Chapter 2
Definitions and Basic Concepts

In this chapter we define the notion of hierarchy, more precisely, the dominant-subordinate relationship. Based on this concept, the hierarchical nature of the entire system can be measured and analyzed, for which several techniques and approaches are reviewed. We also discuss the main characteristics of hierarchical systems, such as their context and time dependency. Definitions and examples are also provided for various types of hierarchies. Finally, visualization techniques are overviewed, both for small and big graphs.

As we indicated in the introduction, the notion of hierarchy applies to a great variety of topics and contexts, be it the social structure of animal groups, human virtues, psychological needs or the structure of a computer program. Accordingly, it does not have a compact, precise, widely accepted definition that would be applicable to all cases. Available definitions usually bypass the problem of clarification by using synonymous words, which are, unfortunately, similarly unclear. For example, according to the Cambridge dictionary, hierarchy is “a system in which people or things are arranged according to their importance.” Here, “importance” is the keyword, but importance is highly subjective: something that is important in a given context might not be important at all from another point of view. Here, we also find that hierarchy corresponds to “the people in the upper levels of an organization who control it.” So, we learn that it is about control, but according to this definition, hierarchy is restricted to people in an organization, which is a very narrow interpretation. Checking a very popular site, Wikipedia, we find that “A hierarchy (from the Greek “hierarchia,” “rule of a high priest,” from “hierarches,” “leader of sacred rites”) is an arrangement of items (objects, names, values, categories, etc.) in which the items are represented as being “above,” “below,” or “at the same level as” one another.” However, this interpretation does not inform us about the basic aspects of the arrangement, which represent, on the other hand, the heart of the problem.

As we shall see, it turns out from more strict investigations that we usually talk about hierarchy in situations in which the entities of a system can be classified into levels in a way that elements of a higher level determine or constrain the
**Definition** A system is hierarchical if it has elements (or subsystems) that are in dominant-subordinate relation to each other. *A unit is dominant over another unit to the extent of its ability to influence the behaviour of the other. In this relationship, the latter unit is called subordinate.*

A typical hierarchical structure can be seen in Fig. 2.1, depicting the ranks within a pigeon flock. The inner structure of the group has been established by observing and measuring the feeding-queueing behaviour of its members (Nagy et al. 2013).

Note that this definition does not tell us how hierarchical the system is. Instead, it expresses whether its elements (or subsystems) are in hierarchical relation or not (manifesting itself in a dominant-subordinate relationship). Furthermore, it tells the origin (reason) and extent of the dominant-subordinate relationship. Consider, for example, the rock–paper–scissors game. According to the rules,

- The rock blunts the scissors (and hence “disarms” it, beats it)
- The scissors cut the paper, and
- The paper covers the rock.

Figure 2.2 shows how the elements overpower each other. Based on the above definition, the hierarchical (dominant-subordinate) relationship among the units is clear, but the hierarchical nature of the whole system is not: is this network hierarchical at all?

In other words, from a graph-theoretical point of view, the above definition gives a lead regarding the *arrows* (where they should be and what their deeper meaning is) but it does not tell us how hierarchical the entire system is. At this point, we
choose to keep it this way, mainly because the extent of hierarchy within a system has subjective aspects: for some, the rock-paper-scissors game is “fully” hierarchical, since its elements are clearly in hierarchical relation. For others, it is not, because no source (leader) can be determined.

Many approaches have been proposed to measure the hierarchy of a network, but none of them is “universal,” or accepted by everyone for all cases. Section 2.1.2, “Measuring the level of hierarchy,” gives an overview of these measures and algorithms.

A few comments related to the definition:

- During different group activities, the influence of the members might vary. In other words, hierarchy is context/task-sensitive, even within the same group! For example, as we shall see in Sect. 3.1.3, “Leadership versus dominance,” the members of the same pigeon flock arrange themselves into different hierarchies according to the activity being undertaken: when they feed, the ranks are entirely different from those that can be observed during flight. This phenomenon is even more starkly expressed in human groups.
- Hierarchy might vary over time. As certain characteristics of the group members change (for example, the physical strength of the individuals in a pack of wolves), so do their ranks.
- This definition implies that the units behave somehow, or have some observable characteristics. In other words, entities without observable behaviour or characteristics cannot form a hierarchical structure.
- The influence can either be forced by the higher-ranked individual (e.g., when a higher-ranked pigeon does not let a lower-ranked one near the food source), or it can be voluntary (for example, leader-follower relationships during flight).
- A higher-ranked unit, by influencing the behaviour of other units more extensively, has a larger effect on the collective (emergent) group behaviour as well.

Fig. 2.2 Graph representation of the rock-paper-scissors game. The dominant-subordinate relationship among the elements is clear, but the hierarchical nature of the entire system is not
Hierarchical systems can be classified into the following subtypes:

1. **Order** hierarchy is basically an ordered set, in which a value is assigned to each element characterizing one of its arbitrarily chosen features. This assigned value defines the rank of the entity within the hierarchy. An example of this can be the ranking of artists, e.g., painters or sculptors, based on the average price of their artworks. In this example, the “set” is composed of the artists, and the feature is the average price of their artwork. Another example can be a hierarchy of firms, ordered by, say, their number of employees. In this type, the network behind the system is neglected or it does not exist. More formally, this type of hierarchy is “equivalent to an ordering induced by the values of a variable defined on some set of elements” (Lane 2006).

2. **Nested** (or **embedded**, **containment**, **inclusive**) hierarchy is a structure in which entities are embedded into each other. Higher level entities consist of and contain lower level entities, or, as Wimberley (2009) has formulated it, “larger and more complex systems consist of and are dependent upon simpler systems and essential system-component entities.” (According to some categorizations, a nested hierarchy can contain only one entity at each lower level, a bit like the case of the Russian Matryoshka dolls, while a generalized nested hierarchy allows for multiple objects.) Uncovering a nested hierarchy structure within a system is closely related to community detection in graphs.

Containment hierarchy has two sub-types:

- **A subsumptive containment** hierarchy (a.k.a. **taxonomic hierarchy**) is a structure in which items are classified from specific to general. For example, domestic cats, lions, tigers and cheetahs (gepards) belong to the family of cats called “Felidae,” dogs, foxes and wolves belong to the family of carnivorans, a.k.a. “Canidae,” Canidae and Felidae both belong to the order of Carnivora, etc. (see Fig. 2.3a). Entities are containers, containing other containers. Mathematically, this arrangement can be formulated as:
  
  Foxes $\subset$ Canidae $\subset$ Carnivora (and Carnivora $\subset$ Mammals $\subset$ Animals, to go further on). Each entity on a lower level “is an” entity of a higher level: a fox “is a” Canidae, a Canidae “is a” Carnivora, a fox “is a” mammal, etc. It is assumed that entities on a lower level are proper (or strict) subsets of the entities on a higher level.

