Lemma 8.23(c) says that the number of RELABEL operations is $O(n^2)$. Since each RELABEL operation takes $O(n)$ time, and scanning the neighbours of a vertex v in the DISCHARGE procedure between two relabelings of v takes $O(n)$ time again, we have an overall running time of $O(n^3)$ plus $O(1)$ times the number of nonsaturating pushes. (Note that each PUSH operation can be done in constant time.)

Since there are at most $O(n^3)$ nonsaturating pushes by Lemma 8.26, the theorem is proved. □

8.6 Gomory-Hu Trees

Any algorithm for the MAXIMUM FLOW PROBLEM also implies a solution to the following problem:

MINIMUM CAPACITY CUT PROBLEM	
Instance:	A network (G, u, s, t)
Task:	An s-t-cut in G with minimum capacity.

Proposition 8.28. *The* MINIMUM CAPACITY CUT PROBLEM *can be solved in the same running time as the* MAXIMUM FLOW PROBLEM, *in particular in* $O(n^3)$ *time.*

Proof: For a network (G, u, s, t) we compute a maximum s-t-flow f and define X to be the set of all vertices reachable from s in G_f. X can be computed with the GRAPH SCANNING ALGORITHM in linear time (Proposition 2.17). By Lemma 8.3 and Theorem 8.5, $\delta_G^+(X)$ constitutes a minimum capacity s-t-cut. □

In this section we consider the problem of finding a minimum capacity s-t-cut for each pair of vertices s, t in an undirected graph G with capacities $u : E(G) \to \mathbb{R}_+$.

This problem can be reduced to the above one: For all pairs $s, t \in V(G)$ we solve the MINIMUM CAPACITY CUT PROBLEM for (G', u', s, t), where (G', u') arises from (G, u) by replacing each undirected edge $\{v, w\}$ by two oppositely directed edges (v, w) and (w, v) with $u'((v, w)) = u'((w, v)) = u(\{v, w\})$. In this way we obtain a minimum s-t-cut for all s, t after $\binom{n}{2}$ flow computations.

This section is devoted to the elegant method of Gomory and Hu [1961], which requires only $n - 1$ flow computations. We shall see some applications in Sections 12.3 and 20.2.

Definition 8.29. *Let G be an undirected graph and $u : E(G) \to \mathbb{R}_+$ a capacity function. For two vertices $s, t \in V(G)$ we denote by λ_{st} their* **local edge-connectivity**, *i.e. the minimum capacity of a cut separating s and t.*

The edge-connectivity of a graph is obviously the minimum local edge-connectivity with respect to unit capacities.

Lemma 8.30. *For all vertices $i, j, k \in V(G)$ we have $\lambda_{ik} \geq \min(\lambda_{ij}, \lambda_{jk})$.*

Proof: Let $\delta(A)$ be a cut with $i \in A$, $k \in V(G) \setminus A$ and $u(\delta(A)) = \lambda_{ik}$. If $j \in A$ then $\delta(A)$ separates j and k, so $u(\delta(A)) \geq \lambda_{jk}$. If $j \in V(G) \setminus A$ then $\delta(A)$ separates i and j, so $u(\delta(A)) \geq \lambda_{ij}$. We conclude that $\lambda_{ik} = u(\delta(A)) \geq \min(\lambda_{ij}, \lambda_{jk})$. □

Indeed, this condition is not only necessary but also sufficient for numbers $(\lambda_{ij})_{1 \leq i,j \leq n}$ with $\lambda_{ij} = \lambda_{ji}$ to be local edge-connectivities of some graph (Exercise 20).

Definition 8.31. *Let G be an undirected graph and $u : E(G) \to \mathbb{R}_+$ a capacity function. A tree T is called a* **Gomory-Hu tree** *for (G, u) if $V(T) = V(G)$ and*

$$\lambda_{st} = \min_{e \in E(P_{st})} u(\delta_G(C_e)) \quad \text{for all } s, t \in V(G),$$

where P_{st} is the (unique) s-t-path in T and, for $e \in E(T)$, C_e and $V(G) \setminus C_e$ are the connected components of $T - e$ (i.e. $\delta_G(C_e)$ is the fundamental cut of e with respect to T).

