6. Spanning Trees and Arborescences

Consider a telephone company that wants to rent a subset from an existing set of cables, each of which connects two cities. The rented cables should suffice to connect all cities and they should be as cheap as possible. It is natural to model the network by a graph: the vertices are the cities and the edges correspond to the cables. By Theorem 2.4 the minimal connected spanning subgraphs of a given graph are its spanning trees. So we look for a spanning tree of minimum weight, where we say that a subgraph T of a graph G with weights $c : E(G) \to \mathbb{R}$ has weight $c(E(T)) = \sum_{e \in E(T)} c(e)$.

This is a simple but very important combinatorial optimization problem. It is also among the combinatorial optimization problems with the longest history; the first algorithm was given by Borůvka [1926a,1926b]; see Nešetřil, Milková and Nešetřilová [2001].

Compared to the DRILLING PROBLEM which asks for a shortest path containing all vertices of a complete graph, we now look for a shortest tree. Although the number of spanning trees is even bigger than the number of paths (K_n contains $\frac{n!}{2}$ Hamiltonian paths, but, by a theorem of Cayley [1889], as many as n^{n-2} different spanning trees; see Exercise 1), the problem turns out to be much easier. In fact, a simple greedy strategy works as we shall see in Section 6.1.

Arborescences can be considered as the directed counterparts of trees; by Theorem 2.5 they are the minimal spanning subgraphs of a digraph such that all vertices are reachable from a root. The directed version of the MINIMUM SPANNING TREE PROBLEM, the MINIMUM WEIGHT ARBORESCENCE PROBLEM, is more difficult since a greedy strategy no longer works. In Section 6.2 we show how to solve this problem.

Since there are very efficient combinatorial algorithms it is not recommended to solve these problems with LINEAR PROGRAMMING. Nevertheless it is interesting that the corresponding polytopes (the convex hull of the incidence vectors of spanning trees or arborescences; cf. Corollary 3.28) can be described in a nice way, which we shall show in Section 6.3. In Section 6.4 we prove some classical results concerning the packing of spanning trees and arborescences.

6.1 Minimum Spanning Trees

In this section, we consider the following two problems:

MAXIMUM WEIGHT FOREST PROBLEM	
Instance:	An undirected graph G, weights $c : E(G) \to \mathbb{R}$.
Task:	Find a forest in G of maximum weight.

MINIMUM SPANNING TREE PROBLEM	
Instance:	An undirected graph G, weights $c : E(G) \to \mathbb{R}$.
Task:	Find a spanning tree in G of minimum weight or decide that G is not connected.

We claim that both problems are equivalent. To make this precise, we say that a problem $\mathcal{P}$ **linearly reduces** to a problem $\mathcal{Q}$ if there are functions f and g, each computable in linear time, such that f transforms an instance x of $\mathcal{P}$ to an instance $f(x)$ of $\mathcal{Q}$, and g transforms a solution of $f(x)$ to a solution of x. If $\mathcal{P}$ linearly reduces to $\mathcal{Q}$ and $\mathcal{Q}$ linearly reduces to $\mathcal{P}$, then both problems are called **equivalent**.

Proposition 6.1. *The* MAXIMUM WEIGHT FOREST PROBLEM *and the* MINIMUM SPANNING TREE PROBLEM *are equivalent.*

Proof: Given an instance (G, c) of the MAXIMUM WEIGHT FOREST PROBLEM, delete all edges of negative weight, let $c'(e) := -c(e)$ for all $e \in E(G')$, and add a minimum set F of edges (with arbitrary weight) to make the graph connected; let us call the resulting graph G'. Then instance (G', c') of the MINIMUM SPANNING TREE PROBLEM is equivalent in the following sense: Deleting the edges of F from a minimum weight spanning tree in (G', c') yields a maximum weight forest in (G, c).

Conversely, given an instance (G, c) of the MINIMUM SPANNING TREE PROBLEM, let $c'(e) := K - c(e)$ for all $e \in E(G)$, where $K = 1 + \max_{e \in E(G)} c(e)$. Then the instance (G, c') of the MAXIMUM WEIGHT FOREST PROBLEM is equivalent, since all spanning trees have the same number of edges (Theorem 2.4). □

We shall return to different reductions of one problem to another in Chapter 15. In the rest of this section we consider the MINIMUM SPANNING TREE PROBLEM only. We start by proving two optimality conditions:

Theorem 6.2. *Let (G, c) be an instance of the* MINIMUM SPANNING TREE PROBLEM*, and let T be a spanning tree in G. Then the following statements are equivalent:*

(a) *T is optimum.*
(b) *For every $e = \{x, y\} \in E(G) \setminus E(T)$, no edge on the x-y-path in T has higher cost than e.*
(c) *For every $e \in E(T)$, e is a minimum cost edge of $\delta(V(C))$, where C is a connected component of $T - e$.*

Proof: (a)⇒(b): Suppose (b) is violated: Let $e = \{x, y\} \in E(G) \setminus E(T)$ and let f be an edge on the x-y-path in T with $c(f) > c(e)$. Then $(T - f) + e$ is a spanning tree with lower cost.

(b)⇒(c): Suppose (c) is violated: let $e \in E(T)$, C a connected component of $T - e$ and $f = \{x, y\} \in \delta(V(C))$ with $c(f) < c(e)$. Observe that the x-y-path in T must contain an edge of $\delta(V(C))$, but the only such edge is e. So (b) is violated.

(c)⇒(a): Suppose T satisfies (c), and let T^* be an optimum spanning tree with $E(T^*) \cap E(T)$ as large as possible. We show that $T = T^*$. Namely, suppose there is an edge $e = \{x, y\} \in E(T) \setminus E(T^*)$. Let C be a connected component of $T - e$. $T^* + e$ contains a circuit D. Since $e \in E(D) \cap \delta(V(C))$, at least one more edge f ($f \neq e$) of D must belong to $\delta(V(C))$ (see Exercise 9 of Chapter 2). Observe that $(T^* + e) - f$ is a spanning tree. Since T^* is optimum, $c(e) \geq c(f)$. But since (c) holds for T, we also have $c(f) \geq c(e)$. So $c(f) = c(e)$, and $(T^* + e) - f$ is another optimum spanning tree. This is a contradiction, because it has one edge more in common with T. □

The following "greedy" algorithm for the Minimum Spanning Tree Problem was proposed by Kruskal [1956]. It can be regarded as a special case of a quite general greedy algorithm which will be discussed in Section 13.4. In the following let $n := |V(G)|$ and $m := |E(G)|$.

Kruskal's Algorithm

Input: A connected undirected graph G, weights $c : E(G) \to \mathbb{R}$.

Output: A spanning tree T of minimum weight.

① Sort the edges such that $c(e_1) \leq c(e_2) \leq \ldots \leq c(e_m)$.

② Set $T := (V(G), \emptyset)$.

③ **For** $i := 1$ **to** m **do**:
If $T + e_i$ contains no circuit **then** set $T := T + e_i$.

Theorem 6.3. Kruskal's Algorithm *works correctly.*

Proof: It is clear that the algorithm constructs a spanning tree T. It also guarantees condition (b) of Theorem 6.2, so T is optimum. □

The running time of Kruskal's Algorithm is $O(mn)$: the edges can be sorted in $O(m \log m)$ time (Theorem 1.5), and testing for a circuit in a graph with at most n edges can be implemented in $O(n)$ time (just apply DFS (or BFS) and check if there is any edge not belonging to the DFS-tree). Since this is repeated m times, we get a total running time of $O(m \log m + mn) = O(mn)$. However, a more efficient implementation is possible:

Theorem 6.4. KRUSKAL'S ALGORITHM *can be implemented to run in* $O(m \log n)$ *time.*

Proof: Parallel edges can be eliminated first: all but the cheapest edges are redundant. So we may assume that $m = O(n^2)$. Since the running time of ① is obviously $O(m \log m) = O(m \log n)$ we concentrate on ③. We study a data structure maintaining the connected components of T. In ③ we have to test whether the addition of an edge $e_i = \{v, w\}$ to T results in a circuit. This is equivalent to testing if v and w are in the same connected component.

Our implementation maintains a branching B with $V(B) = V(G)$. At any time the connected components of B will be induced by the same vertex sets as the connected components of T. (Note however that B is in general not an orientation of T.)

