Maximum Flow Minimum Cost Problem In A Network

COLLEGE OF COMPUTATIONAL AND NATURAL SCIENCES
DEPARTMENT Of MATHEMATICS
STREAM:OPTIMIZATION

A Project Submitted "In Partial Fulfillment Of The Requirement Of The Degree Of Master Of Science In Mathematics"

BY:

Tsadkan Tsegay

Advisor:

Berhanu Guta(PhD)

June 2014
Addis Ababa, Ethiopia

ADDIS ABABA UNIVERSITY
DEPARTMENT OF MATHEMATICS

The undersigned hereby certify that they have read and recommend to the department of Mathematics for acceptance of this project entitled **Maximum Flow Minimum Cost Problem In A Network** by **Tsadkan Tsegay** in partial fulfillment of the requirements for the degree of Master of Science in Mathematics.
Advisor: Dr.Berhanu Guta(PhD)
Signature:_____

Date_____

Examiner 1: Dr. _____
Signature:_____

Date_____

Examiner 2: Dr. _____
Signature:_____

Date_____

Contents

List of Notations and Abbreviations

- ✓ C_{ij} = Cost from node i to node j
- ✓ X_{ij} = Flow from node i to node j
- ✓ U_{ij} = Capacity from node i to node j
- ✓ G_X = Residual network
- ✓ $MFMC$ = Maximum flow minimum cost
- ✓ $[S, \bar{S}]$ = $s-t$ cut
- ✓ $G(N, A)$ = Directed network with N nodes and A directed arcs
- ✓ FAP = Flow augmenting path

Acknowledgment

I would like to thank Dr.Berhanu Guta(PhD) for his helpful comments and constructive suggestions for this project.Next my thank goes to Addis Ababa University Female Scholarship for its financial support in attending the degree of masters of science.More over I would like to express my special thanks to the department of Mathematics for which I am able to find enough materials like internet access,printing materials and others.

Abstract

Maximum flow minimum cost(MFMC) problem combines sending as much flow as possible from the source node to the sink node with minimum cost.Here the attributes for the network are the capacity,flow and cost per unit flow of an arc which varies linearly with the amount of flow.The problem has two special problems namely the shortest path problem and maximum flow problem for which we set all the capacities and cost are set to be zero respectively.Solving maximum flow minimum cost involves two steps;finding the maximum flow and then find the minimum cost using the feasible flow as an input.

Keywords:Maximum-Flow Minimum Cost,$s-t$ cut,Ford Fulkerson labeling algorithm,Network simplex algorithm

Introduction

This project aims at obtaining maximum flow from the source node (origin) to the sink (destination) node with in a minimum cost.This type of problem is special in the case that it is a combination of two types of problem where the first one is as in put for the second problem.In other words it first finds the maximum flow say X^* and then using $b(s) = X^*$,$b(t) = -X^*$ it finds the minimum cost flow in the given network problem.This type of problem occurs in many aspects of the real world including in computer networking,telecommunications,transportation and any others.

Several solution procedures are applied to this problem in different literatures.

Nazimuddin Ahmed & S. Das & S. Purusotham [8]solved the maximum flow minimum cost problem using the lexicographic search technique.Pandit's Max-Min algorithmic approach is proposed to solve the maximum flow with minimum cost[7].But in this project solving the problem involves two main steps.The first one is to find the maximum flow using the capacity and flow as an input.This can be done using different ways like identifying the flow augmenting path,finding an $s-t$ cut having with the minimum capacity,Ford Fulkerson labeling algorithm and others.

The second step is to solve the minimum cost flow problem by setting the supply at the source node s to be the maximum flow value and the demand at the sink node t to be the negative of the flow value which is obtained in step one.Here all nodes other than the source and sink nodes are transshipment nodes.

The first chapter deals with some terms and illustrative examples for the Maximum flow Minimum Cost problem.Chapter 2 deals with the background theories for solving the problem.In addition it attempts to deal with the solution methods and optimality conditions.This chapter is the key for solving the maximum flow minimum cost problem.Chapter 3 attempts to elaborate about the solution procedures for the Maximum Flow Minimum Cost Problem together with an example.

Chapter 1

Introduction And Some Terminologies For Maximum Flow Minimum Cost Problem

1.1 Introduction

The *Maximum Flow Minimum Cost* (MFMC) problem is defined on a directed network $G(N, A)$ where by we wish to send the maximum amount of flow from a source node s to a sink node t at the minimum possible total cost.Here N represents the set of nodes and A represents the set of arcs.The attributes for the network are the capacity U_{ij},cost C_{ij},flow X_{ij} of an edge and we will discuss them in detail in the next section. Graphically,

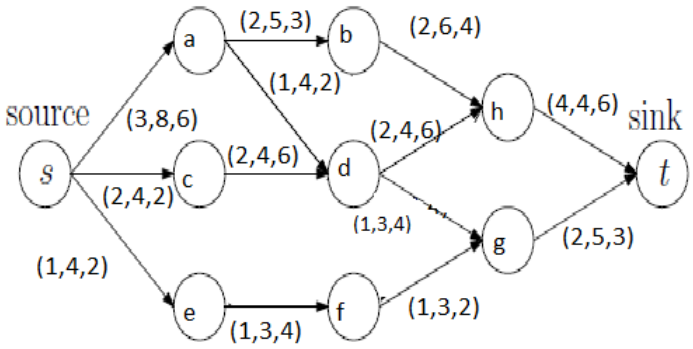


Figure 1.1 (X_{ij}, U_{ij}, C_{ij}) of each edge represents the flow, the capacity and the cost

The maximum flow minimum cost problem can be specialized in to two problems.

- If all the costs are set to be zero,then the problem is reduced in to the maximum flow problem.

- If all the capacities are set to be zero,then the problem is reduced in to the shortest path problem.

A network can be used to model traffic in a road system,fluids in pipes,currents in an electrical circuit,or anything similar in which something travels through a network of nodes. Network flow problems form an important class of optimization problems on graphs with applications to several areas including computer networking,public policy,scheduling,telecommunications,transportation.For example a company wants to transport(distribute) commodities or products from the the source(the place where by the commodities are produced) to the sink(the place where by the products are consumed).In this case the company wants as much flow as possible to fulfill the demands of customers as well as to send the products using the most minimum cost.A network consists of a set of points and a set of lines connecting certain pairs of the points.
The points are called **nodes** or vertices and the lines are called **arcs**.The arcs may have a direction on them,in which case they are called *directed arcs*.In other words an arc is said to be *directed* or *oriented* if it allows positive flow in one direction and zero flow in the opposite direction.If an arc has no direction,it is often called a *link* or *undirected*.If all the arcs in a network are directed,the network is a directed network otherwise undirected network [2]. Each node represents a location or city and each arc represents the connection or road link between two different locations (cities).The number on each arc represents the distance or cost,time between the two locations.
The following table shows some typical associations for the nodes, arcs, and flows in a variety of physical networks which is taken from [5].

Nodes	Arcs	Flow
intersections	roads	vehicles
airports	air lanes	aircraft
switching points	wires,channels	messages
pumping stations	pipes	fluids
work centers	material handling routes	jobs

Important subclasses of network flow problems are the multi commodity flow problems, generalized flow problems (having flows with losses and gains),minimum cost flow problems,maximum flow problems, and **maximum flow minimum cost problem** which is the main focus of this project.
Multi commodity flow problems involves physical commodities like vehicles,messages each governed by their own network flow constraints share the same network.In other words in multi commodity flow problems each individual commodities share common arc capacities.That is each arc has a capacity U_{ij} that restricts the total flow of all commodities on that arc. In this project work we have considered,the objective function was separable in the sense that the different flow

variables X_{ij} appeared in separate terms $C_{ij} \times X_{ij}$.But in case of convex cost flow problems this property is not applied but we permit the separable terms to be nonlinear functions of the form $C_{ij}(X_{ij})$.More over Each function $C_{ij}(X_{ij})$ is convex that is for any $\lambda \in [0,1]$ X_{ij} and $Y_{ij}$are any two points within the flow bounds of X_{ij} ,then

$$C_{ij}(\lambda X_{ij} + (1-\lambda)Y_{ij}) \leq \lambda C_{ij}(X_{ij}) + (1-\lambda)C_{ij}(Y_{ij})$$

. In the *generalized network flow problems* there is a positive multiplier β_{ij} associated with each arc (i,j) such that if a flow X_{ij} units enter an arc (i,j) at node i,then by the time it reaches node j,the flow is $\beta_{ij} \times X_{ij}$ units.If $0 < \beta_{ij} < 1$,then arc (i,j) is *lossy* and if $1 < \beta_{ij} < \infty$,it is said to be *gainy*.In other words here in the case of the generalized network flow problem flow conservation is violated.

Example 1. *Transmission of a volatile gas,we might lose flow because of evaporation,in the transmission of liquids such as raw petroleum crude,we might lose flow due to leakage.In this case the flow is* lossy.

Example 2. *If we transmit money from one period to the next by holding it in a bank account,the amount reaching the destination will be more than the amount than left in the origin.This is due to the interest earned and the flow is* gainy.

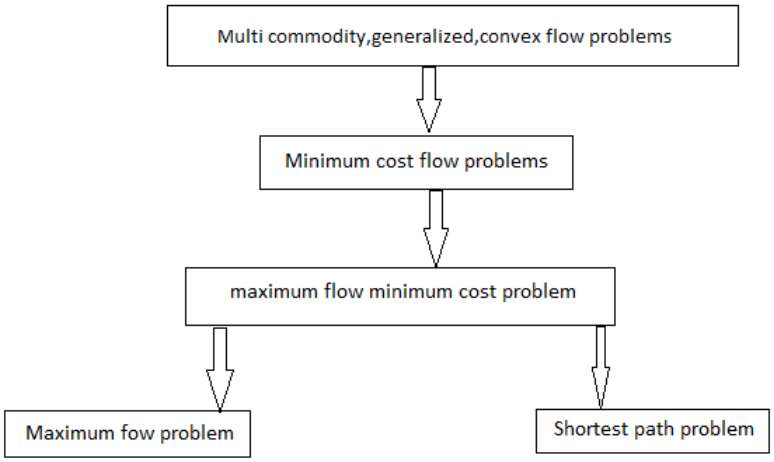


Figure 1.2

If all arc flows has $\beta_{ij} = 1$,then the resulting network is known as pure network flow and flow problems on them have been called pure network flow problems like pure minimum cost flow,pure maximum flow problems which are the main focus of this project.That is we are going to see for network flow problems having $\beta_{ij} = 1$.

