ON APPLICATION OF NETWORK SIMPLEX METHOD TO SOLVE MINIMUM COST NETWORK FLOW PROBLEM

ADDIS ABABA UNIVERSITY

COLLEGE OF NATURAL AND COMPUTATIONAL SCIENCE

DEPARTMENT OF MATHEMATICS

A Thesis Submitted to the Department of Mathematics, Addis Ababa University in Partial Fulfillment of the Requirements for the Degree of Master of Science in Mathematics

By: Getachew Beyene

Advisor: Mesfin Masre (PhD)

September, 2024

Addis Ababa, Ethiopia

Table of Contents

Permission

This is to certify that this thesis is prepared by Mr. Getachew Beyene in the Department of mathematics, Addis Ababa University, under my supervision. I hereby also confirm that the thesis can be presented for evaluation by examiners and eventual defense.

Advisor's Name	Signature	Date

ADDIS ABABA UNIVERSITY

DEPARTMENT OF MATHEMATICS

THESIS FINAL SUBMISSION

APPROVAL SHEET

We, the undersigned members of the boarded of the examiners of the final open defense by: Getachew Beyene have read and evaluated his thesis report entitled “On application of Network Simplex Method to solve Minimum Cost Network Flow Problem ”, and examined the candidate. This is therefore to certify that the thesis has been accepted in partial fulfillment of the requirement for the degree of Master of Science in Mathematics.

.

_____	_____	_____
Name of the chairperson	Signature	Date
_____	_____	_____
Name of Advisor	Signature	Date
_____	_____	_____
Name of Examiner 1	Signature	Date
_____	_____	_____
Name of Examiner 2	Signature	Date

ACKNOWLEDGEMENT

First of all, I would like to express my deepest gratitude to God and his mother Saint Marry who gave me patience, health and enabled me pass all the ups and down I faced since I started this post graduate program to accomplish this paper. I am also gratefully acknowledging my advisor Dr. Mesfin Masre , for his unreserved support and continued guidance in correcting this study by giving me conceptual and methodological feed backs. Then, I would like to sincerely thank my families who have shouldered the burdens of my responsibilities at home and for their patience while I was attending the summer courses and doing this project.

ABSTRACT

The problem of the minimum cost flow (MCF) is to send a flow from a set of supply nodes to a set of demand nodes via the arc of a network at a minimum total cost, without violating the lower and upper limits of the flow through the arc. The framework work of the MCF is particularly broad and can be used to model a number of specialized network problems, including assignment, transport and transfer problems, the shortest path problem and the maximum flow problem.

The network simplex method is described to solve the minimum cost network flow problem, one of the most fundamental and a significant problem of network optimal design, and is applied to the network flow programming problem using simplex algorithms. The Network Simplex method describes the basic solutions for the problem of network flow programming and provides procedures for calculating the basic and double solutions associated with a given basis to find the optimal solution.

CHAPTER ONE

INTRODUCTION AND PRELIMINARIES

1.1 Basic terminologies in graph theory

Graph theory is a branch of mathematics that deals with arrangements of certain objects and relation ships between these objects. Graph theory is broadly classified into two: nondirected graphs and directed graphs (digraphs).

A directed graph $G = (V, E)$ consists of a set V of vertices and a set $E \subseteq V \times V$ of edges. When the relation E is symmetric, G is called an undirected graph, and we can write edges as unordered pairs $\{u, v\} \in E$ for $u, v \in E$. The degree of vertex $u \in V$ in graph G is the number $\{v \in V : (u,v) \in E \text{ or } (v,u) \in E\}$ of other vertices connected to it by an edge. A walk from $u \in V$ to $w \in V$ is a sequence of vertices $v_1,\ldots, v_k \in V$ such that $v_1 = u$, $v_k = w$, and for $i = 1,\ldots,k-1$. In a directed graph, we can also consider an undirected walk where $(V_i,V_{i+1}) \in E$ or $(V_{i+1},V_i) \in E$ for $i = 1,\ldots,k-1$. A walk is a path if $v_1,\ldots, v_k$ are pairwise distinct, and a cycle if $v_1,\ldots, v_{k-1}$ are pairwise distinct and $v_1 = v_k$. A graph that does not contain any cycles is called acyclic. A graph is called connected if for every pair of vertices $u, v \in E$ there is an undirected path from u to v. A tree is a graph that is connected and acyclic. A graph $G' = (V',E')$ is a sub graph of graph $G = (V,E)$, if $V' \subseteq V$ and $E' \subseteq E$. In the special case where G' is a tree and $V' = V$, it is called a spanning tree of G. In a network, each arc or edge is associated with some numerical value (real number), variably called its cost, weight, length (distance) or some other variable depending on the application. It is denoted as c_{ij} for each arc $(i,j) \in A$. Weights of the arcs are very important because some algorithms impose further restrictions on weights.

Definition 1.2 : Two or more edges joining the same pair of vertices are called multiple (parallel) edges. An edge joining a vertex to itself is called a Loop.

Depending on loops (self loops) and parallel edges we state the following types of graphs.

Simple graph**:** A graph with no loops and parallel edges is called simple graph.

Multigraph**:** A graph which consists of parallel (multiple edges) is called a multigraph.

Directed graphs are graphs in which the edges are one way. Such graphs are frequently more useful in various dynamical systems such as: Digital computer, Flow system, Communication system, Transportation system.

Definition 1.3 : A digraph D is a graph consisting of two things:

i) A set V whose elements are called vertices, Points or node of D

ii) A set E whose elements are order pairs (u,v) of distinct vertices called arcs or directed edges of D.

Suppose e = (u,v) is a directed edge in a digraph D. Then the following terminologies are used. Some are e begins at u and ends at v, u is the origin or initial point of e where as v is destination or terminal point of e, **v** is the successor of **u** and **u** is the predecessor of **v,** u is adjacent to v where as v is adjacent from u, If u = v then e is a loop, and The set of all successor of a vertex u is : suss(u) = $\{v \in V : (u,v) \in E\}$

The sum of the out degrees of the vertices of the diagraph G equals the sum of the in degrees of the vertices, which equals the number of edges in G. A vertex u in a diagraph with zero in degree is called a **source** and a vertex u with zero outdegree is called a **sink.**

Undirected graph: - It can be defined in the same way as the directed graph. An arc that connects the node pairs u and v can be thought of unordered pairs as {u,v}. It is possible to think of an undirected arc (u, v) as a two-way flow that goes from node u to node v or from v to u. The degree of a vertex in an undirected graph is the number of edges incident with it, except that loop at a vertex contributes twice to the degree of that vertex. A degree in a graph is mentioned to be the number of edges connected to a vertex. It is denoted deg(v), where v is a vertex of the graph.

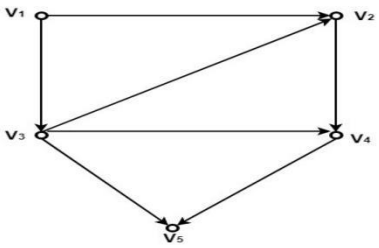


fig 1.1 directed graph

Subgraph : A graph H is called a subgraph of G if every vertex of H is also a vertex of G and every edge of H is also an edge of G. Symbolically H is a subgraph of G if $V(H) \subseteq V(G)$ *and* $E(H) \subseteq E(G)$.

Note: A graph and its null graph are trivial subgraphs.

- A subgraph H of G is called a spanning subgraph of G iff V(H) = V(G).
- A subgraph H of G is called a proper subgraph if $H \neq G$.

Definition 1.4 : A path in a graph G consists of an alternating finite sequence of vertices and edges of the form: $v_0,e_1,v_1,e_2,v_2,\ldots e_{n-1},v_{n-1},e_n,v_n$ Where each edge e_i contains the vertices v_{i-1} and v_i (which appears on the sides of e_i in the sequence). Some times, with the understanding that consecutive vertices are adjacent, the path can be written as: $(v_0 ,v_1 ,...v_n)$ with the idea that consecutive vertices are adjacent.

Given the path $p = (v_0,e_1,v_1,e_2,v_2,\ldots,v_{n-1},e_n,v_n)$. Then:

- P is said to **traverse** the edges e1,e2,---,en and **visit** the vertices $v_0,v_1,\ldots,v_{n-1},v_n$

- v_0 is called the initial vertex and v_n is the terminal vertex of p.

- The number n of edges is called the length of the path.

- A path is said to be closed if $v_0 = v_n$.

- P is said to be open if $v_0 \neq v_n$.

- P is called a simple path if all the vertices are distinct.

- P is called a cycle if it is a closed simple path (i.e., all the vertices are distinct except v_0 and v_n).