When checking an edge $e_i = \{v, w\}$ in ③, we find the root r_v of the arborescence in B containing v and the root r_w of the arborescence in B containing w. The time needed for this is proportional to the length of the r_v-v-path plus the length of the r_w-w-path in B. We shall show later that this length is always at most $\log n$.

Next we check if $r_v = r_w$. If $r_v \neq r_w$, we insert e_i into T and we have to add an edge to B. Let $h(r)$ be the maximum length of a path from r in B. If $h(r_v) \geq h(r_w)$, then we add an edge (r_v, r_w) to B, otherwise we add (r_w, r_v) to B. If $h(r_v) = h(r_w)$, this operation increases $h(r_v)$ by one, otherwise the new root has the same h-value as before. So the h-values of the roots can be maintained easily. Of course initially $B := (V(G), \emptyset)$ and $h(v) := 0$ for all $v \in V(G)$.

We claim that an arborescence of B with root r contains at least $2^{h(r)}$ vertices. This implies that $h(r) \leq \log n$, concluding the proof. At the beginning, the claim is clearly true. We have to show that this property is maintained when adding an edge (x, y) to B. This is trivial if $h(x)$ does not change. Otherwise we have $h(x) = h(y)$ before the operation, implying that each of the two arborescences contains at least $2^{h(x)}$ vertices. So the new arborescence rooted at x contains at least $2 \cdot 2^{h(x)} = 2^{h(x)+1}$ vertices, as required. □

The above implementation can be improved by another trick: whenever the root r_v of the arborescence in B containing v has been determined, all the edges on the r_v-v-path P are deleted and an edge (r_x, x) is inserted for each $x \in V(P) \setminus \{r_v\}$. A complicated analysis shows that this so-called path compression heuristic makes the running time of ③ almost linear: it is $O(m\alpha(m, n))$, where $\alpha(m, n)$ is the functional inverse of Ackermann's function (see Tarjan [1975,1983]).

We now mention another well-known algorithm for the MINIMUM SPANNING TREE PROBLEM, due to Jarník [1930] (see Korte and Nešetřil [2001]), Dijkstra [1959] and Prim [1957]:

PRIM'S ALGORITHM

Input: A connected undirected graph G, weights $c : E(G) \to \mathbb{R}$.

Output: A spanning tree T of minimum weight.

① Choose $v \in V(G)$. Set $T := (\{v\}, \emptyset)$.

② **While** $V(T) \neq V(G)$ **do**:
Choose an edge $e \in \delta_G(V(T))$ of minimum weight. Set $T := T + e$.

Theorem 6.5. PRIM'S ALGORITHM *works correctly. Its running time is* $O(n^2)$.

Proof: The correctness follows from the fact that condition (c) of Theorem 6.2 is guaranteed.

To obtain the $O(n^2)$ running time, we maintain for each vertex $v \in V(G) \setminus V(T)$ the cheapest edge $e \in E(V(T), \{v\})$. Let us call these edges the candidates. The initialization of the candidates takes $O(m)$ time. Each selection of the cheapest edge among the candidates takes $O(n)$ time. The update of the candidates can be done by scanning the edges incident to the vertex which is added to $V(T)$ and thus also takes $O(n)$ time. Since the while-loop of ② has $n - 1$ iterations, the $O(n^2)$ bound is proved. □

The running time can be improved by efficient data structures. Denote $l_{T,v} := \min\{c(e) : e \in E(V(T), \{v\})\}$. We maintain the set $\{(v, l_{T,v}) : v \in V(G) \setminus V(T), l_{T,v} < \infty\}$ in a data structure, called priority queue or heap, that allows inserting an element, finding and deleting an element (v, l) with minimum l, and decreasing the so-called key l of an element (v, l). Then PRIM'S ALGORITHM can be written as follows:

① Choose $v \in V(G)$. Set $T := (\{v\}, \emptyset)$.
Let $l_w := \infty$ for $w \in V(G) \setminus \{v\}$.

② **While** $V(T) \neq V(G)$ **do**:
For $e = \{v, w\} \in E(\{v\}, V(G) \setminus V(T))$ **do**:
If $c(e) < l_w < \infty$ **then** set $l_w := c(e)$ and DECREASEKEY(w, l_w).
If $l_w = \infty$ **then** set $l_w := c(e)$ and INSERT(w, l_w).
$(v, l_v) :=$ DELETEMIN.
Let $e \in E(V(T), \{v\})$ with $c(e) = l_v$. Set $T := T + e$.

There are several possible ways to implement a heap. A very efficient way, the so-called Fibonacci heap, has been proposed by Fredman and Tarjan [1987]. Our presentation is based on Schrijver [2003]:

Theorem 6.6. *It is possible to maintain a data structure for a finite set (initially empty), where each element u is associated with a real number* $d(u)$*, called its key, and perform any sequence of*

- p INSERT*-operations (adding an element u with key* $d(u)$*);*
- n DELETEMIN*-operations (finding and deleting an element u with* $d(u)$ *minimum);*
- m DECREASEKEY*-operations (decreasing* $d(u)$ *to a specified value for an element u)*

in $O(m + p + n \log p)$ *time.*

Proof: The set, denoted by U, is stored in a Fibonacci heap, i.e. a branching (U, E) with a function $\varphi : U \to \{0, 1\}$ with the following properties:

(i) If $(u, v) \in E$ then $d(u) \leq d(v)$. (This is called the heap order.)
(ii) For each $u \in U$ the children of u can be numbered $1, \ldots, |\delta^+(u)|$ such that the i-th child v satisfies $|\delta^+(v)| + \varphi(v) \geq i - 1$.
(iii) If u and v are distinct roots ($\delta^-(u) = \delta^-(v) = \emptyset$), then $|\delta^+(u)| \neq |\delta^+(v)|$.

Condition (ii) implies:

(iv) If a vertex u has out-degree at least k, then at least $\sqrt{2}^k$ vertices are reachable from u.

We prove (iv) by induction on k, the case $k = 0$ being trivial. So let u be a vertex with $|\delta^+(u)| \geq k \geq 1$, and let v be a child of u with $|\delta^+(v)| \geq k - 2$ (v exists due to (ii)). We apply the induction hypothesis to v in (U, E) and to u in $(U, E \setminus \{(u, v)\})$ and conclude that at least $\sqrt{2}^{k-2}$ and $\sqrt{2}^{k-1}$ vertices are reachable. (iv) follows from observing that $\sqrt{2}^k \leq \sqrt{2}^{k-2} + \sqrt{2}^{k-1}$.

In particular, (iv) implies that $|\delta^+(u)| \leq 2 \log |U|$ for all $u \in U$. Thus, using (iii), we can store the roots of (U, E) by a function $b : \{0, 1, \ldots, \lfloor 2 \log |U| \rfloor\} \to U$ with $b(|\delta^+(u)|) = u$ for each root u.

In addition to this, we keep track of a doubly-linked list of children (in arbitrary order), a pointer to the parent (if existent) and the out-degree of each vertex. We now show how the INSERT-, DELETEMIN- and DECREASEKEY-operations are implemented.

INSERT$(v, d(v))$ is implemented by setting $\varphi(v) := 0$ and applying

PLANT(v):

① Set $r := b(|\delta^+(v)|)$.
if r is a root with $r \neq v$ and $|\delta^+(r)| = |\delta^+(v)|$ **then**:
if $d(r) \leq d(v)$ **then** add (r, v) to E and PLANT(r).
if $d(v) < d(r)$ **then** add (v, r) to E and PLANT(v).
else set $b(|\delta^+(v)|) := v$.

As (U, E) is always a branching, the recursion terminates. Note also that (i), (ii) and (iii) are maintained.

DELETEMIN is implemented by scanning $b(i)$ for $i = 0, \ldots, \lfloor 2 \log |U| \rfloor$ in order to find an element u with $d(u)$ minimum, deleting u and its incident edges and successively applying PLANT(v) for each (former) child v of u.

DECREASEKEY$(v, (d(v))$ is a bit more complicated. Let P be the longest path in (U, E) ending in v such that each internal vertex u satisfies $\varphi(u) = 1$. We set $\varphi(u) := 1 - \varphi(u)$ for all $u \in V(P) \setminus \{v\}$, delete all edges of P from E and apply PLANT(z) for each deleted edge (y, z).

