Chapter 5

Maximum Flows

5.1 Introduction

The Maximum Flow Problem (MFP) is easy to state: in a capacitated network, we wish to send as much flow as possible between two special nodes, a source node s and a sink node t, without exceeding the capacity of any arc.

5.1.1 Notation

We consider a capacitated graph $G = (N, A)$ with a non-negative capacity u_{ij} associated with each arc $(i, j) \in A$. Let $\overline{u} = \max\{u_{ij} \mid (i, j) \in A\}$. Moreover, let $\delta_i^+ \subseteq N$ ($\delta_i^- \subseteq N$, respectively) the set of successors (predecessors, resp.) of node $i \in N$, i.e., $\delta_i^+ = \{j \in N \mid (i, j) \in A\}$ ($\delta_i^- = \{j \in N \mid (j, i) \in A\}$, resp.). To define the MFP, we distinguish two special nodes in the graph G: a source node s and a sink node t. We wish to find the maximum flow from s to t that satisfies the arc capacities and mass balance constraints at all nodes. We can formally state the problem formally as

$$
\begin{aligned}
\max \quad & v && && (5.1a)\\
\text{s.t.} \sum_{j \in \delta_s^+} x_{sj} - \sum_{i \in \delta_s^-} x_{is} &= v && && (5.1b)\\
\sum_{j \in \delta_k^+} x_{kj} - \sum_{i \in \delta_k^-} x_{ik} &= 0 && \forall k \in N \setminus \{s, t\} && (5.1c)\\
\sum_{j \in \delta_t^+} x_{tj} - \sum_{i \in \delta_t^-} x_{it} &= -v && && (5.1d)\\
0 \leq x_{ij} &\leq u_{ij} && \forall (i, j) \in A && (5.1e)
\end{aligned}
$$

We refer to a vector $\boldsymbol{x} \in \mathbb{R}_+^{|A|}$ satisfying constraints (5.1b)-(5.1e) as a *flow* and the corresponding value of v as the *value of the flow*.

5.1.2 Assumptions

We consider the MFP subject to the following assumptions:

Assumption 5.1. *The graph is directed.*

We can always fulfill this assumption by transforming a undirected graph into a directed graph.

Assumption 5.2. *All capacities are non-negative integers.*

The integrality assumption is not restrictive because all computers store capacities as rational numbers and we can always transform rational numbers to integer numbers by multiplying them by a suitably large number.

Assumption 5.3. *The graph does not contain a directed path from node s to t composed only of infinite capacity arcs.*

Whenever every arc on a directed path P from s to t has infinite capacity, the maximum flow value is unbounded.

Assumption 5.4. *Whenever an arc (i, j) belongs to A, arc (j, i) also belongs to A.*

This assumption is nonrestrictive because we allow arcs with zero capacity.

5.2 Applications

The MFP arises in a wide variety of situations. We describe a few such applications.

5.2.1 Feasible Flow Problem

The feasible flow problem requires that we identify a flow $\boldsymbol{x}$ in a graph $G = (N, A)$ satisfying the following constraints

$$\sum_{j\in\delta_k^+} x_{kj} - \sum_{i\in\delta_k^-} x_{ik} = b_k \quad \forall k \in N \tag{5.2a}$$

$$0 \leq x_{ij} \leq u_{ij} \qquad \forall (i, j) \in A \tag{5.2b}$$

where we assume that $\sum_{k\in N} b_k = 0$.

This problem arises, for example, in distribution. Suppose that merchandise is available at some seaports and is desired by other ports. We know the stock of merchandise available at the ports, the amount required at the other ports, and the maximum quantity of merchandise that can be shipped on a particular sea route. We wish to know if we can satisfy all of the demands with the supplies.

We can solve the feasible flow problem by solving a MFP defined on an augmented network as follows. We introduce two new nodes, a source node s and a sink node t. For each node i with $b_i > 0$, we add an arc (s, i) with capacity b_i, and for each node i with $b_i < 0$, we add an arc (i, t) with capacity $-b_i$. We solve a MFP from s to t in this augmented network. If the maximum flow saturates all the source and sink arcs, the feasible flow problem has a feasible solution; otherwise, it is infeasible.

