4. Graph-Theoretic Algorithms - NB

1. Maximum Flows

The links (edges) in a telecommunication network are associated with cost and speed (bandwidth, or capacity). The following types of problems are studied in this section.

(a) Assume that a network is specified by link capacity metric. A network of this type is called a capacitated network. It is interesting to determine the maximum possible traffic flow in this network.

(b) A network is specified by link cost and capacity metrics, and certain traffic flow requirement between source and destination nodes. A least cost scheme for transfer of traffic has to be determined.

In this section, abstraction of such problems are studied. Specifically, the problems associated with maximum flows, and minimum cost flows are examined. In summary, traffic *flows* in a network are studied. It is implicitly assumed in this section that all the digraphs are simple and connected.

In the maximum traffic flow problem, a network with nonnegative link capacities is specified. The goal of the problem is to determine maximum flow that is possible between the source node and the sink (destination) node. Minimum cost flow algorithms are also discussed in this section. In addition to the capacity constraints in a network, each link is assigned a cost. The purpose of this algorithm is to determine the least cost transportation of goods (network traffic) such that the demands at specified vertices are met by supplies available at the source vertices. This formulation of the problem is a generalization of the maximum flow problem, and the shortest path problem.

The theoretical details developed in this section can also be used to justify the correctness of the edge-disjoint shortest paths algorithm. The edge-disjoint shortest paths algorithm was developed earlier in the chapter.

1.1. Maximum Flows.

The maximum flow problem determines the maximum possible flow in a capacitated network. The network variables are defined below.

Definition

(a) *Let $G = (V, E)$ be a directed graph, where V is the vertex (node) set, E is the set of edges, $|V| = n$, $|E| = m$, and $V = \{1, 2, \ldots, n\}$.*

(b) *Associated with this graph is a mapping $u : E \to \mathbb{R}_0^+$, that is for each edge $e \in E$ there is a value $u(e) \geq 0$ called the capacity of the edge e. If an arc is denoted by (i, j), where i and j are the vertices, then its capacity is specified by u_{ij}. For simplicity assume that the capacities are finite. Notice that arcs with zero capacity are also allowed. The mapping u is generally specified by an $n \times n$ matrix denoted by U, where $U = [u_{ij}]$.*

(c) *The graph does not contain parallel arcs. That is two or more arcs with the same head and tail nodes are not permissible. Also assume that the graph is connected.*

(d) *In this graph there are two special vertices called the source node s and the sink (target, destination) node t, such that t is accessible from s.*

(e) *The flow network or a capacitated network is specified by $N = (G, u, s, t)$.* □

The maximum flow scheme routes maximum flow of traffic from the source to the sink node. Some more notation and assumptions are listed below.

Definitions. *Let $N = (G, u, s, t)$ be a capacitated network.*

1. *A feasible flow on N is a mapping $x : E \to \mathbb{R}_0^+$ which satisfies the following conditions*:

   (a) *Capacity constraints.* $0 \le x_{ij} \le u_{ij}$ *for all arcs* $(i, j) \in E$.

   (b) *Flow conservation.* $\sum_{\{j|(i,j)\in E\}} x_{ij} = \sum_{\{j|(j,i)\in E\}} x_{ji}$ *for all* $i \in V - \{s, t\}$.
   In this statement, $\sum_{\{j|(i,j)\in E\}} x_{ij}$ *represents the total flow out of node* i, *and* $\sum_{\{j|(j,i)\in E\}} x_{ji}$ *represents the total flow into the node* i.

   The mapping x is generally specified by an $n \times n$ matrix denoted by x, such that $x = [x_{ij}]$.

2. *The value of flow x is $v = \sum_{\{j|(s,j)\in E\}} x_{sj}$. It is equal to the total flow leaving the source node.*

3. *A maximum flow on N is a flow for which v has a maximum value.* □

Observation. The capacity constraint implies that each edge can carry a nonnegative flow bounded from above by the capacity of the edge. The flow conservation constraint means that flows are preserved at all vertices except at the source and sink nodes. That is the amount which flows into a node (other than the source and sink nodes) equals the amount that flows out of it. Thus for any $i \in V$

$$\sum_{j\in V} x_{ji} - \sum_{j\in V} x_{ij} = \begin{cases} -v, & i = s \\ 0, & i \neq s, t \\ v, & i = t \end{cases}$$

□

The maximum flow problem in a capacitated network is next stated formally. *In a capacitated network $N = (G, u, s, t)$, maximize the value of v, which is the flow between the source node s and the sink node t, and determine the corresponding flow x.* In order to develop the theory further, the concept of cut of a set is introduced.

