# CO351: Network Flow

Felix Zhou

Fall 2019, University of Waterloo
from Kazuhiro Nomoto's Lectures

# Contents

# 1 Preface

## 1.1 Assignments

10, with the lowest 2 dropped.

## 1.2 Midterms

2 each worth potentially 15% each potentially in EV3

## 1.3 Final Exam

worth either 55% or 70%

# 2 Graph Theory

## 2.1 Undirected Graphs

**Definition 2.1.1 (Graph)**
$G = (V, E)$ set of vertices and unordered pair of elements of $V$ called edges.

**Definition 2.1.2**

- degree
- walk
- path
- cycle
- connected

**Definition 2.1.3 (Cut)**
Let $G = (V, E)$ be a graph, $S \subseteq V$.
The cut $\delta(S)$ induced by $S$ is the set of all edges with exactly one vertex in $S$

If $s, t \in V$ and $s \in S, t \notin S$ then $\delta(S)$ is an $st$-cut.

**Theorem 2.1.1**
There is an $st$-path if and only if every $st$-cut is non-empty.

**Definition 2.1.4**

- tree
- forest
- spanning-tree

**Proposition 2.1.2**

- a tree with $n$ vertices has exactly $n-1$ edges.
- if $T$ is a tree and $u, v \in T$ are NOT adjacent in $T$
  1. $T + uv$ has exactly one cycle $C$
  2. if $xy \in C$ then $T + uv + xy$ is a tree

## 2.2 Directed Graphs (Di-Graphs)

**Definition 2.2.1 (Di-Graph)**
$D = (N, A)$ set of nodes and ordered pair of elements of $N$ called arcs.

**Definition 2.2.2**

- tail
- head
- out-degree
- in-degree
- di-walk
- di-path
- di-cycle

**Definition 2.2.3 (Cut)**
Let $D = (N, A)$ be a digraph and $S \subseteq N$.
The cut induced by $S$ is

$$\delta(S) := \{uv \in A : u \in S, v \notin S\}$$

We can also define

$$\delta(\bar{S}) := \{uv \in A : u \notin S, v \in S\}$$

**Definition 2.2.4**
If $s \in S, t \notin S$ then $\delta(S)$ is an $s,t$-cut.

**Theorem 2.2.1**
Let $D = (N, A)$ be a digraph and $s, t \in N$.
There is an $s, t$ di-path if and only if every $s, t$-cut is non-empty.

**Proof ( $\Longrightarrow$ )**
Any $s, t$-cut intersects any $s, t$-path.

**Proof ( $\Longleftarrow$ )**
Consider $S$, the set of all nodes such that for every $v \in S$, there is an $s, v$ di-path.

If $t \in S$, we are done.

Else $\delta(S)$ is an $s, t$-cut and must be non-empty. But then we can extend a path in $S$ with the arcs in $\delta(S)$ and it should belong in $S$ in the first place.

This is the desired contradiction.

## 2.3 Underlying Graph

**Definition 2.3.1 (Underlying Graph)**
of $D = (N, A)$ is the undirected graph $G = (N, E)$ where $E$ is the set of arcs of $D$ but with directions removed.

**Definition 2.3.2 (Cycle in Digraph)**
A cycle in the underlying graph.

By fixing an orientation of the cycle, arcs in the direction of the orientation are

**Definition 2.3.3 (Forward Arcs)**

and otherwise

**Definition 2.3.4 (Backward Arcs)**

**Definition 2.3.5 (Connected)**
A digraph is connected if the underlying graph is connected.

**Definition 2.3.6 (Strongly Connected)**
Let $D = (N, A)$ be a digraph. It is strongly connected if for all $s, t \in N$ there is both a $s, t$ di-path and a $t, s$ di-path.

**Definition 2.3.7 (Spanning Tree)**
of a directed graph is a spanning tree of the underlying graph.

# 3 Transshipment Problem (TP)

Let $D = (N, A)$, node demand $b \in \mathbb{R}^N$, and arc costs $w \in \mathbb{R}^A$.

We generally assume $\sum b_i = 0$, and that $D$ is connected.

A solution is a flow $x \in \mathbb{R}^A$.

We wish to minimize $w^T x$ such that

$$\sum_{iv \in A} x_{iv} - \sum_{vj \in A} x_{vj} = b_v$$

In other words, the in-flow less the out-flow is the demand of the node.

Given a a vector (flow) $x$, and $A' \subseteq A$ define

$$x(A') = \sum_{e \in A'} x(e)$$

## 3.1 LP Formulation of TS

$\min w^T x$ such that $x \geq 0$ and

$$x(\delta(\bar{v})) - x(\delta(v)) = b_v$$

for all $v \in N$

We can write the constraints in matrix form with a $|N| \times |A|$ matrix such that

**Definition 3.1.1 (Incidence Matrix)**

$$a_{v,ij} = \begin{cases} -1, & i = v \\ 1, & j = v \\ 0, & \text{else} \end{cases}$$

Note that each arc "appears" twice, once as head and once as tail in the matrix.

## 3.2 dual LP for TP

There is one constraint in the LP for every node. So we have one dual variable for each node $y \in \mathbb{R}^N$.

The dual is a maximization problem so the objective function is

$$\max y^T b$$

the constraints are such that for each arc $uv \in A$

$$y_v - y_u \leq w_{uv}$$

note that $y$ is free since the original LP is an equivalence LP.

## 3.3 Complementary Slackness Condition

**Theorem 3.3.1**
$x, y$ are optimal solutions for the primal and dual if and only if

$$x_{uv} = 0 \vee y_v - y_u = w_{uv}$$

for each $uv \in A$.

Note that since we have equality constraints in the primal, there are no complementary slackness conditions in the dual.

If a flow is optimal, then the Complementary Slackness Conditions say that there must be a $y$ such that $x_{uv} = 0$ or

$$y_v - y_u = w_{uv}$$

for each $uv \in A$.

If the $y$ is then feasible (for all other non-active conditions, the constaints still holds), then we have verified optimality of the solution for the primal.

## 3.4 Simplex Applied to TP

### 3.4.1 Linear Algebra

What is a basis?

Consider an incidence matrix, we have $|N|$ rows. If we add all the rows of the incidence matrix, we get the zero vector, meaning that the rows are linearly dependant.

So the rank of the incidence matrix is at most $|N| - 1$. We need to choose at most $|N| - 1$ linearly independent columns.

We now show that if $D = (N, A)$ is connected, the rank is precisely $|N| - 1$.

From now on, we will assume $D$ is connected.

**Proposition 3.4.1**
The columns of the incidence matrix corresponding to a cycle are linearly dependent.

Recall for an (undirected) cycle $v_0, v_1, \ldots, v_{k-1}$ in a digraph $D = (N, A)$, an arc of the form $v_i v_{i+1}$ are forward arcs and $v_{i+1} v_i$ are backward arcs.

**Proof**
Let $C$ be a cycle and let $M$ be its incidence matrix.

Let $F, B$ be the forward, backward arcs of $C$ respectively.

Now, suppose $M'$ is obtained by multiplying columns of arcs in $B$ by $-1$.

For each node $v, d(v) = d(\bar{v}) = 1$. So for each row, representing $v \in M'$, there is only one entry with 1, only one with $-1$.

So the sum of the entries of each row is 0. Hence, the sum of the columns of $M'$ is 0.

This shows that the sum is a linear combination of columns of $M$. So $M$ is linearly dependent.

**Proposition 3.4.2**
A set of linearly dependent columns of the incidence matrix must include a cycle.

**Proof**
Assignment 3.

Note that a basis consists of a spanning tree.

**Theorem 3.4.3**
Let $M$ be the incidence matrix of a digraph $D = (N, A)$.
Then a set of $|N| - 1$ columns of $M$ is a basis if and only if the corresponding arcs form a spanning tree of $D$.

Note that the above shows that $\operatorname{rank} M = |N| - 1$.

### 3.4.2 Finding a Flow

We can find flow based on a tree $T$ by

(1) Solving $A_T x = b_T$

(2) work in an iterative way starting at a leaf

Note that it is entirely possible that a tree does not give a feasible flow. So how do we find an initial BFS?

For now, let us assume we have a feasible BFS. How do we pick an entering variable?

At each step of the Simplex Method, we are finding $y$ such that

$$A_B^T y = w_B$$

where $B$ is basis.

We have an "optimal" solution if

$$w_N - A_N^T y \geq 0$$

If not, there is $uv \in N$ such that

$$w_{uv} - (y_v - y_u) < 0$$

so it is not optimal.

If such an arc exists, have $uv$ enter the basis.

### 3.4.3 The Dual

**Definition 3.4.1 (Node Potential)**
A vector $y \in \mathbb{R}^N$ is called a node potential.

**Definition 3.4.2 (Reduced Cost)**
Given potential $y$, the reduced cost of arc $uv$ is

$$\bar{w}_{uv} = w_{uv} + y_u - y_v$$

**Definition 3.4.3 (Feasible)**
We say a node potential is feasible if

$$\bar{w}_{uv} \geq 0$$

for all $uv \in A$.

### 3.4.4 Tree Flow

**Definition 3.4.4 (Tree Flow)**
A BFS in a TP is a spanning tree with a feasible flow called the tree flow

For a given basis $B$, we calculate $A_B^T y = w_B$ (or $y_u + w_{uv} - y_v = 0$).

If $w_N - A_N^T y \geq 0$, then we are done.

If not, there is $uv \in B$ such that $\bar{w}_{uv} = y_u + w_{uv} - y_v < 0$. Put $uv$ in my basis.

### 3.4.5 Selecting a Leaving Arc

If we add arc $uv$ into the tree $T$, then this creates exactly one cycle $C$.

To ensure that the leaving arc gives a spanning tree, the leaving arc needs to be on $C$.

We want to add a flow of $t$ to the arc $uv$. To make sure that the feasibility conditions are met, any forward arcs in $C$ with respect to $uv$ must have additional flow of $t$. Any backward arc must substract a flow of $t$.

### 3.4.6 How Much Flow to Introduce

Look at all the backward arcs, find the minimum flow and set $t$ to be this value.

### 3.4.7 Full Algorithm

Given $D = (N, A)$ connected, with arc costs $w$, and node demands $b$.

1) Start with a spanning tree $T$ which has a feasible tree flow $x$.
2) Calculate potentials $y$ such that $\bar{w}_{uv} = 0$ when $uv \in T$.
3) Calculate $\bar{w_{uv}}$ for any $uv$ not in $T$.
4) While there exists an arc $uv$ with $\bar{w}_{uv} < 0$
   (a) $T + uv$ has a cycle. Consider the direction of $C$ such that $uv$ is a forward arc.
   (b) Pick $pq$ such that $x_{pq} = \min\{ij : ij \text{ is a backward arc of } C\}$. Let $t = x_{pq}$.
   (c) For any forward arc in $C$, add a flow of $t$.
   (d) For any backward arc in $C$, substract a flow of $t$.
   (e) Replace $T$ with $T + uv - pq$.

(f) Go back to 2)

### 3.4.8 Economic Interpretation

Interpretation for dual potentials.

Imagine transporting a set of commodities. The dual potentials represent the prices of these commodities at the nodes. We buy units at the source node, send them over, and sell at sink node.

The reduced cost on an arc $uv$ is such that if it is greater than 0, we do not wish to transport commodities through $uv$ as we are at a deficit. On the other hand, if less than 0, we wish to increase flow as we can make a profit.

### 3.4.9 Unboundedness

In network simplex, unboundedness occurs when no leaving arc can be found, so that you can increase the parameters (entering variable) indefinitely.

Consider a dicycle $C = y_1 \dots y_k y_1$

$$\begin{aligned} y_{v_2} &= y_{v_1} + w_{v_2 v_2} \\ &\dots \\ y_k &= y_{v_1} + w_{v_1 v_2} + \dots + w_{v_k v_1} \end{aligned}$$

So the reduced cost $w_{uv}$ is such that

$$\begin{aligned} \bar{w}'_{v_k v_1} &= y_{v_k} + w_{v_k v_1} - y_{v1} \\ &= (y_{v_1} + \dots) + w_{v_k v_1} - y_{v_1} \\ &= w_{v_k v_1} + \sum_{i=1}^{k-1} w_{v_i v_{i+1}} \end{aligned}$$

So the sum costs on $C < 0$. Because $v_k v_1$ was chosen as an entering arc since $\bar{w}_{uv} < 0$.