1.2 Some Terminologies And Examples

Although there are several terms and concepts to deal with,here are some of them which are helpful in this project.

- Graphs:They are sets of vertices (nodes) connected by set of arcs.
- Connected graph: A graph is said to be connected if there is a path between every pair of its nodes.
- *Network flow*:is a directed graph where each edge has a capacity and each arc (directed edge) receives a flow. In a network flow problem,we assign a flow to each arc.Flow is related to people or materials over a transportation networks,electricity over electrical distribution system where by the flow conservation holds.Flow is simply the rate a material;can be fluid in a pipelines,cars or any thing else moves through the network.In general,the flow in a network is limited by the capacity of its arcs that is can not exceed the capacity of an arc,which may be finite or infinite. In other words a flow can be stated as follows.
  A *flow* is a mapping $X : A \to \mathbb{R}^+$ such that the following two conditions are satisfied:
  Capacity constraint

$$0 \leq x_{ij} \leq U_{ij} \tag{1.1}$$

  and

$$Flow \;\; conservation \sum_{\{j:(i,j)\in A\}} X_{ij} = \sum_{\{j:(j,i)\in A\}} X_{ij} \;\; \forall i \backslash \{s,t\} \tag{1.2}$$

  for a given $G(N, A)$ directed graph and $i \in N$.
  Consider the following (Figure 1.2)directed graph $G(N, A)$;

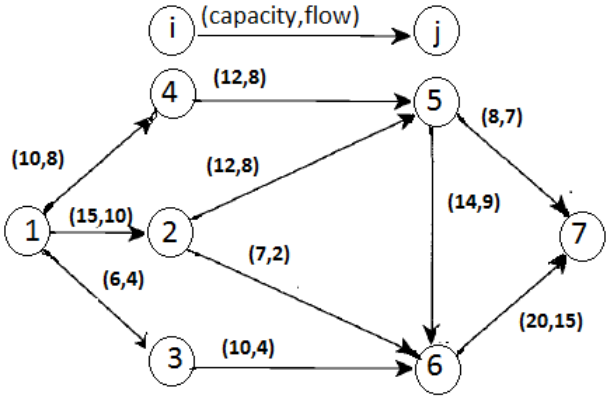


Figure 1.3

For all $(i, j) \in A$ both 1.1 and1.2 holds.

- *Cost*(C_{ij}):cost that must be paid per unit of flow that goes through an edge.The cost of a flow is the sum over all edges of the cost of the edge times the value that the flow assigns to that edge that is ;
  Total cost= $\sum_{(i,j)\in A} X_{ij} \times C_{ij}$
- *Capacity Function*:capacity U of an edge is a mapping $U : E \to \mathbb{R}^+$ denoted by U_{ij} which represents the maximum amount of flow(possibly infinity) that can pass through an edge.The capacity U_{ij} limits the amount of flow we are permitted to send into arc (i, j) .

- A *path*:is a sequence of distinct arcs. A path that begins and ends at the same node is a *cycle* and may be either directed or undirected.
- A cut represented by $[S, \bar{S}]$ is a set of arcs whose deletion disconnects the network into two components S and $\bar{S}$ and no subset of it has this property.
- Let $T = (N, A)$ be a graph.T is said to be a tree if and only if it is a connected graph and contains no cycle.
- For $G = (N, A)$ is a connected graph,a spanning subgraph of G which is a tree is called spanning tree of G.
- Given $G = (V, E)$ which is a connected weighted graph.A spanning tree of G which has the minimum weight is called minimal spanning tree of G.

1.3 Residual Network

The residual network consists of arcs that can admit more flow.The residual network G_X corresponding to a feasible flow X_{ij} is defined as follows:
We replace each arc $(i, j) \in A$ by two arcs (i, j) and (j, i).

- For arcs having only the capacity U_{ij},and flow X_{ij} i.e in case of the maximum flow problems; Arc (i, j) has residual capacity of $r_{ij} = U_{ij} - X_{ij}$ and arc (j, i) has residual capacity of $r_{ij} = X_{ij}$.
- Arcs having cost C_{ij} for each edge the arcs in the residual network can have the following values.
  Arc (i, j) has cost C_{ij} and a residual capacity $r_{ij} = U_{ij} - X_{ij}$, and arc (j, i) has cost C_{ji}=-C_{ij} and a residual capacity r_{ij}= X_{ij}.

The only difference between the two types in the above is;in the second one for all edges there is an associated cost C_{ij}.The residual network consists only of arcs with positive residual capacities.The residual network, denoted by G_x, consists of all arcs whose residual capacity is strictly greater than 0.Thus,for any arc $(i, j) \in A$, the residual network may contain arc (i, j) or (j, i) or both.Lets consider the following notation to describe residual network graphically.

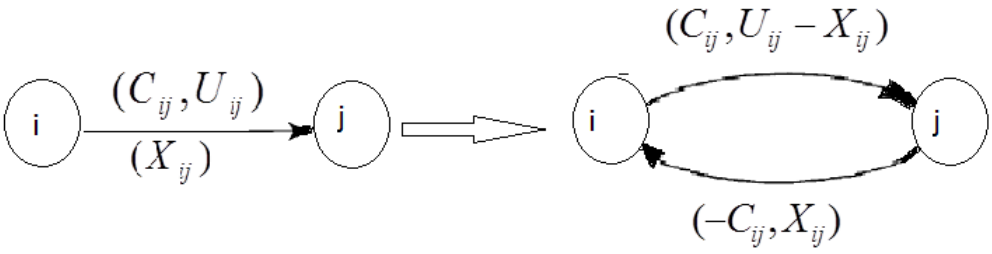


Figure 1.4

Example 3.

If the flow,capacity &*cost* is given as $X_{ij} = 5, U_{ij} = 10 \& C_{ij} = 6$,then the arcs in the residual network will have the following residual capacity and cost.
Arc (i, j) have residual capacity $r_{ij} = U_{ij} - X_{ij} = 5$ & *cost* $C_{ij} = 6$
Arc (j, i) have residual capacity $r_{ji} = X_{ij} = 5$ &*cost* $C_{ji} = -C_{ij} = -6$
But if $U_{ij} = 10, C_{ij} = 6$ and $X_{ij} = 10$,then then we have only to take only the backward arc (j, i) with residual capacity of $r_{ji} = X_{ij} = 10$ and cost of $C_{ji} = -C_{ij} = -6$
Reducing the flow in (i, j) by one unit is equivalent to sending one unit of flow from j to i.It reduces the cost by C_{ij}.The idea is that,in the residual network the capacity of an edge measures *how much more flow* can be pushed along that edge in addition to the flow X_{ij},with out violating the capacity constraints.

Chapter 2

Back Ground Theories And Solution Methods

2.1 Maximum flow problem

Maximum flow problem is to send as much flow as possible between two special nodes, a source node s and a sink node t.All the remaining nodes are intermediate or transshipment nodes.Flow through an arc is allowed only in the direction indicated by the arrow head and the flow in each arc $(i, j) \in A$ is not allowed to be more than a given positive integral value upper bound the capacity U_{ij}. To be precise, in an instance of the maximum flow problem we are given a directed graph $G = (N, A)$, and two specified nodes s and t which are assumed to have no incoming and outgoing arcs,respectively. With each arc $(i, j) \in A$ is associated a positive integral number U_{ij} its capacity.
If V denotes the amount of material sent from the source node s to the sink node t or is the value of the flow,the maximum flow problem can be formulated as follows:

$$\max V$$

subject to;

$$\sum_{\{j:(i,j)\in A\}} X_{ij} - \sum_{\{j:(j,i)\in A\}} X_{ij} = \begin{cases} V & \text{if } i = s \\ 0 & \text{if } \forall i \in N\backslash\{s,t\} \\ -V & \text{if } i = t \end{cases}$$

Capacity Constraint(flow bound constraint)

$$0 \leq X_{ij} \leq U_{ij} \tag{2.1}$$

Some times the flow vector X_{ij} might be required to satisfy lower bound constraints imposed up on the arc flows;i.e if $l_{ij} > 0$ specifies the lower bound on the flow on arc $(i, j) \in A$ we

imposed the condition $X_{ij} \geq l_{ij}$ and this problem is maximum flow problem with non negative lower bounds.Here it is possible to transform a maximum flow problem with non negative lower bounds in to a maximum flow problem with zero lower bound as;

$$l_{ij} \leq X_{ij} \leq u_{ij}$$

The above inequality can be transformed in to the maximum flow problem with 0 lower bound by subtracting the l_{ij} from both sides as follows;
$l_{ij} - l_{ij} \leq X_{ij} - l_{ij} \leq U_{ij} - l_{ij}$ which is the same as;
$0 \leq \bar{X_{ij}} \leq \bar{U_{ij}}$ for $\bar{X_{ij}} = X_{ij} - l_{ij}$ and $\bar{U_{ij}} = U_{ij} - l_{ij}$
But in this project we are going to discuss a special case of this problem with only zero lower bounds. The total amount of flow from s to t is measured in either of two equivalent ways,either the amount leaving the source s or the amount entering the sink t.That is;
Value of flow$(V) = \sum_{j \in NA(s)} X_{sj} = \sum_{j \in NB(t)} X_{jt}$
Assumptions In Maximum Flow Problem
In examining the maximum flow problem the following simplifying assumptions are imposed[1].

- The network is directed.
- All capacities are nonnegative integers.
  For some algorithms this assumption is not restrictive but it is a must for others to take in to consideration.
- The network does not contain a directed path from node s to node t composed only of infinite capacity arcs.
- Whenever an arc (i, j) belongs to A, arc (j, i) also belongs to A.This assumption is nonrestrictive because we allow arcs with zero capacity.
- The network does not contain parallel arcs i.e. two or more arcs with the same tail and head nodes.Here for an arc (i, j),i is tail node and j is head node.

