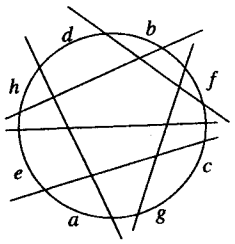
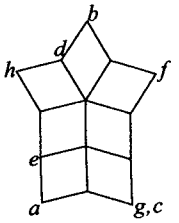
$\{a,b,d,e,h\} \mid \{c,f,g\}$
$\{a,c,d,e,g,h\} \mid \{b,f\}$
$\{a,c,e,g\} \mid \{b,d,f,h\}$
$\{a,c,g\} \mid \{b,d,e,f,h\}$
$\{a,c,e,f,g\} \mid \{b,d,h\}$
$\{a,e,h\} \mid \{b,c,d,f,g\}$

(a) Circular splits



(b) Circular ordering

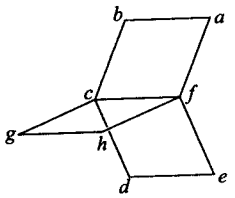


(c) Planar network

Figure 5.9 (a) A set of six circular splits $\mathcal{S}$ on $\mathcal{X} = \{a, b, \ldots, h\}$. (b) An arrangement of the taxa around a circle such that every split $S = A \mid B \in \mathcal{S}$ can be realized by a straight line through the circle that separates the two split parts A and B. A circular ordering is given by (a, g, c, f, b, d, h, e). (c) An outer-labeled planar split network representing $\mathcal{S}$.

$\{a,b\} \mid \{c,d,e,f,g,h\}$
$\{a,b,c,d,e,f\} \mid \{g,h\}$
$\{a,b,c,f,g,h\} \mid \{d,e\}$
$\{a,e,f,h\} \mid \{b,c,d,g\}$

(a) Non-circular splits



(b) Non-planar network

Figure 5.10 (a) A set of four non-circular splits $\mathcal{S}$ on $\mathcal{X} = \{a, b, \ldots, h\}$. (b) A non-planar split network representing $\mathcal{S}$.

5.7 Circular splits and planar split networks

One practical problem that arises when working with split networks is that the networks can be very complicated and thus difficult to visualize in a comprehensible way. Hence, a number of restricted classes of sets of splits have been introduced in an attempt to avoid overly complicated networks. The two most important are *circular* splits, which are the focus of this section, and *weakly compatible* splits, which we introduce in the next section.

Informally, a set of splits $\mathcal{S}$ on $\mathcal{X}$ is called *circular*, if the taxa in $\mathcal{X}$ can be placed around a circle in such a way that each split $S = \frac{A}{B}$ can be realized by a line through the circle that separates the plane into two half-planes, one containing all taxa in A and the other containing all taxa in B (see Figure 5.9). An example of a non-circular set of splits and the corresponding split network is shown in Figure 5.10. More formally [9]:

Definition 5.7.1 (Circular splits) *A set of splits $\mathcal{S}$ on $\mathcal{X}$ is called* circular, *if there exists a linear ordering $(x_1, \ldots, x_n)$ of the elements of $\mathcal{X}$ for $\mathcal{S}$ such that each split*

of ones in the matrix M. This problem, known as the *Optimal Consecutive Ones* problem, is NP-hard.

A useful heuristic for finding an optimal circular ordering for a set of non-circular splits $\mathcal{S}$ on $\mathcal{X}$ is to define a distance matrix D on $\mathcal{X}$ by setting the distance between any two taxa a and b equal to the sum of all weights of the splits that separate them. This distance matrix D is then provided as input to the neighbor-net algorithm, described in Section 10.4, which outputs an ordering of $\mathcal{X}$, that is guaranteed to be circular, whenever the given input set $\mathcal{S}$ is circular [32].

Circular split systems are of particular interest because they can always be represented graphically in an appealing way, namely by a split network that is drawn in the plane without any edges crossing and with all labeled nodes appearing around the outside of the network (see Figure 5.9c), conforming to the following:

Definition 5.7.4 (Outer-labeled planar) *Let G be a graph in which some of the nodes are labeled. We call G* outer-labeled planar, *if there exists a drawing of G in the plane such that no two edges intersect and all labeled nodes lie on the outside of the graph.*

With this definition we have [62]:

Theorem 5.7.5 (Circular implies outer-labeled planar) *A set of splits $\mathcal{S}$ on $\mathcal{X} = \{x_1, \ldots, x_n\}$ is circular if and only if it can be represented by a split network N that is outer-labeled planar.*

Proof If $\mathcal{S}$ is circular, then we can position the elements of $\mathcal{X} = \{x_1, \ldots, x_n\}$ around a circle in the plane in such a way that every split $S \in \mathcal{S}$ is induced by a chord of the circle. Applying de Bruijn's "dualization principle" [29] to the set of these chords produces a "zonotopal graph" that can be labeled so as to form an outer-labeled planar split network for $\mathcal{S}$ [20].

Now assume that we are given a split network N that is embedded in the plane such that all labels occur on the outer boundary of the network. A circular ordering is given by the order in which the taxa are encountered when traveling once around the outside of the embedded network. □

As shown in Section 10.4, the neighbor-net algorithm, which is a widely used method for computing a set of incompatible splits from a distance matrix, always produces a circular set of splits and thus can always be represented by an outer-labeled planar split network.

Exercise 5.7.6 (Splits from one or two trees) *Prove that the set of splits obtained from a single unrooted phylogenetic tree is always circular. Construct an example that shows that the set of splits taken from two unrooted phylogenetic trees is not necessarily circular.*

As we saw above, the number of nodes and edges of a split network N representing a general set of splits can be exponential in the number of splits. However, a set of circular splits can be represented by a split network that is of quadratic size only:

Lemma 5.7.7 (Circular splits have quadratic-size network) *If $\mathcal{S}$ is a set of m circular splits on $\mathcal{X}$, then the number of nodes and edges in an outer-labeled planar split network N representing $\mathcal{S}$ is at most quadratic in m.*

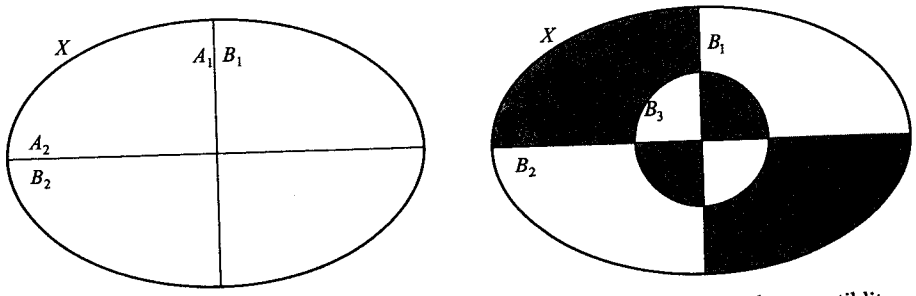
Proof Because $\mathcal{S}$ is circular, we can arrange all taxa on a circle in such a way that every split $S = \frac{A}{B}$ corresponds to some line l_S that passes through the circle and separates all taxa in A from all taxa in B. We assume that all such lines are in *general position*, that is, that no more than two lines intersect at any given point (see Figure 5.9b for an example). This can always be achieved by allowing ourselves to bend lines locally to avoid more than two lines meeting in a point. The collection of lines is then technically what is known as a *pseudo-line arrangement*. As indicated in the proof of Theorem 5.7.5, a split network N can be obtained by dualization of the complete (pseudo) line arrangement. In this process, each line l gives rise to $t + 1$ edges, where t is the number of intersection points that l shares with other lines in the arrangement. As t is at most $m - 1$, it follows that N contains only a quadratic number of edges and, thus also, of nodes. □

Note that the canonical construction is less suited for circular split systems. Indeed, the canonical split network (Buneman graph) N associated with a set of splits $\mathcal{S}$ on $\mathcal{X}$ is not necessary planar when $\mathcal{S}$ is circular. This is a main reason why we make the distinction between the concept of a split network and a canonical split network. In Section 7.2, we present the circular network algorithm, which is guaranteed to produce an outer-labeled planar split network N when given a set of circular splits $\mathcal{S}$ on $\mathcal{X}$.

5.8 Weak compatibility

As discussed above, any *two* splits $S_1 = A_1 \mid B_1$ and $S_2 = A_2 \mid B_2$ on $\mathcal{X}$ are called compatible if one of the four possible intersections, $A_1 \cap A_2$, $A_1 \cap B_2$, $B_1 \cap A_2$ or $B_1 \cap B_2$ is empty. One possible mathematical generalization of compatibility is known as *weak compatibility* [9] and is defined as a relation between any *three* splits, requiring that certain intersections of the split parts are empty, as we discuss below.

Weak compatibility is an important concept for two reasons. First, the set of splits produced by the split decomposition method, which we describe in Section 5.9, is always weakly compatible. The same is also true for the (less widely known) parsimony splits method, which is presented in Section 9.3. Second, in practice the



(a) Venn diagram for compatibility (b) Venn diagram for weak compatiblity

Figure 5.12 (a) Two splits $S_1 = \frac{A_1}{B_1}$ and $S_2 = \frac{A_2}{B_2}$ on $\mathcal{X}$ are compatible, if and only if one of the four regions in this Venn diagram is empty. (b) Three splits $S_1 = \frac{A_1}{B_1}$, $S_2 = \frac{A_2}{B_2}$ and $S_3 = \frac{A_3}{B_3}$ on $\mathcal{X}$ are weakly compatible if and only if at least one of the gray regions and one of the white regions in this Venn diagram are empty.

resulting split networks are often quite close to being planar, as they usually have only a few edges crossing over each other and do not contain any "high-dimensional cubes", which may occur for completely unrestricted sets of splits.