Definitions. *Let $N = (G, u, s, t)$ be a capacitated network, where $G = (V, E)$ is a directed graph.*

1. *The arc $(i, j) \in E$ is saturated in flow x if $x_{ij} = u_{ij}$.*

2. *A cut $\langle S, T\rangle$ is a partition of the vertex set V into two subsets S and T such that $T = (V - S)$. The arcs which are directed from S to T are called forward arcs, and the arcs which are directed from T to S are called backward arcs. The sets of forward and backward arcs are denoted by (S, T) and (T, S) respectively.*

3. *The cut* $\langle S, T\rangle$ *is called an* s-t *cut if* $s \in S$ *and* $t \in T$. *The capacity of the* s-t *cut* $\langle S, T\rangle$ *is denoted by* $u\langle S, T\rangle = \sum_{(i,j)\in(S,T)} u_{ij}$.

4. *The flow across an* s-t *cut is called an* s-t *flow.*

5. *A minimum cut is an* s-t *cut which has a minimum capacity.* □

Theorem. *The value of any* s-t *flow is less than or equal to the capacity of any* s-t *cut in the network.*

Proof. Let x be a flow in the network and $\langle S, T\rangle$ be an s-t cut. If $i \in S$, then $\left\{\sum_{j\in V} x_{ij} - \sum_{j\in V} x_{ji}\right\}$ is equal v if $i = s$, and it is equal to zero if $i \neq s, t$. Therefore

$$\begin{aligned} v &= \sum_{i\in S}\left\{\sum_{j\in V} x_{ij} - \sum_{j\in V} x_{ji}\right\} \\ &= \sum_{i\in S}\sum_{j\in S}(x_{ij} - x_{ji}) + \sum_{i\in S}\sum_{j\in T}(x_{ij} - x_{ji}) \end{aligned}$$

The first set of summations add to zero. Thus

$$v = \sum_{i\in S}\sum_{j\in T}(x_{ij} - x_{ji})$$

This equation implies that the value v of the flow is equal to the net flow through the s-t cut $\langle S, T\rangle$. However $x_{ji} \geq 0$ and $x_{ij} \leq u_{ij}$, consequently

$$v \leq \sum_{(i,j)\in(S,T)} u_{ij} = u\langle S, T\rangle \tag{4.14}$$

□

The result of the above theorem should be intuitively evident. Any s-t flow must pass through any s-t cut in the network. Thus the value of any s-t flow cannot exceed the capacity of an s-t cut with the smallest value.

Corollary. *Let* x *be a feasible flow between a source node* s *and a target node* t, *with value* v. *Assume that* $\langle S, T\rangle$ *is any* s-t *cut in the network, and the net flow across this cut is equal to* v. *Then*

$$v = \sum_{(i,j)\in(S,T)} x_{ij} - \sum_{(i,j)\in(T,S)} x_{ij} \tag{4.15}$$

Proof. See the problem section. □

Corollary. *Let* x *be a feasible flow between a source node* s *and a target node* t, *with value* v. *If* v *is equal to the capacity of some* s-t *cut* $\langle S, T\rangle$ *in the network, then* x *is a maximum flow and* $\langle S, T\rangle$ *is a minimum cut.* □

Example. Consider a graph $G = (V, E)$, where $V = \{1, 2, 3, 4, 5, 6\}$ and the edges are specified by the adjacency matrix A. The capacities of the arcs are specified by the capacity matrix U.

$$A = \begin{bmatrix} 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}, \quad U = \begin{bmatrix} 0 & 5 & 0 & 7 & 0 & 0 \\ 0 & 0 & 6 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 8 \\ 0 & 4 & 0 & 0 & 5 & 0 \\ 0 & 0 & 5 & 0 & 0 & 6 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

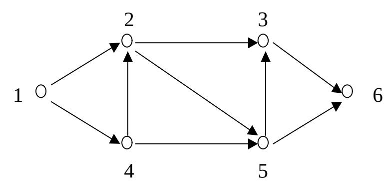

Figure 1.. The graph $G = (V, E)$.