- **Compositional containment** hierarchy (a.k.a. **level hierarchy**) describes how a system is composed of subsystems, which are also composed of subsystems, etc. The “hierarchy of life” is the best example of this structure, describing how organisms are composed of organ systems, which are composed of organs, which are composed of tissues, which are composed of cells, etc. (see Fig. 2.3b). Two important features often (although not always) appear in this type of hierarchy: firstly, there is a “scalar quality,” meaning that entities on higher levels are often bigger than entities on lower levels (a cell is bigger than a molecule). Secondly, there are emergent properties—properties that are not present on lower levels, but due to interactions among the units, appear on higher levels—that also often accompany this structure. For example, consciousness appears on the level of the
brain (which is an organ), but it originates from the interactions of the neuron cells. Emergent properties are of prime importance, since they are a fundamental characteristic of “complex systems.”

3. **Flow (or control)** hierarchy: “intuitively,” this is an acyclic, directed graph. The nodes are layered into levels in such a way that nodes on higher levels influence nodes on lower levels, and the influence is represented by edges. Layers refer to power, that is, an entity on a higher level gives orders or passes on information to entities on lower levels. This is where the name comes from: such a structure represents the flow of orders, or, equivalently, how certain entities control other entities. Armies, churches, schools, political parties and institutions are typically organized in this way. Downwards orders flow along the edges; upwards pointing edges correspond to requests or the sending of information. Technological systems are also often organized in this way. In those cases, a central unit controls particular devices, which control lower level devices, etc. At the bottom-most level, sensors do not control anything directly, but they send

---

**Fig. 2.3** The two types of containment hierarchy: “taxonomic” and “compositional”. a In a taxonomic (or subsumptive) containment hierarchy, entities are containers, containing other containers. b A compositional containment (or level) hierarchy describes how a system is composed of subsystems, which are also composed of subsystems, etc. The best known example of this type of hierarchy is the “hierarchy of life.” b is reproduced from Mader (2010)
information upwards, which is used to refine the decision-making process enacted by devices on higher levels (see Fig. 2.1).

Importantly, these hierarchy types are not independent of each other. On the one hand, many systems can be described by more than one type. For example, members of an army form control hierarchy in such a way that people having higher rank give orders to lower-rank soldiers, but, at the same time, the very same army forms a compositional containment hierarchy as well. This is because an army is composed of various divisions (infantry divisions, motorized divisions, airborne divisions, etc.) that are also composed of smaller contingents, all the way down to the soldiers, who are the “units” in this structure.

On the other hand, both order and nested hierarchies can be converted into a flow hierarchy. In an order hierarchy, a directed edge can be assigned to each pair of adjacent members in the hierarchy, producing a chain of directed edges. In a nested hierarchy, a virtual node is assigned to every sub-graph, and if a sub-graph contains another, then the two corresponding virtual nodes are connected by a directed link, which produces a flow hierarchy along the network of virtual nodes.

Thus, flow hierarchy is the most important variant, and we shall mainly concentrate on its manifestations.

2.1 Describing Hierarchical Structures

In this chapter, we shall briefly summarize the basic concepts related to graphs, the mathematical object most often used in relation to hierarchy. It is important to highlight that graphs and networks are only the models of the real-life systems, not the systems themselves. It is a mathematical representation of the system under investigation, utilized because, using graph theoretical methods and algorithms described in subsequent chapters, they can reveal many important characteristics. An important further comment is that—as is done in the literature—we shall use the term ‘graphs’ for abstract mathematical constructions, while the term ‘networks’ will be associated with the underlying interactions within a real-life structure. Readers familiar with graphs may skip this chapter.

2.1.1 Graphs and Networks

As mentioned above, the most commonly used mathematical tool for describing hierarchical systems is graphs. Primarily, but not exclusively, they are connected to systems embodying flow (or control) hierarchy. Such systems and their graph representations go hand in hand to such an extent that when trying to assign a “hierarchy value” to a system (describing “how hierarchical” the given structure is), it is usually the hierarchy level of the graph (representing the system) that is measured.
The concept behind this representation is rather straightforward: the entities of the systems are the nodes of the graph, and if a pair of entities is in a subordinate-dominant relationship, then there is a directed edge between them.

In the following, we give a short overview of the basic theoretical concepts of graphs:

- A **graph** is a mathematical tool that is appropriate for handling a set of objects with connections among them. The objects are represented by nodes and the connections between them by edges. Formally, \( G = \{ V, E \} \) with a function \( f: E \to V \times V \). The elements of \( V \) are the nodes (or vertices, or points), and the elements of \( E \) are the edges of the graph. The nodes are usually denoted by small Latin letters (e.g. \( i, j, k \)) or by Arabic numbers \( (1, 2, \ldots, N) \). Formally, \( f \) sends edges to pairs of vertices (which are the “endpoints” of the edge), but in practice, we usually forget about the function \( f \) and simply think of \( E \) (the set of edges) as a subset of \( V \times V \). Accordingly, edges are usually given by the starting and nodes, such as \( e = (i, j) \), for any \( e \in E \). The word network is often used as a synonym for graph in cases in which it stands for data that is actually observed.

- A graph can be either **directed** or **undirected**. In the case of a directed graph (or *digraph*), the relation has a special direction as well. For example, in the case of a hierarchy network, the direction can show which element dominates which other. In contrast, in an undirected graph, the connections do not have special directions, as in the network representing the flight connections among cities. Informally speaking, in the case of an undirected graph, the edges are just “lines,” and in the case of digraphs, they are “arrows.”

- A **simple loop** is an edge that connects a node to itself. (An edge whose starting and endpoints are the same vertex.)

- A **path** in a graph is a sequence of connected vertices. (Most definitions specify that the nodes within a path have to be distinct from each other.)

- A **cycle** is a closed path, that is, a path whose starting and endpoints are the same vertex. Many time cycles are also referred to as loops.

- A **tree** is a graph in which there are no loops, cycles or multiple edges. In other words, it is a graph in which any two nodes are connected by exactly one path. There are two special kinds of vertex: (i) the root node, which does not have parents, and the leaves (or end-nodes), which do not have children. Accordingly, in a tree, nodes can be layered into levels.

- A **cluster** (a.k.a. module, community or cohesive group) is a part of the graph in which the units are more densely connected to each other than to the rest of the graph. We will use this elastic description, since the concept does not have a well-defined, widely accepted definition. Importantly, in real-life networks, the presence of such modules is a signature of the hierarchical nature of the structure (see, e.g., Vicsek 2002; Ravasz et al. 2002; Palla et al. 2005).