- P is called a trial if all the edges are distinct (i.e., a path that does not traverse the same edge more than once).

- A loop is a cycle of length 1.

- A simple path of length ≥ 1 with no repeated edges and whose end points are equal is called a circuit.

A graph G is **connected** if there is a path between any two of its vertices. That is, a graph is connected if it is possible to go from any vertex to any other by following the edges of the graph. A graph that is not connected is said to be disconnected.

Definitions 1.5 :

- ✓ A graph T is called a **tree** if T is connected and T has no cycle.
- ✓ A graph without a cycle is said to be cycle-free (**acyclic**) graph.
- ✓ The tree consisting of a single variable with no edges is called the degenerate tree.
- ✓ Since a loop is a cycle of length one, then a tree is a loop free graph.

1.2 Matrix representation of graphs

To represent a graph, we just need the set of vertices, and for each vertex the neighbors of the vertex (vertices which is directly connected to it by an edge). If it is a weighted graph, then the weight will be associated with each edge.

There are different ways to optimally represent a graph, depending on the density of its edges, type of operations to be performed and ease of use.

Graph can be represented in the form of matrix. The matrices that can be formed by a graph are given below.

A. **Incidence Matrix Representation:**

Definition 1.6 : Suppose G is a graph with vertices V = $\{v_1, v_2, v_3, \ldots, v_n\}$. The incidence matrix C= $[c_{ij}]$ of the graph G is given by:

$$C_{ij} = \begin{cases} 1, if\ e_j\ is\ incident\ on\ v_i \\ 0, otherwise \end{cases}$$

Example: Consider the undirected graph G as shown in fig. Find its incidence matrix C.

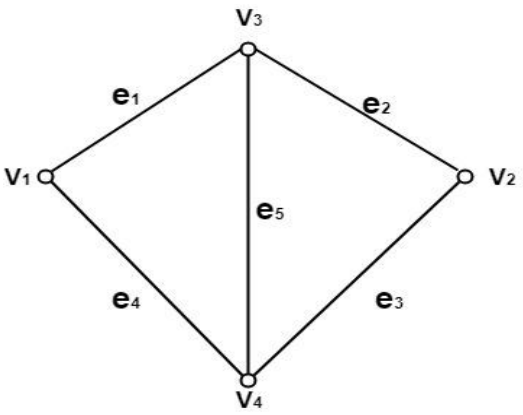


fig 1.2 undirected graph

Solution: The undirected graph consists of four vertices and five edges. Therefore, the incidence matrix is an 4 x 5 matrix, (i,e v in rows, e in column) which is shown in Fig 1.2

$$C = \begin{pmatrix} 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 1 \end{pmatrix}$$

Adjacency Matrix Representation:

Definition 1.7 : Suppose G is a graph with m vertices and suppose the vertices have been ordered, say $v_1, v_2, - - - , v_m$. Then the adjacency matrix $A = [a_{ij}]$ of the graph G is the *mxn* matrix defined by:

$$a_{ij} = \begin{cases} n, \ if \ there \ are \ n \ edges \ joining \ v_i \ and \ v_j \\ 0, otherwise. \end{cases}$$

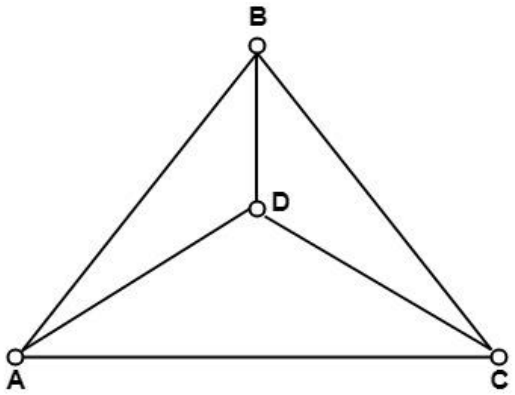


fig 1.3 undirected graph

solutiion: Since graph G consist of four vertices. Therefore, the adjacency matrix wills a 4 x 4 matrix. The adjacency matrix is as follows in fig above:

$$A = \begin{pmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{pmatrix}$$

1.3 Spanning Tree

Definition 1.8 : - A spanning tree of a connected graph G is acyclic connected Subgraph of G which contains all the vertices of G.

One practical pattern for affordably linking each node in a network is a spanning tree. Between any two nodes there is a unique path along the tree; the number of arcs in the spanning tree is equal to the number of nodes minus one. This minimization is well defined since a network can only have a finite number of spanning trees.

Applications include the design of different kinds of distribution networks, where the edges represent communication links (fiberglass phone lines, data transmission lines, cable TV lines, etc.) and the nodes represent cities, centers, etc. High voltage power transmission lines, water and natural gas pipelines, highways, etc. The goal is to create a network architecture that connects every node with the least amount of pipework, cable, or other resource. There's also the minimum cost spanning tree issue.as a sub problem in various routing algorithms, including the traveling salesman problem.

Some basic properties of spanning tree

- Tree is a connected graph/network/ without any cycle / no cycle/,
- A tree with n-1 arcs/edges from a spanning tree in a graph with "n" nodes.
- An arc added to a spanning tree creates a unique cycle.
- When an arc is eliminated from a spanning tree, the tree splits into two distinct trees.
- A collection of arcs connected in a circle is called a circuit in a network, but a collection of arcs connected in a circle without any circuits or other connections is called a tree. A connected subset of the graph devoid of cycles forms a tree.

1.4 Shortest path problems

The basic problem is then to determine one or more shortest (or least cost) routes between a source vertex and a target vertex where a set of edges are given. In several cases SP algorithms

also provide shortest paths from a single source to all or many of the other vertices in a network as a side-effect of the procedure. Such algorithms are known as *single-source* SPAs or single-source queries. The set of shortest paths generated from a single source is known as a shortest path tree (SPT).

The determination of shortest paths can be specified as a linear programming problem, as follows. Let *s* be the source vertex, *t* be the target vertex and let $c_{ij} > 0$ be the cost or distance associated with the link or edge (*i*,*j*). Then we seek to minimize *z*, where:

$$\sum_{ij} c_{ij} x_{ij}$$

$$\sum_j x_{ji} - \sum_k x_{ik} = m$$

Where m=0 for i ≠1

m= 1 for i = t

m= -1 for i = s , $x_{ij} \in \{0, 1\}$

CHAPTER TWO

NETWORK SYSTEMS

2.1 Definition and Examples

A network is made up of a set of lines called edges or arcs that connect a collection of points called vertices or nodes. Nodes can be divided into three categories: demand/destination/node, intermediate/transshipment/node, and supply/source/node. The nodes in a network can be represented as circles, and the arcs that connect them as lines. The lines in a directed network are arrows pointing in the right direction. Many examples of networks come to us, such as

We are all familiar with computer networks, highway networks, telecommunication networks, water delivery network systems, and many more. Networks are particularly useful for modeling due to their straightforward mathematical structure. An easy way to visualize a network is with a graph, denoted G = (N, A), consisting of a set N of n nodes. As well as a group A of m directed arcs.The cost per unit flow on each arc (i, j) in A is represented by a numerical value called its cost, which is indicated by the symbol C_{ij}. It is assumed that the relationship between the flow cost and flow volume is linear. Additionally, we associate a capacity with each arc (i, j) ∈A. The maximum amount that can flow on the arc is indicated by U_{ij}, and the minimum amount that must flow on the arc is indicated by the lower bound, l_{ij}.

We assign an integer value b_i denoting the supply and demand of each node i∈N.

A Node i is a supply node, if $b_i > 0$ and a node i is a demand , if $b_i < 0$.and if $b_i = 0$, node i is a transshipment node.

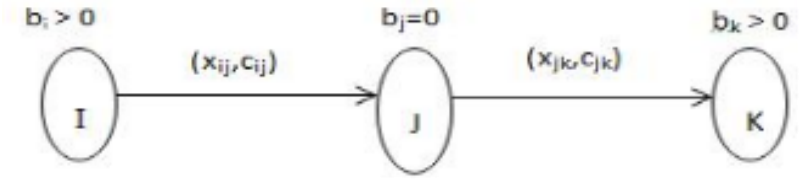


Fig 2.1 Representation of node, arc and Types of nodes

Degree: The number of incoming and outgoing arcs for the node.

Source (initial) node**:** is a node which has only outgoing arrows or edges

Terminal (sink) node: is a node which has only incoming arrows.

Incident arc: If an arc (i, j) originates at node i and descends to node j, it is referred to as an incident arc. Total Price:The sum of all edges' costs multiplied by the edge's value assigned by the flow equals the total cost of the flow. The unit of shipping cost for a unit entity on arc(i,j) is the cost cij.

