MAXIMUM FLOW PROBLEM

COLLEGE OF NATURAL AND COMPUTATIONAL SCIENCE

DEPARTMENT OF MATHEMATICS

Thesis Submitted in Partial Fulfillment of the Requirement for the Degree of Master of Science in Mathematics

(Optimization Stream)

By: Asnake Abrham

Advisor: Berhanu Guta (PhD)

September, 2022

Addis Ababa, Ethiopia

Department of Mathematics

College of Natural Sciences

Addis Ababa University

This is to certify that, the thesis prepared by: Asnake Abrham Getaneh, entitled: "Maximum Flow" and submitted in partial fulfillment for the requirements of the Degree of Master of Science in Mathematics(Optimization) complies with the regulations of the University and meets the accepted standards with respect to originality and quality.

Signed by the Examining Committee

Examiner: 1	Signature	Date
Examiner: 2	Signature	Date
Berhanu Guta(PhD)		
Advisor	Signature	Date

Chair of Department or Graduate Program Coordinator

Acknowledgment

First of all, I would like to thank to my almighty God for everything is possible. Next my heartfelt and best thanks are gladly given to my advisor and instructor, Dr Berhanu Guta, for his invaluable guidance and advice. Also, let me take this opportunity to express my deep sense of gratitude to all instructors in mathematics department. At last but not least, I would like to express my immense gratitude to my parents who are always the source of my motivation and encouragement.

Abstract

In this thesis, we address about the concept of maximum flow problem and the solution methods to solve the problem. In this work the Generic Augmenting Path Algorithm and Ford-Fulkerson Labeling method are presented to find the maximum flow in a network flow problem.

Keywords: Network flows, Maximum Flow Problems, Max-Flow Min-Cut Theorem, Generic Augmenting Path Algorithm, and Ford-Fulkerson Labeling Method.

Table of contents

Contents **Page**

CHAPTER-1

1. Introduction

Network flow problems are central problems in operation research, computer science, and engineering and they arise in many real world applications. Everywhere we look in our daily lives, networks are apparent. Highways, telephone lines, electric power systems, computer chips, water delivery systems, and rail lines; these physical networks, and many others are familiar to all of us. In each of these problem settings, we often wish to send some good(s) (a vehicle or a person, a message, electricity, or water), which we generally call flow, from one point to another typically as efficiently as possible, subject to certain constraint..

Networks consist of special points called nodes and links connecting pairs of nodes called arcs. The maximum flow problem is one of the most fundamental problems in network flow theory and has been investigated extensively. In this thesis we are going to deal with the maximum flow problem and the solution methods to solve it.

Maximum flow problem: Mathematical Formulation

Definition 1.1 ($NA(i)$ *and* $NB(i)$ **)**

Let us define $NA(i) \equiv$ **the set of all nodes just after node *i*.**

$NB(i) \equiv$ **the set of all nodes just before node *i*.**

Given a directed capacitated network $G = (N, A)$ with a single source node (s) and a single sink node(t). We want to formulate the max-flow problem.

- Let f represent the amount of flow in the network from the source node (s) to the sink node (t).
- For each arc $e = (i, j) \in A$, let $f_{ij} \geq 0$ denote the flow sent on arc (i, j)
- For each arc $e = (i, j) \in A$, the flow is bounded from above by the capacity c_{ij} of the arc: $0 \leq f_{ij} \leq c_{ij}$.

- We have to specify the balance equations.
  All the nodes in the network except for the source and the sink node are just "transit" nodes (flow in=flow out)

$$\sum_{k \in NB(i)} f_{ki} - \sum_{j \in NA(i)} f_{ij} = 0 \quad ; \forall\, i \in N \backslash \{s,t\}$$

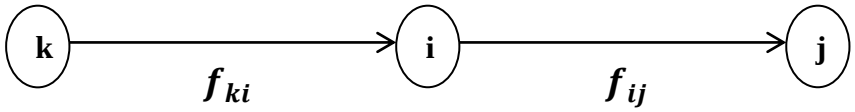

Figure1.1

- The objective is to maximize the out flow(f) from the source node s.

$$f = \sum_{j \in NA(s)} f_{sj}$$

- Alternatively; to maximize the inflow to the sink node t

$$f = \sum_{i \in NB(t)} f_{it}$$

Max-Flow Formulation

$$maximize\ f = \sum_{j \in NA(s)} f_{sj}$$

$$subject\ to \sum_{k\ \in NB(i)} f_{ki} - \sum_{j \in NA(i)} f_{ij} = 0 \ ; \forall\, i \in N \backslash \{s,t\}$$

$$0 \leq f_{ij} \leq c_{ij} \ ; \forall e = (i,j) \in A$$

In all parts of this thesis we assume that the data (flows and arc capacities) are integral.

CHAPTER-2

2. Preliminary Concepts

In this section, we introduce some basic notations and definitions from the graph theory as well as mathematical programing formulations of the maximum flow problem.

2.1. Graphs and Network Flows

Definition 2.1 (Graph)

Graph: A Graph G= (N, A) consists of a non-empty finite set N of nodes (also called vertices) and a finite set A of arcs (also called edges), where each edges links a vertex to a vertex.

Definition 2.2 (Directed Graph)

Directed Graph: A directed graph G= (N, A) is a graph, where all the arcs are directed from one vertex to another. A directed graph sometimes called digraph. When drawing a directed graph, the edges are typically drawn as arrows indicating the directions, as illustrated in figure 2.1 below.

Definition 2.3 (Flow Network)

Flow Network*: A network is a directed weighted graph G= (N, A), where N is the set of* m *nodes and A is the set* n *of directed arcs over these nodes, with the following features.*

- *Each arc* $(i, j) \in A$ *has a nonnegative numerical value, called arc capacity,* c_{ij} *denoting the maximum flow that can be send on that arc.*
- *There is one vertex with no incoming edges, called the source nods*(s).
- *There is one vertex with no outgoing edges, called the sink node*(t) . *Nodes other than s and t are called intermediate nodes.*

Definition 2.4 (s-t flow)

Let $G = (N, A)$ *is a flow network with source node* (s) *and sink node*(t).

An s-t flow is a function $f: A \to R^{+} \cup \{0\}$ *that satisfies*

i. $0 \le f_{ij} \le c_{ij}\ ;\ \forall e = (i,j) \in A$ *(capacity constraint)*
Meaning no arc gets a flow that exceeds its capacity.

ii. $\sum_{k \in NB(i)} f_{ki} - \sum_{j \in NA(i)} f_{ij} = 0\ ;\ \forall i \in N \setminus \{s,t\}$ $(flow\ conservation)$
The flow going into node i *equals the flow coming out of node* i.