We shall see that every graph possesses a Gomory-Hu tree. This implies that for any undirected graph G there is a list of $n - 1$ cuts such that for each pair $s, t \in V(G)$ a minimum s-t-cut belongs to the list.

In general, a Gomory-Hu tree cannot be chosen as a subgraph of G. For example, consider $G = K_{3,3}$ and $u \equiv 1$. Here $\lambda_{st} = 3$ for all $s, t \in V(G)$. It is easy to see that the Gomory-Hu trees for (G, u) are exactly the stars with five edges.

The main idea of the algorithm for constructing a Gomory-Hu tree is as follows. First we choose any $s, t \in V(G)$ and find some minimum s-t-cut, say $\delta(A)$. Let $B := V(G) \setminus A$. Then we contract A (resp. B) to a single vertex, choose any $s', t' \in B$ (resp. $s', t' \in A$) and look for a minimum s'-t'-cut in the contracted graph G'. We continue this process, always choosing a pair s', t' of vertices not separated by any cut obtained so far. At each step, we contract – for each cut $E(A', B')$ obtained so far – A' or B', depending on which part does not contain s' and t'.

Eventually each pair of vertices is separated. We have obtained a total of $n - 1$ cuts. The crucial observation is that a minimum s'-t'-cut in the contracted graph G' is also a minimum s'-t'-cut in G. This is the subject of the following lemma. Note that when contracting a set A of vertices in (G, u), the capacity of each edge in G' is the capacity of the corresponding edge in G.

Lemma 8.32. *Let G be an undirected graph and $u : E(G) \to \mathbb{R}_+$ a capacity function. Let $s, t \in V(G)$, and let $\delta(A)$ be a minimum s-t-cut in (G, u). Let now $s', t' \in V(G) \setminus A$, and let (G', u') arise from (G, u) by contracting A to a single vertex. Then for any minimum s'-t'-cut $\delta(K \cup \{A\})$ in (G', u'), $\delta(K \cup A)$ is a minimum s'-t'-cut in (G, u).*

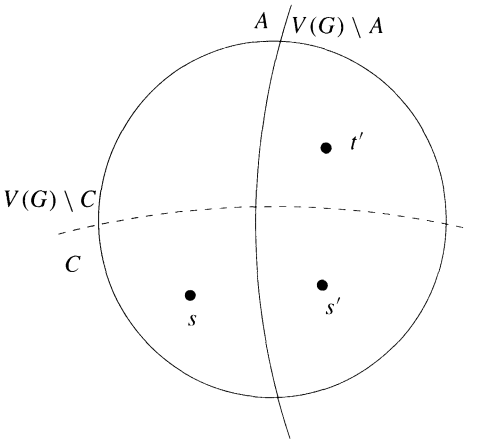

Fig. 8.3.

Proof: Let s, t, A, s', t', G', u' be as above. W.l.o.g. $s \in A$. It suffices to prove that there is a minimum s'-t'-cut $\delta(A')$ in (G, u) such that $A \subset A'$. So let $\delta(C)$ be any minimum s'-t'-cut in (G, u). W.l.o.g. $s \in C$.

Since u is submodular (cf. Lemma 2.1(a)), we have $u(\delta(A)) + u(\delta(C)) \geq u(\delta(A \cap C)) + u(\delta(A \cup C))$. But $\delta(A \cap C)$ is an s-t-cut, so $u(\delta(A \cap C)) \geq \lambda_{st} = u(\delta(A))$. Therefore $u(\delta(A \cup C)) \leq u(\delta(C)) = \lambda_{s't'}$ proving that $\delta(A \cup C)$ is a minimum s'-t'-cut. (See Figure 8.3.) □

Now we describe the algorithm which constructs a Gomory-Hu tree. Note that the vertices of the intermediate trees T will be vertex sets of the original graph; indeed they form a partition of $V(G)$. At the beginning, the only vertex of T is $V(G)$. In each iteration, a vertex of T containing at least two vertices of G is chosen and split into two.

GOMORY-HU ALGORITHM	
Input:	An undirected graph G and a capacity function $u : E(G) \to \mathbb{R}_+$.
Output:	A Gomory-Hu tree T for (G, u).