Residual Graph: Residual graph of a flow network is a graph which indicates additional possible flow. If there is a path from source to sink in residual graph, then it is possible to add flow. Every edge of a residual graph has a value called residual capacity which is equal to original capacity of the edge minus current flow.

Bipartite Graph: A graph G = (N, A) is a bipartite graph if we can partition its node set into two subsets N_1 and N_2 so that for each arc (i, j) in A either $i \in N_1$ and $j \in N_2$, or $j \in N_2$ and $i \in N_1$.

Cycle: A cycle consists of a sequence of adjacent and distinct nodes in a graph. The only exception is that the first and last nodes of the cycle sequence must be the same node.

In this way, we can conclude that every cycle is a circuit, but the contrary is not true. Furthermore, another inferring is that every Hamiltonian circuit is also a cycle. So, we call a graph with cycles of cyclic graphs. Oppositely, we call a graph without cycles of acyclic graphs. Finally, if a connected graph does not have cycles, we call it a tree.

Example 2.1:The image next presents an example of a cyclic graph, acyclic graph, and tree:

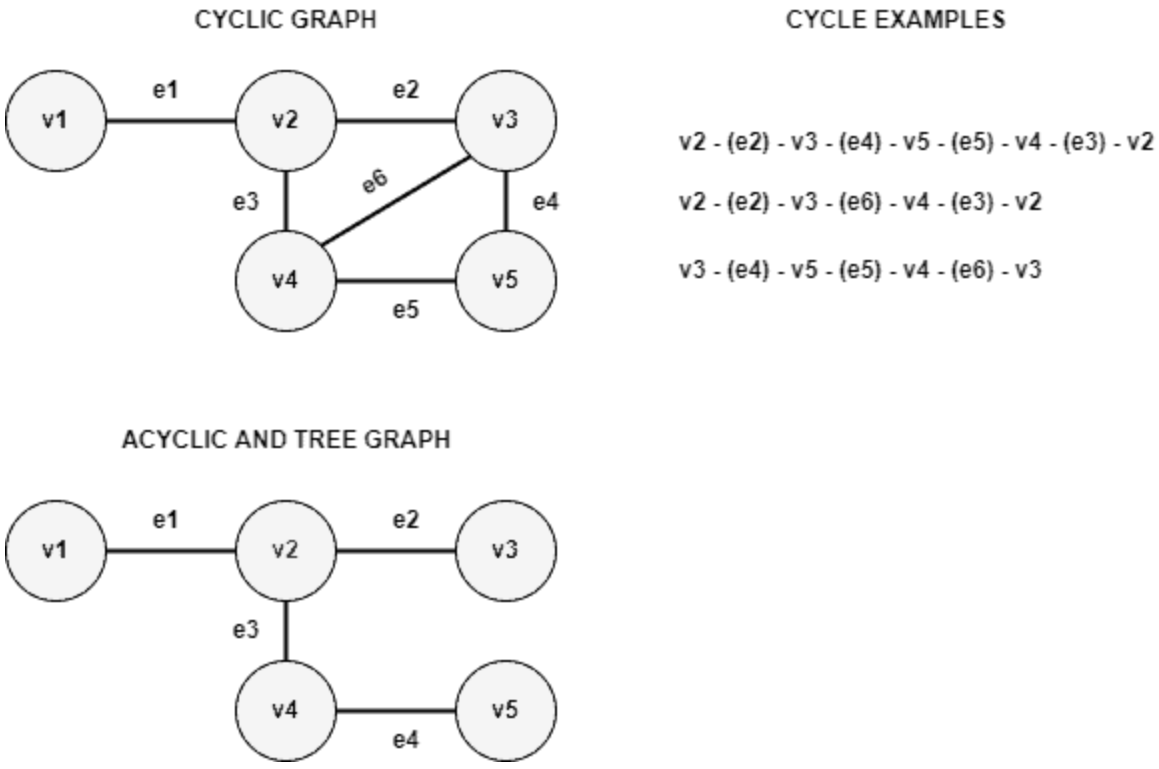


Fig 2.2 acycle graph

Cycle detection is a particular research field in graph theory. There are algorithms to detect cycles for both undirected and directed graphs.

Pivot: - is the process of interchanging the entering and the leaving arcs in network simplex algorithm.

2.2 Network flow problems

The problem domain of network flow is situated at the intersection of various fields of study, such as computer science, applied mathematics, engineering, and management. These fields deal with the optimization of network flows, or the movement of objects within a network. An example of a typical network-flow issue in industrial logistics that involves product distribution from manufacturing facilities (origins) to consumer markets (destinations). The product may be routed via intermediate points, such as warehouses or distribution centers, rather than being sent straight from the source to the destination. Furthermore, some of the shipping links might be limited by capacity restrictions. Reducing the variable cost of product shipping in order to satisfy customer demand is the problem's goal. The network's sources, destinations, and intermediate points are referred to as nodes, and the transportation links that connect them are known as arcs.

The network model describes flow configurations in a networked system, where the flow may include people, money, materials, or other resources. Flow diagrams provide a convenient way to describe these configurations and aid in the creation of reliable spreadsheet models. Network models are a special kind of linear programs because part of the model building can be done with a diagram. There are numerous other models available, such as transportation problems, assignment problems, shortest path problems, maximum flow problems, and minimum cost flow problems, despite the production/distribution problem being the one used as the motivating scenario.

Example 2.2: The graph given below in Fig (2.3) shows network flow problem. The nodes are represented by numbered circles and the arcs by arrows. The arcs are directed so that, material can be sent from node 1 to node 2, but not from node 2 to node 1. Generic arcs will be denoted by i– j, so that 4–5 means the arc from node 4 to node 5. In network flow problem some pairs of nodes, may not directly connected. for example, node 1 and 5, are not connected directly by an arc.

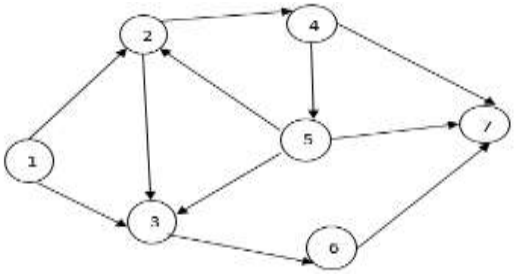


fig 2.3 Network flow model

Three key benefits of network models over linear programming are as follows:These are

1.**They can be solved very quickly**. Problems whose linear program would have 10000 rows and 30,000 columns can be solved in a matter of seconds. This allows net-work models to be used in many applications (such as real-time decision making). For which linear programming would be inappropriate.

They have naturally integer solutions. By recognizing that a problem can be formulated as a network program, it is possible to solve special types of integer programs without resorting to the ineffective and time consuming integer algorithms.

3. **They are intuitive programming**. Network models provide a language for talking about problems that are much more intuitive than the \variables, objective, and constraints" language of linear and integer programming.

2.3 Matrix Representation of Networks

In network representation typically, we focus on two types of information. These are:

- ✓ Network topology, i.e. the network's node and arc structure; and
- ✓ Information about expenses, capabilities, and supply/demand related to the network's node and arc. We require a mathematical and coding representation of networks while working with them. Network space, which is just the collection of all conceivable networks, is where a network exists. Since using standard mathematics in network space is somewhat cumbersome and abstract, it is often preferable to express networks using sets of numbers to provide more concreteness. More precisely, we frequently want to use matrices to represent networks. Networks can be represented using a variety of matrix forms. Some of these renderings are as follows: -

Adjacency matrix Representation

The node-node adjacency matrix/or simply adjacency matrix/ representation, stores the network as an n x n matrix $A = (a_{ij})$The matrix has a row and a column corresponding to every node, and its ij^{th} entry a_{ij} equals 1 if the arrow is from node i to node j , 0 other wise

The adjacency matrix is defined as $A= (a_{ij})$, (already stated in chapter one)

Example 2.3 : let us see the following network /graph/, where the small circles represent nodes with their corresponding representative numbers in them and each directed segment represent ij^{th} arcs/edges/

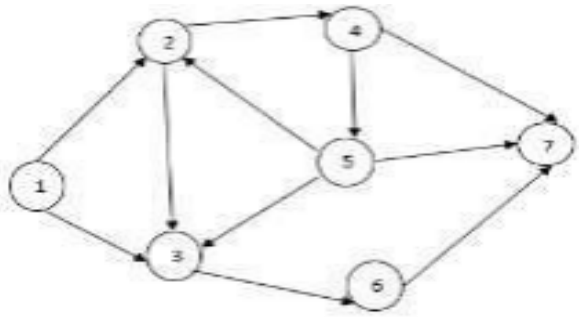


Fig.2.4 Directed graph for adjacent matrix

Example 2.3: From fig 2.4 above graph, adjacent matrix as follows

$$A = \begin{array}{c} \text{node/node} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \end{array} \begin{array}{c} \begin{array}{ccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 \end{array} \\ \begin{bmatrix} 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \end{array}$$

Note:

1. Node 1 is source node, because the first column has elements only zero. i.e. no arrow comes from any other vertices to this node.

