A GRADUATE SEMINAR REPORT ON

THE TRANSPORTATION PROBLEM

BY HENOK GEBREMEDHIN

In partial fulfilment of the requirements for the degree of Master of Science In Mathematics

Department of Mathematics

Advisor: Yirgalem Tsegaye(PhD)

July 2, 2013

ADDIS ABABA UNIVERSITY

DEPARTMENT OF MATHEMATICS

The undersigned hereby certify that they have read and recommend to the School of Graduate Studies for acceptance a project work entitled "The Transportation Problem" by Henok Gebremedhin in partial fulfilment of the requirements for the degree of Master of Science.

date_____

Advisor Signature

Dr. Yirgalem Tsegaye _____

Contents

Abstract

A variety of applications arise in Mathematics with the growing concepts of Graph Theory. Among the different applications; one of the first Optimization problem studied in operations research is the Transportation problem. The problem can be seen as the transport of products from warehouses to stores, with a minimum possible transportation costs, in such a way that the total supplies from all Warehouses are sufficient to meet the demands at all the Stores.

Acknowledgment

From the bottom of my heart, I would like to thank the almighty God for his blessings. Next, I gratefully acknowledge my advisor Dr. Yirgalem Tsegaye for her support. Last but not least, my gratitude goes to my beloved families especially my dad Gebremedhin and brother Shimelis, thank you very much for your valuable support.

Introduction

The Graph Theory has a definite starting place, a paper published in **1736** by the Swiss Mathematician **Leonhard Euler(1707-1783)**. The main idea behind this work grew out of a problem known as the seven bridges of Konigsberg. This area of Mathematics grew fast since it models many problems in science, specially in Computer Science.

An efficient algorithm developed to solve the maximum flow problem was introduced by **Lester R.Ford, Jr.** and **Delbert Ray Fulkerson**. Basically, it is designate to increase the flow in a transport network N iteratively, until no further increase is possible.

Among the various applications of Graph Theory which is studied in Operations research is the **Transportation Problem**. The goal of the problem is to find a routing of all the items from Warehouses to Stores that minimizes the transportation costs.

Chapter 1

Preliminaries

1.1 Undirected Graphs

Definition 1.1.1. *An undirected graph G is an ordered triple (V(G), E(G), ψ_G) consisting of a nonempty set V(G) of vertices, a set E(G) disjoint from V(G), of edges, and an incidence function ψ_G that associates with each edge of G an unordered pair of(not necessarily distinct vertices of G). If e is an edge and u and v are vertices such that $\psi_G(e) = (u, v)$, then e is said to join u and v; the vertices u and v are called the ends of e.*

Example 1.1.1. *Let* $G = (V, E)$, *where* $V = \{v_1, v_2, v_3, v_4, v_5\}$, $E = \{e_1, e_2, e_3, e_4, e_5, e_6, e_7, e_8, e_9, e_{10}, e_{11}\}$ *and* $\psi_G(e_1) = \{v_1, v_3\}$, $\psi_G(e_2) = \{v_1, v_5\}$, $\psi_G(e_3) = \{v_1, v_2\}$, $\psi_G(e_4) = \{v_2, v_5\}$, $\psi_G(e_5) = \{v_2, v_3\}$, $\psi_G(e_6) = \{v_2, v_4\}$, $\psi_G(e_7) = \{v_5, v_5\}$, $\psi_G(e_8) = \{v_3, v_4\}$, $\psi_G(e_9) = \{v_4, v_5\}$, $\psi_G(e_{10}) = \{v_3, v_5\}$ *and* $\psi_G(e_{11}) = \{v_3, v_3\}$.

A graph can be represented using a picture, where points are used to represent vertices, and line segments to represent edges. Note that neither the points nor the line segments have to do with the geometrical concepts of points and line segments. They are simply there to show discrete objects and their

connections.

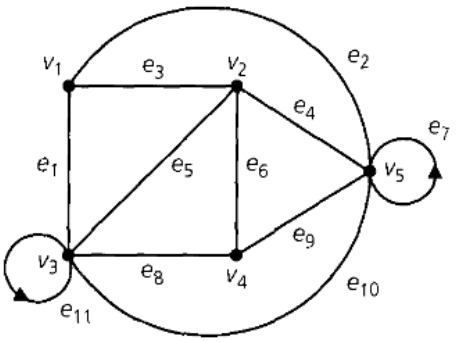


Figure 1.1: A graph G

The definition and concepts in graph theory are suggested by the graphical representation. The ends of an edge are said to be incident with the edges, and viceversa. Two vertices which are incident with a common edge are adjacent, as are two edges which are incident with a common vertex.

Definition 1.1.2. *An edge with identical ends is called a loop, and an edge with distinct ends a link. Two or more links with the same pair of ends are said to be parallel edges. If a vertex of G has no incident with it or a vertex which has no vertices adjacent to it then the vertex is called an isolated vertex.*

Example 1.1.2. *let* $G = (V, E)$, *where* $V = \{u, v, w, x, y\}$,
$E = \{a, b, c, d, e, f, g, h\}$ *and*
$\psi_G(a) = \{v, u\}$, $\psi_G(b) = \{u, u\}$, $\psi_G(c) = \{v, w\}$, $\psi_G(d) = \{w, x\}$,
$\psi_G(e) = \{v, x\}$, $\psi_G(f) = \{w, x\}$, $\psi_G(g) = \{u, x\}$, $\psi_G(h) = \{x, y\}$.
The edge b is a loop and edges d and f are parallel edges. See the graph G on Figure 1.2.

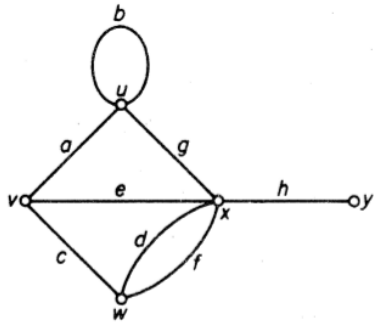


Figure 1.2:

Definition 1.1.3. *A graph is finite if both its vertex set and edge set are finite.*

Note that in this paper we consider only the finite graphs, and so the term 'graph' always means 'finite graph'. Any graph with just one vertex is referred to as trivial and all other graphs are non trivial.

Definition 1.1.4. *let G be a graph then,*
i. G is empty if it has no edge. (a null graph)
ii. A graph G is said to be simple if it has no loop and parallel edges.

Notation 1. *The number of vertices(the order of the graph G) and edges(the size of G) are denoted by $|G|$ and $\|G\|$ respectively.*

Definition 1.1.5. *A graph G is a weighted graph if each edge e of G is associated with a real number, $w(e)$ which is its weight.*

Definition 1.1.6. *A bipartite graph is one whose vertex set can be partitioned into two subsets X and Y, so that each edge has one end in X and one in Y; such a partition (X, Y) is called a bipartition of the graph.*

Definition 1.1.7. *A complete bipartite graph is a simple bipartite graph with bipartition (X, Y) in which each vertex of X is joined to each vertex in Y. If $|X| = m$ and $|Y| = n$, such a graph is denoted by $K_{m,n}$.*

Example 1.1.3. *The figure below is a complete bipartite graph, whose vertex set can be partitioned into two subsets X and Y such that $X = \{x_1, x_2, x_3\}$ and $Y = \{y_1, y_2, y_3\}$.*

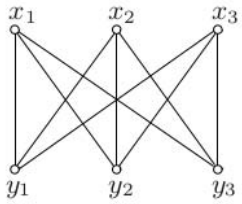


Figure 1.3: $K_{3,3}$

1.1.1 Subgraphs

Definition 1.1.8. *A graph H is a subgraph of G (written $H \subseteq G$) if $V(H) \subseteq V(G)$, $E(H) \subseteq E(G)$ and ψ_H is the restriction of ψ_G to $E(H)$. When $H \subseteq G$ but $H \neq G$, we write $H \subset G$ and called H is a proper subgraph of G.*

Example 1.1.4. *In Figure* 1.4*, we have a graph G and two subgraphs G_1 and G_2 of G such that $V(G_1) \subseteq V(G)$ and $V(G_2) \subseteq V(G)$, $E(G_1) \subseteq E(G)$ and $E(G_2) \subseteq E(G)$, and ψ_{G_1} and ψ_{G_2} are the restriction of ψ_G to $E(G_1)$ and $E(G_2)$ respectively.*

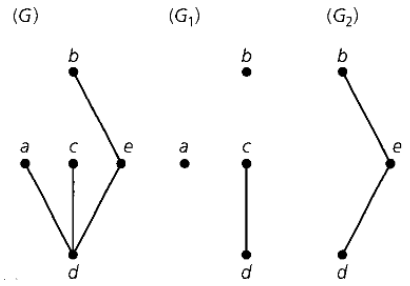


Figure 1.4:

Definition 1.1.9. *A spanning subgraph of G is a subgraph H with $V(H) = V(G)$.*

Definition 1.1.10. *An induced subgraph of G is the subgraph obtained from G by deleting some of the vertices in G together with their incident edges and an edge-induced subgraph of G is the subgraph obtained from G by taking some edge in G with their incident vertices.*

Example 1.1.5. *Consider a graph G with $|G| = 4$ and $\|G\| = 5$.*

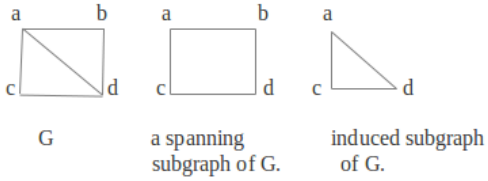


Figure 1.5:

Definition 1.1.11. *The degree of a vertex v in G is the number of edges of G incident with v and denoted by $d_G(v)$, each loop counting as two.*

Notation 2. *We denote the minimum and maximum degrees of vertices of G by $\delta(G)$ and $\triangle(G)$, respectively.*

Example 1.1.6. *See the graph in Figure* 1.6. *$d_G(v) = 4, \ \forall \ v \in V(G)$. Such a graph is known as a K-regular graph, and $K = 4$ in this particular example.*

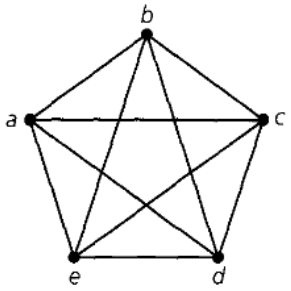


Figure 1.6:

Example 1.1.7. *See the figure below,*

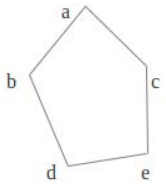


Figure 1.7: 2-regular graph

1.1.2 Paths and Connected graphs

Definition 1.1.12. *A walk in G is a finite nonempty sequence* $W = v_0e_1v_1e_2v_2...e_kv_k$, *of vertices and edges, such that, for* $1 \leq i \leq k$, *the ends of* e_i *are* v_{i-1} *and* v_i.