Flow Augmenting Path Algorithm
We can partition the arcs of a path P into two groups namely forward arcs and backward arcs.If flow through an arc is allowed in only one direction as in a one way street, the arc is said to be a *directed arc*. The direction is indicated by adding an arrowhead at the end of the line representing the arc. On the other hand if flow through an arc is allowed in either direction as in a pipeline that can be used to pump fluid in either direction, the arc is said to be an *undirected arc*.

Definition 1. *A flow augmenting path(FAP)is any directed path P from the source to the sink made up of*

- *The forward arcs with $U_{ij} - X_{ij} > 0$ and*
- *The backward arcs with $X_{ji} > 0$*

The maximum flow that we can augment through a FAP is $\delta = \min\{\delta_1, \delta_2\}$ for

- $\delta_1 = \min\{U_{ij} - X_{ij}\}$ for all forward arcs $(i, j) \in P$
- $\delta_2 = \min\{X_{ji}\}$ for all backward arcs $(j, i) \in P$

Now the new flow in the network is found as follows;

$$X'_{ij} = \begin{cases} X_{ij} + \delta & , \ (i,j) \ \ is \ \ forward \in P \\ X_{ji} - \delta & , (j,i) \ \ is \ \ backward \in P \end{cases}$$

Thus we can send more flow than the current flow and hence FAP can increase the flow.

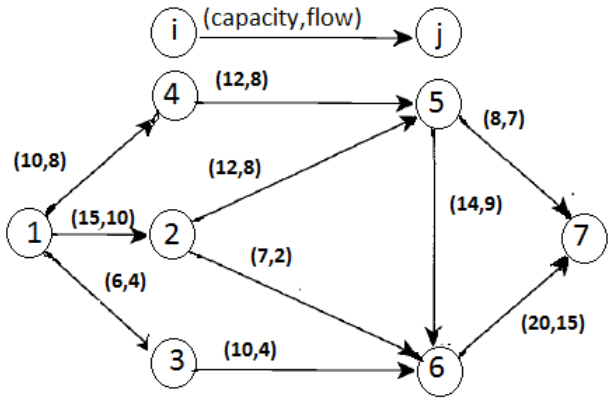


Figure 2.1

In the above figure the current flow is $8 + 10 + 4 = 7 + 15 = 22$ which is not the maximum one because we can send more flow along different flow augmenting paths.In this particular example as the given directed $s-t$ graph with $s = 1$ and $t = 7$ has small number of nodes and arcs one can identify that the flow augmenting paths easily.But later on we will see an algorithm to identify the flow augmenting path in a simple way for directed $s - t$ graphs.For now by augmenting flows along the flow augmenting paths we have found the final graph ;

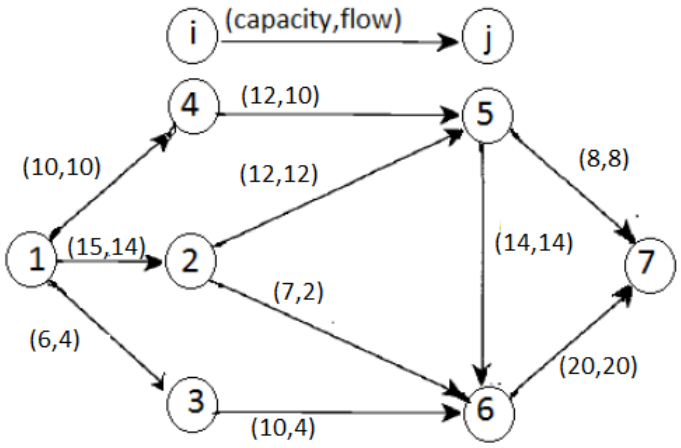


Figure 2.2

Hence the flow is maximum and has a value of; $10 + 14 + 4 = 28 = 8 + 20$

$s - t$ **Cut**

A cut denoted by $[S, \bar{S}]$ defines a set of arcs which when deleted from the network will cause a total disruption of flow between the source s and sink t nodes.In other words a cut is a set of arcs whose endpoints belong to the different subsets S and $\bar{S}$. An arc (i, j) with $i \in S$ and $j \in \bar{S}$ as a forward arc of the cut, and an arc (i, j) with $i \in \bar{S}$ and $j \in S$ as a backward arc of the cut $[S, \bar{S}]$.

Graphically,

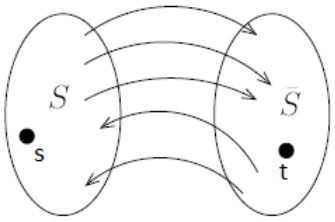


Figure 2.3

The capacity of an $s - t$ cut $U_{[S,\bar{S}]}$ is the sum of the capacities of the *forward arcs* in the cut. That is $U_{[S,\bar{S}]} = \sum_{(i,j)\in[S,\bar{S}]} U_{ij}$

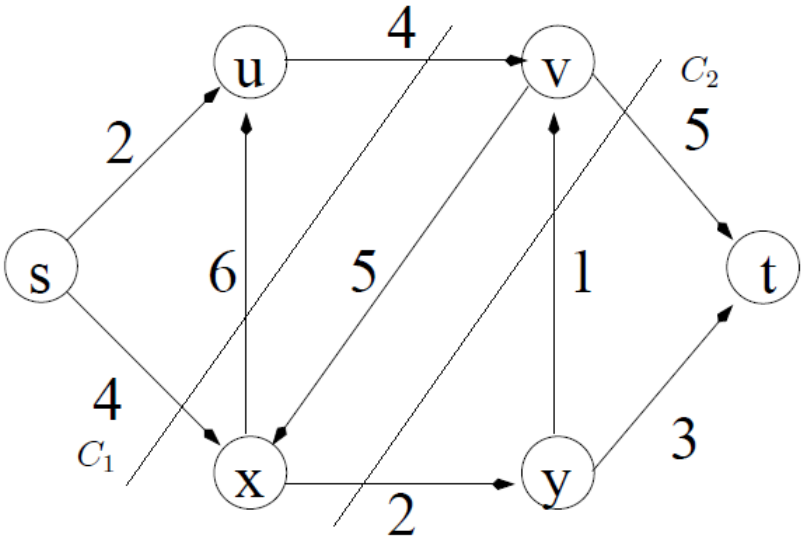


Figure 2.4

For example in the above graph for the cut

- Along $C_1, S = \{s, u\}$ and $\bar{S} = \{v, x, y, t\}$.Hence the cut set is $[S, \bar{S}] = \{(s, x), (u, v)\}$
- Along $C_2, S = \{s, u, v, x\}$ and $\bar{S} = \{y, t\}$.Thus $[S, \bar{S}] = \{(x, y), (v, t)\}$

In addition to that the capacity for $C_1 = 4 + 4 = 8$ and for C_2 is $2 + 5 = 7$
Flow across an $s - t$ cut
The net flow V across any $s - t$ cut is simply adding up the forward flows (flows from nodes in S to nodes in $\bar{S}$) and subtract the backward flows (flows from nodes in $\bar{S}$ to nodes in S) which is given by;

$$V = \sum_{(i,j)\in[S,\bar{S}]} X_{ij} - \sum_{(i,j)\in[\bar{S},S]} X_{ij} \tag{2.2}$$

The net flow across a cut can include negative flows between nodes, but that the capacity of a cut is composed entirely of nonnegative values. That is positive flow from S to $\bar{S}$ is added while positive flow from $\bar{S}$ to S is subtracted.But the capacity of a cut $[S, \bar{S}]$ is computed only from edges going from S to $\bar{S}$. Edges going from $\bar{S}$ to S are not included in the computation of the capacity of the cut $[S, \bar{S}]$. More over among all $s-t$ cuts in the network that separate the source and sink nodes with minimum capacity is called *minimum cut*.More over *minimum cut problem*: Is to find the cut with the minimum capacity from among all cuts in the network that separate the source and sink nodes.

2.1.1 Ford Fulkerson Labeling Algorithm

In the above method of increasing a flow it is simple for graphs having few number of nodes.But for complicated graphs and graphs having large number of nodes the method of identifying

the FAP is difficult.To over come such kind of difficulty Ford Fulkerson Labeling algorithm is applied.Labling process is a systematic search for a flow augmenting path from the source node s to the sink node t.The computation in the algorithm progresses by a sequence of *labelings* each of which either results in a flow of higher value or terminates with the condition that the current flow is maximum.
In the Ford Fulkerson labeling algorithm each node is considered to be in one of the three states (s_1, s_2, s_3).

- Labeled and scanned
- labeled and unscanned
- Unlabeled

The Ford-Fulkerson algorithm is iterative.At each iteration, we increase the flow value by by some $\delta > 0$ by finding a flow augmenting path(FAP),which we can think of simply as a path from the source s to the sink t along which we can send more flow. Repeat the general step until;

1. The sink t is labeled and unscanned where by we can identify the flow augmenting path and do flow augmentation by backtracking using the labels.
2. No more labels can be assigned and the sink is unlabeled for which the algorithm terminates and the flow is maximum.