Definition 5.8.1 (Weak compatibility) *A set of splits $\mathcal{S}$ on $\mathcal{X}$ is called* weakly compatible, *if any* three *distinct splits in $\mathcal{S}$ are weakly compatible. Three such splits $S_1 = \frac{A_1}{B_1}$, $S_2 = \frac{A_2}{B_2}$, and $S_3 = \frac{A_3}{B_3}$ are called* weakly compatible, *if*

(i) *at least one of the following four intersections is empty:*

$$A_1 \cap A_2 \cap A_3, \quad A_1 \cap B_2 \cap B_3, \quad B_1 \cap A_2 \cap B_3 \quad \text{and} \quad B_1 \cap B_2 \cap A_3, \tag{5.18}$$

and, symmetrically,

(ii) *at least one of the following four intersections is empty:*

$$B_1 \cap B_2 \cap B_3, \quad B_1 \cap A_2 \cap A_3, \quad A_1 \cap B_2 \cap A_3 \quad \text{and} \quad A_1 \cap A_2 \cap B_3. \tag{5.19}$$

In Figure 5.12 we show two Venn diagrams illustrating the definitions of compatibility and weak compatibility. If three splits are not weakly compatible, then it can happen that every possible intersection of any three split parts is non-empty and the split network required to represent the three splits consists of the eight edges of a cube.

Exercise 5.8.2 (Number of weakly compatible splits) *What is the maximum number of weakly compatible splits that is possible on a taxon set of size* 4, 5 *or* 6*?*

Consider two unrooted phylogenetic trees T_1 and T_2 on $\mathcal{X}$ and let $\mathcal{S}$ be the set of splits obtained from both trees. Is $\mathcal{S}$ weakly compatible? This question can easily be answered using Figure 5.12: Given any three splits S_1, S_2 and S_3, at least one

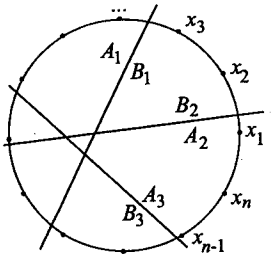


Figure 5.13 For a circular set of splits $\mathcal{S}$ with circular ordering $(x_1, \ldots, x_n)$, we represent three distinct splits $S_1 = \frac{A_1}{B_1}$, $S_2 = \frac{A_2}{B_2}$ and $S_3 = \frac{A_3}{B_3}$ by lines in a configuration in which every pair of lines intersects.

pair of splits, S_1 and S_2, say, must come from the same tree and is thus compatible. Hence, one of the four regions shown in (a) is empty. Comparison of (a) and (b) shows that every such region contains both a gray and a white region of the Venn diagram shown in (b). Therefore, one gray and one white region shown in (b) must be empty and so the three splits are weakly compatible. Moreover, we have that any set of circular splits is weakly compatible:

Lemma 5.8.3 (Circular implies weakly compatible) *Let $\mathcal{S}$ be a set of splits on $\mathcal{X}$. If $\mathcal{S}$ is circular, then $\mathcal{S}$ is weakly compatible.*

Proof Let $\mathcal{S}$ be a circular set of splits on $\mathcal{X}$ and assume that we have placed the taxa around a circle in such a way that each split in $\mathcal{S}$ is represented by a line through the circle. Consider three distinct splits $S_1 = \frac{A_1}{B_1}$, $S_2 = \frac{A_2}{B_2}$ and $S_3 = \frac{A_3}{B_3}$. If all three lines representing these splits intersect each other, as show in Figure 5.13, then the intersections $A_1 \cap A_2 \cap A_3$ and $B_1 \cap B_2 \cap B_3$ are both empty. This implies that the three splits are weakly compatible, as the one intersection is colored gray and the other intersection is colored white in Figure 5.12(b). Otherwise, assume that (at least) two lines do not intersect, then the two corresponding splits are compatible and it follows that all three splits must be weakly compatible, using the same argument as applied above in the analysis of the situation of two trees. □

5.9 The split decomposition

The first practical method proposed for computing a set of incompatible splits from evolutionary data was the split decomposition method, introduced in [9]. As we will see, the split decomposition method takes as input a distance matrix and produces as output a set of weighted splits that is weakly compatible. To motivate this approach, we first discuss the related *Buneman tree* method as proposed by Bandelt and Dress [9].

7

Phylogenetic networks from splits

In Chapter 5, we introduced splits and split networks and presented the Buneman graph as the canonical split network associated with a given set of splits. Splits can be computed from many different types of data, as we discuss in some of the following chapters. The focus of this chapter is on the problem of computing a split network N for a given set of splits $\mathcal{S}$ on $\mathcal{X}$. We present two different approaches. The first is the *convex hull algorithm* that computes the Buneman graph and can be applied to any set of splits, using an exponential number of nodes and edges in the worst case [11]. It is also used in Section 9.4 to compute median networks. The second is the *circular network algorithm*, which can be applied to any set of circular splits and produces an *outer-labeled planar network* with only a quadratic number of nodes and edges [62].

Both algorithms proceed in two steps. In the first step, all trivial splits in $\mathcal{S} = \{S_1, \ldots, S_m\}$ are processed to obtain a *star network* consisting of a central node and one leaf per taxon. Then, in the second step, the remaining splits are inserted one by one so as to obtain the final network.

To avoid some technical complexities, in the following we will assume that $\mathcal{S}$ contains all n trivial splits on $\mathcal{X} = \{x_1, \ldots, x_n\}$. We will use $\mathcal{S}^O = \{S_1^O, \ldots, S_n^O\}$ and $\mathcal{S}^I = \{S_1^I, \ldots, S_r^I\}$ to denote the set of trivial ("outer") and non-trivial ("inner") splits in $\mathcal{S}$, with $n + r = m$.

7.1 The convex hull algorithm

In this section we present the *convex hull algorithm* for constructing a split network. First we introduce the concept of the convex hull of a set of taxa in a split network:

Definition 7.1.1 (Convex hull) *Let $\mathcal{S}$ be a set of splits on $\mathcal{X}$ and let $N = (V, E, \sigma, \lambda)$ be a split network that represents $\mathcal{S}$. Consider a set of taxa $C \subset \mathcal{X}$. The* convex hull *$H(C)$ of C in N is defined as the set of all nodes in N that lie on a shortest path*

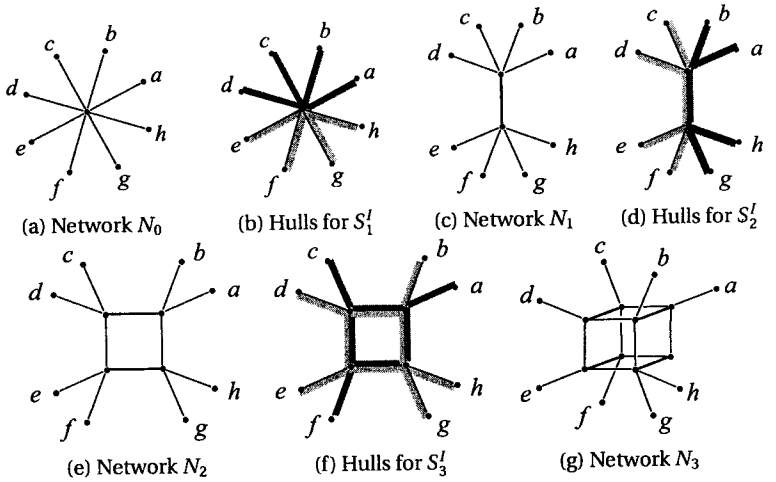


Figure 7.1 Construction of the canonical split network for three non-trivial splits S_1^I, S_2^I and S_3^I on $\mathcal{X} = \{a, \ldots, h\}$. (a) The split network N_0 representing all trivial splits on $\mathcal{X}$. (b) The two convex hulls $H(A_1)$ and $H(B_1)$ for the split $S_1^I = \frac{A_1}{B_1} = \frac{\{a,b,c,d\}}{\{e,f,g,h\}}$ (highlighted in dark gray and light gray, respectively). The intersection $H(A_1) \cap H(B_1)$ consists of the central node. (c) The resulting network N_1 obtained using the convex hull algorithm. (d) The two convex hulls for $S_2^I = \frac{A_2}{B_2} = \frac{\{a,b,g,h\}}{\{c,d,e,f\}}$. (e) The resulting network N_2. (f) The two convex hulls for $S_3^I = \frac{A_3}{B_3} = \frac{\{a,c,f\}}{\{b,d,e,g,h\}}$. (g) The resulting network N_3.

between any two nodes v and w that are contained in $\lambda(C)$, the set of all nodes labeled by taxa in C.

Examples can be found in Figure 7.1.

Exercise 7.1.2 (Alternative definition) *Show that the above definition can be rephrased as follows: the convex hull $H(C)$ consists of all nodes in N that can be reached from some node $v \in \lambda(C)$ using only such edges that are labeled by splits contained in the set $\{\frac{A}{B} \in \mathcal{S} \mid A \cap C \neq \emptyset, B \cap C \neq \emptyset\}$.*

Our aim is to construct a split network $N = (V, E, \sigma, \lambda)$ that represents the set of splits $\mathcal{S} = \mathcal{S}^O \cup \mathcal{S}^I$. To accomplish this, we need to define the set of nodes V, set of edges E, labeling of nodes by taxa λ and the labeling of edges by splits σ. First, we process all the trivial splits so as to obtain a *star network* with one central node and one edge for each trivial split, as shown in Figure 7.1a:

Algorithm 7.1.3 (Star network for trivial splits) *Let $\mathcal{X} = \{x_1, \ldots, x_n\}$ be a set of taxa and let $\mathcal{S}^O = \{S_1^O, \ldots, S_n^O\}$ be the set of all trivial splits on $\mathcal{X}$, with $S_i^O = \frac{\{x_i\}}{\mathcal{X}-\{x_i\}}$ for $i = 1, \ldots, n$. We obtain a split network N_0 representing $\mathcal{S}^O$ as follows: Create a new node v_0. For $i = 1, \ldots, n$ do:*

(i) *Create a new node* v_i *and new edge* $e_i = \{v_0, v_i\}$.
(ii) *Set* $\lambda(x_i) = v_i$ *and* $\sigma(e_i) = S_i^O$.

After constructing the initial star network, we add the non-trivial splits to the network one by one, as follows. For each non-trivial split, compute the convex hull of each of the two split parts, determine and duplicate the intersection of the two hulls, one copy for each hull, and then insert new edges to connect the two copies.

