Office of Graduate Program

Faculty of computer and mathematical Science

Department of Mathematics

Graduate Project Report on

Networks with Applications

A project submitted to the Department of Mathematics in Partial fulfillment of the requirements for the degree of Master of Science in Mathematics.

Compiled By: - Yetneberk Kuma

Advisor: - Dr. Yirgalem Tsegaye

Addis Ababa, Ethiopia

June, 2011

Declaration

I declare that this project has been composed by me and that no part of the project has formed the basis for the award of any Degree, Diploma, Associate ship, Fellowship or any other similar title to me.

Yetneberk Kuma

Signature_____ Date_____

Permission

This is certify that this project is compiled by **Mr. Yetneberk Kuma** in the Department of Mathematics, Addis Ababa University, under my supervision.
I hereby also confirm that the project can be submitted for evaluation by examiners and eventual defense.

Dr. Yirgalem Tsegaye

Signature_____ Date_____

Summary of the project

This project is conserned with a network and its applications. The project begins with pre-concepts of graph theory and focuses on one of the the well known application of graph theory called networks. The project on its main body begins by defining terms like network, flow , net flow, cuts,…,etc.and concentrate on the basic ideas to obtain an optimal flow in a given network. The concept of network is helps to solve prblems related with the movement of commodity (or materials) from one location to another. Forinstance, an oil company must move crude oil from the oil field to its refinery, and a long-distance telephone company must move messages from one city to another.

Even though, there are a lot of applications of networks, but this project deals on two of them, it's application on menger's theorem and obtaining the cardinality of the maximum matching in any bipartite graph. Menger's theorem is proved easily using network flow method, and normally it is very awkward to prove it without using network flow problem. In maximum bipartite matching, the network flow suggests that $|M| = val(f)$ for a flow f in a network and a maximum matching M in a bipartite graph by transforming the bipartite graph G into a network G' .

Acknowledgment

My first special gratitude goes to my advisor Dr. Yirgalem Tsegaye for her valuable enlightening discussions, suggestions, comments and her generous kindness in preparing this Project. It is always a joy to talk with her in every discussion of this project. She teaches me just how to my students in future soon. Next, I would like to thanks professor Melkamu Zeleke for his nice support of this project. Finally, my gratitude goes to the Department of Mathematics for providing the necessary facilities.

Table of content

List of figures

List of Symbols

Symbol	stands for
G	Graph
D	digraph
N	network
$\mathcal{K}(G)$	connectivity of G
$\lambda(G)$	edge- connectivity of G
$k(x,y)$	the minimum number of edges whose removal destroys x,y-path
$\lambda(x,y)$	the maximum number of edge disjoint x,y-path
$\varepsilon(p)$	the tolerance of path p
e	arc/edge as required
c	capacity function
f	network flow function
$f^+(v)$	the total flow leaving vertex v
$f^-(v)$	the total flow entering vertex v
$\sum$	summation
$val(f)$	value of flow f
ij	arc whose tail is i and head j
$f\ (ij)$	flow on arc ij
K	cut
≤	less than or equal
≥	greater than or equal
R	the set of vertices labeled reached
S	the subset of R labeled searched
$\emptyset$	empty set
$R\backslash S$	R without S
$\in$	an element of
$\notin$	not an element of
Max $val(f)$	maximum value of flow f
min$capK$	capacity of minimum cut K
$\forall_e$	for all arc e
$d_{D^*}{}^+(s)$	the number of arcs leaving vertex s in D^*
$d_{D^*}{}^-(t)$	the number of arcs entering vertex t in D^*
$>$	greater than

$<$	less than
$\cup$	union
$\cap$	intersection
$G = (X, Y, E)$	bipartite graph with bipartition X and Y, edge set E
M	matching
$\lvert M \rvert$	the cardinality of M

PRELIMINARIES

1. Graphs and Paths

A graph is a mathematical object, which could be represented by a diagram, consisting of points, called vertices, and line segments each of which join two vertices.

A finite graph G consists of a (finite) set denoted by V, and a collection E or $E(G)$, of unordered Pairs $\{u, v\}$ of (not necessarily distinct) vertices from V**.** Each element of V is called a vertex or a node, and each element of E is called an edge or a link.

The number of vertices in a graph G, the cardinality of V, is called the order of graph G and denoted by $|G|$. The number of edges in G, the cardinality of E, is called the size of graph G and is denoted by $\| G \|$. Given a graph $G = (V, E)$, two vertices $u, v \in V$, are said to be adjacent vertices, if $uv \in E$ and edge uv is said to be an incident to u and v.

The degree of a vertex v, $deg(v)$, is the number of edges incident on v. A graph with no loops (an edge of the form $e = vv$) or no multiple edges (no two edges with same ends) is called *a simple graph*.

Let $G = (V, E)$ be a graph with $V = \{v_1, v_2, \ldots\ldots, v_n\}$ and $E = \{e_1, e_2, \ldots, e_m\}$. Let k be a non-negative integer such that: A non-empty sequence

$W = (v_0, e_1, v_1 e_2, v_2, \ldots\ldots, e_k v_k)$ is said to be as *walk* from v_0 to v_k , or (v_0, v_k)-walk, if e_i is incident with $v_{i-1}$and v_i, for $1 \leq i \leq k$. If a walk does not repeat a vertex, it is called *a path***.** In a (u, v)-path, vertices other than u and v are called internal vertices of the path.

A graph G is said to be connected if there is a path in G between any pair of vertices, otherwise it is said to be disconnected. A component of a graph G is a connected sub graph of G. The number of these components in a graph is denoted by $\omega(G)$.

If $\omega(G) = 1$ then, G is said to be a connected graph.

2. *Connectivity of Graph*

A cut-vertex of a graph G is a single vertex, v of G whose deletion (removal) disconnects the graph i.e. the relation $\omega(G) < \omega(G - v)$ is valid where $\omega(G)$ is the number of components of G. Let $G = (V, E)$ be a connected graph. A vertex cut or separating set of G is a set $S \subseteq V(G)$ of vertices satisfying:

1. G is connected
2. $G - S$ is disconnected but the removal of some (but not all) vertices of S does not disconnect G.

The minimum size of a vertex set S such that $G - S$ is disconnected or has only one vertex is known as the connectivity of G, which is denoted by $k(G)$. A graph G is k-connected if $k(G) > k$. Note that when we remove a vertex, we must also remove the edges incident to it.

A bridge or a cut- edge of a graph G is a single edge e of G whose deletion (removal) disconnects a graph i.e. the relation, $\omega(G) < \omega(G - \mathrm{e})$ is valid. Let $G = (V, E)$ be a connected graph. A cut set is a set F of edges such that $G - F$ is disconnected but the deletion (removal) of some (but not all) of edges of F does not disconnect G.

The edge-connectivity $\lambda(G)$ of a connected graph G is the smallest number of edges whose deletion/removal disconnects G. When $\lambda(G) \geq$ k, the graph G is said to be k-edge- connected. For any connected graph G, we have $k(G) \leq \lambda(G) \leq \sigma(G)$ where the minimum degree of a graph G is $\sigma(G)$.

Given $x, y \in V(G)$, a set $S \subseteq V(G)\backslash\{x, y\}$ is an x, y-separator or x, y-cut if $G - S$ has no x, y -path. Let p_1 and p_2 be two an x, y-paths, they are internally disjoint paths means that they do not share either an edge or vertex. Let $k(x, y)$ be the minimum size of a x, y-cut and let $\lambda(x, y)$ be the maximum size of a set of pair wise internally disjoint x, y paths.

An x, y-cut must contain an internal vertex of every x, y-path, and no vertex cut two internally disjoint x,y-paths. Therefore, always $k(x, y) \geq \lambda(x, y)$

3. Directed Graphs

A directed graph or digraph D consists of a set of elements, called vertices, and a list of ordered pairs of these elements, called arcs. Simply, a directed graph D is a graph G with an order/ orientation on every edges of G. The set of vertices is called the vertex set of D, denoted by $V(D)$, and the list of arcs is called the arc list of D, denoted by $A(D)$. If e is an arc and u and v are vertices such that $e = (u, v)$, then e is said to join u to v; u is the tail of e and v is its head.

With each digraph, D we can associate a graph G on the same vertex set; corresponding to each arc of D there is an edge of G with the same ends. This graph G is the *underlying graph of* D. Conversely, given any graph G, we can obtain a digraph from G by specifying for each edge, an order on its edges. Such a digraph is called the *orientation of* G.

Just as with graphs (undirected), digraphs have simple pictorial representations. A digraph is represented by a diagram of its underlying graph together with arrows on its edges, each arrow pointing towards the head of the corresponding arc. Every concept that is valid for graphs automatically applies to digraphs too. However, there are many concepts that involve the notion of orientation, and these apply only to digraphs. Some of these are listed and defined here. *A direct walk* in D is a finite non-empty sequence $W = (v_0, e_1, v_1 e_2, v_2, \ldots\ldots, e_k v_k)$ whose terms are alternately vertices and arcs, such that for $i =$ 1,2,3 $\ldots\ldots k$, the arc a_i has head V_i and tail V_{i-1}. *A directed path* in D is a directed walk that is a path i.e. *A directed path* in D a path whose edges are directed (oriented). *A directed cycle* in D is a closed directed path. If there is a directed (u, v)-path in D, vertex v is said to be *reachable* from vertex u in D. Two vertices are connected in D if each is reachable from the other.

Let $D = (V, A)$ be a directed graph. If S and T are subsets of $V(D)$, we denote the set of arcs of D whose tails are in S and their heads in T by $[S, T]$.

1. INTRODUCTION

A network flow problems could be effectively analyzed using digraphs. They can also treated in different situations, one can be transportation problem, Assignment or matching problem, Road network problem , short path problem etc.

In transportation problem, commodity produced in different locations need to be transported to different locations. For instance, an oil company shall move crude oil from the oil field to its refinery. In this situation, there is a limitation to the amount of the crude oil that can be moved at one time. Here we assume that flow cannot accumulate at a junction of the pipes, which carry the crude oil. That is, there is a flow of the crude oil and there is a network among all the oil fields, their destinations and some other junctions in between them. All these elements could be represented using a digraph, with some structure added to it. In assignment, problem (or matching) different individuals (or machines) are assigned to different tasks. For instance, in factory different machines can be assigned to different tasks to be performed. Here also the machines and tasks could be represented using a digraph with some more structure added to it.

Section two of this report begins with the formal definition of a network and ends with important theorems, the integral theorem and the max-flow min-cut theorem, which was stated by Ford and Fulkerson in 1956.

A network can be with multiple sources and sinks as shown on figure 2.1.1 and it can be reduced to an ordinary network of a single source and sink in order to find the maximum flow. Hence, this project focuses on networks of single source and sinks. This section also deals with feasible flows and cuts in network.

Finally, section three of this project presents some applications of network flow. Even though network flow has several applications, two of them, menger's theorems and maximum bipartite matching, are presented in detail. The first, Menger's theorem, is proved in simple and short ways as compared to its original proof,

which is very awkward and long stepped. Since Menger's theorems, use the concept of connectivity in both graphs and digraphs as its ground, network flow problems help to prove theorems/cases related to this. The second, maximum bipartite matching, the network flow problems help to find the possible maximum number of matching in a bipartite graph. The method uses two basic steps; first, one is converting the bipartite graph in to a network with multiple sources and sinks by specifying an order on the edges of the graph. Then, by introducing a super source and super sink, the network shall be reduced to a network of single source and sink with an integral flow corresponding to a matching in the bipartite graph and by assigning a unit capacity to each arc of the network.