In the second one the source node is not connected to the sink node in the residual network.The set of arcs leading from labeled to unlabeled nodes is a minimal cut. Suppose at this stage that S is the set of labeled nodes and $\bar{S} = N \backslash S$ is the set of unlabeled nodes.From this $s \in S$ and $t \in \bar{S}$.Since the algorithm cannot label any node in $\bar{S}$ from any node in $S, r_{ij} = 0$ for each $(i, j) \in [S, \bar{S}]$.

$$r_{ij} = (U_{ij} - X_{ij}) + X_{ji}$$

, Moreover Substituting $X_{ij} \leq U_{ij} \& X_{ji} \geq 0$ in to the above equation yields the condition $r_{ij} = 0$ which in turn implies that $X_{ij} = U_{ij}$ for every arc $(i, j) \in [S, \bar{S}]$ and $X_{ji} = 0$ for every arc $(i, j) \in [S, \bar{S}]$.
Now let the net flow across the $s - t$ cut $[S, \bar{S}]$ is denoted by V equals the net flow from left to right across the cut,so that V can not be greater than the total capacity of all arcs from i to j as pictured below;

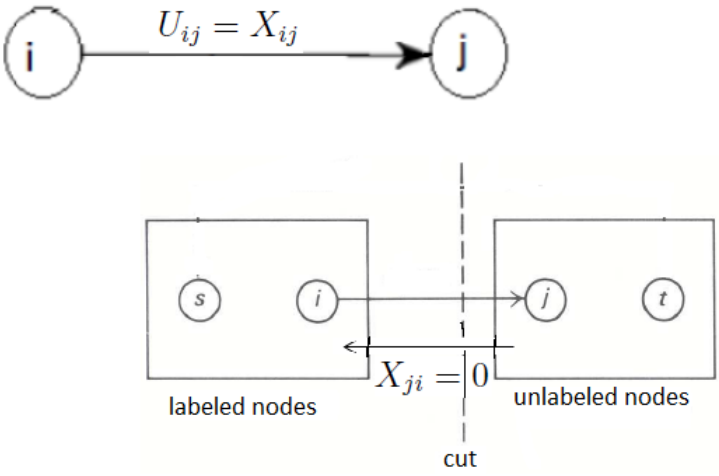


Figure 2.5

$$V = \sum_{(i,j)\in[S,\bar{S}]} X_{ij} - \sum_{(i,j)\in[\bar{S},S]} X_{ij}$$
$$= \sum_{(i,j)\in[S,\bar{S}]} U_{ij} = U_{[S,\bar{S}]}$$

The main reason underlying the computational efficiency of the labeling process is that once a node is labeled and scanned it can be ignored for the remainder of the process.Labeling a node N corresponds to locating a path from s to N that can be the initial segment of a flow augmenting path.

Notation 1. *Given a directed graph* $G = (N, A)$ *and* $i \in N$ *,then*

- *The set of nodes after* i *is denoted by* $NA(i) = \{j \in N : (i, j) \in A\}$
- *The set of nodes before* i *is denoted by* $NB(i) = \{j \in N : (j, i) \in A\}$

Ford Fulkerson Labeling Algorithm

1. Initialization set $s_1 = \emptyset, s_2 = \{1\}, s_3 = \{2, 3, ..., n\}$ for n is the number of nodes
2. Scanning choose $i \in s_2$ that is node i is to be scanned. $s_1 = s_1 \cup \{i\}\&\ \ s_2 = s_2 \backslash \{i\}$
3. Labeling
   - $\forall j \in NA(i)\& \quad \in \quad s_3$ do the following;
     - If $X_{ij} < U_{ij}$
       - label j as $(+, i)$
       - $s_2 = s_2 \cup \{j\}\& \quad s_3 = s_3 \backslash \{j\}$;
     - If $j = n$ identify the flow augmenting path by backtracking using the labels and remove all the labels and return back in to step 1.
   - $\forall j \in NB(i)\& \quad \in \quad s_3$ do the following;
     - If $X_{ij} > 0$ then,
       - label j as $(-, i)$;
       - $s_2 = s_2 \cup \{j\}\& \quad s_3 = s_3 \backslash \{j\}$;
4. If $s_2 \neq \emptyset$ go to step 2;other wise the current flow is maximum and stop.

proposition 1. *For any feasible flow its value V is less than or equal to the capacity of any $s-t$ cut in the network.That is ;*

$$V \leq U_{[S,\bar{S}]}$$

This is found by substituting $X_{ij} \leq U_{ij}$ in the first expression and,$X_{ij} \geq 0$ in the second expression of 2.2.

proposition 2. *The maximum flow value V in the $s-t$ network is at most equal to the capacity of a minimum $s-t$ cut.That is $V \leq U_{[S,\bar{S}]}$.*

Proof. Let y be the capacity of a minimum $s-t$ cut.As V is the value of the feasible flow,from proposition 1 V is less than or equal to the capacity of any $s-t$ cut.Thus $V \leq y$. □

proposition 3. *Let V' be flow amount and Y' be the capacity of a certain $s-t$ cut.If $V' = Y'$,then V' is the maximum flow and Y' must be the capacity of a minimum $s-t$ cut.*

2.1.2 Optimality Conditions

Theorem 1. *[1]* ***(Augmenting Path Theorem)***
A flow is maximum if and only if it admits no augmenting path from s to t.

Proof. proof by contra positive
($\Rightarrow$) Maximum flow $\Rightarrow$ no augmenting path
If there is an augmenting path in the residual network G_X, then from flow augmenting path algorithm of the second step we can push additional flow along that path which implies that X was not a maximum flow.Thus the contrapositive of this is;if a flow V is is maximum then clearly there doesn't exist an augmenting path from the source node s to the sink node t.That is there is no augmenting path such that the current flow can be further increased.
($\Leftarrow$) To prove the reverse implication that is no augmenting path $\Rightarrow$ the flow is maximum ;
By contrapositive $\equiv$ a flow X is not maximum $\Rightarrow$ X has an augmenting path.
From this X is not maximum implies that we can find a flow $\bar{X}$ such that $\mid \bar{X} \mid > \mid X \mid$ this implies that we can find an augmenting path having a positive flow $\delta > 0$ where by we can increase the flow.Therefore the theorem holds. □

Theorem 2. *(Maximum flow minimum cut theorem) The maximum value of the flow from a source node s to a sink node t in a capacitated network equals the minimum capacity among all $s-t$ cuts.*

Proof. Follows from the previous propositions 1 up to 3 together with the termination of the Ford Fulkerson labeling algorithm.When the algorithm terminates the set of arcs leading from labeled to unlabeled nodes determines a cut which is minimum one.That is the arcs from S to $\bar{S}$ (forward arcs) have the property that $X_{ij} = U_{ij}$ and the arcs from $\bar{S}$ to S have the property of $X_{ij} = 0$.
From this we can see that;
$V = \sum_{(i,j)\in[S,\bar{S}]} X_{ij} - \sum_{(i,j)\in[\bar{S},S]} X_{ij}$ becomes
$\sum_{(i,j)\in[S,\bar{S}]} X_{ij} = \sum_{(i,j)\in[S,\bar{S}]} U_{ij}$ and this is the capacity of the $s-t$cut which is minimum. □

2.2 Minimum-cost flow problem

The minimum-cost flow problem is to find the cheapest possible way of sending a certain amount of flow through a flow network. Given a flow network,that is,a directed graph $G(N, A)$ with source $s \in N$ and $t \in N$ where each edge $(i, j) \in A$ has capacity $U_{ij} > 0$,flow $X_{ij} \geq 0$ and cost C_{ij}.We associate with each node $i \in N$ a number $b(i)$ which indicates its supply or demand depending on wether $b(i) > 0$ or $b(i) < 0$.The total cost of sending this flow is $\sum_{i,j)\in A} X_{ij}C_{ij}$.
Now the problem is formulated as follows;
$\min \sum_{(i,j)\in A} X_{ij}C_{ij}$
Subject to ;
$\sum_{\{j:(i,j)\in A\}} X_{ij} - \sum_{\{j:(j,i)\in A\}} X_{ij} = b(i) \quad \forall i \in N$
$l_{ij} \leq X_{ij} \leq U_{ij}$

$$b(i) = \begin{cases} > 0 & \text{for } i = s \\ 0 & \text{for } \forall i \in N \backslash \{s, t\} \\ < 0 & \text{for } i = t \end{cases}$$

For $b(i)$ is positive implies that node i is supply node,for negative case node i is a demand node

and otherwise transshipment nodes.
If we have multiple supply(source) nodes and demand(sink) nodes we can transform it in to a single supply node and single sink node.To do this;

- If $b(i) > 0$, create arc (s, i) with $U_{si} = b(i), C_{si} = 0$
- If $b(k) < 0$, create arc (k, t) with $U_{kt} = |b(k)|, C_{kt} = 0$
- The supply say B at node s is the sum of all supplies and also the demand $(-B)$ at node t is the sum of all demands.

Example 4.

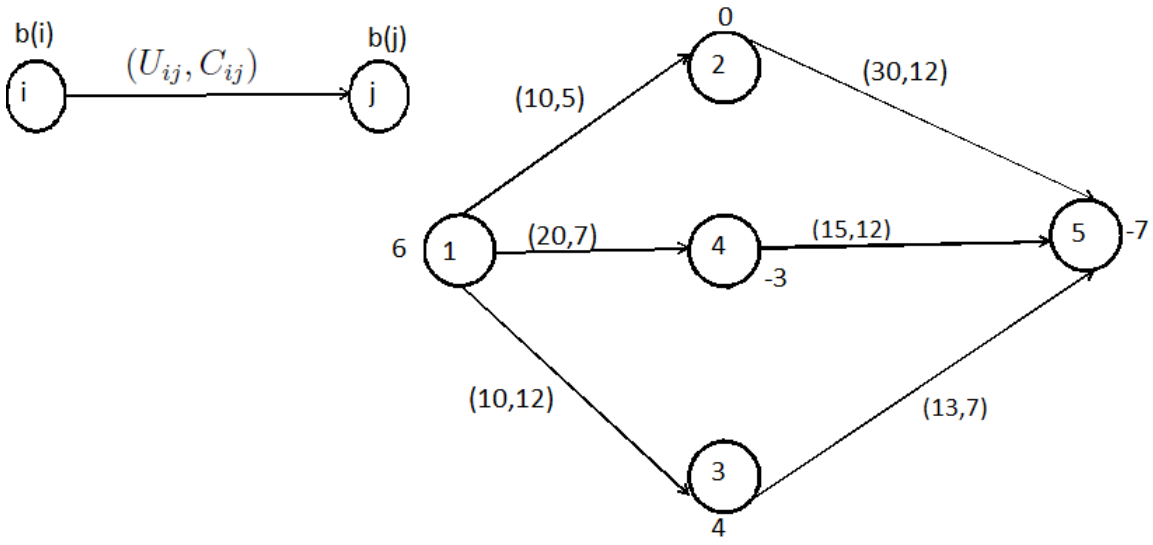


Figure 2.6

Figure 2.6 can be transformed in to an $s - t$ minimum cost flow problem as follows;

- Add node s having $U_{si} = b(i), C_{si} = 0$ for $i = 1\&3$
- Add node t having $U_{jt} =\mid b(j) \mid, C_{jt} = 0$ for $j = 5\&4$
- The supply at node s is the sum of all supplies (10) and also the demand at node t is the sum of all demands(-10).