A flow that satisfies (i) *and* (ii) *is said to be a feasible flow.*

Definition 2.5 (Flow Value)

The value of the flow, denoted by $val(f)$ *is the total value of the flow leaving the source node*(s)*;* $val(f) = \sum_{j \in NA(s)} f_{sj}$

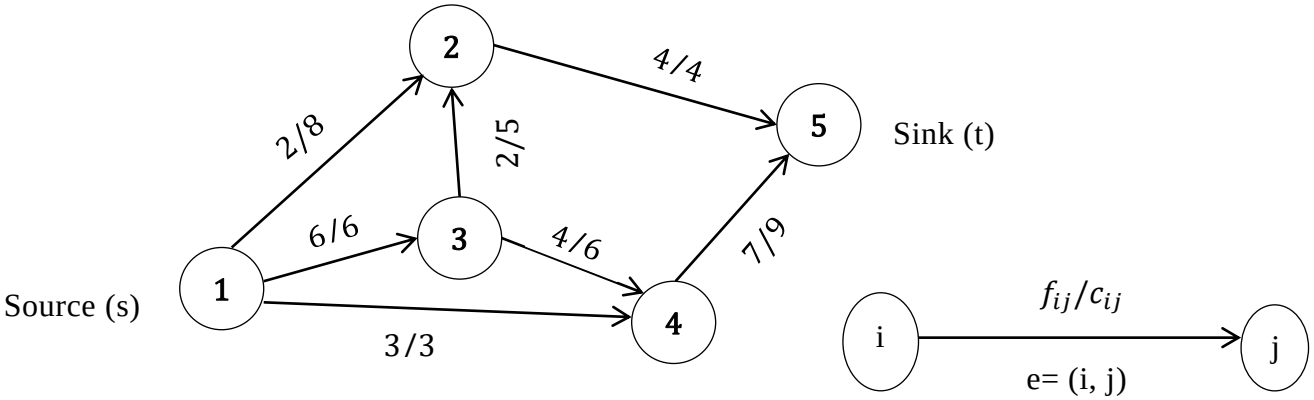

Figure 2.1 Sample flow network graph

As depicted in figure 2.1 the graph has 5 nodes and 7 arcs. That is

N={1 , 2 ,3 ,4 ,5} and A={(1, 2) ,(1, 3),(1, 4),(2 ,5),(3, 2),(3 ,4),(4, 5) }

Each arc $(i,j) \in A$ *has a flow* f_{ij} *that defines the number of units of the commodity that flows from node* i *to node* j. *Each arc also has a capacity* c_{ij} *that constrains the maximum number of units that can flow over that arc.*

In figure 2.1, each vertex is numbered and each arc is labeled as f_{ij}/c_{ij}, *showing the flow over that arc and the maximum possible flow. The edge between node*(1) *and node*(2), *for example, is labeled* $2/8$ *meaning that* 2 *units flow over that edge, which can sustain a capacity of up to* 8.

The flow conservation constrains in the network

- ✓ *Flows into node* (2) *is* $2 + 2 = 4$ *and flows out of this node is* 4
- ✓ *Flows into node*(3) *is* 6 *and flows out of this node is* $2 + 4 = 6$
- ✓ *Flows into node*(4) *is* $3 + 4 = 7$*and flows out of this node is* 7
- ✓ *the value of the flow is;* $val(f) = \sum_{j \in NA(s)} f_{sj} = 2 + 6 + 3 = 11$

$$= \sum_{i \in NB(t)} f_{it}$$

$$= 4 + 7$$

$$= 11$$

Moreover the problem can be expressed as linear programming problem (LPP) as

$$\textbf{maximize } \boldsymbol{f = f_{12} + f_{13} + f_{14}}$$

$$\boldsymbol{\text{subjected to } f_{12} + f_{32} - f_{25} = 0}$$

$$\boldsymbol{f_{13} - f_{32} - f_{34} = 0}$$

$$\boldsymbol{f_{14} + f_{34} - f_{45} = 0}$$

$$\boldsymbol{0 \leq f_{ij} \leq c_{ij};\ \forall (i,j) \in A}$$

CHAPTER-3

3. Flows and Cuts

In this chapter we discuss some properties of flows and cuts. We use these properties to prove the max-flow min-cut theorem to establish the correctness of the Ford-Fulkerson labeling algorithm.

Definition 3.1($s-t$ flow)

An $s-t$ flow is a function that satisfies;

i. $0 \leq f(e) \leq c(e);\ for\ all\ e=(i,j) \in A$ (capacity constraint)

ii. $\sum_{e\ into\ v} f(e) = \sum_{e\ out\ of\ v} f(e)\ ;\ for\ all\ v \in N \setminus \{s,t\}$ (*conservation constra* int)

Definition3.2 (Flow Value)

The value of the flow is defined as; $Val(f) = \sum_{e\ out\ of\ s} f(e)$

Since s and t are the only nodes that do not conserve flow, the value of f can be equivalently stated as the amount of flow entering into the sink node (t).

Proposition 3.1

For any feasible flow; $Val(f) = \sum_{e\ out\ of\ s} f(e) = \sum_{e\ into\ t} f(e)$

Proof:

This follows directly from conservation of flows.

$$Val(f) = \sum_{e\ out\ of\ s} f(e)$$

$$= \sum_{e\ out\ of\ s} f(e) - \sum_{v \in N\setminus\{s,t\}} \left(\sum_{e\ into\ v} f(e) - \sum_{e\ out\ of\ v} f(e) \right) \quad \text{.....(1)}$$

$$= \sum_{e\ into\ t} f(e)$$

Where the last line is due to the fact that each arc e appears in (1)-once as a leaving arc (with positive sign) and once as entering arc (with negative sign)-except the sink node (t) those arcs entering node t which appear exactly once and with positive sign.

Example 3.1

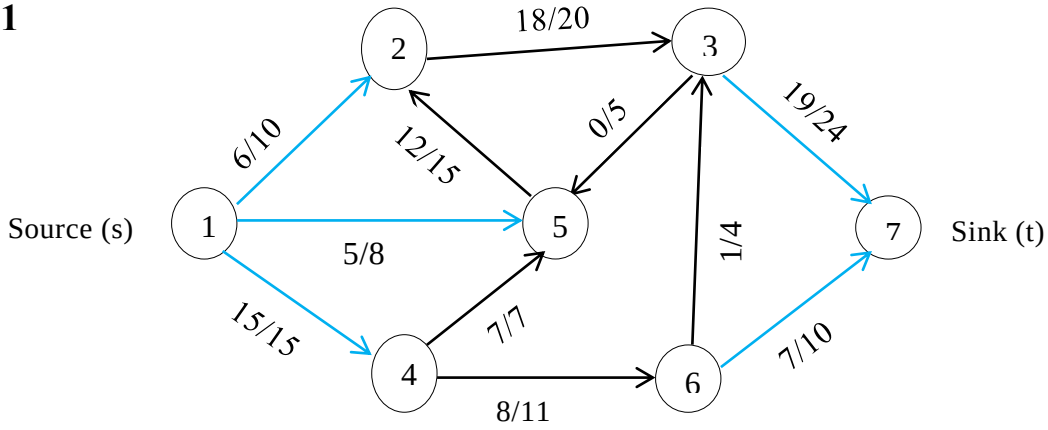

Figure 3.1

$$Flow\ value; Val(f) = \sum_{e\ out\ of\ s} f(e) = 6+5+15 = 26 = 19+7 = \sum_{e\ into\ t} f(e)$$

Definition3.3 $(p^+$ ***and*** $p^-)$

For any s-t path(p)*; let* $p^+ = the\ set\ of\ all\ forward\ arcs\ on\ p.$

$$p^- = the\ set\ of\ all\ backward\ arcs\ on\ p.$$

Flow augmenting path (FAP): *- intuitively we can think of simply as a path (s-t path) from the source s to the sink t along which we can send more flow than is currently in the network. A path is flow augmenting if it's all arcs have*

additional (spare) capacity. The augmented flow is the minimum of the spare capacities of the arcs.

Definition 3.4(Flow Augmenting Path (FAP))

A path (p) from s to t is a flow augmenting path (FAP) if and only if

i. $f_{ij} < c_{ij}$; $\forall e = (i,j) \in p^{+}$: $(non - full\ \ forward\ \ arcs\ on\ \ p.$

ii. $f_{ij} > 0$; $\forall e = (i,j) \in p^{+}$: $(non - empty\ \ backward\ arcs\ on\ p.$

Definition 3.5($s - t$ cut)

Let the node set N be partitioned into two non-empty disjoint subsets A and B = N\A; where $s \in A$ and t ∈ B.

An $\boldsymbol{s - t\ cut}$*, denoted by* $[A \backslash B]$*, is the set of arcs whose endpoints belong to the different subsets A and B.*

Definition3.6

Let $(A,B) = \{(i,j)\,; i \in A\,, j \in B\}$ *denote the set of forward arcs in the cut* $[A \backslash B]$*, and* $(B,A) = \{(i,j); i \in B\,, j \in A\}$ *denote the set of backward arcs in the cut* $[A \backslash B]$.