5.2.2 Problem of Representatives

A town has r residents $R_1, R_2, \dots, R_r$; q clubs $C_1, C_2, \dots, C_q$; and p political parties $P_1, P_2, \dots, P_p$. Each resident is a member of at least one club and belongs to exactly one political party. Each club must nominate one of its members to represent it on the town's governing council so that the number of council members belonging to the political party P_k is at most u_k. Is it possible to find a council that satisfies this balancing property?

We illustrate this formulation with an example with $r = 7$, $q = 4$, and $p = 3$. We formulate it as a MFP (see Figure 5.1). The nodes $R_1, R_2, \dots, R_7$ represent the residents, the nodes C_1, C_2, C_3, C_4 the clubs, and the nodes P_1, P_2, P_3 the political parties.

The graph also contains a source node s and a sink node t. It contains an arc (s, C_i) for each club C_i, an arc (C_i, R_j) whenever the resident R_j is a member of the club C_i, and an arc (R_j, P_k) if the resident R_j belongs to the political party P_k. Finally, we add an arc (P_k, t) for each party P_k of capacity u_k; all other arcs have unit capacity. We find a maximum flow in this graph. If the maximum flow value equals q, the town has a balanced council; otherwise, it does not.

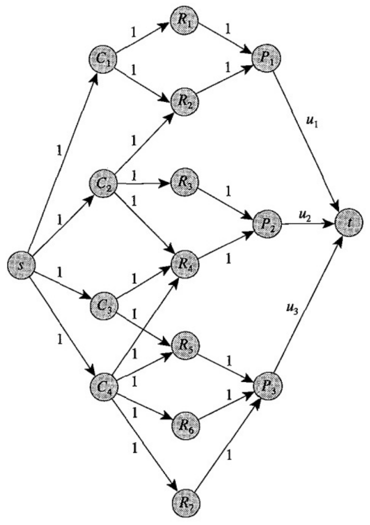


Figure 5.1: Problem of representatives

Exercise 5.1.

Several families go out to dinner together. To increase their social interaction, they would like to sit at tables so that no two members of the same family are at the same table. Show how to formulate finding a seating arrangement that meets this objective as a MFP. Assume that the dinner contingent has p families and that the ith family has a_i members. Also assume that q tables are available and that the jth table has a seating capacity of b_j.

Construct the graph $G = (N_1 \cup N_2 \cup \{s, t\}, A)$, where N_1 contains p nodes (one per family) and N_2 contains q nodes (one per table). The source node s is connected to each node $i \in N_1$ by an arc (s, i) of capacity a_i. Each node $j \in N_2$ is connected to the sink node t by an arc (j, t) of capacity b_j. An arc (i, j) of unit capacity exists between every pair of nodes (i, j) such that $i \in N_1$ and $j \in N_2$. A feasible seating arrangement exists if and only if all the arcs emanating from node s are saturated in a maximum flow from s to t. Such a maximum flow defines a feasible seating arrangement in which a person from family i is to sit on table j if and only if arc (i, j) carries unit flow.

Exercise 5.2.

We are given a directed graph $G = (N, A)$. The set N consists of two sets V and W. Each node $i \in V$ has a supply $a_i \in \mathbb{Z}_+$, and each node $j \in W$ has a demand $b_j \in \mathbb{Z}_+$ – we assume that $\sum_{i \in V} a_i = \sum_{j \in W} b_j$. We can send a flow from each node $i \in V$ to each node $j \in W$, i.e., $A = \{(i, j) \mid i \in V, j \in W\}$. The unit cost to send a flow from i to j is c_{ij}. We wish to find a flow $\boldsymbol{x}$ from the nodes of the set V to the nodes of the set W such that $\max\{c_{ij} x_{ij} \mid (i, j) \in A\} \leq \lambda$, where λ is a given parameter. Show how to formulate this problem as a MFP

We solve a MFP on a directed graph $G' = (N', A')$ defined as follows. The node set N' contains the sets V and W plus two dummy nodes s and t, i.e., $N' = V \cup W \cup \{s, t\}$. The arc set is defined as $A' = A^s \cup A \cup A^t$, where $A^s = \{(s, i) \mid i \in V\}$ and $A^t = \{(j, t) \mid j \in W\}$. The arc capacities are defined as follows: (i) for each $(s, i) \in A^s$, $u_{si} = a_i$; (ii) for each $(i, j) \in A$, $u_{ij} = \lfloor \frac{\lambda}{c_{ij}} \rfloor$; (iii) for each $(j, t) \in A^t$, $u_{jt} = b_j$.