See Figure 1. Let the source node be $s = 1$, and the sink node be $t = 6$. A possible feasible flow x is specified by:

$$x_{12} = 3,\ x_{14} = 4,\ x_{23} = 4,\ x_{25} = 2,\ x_{36} = 5,\ x_{42} = 3,\ x_{45} = 1,\ x_{53} = 1,\ \text{and } x_{56} = 2$$

The value of this flow is $v = 7$. Note that $v = x_{12} + x_{14} = x_{36} + x_{56} = 7$. Also, none of the arcs is saturated, because all arc flows are less than their respective capacities. Observe that, flow conservation is maintained at all the nodes. The flow across the cut $\langle S, T\rangle$ where $S = \{1, 2\}$ and $T = \{3, 4, 5, 6\}$ is next evaluated. This flow is equal to $x_{14} + x_{23} + x_{25} - x_{42} = 4 + 4 + 2 - 3 = 7 = v$. The capacity of this cut is computed to be

$$u\langle S, T\rangle = u_{14} + u_{23} + u_{25} = 7 + 6 + 3 = 16$$

Therefore the value of the flow in the network is bounded from above by 16. The capacity of the cut $\langle S', T'\rangle$ where $S' = \{1\}$ and $T' = \{2, 3, 4, 5, 6\}$ is computed to be

$$u\langle S', T'\rangle = u_{12} + u_{14} = 5 + 7 = 12$$

This cut capacity provides an improved value for the upper bound on the flow. Consider another feasible flow x'.

$$x'_{12} = 5,\ x'_{14} = 7,\ x'_{23} = 6,\ x'_{25} = 2,\ x'_{36} = 6,\ x'_{42} = 3,\ x'_{45} = 4,\ x'_{53} = 0,\ \text{and } x'_{56} = 6$$

The value of this flow is $v' = 12$, which is equal to the capacity $u\langle S', T'\rangle$. Consequently, $\langle S', T'\rangle$ is the minimum cut and the value of the maximal flow is equal to 12. $\square$

It is next proved that the maximum value of a flow in a capacitated network equals the minimum capacity of all s-t cuts. An algorithm for determining maximum flows is also developed. These are based upon the concept of *augmenting path* and *residual network.*

The concept of a residual network of a graph G is based upon the following simple idea. If the edge $(i, j) \in E$ of capacity u_{ij} carries a flow of x_{ij} units, then it can potentially carry another $(u_{ij} - x_{ij})$ units of flow on this edge. Also observe that the edge (i, j) can carry an additional x_{ji} units, which is the flow from node j to node i. Thus the residual network $G(x)$ with respect to the flow x is defined as follows. Replace *each* edge (i, j) in the graph G by *two* edges (i, j) and (j, i). Let the capacity of these two edges be $r_{ij} = (u_{ij} - x_{ij})$ and $r_{ji} = x_{ij}$ respectively.

However, if the initial network specified by the graph G has edges (i, j) and (j, i) then the residual graph will have two parallel edges from node j to node i. This restriction can be overcome by a suitable transformation of the original graph.

Definitions. *Assume that $N = (G, u, s, t)$ is a capacitated network.*

1. *For a given flow x, the residual capacity r_{ij} of an arc $(i, j) \in E$ is the maximum additional flow which can be sent from vertex i to vertex j using the arcs (i, j) and (j, i). The residual capacity $r_{ij} = (u_{ij} - x_{ij}) + x_{ji}$. Observe that $(u_{ij} - x_{ij})$ represents the unused capacity of the arc (i, j); and x_{ji} is the value of flow on arc (j, i) that can be cancelled to increase the flow from vertex i to vertex j. The capacities in the residual network are specified in an $n \times n$ matrix r.*

2. *The residual network $G(x)$ for a specified value of flow x are those arcs in G which have a positive residual capacity.*