Definition 1.1.13. *A walk is said to be a path, if the vertices* $v_0,\ v_1,\ v_2, ...,\ v_k$ *of a walk are distinct and a walk is said to be a trail, if the edges* $e_1,\ e_2, ...,\ e_k$ *of a walk are distinct.*

Note 1.1.1. *In a simple graph, a walk* $v_0e_1v_1e_2v_2...e_kv_k$ *is determined by the sequence* $v_0v_1v_2...v_k$ *of its vertices; hence a walk in a simple graph can be specified simply by its vertex sequence.*

Example 1.1.8. *Given a graph* $G = (V, E)$. *See the figure below*

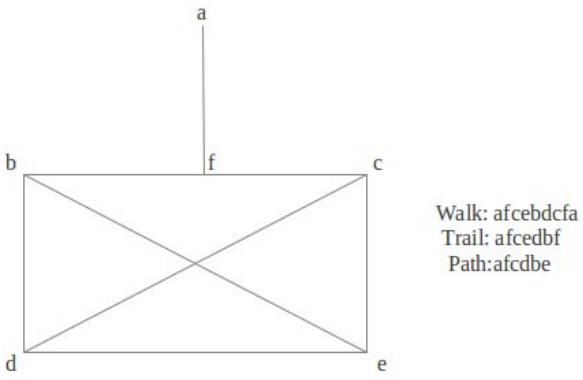


Figure 1.8:

Definition 1.1.14. *Two vertices* u *and* v *of* G *are said to be connected if there is a* $(u,\ v)$ *path in* G.

Theorem 1.1.1. *Connectedness is an equivalent relation on the vertex set* V.

Proof. We have a relation, a is connected to b, for $a,\ b \in V$. Now we need to show that this relation is an equivalent relation by showing the relation is reflexive, symmetric and transitive.
(i). We have a (u, u) path with length zero, from the definition u is connected to u. The relation is reflexive.
(ii). Let consider a (u, v) path; u is connected to v, by reversing this path we have another (v, u) path so by definition v is connected to u, then the relation is symmetric.
(iii). Let we have (u, v) and (v, w) paths; u is connected to v and v is connected to w, by concatenating the two path at v we form another (u, w)

path, then by definition u is connected to w. Then the relation is transitive. Therefore by (i), (ii) *and* (iii) the relation is an equivalent relation. □

Note 1.1.2. *The proof of above Theorem is important for the partitioning of the vertex set V into non empty subsets* $V_1,\ V_2, ...,\ V_\omega$.

If there is a partition of V into non empty subsets $V_1,\ V_2, ...,\ V_\omega$ then two vertices u and v are connected if and only if both u and v belong to the same set V_i. The subgraph G[V_1], G[V_2],..., G[V_ω] are called the components of G. If G has exactly one component then G is connected; otherwise G is disconnected. We denote the number of component of G by $\omega(G)$.

Example 1.1.9.

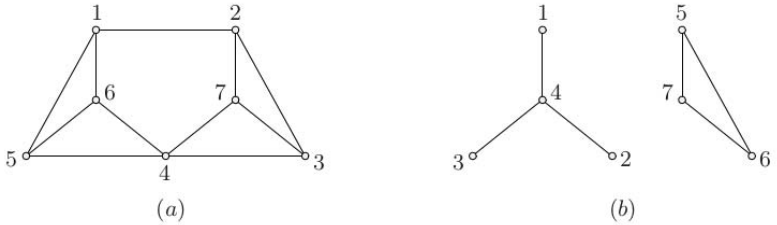


Figure 1.9: (a) A connected graph G, and (b) A disconnected graph G

cycles

Definition 1.1.15. *A walk is closed if it has a positive length and its origin and terminus are the same. A closed trail whose origin and internal vertices are distinct is a cycle. A cycle of length K is called a K-cycle; a K-cycle is odd or even according as K is odd or even.*

Example 1.1.10. *See the graph in Figure* 1.10.

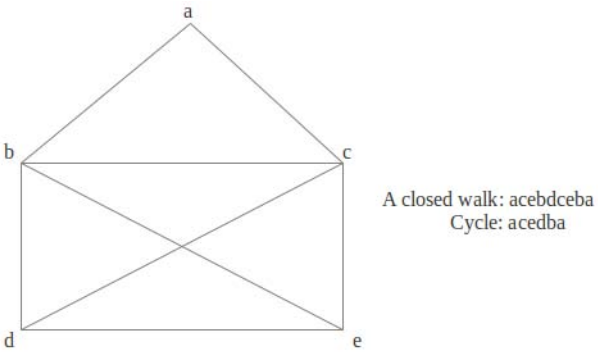


Figure 1.10:

1.1.3 Trees

Definition 1.1.16. *An acyclic graph is a graph G which doesn't contain a cycle. A tree is a connected acyclic graph.*

Example 1.1.11. *The Figure* 1.11 *is the different form of trees in which each tree has six vertices.*

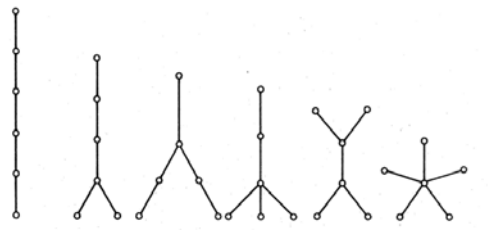


Figure 1.11: The trees on six vertices

proposition 1. *In a tree, any two vertices are connected by a unique path.*

Proof. We shall use proof by contradiction. Let G be a tree, and assume that there are two distinct $(u,\ v)$-paths P_1 *and* P_2 in G. Since $P_1 \neq P_2$, there is an

edge $e = xy$ of P_1 that is not an edge of P_2. Clearly the graph $(P_1 \cup P_2) - e$ is connected. It therefore contain an (x, y)-path P. But then P+e is a cycle in the acyclic graph G, a contradiction that G is a tree. □

Theorem 1.1.2. *If the graph G is a tree, then* $\|G\| = |G| - 1$

Proof. By induction on v

when $|G| = 1$, G has no edge and $\|G\| = 0 = |G| - 1$. Suppose the theorem is true for all trees on fewer than v vertices and let G be a tree on $|G| \geqslant 2$ vertices.

Let $uv \in E$. Then $G - uv$ contains no (u, v)-path, since uv is unique (u, v)-path in G. Thus $G - uv$ is disconnected and so $\omega(G - uv) = 2$, The components G_1 and G_2 of $G - uv$ being acyclic are trees moreover, each has fewer than v vertices. Therefore by the induction hypothesis

$\|G_i\| = |G_i| - 1$ for $i = 1, 2$

Thus $\|G\| = \|G_1\| + \|G_2\| + 1$

$= |G_1| + |G_2| - 1$

$= |G| - 1.$ □

Definition 1.1.17. *A Spanning Tree is a Spanning Subgraph which is a tree.*

Cut edges and Cut vertices

Definition 1.1.18. *A cut edge of G is an edge e such that* $\omega(G-e) > \omega(G)$.

Theorem 1.1.3. *An edge e of G is a cut edge of G if and only if e is not contained in a cycle of G.*

Proof. Let e be a cut edge of G. Since $\omega(G-e) > \omega(G)$ there exist vertices u and v of G that are connected in G but not in G-e. There is therefore some (u, v) path P in G which, necessarily, traverse e. Suppose that x and y are

the ends of e, and that x precedes y on P. In G-e, u is connected to x by a section of P and y is connected to v by a section of P. If e were in a cycle C, x and y would be connected in G-e by the path C-e. Thus u and v would be connected in G-e a contradiction.

Conversely, suppose that e=xy is not a cut edge of G; $\omega(G-e) = \omega(G)$. Since there is an (x, y)-path(namely xy) in G, x and y are in the same component of G. It follows that x and y are in the same component of G-e and hence that there is an (x, y)-path P in G-e. But then e is in the cycle P+e of G. □

1.1.4 Matching

Definition 1.1.19. *A subset M of E is called a matching in G if its elements are links and no two of them are adjacent in G.*

Example 1.1.12. *In the given Peterson graph in Figure* 1.12, $M = \{af, ej, di, hc, gb\}$ *is a matching.*

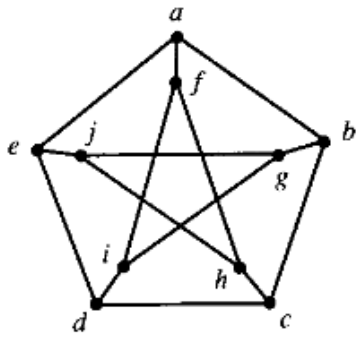


Figure 1.12: Petersen Graph

A matching M saturates a vertex v, and v is said to be M-saturated if some edge of M is incident with v; otherwise, v is M-unsaturated. If every vertex of G is M-saturated then the matching M is perfect.

Definition 1.1.20. *A matching M is a maximum matching if G has no matching $\bar{M}$ with $|\bar{M}| > |M|$.*

Theorem 1.1.4. *Every perfect matching is a Maximum matching.*

Proof. let M be a perfect matching
suppose M is not a maximum matching, from the definition there exist a matching $\bar{M}$ such that $|\bar{M}| > |M|$, so it contradict that M is perfect matching. But the converse is not true, we give a counter example which is a maximum matching but not a perfect matching see the figure below whose dark edges are in the set of maximum matching M but every vertex of the graph is not M-saturated so its not perfect matching. □

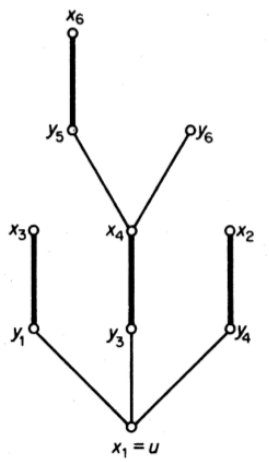


Figure 1.13:

Definition 1.1.21. *Let M be a matching in G,*
(i) An M-alternating path in G is a path whose edges are alternately in $E \setminus M$ and M.
(ii) An M-augmenting path is an M-alternating path whose origin and terminus are M-unsaturated.

1.2 Directed Graphs

Definition 1.2.1. *A directed graph D is an ordered triple (V(D), A(D), ψ_D) consisting of a non empty set V(D) of vertices, a set A(D), disjoint from V(D), of arcs, and an incidence function ψ_D that associates with each arc of D an ordered pair of(not necessarily distinct vertices of D). If a is an arc and u and v are vertices such that $\psi_D = (u, v)$ then a is said to join u to v; u is the tail of a and v is its head. For convenience, we shall abbreviated 'directed graph' to digraph.*

Example 1.2.1. *let $D = (V, A)$ be a digraph such that $V = \{v, w, y, z, x\}$ and $A = \{vw, yw, zy, yv, vz, zx, xv\}$.*

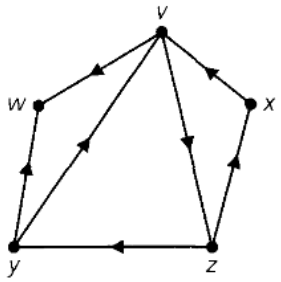


Figure 1.14: A digraph D

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 such a graph is underlying graph G. Conversely, given any graph G, we can obtain a digraph from G by specifying, for each link an order on its ends. Such a digraph is called an orientation of G.

Example 1.2.2.