More formally, we process the set of non-trivial splits $\mathcal{S}^I$ by applying the following algorithm to each split in $\mathcal{S}^I = \{S_1^I, S_2^I, \ldots, S_r^I\}$ in turn (see Figure 7.1b–g):

Algorithm 7.1.4 (Convex hull algorithm) *Let* $\mathcal{S}^O$ *be the set of all trivial splits on* $\mathcal{X}$ *and let* $\mathcal{S}_t^I = \{S_1^I, S_2^I, \ldots, S_t^I\}$ *be a set of* $t \geq 0$ *non-trivial splits on* $\mathcal{X}$. *Assume that we have already computed a split network* N_t *for* $\mathcal{S}^O \cup \mathcal{S}_t^I$. *To obtain a split network for* $\mathcal{S}^O \cup \mathcal{S}_t^I \cup \{S_{t+1}^I\}$, *where* $S_{t+1}^I = \frac{A}{B}$ *is a new non-trivial split, modify* N_t *as follows:*

(i) *Compute the two convex hulls* $H(A)$ *and* $H(B)$ *in* N_t *and let* M *be the split graph induced in* N_t *by the set of nodes* $H(A) \cap H(B) \neq \emptyset$.
(ii) *Create a copy* M' *of* M *and for each node* v *and edge* e *in* M *let* v' *and* e' *denote their copies in* M', *respectively.*
(iii) *For every edge* f *that leads from some node* u *in* $H(B) - H(A) \neq \emptyset$ *to a node* v *in* M, *redirect the edge* f *so that it leads from* u *to* v'.
(iv) *Connect each pair of nodes* v *in* M *and* v' *in* M' *by a new edge* $f = (v, v')$ *and set* $\sigma(f) = S_{t+1}$.

The algorithm makes two assumptions that we should briefly discuss. Let N be a split network for a set of splits $\mathcal{S}$ on $\mathcal{X}$ and let $S = \frac{A}{B}$ be a split of $\mathcal{X}$ that is not contained in $\mathcal{S}$. Then $H(A) \cap H(B) \neq \emptyset$ must hold, as otherwise there would already exist a set of edges in N that separates $\lambda(A)$ from $\lambda(B)$. This is used in step (i) of the algorithm. Further, the assumption that $\mathcal{S}$ contains all trivial splits of $\mathcal{X}$ implies $H(A) - H(B) \neq \emptyset$ and $H(B) - H(A) \neq \emptyset$. This is used in step (iii) of the algorithm.

In the worst case, $H(A) \cap H(B)$ will consist of all non-leaf nodes of N_t and then N_{t+1} will have twice as many internal nodes as N_t. As this can happen for every split that is processed by the algorithm, the number of nodes and edges in the final split network can be exponential in the number of splits.

Note that the split network N constructed by this algorithm is uniquely defined and is the same as the one specified by the Buneman graph construction in Section 5.6. Although this is not easy to prove, we leave the proof as an exercise:

Exercise 7.1.5 (Convex hull algorithm computes Buneman graph) *Prove that the convex hull algorithm computes the Buneman graph.*

7.2 The circular network algorithm

A network N is called *planar*, if it can be drawn in the euclidean plane in such a way that no two edges intersect. If a concrete drawing of N is given, then we refer to this as a *plane network* and, for brevity, also denote it by N. A *topological embedding* of N is given by a mapping τ that assigns to each node $v \in V$ an ordering $\tau(v) = (v_1, \ldots, v_k)$ of the set of all its adjacent nodes $\{v_1, \ldots, v_k\}$.

As above, assume that we are given a set of splits $\mathcal{S} = \mathcal{S}^O \cup \mathcal{S}^I$, where $\mathcal{S}^O = \{S_1^O, \ldots, S_n^O\}$ is the set of all trivial splits on $\mathcal{X}$ and $\mathcal{S}^I = \{S_1^I, \ldots, S_r^I\}$ is a set of non-trivial splits. Additionally, we assume that $\mathcal{S}$ is *circular* with respect to the ordering $(x_1, \ldots, x_n)$, as defined in Section 5.7.

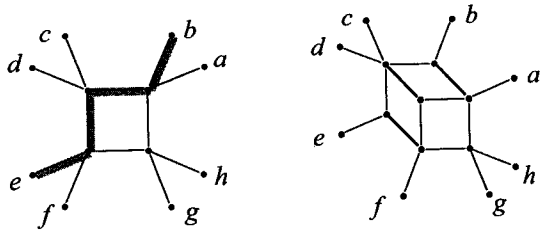
Our aim is to construct a planar split network N that represents $\mathcal{S}$. This can be done using the *circular network algorithm*, which operates in a similar fashion to the convex hull algorithm. Just as in that algorithm, the first step is to obtain a star network with one central node and one edge for each trivial split using Algorithm 7.1.3 (see Figure 7.1a).

The main difference between the circular network algorithm and the convex hull algorithm is in the choice of the subnetwork M to be duplicated when processing a non-trivial split: in the circular network algorithm, the subnetwork M will be a single path along the outside of the existing network.

Let $\mathcal{S}$ be a set of splits on $\mathcal{X}$ and suppose we are given an outer-labeled planar split network $= (V, E, \sigma, \lambda)$ for $\mathcal{S}$, in which the taxa appear in the circular order $(x_1, \ldots, x_n)$ as one circles around the network in mathematically positive orientation. For a new non-trivial split $S = \frac{\{x_p, \ldots, x_q\}}{\mathcal{X} - \{x_p, \ldots, x_q\}} \notin \mathcal{S}$, with $1 < p < q \leq n$, we define $M(x_p, x_q)$ to be the unique shortest path from $\lambda(x_p)$ to $\lambda(x_q)$ that runs around the outside of the embedded network N and is incident to every leaf edge that is connected to a node labeled by one of the taxa in $\{x_p, \ldots, x_q\}$.

The second step is to process the non-trivial splits one after the other (see Figure 7.2). To ensure that the resulting network has a minimal number of nodes and edges, and possesses an embedding in the plane in which all edges corresponding to the same split can be drawn as parallel lines of the same length, the algorithm must process the non-trivial splits in order of non-increasing cardinality of the split part that does not contain taxon x_1.

Algorithm 7.2.1 (Circular network algorithm) *Let $\mathcal{S}^O$ be the set of all trivial splits on $\mathcal{X}$ and let $\mathcal{S}_t^I$ be a set of $t \geq 0$ non-trivial splits that is circular on $(x_1, \ldots, x_n)$. Assume that we have already computed a split network N_t for $\mathcal{S}^O \cup \mathcal{S}_t^I$ using this algorithm. To obtain a split network for $\mathcal{S}^O \cup \mathcal{S}_t^I \cup \{S_{t+1}^I\}$, where $S_{t+1}^I = \frac{\{x_p, \ldots, x_q\}}{\mathcal{X} - \{x_p, \ldots, x_q\}}$ is a new non-trivial split (obeying the ordering requirement formulated above), modify N_t as follows:*



(a) Network N_2 (b) Network N_3

Figure 7.2 To add the split $\frac{A}{B} = \frac{\{a,f,g,h\}}{\{b,c,d,e\}}$ to the split network N_2 shown in (a), with $x_1 = a$, $x_2 = b$, etc., the circular network algorithm duplicates the internal edges of the path $M(b, e)$ leading from the b to e (highlighted in gray) to obtain the network N_3 shown in (b), in which the three edges representing $\frac{A}{B}$ are highlighted in bold.

(i) *Determine the path $M(x_p, x_q)$ and let $\dot{M}$ denote the path obtained by removing the first and last (leaf) edges from $M(x_p, x_q)$.*
(ii) *Create a copy $\dot{M}'$ of $\dot{M}$ and for each node v and edge e in $\dot{M}$ let v' and e' denote their copies, respectively.*
(iii) *Redirect every edge f that leads from some node $u = \lambda(x_i)$ to some node v in $\dot{M}$ so that it leads from u to v', for all $i = p, \ldots, q$.*
(iv) *Connect each pair of nodes v in $\dot{M}$ and v' in $\dot{M}'$ by a new edge $f = (v, v')$ and set $\sigma(f) = S_{t+1}$.*

The number of nodes and edges produced by this algorithm is at most quadratic in the number of splits m. To see this, note that the path $M(x_p, x_q)$ in the algorithm contains at most one edge per split and so processing of a single non-trivial split produces a linear number of edges.

Indeed, $M(x_p, x_q)$ cannot contain two edges corresponding to the same split $S' \in \mathcal{S}_t^I$ because this would imply that the split part of S' that does not contain x_1 is smaller than the split part of S_{t+1}^I that does not contain x_1, which is impossible due to the ordering in which the algorithm processes the splits. The requirement that the non-trivial splits must be processed in order of non-increasing size of the split parts not containing x_1 is not only important for minimizing the number of edges in the network. If we disregard this requirement, then the resulting split network may fail to have the property that edges corresponding to the same split can be drawn as parallel line segments of the same length:

Exercise 7.2.2 (Correct order is required) *Consider the split network N_3 shown in Figure 7.2. Using the circular network algorithm, attempt to modify N_3 so as to obtain a new split network N_4 that also represents the split $S_4^I = \frac{\{a,h\}}{\{b,c,d,e,f,g\}}$ as well. What goes wrong? Rerun the algorithm on all four splits in the correct order.*

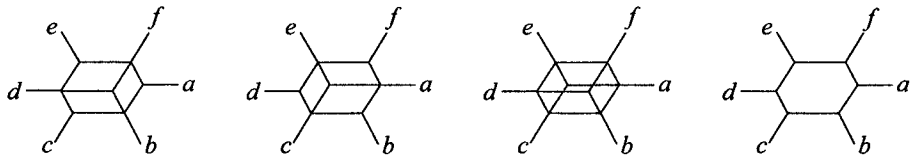


Figure 7.3 All four different split networks shown here represent the same set of splits. The split networks shown in (a) and (b) were computed using the circular network algorithm processing the splits and taxa in two different orders. The one shown in (c) was constructed using the convex hull algorithm. The split network shown in (d) can be obtained by deleting superfluous edges in any of the first three.

Unlike in the case of the convex hull algorithm, the split network constructed by this algorithm is not uniquely defined, as the resulting network depends on the order in which the splits are processed. In Figure 7.3, we show four different representations of the split set $\mathcal{S}$ consisting of all six trivial splits on $\mathcal{X} = \{a, \ldots, f\}$ and the three non-trivial splits $S_1 = \frac{\{a,b,c\}}{\{d,e,f\}}$, $S_2 = \frac{\{b,c,d\}}{\{e,f,a\}}$ and $S_3 = \frac{\{c,d,e\}}{\{f,a,b\}}$. The first two networks were produced by the circular network algorithm and the third by the convex hull algorithm. The fourth network can be obtained by the deletion of superfluous edges in any of the first three. This example also illustrates that the convex hull algorithm does not necessarily produce a planar network, even when given a circular set of splits.