A feasible solution to the problem exists if and only if the maximum flow from s to t has a value $v = \sum_{i \in V} a_i$. Otherwise, no feasible solution such that $\max\{c_{ij} x_{ij} \mid (i, j) \in A\} \leq \lambda$ exists.

5.3 Flows and Cuts

In this section, we discuss some elementary properties of flows and cuts. We use these properties to prove the max-flow min-cut theorem to establish the correctness of the augmenting path algorithm.

Residual Graph Given a flow $\boldsymbol{x}$, the residual capacity r_{ij} of any arc $(i, j) \in A$ is the maximum additional flow that can be sent from i to j using the arcs (i, j) and (j, i). The residual capacity r_{ij} has two components: (i) $u_{ij} - x_{ij}$, the unused capacity of arc (i, j), and (ii) the current flow x_{ji} on arc (j, i), which we can cancel to increase the flow from i to j. Consequently, $r_{ij} = u_{ij} - x_{ij} + x_{ji}$. We refer to the graph $G(\boldsymbol{x})$ consisting of the arcs with positive residual capacities as the *residual graph* (with respect to the flow $\boldsymbol{x}$). Figure 5.2 gives an example of a residual graph

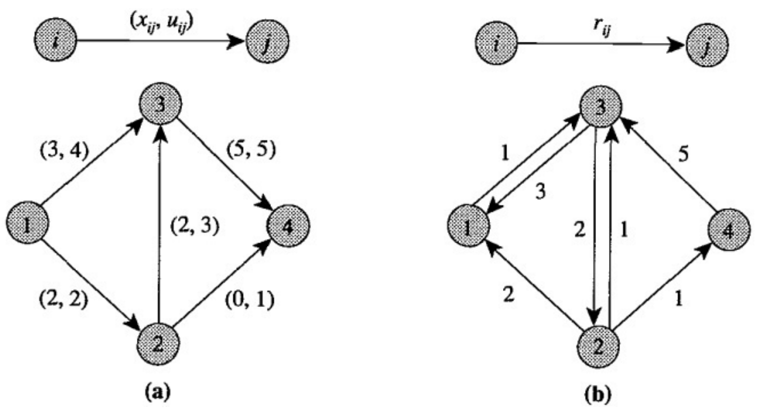


Figure 5.2: A residual graph: (a) original graph G with a flow $\boldsymbol{x}$; (b) residual graph $G(\boldsymbol{x})$

$s - t$ cut A cut is a partition of the node set N into two subsets S and $\overline{S} = N \setminus S$; we represent this cut with $(S, \overline{S})$. We refer to a cut as an $s - t$ cut if $s \in S$ and $t \in \overline{S}$. A forward arc (i, j) of the cut is such that $i \in S$ and $j \in \overline{S}$, and a backward arc (i, j) of the cut is such that $i \in \overline{S}$ and $j \in S$. Let $A(S, \overline{S})$ denote the set of forward arcs in the cut, and let $A(\overline{S}, S)$ denote the set of backward arcs. For example, in Figure 5.3, the dashed arcs constitute an $s - t$ cut, where $S = \{1, 3, 5\}$, $\overline{S} = \{2, 4, 6\}$, $s = 1$, $t = 6$, $A(S, \overline{S}) = \{(1, 2), (3, 4), (5, 6)\}$, and $A(\overline{S}, S) = \{(2, 3), (4, 5)\}$

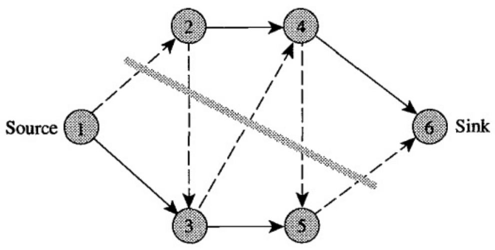


Figure 5.3: Example of an $s - t$ cut

Capacity of an $s - t$ cut The capacity $u(S, \overline{S})$ of an $s - t$ cut $(S, \overline{S})$ is the sum of the capacities of the forward arcs in the cut, i.e., $u(S, \overline{S}) = \sum_{(i,j)\in A(S,\overline{S})} u_{ij}$.