Thus the final graph is;

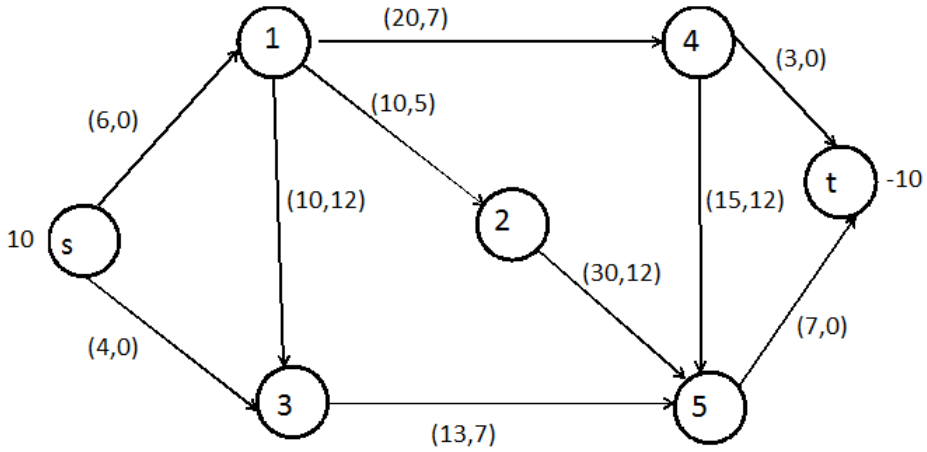


Figure 2.7

Assumptions in minimum cost flow problem
Here are some simplifying assumptions for the minimum cost flow problem which are taken from [5].

- All data (cost, supply/demand. and capacity) are integral.
- The network is directed.
- The supplies/demands at the nodes satisfy the condition $\sum_{i \in V} b(i) = 0$.Total flow being generated at the supply nodes equals the total flow being absorbed at the demand nodes other wise no feasible solution.
- All arc costs are nonnegative.
- For any pair i and j of nodes,the network does not contain both the arcs (i, j) and (j, i).

2.3 Solution Method For Minimum Cost Flow Problem

2.3.1 Cycle Canceling Algorithm

This algorithm maintains a feasible solution and at every iteration attempts to improve its objective function that is to minimize the total cost.The algorithm first establishes a feasible flow X in the network by solving a maximum flow problem (Section 2.1).Then it iteratively finds negative cost-directed cycles in the residual network and augments flow on these cycles.The algorithm terminates when the residual network contains no negative cost-directed cycle and it has found a minimum cost flow.

- Find the feasible flow(X).
- Construct the residual network G_X.
- Check the existence of directed negative cost cycle C in the residual network G_X.
- If the residual network G_X contains a negative cost cycle C decrease the flow by $\delta = \min_{(i,j)\in C}\{U_X(ij)\}$ units in the forward arcs and increase the flow by $\delta = \min_{(i,j)\in C}\{U_X(ij)\}$ units in the back ward arcs in C.Other wise the flow X is optimal that is X is a feasible flow of minimum cost.
- Again construct the next residual network until there is no negative cost cycle C in the residual network G_X. Here the total amount of flow does not change but the only change is that the cost of the flow which is with least one.

Definition 2. *Augmenting Cycle:it is a directed cycle C having negative cost cycle in a residual graph.If we have a negative cost cycle,then we can augment a flow around the cycle to get another flow of same value but smaller cost.That is;*

An augmenting cycle;

- Can send flow around a cycle i.e it decreases the flow on forward arcs in the cycle C and increases the flow on backward arcs in the cycle
- It strictly decreases cost
- Preserves feasibility

Definition 3. *For $d(j)$ is the shortest path from node 1 to node j ,the node potential Π is the negative of the shortest path that is $\Pi_j = -d(j)$.It is the cost of sending 1 unit from j to 1. along the tree path.*

For a given node potential Π, we define the reduced cost of an arc (i, j) by

$$C_{ij}^{\Pi} = C_{ij} - \Pi_i + \Pi_j$$

The reduced cost of an arc measures the cost of the arc relative to the shortest path distances Π_i and Π_j.

property 1. *For any directed path P from node i to node j*
$\sum_{(i,j)\in P} C_{ij}^{\Pi} = \sum_{(i,j)\in P} C_{ij} - \Pi_i + \Pi_j$

property 2. *For any directed cycle C, $\sum_{(i,j)\in C} C_{ij}^{\Pi} = \sum_{(i,j)\in C} C_{ij}$*

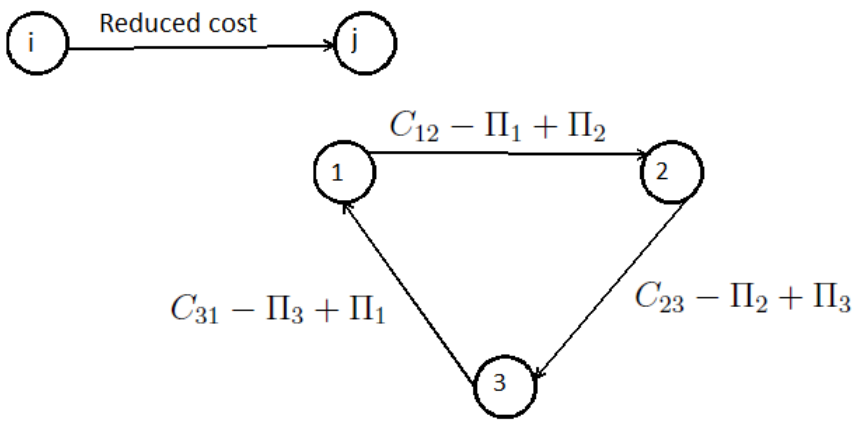


Figure 2.8

$$\sum_{(i,j)\in C} C_{ij}^{\Pi} = C_{12} - \Pi_1 + \Pi_2 + C_{23} - \Pi_2 + \Pi_3 + C_{31} - \Pi_3 + \Pi_1 = C_{12} + C_{23} + C_{31}$$

which is equal to the sum of all the costs in the cycle $1 - 2 - 3 - 1$.The distance label $d(j)$ is an estimate of the shortest path distance from the source node s to node j.Let $d(j)$ for j is different from the source node s denote the length of a shortest path from the source node to the node j.Set $d(s) = 0$ as it represents the shortest distance from s to itself s which is zero. If the distance labels are shortest path distances,they must satisfy:

$$d(j) \leq d(i) + C_{ij}, forall(i,j) \in A$$

The length of the shortest path to node j is is less than the length of the shortest path to node i plus the length of the arc (i,j) otherwise if for some arc $(i,j) \in A$ satisfy the condition $d(j) > d(i) + C_{ij}$; We could improve the length of the shortest path to node j by passing through node i, there by contradicting the optimality of distance labels $d(j)$.

2.3.2 The Network Simplex Algorithm

For problems having small number of nodes and arcs we can iteratively augment flows along a series of negative cost cycles to generate an improving sequence of solutions to the minimum cost flow problem.But it is very difficult and takes long time for problems having large number of nodes and arcs.Hence the *network simplex algorithm* is applied to eliminate this difficulty.The linear programming minimum cost flow problem has n constraints corresponding to each node and any one of the node constraints is redundant.The reason is that summing all these constraint equations yields nothing but zeros on both sides.So the negative of any one of these equations equals the sum of the rest of the equations.With just $n-1$ non redundant node constraints,these equations provide just $n-1$ basic variables for a basic feasible solution.These basic constraints correspond to the $n-1$ arcs of the spanning tree(a spanning subgraph of G which is tree) of the graph G. The network simplex algorithm uses a particular strategy for generating negative cost cycles.Note that the cost in the cycle is the sum of the costs of forward arcs minus the sum of the costs of backward arcs in the cycle.The central concept underlying the network simplex algorithm is the notion of spanning tree solutions,which are solutions that we obtain by fixing the flow of every arc not in a spanning tree either at value zero or at the arc's flow capacity.We can solve uniquely for the flow on all the arcs in the spanning tree.Here the minimum cost flow problem always has at least one optimal spanning tree solution and that it is possible to find an optimal spanning tree solution by moving from one such solution to another,at each step introducing one new non tree arc into the spanning tree in place of one tree arc.Finally it finds a spanning tree solution that satisfies the network optimality conditions.This method is known as the *network simplex algorithm* because spanning trees correspond to the so-called basic feasible solutions of linear programming,and the movement from one spanning tree solution to another corresponds to the pivot operation of the general simplex method.The essential steps of the network simplex algorithm,for example,determining node potentials or moving from one spanning tree to another,are specializations of the usual steps of the simplex method for solving linear programs.
For any feasible solution X, we say that an arc (i,j) is;

1. A free arc if $0 < X_{ij} < U_{ij}$ where by we can both increase and decrease flow while honoring the bounds on arc flows.

2. A restricted arc if $X_{ij} = 0$ or $X_{ij} = U_{ij}$ for which in case of $X_{ij} = 0$ we can only increase a flow but for $X_{ij} = U_{ij}$ we can only decrease flow.

We refer to a solution X as a *cycle free solution* if the network contains no cycle composed only of free arcs.In a cycle free solution, we can augment flow on any augmenting cycle in only a single direction since some arc in any cycle will restrict us from either increasing or decreasing that arc's flow.In addition feasible solution X and an associated spanning tree of the network as a *spanning tree solution* if every non tree arc is a restricted arc.In this context we are not referring that the associated tree. Note:in a spanning tree solution,the tree arcs can be free or restricted.If every tree arc in a spanning tree solution is a free arc,we say that the spanning tree is non degenerate;otherwise,a degenerate spanning tree.

