2.1

Notation

Katta G. Murty, IOE 612 Lecture slides 2

1. Paths A path $\mathcal{P}$ from the *origin node* x_1 to the *destination node* x_k in G is a sequence of lines connecting x_1 to x_k, but not all these lines may be directed towards x_k. Written as:

$$x_1, e_1, x_2, e_2, \ldots, x_{k-1}, e_{k-1}, x_k$$

where e_r is either (x_r, x_{r+1}) or (x_{r+1}, x_r), or edge $(x_r; x_{r+1})$ with some orientation selected for it, and will be treated as an arc with that orientation.

1.1 Forward, reverse arcs on a path: On path $\mathcal{P}$ an arc is **forward** (*reverse*) if its orientation coincides with (*is opposite to*) direction of travel from origin to destination.

1.2 Simple path: If no point and no line is repeated on it.

1.3 Underlying partial subnetwork of a path: Let:

N = set of distinct nodes appearing on path $\mathcal{P}$.

A = set of distinct lines appearing on $\mathcal{P}$.

Then (N, A) is called the *underlying partial subnetwork of path* $\mathcal{P}$.

1.4 Cycle: Path $\mathcal{P}$ called a **cycle** if:

(i) origin and destination node on $\mathcal{P}$ are same

(ii) and underlying partial subnetwork (N, A) of $\mathcal{P}$ is Eulerian.

1.5 Simple cycle: If no node is repeated on it (with exception of origin = destination).

1.6 Predecessor labels to represent simple paths: Consider simple path $\mathcal{P}$: $x_1, e_1, x_2, e_2, \ldots, x_{k-1}, e_{k-1}, x_k$.

Origin node x_1 has no predecessor, is labeled with $\emptyset$.

For $2 \leq r \leq k$, x_{r-1} is **immediate predecessor** of x_r on $\mathcal{P}$, and is called its **predecessor index**.

e_{r-1} is arc leading into x_r from its immediate predecessor x_{r-1}.

If e_{r-1} forward [*reverse*] label x_r with $(x_{r-1}, +)$ $[(x_{r-1}, -)\ ]$.

Entire path can be retrieved by a **backwards trace** of predecessor labels beginning at destination node.

2. Chains: Paths in which all arcs are forward

2.1 Simple chain: If no point and no line repeated on it. Can be stored using predecessor indices without the $+, -$ labels.

2.2 Circuit: is a chain satisfying (i) and (ii) of 1.4.

2.3 Simple Circuit: is a circuit in which no nodes or arcs are repeated (except origin = destination).

3. Connectedness: Network $G = (\mathcal{N}, \mathcal{A})$ **connected** if there exists a path between every pair of nodes in it, **disconnected** otherwise.

3.1 Connected components: A disconnected network is several connected components put together.

3.2 Strongly connected network: Directed network in which there exists a chain from every node to every other node.

3.3 Strongly connected components: In a directed network, each strongly connected component is a maximal partial network that is strongly connected.

4: Cuts in connected networks

4.1: Cut in a connected undirected network: is a subset of edges, deletion of which makes network disconnected; i.e., it has **path blocking property**. Used in study of electrical networks.

Consider $G = (\mathcal{N}, \mathcal{A})$, let $X \subset \mathcal{N}$, $\bar{X} = \mathcal{N} \backslash X$, with both $X, \bar{X} \neq \emptyset$. This partition yields the cut

$(X; \bar{X})$ = set of all edges in $\mathcal{A}$ with one node in X, and another in $\bar{X}$.

4.2: Cutset: A minimal cut. No proper subset of a cutset has cut property. So, removal of a cutset disconnects G into exactly 2 connected components.

In network below $(\{1, 2, 3\}; \{4, 5, 6\}) = \{e_3, e_4, e_6, e_7\}$ is a cutset. $(\{1, 6\}; \{2, 3, 4, 5\}) = \{e_1, e_9, e_2, e_{10}\}$ is a cut but not a cutset.

4.3: Cuts in Connected Directed Networks: Let $G = (\mathcal{N}, \mathcal{A}, \ell, k)$ be connected directed network with ℓ, k as arc lower bound, capacity vectors.