2.The number of +1 in each row shows the number of arrows coming to node.

3.Node 7 is terminal/sink/ node, since all elements in row 7 are zero i.e. there is no outgoing arrows or edges from this node.

Incidence Matrix Representation

In this representation, the network is kept as a n x m matrix A, with one row representing each network node and one column representing each arc. Only two entries in the column that corresponds to arc (i, j) are non-zero. The row that corresponds to node i has a +1, whereas the row that corresponds to node j has a -1. This representation is significant because it has various

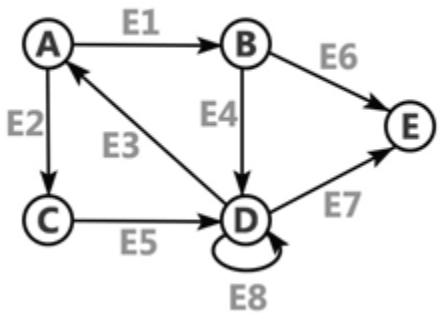
intriguing theoretical aspects and shows the constraint matrix of the least cost flow issue in an understandable manner.

If a directed graph G consists of n vertices and m edges, then the incidence matrix is an n x m matrix A = (a_{ij}) and defined by

$$a_{ij} = \begin{cases} 1, if\ arc\ k_j\ \ starting\ in\ node\ i \\ -1, if\ arc\ k_j\ \ ending\ in\ node\ i \\ 0, otherwise \end{cases}$$

For i = 1,2,3,…,n , j = 1,2,3,…,m

Example 2.4: Consider the following directed graph incidence matrix



	E1	E2	E3	E4	E5	E6	E7	E8
A	1	1	-1	0	0	0	0	0
B	-1	0	0	1	0	1	0	0
C	0	-1	0	0	1	0	0	0
D	0	0	1	-1	-1	0	1	1
E	0	0	0	0	0	-1	-1	0

CHAPTER THREE

MINIMUM COST NETWORK FLOW

3.1 Characteristic of Minimum Cost Network Flow

A minimum cost network flow program has the following characteristic:

- **Variables**

The unknown flows in the arcs, the x_{ij} , are the variables.

- **Flow conservation at the nodes**

The total flow in to a node equals the total flow out of a node. It makes things easier later if we follow the convention of writing the flow conservation equation at a node as:

$$\sum_{i=1}^{n} b_i = 0$$

- **Source and sink nodes**

Some nodes are connections to the environment surrounding the network. At these entryway nodes, there may be a net gain of flow into the network (source node), or a net loss of flow out of the network (sink node). To emphasize that flow conservation still holds at source and sink nodes shown on the network diagram. The phantom arc will be an inflow for a source node, and an out flow for a sink node.

- **Capacity property**: Capacity Constraint makes sure that the flow through each edge/arc/ is not greater than the capacity.
- **Integrality Property:** In minimum-cost flow problem, assuming that the upper and lower bounds on the variables are integers and the right hand-side values for the flow-balance equations are integers, the values of the basic variables are also integers when the non-basic variables are set to their upper or lower bounds.
- It is simple to determine what kind of node is connected to each relationship when all of the flow conservation equations (or inequalities) are written using the outflows-inflows paradigm by examining the value of the right-hand side constant. For a basic flow conserving node, the constant will be zero; for a source node, it will be positive; and for a sink node, it will be negative.

Bounds on the arc flows

The variables in the model, or the flows in the arcs, could have upper and lower boundaries. On an arc flow, $x_j \leq u_j$ is an upper bound while $x_j \geq l_j$ is a lower bound. An upper bound is used, for instance, when the maximum water flow through a certain pipe is restricted due to the pipe's diameter and internal roughness.

Although upper bounds are simple to comprehend, why might a lower bound possibly exist?

- By default, there is no upper bound and a lower bound of zero for arc flow. In a network model, it is common for nearly all of the arcs to have the default bounds, with only a small number of arcs having specified upper or lower bounds. These arcs are usually those at the "edges" of the network, which could represent upper limits on the rates at which raw materials flow into the network or lower limits on the rates at which production occurs at the main outflow from the network.

- There are modeling systems that permit an arc flow to have a negative lower bound. I strongly advise against doing this! A negative flow in the arc is interpreted as a flow that is going backwards.

  The most advantageous aspect of a network model is instantly destroyed by this: the intuitive comprehension of the system that one gains by examining the network diagram. Although you would expect the flows to follow the arrowheads' directions, allowing negative flows could cause the arrowheads to lie. If required, two-way flow can also be accommodated by coupling arcs with opposite orientations between two nodes. Price per flow unit.Every arc has an associated cost per unit of flow, c_j. Most arcs in many network models have no cost per unit of flow; arcs at the network's "edges" are usually the ones that incur charges. C_j has a default value of zero.

Fundamental information concerns about minimum cost network flow:

- Transportation costs between two nodes;

- The amount of flow produced or consumed at each node;
- Possibly, an upper limit on maximum flow on each arc (i.e., capacity).

An (integer) value bi, denoting the quantity of flow generated (if $b_i > 0$) or consumed (if $b_i < 0$) at node i, is provided for each i, i =1,..., n. The terms "source" and "supply" are occasionally used to describe the nodes that generate flow. Sink nodes are those that consume flow, and b i is referred to as demand. If $b_i=0$, meaning that node i is a transit node because it neither produces nor consumes flow. Keep in mind that this division of nodes into three categories is solely based on supply and demand, and is totally independent of the network's structure.

Network problems can be cast as minimum cost network flow programs.

Consider G = (N, A) be a directed network defined by a set of nodes N={1,2,3,…,n} and a set of directed arc A= {1,2,3,…,m} linking pairs of nodes in N. Each arc (i, j) ∈ A has an associated cost C_{ij} that denotes the cost per unit flow on that arc. We assume that the flow cost varies linearly with the amount of flow. We also associate with each arc (i, j) ∈ A a capacity U_{ij} that denotes the maximum amount that can flow on the arc and a lower bound l_{ij} that denotes the minimum amount that must flow on the arc. We associate with each node i ∈ N an integer number b_i representing its supply/demand. If $b_i > 0$, node i is a supply node; if $b_i < 0$, node i is a demand node. if $b_i = 0$, node i is a transshipment node.The minimum Cost Flow (MCF) problem is to send the required flows from the supply node to the demand nodes(satisfying the demand constraints), at the minimum cost. The flow bound constraints.The decision variables in the minimum cost flow problem are arc flows and we represent the flow on an arc(i, j) ∈ A by X_{ij}.

x_{ij} = number of units of flow sent from node i to node j through arc (i, j).

c_{ij} = cost of transporting one unit of flow from node i to node j through arc (*i, j*).

l_{ij} = lower bound on flow through arc (i, j). If there is no lower bound, let $l_{ij} = 0$.

u_{ij} = upper bound on flow (Capacity of) through arc (i, j), if there is no upper bound, let ($u_{ij} = \infty$)

The minimum cost flow problem is an optimization model formulated as follows:

Minimize $\sum_{(i,j)\in A} C_{ij}\, x_{ij}$ ……(1)

Subject to $\sum_{j=1}^{n} x_{ij} - \sum_{k=1}^{n} x_{ki} = b_i$, i ,j = 1,2,3,…,n …….(2)

$0 \le l_{ij} \le x_{ij} \le U_{ij}$, $x_{ij} \ge 0$ for all (i, j) ∈A ……(3)

The Total net supply must equal equal zero can be seen by summing the flow balance equations over all i ∈ N resulting in $\sum_{j=1}^{n} x_{ij} - \sum_{k=1}^{n} x_{ki} = \sum_{i=1}^{n} b_i = 0$

In matrix form we represent the minimum cost flow problem as follows:

Minimize CX

Subject to AX = b and $0 \leq l_{ij} \leq x \leq U_{ij}$

Where A is a node - arc incident matrix of the minimum flow problem having order n x m and a row for each node and a column for each arc.