- A **directed community** is simply a community in a directed graph. Here, the nodes can be related to each other based on the number of incoming and outgoing links connecting them to other nodes within the same module. A node
having more outgoing edges towards other members of the module is more like a “source”-node, whereas a node with mostly incoming links from these members is more like a “drain” (Palla et al. 2007).

- Vertices can be characterised by the number of links they have, reflecting how “strongly” they are connected to other nodes. Accordingly, the \textit{degree} of a node in an undirected graph is simply the number of its edges. In a directed graph, vertices can be characterised by their \textit{in-degree} and \textit{out-degree} values: the \textit{in-degree} value refers to the number of links pointing \textit{towards} the given node, whereas the \textit{out-degree} value refers the number of links going outwards from the vertex.

\section*{2.1.1.1 Centralities and Complexity Measures of Networks}

Once a graph/network is defined, a number of interesting local and global quantities can be introduced for its characterization. Here, we first mention a few commonly used quantities of this kind, and next, we turn to measures that are essential from the point of view of hierarchy.

Centrality values are usually associated with a single node or edge, but typically depend on the global structure of the network. Perhaps the best known and most used quantity in this context is the so-called ‘betweenness’ centrality, which measures the centrality of a node in a connected graph based on the shortest paths. For every pair of nodes, there exists at least one shortest path between the nodes. Betweenness centrality for each node is then the number of these paths that pass through the node (Freeman 1977). Obviously, it measures the level of importance of a given node in the context of information transfer within the network.

While betweenness centrality has a unique definition, the situation is much less well defined when one considers the level of complexity of a network. The situation is similar to what holds for the definitions of hierarchy measures, and, indeed, although complexity and hierarchy are two different notions, they are intimately related. Several complexity measures have been defined in the literature, and we reproduce only a few of them here in order to illustrate their nature. The algorithmic complexity of a network is given by the shortest possible program that can be used to generate its structure. Complexity measures that involve hierarchy are typically specified for networks with an underlying tree-like structure. For example, the so-called cyclomatic number (Temperley 1981) measures the complexity of an expression, represented as a tree. Expressions with either identical nodes or with all different nodes are the extremes in an “entropy” scale, because they are either trivial or impossible to compress. The more complex ones in the cyclomatic sense are those whose branches are different, yet some subtrees are reused across branches. Bochev and Buck (2005) discuss this and further relevant measures of network complexity in detail.
2.1.2 Measuring the Level of Hierarchy

In this section, we shall focus on measures for flow hierarchies. More precisely, we consider measures for graphs representing flow hierarchy. We have two main reasons to do so: (i) observations and experiments, as well as results of computer simulations, are likely to return flow hierarchy; (ii) all other hierarchy types can be transformed into flow hierarchy in a rather straightforward way. For example, considering a containment hierarchy, its clusters can be identified with the nodes of a graph in which the directed edges will indicate the containment relation. That is, in the graph, there will be an edge pointing from node A to node B, if cluster B fully contains cluster A in the original structure (Nepusz 2013).

Most of the proposed measures take values on the [0, 1] interval, returning nearly 0 for a completely hierarchy-less structure, like a full graph or a circle, and returning a value close to 1 for “completely hierarchical” structures, like a directed tree. Values for transient structures rely on “intuition,” and intuition differs from person to person. This is one of the main reasons why there is no “most appropriate” measure serving all needs. The measures reviewed in the present book have values on the [0, 1] interval, with higher values representing higher degrees of hierarchy.

This section of the book is relatively extensive for two reasons: (i) it is about an obviously central quantitative characteristic of a hierarchical structure; (ii) in spite of its essential importance, there is no unique definition of the level of hierarchy of a system.

This latter situation is analogous to that of the definition of a community in a network. The notion itself is so complex that, depending on the aspect that we are interested in, a suitable definition should be chosen. For example, a community (cluster) in a network can be defined as a sub-network of nodes that have a relatively greater number of connections among them than with the other nodes. However, we can require this “relatively more” to manifest itself in various ways. Directed, weighted and connections specified according to further criteria make the problem of defining clusters in a network an even more open problem.

To introduce this aspect of the problem of finding the best measure of hierarchy, the reader is asked to consider the following question: which structure is more hierarchical? A set of nodes arranged into layers connected by directed edges, all directing from an upper to a lower layer, or a “star” consisting of a central node from which a number of directed edges lead to the other nodes of the network? To us, the right answer is: it depends on the context, on the function, etc. Next, we account for a number of relevant possible angles from which such a question can be approached.
2.1.2.1 Global Reaching Centrality

The central idea of this approach is to give a rank to each node by measuring its “impact” on other nodes. Impact is defined by the ratio of vertices that can be reached from the focal node \( i \). Local reaching centrality, \( C_R(i) \) defines this quantity exactly: in a directed, un-weighted graph, \( C_R(i) \) is the maximum number of vertices that can be reached from node \( i \), divided by \( N-1 \). Then, the level of hierarchy is inferred from the distribution of the local reaching centralities: the more heterogeneous the distribution, the more hierarchical the corresponding graph/network. In order to demonstrate this statement (namely, that the distributions of the local reaching centralities reveal the hierarchical nature of a network), three different graph types are compared in Fig. 2.4: an Erdős-Rényi (random) graph (which is not hierarchical), a tree (which is highly hierarchical), and a scale-free graph (which is “moderately” hierarchical). The most homogeneous \( C_R(i) \) distribution belongs to the Erdős-Rényi (ER) graph: the \( C_R(i) \) values are either 0 or close to 1, marked by the two narrow spikes at these values with a solid black line. In contrast, we find all kinds of \( C_R(i) \) value in a tree, as indicated by the red line in Fig. 2.4 (note the log-log scale).

![Fig. 2.4 Distributions of the local reaching centralities for three kinds of directed network: Tree, Erdős-Rényi (ER) and scale-free (SF). All the curves are averages of 1000 graphs with \( N = 2000 \), of the appropriate graph type. Reproduced from Mones et al. (2012)
This distribution follows a power law that is distorted due to the random branching numbers. The blue dashed line belongs to the “moderately hierarchical” scale-free graph, marking a “moderately heterogeneous” distribution.