To see that this maintains (ii) we only have to consider the parent of the start vertex x of P, if existent. But then x is not a root, and thus $\varphi(x)$ changes from 0 to 1, making up for the lost child.

We finally estimate the running time. As φ increases at most m times (at most once in each DECREASEKEY), φ decreases at most m times. Thus the sum of the length of the paths P in all DECREASEKEY-operations is at most $m+m$. So at most $2m+2n\log p$ edges are deleted overall (as each DELETEMIN-operation may delete up to $2\log p$ edges). Thus at most $2m+2n\log p+p-1$ edges are inserted in total. This proves the overall $O(m+p+n\log p)$ running time. □

Corollary 6.7. PRIM'S ALGORITHM *implemented with Fibonacci heap solves the* MINIMUM SPANNING TREE PROBLEM *in* $O(m+n\log n)$ *time.*

Proof: We have at most $n-1$ INSERT-, $n-1$ DELETEMIN-, and m DECREASEKEY-operations. □

With a more sophisticated implementation, the running time can be improved to $O(m\log\beta(n,m))$, where $\beta(n,m)=\min\left\{i:\log^{(i)}n\le\frac{m}{n}\right\}$; see Fredman and Tarjan [1987], Gabow, Galil and Spencer [1989], and Gabow et al. [1986]. The fastest known deterministic algorithm is due to Chazelle [2000] and has a running time of $O(m\alpha(m,n))$, where α is the functional inverse of Ackermann's function.

On a different computational model Fredman and Willard [1994] achieved linear running time. Moreover, there is a randomized algorithm which finds a minimum weight spanning tree and has linear expected running time (Karger, Klein and Tarjan [1995]; such an algorithm which always finds an optimum solution is called a Las Vegas algorithm). This algorithm uses a (deterministic) procedure for testing whether a given spanning tree is optimum; a linear-time algorithm for this problem has been found by Dixon, Rauch and Tarjan [1992]; see also King [1995].

The MINIMUM SPANNING TREE PROBLEM for planar graphs can be solved (deterministically) in linear time (Cheriton and Tarjan [1976]). The problem of finding a minimum spanning tree for a set of n points in the plane can be solved in $O(n\log n)$ time (Exercise 9). PRIM'S ALGORITHM can be quite efficient for such instances since one can use suitable data structures for finding nearest neighbours in the plane effectively.

6.2 Minimum Weight Arborescences

Natural directed generalizations of the MAXIMUM WEIGHT FOREST PROBLEM and the MINIMUM SPANNING TREE PROBLEM read as follows:

MAXIMUM WEIGHT BRANCHING PROBLEM	
Instance:	A digraph G, weights $c: E(G)\to\mathbb{R}$.
Task:	Find a maximum weight branching in G.

MINIMUM WEIGHT ARBORESCENCE PROBLEM	
Instance:	A digraph G, weights $c : E(G) \to \mathbb{R}$.
Task:	Find a minimum weight spanning arborescence in G or decide that none exists.

Sometimes we want to specify the root in advance:

MINIMUM WEIGHT ROOTED ARBORESCENCE PROBLEM	
Instance:	A digraph G, a vertex $r \in V(G)$, weights $c : E(G) \to \mathbb{R}$.
Task:	Find a minimum weight spanning arborescence rooted at r in G or decide that none exists.

As for the undirected case, these three problems are equivalent:

Proposition 6.8. *The* MAXIMUM WEIGHT BRANCHING PROBLEM*, the* MINIMUM WEIGHT ARBORESCENCE PROBLEM *and the* MINIMUM WEIGHT ROOTED ARBORESCENCE PROBLEM *are all equivalent.*

Proof: Given an instance (G, c) of the MINIMUM WEIGHT ARBORESCENCE PROBLEM, let $c'(e) := K - c(e)$ for all $e \in E(G)$, where $K = 2\sum_{e \in E(G)} |c(e)|$. Then the instance (G, c') of the MAXIMUM WEIGHT BRANCHING PROBLEM is equivalent, because for any two branchings B, B' with $|E(B)| > |E(B')|$ we have $c'(B) > c'(B')$ (and branchings with $n - 1$ edges are exactly the spanning arborescences).

Given an instance (G, c) of the MAXIMUM WEIGHT BRANCHING PROBLEM, let $G' := (V(G) \dot\cup \{r\}, E(G) \cup \{(r, v) : v \in V(G)\})$. Let $c'(e) := -c(e)$ for $e \in E(G)$ and $c(e) := 0$ for $e \in E(G') \setminus E(G)$. Then the instance (G', r, c') of the MINIMUM WEIGHT ROOTED ARBORESCENCE PROBLEM is equivalent.

Finally, given an instance (G, r, c) of the MINIMUM WEIGHT ROOTED ARBORESCENCE PROBLEM, let $G' := (V(G) \dot\cup \{s\}, E(G) \cup \{(s, r)\})$ and $c((s, r)) := 0$. Then the instance (G', c) of the MINIMUM WEIGHT ARBORESCENCE PROBLEM is equivalent. □

In the rest of this section we shall deal with the MAXIMUM WEIGHT BRANCHING PROBLEM only. This problem is not as easy as its undirected version, the MAXIMUM WEIGHT FOREST PROBLEM. For example any maximal forest is maximum, but the bold edges in Figure 6.1 form a maximal branching which is not maximum.

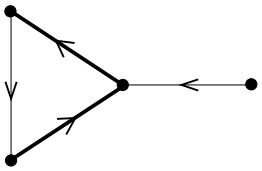

Fig. 6.1.

Recall that a branching is a graph B with $|\delta_B^-(x)| \leq 1$ for all $x \in V(B)$, such that the underlying undirected graph is a forest. Equivalently, a branching is an acyclic digraph B with $|\delta_B^-(x)| \leq 1$ for all $x \in V(B)$; see Theorem 2.5(g):

Proposition 6.9. *Let B be a digraph with $|\delta_B^-(x)| \leq 1$ for all $x \in V(B)$. Then B contains a circuit if and only if the underlying undirected graph contains a circuit.* □

Now let G be a digraph and $c : E(G) \to \mathbb{R}_+$. We can ignore negative weights since such edges will never appear in an optimum branching. A first idea towards an algorithm could be to take the best entering edge for each vertex. Of course the resulting graph may contain circuits. Since a branching cannot contain circuits, we must delete at least one edge of each circuit. The following lemma says that one is enough.

Lemma 6.10. (Karp [1972]) *Let B_0 be a maximum weight subgraph of G with $|\delta_{B_0}^-(v)| \leq 1$ for all $v \in V(B_0)$. Then there exists an optimum branching B of G such that for each circuit C in B_0, $|E(C) \setminus E(B)| = 1$.*

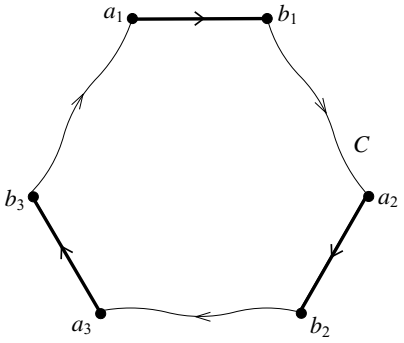

Fig. 6.2.

Proof: Let B be an optimum branching of G containing as many edges of B_0 as possible. Let C be some circuit in B_0. Let $E(C) \setminus E(B) = \{(a_1, b_1), \ldots, (a_k, b_k)\}$; suppose that $k \geq 2$ and $a_1, b_1, a_2, b_2, a_3, \ldots, b_k$ lie in this order on C (see Figure 6.2).

We claim that B contains a b_i-b_{i-1}-path for each $i = 1, \ldots, k$ ($b_0 := b_k$). This, however, is a contradiction because these paths form a closed edge progression in B, and a branching cannot have a closed edge progression.

Let $i \in \{1, \ldots, k\}$. It remains to show that B contains a b_i-b_{i-1}-path. Consider B' with $V(B') = V(G)$ and $E(B') := \{(x, y) \in E(B) : y \neq b_i\} \cup \{(a_i, b_i)\}$.

B' cannot be a branching since it would be optimum and contain more edges of B_0 than B. So (by Proposition 6.9) B' contains a circuit, i.e. B contains a

b_i-a_i-path P. Since $k \geq 2$, P is not completely on C, so let e be the last edge of P not belonging to C. Obviously $e = (x, b_{i-1})$ for some x, so P (and thus B) contains a b_i-b_{i-1}-path. □

The main idea of Edmonds' [1967] algorithm is to find first B_0 as above, and then contract every circuit of B_0 in G. If we choose the weights of the resulting graph G_1 correctly, any optimum branching in G_1 will correspond to an optimum branching in G.