3. *An augmenting path in $G(x)$ is a directed path from vertex s to vertex t in residual network $G(x)$.*

4. *The capacity of a directed path is the smallest value of arc capacity which lies on the path.*

5. *The residual capacity of an s-t cut $\langle S, T\rangle$ with respect to a flow x is the sum of residual capacities of forward arcs in the cut. It is denoted by $r\langle S, T\rangle$. Thus $r\langle S, T\rangle = \sum_{(i,j)\in(S,T)} r_{ij}$.* □

Lemma. *A flow x in a network $N = (G, u, s, t)$ has a value v. The extra s-t flow that is possible in this network is equal to the residual capacity of any s-t cut.*

Proof. Let x' be a flow of value $(v + \Delta v)$, where $\Delta v \geq 0$. Therefore

$$v + \Delta v \leq \sum_{(i,j)\in(S,T)} u_{ij}$$

Using the result

$$v = \sum_{(i,j)\in(S,T)} x_{ij} - \sum_{(i,j)\in(T,S)} x_{ij}$$

yields

$$\Delta v \leq \sum_{(i,j)\in(S,T)} (u_{ij} - x_{ij}) + \sum_{(i,j)\in(T,S)} x_{ij} = \sum_{(i,j)\in(S,T)} (u_{ij} - x_{ij} + x_{ji}) = \sum_{(i,j)\in(S,T)} r_{ij}$$

Therefore

$$\Delta v \leq \sum_{(i,j)\in(S,T)} r_{ij}$$

□

The following fact is immediate from the above result.

Lemma. *For any network flow x of value v, the additional flow that is possible between the source and sink nodes is less than or equal to the residual capacity of any s-t cut.* □

Using the above definitions, properties of augmenting paths are established. The next theorem is called the *augmenting path theorem.*

Theore. *A flow x in a flow network $N = (G, u, s, t)$ is maximal if and only if there are no augmenting path from s to t.*

Proof. It is evident that if there is an augmenting path, then the flow is not maximal. Conversely, assume that the flow x does not permit an augmenting path in the network. Let S be the set of all nodes i, including the source node s for which there is an augmenting path from s to node i. Furthermore, let $T = V - S$. Using the definition of augmenting path, and of S and T; it follows that $x_{ij} = u_{ij}$ and $x_{ji} = 0$ for all values of $i \in S$ and $j \in T$. Therefore

$$\begin{aligned} v &= \sum_{i \in S} \sum_{j \in T} (x_{ij} - x_{ji}) \\ &= \sum_{i \in S} \sum_{j \in T} u_{ij} = u \langle S, T \rangle \end{aligned}$$

which is the capacity of the cut $\langle S, T \rangle$. But $v \leq u \langle S, T \rangle$ from theorem 4.3, therefore the flow is maximal. □

Corollary. *A flow x is a maximum flow iff the residual network specified by $G(x)$ has no augmenting path.* □

The next set of three theorems are useful in developing a useful and intuitive maximal flow algorithm. The integral property of the flow is proved in the next theorem. It is called the *integral flow theorem.*

Theorem. *Let $N = (G, u, s, t)$ be a flow network in which all capacities are integers. There exists a maximal flow on N such that the value of flow on any arc is integral.*

Proof. As per the hypothesis of the theorem, all capacities are integers. Initially assume that the flow x'_{ij} on arc (i, j) is equal to zero for all $i, j \in V$. Let the corresponding value of the flow be v'. If this flow is not maximal, then an augmenting path exists such that there is an integral flow $v'' > v'$. If v'' is not maximal, then the flow network admits an augmenting path, and so on. This process can be continued iteratively till the flow does not permit any more augmenting paths. Note that in this process the value of the flow increases from the previous step by at least unity. Consequently at the end of this process the value of the maximal flow is integral, and there is no augmenting path. □

The celebrated *max-flow min-cut theorem* was proved by L. R. Ford and D. R. Fulkerson and also independently by P. Elias, A. Feinstein and C. E. Shannon in the year 1956.

Theorem. *Consider a flow network $N = (G, u, s, t)$. The maximum value of an s-t flow on the network N is equal to the minimum capacity of an s-t cut.*

Proof. The inequality $v \leq u \langle S, T \rangle$, the augmenting path and integral flow theorems together imply this theorem if the capacities are all integers.