These curves represent distributions, while for a measure, we expect a number. The definition proposed by Mones et al. (2012) grasps the heterogeneity of the \( C_R(i) \) distribution as follows: Let \( C_R^{\text{max}} \) denote the highest local reaching centrality in a graph \( G = (V, E) \). Then, the Global Reaching Centrality, \( GRC \), is defined as:

\[
GRC = \frac{\sum_{i \in V} [C_R^{\text{max}} - C_R(i)]}{N - 1},
\]

where \( V \) is the set of nodes and \( N \) is the number of nodes in \( G \). The GRC values for our three example graphs (Tree, Scale-free and Erdős-Rényi), are as follows:

- Tree: 0.997 ± 0.001, which is the highest.
- Scale-free: 0.127 ± 0.008, that is, SF networks are slightly hierarchical.
- Erdős-Rényi: 0.058 ± 0.005, that is, these are not hierarchical at all.

These values, the means and variances are calculated for an ensemble of 1000 graphs, and they demonstrate that the returned values are close to our “intuition.” Equation (2.1) applies to directed, un-weighted graphs. Its generalized version is suitable for analysing weighted and/or undirected graphs by an appropriate modified definition of the local reaching centrality (Mones et al. 2012).

2.1.2.2 Random Walk Measure

The main motivation for this approach is the claim that—in contrast to the assumptions behind most of the proposed methods—it is not correct to treat all directed acyclic graphs as already being maximally hierarchical, independent of their inner structure. This observation is based on the common intuition that a hierarchical structure often corresponds to a multi-level pyramid in which the levels become more and more wide as one descends from the higher levels towards the lower ones.

The measure proposed by Czégel and Palla (2015) is based on properties of random walks within the graph, and, in accordance with the above-mentioned claim, directed trees corresponding to multi-level pyramidal structures obtain higher values of hierarchy than directed stars or chains.

Intuitively, the method is based on the assumption that there is information flow coming from the high-ranking nodes towards the ones at the bottom, similarly to the case of an army or company, in which the leaders send instructions downwards along the links. In order to track the sources of the instructions/information/etc., random walkers are dropped onto the nodes who then move backwards along the links. Once a steady state is reached, the density of such random walkers (the number of them visiting a given node) can be interpreted as being proportional to the rank of this node: high random walker density indicates that the vertex is a
source of information, low density indicates the vertex is more likely to be just a “receiver” of orders—that is, low in rank. The hierarchical nature of the network is then estimated based on the distribution of these random walker densities: if the distribution is homogeneous, the source of information/order cannot be pinpointed, thus, the network is not hierarchical. In contrast, inhomogeneous distribution indicates clear information sources: the network is hierarchical. This homogeneity/inhomogeneity is measured with a value called $H$, with higher values reflecting more hierarchical structures (bigger inhomogeneity) and lower values less hierarchical networks.

The largest $H$ values belong to regulatory networks, electric circuits and food webs, whereas the lowest ones belong to the informal networks of acquaintances in different organizations (Fig. 2.5). Moderately hierarchical examples include the Internet and various citation-, metabolic-language and trust networks, the results of which are in good accordance with our intuitive expectations.

An even clearer picture regarding the hierarchical nature of a network can be obtained by “normalizing” the hierarchy measure $H$ against the hierarchy measure of the same network, but under the assumption of random connections. This is the “$z$-score”, defined as
where $H$ is the hierarchy score, $\langle H \rangle$ is the expected $H$ value of the randomized graph, and $\sigma(H)$ is the standard deviation of $H$ in the randomized ensemble.

In the rest of the section, we will give an overview of some further measures, focusing on the main ideas behind them. Here, our aim is not to give a detailed description of the techniques, but rather to flip through the type of concepts that have been proposed so far regarding the problem of measuring the hierarchy level of a graph.

### 2.1.2.3 A Measure for Undirected Networks

The measure proposed by Trusina et al. (2004) quantifies the flow hierarchy of undirected networks. It is based on the assumption that every vertex already has a rank associated with it by denoting its place in the global hierarchy. This estimate for the rank can be the degree of the node (originally proposed by the authors), but can be other conceivable measures as well, such as betweenness centrality or eigenvector centrality. With these assumptions, the hierarchy measure is given by the fraction of shortest directed paths going strictly upwards in the hierarchy.

More precisely, this method assumes that the shortest paths in the network consist of a part going upward along the hierarchy (towards more important nodes), followed by a part going downward along the hierarchy (towards less important nodes). Either part may be empty, of course, but one should not turn back upwards after the downward part again. Paths of this type are said to be hierarchical, and the measure simply calculates the fraction of vertex pairs that are connected by the shortest hierarchical path.

### 2.1.2.4 Determining the Levels of Organizations

One of the first methods was proposed by Krackhardt (1994), whose main motivation was to measure the levels of hierarchy of organizations. He defined four measures that can be used together as an estimate of the extent of flow hierarchy in networks. These measures are:

- **Hierarchy**: The fraction of unordered vertex pairs $(i, j)$ such that vertex $i$ is reachable from vertex $j$ but vertex $j$ is *not* reachable from vertex $i$, or vice versa. It works on directed graphs only.

\[
z = \frac{H - \langle H \rangle}{\sigma(H)}, \tag{2.2}
\]
- **Connectedness**: The fraction of unordered vertex pairs \((i, j)\) such that vertex \(j\) is reachable from vertex \(i\) via a directed path or vertex \(i\) is reachable from vertex \(j\).

- **Efficiency**: One minus the proportion of possible “extra” edges that are not needed to maintain connectedness of the components. It is assumed that each component should be an out-tree (as an archetype of perfect hierarchy), and thus a component of size \(N\) must have, at most, \(N-1\) links; any more than that is a violation of efficiency. This measure obviously penalizes cases in which there are two separate paths leading upwards along the hierarchy from a node \(A\) to its superior \(B\); one of the paths is not required to maintain connectedness, hence the structure is inefficient.

- **LUBness**: For each unordered pair of vertices \((i, j)\), the lowest upper bound (LUB) is a vertex \(k\) such that both \(i\) and \(j\) are reachable from \(k\). LUBness is the fraction of pairs having a LUB. This definition can be explained by Krackhardt’s assumption that out-tree is the most perfect hierarchy one can achieve.

Each of these metrics may take values from zero to one, and each metric measures some kind of a “deviation” from the perfect hierarchy assumed by Krackhardt, i.e., a directed out-tree. (It also applies for in-trees if we reverse the edge directions in the definition of LUBness). However, these measures (with the exception of efficiency) can be calculated only for directed networks.

### 2.1.2.5 Concept for Containment Hierarchies

Unlike the measures presented so far, the concept by Ravasz and Barabási (2003) addresses the notion of containment hierarchies. They observed that \(\log k\) and \(\log C\) are correlated in many real-world networks (where \(k\) is the vertex degree and \(C\) is the local clustering coefficient).