Minimum cut An $s - t$ cut whose capacity is minimum among all $s - t$ cuts is a *minimum cut*

Residual capacity of an $s - t$ cut The residual capacity $r(S, \overline{S})$ of an $s - t$ cut $(S, \overline{S})$ is the sum of the residual capacities of forward arcs in the cut

Flow across an $s - t$ cut Let $\boldsymbol{x}$ be a flow in the graph. Adding the mass balance constraint (5.1b)-(5.1d) for the nodes in S, we have

$$v = \sum_{i\in S} \Big(\sum_{(i,j)\in A} x_{ij} - \sum_{(j,i)\in A} x_{ji} \Big) = \sum_{(i,j)\in A(S,\overline{S})} x_{ij} - \sum_{(j,i)\in A(\overline{S},S)} x_{ji} \le \sum_{(i,j)\in A(S,\overline{S})} u_{ij} = u(S, \overline{S}) \tag{5.3}$$

Thus, the value of any flow is less than or equal to the capacity of any $s - t$ cut: indeed, any flow from s to t must pass through every $s - t$ cut in the graph

Property 5.1. *The value of any flow is less than or equal to the capacity of any $s - t$ cut in the graph.*

If we discover a flow $\boldsymbol{x}$ whose value equals the capacity of some $s - t$ cut $(S, \overline{S})$, then $\boldsymbol{x}$ is a maximum flow and the cut $(S, \overline{S})$ is a minimum cut.

Suppose that $\boldsymbol{x}$ is a flow of value v and that $\boldsymbol{x}'$ is a flow of value $v + \Delta v$ for some $\Delta v \ge 0$. The inequality (5.3) implies that

$$v + \Delta v \le \sum_{(i,j)\in A(S,\overline{S})} u_{ij} \quad \Longrightarrow \quad \Delta v \le \sum_{(i,j)\in A(S,\overline{S})} (u_{ij} - x_{ij}) + \sum_{(j,i)\in A(\overline{S},S)} x_{ij} = \sum_{(i,j)\in A(S,\overline{S})} (u_{ij} - x_{ij} + x_{ji}) = \sum_{(i,j)\in A(S,\overline{S})} r_{ij}$$

Property 5.2. *For any flow $\boldsymbol{x}$ of value v, the additional flow that can be sent from s to t is less than or equal to the residual capacity of any $s - t$ cut.*

5.4 Generic Augmenting Path Algorithm

In this section, we describe one of the simplest algorithms, known as the *augmenting path algorithm*, for solving the MFP. We refer to a directed path from the source to the sink in the residual graph as an *augmenting path*. We define the *residual capacity* of an augmenting path as the minimum residual capacity of any arc in the path, which is always positive by definition. Whenever the graph contains an augmenting path, we can send additional flow from the source to the sink. The algorithm (see Algorithm 3) proceeds by identifying augmenting paths and augmenting flows on these paths until the graph contains no such path.

Algorithm 3: Augmenting Path Algorithm (Ford-Fulkerson Algorithm)

```
1 $x \leftarrow 0$;
2 while $G(x)$ contains a directed path from $s$ to $t$ do
3     Identify an augmenting path $P$ from $s$ to $t$;
4     $\delta \leftarrow \min\{r_{ij} \mid (i,j) \in P\}$;
5     Augment $\delta$ units of flow along $P$ and update $G(x)$;
```

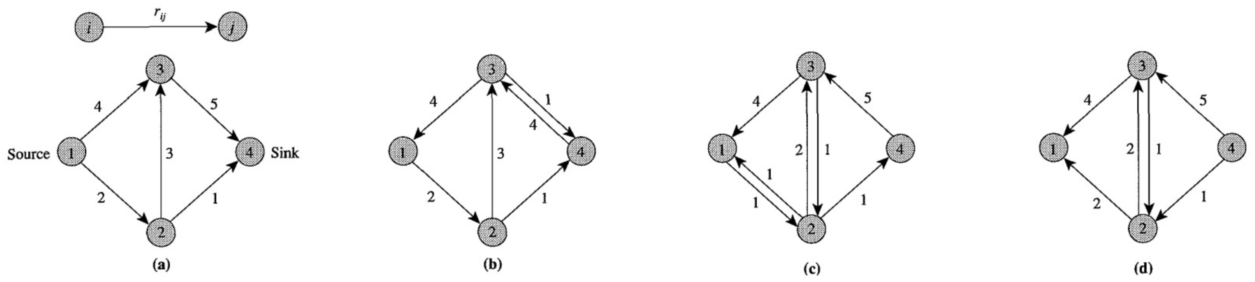


Figure 5.4: Illustrating the generic augmenting path algorithm: (a) residual graph for the zero flow; (b) graph after augmenting four units along 1 – 3 – 4; (c) graph after augmenting one unit along 1 – 2 – 3 – 4; (d) graph after augmenting one unit along 1 – 2 – 4