We call $C$ a negative dicycle

**Theorem 3.4.4**
A feasible TP is unbounded if and only if there is a negative dicycle.

**Proof ( $\Longrightarrow$ )**
Assignment.

**Proof ( $\Longleftarrow$ )**
Let $C$ be a negative dicycle so $w(C) < 0$.

Let $x^*$ be a feasible flow and define a flow

$$x^C := \begin{cases} t, & e \in C \\ 0, & e \notin C \end{cases}$$

So $x^C(\delta(\bar{v})) - x^C(\delta(v)) = 0$ for each $v \in N$.

As $x^*$ does not change the net flow on any node

$$x^* + x^C$$

is a feasible flow

$$(x^*+x^C)(\delta(\bar{v}))-(x^*+x^C)(\delta(\bar{v})) = [x^*(\delta(\bar{v})) - x^*(\delta(\bar{v}))]+[x^c(\delta(\bar{v})) - x^c(\delta(v))] = b_v-0 = b_v$$

The objective value however, is

$$w^T(x^* + x^C) = w^T x^* + w^T x^C = w^T x^* + t \cdot w(C) \to \infty$$

as $w(C) < 0$.

### 3.4.10 Finding an Initial BFS

We can apply the same trick and augment $D = (N, A)$ so that there is an extra node with zero $b$-value and arcs of cost 1 with tails in the demand nodes and heads in the supply nodes. All other arc costs are zeroed out.

**Definition 3.4.5**
Given a digraph $D = (N, A)$ and node demands $b \in \mathbb{R}^N$, the auxiliary is given by $D' = (N', A')$ with $b \in \mathbb{R}^{N'}$ and arc costs $w' \in \mathbb{R}^{A'}$.

$$\begin{aligned} N' &= N \cup \{z\} && \text{new node} \\ A' &= A \cup \{vz : v \in N, b(v) < 0\} \cup \{zv : v \in N, b(v) < 0\} \\ b'(z) &= 0, \forall v \in N, b'(v) = b(v) \\ w'(e) &= \begin{cases} 0, & e \in A \\ 1, & e \in A' \setminus A \end{cases} \end{aligned}$$

Note that the auxiliary TP has a natural feasible flow (everything through $z$).

It is bounded below by 0, and so has an optimal solution.

**Theorem 3.4.5**
A TP is feasible if and only if its auxiliary TP has optimal value 0.

### 3.4.11 Feasibility Characterization

Let us assume that our TP is not feasible. This means that the auxiliary TP has an optimal solution with positive objective value. Let $x^*$ be an optimal solution that is a tree.

Let us set $y_z := 0$. For our tree flow, all demand nodes have potential 1, and supply nodes potential $-1$.

This is because any nodes with arcs going into $z$ or out of $z$ are forced to have potential $-1, 1$ respectively. But all other nodes are connected to at least one of these nodes but every non-augmented arc has cost 0, so the node potentials just spread to each other.

Let the sets of such nodes be denoted $S_+, S_-$.

Suppose there is an arc $e$ from $S_-$ to $S_+$. But then the reduced cost is strictly negative which contradicts optimality so no such arcs exist.

$$\bar{e} = 0 - (1 - (-1)) = -2 < 0$$

If $e$ is an arc from $S_+$ to $S_-$ but then the reduced cost is 2 and so it is a non basic arc. In other words

$$x_e = 0$$

So the only arcs in our tree are ones from $S_-$ to $z$, and $z$ to $S_+$.

Restricting our solution to non-zero arcs, we get

$$b(S_-) = \sum_{v \in S_-} b_v < 0$$

is the net flow of $S_-$.

**Theorem 3.4.6**
A TP with digraph $D = (N, A)$ and node demands $b \in \mathbb{R}^n$ is infeasible if and only if there are a set of nodes $S \subseteq N$ such that

$$b(S) < 0$$

and

$$\delta(S) = \varnothing$$

Essentially, there is more supply than demand in this subgraph and no arcs for the supply to leave.

**Proof ( $\implies$ )**
Assume our TP is infeasible. Using the theorem, the auxiliary TP has an optimal solution with positive optimal value.

So there is a tree flow $x$.

Let $y$ be the corresponding feasible potential with $y_z = 0$.

Then we can partition other nodes by whether the feasible potential is $1, -1$. This is possible since we are essentially setting potentials from a tree with root $z$.

We claim that $\delta(S_-) = \varnothing$.

If not, then there is an arc $uv$ so that $u \in S_-, v \in S_+$. But then

$$\bar{w}_{uv} = w_{uv} - y_v + y_u = -2 < 0$$

which means $x$ is not optimal!

We also claim that $x\delta(S_+) = x\delta(\bar{S}_-) = 0$. Indeed, if there is an arc from $S_+ \to S_-$ in our tree, the reduced cost is $2 \neq 0$, meaning it is non-basic.

By definition

$$x\delta(\bar{S}_-) - x\delta(S_-) = \sum_{v \in S_-} [x\delta(\bar{v}) - x\delta(v)] \overset{\text{by feasibility}}{=} \sum_{v \in S_-} b(v) = b(S_-) < 0$$

**Proof ( $\Longleftarrow$ )**
Suppose that there is $S \subseteq N$ where $b(S) < 0$ and $\delta(S) \neq \varnothing$.

Suppose for a contradiction that there is a feasible solution.

$$\begin{aligned} 0 > b(S) &= \sum_{v \in S} b(v) \\ &= \sum_{v \in S} x\delta(\bar{v}) + x\delta(v) && \text{by feasibility} \\ &= x\delta(\bar{S}) - x\delta(S) \\ &= x\delta(\bar{S}) \end{aligned}$$

But $x \geq 0$ so $x\delta(\bar{S}) \geq 0$ which is a contradiction.

# 4 Minimum Cost Flow Problem (MCFP)

Given digraph $D = (N, A)$, node demands $b \in \mathbb{R}^N$, arc costs $w \in \mathbb{R}^A$, and arc capacities $x \in \mathbb{R}^A_+$, our goal is to find a feasible flow satisfying node demands and capacities while minimizing total cost.

## 4.1 LP Formulation

$\min w^T x$ such that

$$\begin{aligned} Mx &= b \\ -Ix &\geq -c \\ x &\geq 0 \end{aligned}$$

$$\begin{aligned} x\delta(\bar{v}) - x\delta(v) &= b_v && v \in N \\ x_e &\leq c_e && e \in A \\ x &\geq 0 \end{aligned}$$

$\min w^T x$ such that

$$\begin{aligned} \begin{bmatrix} M & 0 \\ I_A & I_A \end{bmatrix} \begin{bmatrix} x \\ s \end{bmatrix} &= \begin{bmatrix} b \\ c \end{bmatrix} \\ x, s &\geq 0 \end{aligned}$$

$$\begin{aligned} x\delta(\bar{v}) - x\delta(v) &= b_v && v \in N \\ x_e + s_e &= c_e && e \in A \\ x, s &\geq 0 \end{aligned}$$

### 4.1.1 Dual of MCFP LP

$\max b^T y - c^T z$ such that

$$\begin{bmatrix} M^T & -I \end{bmatrix} \begin{bmatrix} y \\ z \end{bmatrix} \leq w$$

$$\begin{aligned} -y_u + y_v - z_{uv} &\leq w_{uv} && uv \in A \\ z &\geq 0 \end{aligned}$$

### 4.1.2 Rank of Matrix

Consider the incidence matrix which is a part of the matrix of constraints and has rank $|N| - 1$.

We extended this by essentially adding the identity corresponding to the upper bound of constraints into the set so the rank of this matrix is

$$|N| - 1 + |A|$$

This means a basis has $|N| + |A| - 1$ variables. All non-basic variables are set at 0.

We will assume that $c_e > 0$ for all $e \in A$. Otherwise, we can remove that arc.

For each $e \in A$ at least one of $x_e, s_e$ is in the basis. If not, $x_e, s_e$ are non-basic, so they are set to zero. But

$$0 = x_e + x_e = c_e > 0$$

which violats feasibility.

For each arc $e \in A$, we have the following three possibilities

(1) only $x_e$ is in the basis ($s_e$ is not)

(2) only $s_e$ is in the basis ($x_e$ is not)

(3) both $x_e, s_e$ are in the basis

Now let us try to count how many arcs are in case (3).

Let $k$ denote this number. Remark that each arc of this case "takes up" two elements in our basis.

Then $|A| - k$ arcs are in (1), (2), each taking only one element in our basis.

The size of the basis is $|N| + |A| - 1$ so

$$2k + |A| - k = |N| + |A| - 1$$

as there are $2k$ variables in case (3) (generate two elements each) and $|A| - k$ variables in (1), (2).

Hence

$$k = |N| - 1$$

Can the case (3) arcs include a cycle?

No. To see the reason, first notice that $x$ takes a linear combination of the contraint matrix, so each basis element correponds to one column (not injective). We can column reduce the columns of case (3) arcs to an "incidence matrix" with 0s on the bottom rows.

Then there is a cycle if and only if the columns are linearly dependent. But we already know that the incidence matrix has rank $|N| - 1$. Therefore, the $|N| - 1$ arcs in (3) correspond to a spanning tree of $D = (N, A)$.

### 4.1.3 Summary

A tree flow in MCFP consists of a spanning tree $T$ and a feasible flow where all arcs not in $T$ satisfies

$$x_e \in \{0, c_e\}$$

## 4.2 Complementary Slackness Conditions

Recall the LP formulation: $\min w^T x$ such that

$$x\delta(\bar{v}) - x\delta(v) = b_v, x_e \leq c_e, x \geq 0$$

Dual: $\max b^T y - c^T z$ such that

$$-y_u + y_v - z_{uv} \leq w_{uv}, z_{uv} \geq 0$$

(1) $z_{uv} > 0 \implies x_{uv} = c_{uv}$

(2) $-y_u + y_v - z_{uv} < w_{uv} \implies x_{uv} = 0$

The second condition is annoying to check. As before, the reduced cost $\bar{w}_{uv}$ for arc $uv$ is

$$\bar{w}_{uv} = y_u + w_{uv} - y_v$$

The dual constraints become

$$z_{uv} \geq -\bar{w}_{uv}, z_{uv} \geq 0$$

The objective function contains $-c^T z$ where $c > 0$, so maximizing this is the same as minimizing $z$.

In other words, any partial optimal solution $z$ of the dual must satisfy

$$z_{uv} = \max\{0, -\bar{w}_{uv}\}$$

Le us rewrite (1)

$$z_{uv} > 0 \implies x_{uv} = c_{uv}$$

But

$$z_{uv} > 0 \iff \bar{w}_{uv} < 0$$

So we can write

$$\bar{w}_{uv} < 0 \implies x_{uv} = c_{uv}$$

Now consider (2)

$$-y_u + y_v - z_{uv} < w_{uv} \implies x_{uv} = 0$$

But

$$-y_u + y_v - z_{uv} < w_{uv} \iff z_{uv} > -\bar{w}_{uv} \iff \bar{w}_{uv} > 0$$

We end up with

(1) $\bar{w}_{uv} < 0 \implies x_{uv} = c_{uv}$
(2) $\bar{w}_{uv} > 0 \implies x_{uv} = 0$
(3) $0 < x_{uv} < c_{uv} \implies \bar{w}_{uv} = 0$ [contrapositive of (1), (2)]

### 4.2.1 Economic Interpretation

Consider the dual potentials as the prices of goods.

The reduced costs are the cost of buying at $u$, transporting through $uv$, and selling at $v$.

Then $\bar{w}_{uv} < 0$ means we make a profit thorugh $uv$ and send as much flow as possible.

If $\bar{w}_{uv} > 0$, then we lose money through $uv$ and do not send anything.

## 4.3 Network Simplex for MCFP

Same as TP except in selecting entering and leaving arcs.