However, if the capacities are rational numbers, then it can be reduced to the integer case by multiplying all the capacities by the lowest common multiple of the denominators. This is the case when all the capacities are represented on a computer. The case of real-valued capacities can be addressed by using a continuity argument and observing that v is a continuous function of the x_{ij}'s.

Furthermore, the existence of a cut with minimum capacity is guaranteed because there are only a finite number of cuts. □

A Max-Flow Min-Cut Algorithm

An augmenting path algorithm for determining maximal flow in a network is outlined below. It is based upon the above theoretical details.

Algorithm. *Outline of Max-Flow Min-Cut Algorithm.*

Input: A flow network $N = (G, u, s, t)$. The capacity matrix is U.
Output: Maximum flow x.
begin
    $x \leftarrow 0$
    $r \leftarrow U$
    while ($G(x)$ contains a directed path between vertices s and t) **do**
    begin
        find an augmenting path P in $G(x)$
        $\theta \leftarrow \min \{r_{ij} \mid (i, j) \in P\}$
        augment the flow along the path P by θ units
        update the residual network $G(x)$
    end (*end while loop*)
    find the maximal flow from the final residual network $G(x)$
end (*end of the outline of max-flow min-cut algorithm*)

This algorithm requires a technique to determine an augmenting path P in the residual network $G(x)$. Furthermore, it is not clear if it terminates in a finite number of steps. The above algorithm has to be further developed to meet this goal. This is done via a *labeling procedure*. The purpose of this labeling procedure is to determine via a search algorithm an augmenting path which starts at node s and terminates at node t.

In order to reach the target node t from the source node s, the algorithm determines via a *fanning out* process, all the nodes which are reachable from the source node s along a directed path in the residual network $G(x)$. In this process, the algorithm splits the nodes in the network into two parts: *labeled* and *unlabeled nodes*. The labeled nodes of the network $G(x)$ are a set of nodes which have been reached from s during the fanning out process. Thus, there is a direct path from s to all the labeled nodes. The unlabeled nodes are the nodes which have not yet been reached from the node s during the fanning out process.

The algorithm essentially selects a labeled node in the network $G(x)$, and scans its adjacency list of vertices for unlabeled nodes. It then labels these additional nodes. Iteration of this process ultimately results in labeling the sink node t. When this occurs, the algorithm transfers maximum possible flow along this directed path from node s to node t. Based upon this path and its flow, the algorithm determines a new residual network. The algorithm unlabels all the nodes, and repeats the process. The algorithm eventually terminates when it has scanned all the labeled nodes and it was not successful in labeling the sink node t. The above algorithm, along with this procedure to find the augmenting paths is listed below.

Recall that the vertex i is the predecessor of vertex j if the arc $(i, j) \in E$. This is denoted by predecessor$(j) = i$.

Algorithm. *Max-Flow Min-Cut Labeling Algorithm.*

Input: A flow network $N = (G, u, s, t)$. The capacity matrix is U.
Output: Maximum flow x.
begin
    $x \leftarrow 0$
    $r \leftarrow U$
    label node t
    while (node t is labeled) **do**
    begin
        unlabel all nodes in the network
        label node s
        $L \leftarrow \{s\}$
        predecessor$(j) \leftarrow \varnothing$ for all $j \in V$
        while $((L \neq \varnothing) \wedge (t$ is unlabeled$))$ **do**
        begin
            (*remove a node i from the set L*)
            $L \leftarrow L - \{i\}$
            for (all arcs (i, j) in the residual network $G(x)$) **do**
            begin
                if $(r_{ij} > 0$ and node j is unlabeled$)$ **then**
                begin
                    predecessor$(j) \leftarrow i$
                    label node j
                    $L \leftarrow L \cup \{j\}$
                end (*end if condition*)
            end (*end for loop*)
        end (*end while loop*)
        if (t is labeled) **then**
        begin
            (*augmentation of the flow*)

determine the augmenting path P in $G(x)$ from s to t
by using the predecessor table
$\theta \leftarrow \min\{r_{ij} \mid (i,j) \in P\}$
augment the flow along the path P by θ units
update $G(x)$
end (*end if condition*)
end (*end while loop*)
find the maximal flow from the final residual network $G(x)$
end (*end of max-flow min-cut labeling algorithm*)