We refer to an arc $(i\,, j)$ *with* $i \in A$ *and* $j \in B$ *as a forward arc of the cut* $[A \backslash B]$ *and an arc* $(i\,, j)$ *with* $i \in B$ *and* $j \in A$ *as a back ward arc of the cut* $[A \backslash B]$

For example, in figure 3.2 the dashed arcs constitute an s – t cut *. For this cut*

$A = \{1\,, 3, 5\}, B = \{2, 4, 6\}$ *, then* $(A,B) = \{(1,2), (3,4), (5,6)\}$ *and*

$(B,A) = \{(2,3), (4,5)\}$

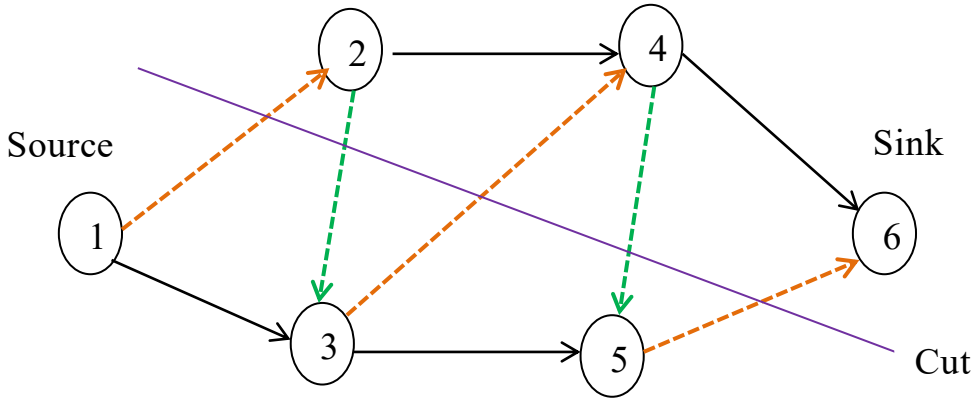

Figure 3.2 Example of s-t cut

Definition 3.7(Capacity of an $s-t$ cut $[A,\backslash B]$***)***

We define the capacity $cap[A\backslash B]$ *of an* $s-t$ *cut* $[A\backslash B]$ *as the sum of the capacities of forward arcs in the cut. That is,*

$$cap[A\backslash B]=\sum_{(i,\ j)\in(A,B)} c_{ij} = \sum_{e\ out\ of\ A} c(e)$$

Example 3.3

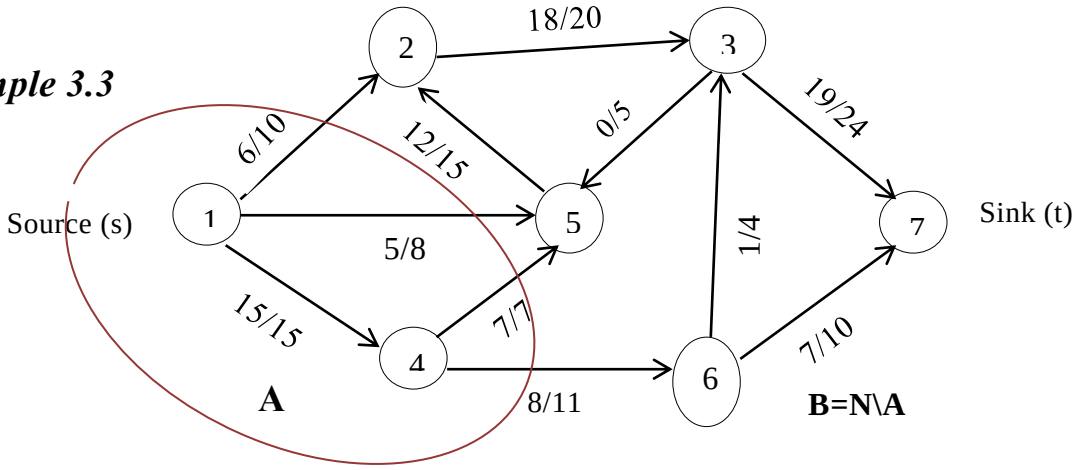

Figure 3.3

$N=\{1,2,3,4,5,6,7\}, A=\{1,4\}$ *and* $B=N\backslash A=\{2,3,5,6$ *then*

$cap[A\backslash B]=10+8+7+11=36$

Example 3.4

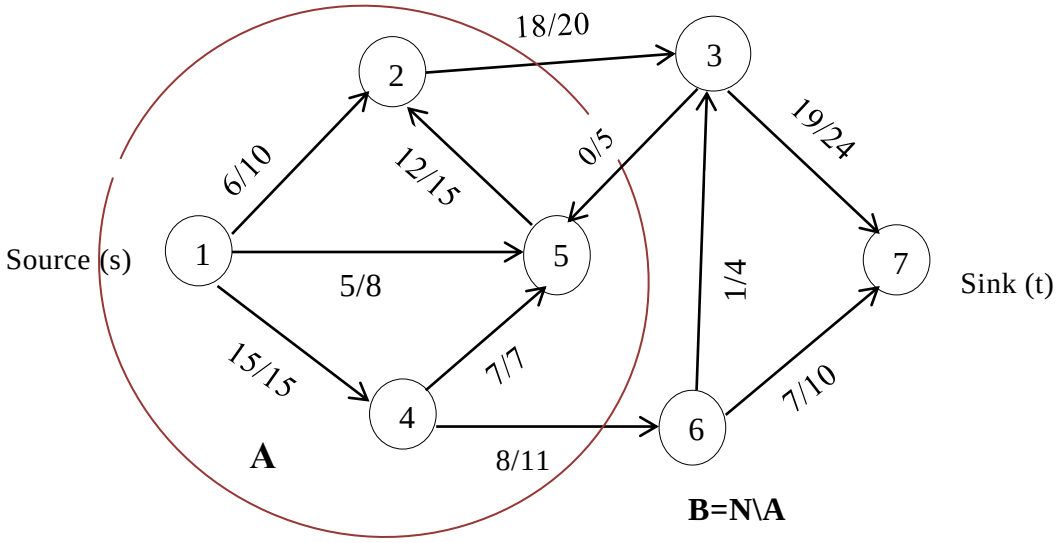

Figure 3.4

$$cap[A \setminus B] = 20 + 11 = 31$$

Definition 3.8(Minimum Cut)

Among all $s-t$ cuts in a network, a cut with the minimum capacity is called minimum cut. That is, an $s-t$ cut having a minimum capacity.