① Set $V(T) := \{V(G)\}$ and $E(T) := \emptyset$.

② Choose some $X \in V(T)$ with $|X| \geq 2$. **If** no such X exists **then go to** ⑥.

③ Choose $s, t \in X$ with $s \neq t$.
For each connected component C of $T - X$ **do**: Let $S_C := \bigcup_{Y \in V(C)} Y$.
Let (G', u') arise from (G, u) by contracting S_C to a single vertex v_C for each connected component C of $T - X$.
(So $V(G') = X \cup \{v_C : C$ is a connected component of $T - X\}$.)

④ Find a minimum s-t-cut $\delta(A')$ in (G', u'). Let $B' := V(G') \setminus A'$.
Set $A := \left(\bigcup_{v_C \in A' \setminus X} S_C \right) \cup (A' \cap X)$ and $B := \left(\bigcup_{v_C \in B' \setminus X} S_C \right) \cup (B' \cap X)$.

⑤ Set $V(T) := (V(T) \setminus \{X\}) \cup \{A \cap X, B \cap X\}$.
For each edge $e = \{X, Y\} \in E(T)$ incident to the vertex X **do**:
If $Y \subseteq A$ **then** set $e' := \{A \cap X, Y\}$ **else** set $e' := \{B \cap X, Y\}$.
Set $E(T) := (E(T) \setminus \{e\}) \cup \{e'\}$ and $w(e') := w(e)$.
Set $E(T) := E(T) \cup \{\{A \cap X, B \cap X\}\}$ and
$w(\{A \cap X, B \cap X\}) := u'(\delta_{G'}(A'))$.
Go to ②.

⑥ Replace all $\{x\} \in V(T)$ by x and all $\{\{x\}, \{y\}\} \in E(T)$ by $\{x, y\}$. **Stop**.

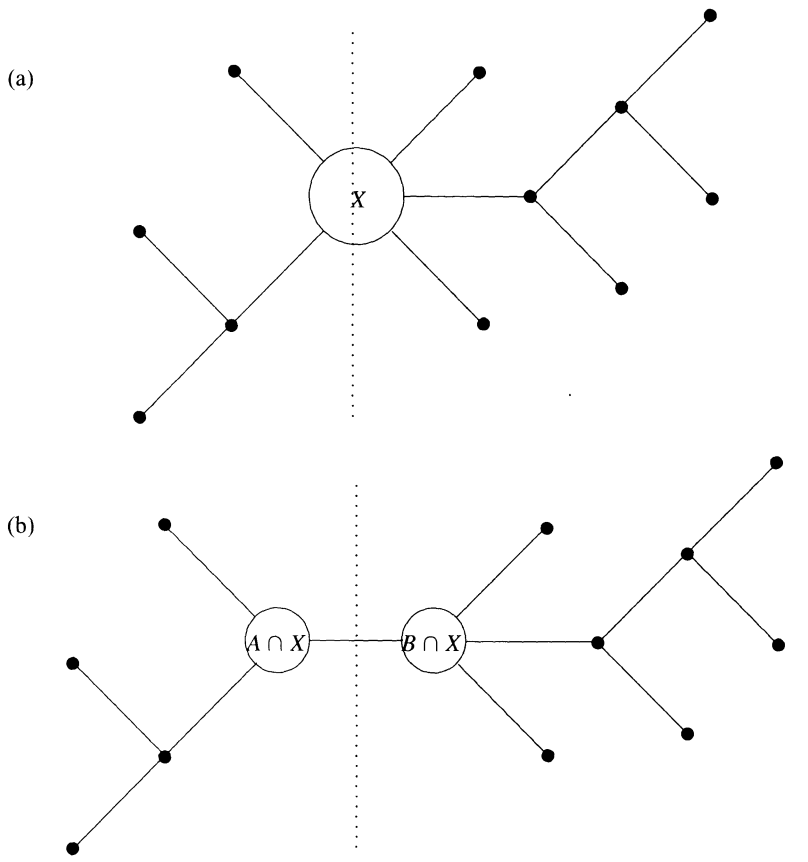

Fig. 8.4.