They argue that this is due to a containment hierarchy in the network (although they have not used the word “containment”). In order to determinate this, they proposed a simple recursive generation process that creates graphs with a power-law degree distribution, a linear dependence between \(\log k\) and \(\log C\), and multiple levels of hierarchies contained within each other. The bottom line of their argument is that hierarchy in undirected networks can be quantified by looking at the \(\log k\) vs. \(\log C\) plot and fitting a straight line to the data; the larger the slope of the line, the more hierarchical the network.

### 2.1.2.6 Layout-Motivated Measure

Carmel et al. (2002) proposed a layout-based metric for measuring the amount of hierarchy in a directed graph. They have conceived a layout algorithm that places the nodes of the graph in 2D space such that a set of constraints related to the target level differences are taken into account as much as possible. More formally, this means the following. For each \(i-j\) edge, we assign a measure that describes the
desired difference between the \( y \) coordinates of vertex \( i \) and vertex \( j \). The graph is then laid out using their algorithm, and the difference between the maximal (\( \text{max}Y \)) and minimal \( y \) coordinates (\( \text{min}Y \)) is compared to the diameter of the graph. A strictly hierarchical graph with no cycles can be laid out in such a way that the distance between levels is 1, thus the difference between \( \text{max}Y \) and \( \text{min}Y \) is equal to the diameter, while a cycle (i.e., a perfectly un-hierarchical graph) would be laid out with equal \( y \) coordinates, yielding a hierarchy measure of zero.

The disadvantages of this method are twofold:

- In the general case, it is not possible to assign desired target level differences to the edges. We could simply say that the desired difference is 1 for all of the edges, but this would work only if none of the edges span more than one layer. Edges skipping layers, but otherwise pointing in the right direction, would skew the layout and decrease the hierarchy measure.
- This measure is not applicable to undirected graphs.

2.1.2.7 Measures for Structures “From Down to Top”

Next, in contrast to the situation we assumed above, we shall consider the edges of directed networks to be oriented upwards (i.e., from lower to higher levels), as in a who-reports-to-whom organizational diagram. We do so in order to follow the terminology of the related literature. It is usually a straightforward thing to apply the definitions to directed networks that use the opposite convention.

Sometimes, we will talk about layers or levels (sets of nodes with the same rank). Layers are indexed from 1 upwards, and a lower layer index corresponds to a higher rank.

Some of these measures will work on networks in which the ranks of individual nodes are not known in advance; others are defined for a network and a corresponding ranking of nodes, and therefore must be optimized through some optimization procedure when the ranks are unknown.

2.1.2.8 Fraction of Edges Participating in Cycles

Here, the main idea is to reveal somehow the possible asymmetry between nodes by assuming some sort of flow along the links, and then checking whether or not these flows exhibit any kind of overall directionality. One way to do so is to find all of the elementary cycles in the network, count the edges participating in them, and divide this number by the total number of edges. This approach works for both directed and undirected graphs; in directed graphs, only directed cycles matter. (A cycle is elementary if no vertex appears in it twice.)

All of the elementary cycles in a directed graph can be found simply by using Johnson’s algorithm (Johnson 1975), which is \( O((N + E)(c + 1)) \), where \( N \) is the number of nodes, \( E \) is the number of edges and \( c \) is the number of elementary
cycles. The case of undirected graphs is a bit trickier, as the union of two elementary cycles with at least one shared edge is also an elementary cycle (after removing the shared edges from the union), and thus, we can expect a far greater number of cycles than that for directed graphs for which this condition does not hold. It is therefore common to search for a cycle base instead, i.e., a set of cycles such that every other cycle can be reproduced from selected base cycles by taking their disjoint unions. Since every edge that participates in a cycle must also participate in one of the base cycles, finding a cycle base is enough for our purposes.

Luo and Magee (2011) proposed the opposite of this measure (i.e., the fraction of edges not participating in cycles) as a hierarchy measure for directed networks. One big advantage of this approach is its simplicity.

### 2.1.2.9 Minimum Fraction of Edges to be Removed to Make the Graph Cycle-Free

This approach is slightly different from the one called “fraction of edges participating in cycles.” For instance, consider a graph consisting of two interlocking directed links sharing an edge. In this graph, all of the edges participate in cycles (hence the previous measure would be 1.0), but removing the shared edge would make the graph entirely cycle-free. We call a set of edges whose removal makes the graph cycle-free a feedback arc set.

Note that although Fig. 2.6 shows a directed graph, this measure works just as well for undirected graphs—but the number of edges to be removed may be different! For instance, the graph with the two rings on the left side of Fig. 2.6

![Fig. 2.6](image_url)

**Fig. 2.6** Illustration of the difference between “the fraction of edges participating in cycles” and the “fraction of edges to be removed to make the graph cycle-free.” Subfigure **a** shows a graph in which all of the edges participate in cycles. However, as can be seen in **b**, it is enough to remove a single edge (from J to A) to break both cycles and obtain a perfect hierarchy.
becomes cycle-free through the removal of one single edge if the edges are directed, but one has to remove two edges to make it cycle-free in the undirected case.

This measure is very easy to calculate for connected undirected simple graphs. Since the graph is connected, the minimum number of edges required to connect \( N \) vertices is \( N-1 \). Adding any extra edge on top of these \( N-1 \) edges necessarily creates a cycle, thus the number of edges one has to remove from an undirected simple connected graph with \( N \) vertices and \( M \) edges is \( M-N+1 \), and the fraction of such edges is therefore \( 1-(N-1)/M \).

For directed graphs, finding a minimum feedback arc set is an NP-hard problem (Healy and Nikolov 2013), but heuristic procedures exist to find an approximation. One such procedure is the greedy cycle removal algorithm by Eades et al. (1993) Namely:

1. Create an empty “deque” (double-ended queue).
2. If the graph is empty, we are done.
3. If there are sink vertices in the graph, remove them one by one and add them to the beginning of the deque.
4. If there are source vertices in the graph, remove them one by one and append them to the deque (add them to the end of the deque).
5. If no sinks and sources remain, find a vertex where the difference between the out-degree and the in-degree is as large as possible, remove it from the graph, append it to the deque, and return to step 2.

At the end of the algorithm, the deque contains a possible ordering of vertices where ordinary edges point “forward” in the ordering and feedback arcs point “backward.” The cardinality of the feedback arc set found by this heuristic is, at most, \( M/2-N/6 \), where \( M \) is the number of edges and \( N \) is the number of vertices.

Another heuristic is as follows. Scan each edge of the graph one by one and maintain two sets, \( S \) and \( T \). At each step, check whether edge \( e \) forms a cycle with the edges already in \( S \). If not, add \( e \) to \( S \); otherwise add \( e \) to \( T \). In the end, both \( S \) and \( T \) are acyclic, and the smaller of the two sets gives a feedback arc set with, at most, half of all of the edges. More sophisticated approximations are to be found in Even et al. (1995) and Saab (2001).

