20

Biological Networks

Christian Bachmaier
University of Passau

Ulrik Brandes
University of Konstanz

Falk Schreiber
IPK Gatersleben and
University of Halle-Wittenberg

20.1 Introduction

Biological processes are often represented in the form of networks such as protein-protein interaction networks and metabolic pathways. The study of biological networks , their modeling, analysis, and visualization are important tasks in life science today. An understanding of these networks is essential to make biological sense of much of the complex data that is now being generated. This increasing importance of biological networks is also evidenced by the rapid increase in publications about network-related topics and the growing number of research groups dealing with this area. Most biological networks are still far from being complete and they are usually difficult to interpret due to the complexity of the relationships and the peculiarities of the data. Network visualization is a fundamental method that helps scientists in understanding biological networks and in uncovering important properties of the underlying biochemical processes. This chapter therefore deals with major biological networks, their visualization requirements and useful layout methods. We start with some basic biology and important biological networks.

20.1.1 Molecular Biological Foundations

A cell consists of many different (bio-)chemical compounds. A crucial macromolecule in organisms is DNA (deoxyribonucleic acid), which is the carrier of genetic information. But DNA itself is not able to provide the structure of a cell, to act as a catalyst for chemical reactions or to sense changes in the cell's environment. Such functions are carried out by proteins, large molecules which are built according to information stored in DNA sequences. The central dogma of molecular biology deals with the information transfer from DNA to proteins. It states that proteins do not code for the production of other proteins, DNA

or RNA (ribonucleic acid), i.e., that information cannot be transferred from one protein to another protein directly or from a protein back to nucleic acid. Instead, the standard pathway of information flow is from DNA to RNA to protein. Genes represented by DNA sequences are transcribed into RNA sequences which are then translated into proteins, see Figure 20.1. These proteins have different types such as structural components (which give cells their shape and help them move), transport proteins (which carry substances such as oxygen), enzymes (which catalyze most chemical processes in cells and help change metabolites into each other) and regulatory proteins (which regulate the expression of other genes). Crick summarized the standard pathway of information flow as "DNA makes RNA, RNA makes protein and proteins make us" [Kel00].

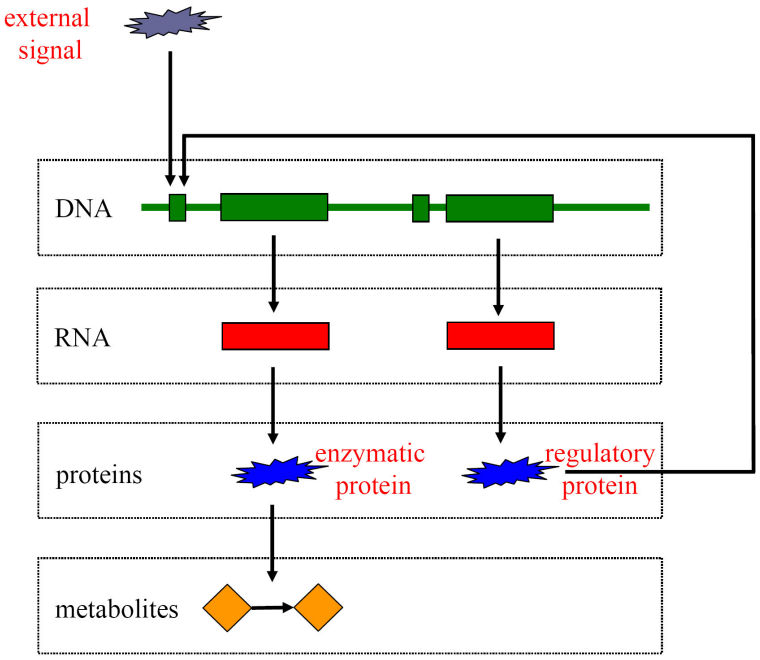


Figure 20.1 The standard pathway of information flow: DNA→RNA→protein. Two kinds of proteins (enzymatic and regulatory proteins) are shown as well as two types of gene regulation (via regulatory protein and external signal).

20.1.2 Biological Networks

Several highly important biological networks are related to molecules such as DNA, RNA, proteins and metabolites and to interactions between them. Gene regulatory and signal transduction networks describe how genes can be activated or repressed and therefore which proteins are produced in a cell at a particular time. Such regulation can be caused by regulatory proteins or external signals. The related networks are considered in Section 20.2. Protein-protein interaction networks represent the interaction between proteins such as the building of protein complexes and the activation of one protein by another protein. Section 20.3 deals with these networks and their visualization in detail. Metabolic networks

show how metabolites are transformed, for example to produce energy or synthesize specific substances. Metabolic and closely related networks are studied in Section 20.4. In Section 20.5 we consider phylogenetic trees, special networks or hierarchies which are often built on information from molecular biology such as DNA or protein sequences. Phylogenetic trees represent the ancestral relationships between different species. They are used to study evolution, which describes and explains the history of species, i.e., their origins, how they change, survive, or become extinct. Finally, signal transduction, gene regulatory, protein-protein interaction and metabolic networks interact with each other and build a complex network of interactions; furthermore these networks are not universal but species-specific, i.e., the same network differs between different species. These topics are discussed in Section 20.6.

Often established layout methods as described in the previous chapters are used to visualize biological networks. Sometimes these methods are slightly modified, e.g., by adding extra forces to force-directed approaches. We will not discuss all these modifications in detail for each network, instead we focus on two topics: metabolic networks and phylogenetic trees. Metabolic networks have been studied for a long time in biology and biochemistry, and specific visualization requirements are given, e.g., by established drawing styles. We present some algorithmic extensions of the hierarchical layout approach which aim to fulfil these requirements. Phylogenetic tree visualizations are quite different to usual tree drawings. Therefore we discuss specific algorithms which have been developed to produce information-rich layouts of phylogenetic trees.

20.2 Signal Transduction and Gene Regulatory Networks

A key issue in biology is the response of a cell to internal and external stimuli and the subsequent regulation of its genetic activity. Signal transduction and gene regulatory pathways and networks describe processes to coordinate the cell's response to such stimuli. Here we consider both networks together as the underlying mechanisms have many similarities, the networks share some common elements and both often result in the regulation of gene expression. Consequently, similar visualization approaches are used for signal transduction and gene regulatory pathways and networks.

20.2.1 Definition

Signal transduction is a communication process within a cell to coordinate its responses to an environmental change. The *stimulus* comes from the cell's environment, e.g., molecules such as hormones. The *response* is a reaction of the cell, e.g., the activation of a gene or the production of energy. A *signal transduction pathway* is a directed network of chemical reactions in a cell from a stimulus (an external molecule which binds to a receptor on the cell membrane) to the response (e.g., the activation of a gene). Here we focus on signal transduction pathways that aim at transcription factors and thus alter the expression of genes in a cell. The *signal transduction network* of a cell is the complete network of all signal transduction pathways. A *signaling cascade* is a process where signal transduction involves an increasing number of molecules in the steps from the stimulus to the response.

Gene regulation is a general term for cellular control of the synthesis of proteins at the transcription step. Gene regulation can also be seen as the response of a cell to an internal stimulus. Often one gene is regulated by another gene via the corresponding protein (called transcription factor), thus gene regulation is coordinated in a *gene regulatory network*. This network directs the level of expression for each gene in the cell by controlling whether and

how often that gene will be transcribed into RNA. Similar to signaling cascades in signal transduction networks a gene can activate more genes in turn and an initial stimulus can trigger the expression of large sets of genes.

As mentioned above we study signal transduction and gene regulation together. Figure 20.1 sketches both processes with signal transduction going from an external signal via several steps to the activation of a gene as one possible response and gene regulation going from a gene via a protein to another gene.

Events of signal transduction and gene regulatory processes occur in different parts of a cell (cellular compartments). To represent compartments these networks can be modeled as clustered graphs. A clustered graph $C = (G, T)$ consists of a directed graph $G = (V, E)$ and a rooted tree T, such that the leaves of T are exactly the nodes of G. The nodes $v \in V$ of the graph are chemical and biochemical compounds (ranging from ions, to small molecules, macromolecules and genes) and the edges $e \in E$ are biochemical events (e.g., binding, transportation and reaction). The occurrence of signal transduction and gene regulatory events in different cellular compartments can be modeled be the tree T. Each node $t \in T$ represents a cluster of nodes of G consisting of the leaves of the subtree rooted at t. The modeling of such networks based on clustered graphs can be used for cluster-preserving layout algorithms [EH00]. However, as it is only partly known in which compartment an event occurs, signal transduction and gene regulatory processes are usually modeled by graphs. The pathways and networks can be derived from databases such as KEGG [KGKN02, KGH$^+$06] and TransPath [KVC$^+$03] (for an overview of biological databases see, for example, [CG10]).

20.2.2 Visualization Requirements

Important goals of the visualizations of signal transduction and gene regulatory pathways are the understanding of the regulation of cellular processes by external and internal signals, the flow of information through the pathways and networks, the interconnection of genes, the discovering of master-genes responsible for the regulation of larger sets of genes, and the identification of main and alternative regulatory paths.

The main visualization requirements are:

- *Pathways*: The main direction of the processes (e.g., from top to bottom) should be clearly visible to express the temporal order of the events.
- *Compartments*: Events of signal transduction and gene regulation occur in different cellular compartments and this information should be visually represented.
- *Complexes*: Especially during signal transduction one event occurring frequently is the building of molecular complexes. Their structure and how they are built by interacting molecules should be displayed.

Signal transduction and gene regulatory pathways often contain metabolic reactions, therefore the visualization requirements discussed in Section 20.4 are also of interest. However, there is no need for the consideration of open and closed cycles (see Section 20.4.2) and usually co-substances are not considered.

20.2.3 Layout Methods

There are two established approaches to visualize signal transduction and gene regulatory pathways and networks: force-directed and hierarchical layout methods. It should be noted that some visualizations of gene regulatory networks in books and articles also use orthogonal or grid-based drawing styles.

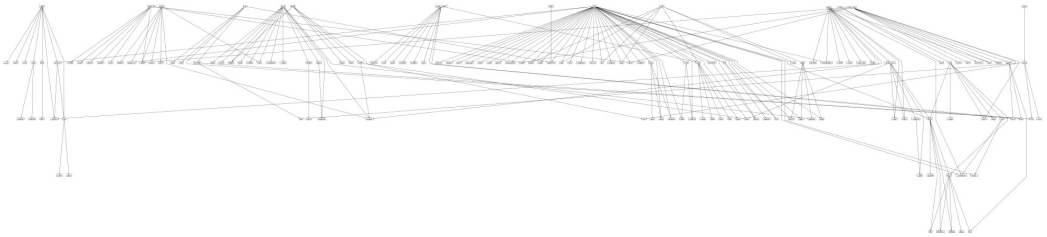


Figure 20.2 A hierarchical layout of a part of the gene regulatory network of *E. coli*.

There are some systems supporting force-directed layouts for the visualization of signal transduction and gene regulatory pathways and networks. These tools are either based on re-implementations of well-known algorithms or on existing layout libraries. Usually the visualizations do not meet the main requirements, especially the main direction and the consideration of compartments. There are a few approaches to improve the general force-directed method. Examples are the PATIKA system [DBD+02, GD06] where the force-directed layout has been extended to deal with several application specific requirements, e.g., cellular compartments, and the approach presented in [SDMW09] where placement, directional, compartmental and other constraints are considered.

Another common approach for the visualization of signal transduction and gene regulatory networks are graph drawing solutions based on hierarchical layout methods, see Figure 20.2. There exist several systems which use hierarchical layouts for the visualization of these networks, e.g., TransPath [KVC+03]. Most are based on existing layout libraries such as dot [KN95] and Pajek [BM02]. These approaches meet some visualization requirements such as the main direction of pathways.

20.3 Protein-Protein Interaction Networks

Proteins are one of the most important molecule groups for living cells. For example, they serve as enzymes for catalysis of metabolic processes, signaling substances (hormones), structural or mechanical material (hair), or transporters for other substances (oxygen). The primary structure of a protein is a long sequence out of essentially twenty different *amino acids* connected by *peptide bonds*.

20.3.1 Definition

A protein can interact with another protein, e.g., to build a protein complex or to activate it. *Protein-protein interactions* form large networks. Their visualization aids biologists in pinpointing the role of proteins and in gaining new insights about the processes within and across cellular processes and compartments, e.g., for formulating and experimentally testing specific hypotheses about gene function.