EDMONDS' BRANCHING ALGORITHM

Input: A digraph G, weights $c : E(G) \to \mathbb{R}_+$.

Output: A maximum weight branching B of G.

① Set $i := 0$, $G_0 := G$, and $c_0 := c$.

② Let B_i be a maximum weight subgraph of G_i with $|\delta^-_{B_i}(v)| \leq 1$ for all $v \in V(B_i)$.

③ **If** B_i contains no circuit **then** set $B := B_i$ and **go to** ⑤.

④ Construct (G_{i+1}, c_{i+1}) from (G_i, c_i) by doing the following for each circuit C of B_i:
Contract C to a single vertex v_C in G_{i+1}
For each edge $e = (z, y) \in E(G_i)$ with $z \notin V(C)$, $y \in V(C)$ **do**:
Set $c_{i+1}(e') := c_i(e) - c_i(\alpha(e, C)) + c_i(e_C)$ and $\Phi(e') := e$,
where $e' := (z, v_C)$, $\alpha(e, C) = (x, y) \in E(C)$,
and e_C is some cheapest edge of C.
Set $i := i + 1$ and **go to** ②.

⑤ **If** $i = 0$ **then stop**.

⑥ **For** each circuit C of B_{i-1} **do**:
If there is an edge $e' = (z, v_C) \in E(B)$
then set $E(B) := (E(B) \setminus \{e'\}) \cup \Phi(e') \cup (E(C) \setminus \{\alpha(\Phi(e'), C)\})$
else set $E(B) := E(B) \cup (E(C) \setminus \{e_C\})$.
Set $V(B) := V(G_{i-1})$, $i := i - 1$ and **go to** ⑤.

This algorithm was also discovered independently by Chu and Liu [1965] and Bock [1971].

Theorem 6.11. (Edmonds [1967]) EDMONDS' BRANCHING ALGORITHM *works correctly.*

Proof: We show that each time just before the execution of ⑤, B is an optimum branching of G_i. This is trivial for the first time we reach ⑤. So we have to show that ⑥ transforms an optimum branching B of G_i into an optimum branching B' of G_{i-1}.

Let B^*_{i-1} be any branching of G_{i-1} such that $|E(C) \setminus E(B^*_{i-1})| = 1$ for each circuit C of B_{i-1}. Let B^*_i result from B^*_{i-1} by contracting the circuits of B_{i-1}. B^*_i

is a branching of G_i. Furthermore we have

$$c_{i-1}(B^*_{i-1}) \;=\; c_i(B^*_i) + \sum_{C\text{: circuit of } B_{i-1}} (c_{i-1}(C) - c_{i-1}(e_C)).$$

By the induction hypothesis, B is an optimum branching of G_i, so we have $c_i(B) \geq c_i(B^*_i)$. We conclude that

$$\begin{aligned} c_{i-1}(B^*_{i-1}) &\leq c_i(B) + \sum_{C\text{: circuit of } B_{i-1}} (c_{i-1}(C) - c_{i-1}(e_C)) \\ &= c_{i-1}(B'). \end{aligned}$$

This, together with Lemma 6.10, implies that B' is an optimum branching of G_{i-1}. □

This proof is due to Karp [1972]. Edmonds' original proof was based on a linear programming formulation (see Corollary 6.14). The running time of Edmonds' Branching Algorithm is easily seen to be $O(mn)$, where $m = |E(G)|$ and $n = |V(G)|$: there are at most n iterations (i.e. $i \leq n$ at any stage of the algorithm), and each iteration can be implemented in $O(m)$ time.

The best known bound has been obtained by Gabow et al. [1986] using a Fibonacci heap: their branching algorithm runs in $O(m + n \log n)$ time.

6.3 Polyhedral Descriptions

A polyhedral description of the Minimum Spanning Tree Problem is as follows:

Theorem 6.12. (Edmonds [1970]) *Given a connected undirected graph G, $n := |V(G)|$, the polytope $P :=$*

$$\left\{ x \in [0,1]^{E(G)} : \sum_{e \in E(G)} x_e = n-1, \sum_{e \in E(G[X])} x_e \leq |X| - 1 \text{ for } \emptyset \neq X \subset V(G) \right\}$$

is integral. Its vertices are exactly the incidence vectors of spanning trees of G. (P is called the **spanning tree polytope** *of G.)*

Proof: Let T be a spanning tree of G, and let x be the incidence vector of $E(T)$. Obviously (by Theorem 2.4), $x \in P$. Furthermore, since $x \in \{0,1\}^{E(G)}$, it must be a vertex of P.

On the other hand let x be an integral vertex of P. Then x is the incidence vector of the edge set of some subgraph H with $n-1$ edges and no circuit. Again by Theorem 2.4 this implies that H is a spanning tree.

So it suffices to show that P is integral (recall Theorem 5.12). Let $c : E(G) \to \mathbb{R}$, and let T be the tree produced by Kruskal's Algorithm when applied to (G, c) (ties are broken arbitrarily when sorting the edges). Denote

$E(T) = \{f_1, \ldots, f_{n-1}\}$, where the f_i were taken in this order by the algorithm. In particular, $c(f_1) \leq \cdots \leq c(f_{n-1})$. Let $X_k \subseteq V(G)$ be the connected component of $(V(G), \{f_1, \ldots, f_k\})$ containing f_k $(k = 1, \ldots, n-1)$.

Let x^* be the incidence vector of $E(T)$. We show that x^* is an optimum solution to the LP

$$\begin{array}{lll} \min & \displaystyle\sum_{e \in E(G)} c(e)x_e & \\ \text{s.t.} & \displaystyle\sum_{e \in E(G)} x_e = n-1 & \\ & \displaystyle\sum_{e \in E(G[X])} x_e \leq |X|-1 & (\emptyset \neq X \subset V(G)) \\ & x_e \geq 0 & (e \in E(G)). \end{array}$$

We introduce a dual variable z_X for each $\emptyset \neq X \subset V(G)$ and one additional dual variable $z_{V(G)}$ for the equality constraint. Then the dual LP is

$$\begin{array}{lll} \max & -\displaystyle\sum_{\emptyset \neq X \subseteq V(G)} (|X|-1)z_X & \\ \text{s.t.} & -\displaystyle\sum_{e \subseteq X \subseteq V(G)} z_X \leq c(e) & (e \in E(G)) \\ & z_X \geq 0 & (\emptyset \neq X \subset V(G)). \end{array}$$

Note that the dual variable $z_{V(G)}$ is not forced to be nonnegative. For $k = 1, \ldots, n-2$ let $z^*_{X_k} := c(f_l) - c(f_k)$, where l is the first index greater than k for which $f_l \cap X_k \neq \emptyset$. Let $z^*_{V(G)} := -c(f_{n-1})$, and let $z^*_X := 0$ for all $X \notin \{X_1, \ldots, X_{n-1}\}$.

For each $e = \{v, w\}$ we have that

$$-\sum_{e \subseteq X \subseteq V(G)} z^*_X = c(f_i),$$

where i is the smallest index such that $v, w \in X_i$. Moreover $c(f_i) \leq c(e)$ since v and w are in different connected components of $(V(G), \{f_1, \ldots, f_{i-1}\})$. Hence z^* is a feasible dual solution.

Moreover $x^*_e > 0$, i.e. $e \in E(T)$, implies

$$-\sum_{e \subseteq X \subseteq V(G)} z^*_X = c(e),$$

i.e. the corresponding dual constraint is satisfied with equality. Finally, $z^*_X > 0$ implies that $T[X]$ is connected, so the corresponding primal constraint is satisfied with equality. In other words, the primal and dual complementary slackness conditions are satisfied, thus (by Corollary 3.18) x^* and z^* are optimum solutions for the primal and dual LP, respectively. □

Indeed, we have proved that the inequality system in Theorem 6.12 is TDI. We remark that the above is also an alternative proof of the correctness of KRUSKAL'S ALGORITHM (Theorem 6.3). Another description of the spanning tree polytope is the subject of Exercise 13.