Chapter notes

In many cases, direct application of the convex hull algorithm leads to an over-complicated network, as indicated in Figure 7.3. In practice, a useful heuristic is to first choose an order of the taxa such that a large subset of the given set of splits is circular. This subset of splits is then processed using the circular network algorithm to obtain an outer-labeled planar network. The remaining splits are then processed using the convex hull algorithm, which will add some non-planar parts to the network. This is the default approach used in SplitsTree [125].

In the following chapters we describe a number of different methods for computing a set of splits from sequences, distances, trees or quartets. All these methods are used in conjunction with the convex hull and circular network algorithms to obtain a split network from biological data.

9

Phylogenetic networks from sequences

In this chapter we describe a number of methods that require sequences, or more precisely, a multiple sequence alignment, as input. To this end, we first introduce the useful concept of a condensed alignment and briefly discuss the relationship between binary sequences and splits. We then discuss a number of methods that can be used to compute unrooted phylogenetic networks from sequences, including the median network, quasi-median network and median-joining methods. Finally, we look at two different ways of computing rooted networks from sequences, both aimed at explaining differences in terms of mutations and recombinations.

9.1 Condensed alignments

For computational reasons, it is sometimes useful to assume that a given multiple sequence alignment M on $\mathcal{X}$ is *condensed* in the sense that no two sequences are identical and no two columns display the same *pattern* of states, that is, induce the same partitioning of the taxa. Additionally, we assume that a condensed alignment does not contain any constant columns (in which all sequences have the same state). We shall sometimes refer to the sequences of a condensed alignment as *haplotypes*.

If the original input multiple sequence alignment M_0 is not condensed, then we will usually *condense* it in a preprocessing step as follows: first, each set of identical sequences is pooled into a single haplotype, then all constant columns are removed and, finally, every set of columns that have the same pattern is replaced by a single column i that is assigned a weight $\omega(i)$ that equals the number of columns in the represented set. Also, in some applications, one may want to consider the haplotypes as weighted by the number of original sequences that they represent. An example is shown in Figure 9.1.

9.2 Binary sequences and splits

Some of the algorithms discussed in this chapter assume that the input consists of a multiple alignment M of *binary* sequences on $\mathcal{X}$. However, in practice, the original

	1	2	3	4	5
a	A	A	G	T	T
b	C	A	C	A	T
c	C	A	C	A	A
d	T	A	T	C	A
e	A	A	G	T	T

(a) Alignment M_0

	1	2
a	A	A
b	B	A
c	B	B
d	C	B

(b) Condensed M

Figure 9.1 In the multiple alignment of DNA sequences M_0 shown in (a), the columns 1, 3 and 4 have the same pattern and column 2 is constant. The sequences for *a* and *e* are identical. (b) The resulting condensed alignment M contains four sequences of length two. The sequence for *a* represents the sequences for both *a* and *e* in the original alignment and has weight 2. The first column represents the three original columns 1, 3 and 4 and has weight $\omega(1) = 3$. The second column corresponds to the fifth original column and has weight $\omega(2) = 1$.

input data will usually be a multiple alignment of DNA sequences M_0 on $\mathcal{X}$. In this section, we assume that the given multiple sequence alignment does not contain any gaps. This assumption can be met by ignoring all columns that contain a gap.

There are a number of different ways that such data can be converted into binary sequences. One possibility is to extract all those columns in M_0 that contain precisely two different character states and then to replace the two character states by 0 and 1, perhaps in such a way that the first sequence is always given the state 0, say. A justification for discarding all columns that contain more than two characters states could be that such sites are "hypervariable" and so may fail to be homologous or may have been saturated by multiple substitutions [46].

Another possibility is to use all columns of M_0 and to replace any occurrence of either of the two nucleotides A and G (the two purines) by 0, and to replace any occurrence of the other two nucleotides C and T (the two pyrimidines) by 1. This approach is based on the observation that transitions (changes between A and G or between C and T) are more frequent than transversions (all other possible changes) and the latter are more informative.

If M is a multiple alignment of binary sequences on $\mathcal{X}$, then each non-constant column i of M corresponds to a split $S = \frac{A}{B}$ of $\mathcal{X}$. It is defined by setting A equal to the set of all taxa whose state in column i agrees with that of the first sequence and setting $B = \mathcal{X} - A$. If M is a condensed alignment, then the split S is unique and it is assigned a weight $\omega(S) = \omega(i)$ that equals the number of original columns that were condensed into the column i. We use $\mathcal{S}(M)$ to denote the set of all splits obtainable in this way.

The columns of a multiple sequence alignment M on $\mathcal{X}$ are also called *characters* or *positions*. We say that two characters or positions of a multiple alignment of binary sequences are *compatible* if and only if the corresponding splits are, and *incompatible* otherwise.

Exercise 9.2.1 (Incompatibility of binary characters) *Show that two characters i and j in a multiple alignment M of binary sequences are incompatible if and only if there exist four different sequences in the alignment that equal* `00`, `01`, `10` *and* `11`, *when restricted to the characters i and j.*

9.3 Parsimony splits

The split decomposition algorithm discussed in Section 5.9.4 takes as input a distance matrix D on $\mathcal{X}$ and produces as output a set of weakly compatible splits $\mathcal{S}$ on $\mathcal{X}$. A main feature of the algorithm is that it employs a distance-based "isolation index" that evaluates to a positive number for, at most, two of the three possible non-trivial splits on any quadruple Q of taxa in $\mathcal{X}$.

Let M be a multiple sequence alignment on $\mathcal{X}$. To obtain a similar algorithm that operates on a multiple sequence alignment rather than on a distance matrix, we must replace the computation of the isolation index (which requires a distance matrix) by an analogous calculation that is based directly on the alignment.

In the *parsimony splits* method [8], the basic idea is to use an index that selects, at most, two of the three most parsimonious bifurcating tree topologies for any given quadruple, as the name suggests. In more detail, for each quadruple of taxa $Q = \{w, x, y, z\}$ on $\mathcal{X}$ with corresponding aligned sequences $a = a_1 \dots a_L$, $b = b_1 \dots b_L$, $c = c_1 \dots c_L$ and $d = d_1 \dots d_L$ in M, we define

$$p_M\left(\frac{\{w, x\}}{\{y, z\}}\right) = |\{i : a_i = b_i \text{ and } c_i = d_i\}| \tag{9.1}$$

as the number of positions $i \in \{1, \dots, L\}$ in the alignment M at which the character states are the same for the sequences a and b and also for c and d, but not necessarily for all four sequences.

To set things up so for any quadruple of taxa $Q = \{w, x, y, z\}$, we select the quartet topology $\frac{\{w,x\}}{\{y,z\}}$ if and only if

$$p_M\left(\frac{\{w, x\}}{\{y, z\}}\right) > \min\left(p_M\left(\frac{\{w, y\}}{\{x, z\}}\right), p_M\left(\frac{\{w, z\}}{\{x, y\}}\right)\right). \tag{9.2}$$

As this condition is a proper inequality, at most two quartet topologies are chosen for any given quadruple of taxa.

To obtain a set of splits that induce only the selected quartet topologies, we define the *parsimony index* of a split $S = \frac{A}{B}$ as

$$p_M(S) = \min_{\substack{w,x \in A \\ y,z \in B}} \left(p_M\left(\frac{\{w, x\}}{\{y, z\}}\right) - \min\left(p_M\left(\frac{\{w, x\}}{\{y, z\}}\right), p_M\left(\frac{\{w, y\}}{\{x, z\}}\right), p_M\left(\frac{\{w, z\}}{\{x, y\}}\right)\right)\right) \geq 0. \tag{9.3}$$

9.4 Median networks

A split S is called a *p-split*, if its parsimony index $p_M(S)$ is positive. The set of all p-splits associated with M is weakly compatible by Lemma 5.9.4. It can be computed using a simple modification of the split decomposition algorithm as follows:

Algorithm 9.3.1 (Parsimony splits) *Given a multiple sequence alignment M on $\mathcal{X} = \{x_1, \ldots, x_n\}$, compute the set of all p-splits on $\mathcal{X}$ as follows:*

Initially, set $\mathcal{X}_1 = \{x_1\}$ and $\mathcal{S}_1 = \emptyset$. Now, assume that we have computed the set of all p-splits $\mathcal{S}_i$ on the first i taxa $\mathcal{X}_i = \{x_1, \ldots, x_i\}$. To obtain $\mathcal{S}_{i+1}$ on $\mathcal{X}_{i+1} = \{x_1, \ldots, x_{i+1}\}$, for each split $A \mid B \in \mathcal{S}_i$ do:

(i) *If* $p_M\left(\frac{A \cup \{x_{i+1}\}}{B}\right) > 0$, *then add* $\frac{A \cup \{x_{i+1}\}}{B}$ *to* $\mathcal{S}_{i+1}$.

(ii) *If* $p_M\left(\frac{A}{B \cup \{x_{i+1}\}}\right) > 0$, *then add* $\frac{A}{B \cup \{x_{i+1}\}}$ *to* $\mathcal{S}_{i+1}$.

(iii) *If* $p_M\left(\frac{\mathcal{X}_i}{\{x_{i+1}\}}\right) > 0$, *then add* $\frac{\mathcal{X}_i}{\{x_{i+1}\}}$ *to* $\mathcal{S}_{i+1}$.

The result is given by $\mathcal{S}_n$.

The splits computed by the algorithm are then usually processed by a method such as the convex hull algorithm (see Section 7.1) to produce a split network.

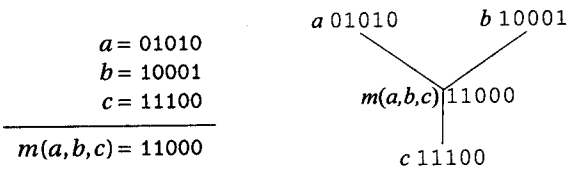
9.3.1 Application

The *parsimony-splits* method has not been used much in the literature, probably because the resulting set of splits is usually very similar to the one obtained by the more widely known split-decomposition method. An example of the application of the method can be found in a study that compares the phylogeny of different Hox clusters in sharks and mammals [200]. The parsimony-splits algorithm is used together with a number of other phylogenetic methods to support the hypothesis that the shark *HoxN* cluster is orthologous to the mammalian *HoxD* cluster.