Figure 8.4 illustrates the modification of T in ⑤. To prove the correctness of this algorithm, we first show the following lemma:

Lemma 8.33. *Each time at the end of ④ we have*

(a) $A \dot{\cup} B = V(G)$
(b) $E(A, B)$ *is a minimum* s-t-*cut in* (G, u).

Proof: The elements of $V(T)$ are always nonempty subsets of $V(G)$, indeed $V(T)$ constitutes a partition of $V(G)$. From this, (a) follows easily.

We now prove (b). The claim is trivial for the first iteration (since here $G' = G$). We show that the property is preserved in each iteration.

Let $C_1, \ldots, C_k$ be the connected components of $T - X$. Let us contract them one by one; for $i = 0, \ldots, k$ let (G_i, u_i) arise from (G, u) by contracting each of $S_{C_1}, \ldots, S_{C_i}$ to a single vertex. So (G_k, u_k) is the graph which is denoted by (G', u') in ③ of the algorithm.

Claim: For any minimum s-t-cut $\delta(A_i)$ in (G_i, u_i), $\delta(A_{i-1})$ is a minimum s-t-cut in (G_{i-1}, u_{i-1}), where

$$A_{i-1} := \begin{cases} (A_i \setminus \{v_{C_i}\}) \cup S_{C_i} & \text{if } v_{C_i} \in A_i \\ A_i & \text{if } v_{C_i} \notin A_i \end{cases}.$$

Applying this claim successively for $k, k-1, \ldots, 1$ implies (b).

To prove the claim, let $\delta(A_i)$ be a minimum s-t-cut in (G_i, u_i). By our assumption that (b) is true for the previous iterations, $\delta(S_{C_i})$ is a minimum s_i-t_i-cut in (G, u) for some appropriate $s_i, t_i \in V(G)$. Furthermore, $s, t \in V(G) \setminus S_{C_i}$. So applying Lemma 8.32 completes the proof. □

Lemma 8.34. *At any stage of the algorithm (until ⑥ is reached) for all* $e \in E(T)$

$$w(e) = u\left(\delta_G\left(\bigcup_{Z \in C_e} Z\right)\right),$$

where C_e *and* $V(T) \setminus C_e$ *are the connected components of* $T - e$. *Moreover for all* $e = \{P, Q\} \in E(T)$ *there are vertices* $p \in P$ *and* $q \in Q$ *with* $\lambda_{pq} = w(e)$.

Proof: Both statements are trivial at the beginning of the algorithm when T contains no edges; we show that they are never violated. So let X be vertex of T chosen in ② in some iteration of the algorithm. Let s, t, A', B', A, B be as determined in ③ and ④ next. W.l.o.g. assume $s \in A'$.

Edges of T not incident to X are not affected by ⑤. For the new edge $\{A \cap X, B \cap X\}$, $w(e)$ is clearly set correctly, and we have $\lambda_{st} = w(e)$, $s \in A \cap X$, $t \in B \cap X$.

So let us consider an edge $e = \{X, Y\}$ that is replaced by e' in ⑤. We assume w.l.o.g. $Y \subseteq A$, so $e' = \{A \cap X, Y\}$. Assuming that the assertions were true for e we claim that they remain true for e'. This is trivial for the first assertion, because $w(e) = w(e')$ and $u\left(\delta_G\left(\bigcup_{Z \in C_e} Z\right)\right)$ does not change.

To show the second statement, we assume that there are $p \in X, q \in Y$ with $\lambda_{pq} = w(e)$. If $p \in A \cap X$ then we are done. So henceforth assume that $p \in B \cap X$ (see Figure 8.5).

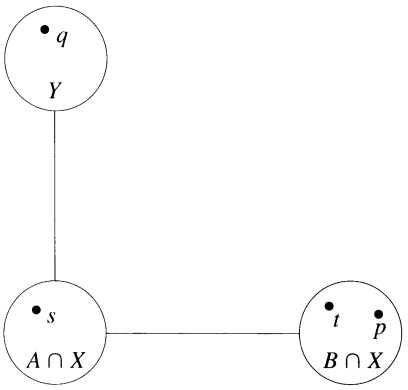

Fig. 8.5.