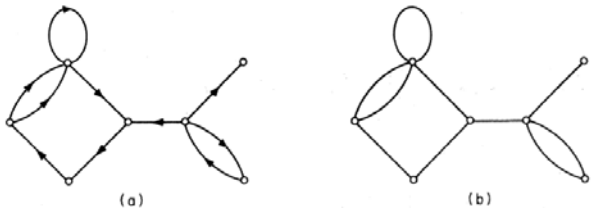


Figure 1.15: (a) digraph D and (b) underlying graph G.

Definition 1.2.2. *A digraph $\bar{D}$ is a subdigraph D if $V(\bar{D}) \subseteq V(D)$, $A(\bar{D}) \subseteq A(D)$ and $\psi_{\bar{D}}$ is the restriction of ψ_D to $A(\bar{D})$.*

Note 1.2.1. *Every concept that is valid for graphs is automatically applied for digraph too.*

Definition 1.2.3. *A directed walk in D is a finite non-null sequence $W = v_0a_1v_1...a_kv_k$ of vertices and arcs, such that for $i = 1, 2, ..., k$. The arc a_i has head v_i and tail v_{i-1}.*

Definition 1.2.4. *The in-degree of a vertex v in D, denoted $d_D^-(v)$ is the number of arcs with head v and the out-degree of v in D is denoted by $d_D^+(v)$; the number of arcs with tail v. We denote the minimum and maximum in-degree and out-degree in D by $\delta^-(D)$, $\Delta^-(D)$ and $\delta^+(D)$, $\Delta^+(D)$, respectively.*

Definition 1.2.5. *A digraph is strict if it has no loops and parallel arcs(no two arcs with the same head and the same tail).*

Chapter 2

Transport Networks

2.1 Introduction

Definition 2.1.1. *A network N is a digraph D(the underlying digraph of N) with two distinguished nonempty and disjoint subsets of vertices, X and Y, and a non negative integer valued function c defined on its arc set A. The vertices in X are the sources of N and those in Y are the sinks of N. In a transport network they correspond to production centres and markets, respectively. Vertices which are neither sources nor sinks are called intermediate vertices; the set of such vertices will be denoted by I. The function c is the capacity function of N and its value on an arc a is the capacity of a. The capacity of an arc can be thought of as representing the maximum rate at which a commodity can be transported along it.*

This chapter provides an application for weighted directed graph D to the flow of commodity from a source to a prescribed destination. Such commodity may be gallons of oil that flow through pipelines or number of telephone calls transmitted in a communication system. In modelling such situations we interpret the weight of an edge in a graph as a capacity that places an

upper limit on, for example, the amount of oil that can flow through a system of pipelines.

Definition 2.1.2. *Let $N = (V, A)$ be a loop-free connected directed graph. Then N is called a* ***Transport network****, if the following conditions are satisfied;*

(a) There exist a unique vertex $x \in V$ with $d_N^-(x)$ equals to 0. This vertex x is called the source.

(b) There is a unique vertex $z \in V$, called the sink, where $d_N^+(z)$ is equal to 0.

(c) There is a nonnegative integer-valued function c from A which assigns to each arc $a = (v, w) \in A$ a capacity, denoted by $c(e) = c(v, w)$.

Example 2.1.1. *See the Figure* 2.1

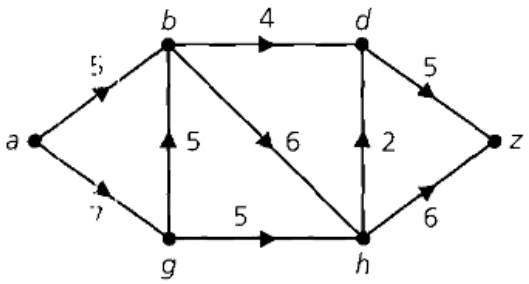


Figure 2.1: A network N

Let f be a real valued function defined on the arc set A of N, and K $\subseteq$ A we denote $\sum_{a \in K} f(a)$ by $f(K)$. Let $\phi \neq S \subset V_n$, $V \setminus S$ is denoted by $\bar{S}$, $\phi \neq \bar{S}$. If k is a set of arcs having tail in S and head in $\bar{S}$, we denote the set by $(S, \bar{S})$ i.e.

$(S, \bar{S}) = \{ a \in A \mid a = (u, v),\ u \in S \text{ and } v \in \bar{S}\}$

$$f(S,\bar{S}) \;=\; f^{+}(S) \;=\; \sum_{a\in(S,\bar{S})} f(a) \tag{2.1}$$

and

$$f(\bar{S},S) \;=\; f^{-}(S) \;=\; \sum_{a\in(\bar{S},S)} f(a) \tag{2.2}$$

$$\textit{In particular, } f^{+}(v) \;=\; \sum_{u\in V_n} f(vu) \textit{ and } f^{-}(v) \;=\; \sum_{u\in V_n} f(uv) \tag{2.3}$$

2.2 Flow

Definition 2.2.1. *A flow in a transport network $N = (V, A)$ is a non negative integer valued function f defined on A satisfying*
(a) $0 \leqslant f(a) \leqslant c(a), \;\forall\; a =(u, v)\in A$ (capacity constraint).
(b) $f^{+}(v) \;=\; f^{-}(v), \quad \forall\; v \;\in I$ (conservation condition).

Note 2.2.1. *The value $f(a)$ can be taken to be material is transported along a under the flow f.*

Notation 3. *$f^{+}(S)$ and $f^{-}(S)$ stand for the total flow out of and into S. In particular $f^{+}(v)$ and $f^{-}(v)$ are the total flows on edges leaving and entering v, respectively.*

Example 2.2.1. *For the network in Figure 2.2, the label (m, n) on each arc a stands for the capacity and the flow in the arc i.e. $m = c(a)$ and $n = f(a)$. The label on each arc a satisfies $f(a) \leqslant c(a)$. In example Figure 2.2(a) the flow into vertex g is 5 but the flow out of g is 4. Hence the function f is not a flow. The function f in (b) satisfies both conditions, so f is a flow.*

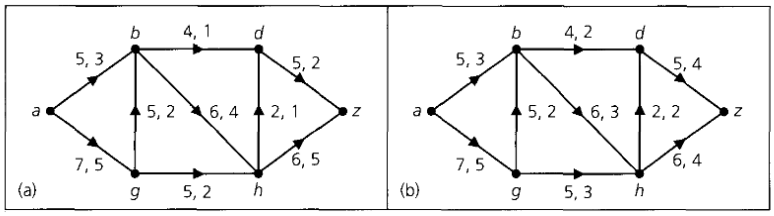


Figure 2.2:

Every network has atleast one flow, because the function by $f(a) \equiv 0$ (the zero flow), clearly satisfies the properties of flow.

Definition 2.2.2. *If S is a set of vertices and f is a flow in a network N, then $f^+(S)$ - $f^-(S)$ is called the Resultant flow out of S, and $f^-(S)$ - $f^+(S)$ is the Resultant flow into S, relative to f.*

Theorem 2.2.1. *Given a flow f in a network N, the net flow out of the source x equal to the net flow into the sink z. i.e $f^+(x) = f^-(z)$*

Proof. let V and A be the set of vertices and arcs of N respectively. We have,

$$\sum_{j\in V}\sum_{i\in V} f(i,j) = \sum_{a\in A} f(a) = \sum_{j\in V}\sum_{i\in V} f(j,i)$$

where $(i,j) \in A$. that is

$$\sum_{j\in V}\sum_{i\in V} f(i,j) = \sum_{j\in V}\sum_{i\in V} f(j,i),$$

this implies that

$$0 = \sum_{j\in V}(\sum_{i\in V} f(i,j) - \sum_{i\in V} f(j,i))$$

separating the terms for j = x and j = z and leaving the rest together , we get

$$0 = \sum_{i\in V} f(i,z) - \sum_{i\in V} f(z,i) + \sum_{i\in V} f(i,x) - \sum_{i\in V} f(x,i) + \sum_{j\in V, j\neq x,z} (\sum_{i\in V} f(i,j) - \sum_{i\in V} f(j,i))$$

$$0 = (\sum_{i \in V} f(i,z)) - (\sum_{i \in V} f(x,i)),$$

since $f(z,i) = f(i,x)$=0 for all $i \in V$ and conservation of flow,

$$(\sum_{i \in V} f(i,j) - \sum_{i \in V} f(j,i)) = 0 \ for \ j \in V \ and \ j \neq x, z.$$

since by definition,

$$f^+(x) = (\sum_{i \in V} f(x,i)) \ and \ f^-(z) = (\sum_{i \in V} f(i,z)). \ from,$$

$$0 = (\sum_{i \in V} f(i,z)) - (\sum_{i \in V} f(x,i)) \ equal \ to,$$

$$(\sum_{i \in V} f(x,i)) = (\sum_{i \in V} f(i,z)).$$

$$\therefore \ f^+(x) = f^-(z).$$

□

Note that also, the value of a flow f $(valf)$ is the net flow $f^-(z)$-$f^+(z)$ into the sink z or the net flow $f^+(x) - f^-(x)$ and we can express it as $val(f) = f^+(x) = f^-(z)$

Definition 2.2.3. *Let f be a flow for a transport network N*
(i) An arc a is said to be $f-$positive if $f(a) > 0$, $f-$unsaturated if $f(a) < c(a)$ and $f-$saturated if $f(a) = c(a)$.
(ii) If x is the source of N, then $val(f) = \sum_{v \epsilon V} f(x,v)$ is called a value of the flow.

Example 2.2.2. *For the Transport network in Figure 2.2(b), the arc (h,d) is $f-$saturated. The arcs (a,b), (a,g), (g,b), (b,d), (b,h), (g,h), (d,z) and (h,z) are $f-$unsaturated.*

Definition 2.2.4. *A flow f in N is a maximum flow if there is no flow $\bar{f}$ in N such that $val\bar{f} > valf$.*

cuts

Definition 2.2.5. *A cut in a network N with single source x and single sink z is a set of arcs of the form* $(S, \bar{S})$*, where* $x \in S$ *and* $z \in \bar{S}$*. Let K=*$(S, \bar{S})$ *be a cut, then the capacity of a cut K is the sum of the capacities of its arcs.*

$$capK = \sum_{a \in K} c(a). \tag{2.4}$$

Lemma 2.2.1. *Given a flow f in a network N and any non empty set* $S \subseteq V$ *then,*

$$\sum_{v \in S} (f^+(v) - f^-(v)) \;=\; f^+(S) - f^-(S).$$

Proof. Now we consider of the flow $f(x, y)$ on the arc (x, y) to each side. Thus, we have the following four cases.

case1: if $x, y \in S$, then $f(x, y)$ is not counted on the right, but it contributes positively(vie $f^+(x)$) and negatively (vie $f^-(x)$) on the left

$$\sum_{x \in S} (f^+(x) - f^-(x)) = 0,$$

since $f^+(x) - f^-(x) = f(x, y) - f(x, y)$=0

thus in this case satisfies the equality

i.e

$$0 = f^+(S) - f^-(S) = \sum_{x \in S} (f^+(x) - f^-(x))$$

$$0 = \sum_{x,y \in S} (f(x, y) - f(y, x)) = \sum_{x,y \in S} f(x, y) - \sum_{x,y \in S} f(y, x) \ldots 1$$

case 2: if $x, y \notin S$ then $f(x, y)$ contributes to neither sum. This one is neglected.