9.4 Median networks

Median networks were developed as a means of visualizing the variation in mitochondrial DNA (mtDNA) sequences in the study of human evolution [11]. They are usually constructed for closely related sequences that have evolved without recombinations. In a median network N, every sequence of a given multiple sequence alignment M on $\mathcal{X}$ is represented by a node and additional nodes are said to represent further *unobserved* sequences. Two nodes are connected by an edge if and only if they differ by exactly one mutation (under the assumption that the alignment is condensed).

In more detail, assume that we are given a multiple alignment of DNA sequences M_0 on $\mathcal{X}$. As we shall see, the median network algorithm works only for binary sequences. Thus, the first step is to convert the original input data into a condensed



(a) Median calculation (b) Parsimonious tree

Figure 9.2 (a) The median $m(a, b, c)$ of three binary sequences a, b and c. The median $m(a, b, c)$ minimizes the parsimony score of the unrooted phylogenetic tree on $\{a, b, c\}$ shown in (b), which in this case is five.

multiple alignment of binary sequences M on $\mathcal{X}$, as described in Section 9.2. Recall that condensed means that no two columns of M exhibit the same pattern, there are no constant columns and no two taxa have the same sequence.

Consider three binary sequences $a = a_1 a_2 \ldots a_L$, $b = b_1 b_2 \ldots b_L$ and $c = c_1 c_2 \ldots c_L$. We define their *median* as the sequence $m(a, b, c)$ whose i-th character state is the majority state of a_i, b_i and c_i, which is 1, if $a_i + b_i + c_i \geq 2$ and 0, else. An example is shown in Figure 9.2a.

The median sequence $\mu = m(a, b, c)$ has the property that it minimizes the sum of pairwise Hamming distances (see Section 3.12) $H(a, \mu) + H(b, \mu) + H(c, \mu)$ and provides a most parsimonious sequence for the internal node of the unrooted phylogenetic tree on $\{a, b, c\}$, as illustrated in Figure 9.2b.

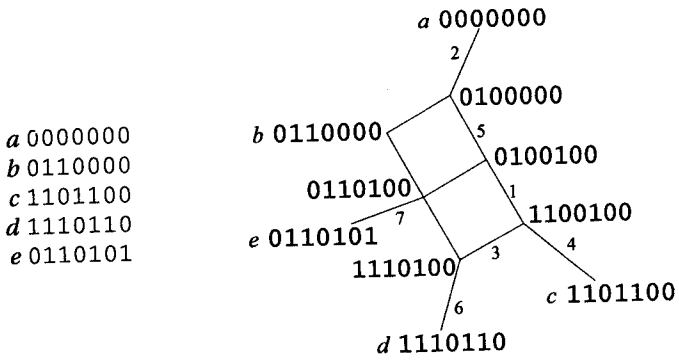
Let M be a multiple alignment of binary sequences on $\mathcal{X}$. The *median closure* of M is defined as the set $\bar{M}$ of all binary sequences that can be obtained by repeatedly taking the median of any three sequences in the set and adding it to the set, until no new sequences can be produced. The median network is defined as follows:

Definition 9.4.1 (Median network) *Let M be a condensed multiple alignment of binary sequences on $\mathcal{X}$. The* median network *associated with M is a phylogenetic network $N = (V, E, \sigma, \lambda)$ whose node set is given by the median closure $V = \bar{M}$ and in which any two nodes a and b are connected by an edge e of color $\sigma(e) = i$ in E, if any only if they differ exactly in their i-th position (as haplotypes). An associated taxon labeling $\lambda : \mathcal{X} \to V$ maps each taxon x onto the node $\lambda(x)$ that represents the corresponding sequence.*

An example of a median network is shown in Figure 9.3.

Let M be a condensed multiple alignment of binary sequences on $\mathcal{X}$. Definition 9.4.1 suggests the following *naive* algorithm for computing the median network N for M: First compute the median closure $\bar{M}$. Then construct the graph N that has node set $\bar{M}$ and in which any two nodes are connected by an edge e of color $\sigma(e) = i$ if any only if they differ exactly in their i-th position (as haplotypes). Alternatively, one can compute the canonical split network $N(\mathcal{S}(M))$ for the set

9.4 Median networks



(a) Alignment M (b) Median network N

Figure 9.3 (a) A condensed multiple alignment M of binary sequences on $\mathcal{X} = \{a, \ldots, e\}$. (b) The median network N that represents M. The nodes are labeled by haplotypes and the edges are labeled by the corresponding columns in M.

of splits $\mathcal{S}(M)$ associated with M using the convex hull algorithm, based on the following result [6]:

Theorem 9.4.2 (Median network is split network) *Let M be a condensed multiple alignment of binary sequences on $\mathcal{X}$. The naive median network algorithm and the convex hull algorithm both produce the same network.*

An important consequence of this result is the following:

Corollary 9.4.3 (Median network is connected) *Let M be a condensed multiple alignment of binary sequences on $\mathcal{X}$. The median network N associated with M is connected.*

Let M be a condensed multiple alignment of binary sequences on $\mathcal{X}$. If all pairs of columns in M are compatible, it follows from Theorem 9.4.2 that the resulting median network will be a phylogenetic tree T. Trivially, this tree will be the most parsimonious tree for M (see Section 3.7). More generally we have the following result [11]:

Theorem 9.4.4 (All most parsimonious trees) *Let M be a condensed multiple alignment of binary sequences on $\mathcal{X}$. The median network N associated with M contains all phylogenetic trees T on $\mathcal{X}$ whose parsimony score $PS(T, M)$ is minimum.*

9.4.1 Reduced median networks

The main application of a median network is to provide a useful visualization of probable evolutionary pathways. Unfortunately, a median network can have a very

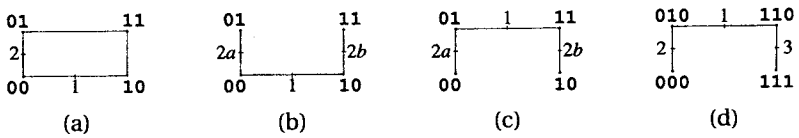
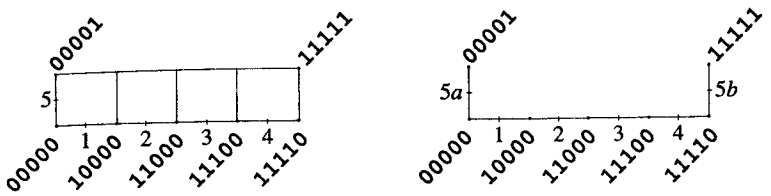


Figure 9.4 (a) A median network restricted to two incompatible binary characters, where the weight of the first character is much higher than the weight of the second one. In (b) and (c) we show the two possible ways of resolving the incompatibility by postulating two parallel changes of the second character. We choose the reduction shown in (b), if the two lower sequences occur in the given multiple sequence alignment more frequently that the two upper ones, and we choose the reduction shown in (c) otherwise. (d) The resolution indicated in (c) can be interpreted as the replacement of the second character by two new characters.

large number of nodes and edges, even for a small number of taxa and characters, if there are a lot of incompatibilities among the characters. In the absence of recombination, many of the incompatibilities will be due to parallel- or back-mutations. Hence, one idea to reduce the complexity of the network is to try and resolve incompatibilities by identifying likely parallel mutations and making them explicit. The resulting network is called a *reduced median network*.

In Figure 9.4a, we display a section of a median network restricted to four binary sequences `00`, `01`, `10` and `11`. Let us assume that the first binary character has a substantially higher weight than the second one, for example $\omega(1) \geq 2\,\omega(2)$. In this case we will assume that the mutation represented by the second character occurred twice, in parallel, as indicated in parts (b) and (c) of the figure. To decide which of the two depicted scenarios is more likely, we count the frequencies of common occurrences of the patterns `00` or `10`, and of the patterns `01` or `11`, in the given multiple sequence alignment M. If the former two patterns occur with a higher frequency than the latter two, then we will resolve the parallel mutations as shown in (b), otherwise we will resolve them as shown in (c). One way to implement this step computationally is replace the character in which a parallel mutation is assumed to occur by two new characters that each mutate only once. For example, for the resolution indicated in (c), we replace the second character in the sequence by two new characters to obtain the four sequences `000`, `010`, `111` and `110` in which all characters are compatible with each other, as illustrated in (d).

When one character in the multiple sequence alignment M is incompatible with a whole set of other characters that are all compatible with each other, then again the median network can be simplified by postulating an appropriate parallel mutation, as indicated in Figure 9.5. We do not describe in detail how to systematically detect all opportunities for reduction in a median network.



9.5 (a) This median network shows that the fifth position of the given multiple sequence alignment M is incompatible with the other four positions. (b) The reduced median network obtained after postulating a parallel mutation at position five of the alignment.

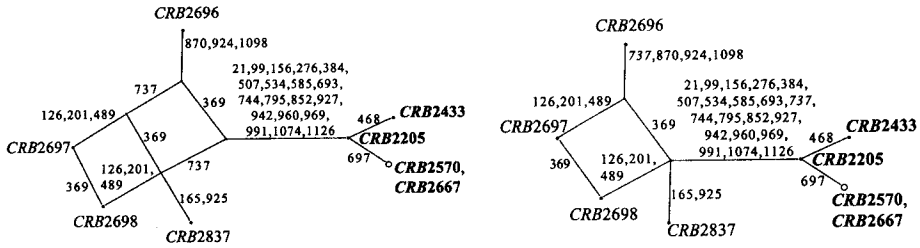


Figure 9.6 (a) The median network for eight specimens of *Callicebus lugens*, based on cytochrome b sequences. Specimens from the right bank of Rio Negro are shown in plain font and those from the left bank are shown in bold font. (b) The reduced median network obtained by postulating a parallel mutation at position 737.

9.4.2 Application

One application domain of median networks is phylogeographic studies. For example, in [44] the distribution of *Callicebus lugens* (Platyrrhini, Primates) at the Rio Negro, in Brazil, is reported. The study focuses on eight specimens, one group of four taken from the left bank of the river and another group of four taken from the right bank. It is based on a multiple alignment M of cytochrome b DNA sequences of length 1140. A median-network analysis shows a clear separation of the two groups, see Figure 9.6. Note that only 35 columns are retained in the condensed version of M (the ones labeling the edges).