2. NETWORK FLOW PROBLEM

Many practical problems require the movement of some commodity from one location to another. For example, an oil company must move crude oil from the oil field to its refinery, and a long-distance telephone company must move messages from one city to another. In both of these situations, there is a limitation to the amount of the commodity that can be moved at one time. The volume of the crude oil that the oil company can move, for instance, is limited by the capacity of the pipeline through which the oil must flow. In addition, the number of telephone calls that the phone company can handle is limited by the capacity of its cable and its switching equipment. This type of problem, in which some commodity must be moved from one location to another subject to the restriction that certain capacities not be exceeded, is called *a network flow problem.*

2.1. Networks

Definition 2.1.1[2]*:* **-** A network N is a digraph $D = (V, A)$ (underlying digraph of N) with two non-empty and disjoint sets of vertices X and Y,and a non-negative integer valued function c defined on its arc set A, where it's value on an arc e is the natural restriction to the arc e.

The vertices in X are the sources of N and those in Y are the sinks of N. They correspond to production centers and markets, respectively, in transportation problem vertices, which are neither sources nor sinks, are called intermediate vertices**,** which are denoted by I.

The function c is the capacity function of N and its value on an arc e is said to be *the capacity of* e. The capacity of an arc can be thought of as a value representing the maximum rate at which a commodity can be transported along it. A network is represented by drawing its underlying digraph and labeling each arc with its capacity.

Example 2.1.1[2]*:* Figure 1.1.1 shows a network with two sources x_1 and x_2, three sinks $y_1, y_2 \text{ and } y_3$ and four intermediate vertices $v_1, v_2, v_3 \text{ and } v_4$.

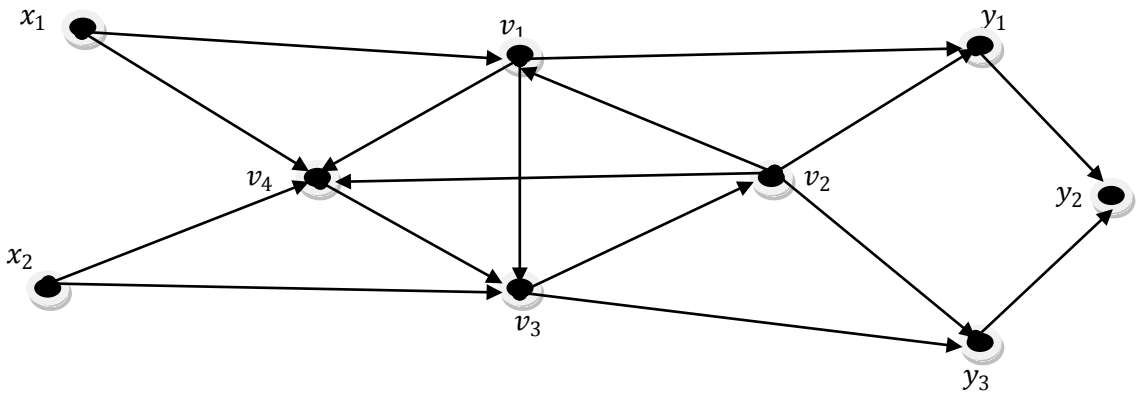


Figure 2.1.1 a network with two sources and three sinks

A network with multiple sources and multiple sinks can be reduced to a network of single source and sink. We will discuss this in detail in section 3.2. In this section, we will deal with networks with single source and single sink. So, let us define a network of single source and sink as a specific case definition 2.1.1.

Definition 2.1.2 [1]*:-* A network is defined as a digraph with a non-negative capacity $c(e)$ on each edge e and distinguished source vertex s and sink t.

*Example 2.1.2***:-**Figure 2.1.2 is a network with single source s and single sink t.

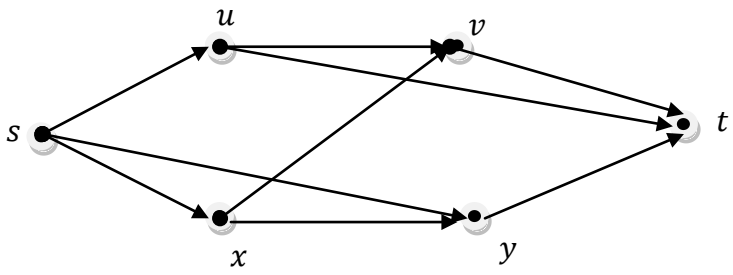


Figure 2.1.2 network of single source and sink

2.2. Flows

A flow f in a network N is an integer valued function defined on the arc set A of N [1]. The value $f(e)$ of f on an arc e can be taken as the rate at which material is transported along e under the flow f in transportation network.

Notations**: -** $f^+(v)$ stands for the total flow on edges leaving v and $f^-(v)$ for the total flow on edges entering v.

Definition 2.2.2 [1]*:*-A flow f is said to be feasible if it satisfies:

1) capacity constraints;

$$0 \leq f(e) \leq c(e) \text{ for each arc or edge } e$$

2) conservation constraints;

$$f^+(v) = f^-(v) \text{ for each vertex } v \notin \{s, t\}.$$

If S is a subset of the vertex set in a network N and f is a flow in N, then $f^-(S) - f^+(S)$ is called the resultant(net) flow into S, and $f^+(S) - f^-(S)$ is called the resultant (net) flow out of S, relative to f. From the conservation condition, one can easily see that the net flow out of any intermediate vertex is zero i.e.$f^+(v) = f^-(v)\ \forall v \in I$.

Theorem 2.2.1 [4]*:* - Given a flow f in a network N, the net flow out of the source s equals to the net flow into the sink t. i.e. $f^+(s) = f^-(t)$. This is what we call the value of the flow. i.e. $val(f) = f^+(s) = f^-(t)$

Proof: - Let V and A be the set of vertices and arcs of N respectively. We have $\sum_{j\in V}\sum_{i\in V} f_{ij} = \sum_{e\in A} f(e) = \sum_{j\in V}\sum_{i\in V} f_{ji}$ where $ij \in A$. That is

$\sum_{j\in V}\sum_{i\in V} f_{ij} = \sum_{j\in V}\sum_{i\in V} f_{ji}$. This implies that

$0 = \sum_{j\in V}[\sum_{i\in V} f_{ij} - \sum_{i\in V} f_{ji}]$ Separating the terms for $j = t$ and $i = s$, leaving the rest together, we get

$$0 = (\sum_{i\in V} f_{it} - \sum_{i\in V} f_{ti}) + (\sum_{i\in V} f_{is} - \sum_{i\in V} f_{si}) + \sum_{\substack{j\in V \\ j\neq s,t}} (\sum_{i\in V} f_{ij} - \sum_{i\in V} f_{ji})$$

$= \sum_{i\in V} f_{it} - \sum_{i\in V} f_{si}$, since $f_{ti} = f_{si} = 0$ for all $i \in V$ and conservation of flow, $\sum_{i\in V} f_{ij} - \sum_{i\in V} f_{ji} = 0$ if $j \in V\backslash\{s,t\}$.

Since by definition, $f^+(s) = \sum_{i\in V} f_{si}$ and $f^-(t) = \sum_{i\in V} f_{it}$

Therefore, $f^+(s) = f^-(t)$ ■

Note that also, the value of a flow f $val(f)$ is the net flow $f^+(t) - f^-(t)$ into the sink t or is the net flow $f^-(s) - f^+(t)$ out of the source s, and we can express it as $val(f) = f^+(t) - f^-(t) = f^-(s) - f^+(t)$.

Example 2.2.1: - From the assignments of figure 2.2.1 given in the following table

e	sb	bc	ct	sd	dc	de	et
$f(e)$	2	2	3	3	1	2	2

For example, the flow into vertex d $f_{sd} = 3$ is the same as the flow out of vertex d,i.e. $f_{sd} = f_{dc} + f_{de} = 1 + 2 = 3$ ■

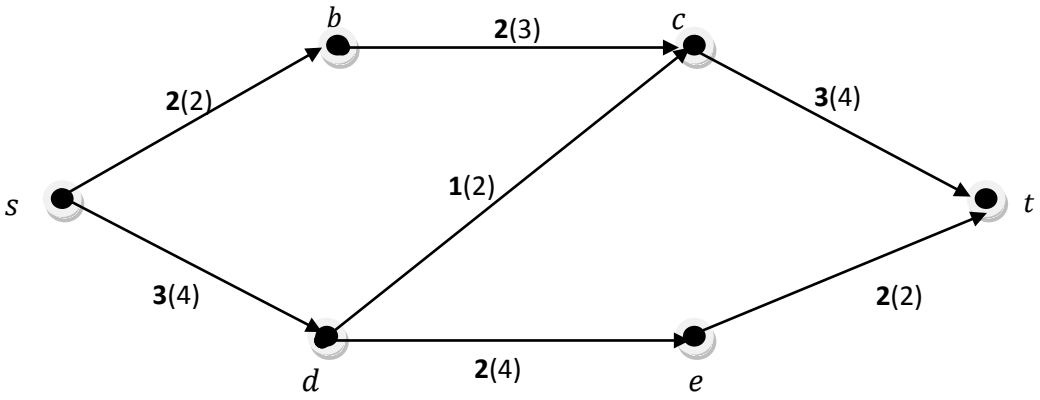


Figure 2.2.1: flow and capacity weighted network, $\boldsymbol{f(c)}$

A flow f in N is said to be a maximum flow if there is no flow f' in N such that $val(f') > val(f)$ [2]. A maximum flow is a feasible flow of maximum value.

As [2] puts the problem of determining a maximum flow in an arbitrary network with multiple sources and sinks can be reduced to the case of networks that have just one source and one sink by means of simple devices. Given a network N, construct an equivalent network N' as follows:

i. Adjoin two new vertices s and t to N;
ii. Join s to each vertex in X by an arc of unlimited capacity;
iii. Join each vertex in Y to t by an arc of unlimited capacity ;
iv. Designate s as the source and t as the sink of N'.

Figure 2.2.2 illustrates this procedure as applied to the network N of figure 2.1.1.

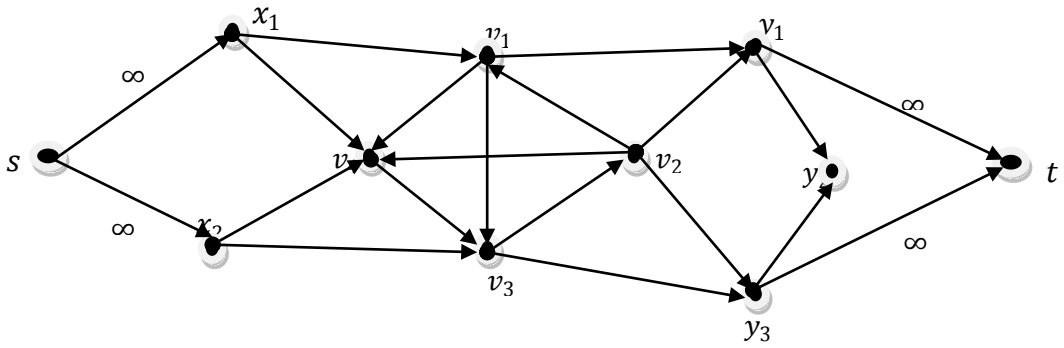


Figure 2.2.2 an equivalent network with single source and sink

Flows in N and N' correspond to one another in a simple way. If f is a flow in N such that the resultant flow out of each source and into each sink is non negative (it suffices to restrict our attention to such flows) then the function f' defined by

$$f'(e) = \begin{cases} f(e), & \text{if } e \text{ is an arc of N} \\ f^+(v) - f^-(v), & \text{if } e = (s, v) \\ f^-(v) - f^+(v), & \text{if } e = (v, t) \end{cases}$$

is a flow in N' such that $val(f') = val(f)$. Conversely, the restriction to the arc set of N of a flow in N' is a flow in N having the same value.

A path from the source to the sink with excess capacity would allow us to increase flow. In example 2.2.1, no path remains with excess capacity, but the flow f' with $f'(vs) = 0$ and $f(e) = 1$ for $e \neq vs$ has value 2. The flow f is "maximal" in the sense

that no other feasible flow can be found by increasing the flow on some edges, but f is not a maximum flow. We need a more general way to increase flow in network.