Relationship between Flows and Cuts

Definition 3.9 (Net flow across a cut)

Let $[A \setminus B]$ be any $s-t$ cut. The net flow across a cut $[A \setminus B]$ is the sum of the flows on its edges from A to B minus the sum of the flows on its edges from B to A. That is, $N[A \setminus B] = \sum_{e \text{ out of } A} f(e) - \sum_{e \text{ into } A} f(e)$

Theorem 3.1 (Flow Value Lemma)

Let f be any flow and $[A \setminus B]$ be any cut. The value of the flow f equals the net flow across the cut $[A \setminus B]$. That is,

$$Val(f) = N[A \setminus B]$$

Proof:

$$val(f) = \sum_{e\ out\ of\ s} f(e)$$

$$= \sum_{e\ out\ of\ s} f(e) - \sum_{e\ into\ A} f(e) + \sum_{v \in A/\{s\}} \left(\sum_{e\ out\ of\ v} f(e) - \sum_{e\ into\ v} f(e) \right)$$

$$= \sum_{v \in A} \left(\sum_{e\ out\ of\ v} f(e) - \sum_{e\ into\ v} f(e) \right)$$

$$= \sum_{e\ out\ of\ A} f(e) - \sum_{e\ into\ A} f(e)$$

$$= N[A \setminus B]$$

Example 3.5

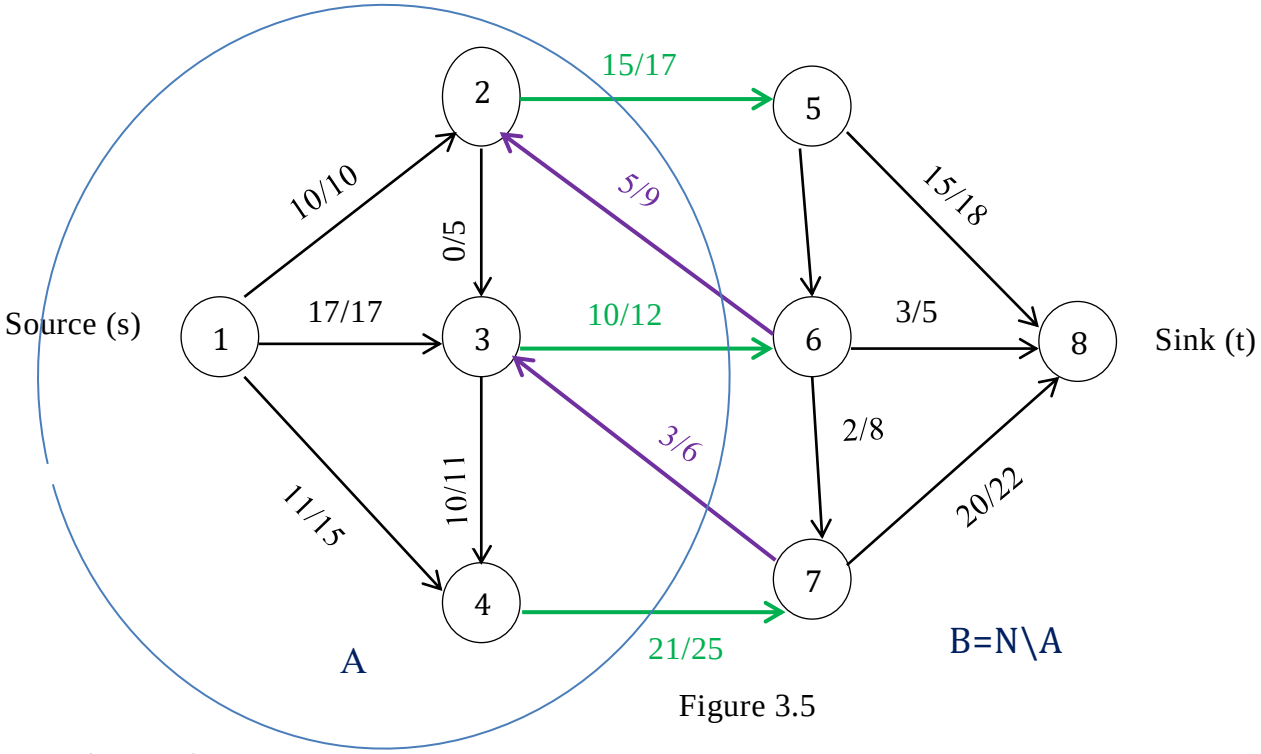

Figure 3.5

Flow value; $val(f) = 10 + 17 + 11 = 38 = 15 + 3 + 20$

Net flow across a cut; $N[A \setminus B] = \sum_{e\ out\ of\ A} f(e) - \sum_{e\ into\ A} f(e)$

$$= (15 + 10 + 21) - (5 + 3)$$

$$= 46 - 8$$

$$= 38$$

Theorem 3.2 (Weak Duality)

The value of any flow is less than or equal to the capacity of any cut in the network. That is, if f is any flow and $[A \setminus B]$ is any $s-t$ cut, then

$val(f) \leq cap[A \setminus B]$.

Proof:

$$val(f) = \sum_{e\ out\ of\ A} f(e) - \sum_{e\ into\ A} f(e) \text{ .....(flow value lemma)}$$

$$\leq \sum_{e\ out\ of\ A} f(e)$$

$$\leq \sum_{e\ out\ of\ A} c(e) = cap[A \setminus B] \text{ .....(capacity constraint)}$$

Theorem 3.3 (Optimality Certificate)

Let f be any flow and $[A \setminus B]$ be any cut. If $val(f) = cap[A \setminus B]$, then the flow is maximum and $[A \setminus B]$ is the minimum cut.

Proof:

Suppose $val(f) = cap[A \setminus B]$, then we need to show $val(f)$ is maximum and $[A \setminus B]$ is minimum cut.

- *For any flow f'; $val(f') \leq cap[A \setminus B] = val(f)$*
- *For any cut $[A' \setminus B']$; $cap[A' \setminus B'] \geq val(f) = cap[A \setminus B]$*

Theorem3.4 (Augmenting Path Theorem)

A flow is maximum flow if and only if there is no augmenting path.

Theorem 3.5 (Max-Flow Min-Cut Theorem)

Let f be any flow and $[A \setminus B]$ be any $s-t$ cut in G. The maximum flow value is equal to the capacity of the minimum cut.

We prove both theorems simultaneously by showing the following are equivalent.

i. *$val(f)$ is maximum flow in G.*

ii. *There is no augmenting path with respect to* $val(f)$.

iii. *There exists an* $s-t$ *cut* $[A \setminus B]$ *such that* $val(f) = cap[A \setminus B]$.

Proof:

We prove the implications cyclically.

(i) $\Rightarrow$ *(ii) equivalent to not (ii)* $\Rightarrow$ *not (i)*

We prove the contrapositive. i.e., if there is augmenting path then f *is not a maximal flow. If* f' *is the flow achievable in an augmenting path, then* $f + f'$ *is a flow in* G*, which means* f *is not a maximum.*

(ii) $\Rightarrow$ *(iii) Suppose there is no flow augmenting path with respect to* $val(f)$.

Let $[A \setminus B]$ *be any* $s-t$ *cut where* A *is the set of nodes reachable from* s *in the network* G.

- *by definition of* A*:* $s \in A$
- *by definition of* f*;* $t \notin A$

$$val(f) = \sum_{e \text{ out of } A} f(e) - \sum_{e \text{ into } A} f(e) \qquad (\textit{flow value lemma})$$

$$= \sum_{e \text{ out of } A} c(e) - 0 \quad (\textit{reachability implies } f(e) = 0 \textit{ for backward arcs in the cut } [A \setminus B])$$

$$= cap[A \setminus B] \qquad (\textit{reachability implies } f(e) = c(e) \textit{ for forward arcs in the cut } [A \setminus B])$$

(iii) $\Rightarrow$ *(i) this is a consequence of weak duality.*

Suppose that $[A \setminus B]$ *is any cut with* $val(f) = cap[A \setminus B]$. *Then, the value of any flow* $f' \leq cap[A \setminus B] = val(f)$.

Thus, $val(f)$ *is a maximum flow.*

Calculating a min-cut from a max-flow

The max-flow min-cut theorem establishes an important correspondence between flows and cuts in networks. By solving a maximum flow problem, we also solve a complementary minimum cut problem. The minimum cut is actually simple to find after the Ford-Fulkerson algorithm has been completed.

To compute min-cut $[A \setminus B]$ *from a max-flow*

We can divide the network into two sets of nodes A *and* $B = N \setminus A$ *based on the flows.* Let A=the set of reachable node(s) from the source node s in the graph, so that all arcs (e) leaving A in the network have f (e) =c (e) (full forward arcs) and all arcs (e) entering A in the network have f (e) =0 (empty backward arcs).