Often only the existence of an interaction between two proteins is known, but the interaction type, such as activation, binding to, or phosphorylation, remains unknown. However, for the understanding of biological processes, information about the interaction type is crucial, although up to now databases contain little information about that. Therefore we define a protein-protein interaction network as a directed graph $G = (V, E, \tau)$ where V is the set of proteins, E the set of directed interactions (the initiator defines the source), and $\tau\colon E \to T$ defines the type of each edge (interaction type). Protein-protein interaction networks can be derived from databases such as BIND [BDH03] and DIP [XFS+01].

20.3.2 Visualization Requirements

Important goals of the visualization of protein-protein interaction networks are the understanding of the overall structure of the interactions, the interactions between two proteins, and the functions of proteins by investigating the functions of their neighbors or of all proteins within a cluster the protein belongs to. These networks are inherently complex: large, non-planar with many edge crossings, many separate components, and nodes of a wide range of degrees [HJP02]. Thus, the main visualization requirements are the common aesthetic criteria for graph layouts such as even node distribution, symmetry, uniform edge lengths, or Euclidian distances reflecting graph-theoretic distances.

20.3.3 Layout Methods

The established approach for the visualization of protein-protein interaction networks is the force-directed layout method. For drawing networks where interactions are not typed or not of interest accelerated force-directed methods are used: Basalaj and Eilbeck [BE99] use an incremental multidimensional scaling heuristic [Bas99] and Han, Ju and Park [HJP02] use Walshaw's algorithm [Wal02], which is a multi level variant of the original algorithm of Fruchterman and Reingold [FR91]. Both algorithms can generate two and three dimensional drawings. For example, Figure 20.3 shows a force-directed layout of interactions in yeast (Saccharomyces cerevisiae).

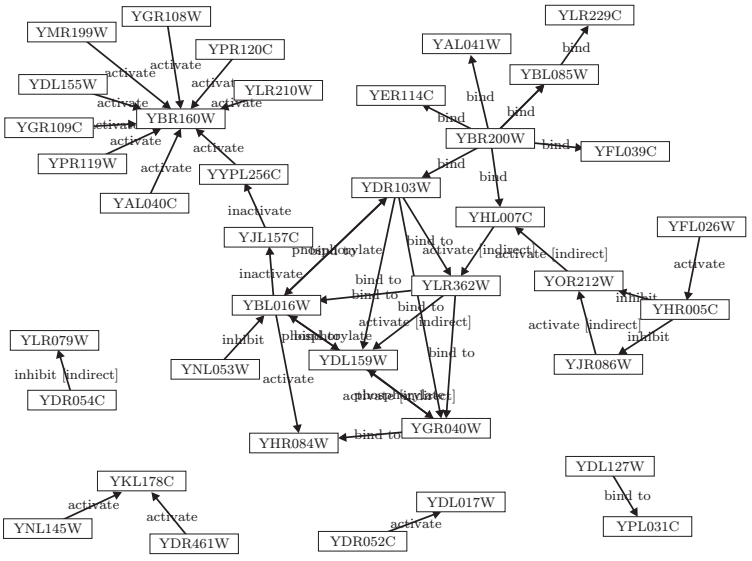


Figure 20.3 A force-directed 2D layout of protein-protein interactions in yeast (redrawn from [FS03]).

However, the general methods cannot cope well with the complexity of protein-protein interaction networks containing typed interactions. In those networks it is not only necessary to show the interactions, but also to explore their different type. For computing visual representations of a network depending on the type of interaction a combination of circular and force-directed algorithms has been suggested [FS03]: Proteins not supporting a selected type of interaction $t \in T$ are placed on an outer circle, whereas proteins that support

that type, i.e., to which an edge of type t is incident, are clustered inside the circle, see Figure 20.4. Thereby the radius of the circle is chosen as big as possible while still fitting in the drawing canvas. As the node labels have a font and thus a fixed height, the circular placement is done with constant vertical distance between them rather than with equal distribution. In the second phase, the positions of the nodes which are involved in the selected interaction are recomputed. Let $G' = (V', E')$ with $E' = \{ e \in E \mid \tau(e) = t \}$ and $V' \subseteq V$ the set of vertices adjacent to an edge in E' be the subgraph representing the interaction t. Based on a variation of the force-directed GEM layout [FLM95] the drawing of G' is generated. GEM optimizes minimal node distances and constant edge lengths while it also tends to display symmetries. However, the gravity force to attract nodes to the center is not suitable to keep all nodes in V' inside the circle. Either the gravity force has to be set so high that it distorts the drawing, or it is not strong enough to prevent nodes from escaping the circle. Thus, a reflective barrier at 80% of the circle radius is introduced. Any node which is about to leave this perimeter is reflected toward the interior of the circle while the energy acting on it is slightly dampened.

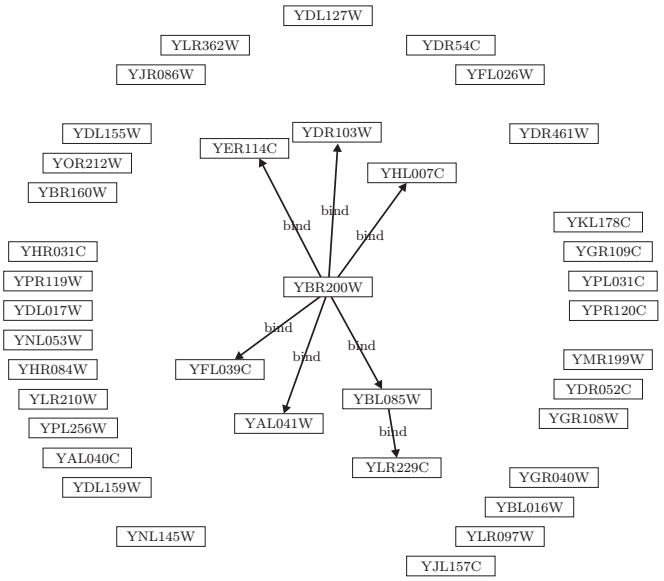


Figure 20.4 The graph of Figure 20.3 with focus on interaction "bind" (redrawn from [FS03]).

While working with a visualization focusing on a special type of interaction, users build a mental map of the picture. Thus, when working with a dynamic visualization tool which allows frequent changes of the interaction type of interest, it is important to help the user in maintaining the mental map. In the described method [FS03] animations are used to provide smooth transitions between different visualizations and ensure that the position of the nodes on the outer circle are fixed over all types of interactions. After computing the new drawing, the nodes are moved on straight lines from their initial positions to their final positions. Thereby the node speed is increased in the beginning and decreased toward the end to allow an easy perception. Edges which have been visible in the initial drawing fade into the background while newly active edges fade from background to foreground color.

20.4 Metabolic Networks

Metabolic reactions are fundamental to life processes, e.g., for the production of energy and the synthesis of substances. A huge number of reactions occur at any time in living cells and the product of one reaction is usually used by another reaction, thus metabolic reactions are strongly interconnected and form metabolic pathways and networks.

20.4.1 Definition

A *metabolic reaction* R is a transformation of chemical substances or metabolites (*reactants*) into other substances (*products*) usually catalyzed by *enzymes*. In general metabolic reactions are reversible, that is, they occur in both directions. Such reactions are characterized by a steady state, i.e., if occurring isolated they reach a state where the amount of change in both directions is equal. A cell is in a constant exchange of substances with its environment. Furthermore, many reactions are regulated, i.e., they are suppressed or enhanced by other factors (allosteric control). This shifts the steady state and together with the steady supply of substances from outside and their final use, e.g., by exporting them from the cell, one can consider a main direction of a reaction. This is also expressed by the differentiation of substances into reactants and products. As already seen, metabolic reactions interact with each other, i.e., the product of one reaction is usually a reactant of another reaction. A *metabolic path* $P = (R_1, \ldots, R_n)$ is a sequence of metabolic reactions where for all $1 \leq i < n$ at least one product of reaction R_i is a reactant of reaction R_{i+1}. The *metabolic network* or *metabolism* of a particular cell or an organism is the complete network of metabolic reactions of this cell or organism. A *metabolic pathway* is a connected sub-network of the metabolic network either representing specific processes or defined by functional boundaries, e.g., the network between an initial and a final substance as shown in Figure 20.5.

From a formal point of view a metabolic pathway is a hyper-graph. The nodes represent the substances and the hyper-edges represent the reactions. A hyper-edge connects all substances of a reaction, is directed from reactants to products and is labeled with the enzymes that catalyze the reaction. Hyper-graphs can be represented by bipartite graphs. Additionally to the nodes representing substances, the reactions are nodes (either labeled with the enzymes or with further nodes for enzymes) and edges are binary relations connecting the substances of a reaction with the corresponding reaction node. This is a common modeling of metabolic pathways, e.g., for their simulation using Petri-nets [HT98, RML93]. For the analysis and visualization of metabolic pathways substances are often divided into two types [MZ03]: *main* substances and *co*-substances. Co-substances are usually small or current metabolites, e.g., ATP, ADP, H_2O, NH_3 and NADH. These substances normally transfer electrons or functional groups such as phosphate and amino groups [NIS90]. Main substances are all other metabolites. However, this is not a global property but is given according to the reaction [MZ03], and a small metabolite such as ATP may be considered as main substance in a particular reaction. For visualization purposes this distinction is important as main substances and co-substances are often differently visually represented.

Here a metabolic pathway is modeled as directed bipartite graph $G = (V_S, V_R, E)$ with nodes $u_1, \ldots, u_n, w_1, \ldots, w_m \in V_S$ representing substances, nodes $v \in V_R$ representing reactions (including the enzyme(s) catalyzing the reaction) and directed edges $(u_1, v), \ldots, (u_n, v)$, $(v, w_1), \ldots, (v, w_m) \in E$ representing the transformation of substances $u_1, \ldots, u_n$ to substances $w_1, \ldots, w_m$ by the reaction v. A reversible reaction does not contain backward edges as in some models for simulation purposes, instead this property of an reaction is represented by an attribute. Another attribute is used to mark main and co-substances.

