Now the theorem follows from Lemma 9.1.1 applied to each term above and the fact that $K_{\bar{1}}(\hat{D}) = K_{\bar{1}}(D)$. □

Now we can prove Kirchhoff's famous formula for the number of spanning trees in an undirected graph G.

Corollary 9.1.3 (Kirchoff's Matrix Tree Theorem) *[578] The number of spanning trees in an undirected graph G on n vertices is equal to any one of* $\det(K(\overleftrightarrow{G_{\bar{i}}}))$ $i \in [n]$.

Proof: Fix an arbitrary vertex r in G and observe that every spanning tree T in G corresponds to a unique out-branching B_r^+ in $\overleftrightarrow{G}$. Now the claim follows from Theorem 9.1.2. □

Since the determinant of a matrix can be calculated efficiently we obtain the following (see, e.g., [395, page 53]).

Corollary 9.1.4 *There is an $\mathcal{O}(n^3)$ algorithm for finding the number of out-branchings rooted at a given of a directed multigraph on n vertices.* □

If we actually wanted to list all out-branchings in a digraph D we clearly have to spend time at least proportional to the number of such branchings in D. In [565] Kapoor and Ramesh give an $\mathcal{O}(Nn + n^3)$ algorithm for listing all out-branchings in a directed multigraph on n vertices and N out-branchings. The algorithm is based on generating one out-branching from another by a series of arc swaps.

9.2 Optimum Branchings

Given a directed multigraph $D = (V, A)$ a special vertex s and a non-negative cost function w on the arcs. What is the minimum cost of an out-branching B_s^+ rooted at s in D? This problem, which is a natural generalization of the minimum spanning tree problem for undirected graphs (Exercise 9.6), is called the MINIMUM COST BRANCHING PROBLEM. The problem arises naturally in applications where one is seeking a minimum cost subnetwork which allows communication from a given source to all other vertices in the network (see the discussion at the end of the section).

It is easy to find a minimum spanning tree in an undirected graph. The greedy approach works as follows: order the edges according to their weights in increasing order $\mathcal{E} = \{e_1, e_2, \ldots, e_m\}$. Start from $T = \emptyset$ and go through $\mathcal{E}$ while always adding the next edge to T if it can be added without creating a cycle. This is the so-called Kruskal algorithm (see, e.g., [226]). It is not difficult to construct examples which show that using a similar greedy approach to find a minimum cost out-branching in a directed multigraph may be incorrect (Exercise 9.2).

The minimum cost branching problem was first shown to be polynomially solvable by Edmonds [275]. Later Fulkerson [357] gave a two phase greedy algorithm which solves the problem very elegantly. The fastest algorithm for the problem is due to Tarjan [819]. Tarjan's algorithm solves the problem in time $O(m \log n)$, that is, with the same time complexity as Kruskal's algorithm for undirected graphs [225]. The purpose of this section is to describe two different algorithms for finding minimum cost out-branching in a weighted directed multigraph. First we show how to solve the problem using matroids and then we give a simple direct algorithm based on Edmonds' original algorithm.

9.2.1 Matroid Intersection Formulation

To illustrate the generality of matroids, let us show how to formulate the minimum cost branching problem as a weighted matroid intersection problem. We refer to Section 18.8 for relevant definitions on matroids.

Let $D = (V, A)$ be a directed multigraph and let $r \in V$ be a vertex which can reach all other vertices by directed paths. We define $M_1 = (A, \mathcal{I}_1)$ and $M_2 = (A, \mathcal{I}_2)$ as follows (here $\mathcal{I}_1, \mathcal{I}_2 \subseteq 2^A$):

- $A' \in \mathcal{I}_1$ if and only if no two arcs in A' have a common head and no arc has head r,
- $A'' \in \mathcal{I}_2$ if and only if $UG(D\langle A''\rangle)$ has no cycle.

It follows from the definition of M_2 that M_2 is the circuit matroid of $UG(D)$ (see Section 18.8). It is easy to show that M_1 satisfies the axioms (I1)-(I3) and hence is a matroid. In particular, all maximal members of $\mathcal{I}_1$ have the same size $n-1$ (by our assumption, every vertex in $V - r$ has at least one in-neighbour) and thus the rank of M_1 is $n-1$.

Since r can reach all other vertices, $UG(D)$ is connected and hence the rank of M_2 is also $n-1$. We claim that every common base of M_1 and M_2 is an out-branching with root r. This follows easily from the definition of an out-branching and the fact that any common base corresponds to a spanning tree in $UG(D)$, since M_2 has rank $n-1$.

Thus we can find an out-branching with root r by applying the algorithm for matroid intersection of Theorem 18.8.11 to M_1, M_2. Of course such an out-branching can be found much easier by using, e.g., DFS starting from r. However, the point is that using the algorithm for weighted matroid intersection, we can find a minimum cost out-branching B_r^+ in D. It is easy to see that the required oracles for testing independence in M_1 and M_2 can be implemented very efficiently (Exercise 9.3). In fact (and much more importantly in the light of the existence of other and more efficient algorithms for minimum cost branchings), using matroid intersection algorithms we can even find a minimum cost subdigraph which has k out-branchings with a specified root s in a directed multigraph with non-negative weights on the arcs (Exercise 9.4). Furthermore, it is shown in Exercise 9.5 that using matroid intersection we

can also solve the following problem; Given directed multigraphs $D = (V, A)$ and $D' = (V, A')$ on the same vertices, a cost function c on A', a natural number k and a vertex $s \in V$. Find a minimum cost set of arcs $A^* \subseteq A'$ such that the directed multigraph $D^* = (V, A \cup A^*)$ has k-arc-disjoint out-branchings rooted at s. Clearly, the minimum cost branching problem corresponds to the case when $A = \emptyset$. Hence using matroid intersection formulations one can in fact solve problems which are much more general than the minimum cost branching problem.