The arcs between A and B constitute an (s-t) cut in the graph: the source is on one side of the cut and the sink is on the other. All flows from s to t has to flow across this cut, and so that there is no way to increase the flow further, because all the arcs in the cut pointing from A to B are saturated and all arcs pointing in the reverse direction are empty. The capacity of the cut is the sum of the capacities of the arcs in the cut pointing from A to B. Then $[A \setminus B]$is the minimum cut. It is a fundamental result that $Max\ Flow = Min\ Cut$.

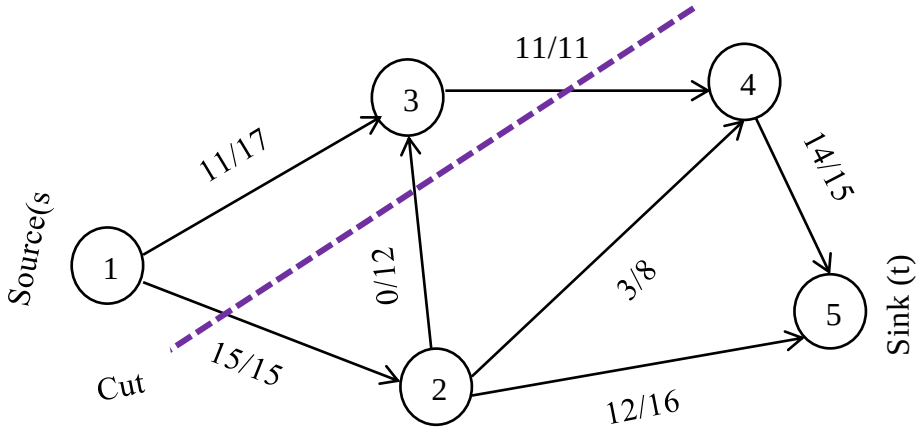

Figure 3.6

Consider an $s-t$ *cut* $[A \setminus B]$ *with* $A = \{1, 3\}$ *and* $B = N \setminus A = \{2,4,5\}$.

Then, an $s-t$ *cut* $[A \setminus B]$ *is the minimum cut with capacity=26=*$val(f)$.

CHAPTER-4

4. Methods for Solving Maximum Flow Problem

The maximum flow problem is one of the most fundamental problems in network flow theory and has been investigated extensively. This problem involves a direct network with arcs carrying flow. The only relevant parameter is the upper bound on arc flow, called arc capacity. The problem is to find the maximum flow that can be sent through the arcs of the network from some specified node (s) called the source, to a second specified node (t) called the sink.

There are several methods available for the solution of maximum flow network problems. In this chapter we are going to deal with the generic augmenting path algorithm (method) and Ford-Fulkerson labeling algorithm (method) for solving the maximum flow problem.

4.1. Generic Augmenting Path Algorithm (Method)

To find the maximum flow value of the flow that can be send from the source s to the sink t in the network using generic augmenting path algorithm (method), do the following until no flow augmenting path may be found.

Step (1). Find a flow augmenting path.

Step (2). Determine the maximum flow increase (δ) in the path(p).

$\delta = \min \begin{cases} \delta_1 = \min\{c_{ij} - f_{ij} : (i,j) \in p^+\} \\ \delta_2 = \min\{f_{ij} : (i,j) \in p^-\} \end{cases}$ is the minimum residual capacity of any arc in path p, (also called bottleneck capacity of an augmenting path), is the maximum flow that can be send through p.

Step (3). Update the flow in each arc on the path(p).

Let f_{ij}^{*} be the new value of the flow in the arc (i,j), then for every $arc(i,j) \in p$; $f_{ij}^{*} = \begin{cases} f_{ij} + \delta & for (i,j) \in p^{+} \\ f_{ij} - \delta & for (i,j) \in p^{-} \end{cases}$ and this called flow augmentation.

Example 4.1 Consider the following flow network and find the optimal flow.

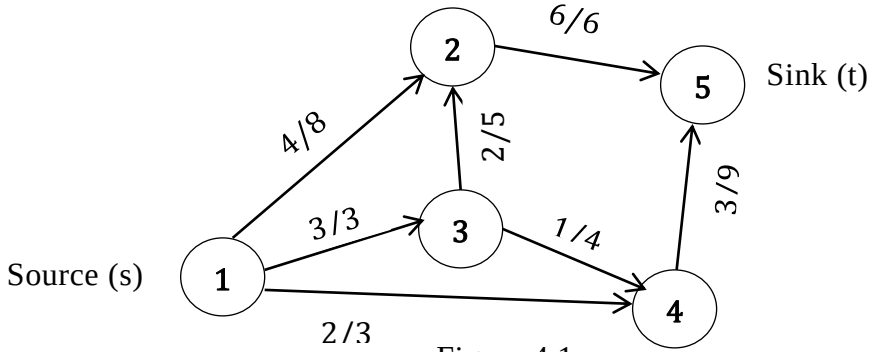

Figure 4.1a

Solution:

One flow augmenting path is: $p_1 = 1 \to 2 \to 3 \to 4 \to 5$

$p^{+} = \{(1,2),(3,4),(4,5)\}$; $p^{-} = \{(2,3)\}$

$\delta_1 = \min\{c_{12} - f_{12}, c_{34} - f_{34}, c_{45} - f_{45}\} = \min\{8-4, 4-1, 9-3\} = \min\{4, 3, 6\} = 3$

$\delta_2 = \min\{f_{23}\} = \min\{2\} = 2$

$\delta = \min\{\delta_1, \delta_2\} = \min\{3, 2\} = 2$

Here the minimum residual capacity along the path p_1 is 2. Then the flow augmentation, along the path p_1 gives $f_{12}^{*} = 4+2 = 6$, $f_{23}^{*} = 2-2 = 0$, $f_{34}^{*} = 1+2 = 3$ *and* $f_{45}^{*} = 3+2 = 5$

After this flow augmentation, the flow in the network is

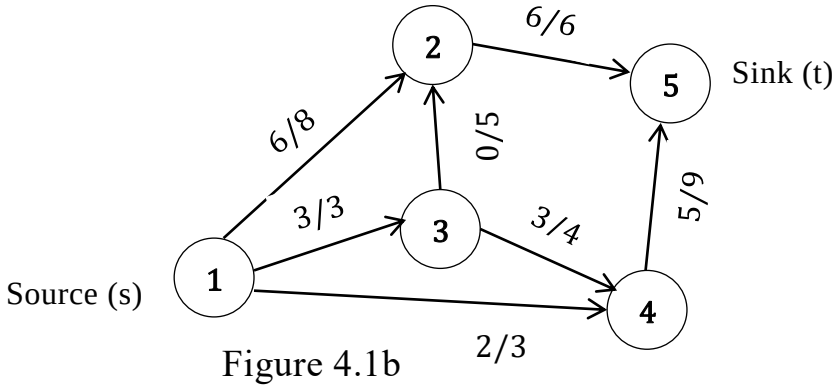

Figure 4.1b

There is now a flow augmenting path in the graph that result in the direction of flow on arc$(1,4)$. So consider a flow augmenting path $p_2 = 1 \to 4 \to 5$

$p^+ = \{(1,4),(4,5)\}$; $p^- = \phi$ (There is no backward arc along this path)

$\delta = \delta_1 = \min\{c_{14} - f_{14}, c_{45} - f_{45}\} = \min\{3-2,\ 9-5\} = \min\{1,4\} = 1$