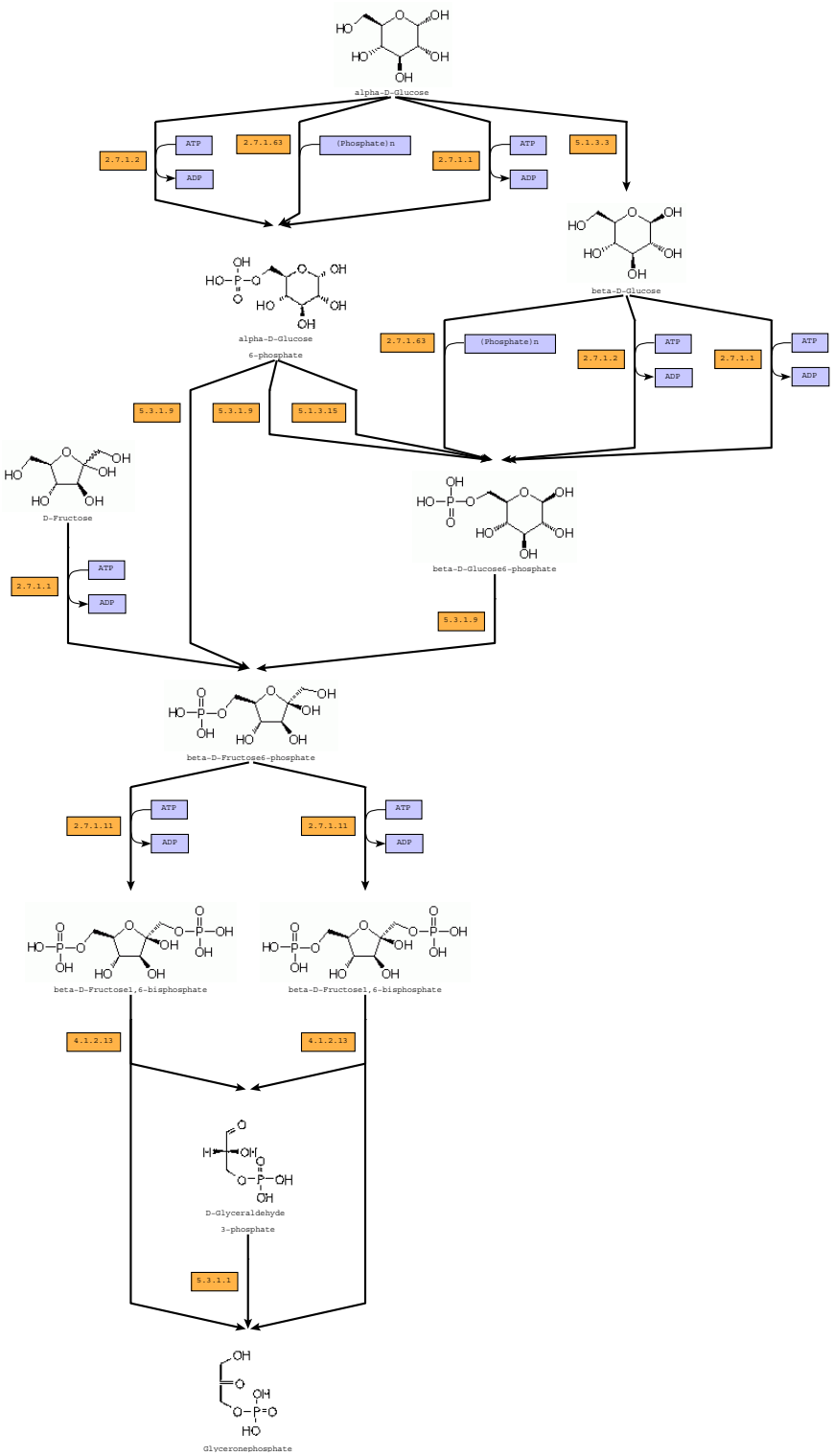
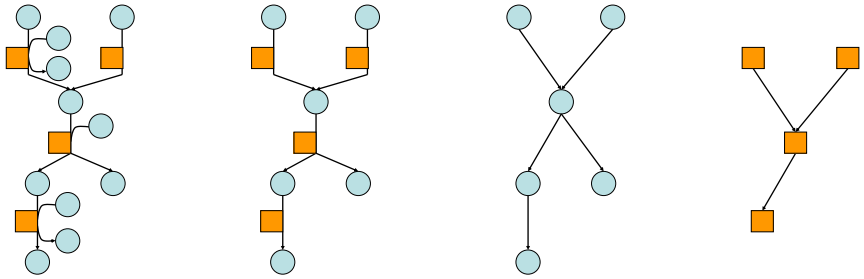


Figure 20.5 An example of a metabolic pathway.

There are several networks which are closely related to metabolic pathways or networks (see Figure 20.6):

- *Simplified metabolic network* : A network which contains reactions, enzymes and main substances, but no co-substances.
- *Metabolite network and simplified metabolite network*: A network which consists only of substances (metabolites); in the simplified case only of main substances.
- *Enzyme network* : A network which consists only of the enzymes catalyzing the reactions.



(a) (b) (c) (d)

Figure 20.6 A metabolic network (a) and corresponding networks: (b) the simplified metabolic network, (c) the simplified metabolite network and (d) the enzyme network. Circles denote metabolites and rectangles represent enzymes

These networks are not always directly associated with a metabolic network. For example, the metabolites in a metabolite network are not necessarily connected according to the reactions of a metabolic network, but can be established by correlation analysis of metabolite profiles [KWLF01]. An enzyme network can be derived from a protein-protein interaction network. Again for relations in such a network a corresponding (connecting) substance cannot always be found within the metabolic network and protein-protein interaction networks may be undirected.

Metabolic pathways can be derived from several databases such as EcoCyc [KRS+00], UM-BBD [EHW00], and MetaCrop [GBWK+08]. For an overview and comparison between different databases see the work of Baxevanis, Wittig and De Beuckelaer [Bax03, WB01]. Simplified metabolic networks are widely used, a popular example is the KEGG/LIGAND database [KGKN02].

20.4.2 Visualization Requirements

The focus of this and the following section is the visualization of (simplified) metabolic pathways and networks. Undirected metabolite networks and enzyme networks as a subset of protein-protein interaction networks have been discussed in Section 20.3.

Visual representations of metabolic pathways are widely used and help scientists to understand the complex relationships between the components of the networks. However, the style of pathway visualizations varies significantly [Mic98]. Examples are biochemical and biological textbooks [Cam96, LNC93, Mic99], pathway posters [Mic93, Nic97] and electronic

databases [ABH94, KGKN02, OLP+00]. Visualizations of metabolic pathways should help understanding the interconnections between metabolites, analyzing the flow of substances through the network and identifying main and alternative paths. The established presentation styles and discussions with users result in several visualization requirements [Sch02]:

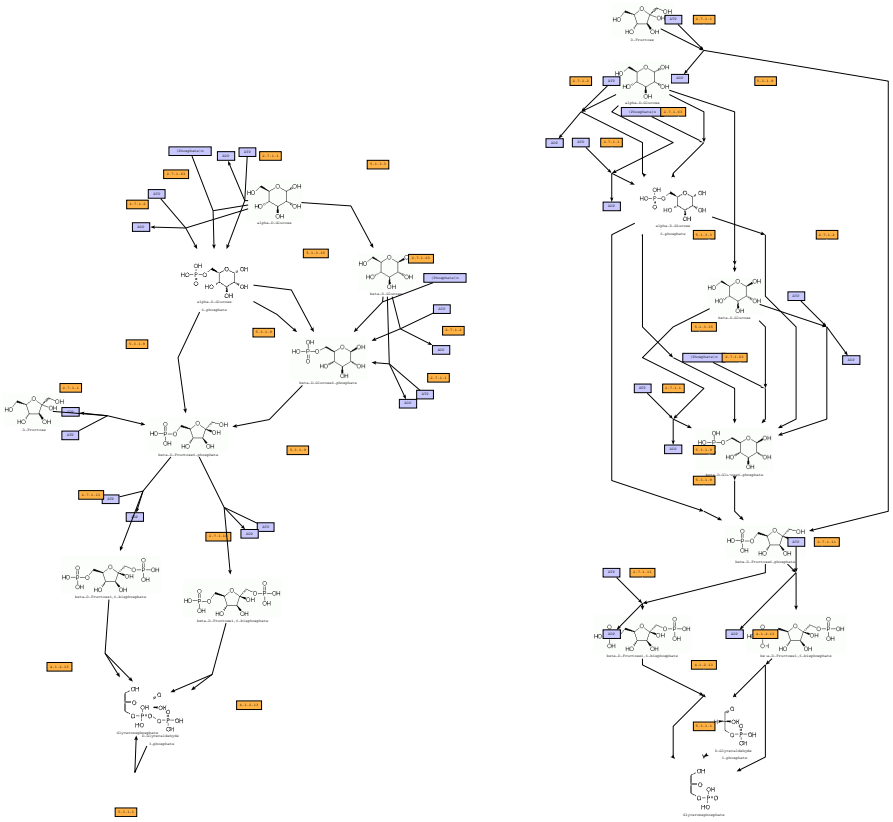
1. *Parts of reactions*: The display of substances and enzymes is application and user-specific. Usually for main substances their name, structural formula or both should be shown. Co-substances should be displayed using their name or abbreviation and enzymes should be represented by their name or EC-number [Int92].
2. *Reactions*: The reaction arrow(s) should be shown from the reactants to the products with enzymes placed on one side of the reaction arrow and co-substances on the opposite side. The reversibility of a specific reaction should be clearly visible. For co-substances their temporal order, which depends on the reaction mechanism, is important, and they should be placed according to this order.
3. *Pathways*: The main direction of reactions (e.g., from top to bottom) should be clearly visible to express the temporal order of reactions. There are important exceptions to the main direction used for the visualization of specific pathways, e.g., the citrate acid cycle or the fatty acid synthesis. The structure of these cyclic reaction chains should be emphasized. Such pathways are characterized by the continuous repetition of a reaction sequence in which the product of the sequence re-enters in the next loop as a reactant. There are two mechanisms. First, the reactant and the product of the reaction sequence are identical from loop to loop (e.g., citrate acid cycle)— a mechanism called a *closed cycle*. Second, the reactant of the reaction sequence varies slightly from the product (e.g., fatty acid cycle) - this is called an *open cycle*.

Besides usual quality criteria, e.g., low number of edge crossings, these visualization requirements result in some specific layout criteria: the hierarchical placement of nodes depending on the structure of the network, the treatment of nodes of varying sizes and the consideration of layout constraints for the order of co-substances and the visualization of specific pathways. Often closed and open cycles are displayed as circles and spirals, respectively. In a spiral related reaction steps from different loops and corresponding substances are placed side by side to emphasis the cyclic structure. As this drawing style needs much space and makes it difficult for a user to trace the reaction sequence of long pathways, an alternative visualization would be to unravel the spiral and align related reactions and substances horizontally.

20.4.3 Layout Methods

There are two established approaches to visualizing metabolic pathways and networks: force-directed and hierarchical layout methods.

Force-directed methods are often used and several pathway analysis tools support such layout. Frequently they visualize not only metabolic and metabolite pathways, but different types of biochemical pathways and networks. Examples are PathwayAssist [NEDM03], PathDB [MBF+00] and pathSCOUT [MdRW03]. These tools use either their own implementations of well-known algorithms or are based on existing layout libraries. For example, VisANT [HMWD04] contains an algorithm based on the layout method of Eades [Ead84], and the method described by Rojdestvenski [Roj03] is based on the force-directed method of Kamada and Kawai [KK89]. On the other hand Cytoscape [SMO+03] uses the yFiles li-



(a) (b)

Figure 20.7 Visualizations of the metabolic pathway shown in Figure 20.5 using (a) a force-directed algorithm [KK89] and (b) a hierarchical approach [STT81].

brary [WEK01] and the layout of BioJAKE [SMKS99], a tool for the creation, visualization and manipulation of metabolic pathways, which is based on Graphviz [EGK+01].

Force-directed approaches do not meet the visualization requirements described in the previous section and visualizations based on this method are very different to the diagrams in posters and books, see Figure 20.7 (a). Different node sizes, the special placement of co-substances and enzymes, the partitioning of substances into reactants and products as well as the general direction of pathways are not considered. A few approaches extend this layout method to deal with application specific requirements. Advanced approaches are the algorithms described in [DBD+02, GD06] where directional and rectangular regional constrains are considered which can be used to enforce different node types (e.g., main and co-substances), layout directions and subcellular locations (cellular compartments), and in [SDMW09] where placement, directional, compartmental and other constraints are considered.

The second layout method for (simplified) metabolic pathways is hierarchical layout. Tools supporting this layout are largely based on existing libraries. Such solutions show the main direction of reactions and are sometimes able to deal with different node sizes. However, there is no specific placement of co-substances, furthermore, open and closed cycles are not emphasized. Figure 20.7 (b) shows a typical example of such a visualization. For example, PathFinder [GHM+02] is restricted to acyclic pathways which are modeled as

directed acyclic graphs and drawn using the VCG library [San95]. The hierarchical layout of BioMiner [SSE+02] is based on yFiles [WEK01]. Some improved approaches consider cyclic structures within the network or depict pathways of different topology using different layouts, e.g., linear, circular and tree structured. Becker and Rojas [BR01] present a graph layout algorithm for drawing metabolic pathways which emphasizes cyclic structures. However, these cycles are computed based on the topology of the network and not on biological knowledge. Therefore pathways may be shown as circles even if they are not closed cycles and closed cycles may not be emphasized by this method, e.g., if they contain shortcuts within the cycle. Furthermore, open cycles are not considered. PathDB contains a component for the visualization of metabolic pathways based on hierarchical layout which allows co-substances to be represented in a smaller font on the side of the reaction arrow [Men00, MBF+00].

The most advanced algorithms try to consider all the visualization requirements discussed in Section 20.4.2. The approach of Karp *et al.* [KP94, KPR02] based on the Grasper-CL system [KLSW94] depicts pathways of different topology using different layout algorithms (linear, circular, tree, hierarchical). It places co-substances and enzymes beside reaction arrows, but has restrictions concerning the order of co-substances or the layout of open cycles. Another approach [Sch02] extends the hierarchical layout for different node sizes; consideration of co-substances and enzymes and special layout of open and closed cycles is implemented in the BioPath system [BFP+04]. The algorithm temporarily builds larger nodes containing the layout of co-substances and enzymes for each reaction, extends the layering step of hierarchical layouts by a local layering [FS04] and the crossing reduction step by constraint crossing reduction [For04]. A drawing produced with this method is shown in Figure 20.5.