Look for an arc $e$ with either

(i) $\bar{w}_e < 0, x_e = 0$
(ii) $\bar{w}_e > 0, x_e = c_e$

Then $T + e$ has a cycle so

(1) orient $C$ such that $e$ is a forward arc.
(2) orient $C$ such that $e$ is a backward arc.

Find $t$ that is the minimum of

$$\{c_f - x_f : f \text{ is a forward arc}\}$$

and

$$\{x_f : f \text{ is a backward arc of } C\}$$

Pick $f$ to be an arc in $C$ that is used to pick $t$.

Update flow: add $t$ to forward arcs and subtract $t$ from backward arcs.

Replace T with $T + e - f$.

## 4.4 Feasibility Characterization

To solve the feasibility problem, we use the same auxiliary digraph as before.

The cap for the auxiliary arcs is removed ($\infty$), or we can choose a constant larger than the sum of all supply.

**Theorem 4.4.1**
A MCFP is feasible if and only if its corresponding auxiliary MCFP has optimal value 0.

**Theorem 4.4.2**
A MCFP is infeasible if and only if there exists $S \subseteq N$ such that

$$b(S) > c(\delta(\bar{S}))$$

or

$$b(S) < -c(\delta(S))$$

This means that there is some subset of the nodes with too much demand, more than the sum of the upperbounds for the in arcs.

**Proof ( $\implies$ )**
Suppose our MCFP is infeasible.

Then the auxiliary MCFP has optimal value greater than 0.

Take $x$ to be a corresponding optimal solution.

Set potentials $y$ so that $y_z = 0$.

All other potentials are either $1, -1$.

Define $S_+, S_-$ as before.

Consider $e$ from $S_- \to S_+$ and $f$ from $S_+ \to S_-$.

$$\bar{w}_e = 0 - 1 - 1 = -2 < 0 \implies x_e = c_e$$

and

$$\bar{w}_f = 2 > 0 \implies x_f = 0$$

Consider $S_+$. Its net flow is just the inflow coming into $S_+$ as there are no arcs leaving $S_+$. Conversely, the out flow from $S_-$ are all at capacity, with no in flow.

The optimal solution $x$ to the auxiliary MCFP is feasible, and so satisfies

$$\begin{aligned} b(S_+) &= x(\delta(\bar{S}_+)) - \underbrace{x(\delta(S_+))}_{=0} \\ &= \sum_{u \in S_-, v \in S_+} c_{uv} + \underbrace{\sum_{w \in S_+} x_{zw}}_{\text{positive by assumption}} \\ &> c(\delta(\bar{S}_+)) \end{aligned}$$

with the optimal value of AUX-P being positive so there is an active arc flowing from $z$.

This means

$$b(S_+) > c(\delta(\bar{S}_+))$$

**Proof ( $\Longleftarrow$ )**
Suppose there exists $S \subseteq N$ such that

$$b(S) > c(\delta(\bar{S}))$$

Suppose for a contradiction that there is a feasible flow $x$.

$$\begin{aligned} c(\delta(\bar{S})) &< b(S) \\ &= x(\delta(\bar{S})) - x(\delta(S)) \\ &\leq c(\delta(\bar{S})) - 0 \\ &= c(\delta(\bar{S})) \end{aligned}$$

which is a contradiction.

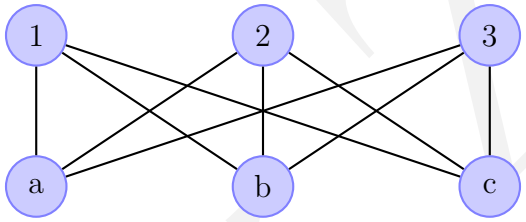
## 4.5 Applications of TP and MCFP

### 4.5.1 Minimum Bipartite Perfect Matching

**Definition 4.5.1 (Bipartite)**
Undirected graph $G(V, E)$ is bipartite if $V = A \cup B$ disjoint and $\delta(A) = E$.

**Definition 4.5.2 (Matching)**
A subset of edges with no common endpoints.

**Definition 4.5.3 (Perfect Matching)**
A matching is perfect if it uses all vertices.




We wish to find a perfect matching in $G$ of minimum total cost. Assume $V = A \cup B$ disjoint with $|A| = |B|$.

Let us formulate this as a MCFP.

(1) direct each edge from $A \to B$.

(2) set the capacity for each arc to be 1.

(3) set

$$b_v = \begin{cases} -1, & v \in A \\ 1, & v \in B \end{cases}$$

(4) arc costs stay the same.

We wish to show a correspondance between optimal solutions to MCFP and MBPM.

**Theorem 4.5.1**
If a MCFP has an optimal solution, and all capacities and node demands are integers, then there exists an integral optimal solution.

**Proof**
Check the network simplex.

Since the node demands are integers, we have an integral basic feasible solution.

But the capacities are also integers, so at each iteration, we remain integral.

For a MCFP formulation, there exists an integer-valued optimal solution $x$. So $x_e = 0$ or $x_e = 1$ for each arc $e$.

$$M = \{e : x_e = 1\}$$

is a perfect matching becuase each node is incident with at most one edge.

Also, any PM corresponds to an integral flow.

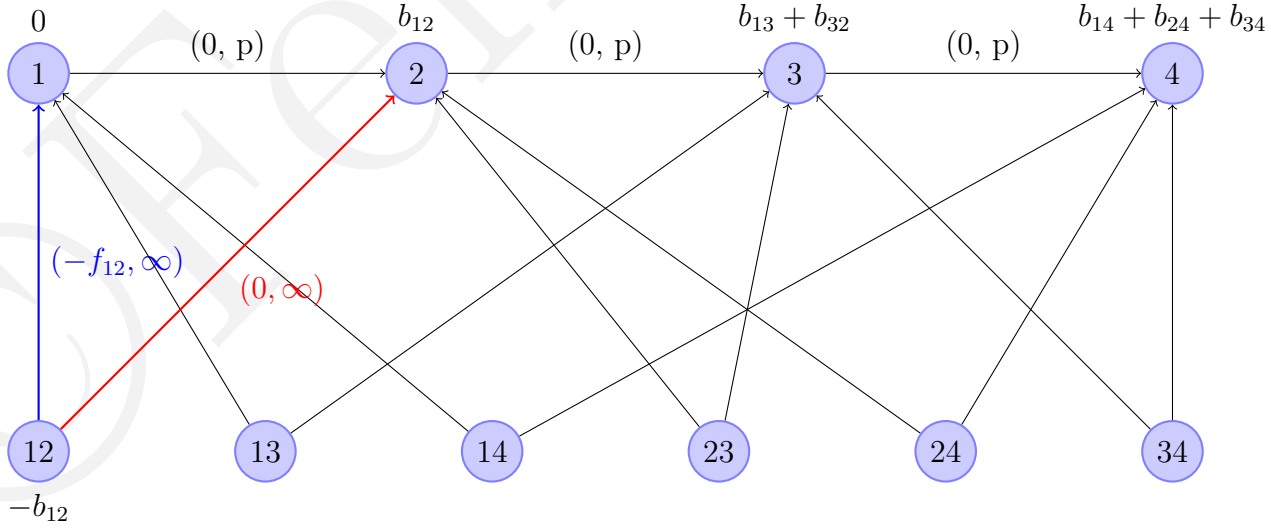
### 4.5.2 Airline Scheduling

A plane visits cities $1, \ldots, n$ in this order:

- There are $b_{i,j}$ passengers from city $i$ to city $j$ $(i < j)$
- The ticket costs are $f_{i,j}$ $(i < j)$
- The plane has capacity $P$

We wish to maximize ticket costs subject to plane capacity (minimize negative costs).

$$\min \sum_{i=1}^{n-1} \sum_{j=2}^{n} -f_{ij} x_{ij,i}$$



We have nodes and arcs that represent the path of the plane, capacity $p$, costs 0.

Each node $i, j$ takes passengers from $i$ to $j$ either through the plane, or by other means. If the passenger takes the plane, we gain $f_{12}$, the price of the plane ticket (negative so its a minimization problem). Elsewise, the customer does not take our flight, and there is no gain whatsoever.

Notice that the negative plane costs "push" passengers to take the plane whenever possible, certainly not the most intuitive assumptions.

### 4.5.3 Catering

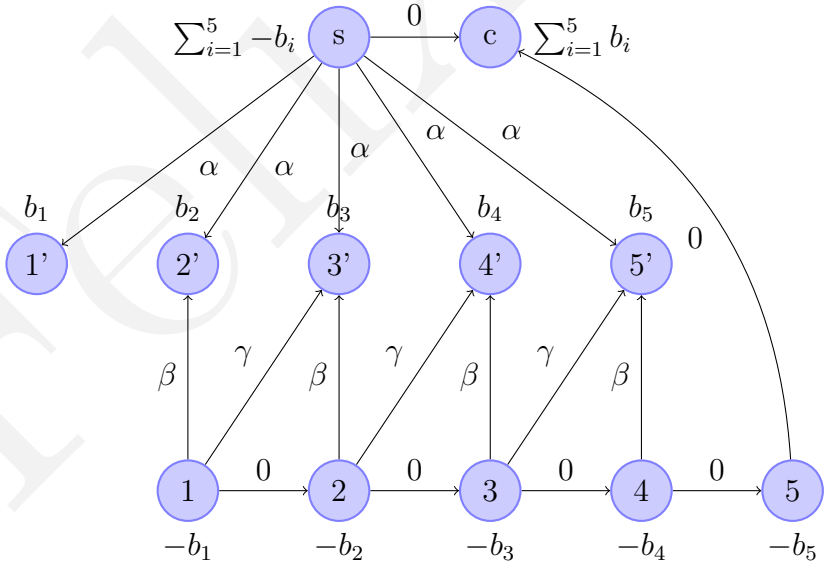
A caterer requires $b_i$ clean napkins for each day $1, \ldots, n$.

They can buy new ones from the store for a cost of $\alpha$.

Used napkins can be treated in 3 different ways.

- 1-day service for a cost of $\beta$
- 2-day service for cost $\gamma$
- kept in storage at no cost

Suppose that clean napkins become dirty at the end of the day. We wish to minimize the total cost of napkins.




the $i$'s are the dirty napkins generated everyday and the $i'$'s are the required number of napkins each day.

### 4.5.4 Matrix with Consecutive 1's

Minimize $w^T x$ subject to

$$Ax \geq b, x \geq 0$$

where the each column of $A$ has one block of consecutive 1's and only 0's otherwise.

$$b = \begin{bmatrix} 5 \\ 12 \\ 10 \\ 6 \end{bmatrix}, A = \begin{bmatrix} 0 & 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 \end{bmatrix}$$

Say $A$ has $p$ rows. We add $p$ slack variables of $-1$'s. Add also a redundant row of 0's.

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

In the order of $i = p, p-1, p-2, \ldots, 1$, substract the $i$-th constraint from the $(i+1)$-th constraint. This produces the incidence matrix of a directed graph!

$$A' = \begin{bmatrix} 0 & 1 & 0 & 1 & 1 & -1 & 0 & 0 & 0 \\ 1 & 0 & 0 & -1 & 0 & 1 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 & 1 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 \\ -1 & -1 & -1 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}, b' = \begin{bmatrix} 5 \\ 7 \\ -2 \\ -4 \\ -6 \end{bmatrix}$$

Note that the new demands actually add to 0! This is due to the telescoping sum

$$\sum b'_i = (0 - b_p) + (b_p - b_{p-1}) + \cdots + (b_2 - b_1) + b_1 = 0$$

**Example 4.5.2**
A manufacturing comapny must contract for $d(i)$ units of storage.

For the time periods $i = 1, \ldots, n$.

$w_{ij}$ denotes the cost of 1 unit of storage at the beginning of period $i$, through the period

$$[i, j)$$

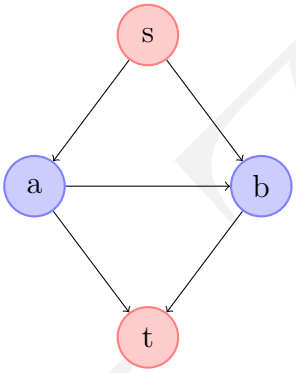
We wish to know how much capacity to acquire, at what times, and for how many period.