9.2.2 A Simple Algorithm for Finding a Minimum Cost Out-Branching

Below we will often call a minimum cost out-branching an **optimum** out-branching. Let $D = (V, A)$ be a directed multigraph with a designated root $r \in V$ and c a non-negative cost vector on A. Denote by y_v, $v \in V - r$ the minimum cost of an arc entering v. The following easy observation is the key to the algorithm below.

Lemma 9.2.1 *Let c' be the cost function on A defined by $c'(uv) = c(uv) - y_v$. Then B_r^+ is an optimum branching with respect to c if and only if it is optimum with respect to c'.*

Proof: Since every vertex except r has precisely one arc entering it in any out-branching, $c(B_r^+) = c'(B_r^+) + \sum_{v \in V-r} y_v$ holds for an arbitrary out-branching B_r^+ and the claim follows. □

For a given directed multigraph D and weight function c, let F^* be a subdigraph of D obtained by taking a minimum cost arc entering each vertex except r, that is, $d^-_{F^*}(v) = 1$ for $v \neq r$. Note that the cost of F^* is zero with respect to c' and hence the following holds, by Lemma 9.2.1.

Lemma 9.2.2 *If F^* is an out-branching then it is optimum.* □

The following result is due to Karp.

Lemma 9.2.3 *[567] There exists an optimum out-branching with root r which contains all but one arc of every cycle C in F^*.*

Proof: Let B_r^+ be an optimum out-branching which contains the maximum number of arcs from F^*. If F^* is itself a branching then by Lemma 9.2.2 we have $B_r^+ = F^*$ so assume that C is a cycle in F^* and suppose $A(C) - A(B_r^+) = \{u_1v_1, u_2v_2, \ldots, u_rv_r\}$ has at least 2 arcs and occurring in that order on C. Consider an arbitrary vertex v_i, $i \in [r]$ and denote by $a(v_i)$ the arc entering v_i in B_r^+. By the the choice of B_r^+, $H_i = B_r^+ + u_iv_i - a(v_i)$ is not an out-branching. This implies that H_i contains a cycle which consists of the arc u_iv_i and a path P_i which starts in v_i and ends in u_i. Consider the last arc xy of P_i which does not belong to C. As H_i contains all the arcs of $C[v_{i-1}, u_i]$

and every vertex of C has in-degree one in H_i it follows that $y = v_{i-1}$ (indices are taken modulo r). Thus we have shown that H_i and hence B_r^+ contains a (v_i, v_{i-1})-path. However, this holds for every $i \in [r]$ and so B_r^+ contains a directed cycle[1], a contradiction. Hence we have shown that B_r^+ contains all but one arc of C. □

When we contract the cycle C below to get the weighted directed multigraph D/C, the arcs incident to v_C inherit the costs from the original arcs between C and $V - C$.

Lemma 9.2.4 *If C is a cycle in F^* and W_r^+ is an optimum out-branching in D/C (the directed multigraph obtained by contracting C to a vertex v_C), then we can obtain an optimum branching B_r^+ in D by replacing v_C by C minus one arc.*

Proof: Let xv_C be the unique arc of W_r^+ entering v_C and let $y \in V(C)$ be chosen so that $xy \in A$ and has the same cost as xv_C in D/C. Clearly we can extend W_r^+ to an out-branching B_r^+ of D by blowing up C again and deleting the unique arc of C which enters y (arcs leaving v_C in W_r^+ are replaced by corresponding arcs starting in vertices from C). By Lemma 9.2.3, there exists and optimum out-branching $\hat{B}_r^+$ containing all but one arc of C and contracting C will tranform $\hat{B}_r^+$ into an out-branching $\tilde{B}_r^+$ in D/C. Now, by Lemma 9.2.1, it follows from the fact that W_r^+ is an optimum out-branching in D/C and C has cost zero w.r.t. c' that $c'(\hat{B}_r^+) \geq c'(B_r^+)$, implying that B_r^+ is an optimum out-branching. □

Theorem 9.2.5 *[275] There is a polynomial algorithm for the minimum cost out-branching problem.*

Proof: We may assume that the root r can reach every other vertex, as otherwise no branching exists. The algorithm is very simple. First construct F^* and search for a cycle in it. If F^* is acyclic it is the desired branching. If F^* contains a cycle C, let $D' = D/C$ and solve the problem recursively in D'. Finally convert the optimum out-branching in D' to an optimum out-branching of D as described in the proof of Lemma 9.2.4. This algorithm can easily be implemented as an $O(n(n+m))$ algorithm. □

9.3 Arc-Disjoint Branchings

This section is devoted to a very important result due to Edmonds [277]. The result can be viewed as just a fairly simple generalization of Menger's theorem. However, as will be clear from the next subsections, it has many important consequences.

[1] Note that when $r = 1$ we do not get the contradiction since P_1 is simply $C[v_1, u_1]$.