Let $X \subset \mathcal{N}$, $\bar{X} = \mathcal{N} \backslash X$, with both $X, \bar{X} \neq \emptyset$, be a partition of $\mathcal{N}$. This generates a **Cut** $[X, \bar{X}]$ **in** G, consisting of two sets of arcs:

$(X, \bar{X})$ = Set of **forward arcs of the cut**$[X, \bar{X}] = \{(i, j) : i \in X, j \in \bar{X}$, and $(i, j) \in \mathcal{A}\}$.

$(\bar{X}, X)$ = Set of **reverse arcs of the cut** $[X, \bar{X}] = \{(i, j) : i \in \bar{X}, j \in X$, and $(i, j) \in \mathcal{A}\}$.

Set of forward arcs of the cut by themselves have the **Chain breaking property**, i.e., if they are all removed, there will be no chain from any node in X to any node in $\bar{X}$. If all reverse arcs are also removed, there will be no path between any node in X and any node in $\bar{X}$.

4.4 : $s - t$ Cuts, or Cuts separating source node s and sink node t: In study of single commodity flow problems

with single source node s and single sink node t, these special cuts play a significant role. They are $[X, \bar{X}]$ with $s \in X$, $t \in \bar{X}$.

In network below, $[\{1,2\}, \{3 \text{ to } 8\}] = \{(2,3),(2,6)\} \cup \{(4,2)\}$

$[\{1,2,3,7,4,5\}, \{6,8\}] = \{(2,6),(7,8)\}$

If all forward arcs in an $s-t$ cut are removed, there remains no chain from s to t, hence no flow is possible in remaining network from s to t.

4.5: Capacity of a cut $[X, \bar{X}]$: in $G = (\mathcal{N}, \mathcal{A}, \ell, k)$ is defined to be:

$k(X, \bar{X}) - \ell(\bar{X}, X) = \Sigma_{(i,j)\in(X,\bar{X})}\, k_{ij} - \Sigma_{(i,j)\in(\bar{X},X)}\, \ell_{ij}$

Cuts play a role dual to that of path in network algorithms.

5: Forests and Trees

5.1: Forest: is a partial subnetwork that contains no cycles.

5.2: Tree: is a connected partial subnetwork that contains no cycle.

Each connected component in a forest is a tree.

5.3: Spanning Tree: A tree that includes all the nodes in the network, i.e., it is a subnetwork that is a tree.

5.4: Trivial Tree: A single isolated node by itself.

5.5: Terminal node, End node, Pendant node, or Leaf node: in a tree is a node that has only one line of the tree incident at it.

5.6: Leaves: of a tree are its leaf nodes.

5.7: Leaf arc, leaf edge: The arc or edge of a tree that is incident to a leaf node.

5.8: In-tree arc or edge: Given a tree T in G, this is an arc or edge in G that is in T.

5.9: Out-of-Tree arc or edge: Given a tree T in G, this is an arc or edge in G that is not in T.

5.10: Cotree: Set of out-of-tree arcs corresponding to a spanning tree.

5.11: Rooted tree: A tree with one of its nodes identified as its **root**.

We always think of a rooted tree as hanging down from its

root. This defines a unique **predecessor-successor (parent-child)** relationship among the nodes of the tree.

5.12: Fundamental cutsets: Let T be a spanning tree in $G = (\mathcal{N}, \mathcal{A})$.

Each in-tree line defines a unique cutset in G called the **Fundamental cutset** of that in-tree line.

After deleting the in-tree line if we add any line in its fundamental cutset, we again get another spanning tree.

5.13: Fundamental cycles: Let T be a spanning tree in $G = (\mathcal{N}, \mathcal{A})$.

Each out-of-tree line defines a unique cycle in G called the **Fundamental cycle** of that out-of-tree line.

After adding the out-of-tree line if we delete any line in its fundamental cycle, we again get another spanning tree.

Properties of nontrivial Trees

a) At least two leaf nodes.

b) If it has n nodes, it has $n - 1$ lines.

c) It has a unique simple path between any pair of nodes in it.

5.14: Tree labels: Let T be a rooted spanning tree in $G = (\mathcal{N}, \mathcal{A})$.

Then T can be stored by storing the **Tree labels** (**Predecessor indices, successor indices, younger and elder brother indices, thread label.**

5.15: Predecessor path of a node in a rooted tree: is the unique path from that node to the root. Can be traced using only the predecessor indices beginning with that node.

5.16: Ancestors or predecessors of a node in a rooted tree: Nodes on its predecessor path.