# 5 Shortest Dipaths

Although Network Simplex is powerful, it is not polynomial time. We will show some more specialized algorithms which are more efficient.

## 5.1 Problem

Given a digraph $D = (N, A)$, arc costs $w \in \mathbb{R}^A$, two distinct nodes $s, t \in N$, we wish to find a minimum cost $st$-dipath.




**Example 5.1.1**

## 5.2 LP Formulation

**Definition 5.2.1 (Characteristic Vector)**
Let $P$ be a $st$-dipath.
The characteristic vector of $P$ is $x^P \in \mathbb{R}^A$ such that

$$x_e^p = \begin{cases} 1, & e \in A(P) \\ 0, & e \notin A(P) \end{cases}$$

we want: $\min w^T x$ such that

$$x(\delta(\bar{v})) - x(\delta(v)) = \begin{cases} -1, & v = s \\ 1, & v = t \\ 0, & \text{else} \end{cases}$$

and $x \geq 0$ for all $v \in N$.

Note that the characteristic vector of any $st$-dipath is an integral feasible solution. NOte that also this is a TP.

### 5.2.1 Dual

$\max y_t - y_s$ subject to

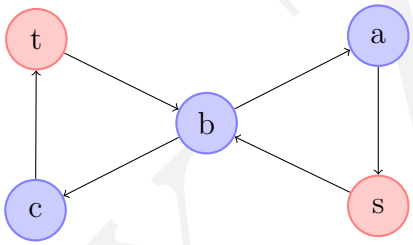
$$y_v - y_u \leq w_{uv}, uv \in A$$

where we write

$$\bar{w}_{uv} := w_{uv} - y_v + y_u$$

## 5.3 Integral Feasible Solutions

to the LP might not be an $st$-dipath.




**Example 5.3.1**

However, it will contain a $st$-dipath. We can subtract $x^P$ from this flow and get a set of dicycles.

**Theorem 5.3.2**
If $\bar{x}$ is an integral feasible solution to LP, then $\bar{x}$ is a sum of the characteristic vector of an $st$-dipath and a collection of dicycles.

**Proof**
Let $F = \{e \in A : \bar{x}_e > 0\}$.

We first show the existence of a dipath. Let $\delta(S)$ be an $st$-cut. The net flow of $S$ is $-1$, so there must be at least one arc in $\delta(S)$ with non-zero flow (ie it is in $F$).

So there is an $st$-dipath $P$ using arcs of $F$.

Let $x' = \bar{x} - x'^P$. Since $\bar{x}, x^P$ satisfy the flow constraints, it is a feasible solution ie

$$\forall v \in N, x'(\delta(\bar{v})) - x'(\delta(v)) = 0$$

Let

$$F' = \{e \in A : x'_e > 0\}$$

If $F' = \varnothing$, we are done. Else suppose $F' \neq \varnothing$.

Take a longest dipath

$$v_1, \dot{,} v_k$$

such that $v_i v_{i+1} \in F'$.

Since

$$x'(\delta(\bar{v}_k)) - x'(\delta(v_k)) = 0$$

there is an arc $v_k v_i$ for some $i < k$ (cannot be outside path since we took longest di-path).

This form s dicycle $C$

$$v_i, v_{i+1}, \ldots, v_k, v_i$$

Then

$$x^{(2)} = x' - x^C$$

satisflies flow contraints and the sum of all flows has decreased by at least 1. By induction, we are done.

### 5.3.1 Consequences

**Theorem 5.3.3**
If there are no negative dicycles, then our LP formulation has an optimal solution that is the characteristic vector of an *st*-dipath.

**Proof**
Suppose $\bar{x}$ is an optimal integral solutionto our LP.

If $\bar{x}$ is the characteristic vector of an *st*-dipath, then we are done.

Else

$$\bar{x} = x^P + x^{C_1} + \cdots + x^{C_k}$$

where $P$ is an *st*-dipath and $C_i$'s are all dicycles.

If there are no negative dicycles, then

$$w^T \bar{x} \geq w^T x^p$$

So $P$ is an optimal solution in the form of an $st$-path.

If there are negative dicycles, then the LP is unbounded.

In the original problem, the minimum must exist as there are finitely many $st$-dipaths.

So the LP formulation (techniques to solve LP) will not help.

## 5.4 Terminology

**Definition 5.4.1 (Equality Arc)**
Given a node potential $y$, if an arc $uv$ satisfies

$$\bar{w}_{uv} = w_{uv} - y_v + y_u = 0$$

Then we call $uv$ an equality arc.

So finding a shortest $st$-dipath gives us in a sense, more information than solving an LP.

**Theorem 5.4.1**
Let $D = (N, A)$ be a digraph with arc costs $w \in \mathbb{R}^A$ and no negative dicycles.
If

$$v_1, v_2, \ldots, v_k$$

is a shortest $v_1, v_k$-dipath, then

$$v_1, v_2, \ldots, v_i$$

is a shortest $v_1, v_i$-dipath.

**Proof**
Since there are no negative dicycles, our LP has an optimal solution which is the characteristic vector of an $v_1, v_k$-dipath.

Then there is a corresponding optimal dual solution $y$ where all arcs of $P$ are equality arcs (CS Conditions).

Let

$$P' = v_1, \ldots, v_i$$

Then $y$ is still feasible for the dual LP of the shortest $v_1, v_i$-dipath problem.

Any arc in $P'$ is in $P$, so all arcs of $P'$ are equality arcs.

It follows that the CS Conditions are satisfied for $v_1, v_i$-dipath problem.

**Definition 5.4.2 (Rooted)**
A tree $T$ is rooted at $s$ if for all $t \in N(T)$, the unique $s,t$-dipath in $T$ is an $s,t$-dipath.

Can we find all shortest $s,v$-dipath for all $v \in N$ in one go?

Yes! As long as there are no negative dicycles.

Ford's Algorithm, Bellman-Ford Algorithm.

## 5.5 Results on Rooted Trees

Let $D = (N, A), x \in N$, then there is an $st$-dipath in $D$ for all $t \in N$ if and only if there is a spanning tree in $D$ rooted at $s$.

**Proposition 5.5.1**
Let $T$ be a spanning tree in $D$.
Then $T$ is rooted at$s$ if and only if the in-degree of $s$ is 0, and the in-degree of all other nodes is 1 in $T$.

**Proof**
Assignment.

**Definition 5.5.1 (Predecessor)**
For a spanning tree $T$, rooted at $s$, for each node $v$ other than $S$, its predecessor is the unique node $u$ such that

$$uv \in A(T)$$

We will assume that every node can be reached from $s$ via a dipath.

## 5.6 Ford's Algorithm

This produces a feasible potential ($\forall e, \bar{w}_e \geq 0$) and a rooted spanning tree at $s$, so that the arcs of this tree are equality arcs.

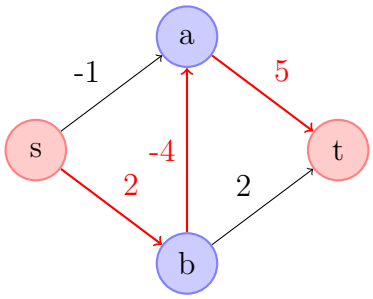
At each step of the algorithm, we maintain a potential, and the predecessor of each node.

### 5.6.1 The Algorithm

1) Set $y_s = 0$ and $y_v = \infty$ for $v \in N \setminus \{s\}$. Set predecessor, $p_v = \varnothing, \forall v \in N$
2) While $y$ is not feasible

(a) Find an arc $uv \in A$, where $\bar{w}_{uv} < 0$

(b) Set $y_v = y_u + w_{uv}$. Set $p_v = u$




### 5.6.2 Remarks

Observe that $y_v$ never increases since by setting $y_v = y_u + w_{uv}$ where $y_v > y_u + w_{uv}$ to begin with, we decrease $y_v$.

**Definition 5.6.1 (Predecessor Digraph)**
At any point in the algorithm, the precessor digraph, denote $D_p$ is one where

$$N(D_p) = N$$

and

$$A(D_p) = \{p_v v : v \in N \setminus \{s\}\}$$

**Proposition 5.6.1**
Throughout the algorithm, $\bar{w}_e \leq 0$ for all arcs in $D_p$.

**Proof**
Focus on an arbitrary node $v$ and its unique predecessor $p_v$.

When, a correction takes place with an arc whose head is in $v$, then $\bar{w}_{p_v v} = 0$ until the predecessor of $v$ is changed again.

The reduced costs stays non-positive, as only $y_{p_v}$ can change (due to correcting other arcs), and by observation, $y_{p_v}$ decreases so

$$\bar{w}_{p_v v} = w_{p_v v} - y_v + y_{p_v}$$

also decreases.

**Proposition 5.6.2**
If $D_p$ contains a dicycle (at any point in the algorithm), then $D$ contains a negative dicycle, and the algorithm does not terminate.

**Proof**
Suppose we produce a dicyle $C$ in $D_p$ by connecting the arc $uv$.

Say

$$C = v = v_1, v_2, \ldots, v_k = u, v_1$$

Since we connected $uv$, it must be true that in the previous iteration

$$y_u + wuv - y_v < 0$$

which means

$$y_{v_k} + w_{v_k v_1} - y_{v_1} < 0$$

From our previous work $\bar{w}_e \leq 0$ for all $e \in A(D_p)$ so

$$y_{v_i} + w_{v_i v_{i+1}} - y_{v_{i+1}} \leq 0, 1 \leq i < k$$

Since $C$ is a dicycle, when adding all these inequalities, the $y$'s cancel. We get

$$\sum_{i=1}^{k-1} w_{v_i v_{i+1}} + w_{v_k v_1} < 0$$

which shows that $C$ is a negative dicycle.

But if there is a negative dicycle, there cannot be a feasible potential (exercise), so the algorithm never terminates.

**Proposition 5.6.3**
If $s$ has a predecessor, then $D$ contains a negative dicycle, and the algorithm does not terminate.

**Proof**
Exercise.

### 5.6.3 Termination

**Proposition 5.6.4**
Suppose the algorithm terminates. Then $D_p$ is a spanning tree of shortest dipaths rooted at $s$.
Furthermore, $y_v$ will bete cost of a shortest $s, v$-dipath.

**Proof**
Since the algorithm terminates, $D_p$ cannot contain a dicycle, and $s$ does not have a predecessor.

So $D_p$ is a rooted spanning tree.

Since all nodes other than $s$ has a predecessor, the in-degree is 1, and $D_p$ is rooted at $s$.

Now, all arcs in $D_p$ are equality arcs, since we know $\bar{w}_e \leq 0$ for all $e \in A(D_p)$, and $\bar{w}_e < 0$ is impossible since $y$ is feasible, by termination.

For $v \in N$, let $P$ be the $s, v$-dipath in $D_p$. Consider the LP formulation of the shortest $s, v$-dipath problem.

Then $x^P$ is feasible for the primal and $y$ is feasible for the dual.

The CS Conditions hold since all arcs in $P$ are equality arcs. So $x^P$ is an optimal and the objective of the dual is

$$y_v - y_s = y_v - 0 = y_v$$

So $y_v$ is the cost of a shortest $s, v$-dipath.

## 5.7 The Bellman-Ford Algorithm

The idea is to go through ars in "passes".

### 5.7.1 The Algorithm

1) Initialisation is the same as Ford, with an additional variable $i = 0$
2) While $i < |N| - 1$
    a) for each $uv \in A$, if $\bar{w}_{uv} < 0$, then set $y_v = y_u + w_{uv}, p_v = u$.
    b) increment $i$

Notice that if a feasible potential is found, then everything from Ford's Algorithm applies, and we have an optimal rooted tree.

Else, if we have an infeasible potential after $|N| - 1$ steps, we show that we have a negative dicycle.

### 5.7.2 Proof of Correctness

Let us write $d_v$ for the cost of a shortest $s, v$-dipath $y_v = d_v$.