Definition 2.2.4[2]*:-*An arc e is said to be f**-***zero* if $f(e) = 0$, f-posetive if $f(e) > 0$, f**-** unsaturated if $f(e) < c(e)$ and f**-** saturated if $f(e) = c(e)$.

Example 2.2.2 In figure 2.2.3, arc dc is an f-zero arc, because $f(dc) = 0$, all the rest arcs in the network are f-posetive, arc ct, sd, bc and de are f**-** unsaturated and finally, arc sb and et are f**-** saturated.

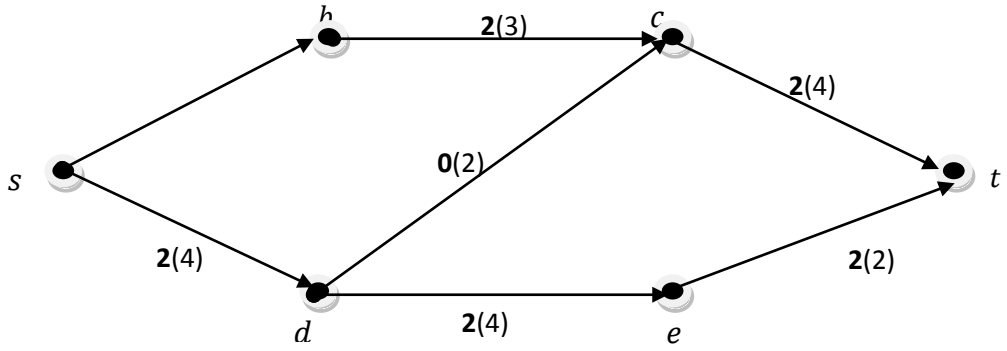


Figure 2.2.3 a network with flow value and capacity.

An edge e of p is said be a forward arc of p if it is directed from the source vertex to the sink and backward arc of p if it is directed from the sink vertex to the source. For instance figure 2.2.4 is a directed path graph, edge sb, bc, de and et are a forward on p and edge cd is a backward on p.

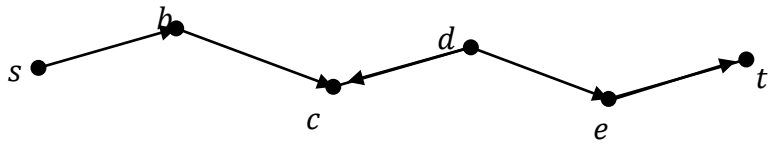


Figure 2.2.4 a directed $(\boldsymbol{s}, \boldsymbol{t})$- path

Definition 2.2.5 [1]*:-*For a feasible flow f in a network N, an f*-augmenting path* is a source-to-sink path p in the underlying graph G such that for each $e \in E(p)$,

a) $f(e) < c(e)$, if p follows e in the forward direction; and

b) $f(e) > 0$, If p follows e in the backward direction,

Definition 2.2.6 [1]*:* - Let $\varepsilon\,(e) = c(e) - f(e)$ when e is forward on p, and let $\varepsilon(e) = f(e)$ when e is backward on p.The *tolerance* of p is $\min_{e\in E(p)}\{\varepsilon(e)\}$

Note that the definition of f-augmenting path ensures that the tolerance is positive; as it is stated in the following lemma, this amount is the increase in the flow value.

Lemma 2.2.1 [1]*:-* If p is an f-augmenting path with tolerance z, then changing flow by $+z$ on edges followed forward by p and by $-z$ on edges followed backward by p produces a feasible flow f' with $val(f') = val(f) + z$.

Proof: - Here f' is defined as $f'(e) = \begin{cases} f(e) + z, & \text{if } e \text{ is a forward arc of } p \\ f(e) - z, & \text{if } e \text{ is a backward arc of } p \end{cases}$ (1)

Thus, we shall show that:

i. f' is a feasible flow , and

ii. $val(f') = val(f) + z$

i. To show f' is a feasible flow; we need to show that the *capacity* and *conservation* conditions are satisfied.

From the case of equation (1), we have $f'(e) = f(e) + z$ for every arc e on p. In addition, since always tolerance of a path is positive, i.e. $z \geq 0$

$f'(e) \geq f(e) \geq 0$ for every forward arc e on p (2)

From the definition of tolerance, $z = \min_{e\in E(p)} \varepsilon(e)$ where

$\varepsilon\,(e) = \begin{cases} c(e) - f(e), & \text{if } e \text{ is forward on } p \\ f(e), & \text{if } e \text{ is backward on } p \end{cases}$

$f'(e) = f(e) + \min_{e\in E(p)} \varepsilon(e)$ this gives that

$f'(e) \leq f(e) + c(e) - f(e)$ since $\varepsilon(e) = c(e) - f(e)$ for each arc e forward on p.

Thus, $f'(e) \leq c(e)$ for each arc e forward on p (3)

Therefore, by (2) and (3), we have

$0 \leq f'(e) \leq c(e)$ for each arc e forward on p (4)

Since $f'(e) = f(e)$ for those arc e not on p. Since f feasible flow and by (4), for all arc e of the network, we have $0 \leq f'(e) \leq c(e)$ (5)

The second case of equation (1), $f'(e) = f(e) - z$, if e is a backward arc of p

$f'(e) = f(e) - \min_{e \in E(p)} \varepsilon(e)$ and $\varepsilon(e) = f(e)$ for arc e is a backward arc of p.

Thus, $f'(e) \geq f(e) - f(e) = 0$ (6)

and from $f'(e) = f(e) - z$ we can have $f'(e) \leq f(e)$ this implies that

$0 \leq f'(e) \leq c(e)$ for e is a backward arc of p (7)

by (7) and since f feasible flow , we have

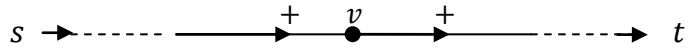
$0 \leq f'(e) \leq c(e)$ for all arc $e \in A(N)$ (8)

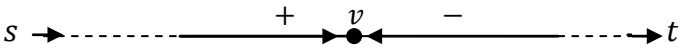
Thus by(5) and (8), the capacity constraint holds.

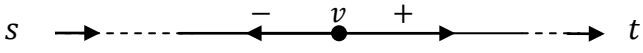
Now we need to show the conservation constraints. Since flow on vertices, which are not on p has not changed i.e.

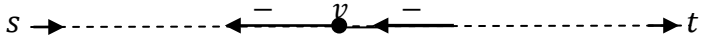
$(f')^+(v) = (f')^-(v)$ and $f^+(v) = f^-(v)$ for all $v \notin V(p)$ (9)

We need to check only vertices of p for the conservation constraints. The arcs of p incident to an internal vertex, v of p occurs in one of the following four ways as shown below.

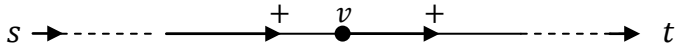
1) 

2) 

3) 

4) 

Now let us check each case.

1) 

$f'^-(v) = f^-(v) + z$ (10)

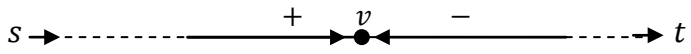
And $f'^+(v) = f^+(v) + z$ (11)

Then subtracting (10) from (11), we get

$$f'^{+}(v) - f'^{-}(v) = f^{+}(v) - f^{-}(v) = 0$$

Since f is feasible; it follows that $f'^{+}(v) - f'^{-}(v) = 0$

Therefore $f'^{+}(v) = f'^{-}(v)$ *for each internal vertex v of* p

2) 

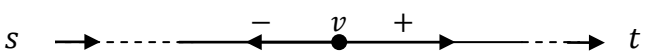
$$f'^{-}(v) = f^{-}(v) + z \quad (12)$$

And $f'^{-}(v) = f^{-}(v) - z$ (13)

Then summing (12), (13) and dividing both sides by 2 will give us

$$f'^{-}(v) = f^{-}(v) \quad (14)$$

Therefore, by (9) and (14), $f'^{+}(v) = f'^{-}(v)$ *for each internal vertex v of* p.

3) 

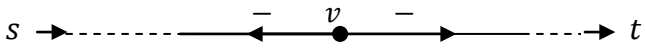
$$f'^{+}(v) = f^{+}(v) - z \quad (15)$$

and $f'^{+}(v) = f^{+}(v) + z$ (16)

Then summing (15), (18) and dividing both sides by 2 will give us

$$f'^{+}(v) = f^{+}(v) \quad (17)$$

Therefore, by (9) and (17), $f'^{+}(v) = f'^{-}(v)$ *for each internal vertex v of* p.

4) 

$$f'^{+}(v) = f^{+}(v) - z \quad (18)$$

and $f'^{-}(v) = f^{-}(v) - z$ (19)

Then subtracting (18) from (19), and since f is feasible flow we will have

$f'^{+}(v) - f'^{-}(v) = f^{+}(v) - f^{-}(v) = 0$ This implies $f'^{+}(v) - f'^{-}(v) = 0$

Therefore $f'^{+}(v) = f'^{-}(v)$ *for each internal vertex v of* p

Thus, in any of the four cases the conservation condition is satisfied. Hence f' is feasible.

ii. For the second part, let $T = \{v_1, v_2, \dots \dots \dots, v_n\}$ be the set of vertices, which are joined to the source s. That is, T is the head set of all arcs joined to s.

Clearly, no edges enter to s in the network $val(f') = f'^{+}(s)$

$$(f')^{+}(s) = \sum_{i=1}^{n} f'_{sv_i} \tag{20}$$

and $$f^{+}(s) = \sum_{i=1}^{n} f_{sv_i} \tag{21}$$

the f-augmenting path p uses exactly one arc among all arcs $\{sv_1, sv_2, \dots \dots \dots, sv_n\}$ with out low of generality let it be sv_2, so

$$f'_{sv_i} = f_{sv_i} \text{ for } i \neq 2 \tag{22}$$

$$f'^{+}(s) = \sum_{i=1\&i\neq 2}^{n} f'_{sv_i} + f'_{sv_2} \tag{23}$$

In addition, since sv_2 is a forward arc of p,

$$f'_{sv_2} = f_{sv_2} + z \tag{24}$$

Therefore, from (22), (23) and (24) we have

$$f'^{+}(s) = \sum_{i=1\&i\neq 2}^{n} f_{sv_i} + f(sv_2) + z$$

$$= \sum_{i=1}^{n} f_{sv_i} + z \tag{25}$$

Now by (13), $f'^{+}(s) = f^{+}(s) + z$

Therefore, $val(f') = val(f) + z$

Thus, the existence of an f-augmenting path p in a network is significant since it implies that f is not a maximum flow; in fact, by sending an additional flow of $\varepsilon(p)$ along p.■

Example 2.2.2:-if f is the flow indicated in the network of figure 2.2.5**,** then one f-augmenting path is the path $p = sv_1v_2v_3t$. The forward arcs of p are (s, v_1) and (v_3, t) and $\varepsilon(p) = 2$. This amount is the increment in the flow value of network of figure 2.2.5. Figure 2.2.6, shows the revised flow for the network figure 2.2.5, based on the f-augmenting path $sv_1v_2v_3t$. However, the increment in the flow value of

network of figure 2.2.6 cannot be increased because it does not have an f'-augmenting path where f' is the revised flow. This case of relationship between a maximum flow and an f-augmenting path is discussed under theorem 2.3.1.