5.17: Descendents or successors of a node in a rooted tree: All nodes for which this node is an ancestor.

I) How to find fundamental cycle of an out-of-tree arc using tree labels?

II) How to find fundamental cutset of an in-tree arc using tree labels?

5.17 In-tree arc directed towards (*away from*) root node: On each in-tree arc, one node is parent and the other its child. in-tree Arc (i, j) is said to be directed towards (*away from*) root node if $\text{son}(i, j) = i$ ($Son(i, j) = j$).

5.18: Thread Labels: A $1-1$ correspondence $\mathcal{N} \to \mathcal{N}$ denoted $t(i)$ for $i \in \mathcal{N}$; i.e., $\{t(1), \ldots, t(n)\} = \mathcal{N}$; satisfying property:

PROPERTY: If node i has $NS(i)$ descendents, then its set of descendents is: $\{t(i), t^2(i), \ldots, t^{NS(i)}(i)\}$.

Commonly used thread label is:

$$t(i) = \begin{cases} \text{eldest son of } i \text{ if } S(i) \neq \emptyset \\ \\ YB(i) \text{ if } S(i) = \emptyset \text{ but } YB(i) \neq \emptyset \\ \\ \text{if } YB(i) = S(i) = \emptyset, \text{ it's } YB(j),\ j \text{ 1st ancestor of } i \\ \text{with a YB} \\ \\ \text{root node, otherwise.} \end{cases}$$

5.19: Tree Growth subroutines in Network Algorithms

Many network algorithms have subroutines to **Grow Trees**. In these subroutines nodes in network will be in two classes called *Labeled, unlabeled*. Usually these subroutines consist of following steps.

TREE PLANTING STEP: Plant a tree at a *suitable node*, i.e., select a suitable (this is defined by the algorithm) node as the root node of a tree, and label it with $\emptyset$. The root is now labeled, all other nodes are unlabeled.

GENERAL TREE GROWTH OR BRANCHING STEP: Select a suitable out-of-tree line joining a labeled node (i say), to an unlabeled node (j say). Label j with i as its predecessor index. This makes the line joining these two nodes an in-tree line (it is called *the line used to label* j) and j an in-tree node.

The subroutine repeats the branching step as often as possible. At any stage:

Set of all in-tree nodes = set of labeled nodes.

Set of in-tree lines = set of lines, each joining a labeled node to its PI.

Structure being grown is always a tree because

- Connectivity is always maintained
- Always, no. of lines = no. of nodes − 1.

5.20: Methods for selecting a spanning Tree in G

Primarily based on above tree growth subroutine. Two particular implementations called **BrFS (Breadth First Search), DFS (Depth First Search)** are important. Here nodes undergo two processes, the first is *labeling* as in the above algorithm, and each labeled node then undergoes a *scanning* operation.

BrFS: Nodes can be in 3 states, *unlabeled, labeled and unscanned, labeled and scanned.* At any stage **LIST** = set of labeled and unscanned nodes, maintained as a Queue, i.e., newly labeled nodes are added at the top of the list, and labeled nodes to scan are taken from the bottom of the list, i.e. using the FIFO queue discipline.

TREE PLANTING STEP: Same as in 5.19 above.

GENERAL STEP: a) If no unlabeled nodes, tree is a spanning tree, terminate.

b) If there are some unlabeled nodes, but list $= \emptyset$, G not connected. Present tree is spanning in connected component contain-

ing root node, terminate.

c) If list $\neq \emptyset$, delete a node from its bottom, say i to scan.

SCANNING NODE i: Find all unlabeled nodes j joined to i by a line. Label each of these nodes j with i as their PI, and include them at the top of the list. Go to next step.

THEOREM: In the BrFS spanning tree, for each $i \in \mathcal{N}$, the predecessor path of i is the shortest path between node i and the root node, in terms of the number of lines.

6: Bipartite networks

$G = (\mathcal{N}, \mathcal{A})$ is said to be **bipartite** if there exists a partition on $\mathcal{N}$ as $(\mathcal{N}_1, \mathcal{N}_2)$ (called a **bipartition** for G) s. th. every line in $\mathcal{A}$ joins a point in $\mathcal{N}_1$ to a point in $\mathcal{N}_2$.