For graphs up to a couple of hundred nodes, one can use the following strategy as well:

1. If the graph is undirected, break it down into components and calculate the sum of \( M-N+1 \) for each component, where \( M \) is the number of edges in the component and \( N \) is the number of vertices. This is the total number of edges to be removed in order to make the graph cycle-free; the fraction follows from a straightforward division.
2. If the graph is directed, break it down into weakly connected components and estimate the number of edges to be removed from each of the components as follows:
• If the component is acyclic (i.e., it has a topological ordering), no edges have to be removed at all.
• If the component has less than 20 edges, use a brute-force search to find the minimum number of edges to be removed in order to make it cycle-free.
• Otherwise, find a minimum cut of the component, add the edges of the cut to the feedback edge set, and proceed recursively with each side of the cut.

2.1.2.10 Fraction of Hierarchy-Violating Edges

A hierarchy-violating edge is one that originates at a higher level and terminates at a lower level, meaning that someone higher up in the hierarchy “reports to” someone on the lower level. This is a clear violation. Naturally, this measure requires the ranks to be known in advance, as it is otherwise impossible to decide which edges violate the hierarchy.

Another, stricter definition of a hierarchy-violating edge is that it is an edge for which subtracting the rank of the origin from the rank of the target yields a result that is not zero and not one. This definition penalizes not only the edges that go “the wrong way” in a hierarchy, but also the edges that skip levels.

In the absence of ranks, one has to find the ranking that minimizes the fraction of hierarchy-violating edges, which leads to a problem that may be familiar from community detection. A trivial way to minimize the number of hierarchy-violating edges is to use the same rank for every node, assuming that edges between peers (i.e., nodes with the same rank) are allowed. A possible solution is to disallow edges between peers, which effectively reproduces the feedback arc set problem, since a directed graph minus a minimum feedback arc set is a directed acyclic graph that can then be decomposed into layers. Each feedback arc is then a hierarchy-violating edge.

2.1.2.11 Average Expected Downstream Path Length

This measure is based on random walks, more precisely, the expected length of a path a random walker is allowed to take on the graph with the following constraints:

1. The walker is only allowed to step downstream in the graph, i.e., towards lower layers. A path that goes downward in a layered hierarchy is called a downstream path (hence the name of the measure).
2. The transition matrix of the random walk is a usual right-stochastic matrix derived from the weighted adjacency matrix of the graph (loop edges are not allowed).
3. The random walk terminates as soon as the walker ends up in a sink node or in a node that has neighbours in higher layers only.
The measure also requires an a priori layer assignment, and it is an open problem to find the optimal assignment given the graph alone. When the layers are known, the measure can be calculated very easily: one has to proceed recursively from the lowermost layer towards the uppermost layer and make use of the following two equations:

1. If a vertex \( v \) is a sink, then the expected length of downstream paths from \( v \) is zero.
2. If \( v \) is not a sink, the expected length is one more than the expected length of downstream paths from its lower-level neighbours, weighted by the probabilities of reaching those neighbours from \( v \) in a single step. Note that only the expected length of downstream paths for vertices in layers lower than \( v \) has to be known, therefore, a single sweep from lower layers to the uppermost layer is enough.

To make graphs with different numbers of layers comparable, it is advised to normalize this measure as follows.

Suppose that vertex \( v \) is at layer \( l(v) \) and there are \( k \) layers. The maximal value of the expected downstream path length originating from \( v \) (denoted by \( h(v) \)) is then \( k - l(v) \). The normalized variant of the measure takes the average of \( h(v)/(k - l(v)) \) for all non-sink vertices, assuming that 0/0 is 0.

The above overview of the “further hierarchy measures” was composed using the working paper by Nepusz (2013).

### 2.1.3 Classification of Hierarchical Networks

The methods overviewed in Sect. (2.1.2) assign a value for each graph, reflecting the extent to which the input network is hierarchical. Now, we will reverse the direction and show an algorithm that creates a graph based on an input parameter \( p \) (taking values on the \([0, 1]\) interval), indicating how hierarchical the output graph should be. \( p = 0 \) refers to non-hierarchical structures and \( p \) close to 1 refers to strongly hierarchical structures. The method was proposed by Mones et al. (2012).

The construction of a graph with tunable levels of hierarchy goes as follows (Fig. 2.7a):

- A level-value (\( \ell \)) is assigned to every node in a directed tree in the following way:
  - The nodes at the “bottom-level” (that is, the leaves) are assigned \( \ell = 1 \).
  - The level-value of the root node is equal to the number of hierarchical levels in the tree (for example, \( \ell = 5 \) in Fig. 2.7a of the root node).
  - All children of a node with level-value \( \ell \) will have \( \ell - 1 \) as their level-value.
- Next, a given number of randomly directed edges are added to the tree according to the following rules:
– $1-p$ proportions of these edges are added completely randomly by choosing their starting point ($A$) and end-node ($B$) with probability $1/N$ ($N$ is the number of nodes in the graph). In a case in which there is no directed edge pointing from $A$ to $B$, such an edge is added to the graph.

– Regarding the remaining edges (accounting for the $p$ proportion of the “extra” edges), they are added only if $\ell_A > \ell_B$.

Figure 2.7b depicts the GRC values (see Sect. 2.1.2) for hierarchical graphs created with the above algorithm, for $p = 0.0, 0.2, 0.4, 0.6, 0.8$ and $1.0$. 

---

**Fig. 2.7**

a) The different types of edges while constructing a hierarchical graph based on an input parameter $p$. Solid blue edges belong to the original tree used as the backbone of the output graph. Edges pointing downwards (green) conserve the hierarchy, horizontal edges (orange) have a slight influence, and finally, the ones directed upwards (marked with red) cause strong change within the structure.

b) Distribution of the local reaching centrality (see Sect. 2.1.2) values for adjustable hierarchical networks with various $p$ values. Each curve is an average of 1000 networks with $N = 2000$ nodes for $<k> = 3$. Note that from the highly random ($p = 0$) to the highly hierarchical ($p = 1$) state, the distribution changes continuously and monotonously with $p$. Reproduced from Mones et al. (2012)
Similarly to the problem of measuring hierarchy, the problem of classifying hierarchical structures is not trivial either. Next, we will give an overview of a method proposed by Corominas-Murtra et al. (2013), which is based on three expectations towards hierarchical systems. These are: (i) treeness, (ii) feed-forwardness, and (iii) orderability. (See later for greater detail.)