Minimum cost flow problems have both optimal cycle free and spanning tree solutions.But The network simplex algorithm will restrict its search for an optimal solution to only spanning tree solutions.If a network contains positive flow $\delta > 0$ around a cycle we can augment a flow in the cycle and this augmentation decreases the cost.The orientation of the cycle depends on the nature of the arcs present in the cycle.we set δ as large as possible while preserving nonnegativity of all arc flows. The augmentation increases the flow on arcs along the orientation of the cycle (forward arcs) by δ units and decreases the flow on arcs opposite to the orientation of the cycle (backward arcs) by δ units. We can convert a cycle free solution into a spanning tree solution by adding some restricted arcs to the collection of node-disjoint trees and produce a spanning tree. A spanning tree solution partitions the arc set A into three subsets such that $T \cup L \cup U = A$ with;

1. T, the arcs in the spanning tree (basic arcs)
2. L, the non tree arcs whose flow is restricted to value zero ($X_{ij} = 0$)
3. U, the non tree arcs whose flow is restricted in value to the arcs's flow capacities ($X_{ij} = U_{ij}$)

Hence we refer to the triple (T, L, U) as a spanning tree structure.The arc flows in T are selected so that each node satisfies its supply/demand constraint.A spanning tree flow is called feasible if it satisfies its upper and lower bound that is $0 \leq X_{ij} \leq U_{ij}$.Otherwise,it is infeasible.If we already have a feasible flow X,we can augment the flow (without increasing the cost) until the set of free arcs is a forest(collection of disjoint nodes).Then we add restricted arcs until a spanning tree T results.L and U are determined by the resulting feasible solution.The network simplex algorithm maintains a feasible spanning tree structure and moves from one spanning tree structure to another until it finds an optimal structure.At each iteration,the algorithm adds one arc to the spanning tree in place of one of its current arcs.The entering arc is a non tree arc violating its optimality condition.

Main computations in the network simplex algorithm

1. Adds a non tree arc violating its optimality condition to the spanning tree, creating a negative cycle (which might have zero residual capacity).
2. Sends the maximum possible flow in this cycle until the flow on at least one arc in the cycle reaches its lower or upper bound.
3. Drops an arc whose flow has reached its lower or upper bound, giving us a new spanning tree structure.

We consider the tree as hanging from a specially designated node, called the *root*.In this project work we assume that node s is the root node.The tree arcs either are upward pointing toward the root node or downward pointing away from the root node.

Computing node potentials

The network simplex algorithm maintains the condition that the reduced cost $C_{ij}^{\Pi} = C_{ij} - \Pi_i + \Pi_j$ of every arc (i, j) in the current spanning tree is zero.Given the current spanning tree structure (T, L, U),the method first determines values for the node potentials Π that will satisfy this condition for the tree arcs.This is computed as follows;

$$C_{ij}^{\Pi} = C_{ij} - \Pi_i + \Pi_j = 0 \quad \forall (i,j) \in T \tag{2.3}$$

From equation 2.5 if we know one of the node potentials Π_i or Π_j we can easily compute the other one.That is

- $\Pi_i = \Pi_j + C_{ij}$ if Π_j is known
- $\Pi_j = \Pi_i - C_{ij}$ if Π_i is known

Optimality Conditions

Theorem 3. *[1](Reduced Cost Optimality Conditions) A flow f^* is an optimal solution of the minimum cost flow problem if and only if there is a vector π so that $C_{ij}^{\Pi} \geq 0 \quad \forall (i,j) \in G_{f^*}$.*

Proof. ($\Rightarrow$) Assume that f^* is an optimal solution of the minimum cost flow problem that is G_{f^*} contains no negative cycle.We need to show that $C_{ij}^{\Pi} \geq 0 \quad \forall (i,j) \in G_{f^*}$ and for some node potential π.
If the network contains no negative cost cycle,the distance labels are well defined and satisfy

$$d(j) \leq d(i) + C_{ij}$$

Rearranging the above equation yields

$$C_{ij} - d(j) + d(i) = C_{ij} + \Pi_j - \Pi_{ij} \geq 0$$

For $\Pi = -d$. This implies that $C_{ij}^{\Pi} \geq 0 \quad \forall (i,j) \in G_{f^*}$.
($\Leftarrow$) Suppose that $C_{ij}^{\Pi} \geq 0 \quad \forall (i,j) \in G_{f^*}$ for all directed cycle C in the residual network G_{f^*} we need to show that f^* is an optimal solution of the minimum cost flow problem.
$C_{ij}^{\Pi} \geq 0 \quad \forall (i,j) \in G_{f^*}$ from property 1 this implies that

$$\sum_{(i,j) \in C} C_{ij} \geq 0 \quad \forall (i,j) \in G_{f^*}$$

Hence G_{f^*} contains no negative cost cycle which in turn implies that f^* is an optimal solution for the minimum cost flow problem.

□

Theorem 4. *[1](Complementary Slackness Optimality Conditions).A feasible solution X is an optimal solution of the minimum cost flow problem if and only if for some set of node potentials Π,the reduced costs and flow values satisfy the following complementary slackness optimality conditions for every arc $(i, j) \in A$:*

1. *If* $C_{ij}^{\Pi} > 0$, *then* $X_{ij} = 0$.

2. *If* $0 < X_{ij} < U_{ij}$, *then* $C_{ij}^{\Pi} = 0$.

3. *If* $C_{ij}^{\Pi} < 0$, *then* $X_{ij} = U_{ij}$.

Proof. 1.If $C_{ij}^{\Pi} > 0$,the residual network can not contain the arc (j, i) since $C_{ji}^{\Pi} = -C_{ij}^{\Pi}$,which is a contradiction.Thus $X_{ij} = 0$.
2.If $0 < X_{ij} < U_{ij}$,the residual network contains both the arcs (i, j) and (i, j).From reduced cost optimality conditions $C_{ij}^{\Pi} \geq 0$ and $C_{ji}^{\Pi} \geq 0$.But $C_{ji}^{\Pi} = -C_{ij}^{\Pi}$ and from those inequality $C_{ji}^{\Pi} = C_{ij}^{\Pi} = 0$
3.If $C_{ij}^{\Pi} < 0$,the residual network cannot contain the arc (i, j) because $C_{ij}^{\Pi} < 0$ for that arc which contradicts reduced cost optimality conditions.Therefore $X_{ij} = U_{ij}$. □

Theorem 5. *A spanning tree structure* (T, L, U) *is an optimal spanning tree structure of the minimum cost flow problem if it is feasible and for some choice of node potentials* Π*,the arc reduced costs* C_{ij}^{Π} *satisfy the following conditions:*

1. $C_{ij}^{\Pi} = 0 \quad \forall (i, j) \in T$

2. $C_{ij}^{\Pi} \geq 0 \quad \forall (i, j) \in L$

3. $C_{ij}^{\Pi} \leq 0 \quad \forall (i, j) \in U$

Proof. Follows from theorem 4. □

Network simplex algorithm for minimum cost flow problem

1. Initialization
   - Start with a feasible flow X in a spanning tree structure (T, L, U)
   - Determine a node potential Π
2. Optimality Test
   - If the current flow in (T, L, U) satisfies the Optimality Test,stop
   - Otherwise, go to Step 3.
3. Basis exchange
   Check to see whether the spanning tree structure satisfies the following conditions;
   - $C_{ij}^{\Pi} \geq 0 \quad \forall\ (i, j) \in L$.
   - $C_{ij}^{\Pi} \leq 0 \quad \forall\ (i, j) \in U$.

Select a non tree arc violating the above optimality conditions to be introduced into the tree.Two types of arcs (violating arcs) are eligible to enter the tree:

(a) Any arc $(i,j) \in L$ with $C_{ij}^{\Pi} < 0$

(b) Any arc $(i,j) \in U$ with $C_{ij}^{\Pi} > 0$

Entering Arc

There are different rules for selecting the entering arc.Some of them are listed below.

- Dantzig's pivot rule:as suggested by George B. Dantzig,it selects an arc with the maximum violation to enter the tree.This is because of the arc with the maximum violation causes the maximum decrease in the objective function per unit change in the value of flow on the selected arc.

- First eligible arc pivot rule:This rule scans the arc list sequentially and select the first eligible arc to enter the tree.This rule quickly identifies the entering arc.

- Candidate list pivot rule:It selects the entering arc using a two-phase procedure namely major iterations and minor iterations where by in the first phase construct a candidate list of eligible arcs and then select an eligible arc from the candidate list with the maximum violation arcs. In a major iteration first examine arcs emanating from node 1 and add eligible arcs to the candidate list. We repeat this process for nodes 2,3,...,until we have examined all the nodes.The next major iteration begins with the node where the previous major iteration ended and examines nodes.
  In a minor iteration,we scan the arcs, we update the candidate list by removing those arcs that are no longer eligible.Once the candidate list becomes empty,rebuild the candidate list by performing another major iteration.

Leaving Arcs

While we select arc (i,j) as the entering arc,its addition to the tree T creates exactly one cycle C which we refer to as the pivot cycle that is $T \cup (i,j)$ has a cycle. The orientation of the cycle C is the same as that of (i,j) if $(i,j) \in L$ and opposite the orientation of (i,j) if $(i,j) \in U$. Sending additional flow around the pivot cycle C in the direction of its orientation strictly decreases the cost of the current solution at the per unit rate of $\mid C_{ij}^{\Pi} \mid$.Note that augmenting flow ($\beta = min\{\beta_{ij} : (i,j) \in C\}$) along C increases the flow on forward arcs and decreases the flow on backward arcs and;

$$\beta_{ij} = \begin{cases} U_{ij} - X_{ij} & \text{if } \forall \ \ forwardarcs(i,j) \in C \\ X_{ij} & \text{if } \forall \ \ backwardarcs(i,j) \in C \end{cases}$$

As mentioned before augmenting a flow in the pivot cycle C decreases the cost.After augmentation of the flow by β units around the pivot cycle C select an arc (p,q) which is the leaving arc with $\beta_{pq} = \beta$.Here if $\beta > 0$,then the pivot iteration is a non degenerate iteration and degenerate iteration if$\beta = 0$.