THEOREM: A network $G = (\mathcal{N}, \mathcal{A})$ is bipartite iff it contains no **odd cycles**.

7: Acyclic networks:

A directed network that contains **no circuits** (it may contain cycles).

THEOREM: A directed network $G = (\mathcal{N}, \mathcal{A})$ is acyclic iff its nodes can be numbered s. th. for each $e \in \mathcal{A}$

number of $tail(e)$ < number of $head(e)$.

Such a numbering is called **acyclic numbering** or **topological ordering**.

8: Node-Arc Incidence Matrices:

Defined for directed networks with no self-loops. $G = (\mathcal{N}, \mathcal{A})$ with $|\mathcal{N}| = n, |\mathcal{A}| = m$.

$E_{n \times m}$ with each row corresponding to a node, and each col. corresponding to an arc in G. Entry in row of node i and col. of arc e is:

$$\begin{array}{rl} 1 & \text{if } i \text{ is tail of } e \\ -1 & \text{if } i \text{ is head of } e \\ 0 & \text{otherwise} \end{array}$$

8.1: Triangular matrices: Concept defined only for square matrices. $D_{n\times n} = (d_{ij})$ is said to be:

Upper triangular	if D nonsingular, and $d_{ij} = 0 \quad \forall i > j$
Lower triangular	if D nonsingular, and $d_{ij} = 0 \quad \forall i < j$
Triangular	if nonsingular and can be made lower triangular by permuting its cols./rows.

Algo. to Check triangularity: To check if a square matrix is triangular:

An element of a matrix is said to be a **single nonzero entry** if it is nonzero, and it is the only nonzero entry either in its row or in its col. or both.

1. Matrix must have a *single nonzero entry*.

2. Submatrix obtained from matrix by striking off the row and column of a *single nonzero entry* must again satisfy 1. Same process continues unitl all rows and columns of matrix are struck off.

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

THEOREM: E is the node-arc incidence matrix of a directed network G. Then every nonsingular square submatrix of E is **triangular**.

COROLLARY: Determinant of every square submatrix of E is 0, or ± 1, i.e., E is **TU (Totally Unimodular)**.

THEOREM: If G connected, rank of E is $n-1$. Any row of E can be struck off to make remaining matrix of full row rank.

THEOREM: Let $\bar{E}$ be the $(n-1) \times m$ matrix obtained by striking off a row (say that of node n) from E.

a) Let B be a basis for $\bar{E}$. Then cols. of B correspond to arcs in a spanning tree of G and vice versa.

b) For any RHS vectors d, c, the equations $By = d$, $\pi B = c$ can both be solved by **backsubstitution**, and the solutions are of form: $(y_j), (\pi_i)$ where $y_j = \Sigma_{t=1}^{n-1} \alpha_t d_t$, $\pi_i = \Sigma_{t=1}^{n-1} \beta_t c_t$ in which all α_t, β_t are $0, \pm 1$.

Hence if d, c are integer vectors, the solutions y, π are also integer vectors.

8.2: Integer Property: Let $A_{p\times q}$, and let $K(b)$ be set of feasible sols. of $Ax \leq b, \quad x \geq 0$.

Then following 3 conds. are equivalent.

i) A is TU

ii) $\forall$ integral b, all extreme pts. of $K(b)$ are integral.

iii) Every nonsingular square submatrix of A has an integer inverse.

8.3: Fundamental cycle−Arc incidence matrices, In-tree arc−Fundamental cycle incidence matrices

Let T be a spanning tree in G with in-tree arcs $e_1, \ldots, e_{n-1}$. Out-of-tree arcs are: $e_n, \ldots, e_m$. Orient fundamental cycle of each out-of-tree arc s. th. the out-of-tree arc is a forward arc in it.

For $t = 1$ to m, $p = n$ to m define:

$$\lambda_{tp} = \begin{cases} 0 & \text{if } e_t \notin \text{funda. cycle of } e_p \\ +1 & \text{if } e_t \text{ reverse on this cycle} \\ -1 & \text{if } e_t \text{ forward on this cycle} \end{cases}$$

For $p = n$ to m,

Incidence vector of funda. cycle of $e_p = (\lambda_{1p}, \ldots, \lambda_{mp})$.