The correctness of the above algorithm is next established. At the end of each iteration, the above algorithm either terminates or finds an augmenting path. If the algorithm terminates, it has to be proved that a maximum flow x has been determined. At the end of the algorithm, denote the labeled nodes as S and the unlabeled nodes as T, which is equal to $(V - S)$. Observe that $s \in S$ and $t \in T$. The algorithm fails to label any node in T from any node in S. This implies $r_{ij} = 0$ for all pairs $(i,j) \in S \times T$. Also $r_{ij} = (u_{ij} - x_{ij}) + x_{ji}$; and $(u_{ij} - x_{ij}) \geq 0$ and $x_{ji} \geq 0$ implies $u_{ij} = x_{ij}$ for all arcs $(i,j) \in S \times T$ and $x_{ij} = 0$ for all $(i,j) \in T \times S$. Therefore

$$v = \sum_{(i,j)\in(S,T)} x_{ij} - \sum_{(i,j)\in(T,S)} x_{ij} = \sum_{(i,j)\in(S,T)} u_{ij} = u\langle S,T\rangle$$

That is the value of the flow v is equal to the capacity of the cut $\langle S,T\rangle$. But it is known that $v \leq u\langle S,T\rangle$ for all cuts, therefore v is the maximum flow, and $\langle S,T\rangle$ is the minimum cut. This algorithm can also be regarded as a constructive proof of the max-flow min-cut theorem with integer capacities. It should be noted that the labeling algorithm may not terminate if the capacities are irrational numbers. However the max-flow min-cut algorithm is valid for irrational capacities. Several other superior algorithms exist in literature to determine maximum flow in a network. These are not discussed in this textbook.

Complexity of the Labeling Algorithm

Observe that in each path augmentation step, the algorithm scans any edge or node at most a single time. Therefore each path augmentation step requires $O(m)$ steps, where m is equal to the number of edges in the graph. The maximum flow is bounded by $nu_{\max}$, where n is equal to the number of vertices in the graph, and $u_{\max} = \max_{(i,j)\in E}\{u_{ij}\}$. If the network capacities are integers, the algorithm increments the value of the flow by at least a single unit. Therefore the maximum number of augmentations is $nu_{\max}$. Thus the computational complexity of the algorithm for integral values of the edge capacities is $O(nmu_{\max})$. The above labeling algorithm is illustrated below via a simple example.

It should be noted that for nonintegral values of the capacities, the algorithm can sometimes run forever. However, the max-flow min-cut theorem is still true.

Example. Consider a graph $G = (V,E)$, where $V = \{1,2,3,4\}$ and the edges are specified by the adjacency matrix A. The capacities of the arcs are specified in the capacity matrix U. The source and sink nodes are 1 and 3 respectively. Maximum flow in the graph G is determined.

$$A = \begin{bmatrix} 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}, \quad U = \begin{bmatrix} 0 & 4 & 0 & 7 \\ 0 & 0 & 8 & 3 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 5 & 0 \end{bmatrix}$$

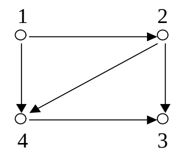

Figure 2. Maximum flow computation for the graph $G = (V, E)$.

See Figure 2. This network has four cuts. Their capacities are listed in Table 4.1. It can be observed that the cut with the minimum capacity is $\langle S, T \rangle$, where $S = \{1, 4\}$, $T = \{2, 3\}$, and its capacity is $u \langle S, T \rangle = 9$. Therefore the value of the maximum flow is equal to 9. This can indeed be checked by following the algorithm. The iteration numbers are with respect to the outer while-loop of the labeling algorithm.

S	T	$u \langle S, T \rangle$
$\{1\}$	$\{2, 3, 4\}$	11
$\{1, 2\}$	$\{3, 4\}$	18
$\{1, 4\}$	$\{2, 3\}$	9
$\{1, 2, 4\}$	$\{3\}$	13

Table 4.1. s-t cuts and their capacities.