The extensions of layering and crossing reduction are of interest also for other graph drawing applications. Usually the layering step of hierarchical layouts computes a global layering, i.e., a layering where nodes belong to a particular layer depending on the topological sorting of the graph. Global layering of graphs tends to produce large drawings as the distance between two layers is determined by the highest node of the layer. An algorithm to compute a local layering, i.e., a layering where each node may be assigned to its own layer depending only on the layers of its direct predecessors and their particular heights is shown in Figure 20.8. It computes the layers from top to bottom. The y-coordinate of a node, i.e., the upper boundary of the rectangle representing the node, and its layer are computed together. Nodes can be split such that a high node may belong to a number of consecutive layers. To reduce the number of layers and dummy nodes layers are joined together if they are situated in an area starting from the current layer with depth y_d. For local and global layering the final part is the replacement of each edge-layer crossing by a dummy node in order to compute a so called *proper* layering. This part is not shown in the algorithm, but takes $\mathcal{O}(|V| * |E|)$ in both the global and the local layering method. This is also the overall running time for these algorithms.

For constraint crossing reduction Forster [For04] presents a heuristic shown in Algorithm 20.9 which extends the well known barycenter heuristic [DETT99]. It starts with partitioning the node set V_2 into ordered node lists with one singleton list $L(v) = \langle v \rangle$ for each node v. Later these lists are pairwise concatenated according to violated constraints. Each violated constraint $c = (s, t)$, i.e., a constraint that node s should be placed left of node t, is removed. The lists containing s and t are concatenated in the required order and treated as a cluster of vertices. The nodes s and t are replaced by a node v_c to represent the concatenated list $L(v_c) = L(s) \circ L(t)$. This node has a barycenter value which is computed as if all edges incident to a node in $L(v_c)$ were incident to v_c. After all violated constraints have been removed the remaining nodes/node lists are sorted according to their barycenter

Input: $G = (V, E)$, height of nodes ($h\colon V \to \mathbb{R}$), minimum node distance d, depth of area where layers are joint y_d
Output: Coordinates $y\colon V \to \mathbb{R}$ and layers $l\colon V \to \mathbb{N}$
Data: Min-heap H, counter $c\colon V \to \mathbb{N}$ for the nodes

```
y ← y_next ← 0; l ← 0
for all v ∈ V do
  c(v) ← indegree(v); h(v) ← h(v) + d
  if c(v) = 0 then
    H.insert(v)
  end if
end for
while !H.isEmpty() do
  {Place nodes on current and consecutive layers within y_d in one layer}
  l ← l + 1; y ← y_next
  v ← H.delmin(); l(v) ← l; y(v) ← y
  y_next ← y + h(v)
  while (y + h(H.top())) ≤ (y_next + y_d) do
    v ← H.delmin();
    l(v) ← l; y(v) ← y
    for all u ∈ children(v) do
      c(u) ← c(u) − 1;
    end for
  end while
  y_next := y + h(v);
  for all v ∈ H do
    {Split large nodes (on this and next layer)}
    In G = (V, E) replace v by v_1, v_2 and the edge (v_1, v_2);
    l(v_1) ← l; y(v_1) ← y; h(v_1) ← y_next − y; h(v_2) ← h(v) − h(v_1)
    Replace in heap H node v by node v_2
  end for
  for all v ∈ V do
    if v ∉ H and v not already placed and c(v) = 0 then
      H.insert(v)
    end if
  end for
end while
```

Figure 20.8 Computing a local layering of the nodes

value. The result is a vertex permutation that satisfies all constraints and has few crossings. During the algorithm the violated constraints have to be considered in an order which avoids the generation of constraint cycles. This is done by the procedure FIND-VIOLATED-CONSTRAINT(V, C) and with the $\mathcal{O}(|C|)$ algorithm for this procedure [For04] the running time of the complete algorithm is $\mathcal{O}\left(|V_2| \log |V_2| + |E| + |C|^2\right)$.

20.5 Phylogenetic Trees

A fundamental issue in biology is the hierarchical classification of organisms in an evolutionary context, i.e., reconstruction of ancestral relationships between different *taxons*, e.g.,

Input: A two-level graph $G = (V_1, V_2, E)$, acyclic constraints $C \subseteq V_2 \times V_2$
Output: A permutation of V_2 (result in L)
Data: singleton lists L and barycenter $b\colon V \to \mathbb{Q}_0^+$ for all nodes

```
for all v ∈ V2 do
  b(v) ← Σ_{u∈V} position(u)/ degree(v)
  L(v) ← ⟨v⟩
end for
V ← { s,t | (s,t) ∈ C }        {constrained vertices}
V' ← V2 − V
while (s,t) ← FIND-VIOLATED-CONSTRAINT(V,C) ≠ ⊥ do
  create new vertex v_c
  degree(v_c) ← degree(s) + degree(t)        {update barycenter value}
  b(v_c) ← (b(s)·degree(s) + b(t)·degree(t))/ degree(v_c)
  L(v_c) ← L(s) ∘ L(t)
  for all c ∈ C do
    if c is incident to s or t then
      make c incident to v_c instead of s or t
    end if
  end for
  C ← C − {(v_c, v_c)}        {remove self loops}
  V ← V − {s,t}
  if v_c has incident constraints then
    V ← V ∪ {v_c}
  else
    V' ← V' ∪ {v_c}
  end if
end while
V'' ← V ∪ V'
sort V'' by b()
L ← ⟨⟩        {concatenate vertex lists}
for all v ∈ V'' do
  L ← L ∘ L(v)
end for
```

Figure 20.9 Computing a constrained crossing reduction

species, genes, or DNA sequences. The common approach for determining such relations is the construction of a phylogenetic tree.

20.5.1 Definition

For hierarchical classification of a set of taxons A there are two common types of approaches: The first are the *phenetic* methods, which have an $|A| \times |A|$ *distance matrix* Δ assigning each pair of taxons a quantitative difference as input. The goal is to group (commonly two) most similar taxons/ancestors and thus to find out how an ancestor of theirs may look like according to the principle of *minimum evolution*. This is done recursively until a common ancestor is reached and a phylogenetic tree is obtained. All these methods are based on clustering and thus explicitly do not consider evolutionary history. The second type of approach is the *cladistic* methods, which have an $|A| \times |M|$ *characteristic matrix* Γ assigning each taxon $|M|$ characteristics like number of legs, ability to fly, or color of skin as input. These methods try to find out the actual genealogy according to a model of

the real evolutionary development assuming that identical characteristics of different taxons indicate a common ancestry.

A *phylogenetic tree* (in literature also called *evolutionary tree*) $T = (V, E, \delta)$ is a tree consisting of nodes V (taxons) and edges E (links). Leave nodes, i.e., nodes with exactly one link, represent species, sequences, or similar entities; they are called *operational taxonomical units* (and are represented by $A \subseteq V$). Internal nodes represent (hypothetical) ancestors generated from phylogenetic analysis; they are called *hypothetical taxonomic units*. The lengths of the edges $\delta\colon E \to \mathbb{R}_0^+$ quantify the biological divergence between the incident nodes, e.g., biological time or genetic distance. Phylogenetic trees are often stored in the Newick file format [Fel95], which makes use of the correspondence between trees and nested parentheses.

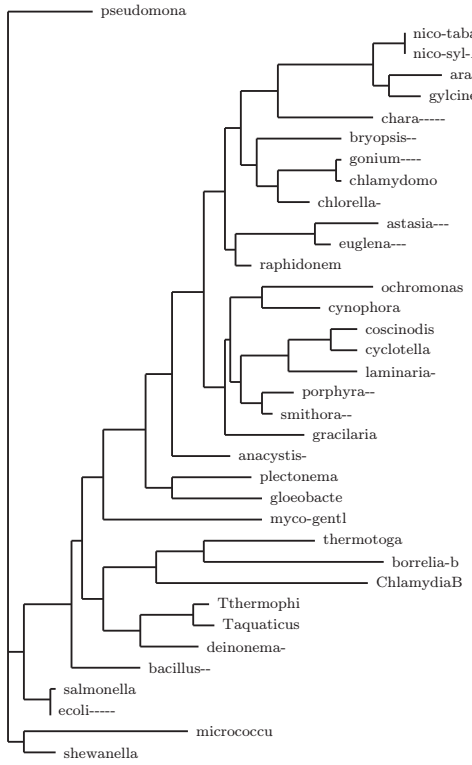


Figure 20.10 An example of a phylogenetic tree (phylogram, redrawn from [DS04]).

A simple phenetic representative for creating a phylogenetic tree $T = (V, E, \delta)$ is the $\mathcal{O}\left(|A|^2 \log |A|^2\right)$ time "Unweighted Pair Group Method with Arithmetic Mean" (UPGMA) [MS57]: Initially define clusters $C \leftarrow \{\, c_i \mid 1 \le i \le |A| \,\}$, each containing one taxon of A, set the cluster sizes $s(c_i) \leftarrow 1$, and let $V \leftarrow C$. Then iterate until there is only one cluster left: Find two closest clusters $c_i \neq c_j$ according to Δ (with the help of a priority queue over the $|A|^2$ elements of Δ). Join the clusters c_i and c_j to a new cluster c_p by $C \leftarrow C \cup \{c_p\} - \{c_i, c_j\}$ with $s(c_p) \leftarrow s(c_i) + s(c_j)$, and add it to T with $V \leftarrow V \cup \{c_p\}$. Introduce new edges $E \leftarrow E \cup \{(c_p, c_i), (c_p, c_j)\}$ with $\delta\left((c_p, c_i)\right) \leftarrow \delta\left((c_p, c_j)\right) \leftarrow \frac{\Delta_{ij}}{2}$. Then compute the distances from c_p to all clusters $c_k \in C$ with $k \neq i, j$:

$$\Delta_{pk} \leftarrow \Delta_{kp} \leftarrow \frac{s(c_i)}{s(c_p)} \cdot \Delta_{ik} + \frac{s(c_j)}{s(c_p)} \cdot \Delta_{jk} \tag{20.1}$$

At the end of the iteration delete the two columns i and j and the two rows i and j in Δ.

If Δ is an ultrametric matrix, then UPGMA guarantees for the unique way W between any two nodes $v_i, v_j \in V : \sum_{e \in W} \delta(e) = \Delta_{ij}$ and T is said to be ultrametric, too. Otherwise, UPGMA is a heuristic.

Another common phenetic approach is the $\mathcal{O}\left(|A|^3\right)$ time "Neighbor-Joining" (NJ) method [SN87] which is an enhancement of UPGMA especially for protein and nucleotide data (DNA does not evolute by accident, but follows some constraints which can be included in the computation of NJ). The idea of NJ is to join clusters which are not only close to each other, but also far from the rest. The initialization is the same as in UPGMA, whereas the iteration for $|C| > 2$ is the following: For each cluster c_i compute the mean distance to an arbitrary other cluster $c_k \in C$ by $d(c_i) \leftarrow \sum_{k \neq i} \frac{\Delta_{ik}}{|C|-2}$. Find two closest clusters $c_i \neq c_j$ with least $\Delta_{ij} - (d(c_i) + d(c_j))$. Join the clusters c_i and c_j to a new cluster c_p by $C \leftarrow C \cup \{c_p\} - \{c_i, c_j\}$, and add it to T with $V \leftarrow V \cup \{c_p\}$. Introduce new edges $E \leftarrow E \cup \{(c_p, c_i), (c_p, c_j)\}$ with lengths as shown in (20.2) and compute the distances from c_p to all clusters $c_k \in C$ with $k \neq i, j$ with (20.3).

$$\delta\left((c_p, c_i)\right) \leftarrow \frac{1}{2}\Delta_{ij} + \frac{1}{2}\left(d(c_i) - d(c_j)\right), \qquad \delta\left((c_p, c_j)\right) \leftarrow \frac{1}{2}\Delta_{ij} + \frac{1}{2}\left(d(c_j) - d(c_i)\right) \tag{20.2}$$

$$\Delta_{pk} \leftarrow \Delta_{kp} \leftarrow \frac{\Delta_{ik} + \Delta_{jk} - \Delta_{ij}}{2} \tag{20.3}$$

Delete the two columns and the two rows i and j in Δ. If $|C| = 2$, i.e., $C = \{c_s, c_t\}$, then connect $c_s, c_t \in V$ by $E \leftarrow E \cup \{(c_s, c_t)\}$ with $\delta\left((c_s, c_t)\right) \leftarrow \Delta_{st}$ and stop.