Updating the Tree

Once the network simplex algorithm has determined a leaving arc (p,q) for a given entering arc

(i, j), it updates the tree structure.
Case 1.If the leaving arc (p, q) is the same as entering arc (i, j),this happens when $\beta = \beta_{ij} = U_{ij}$.Here the tree does not change but the arc (p, q) moves from the set L to the set U,or vice versa.
Case 2.If the leaving arc (p, q) is different from the entering arc (i, j),the tree completely changes.For the new spanning tree,arc (p, q) becomes non tree at its lower or upper bound depending on wether $X_{pq} = 0$ or $X_{pq} = U_{pq}$.In addition the node potentials are also changed.The deletion of the arc (p, q) from the previous tree decomposes the set of nodes into two subtrees,T_1 containing the root node s,and the other T_2 which does not contain the root node s.From this T_1 hangs from the root node where as T_2 hangs from node either i or node j.The entering arc (i, j) has one end point in T_1 and the other in T_2 which forms another new spanning tree $\dot{T}$.As $\Pi_s = 0$ and $C^{\Pi}_{ij} = C_{ij} - \Pi_i + \Pi_j = 0$ for all arcs in the new tree imply that the potentials of nodes in the subtree T_1 remain unchanged, and the potentials of nodes in T_2 change by a constant amount δ.Hence the node potentials are updated as;

- $\bar{\Pi}_i = \Pi_i \quad \forall i \in T_1$
- $\bar{\Pi}_i = \Pi_i + \delta \quad \forall i \in T_2$

The choice of δ depends on;

- Case 1.If $p \in T_1$ and $q \in T_2$ increase all node potentials in T_2 by $-C^{\Pi}_{pq}$
- Case 2.If $p \in T_2$ and $q \in T_1$ increase all node potentials in T_2 by C^{Π}_{pq}

Termination
Generally in the network simplex algorithm if each pivot operation in the algorithm is non degenerate,the algorithm terminates finitely.But on the other hand if there exists degenerate pivot operation,algorithm might not terminate finitely.Unless we perform pivots carefully that is bad choices of pivoting leads to cycling.In other words the algorithm does not terminate.But the implementation of strongly feasible spanning tree guarantees finite convergence of the network simplex algorithm.In this case a given spanning tree is strongly feasible spanning tree if;

- Every tree arc with zero flow is pointing to the root node and every tree arc whose flow equals its capacity is pointing away from the root node
- We can send a positive amount of flow from any node to the root along the tree path without violating any flow bound.

Chapter 3

Solution Methods For Maximum Flow Minimum Cost problem In A Network

3.1 Introduction

In *maximum flow problem* the main objective is to maximize the total amount of flow from the source to the sink subject to the capacity constraint(1.1) and the flow conservation (1.2).The amount of the flow is measured in either of two equivalent ways, namely either the amount leaving the source or the amount entering the sink.All the remaining nodes (other than the source and sink nodes) are transshipment nodes.Here we have given simply that the capacity U_{ij},flow X_{ij} (optional) of an arc (i, j).Moreover if the flow X_{ij} an arc is given it does not exceed the capacity of that arc.Different solution methods are applied to solve such kind of problem.In this project work Ford Fulkerson labeling algorithm is applied to find the maximum flow from the source node to the sink node as explained in section 2.1.1.In addition to that as maximum flow problem is a special case of the minimum cost flow problem the network simplex algorithm is also applicable [1].
Given a directed graph $G = (N, A)$ with each arc (i, j) having C_{ij} per unit flow,minimum cost flow problem is to minimize the total cost of sending the available supply through the network to satisfy the given demand.In this case for each node $i \in N$ there is an associated number b_i representing its supply/demand.Like the maximum flow problem,it considers flow through a network with limited arc capacities. There are different solution methods for the minimum cost flow problem like the negative cycle canceling ,successive shortest path,network simplex algorithms.But cycle canceling and network simplex algorithms are explained in section 2.3.1 and 2.3.2 respectively. A variation of this problem is to find a flow which is maximum, but has the lowest cost.This could be called a **maximum-flow minimum-cost problem**.It combines getting as much flow as possible from the source node to the sink node with minimum cost.Each arc (i, j) has flow X_{ij},capacity U_{ij} and cost C_{ij} an amount of money per unit flow flowing through an arc (i, j).It can be specialized in to two problems as

1. The shortest path problem :Again this problem is found by setting all the capacities of an arcs to be zero.Consider a directed network $G = (N, A)$ with an arc length (or arc cost) C_{ij}

associated with each arc $(i, j) \in A$. The shortest path problem is to determine for every node i different from the source node in N a shortest length directed path from node s to node i.Several types of shortest path include;

- Shortest paths from every node to every other node(more general version)
- Shortest paths from one node to all other nodes for networks with arbitrary arc lengths
- Shortest paths from one node to all other nodes when arc lengths are nonnegative
- ...

Depending on the type of the shortest path problem ,several algorithms are applied.For example Dijkstra's algorithm is applied only for non negative arc lengths,Bellman-Ford algorithm is applicable for negative arc lengths[6]. In addition as the shortest path problem is also a special case of the minimum cost flow problem network simplex algorithm is also applicable for the shortest path problem [1].But in this project we have only focused on the solution methods for the maximum flow problem as it is one part of the work.

2. The maximum flow problem:This problem is found by setting all the costs of the arcs to be zero and hence the objective is to find the maximum flow value by ignoring the arc costs.

3.2 Network Simplex Algorithm For Maximum Flow Problem

We have seen that some of the solution methods for this problem in the second chapter.But since the maximum flow is the special case of the minimum cost flow problem the network simplex algorithm is also applied to solve it. We can transform the maximum flow problem in to an equivalent minimum cost flow problem by;

- Introducing an additional arc (t, s) having cost $C_{ts} = -1$ and capacity $U_{ts} = 0$
- Setting $C_{ij} = 0$ for all the original arcs $(i, j) \in A$

More over the set of arcs in the network is changed in to $A = A \cup \{(t, s)\}$.Thus the problem can be formulated as;

$$\min -X_{ts}$$

subject to;

$$\sum_{j \in NA(i)} X_{ij} - \sum_{j \in NB(i)} X_{ij} = 0 \quad \forall i \in Nand \quad 0 \leq X_{ij} \leq U_{ij} \quad \forall (i, j) \in A$$

From the above formulation we can observe that minimizing $-X_{ts}$ is equivalent to maximizing X_{ts}, which is equivalent to maximizing the net flow sent from the source to the sink,since this

flow returns to the source via arc (t, s).Note that in any feasible spanning tree solution that carries a positive amount of flow from the source to the sink arc (t, s) must be in the spanning tree.
We can find the node potentials for the feasible spanning tree of the maximum flow by first putting one node potential zero let $\Pi_t = 0$.Now since the reduced cost for the feasible spanning tree and the costs for those arcs is zero,it follows that;

- $\Pi_s = \Pi_t - C_{ts} = 1$
- $\Pi_i = 1 \quad \forall i \in T_s$
- $\Pi_i = 0 \quad \forall i \in T_t$

Every spanning tree solution of the maximum flow problem defines an $s - t$ cut $[S, \bar{S}]$ in the original network obtained by setting $S = T_s$ and $\bar{S} = T_t$.Each arc in this cut is a non tree arc in which its flow has value zero or equals the arc's capacity.For every forward arc (i, j) in the cut, $C_{ij}^{\Pi} = -1$, and for every backward arc (i, j) in the cut, $C_{ij}^{\Pi} = 1$. Moreover for every arc (i, j) not in the cut,$C_{ij}^{\Pi} = 0$.As a result,if every forward arc (i, j) in the cut satisfies $X_{ij} = U_{ij}$ and every backward arc satisfies $X_{ij} = 0$,this spanning tree solution satisfies the optimality conditions in theorem 5,and therefore it must be optimal.But if in the current spanning tree solution,some forward arc in the cut has a flow of value zero or the flow on some backward arc equals the arc's capacity,all these arcs have a violation of 1 unit.If we select arc (i, j) to enter in to the tree which creates a cycle that contains arc (t, s) as a forward arc.Next augment flow around the cycle to identify the leaving arc and the new tree constitutes a spanning tree for the next iteration. The network simplex algorithm gives another proof of the maximum flow minimum cut theorem which implies that the current maximum flow value equals the capacity of the $s - t$ cut defined by the subtrees T_s and T_t,and thus the value of a maximum flow from node s to node t equals the capacity of the minimum $s - t$ cut.

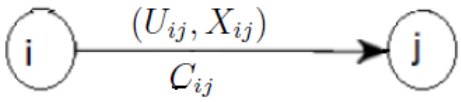
3.3 Solution Methods

Finding the maximum flow minimum cost instead is straightforward.It is a lot harder than the regular maximum flow,shortest path problems which are the special cases of the problem.But in this project we are going to solve it in to two successive steps which are the combination of the previous algorithms as in section 2.1.1 and 2.3.1
Step 1; Solve $s-t$ maximum flow using a procedure in 2.1 let the resulting maximum flow is X^*.
Step 2;Solve the minimum cost flow problem (using the procedure in 2.2) by setting the supply at node s to be X^*,the demand at t to be $-X^*$ and all the other nodes as transshipment nodes.Here we can apply the cycle canceling algorithm,network simplex algorithm to find an optimal solution.The problem is formulated as;
$\min \sum_{(ij)\in A} X_{ij}C_{ij}$
Subject to ;
$\sum_{\{j:(i,j)\in A\}} X_{ij} = \sum_{\{j:(j,i)\in A\}} X_{ij} = b(i) \quad \forall i \in N$
$0 \leq X_{ij} \leq U_{ij}$

$$b(i) = \begin{cases} X^* & \text{if } i = s \\ 0 & \text{if } \forall i \in N\backslash\{s,t\} \\ -X^* & \text{if } i = t \end{cases}$$

Here is an illustrative example for finding a maximum flow with a least(smallest) cost.

Example 5.