Funda. cycle-arc incidence matrix of G WRT T is:

$$L_{(m-n+1)\times m} = \begin{pmatrix} \lambda_{1n} & \dots & \lambda_{m,n} \\ \lambda_{1,n+1} & \dots & \lambda_{m,n+1} \\ \vdots & & \vdots \\ \lambda_{1m} & \dots & \lambda_{mm} \end{pmatrix}$$

The in-tree arc−Funda. cycle incidence matrix for G WRT T is:

$$\lambda_{(n-1)\times(m-n+1)} = \begin{pmatrix} \lambda_{1n} & \dots & \lambda_{1m} \\ \vdots & & \vdots \\ \lambda_{n-1,n} & \dots & \lambda_{n-1,m} \end{pmatrix}$$

For $t = 1$ to m, $E_{.t}, \bar{E}_{.t}$ are the col. vectors of $E, \bar{E}$ corresponding to arc e_t.

RESULT: For $p = n$ to m

$$\begin{aligned} E_{.p} &= \sum_{t=1}^{n-1} \lambda_{tp} E_{.t} \\ \bar{E}_{.p} &= \sum_{t=1}^{n-1} \lambda_{tp} \bar{E}_{.t} \end{aligned}$$

So, $(\lambda_{1p}, \ldots, \lambda_{n-1,p})$ is the vector of coefficients in the representation of node-arc incidence col. of e_p, as a linear comb. of node-arc incidence cols. of in-tree arcs.

RESULT:

$$\bar{E} = \begin{pmatrix} \text{in-tree} & & \text{out-of-tree} \\ B & \vdots & D \end{pmatrix}$$

Then $B_{(n-1)\times(n-1)}$ is a basis for $\bar{E}$. When we multiply $\bar{E}$ on left by B^{-1} we get $B^{-1}\bar{E} = (I \ \vdots \ B^{-1}D) = (I \ \vdots \ \lambda)$.

This result is named **Dantzig Property**.

8.4: Vertex-Edge incidence matrix of an undirected network

$G = (\mathcal{N}, \mathcal{A})$ umdirected with $|\mathcal{N}| = n$, $|\mathcal{A}| = m$. The **Vertex-Edge incidence matrix of** G is the $n \times m$ matrix with: a_{it} = entry in this matrix in row of node i and column of edge e_t given by

$$a_{it} = \begin{cases} 1 & \text{if } i \text{ on } e_t \\ 0 & \text{otherwise} \end{cases}$$

Write the node-edge incidence matrix of the network for Konigsberg bridges.

Verify that the determinant of the node-edge incidence matrix of any odd simple cycle is -2.

THEOREM: The node-edge incidence matrix of an undirected network is TU iff network is bipartite.

9: Single commodity flow models

$G = (\mathcal{N}, \mathcal{A}, \ell, k, s, t)$ Directed. Two ways of representing flows in G.

9.1: Node-Arc flow models: One variable associated with each arc in the network.

f_{ij} = amount shipped across (i, j) from tail node i to head node j.

So, in this model, we treat flow as it takes place one arc at a time. $f = (f_{ij})$ called **node-arc flow vector** has to satisfy following conditions for feasibility.

Flow conservation: 1. At every transit node i , $f(\mathcal{N}, i) = \Sigma_{j \in B(i)} f_{ji}$ = total flow coming to i thro' arcs in its reverse star = $f(i, \mathcal{N}) = \Sigma_{j \in A(i)} f_{ij}$ = total flow going out of i thro' arcs in its forward star.

2. Value of flow vector f = net flow going out of source node = net flow reaching sink node, denoted by v.

Bounds: f_{ij} honors the lower bound and capacity for flow of arc (i, j).

So, these cods. are:

$$\begin{aligned} f(s, \mathcal{N}) - f(\mathcal{N}, s) &= v \\ f(i, \mathcal{N}) - f(\mathcal{N}, i) &= 0, \quad i \neq s, t \\ f(t, \mathcal{N}) - f(\mathcal{N}, t) &= -v \end{aligned}$$

$$\ell \leq f \leq k$$

This model looks strange for physical shipping, because trucks usually loaded at source, and never unload until they reach sink.

2nd type of model called **Arc-Chain flow model** gives much more realistic portrayal of physical shipping.

But in many network models, each arc (i, j) represents a channel for transporting intermediate product at some stage represented by node i to next processing operation represented by node j (e.g., chair-making example). In these cases, material actually unloaded after traveling thro' each arc (i, j), processed, then sent along another arc, and continues same manner on its way to becoming finished product. Node-arc flow model directly applicable to these situations. As most network applications are of this type, we will concentrate on node-arc flow model.