A typical representative of the cladistic category is the "Maximum Parsimony" (MP) method. The idea is to define the (non-unique) tree T as optimal, which posits fewest mutations as possible. For the "Small Parsimony" problem the topology of T is already given and only the labels $l(v) = \bigcup_{1 \leq j \leq |M|} l_j(v)$ of the inner nodes $v \in V$, i.e., the position $l_j(v)$ of each characteristic $m_j \in M$ has to be determined. It can be solved in $\mathcal{O}\left(|A||M| \cdot \max\{\,|\operatorname{dom}(m_j)| \mid m_j \in M\,\}\right)$ time [Fit71], where $\operatorname{dom}(m_j)$ is the set of all possible values which a taxon can adopt for m_j. A solution is the following algorithm: Assign each $v_i \in V$ for each $m_j \in M$ in a postorder traversal of T a set $S_j(v_i) \subseteq \operatorname{dom}(m_j)$ with (20.4), where $w_1, w_2 \in V$ are the children of v_i.

$$S_j(v_i) \leftarrow \begin{cases} \Gamma_{ij}, & \text{if } v_i \text{ is a leaf,} \\ S_j(w_1) \cap S_j(w_2), & \text{if } S_j(v_1) \cap S_j(v_2) \neq \emptyset, \\ S_j(w_1) \cup S_j(w_2), & \text{otherwise.} \end{cases} \tag{20.4}$$

In a subsequent preorder traversal of T for each node $v \in V$ which has a parent u with $l_j(u) \in S_j(v)$ set $l_j(v) \leftarrow l_j(u)$. If no such u exists or v is a leaf set $l_j(v)$ to an arbitrary element of $S_j(v)$. The number of (independent) mutations in T is equal to how many times the third item of (20.4) was used.

In the "Weighted Small Parsimony" version the probability of different mutations is not unique, i.e., $p_j(a, b)$ defines the "price" of a change for a characteristic $m_j \in M$ from state $a \in \operatorname{dom}(m_j)$ to $b \in \operatorname{dom}(m_j)$. The goal is not to minimize the number of mutations, but the sum of their prizes while the topology of T again is given. For that we present the $\mathcal{O}\left(|A||M| \cdot \max\{\,|\operatorname{dom}(m_j)| \mid m_j \in M\,\}\right)$ time algorithm [San75], which is a generalization

of [Fit71]: Assign in a postorder traversal of T to each $v_i \in V$ quantities $S_j(v_i, t_k(m_j))$ for each m_j and all values $t_k(m_j) \in \mathrm{dom}(m_j)$ with (20.5) for a leaf v_i and (20.6) for an internal node v_i, where $w_1, w_2 \in V$ are the children of v_i. Considering only mutations of characteristic m_j, then $S_j(v, t_k(m_j))$ is the minimum total cost for the subtree rooted at v_i if $l_j(v_i)$ was set to $t_k(m_j)$.

$$S_j(v_i, t_k(m_j)) \leftarrow \begin{cases} 0, & \text{if } \Gamma_{ij} = t_k(m_j), \\ \infty, & \text{otherwise.} \end{cases} \tag{20.5}$$

$$\begin{aligned} S_j(v_i, t_k(m_j)) \leftarrow \quad & \min\{p_j(t_k(m_j), t) + S_j(w_1, t) \mid t_k(m_j) \neq t \in \mathrm{dom}(m_j)\} \\ & + \min\{p_j(t_k(m_j), t) + S_j(w_2, t) \mid t_k(m_j) \neq t \in \mathrm{dom}(m_j)\} \end{aligned} \tag{20.6}$$

The minimum total cost of T with root r is $\sum_{m_j \in M} \min\{S_j(r,t) \mid t \in \mathrm{dom}(m_j)\}$. In a subsequent preorder traversal of T update the labels of each $v_i \in V$, where u is the parent of v_i:

$$l_j(v_i) \leftarrow \begin{cases} \arg\min\{S_j(r,t) \mid t \in \mathrm{dom}(m_j)\}, & \text{if } v_i = r, \\ \arg\min\{p_j(l_j(u), t) + S_j(v_i, t) \mid t \in \mathrm{dom}(m_j)\}, & \text{otherwise.} \end{cases} \tag{20.7}$$

In contrast to the above, the "Large Parsimony" problem, where the topology of T is not given, is $\mathcal{NP}$-hard, regardless if discrete or weighted. However, there are some heuristics, e.g., [HP82] which uses branch&bound to find the cheapest tree T among all trees. This approach guarantees to find T, but its time complexity is in the worst case exponential in $|A|$ (exhaustive search). Another heuristic is "Nearest Neighbor Interchange" (NNI) [MGB73], which defines a relation between each pair of trees and then uses well-known concepts like greedy algorithms or simulated annealing to find a (local) optimum.

Given a tree T with known edge lengths δ, the likelihood of T is $P(M|T)$. It is a statistical measure of how well it describes the biological data. Let $P_{a \to b}(\delta(e))$ be the probability that character $a \in \mathrm{dom}(m_j)$ will transform to $b \in \mathrm{dom}(m_j)$ within the time $\delta(e)$, $P(a)$ be character frequency of $a \in \mathrm{dom}(m_j)$ fixed throughout biological history, L be the set of all reconstructions of T, i.e., all full labelings of internal nodes, and $r \in V$ be the root of T. Then [Fel73]:

$$P(M|T) = \prod_{j \in M} \left(\sum_{l \in L} \left(\left(P(l_j(r)) \cdot \prod_{(u,v) in E} P_{l_j(u) \to l_j(v)}(\delta((u,v))) \right) \right) \right) \tag{20.8}$$

If the character substitution is reversible, i.e., $P_{a \to b}(\delta(e)) = P_{b \to a}(\delta(e))$, then T is unrooted and r can be chosen arbitrarily without changing $P(M|T)$. The "Maximum Likelihood" method (ML) [Fel73] computes the likelihood of a tree with dynamic programming in $\mathcal{O}(|A||M| \cdot \max\{|\mathrm{dom}(m_j)| \mid m_j \in M\})$ time, i.e., it computes the likelihood of each bifurcation and declares the tree with the greatest sum of likelihoods as the best. There are also statistical methods for computing the optimum edge lengths δ for a given tree T with regard to a maximum tree likelihood [SL99].

The topology of T is fixed. However, there is in most cases the freedom of permutation of each node's children and thus there are $2^{|V|-1}$ possible linear leaf orderings consistent with the structure of a binary T. From a biological view it makes sense to order the leaves such that similar leaves are close together. Remember, the dissimilarity of each pair of leaves is stored in the distance matrix Δ. Therefore, the goal is to minimize the sum of the lengths of the ways from each leaf to each other. In an optimal tree the lengths of

all ways correspond exactly to the entries in Δ. Since in the general case no such optimal tree exists (Δ represents a complete graph and not only a tree), leaf ordering makes sense. It can be done, e.g., with the dynamic programming approach [BJDG+03] which needs $\mathcal{O}\left(4^k|V|^3\right)$ time for a k-ary T. There, an optimal leaf ordering consistent with a binary tree T is determined by a bottom-up computation of subintervals. Define $M(u, w_l, w_r)$ to be the cost of the best linear order of the leaves in the subtree $T(u)$ induced by $u \in V$ that begins with leaf w_l and ends with leaf w_r. If u is a leaf, then $M(u, u, u) \leftarrow 0$. Otherwise, let v_1 and v_2 be the children of u such that $w_l \in T(v_1)$ and $w_r \in T(v_2)$. Then the optimality criterion of (20.9) holds. For a k-ary tree, denote the children of u by $v_1, \ldots, v_p$, $1 \leq p \leq k$. If $w_l \in T(v_1)$ and $w_r \in T(v_p)$, any ordering of $v_2, \ldots, v_{p-1}$ is possible. Thus for each of the $p!$ orderings $M(u, w_l, w_r)$ is computed in the same way as for binary trees by inserting $k-1$ internal binary dummy nodes while maintaining the current order.

$$M(u, w_l, w_r) \leftarrow \min \{ M(v_1, w_l, a_i) + \Delta_{ij} + M(v_2, b_j, w_r) \mid \text{leaf } a_i \in T(v_1), \text{leaf } b_j \in T(v_2) \} \quad (20.9)$$

20.5.2 Visualization Requirements

As seen earlier, the graphs to visualize are directed (and thus rooted) or undirected trees $T = (V, E, \delta)$ with given edge lengths δ. T is either a binary tree or very similar to a binary tree, i.e., there are view nodes with a degree higher than three. Irrespectively of edge direction, T should be laid out hierarchically to visualize the ancestral relationships between taxons. Since the sum over the edge lengths on the unique path from one taxon to another is the evolutionary distance, it is desirable to reflect this in the lengths of the curves drawn for the edges. This means in the most simple case that $\delta(e)$ is the curve length of $e \in E$. Traditional algorithms for drawing trees explicitly do not consider given edge lengths. They follow aesthetic criteria as edges should have the same length and nodes of the same depth should be drawn on the same y-coordinate [RT81, Wal90, WS79] or radius [Ead92]. In most cases the nodes as well as the edges contain labels, which should be drawn non-overlapping. Further a good layout follows common criteria for graph/tree layout like no unnecessary edge crossings, compactness, and use of the entire available drawing area.

As we will see in the next section, some layout methods will use the freedom of permuting children to generate nice drawings. However, if not especially mentioned, we assume to have already a fixed leave ordering given.

Although there is need to edit layouts dynamically [Car04a], e.g., collapsing and expanding subtrees or editing annotations, for an easy understanding of large trees, we restrict ourselves to static layouts for the sake of simplicity. Since there is an ongoing trend to larger trees, which may contain several hundred thousand of nodes, a layout algorithm must be efficient.

20.5.3 Layout Methods

The most common layouts for phylogenetic trees are vertical or circular *dendrograms* or radial drawings [Car04b]. The typical representatives of the first group are the orthogonal *phylograms* (see Figure 20.10), where the tree is drawn hierarchically and from left to right and thus the vertices vertically from top to bottom. Each edge $e = (u, v)$ has exactly one bend b at the x-coordinate of u and at the y-coordinate v. The length of the horizontal edge segment (b, v) represents $\delta(e)$. A parent node is vertically placed, e.g., in the middle between

its extremal children or in the arithmetic mean of all its children. Since the topology of the tree, the horizontal edge lengths, and the leave ordering (and thus the y-coordinates of the leaves) are already fixed, the layout is already fixed and can be computed by the $\mathcal{O}(|V|)$ time algorithm in Figure 20.11. Phylograms are easy to interpret and leave space for edge annotations [Car04b]. *Cladograms* and *curvograms* drawing edges as straight lines or splines are subtypes of phylograms and thus are not treated separately.

Input: $T = (V, E, \delta)$, y-coordinates of leaves
Output: Coordinates $x, y \colon V \to \mathbb{R}$ for the nodes and $x_b, y_b \colon E \to \mathbb{R}$ for the bends
Data: Stack S

```
r ← root(T)
S.push(r)
x(r) ← 0
while !S.isEmpty() do
  v ← S.top()
  if v has an unmarked child w then
    mark w; S.push(w)
    x_b((v,w)) ← x(v)
    x(w) ← x(v) + δ((v,w))
  else
    S.pop()
    if v is an internal node then
      y(v) ← ½ (min{ y(w) | w is a child of v } + max{ y(w) | w is a child of v })
    end if
    if v ≠ r then
      u ← S.top()    {the parent of v}
      y_b((u,v)) ← y(v)
    end if
  end if
end while
```

Figure 20.11 Computing coordinates for drawing a phylogram.