**Theorem 5.7.1**
Suppose no negative dicycles exist.
After the $i$-th iteration, if there is a shortest $s, v$-dipath using at most $i$ arcs, then $y_v = d_v$.

**Proposition 5.7.2**
Suppose there are no negative dicycles.
Then at any point in the algorithm, $y_v \geq d_v$.

**Proof**
This is true at initialisation.

If $y_v \neq \infty$, then we can trace $v$ bac to $s$ using $D_p$.

For each of the arcs $e$, $\bar{w}_e \leq 0$. Adding all these inequalities $\bar{w}_e \leq 0$ for all arcs in this $s, v$-dipath.

We obtain $w(P) - y_v \leq 0$. So

$$d_v \leq w(P) \leq y_v$$

as desired.

**Proof (theorem)**
We argue by induction on $i$.

When $i = 0$, this is trivial.

Assume that this is true after the $i$-th iteration. We wish to show this for the $(i+1)$-th iteration.

Pick $v$ which has a shortest $s, v$-dipath that uses at most $i + 1$ arcs. If there is a shortest $s, v$-dipath that uses at most $i$ arcs, by induction, $y_v = d_v$ so that by the proposition, $y_v$ will not change.

Suppose there is a shortest $s, v$-dipath that uses exactly $i + 1$ arcs, say

$$s = v_0, v_1, \ldots, v_{i+1} = v$$

Since no negative dicycles exist, $v_1, \ldots, v_i$ is a shortest $s, v_i$-dipath which uses $i$ arcs.

By induction, $y_{v_i} = d_{v_i}$ after the $i$-th iteration, and this does not change after the $i$-th iteration.

Consider $\bar{w}_{v_i v_{i+1}}$, if it is 0, this means

$$\begin{aligned} y_{v_{i+1}} &= y_{v_i} + w_{v_i v_{i+1}} \\ &= d_{v_i} + w_{v_i v_{i+1}} \\ &= w(P) \\ &= d_{v_{i+1}} \end{aligned}$$

Elsewise, if $\bar{w}_{v_i v_{i+1}} > 0$, this means

$$\begin{aligned} y_{v_{i+1}} &< y_{v_i} + w_{v_i v_{i+1}} \\ &= d_{v_i} + w_{v_i v_{i+1}} \\ &= w(P) \\ &= d_{v_{i+1}} \end{aligned}$$

which is a contradiction.

Finally, if $\bar{w}_{v_i v_{i+1}} < 0$, this means the algorithm must correct the arc.

$$\begin{aligned} y_{v_{i+1}} &= y_{v_i} + w_{v_i v_{i+1}} \\ &\ldots \\ &= d_{v_{i+1}} \end{aligned}$$

so we are done.

**Corollary 5.7.2.1**
At the end of Bellman-Ford, if $y$ is feasible, the $y_v = d_v$ for all $v \in N$.
Otherwise, we can conclude that there is a negative dicycle.

**Proof**
Bellman-Ford runs in $|N| - 1$ iterations.

Any shortest $s, v$-dipath uses at most $|N| - 1$ arcs.

If $y$ is feasible, then there are no negative dicycles, so by the theorem, $y_v = d_v$ for all $v \in N$.

If not, there is a negative dicycle.

## 5.8 Dijkstra's Algorithm

Consider the case when there are no negative costs. We can apply a greedy algorithm!

This will be faster than the Bellman-Ford Algorithm.

### 5.8.1 Motivation

If there are no negative arcs, $y = 0$ is a feasible potential for the dual.

We wish to raise potentials by $t$ for non-tree nodes while maintaining feasibility.

### 5.8.2 Details

Case I: Consider $u, v \notin N(T)$, $T$ is our current tree. Then both $y_u, y_v$ increases by $t$, so $\bar{w}_{uv} = w_{uv} - y_v + y_u$ does not change.

Case II: If $u, v \in N(T)$, then we do not change the potentials and $\bar{w}_{uv}$ does not change.

Case III: If $u \notin N(T) v \in N(T)$, then $\bar{w}_{uv}$ increases and that does not introduce infeasibility of the potentials.

Case IV: Otherwise, if $u \in N(T), v \notin N(T)$, then $\bar{w}_{uv}$ decreases by $t$. We can choose $t$ to be the minimum among all such arcs.

Now, the arc which determined the minimum becomes an equality arc and we can add it to the tree.

### 5.8.3 The Algorithm

1) Initialize $y_v = 0$, and $T$ to be the single node tree $s$
2) While $T$ is not a spanning tree
3) (a) Pick $uv \in \delta(N(T))$ such that $\bar{w}_{uv} = \min\{\bar{w}_e : e \in \delta(N(T))\}$
   (b) $y_w := y_w + \bar{w}_{uv}$ for all $w \in N(D - T)$
   (c) Add $uv$ and $v$ to $T$

### 5.8.4 Proof of Correctness

$y$ is always feasible by our work above. This includes at initialization as we do not have negative costs.

All arcs in $T$ are equality arcs.

In addition, The algorithm produces a spanning tree rooted at $S$.

So the same LP argument gives that it must be a tree of shortest $s, v$-dipaths for all $v \in N$.

### 5.8.5 Running Time

$O(|N| \log |N| + |A|)$, which is faster than Bellman-Ford as we can take advantage of the greedy approach.

## 5.9 Applications of Shortest Path

### 5.9.1 Network Reliability

Given a network $D = (N, A)$, each arc $e$ is assigned an associated reliability $r_e \in (0, 1]$. Think of this as a probability that $r_e$ is operational.

For a given dipath $P$, the reliability of $P$ is

$$r(p) = \prod_{e \in P} r_e$$

Our goal is to maximize reliability amongst all $s, t$-dipaths.

Notice that $\log r(P) = \sum_{e \in P} \log r_e$ and log is strictly increasing so it suffices to compare logarithms of reliability.

We also make this a minimization problem by having negative arc costs.

The cost $ij$ is

$$-\log r_{ij}$$

**Proposition 5.9.1**
Modify Djikstra's Algorithm to solve this problem without taking logs.

**Proof**
Exercise.

### 5.9.2 Currency Exchange

We have a set of currencies and an exchange rate $r_{uv}$ representing the number of units of $v$ currencies which 1 unit of $u$ currency exchanges.

Our goal is to make a profit through a series of exchanges.

If we can make a profit, there must be a cycle $C$ such that

$$\prod_{e \in C} r_e > 1 \iff \sum_{e \in C} -\log r_e < 0$$

Label each arc cost with $-\log r_e$ and simply run Bellman-Ford.

# 6 Maximum Flow

## 6.1 LP Formulation

Given $D = (N, A)$ and $s, t \in N$ as source and sink, we wish to maximize total flow from $s$ to $t$.

This is given by the LP (P)

$$\begin{aligned} \max x(\delta(s)) - x(\delta(\bar{s})) & \qquad \text{maximize net out flow of } s \\ x(\delta(\bar{v})) - x(\delta(v)) = 0 & \qquad \forall v \in N \setminus \{s, t\} \\ x_e \in [0, c_e] & \qquad \forall e \in A \end{aligned}$$

Note that we can turn this into a MCFP.

## 6.2 Ford-Fulkerson Algorithm

Given $D, c, x$ (flow), its corresponding residual digraph $D'$ is defined by

$$N(D') = N(D)$$

$$\forall uv \in A \begin{cases} c_{uv} > x_{uv} & \implies uv \in A(D'), c'_{uv} = c_{uv} - x_{uv} \\ 0 < x_{uv} & \implies vu \in A(D'), c'_{vu} = x_{uv} \end{cases}$$

Now, notice that if there is an $s, t$-dipath in $D$', we can push flow in $D$.

Every time the flow changes, the residual graph changes as well.

### 6.2.1 The Algorithm

**Definition 6.2.1 (Augmenting Path)**
An augmenting path is an $s, t$-dipath in $D'$ the residual digraph.

1) Initialize $x_e = 0$ for all $e \in A$. Form the residual digraph $D'$
2) While $D'$ has an augmenting path ($st$-dipath in $D'$)
   (a) find augmenting path $P$ in $D'$
   (b) push additional flow $r$ along $P$ ($+r$ for forward arcs, $-r$ for backward arcs)
3) Update $D'$ with the abridged flow

### 6.2.2 Termination

If $c$ is integral, then at each step we increase the flow by at least 1.

If $c$ is rational, we can multiply $c$ by some large integer to make everything into integers.

If $c$ is irrational, then is it possile that this algorithm does not terminate.

### 6.2.3 Running Time

The running time is polynomial in the numeric values of the input.

## 6.3 Edmonds-Karp

If we pick an augmenting path in $D'$ with the fewest number of arcs (with breadth-first search), then we only need $|N||A|$ iterations.

### 6.3.1 Running Time Analysis

**Lemma 6.3.1**
let $d_f(s,v)$ denote the minimal number of edges in a $sv$-dipath in the residual digraph $D_f$ corresponding to the flow $f$.
Let $f'$ be the flow of the next iteration of Edmonds-Karp.
We claim $d_f(s,v) \leq d_{f'}(s,v)$.

**Proof**
Suppose for a contradiction that some $v \in N$ violates the claim and let $v$ be a minimal counterexample with respect to $d_{f'}(s,v)$.

Let $P$ be a path attaining $d_{f'}(s,v)$ in the digraph. Clearly $v \neq s$ so there is at least one edge $uv \in A(P)$.

But then $u$ is not a counterexample so $d_{f'}(s,u) \geq d_f(s,u)$.

Case I: $uv \in A(D'_f)$.

We have

$$\begin{aligned} d_f(s,v) &\leq d_f(s,u) + 1 \\ &\leq d_{f'}(s,u) + 1 \\ &= d_f(s,v) \end{aligned}$$

which contradicts the assumption that $v$ is a counterexample.

Case II: $uv \notin A(D'_f)$

But $uv \in A(D'_{f'})$. This means flow was pushed through $vu \in A(D'_f)$. But the algorithm chooses the shortest path. In particular

$$\begin{aligned} d_f(s,v) &= d_f(s,u) - 1 \\ &\leq d_{f'}(s,u) - 1 \\ &= d_{f'}(s,v) - 2 \\ &< d_{f'}(s,v) \end{aligned}$$

which is again a contradiction.

**Definition 6.3.1 (Critical)**
We say $uv \in A(P)$ on an augmenting path $P$ in $D'_f$ is critical if

$$c'(P) = c'_{uv}$$

**Lemma 6.3.2**
Fix $uv$, it can be critical at most $\frac{|N|}{2} - 1$ number of times.

**Proof**
Suppose $uv$ is critical on the current iteration $i$ for flow $f$. Then

$$d_f(s,v) = d_f(s,u) + 1$$

and it disappears from the digraph in the next iteration.

Now, suppose $uv$ appears again in iteration $j$. We must have pushed flow on $vu$ in some iteration after $i < k < j$.

Let $f'$ be the flow on iteration $k$. By the previous lemma

$$\begin{aligned} d_{f'}(s,u) &= d_{f'}(s,v) + 1 \\ &\geq d_f(s,v) + 1 \\ &= d_f(s,u) + 2 \end{aligned}$$

But $d_f(s,u) \leq |N| - 2$ for the duration of the algorithm since $uv$ being critical means $u \neq t$. $uv$ can indeed only be critical at most

$$\frac{|N| - 2}{2}$$

as required.

**Proposition 6.3.3**
Edmonds-Karp terminates in $O(|N||A|^2)$ time.

**Proof**
There are at most $O(|N||A|)$ iterations since there are $|A|$ arcs, each iteration has a critical arc, and each arc can be critical $O(|N|)$ times.

Each iteration runs in $O(|A|)$ time so the overall running time follows.

### 6.3.2 Proof of Correctness

We ask whether Edmonds-Karp terminates with a maximum flow.

The algorithm terminates when there is no augmenting path in $D'$. This means that there is an $s,t$-cut $\delta(S)$ that is empty in $D'$.

Arcs in $\delta_D(S)$ have no forward arc so the flow is at capacity.

Arcs in $\delta_D(\bar{S})$ has no backward arcs, so the flow is 0.