5.4.1 Relationship between the Original and Residual Graphs

Suppose that we find an augmenting path P in the residual graph and send δ units of flow through P. What is the effect on the arc flows x_{ij}? The definition of the residual capacity (i.e., $r_{ij} = u_{ij} - x_{ij} + x_{ji}$) implies that an additional flow of δ units on arc (i, j) in the residual graph corresponds to (i) an increase in x_{ij} by δ units in the original graph, or (ii) a decrease in x_{ji} by δ units in the original graph.

Consider Figure 5.5(a) and the corresponding residual graph of Figure 5.5(b). Augmenting 1 unit of flow on the path 1 – 2 – 4 – 3 – 5 – 6 produces the residual graph in Figure 5.5(c) with the corresponding arc flows shown in Figure 5.5(d). Comparing the solution in Figure 5.5(d) with that in Figure 5.5(a), we find that the flow augmentation increases the flow on arcs (1, 2), (2, 4), (3, 5), (5, 6), and decreases the flow on arc (3, 4).

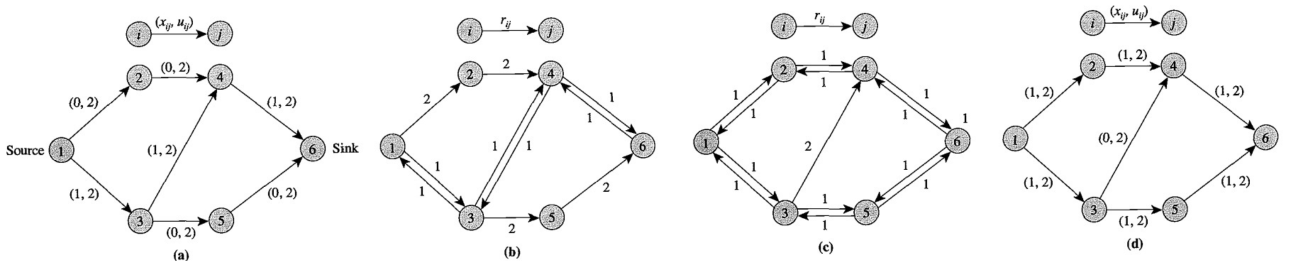


Figure 5.5: (a) original graph with a flow x; (b) residual graph for flow x; (c) residual graph after augmenting one unit along 1 – 2 – 4 – 3 – 5 – 6; (d) flow in the original graph after the augmentation

5.4.2 Effect of Augmentation on Flow Decomposition

Let us illustrate the effect of an augmentation on the flows of the example of Figure 5.5. Figure 5.6(a) gives the decomposition of the initial flow, and Figure 5.6(b) gives the decomposition of the flow after augmenting one unit of flow on the path 1 – 2 – 4 – 3 – 5 – 6. Although we augmented one unit of flow along the path 1 – 2 – 4 – 3 – 5 – 6, the flow decomposition contains no such path.

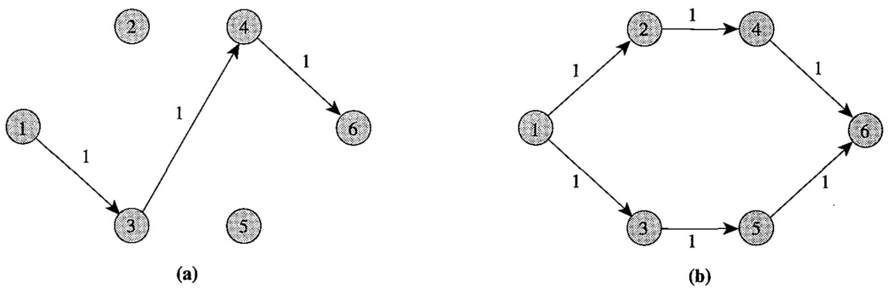


Figure 5.6: Flow decomposition of the solution in (a) Figure 5.5(a) and (b) Figure 5.5(d)