If we replace the constraint $\sum_{e\in E(G)} x_e = n-1$ by $\sum_{e\in E(G)} x_e \leq n-1$, we obtain the convex hull of the incidence vectors of all forests in G (Exercise 14). A generalization of these results is Edmonds' characterization of the matroid polytope (Theorem 13.21).

We now turn to a polyhedral description of the MINIMUM WEIGHT ROOTED ARBORESCENCE PROBLEM. First we prove a classical result of Fulkerson. Recall that an r-cut is a set of edges $\delta^+(S)$ for some $S \subset V(G)$ with $r \in S$.

Theorem 6.13. (Fulkerson [1974]) *Let G be a digraph with weights $c : E(G) \to \mathbb{Z}_+$, and $r \in V(G)$ such that G contains a spanning arborescence rooted at r. Then the minimum weight of a spanning arborescence rooted at r equals the maximum number t of r-cuts $C_1, \ldots, C_t$ (repetitions allowed) such that no edge e is contained in more than $c(e)$ of these cuts.*

Proof: Let A be the matrix whose columns are indexed by the edges and whose rows are all incidence vectors of r-cuts. Consider the LP

$$\min\{cx : Ax \geq \mathbb{1},\ x \geq 0\},$$

and its dual

$$\max\{\mathbb{1}y : yA \leq c,\ y \geq 0\}.$$

Then (by part (e) of Theorem 2.5) we have to show that for any nonnegative integral c, both the primal and dual LP have integral optimum solutions. By Corollary 5.14 it suffices to show that the system $Ax \geq \mathbb{1}$, $x \geq 0$ is TDI. We use Lemma 5.22.

Since the dual LP is feasible if and only if c is nonnegative, let $c : E(G) \to \mathbb{Z}_+$. Let y be an optimum solution of $\max\{\mathbb{1}y : yA \leq c, y \geq 0\}$ for which

$$\sum_{\emptyset \neq X \subseteq V(G)\setminus\{r\}} y_{\delta^-(X)}|X|^2 \tag{6.1}$$

is as large as possible. We claim that $\mathcal{F} := \{X : y_{\delta^-(X)} > 0\}$ is laminar. To see this, suppose $X, Y \in \mathcal{F}$ with $X \cap Y \neq \emptyset$, $X \setminus Y \neq \emptyset$ and $Y \setminus X \neq \emptyset$ (Figure 6.3). Let $\epsilon := \min\{y_{\delta^-(X)}, y_{\delta^-(Y)}\}$. Set $y'_{\delta^-(X)} := y_{\delta^-(X)} - \epsilon$, $y'_{\delta^-(Y)} := y_{\delta^-(Y)} - \epsilon$, $y'_{\delta^-(X\cap Y)} := y_{\delta^-(X\cap Y)} + \epsilon$, $y'_{\delta^-(X\cup Y)} := y_{\delta^-(X\cup Y)} + \epsilon$, and $y'(S) := y(S)$ for all other r-cuts S. Observe that $y'A \leq yA$, so y' is a feasible dual solution. Since $\mathbb{1}y = \mathbb{1}y'$, it is also optimum and contradicts the choice of y, because (6.1) is larger for y'. (For any numbers $a > b \geq c > d > 0$ with $a + d = b + c$ we have $a^2 + d^2 > b^2 + c^2$.)

Now let A' be the submatrix of A consisting of the rows corresponding to the elements of $\mathcal{F}$. A' is the one-way cut-incidence matrix of a laminar family (to be precise, we must consider the graph resulting from G by reversing each edge). So by Theorem 5.27 A' is totally unimodular, as required. □

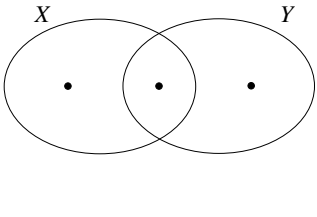

Fig. 6.3.

The above proof also yields the promised polyhedral description:

Corollary 6.14. (Edmonds [1967]) *Let G be a digraph with weights $c : E(G) \to \mathbb{R}_+$, and $r \in V(G)$ such that G contains a spanning arborescence rooted at r. Then the LP*

$$\min \left\{ cx : x \geq 0, \sum_{e \in \delta^+(X)} x_e \geq 1 \text{ for all } X \subset V(G) \text{ with } r \in X \right\}$$

has an integral optimum solution (which is the incidence vector of a minimum weight spanning arborescence rooted at r, plus possibly some edges of zero weight). □

For a description of the convex hull of the incidence vectors of all branchings or spanning arborescences rooted at r, see Exercises 15 and 16.

6.4 Packing Spanning Trees and Arborescences

If we are looking for more than one spanning tree or arborescence, classical theorems of Tutte, Nash-Williams and Edmonds are of help. We first give a proof of Tutte's Theorem on packing spanning trees which is essentially due to Mader (see Diestel [1997]) and which uses the following lemma:

Lemma 6.15. *Let G be an undirected graph, and let $F = (F_1, \ldots, F_k)$ be a k-tuple of edge-disjoint forests in G such that $|E(F)|$ is maximum, where $E(F) := \bigcup_{i=1}^{k} E(F_i)$. Let $e \in E(G) \setminus E(F)$. Then there exists a set $X \subseteq V(G)$ with $e \subseteq X$ such that $F_i[X]$ is connected for each $i \in \{1, \ldots, k\}$.*

Proof: For two k-tuples $F' = (F'_1, \ldots, F'_k)$ and $F'' = (F''_1, \ldots, F''_k)$ we say that F'' arises from F' by exchanging e' for e'' if $F''_j = (F'_j \setminus e') \cup e''$ for some j and $F''_i = F'_i$ for all $i \neq j$. Let $\mathcal{F}$ be the set of all k-tuples of edge-disjoint forests arising from F by a sequence of such exchanges. Let $\overline{E} := E(G) \setminus \big(\bigcap_{F' \in \mathcal{F}} E(F')\big)$ and $\overline{G} := (V(G), \overline{E})$. We have $F \in \mathcal{F}$ and thus $e \in \overline{E}$. Let X be the vertex set

of the connected component of $\overline{G}$ containing e. We shall prove that $F_i[X]$ is connected for each i.

Claim: For any $F' = (F'_1, \ldots, F'_k) \in \mathcal{F}$ and any $\bar{e} = \{v, w\} \in E(\overline{G}[X]) \setminus E(F')$ there exists a v-w-path in $F'_i[X]$ for all $i \in \{1, \ldots, k\}$.

To prove this, let $i \in \{1, \ldots, k\}$ be fixed. Since $F' \in \mathcal{F}$ and $|E(F')| = |E(F)|$ is maximum, $F'_i + \bar{e}$ contains a circuit C. Now for all $e' \in E(C) \setminus \{\bar{e}\}$ we have $F'_{e'} \in \mathcal{F}$, where $F'_{e'}$ arises from F' by exchanging e' for $\bar{e}$. This shows that $E(C) \subseteq \overline{E}$, and so $C - \bar{e}$ is a v-w-path in $F'_i[X]$. This proves the claim.

Since $\overline{G}[X]$ is connected, it suffices to prove that for each $\bar{e} = \{v, w\} \in E(\overline{G}[X])$ and each i there is a v-w-path in $F_i[X]$.

So let $\bar{e} = \{v, w\} \in E(\overline{G}[X])$. Since $\bar{e} \in \overline{E}$, there is some $F' = (F'_1, \ldots, F'_k) \in \mathcal{F}$ with $\bar{e} \notin E(F')$. By the claim there is a v-w-path in $F'_i[X]$ for each i.

Now there is a sequence $F = F^{(0)}, F^{(1)} \ldots, F^{(s)} = F'$ of elements of $\mathcal{F}$ such that $F^{(r+1)}$ arises from $F^{(r)}$ by exchanging one edge ($r = 0, \ldots, s-1$). It suffices to show that the existence of a v-w-path in $F_i^{(r+1)}[X]$ implies the existence of a v-w-path in $F_i^{(r)}[X]$ ($r = 0, \ldots, s-1$).