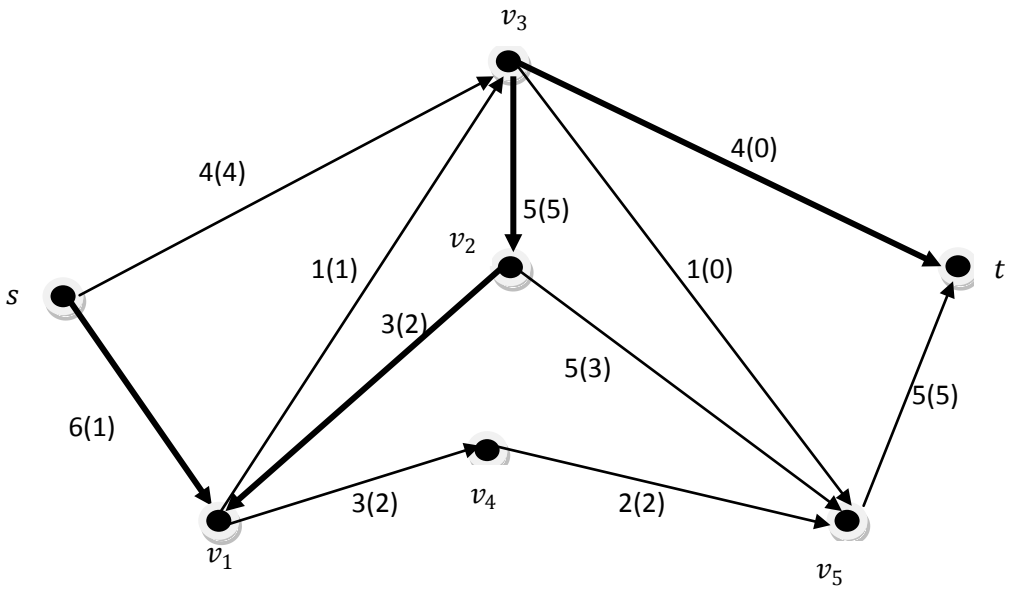


Figure 2.2.5 a network with an $\boldsymbol{f}$-augmenting path

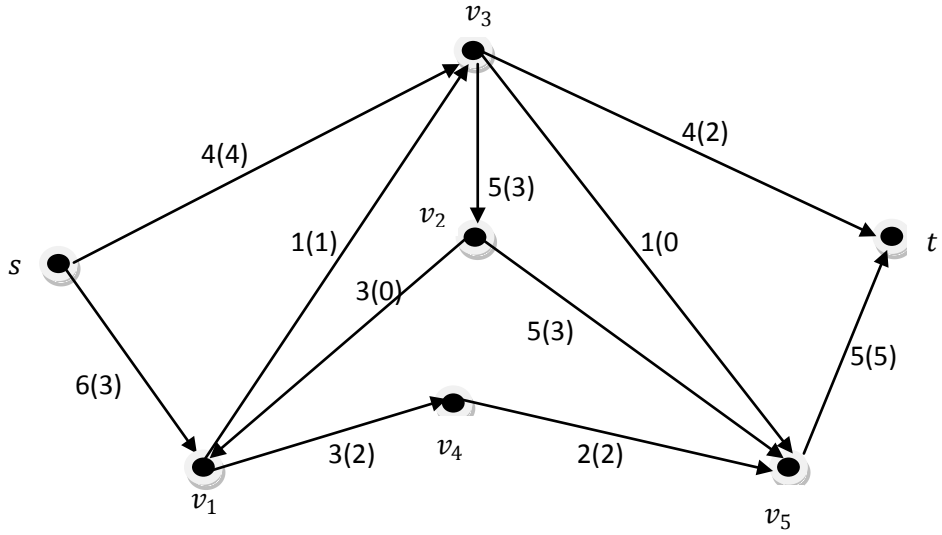


Figure 2.2.6 a revised flow

2.3. Cuts

Definition 2.3.1 [2]:-Let N be a network with a single source s and a single sink t. A cut in N is a set of arcs of the form $[S, \overline{S}]$, where $S \subseteq V(N)$. In other words, a set of arcs of the form $(v_{i,} v_j)$ where $v_{i,} \in S$ and $v_j \in \overline{S}$ is a cut in N, and denoted by$[S, \overline{S}]$. In the network of figure 2.3.1, heavy lines indicate a cut.

The capacity of a cut K is the sum of the capacities of its arcs. We denote the capacity of K by $capK$; thus

$$capK = \sum_{e \in K} c(e)$$

Example 2.3.1: - In figure 2.3.1, the cut indicated by the heavy lines has capacity 16. By taking $S = \{x, x_1, x_2, v_3, v_4\}$

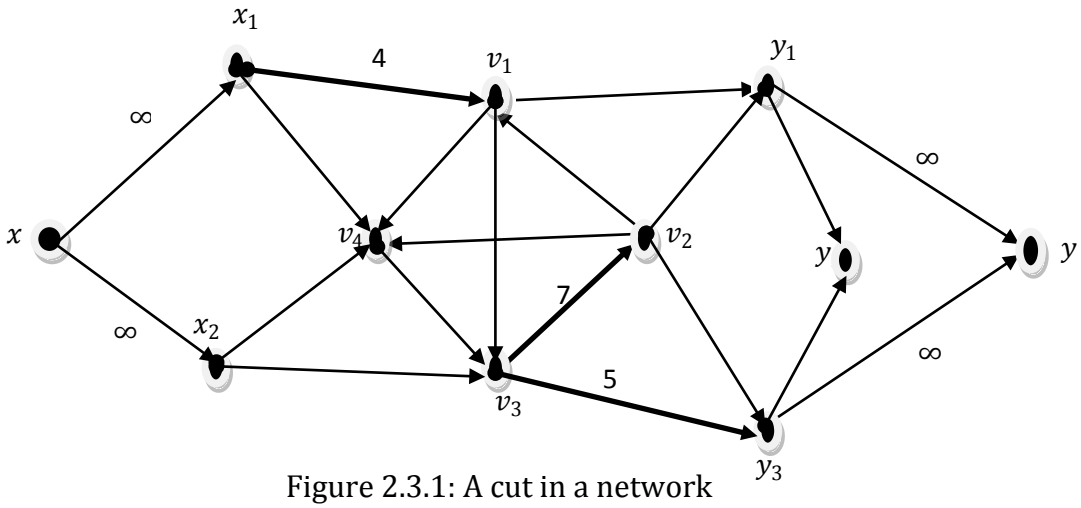


Figure 2.3.1: A cut in a network

Definition 2.3.1 [1]*:* - In a network, a source/sink cut $[S, \overline{S}]$ consists of the edges from a source set S to a sink set $\overline{S}$, where S and $\overline{S}$ partition the set of vertices, with $s \in S$ and $t \in \overline{S}$.in other words, a cut $[S, \overline{S}]$ where $s \in S$ and $t \in \overline{S}$ is known as a source/sink cut.

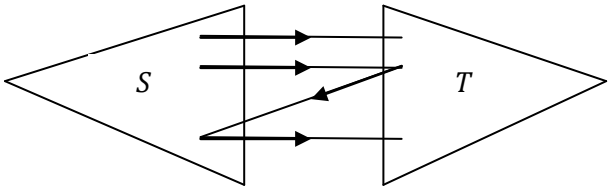


Figure 2.3.2 a cut

Recall that in a digraph $[S,\overline{S}]$, denotes the set of edges with tail in S and head in $\overline{S}$. Thus the capacity of a cut $[S,\overline{S}]$ is completely unaffected by edges from $\overline{S}$ to S. Given a cut $[S,\overline{S}]$, every s,t-path uses at least one edge of $[S,\overline{S}]$, so intuition suggests that the value of a feasible flow should be bounded by $cap(S,\overline{S})$. To make this precise, we extend the notion of net flow to the sets of vertices. Let $f^+(U)$ denote the total flow on edges leaving U, and let $f^-(U)$ be the total flow on edges entering U, for any set of vertices such that $U \subseteq V(N)$. The net flow out of U is then $f^+(U) - f^-(U)$.

Lemma 2.3.1 [1]*: -* If U is a set vertices in a network, then the net flow out of U is the sum of the net flows out of the vertices in U. In particular, if f is a feasible flow and $[S,\overline{S}]$, is a source/sink cut, then the net flow out of S and the net flow into $\overline{S}$ are both equal to $val(f)$.

Proof: - The stated claim is that

$$f^+(U) - f^-(U) = \sum_{x\in U}[f^+(x) - f^-(x)] \qquad (1)$$

We consider the contribution of the flow f_{xy} on the arc xy to each side of (1). Thus, we have the following four cases.

Case 1:- If $x, y \in U$, then f_{xy} is not counted on the left, but it contributes positively (via $f^+(x)$) and negatively (via $f^-(x)$) on the right of (1) that is the right side of (1), $\sum_{x\in U}[f^+(x) - f^-(x)] = 0$ Since $f^+(x) - f^-(x) = f_{xy} - f_{xy} = 0$ thus, this case satisfies the equality.

i.e. $0 = f^+(U) - f^-(U) = \sum_{x\in U}[f^+(x) - f^-(x)]$ (1)

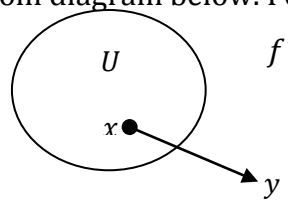
and (1) can be expressed as

$0 = f^+(U) - f^-(U) = \sum_{x\in U}[f^+(x) - f^-(x)]$

$$0 = \sum_{x,y \in U}[f_{xy} - f_{yx}] = \sum_{x,y \in U} f_{xy} - \sum_{x,y \in U} f_{yx} \quad (2)$$

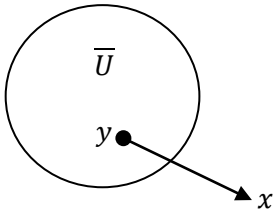
Case 2:- If $x, y \notin U$, then f_{xy} contributes to neither sum. This one is neglected.

Case 3:- If $xy \in [U, \overline{U}]$, then f_{xy} contributes positively to each sum. That is, as we can see from diagram below. For $xy \in [U, \overline{U}]$,



$$f^+(U) = \sum_{xy \in [U,\overline{U}]} f_{xy} \quad (3)$$

Case 4:- If $xy \in [\overline{U}, U]$, then f_{xy} contributes negatively to each sum. That is, as we can see from the diagram below. For $xy \in [\overline{U}, U]$



$$-f^-(U) = -\sum_{yx \in [\overline{U},U]} f_{yx} \quad (4)$$

Therefore, by adding (2), (3) and (4) that is

$$+ \begin{cases} 0 = \sum_{x,y \in U} f_{xy} - \sum_{x,y \in U} f_{yx}, \\ f^+(U) = \sum_{xy \in [U,\overline{U}]} f_{xy} \quad \text{and} \\ -f^-(U) = -\sum_{yx \in [\overline{U},U]} f_{yx} \end{cases}$$

$$f^+(U) - f^-(U) = \sum_{x,y \in U} f_{xy} - \sum_{x,y \in U} f_{yx} + \sum_{yx \in [U,\overline{U}]} f_{yx} - \sum_{xy \in [\overline{U},U]} f_{xy}$$

$$f^+(U) - f^-(U) = (\sum_{xy \in [U,\overline{U}]} f_{xy} + \sum_{x,y \in U} f_{xy}) - (\sum_{yx \in [\overline{U},U]} f_{yx} + \sum_{x,y \in U} f_{yx}) \quad (5)$$

But the right side of (5) can be expressed separately as

$$\sum_{x \in U} f^+(x) = \sum_{xy \in [U,\overline{U}]} f_{xy} + \sum_{x,y \in U} f_{xy} \quad \text{and}$$

$$\sum_{x \in U} f^-(x) = \sum_{yx \in [\overline{U},U]} f_{yx} + \sum_{x,y \in U} f_{yx}$$

therefore, from (5) we get

$f^+(U) - f^-(U) = \sum_{x \in U} f^+(x) - \sum_{x \in U} f^-(x)$ and taking the sum overall $x \in U$ yields

$$f^+(U) - f^-(U) = \sum_{x \in U} [f^+(x) - f^-(x)] \quad \blacksquare$$

- But note that, $\sum_{v \in U} f^+(v) \neq f^+(U)$ and $\sum_{v \in U} f^-(v) \neq f^-(U)$.