Initialization: $x \leftarrow 0, r \leftarrow U$, and label the sink node 3.

Iteration 1: The path $P = (1, 2, 3)$. Also $\theta = \min \{r_{12}, r_{23}\} = \min \{4, 8\} = 4$. Therefore $x_{12} = x_{23} = 4$. The values $x_{24} = x_{14} = x_{43} = 0$ remain unchanged. The capacities of the residual network are $r_{12} = 0$, $r_{21} = 4$, $r_{23} = 4$, and $r_{32} = 4$. The capacities $r_{24} = 3$, $r_{14} = 7$, and $r_{43} = 5$ remain unchanged.

Iteration 2: The path $P = (1, 4, 3)$. Also $\theta = \min \{r_{14}, r_{43}\} = \min \{7, 5\} = 5$. Therefore $x_{14} = x_{43} = 5$. The values $x_{12} = x_{23} = 4$, and $x_{24} = 0$ remain unchanged. The capacities of the residual network are $r_{14} = 2$, $r_{41} = 5$, $r_{43} = 0$, and $r_{34} = 5$. The capacities $r_{12} = 0$, $r_{21} = 4$, $r_{23} = 4$, $r_{32} = 4$, and $r_{24} = 3$ remain unchanged.

Iteration 3: No complete path exists from the source node 1 to sink node 3. Therefore the algorithm terminates. Thus the flow x is

$$x_{12} = 4, \ x_{23} = 4, \ x_{24} = 0, \ x_{14} = 5, \text{ and } x_{43} = 5$$

The value of the maximal flow $v = (x_{12} + x_{14}) = (x_{23} + x_{43}) = 9$. □

Extensions of the Single-Source and Single-Sink Maximum Flow Problem

1. *A network with multiple sources and sinks*: Assume that a graph has n_s sources and n_t sinks. The problem is to find maximum flow from all sources to all sinks. This

problem can be converted to a single source and a single sink problem by adding a master source node and a master sink node.

A direct edge is added from the master source node to each source node. The capacity of each of these links is set to infinity, or equal to the finite value if the supply at that source is limited.

Also a direct edge is added from each of the original sink nodes to the master sink node. The capacity of each of these links is set to infinity, or equal to the finite value if the demand at the sink is limited.

2. *Graphs with both arc and vertex capacities*: In a single source and a single sink graph, the arcs and the nodes have finite capacities. The total flow entering a node $i \in V$ has a capacity w_i for all $1 \leq i \leq n$. This problem can be converted to a capacitated single-source single-sink maximum flow problem.

   Replace each interior node i (nodes other than the source and sink nodes) by a pair of nodes i' and i'', and an arc (i'', i'). The nodes i' and i'' are the called the *out-node* and the *in-node* respectively. The capacity of the arc (i'', i') is equal to w_i. Furthermore, in the modified network the capacity of the arc (i', j'') is equal to u_{ij}.

3. *Flows with nonnegative lower and upper bounds*: The problem with nonnegative lower and upper bound can also be studied using the techniques discussed in this section. Observe that the maximum flow problem always has a feasible solution. However if there is a restriction on the lowest value of flow on the arcs, then it is not always possible to obtain a feasible solution.

   This problem is not discussed any further. It is addressed extensively in books on network flows and optimization.

Combinatorial Application of Max-Flow and Min-Cut Theorem

Some problems in network connectivity can be studied using the max-flow min-cut theorem. Two directed paths from the source node s to the sink node t are said to be *arc-disjoint* if they do not have any arc in common. *Node-disjoint* paths are defined similarly. Two paths are node-disjoint if they do not have any common nodes, except the nodes s and t. The following observations can be made about these applications of max-flow min-cut theorem.

Observations.

1. The maximum number of arc-disjoint paths from the source node s to the sink node t, is equal to the minimum number of arcs required to be removed from the network to disconnect all paths from the node s to the node t.

   This result can be established by assigning a capacity of 1 to each arc in the network, and using the max-flow min-cut theorem.

2. Similarly, the maximum number of node-disjoint paths from the source node s to the sink node t, is equal to the minimum number of nodes required to be removed from the network to disconnect all paths from the node s to the node t. □

The proofs of the above observations are left to the reader.