For dual case the formulation of the problem is as follows: Associate the dual variable Z_i with the flow balance constraints for node i in (2) and U_{ij} is upper bound constraint of arc (i,j) in (3)

The dual problem is then : Maximize $\sum_{i \in N} b_i z_i$

Subjected to $z_i - z_j - u_{ij} \leq c_{ij}$, uij ≥ 0, for all $(i,j) \in A$

The objective function in the aforementioned equations is represented by equation (1), which adds up the costs of each arc in the network flow problem. The flow balance, or continuous flow conservation, is described by equation (2). The flow out of node i is represented by the first summation in (2), and the flow in of node i is represented by the second. The net flow produced at node i is represented by the difference between the two.Values b_i on the right side of (2) are zero otherwise, negative when node i consumes flow, and positive when it supplies flow. The node's outflow is more than its inflow if it is a supply node, its inflow is greater than its outflow if it is a demand node, and its outflow is equal to its inflow if it is a transshipment node. The lower bound and capacity restrictions, which we refer to as flow bound constraints, must also be satisfied by the flow. Usually, the flow boundaries simulate physical limitations or capabilities placed on the flows' operational ranges. Since the lower bounds on arc flows are usually zero in applications, we presume that they are also zero in any situation where they are not state.

Example 3.1 : The following example of a numerical model can be used to represent the minimum cost flow problem.

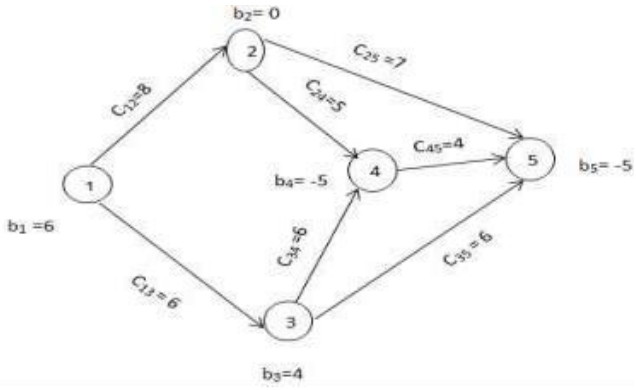


Fig 3.1 minimum cost flow problem model

The numerical model example (Fig. 3.1) above can be used to represent the minimum cost flow problem. As shown in this graphic, there are five nodes and seven arcs in this network flow problem. Every one of these arcs has an arch weight, which stands for the corresponding cost of flow or conveyance. Numbered circles stand in for the nodes, while arrows indicate the arcs. For example, material can be transferred from node 1 to node 2, but not from node 2 to node 1. This is because the arcs are oriented in that way. i – j or (i, j) is used to indicate generic arcs; so, 3–5 represents the arc from node 3 to node 5. Keep in mind that not all node pairs—like 1 and 5 for instance—have an arc connecting them directly. In this network flow problem, for instance, moving a unit from node 1 to node 2 costs 8, whereas moving it from node 1 to node 3 costs 6. We also have some values connected to these nodes. When b_1 equals 6, it indicates that there are six units available in node 1, zero units in node 2, four units in node 3, and five units needed in node 4, with the requirement represented by a negative value. Minus 5 denotes that node 4 requires 5 units, and again, minus 5 denotes that node 5 requires 5 units. The supply nodes are nodes 1 and 3. Positive quantities are accessible at supply nodes. Node 2 is a transitional node.

The zero value of the intermediate node might be interpreted as either a zero supply or a zero requirement. The requirements for destination nodes 4 and 5 are indicated by these negative numbers. At this point, we can also see that the problem is balanced because the total supply is equal to 6 + 4, or 10, and the total requirement is equal to 5 + 5, or 10. Therefore, the issue is balanced. Generally speaking, there are a number of nodes in this problem: some are supply points, some are destination sites, and some are intermediary points, also known as transshipment points. The challenge is to move a single good, which is available in 6 and 4 places here, respectively, to 5 and 5 places here while paying the least amount of transportation expenses.

As an illustration: We can see the feasibility property in the network flow model in section (fig 3.1), which is provided above. That is, supply and demand added together equals zero.

3.2 Linear programming formulation of MCNF

3.2.1 The Assignment problem

The assignment problem is a specific instance of the transportation problem that arises when all supply and demand are 1.The components of the graph G = (N, A) are as follows: N_1 and N_2 are two disjoint sets of equal size, hence $N = N_1 \cup N_2$. Assigning each element in N_1 to a single

element in N_2 is the challenge (people to machines, for example). It entails dividing up a variety of tasks among employees in an economical and efficient way. $X_{ij} = 1$, if element $i \in N_1$ is assigned to element $j \in N_2$. If not, $X_{ij} = 0$. The value of C_{ij} represents the cost of allocating $i \in N_1$ to $j \in N_2$. $b_i = 1$ for every i in N_1 and $b_j = -1$ for every j in N_2. Every upper bound is one, while every lower bound is zero. The LP formulation is: -

$$\text{Minimize} \quad \sum_{(i,j)\in A} C_{ij} X_{ij}$$

$$\text{Subject to} \quad \sum_{j:(i,j)\in A} X_{ij} = 1 \qquad \text{for all } i \in N_1$$

$$\sum_{i:(i,j)\in A} -X_{ij} = 1 \qquad \text{for all } j \in N_2$$

$$X_{ij} \in \{0, 1\}, \quad \text{for all } (i, j) \in A$$

3.2.2 The Transportation Problem

The network flow model without intermediate points is the transportation problem. It is clear that supply chain management and logistics depend heavily on transportation issues in order to save costs and enhance services. The Transportation Problem is based on a bi-partite graph, the set of arcs defined by $A = \{(i, j) /, i \in N_1; j \in N_2\}$ where $N_1$is the set of source nodes, and N_2 is the set of sink or destination nodes, such that $N = N_1 \cup N_2$. The goal is to determine the least expensive shipping strategy from the supply sources (N_1) to the destinations (N_2), where the cost of shipping a single unit from source i to destination j is denoted by C_{ij} and the number of units delivered (X_{ij}). The symbols b_i and $-b_i$ represent the supply and demand at the sources and destinations, respectively. The cost C_{ij} per unit of flow is equal to the cost per unit disseminated. All of the $u_{ij} = \infty$ since the transportation problem does not place upper bound constraints on any particular X_{ij}. This suggests that the top bound of the transportation problem is infinite. In LP form, the problem is stated:

$$\text{Minimize} \quad \sum_{(i,j)\in A} C_{ij} X_{ij}$$

$$\text{Subjected to} \quad \sum_{j:(i,j)\in A} X_{ij} = b_i \quad \text{for all } i \in N_1$$

$$\sum_{i:(i,j)\in A} X_{ij} = -b_j \quad \text{for all } j \in N_2$$

$$x_{ij} \geq 0, \text{ for all } (i, j) \in A$$

CHAPTER FOUR

NETWORK SIMPLEX METHOD

4.1 An algorithm for network simplex method

The network simplex algorithm is the simplex algorithm. Simply we can compute the primal solution, the simplex multipliers and reduced costs directly on the network.

- ✓ The Bases correspond to spanning trees
- ✓ The basic feasible solution is found by sending flow in arcs
- ✓ The reduced costs are found by finding the simplex multipliers explicitly.
- ✓ The leaving arc is found by sending flow around a cycle.

Feasible solution and Optimality condition

To find the optimal solution of minimum cost flow problem; first we must have a network with n nodes which is a spanning tree. In order to show that the variables corresponding to the arcs in the tree constitute a basis, it is sufficient to show that the (n – 1) tree variables are uniquely determined. In the simplex method, this corresponds to setting the non-basic arcs to specific values and uniquely determining the basic variables.

Therefore, (n – 1) arcs must exist in the original network with n nodes. Next, we demonstrate that a sub network connected to n nodes is a spanning tree if it has (n – 1) arcs and no loops.

A base for the Simplex method and a network's spanning tree match in a minimum-cost flow model. The spanning tree does not have any loops, hence there must be at least two endpoints. We do not yet know if this workable approach is the best one. Therefore, we should use reduced cost optimality, negative cycle optimality, or complementary slackness optimality requirements to determine whether a basis feasible solution is optimal. In this project, I attempt to determine if a feasible solution is optimal or not by applying the reduced cost optimality criterion to my illustrative scenario. As a result, we should first determine the node potentials for basis arcs and use these node potentials to compute decreased costs for all non-basic arcs.