case 3: $(x, y) \in (S, \bar{S})$, then $f(x, y)$ contributes positively to each sum that

is,

$$f^+(S) = \sum_{(x,y)\in(S,\bar{S})} f(x,y),\ f^-(S) \ = \ 0 \ and \ f^-(v) \ = \ 0.$$

$\forall v \in S \ldots 2.$

case 4: if $(x,y) \in (\bar{S},S)$, then $f(x,y)$ contributes negatively to each sum that is,

$$-f^-(S) = -\sum_{y,x\in(\bar{S},S)} f(y,x),\ f^+(S) = 0 \ and \ f^+(v) = 0 \ \ \forall v \in S \ldots 3$$

Therefore, by adding $1, 2$ *and* 3 we get,

$$0 = \sum_{x,y\in S} f(x,y) - \sum_{x,y\in S} f(y,x)$$

$$f^+(S) = \sum_{x,y\in(S,\bar{S})} f(x,y),$$

$$-f^-(S) = -\sum_{y,x\in(\bar{S},S)} f(y,x)$$

$$f^+(S) - f^-(S) = \sum_{x,y\in S} f(x,y) - \sum_{x,y\in S} f(y,x) + \sum_{x,y\in(S,\bar{S})} f(x,y) - \sum_{x,y\in(\bar{S},S)} f(y,x)$$

$$= (\sum_{x,y\in S} f(x,y) + \sum_{x,y\in(S,\bar{S})} f(x,y)) - (\sum_{x,y\in S} f(y,x) + \sum_{x,y\in(\bar{S},S)} f(y,x)) \ldots 4$$

but the right side of (4) can be expressed separately as

$$\sum_{x\in S} f^+(x) = \sum_{x,y\in S} f(x,y) + \sum_{x,y\in(S,\bar{S})} f(x,y)$$

$$\sum_{x\in S} f^-(x) = (\sum_{x,y\in S} f(y,x) + \sum_{x,y\in(\bar{S},S)} f(y,x)),$$

therefore from (4) we get

$$f^+(S) - f^-(S) = \sum_{x\in S} f^+(x) - \sum_{x\in S} f^-(x) = \sum_{x\in S}(f^+(x) - f^-(x)).$$

But note that

$$\sum_{v\in S} f^+(v) \neq f^+(S) \text{ and } \sum_{v\in S} f^-(v) \neq f^-(S).$$

□

Theorem 2.2.2. *For any flow f and cut $(S, \bar{S})$ in N*

$$val(f) \;=\; f^+(S) - f^-(S).$$

Proof. let f be a flow, $(S, \bar{S})$ be a cut in N and x be a source then
From the definition of flow and value of flow, we have

$$f^+(v) - f^-(v) = \begin{cases} val(f) & for\ v = x \\ 0 & for\ v \in I \end{cases}$$

summing these equation over S,

$$\sum_{v\in S}(f^+(v) - f^-(v)) = valf$$

$$by\ above\ lemma,\ valf = \sum_{v\in S}(f^+(v) - f^-(v)) = f^+(S) - f^-(S).$$

□

Theorem 2.2.3. *For any flow f and any cut $(S, \bar{S})$ in N, the $val(f)$ cannot exceed $c(S, \bar{S})$. That is, $valf \leqslant c(S, \bar{S})$.*

Proof. let x and z be the source and sink of a network N, respectively. Since $d_N^-(x) \;=\; 0$, it follow that for any $w \in V \quad f(w, x) \;=\; 0$. Consequently,

$$val(f) = \sum_{v\in V} f(x, v) = \sum_{v\in V} f(x, v) - \sum_{w\in V} f(w, x).$$

By the second condition of the definition of flow, For any $a \in S$, $a \neq x$,

$$\sum_{v \in V} f(a,v) - \sum_{w \in V} f(w,a) = 0.$$

Adding the results in the above equations yields

$$\begin{aligned} val(f) &= [\sum_{v \in V} f(x,v) - \sum_{w \in V} f(w,x)] + \sum_{a \in S, a \neq x} [\sum_{v \in V} f(a,v) - \sum_{w \in V} f(w,a)] \\ &= \sum_{a \in S, v \in V} f(a,v) - \sum_{a \in S, w \in V} f(w,a) \\ &= [\sum_{a \in S, v \in S} f(a,v) + \sum_{a \in S, v \in \bar{S}} f(a,v)] - [\sum_{a \in S, w \in S} f(w,a) + \sum_{a \in S, w \in \bar{S}} f(w,a)] \\ &= \sum_{a \in S, v \in \bar{S}} f(a,v) - \sum_{a \in S, w \in \bar{S}} f(w,a). \end{aligned}$$

Since

$$\sum_{a \in S, v \in S} f(a,v) \text{ and } \sum_{a \in S, w \in S} f(w,a),$$

are summed over the same set of all ordered pair in $S \times S$, these summations are equal. Consequently,

$$val(f) = \sum_{a \in S, v \in \bar{S}} f(a,v) - \sum_{a \in S, w \in \bar{S}} f(w,a).$$

For $a, w \in V$, $f(w,a) \geqslant 0$, so

$$\sum_{a \in S, w \in \bar{S}} f(w,a) \geqslant 0.$$

and,

$$val(f) \leqslant \sum_{a \in S, v \in \bar{S}} f(a,v) \leqslant \sum_{a \in S, v \in \bar{S}} c(a,v) = c(S, \bar{S})$$

□

Note 2.2.2. *We call $(S, \bar{S})$ an $a - z$ cut if $a \in S$ and $z \in \bar{S}$ for a and z be the source and sink of the network N, respectively.*

Corollary 2.2.1. *For any $a-z$ flow f and any $a-z$ cut $c(S,\bar{S})$ in a network N,*

$val f = c(S,\bar{S})$ if and only if ;

(i) For each arc $a \in (\bar{S},S)$, $f(a) = 0$.

(ii) For each arc $a \in (S,\bar{S})$, $f(a) = c(a)$.

Proof. Given $val(f) = c(S,\bar{S})$, from the above theorem,

$$val(f) = f^+(S) - f^-(S) \leq \sum_{a\in(S,\bar{S})} c(a) = c(S,\bar{S})$$

we have, $f^+(S) - f^-(S) = \sum_{a\in(S,\bar{S})} c(a)$

$$\sum_{a\in(S,\bar{S})} f(a) - \sum_{a\in(\bar{S},S)} f(a) = \sum_{a\in(S,\bar{S})} c(a)$$

$$-\sum_{a\in(\bar{S},S)} f(a) = 0, \ for\ a \in (S,\bar{S})$$

$$\sum_{a\in(\bar{S},S)} f(a) = 0 \ implies\ f(a) = 0 \ and\ for\ each\ a \in (S,\bar{S})\ we\ have,$$

$$\sum_{a\in(S,\bar{S})} f(a) = \sum_{a\in(S,\bar{S})} c(a)\ since,\ \sum_{a\in(\bar{S},S)} f(a) = 0$$

$\therefore\ f(a) = c(a).$

Conversely, $val(f) = f^+(S) - f^-(S) \leq \sum_{a\in(S,\bar{S})} c(a) = c(S,\bar{S})$

$$= f^+(S) - f^-(S) \leq \sum_{a\in(S,\bar{S})} c(a) = c(S,\bar{S})$$

$val(f) = c(S,\bar{S}),\ since$

$$\sum_{a\in(\bar{S},S)} f(a) = 0\ by\ (i)\ and\ \sum_{a\in(S,\bar{S})} f(a) = \sum_{a\in(S,\bar{S})} c(a)\ by\ (ii).$$

□

Definition 2.2.6. *A cut K in N is a minimum cut if there is no cut $\grave{K}$ in N such that $cap\grave{K} < capK$. If $\hat{f}$ is a maximum flow and $\check{K}$ is a minimum cut then $val\hat{f} \leq cap\check{K}$.*

Corollary 2.2.2. *Let f be a flow and K be a cut such that $valf = capK$, Then f is a maximum flow and K is a minimum cut.*

Proof. Let $\hat{f}$ be a maximum flow and $\check{K}$ be a minimum cut. By above definition,
$valf \leq val\hat{f} \leq cap\check{K} \leq capK$ since, by hypothesis, $valf = capK$, it follows that $valf = val\hat{f}$ and $capK = cap\check{K}$. Thus f is a maximum flow and K is a minimum cut. □

2.3 Maximum flow problem

In this section we develop an efficient algorithm to solve the maximum flow-minimum cut(max-flow min-cut) problem, and establishing the max-flow min-cut theorem. The algorithm we introduce was initially presented in the work of **Lester R.ford,Jr.** and **Delbert Ray Fulkerson**. Basically, it is designate to increase the flow in a transport network N iteratively, until no further increase is possible[6].

Definition 2.3.1. *Let $N = (V, A)$ be a transport network and let P be a path from the source to the sink, $s = v_0a_1v_1a_2v_2...v_{n-1}a_nv_n = t$, be an alternating sequence of vertices and edges, where the edges taken from the underlying graph associate with D. For $2 \leq i \leq n-1$, if $a_i = (v_{i-1}, v_i)$ that is a_i is the directed arc in N from v_{i-1} to v_i then a_i is called a forward arc and if $a_j = (v_j, v_{j-1})$ that is (v_{j-1}, v_j) is the actual directed edge in N then a_j is called a backward arc.*

Definition 2.3.2. *For a flow f in a transport network N, an f- augmenting path is a source to sink path P in the underlying graph G such that for each $a \in A(P)$,*

(i) $f(a) < c(a)$, *for a forward arc and*
(ii) $f(a) > 0$, *for a backward arc.*

From above definition we see that along an $f-$*augmenting path* P, the flow on a forward arc can be increased for no such forward arc is saturated and for each backward arc the flow is positive, so it can be decreased. The maximum possible increase or decrease is given in terms of $\triangle_a$, the tolerance on an arc a.

Definition 2.3.3. *Let P be an $f-$augmenting path in a transport network N, for each arc a on the path P,*

$$\triangle_a = \begin{cases} c(a) - f(a) \ for \ a \ forward \ arc \\ f(a) \ for \ a \ backward \ arc \end{cases}$$

The quantity $\triangle_a$ is often called the tolerance on arc a.

note that, we have $\triangle_a > 0$ for each a on P. Further, we find that

$$\triangle_p = min_{a \in P}\{\triangle_a\}$$

is the maximum increase (for the forward arc) and maximum decrease (for the backward arc) that we can have and still maintain the conservation condition of flow.

Lemma 2.3.1. *Let f be a flow in a transport network N, and let P be an $f-$augmenting path in N with $\triangle_p = min_{a \in P}\{\triangle_a\}$. Define a flow f_1 by*

$$f_1(a) = \begin{cases} f(a) + \triangle_p; \ a \ is \ a \ forward \ arc \ of \ P \\ f(a) - \triangle_p; \ a \ is \ a \ backward \ arc \ of \ P \\ f(a); \ a \ is \ not \ in \ P. \end{cases}$$

Then f_1 is a flow in N with $val(f_1) = val(f) + \triangle_p$

Proof. Here f_1 is defined as

$$f_1(a) = \begin{cases} f(a) + \triangle_p;\ a\ is\ a\ forward\ arc\ of\ P \\ f(a) - \triangle_p;\ a\ is\ a\ backward\ arc\ of\ P \\ f(a);\ a\ is\ not\ in\ P. \end{cases} \quad \ldots 1$$

Thus, we shall show that:

(i)f_1 is a flow, and

(ii)$val(f_1) = val(f) + \triangle_p$

(i) To show f_1 is a flow; we need to show that the capacity constraint and conservation conditions are satisfied.