To see this, suppose that $F_i^{(r+1)}[X]$ arises from $F_i^{(r)}[X]$ by exchanging e_r for e_{r+1}, and let P be the v-w-path in $F_i^{(r+1)}[X]$. If P does not contain $e_{r+1} = \{x, y\}$, it is also a path in $F_i^{(r)}[X]$. Otherwise $e_{r+1} \in E(\overline{G}[X])$, and we consider the x-y-path Q in $F_i^{(r)}[X]$ which exists by the claim. Since $(E(P) \setminus \{e_{r+1}\}) \cup Q$ contains a v-w-path in $F_i^{(r)}[X]$, the proof is complete. □

Now we can prove Tutte's theorem on disjoint spanning trees. A **multicut** in an undirected graph G is a set of edges $\delta(X_1, \ldots, X_p) := \delta(X_1) \cup \cdots \cup \delta(X_p)$ for some partition $V(G) = X_1 \dot\cup X_2 \dot\cup \cdots \dot\cup X_p$ of the vertex set into nonempty subsets. For $p = 3$ we also speak of 3-cuts. Observe that cuts are multicuts with $p = 2$.

Theorem 6.16. (Tutte [1961], Nash-Williams [1961]) *An undirected graph G contains k edge-disjoint spanning trees if and only if*

$$|\delta(X_1, \ldots, X_p)| \;\geq\; k(p-1)$$

for every multicut $\delta(X_1, \ldots, X_p)$.

Proof: To prove necessity, let $T_1, \ldots, T_k$ be edge-disjoint spanning trees in G, and let $\delta(X_1, \ldots, X_p)$ be a multicut. Contracting each of the vertex subsets $X_1, \ldots, X_p$ yields a graph G' whose vertices are $X_1, \ldots, X_p$ and whose edges correspond to the edges of the multicut. $T_1, \ldots, T_k$ correspond to edge-disjoint connected subgraphs $T'_1, \ldots, T'_k$ in G'. Each of the $T'_1, \ldots, T'_k$ has at least $p-1$ edges, so G' (and thus the multicut) has at least $k(p-1)$ edges.

To prove sufficiency we use induction on $|V(G)|$. For $n := |V(G)| \leq 2$ the statement is true. Now assume $n > 2$, and suppose that $|\delta(X_1, \ldots, X_p)| \geq k(p-1)$ for every multicut $\delta(X_1, \ldots, X_p)$. In particular (consider the partition into singletons) G has at least $k(n-1)$ edges. Moreover, the condition is preserved

when contracting vertex sets, so by the induction hypothesis G/X contains k edge-disjoint spanning trees for each $X \subset V(G)$ with $|X| \geq 2$.

Let $F = (F_1, \ldots, F_k)$ be a k-tuple of edge-disjoint forests in G such that $|E(F)| = \left|\bigcup_{i=1}^{k} E(F_i)\right|$ is maximum. We claim that each F_i is a spanning tree. Otherwise $E(F) < k(n-1)$, so there is an edge $e \in E(G) \setminus E(F)$. By Lemma 6.15 there is an $X \subseteq V(G)$ with $e \subseteq X$ such that $F_i[X]$ is connected for each i. Since $|X| \geq 2$, G/X contains k edge-disjoint spanning trees $F_1', \ldots, F_k'$. Now F_i' together with $F_i[X]$ forms a spanning tree in G for each i, and all these k spanning trees are edge-disjoint. □

We now turn to the corresponding problem in digraphs, packing spanning arborescences:

Theorem 6.17. (Edmonds [1973]) *Let G be a digraph and $r \in V(G)$. Then the maximum number of edge-disjoint spanning arborescences rooted at r equals the minimum cardinality of an r-cut.*

Proof: Let k be the minimum cardinality of an r-cut. Obviously there are at most k edge-disjoint spanning arborescences. We prove the existence of k edge-disjoint spanning arborescences by induction on k. The case $k = 0$ is trivial.

If we can find one spanning arborescence A rooted at r such that

$$\min_{r \in S \subset V(G)} \left|\delta_G^+(S) \setminus E(A)\right| \geq k - 1, \tag{6.2}$$

then we are done by induction. Suppose we have already found some arborescence A rooted at r (but not necessarily spanning) such that (6.2) holds. Let $R \subseteq V(G)$ be the set of vertices covered by A. Initially, $R = \{r\}$; if $R = V(G)$, we are done.

If $R \neq V(G)$, we call a set $X \subseteq V(G)$ critical if

(a) $r \in X$;
(b) $X \cup R \neq V(G)$;
(c) $|\delta_G^+(X) \setminus E(A)| = k - 1$.

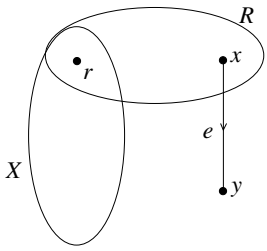

Fig. 6.4.

If there is no critical vertex set, we can augment A by any edge leaving R. Otherwise let X be a maximal critical set, and let $e = (x, y)$ be an edge such that

$x \in R \setminus X$ and $y \in V(G) \setminus (R \cup X)$ (see Figure 6.4). Such an edge must exist because

$$|\delta^+_{G-E(A)}(R \cup X)| \;=\; |\delta^+_G(R \cup X)| \;\geq\; k \;>\; k-1 \;=\; |\delta^+_{G-E(A)}(X)|.$$

We now add e to A. Obviously $A+e$ is an arborescence rooted at r. We have to show that (6.2) continues to hold.

Suppose there is some Y such that $r \in Y \subset V(G)$ and $|\delta^+_G(Y) \setminus E(A+e)| < k-1$. Then $x \in Y$, $y \notin Y$, and $|\delta^+_G(Y) \setminus E(A)| = k-1$. Now Lemma 2.1(a) implies

$$\begin{aligned} k-1+k-1 \;&=\; |\delta^+_{G-E(A)}(X)| + |\delta^+_{G-E(A)}(Y)| \\ &\geq\; |\delta^+_{G-E(A)}(X \cup Y)| + |\delta^+_{G-E(A)}(X \cap Y)| \\ &\geq\; k-1+k-1\,, \end{aligned}$$

because $r \in X \cap Y$ and $y \in V(G) \setminus (X \cup Y)$. So equality must hold throughout, in particular $|\delta^+_{G-E(A)}(X \cup Y)| = k-1$. Since $y \in V(G) \setminus (X \cup Y \cup R)$ we conclude that $X \cup Y$ is critical. But since $x \in Y \setminus X$, this contradicts the maximality of X. □

This proof is due to Lovász [1976]. A generalization of Theorems 6.16 and 6.17 was found by Frank [1978]. A good characterization of the problem of packing spanning arborescences with arbitrary roots is given by the following theorem, which we cite without proof:

Theorem 6.18. (Frank [1979]) *A digraph G contains k edge-disjoint spanning arborescences if and only if*

$$\sum_{i=1}^{p} |\delta^-(X_i)| \;\geq\; k(p-1)$$

for every collection of pairwise disjoint nonempty subsets $X_1, \ldots, X_p \subseteq V(G)$.

Another question is how many forests are needed to cover a graph. This is answered by the following theorem:

Theorem 6.19. (Nash-Williams [1964]) *The edge set of an undirected graph G is the union of k forests if and only if $|E(G[X])| \leq k(|X|-1)$ for all $\emptyset \neq X \subseteq V(G)$.*

Proof: The necessity is clear since no forest can contain more than $|X|-1$ edges within a vertex set X. To prove the sufficiency, assume that $|E(G[X])| \leq k(|X|-1)$ for all $\emptyset \neq X \subseteq V(G)$, and let $F = (F_1, \ldots, F_k)$ be a k-tuple of disjoint forests in G such that $|E(F)| = \left|\bigcup_{i=1}^k E(F_i)\right|$ is maximum. We claim that $E(F) = E(G)$. To see this, suppose there is an edge $e \in E(G) \setminus E(F)$. By Lemma 6.15 there exists a set $X \subseteq V(G)$ with $e \subseteq X$ such that $F_i[X]$ is connected for each i. In particular,

$$|E(G[X])| \;\geq\; \left|\{e\} \,\dot\cup \bigcup_{i=1}^{k} E(F_i[X])\right| \;\geq\; 1 + k(|X| - 1),$$

contradicting the assumption. □

Exercise 21 gives a directed version. A generalization of Theorems 6.16 and 6.19 to matroids can be found in Exercise 18 of Chapter 13.