9.5 Quasi-median networks

In Section 9.4, we show that median networks are designed for representing binary sequences. To apply them to DNA (or other multi-state sequences), one must first convert the sequences into binary sequences as described in Section 9.2. Because such a conversion usually discards phylogenetic information, the question arises:

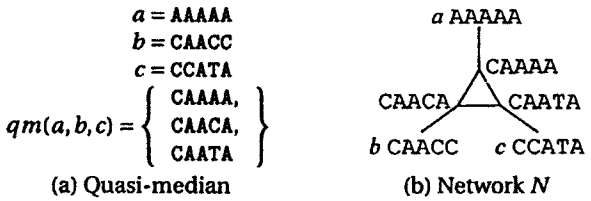


Figure 9.7 (a) The quasi-median $qm(a, b, c)$ for three sequences a, b and c. (b) The corresponding quasi-median network N.

How to modify the median calculation so that it works directly with multi-state data? The answer is simple: whenever a column of three different states occurs in the computation of the median, produce three median sequences, one for each of the different states. More formally, we introduce the concept of a quasi-median:

Definition 9.5.1 (Quasi-median) *Consider three DNA sequences $a = a_1 \dots a_L$, $b = b_1 \dots b_L$ and $c = c_1 \dots c_L$. The* quasi-median *of a, b and c is the set $qm(a, b, c)$ of all sequences $d = d_1 \dots d_L$ that have the property that the state d_i occurs in the set $\{a_i, b_i, c_i\}$ at least as many times as any other state, for each position $i = 1, \dots, L$.*

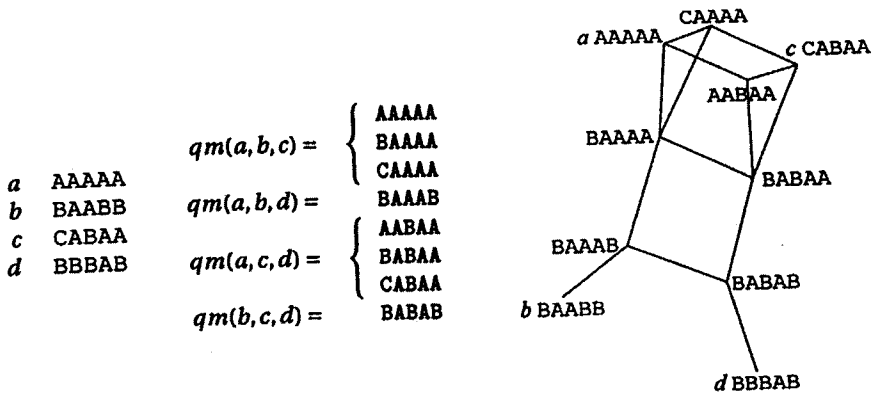
In other words, $d = d_1 \dots d_L$ is a member of $qm(a, b, c)$ if and only if either the i-th column (of the alignment of a, b and c) contains at most two different states and d_i is the majority state, or the column contains three different states and d_i is any one of them, for all i. If the number of columns containing three different states is k, then $qm(a, b, c)$ will contain 3^k different sequences. An example with $k = 1$ is shown in Figure 9.7.

Let M be a multiple alignment of DNA sequences on $\mathcal{X}$. We define the *quasi-median closure* $\bar{M}$ as the set of all sequences that can be obtained by repeatedly taking the quasi-median of any three sequences in the set and then adding the result to the set.

A *quasi-median network* is a generalization of the concept of a median network that takes *all* non-constant columns of a multiple alignment of DNA sequences [10] into account:

Definition 9.5.2 (Quasi-median network) *Let M be a condensed multiple alignment of DNA sequences on $\mathcal{X}$. The* quasi-median network *associated with M is a phylogenetic network $N = (V, E)$ whose node set is given by the quasi-median closure $N = \bar{M}$ and in which any two nodes are joined by an edge in E if and only if they differ in exactly one position (as haplotypes). The associated taxon labeling $\lambda : \mathcal{X} \to V$ maps each taxon x onto the node $\lambda(x)$ that represents the corresponding sequence.*

9.5 Quasi-median networks



(a) Input M (b) Quasi-medians for M (c) Quasi-median network N

Figure 9.8 For the multiple sequence alignment M of multi-state characters in (a) we list all quasi-medians in (b). Note that taking medians of any of the sequences listed in (a) and (b) does not produce any new sequences, and so in this example the quasi-median closure $\bar{M}$ equals the set of listed sequences. In (c) we show the corresponding quasi-median network N.

The quasi-median network of a condensed multiple sequence alignment M is uniquely defined. Moreover, one can show that it is connected [10]. An example is shown in Figure 9.8. Recall that each column in a condensed alignment corresponds to a *pattern* in the original DNA alignment, which we describe using letters A, B, . . . Note that the depicted network contains cycles of length three ("triangles"), thus illustrating that a quasi-median network is *not* a split network in general.

9.5.1 Algorithm

In theory, the quasi-median network can be constructed by first computing the quasi-median closure $\bar{M}$ of the given input dataset M and then producing the phylogenetic network whose node set is given by $\bar{M}$ and whose edges are given by all pairs of haplotypes in $\bar{M}$ that differ in precisely one position, as shown in Figure 9.8.

In practice, it is more efficient to employ a modification of the strategy used for median networks [5]. The basic idea is to encode each column of multi-state data as a group of binary columns, to then compute the median network for the alignment of binary sequences, and then finally to convert back to multi-state characters, being careful to expand each encountered "virtual median" into a set of sequences that display all appropriate multi-state characters.

In more detail, assume that we are given a condensed multiple alignment M of DNA sequences on $\mathcal{X}$. For ease of exposition, assume that the characters in M are labeled A, B, . . ., E, representing the four DNA bases and the gap character.

A	0	A	0	0	0	A	0	0	0	0	A	0	0	0	0	0
B	1	B	1	1	0	B	1	1	0	0	B	1	1	0	0	0
		C	1	0	1	C	1	0	1	0	C	1	0	1	0	0
						D	1	0	0	1	D	1	0	0	1	0
											E	1	0	0	0	1

(a) 2 states (b) 3 states (c) 4 states (d) 5 states

Figure 9.9 The binary expansion M_1 of a condensed multiple alignment M of DNA sequences is obtained by replacing each column that contains 2, 3, 4 or 5 different states by a group of 1 to 5 binary columns, converting each state A–E into a corresponding binary pattern, as indicated in (a) to (d), respectively.

The first step is to compute the *binary expansion* M_1 of M in which every column of M is represented by a *group* of binary columns in M_1. For each column C of M, we compute the corresponding group of columns in M_1 as follows: If the column C contains only two different states, then the corresponding group of columns in M_1 contains only one column, which is obtained from C by changing all occurrences of A by 0 and all occurrences of B by 1. If the number of different states contained in the column C is $d > 2$, then the column C gives rise to a group of d binary columns, in which each of the letters A–E is represented by a specific *binary pattern* as prescribed in Figure 9.9.

The second step is to compute the median closure $M_2 = \bar{M}_1$. For computational reasons, one should first collapse all identical columns in M_1 before computing $\bar{M}_1$ and then expand them all again to obtain M_2.

The third step is to convert M_2 back to a representation M_3 of multi-state data. To do this, each group of columns is converted back to a single column using the tables shown in Figure 9.9. Note that a group of three or more columns may contain binary patterns that are not listed in the tables. Such binary patterns are produced by the median operation and we refer to them as *virtual medians*, as they are not the medians of binary characters but rather of our binary encoding of multi-state characters. Virtual medians are represented in M_3 by the state '$*$'.

By definition, a virtual median is obtained as the median of three different binary patterns that represent three different multi-state characters, according to the tables shown in Figure 9.9. (Please convince yourself that the binary patterns listed in each of the tables have the property that the median of any three different ones is not contained in the table.)

In the fourth step we obtain a new multiple alignment M_4 of multi-state characters from M_3 by expanding each sequence M_3 that contains a virtual median $*$ into a set of new sequences, each one obtained by replacing each virtual median by any one of the states A–E that occur in the column. Remove any duplicate columns from M_4.

Finally, the quasi-median network N associated with the given multiple alignment of DNA sequences M is constructed as the graph $N = (V, E)$ that has node set $V = M_4$ and in which any two nodes are connected by an edge in E if and only if the two nodes differ in exactly one position (as haplotypes). Here is a summary of the algorithm [5].

Algorithm 9.5.3 (Quasi-median network) *Let M be a condensed multiple alignment of DNA sequences on $\mathcal{X}$. The quasi-median network N associated with M is computed in the following steps:*

(i) *Compute the binary expansion M_1 for M.*
(ii) *Compute the median closure M_2 for M_1.*
(iii) *Compute the multi-state representation M_3 for M_2.*
(iv) *Compute M_4 by expanding all virtual medians in M_3 and then removing any duplicate copies.*
(v) *Construct the graph $N = (V, E)$ that has node set $V = M_4$ and in which any two nodes v and w are connected by an edge e in E with label i, if and only if v and w differ only in their i-th position (as haplotypes).*

An example illustrating this algorithm is presented in Figure 9.10.

9.6 Median-joining

The number of nodes of the quasi-median network associated with a condensed multiple alignment of DNA sequences M on $\mathcal{X}$ can be very large, even for a small number of short sequences. Thus, the quasi-median network is rarely useful in practice. In this section, we describe the *median-joining* algorithm [10] that aims at computing a phylogenetic network that is as informative as a quasi-median network, but usually much smaller. As we will see, the algorithm has a parameter Δ that is used to control how complex the resulting phylogenetic network will be.

Let M be a condensed multiple alignment of DNA sequences on $\mathcal{X}$. The median-joining method brings together two different algorithmic ideas. On the one hand, it repeatedly uses the concept of a "(relaxed) minimum spanning network", and on the other hand, it repeatedly employs the quasi-median calculation. While the former construction, on its own, will produce too few nodes to be useful, the latter construction, on its own, will produce too many nodes. Using both together, the median-joining method attempts to provide a useful network of intermediate size.