The path 1 – 3 – 4 – 6 defining the flow in Figure 5.5(a) contains three segments: the path up to node 3, arc (3, 4) as a forward arc, and the path up to node 6. We can view this path as an augmentation on the zero flow. Similarly, the path 1 – 2 – 4 – 3 – 5 – 6 contains three segments: the path up to node 4, arc (3, 4) as a backward arc, and the path up to node 6. We can view the augmentation on the path 1 – 2 – 4 – 3 – 5 – 6 as linking the initial segment of the path 1 – 3 – 4 – 6 with the last segment of the augmentation path, linking the last segment of the path 1 – 3 – 4 – 6 with the initial segment of the augmentation path, and canceling the flow on arc (3, 4), which then drops from both the path 1 – 3 – 4 – 6 and the augmentation path. In general, we can view each augmentation as "pasting together" segments of the current flow decomposition to obtain a new flow decomposition.

5.5 Labeling Algorithm and the Max-Flow Min-Cut Theorem

Now, we discuss the augmenting path algorithm in detail. So far, we did not discuss some important details, such as (i) how to identify an augmenting path or show that the graph contains no such path, and (ii) whether the algorithm terminates in finite number of iterations, and when it terminates, whether it has obtained a maximum flow. We consider these issues for a specific implementation of the generic augmenting path algorithm known as the *labeling algorithm.*

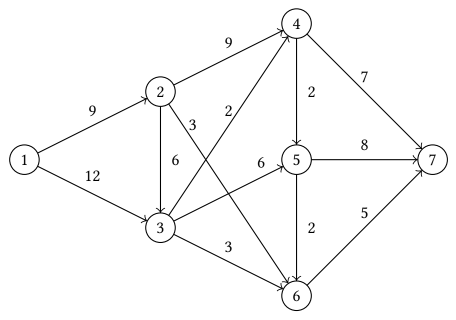
The labeling algorithm uses a search technique to identify a directed path in $G(\boldsymbol{x})$ from s to t. The algorithm fans out from s to find all nodes reachable from s along a directed path in the residual graph. At any step, the algorithm maintains two sets of nodes: the *labeled* nodes (L) and the *temporary* nodes (T). Labeled nodes are those nodes reached in the fanning out process, so the algorithm has determined a directed path from s to these nodes in the residual graph. The temporary nodes are those nodes that are examined in the fanning-out process. The algorithm iteratively selects a temporary node and scans its arc adjacency list (in the residual graph) to reach and label additional nodes. Eventually, t becomes labeled, and the algorithm sends the maximum possible flow on the path from s to t. It then erases the labels and repeats this process. The algorithm terminates when it has scanned all the labeled nodes and t remains unlabeled, implying that s is not connected to t in the residual graph. Algorithm 4 gives a step-by-step description of the labeling algorithm.

Algorithm 4: Labeling Algorithm (Ford-Fulkerson Algorithm)

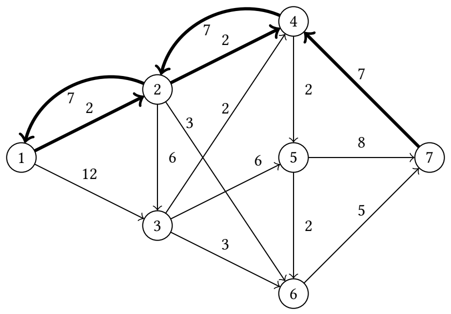
```
Set $\boldsymbol{x} \leftarrow \mathbf{0}$, $L \leftarrow \{t\}$, and create the residual graph;
while $t \in L$ do
    Set $L \leftarrow \{s\}$, $T \leftarrow \{s\}$, $pred(j) \leftarrow 0$ for each $j \in N$;
    // Identify an augmenting path
    while $T \neq \emptyset$ and $t \notin L$ do
        Remove a node $i$ from $T$;
        for each arc $(i, j)$ emanating from node i in the residual graph do
            if $r_{ij} > 0$ and node $j \notin L$ then
                Set $pred(j) \leftarrow i$, $L \leftarrow L \cup \{j\}$, $T \leftarrow T \cup \{j\}$;
    // Augmenting phase
    if $t \in L$ then
        Obtain an augmenting path $P$ from $s$ to $t$ by using the predecessor labels;
        Compute $\delta = \min\{r_{ij} : (i, j) \in P\}$;
        Augment $\delta$ units of flow along $P$, and update the residual capacities;
```

To better understand the labeling algorithm, consider the following graph, where the label close to each arc is the corresponding capacity, u_{ij}. We search for the maximum flow we can send from source 1 to sink 7. We describe the steps the algorithm goes through in detail.