Another style of dendrograms is the *circle layout*, which draws the trees concentric around the root with an unique radius for the leaves. Again, each edge $e = (u, v)$ bends exactly once at the radius of the parent u. The "vertical" segment is drawn as a segment of a circle, whereas the "horizontal" one is an interval of a straight line from the root through the child v, see Figure 20.13. The algorithm for computing a circle layout is similar to Algorithm 20.11 if treating x as levels ($x, x_b \colon V \to \{0, 1, \ldots, \mathsf{height}(T)\}$) with $x(r) = 0$ and y as angles ($y, y_b \colon V \to [0, \ldots, 2\pi]$). Instead of the Cartesian coordinates, the algorithm needs the polar angles of the leaves distributed uniformly on a circle as input. Since the radius now is unique for all leaves, we set $x(w) \leftarrow x(v)+1$ instead of $x(w) \leftarrow x(v)+\delta((v,w))$ for each edge (v, w). This ignores edge lengths δ, however. Another approach [BBS05] which considers edge lengths is to distribute the leaves uniformly on a circle, to set each inner node v on the weighted Cartesian barycenter of its parent u and its children W as shown in (20.10), and to draw each edge as a straight line. See Figure 20.13 for an example. The arising equation system can be solved in $\mathcal{O}(|V|)$ time. Algorithm 20.12 shows the computation in a unit circle. If reordering of the leaves is acceptable, the postorder traversal of the children w of each node v can be ordered according to ascending height of $T(w)$ (in terms of δ) plus

$\delta((v,w))$. This should support the algorithm to draw edges with their desired length, but raises the running time to $\mathcal{O}(|V| \log |V|)$, however. Since even this cannot guarantee exact lengths, the edges are colored, i.e., blue color means too short and red color too large, such that the color saturation reflects the multiplicative failure.

$$((x(v), y(v)) \leftarrow \frac{(x(u), y(u))}{\delta((u,v))} + \sum_{w \in W} \frac{(x(w), y(w))}{\delta((v,w)) \cdot |W|} \tag{20.10}$$

Input: $T = (V, E, \delta)$ with $\delta(e) > 0$ for all edges e
Output: Coordinates $x, y \colon V \to \mathbb{R}$ for the nodes
Data: Coefficients $c \colon V \to \mathbb{R}$, offsets $d \colon V \to \mathbb{R}^2$, and edge weights $s \colon E \to \mathbb{R}$

```
for each v ∈ V if deg(v) = 1 then l ← l + 1
i ← 0
postorder_traversal(root(T))
preorder_traversal(root(T))

procedure postorder_traversal(node v)
  for each child w of v do postorder_traversal(w) {optionally ordered}
  if v is a leaf or (v = root(T) and deg(root(T)) = 1) then
    c(v) ← 0; d(v) ← (cos(2πi/l), sin(2πi/l))    {fix vertex on circle}
    i ← i + 1
  else
    s ← 0
    for each adjacent edge e ← {u, v} do
      if v = root(T) or w is the parent of v then s(e) ← 1/δ(e)
      else s(e) ← 1/(δ(e)·(deg(v)−1))
      s ← s + s(e)
    end for
    t ← t' ← 0
    for each outgoing edge e ← (v, w) do t ← t + (s(e)/s)·c(w); t' ← t' + (s(e)/s)·d(w)
    if v ≠ root(T) then let e be the incoming edge of v; c(v) ← s(e)/(s·(1−t))
    d(v) ← t'/(1−t)
  end if
end procedure

procedure preorder_traversal(node v)
  if v = root(T) do x(v) ← d(v)
  else let u be the parent of v; x(v) ← c(v)·x(u) + d(v)
  for each child w of v do preorder_traversal(w)
end procedure
```

Figure 20.12 Cartesian barycenter method for generating a circle layout.

Circle layouts provide the best use of the available space for trees with more than 100 leaves [Car04b]. Dendrograms in general are a good choice to visualize the leaf ordering.

The second type of drawings are the radial tree drawings [BBS05], which are preferred for visualizing unrooted trees. Their edges are drawn as straight lines. To obtain coordinates for the vertices, Algorithm 20.14 traverses T in preorder (here, breadth first search) from a given root to the leaves. Thereby it assigns each subtree a wedge according to its size,

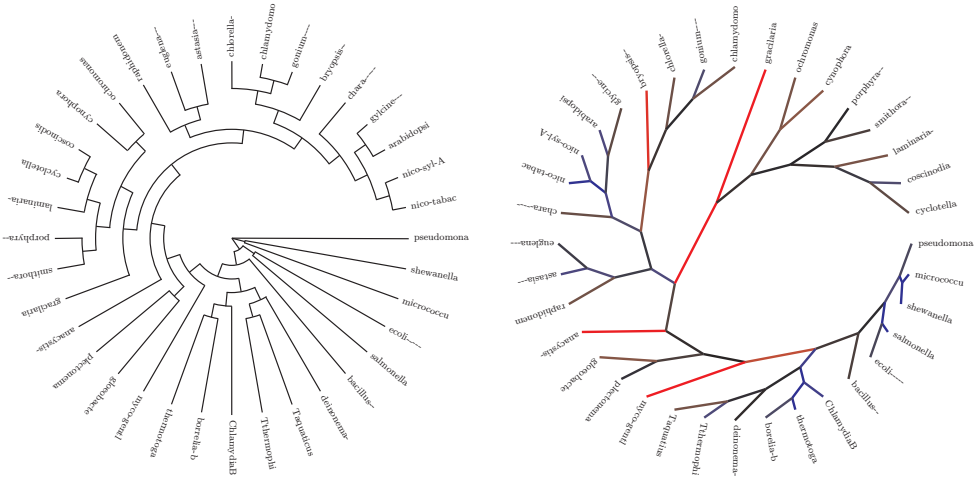


Figure 20.13 Circle layouts with levels and weighted Cartesian barycenter.

i.e., according to its number of leaves (leafcount). Note that here all degree one vertices are treated as leaves. Since the wedge sizes are independent of the root, rerooting the tree only results in a different ordering of the children of the new root.

Input: $T = (V, E, \delta)$
Output: Coordinates $x, y \colon V \to \mathbb{R}$ for the nodes
Data: Queue Q, leafcount$\colon V \to \mathbb{N}^+$ {from a previous postorder traversal}

```
r ← root(T)
Q.insert(r)
rightborder(r) ← 0
wedgesize(r) ← 2π
x(r) ← y(r) ← 0
while !Q.isEmpty() do
  v ← Q.delete_first()
  η ← rightborder(v)
  for each child w of v do
    Q.insert(w)
    rightborder(w) ← η
    wedgesize(w) ← (2π·leafcount(w)) / leafcount(r)
    α ← rightborder(w) + wedgesize(w)/2
    x(w) ← x(v) + cos(α) · δ((v,w)); y(w) ← y(v) + sin(α) · δ((v,w))
    η ← η + wedgesize(w)
  end for
end while
```

Figure 20.14 Computing coordinates for drawing of radial tree drawings.

Clearly, Algorithm 20.14 has an $\mathcal{O}(|V|)$ running time if newly discovered children are distributed in random order around their parent, e.g., as they occur in the adjacency list. Advanced versions use the freedom of reordering the children. The first aims to reach a symmetric layout: For each child v the metric of (20.11) is computed with a postorder

traversal of T. It is a measure of how far the biological development goes on in the induced subtree of v. Alternating, depending on the depth of the parent node, the child with higher value is drawn on the left or on the right side of the corresponding wedge. If the parent has more than two children, then the child with highest value is drawn in the middle and the other children on its left and right side according to descending m. The second method is to put evolutionary closely related children on near positions. For this (20.12) is used to order the children ascending according to average distance of the leaves in the induced subtree to the parent. However, in both cases the running time raises to $\mathcal{O}(|V| \log |V|)$ and ordering of children makes no sense for UPGMA-trees, since each child will have the same m-value.

$$m(v) \leftarrow \begin{cases} \delta((u,v)), & \text{if } v \text{ is a leaf,} \\ \delta((u,v)) + \max\{\, m(w) \mid w \text{ is a child of } v \,\}, & \text{otherwise.} \end{cases} \tag{20.11}$$

$$m(v) \leftarrow \begin{cases} \delta((u,v)), & \text{if } v \text{ is a leaf,} \\ \frac{\sum_{(v,w)} (\delta((u,v)) + m(w))}{|\{\, w | w \text{ is a child of } v \,\}|}, & \text{otherwise.} \end{cases} \tag{20.12}$$

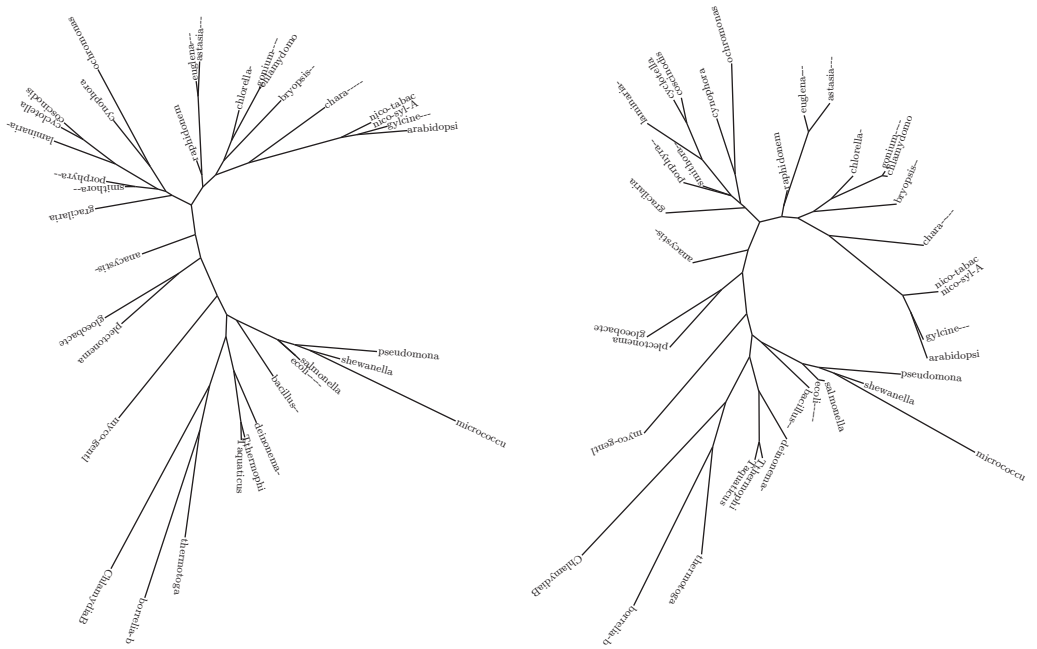


Figure 20.15 Radial tree layout with the same root as in Figure 20.10 and leaf reordering for drawing those closely related near. The right drawing is with spreading.

A lot of space is wasted by simply giving the wedge for a child v from the parent u to v, i.e., the area between the pairwise parallel wedge borders. This can be avoided by spreading (the subtrees induced by) the children w of v to use the full wedge of v originated at u and not at v except of a small buffer. Spreading is done in a postprocessing step and needs $\mathcal{O}(|V|^2)$ time. Each label is drawn as an extension of the incoming edge of the corresponding leaf, i.e., in the corresponding wedge. To leave space for labels in spreaded layouts, the lengths of the labels are added to the δ values of the respective incoming edges,

for computation only. Another more simple solution is to draw the labels with an angle of a ray from the root through the leaves. Figure 20.15 shows a standard and a spreaded layout of our running example. To overcome the problem of zero edge lengths, e.g., incoming edges of ecoli----- or nico-tabac and nico-syl-A, a user definable minimum edge length is useful to indicate edges and to simplify the labeling.

20.6 Discussion

In this chapter we discussed the visualization of biological networks. We focused on important networks closely related to molecular biology: gene regulatory, signal transduction, protein-protein interaction and metabolic networks. Furthermore, we studied the visualization of phylogenetic trees, hierarchies which are often built on information from molecular biology such as DNA or protein sequences. However, there are many more networks in biology: ecological networks such as food-webs, biological data analysis networks such as correlation networks, and neuronal networks to name just a few. Moreover, even for the networks discussed we presented only some visualization aspects.

Other topics of particular importance in the visualization of biological networks are, for example, visual network comparison, exploration of network based phylogenetic trees, visualization of data in the network context, and the exploration of integrated networks. The same network often has to be compared in different organisms for applications such as drug discovery and evolutionary studies. Several methods for the visual comparison of biological networks, especially metabolic pathways, have been already developed [BDS04b, GHM+02, Sch03], see also Figure 20.16. Differences in the network between different species can be used to compute phylogenetic trees [MZ04, HS03] and methods for the interactive visualization and triangulation of this complex structure (a tree built over networks) have been developed [BDS04a].