Given that $[S, \overline{S}]$ is a source/sink cut, a source vertex s with sink vertex t and f is feasible flow, we can use our result to show that

$$val(f) = f^+(S) - f^-(S) \text{ and } \; f^+(\overline{S}) - f^-(\overline{S}) = -val(f)$$

$$\sum_{x \in S}[f^+(x) - f^-(x)] = \begin{cases} f^+(s) - f^-(s), & \text{if } x = s \\ 0 \quad , & \text{if } x \neq s \end{cases}$$ - If x is intermediate vertex

Therefore, the sum reduced to

$$\sum_{x \in S}[f^+(x) - f^-(x)] = f^+(s) - f^-(s) \qquad (6)$$

This gives that $val(f) = f^+(s) - f^-(s) = f^+(S) - f^-(S)$; the net flow from vertices of S sums to $f^+(s) - f^-(s)$ and similarly the net flow from vertices of $\overline{S}$ sums to $f^+(t) - f^-(t)$, which equals – $val(f)$.

I.e. $f^+(\overline{S}) - f^-(\overline{S}) = -val(f)$ therefore, for any flow f and a source/sink cut $[S, \overline{S}]$ in N. $val(f) = f^+(S) - f^-(S)$, or $val(f) = f^-(\overline{S}) - f^+(\overline{S})$ or in other words, the net flow across any source/sink cut equals both the net flow out of s and the net flow into t.

Corollary 2.3.1[1] **(**Weak duality):-

If f is a feasible flow and $[S, \overline{S}]$ is a source/sink cut, then $val(f) \leq cap(S, \overline{S})$. Furthermore [2], equality holds if and only if each arc in $[S, \overline{S}]$ is f-saturated and each arc in $[\overline{S}, S]$ is f-zero.

Proof:-by lemma 3.3.1, the value of f equals the net flow out of S. Thus

$$val(f) = f^+(S) - f^-(S) \leq f^+(S)$$

$$val(f) \leq f^+(S) \qquad (1)$$

Since the flow into S, $f^-(S) \geq 0$. And by definition

$$f^+(S) = \sum_{e \in [S, \overline{S}]} f(e) \leq \sum_{e \in [S, \overline{S}]} c(e) = cap(S, \overline{S})$$

Therefore $\; f^+(S) \leq cap(S, \overline{S}) \qquad (2)$

Hence $val(f) \leq cap(S, T)$.

The second statement follows, $f^+(S) = cap(S, \overline{S})$ if and only if $f(e) = c(e)$, for all arc $e \in [S, \overline{S}]$ and $f^-(S) = 0$ if and only if $f(e) = 0$, for all arc $e \in [\overline{S}, S]$. Thus,

equation (1) becomes $val(f) = f^{+}(S) = cap(S, \overline{S})$ if and only if each arc in $[S, \overline{S}]$ is f-saturated and each arc in $[\overline{S}, S]$ is f-zero ■

A cut K in N is a *minimum cut* if there is no cut K' in N such that $capK' < capK$. If f^* is a maximum flow, and $\widetilde{K}$ is a minimum cut [2], we have a special case for the weak duality stated above which would be seen later on.

Corollary 2.3.2 [2]*: -* Let f be a flow and K be a cut such that $val(f) = capK$.Then f is a maximum flow and K is a minimum cut.

Proof: - Let f^* be a maximum flow and $\overline{K}$ a minimum cut. Then by weak duality condition, for any flow f and cut $K = [S, \overline{S}]$, $val(f) \leq capK$. Therefore, for maximum f^* flow and minimum cut $\overline{K}$. We have

$$val(f) \leq val(f^*) \leq cap\overline{K} \leq capK$$

Since, by hypothesis, $val(f) = capK$ it follows that $val(f^*) = val(f)$ and $capK = cap\widetilde{K}$. Thus, f is a maximum flow and K is a minimum cut.

*Theorem 2.3.1:-*A flow f in a network N is a maximum flow if and only if N contains no f-augmenting path.

Proof:-Let f be a maximum flow in N.

Suppose N contains an f-augmenting path p. This implies that f cannot be a maximum flow since f', the revised flow on p has a large value. Thus it contradicts the fact that f is a maximum flow.

Conversely, suppose that N contains no f-augmenting path.

<u>*Claim*</u>*: -* f is a maximum flow

Let S denote the set of all vertices to which s is connected by f-unsaturated paths in N. Clearly $s \in S$ also, since N has no f-augmenting path $t \in \overline{S}$. Thus $K = (S, \overline{S})$ is a cut in N.

We shall show that:-

a. Each arc in $(S, \overline{S})$ is f-saturated, and
b. Each arc in $(\overline{S}, S)$ is f-zero.

Consider an arc e with tail $u \in S$ and head $v \in \overline{S}$. Since $u \in S$, there exists an f-unsaturated (s,u)- path Q. If $e = (u,v)$ f-unsaturated, then Q could be extended by the arc e to yield an f-unsaturated (s,v)-path;-$Q + e$.

However $v \in \overline{S}$, and so there is no such path. Therefore, e must be f-saturated.

Similarly, consider an arc e' with tail $u \in \overline{S}$ and head $v \in S$ i.e. $e' \in (\overline{S}, S)$, since N does not contain an f-augmenting path, $f(e^{'}) = 0$ for all $e^{'} \in (\overline{S}, S)$. Therefore, e' must be f-zero.

Since the duality theorem equality holds if and only if each arc in $(S, \overline{S})$, is f-saturated and each arc in $(\overline{S}, S)$ is f-zero. We obtain

$val(f) = capK$ By corollary 2.3.2, this implies that f is maximum flow■.

Ford-Fulkerson labeling algorithm

This algorithm helps us to find a maximum flow in a network [2]. It is also called as *the labeling method.* Starting with a known flow, for instance the zero flow, it recursively constructs a sequence of flows of increasing value, and terminates with a maximum flow. After the construction of each new flow f, a subroutine called the labeling procedure is used to find an f-augmenting path, if one exists. If such a path p is found, then f^*, the revised flow based on p, is constructed and taken as the next flow in the sequence. If there is no such a path, the algorithm terminates and the original flow is still a maximum flow. The algorithm is given below as [1] put in precise manner.

Input: A feasible flow f in a network,

Output: An f- augmenting path or a cut with capacity $val(f)$.

Idea: Find the vertices reachable from s by paths with positive tolerance. Reaching t completes an f-augmenting path. During the search, R is the set of vertices labeled reached, and S is the subset of R labeled searched.

Initialization: $R = \{s\},\ S = \emptyset$.

Iteration: Choose $v \in R\backslash S$

For each existing edge vw with $f(vw) < c(vw)$ and $w \notin R$, add w to R.

For each entering edge uv with $f(uv) > 0$ and $u \notin R$, add u to R.

Label each vertex added to R as "reached", and record v as the vertex reaching it. After exploring all edges at v, add v to S. If the sink t has been reached (put in R), then trace the path reaching t to report an f-augmenting path and terminate.

If $R = S$, then return the cut $[S, \overline{S}]$ and terminate. Otherwise, iterate.

Example 2.3.2: On the figure, 2.3.3(a) is a network with the flow f. We run the labeling algorithm. First, we search from s and find excess capacity to u and x, labeling them reached. Now we have $u, x \in R\backslash S$. There is no excess capacity on uv or xy, so searching from u reaches nothing, and searching from x does not reach y. However, there is nonzero flow on vx. Thus, we label v from x now v is the only element of $R\backslash S$, and searching from v reaches t. We labeled t from v, v from x, and x from s, so we have found the augmenting path s, x, v, t. In figure 3.3.3(a) and (b), the capacities are in bold and flows are in brace.

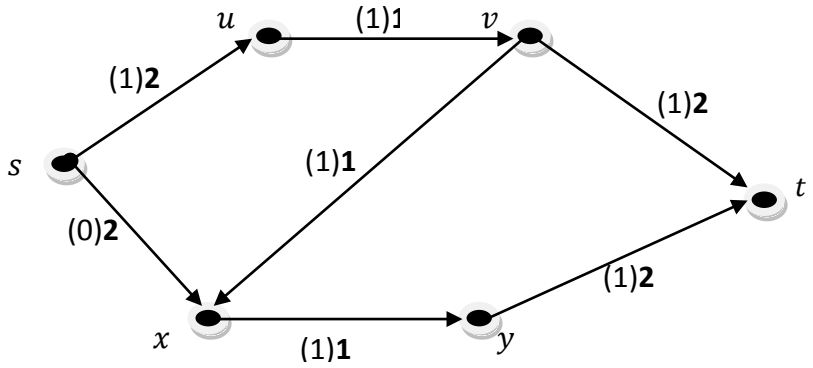


Figure 2.3.3(a): Network flow

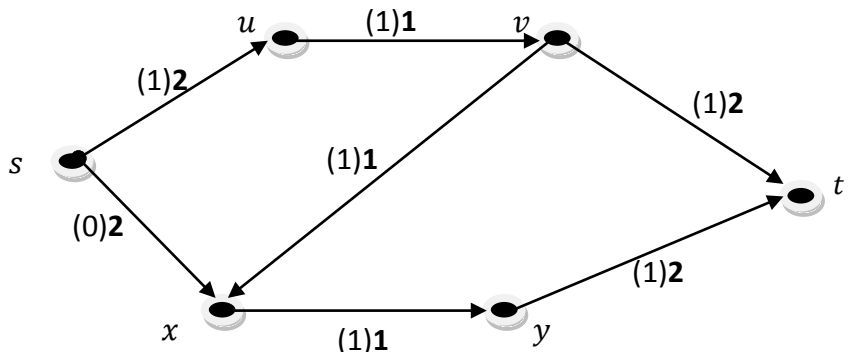


Figure 3.3.3(b): Network flow

The tolerance on this path is one, so the augmentation increases the flow value by one. In the new flow f' shown on the right, every edge has unit flow except $f'(vx) = 0$. When we run the labeling algorithm again, we have excess capacity on su and sx and label $\{u, x\}$, but from these vertices we can label no others. We terminate with $R = S = \{s, u, x\}$. The capacity of the resulting cut $[S, \overline{S}]$ is two, which equals $val(f')$ proves that f' is a maximum flow.

Note that repeated use of the labeling algorithm allows us to solve the maximum flow problem and prove the strong duality relationship [1].

2.4. Maximum flow and Minimum cut (Dual optimization problems)

In this section, we shall state the integral theorem and the max-flow min-cut theorem, which we use them for further applications and use them to obtain or prove different theorems in graph theory.

Definition 2.4.1:- A flow having an integer amount on each edge of a network is said to be an integral flow.

In combinatorial applications, we typically have integer capacities and want a solution in which the flow on each edge is an integer.

Theorem 2.4.1 (Integral theorem) [1]*:-*

If all capacities in a network are integer, then there is a maximum flow assigning integral flow to each edge. Furthermore, some maximum flow can be partitioned into flows of unit value along paths from source to sink.

Proof: In the labeling algorithm of Ford and Fulkerson, the change flow value when an augmenting path is found is always a flow value or the difference between a flow value and capacity. When these are integers, the difference is also an integer. Starting with the zero flow, this implies that there is no first time when a non-integer flow appears.

The algorithm thus produces a maximum flow with integer flow on each edge. At each internal node, we now match units of entering flow to units of exiting flow. This forms s, t-path and perhaps cycles. If a cycle arises, then we decrease flow on its edges by one to eliminate it without changing the flow value. This leaves $val(f)$ paths from to t, each corresponding to a unit of flow ■

The integrality theorem yields paths of unit flow. In applications, we build networks where these units of flow have meaning.

In section 2.2, we have introduced a non-negative number $\varepsilon(p)$, *the tolerance of a path* p in a network N, and a flow f in N as

$\varepsilon(p) = \min_{e \in E(p)} \varepsilon(e)$, where $E(p)$ is the edge set of the path p and

$$\varepsilon(e) = \begin{cases} c(e) - f(e), & if\ e\ is\ a\ frward\ arc\ of\ p \\ f(e), & if\ e\ is\ a\ backward\ \ arc\ of\ p \end{cases}$$

As may easily be seen, $\varepsilon(p)$ is the largest amount by which the flow along p can be increased (relative to f) without violating capacity constraints.