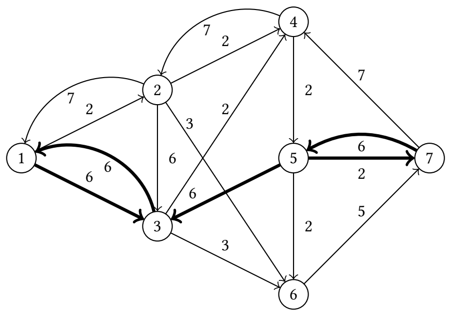
1. As the initial flow is set to 0 in every arc, this graph also corresponds to the residual graph.



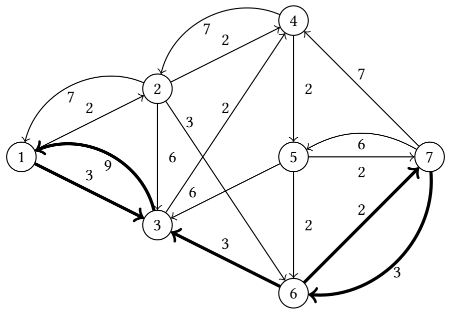
2. Let us assume that, in the first iteration, the algorithm identifies the augmenting path 1 – 2 – 4 – 7 with an augmenting flow of 7. The corresponding residual graph is



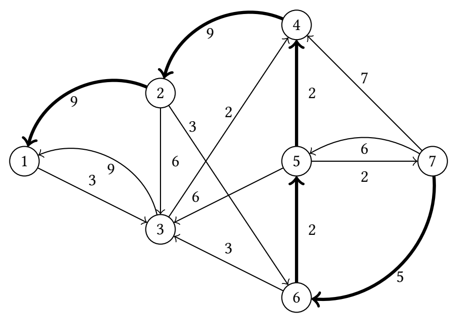
3. We now assume that, in the second iteration, the algorithm identifies the augmenting path 1 – 3 – 5 – 7 with an augmenting flow of 6. The corresponding residual graph is



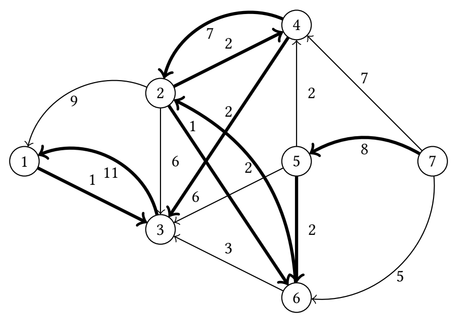
4. In the next iteration, the algorithm can find the augmenting path 1 – 3 – 6 – 7 with an augmenting flow of 3. The corresponding residual graph is



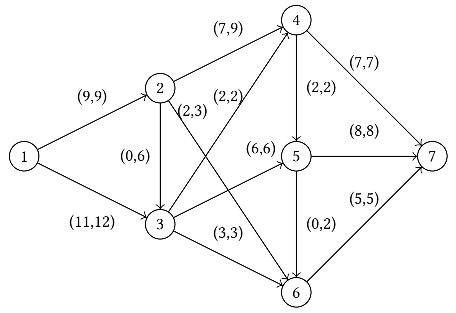
5. In the next iteration, the algorithm can find the augmenting path 1 – 2 – 4 – 5 – 6 – 7 with an augmenting flow of 2. The corresponding residual graph is



6. Then, the algorithm can find the augmenting path 1 − 3 − 4 − 2 − 6 − 5 − 7 with an augmenting flow of 2. The corresponding residual graph is



This is the final residual graph. The algorithm finds out that the maximum flow from node 1 to node 7 is 20. In particular: 7 units go through path 1 − 2 − 4 − 7, 2 units go through path 1 − 2 − 6 − 7, 3 units go through path 1 − 3 − 6 − 7, 6 units go through path 1 − 3 − 5 − 7, and 2 units go through path 1 − 3 − 4 − 5 − 7. This residual graph corresponds to the following solution, where the label close to each arc indicates the flow and the capacity, i.e., (x_{ij}, u_{ij}).