Advances in high-throughput methods such as metabolite profiling and automatized enzyme assays have increased the need for automatized data analysis and visual exploration

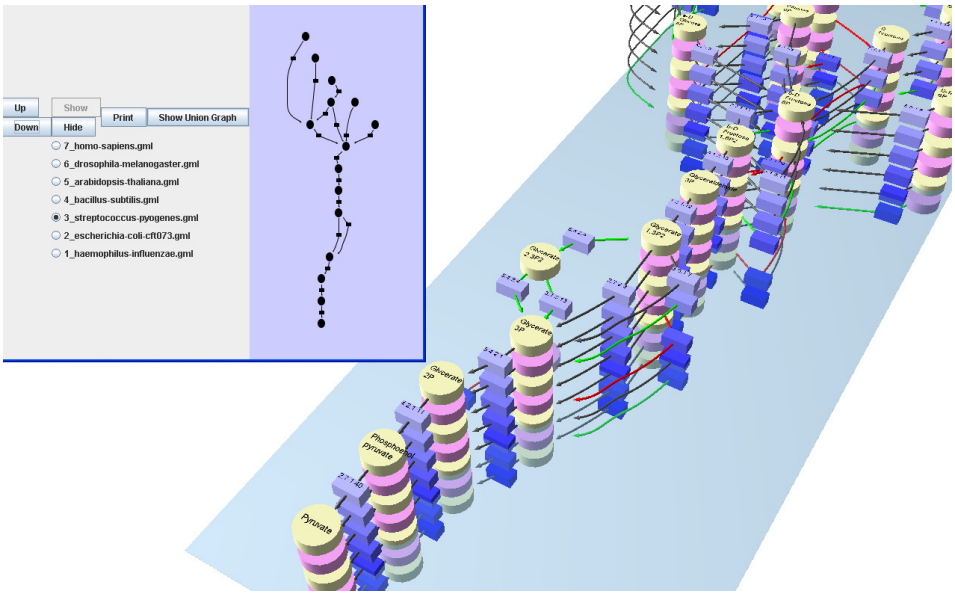


Figure 20.16 Visual comparison of metabolic pathways in $2\frac{1}{2}$ dimensions.

techniques to deduct biologically meaningful interpretations from the large amount of experimental data. The visualization of these data-rich networks provides new challenges for algorithms such as the consideration of complex graphical elements and of different node sizes. There is an increasing amount of approaches which look into this area, early approaches were, for example, [BHK+05, DRS04, JKS06, TSS+05], and a comparison is given in [KAO+09]. Also, the integration of different networks is increasingly important. Elements of one biological network often belong to several networks. For example, a protein of a protein-protein interaction network may be an enzyme of a metabolic network, an element of a gene regulatory network, or a leaf of a phylogenetic tree. This complex structure of interwoven networks requires new visualization and exploration methods which are the topic of current research. Finally, the standardization of the visual representation of elements of biological networks has been the focus of recent developments. The Systems Biology Graphical Notation (SBGN) [LHM+09] provides a set of standards for graphically representing biological information. It can be considered as the biology equivalent of the circuit diagram in electronics. The standard also contains layout requirements for SBGN maps.

A detailed presentation of the above-mentioned and newly emerging topics would easily fill not only another chapter, but a book. Biological network visualization is growing at an extremely fast pace. However, our sole intention in this chapter was to raise awareness of the relevance of graph drawing for the area of biological networks and provide an introduction to this topic. The interested reader is referred to journals such as *Bioinformatics* and *BMC Bioinformatics* as well as newly founded conferences such as *VIZBI* (since 2010) or *IEEE BioVis* (since 2011) for ongoing developments.

References

[ABH94] R. D. Appel, A. Bairoch, and D. F. Hochstrasser. A new generation of information retrieval tools for biologists: The example of the ExPASy WWW server. *Trends Biochemical Sciences*, 19:258–260, 1994.

[Bas99] W. Basalaj. Incremental multidimensional scaling method for database visualization. In R. F. Erbacher, P. C. Chen, and C. M. Wittenbrink, editors, *Visual Data Exploration and Analysis VI (Proc. SPIE)*, volume 3643 of *Proceedings of SPIE*, pages 149–158, 1999.

[Bax03] A. D. Baxevanis. The molecular biology database collection: 2003 update. *Nucleic Acids Research*, 31(1):1–12, 2003.

[BBS05] C. Bachmaier, U. Brandes, and B. Schlieper. Drawings of phylogenetic trees (extended abstract). In X. Deng and D. Du, editors, *Algorithms and Computation, Proc. ISAAC 2005*, volume 3827 of *LNCS*, pages 1110–1121. Springer, 2005.

[BDH03] G. D. Bader, D. Betel D, and C. W. Hogue. BIND: the biomolecular interaction network database. *Nucleic Acids Research*, 31(1):248–250, 2003.

[BDS04a] U. Brandes, T. Dwyer, and F. Schreiber. Visual triangulation of network-based phylogenetic trees. In O. Deussen, C. Hansen, D. Keim, and D. Saupe, editors, *Data Visualization (Proc. VisSym'04)*, pages 75–84. Eurographics Association, 2004.

[BDS04b] U. Brandes, T. Dwyer, and F. Schreiber. Visual understanding of metabolic pathways across organisms using layout in two and a half dimensions. *Journal of Integrative Bioinformatics*, 1:2 (EPub), 2004.

[BE99] W. Basalaj and K. Eilbeck. Straight-line drawings of protein interactions (system demonstration). In J. Kratochvíl, editor, *Graph Drawing (Proc. GD '99)*, volume 1731 of *Lecture Notes Comput. Sci.*, pages 259–266. Springer-Verlag, 1999.

[BFP+04] F. J. Brandenburg, M. Forster, A. Pick, M. Raitner, and F. Schreiber. *Graph Drawing Software*, chapter BioPath – Exploration and Visualization of Biochemical Pathways, pages 215–236. Springer Mathematics and Visualization Series, 2004.

[BHK+05] L. Borisjuk, M.-R. Hajirezaei, C. Klukas, H. Rolletschek, and F. Schreiber. Integrating data from biological experiments into metabolic networks with the DBE information system. *In Silico Biology*, 5(2):93–102, 2005.

[BJDG+03] Z. Bar-Joseph, E. D. Demaine, D. K. Gifford, A. M. Hamel, T. S. Jaakkola, and N. Srebro. K-ary clustering with optimal leaf ordering for gene expression data. *Bioinformatics*, 19(9):1070–1078, 2003.

[BM02] V. Batagelj and A. Mrvar. Pajek – analysis and visualization of large networks. In P. Mutzel, M. Jünger, and S. Leipert, editors, *Graph Drawing (Proc. GD '01)*, volume 2265 of *Lecture Notes Comput. Sci.*, pages 477–478, 2002.

[BR01] M. Y. Becker and I. Rojas. A graph layout algorithm for drawing metabolic pathways. *Bioinformatics*, 17(5):461–467, 2001.

[Cam96] N. A. Campbell. *Biology*. The Benjamin-Cummings Publishing Company, 1996.

[Car04a] S. F. Carrizo. Phylogenetic trees: An information visualization perspective. In Y.-P. Phoebe Chen, editor, *Bioinformatics (Proc. APBC 2004)*, volume 29 of *Conf. Res. Pract. Inform. Techn.*, pages 315–320, 2004.

[Car04b] S. F. Carrizo. A survey of phylogenetic researchers: Results. `http://www.cs.usyd.edu.au/~scarrizo/Carrizo_PhylogeneticsSurveyResults.doc`, January 2004.

[CG10] G. R. Cochrane and M. Y. Galperin. The 2010 Nucleic Acids Research database issue and online database collection: a community of data resources. *Nucleic Acids Research*, 38:D1–D4, 2010.

[DBD+02] E. Demir, O. Babur, U. Dogrusöz, A. Gürsoy, G. Nisanci, R. Çetin Atalay, and M. Ozturk. PATIKA: an integrated visual environment for collaborative construction and analysis of cellular pathways. *Bioinformatics*, 18(7):996–1003, 2002.

[DETT99] G. Di Battista, P. Eades, R. Tamassia, and I. G. Tollis. *Graph Drawing*. Prentice Hall, Upper Saddle River, NJ, 1999.

[DRS04] T. Dwyer, H. Rolletschek, and F. Schreiber. Representing experimental biological data in metabolic networks. In Y. P. Chen, editor, *Bioinformatics (Proc. APBC'04)*, volume 29 of *Conf. Res. Pract. Inform. Techn.*, pages 13–20, 2004.

[DS04] T. Dwyer and F. Schreiber. Optimal leaf ordering for two and a half dimensional phylogenetic tree visualization. In N. Churcher and C. Churcher, editors, *Information Visualisation (Proc. invis.au 2004)*, volume 35 of *Conf. Res. Pract. Inform. Techn.*, pages 109–115, 2004.

[Ead84] P. Eades. A heuristic for graph drawing. *Congr. Numer.*, 42:149–160, 1984.

[Ead92] P. D. Eades. Drawing free trees. *Bulletin of the Institute for Combinatorics and its Applications*, 5:10–36, 1992.

[EGK+01] J. Ellson, E. R. Gansner, E. Koutsofios, S. C. North, and G. Woodhull. Graphviz – open source graph drawing tools. In P. Mutzel, M. Jünger, and S. Leipert, editors, *Graph Drawing (Proc. GD'01)*, volume 2265 of *Lecture Notes Comput. Sci.*, pages 483–484, 2001.

[EH00] P. Eades and M. L. Huang. Navigating clustered graphs using force-directed methods. *Journal of Graph Algorithms Applications*, 4(3):157–181, 2000.

[EHW00] L. B. Ellis, C. D. Hershberger, and L. P. Wackett. The university of minnesota biocatalysis/biodegradation database: Microorganisms, genomics and prediction. *Nucleic Acids Research*, 28(1):377–379, 2000.

[Fel73] J. Felsenstein. Maximum likelihood and minimum-steps methods for estimating evolutionary trees from data on discrete characters. *Systematic Zoology*, 22:240–249, 1973.

[Fel95] J. Felsenstein. The newick tree format. `http://evolution.gs.washington.edu/phylip/newicktree.html`, 1995.

[Fit71] W. M. Fitch. Toward defining the course of evolution: Minimum change for a specified tree topology. *Systematic Zoology*, 20:406–416, 1971.

[FLM95] A. Frick, A. Ludwig, and H. Mehldau. A fast adaptive layout algorithm for undirected graphs. In R. Tamassia and I. G. Tollis, editors, *Graph Drawing (Proc. GD '94)*, volume 894 of *Lecture Notes Comput. Sci.*, pages 388–403. Springer-Verlag, 1995.

[For04] M. Forster. A fast and simple heuristic for constrained two-level crossing reduction. In *Graph Drawing (Proc. GD'04)*, volume 3383 of *Lecture Notes Comput. Sci.*, pages 206–216, 2004.

[FR91] T. Fruchterman and E. Reingold. Graph drawing by force-directed placement. *Softw. – Pract. Exp.*, 21(11):1129–1164, 1991.

[FS03] C. Friedrich and F. Schreiber. Visualisation and navigation methods for typed protein-protein interaction networks. *Applied Bioinformatics*, 2(S3):19–24, 2003.

[FS04] C. Friedrich and F. Schreiber. Flexible layering in hierarchical drawings with nodes of arbitrary size. In V. Estivill-Castro, editor, *Computer Science (Proc. ACSC 2004)*, volume 26 of *Conf. Res. Pract. Inform. Techn.*, pages 369–376, 2004.

[GBWK+08] E. Grafahrend-Belau, S. Weise, D. Koschützki, U. Scholz, B. H. Junker, and F. Schreiber. MetaCrop – a detailed database of crop plant metabolism. *Nucleic Acids Research*, 36:D954–D958, 2008.

[GD06] B. Genc and U. Dogrusöz. A layout algorithm for signaling pathways. *Information Sciences*, 176:135–149, 2006.

[GHM+02] A. Goesmann, M. Haubrock, F. Meyer, J. Kalinowski, and R. Giegerich. PathFinder: reconstruction and dynamic visualization of metabolic pathways. *Bioinformatics*, 18(1):124–129, 2002.

[HJP02] K. Han, B.-H. Ju, and J. H. Park. InterViewer: Dynamic visualization of protein-protein interactions. In M. T. Goodrich and S. G. Kobourov, editors, *Graph Drawing (Proc. GD '02)*, volume 2528 of *Lecture Notes Comput. Sci.*, pages 364–365. Springer-Verlag, 2002.