9.6.1 The relaxed minimum spanning network

Let M be a condensed multiple alignment of DNA sequences $\mathcal{X}$. To be able to employ the concept of a minimum spanning network, we need to define a distance

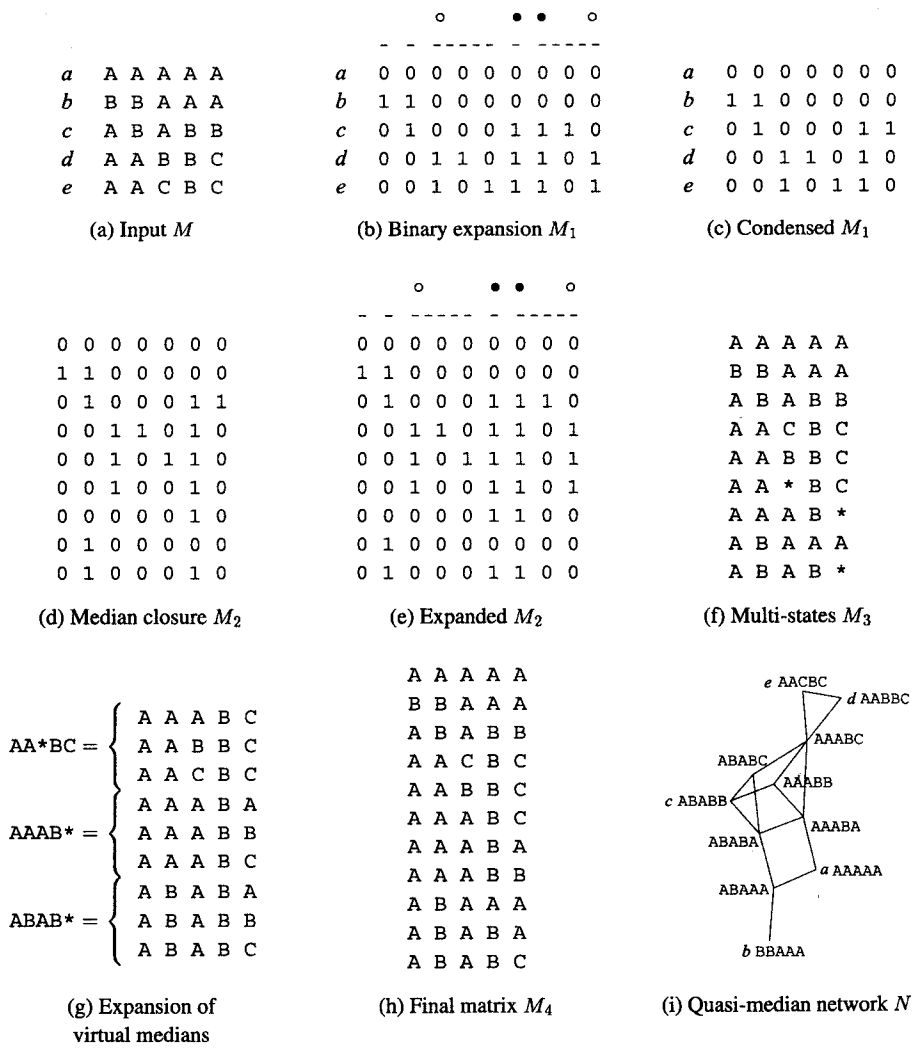


Figure 9.10 For the condensed multiple sequence alignment M of multi-state characters in (a) we first compute the binary expansion M_1 (b). We then generate the median closure $M_2 = \bar{M}_1$, first collapsing all identical columns in M_1 (c) before computing $\bar{M}_1$ (d). We record which columns are collapsed together (marked ∘ and • in (b)) and then expand them all again to obtain M_2 (e). We then compute the multi-state representation M_3 for M_2 (f). Finally, we obtain the matrix M_4 by expanding all virtual medians in M_3 (g) and then removing any duplicate copies (h). In (i) we show the quasi-median network N for M.

matrix D on $\mathcal{X}$. A natural choice is to use the (unnormalized) Hamming distance $H(a, b)$ between any two sequences a and b in M.

In practice, the multiple sequence alignment M will usually be obtained by condensing some original input multiple sequence alignment M_0. To take this into

account, whenever we compare two sequences in M we must weight each position in M by the number of original positions in M_0 that it represents. This gives rise to a distance matrix D in which the distance $d(a, b)$ between any two sequences a and b in M is given by the sum of weights of all positions at which a and b differ.

Consider the graph $G = (V, E, \omega)$ that has node set $V = \mathcal{X}$ and whose edge set E contains all possible edges between any two nodes in V. Moreover, let $\omega : E \to \mathbb{R}^{\geq 0}$ be an edge weighting that reflects D, that is, with $\omega(e) = d(v, w)$ for every edge $e = \{v, w\}$ in E. We will call G the *distance graph* associated with M.

A *minimum spanning tree* for G is any subtree T of G that connects all nodes of G and that minimizes the sum of edge weights given by $\omega(T) = \sum_{e \text{ in } T} \omega(e)$. There are two different algorithms for computing a minimum spanning tree, Prim's algorithm [199] and Kruskal's algorithm [155].

The latter starts with a subgraph T that contains only the nodes of G. It then considers each edge e of G in ascending order of weight and adds e to T, if e joins two different connected components of T. The algorithm terminates as soon as the subgraph T becomes connected.

In the algorithm all edges that have the same weight are processed consecutively in some order. This order serves as an implicit "tie-breaking" rule and different orderings may give rise to different minimum spanning trees.

The *minimum spanning network* for M is defined as the subgraph N of the corresponding distance graph G whose edge set is obtained as the union of the edge sets of all minimum spanning trees of G. It can be computed as follows [10]:

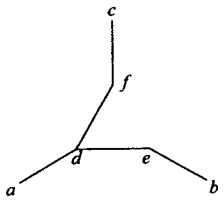
Algorithm 9.6.1 (Minimum spanning network) *For a given edge-weighted, connected graph $G = (V, E, \omega)$, the minimum spanning network N is computed as follows: Initially, let N be the subgraph of G that contains all nodes of G and no edges. Let $w_1 < w_2 \ldots < w_t$ denote the different edge weights that occur in G. For $i = 1, \ldots, t$ repeat the following steps:*

(i) *Add all edges of weight w_i to N that join different connected components.*
(ii) *If N is connected, terminate.*

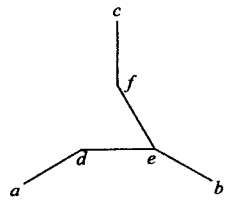
In this algorithm, all edges of the same weight are treated equally, that is, either they are all put into the minimum spanning network, or they are all left out. For real data, it can be useful to introduce a tolerance Δ up to which we do not distinguish between different distance values. Let G be a distance graph and let N be the corresponding minimum spanning network. We define the *relaxed minimum spanning network* N' with parameter $\Delta > 0$ as the graph that contains all edges of

	a	b	c	d	e	f
a	0	3	3	1	2	2
b	3	0	3	2	1	2
c	3	3	0	2	2	1
d	1	2	2	0	1	1
e	2	1	2	1	0	1
f	2	2	1	1	1	0

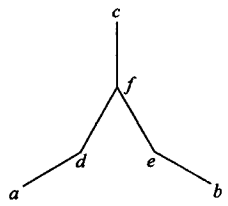
(a) Distances M



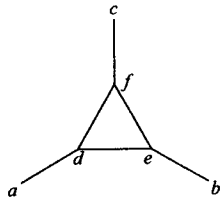
(b) Minimum spanning tree T_1



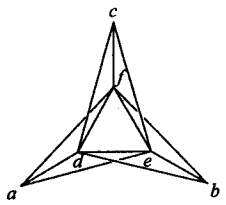
(c) Minimum spanning tree T_2



(d) Minimum spanning tree T_3



(e) Minimum spanning network N



(f) Relaxed minimum spanning network N'

Figure 9.11 The distance matrix M on $\mathcal{X} = \{a, \ldots, f\}$ shown in (a) gives rise to three different minimum spanning trees T_1, T_2 and T_3, shown in (b), (c) and (d), respectively. The corresponding minimum spanning network N is shown in (e). The relaxed minimum spanning network N' for $\Delta = 1$ is shown in (f).

N, and, in addition, also those heavier edges of G whose weights do not exceed the heaviest weight in N by more than Δ.

Exercise 9.6.2 *Modify Algorithm 9.6.1 so that it computes the relaxed minimum spanning network for parameter* Δ.

An alternative method of computing the minimum spanning network can be found in [72].

In Figure 9.11, we show a distance matrix D on $\mathcal{X} = \{a, \ldots, f\}$ that gives rise to three different minimum spanning trees T_1, T_2 and T_3. The displayed minimum spanning network N clearly contains all edges of the three trees. As the heaviest weight in N is one, the relaxed minimum spanning network N' with $\Delta = 1$ additionally contains six edges of weight two.

9.6.2 The median-joining algorithm

Let M be a condensed multiple alignment of DNA sequences on $\mathcal{X}$. The median-joining algorithm uses the quasi-median operation to compute new sequences from existing ones. To avoid computing the full quasi-median closure, the algorithm is guided by the relaxed minimum spanning network N_Δ of the distance graph G

associated with M, for a given user-specified tolerance $\Delta \geq 0$. The idea is to apply the quasi-median calculation only to sets of sequences that lie close to each other in the relaxed minimum spanning network N_Δ and are not already connected to each other by chains of short edges [10]:

Algorithm 9.6.3 (Median-joining) *Let M be a condensed multiple alignment of DNA sequences on $\mathcal{X}$ and $\Delta \geq 0$. The associated median-joining network N is computed in the following steps:*

(i) *Set S equal to the set of sequences contained in M.*

(ii) *Compute the relaxed minimum spanning network N_Δ for the current set of sequences S. Determine the set of all* couples *$\{a, b\}$ of sequences a and b in S that are connected by an edge e in N_Δ and are not connected by any chain of edges $e_1, \ldots, e_t$ in which each edge has a weight of at most $\omega(e) - \Delta$.*

(iii) *Any node contained in N_Δ that does not represent one of the original input sequences in M and is only contained in at most two couples is deemed* obsolete *and is removed from S. If any removals occur, return to step (ii).*

(iv) *Decide which new sequences to add to S in the following steps.*

(a) *Let T be the set of all triples $\{a, b, c\}$ of sequences in S for which at least two of the three subsets $\{a, b\}, \{b, c\}, \{a, c\}$ are couples. Compute $\mu = qm(a, b, c)$ for all $\{a, b, c\}$ in T.*

(b) *For each sequence f in μ that is not already contained in S, compute its* connection cost *as $d(f, a) + d(f, b) + d(f, c)$, where $d(x, y)$ is computed as the weighted number of differences between sequences x and y.*

(c) *Let λ be the minimum connection cost seen in (b) over all triples in T.*

(d) *For each triple $\{a, b, c\}$ in T, compute $\mu = qm(a, b, c)$, and then, for every sequence f in μ, add f to S, if the connection cost for f does not exceed $\lambda + \Delta$.*

(e) *If any new sequences are added to S in (d), return to step (ii).*

(v) *In this step we generate the final network. Compute the minimum spanning network N for S (with parameter $\Delta = 0$) and determine the set of couples, as in step (i). If any obsolete nodes are present, remove them and return to step (iv). Otherwise, the set of couples defines the edge set of the final network N.*

9.6.3 Application

Median-joining is a widely used method. An interesting application of this method can be found in [151], where a median-joining network on human populations is computed for mitochondrial DNA samples. The network (displayed in Figure 9.12) is consistent with the out-of-Africa model of human origins, which suggests that all non-African populations are derived from one African lineage.