Definition: An edge (i,j) has a decreased cost of (i, j) is $r_{ij} = c_{ij} + z_i - z_j$. The cost of purchasing a widget at i, transporting it to j, and selling it there can be compared to the decreased cost. Keep in mind that we would not transport the item from i to j if r is positive. If there isn't a residual edge with a negative decreased cost, then a pricing function for a residual graph is viable. A

viable solution x* is an optimal solution for the least cost flow issue under the following assumptions, where z is the optimal solution for the dual problem:

Reduced Cost Optimality Conditions:

A workable resolution the minimal cost flow issue has an optimum solution x^* only when a certain set of node potentials z meets the following reduced-cost optimality requirements: For each arc (i, j) in $G(x^*)$, $r_{ij} \geq 0$.

Assume that we assign a real integer (N), with no restrictions on its sign, to every node within N. We denote the potential of node i as z_i. The dual variable for linear programming that corresponds to node i's mass balance constraint is z_i. We define the reduced cost of an arc as (i, j) as $r_{ij} = c_{ij} - z_i + z_j$.for a given set of node potentials z_i.Thus, under reduced cost optimality conditions, a viable solution x* is an optimal solution of the minimum cost flow issue if and only if a certain set of node potentials z meet the requirements for reduced cost optimality, that is, $r_{ij} > 0 \; for \; every \; arc \; (i, j)$ the non-basis arcs.

Complementary Slackness Optimality Conditions

A feasible solution x^* is optimal if and only if for some set of node potentials z, the reduced costs r_{ij} and flow values satisfy the following complementary slackness optimality conditions for every arc (i, j) ∈ A

1. $r_{ij} > 0$, if $x_{ij} = 0$. for non-basic arcs(i,j)
2. $r_{ij} = 0$, If $0 < x_{ij}^* < u_{ij}$
3. $r_{ij} < 0$, if $x^*_{ij} = u_{ij}$

If it satisfies the optimality condition the current basic feasible solution is optimal. If r_{ij} has some negative components, we can, in principle, lower the cost by letting those components of non-basic arc X_N become positive. Choose an arc (i,j) with negative reduced cost to send as much flow as possible around the basic cycle and adjust the new spanning tree.

4.2. Application of NSM to solve MCNFP

The network simplex method maintains a feasible spanning tree structure at each iteration and successively transforms it into an improved spanning tree structure until it becomes optimal. **Initialization :** In this step our first task is determining the initial basic feasible spanning tree. There are different strategies that, we can apply to find the initial basic spanning tree for a given network flow problem. Connectedness assumption and two phase method are some of these strategies. For the given problem, we can identify a finite number of spanning trees under the connectedness assumption. We can use one of these as the starting basic spanning tree and keep

running the algorithm until we find the best answer. In contrast, each node in the two-procedure technique introduces an artificial arc with a unit cost in relation to the root node, r.

Find the basic solution for the spanning tree using flow balance equation $A*X_B=b*$. If all flows are non-negative $X_B \geq 0$,then the spanning tree flow is basic feasible solution.

Main Iterative steps

- The main step start with the basic feasible solution .To check optimality
- First compute the simplex multipliers (node potential z_i) for the arc of the basis where $c_{ij} = z_i - z_j$
- Secondly calculate the reduce cost (r_{ij}) by using the simplex multipliers. then
- Check Complementary Slackness optimality condition

The current fundamental viable solution is optimal if it meets the optimality criteria.

In theory, we can reduce the cost if r_{ij} contains any negative components by allowing those components of the non-basic arc X_N to become positive.

To send as much flow around the basic cycle as possible, choose an arc (i, j) with negative reduced cost, then modify the new spanning tree.

Identify the entering arc and Testing optimality

If all $x_{ij} \geq 0$, for each (i, j) elements of initial basis spanning tree, it is better to check that whether the spanning tree structure satisfies the optimality conditions. At this step we must find all node potentials/simplex multiplier/ of all nodes in the spanning tree and reduced cost of all non-basis arcs. The reduced cost, r_{ij} for a non-basic arc (i, j) is the cost of increasing the flow on that arc by one unit, and can be expressed as: $r_{ij} = c_{ij} - z_i + z_j$. If this spanning tree structure satisfies the optimality conditions (if the reduced costs of all non-basis arcs are non-negative) then, it is optimal and the algorithm terminates. That is the solution is optimal, if one of the following conditions must hold for each non-basic arc (i, j): if $x_{ij}= 0$, then $r_{ij} > 0$, if $x_{ij} = u_{ij}$ then $r_{ij} < 0$. If not the algorithm selects a non-basic tree arc that mostly violating the optimality condition to introduce into the new tree. Many different rules, called pivot rules, are possible for choosing the entering arc.

For every unit change in the flow value on the chosen arc, the arc with the biggest violation results in the greatest decrease in the goal function. Therefore, if the average rise in the selected arc's value were the same for all arcs, adding this arc to the spanning tree would result in the largest drop per pivot. The algorithm runs fewer iterations than alternative choices for the pivot rule, according to computational studies that validate this choice of the entering arc tends to yield reasonably big drops in the objective function each iteration.

Leaving arc and cycle rule

To create a new spanning tree, we must first identify the leaving arc and then the intersecting arc. Some of the prior basis arcs form a cycle with the arc entering the spanning tree. To determine the departing arc in this cycle, we must select a leaving arc with the lowest flow of all such arcs and one that satisfies a flow in the opposite direction as the entering arc.

Update Flows

Let the arc's flow out of the spanning tree be represented by $\mathbf{X}_{min}$. Keep in mind that there is a cycle if there are entering and exiting arcs. This only applies to the arcs in this cycle: deduct $\mathbf{X}_{min}$ from the flows corresponding to the arcs that are flowing in the same direction as the leaving arc, and add $\mathbf{X}_{min}$ to the arcs whose flow is opposite the leaving arc.

Update the simplex multipliers.

Here, we can allow $z_i = 0$ for each node in the spanning tree in order to compute z_1, z_2, z_3,...., z_n (sometimes z termed the simplex multipliers / node potential / dual variable). principally z_n or $z_1 = 0$. Reduce the dual in the non-root tree by the incoming arc's cost variable if the entering arc connects the non-root sub tree to the root sub tree. Stopping Criterion .

We find a solution to our problem if the present extreme point is in fact optimal and all components of the lowered cost of the non-basis arc turn out to be non-negative ($r_{ij} > 0$). As a result, the procedure ends.

No solution

Even if we can lower and lower the cost, (some components of $r \leq 0$) and none of the components in X_b will ever reach zero (on choice as leaving arc). The problem does not admit an optimal solution because we can reduce the cost indefinitely.

Example 4.1: Let us consider following network flow problem/graph /that minimizes the total cost for trafficking, by using the network simplex algorithm.

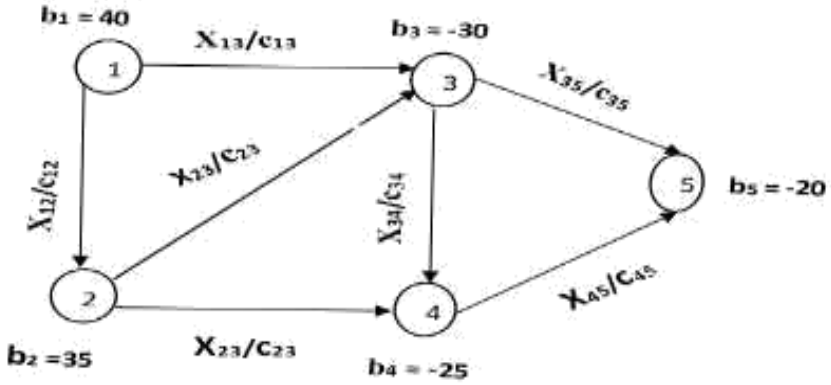


Fig 4.1 Network flow problem graph

For this graph / network flow problem /

Among 5 nodes and 7 arcs, node 1 and node 2 are supply nodes(since $b_1, b_2 > 0$) others node in the network (node 3,4,5) are demand nodes (since $b_3, b_4, b_5 < 0$)

The cost for traffic in the link between node i and j is represented by c_{ij}.

The objective of the problem is to minimize the total cost.

The problem is equilibrium because Total supply = total demand

Representation of network flow problem by incidence matrix

we must recall that,

$$\mathbf{a_{ij}} = \begin{cases} 1, & \text{if an arc e starts from node i and ends at j} \\ -1, & \text{if an arc e starts at node j and ending at node i ,} \\ 0, & \text{otherwise} \end{cases}$$

for ij^{th} entry of the incidence matrix A

Therefore, the incident matrix representation of the network given above in fig 4.1 is given as follows:

$$A = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \begin{array}{ccccccc} x_{12} & x_{13} & x_{23} & x_{24} & x_{34} & x_{35} & x_{45} \end{array} \\ \begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & -1 & -1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & -1 & -1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & -1 & -1 \end{pmatrix} \end{array}$$

Cast as LPP from the network optimization problem

N = set of nodes.