From the case of equation (1), we have, $f_1(a) = f(a) + \triangle_p$ for every forward arc a on P. In addition, since always tolerance of a path is positive, i.e $\triangle_p > 0$.

$f_1(a) \geq f(a) \geq 0$ for every forward arc a of P $\ldots 2$

From the definition of tolerance $\triangle_a$.

$$\begin{aligned} f_1(a) &= f(a) + min_{a \in P}\{\triangle_a\} \\ f_1(a) &\leq f(a) + \triangle_a \\ f_1(a) &\leq f(a) + c(a) - f(a) \\ f_1(a) &\leq c(a) \ldots 3 \end{aligned}$$

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

$$0 \leq f_1(a) \leq c(a)\ for\ a\ forward\ arc\ on\ P \ldots 4$$

From the second case of equation(1), we have

$$f_1(a) = f(a) \; - \; min_{a \in P}\{\triangle_a\} \; for \; every \; backward \; arc \; on \; P.$$

$$f_1(a) = f(a) \; - \; f(a) = 0 \ldots 5$$

since $\triangle_a$ *is positive*,

$$f_1(a) \leq \; f(a) \; this \; implies \; that$$

$$0 \leq f_1(a) \leq c(a) \; for \; every \; backward \; arc \; on \; P \ldots 6$$

From the third case of equation (1), we have
$f_1(a) = f(a)$ for those arc a not on P, since f is a flow for all arc on the network, so we have

$$0 \leq f_1(a) \leq c(a) \ldots 7$$

Thus by 4 , 6 *and* 7 *it satisfy the capacity constraint.*

Now we need to show the conservation condition. Since flow on vertices,which are not on P has no changed i.e.
$f_1^+(v)$=$f_1^-(v)$ and $f^+(v)$=$f^-(v)$ for all $v \notin V(P) \ldots 8$.
We need to check only vertices of P for the conservation condition. The arcs of P incident to an internal vertex, v of P occurs in one of the following four cases shown below.
case (1): in this case,

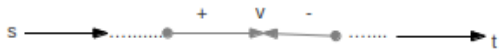
$f_1^-(v)$=$f^-(v)$+ $\triangle_p \ldots 9$ and
$f_1^+(v)$=$f^+(v)$+ $\triangle_p \ldots 10$, then subtracting 9 from 10 we get,
$f_1^+(v) - f_1^-(v)$=$f^+(v) - f^-(v)$=0. Since f is a flow, it follow that $f_1^+(v)$ –

$f_1^-(v)$=0.

Therefore, $f_1^+(v) = f_1^-(v)$, for each internal vertex v on P.

case (2):

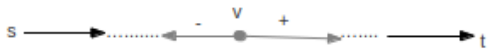
$f_1^-(v)$=$f^-(v)$+ $\triangle_p$ ... 11 and



$f_1^-(v)$=$f^-(v)$− $\triangle_p$ ... 12, then summing 11 and 12 and divide both side by 2 will give us, $f_1^-(v) = f^-(v)$ ... 13. Therefore by 8 and 13, $f_1^-(v) = f_1^+(v)$ for each internal vertex v of P.

case(3):

$f_1^+(v) = f^+(v)$− $\triangle_p$ ... 14



$f_1^+(v) = f^+(v)$+ $\triangle_p$ ... 15. then summing 14 and 15 and divide both side by 2 will give us,$f_1^+(v) = f^+(v)$ ... 16.

Therefore, by 8 and 16 $f_1^+(v)$=$f_1^-(v)$, for each internal vertex v of P.

case (4):

$f_1^+(v) = f^+(v)$− $\triangle_p$ ... 17 and

$f_1^-(v) = f^-(v)$− $\triangle_p$ ... 18. Subtracting (17) and (18), and since f is a flow then we will have,

$f_1^+(v) - f_1^-(v) = f^+(v) - f^-(v)$=0. This implies $f_1^+(v) - f_1^-(v)$=0.

Therefore, $f_1^+(v) = f_1^-(v)$ for each internal vertex v of P.

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

f_1 is a flow.

(ii) For the second part,

Let $T = \{v_1, v_2, ..., v_n\}$ be the set of vertices, which are joined to the source a that is T is the head set of all arcs joined to a.

Clearly, no edge enter to a in the network; $val(f_1) = f_1^+(a)$

$$f_1^+(a) = \sum_{i=1}^{n} f(a, v_i) \dots \ 19$$

and

$$f^+(a) = \sum_{i=1}^{n} f(a, v_i) \dots \ 20.$$

f augmenting path P uses exactly one arc among all arcs,$\{(a, v_1), (a, v_2), ..., (a, v_n)\}$. WLOG, let it be (a, v_3) so,

$$f_1(a, v_i) = f(a, v_3) \ for \ i \neq 3 \dots 21.$$

$$f_1^+(a) = \sum_{i=1, \neq 3}^{n} f(a, v_i) + f_1(a, v_3) \dots 22.$$

In addition, since (a, v_3) is a forward arc of P.

$$f_1(a, v_3) = f(a, v_3) + \triangle_p \dots 23.$$

Therefore from 21,22 and 23 we have,

$$f_1^+(a) = \sum_{i=1, i\neq 3}^{n} f(a, v_i) + f(a, v_3) + \triangle_p$$

$$= \sum_{i=1}^{n} f(a, v_i) + \triangle_p \dots 24.$$

now, we have

$$f_1^+(a) = f^+(a) + \triangle_p$$

Therefore,

$$val(f_1) = val(f) + \triangle_p$$

Thus, the existence of an f *augmenting path* P in a network is significant since it implies that f is not a maximum flow. □

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 contain f *augmenting path* P. This implies that f cannot be a maximum flow since f_1, the revised flow on P has a large value. Thus it contradict the fact that f is a maximum flow.

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

Claim: 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* $t \in \bar{S}$. Thus K=$(S, \bar{S})$ is a cut in N.

We shall show that;

(i). Each arc in $(S, \bar{S})$ is $f -$ *unsaturated*, and

(ii) Each arc in $(\bar{S}, S)$ is $f -$ *zero.*

Consider an arc a with tail $u \in S$ and head $v \in \bar{S}$. Since $u \in S$, there exists an $f-$*unsaturated* $(s,u)-$*path* Q. If a=(u,v) $f-$*unsaturated*, then Q could be extended by the arc a to yield an $f -$ *unsaturated* $(s,v) -$ *path* $Q + a$. However $v \in \bar{S}$, and so there is no such path. Therefore, a must be $f -$ *saturated.* Similarly, consider an arc $\grave{a}$ with tail $u \in \bar{S}$ and head $v \in S$ i.e. $\grave{a} \in (S, \bar{S})$,since N does not contain an $f -$ *augmenting path,* $f(\grave{a}) = 0$ *for all* $\grave{a} \in (S, \bar{S})$. Therefore, $\grave{a}$ must be $f -$ *zero.*

Since by **corollary 2.1.1** the equality hold, *i.e.* $val(f) = cap(K)$ if and only if each arc in $(S, \bar{S})$ is $f-$*saturated* and each arc in $(\bar{S}, S)$ is $f-$*zero.* Hence f is a maximum flow. □

Theorem 2.3.2. ***max-flow min-cut theorem***

For a transport network N, the maximum flow value that can be attained in N is equal to the minimum capacity over all cuts in the network.

Proof. Let f be a flow for which $val(f)$ is a maximum. Then let $(S, \bar{S})$ be the cut constructed as in *Theorem* 2.1.4. We know from *Corollary* 2.1.1 that $val(f) = C(S, \bar{S})$ and then *Corollary* 2.1.2 shows us that $(S, \bar{S})$ is a minimum cut. □

Ford-Fulkerson labelling algorithm

This algorithm helps us to find a maximum flow in a given network N. It is also called as the labelling method. Starting with known flow, for instance the zero flow, it recursively construct 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 labelling procedure is used to find an $f - augmenting\ path$, if one exists. If such a path P is found, then $\hat{f}$, 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 terminate and the original flow is still a maximum flow. See the algorithm below,

Step 1: Given a network N, define an initial flow f in N by $f(a) = 0$ for every $a \in A$(the function f satisfies the condition of the flow)

Step 2: Label the source a with $(-, \infty)$. This label indicates that from the source a as much material is available as is needed to achieve a maximum flow.

Step 3: For any vertex v that is adjacent with a, label v as follows;

(a). If $c(a, v) - f(a, v) > 0$, define $\triangle\ (v) = c(a, v) - f(a, v)$ and label vertex v with $(a^+, \triangle\ (v))$.

(b). If $C(a,v) - f(a,v)$=0, leave vertex v unlabelled.

[*the label* $(a^+, \triangle\,(v))$ *indicates that the present flow from a to v can be increased by the amount* (v) *with the* $\triangle\,(v)$ *additional units supplied from the source a.*

Step 4: As long as there exists $v(\neq a) \in V$ such that v is labelled, and there is an edge (v,w) where w is not labelled, label vertex w as follows;

(a). If $c(v,w) - f(v,w) > 0$, define $\triangle\,(w) = min\{\triangle\,(v), c(v,w) - f(v,w)\}$ and label vertex w as $(v^+, \triangle\,(w))$

(b). $C(v,w) - f(v,w) = 0$, leave vertex w unlabelled. Likewise, as long as there is a vertex $v \neq a$ where v is labelled, and there is an edge (w,v) where w is not labelled, label vertex w as follows;

(c). If $c(w,v) - f(w,v) > 0$, label vertex w as follows $(v^-, \triangle\,(w))$, where $\triangle\,(w) = min\{\triangle\,(v), f(w,v)\}$

(d). If $f(w,v) = 0$, leave vertex y unlabelled.

[*The label* $(v^-, \triangle\,(w))$] *tells us that by decreasing the flow from w to v, the total flow out of w to the labelled vertices can be decreased by* $\triangle\,(w)$.

Step 5. If the sink z has been labelled, go to **Step 6**. Otherwise choose another labelled vertex to be scanned (*which was not previously scanned*) and go to **Step 4**. If there are no more labelled vertices to scan let S be the set of labelled vertices,and now $(S, \bar{S})$ is a saturated $a - z$ cut. Moreover, $valf = C(S, \bar{S})$ and thus f is maximum.

Step 6. Find an $a - z$ flow by backtracking with the labels vertices, the desired augmenting path increase a flow in the forward arc on P by $\triangle\,(z)$ units and decrease flow in the backward arc on P.

Example 2.3.1. *Let* $N = (V, A)$ *be a transport network shown in Figure* 2.3 *part*(i). *Find the maximum flow of the given network using Ford-Fulkerson labelling algorithm.*

The given flow f *is not a maximum flow, since we have an* $f-$*augmenting path*

of $a-z$ flow.