We cannot do any better than this as any $s,t$-cut is an upper bound of the sum of the capacities of the leaving arcs (less the sum of entering arcs which is zero in this case).

## 6.4 Maximum-Flow Minimum-Cut

Does Ford-Fulkerson give a maximal flow if it terminates?

**Proposition 6.4.1**
Given $D = (N, A)$ and $s, t \in N$. The value of any $st$-flow is at most the capacity of any $st$-cut.

**Proof**
For any $st$-flow $x$, and any $st$-cut $\delta(S)$, the net flow of $S$ is equal to the net flow of $s$. (Net flow of other nodes is 0).

The value of flow $s$ is

$$\begin{aligned} x(\delta(s)) - x(\delta(\bar{s})) &= s(\delta(S)) - x(\delta(\bar{S})) \\ &\leq x(\delta(S)) \\ &\leq c(\delta(S)) \end{aligned}$$

So the value of any $st$-flow is at most the capacity of $\delta(S)$.

**Theorem 6.4.2 (Max-Flow Min-Cut)**
Given $D = (N, A), s, t \in N, c$, the maximum value of an $st$-flow if equal to the minimum capacity of an $st$-cut.

**Proof**
Let $x$ be a maxium $st$-flow.

Then there is no $st$-dipath in the residual digraph $D$. This means there is an empty $st$-cut $\delta_{D'}(S)$ in $D'$.

If $uv \in \delta_D(S)$, then $uv$ is not an arc in $D'$. So $x_{uv} = c_{uv}$.

If $uv \in \delta_D(\bar{S})$, then $vu$ is not an arc in $D'$. So $x_{uv} = 0$.

Therefore, the value of $st$-flow $x$ is

$$\begin{aligned} x(\delta(S)) - x(\delta(\bar{S})) &= c(\delta(S)) - 0 \\ &= c(\delta(S)) \end{aligned}$$

By the propositions above, $x$ is a max $st$-flow, and $\delta(S)$ is a minimum capacity $st$-cut.

By Max-Flow Min-Cut, Ford-Fulkerson is correct.

### 6.4.1 Combinatorial Applications of Max-Flow Min-Cut

**Definition 6.4.1**
$A' \subseteq A$ disconnects $s$ from $t$ if there is no $st$-dipath in $D' = (N, A - A')$.

**Lemma 6.4.3 (Flow Decomposition)**
Given $D = (N, A)$ and nodes $s, t$ with integer capacities. If $x$ is an $st$-flow of value $k$, and $x$ is integral, then $x$ is the sum of characteristic vectors of $k$ $st$-dipaths.

**Proof**
If $k = 0$, then $x$ is a circulation. So $x$ is the sum of characteristic vectors of dicycles (exercise).

If $k > 0$, then there is an $st$-dipath $P$ (exercise).

Notice that $x - x^P$ is an $st$-flow of value $k - 1$. So we are done by induction on $k$.

**Theorem 6.4.4 (Menger, Arc-Disjoint)**
Given a digraph $D = (N, A)$, and nodes $s, t$, the maximum number of arc-disjoint $st$-dipaths is equal to the minimum number of arcs that disconnect $s$ from $t$.

Recall that a feasible integral flow $x$ is the sum of characteristic vectors of $k$ $st$-dipaths and any number of dicycles.

**Proof**
By Max-Flow Min-Cut.

If there are $k$ arc-disjoint $st$-dipaths then we must remove at least one arc from each of these dipaths.

The inequality is trivial, so we wish to prove equality.

Apply the max flow min cut theorem with $c = 1$.

By the max flow min cut theorem, there is an $st$-flow with the same value as the capacity of an $st$-cut $\delta(S)$. Let us say $k$.

By flow decomposition, $x$ is the sum of $k$ $st$-dipaths and some dicycles. Since $c = 1$, each arc is used in at most 1 $st$-dipath.

Hence, these $st$-dipaths are all arc-disjoint. By removing all $k$ arcs in $\delta(S)$, then $S$ is disconnected from $t$.

If there are $k$ node-disjoint $st$-dipaths, then we need to remove at least $k$ nodes to disconnect $s, t$.

**Definition 6.4.2 ($st$-Separator)**
$X \subseteq N$ such that $s \not\rightsquigarrow t$ in $G - X$.

**Theorem 6.4.5 (Menger, Node Disjoint)**
If $st$ is not an arc, then the maximum number of node-disjoint $st$-dipaths is equal to the minimal size of an $st$-separator.

**Proof (Sketch)**
For every node, generate two nodes with a single edge between them, $v^-, v^+$.

Arcs of the form $uv$ map to $u^+v^-$ with

$$c_{u^+v^-} = \infty$$

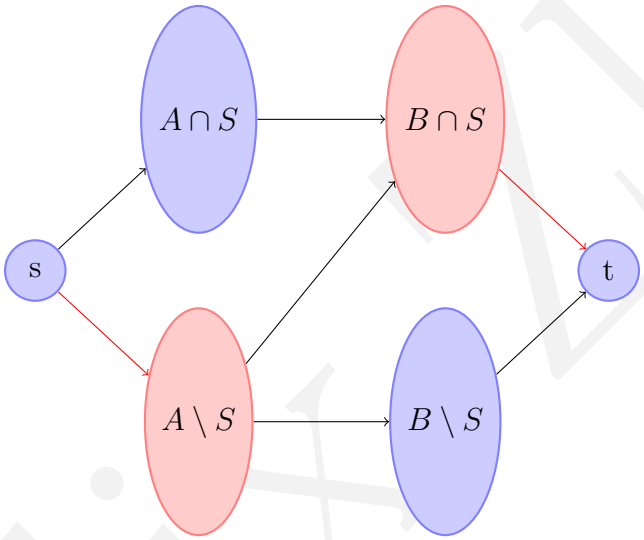
All newly created capacities are 1.

We can then simply apply Menger's Theorem for arc-disjoint paths and conclude the proof.

Then minimal cut cannot use $\infty$ arcs and thus only newly created edges (ie nodes that disconnect $s$ from $t$).

**Theorem 6.4.6 (König)**
$\nu(G) = \tau(G)$ in a bipartite graph.

Observe that $\nu(G) \leq \tau(G)$ trivially since we can take half one vertex per edge in the matching to obtain a lower bound.




**Proof**
Let $A, B$ be a bipartition of $V$ and create $a, b$ with edges $aA, Bb$. Then orient the edges $a \to b$.

A maximum flow corresponds to a maximum matching.

There is a corresponding minimum $st$-cut $\delta(S)$. In the illustration above, $S = \{s\} \cup (S \cap A) \cup (S \cap B)$.

Notice that $\delta(S)$ cannot have any of the original edges-turned-arcs or else it contains one of infinite capacity. The red arcs are the arcs of $\delta(S)$.

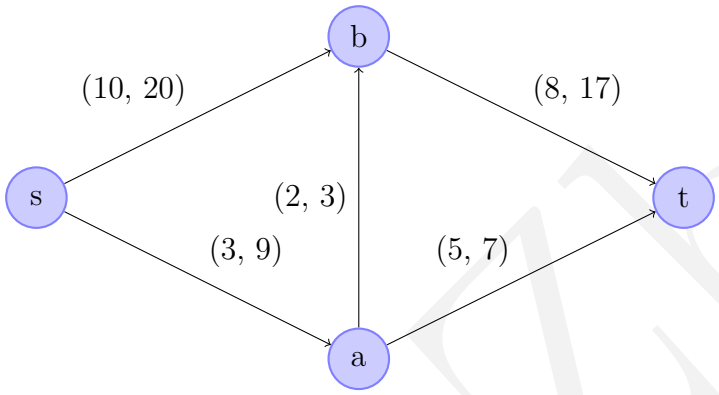
We claim $A \setminus S, B \cap S$ is a vertex cover. This shows the claim.

## 6.5 Flows with Lower Bounds

We know introduce non-negative lower bounds $\ell_e \in \mathbb{R}^A$ for the arcs. A flow must satisfy

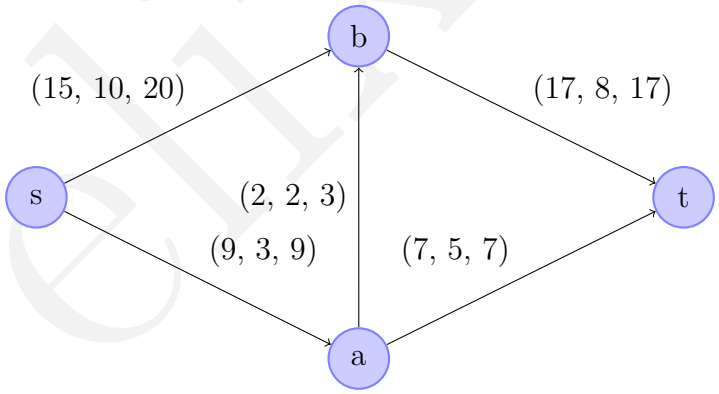
$$\ell_e \leq x_e \leq c_e$$

for all $e \in A$.



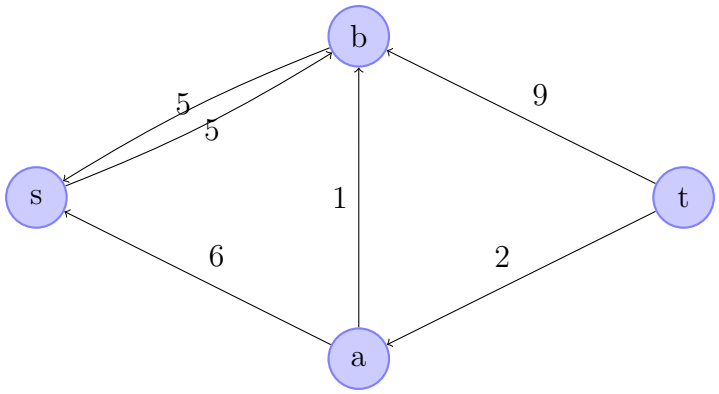

### 6.5.1 Modification of Ford-Fulkerson

Given a feasible flow, we form the residual digraph $D'$, where for the backward arcs, we put the residual $x_e - \ell_e$. Otherwise, the algorithm is exactly the same.




where the labels are $(x_e, \ell_e, c_e)$.

Consider the residual digraph.




There is clearly an empty $st$-cut and so the algorithm terminates.

**Proposition 6.5.1**
For any $st$-flow $x$, and any $st$-cut $\delta(S)$, the value of $x$ is at most

$$c(\delta(S)) - \ell(\delta(\bar{S}))$$

**Proof**
Exercise.

At the end of Ford Fulkerson, there is no $st$-dipath in $D'$, meaning that there is an empty $st$-cut $\delta(S)$.

The net outflow of this flow $x$ (at termination) is

$$c(\delta(A)) - \ell(\delta(\bar{S}))$$

By the proposition, FF finds a maximum $st$-flow.

**Theorem 6.5.2 (Generalized Maximum-Flow Minimum-Cut)**
Given $D = (N, A), c, \ell$ and nodes such that there is a maxium $st$-flow $x$ whose value is the same as the minimum value of

$$c(\delta(S)) - \ell(\delta(\bar{S}))$$

(provided there IS a feasible solution).

### 6.5.2 Feasibility Characterization

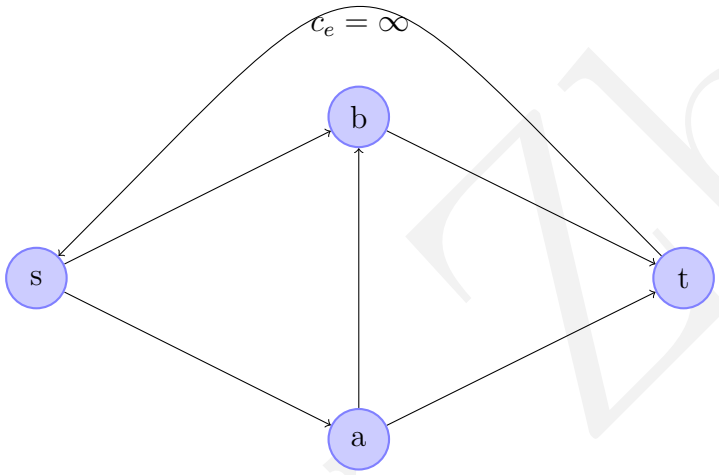
The network is infeasible if and only if there is an $st$-cut $\delta(S)$ such that

$$c(\delta(S)) - \ell(\delta(\bar{S})) < 0$$

**Proof**
Exercise.

### 6.5.3 Aside: Finding an Initial Solution

Recall that it was mentioned the possibility of formulating this problem as a MCFP.