B = set of arcs.

N = {1, 2, 3, 4, 5}

B = {(x_{12}, x_{13}, x_{23}, x_{24}, x_{34}, x_{35}, x_{45})} = {(1,2), (1,3), (2,3), (2,4), (3,4), (3,5), (4,5)}.And Let C_{ij} = {c_{12}, c_{13}, c_{23}, c_{24}, c_{34}, c_{35}, c_{45}} = {2, 5, 2, 2, 1, 1, 2}

b_i = {b_1, b_2, b_3, b_4, b_5} = {40, 35, -30, -25, -20}.

[X] =[x_{12},x_{13},x_{23},x_{24},x_{34},x_{35},x_{45}]T, A= the incident matrix above , [b] = [b_1,b_2,b_3,b_4,b_5]

Finding the best flows throughout the network's arcs to minimize overall costs while maintaining flow conservation at each node is the goal. Decision variables, or the number of flows, will be used to combine the objective function and the constraints. Hence, the Network Optimization Problem cast as LPP.

AX=b

Minimize $\sum c_{ij}x_{ij}$

S.t $x_{12}+x_{13} = b_1 = 40$

$-x_{12}+x_{23} + x_{24} = b_2 = 35$

$-x_{13} - x_{23} + x_{34} + x_{35} = b_3 = -30$

$-x_{24} - x_{34} + x_{45} = b_4 = -25$

$- x_{35} - x_{45} = b_5 = -20$

$x_{ij} \geq 0$, for all flows in the networks.

Since, n = 5, there are n - 1 = 5 - 1 = 4 arcs in a basis feasible solution.

Some of the spanning trees of the network given by fig4.3 above are: -

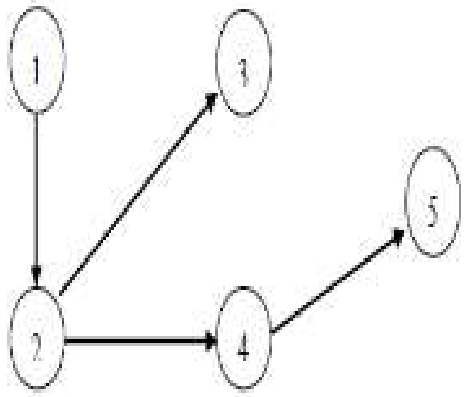


Fig4.2(a)

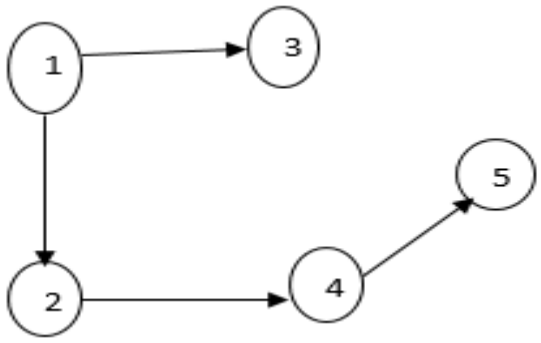


fig4.2(b)

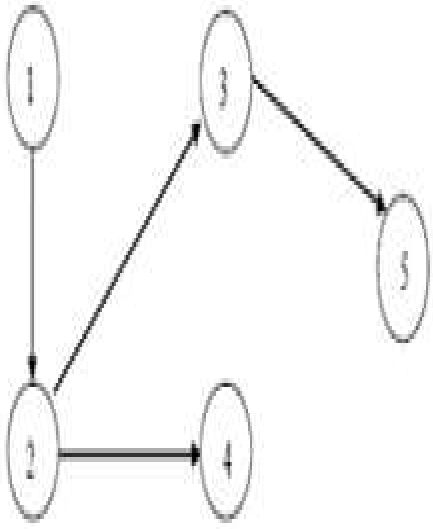


Fig4.2(c)

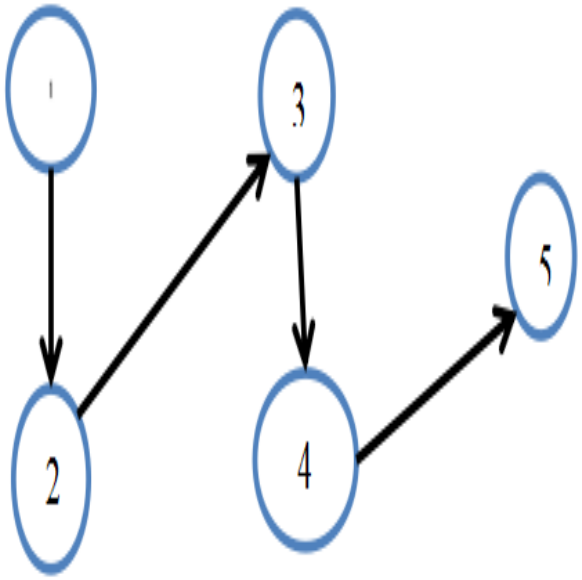
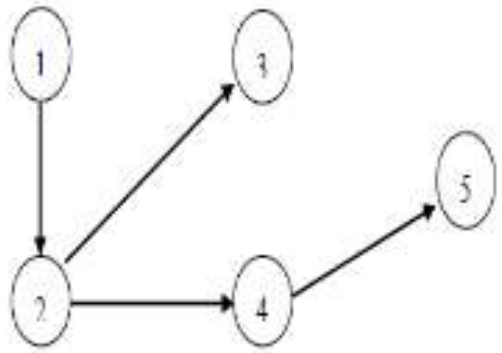


fig4.2(d)

Fig 4.2 spanning trees

First Iteration

Let take fig4.2(a) spanning tree and apply network simplex method on it.



In this case,

N = {1, 2, 3, 4, 5}, A_B = {(1, 2), (2, 3), (2, 4), (4, 5)} are basic arcs

$A_{N=}$ {(1, 3), (3, 4),(3, 5)} are the non-basic arcs.

C_{ij} = {2, 2, 2, 2}

b_i = {b_1, b_2, b_3, b_4, b_5} = {40 , 35, -30, -25, -20}

The basic incident matrix is given by:

$$A_B = \begin{bmatrix} 1 & 0 & 0 & 0 \\ -1 & 1 & 1 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & 1 \end{bmatrix}$$

Now determining feasible basic solution of the given spanning tree

$A_B X_B = b$ ⟹

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ -1 & 1 & 1 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & 1 \end{pmatrix} \begin{pmatrix} x12 \\ x23 \\ x24 \\ x45 \end{pmatrix} = \begin{pmatrix} 40 \\ 35 \\ -30 \\ -25 \end{pmatrix}$$

Which is,

$X_{12}=40$

$-X_{12}+X_{23}+X_{24}=35$

$-X_{23}=-30$

$-X_{24}+X_{45}=-25$,By solving , $X_{12} = 40$, $X_{23} = 30$, $X_{24} = 45$, $X_{45} = 20$

$X_B = \{X_{12}, X_{23}, X_{24}, X_{45}\} = \{40, 30, 45, 20\}$ which are all non-negative. Hence X_B is feasible basic solution. The total cost associated with this initial basis feasible solution will be,

$\sum C_{ij}X_{ij} = C_{12}X_{12}+C_{23}X_{23}+C_{24}X_{24}+C_{45}X_{45} = 2*40+2*30+2*45+2*20 = 270.$

To check is this optimal ?

Now let us find simplex multipliers/dual variable (z) for all basic arcs and reduced costs r_{ij} for all non-basic arcs, based on $\{c_{12}, c_{13}, c_{23}, c_{24}, c_{34}, c_{35}, c_{45}\} = \{2, 5, 2, 2, 1, 1, 2\}$

The node potential of each basic arc {(1, 2), (2, 3), (2, 4), (4, 5)} is given as follows,

Assume $z_n = 0$, n = 5, hence $z_5 = 0$

$z_i - z_j = c_{ij}$ for all $(i, j) \in A_B$

$z_4 - z_5 = c_{45}$	$z_2 - z_4 = c_{24}$	$z_2 - z_3 = c_{23}$	$z_1 - z_2 = c_{12}$
$z_4 - 0 = 2$	$z_2 - 2 = 2$	$4 - z_3 = 2$	$z_1 - 4 = 2$
$z_4 = 2$	$z_2 = 4$	$z_3 = 2$	$z_1 = 6$

Determine reduced cost of each non-basic arc

$r_{ij} = c_{ij} - z_i + z_j$, for non-basic arcs, $A_N = \{(1, 3), (3, 4), (3, 5)\}$

The corresponding non basic matrix is :-

$$A_N = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ -1 & 1 & 1 \\ 0 & -1 & 0 \end{bmatrix}$$

$r_{13} = c_{13} - z_1 + z_3 = 5 - 6 + 2 = 1$

$r_{34} = c_{34} - z_3 + z_4 = 1 - 2 + 2 = 1$

$r_{35} = c_{35} - z_3 + z_5 = 1 - 2 + 0 = -1.$

- The reduced cost $r_{35} < 0$, (for non-basic arc (3,5)), which violates the optimality condition. Therefore, the initial basic feasible solution(X_B) is not optimal.