Step 1: we label the source a with $(-,\infty)$.

Step 2: let $b \in V$, b is adjacent with a, then we label b as follows; $(a^+,4)$, where $\triangle\ (b)$=$c(a,b)-f(a,b)=8-4$.

Step 3: b is labelled and there exist a sink z which is not labelled and adjacent with b so we label z by $(b^+,2)$, since $\triangle\ (z)=min\{\triangle\ (b)$,$c(b,z)-f(b,z)\}=min\{4,8-6\}=2$.

Here we get an $f-$augmenting path of an $a-z$ flow that is the path $(a,b),(b,z)$ then we add $\triangle_p$= $min\{4,2\}=2$ to the present flow in the path $(a,b),(b,z)$. See Figure 2.3(iii).

Since the present flow is not maximum we still find another $f-$augmenting path in the $a-z$ flow. Consider the path (a,d),$(d,g),(g,z)$ it is an $f-$augmenting path in the $a-z$ flow. See the Figure 2.3(iv), now we label the vertices in the path, $(a^+,2),(d^+,2),(g^+,2)$ be labels of d,g and z respectively. We add 2 in the present flow along the path (a,d),(d,g) and (g,z). See the Figure 2.3(v).

If we consider arcs (a,d) and (b,z) both arcs are $f-$saturated so we cannot have an $f-$augmenting path containing the two arcs.

Still we can construct an $f-$augmenting path since the flow in the present solution is not a maximum flow. Consider the path $(a,b),(b,d),(d,g),(g,z)$ See Figure 2.3(vi), the arcs $(a,b),(d,g)$ and (g,z) are forward arcs and (b,d) is backward arc.

Now we labels the vertices as follow $(a^+,2),(b^-,2),(d^+,1)$ and $(g^+,1)$ then we add $\triangle_p$= $min\{2,2,1,1\}=1$ to the present flow along the path $(a,b),(b,d),(d,g)$ and (g,z). See Figure 2.3(vii), now the arcs (b,z) and (g,z) are $f-$saturated so we cannot have an $f-$augmenting path of $a-z$ flow.

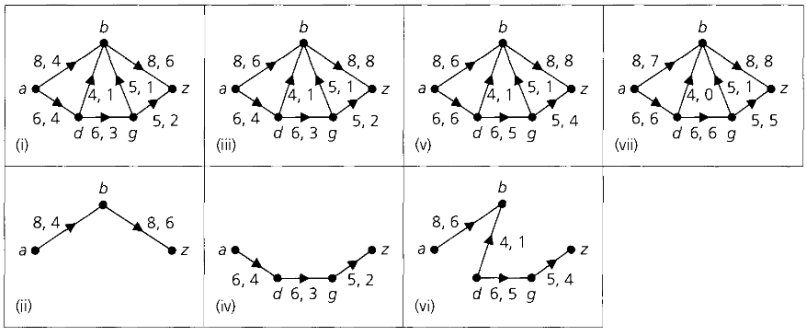


Figure 2.3:

Therefore the flow given in *Figure* 2.3(*vii*) is maximum.

Chapter 3

The Transportation Problem

3.1 Introduction

In this chapter we apply our knowledge of spanning trees, network flows and matchings to **study minimum-cost network flow**. To simplify the problem, consider networks whose underlying graphs are complete bipartite. The edges here have unlimited capacity and the vertices have supplies and demands. We refer to the supply vertices as **Warehouses** and the demand vertices as **Stores** and edge (i, j) from each warehouse w_i to each store s_j. What is new is that there is a cost **c(i,j)** charged for shipping an item on edge (i, j). **The goal is to find a routing of all the items from warehouses to stores that minimizes the transportation costs**. This optimization problem is appropriately called the **Transportation Problem**. It is one of the first optimization problems studied in operations research.

In general, warehouse i has a supply of size $s(i)$ and store j has a demand of size $d(j)$. We assume that the total supplied summed over all warehouses, are adequate to meet the demand at all the stores.

A solution to the transportation problem specifies the amount $x(i, j)$ to ship

on each edge (i, j) so that the total shipments leaving each warehouse do not exceed the warehouse's supply and the total shipments arriving at each store equal the demand of that store. **The goal is to find, among all solution, a minimum-cost solution**. A mathematical statement of the transportation problem is:

$$\textbf{minimize} \quad \sum_{i,j} c(i,j)x(i,j) \ \textit{such that}$$

$$\sum_{j} x(i,j) \leq s_i \ \ \textit{for each } i, \quad \sum_{i} x(i,j) = d_j \ \ \textit{for each } j, \ \textit{and} \ \ x(i,j) \geq 0.[1]$$

Definition 3.1.1. *The data describing a Transportation problem are usually presented in a table called a* ***Transportation Tableau****.*

Example 3.1.1. *See the tableau and associated bipartite graph for a sample transportation problem in Figure* 3.1*. The supplies $s(i)$ of the warehouses appear on the right side of the tableau, and the demands $d(j)$ of the stores appear at the bottom. The shipping cost $c(i, j)$ for (i, j) appears in the upper right half of entry (i, j) in the tableau. The number $x(i, j)$ in the lower left half of the entry (i, j) tells how much is shipped from warehouse i to store j. Note that we refer interchangeably to (i, j) as an edge or an entry in the tableau.*

In the above example, the $x(i, j)'s$ (non optimal) solutions for this transportation problem. In the graph in *Figure* 3.1, we have thickened the edges used in the solution. We observe that

$$\sum_{j} x(i,j) \leq s_i \ \ \textit{in each row } i, \ \textit{and} \ \sum_{i} x(i,j) = d_j \ \textit{in each column } j.$$

The cost of the sample solution in **Figure 3.1** tableau, which use the edges $(1, 1), (2, 1), (2, 2)$ *and* $(3, 2)$ is calculated by using

$$\sum_{i,j} c(i,j)x(i,j) \ \textit{is equal to } 30 \times 4 + 30 \times 6 + 30 \times 7 + 20 \times 6 = 630.$$

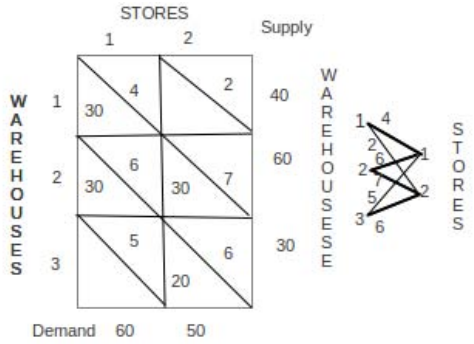


Figure 3.1:

Consider the above example the total supplies exceed the total demands. In this type of problem we add a **Dummy** store to balance the problem but the addition of a new store will not have any effect in overall transportation cost since we assign a cost zero for shipping. Such a transportation problem is called **Balanced Transportation Problem**. In Figure 3.1, the total of the supplies is 130 and the total of demands is 110. Here we have excess supplies of 20 unit so this goes to a demand of the dummy store. See the Figure 3.2.

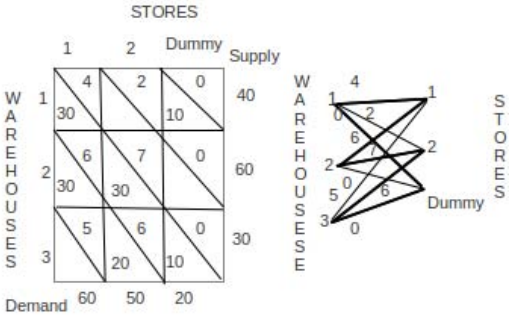


Figure 3.2:

Lemma 3.1.1. *Let S be a solution to a transportation problem involving a set of edges $E(S)$ that contains a cycle. Then there is another solution $\hat{S}$ that costs the same or less than S, where $E(\hat{S})$ is a subset of $E(S)$ containing no cycles.*

Proof. We will not give a formal proof of this lemma, but rather illustrate it with an example and explain how this example generalized to all solution for all transportation problems. □

Consider the solution displayed in Figure 3.2. Observe that the edges used for shipments in S, $(1,1),(2,1),(2,2),(3,2),(3,3),(1,3)$, form a cycle. We can modify S by increasing the shipments 1 on odd edges and reducing the shipment 1 on even edges. In the given above example the odd edges in S are $(1,1),(2,2),(3,3)$ and even edges in S are $(2,1),(3,2),(1,3)$. This balancing changes keep the row and column sums of the modified $x(i,j)'s$ the same, and the modified shipments are again a valid solution. Let us check how the original cost of the solution changes with this modification of S.
One more unit along odd edges increases the cost by, $4+7+0=11birr$. One less unit along even edges reduces the cost by, $6+6+0=12birr$. The net change is $11-12=-1$. Thus, this modification reduces the cost of the solution by $1birr$. To get the largest reduction in cost possible by this modification, let us increase the shipments in the odd edges of this cycle, with corresponding decreases on even edges of the cycle. The critical constraint in our modification is that we cannot decrease the shipment in any even edge of the cycle below 0. The smallest number is 10 on edge $(1,3)$. So we can increase the shipment level by 10 on the odd edges and reduce 10 on even edges.
Note that by driving the level of shipment down to 0 on edge $(1,3)$, we have dropped this edge from the subset of edges used in a new solution. Thus, the

new solution is cycle-free. see the Figure 3.3

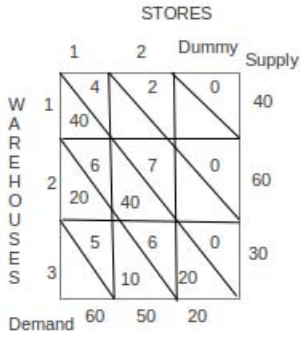


Figure 3.3:

The transportation cost of the new solution is $40 \times 4 + 20 \times 6 + 40 \times 7 + 10 \times 6 + 0 = 620 birr$, which is $10 birr$ less than the cost of the solution of Figure 3.2.

If it had turned out that our modification of increasing shipments by 1 on odd edges and decreasing shipments by 1 on even edges had increased the cost, then we would reverse the strategy and decrease shipments on odd edges as much as possible until one of the odd edges has a shipment of zero and increase shipments on even edges correspondingly. Also, if our modification resulted in no change in the cost, we would still follow the original strategy of increasing odd edge shipments and decreasing even edge shipments as much as possible in order to get a new solution of the same cost that was cycle-free. From lemma,it follows that we only need to consider solutions which contain no cycle, that is solutions which are spanning trees or spanning forest.

For n warehouses and m stores, If the number of entries in the solution are $n+m-1$ then the solution is called **basic feasible solution** and if its allocations in a feasible solution are less than $m + n - 1$, then the solution is called **Degenerate basic feasible solution**. In such solution further improvement in the solution will face problem since we consider cycles for improvement

and we add unused entries with zero shipping unit in the solution to get the better solution so that we consider the solution of the transportation problem is a non-degenerate basic feasible solution or entries in the optimal solution form a spanning tree.