Definition 2.4.2 [2]*:* The path p is said to be f*-saturated* if $\varepsilon(p) = 0$ and f*-unsaturated* if $\varepsilon(p) > 0$ (or, equivalently, if each forward arc of p is f-unsaturated and each reverse (backward direction) arc of p is f-posetive). Simply, an f-unsaturated path is one that is not being used to its full capacity. Thus, we can define an f-augmenting path as it is an f-unsaturated path from the source s to the sink t.

Theorem 2.4.2 [Max-flow min-cut theorem] [1]

In any network, the value of a maximum flow is equal to the capacity of a minimum cut.

Proof:-In max flow problem, the zero flow ($f(e) = 0, \forall_e$) is always a feasible flow and gives as a place to start. Given a feasible flow, we apply the labeling algorithm. It iteratively adds vertices to S(each vertex at most once) and terminates with $t \in R$, or with $S = R$.

In the case where it terminates with $t \in R$, we have an f-augmenting path and increase the flow value. We then repeat the labeling algorithm.

When the capacities are rational, each augmentation increases the flow by a multiple $1/a$, where a is the least common multiple of the denominators, so after finitely many augmentations the capacity of some cut is reached. The labeling algorithm then terminates with $S = R$.

When terminating this way, we claim that $[S, T]$ is a source/sink cut with capacity $val(f)$, where $T = \overline{S}$ and f is the present flow. It is a cut because $s \in S$ and $t \notin R = S$.

Since applying the labeling algorithm to the flow f introduces no vertices of T into R, no edge from S to T has excess capacity, and no edge from T to S has non zero flow in f.

Hence $f^+(S) = cap(S,T)$ and $f^-(S) = 0$

Since the net flow out of any set containing the source but not the sink is $val(f)$,

Thus, $val(f) = f^+(S) - f^-(S) = f^+(S) = cap(S,T)$

This gives that $val(f) = cap(S,T)$ ■

3. APPLICATIONS OF NETWORKS

3.1. Menger's theorems

Now we shall use the max flow min cut theorem and integral theorem to obtain different theorems as well as menger's theorem. The following lemma provides a basic link.

Lemma 3.1.1 [2]*: -* Let N be a network with source s and sink t in which each arc has unit capacity. Then

a) The value of a maximum flow in N is equal to the maximum number m of arc-disjoint directed (s,t)-paths in N; and
b) The capacity of a minimum cut in N is equal to the minimum number n of arcs whose deletion destroys all directed (s,t)-paths in N.

Proof: - (a) let f^* be a maximum flow in N and let D^* dente the digraph obtained from D by deleting all f^*-zero arcs. Since each is of N has unit capacity. $f^*(e) = 1$ for all $e \in A(D^*)$ (the arc set of D^*). It follows that

i. $d_{D^*}{}^+(s) - d_{D^*}{}^-(s) = val(f^*) = d_{D^*}{}^-(t) - d_{D^*}{}^+(t)$; and
ii. $d_{D^*}{}^+(v) = d_{D^*}{}^-(v)$ for all $v \in V(D^*)\backslash\{s,t\}$

Recall that if D is a digraph satisfying [2]

i. $d_D{}^+(s) - d_D{}^-(s) = l = d_D{}^-(t) - d_D{}^+(t)$; and
ii. $d_D{}^+(v) = d_D{}^-(v)$ For all $v \in V(D)\backslash\{s,t\}$ then there exist l-arc-disjoint directed (x,y)-paths in D

Therefore, there exist $val(f^*)$ arc-disjoint directed (s,t)-paths in D^*, and hence also in D.

Thus $val(f^*) \leq m$ (1)

Now let $p_1, p_2, \ldots\ldots, p_m$ be any system of m arc-disjoint directed (s,t)-paths in N, and define a function f on A by

$$f(e) = \begin{cases} 1, & \text{if e is an arc of } \bigcup_{i=1}^{m} p_i \\ 0, & \text{otherwise} \end{cases} \quad (2)$$

Since $val(f) = f^+(S) - f^-(S)$ and $f^-(S) = 0$ by (2) thus $val(f) = f^+(S) = m$

Since f^* is a maximum flow, we have

$$val(f^*) \geq val(f) = m \quad (3)$$

Therefore, by (1) and (3), we have

$$val(f^*) = m \blacksquare$$

b) Let $\overline{K} = (S, \overline{S})$ be a minimum cut in N. Then, in $N \backslash \overline{K}$, no vertex of $\overline{S}$ is reachable from any vertex in S; In particular, it is not reachable from s.

Thus $\overline{K}$ is a set of arcs whose deletion destroys all directed (s, t)- paths, and we have

$$cap\ \overline{K} = \overline{|K|} \geq n \quad (4)$$

Now, let Z be the set of n arcs whose deletion destroys all directed (s, t)-paths, and denote the set of all vertices reachable from s in $N \backslash Z$ by S.

Since $s \in S$ and $t \in \overline{S}$, $K = (S, \overline{S})$ is a cut in N. Moreover, by definition of a cut set, $N \backslash Z$ can contain no arc of $(S, \overline{S})$, and so $K \subseteq Z$. Since $\overline{K}$ is a minimum act, we conclude that

$$cap\ \overline{K} \leq cap\ K = |K| \leq |Z| = n$$

$$cap\overline{K} \leq n \quad (5)$$

Therefore, by (4) and (5), we have $cap\overline{K} = n \blacksquare$

Theorem 3.1.1[2]: - Let s and t be two vertices of a digraph D. Then the maximum number of arc-disjoint directed $(s,)$-paths in D is equal to the minimum number of arcs whose deletion destroys all directed (s, t)- paths in D.

Proof: - we obtain a network N with source s, and sink t by assigning unit capacity to each arc of D.

Let m be the maximum number of arc-disjoint directed (s, t)-paths in N.

Let n be the minimum number of arc whose deletion destroys all directed (s, t) paths in N.

Claim: $m = n$

Let f be the maximum flow in N. Thus, by lemma 4.1.1 (a),

$$val\ (f) = m \quad (1)$$

Let S be the set of vertices of N containing the source s not the sink t, such that $K = (S, \overline{S})$ is the minimum cut in N with $c(e) = 1$ for each $e \in K$

Thus by lemma 3.1.1(b),

$$cap\ K = n \quad (2)$$

By max-flow min-cut theorem, the maximum flow in a network is equal to the minimum cut in a network

$$cap\ K = val\ f \quad (3)$$

Therefore, by (1), (2) and (3) $m = n$ ■

Theorem 3.1.2 [2]*:* - let s and t be to two vertices of a graph G. Then the maximum number of edge-disjoint (s, t)-paths in G is equal to the minimum number of edges whose deletion descroys all (s, t)-paths in G.

Proof: - let m = the maximum number of edge-disjoint (s, t)- paths in G; and

n = the minimum number of edges whose deletion destroys all (s, t)-paths in G.

Claim**:** $m = n$

Let $D(G)$ be the associated digraph of G.i.e.we replace each edge e of G by two oppositely oriented arcs with the same ends as e. Thus, by theorem 3.1.1, the maximum number of arc- disjoint directed (s, t) paths in $D(G)$ is equal to the minimum number of arcs whose deletion destroys all directed (s, t) paths in $D(G)$. Since there is a one-to-one correspondence between paths in G and directed paths in $D(G)$.

m = the maximum number of arc disjoint directed (s, t)-paths in $D(G)$.

In addition, because of the bijection "$D(G)$ is k-arc-connected if and only if G is k-edge-connected." The minimum cut of G is equal to the minimum cut of a digraph $D(G)$. By the same argument

n = the minimum number of arcs whose deletion destroys all directed (s,t)-paths in D (G).

Therefore, since $D(G)$ is a directed graph containing s & t, by theorem 3.1.1,

$$m = n \blacksquare$$

Theorem 3.1.3 [2]*:* - Let s & t be two vertices of a digraph D, such that s is joined to t. Then the minimum number of internally disjoint directed (s,t)-paths in D is equal to the minimum number of vertices whose deletion destroys all directed (s,t)-paths in D.

Proof: - Construct a new digraph D' from D as follows:

i. Split each vertex $v \in V \setminus \{s,t\}$ in to two new vertices v' and v'', and join them by an are (v', v'');

ii. Replace each arc of D with head $v \in V \setminus \{s,t\}$ by a new arc with head v', and each arc of D with tail $v \in V \setminus \{s,t\}$ by a new arc with tail v''. For instance, see figure 3.1.1, which illustrates this construction as an example.

Now to each directed (s,t)-path in D' there corresponds a directed (s,t)-path in D obtained by contracting all arcs of (v', v''); and conversely, to each directed (s,t)-path in D, there corresponds a directed (s,t)-path in D obtained by splitting each internal vertex of the path.

Furthermore, two directed (s,t)-paths in D' are arc-disjoint if and only if the corresponding paths in D are internally disjoint. It follows that the maximum number of arc-disjoint directed (s,t)-paths in D' is equal to the maximum number of internally disjoint directed (s,t)-paths in D.

Similarly, to each arc of type (v', v'') in D' there corresponds to a vertex v of D, obtained by removing the arc and over lapping them. Conversely, for each $v \in V \setminus \{s,t\}$ in D, there corresponds to an arc (v', v'') in D obtained by splitting v and joining them.

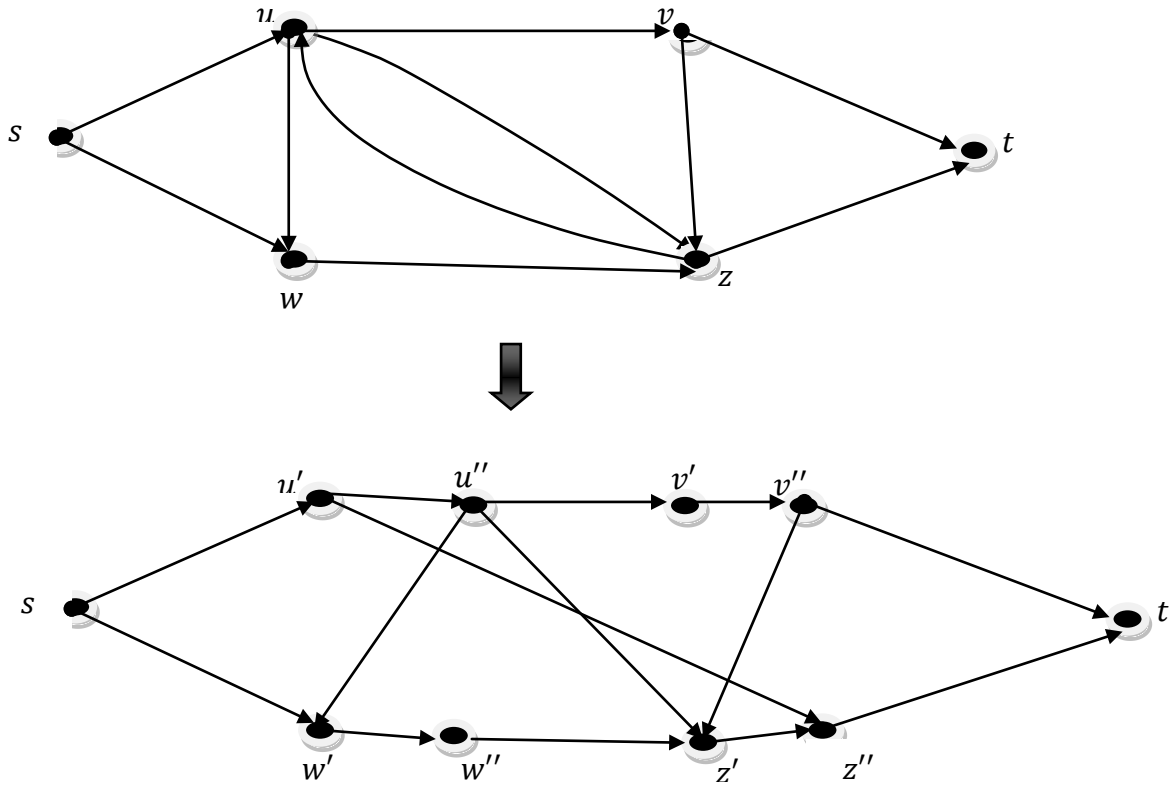


Figure 3.1.1 construction of a network

Let S be the set of arcs of type $(v^{'}, v^{''})$ in D' ; and T be the set of vertices, which obtained by over lapping the ends of an arc of type $(v^{'}, v^{''})$.