We claim that $\lambda_{sq} = \lambda_{pq}$. Since $\lambda_{pq} = w(e) = w(e')$ and $s \in A \cap X$, this will conclude the proof.

By Lemma 8.30,

$$\lambda_{sq} \geq \min\{\lambda_{st}, \lambda_{tp}, \lambda_{pq}\}.$$

Since by Lemma 8.33(b) $E(A, B)$ is a minimum s-t-cut, and since $s, q \in A$, we may conclude from Lemma 8.32 that λ_{sq} does not change if we contract B. Since $t, p \in B$, this means that adding an edge $\{t, p\}$ with arbitrary high capacity does not change λ_{sq}. Hence

$$\lambda_{sq} \geq \min\{\lambda_{st}, \lambda_{pq}\}.$$

Now observe that $\lambda_{st} \geq \lambda_{pq}$ because the minimum s-t-cut $E(A, B)$ also separates p and q. So we have

$$\lambda_{sq} \geq \lambda_{pq}.$$

To prove equality, observe that $w(e)$ is the capacity of a cut separating X and Y, and thus s and q. Hence

$$\lambda_{sq} \leq w(e) = \lambda_{pq}.$$

This completes the proof. □

Theorem 8.35. (Gomory and Hu [1961]) *The* GOMORY-HU ALGORITHM *works correctly. Every undirected graph possesses a Gomory-Hu tree, and such a tree is found in* $O(n^4)$ *time.*

Proof: The complexity of the algorithm is clearly determined by $n-1$ times the complexity of finding a minimum s-t-cut, since everything else can be implemented in $O(n^3)$ time. Using the GOLDBERG-TARJAN ALGORITHM (Theorem 8.27) we obtain the $O(n^4)$ bound.

We prove that the output T of the algorithm is a Gomory-Hu tree for (G,u). It should be clear that T is a tree with $V(T)=V(G)$. Now let $s,t \in V(G)$. Let P_{st} be the (unique) s-t-path in T and, for $e \in E(T)$, let C_e and $V(G) \setminus C_e$ be the connected components of $T-e$.

Since $\delta(C_e)$ is an s-t-cut for each $e \in E(P_{st})$,

$$\lambda_{st} \leq \min_{e \in E(P_{st})} u(\delta(C_e)).$$

On the other hand, a repeated application of Lemma 8.30 yields

$$\lambda_{st} \geq \min_{\{v,w\} \in E(P_{st})} \lambda_{vw}.$$

Hence applying Lemma 8.34 to the situation before execution of ⑥ (where each vertex X of T is a singleton) yields

$$\lambda_{st} \geq \min_{e \in E(P_{st})} u(\delta(C_e)),$$

so equality holds. □

A similar algorithm for the same task (which might be easier to implement) was suggested by Gusfield [1990].

8.7 The Minimum Cut in an Undirected Graph

If we are only interested in a minimum capacity cut in an undirected graph G with capacities $u : E(G) \to \mathbb{R}_+$, there is a simpler method using $n-1$ flow computations: just compute the minimum s-t-cut for some fixed vertex s and each $t \in V(G) \setminus \{s\}$. However, there are more efficient algorithms.

Hao and Orlin [1994] found an $O(nm \log \frac{n^2}{m})$-algorithm for determining the minimum capacity cut. They use a modified version of the GOLDBERG-TARJAN ALGORITHM.

If we just want to compute the edge-connectivity of the graph (i.e. unit capacities), the currently fastest algorithm is due to Gabow [1995] with running time $O(m+\lambda^2 n \log \frac{n}{\lambda(G)})$, where $\lambda(G)$ is the edge-connectivity (observe that $m \geq \lambda n$). Gabow's algorithm uses matroid intersection techniques. We remark that the MAXIMUM FLOW PROBLEM in undirected graphs with unit capacities can also be solved faster than in general (Karger and Levine [1998]).

Nagamochi and Ibaraki [1992] found a completely different algorithm to determine the minimum capacity cut in an undirected graph. Their algorithm does not use max-flow computations at all. In this section we present this algorithm in a simplified form due to Stoer and Wagner [1997] and independently to Frank [1994]. We start with an easy definition.