Second iteration

Determining the Entering arc

- The arc which has least reduced cost inters to the basis. Hence, arc (3, 5) is the entering arc.

Or by applying the complete slackness condition, $z_i - z_j - c_{ij} = 0$ where (i, j) is non-basic, we can determine the entering arc. (The arc which has highest /higher/ $z_i - z_j - c_{ij}$ enters the basis).

Therefore,Arc (3, 5) has highest value of $z_i - z_j - c_{ij}$, arc (3, 5) is entering arc.

Determine the leaving arc

Which arc is leaving from the basis tree?

The newly entered basic arc and some of the previous basis arc form a cycle in the direction of the entering arc. From this cycle, the previous basic arc that has opposite directional flow with

minimum amount of flow leaves the basis. Here for the above example the previous basic arcs (2, 3), (2, 4), (4, 5) and the newly entered arc (3, 5) form a cycle and arcs (2, 4) and (4,5) have opposite direction to the cycle with x_{24} = 45 and x_{45} = 20. Since arc (4, 5) has less amount of flow than arc (2, 4), arc (4,5) is leaving arc.(Let w = θ)

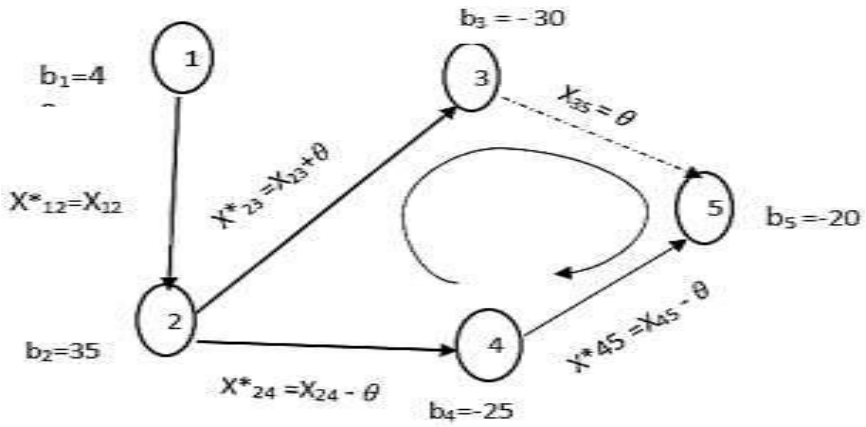


Fig 4.3 graph of Augmenting cycle

$X^*_{23} = X_{23}+w= 30 +w$

$X^*_{24}=X_{24}–w=45– w$

$X^*_{45} = X_{45} –w = 20 – w$ (where x^*_{ij} is flow the new basis arcs (i,j) where X_{ij} is flow of the old basic arc). $W_{max}=20$ when $X_{45}= 0$

The augmented w amount of flow can be increased on arcs (2, 3) and (3, 5) and decreases on arcs (2, 4) and (4, 5) until it reaches to 20. That is until x_{45} = 0. Then x_{45} exits/leaves/ the basis. Hence arcs (1, 2) (2, 3), (2, 4) and (3, 5) are now the new basis. (i.e. $X^*_B = \{x_{12,} x_{23}, x_{24}, x_{35}\}$).

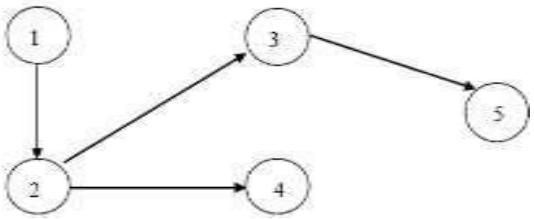


Fig 4.4 New spanning tree

The incident matrix form of these basic arcs(new spanning tree) is A*

$A^*X^*_B= b$, We get $X^*_{12} = 40$

$-X^*_{12}+X^*_{23}+X^*_{24} = 35$

$-X^*_{23}+X^*_{35}= -30$

$-X^*_{24}= -25$

Resulting $X^*_B =$ [40,50,25,20] all non negative, which is new basic feasible solution.

For basic arcs {(1, 2), (2, 3), (2, 4) (3, 5)}, $c_{12} = 2$, $c_{23} = 2$, $c_{24} = 2$, $c_{35} = 1$

$\sum C_{ij}X_{ij} = c_{12}x_{12} + c_{23}x_{23} + c_{24}x_{24} + c_{35}x_{35} = 2*40 + 2*50 + 2*30 + 1*20 = 260$

Now the total cost decreases by 270 – 260 = 10 units.

Is this basis solution optimal? **check optimality**

Determining simplex multipliers.($z_i - z_j = c_{ij}$) for all (i,j)$\in A^*_B$

Let $z_n = 0$, n = 5, hence $z_5 = 0$

$z_3 - z_5 = c_{35}$	$z_2 - z_4 = c_{24}$	$z_2 - z_3 = c_{23}$	$z_1 - z_2 = c_{12}$
$z_3-0=1$	$3-z_4=2$	$z_2-1=2$	$z_1-3=2$
$z_3=1$	$z_4=1$	$z_2=3$	$z_1= 5$

Determine the reduced cost for non-basic arcs

For non-basic arcs (1, 3), (3, 4) and (4, 5) with $c_{13} = 5$, $c_{34} = 1$ and $c_{45}= 2$,

The reduced costs are: -

$r_{13} = c_{13} - z_1 + z_3 = 5-5+1 = 1$

$r_{34} = c_{34} - z_3 + z_4 = 1-1+1 = 1$

$r_{45} = c_{45} - z_4 + z_5 = 2-1+0 = 1$

All $r_{ij} > 0$, for all (i, j)$\in$ non basic arcs. This tells us optimality condition is satisfied. Therefore, the basic feasible solution X^*_B is optimal. Hence the iteration is stop.

CONCLUSICON

I derive the following conclusions from the discussion above.

Networks are a significant subclass of linear programs that are helpful for simulating business challenges and are simple to understand and solve. Even in situations when there are extra variables or restrictions that make it impossible to model the entire problem using networks, networks offer a helpful framework for problem-solving. With a long history, the Minimum Cost Flow Problem provides a framework for the formulation and solution of several theoretical and real-world issues. The MCF problem has numerous applications and is essential to network flow theory.This project describes how to solve the minimal cost network flow problem using an implementation of the network simplex algorithm. The network problem displaying complete computations on an illustrative case has been solved using the NSM dual viable solution in order to determine the optimal flows. Ultimately, the findings demonstrate that NSM can be helpful in solving network flows and identifying the best solution when the network size is moderate.

BIBILOGRAPHY

[1] Moktar.S Bazarra, John J. Jarvis, and Hanif. D.Sherali (1977) Linear Programming and Network Flows (John Wiley & Sons Inc.publication) Fourth Edition-Wiley-Interscience (2009)

[2] R. K. Ahuja, T. L. Magnanti and J. B. Orlin, "Network Flows: Theory, Algorithms and Applications," Prentice Hall, Englewood Clis, New Jersey, U.S.A., 1993.

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

[4] Damian J. Kelly, B. Comm and Garrett M. O'Neill, B. Comm at University of Ireland [5] R. Barr, F. Glover and D. Klingman, Enhancements to spanning tree labeling procedures for network optimization, INFOR, 17 (1979), 16–34.

[6] E. Cohen and N. Megiddo, Algorithms and complexity analysis for some flow problems, and Algorithmic

[7] Hans JakobLiithi system modeling and optimization theory of linear programming.

[8] Michel Gondran , Michel Minoux (1984) GRAPHS AND ALGORITHMS (John Wiley & Sons).

[9] PabloPedregal, introduction to optimization.

[10] EUGENE L. LAWLER (1976) Combinatorial Optimization: Networks and Matroids (Holt, Rinehart and Winston

[11] Ann-Brith Stromberg lecture note on shortest paths and network flow models

[12] Cunningham, W.H. (1979) 'Theoretical Properties of the Network Simplex Method' Mathematics of Operations Research Volume 4 p196-208.