Also, any solution of node-arc flow type, can be converted into the arc-chain flow type. Hence, we will see that node-arc flow model is convenient to study physical shipping problems too.

9.2: Min cost flow problems with exogenous flows

Let E be node-arc incidence matrix of directed $G = (\mathcal{N}, \mathcal{A}, \ell, k, c, V)$ with $c =$ cost vector for arc flows, $V = (V_i) =$ vector of exogenous flow amounts at the nodes. The model is:

$$\begin{aligned} & \min cf \\ \text{s. to} \quad & Ef = V \\ & \ell \leq f \leq k \end{aligned}$$

Since sum of row vectors of E is 0, we have a

Necessary cond. for existence of a feasible flow vector

$$\sum_{i \in \mathcal{N}} V_i = 0$$

9.3: The Arc-Chain flow model

Consider shipping maximum possible amount from s to t in directed flow network $G = (\mathcal{N}, \mathcal{A}, 0, k, s, t)$.

Make a list of different chains from s to t in G. Suppose these are:

$\mathcal{C}_h$, $h = 1$ to p. Let x_h denote amount shipped from s to t along the chain $\mathcal{C}_h$.

$x = (x_h) = (x_1, \ldots, x_p)$ is called the **Arc-Chain flow vector**. Its **Value** = amount reaching t from s in it $= \Sigma_{h=1}^{p} x_h$.

So, arc-chain formulation of maximum flow problem is:

$$\begin{array}{lll} \max & \Sigma_h x_h & \\ \text{s. to} \quad \Sigma(x_h : \text{over } h \quad \text{s. th. } (i,j) \in \mathcal{C}_h) & \leq k_{ij} & \forall (i,j) \in \mathcal{A} \\ x_h & \geq 0 & \forall h. \end{array}$$

HOW TO GET A NODE-ARC FLOW VECTOR CORRESPONDING TO A GIVEN ARC-CHAIN FLOW VECTOR?

Let $x = (x_h)$ be an arc-chain flow vwctor of value v. Define:

$$f_{ij}(x) = \Sigma_h \quad {}_{s.th.} \quad {}_{(i,j)\in\mathcal{C}_h} \, x_h, \qquad \forall (i,j) \in \mathcal{A}$$

Let $f(x) = (f_{ij}(x))$. Verify $f(x)$ is a feasible node-arc flow vector of same value as x, it is called the node-arc flow vector corresponding to the arc-chain flow vector x.

HOW TO GET AN ARC-CHAIN FLOW VECTOR CORRESPONDING TO A GIVEN NODE-ARC FLOW VECTOR?

THEOREM: Let $\bar{f} = (\bar{f}_{ij})$ be a feasible node-arc flow vector of value $\bar{v}$ in G. If $\bar{v} > 0$, there exists a chain from s to t such that $\bar{f}_{ij} > 0$ on all arcs (i, j) on that chain.

10: Flow Augmenting Paths (FAP)

Let f be a feasible flow vector in directed single commodity flow vector $G = (\mathcal{N}, \mathcal{A}, \ell, k, s, t)$.

A path $\mathcal{P}$ from s to t in G is said to be a **flow augmenting path (FAP) WRT** f if it satisfies:

$$f_{ij} \begin{cases} < k_{ij} & \text{for } (i,j) \text{ forward on } \mathcal{P} \\ > \ell_{ij} & \text{for } (i,j) \text{ reverse on } \mathcal{P} \end{cases}$$

The **Residual capacity** $\epsilon > 0$ of this FAP is defined to be:

$$\min\{k_{ij} - f_{ij} : (i,j) \text{ forward on } \mathcal{P}\} \cup \{f_{ij} - \ell_{ij} : (i,j) \text{ reverse on } \mathcal{P}\}$$

Given an FAP WRT f with residual capacity ϵ, the **flow augmentation step using it** generates the new flow vector $\bar{f} = (\bar{f}_{ij})$ where:

$$\bar{f}_{ij} = \begin{cases} f_{ij} & \text{if } (i,j) \notin \mathcal{P} \\ f_{ij} + \epsilon & \text{if } (i,j) \text{ forward on } \mathcal{P} \\ f_{ij} - \epsilon & \text{if } (i,j) \text{ reverse on } \mathcal{P} \end{cases}$$

$\bar{f}$ is feasible and its value is ϵ + value of f.