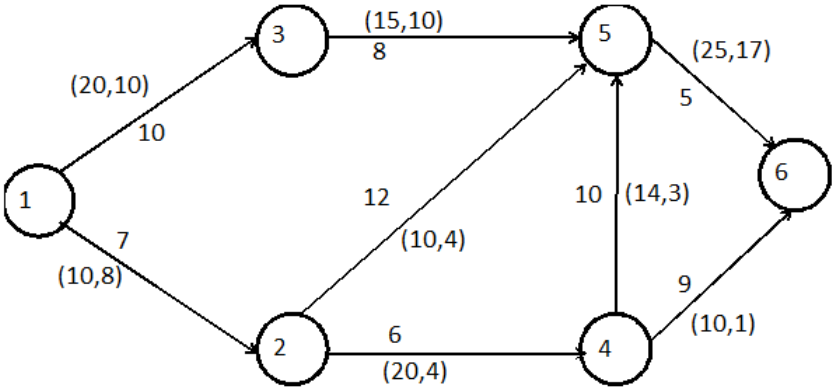


Figure 3.1

Step 1.First apply Ford Fulkerson labeling algorithm to find the maximum flow for ;

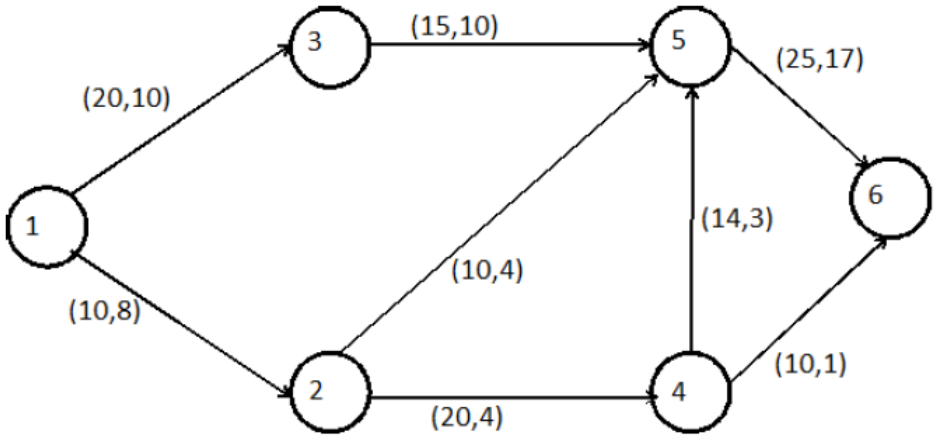
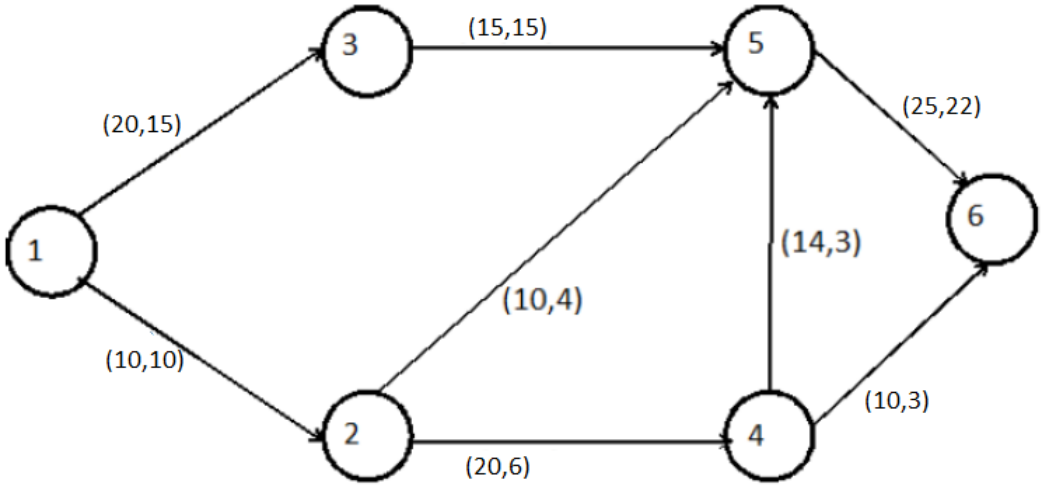


Figure 3.2

Applying all the procedures in Ford Labeling algorithm iteratively we have fond that two distinct flow augmenting paths $p_1 = 1-2-4-6$ and $p_2 = 1-3-5-6$.At final step we have found that ; $s_1 = \{1,3\}, s_2 = \{\emptyset\}$ and $s_3 = \{2,4,5,6\}$.
Here also the $s-t$ cut from s_1 (set of labeled nodes) to s_3 (set of unlabeled nodes) defines the minimum cut with $s_1 = S$ and $s_3 = \bar{S}$ for $[S, \bar{S}]$ defines an $s-t$ cut as explained in chapter 2. The final graph is;



Now the maximum flow is 3+22 = 25.More over the capacity of the cut $S = \{1, 3\}, \bar{S} = \{2, 4, 5, 6\}$ is also $15 + 10 = 25$ which is the smallest one from all the possible cuts.
Step 2.Using $b(1) = 25$ and $b(6) = -25$ it remains to find the minimum cost flow problem;
$\min \sum_{(ij)\in A} X_{ij}C_{ij}$
Subject to ;
$\sum_{\{j:(i,j)\in A\}} X_{ij} - \sum_{\{j:(j,i)\in A\}} X_{ji} = b(i) \quad \forall i \in N$
$0 \leq X_{ij} \leq U_{ij}$

$$b(i) = \begin{cases} 25 & \text{if } i = 1 \\ 0 & \text{if } \forall i \in N \backslash \{1, 6\} \\ -25 & \text{if } i = 6 \end{cases}$$

Applying the network simplex algorithm
We have found that the tree structure (T, L, U) with $T = \{(1, 3), (2, 4), (4, 5), (4, 6)\}, L = \{(1, 2)\}, U = \{(3, 5), (2, 5)(5, 6)\}$
But here as the network simplex algorithm restricts to find only to spanning tree solution, we can convert the above structure in to spanning tree solution.We have already a feasible flow X,we can augment the flow (without increasing the cost) until at least one of the set of free arcs in the cycle becomes restricted that is they are a forest(collection of node disjoint trees).This solution is cycle free and then we can add restricted arcs until a spanning tree T results.L and U are determined from the remaining arcs.For this particular example if we add arc $(3, 5)$ and $(2, 5)$ we have;

1. Initialization
   - The spanning tree structure is $T = \{(1, 3), (2, 5), (2, 4), (5, 6), (3, 5)\}, L = \{(4, 5)\}, U = \{(1, 2), (4, 6)\}$
   - The node potentials $(\Pi_1, \Pi_2, \Pi_3, \Pi_4, \Pi_5, \Pi_6) = (0, -6, -10, -12, -18, -23)$
2. Optimality Test
   - $C_{ij}^{\Pi} \geq 0 \ \ \forall \ (i, j) \in L.$
   - $C_{ij}^{\Pi} \leq 0 \ \ \forall \ (i, j) \in U.$
3. Basis exchange
   From the above arc $(1, 2)$ violates optimality conditions with $C_{12}^{\Pi} = 7-0-6 = 1 > 0$.Hence arc $(1, 2)$ enters in to T creating the cycle $1-3-5-2-1$.Now $\delta = \min\{5, 0, 0, 10\} = 0$ The new spanning tree structure is $T = \{(1, 3), (1, 2), (2, 4), (5, 6), (3, 5)\}, L = \{(4, 5), (2, 5)\}, U = \{(4, 6)\}$ From this arc $(2, 5)$ leaves the tree and its deletion creates $T_1 = (1, 3, 5, 6), T_2 = (2, 4)$.The node potentials in T_1 remains the same.But the node potentials in T_2 is increased by $C_{12}^{\Pi} = 1$ as $2 \in T_2$ and $5 \in T_1$.That is $(\Pi_1, \Pi_2, \Pi_3, \Pi_4, \Pi_5, \Pi_6) = (0, -5, -10, -11, -18, -23)$
   Again checking the optimality conditions this new spanning structure is optimal and hence the algorithm terminates.

More over as we can have more than one spanning tree solutions corresponding to the cycle free solution,for this example we can have another spanning tree solution which is obtained by adding the restricted arcs $(1,2)$ and $(2,5)$.That is the spanning tree structure is;
$T = \{(1,3),(1,2),(2,4),(5,6),(2,5)\}, L = \{(4,5)\}, U = \{(4,6),(3,5)\}$ and the node potentials are $(\Pi_1, \Pi_2, \Pi_3, \Pi_4, \Pi_5, \Pi_6) = (0, -7, -10, -13, -19, -24)$
The above spanning tree structure satisfies the optimality conditions and hence it is optimal which implies that the algorithm terminates here.

Bibliography

[1] Ravindra K.Ahuja,Thomas L.Magnanti James B. Orlin *Network Flows: Theory,Algorithms and Applications*.

[2] Hamdy Taha, *Operations Research An Introduction*, Eighth Edition.

[3] L.R. Ford,D.R. Fulkerson, *Flows in Networks* ,Princeton University Press, Princenton,New Jersey 1962.

[4] Eugenel Lawler, *Combinatorial Optimization:Networks & matroids*.

[5] Frederik S.,Hillier,Gerald J.Lieberman, *Introduction To Operation Research*,seventh edition.

[6] Georege L.Nemhauser & Laurence A.Wolsey, *Integer And Combinatorial Optimization*.

[7] K.Chendra,U.Balakrishna,Purusotham Singamsetty,M. Sundara Murthy, *Maximum Flow with Minimum Cost in A Capacitated and Directed Network Problem*,Volume.1 Number.1 January June 2013, pp 20-30.

[8] Nazimuddin Ahmed,S. Das ,S. Purusotham, *The problem of maximum flow with minimum attainable cost in a network*,OPSEARCH (AprJun 2013) 50(2):pp 197214.

[9] Ravindra K.Ahuja,James B. Orlin,and Robert E. Tarjan, *Improved Time Bounds For The Maximum Flow Problem** ,Vol.18,No.5,pp.939-954,October 1989.

[10] Ravindra K.Ahuja, James B. Orlin, *The Scaling Network Simplex Algorithm*, Operations Research:Vol.40 Supp No 1,January -February 1992.

[11] James B. Orlin, *A polynomial time primal network simplex algorithm for minimum cost flows*,Mathematical Programming,78 (1997), pp 109-129.

[12] Donald Goldfarb,Jianxiu Hao, *A primal simplex algorithm that solves the maximum flow problem in at most nm pivots and* $O(n^2m)$ *time*, Mathematical Programming 47 (1990) 353-365.