Then the flow augmentation along this path (p_2) gives

$f_{14}^* = 2+1 = 3$ *and* $f_{45}^* = 5+1 = 6$

After this flow augmentation, the flow in the network is

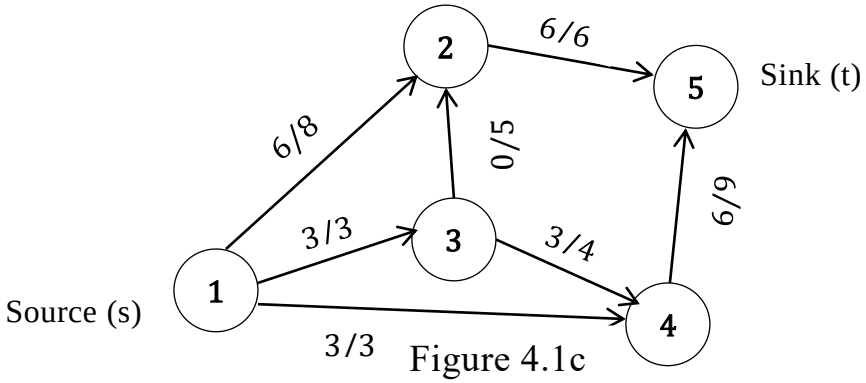

Figure 4.1c

Now we cannot find any more FAP in the network. If we try to augment flow further, we cannot push flow along the arcs $(1,3)$ *and* $(1,4)$. We can push flow

along arc (1,2) but no further (2,5) is saturated, and the arc (2,3) entering node 2 is empty. So the current flow is optimal with flow value $val(f) = 12 = 6 + 6$

Example 4.2

Consider the following capacitated network and then find maximum flow.

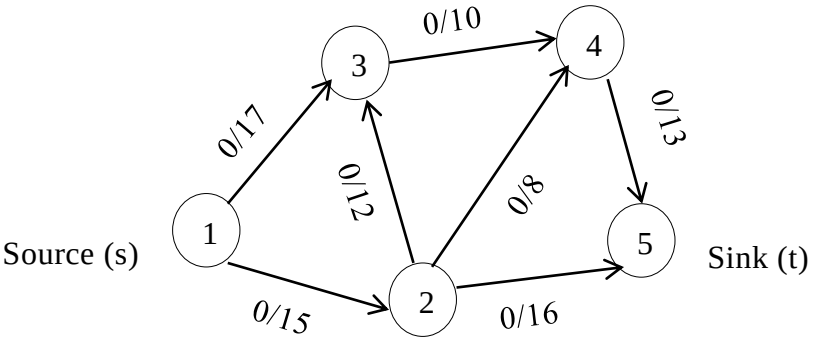

Figure 4.2a

Solution:

There is no initial flow.

One augmenting path is p_1=$1 \to 2 \to 3 \to 4 \to 5$

$\delta = 10\, units$ can be send along this path, at which point arc $(3\,,4)$ is saturated.

Then after doing the flow augmentation, the flow network is

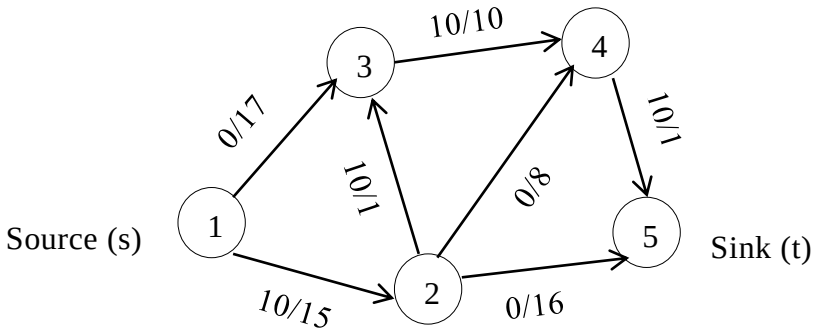

Figure 4.2b

Again consider a FAP: $p_2 = 1 \to 3 \to 2 \to 5$

The maximum possible flow value that can be send in this path is $\delta = 10$, gives the overall flow below.

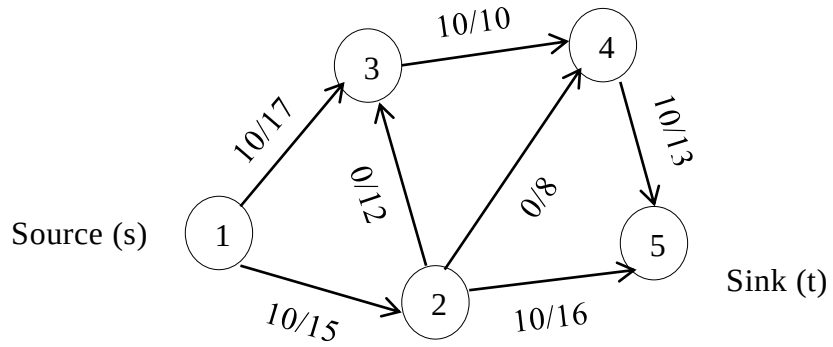

Figure 4.2c

There is another flow augmenting path; $p_3 = 1 \rightarrow 2 \rightarrow 5$ with both arcs used in the forward direction. The maximum possible flow value that can be sent along this path is $\delta = 5$, giving the overall flow below.

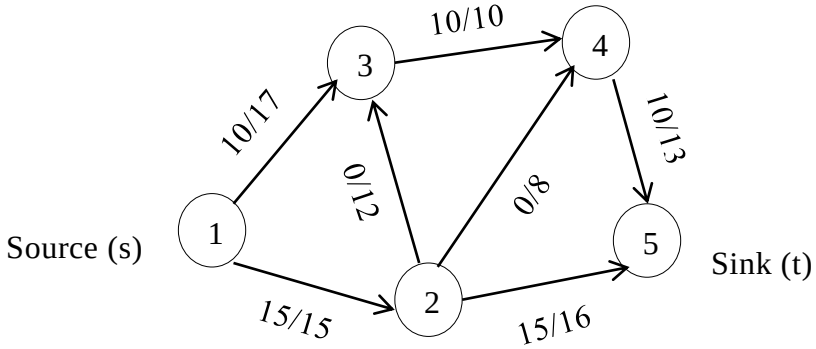

Figure 4.2d

Since we cannot find any more FAP in the network, now the current flow is optimal with flow value; $val(f) = 25 = 10 + 15$

Here getting a flow augmenting path from a given network (especially for large networks) is not easy. For large networks and for computer implementation, a formal procedure is required. The Ford-Fulkerson labeling

algorithm which is described below is very important to determine the flow augmenting path (FAP) of the network.

4.2. Ford-Fulkerson Labeling Method (Algorithm)

Let $G = (N, A)$ *be a capacitated network with* $N = \{s = 1, 2, 3, \ldots, m = t\}$ *where* 1 *is the source node and* m *is the sink node. And for each arc* $e = (i, j) \in A$*, let* c_{ij} *be the capacity and* f_{ij} *be the flow (initially* $f_{ij} = 0$; $\forall (i,j) \in A$*), then we want to find the maximum flow from the source node*(1) *to the sink node*(m).

The Ford-Fulkerson method (named for L.R. Ford, Jr and D.R. Fulkerson) is an algorithm which computes the maximum flow in a flow network based on the idea of FAP.

The Ford-Fulkerson method is iterative. We start with $f_{ij} = 0$ for all $\forall (i,j) \in A$, giving an initial flow of value 0. At each iteration, we increase the flow value by finding an "augmenting path," which we can think of simply as a path from the source s to the sink t along which we can send more flow, and then augmenting the flow along this path. We repeat this process until no augmenting path can be found. The max-flow min-cut theorem will show that upon termination, this process yields a maximum flow.

The idea behind this algorithm is simple. As long as there is an s-t path, with available FAP on all edges in the path, we send flow along one of this path. Then we need to find another path, and so on.