Verify that $\mathcal{P}$ is not an FAP WRT $\bar{f}$.

Augmenting Path Methods: Those based on FAPs for solving network flow problems.

10.1: Flow Augmenting Chain (FAC) from s to t WRT f: is an FAP from s to t WRT f in which all arcs are forward arcs.

Maximal or Blocking flow in G: is a feasible flow vector WRT which no FAC from s to t exists.

Maximum flow in G: is a feasible flow vector in G associated with the maximum possible value.

10.2: Saturated arc: In a flow vector $f = (f_{ij})$ in G, an arc (i, j) is said to be **saturated** if $f_{ij} = k_{ij}$.

10.3: An approach for finding a maximum flow in $G = (\mathcal{N}, \mathcal{A}, 0, k, s, t)$

1. Begin with $f = 0$.

2. Let $\bar{f}$ be current flow vector. Delete all saturated arcs from G from further consideration. Let $\bar{G}$ denote remaining network.

Find a chain from s to t in $\bar{G}$. If none, terminate with $\bar{f}$ as the answer to the problem.

If a chain $\mathcal{C}$ from s to t in $\bar{G}$ is found, augment $\bar{f}$ using it as an FAC. Repeat this Step 2 with the new flow vector.

11: Reduced Cost Coefficients

Consider min cost flow problem in directed single commodity flow network $G = (\mathcal{N}, \mathcal{A}, \ell, k, c, V)$ with $|\mathcal{N}| = n, |\mathcal{A}| = m$, and E as the node-arc incidence matrix. The problem is:

$$\begin{aligned} & \min cf \\ \text{s. to} \quad & Ef = V \\ & \ell \leq f \leq k \end{aligned}$$

Dual problem has variables $\pi_i, \quad i \in \mathcal{N}$, called **Node potentials or prices**. Given node price vector $\pi = (\pi_i)$, the **relative** or **reduced cost coefficient of arc** (i, j) **WRT** π is:

$$\bar{c}_{ij} = c_{ij} - (\pi_j - \pi_i)$$

THEOREM: Minimizing cf or $\bar{c}f$ in above problem leads to the same set of optimal flows. So, can replace c by $\bar{c}$ in above min cost flow problem.

THEOREM: Let $\mathcal{P}$ be a path from p to q in G; and C any oriented cycle. Define: cost of $\mathcal{P}$ or C to be = (sum of cost coeffs. of forward arcs) − (sum of cost coeffs. of reverse arcs). Then:

Cost of $\mathcal{P}$ WRT $\bar{c}$ = (Cost of $\mathcal{P}$ WRT c) + $(-\pi_p + \pi_q)$.

Cost of C WRT $\bar{c}$ = Cost of C WRT c.

12: Residual cycles

Let $f = (f_{ij})$ be a (not necessarily feasible) flow vector satisfying the bound conds. on all the arcs in $G = (\mathcal{N}, \mathcal{A}, \ell, k, c, V)$. An oriented cycle C in G is said to be a **Residual cycle WRT** f if:

$$f_{ij} < k_{ij} \quad \text{on all forward arcs of } C$$
$$f_{ij} > \ell_{ij} \quad \text{on all reverse arcs of } C$$

The **capacity of residual cycle** C is defined to be:

$$\min\{k_{ij} - f_{ij} : (i,j) \text{ forward on } C \} \cup \{f_{ij} - \ell_{ij} : (i,j) \text{ reverse on } C \}$$

13: Residual arcs; Residual Networks $G(f)$, $G(f, \pi)$

Let $f = (f_{ij})$ be a flow vector (not necessarily feasible) satisfying the bound constraints in $G = (\mathcal{N}, \mathcal{A}, \ell, k, c, V)$.

$(i, j) \in \mathcal{A}$ called a **residual arc WRT** f if $f_{ij} < k_{ij}$.

If $(i, j) \in \mathcal{A}$, and $f_{ij} > \ell_{ij}$, then we say that the arc (j, i) (it may not exist in $\mathcal{A}$) is a **residual arc WRT** f.