## 6.6 Applications of Maximum Flow

### 6.6.1 Matrix Rounding

Round the matrix $M \in R^{m \times n}$ elements either by taking the ceiling or floor such that the sum of the columns and sum of rows are floored or ceilinged.

We can describe this as an instance of an $st$-flow problem.

Let $s, t$ be dummy nodes, and $r_i, 1 \leq i \leq m, c_j, 1 \leq j \leq n$. There is an arc $su_i$ with bounds floor/ceiling of row $i$ There is an arc $v_j t$ with bounds floor/ceiling of column $j$. Finally, there is an arc $u_i v_j$ with bounds $(\lfloor m_{ij} \rfloor, \lceil m_{ij} \rceil)$.

### 6.6.2 Maximum Closure

We have a set of tasks which may depend on each other. So if $uv \in A$ then taking task $u$ requires taking task $v$.

In order to maximize profit, we want $S \subseteq N$ such that $\delta(S) = \varnothing$.

**Definition 6.6.1 (Closure)**
A set of nodes $S$ such that

$$\delta(S) = \varnothing$$

Our goal is to find a closure with maximum total weight (a function of nodes). Let $s, t$ be two new nodes and for every node with $w(u) > 0$, add an arc $su$, with weight

$$w_{su} = w_u$$

for every node with $w(v) < 0$, add an arc $vt$ with weight

$$w_{vt} := -w_v$$

Finally, all original arcs have capacity $\infty$.

**Proposition 6.6.1**
An $st$-cut $\delta(S)$ in the new digraph has finite capacity if and only if

$$S \setminus \{s\}$$

is a closure in the original digraph.

**Proof (sketch)**
Any cut with an arc of capacity $\infty$ has an arc from the original digraph.

**Proposition 6.6.2**
The weight of the closure is

$$c(\delta(s)) - c(\delta(S))$$

**Proof (sketch)**
Let $v \in N$ and $S \subseteq N$ a closure of the original digraph $D$.

Case I: $v \in S, w(v) > 0$

Then $sv \in \delta(s)$.

Case II: $v \in S, w(v) < 0$

Then $sv \notin \delta(s), vt \in \delta(S)$.

Case III: $v \notin S, w(v) > 0$

Then $sv \in \delta(s), sv \in \delta(S)$.

Case IV: $v \notin S, w(v) < 0$

Then $vt \notin \delta(s), vt \in\notin \delta(S)$.

**Proposition 6.6.3**
Now, $c(\delta(s))$ is fixed so to maximize the weight of the closure, we minimize

$$c(\delta(S))$$

We can solve this with maximum flow.

## 6.7 Preflow-Push Algorithm

With Ford-Fulkerson, we only push an additional flow of 1 each time. Instead, we want to make "local adjustments".

The idea is to introudce a "height" for each node. We keep the heights so that we always push flow downwards, however not too steeply.

1) push as much flow as possible from $s$
2) while the flow is not feasible
3) (a) if we can push flow on $uv$ "gently" (some requirements), do it
   (b) otherwise increase height of $u$.

The main difference between FF, PP is that FF always maintains feasibility while PP works towards feasibility.

The running time for EK is $O(nm^2)$.

PP has running time $O(n^2m)$.

Where $n$ is the number of vertices and $m$ is the number of edges.

**Definition 6.7.1 (Preflow)**
An $st$-preflow $x$ is a flow which satisfies $0 \leq x \leq c$ and

$$x(\delta(\bar{v})) \geq x(\delta(v))$$

for all $v \in N(D) \setminus \{s, t\}$.

**Definition 6.7.2 (Excess)**
The excess at $v$ is

$$e_x(v) = x(\delta(\bar{v})) - x(\delta(v))$$

Consider the residual digraph $D'$, defined the same way as before. Use $r_e$ to represent the residual of each arc in $A(D')$.

Think of embedding the digraph in $\mathbb{R}^3$ and only allowing changes with "gentle" slopes.

For each $v \in N$, we have a height function $h(v)$

**Definition 6.7.3 (Compatible)**
A set of heights $h$ is compatible with a preflow $x$ if

(a) $h(s) = |N|$

(b) $h(t) = 0$

(c) $h(v) \geq h(u) - 1$ for all $uv \in A(D')$

This means that any arc in $D'$ does NOT point down too steeply.

### 6.7.1 The Algorithm

1) Initialize $h(s) = |N|, h(v) = 0$ for all other nodes $h(v)$. Set preflow $x$ with $x_e = c_e$ for all $e \in \delta_D(s)$ and $x_e = 0$ otherwise
2) While there is $u \in N \setminus \{s, t\}$ where $e(u) > 0$
   (a) if there is $uv \in A(D')$ where $h(v) = h(u) - 1$, then push $\min\{r_{uv}, e(u)\}$ on $uv$ (push means add flow on forward arcs, substract flow on backward arcs)
   (b) else $h(u) + +$ ("relabel" operation)

### 6.7.2 Correctness of Preflow-Push

**Lemma 6.7.1**
If preflow $x$ and height $h$ are compatible, then $D'$ has no $st$-dipath.

**Proof**
Suppose not, so an $st$-dipath exists. Let

$$S : s = v_0 v_1 \dots v_k = t$$

in $D'$ with

$$h(v_{i+1}) \geq h(v_i) - 1$$

Adding all these inequalities gives

$$h(v_k) \geq h(v_0) - k$$

so $0 \geq |N| - k$ and

$$|N| \leq k$$

which is a contradiction as there are $k+1$ nodes in the path but only $|N|$ nodes in the digraph.

**Corollary 6.7.1.1**
If $x$ is a feasible flow, with compatible heights $h$, then $x$ is a maximum flow.

**Proof**
There are no $st$-dipaths in $D'$, $x$ is feasible, so no agumenting paths exist, and $x$ is an optimal solution.

**Theorem 6.7.2**
The algorithm maintains a preflow and a height function that are compatible with each other.

**Proof**
At initialization, we have a preflow. We saturate all arc in $\delta(s)$, and so the heights are compatible.

Suppose we push some flow, then we push

$$\min\{e_u, r_{uv}\}$$

and the resulting flow is a preflow.

We may create new arcs, but the only arc created is actually $vu$ (if we are pushing flow on $uv$). But the arc $vu$ is ok.

Suppose we relabel. The reason for performing a relabel operation is that

$$h(u) \leq h(v)$$

for all arcs $uv \in \delta_{D'}(u)$.

It follows that when we add 1 to $h(u)$, then we have

$$h(u) - 1 \leq h(v)$$

for all arcs $uv \in A(D')$.

Notice that if the algorithm terminates, the theorem above guaratees that we are happy.

### 6.7.3 Termination & Running Time Analysis

We will bound the number of push/relabel operations.

To bound the number of relabel operations, we bound the maximum value of $h(v)$, and hence the number of relabel operations is finite.

**Lemma 6.7.3**
If $e(u) > 0$, then there is a $us$-dipath in $D'$.

**Proof**
We will show contrapositive.

Let $T$ be the set of all nodes $u \in N$ with no $u, s$-dipath (we want to show $e(u) = 0$) in $D'$.

We note that

$$\delta_{D'}(T) = \varnothing$$

since otherwise, $uv \in \delta_{D'}(T)$ means $vs$-dipath $P$ exists and thus $uP$ is a $us$-dipath.

This gives

$$\sum_{v \in T} e(v) = \cdots = 0 - c(\delta(T)) \leq 0$$

but since excess is non-negative, this can only happen if $e(v) = 0$ for each $v \in T$.

**Corollary 6.7.3.1**
$h(u) \leq 2|N| - 1$ for all $u \in N$ throughout the algorithm.

**Proof**
Suppose that at some point in the algorithm, the height $h(u)$ increases to $2|N|$.

This means that $e(u) > 0$. By the lemma, there is a $us$-dipath in $D'$

$$P : u = v_0 v_1 \dots v_k = s$$

By compatibility, height decreases by at most 1 in each arc resulting in

$$h(s) > |N|$$

clearly a contradiction.

**Corollary 6.7.3.2**
The total number of relabel operations is at most

$$2|N| \cdot |N| = 2|N|^2$$

**Definition 6.7.4 (Saturating Push)**
Push an arc $uv \in A(D')$ if we push $r_{uv}$

**Definition 6.7.5 (Non-saturating Push)**
Push an arc if we push $e(u)$.

Observe that if we perform a saturating push on $uv$, then $uv$ disappears from $D'$, and $vu$ is in $D'$.

**Theorem 6.7.4**
The number of saturating pushes is at most $2|N||A|$.

**Proof**
Given $uv \in A(D')$, how many times can we do a saturating push on $uv$?

Suppose we have a saturating push on $uv$

$$h(u) = h(v) + 1$$

then $uv$ disappears from $D'$.

In order to push flow on $uv$ again, we need to first push flow on $vu$. In order to do that, we need to relabel $v$ at least TWICE to obtain

$$h(v) = h(u) + 1$$

So between two saturating pushes on $uv$, at least two relabel operations need to occur.

The maximum number of relabel operations on node $v$ is $2|N|$. Thus the number of saturating pushes on a fixed $uv \in A(D')$ is $|N|$.

There are at most $2|A|$ arcs in $A(D')$ at any given time and the result follows.

For non-saturating pushes, we define the function

$$\Phi(x, h) = \sum_{v \in N, e(v) > 0} h(v)$$

At initialization, we have

$$\Phi = 0$$

and it is always non-negative.

**Theorem 6.7.5**
The number of non-saturating pushes is at most

$$4|N|^2|A|$$

**Proof**
We will determine the effect on $\Phi(x, h)$ for each type of operation

Relabel Operation: The relabel operations is done on an excess node, so $\phi(x, h)$ goes up by 1.

Over the algorithm, the maxmum increase from relabelling is $2|N|^2$.

Saturating Push on $uv$: This decreases $e(u)$ and increases $e(v)$.

If $e(v) = 0$ before the push, then after the push, we add $h(v)$ to $\Phi(x, h)$.

Since $h(v) \leq 2|N| - 1$ (from our last lecture) we add at most $2|N| - 1$.

Over the course of the algorithm, the maximum increase from saturating pushes is

$$2|N||A|(2|N| - 1) \leq 4|N|^2|A|$$

(at most $2|N||A|$ saturating pushes).

Non-saturating Push: Then $e(v)$ becomes positive, but $e(u) = 0$

At worst, we add $h(v)$ to $\Phi(x, h)$ and we subtract $h(u)$ from $\Phi(x, h)$. Since

$$h(u) = h(v) + 1$$

and $\Phi(x, h)$ decreases by at least 1.

Note $\Phi(x, h) \geq 0$ throughout the algorithm. The number of non-saturating pushes is therefore at most

$$\begin{aligned} 2|N|^2 + 2|N||A|(2|N| - 1) &= 2|N|^2 + 4|N|^2|A| - 2|N||A| \\ &\leq 2|N|^2 + 4|N|^2|A| - 2|N||N| \qquad \text{connectedness} \\ &= 4|N|^2|A| \end{aligned}$$

**Corollary 6.7.5.1**
Preflow-push terminates in at most

$$2|N|^2 + 2|N||A| + 4|N|^2|A| \in O(|N|^2|A|)$$

operations.

# 7 Gobal Minimum Cuts

We wish to find a minimum non-trivial cut in $D = (N, A)$ with cap $c$.

## 7.1 First Steps

We can pick all possible pairs $s, t$ and run maximum flow on each of the

$$|N|(|N| - 1)$$

pairs.

Something better would be to pick $s \in N$ and notice that it must reside on either side of a global minimum cut.

We just need to find all $st$-cuts and $ts$-cuts

$$2(|N| - 1)$$

times.