Exercises

1. Prove Cayley's theorem, stating that K_n has n^{n-2} spanning trees, by showing that the following defines a one-to-one correspondence between the spanning trees in K_n and the vectors in $\{1, \ldots, n\}^{n-2}$: For a tree T with $V(T) = \{1, \ldots, n\}$, $n \geq 3$, let v be the leaf with the smallest index and let a_1 be the neighbour of v. We recursively define $a(T) := (a_1, \ldots, a_{n-2})$, where $(a_2, \ldots, a_{n-2}) = a(T - v)$.
(Cayley [1889], Prüfer [1918])
2. Let (V, T_1) and (V, T_2) be two trees on the same vertex set V. Prove that for any edge $e \in T_1$ there is an edge $f \in T_2$ such that both $(V, (T_1 \setminus \{e\}) \cup \{f\})$ and $(V, (T_2 \setminus \{f\}) \cup \{e\})$ are trees.
3. Given an undirected graph G with weights $c : E(G) \to \mathbb{R}$ and a vertex $v \in V(G)$, we ask for a minimum weight spanning tree in G where v is not a leaf. Can you solve this problem in polynomial time?
4. We want to determine the set of edges e in an undirected graph G with weights $c : E(G) \to \mathbb{R}$ for which there exists a minimum weight spanning tree in G containing e (in other words, we are looking for the union of all minimum weight spanning trees in G). Show how this problem can be solved in $O(mn)$ time.
5. Given an undirected graph G with arbitrary weights $c : E(G) \to \mathbb{R}$, we ask for a minimum weight connected spanning subgraph. Can you solve this problem efficiently?
6. Consider the following algorithm (sometimes called Worst-Out-Greedy Algorithm, see Section 13.4). Examine the edges in order of non-increasing weights. Delete an edge unless it is a bridge. Does this algorithm solve the Minimum Spanning Tree Problem?
7. Consider the following "colouring" algorithm. Initially all edges are uncoloured. Then apply the following rules in arbitrary order until all edges are coloured:
Blue rule: Select a cut containing no blue edge. Among the uncoloured edges in the cut, select one of minimum cost and colour it blue.
Red rule: Select a circuit containing no red edge. Among the uncoloured edges in the circuit, select one of maximum cost and colour it red.
Show that one of the rules is always applicable as long as there are uncoloured edges left. Moreover, show that the algorithm maintains the "colour invariant":

there always exists an optimum spanning tree containing all blue edges but no red edge. (So the algorithm solves the MINIMUM SPANNING TREE PROBLEM optimally.) Observe that KRUSKAL'S ALGORITHM and PRIM'S ALGORITHM are special cases.
(Tarjan [1983])

8. Suppose we wish to find a spanning tree T in an undirected graph such that the maximum weight of an edge in T is as small as possible. How can this be done?
9. For a finite set $V \subset \mathbb{R}^2$, the Voronoï diagram consists of the regions

$$P_v := \left\{ x \in \mathbb{R}^2 : ||x - v||_2 = \min_{w \in V} ||x - w||_2 \right\}$$

for $v \in V$. The Delaunay triangulation of V is the graph

$$(V, \{\{v, w\} \subseteq V,\ v \neq w,\ |P_v \cap P_w| > 1\}) .$$

A minimum spanning tree for V is a tree T with $V(T) = V$ whose length $\sum_{\{v,w\} \in E(T)} ||v - w||_2$ is minimum. Prove that every minimum spanning tree is a subgraph of the Delaunay triangulation.
Note: Using the fact that the Delaunay triangulation can be computed in $O(n \log n)$ time (where $n = |V|$; see e.g. Fortune [1987], Knuth [1992]), this implies an $O(n \log n)$ algorithm for the MINIMUM SPANNING TREE PROBLEM for point sets in the plane.
(Shamos and Hoey [1975]); see also (Zhou, Shenoy and Nicholls [2002])

10. Can you decide in linear time whether a graph contains a spanning arborescence?
Hint: To find a possible root, start at an arbitrary vertex and traverse edges backwards as long as possible. When encountering a circuit, contract it.
11. The MINIMUM WEIGHT ROOTED ARBORESCENCE PROBLEM can be reduced to the MAXIMUM WEIGHT BRANCHING PROBLEM by Proposition 6.8. However, it can also be solved directly by a modified version of EDMONDS' BRANCHING ALGORITHM. Show how.

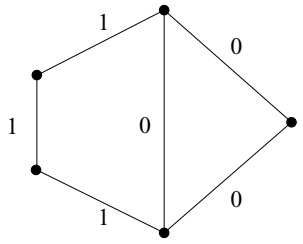

Fig. 6.5.

12. Prove that the spanning tree polytope of an undirected graph G (see Theorem 6.12) with $n := |V(G)|$ is in general a proper subset of the polytope

$$\left\{ x \in [0,1]^{E(G)} : \sum_{e \in E(G)} x_e = n-1, \sum_{e \in \delta(X)} x_e \geq 1 \text{ for } \emptyset \subset X \subset V(G) \right\}.$$

Hint: To prove that this polytope is not integral, consider the graph shown in Figure 6.5 (the numbers are edge weights).
(Magnanti and Wolsey [1995])

∗ 13. In Exercise 12 we saw that cut constraints do not suffice to describe the spanning tree polytope. However, if we consider multicuts instead, we obtain a complete description: Prove that the spanning tree polytope of an undirected graph G with $n := |V(G)|$ consists of all vectors $x \in [0,1]^{E(G)}$ with

$$\sum_{e \in E(G)} x_e = n-1 \text{ and } \sum_{e \in C} x_e \geq k-1 \text{ for all multicuts } C = \delta(X_1, \ldots, X_k).$$

(Magnanti and Wolsey [1995])

14. Prove that the convex hull of the incidence vectors of all forests in an undirected graph G is the polytope

$$P := \left\{ x \in [0,1]^{E(G)} : \sum_{e \in E(G[X])} x_e \leq |X| - 1 \text{ for } \emptyset \neq X \subseteq V(G) \right\}.$$

Note: This statement implies Theorem 6.12 since $\sum_{e \in E(G[X])} x_e = |V(G)| - 1$ is a supporting hyperplane. Moreover, it is a special case of Theorem 13.21.

∗ 15. Prove that the convex hull of the incidence vectors of all branchings in a digraph G is the set of all vectors $x \in [0,1]^{E(G)}$ with

$$\sum_{e \in E(G[X])} x_e \leq |X| - 1 \text{ for } \emptyset \neq X \subseteq V(G) \text{ and } \sum_{e \in \delta^-(v)} x_e \leq 1 \text{ for } v \in V(G).$$

Note: This is a special case of Theorem 14.13.

∗ 16. Let G be a digraph and $r \in V(G)$. Prove that the polytopes

$$\left\{ x \in [0,1]^{E(G)} : x_e = 0 \ (e \in \delta^-(r)), \sum_{e \in \delta^-(v)} x_e = 1 \ (v \in V(G) \setminus \{r\}), \right.$$
$$\left. \sum_{e \in E(G[X])} x_e \leq |X| - 1 \text{ for } \emptyset \neq X \subseteq V(G) \right\}$$

and

$$\left\{ x \in [0,1]^{E(G)} \quad : \quad x_e = 0 \ (e \in \delta^-(r)),\ \sum_{e \in \delta^-(v)} x_e = 1 \ (v \in V(G) \setminus \{r\}),\right.$$
$$\left. \sum_{e \in \delta^+(X)} x_e \geq 1 \text{ for } r \in X \subset V(G) \right\}$$

are both equal to the convex hull of the incidence vectors of all spanning arborescences rooted at r.

17. Let G be a digraph and $r \in V(G)$. Prove that G is the disjoint union of k spanning arborescences rooted at r if and only if the underlying undirected graph is the disjoint union of k spanning trees and $|\delta^-(x)| = k$ for all $x \in V(G) \setminus \{r\}$.
(Edmonds)

18. Let G be a digraph and $r \in V(G)$. Suppose that G contains k edge-disjoint paths from r to every other vertex, but removing any edge destroys this property. Prove that every vertex of G except r has exactly k entering edges.
Hint: Use Theorem 6.17.

∗ 19. Prove the statement of Exercise 18 without using Theorem 6.17. Formulate and prove a vertex-disjoint version.
Hint: If a vertex v has more than k entering edges, take k edge-disjoint r-v-paths. Show that an edge entering v that is not used by these paths can be deleted.