Then removing/deleting the arcs in S destroys all the (s,t)-paths in D' if and only if removing the vertices in T destroys all the (s,t)-paths in D. Therefore, the minimum number of arcs in D' whose deletion destroys all directed (s,t)-paths is equal to the minimum number of vertices in D whose deletion destroys all directed (s,t)-paths.

Thus, from the previous theorem, " the maximum number of arc-disjoint directed (s,t)-path is equal to the minimum number of arcs whose deletion destroys all directed (s,t)-paths.

The maximum number of internally disjoint directed (s,t)-paths in D is equal to the minimum number of vertices whose deletion destroys all directed (s,t)-paths in D.■

Theorem 3.1.4 [1]*:* - Let x and y be two non-adjacent vertices of a graph G. Then the maximum number of internally disjoint (x, y)-paths in G is equal to the minimum number of vertices whose deletion destroys all (x, y)-paths.

Proof: - Let S be a set of vertices such that $S \subseteq V(G) - \{x, y\}$ whose deletion destroys all (x, y)-paths in G. Let $k(x, y)$ be the minimum number of vertices whose deletion destroys all (x, y)-paths, the minimum size of S; and let $\lambda(x, y)$ be the maximum number of internally disjoint (x, y)-paths. Any S must contain an internal vertex of every (x, y)-path, and no vertex can cut two internally disjoint (x, y)-paths.

Therefore, always $k(x, y) \geq \lambda(x, y)$ (1)

Thus, the main proof of this theorem is the converse one, which we need to use network flow method.

Let $D(G)$ be the associated digraph of G. Clearly, x and y are vertices of $D(G)$. Therefore, we can view $D(G)$ as a network with source x and sink y and capacity 1on every edge. By integral theorem, capacity 1 ensures that units flow from x to y correspond to pair wise edge-disjoint (x, y)-paths in $D(G)$. Thus, a flow of value k yields a set of k such paths.

Thus, $\lambda(x, y) \geq$ Max $val(f)$ (2)

Let $K = (R, \overline{R})$ be a cut where $x \in R \subseteq V(D(G))$ but $y \notin R$ also K defines a set of edges whose deletion makes y unreachable from x. Since every capacity is 1 $|K| = capK$. Every (x, y)-path in a network uses at least one edge of a cut K.

Therefore, min$capK \geq k(x, y)$ (3)

By max-flow min-cut theorem, $\lambda(x, y) \geq max\ val(f) = min\ capK \geq k(x, y)$ this gives that, $\lambda(x, y) \geq k(x, y)$ (4)

Hence, by (1) and (4) as claimed■

3.2. Maximum bipartite matching

Under this section, we will discuss one of the applications of a network flow problem, called maximum bipartite matching.

Recall that undirected graph $G = (V, E)$ is a bipartite graph if:

- V can be partitioned into two disjoint sets X and Y i.e.$V = X \cup Y$ and $X \cap Y = \phi$
- All edges in E go between X and Y, no edge is found between two vertices of only X or Y. In addition, we denote this bipartite graph as $G = (X, Y, E)$.

Note that throughout this section we assume that every vertex has at least one incident edge in the bipartite graph.

A matching M in a bipartite graph is a set of edges with no common vertices in other words a matching is a set of independent edges that is each vertex appears in at most one edge in M.

For example, the edge set $M = \{1a, 2b, 3c, 4e\}$ is matching in the bipartite graph shown on figure 3.2.1.

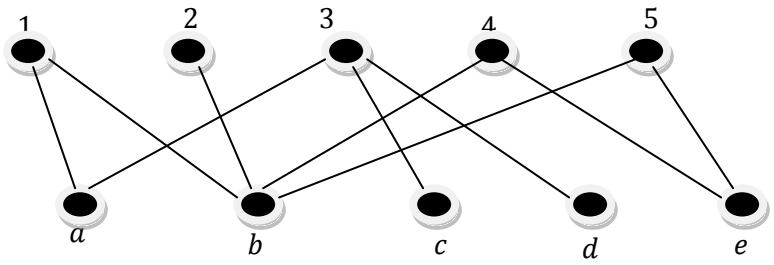


Figure 3.2.1: Bipartite graph

Recall that a matching M is a maximum if the cardinality of M is the highest among all the matching in the graph.

Therefore, the main objective of this section is to determine the cardinality of the maximum matching by using the network flow problem.

Given a bipartite graph $G = (V, E)$ in which $V = X \cup Y$ such that $X = \{x_1, x_2, \dots, x_m\}$ and $Y = \{y_1, y_2, \dots, y_n\}$ whose diagram is given by figure 3.2.2.

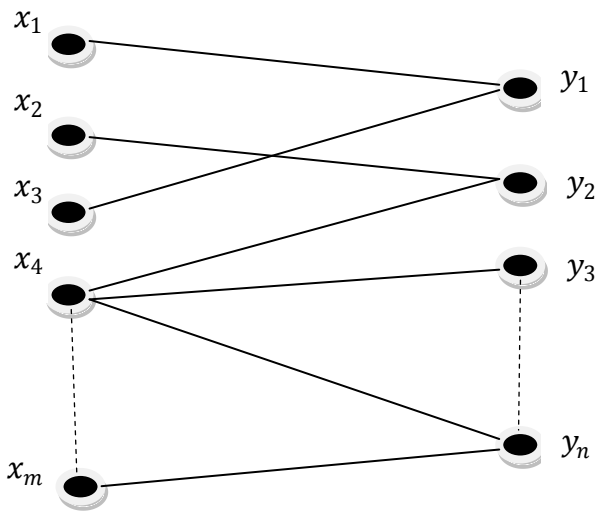


Figure 3.2.2: Bipartite graph with $\boldsymbol{n + m}$ vertices

The maximum bipartite matching can be applied in different areas such as in industrial areas and social life. For instance, we can raise two of them. The first one is matching a set of machines, say X, with a set of tasks, say Y, to be performed simultaneously. In this case, an edge between machine $u \in X$ and task $v \in Y$ indicates that machine u can execute task v. Therefore, we want to maximize the number of tasks executed. The second is Dating Agency; here we could find a matching between a set of girls, say X, with a set of boys, say Y, to establish dating. In this case, an edge indicates potential compatibility. Therefore, we are interested to as many matches as possible. Thus, if all this an important part of this section, how could be done?

The first step is creating a digraph (or, the orientation) of $G = (X, Y, E)$ by specifying for each edge an order on its ends by directing from vertices of X to vertices of Y as shown on figure 3.2.3. Then creating a flow network $N' = (V', A)$ in which flow correspond to matching in original graph $G = (V, E)$, where $V = X \cup Y$.

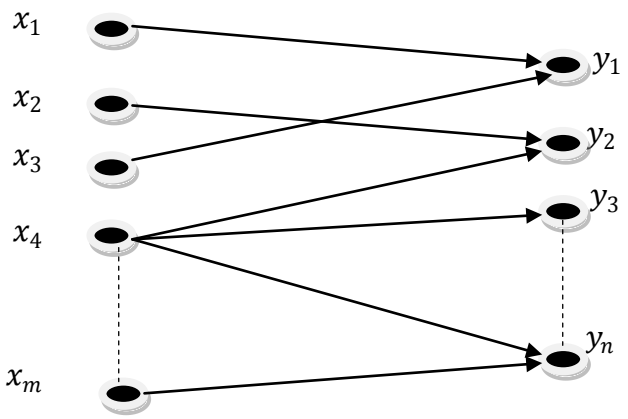


Figure 3.2.3: directed graph of $\boldsymbol{G} = (\boldsymbol{X}, \boldsymbol{Y}, \boldsymbol{E})$

Recall that networks with multiple sources and sinks can be reduced to the case of networks that have just one source and one sink by creating super source s, connected to sources $\{s_1, s_2, \dots, s_m\}$ and super sink t, connected to sinks $\{t_1, t_2, \dots, t_n\}$ and setting $cap(s, s_i) = \infty$ for each $i = 1,2, \dots, m$ and $cap(t_j, t) = \infty$ for each $j = 1,2, \dots, n$). This procedure is illustrated in the following figure 3.2.4 as an example.

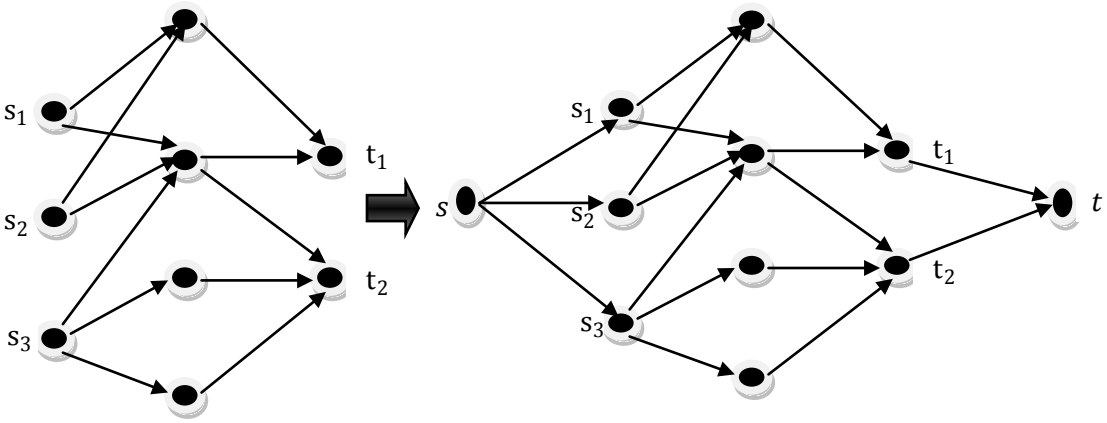


Figure 3.2.4: creating a network of a single source and single sink

Thus, the orientation of the bipartite graph $G = (X, Y, E)$ which shown on figure 3.2.3 can be taken as a network of multiple sources $x_1, x_2, \dots, x_m$ and multiple sinks $y_1, y_2, \dots, y_n$ and so reduced to an ordinary network flow by creating a super source s and super sink t. Let $N = (V', A)$ be the network associated with the bipartite graph $G = (V, E)$ such that

a) $V' = \{s, t\} \cup V$;

b) A is made of directed edges as follows:

- From source s to the vertices in X, $\{(s,u): u \in X\}$;
- From u in X to v in Y, $\{(u,v): u \in X, v \in Y \text{ and } (u,v) \in E\}$; and
- From vertices in Y to sink t, $\{(v,t): v \in Y\}$

That is, $A = \{(s,u): u \in X\} \cup \{(u,v): u \in X, v \in Y \text{ and } (u,v) \in E\} \cup \{(v,t): v \in Y\}$

c) Assign unit capacity to each arc in A, $cap(e) = 1$ for all $e \in A$. Figure 3.2.5 shows what we did so far [3].