3.2 Methods For Solving The Transportation Problem

The method to solve the transportation problem is known as **Transportation Method**(the simplified version of simplex method).
In this method we have two general steps in solving the problem.
(i). **Finding some initial solution**
First, we find some initial spanning tree solution by using one of the following methods
1.**Northwest Corner Rule**
Steps:
(i) Assign largest possible allocation to the edge in the upper left-hand corner of the tableau.
(ii) Repeat step (i) until all allocations have been assigned.
(iii) Stop. Initial solution is obtained.
2.**Minimum Cost Rule**
Steps:
(i) Find the edge that has the least cost.
(ii) Assign as much as allocation to this edge.
(iii) Block the row\column that cannot further allocated.
(iv) Repeat above steps until all allocations have been assigned.

3.**VAM(Vogel's Approximation Method)**

Steps:

(i) For each column and row, determine its penalty cost by subtracting their two of their least cost.

(ii) select row\column that has the highest penalty cost in step (i).

(iii) Assign as much as allocation to the selected row\column that has the least cost.

(iv) Block the row\column that cannot further allocated.

(v) Repeat above steps until all allocations have been assigned.

See the above transportation problem. Find the initial solution using Northwest Corner rule.

Start at the Northwest corner of the tableau, that is entry $(1, 1)$. Make $x(1, 1)$ as large as possible. The value will be the minimum of $s(1)$ and $d(1)$. For our problem $x(1, 1) = min(s(1), d(1)) = min(40, 60) = 40$. We have total used up the supply at warehouse 1, and so we delete row one from the tableau and reduce the demand at store 1 to $60 - 40 = 20$.

The resulting modified table is shown in Figure 3.4.

Figure 3.4:

To meet the rest of the demand at store 1, we turn to the Northwest corner in the remaining tableau shown in Figure 3.4, that is edge $(2, 1)$. Make this entry as large as possible. So $x(2, 1) = min(40, 20) = 20$. Now we satisfied

the demand at store 1 and delete column one from the tableau and reduce the supply at warehouse 2 to 40. See Figure 3.5.

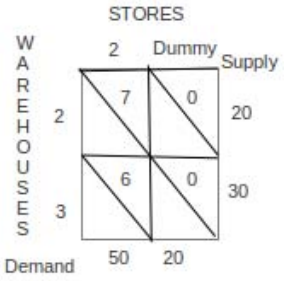


Figure 3.5:

We continue the procedure of repeatedly making the entry in the Northwest of the remaining tableau as large as possible. The complete Northwest Corner Rule solution is shown in Figure 3.6, which we have seen in Figure 3.3.

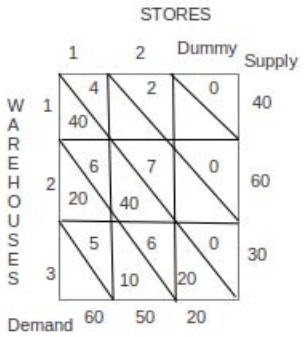


Figure 3.6:

Note 3.2.1. *If the value we assign the current Northwest entry in the current tableau uses up the supplies at the first(remaining) warehouses and also satisfies the demand of the first(remaining) stores, then we would have to delete both the first row and the first column of the current tableau. This will*

lead to a disconnected set of edges in the solution, that is a spanning forest. To avoid this outcome, we arbitrary keep either the first row of first column, although its supplies or demand is 0.

2. **Finding a Better Solution**

The second general step is needed to check whether the given initial solution is optimal or not. This method known as **Modified distributed method(MODI)**. We iteratively follows the steps in MODI until no further decrease the transportation cost

Steps;

$II.A$. Determine selling prices at each warehouses and stores.

$II.B$. Determining which edge to add to the current solution.

$II.C$. Determining the cheaper spanning tree solution.

Step II.A. Determining selling price at warehouses and stores. We now introduce selling prices for the commodity at the warehouses and stores based on the transportation cost in the initial solution. We need to pick an arbitrary price u_1 for the commodity at warehouse 1, let say $u_1 = 10$ *birr*. We use an edge $(1, 1)$ in our solution to transport the commodity from warehouse 1 to store 1 at cost of $c(1, 1) = 4$ *birr* per unit then we should sell the commodity for, 10 *birr* + 4 *birr* = 14 *birr* at store 1. Store 1 also receive shipments from warehouse 2 with a transportation cost of $c(2, 1) = 6$ *birr* per unit. Now we reverse the reasoning used to determine the price at store 1 from the price at warehouse 1. Given that the commodity's price at store 1 is 14 *birr* and it costs 6 *birr* to ship from warehouse 2 to store 1. The price at warehouse 2 should be $14 - 6 = 8$ *birr*. We can continue this process to calculate the selling price for the rest of warehouses and stores.

The complete set of selling price is:

$$Warehouse\ \ 1: \quad u_1 = 10\ birr \qquad Store1: \quad v_1 = 14\ birr$$

$$Warehouse\ \ 2: \quad u_2 = 8\ birr \qquad Store2: \quad v_2 = 15\ birr$$

$$Warehouse\ \ 3: \quad u_3 = 9\ birr \qquad Store3: \quad v_3 = 9\ birr \ldots (1)$$

Summarizing, we set u_1 equal to any arbitrary value. Then we determine successive u_i's and v_j's as in the previous example, using the condition that $v_j - u_i = c(i, j)$, for each edge (i, j) in the current solution. Note that the calculation of price in this fashion is only possible if the edges used in the solution do not form a cycle.

Lemma 3.2.1. *Let S be a spanning tree solution of a transportation problem using edge set $E(S)$, and let u_i and v_j be prices at warehouse i and store j, respectively. Based on solution S according to Step II.A. The transportation cost of this solution,*

$$\sum_{i,j} c(i, j)x(i, j),$$

summed over edges (i, j) in $E(S)$, equals the profit from the sale of the stores' demands at the stores' price minus the cost of the warehouses' supplies at the warehouses' price. That is

$$\sum_{i,j} c(i, j)x(i, j) = \sum_{j} v_j d_j - \sum_{i} u_i s_i \ldots (2)$$

Proof. For each edge (i, j) in $E(S)$, the prices are determined by the condition that $c(i, j) = v_j - u_i$. Then

$c(i, j)x(i, j) = (v_j - u_i)x(i, j) \ldots (3)$

When one sums (3) over all edges in $E(S)$, the total of the shipments $x(i, j)$ out of warehouse i is $s(i)$ and the total of the shipment into store j is $d(j)$. Thus, the right-hand side of the sum of the (3)s over all edges is equal to the

right-hand side of (2). The left-hand side of this sum of (3)s is clearly the left-hand side of (2). □

To illustrate the lemma for the solution in *Figure* 3.6 with the associated prices in (1), the right side of (2) is $(birr\ 14 \times 60 + birr\ 15 \times 50 + birr\ 9 \times 20) - (birr\ 10 \times 40 + birr\ 8 \times 60 + birr\ 9 \times 30) = birr\ 620$. As noted earlier, *birr* 620 is the sum of the transportation costs for this solution.

Step II.B. Determining which edge to add to the current solution

We now look at the edges that are not used in the current solution. Consider edge $(1, 2)$ with cost $c(1, 2) = birr\ 2$. Warehouse 1's selling price is *birr* 10 and Store 2's selling price is *birr* 15. However,if we buy the commodity at warehouse 1 for (*birr* 10) and ship it on edge $(1, 2)$ for *birr* 2, we can sell it at store 2 for $birr\ 10 + birr\ 2 = birr\ 12$, reducing the selling price at Store 2 by *birr* 3 per item. A reduced price at store 2 will cause a reduced profit-store sales minus warehouse purchase, which by above lemma equals a reduced transportation cost. This means that a cheaper solution can be obtained by incorporating edge $(1, 2)$ into the solution. To maintain a spanning trees solution when edge $(1, 2)$ is added, some edge in the current solution would have to be dropped. A choice made in the next Step II.C. Before using edge $(1, 2)$, we check the other edges not in the current solution to see how much each of them could reduce the profit. This result are summarizer below.

$$edge(1,2):\ c(1,2) = birr\ 2 < v_2 - u_1 = birr\ 15 - birr\ 10 = birr\ 5\ decrease\ of\ birr\ 3.$$

$$edge(1,3):\ c(1,3) = birr\ 0 > v_3 - u_1 = birr\ 9 - birr\ 10 = birr\ -1\ increase\ of\ birr\ 1.$$

$$edge(2,3):\ c(2,3) = birr\ 0 < v_3 - u_2 = birr\ 9 - birr\ 8 = birr\ 1\ decrease\ of\ birr\ 1.$$

$$edge(3,1):\ c(3,1) = birr\ 5 < v_1 - u_3 = birr\ 14 - birr\ 9 = birr\ 5\ no\ change \ldots (4)$$

We see that edge $(1, 2)$ yields the greatest reduction. So we add $(1, 2)$ to the current solution. If there is no edge that reduces the cost of the solution,

then the current solution is optimal and we are finished.

Step II.C. Determining a Cheaper Spanning Tree Solution

In Step II.B, an edge was chosen to be added to the current solution. the choice is edge $(1,2)$. When this edge is added to the set of edges in the current solution. We have a unique cycle, for the solution in Figure 3.6. The cycle is $(1,2),(2,2),(2,1),(1,1)$ by lemma 3.1.1, we get a cheaper solution by increasing the flow in the odd edges, $(1,2)$ *and* $(2,1)$. as much as possible while decreasing the flow in the even edges,$(1,1)$ *and* $(2,2)$, correspondingly. The net reduction is $c(1,2)+c(2,1)-c(1,1)-c(2,2)=2+6-4-7=-3$. Note that this calculation confirms our previous analysis that we will save *birr* 3 for each unit shipped on edge $(1,2)$.

The limiting constraint is when the shipment in an even edge decrease to 0. In our example, since the current shipment in both $x(1,1)$ and $x(2,2)$ is 40, we can increase the shipment in $(1,2)$ and $(2,1)$ by 40 and reduce the shipments in $x(1,1)$ by 40. The new solution, cheaper by $40\times birr\ 3 = birr\ 120$ than the previous solution. Note that although the flow in edges $x(1,1)$ and $x(2,2)$ both decreased to 0, we cannot drop both edges. If we did, the new solution would not be a spanning tree. We arbitrary pick one of $x(1,1)$ and $x(2,2)$ to say in new solution but with shipment level 0. See Figure 3.7.

Now we repeat the three parts of Phase II to find a still better solution. In Step II.A, using the solution in Figure 3.7, we compute new prices at warehouses and stores. Starting with $u_1 = birr\ 10$ and using edge $(1,2)$, we find $v_2 = u_1 + c(1,2) = birr\ 10 + birr\ 2 = birr\ 12$. Continuing as before,we obtain the following set of prices.