The Ford-Fulkerson Labeling algorithm has two main steps. The first is a labeling process that searches for a flow augmenting path i.e., a path from node 1 to node m for which $f_{ij} < c_{ij}$ along forward arcs and $f_{ij} > 0$ along backward arcs. If this step finds a flow augmenting path, the second step changes the flow accordingly otherwise, no augmenting path exists, and then we get the maximum flow. The detail step is as follows;

The algorithm begins with a feasible flow (say $f_{ij} = 0$ for all arcs), identifies a flow augmenting path and does augmentation iteratively until no more

augmenting path can be constructed. To identify a flow augmenting path, this algorithm puts label on each node j which indicates the node prior to node j on the current flow augmenting path. If j is labeled by$(+, i)$, then it is to mean that$(i, j) \in p^+$. If j is labeled by $(-, i)$, then it is to mean that$(i, j) \in p^-$.

This algorithm starts from the source node and puts stepwise such labels on nodes. If the algorithm terminates with the sink node is labeled, the flow augmenting path is found by tracing the path backward from m and constructing an s-t path (p) from the labels encountered. If the algorithm terminates with t unlabeled, no flow augmenting path exists.

After a flow augmenting path (say p) is identified, then this algorithm determines the residual capacity δ of p. And perform flow augmentation. If a node j on p has label $(+, i)$, then $f_{ij}^* = f_{ij} + \delta$ and if j on p has label $(-, i)$, then $f_{ij}^* = f_{ij} - \delta$. In general, at each iteration of this algorithm every node is in one of the three states: labeled and scanned (listed in set s_1), labeled and un-scanned (listed in set s_2) or unlabeled (listed in set s_3). Upon entering step-0, all nodes are unlabeled. The first step renders the source node labeled and un-scanned.

Ford-Fulkerson Labeling Algorithm

Step(0). Initial:

Let $s_1 = \emptyset \;\; ; \;\; s_2 = \{1\} \;\; ; \;\; s_3 = \{2, 3, 4, \ldots, m\}$

Step(1). Scanning:

Choose node $i \in s_2$ and then update s_1 *and* s_2

$s_1 = s_1 \cup \{i\} \;\; ; \;\; s_2 = s_2 \backslash \{i\}$

Step(2). Labeling:

- $\forall j \in NA(i) \cap s_3$ do;

If $f_{ij} < c_{ij}$, then assign the label $(+, i)$ to node j and then update s_2 *and* s_3

$$s_2 = s_2 \cup \{j\} \quad ; \quad s_3 = s_3 \backslash \{j\}$$

- $\forall j \in NB(i) \cap s_3$ do ;

If $f_{ij} > 0$, then assign the label $(-, i)$ to node j and update s_2 *and* s_3

$$s_2 = s_2 \cup \{j\} \quad ; s_3 = s_3 \backslash \{j\}$$

If $j = m$, determine a flow augmenting path by tracing backward from the sink node m to the source node 1.; find its δ, do flow augmentation and remove all labels and go to step (0).

Step(3). If $s_2 \neq \emptyset$, go to step (1); else i.e., if $s_2 = \emptyset$, then the current flow is optimal, **STOP**.

Correctness of the Labeling Algorithm

To study the correctness of the labeling algorithm, note that in each iteration, 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 f is a maximum flow. Suppose at this stage that A is the set of labeled nodes and $B = N \backslash A$ is the set of unlabeled nodes. Clearly, $s \in A$ *and* $t \in B$. Since the algorithm cannot label any node in B from any node in A, the residual capacity $\delta = c_{ij} - f_{ij} = 0$ for each arc $(i,j) \in (A,B)$.This implies that $c_{ij} = f_{ij}$ for every arc $(i,j) \in (A,B)$ and $f_{ij} = 0$ for every arc$(i,j) \in (B,A)$.

So, $val(f) = \sum_{(i,j) \in (A,B)} f_{ij} - \sum_{(i,j) \in (B,A)} f_{ij} = \sum_{(i,j) \in (A,B)} c_{ij} = cap[A \backslash B]$

This discussion shows that the value of the current flow $f$equals the capacity of the cut$[A \backslash B]$. But then max-flow min-cut theorem implies that f is maximum flow and [A\B] is a minimum cut. This conclusion establishes the correctness of the labeling algorithm.

Example 4.3

Consider the following flow networks and find the maximum flow using Ford – Fulkerson labeling algorithm.

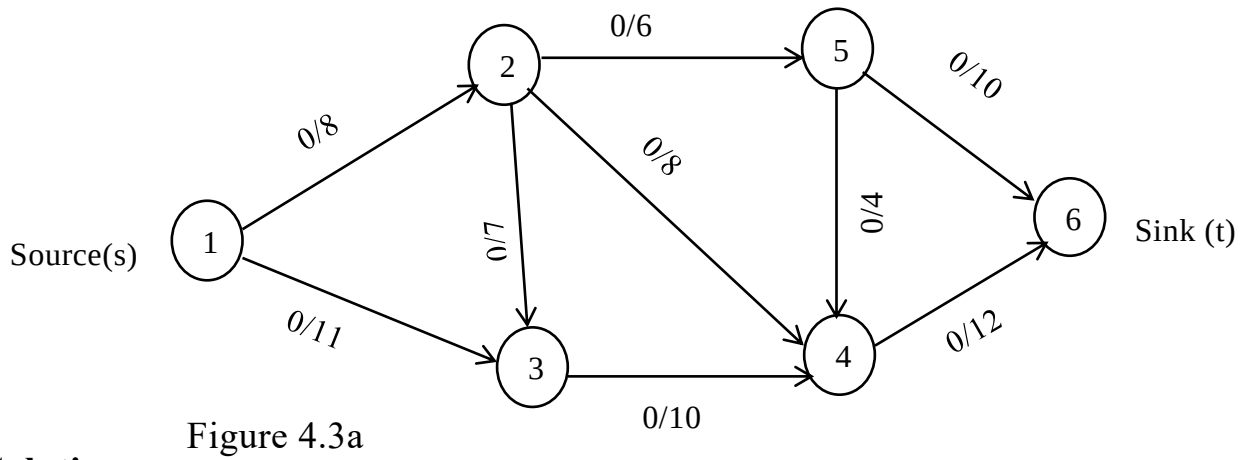

Figure 4.3a

Solution

Step (0). Initial: Let $s_1 \in \emptyset$; $s_2 = \{1\}$; $s_3 = \{2\,,3\,,4\,,\ 5\,,6\}$

Iteration-1

Step (1). Scanning; Choose $i \in s_1 = \{1\}$; $i = 1$

Update $s_1\ and\ s_2$

$s_1 = s_1 \cup \{i = 1\} = \emptyset \cup \{1\} = \{1\}$; $s_2 = s_2 \backslash \{1\} = \{1\} - \{1\} = \emptyset$

Step (1). Labeling

$NA(i = 1) = \{2\,,3\}$

$NA(i = 1) \cap s_3 = \{2\,,3\} \cap \{2\,,3\,,4\,,5\,,6\} = \{2\,,3\}$

$0 = f_{12} < c_{12} = 8$; $0 = f_{13} < c_{13} = 11$

Label $j = 2$ and $j = 3$ by $(+\,,1)$ and, then update $s_2\ and\ s_3$.

$s_2 = s_2 \cup \{j\} = \emptyset \cup \{2\,,3\,\} = \{2\,,3\}$; $s_3 = s_3 \backslash \{j\} = \{2,3,4,5,6\} - \{2,3\} = \{4,5,6\}$

Since $NB(i) \cap s_3 = \emptyset$, go to step (3).