20. Give a polynomial-time algorithm for finding a maximum set of edge-disjoint spanning arborescences (rooted at r) in a digraph G.
Note: The most efficient algorithm is due to Gabow [1995]; see also (Gabow and Manu [1998]).

21. Prove that the edges of a digraph G can be covered by k branchings if and only if the following two conditions hold:
(a) $|\delta^-(v)| \leq k$ for all $v \in V(G)$;
(b) $|E(G[X])| \leq k(|X| - 1)$ for all $X \subseteq V(G)$.
Hint: Use Theorem 6.17.
(Frank [1979])

References

General Literature:

Ahuja, R.K., Magnanti, T.L., and Orlin, J.B. [1993]: Network Flows. Prentice-Hall, Englewood Cliffs 1993, Chapter 13

Balakrishnan, V.K. [1995]: Network Optimization. Chapman and Hall, London 1995, Chapter 1

Cormen, T.H., Leiserson, C.E., and Rivest, R.L. [1990]: Introduction to Algorithms. MIT Press, Cambridge 1990, Chapter 24

Gondran, M., and Minoux, M. [1984]: Graphs and Algorithms. Wiley, Chichester 1984, Chapter 4

Magnanti, T.L., and Wolsey, L.A. [1995]: Optimal trees. In: Handbooks in Operations Research and Management Science; Volume 7: Network Models (M.O. Ball, T.L. Magnanti, C.L. Monma, G.L. Nemhauser, eds.), Elsevier, Amsterdam 1995, pp. 503–616

Schrijver, A. [2003]: Combinatorial Optimization: Polyhedra and Efficiency. Springer, Berlin 2003, Chapters 50–53

Tarjan, R.E. [1983]: Data Structures and Network Algorithms. SIAM, Philadelphia 1983, Chapter 6

Wu, B.Y., and Chao, K.-M. [2004]: Spanning Trees and Optimization Problems. Chapman & Hall/CRC, Boca Raton 2004

Cited References:

Bock, F.C. [1971]: An algorithm to construct a minimum directed spanning tree in a directed network. In: Avi-Itzak, B. (Ed.): Developments in Operations Research. Gordon and Breach, New York 1971, 29–44

Borůvka, O. [1926a]: O jistém problému minimálním. Práca Moravské Přírodovědecké Spolnečnosti 3 (1926), 37–58

Borůvka, O. [1926b]: Příspevěk k řešení otázky ekonomické stavby. Elektrovodních sítí. Elektrotechnicky Obzor 15 (1926), 153–154

Cayley, A. [1889]: A theorem on trees. Quarterly Journal on Mathematics 23 (1889), 376–378

Chazelle, B. [2000]: A minimum spanning tree algorithm with inverse-Ackermann type complexity. Journal of the ACM 47 (2000), 1028–1047

Cheriton, D., and Tarjan, R.E. [1976]: Finding minimum spanning trees. SIAM Journal on Computing 5 (1976), 724–742

Chu, Y., and Liu, T. [1965]: On the shortest arborescence of a directed graph. Scientia Sinica 4 (1965), 1396–1400; Mathematical Review 33, # 1245

Diestel, R. [1997]: Graph Theory. Springer, New York 1997

Dijkstra, E.W. [1959]: A note on two problems in connexion with graphs. Numerische Mathematik 1 (1959), 269–271

Dixon, B., Rauch, M., and Tarjan, R.E. [1992]: Verification and sensitivity analysis of minimum spanning trees in linear time. SIAM Journal on Computing 21 (1992), 1184–1192

Edmonds, J. [1967]: Optimum branchings. Journal of Research of the National Bureau of Standards B 71 (1967), 233–240

Edmonds, J. [1970]: Submodular functions, matroids and certain polyhedra. In: Combinatorial Structures and Their Applications; Proceedings of the Calgary International Conference on Combinatorial Structures and Their Applications 1969 (R. Guy, H. Hanani, N. Sauer, J. Schonheim, eds.), Gordon and Breach, New York 1970, pp. 69–87

Edmonds, J. [1973]: Edge-disjoint branchings. In: Combinatorial Algorithms (R. Rustin, ed.), Algorithmic Press, New York 1973, pp. 91–96

Fortune, S. [1987]: A sweepline algorithm for Voronoi diagrams. Algorithmica 2 (1987), 153–174

Frank, A. [1978]: On disjoint trees and arborescences. In: Algebraic Methods in Graph Theory; Colloquia Mathematica; Soc. J. Bolyai 25 (L. Lovász, V.T. Sós, eds.), North-Holland, Amsterdam 1978, pp. 159–169

Frank, A. [1979]: Covering branchings. Acta Scientiarum Mathematicarum (Szeged) 41 (1979), 77–82

Fredman, M.L., and Tarjan, R.E. [1987]: Fibonacci heaps and their uses in improved network optimization problems. Journal of the ACM 34 (1987), 596–615

Fredman, M.L., and Willard, D.E. [1994]: Trans-dichotomous algorithms for minimum spanning trees and shortest paths. Journal of Computer and System Sciences 48 (1994), 533–551

Fulkerson, D.R. [1974]: Packing rooted directed cuts in a weighted directed graph. Mathematical Programming 6 (1974), 1–13
Gabow, H.N. [1995]: A matroid approach to finding edge connectivity and packing arborescences. Journal of Computer and System Sciences 50 (1995), 259–273
Gabow, H.N., Galil, Z., and Spencer, T. [1989]: Efficient implementation of graph algorithms using contraction. Journal of the ACM 36 (1989), 540–572
Gabow, H.N., Galil, Z., Spencer, T., and Tarjan, R.E. [1986]: Efficient algorithms for finding minimum spanning trees in undirected and directed graphs. Combinatorica 6 (1986), 109–122
Gabow, H.N., and Manu, K.S. [1998]: Packing algorithms for arborescences (and spanning trees) in capacitated graphs. Mathematical Programming B 82 (1998), 83–109
Jarník, V. [1930]: O jistém problému minimálním. Práca Moravské Přírodovědecké Společnosti 6 (1930), 57–63
Karger, D., Klein, P.N., and Tarjan, R.E. [1995]: A randomized linear-time algorithm to find minimum spanning trees. Journal of the ACM 42 (1995), 321–328
Karp, R.M. [1972]: A simple derivation of Edmonds' algorithm for optimum branchings. Networks 1 (1972), 265–272
King, V. [1995]: A simpler minimum spanning tree verification algorithm. Algorithmica 18 (1997), 263–270
Knuth, D.E. [1992]: Axioms and hulls; LNCS 606. Springer, Berlin 1992
Korte, B., and Nešetřil, J. [2001]: Vojtěch Jarník's work in combinatorial optimization. Discrete Mathematics 235 (2001), 1–17
Kruskal, J.B. [1956]: On the shortest spanning subtree of a graph and the travelling salesman problem. Proceedings of the AMS 7 (1956), 48–50
Lovász, L. [1976]: On two minimax theorems in graph. Journal of Combinatorial Theory B 21 (1976), 96–103
Nash-Williams, C.S.J.A. [1961]: Edge-disjoint spanning trees of finite graphs. Journal of the London Mathematical Society 36 (1961), 445–450
Nash-Williams, C.S.J.A. [1964]: Decompositions of finite graphs into forests. Journal of the London Mathematical Society 39 (1964), 12
Nešetřil, J., Milková, E., and Nešetřilová, H. [2001]: Otakar Borůvka on minimum spanning tree problem. Translation of both the 1926 papers, comments, history. Discrete Mathematics 233 (2001), 3–36
Prim, R.C. [1957]: Shortest connection networks and some generalizations. Bell System Technical Journal 36 (1957), 1389–1401
Prüfer, H. [1918]: Neuer Beweis eines Satzes über Permutationen. Arch. Math. Phys. 27 (1918), 742–744
Shamos, M.I., and Hoey, D. [1975]: Closest-point problems. Proceedings of the 16th Annual IEEE Symposium on Foundations of Computer Science (1975), 151–162
Tarjan, R.E. [1975]: Efficiency of a good but not linear set union algorithm. Journal of the ACM 22 (1975), 215–225
Tutte, W.T. [1961]: On the problem of decomposing a graph into n connected factor. Journal of the London Mathematical Society 36 (1961), 221–230
Zhou, H., Shenoy, N., and Nicholls, W. [2002]: Efficient minimum spanning tree construction without Delaunay triangulation. Information Processing Letters 81 (2002), 271–276