Using the above three features, a 3D morphospace ("phenotype-space") can be defined in which the three axes are the tree’s quantifiable features. Placing real-life hierarchical and random null-models into such a coordinate system, fundamental characteristics can be revealed. As it turns out, networks do not occupy the entire morphospace; instead, they accumulate in four major clusters within the large voids, which most probably results from the constraints under which they evolve.

Let us define the proper position of a network \( G(V, E) \) within the morphospace. First, \( G(V, E) = G \) is transformed into its corresponding node-weighted condensed graph \( G_C(V_C, E_C) = G_C \), which is an acyclic feed-forward structure in which the cyclic modules (strongly connected components) of \( G \) are replaced by single nodes. Accordingly, in a node-weighted condensed graph \( G_C \), each node has a weight \( z_i \) indicating the number of nodes it includes from \( G \), the original graph. For example, in Fig. 2.8, subfigure h depicts the node-weighted condensed graph \( G_C \) corresponding to \( G \), the one depicted on subfigure d. In this, node \( S_2 \) includes 3 nodes from \( G \), and \( S_1 \) includes 2. (This method, the localization of strongly connected components, is an often-used approach for identifying subsystems within a graph.)

Then, we calculate the three values using both \( G \) and \( G_C \):

1. “Treeness”, \( T \): Taking values on the \([-1, 1]\) interval, this captures how unambiguous the “chain of command” is within \( G_C \). In hierarchical networks, like the one in Fig. 2.8a and on its corresponding node-weighted graph depicted in e, the chain-of-command is unequivocal, characterized by positive \( T \) values. In a case in which the chain of command is ambiguous, the structure is said to be anti-hierarchical, marked by negative \( T \) values (Fig. 2.8b, f). Intuitively, this feature is calculated by comparing the diversity of choices one can make top-down vs. the uncertainty on the way bottom-up, captured by the concepts of forward and backward entropies.

2. “Feed-forwardness”, \( F \): Since the paths within cyclic modules (like \( S_1 \) and \( S_2 \) in Fig. 2.8h) violate the downstream order within the graph, they are penalized according to their size and position; larger modules closer to the top of \( G \) have greater influence on the overall structure of \( G \) than smaller ones close to the bottom. Accordingly, they introduce a larger penalty. \( F \) is defined on the \([0, 1]\) interval.

3. “Orderability”, \( O \): This is defined as the fraction of nodes that do not belong to any cycle. These nodes are orderable, and accordingly, a bigger ratio results in a higher value of orderability. \( O \) takes values from \([0, 1]\).
Figure 2.9 describes the location of random null models (white circles) and 125 real networks within the morphospace. Since random networks are being created without any selection pressure, they are neither hierarchical nor anti-hierarchical, and they occupy the $T/C_{25}$ segment.

The main observation is that the vast majority of real networks fall into four clusters:

i. Gene regulatory networks (plus a protein kinase NW) occupy the first cluster at the top of the coordinate system (Fig. 2.9), marked as “GRN.” These systems are characterized by very high orderability values ($O$) with variable $F$ values. The broad range of $F$ (feed-forwardness) is caused by various sized modules near the top of the networks, corresponding to a small fraction of genes, (transcription factors) participating in cycles.
ii. Electronic circuits and software graphs are strictly feed-forward ($F \approx 1$) with orderable nodes ($O \approx 1$), biased slightly towards negative $T$ values. This cluster (marked as “TECH” in Fig. 2.9) is located on the top right edge of the morphospace.

iii. The third cluster is defined by the ecological flow graphs, marked as “ECO” in Fig. 2.9. Their positions within the morphospace reveal a certain degree of pyramidal structure combined with the important role played by loops. This special, separated position is consistent with the trophic pyramid mingled with recycling.

iv. And finally, the fourth cluster is composed of metabolic, neural, linguistic, and some social networks (“LANG, MET, NEU”), embedded within the cloud of random graphs. These networks display a large central cycle, much larger than their randomized counterparts, a feature that is most probably due to the advantage of reusing/recycling molecules.

Two of these clusters (LANG/MET/NEU and TECH) overlap with random networks with similar connectivity, suggesting that non-adaptive factors shape the topological nature of these graphs. In contrast, the position of the ECO and GRN clusters indicate that the topological features of the ecological and gene networks are the result of functional constraints.
2.2 Visualization Techniques

2.2.1 A General Overview

The aim of the various visualization techniques is the same: to illustrate the entire network as a single figure in an easily perceptible way, revealing as much information about its hierarchical nature/inner structure as possible. Since (real) hierarchical systems are often complex, with many characteristics, the extent to which a visualization technique reflects the main features of a network is limited. Different visualization tools highlight different characteristics and different hierarchy types require different visualization tools. There exists no “best method”; the appropriate technique depends on the specific characteristics we would like to highlight.

The most simple—and widespread—visualization technique is the pyramid, in which each entity is represented by a layer, and the higher an entity is in the hierarchy, the higher it is in the diagram. Often (for example, in the case of social pyramids), but not always (e.g., Maslow’s hierarchy), the width of the layer reflects the size of the represented layer. The drawback of this technique is that it can reflect only a linear order (a sequence) of the layers and, in some cases, their approximate sizes. In other words, this technique reveals only an order hierarchy of the layers, without giving any description about the inner structure of the given system.

In contrast, graphs are applicable for describing not only order hierarchy, but other hierarchy types as well, most importantly flow hierarchy, meanwhile allowing for a much more detailed visualization of the inner structure of the system as well. For these reasons, visualization of flow hierarchy is the most commonly used technique for representing hierarchical systems.

Because of the lack of loops and cycles, the representation of a “pure” hierarchical system would be a tree. However, in real-life cases, such systems occur only very rarely. Accordingly, trees often correspond to the ideal and/or theoretical case, while graphs that are more complex (have cycles, undirected edges, etc.) are better suited to representing real-life cases.

This representation is closely connected to the concept of control (or flow) hierarchy, in which the entities (which are represented by nodes in the corresponding graph) are organized into a system of subordinate-superordinate relations, which correspond to the edges of the graph. Accordingly, orders and information flow along the edges (hence the name) from the superior unit(s) towards the inferior element(s), while requests and information flow in the opposite direction. Typical examples are the ranks in armies, various state and church organizations, corporations, etc.
2.2.2 Techniques Reflecting the Overall Hierarchy Level

Let us consider a graph representing a (real life or artificial) system. The graph can be large, having many communities and sub-communities, and therefore be difficult to draw in a way that is reasonably accessible for overview. However, we would like to know how hierarchical the original system is, preferably in a visual form.