$Warehouse\ \ 1: \quad u_1 = 10\ birr \quad Store1: \quad v_1 = 11\ birr$

$Warehouse\ \ 2: \quad u_2 = 5\ birr \quad Store2: \quad v_2 = 12\ birr$

$Warehouse\ \ 3: \quad u_3 = 6\ birr \quad Store3: \quad v_3 = 6\ birr \ldots (5)$

In Step II.B, we see if edges not in the current solution produce a reduction in the transportation costs. The calculation yield:

$edge(1,3):\ c(1,3) = birr\ 0 > v_3 - u_1 = birr\ 6 - birr\ 10 = birr\ -4\ increase\ of\ birr\ 4.$

$edge(1,1):\ c(1,1) = birr\ 4 > v_1 - u_1 = birr\ 11 - birr\ 10 = birr\ 1\ increase\ of\ birr\ 3.$

$edge(2,3):\ c(2,3) = birr\ 0 < v_3 - u_2 = birr\ 6 - birr\ 5 = birr\ 1\ decrease\ of\ birr\ 1.$

$edge(3,1):\ c(3,1) = birr\ 5 = v_1 - u_3 = birr\ 11 - birr\ 6 = birr\ 5\ no\ change$

Since an edge $(2,3)$ decrease the cost by *birr* 1 and when we consider a unique cycle $(2,3),(3,3),(3,2),(2,2)$. We cannot decrease a shipment in even edges since edge (2,2) is with shipment 0 so adding edge $(2,3)$ in the current solution and dropping even edge in the cycle does not change the total transportation cost. Thus the current solution is optimal. See the Figure below.
The transportation cost is *birr* 500.

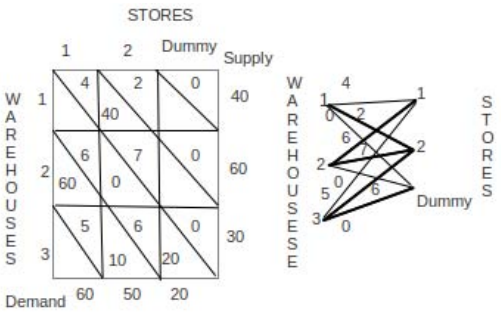


Figure 3.7:

Example 3.2.1. *Solve the transportation problem in which warehouse* 1 *has* 30 *units, warehouse* 2 *has* 30 *units and warehouse* 3 *has* 30 *units and in which store* 1 *needs* 20 *units, store* 2 *needs* 20 *units, and store* 3 *needs* 50 *units. See the transportation tableau of this problem below.*

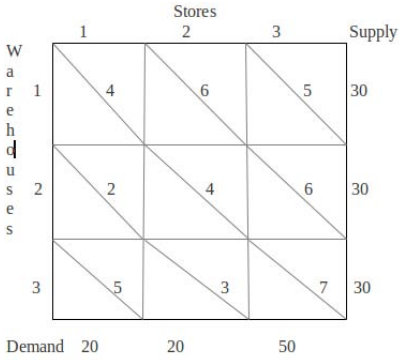


Figure 3.8:

We first find an initial solution using Northwest Corner Rule, Minimum Cost Rule or VAM(Vogel's Approximation Method).

(i). *Northwest corner rule, We start at edge* $(1,1)$, $x(1,1) = min(30,20) = 20$. *We satisfies the demand of store 1. So we delete column 1 and the modified transportation tableau as follows*

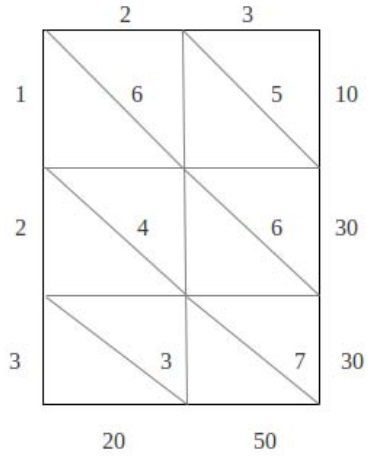


Figure 3.9:

now we take an edge $(1,2)$ *and* $x(1,2) = min(10,20) = 10$, *here we used the supply at warehouse 1 so we delete row 1. See Figure below.*

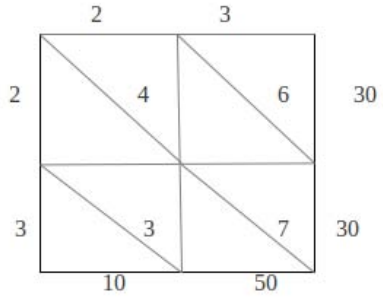


Figure 3.10:

We continuing this procedure of repeatedly making the entries in the Northwest corner of the remaining tableau as large as possible. The complete Northwest corner rule solution is shown below and The total transportation cost is $20 \times birr\ 4 + 10 \times birr\ 6 + 10 \times birr\ 4 + 20 \times birr\ 6 + 30 \times birr\ 7 = birr\ 510$.

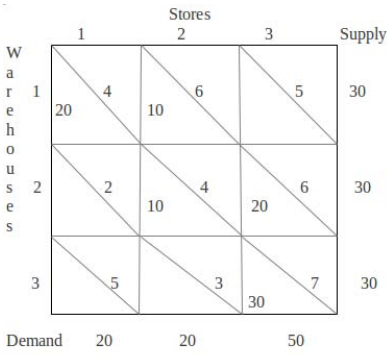


Figure 3.11:

(ii). *The Minimum cost rule, we start at edge with minimum cost, edge* $(2,1)$, $x(2,1) = min(20,30) = 20$, *we satisfies the demand of store one, so we delete the column one and the modified tableau is given below.*

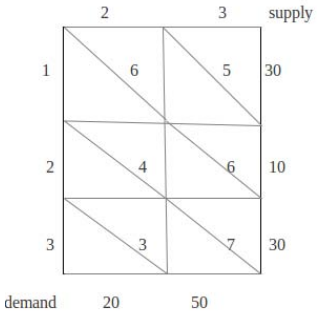


Figure 3.12:

Next, we take an edge with minimum cost in the above tableau, edge $(3, 2)$*,* $x(3, 2) = min(20, 30) = 20$*, we satisfies the demand at store* 2 *so we delete the column. Continuing this procedure of repeatedly until all allocation have been assigned. See the figure below.*

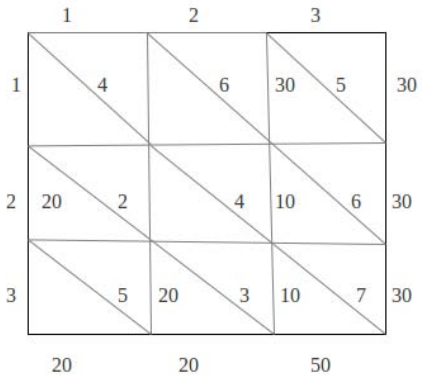


Figure 3.13:

The transportation cost is $30 \times 5 + 20 \times 2 + 10 \times 6 + 20 \times 3 + 10 \times 7 = 380$

(iii)*. The Vogel's Approximation Method. For each column and row, we determine its penalty cost by subtracting their two of their least cost. See the figure below.*

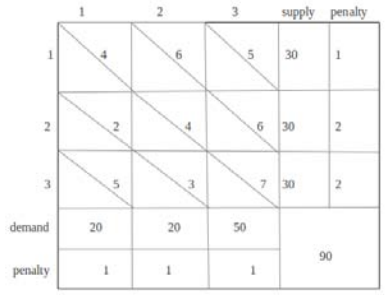


Figure 3.14:

We can select row 2 *or row* 3*, let us take row* 3 *and edge* $(3, 2)$ *with minimum cost,* $x(3, 2) = min(20, 30) = 20$*. We satisfies the demand at store* 2 *so we delete the column and the modified tableau is shown below.*

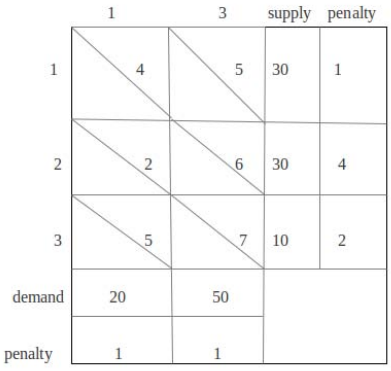


Figure 3.15:

Consider an edge $(2, 1)$ *in figure,* $x(2, 1) = min(2, 1) = 20$*. The demand at store* 1 *is satisfies so we delete column* 1 *and Continuing this procedure repeatedly, we get an initial solution of the problem using VAM, see the figure* 3.16.

The transportation cost is $20 \times 2 + 30 \times 5 + 10 \times 6 + 20 \times 3 + 10 \times 7 = 380$.

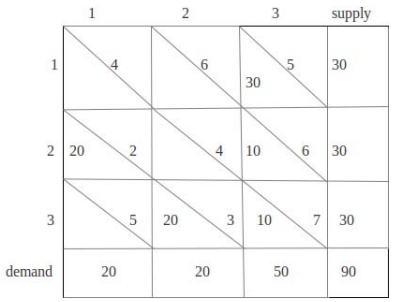


Figure 3.16:

Note

After calculating the cost of transportation by the above three methods, one thing is clear that Vogel's Approximation method and Minimum cost method give a better solution than Northwest corner rule. So it is always worth while to spend sometimes finding a " good" initial solution because it can considerably reduce the total number of iteration required to reach an optimal solution. The second part is finding a better solution using MODI, consider the initial solution we have using VAM.

By step II.A, we determine the selling price of each warehouses and stores. let $u_1 = birr\ 10$ *be a price at warehouse 1. then using the edges in the current solution we have a price as follows;*

$Warehouse\ \ 1: \quad u_1 = 10\ birr \quad Store1: \quad v_1 = 11\ birr$

$Warehouse\ \ 2: \quad u_2 = 9\ birr \quad Store2: \quad v_2 = 11\ birr$

$Warehouse\ \ 3: \quad u_3 = 8\ birr \quad Store3: \quad v_3 = 15\ birr.$ *Step II.B, we determine which edge is to add in the current solution to reduce the transportation cost.*

$$edge(1,1): \ c(1,1) = \ 4 > v_1 - u_1 = \ 11 - \ 10 = 1\ increase\ of\ 3.$$

$$edge(1,2): \ c(1,2) = 6 > v_2 - u_1 = 11 - 10 = 1\ increase\ of\ 5.$$

$$edge(2,2): \ c(2,2) = 4 > v_2 - u_2 = 11 - 9 = 2 \ increase \ of \ 2.$$

$$edge(3,1): \ c(3,1) = 5 > v_1 - u_3 = 11 - 8 = 3 \ increase \ of \ 2.$$

Since non of this edges reduce the cost of the current solution, the current solution is optimal, and we are finished.

Conclusion

The problems of maximizing the flow and finding minimum-cost flow in networks, for which algorithms were described have long been of interest in operations research. The question of finding an efficient algorithms has more recently been the interest of computer scientists as the reference that follows make clear. Networks have been the subject of many variation and generalizations as their applications have warranted.

References

[1]. Alan Tucker, *Applied Combinatorics*: John Wiley and Sons, Inc. 1995.

[2]. Alan Gibbons, *Algorithmic Graph Theory*: Press syndicate, 1985.

[3]. J.A. Bondy and U.S.R Murty, *Graph Theory*: Springer, 2008.

[4]. J.A. Bondy and U.S.R Murty, *Graph Theory With Application*: The macmillan press LTD, 1976.

[5]. Gerd Finke, *Operations Research and Networks*: John Wiley and Sons, Inc. 2008.

[6]. Ralph P.Grimaldi, *Discrete and Combinatorial Mathematics*: Pearson Education INC, 2004.