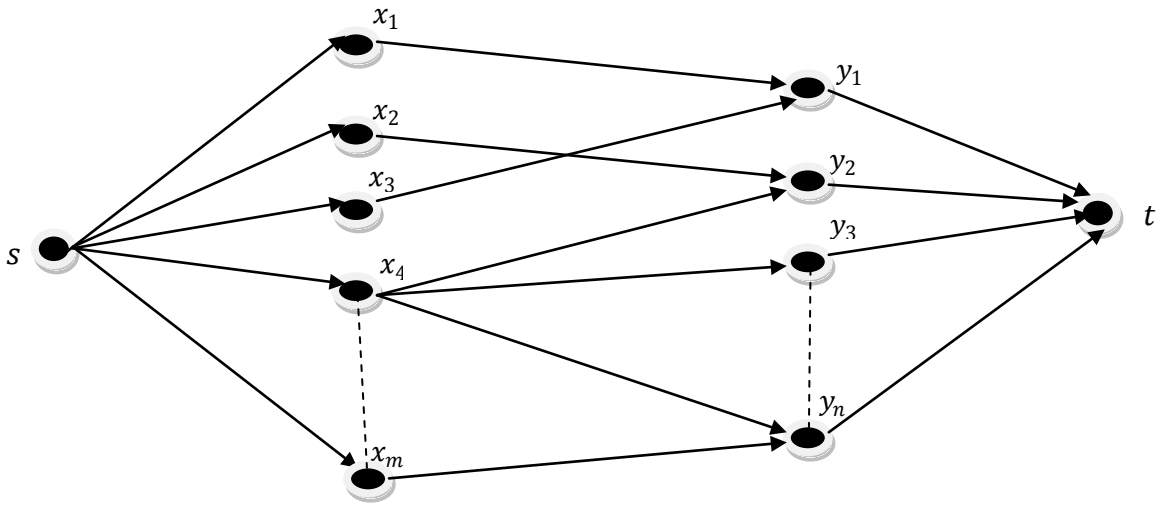


Figure 3.2.5: A network ***N*** associated with ***G***

Theorem 3.2.1[3] let $G = (X, Y, E)$ be a bipartite graph, and let N be the network associated with G. Then,

a) Every flow in N corresponds to a matching M of G, and
b) Every matching M of G corresponds to a flow in N.

This correspondence is such that two vertices are matched in G if and only if there is a unit flow along the corresponding arc in N.

Proof: Recall that $cap(e) = 1$ for all $e \in A(N)$. Let f be a flow in N. By the integral theorem for all $e \in A(N)$ either $f(e) = 0$ or $f(e) = 0$.

a) Let M be the set of edges of the form xy in G for which $x \in X, y \in Y$ and $f_{xy} = 1$. Our claim is, M is a matching of G.

To prove M is a matching of G, we must show that no vertex of G is incident with more than one edge in M.

Let u be any vertex in G. Then u belongs to exactly one of the sets X or Y. Assume without loss of generality that u belongs to X and that u is incident with the edge uv

in M. We will show that u is not incident with any other edge in M. Suppose that uw is another edge in M. Then $f_{uv} = 1$ and $f_{uw} = 1$ by definition of M. Thus in N the total flow out of vertex u is at least two. However, in $N$the only arc, entering u is su, where s is the source vertex and this has capacity 1. So the total flow into vertex u does not equal to the total flow out of vertex u, this contradicts that f is a flow in N. Hence u is incident with at most one edge in M, and so M is a matching of G. This proves that every flow in N corresponds to a matching of G.

b) Now suppose that M is a matching in G. Let N be the network associated with G, and let s and t be the source and sink vertices in N, respectively. For each arc in N, define a function f by

$f_{sx} = 1$ if $x \in X$ and there exist $z \in Y$ such that $xz \in M$

$f_{yt} = 1$ if $y \in Y$ and there exist $w \in X$ such that $wy \in M$

$f_{xy} = 1$ if $x \in X$, $y \in Y$ and $xy \in M$; and

$f_{uv} = 0$ Otherwise

Since each arc e in N has unit capacity,$0 \leq c(e) \leq f(e)$, f satisfies the capacity constraints. Now consider any vertex x of N other than s and t. Such a vertex x is a vertex of G and hence belongs either X or Y. Assume without loss of generality that belongs to X. By definition of f, either $f_{sx} = 0$ or $f_{sx} = 1$.
If $f_{sx} = 0$, then there exists no $z \in Y$ such that $xz \in M$; so the total flow into x and the total flow out of x are both zero. On the other hand, if $f_{sx} = 1$, then there exists $z \in Y$ such that $xz \in M$. Since M is a matching of G, z is unique. Thus, in this case also the total flow into x is equal to the total flow out of x, and so f satisfies the conservation constraints. It follows that f is a flow in N. This proves that every matching M of G corresponds to a flow f in N.■

Theorem 3.2.2 [3]*:* - Let M be a maximum matching in bipartite graph $G = (X, Y, E)$. Let f be a maximum flow in the network G'.Then $|M| = val(f)$.

Example 3.2.1: - Let us take the following bipartite graph G as shown on figure 4.2.6, with partitions $X = \{s_1, s_2, s_3, s_4, s_5\}$ and $Y = \{t_1, t_2, t_3, t_4\}$

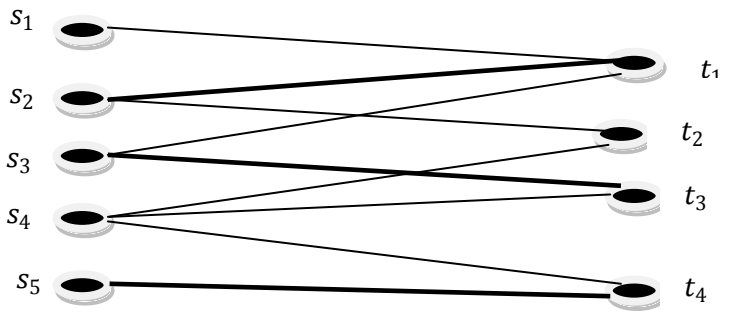


Figure 3.2.6: Bipartite graph ***G***

The matching $M = \{s_2t_1, s_3t_3, s_5t_4\}$ is a maximum matching in G and $|M| = 3$. Now let f is the maximum flow in the network shown on figure 4.2.7.
Now we have to take a flow corresponding to a matching M in the graph G. Let p_1, p_2, p_3 be arc disjoint (s,t)- paths in G', such that $p_1 = ss_2t_1t$, $p_2 = ss_3t_3t$ and $p_3 = ss_5t_4t$. Since the maximum number of arc disjoint (s,t)- paths in G' is three, by lemma 4.1.1 the value of the maximum flow is equal to maximum number of arc disjoint (s,t)- paths in a network.
Therefore, $val(f) = 3$. Hence, $|M| = val(f) = 3$.

Let us come to the proof of theorem 3.2.2
Given a bipartite graph G and a network G', this is given on figure 3.2.3 and figure 3.2.5 respectively. Let M be the maximum matching in G and f be the maximum flow in G'.

Claim: $|M| = val(f)$

(≤) Let M be the maximum matching in G with $|M| = k$. Since our assumption in the formulation of the network $G^{'}$ is $c(e) = 1$ for all arc $e \in E^{'}$. By integral theorem, the maximum flow produces an integer value flow. Thus, $f(e)$ is either 0 or 1. Consider flow f that sends 1 unit along each k paths (paths whose one edge is from M). Let C be the minimum cut in $'$ such that $C = [S.\overline{S}]$ for a set $s \in S \subseteq V'$. Since M may not saturate V, so $|M| \leq |C|$ however, by max flow min cut theorem. We have

$$|M| \leq val(f) \qquad (2)$$

(≥) Let f be the maximum flow in $G^{'}$ with $val(f) = k$. Since $c(e) = 1$ for all $e \in E'$, by integral theorem, we can assume that $f(e)$ is either 0 or 1 for every arc $e \in E'$. Consider M be the set of arcs from X to Y with $f(e) = 1$. This implies that each vertex in X and Y appears in at most one edge in M. By lemma 4.1.1, $val(f) = k =$ the maximum number of arc-disjoint (s,t)-paths in $G^{'}$. let $p_1, p_2, \dots, p_k$ be the k arc-

disjoint (s,t)-paths in G'. Since M is the set of independent edges with possible maximum cardinality, those paths do not share any arc of M.

Therefore, $val(f) \leq |M|$ (3)

Hence, by (2) and (3) we have $|M| \leq val(f)$ ■

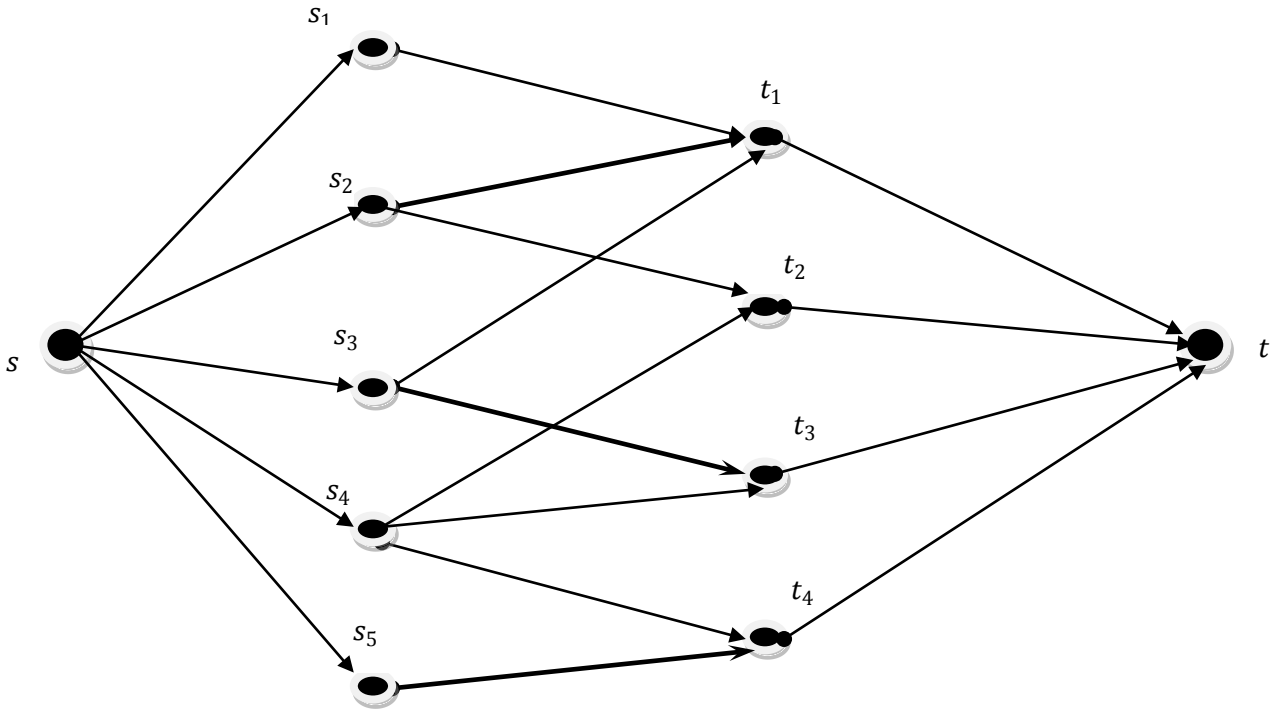


Figure 4.2.7: Network

4. References

[1] Douglas B. West, Introduction to Graph theory, 2nd edition, prentice hall, USA, 2001.

[2] J.A. Bondy and U.S.R Murty, Graph theory with Applications, 5th edition, North-Holland, Canada, 1982.

[3] Jon Kleinberg, Eva Torados, Algorithm Design, 2nd edition, Pearson press, Boston, 2005

5. Bibliography

[1] Douglas B. West, Introduction to Graph theory, 2nd edition, prentice hall, USA, 2001.

[2] J.A. Bondy and U.S.R Murty, Graph theory with Applications, 5th edition, north-Holland, Canada, 1982.

[3] Jon Kleinberg, Eva Torados, Algorithm Design, 2nd edition, Pearson press, Boston, 2005.

[4] John A. Dossey, Albert D.Otto, Lawrence E. Spence, & Charless vanden Enyden, Dicrete Mathematics, 5th edition, Pearson International, Boston, 2006.

[5] Rechard A. Brauldi, Introductory combinatory, 4th Edition, Pearson prentice hall, New Jersey, 2004.

[6] Richard Johnson bough, Discrete Mathematics, 6th edition, prentice hall, USA, 2010.