The most widely accepted method for visualizing the hierarchical nature of small networks is the one proposed by Sugiyama et al. (1981). For such graphs, this technique provides an informative and clear hierarchical layout by layering the vertices into horizontal rows in such a way that the edges are directed downwards. This method is often referred to as the “layered graph drawing” or “hierarchical graph drawing” method.

The main steps are as follows (Fig. 2.10):

(i) **Cycle removal** (a pre-processing step). If the directed input graph is not acyclic, a minimal set of “reversal edges” has to be identified and reversed in order to obtain an acyclic digraph. (Identifying such a minimal edge-set is an NP-complete problem.) (These reversed edges, as well as other changes within the graph, will be restored to their original state in a later step.)

(ii) **Layer assignment.** Partitioning the vertex set of the graph into layers in such a way that each edge is directed from a higher level towards a lower one, with the following properties:
   a. the number of layers is kept small
   b. as few edges as possible span a large number of layers
   c. the assignment of nodes into layers is balanced.

(iii) **Insertion of “dummy vertices.”** “Long” edges (edges spanning multiple layers) are chopped up into a series of shorter ones by inserting so-called “dummy vertices” into the graph. After this step, each edge will connect nodes on adjacent layers.

(iv) **Edge concentration** (optional step). The aim of this step is to reduce the number of edge crossings and the edge density between adjacent levels. It might reduce the number of dummy vertices as well, but, as important drawbacks, it may increase the number of layers and also modify the graph.

(v) **Vertex ordering** (or “crossing minimization”/“crossing reduction” step). The nodes within the layers are permuted in such a way that the numbers of edge-crossings are minimized between the adjacent layers.

(vi) **x-Coordinate assignment.** The aim of this step is to position the nodes (that is, assigning them an x coordinate) within each layer in such a way that the edges become as straight as possible, and the nodes are centred with respect to their neighbours. This positioning should be consistent with the permutation applied in the previous step.
(vii) **Final step.** Changes that have been introduced into the graph in previous steps are reversed so that the edges return to their original state:

a. edges reversed in the “cycle removal” (first) step are reverted to their original direction

b. dummy vertices that have been inserted in step (iii) are removed from the graph, and the corresponding “long” edges are drawn back in a way that avoids intersections and crossings. This might be done by drawing the edges as polygonal chains or spline curves.
For a detailed analysis and description of this method, see also (Healy and Nikolov 2013). Although this method is very popular for small networks, it has some serious drawbacks as well, which become especially important for large graphs:

- for bigger networks (graphs with more than a few hundred nodes), the generated layout becomes difficult to overview/interpret;
- the steps are NP-complete or NP-hard, which makes the usage of several different heuristics necessary, and the results therefore less well-defined.
- independently of the hierarchical nature of the given network, the method provides a hierarchical layout that is often misleading;
- the meaning of the levels is not defined;

Next, we discuss a method proposed by Mones et al. (2012) that solves the above problems and is easily applicable, even for large complex networks (see Fig. 2.11).

Fig. 2.11 The main steps of the visualization process. Firstly, the layout is computed, based on the local reaching centrality, $C_R(i)$, values (top right). Next, the levels are separated with a logarithmic ratio, and then each layout is scaled into the unit square (bottom left). Finally, the rescaled layouts are plotted in the unit square with the obtained node-density (bottom right, see the colour bar as well). In the heat maps, the colour scale shows $\log(\log(\rho(x, y) + 1) + 1)$, where $(x, y)$ is the average density of the ensemble. Reproduced from Mones et al. (2012)
The algorithm of the proposed method is as follows:

1. Rank the nodes according to their local reaching centrality value, \( C_{R}(i) \), where \( C_{R}(i) \) is the ratio of nodes that can be reached from the focal node \( i \), reflecting the “impact” of \( i \) on other nodes (see also Sect. 2.1.3). (Importantly, from the viewpoint of the algorithm, instead of \( C_{R}(i) \), other local quantities can be used as well.)

2. Start to add nodes to the first, bottom-most level of the layout in an increasing order regarding their \( C_{R}(i) \) values, until \( \sigma_L < \varepsilon \sigma_G \). Here, \( \sigma_L \) is the standard deviation of the \( C_{R}(i) \) values within the actual level, whereas \( \sigma_G \) is that within the entire graph. \( \varepsilon \) is an adjustable parameter, defining the “resolution” of the levels.

3. Once \( \sigma_L \geq \varepsilon \sigma_G \), start a new level.

4. Repeat the 2nd and 3rd steps until every node has been put into levels. (Step 2 ensures that nodes with similar \( C_{R}(i) \) values will be on the same level.)

5. In order to get a nice horizontal arrangement, align the centre of mass of each level above that of the next, that is, to the same vertical line.

6. The levels are arranged vertically in such a way that the distances between adjacent levels are proportional to the logarithm of the differences of the averages inside the certain levels: \( (Y_{\ell+1} - Y_{\ell}) \propto \ln \left[ \frac{\langle C_{R} \rangle_{\ell+1} - \langle C_{R} \rangle_{\ell}}{\langle C_{R} \rangle_{\ell}} \right] \). \( Y_{\ell} \) is the vertical position of level \( \ell \), whereas \( \langle C_{R} \rangle_{\ell} \) is the average of the \( C_{R}(i) \) values within level \( \ell \).

---

**Fig. 2.12** Visualization of the three network types studied in Sect. 2.1.3, based on their local reaching centrality values. **a** An Erdős–Rényi (ER) graph. **b** A scale-free (SF) network. **c** A directed tree with random branching number between 1 and 5. For all three graphs, \( N = 1000 \) with the parameter \( \varepsilon \) set to \( 2/N \). In the cases of the ER and SF graphs, \( \langle k \rangle = 3 \). Reproduced from Mones et al. (2012)
Next, set the vertical distances of the levels in such a way that they become proportional to the differences between their average $x_i$ values. Set the smallest distance to the same value as the horizontal distance between two adjacent nodes. Finally, set the distances to be proportional to the logarithm of the original differences in such a way that the height of the graph remains unchanged.

For large graphs, $\varepsilon$ tunes the vertical extension of the layout.

Figure 2.12 shows the result of this method for (a) Erdős-Rényi, (b) scale-free, and (c) directed tree types of graph.

References


Freeman Linton (1977) A set of measures of centrality based on betweenness. Sociometry 40:35–41


Johnson DB (1975) Finding all the elementary circuits of a directed graph. SIAM J Comput 4(1):77–84


Temperley HN (1981) Graph theory and applications. Ellis Horwood, Chichester
Why We Live in Hierarchies?
A Quantitative Treatise
Zafeiris, A.; Vicsek, T.
2018, XIV, 110 p. 42 illus., 39 illus. in color., Softcover
ISBN: 978-3-319-70481-4