Note that in each iteration of the labeling algorithm, the algorithm either performs an augmentation or terminates because it cannot label the sink. In the latter case, we must show that the current flow $\boldsymbol{x}$ is a maximum flow. Suppose, at this stage, that S is the set of labeled nodes (i.e., $S = L$) and $\overline{S} = N \setminus S$ is the set of unlabeled nodes. Clearly, $s \in S$ and $t \in \overline{S}$. Since the algorithm cannot label any node in $\overline{S}$ from any node in S, $r_{ij} = 0$ for each $(i, j) \in A(S, \overline{S})$. Furthermore, since $r_{ij} = (u_{ij} - x_{ij}) + x_{ji}$, $x_{ij} \leq u_{ij}$ and $x_{ji} \geq 0$, the condition $r_{ij} = 0$ implies that $x_{ij} = u_{ij}$ for every arc $(i, j) \in A(S, \overline{S})$ and $x_{ji} = 0$ for every arc $(j, i) \in (\overline{S}, S)$. Substituting these flow values in (5.3), we find that

$$v = \sum_{(i,j)\in A(S,\overline{S})} x_{ij} - \sum_{(j,i)\in A(\overline{S},S)} x_{ji} = \sum_{(i,j)\in A(S,\overline{S})} u_{ij} = u(S, \overline{S})$$

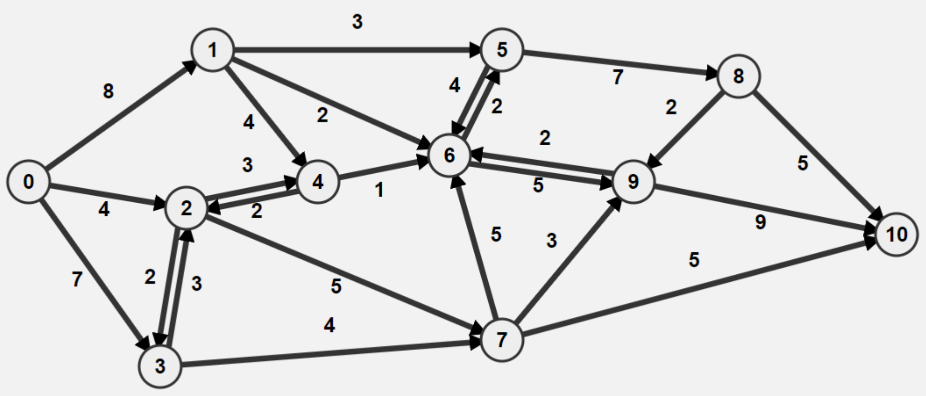
which shows that the value of the current flow $\boldsymbol{x}$ equals the capacity of the $s-t$ cut $(S, \overline{S})$. Therefore, $\boldsymbol{x}$ is a maximum flow, and $(S, \overline{S})$ is a minimum $s-t$ cut.

Theorem 5.1. ***(Max-Flow Min-Cut Theorem)*** *The maximum value of the flow from a source node s to a sink node t in a capacitated graph equals the minimum capacity among all $s-t$ cuts.*

Theorem 5.2. ***(Augmenting Path Theorem)*** *A flow $\boldsymbol{x}^*$ is a maximum flow if and only if the residual graph $G(\boldsymbol{x}^*)$ contains no augmenting path.*

Exercise 5.3.

By applying the labeling algorithm, determine a maximum flow from node $s = 0$ to node $t = 10$ of the following graph. What is the maximum flow? What is the corresponding minimum cut?



The maximum flow has a value of $v = 15$ and corresponds to the $s-t$ cut $(S, \overline{S})$, $S = \{0, 1, 2, 3, 4\}$, $\overline{S} = \{5, 6, 7, 8, 9, 10\}$.

Exercise 5.4.

In some graphs $G = (N, A)$, in addition to arc capacities, each node $i \in N$ might have an upper bound (say w_i) on the flow that can pass through it. In these graphs, we are interested in determining the maximum flow satisfying both the arc and node capacities. Transform this problem to the standard MFP.

Let $G = (N, A)$ be a graph having node capacities. Construct the graph $G' = (N', A')$ in which every node $i \in N$ is replaced by two nodes $i', i'' \in N$. Furthermore, the arc set A' is defined as $A' = \{(i'', j') \,|\, (i, j) \in A\} \cup \{(i', i'') \,|\, i \in N\}$. Each arc $(i', i'') \in A$ has a capacity of w_i. A maximum flow in G' is equal to a maximum flow in G.