Step (3). Since $s_2 = \{2,3\} \neq \emptyset$, go to step (1)

Iteration-2

Step (1). Scanning; Choose $i \in s_2 = \{2\,,3\}$

Let us choose $i = 2$ and update $s_1 \ and \ s_2$

$s_1 = s_1 \cup \{i = 2\} = \{1\} \cup \{2\} = \{1,2\}$; $s_2 = s_2 \backslash \{i\} = \{2,3\} - \{2\} = \{3\}$

Step (2). Labeling

$NA(i = 2) = \{3 , 4, 5\}$

$NA(i = 2) \cap s_3 = \{3 ,4 ,5\} \cap \{4,5,6\} = \{4,5\}$

$0 = f_{24} < c_{24} = 8$; $0 = f_{25} < c_{25} = 6$

Label $j = 4 \ and \ j = 5$ by $(+ , 2)$ and update $s_2 \ and \ s_3$

$s_2 = s_2 \cup \{j\} = \{3\} \cup \{4,5\} = \{3,4,5\}$; $s_3 = s_3 \backslash \{j\} = \{4,5,6\} - \{4 ,5\} = \{6\}$

Since $NB(i = 2) \cap s_3 = \emptyset$, go to step (3).

Step (3). Since $s_2 = \{3 ,4 ,5\} \neq \emptyset$, go to step (1).

Iteration-3

Step(1). Scanning; Choose $i \in s_2 = \{3 , 4 , 5\}$

Let us choose $i = 4$, then update $s_1 \ and \ s_2$

$s_1 = s_1 \cup \{i\} = \{1 , 2\} \cup \{4\} = \{1,2,4\}$;

$s_2 = s_2 \backslash \{i\} = \{3 , 4 ,5\} - \{4\} = \{3 , 5\}$

Step(2). Labeling;

$NA(i = 4) = \{6\}$

$NA(i = 4) \cap s_3 = \{6\} \cap \{6\} = \{6\}$

$0 = f_{46} < c_{46} = 12$, so label $j = 6 \ by \ (+ ,4)$

Now put $s_1 = \{1 , 2 ,4 ,6\}$ and $s_3 = \emptyset$. Since $j = m = 6$, then by tracing backward we get a FAP; $P_1 = 1 \rightarrow 2 \rightarrow 4 \rightarrow 6$ with minimum residual capacity of $\delta = 8$.

Then after doing flow augmentation to this flow augmenting path, we get

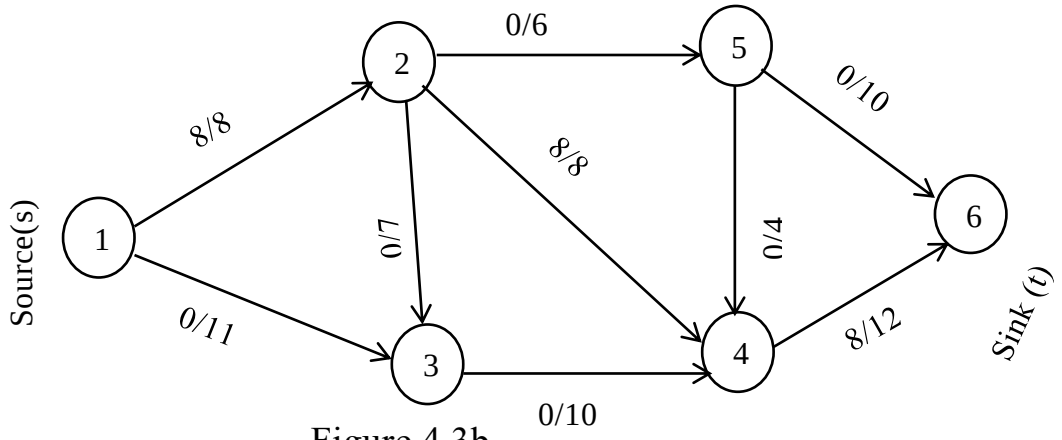

Figure 4.3b

By applying similar procedure as given above we get a FAP;

$p_2 = 1 \rightarrow 3 \rightarrow 4 \rightarrow 6$ with a residual capacity $\delta = 4$.

FAP; $p_3 = 1 \rightarrow 3 \rightarrow 4 \rightarrow 2 \rightarrow 5 \rightarrow 6$ with a residual capacity $\delta = 6$ and then the final network is given below.

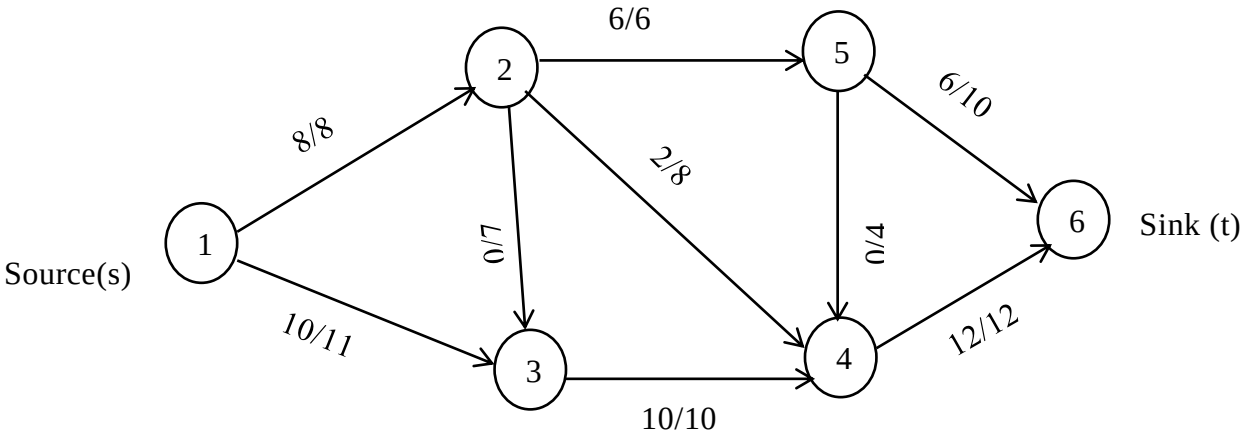

Figure 4.3c

With optimal flow value; $val(f) = 10 + 8 = 6 + 12 = 18$

Conclusion

The maximum flow is one of the most fundamental problems in network flow theory. In this thesis, we give the generic augmenting algorithm and Ford-Fulkerson labeling algorithm for finding the amount of maximum flow from the source to the sink in a flow network. A numerical example is solved to illustrate the algorithm.

The Ford-Fulkerson method is iterative. We start with $f_{ij} = 0$ for all $\forall$ (i, j) $\in$A, giving an initial flow of value 0. At each iteration, we increase the flow value by finding an "augmenting path," which we can think of simply as a path from the source s to the sink t along which we can send more flow, and then augmenting the flow along this path. We repeat this process until no augmenting path can be found. The max-flow min-cut theorem will show that upon termination, this process yields a maximum flow.

The labeling algorithm is possibly the simplest algorithm for solving flow problem. However, the worst case is the number of iteration it takes to find a flow of maximum value is high. The other drawback is that if the capacities are irrational, the algorithm might not terminate.

Bibliography

[1]. Auja, R.K and Orlin, J.B: A fast and simple algorithm for the maximum flow problem. Sloan School of Management, M.I.T., 1987

[2]. Bazarra, M.S., Jarvis, J.J.,and Sherali, H.D: Linear Programming and Network Flows (John Wiley & Sons).1977.

[3]. G. Srinivasan: Operations Research, Principles and Applications. New Delhi; PHI Learning Private limited 2nd ed., 2010.

[4]. Jensen, P.A., Barnes, J,W: Network Flow Programing; J.Wiley & sons. New York.

[5]. L.R. Ford, Jr. and D.R. Fulkerson: Flows in Networks. Princeton University Press, Princeton, 1962.

[6]. Ravindra K. Ahuja,Thomas L.Magnanti, and JamesB. Orlin: Network Flows: Theory, Algorithms, and Applications. Prentice Hall, 1993.

[7]. Thomas H. Cormen, Charles E. Leiserson, Ronald. L. Rivest and Clifford Stein: Introduction to Algorithms. London, England, The MIT Press,3rd ed,2009

[8]. Wayne L. Winston: Operations Research Applications and Algorithms. Book/Cole 4thed, 2004