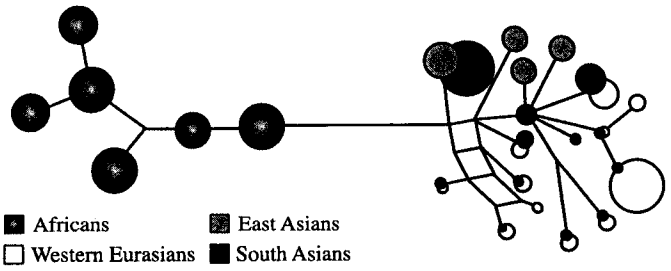


Figure 9.12 A median-joining network on human populations computed from mtDNA (adapted from [60, 151]). Each disk in the tree represents a cluster of human mitochondrial types and its diameter is proportional to the number of sequences represented. For African sequences, edges between individual types are collapsed and not shown. The cluster containing all non-African sequences is shown here in a non-collapsed view. Note that it attaches to only one of the clusters of African lineages. This network is consistent with the out-of-Africa model of human origins, suggesting that all non-African populations are derived from one African lineage.

9.7 Pruned quasi-median networks

As mentioned above, the number of nodes and edges in a quasi-median network can be unmanageably large. We saw that the median-joining method [10] tries to avoid constructing the full quasi-median closure using a heuristic based on the concept of a relaxed minimum spanning network. A more recent method [5] aims at computing a pruned version of the full quasi-median network by considering only those sequences that lie on a *geodesic* between two of the original input sequences. The result is called a *geodesically pruned quasi-median network.*

The definition and computation of geodesics require some additional concepts.

Definition 9.7.1 (Restricted quasi-median) *Consider three DNA sequences a, b and c of length L. Moreover, assume that we are given two* reference sequences *g and h of length L. The* restricted quasi-median *of a, b and c,* relative to *g and h is defined as the set $qm_{g,h}(a, b, c)$ of all sequences d of length L that have the property that d_i is the majority state in (a_i, b_i, c_i), if there is one, and otherwise d_i is contained in $\{a_i, b_i, c_i\} \cap \{g_i, h_i\}$, for all positions $i = 1, \ldots, L$.*

Let M be a condensed multiple alignment of DNA sequences of length L. For each character state c and each position i in M we define the score $s(c, i)$ as the fraction of sequences in M that have state c at position i. The score of an arbitrary sequence a of length L is then defined as

$$s(a) = \log\left(\sum_{i=1}^{L} s(a_i, i)\right). \tag{9.4}$$

If a and b are two sequences of length L that are nodes of some graph G whose nodes are all sequences of length L, then a *geodesic* between a and b is any path $P = (v_1, v_2, \ldots, v_t)$ of nodes from a to b in G that has minimum total score $s(P) = \sum_{i=1}^{t} s(v_i)$. The geodesically-pruned quasi-median network can be computed using the following algorithm [5]:

Algorithm 9.7.2 (Geodesically pruned quasi-median network) *Let M be a condensed multiple alignment of DNA sequences on $\mathcal{X}$. Output is the associated geodesically pruned quasi-median network N. Process each pair of sequences a and b in M as follows:*

(i) *Extract the sub-alignment M_{ab} from M that consists of all columns of M for which the states of a and b differ.*
(ii) *Compute the closure $\bar{M}_{ab}$ of the restricted quasi-median operation on M_{ab} relative to a and b.*
(iii) *Construct the graph G_{ab} with node set $\bar{M}_{ab}$ in which any two nodes are connected by an edge if and only if they differ by exactly one position (as haplotypes).*
(iv) *Remove all sequences from $\bar{M}_{ab}$ that do not lie on a geodesic from a to b in G_{ab}.*
(v) *Let S_{ab} be the result of expanding all sequences in $\bar{M}_{ab}$ back to their original length by reinserting all positions at which a and b have the same state.*

The final network $N = (V, E)$ has node set $V = \bigcup_{a \neq b} S_{ab}$ and an edge set E that consists of all pairs of nodes that differ by exactly one position (as haplotypes). An associated taxon labeling $\lambda : \mathcal{X} \to V$ maps each taxon x onto the node $\lambda(x)$ that represents the corresponding sequence.

9.7.1 Application

An application of the geodesically pruned quasi-median network algorithm can be found in [5], where it is applied to 110 sequences from the spacer regions of the 5S ribosomal DNA loci of sea beet (*Beta vulgaris* ssp. *martima*). The resulting phylogenetic network (reproduced in Figure 9.13) contains 648 nodes. It shows a separation between harbor and cliff top populations, in agreement with previous studies.

9.8 Recombination networks

A recombination network is a rooted phylogenetic network that is used to describe the evolution of a set of closely related sequences (usually from different individuals of a population) in terms of mutations (along edges of the network), speciation events (at tree nodes) and recombination events (at reticulate nodes).

In a simple recombination event, two sequences a and b of the same length L give rise to a new sequence c of length L that consists of a prefix of the sequence a

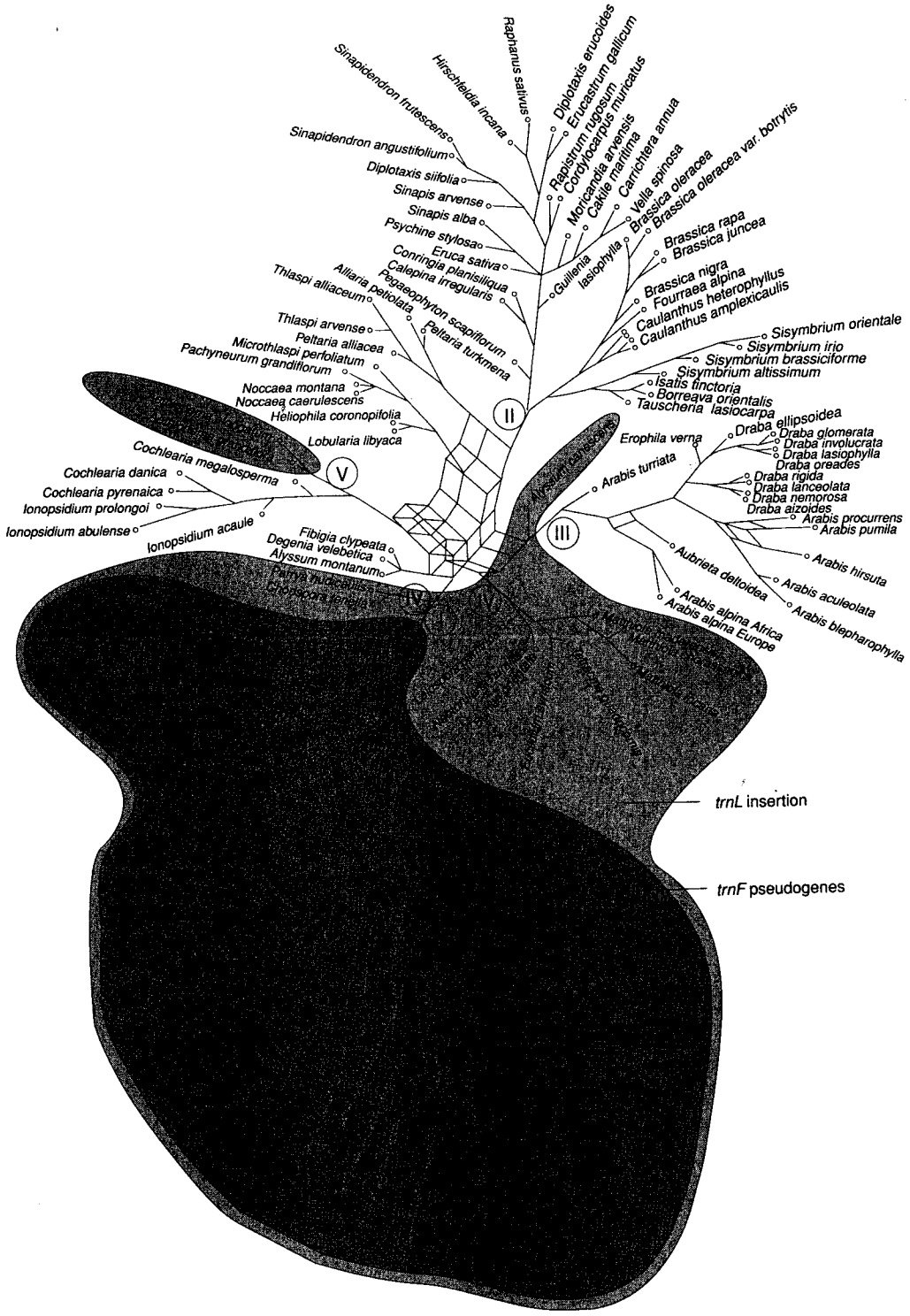


Figure 11.4 A super split network on 71 plant taxa. It was constructed from five different gene trees that were computed using the maximum parsimony method and then processed to remove uncertainties [154]. Reprinted by permission from Oxford University Press: Molecular Biology and Evolution, 24:63–73 ©2007.