[HMWD04] Z. Hu, J. Mellor, J. Wu, and C. DeLisi. VisANT: an online visualization and analysis tool for biological interaction data. *BMC Bioinformatics*, 5(1):17 (EPub), 2004.

[HP82] M. D. Hendy and D. Penny. Branch and bound algorithms to determine minimal evolutionary trees. *Mathematical Bioscience*, 60:133–142, 1982.

[HS03] M. Heymans and A. K. Singh. Deriving phylogenetic trees from the similarity analysis of metabolic pathways. *Bioinformatics*, 19(Suppl. 1):138–146, 2003.

[HT98] R. Hofestädt and S. Thelen. Qualitative modeling of biochemical networks. *In Silico Biology*, 1(1):39–53, 1998.

[Int92] International Union of Biochemistry and Moleculare Biology, Nomenclature Commitee. *Enzyme Nomenclature*. Academic Press, 1992.

[JKS06] B. H. Junker, C. Klukas, and F. Schreiber. VANTED: A system for advanced data analysis and visualization in the context of biological networks. *BMC Bioinformatics*, 7:109, 2006.

[KAO+09] N. Kono, K. Arakawa, R. Ogawa, N. Kido, K. Oshita, K. Ikegami, S. Tamaki, and M. Tomita. Pathway Projector: Web-based zoomable pathway browser using KEGG atlas and Google Maps API. *PLoS ONE*, 4(11):e7710, 2009.

[Kel00] E. F. Keller. *The Century of the Gene*. Harvard University Press, 2000.

[KGH+06] M. Kanehisa, S. Goto, M. Hattori, K. F. Aoki-Kinoshita, M. Itoh, S. Kawashima, T. Katayama, M. Araki, and M. Hirakawa. From genomics to chemical genomics: new developments in KEGG. *Nucleic Acids Research*, 34:D354–357, 2006.

[KGKN02] M. Kanehisa, S. Goto, S. Kawashima, and A. Nakaya. The KEGG databases at GenomeNet. *Nucleic Acids Research*, 30(1):42–46, 2002.

[KK89] T. Kamada and S. Kawai. An algorithm for drawing general undirected graphs. *Inform. Process. Lett.*, 31:7–15, 1989.

[KLSW94] P. D. Karp, J. Lowrance, T. Strat, and D. Wilkins. The Grasper-CL graph management system. *LISP and Symbolic Computation*, 7:245–282, 1994.

[KN95] E. Koutsofios and S. North. Drawing graphs with *dot*. Technical report, AT&T Bell Laboratories, Murray Hill, NJ., 1995. Available from `http://www.research.bell-labs.com/dist/drawdag`.

[KP94] P. D. Karp and S. M. Paley. Automated drawing of metabolic pathways. In H. Lim, C. Cantor, and R. Bobbins, editors, *Proc. of the 3rd International Conference on Bioinformatics and Genome Research*, pages 225–238, 1994.

[KPR02] P. D. Karp, S. M. Paley, and P. Romero. The pathway tools software. *Bioinformatics*, 18(Suppl. 1):S225–S232, 2002.

[KRS+00] P. D. Karp, M. Riley, M. Saier, I. Paulsen, S. M. Paley, and A. Pellegrini-Toole. The EcoCyc and MetaCyc database. *Nucleic Acids Research*, 28(1):56–59, 2000.

[KVC+03] M. Krull, N. Voss, C. Choi, S. Pistor, A. Potapov, and E. Wingender. TRANSPATH: an integrated database on signal transduction and a tool for array analysis. *Nucleic Acids Research*, 31:97–100, 2003.

[KWLF01] F. Kose, W. Weckwerth, T. Linke, and O. Fiehn. Visualizing plant metabolomic correlation networks using clique-metabolite matrices. *Bioinformatics*, 17(12):1198–1208, 2001.

[LHM+09] N. Le Novère, M. Hucka, H. Mi, S. Moodie, F. Schreiber, A. Sorokin, E. Demir, K. Wegner, M. Aladjem, S. M. Wimalaratne, F. T. Bergman, R. Gauges, P. Ghazal, K. Hideya, L. Li, Y. Matsuoka, A. Villéger, S. E. Boyd, L. Calzone, M. Courtot, U. Dogrusoz, T. Freeman, A. Funahashi, S. Ghosh, A. Jouraku, S. Kim, F. Kolpakov, A. Luna, S. Sahle, E. Schmidt, S. Watterson, G. Wu, I. Goryanin, D. B. Kell, C. Sander, H. Sauro, J. L. Snoep, K. Kohn, and H. Kitano. The Systems Biology Graphical Notation. *Nature Biotechnology*, 27:735–741, 2009.

[LNC93] A. L. Lehninger, D. L. Nelson, and M. M. Cox. *Principles of Biochemistry*. Worth Publisher, 1993.

[MBF+00] P. Mendes, D. L. Bulmore, A. D. Farmer, P. A. Steadman, M. E. Waugh, and S. T. Wlodek. PathDB: a second generation metabolic database. In J.-H. S. Hofmeyr, J. M. Rohwer, and J. L. Snoep, editors, *Proc. of the 9th International BioThermoKinetics Meeting*, pages 207–212. Stellenbosch University Press, 2000.

[MdRW03] E. Minch, M. de Rinaldis, and S. Weiss. pathSCOUT: exploration and analysis of biochemical pathways. *Bioinformatics*, 19(3):431–432, 2003.

[Men00] P. Mendes. Advanced visualization of metabolic pathways in PathDB. In *Proc. of the 8th Conference on Plant and Animal Genome*, 2000.

[MGB73] G. W. Moore, M. Goodman, and J. Barnabas. A method for constructing maximum parsimony ancestral amino acid sequences on a given network. *Journal of Theoretical Biology*, 38(3):459–483, 1973.

[Mic93] G. Michal. *Biochemical Pathways (Poster)*. Boehringer Mannheim, 1993.

[Mic98] G. Michal. On representation of metabolic pathways. *BioSystems*, 47:1–7, 1998.

[Mic99] G. Michal. *Biochemical Pathways*. Spektrum Akademischer Verlag, 1999.

[MS57] C. D. Michener and R. R. Sokal. A quantitative approach to a problem in classification. *Evolution*, 11:130–162, 1957.

[MZ03] H. Ma and A.-P. Zeng. Reconstruction of metabolic networks from genome data and analysis of their global structure for various organisims. *Bioinformatics*, 19(2):270–277, 2003.

[MZ04] H. W. Ma and A. P. Zeng. Phylogenetic comparison of metabolic capacities of organisms at genome level. *Molecular Phylogenetics and Evolution*, 31(1):204–213, 2004.

[NEDM03] A. Nikitin, S. Egorov, N. Daraselia, and I. Mazo. Pathway studio – the analysis and navigation of molecular networks. *Bioinformatics*, 19(16):2155–2157, 2003.

[Nic97] D. E. Nicholson. *Metabolic Pathways Map (Poster)*. Sigma Chemical Co., St. Louis, 1997.

[NIS90] F. C. Neidhardt, J. L. Ingraham, and M. Schaechter. *Physiology of the Bacterial Cell: A Molecular Approach*. Sinauer Associates, 1990.

[OLP+00] R. A. Overbeek, N. Larsen, G. D. Pusch, M. D'Souza, E. Selkov Jr., N. Kyrpides, M. Fonstein, N. Maltsev, and E. Selkov. WIT: integrated system for high-throughput genome sequence analysis and metabolic reconstruction. *Nucleic Acids Research*, 28(1):123–125, 2000.

[RML93] V. N. Reddy, M. L. Mavrovouniotis, and M. N. Liebman. Petri net representations of metabolic pathways. In L. Hunter, D. Searls, and J. Shavlik, editors, *Intelligent Systems for Molecular Biology (Proc. ISMB '93)*, pages 328–336, 1993.

[Roj03] I. Rojdestvenski. Metabolic pathways in three dimensions. *Bioinformatics*, 19(18):2436–2441, 2003.

[RT81] E. Reingold and J. Tilford. Tidier drawing of trees. *IEEE Trans. Softw. Eng.*, SE-7(2):223–228, 1981.

[San75] D. D. Sankoff. Minimal mutation trees of sequences. *SIAM J. Appl. Math.*, 28:35–42, 1975.

[San95] G. Sander. Graph layout through the VCG tool. In R. Tamassia and I. G. Tollis, editors, *Graph Drawing (Proc. GD '94)*, volume 894 of *Lecture Notes Comput. Sci.*, pages 194–205. Springer-Verlag, 1995.

[Sch02] F. Schreiber. High quality visualization of biochemical pathways in BioPath. *In Silico Biology*, 2(2):59–73, 2002.

[Sch03] F. Schreiber. Visual comparison of metabolic pathways. *Journal of Visual Languages and Computing*, 14(4):327–340, 2003.

[SDMW09] F. Schreiber, T. Dwyer, K. Marriott, and M. Wybrow. A generic algorithm for layout of biological networks. *BMC Bioinformatics*, 10:375, 2009.

[SL99] G. Shavit and C. Linhart. Algorithms for molecular biology, chapter 9. `http://http://www.math.tau.ac.il/~rshamir/algmb/98/scribe/pdf/lec09.pdf`, January 1999.

[SMKS99] W. Salamonsen, K. Y. Mok, P. Kolatkar, and S. Subbiah. BioJAKE: A tool for the creation, visualization and manipulation of metabolic pathways. In *Proc. of the 4th Pacific Symposium on Biocomputing*, pages 392–400, 1999.

[SMO+03] P. Shannon, A. Markiel, O. Ozier, N. S. Baliga, J. T. Wang, D. Ramage, N. Amin, B. Schwikowski, and T. Ideker. Cytoscape: a software environment for integrated models of biomolecular interaction networks. *Genome Research*, 13(11):2498–2504, 2003.

[SN87] N. Saitou and M. Nei. The neighbor-joining method: A new method for reconstructing phylogenetic trees. *Molecular Biology and Evolution*, 4(4):406–425, 1987.

[SSE+02] M. Sirava, T. Schäfer, M. Eiglsperger, M. Kaufmann, O. Kohlbacher, E. Bornberg-Bauer, and H.-P. Lenhof. BioMiner – modeling, analyzing, and visualizing biochemical pathways and networks. *Bioinformatics*, 18(S2):S219–S230, 2002.

[STT81] K. Sugiyama, S. Tagawa, and M. Toda. Methods for visual understanding of hierarchical systems. *IEEE Trans. Syst. Man Cybern.*, SMC-11(2):109–125, 1981.

[TSS+05] T. Tokimatsu, N. Sakurai, H. Suzuki, H. Ohta, K. Nishitani, T. Koyama, T. Umezawa, N. Misawa, K. Saito, and D. Shibatanenell. KaPPA-View. a web-based analysis tool for integration of transcript and metabolite data on plant metabolic pathway maps. *Plant Physiology*, 138:1289–1300, 2005.

[Wal90] J. Q. Walker II. A node-positioning algorithm for general trees. *Softw. – Pract. Exp.*, 20(7):685–705, 1990.

[Wal02] C. Walshaw. A multilevel algorithm for force-directed graph drawing. In J. Marks, editor, *Graph Drawing (Proc. GD '01)*, volume 1984 of *Lecture Notes Comput. Sci.*, pages 171–182. Springer-Verlag, 2002.

[WB01] U. Wittig and A. De Beuckelaer. Analysis and comparison of metabolic pathway databases. *Briefings in Bioinformatics*, 2(2):126–142, 2001.

[WEK01] R. Wiese, M. Eiglsperger, and M. Kaufmann. yFiles: Visualization and automatic layout of graphs. In P. Mutzel, M. Jünger, and S. Leipert, editors, *Graph Drawing (Proc. GD'01)*, volume 2265 of *Lecture Notes Comput. Sci.*, pages 453–454, 2001.

[WS79] C. Wetherell and A. Shannon. Tidy drawing of trees. *IEEE Trans. Softw. Eng.*, SE-5(5):514–520, 1979.

[XFS+01] I. Xenarios, E. Fernandez, L. Salwinski, X.J. Duan, M. J. Thompson, E. M. Marcotte, and D. Eisenberg. DIP: The database of interaction proteins: 2001 update. *Nucleic Acids Research*, 29(1):239–241, 2001.