The **residual network** $G(f)$ **of** G **WRT** f is an anciliary network $(\mathcal{N}, \mathcal{A}(f), 0, \kappa, c')$ where $\mathcal{A}(f)$ consists of the arcs defined by following rules 1 and 2, with κ, c' defined there.

1. $\forall (i, j) \in \mathcal{A}$ with $f_{ij} < k_{ij}$, include arc (i, j) in $\mathcal{A}(f)$ with a $+$ **label**, lower bound 0, capacity $\kappa_{ij} = k_{ij} - f_{ij}$, and cost coeff. $c'_{ij} = c_{ij}$.

2. $\forall (i, j) \in \mathcal{A}$ with $f_{ij} > \ell_{ij}$, include an arc (j, i) in $\mathcal{A}(f)$ with a $-$ **label**, lower bound 0, capacity $\kappa_{ji} = f_{ij} - \ell_{ij}$, and cost coeff. $c'_{ji} = -c_{ij}$.

$G(f)$ very useful to determine new flow vectors to which we can move from f maintaining bound feasibility. Every FAP from s to t WRT f corresponds to a chain from s to t in $G(f)$ and vice versa.

$(p, q) \in \mathcal{A}(f)$ corresponds to $(p.q)$ $[(q, p)]$ in $\mathcal{A}$ if its label is $+$ $[-]$.

The residual network $G(f, \pi)$ same as $G(f)$ with exception that cost coeffs. are determined based on reduced cost coeffs. $\bar{c}$ rather than original cost coeffs. c.

14: Outtree, Intree

Outtree [**Intree**] are rooted tree in which every arc is directed away from [towards] the root node.

15: Matchings, perfect matchings

Usually defined in undirected networks. A **matching** or **1-factor** is a set of edges that contains at most one edge incident at any node. In figure set of wavy edges is a matching.

A **perfect matching** is a set of edges that contains exactly one edge incident at each node. Set of red edges in figure is a perfect matching.

16: Assignments

Let G be a bipartite network with bipartition $(\{R_1, \ldots, R_n\}, \{C_1, \ldots, C_n\})$. A perfect matching in G is called an **assignment**.

We will represent this bipartite network by the $n \times n$ array with cell (i, j) in the array corresponding to edge (R_i, C_j). So, an assignment is a $n \times n$ $0 - 1$ matrix satisfying:

$$\begin{aligned} \sum_{j=1}^{n} x_{ij} &= 1, \quad i = 1 \text{ to } n \\ \sum_{i=1}^{n} x_{ij} &= 1, \quad j = 1 \text{ to } n \\ x_{ij} &\geq 0, \quad \forall i, j \end{aligned} \tag{1}$$

$$x_{ij} \in \{0, 1\} \quad \forall i, j \tag{2}$$

A feasible solution x to (1) is called a **doubly stochastic matrix**.

An assignment is a $0 - 1$ doubly stochastic matrix. The cells with "1" entries in an assignment are called its **allocated cells**.

17: Transportation problem

Bipartite min cost flow problem. Every node is either a source or a sink. Bipartition is (sources, sinks).

Figure 1.36 Bipartite network representation of the transportation problem. Data on each arc is its lower bound, capacity, and cost per unit flow. Exogenous flow amounts are entered by the side of the nodes.

18: Assignment problem

A transportation problem in which

number of sources = number of sinks

all sources have 1 unit supply, and all sink demands are 1 unit.

EXAMPLE: Corporation introducing new product. 4 marketing zones, each needs marketing director. 4 candidates selected. Company estimates of sales generated given in table, depending on which candidate is appointed in each zone. Assign candidates to zones to maximize total annual sales. This is a special bipartite minimum cost flow problem, or problem of finding a **min cost perfect matching** in a bipartite network.

	Expected annual sales in zone, if candidate assigned to zone.			
Zone →	1	2	3	4
Candidate 1	90	85	139	73
2	60	130	200	112
3	60	130	200	112
4	111	88	128	94

19: Transhipment problem

A min cost flow in a non-bipartite directed network. May have transit nodes, arcs joining pairs of source nodes, or pairs of sink nodes.

Figure 1.37 Numbers on nodes corresponding to plants and packing units are node capacities per day; and those on sales nodes are requirements per day.

Node corresponding to plant 2 is a source , but it can also receive flow from other source nodes like plant 1. That's why

problems like this are called *transshipment problems*.