We will use Hao-Orlin to solve this more quickly, it is a modification of pre-flow push.

## 7.2 Hao-Orlin

**Definition 7.2.1 ($X, t$-cut)**
Given $X \subseteq N, t \notin X$, an $X, t$-cut has the form

$$\delta(S)$$

where $X \subseteq S, t \neq S$.

**Definition 7.2.2 ($s$-cut)**
Given $s \in N$, and $s$-cut has the form

$$\delta(S)$$

where $s \in S$ and $S$ is non-trivial ($S \neq N$).

Remark that to solve the global minimum cut, we just solve the minimum $s$-cut (at most twice) for a fixed $s$.

Indeed, if a global minimum cut contains $s$ then we are done. Elsewise, reverse all the arcs and this finds the global minimum cut.

### 7.2.1 Generic Algorithm for Minimum $s$-Cut

1) $X = \{s\}$
2) While $X \neq N$
   (a) choose $t \notin X$ and find a minimum $Xt$-cut.
   (b) add $t$ to $X$
3) Output minimum over all cuts found

### 7.2.2 Proof of Correctness

Let $\delta(S^*)$ be a minimum $s$-cut. If we only pick $t \in S^*$, there is nothing to prove.

Consider the first time we pick some $t$ not in $S^*$. Immediately before this step, $X \subseteq S^*$ by our choice of $t$.

The algorithm at this step gives us a minimum $Xt$-cut

$$\delta(\hat{S})$$

But $\delta(S^*)$ is also an $Xt$-cut so

$$c(\delta(S^*)) \geq c(\delta(\hat{S}))$$

and $\delta(\hat{S})$ is also an $s$-cut so

$$c(\delta(S^*)) \leq c(\delta(\hat{S}))$$

thus as cuts, they have the same capacity.

It follows that we output the capacity of the minimum $s$-cut since any $X$ during is the algorithm is an $s$-cut and cannot overcome this lowerbound.

### 7.2.3 Preparation for Hao-Orlin

**Definition 7.2.3 ($X$-preflow)**
For $X \subseteq N$, an $X$-preflow is a flow where

$$e(v) \geq 0$$

for $v \notin X$.

**Definition 7.2.4 (Compatible Height)**
Heights $h$ are compatible with an $X$-preflow if

(1) $h(v) = |N|$ for all $v \in X$

(2) $h(t) \leq |X| - 1$

(3) $h(v) \geq h(u) - 1$ for all $uv \in A(D')$

**Definition 7.2.5 (Level)**
A level $k$ consists of all nodes with height $k$ denoted

$$H(k)$$

**Definition 7.2.6 (Cut Level)**
A cut level is a level $k$ where no arcs goes from

$$H(k) \to H(k-1)$$

in $D'$

**Lemma 7.2.1**
If $\delta(S)$ is an $Xt$-cut with

$$\delta_{D'}(S) = \varnothing$$

and $e(v) = 0$ for all $v \in N \setminus (S \cup \{t\})$ then $\delta(S)$ is a minimum $Xt$-cut.

**Proof**
Take any $Xt$-cut $\delta(S)$. The net flow out of $S$ is

$$x(\delta(S)) - x(\delta(\bar{S})) \leq c(\delta(S))$$

The total net inflow of $N \setminus S$ is equal to

$$\sum_{v \in N \setminus S} e(v) \geq e(t)$$

by non-negativity and the fact that $t \in N \setminus S$.

Now, $\delta_{D'}(S) = \varnothing$ just as in the argument for max-flow min-cut. We have

$$x(\delta(S)) - x(\delta(\bar{S})) = c(\delta(S))$$

with the same argument as well as

$$e(t) = \sum_{v \in N \setminus S} e(v)$$

since $e(v) = 0$ for every $v \in N \setminus (S \cup \{t\})$.

So $S$ attains the lower bound and

$$\delta(S)$$

is a minimum $Xt$-cut.

**Corollary 7.2.1.1**
If $\ell$ is a cut level and $e(v) = 0$ for all $v$ with

$$h(v) < \ell$$

(except $t$).
Then

$$\{v : h(v) \geq \ell\}$$

is a minimum $Xt$-cut.

### 7.2.4 The Algorithm

The idea is to run Preflow Push and maintain a cut level $\ell$. We try to get rid of excess on nodes below $\ell$.

The algorithm keeps non-empty levels consecutuve (except $|N|$).

1) Initialisation $X = \{s\}, t \in N \setminus X, h(s) = |N|, h(v) = 0$ otherwise, $\ell = |N| - 1$. Send as much flow out of $s$ as possible.
2) While $X \neq N$
   (a) Run preflow push with these exceptions
   - only select nodes $v$ such that $e(v) > 0$ AND $h(v) < \ell$
   - if we want to relabel $v$, and $v$ is the only node with height $h(v)$. Do not relabel, and instead set $\ell = h(v)$.
   - if we want to relabel $v$ to $\ell$ then relabel! But reset $\ell = |N| - 1$

   (b) When no node satisfies $e(v) > 0$ and $h(v) < \ell$, store the cut $\{v : h(v) \geq \ell\}$ (minimum $Xt$-cut by the corollary) Add $t$ to $X$ and set $h(t) = |N|$. Send as much flow out of $t$ as possible ($\delta_{D'}(t) = \varnothing$). Pick $t$ with lowest height and set $\ell = |N| - 1$.

### 7.2.5 Proof of Correctness

We will show that $X$-preflow and heights are compatible.

$h(v) = |N|$ for all $v \in X$ from the definition of the algorithm.

$h(v) \geq h(u) - 1$ for $uv \in A(D')$ follows from our work with pre-flow push during each iteration.

Suppose at the end of each iteration, we move $t$ to $X$ and push all flow out of $t$. This maintains

$$h(v) \geq h(u) - 1$$

for every $uv \in A(D')$ as $\delta_{D'}(t)$ is now the emptyset.

We need $h(t) \leq |X| - 1$.

**Lemma 7.2.2**
The non-empty levels less than $|N|$ are consecutive.

**Proof**
Initially, we only have one level less than $|N|$ so the result holds trivially.

We do not relabel $v$ when $v$ is the only node of heigh $h(v)$ so this keeps the non-empty levels consecutive. Transitioning to a new iteration, we move $t$ with lowest height to $X$. This means the non-empty levels remain consecutive.

**Lemma 7.2.3**
$h(t) \leq |X| - 1$.

**Proof**
Initially, $|X| = 1$ and $h(t) = 0 \leq |X| - 1 = 0$.

When we move $t$ to $X$, the next $t$ (call it $t'$) has height $h(t)$ or $h(t) + 1$ (since non-empty levels are consecutive).

Originally $h(t) \leq |X| - 1$. We add 1 to $|X|$ after the move and possibly add 1 to $h(t)$.

Either way, the inequality is preserved.

**Corollary 7.2.3.1**
The algorithm maintains compatible $X$-preflow and heights.

We will now show that the store cuts are min $Xt$-cuts.

It suffices to show that $\ell$ is always a cut level, as then our corollary from before applies.

**Lemma 7.2.4**
$h(v) \leq |N| - 2$ for all $v \notin X$.

**Proof**
We proved $h(t) \leq |X| - 1$ (which is the lowest level) and that there are $|N| - |X| - 1$ nodes not in $X \cup \{t\}$.

Since the non-empty levels are consecutive, the highest level outside $X$ is at most

$$(|X| - 1) + (|N| - |X| - 1) = |N| - 2$$

**Lemma 7.2.5**
$\ell$ is always a cut level when $\ell = |N| - 1$.

**Proof**
level $\ell$ is empty, so it is automatically a cut level.

**Corollary 7.2.5.1**
$\ell$ is always a cut level.

**Proof**
Suppose we change $\ell$ to something else (other than $|N| - 1$) when we wish to relabel $v$, but $v$ is the only node with its height. This happens as $e(v) > 0$ and no neighbours of $V$ is one level below in $D'$.

No arcs goes from $h(v)$ to level $h(v) - 1$. So $h(v)$ is a cut level and we can set

$$\ell = h(v)$$

### 7.2.6 Termination & Running Time Analysis

This section is not covered on the final exam.

Relabel ops $h(v) \leq |N| - 2$ so in total is at most $|N|^2$ operations.

Saturating and non-saturating pushes are the same as before.

Level settings operations are stmost the number of relabel operations $+n$ (number of iterations).

Overall, the running time is asymoptotically equivalent as preflow-push.

# 8 Global Minimum Cut in Undirected Graphs

We may wish to find the global minimum cut in undirected graphs.

Consider the undirected graph $G = (V, E)$ with edge capacity $c \in \mathbb{R}_+^E$.

## 8.1 Karger's Algorithm

The idea of the algorithm is as follows

1) Pick a random edge and contract it
2) Keep track of vertices each contracted vertex represents
3) Do this until only two vertices remain
4) output the cut represented by these two vertices

In general, edges with small capacities are more likely to be in a minimum cut. So we try to lower the probability that these edges are selected.

Thus, we set the probability of an edge getting selected as propotional to its capacity.

### 8.1.1 The Algorithm

1) While $|V| > 2$
   (a) pick $uv$ with probability $\frac{c_{uv}}{\sum_{e \in E} c_e}$
   (b) contract $uv$.

### 8.1.2 Proof of Correctness

**Theorem 8.1.1**
Let $\delta(S^*)$ be a global minimum cut.
The probability that the algorithm produces $\delta(S^*)$ is at least

$$\frac{1}{\binom{|V|}{2}}$$

**Proof**
Consider the probability we pick an edge in $\delta(S^*)$ in the first step. The denominator is $\sum c_e$ and the numerator is $c(\delta(S^*))$.

Consider cuts of the form $\delta(v)$ for some vertex $v$. Each edge $uv$ appears in two such cuts

$$\delta(u), \delta(v)$$

Therefore,

$$\begin{aligned}\sum c_e &= \frac{1}{2}\sum c(\delta(v)) \\ &\geq \frac{1}{2}|V|c(\delta(S^*))\end{aligned}$$

since $\delta(S^*)$ is a global minimum cut.

So the probability that an edge in $\delta(S^*)$ is selected is

$$\begin{aligned}\frac{c(\delta(S^*))}{\sum c_e} &\leq \frac{c(\delta(S^*))}{\frac{1}{2}|V|c(\delta(S^*))} \\ &= \frac{1}{\frac{1}{2}|V|} \\ &= \frac{2}{|V|}\end{aligned}$$

The probability that $\delta(S^*)$ survives the first contraction is thus at least

$$1 - \frac{2}{|V|}$$

Suppose we have contracted $k$ edges, and $\delta(S^*)$ is still intact.

We have $|V| - k$ vertices left, and say the graph is

$$G' = (V', E')$$

We want to find the probability of selecting an edge in $\delta(S^*)$. The numerator is $c(\delta(S^*))$. The denominator is $\sum_{e \in E'} c_e$.

Now,

$$\begin{aligned}\sum_{e \in E'} c_e &= \frac{1}{2}\sum_{v \in V'} c(\delta(v)) \\ &\geq \frac{1}{2}\sum_{v \in V'} c(\delta(S^*)) \\ &= \frac{1}{2}(|V| - k)c(\delta(S^*))\end{aligned}$$

so the probability is at most

$$\frac{c(\delta(S^*))}{\frac{1}{2}(|V|-k)c(\delta(S^*))} = \frac{2}{|V|-k}$$

It follows that $\delta(S^*)$ survives this iteration is at least

$$1 - \frac{2}{|V|-k}$$

The largest possible $k$ is $|V|-3$ (since we finish when we have 2 vertices left, this is the last step).

Overall the chance that $\delta(S^*)$ survives all contractions is at least

$$\begin{aligned}\prod_{k=0}^{|V|-3}\left(1-\frac{2}{|V|-k}\right) &= \prod_{k=0}^{|V|-3}\frac{|V|-k-2}{|V|-k} \\ &= \frac{2}{|V|(|V|-1)} \\ &= \frac{1}{\binom{|V|}{2}}\end{aligned}$$

We expect the algorithm to produce a minimum cut if it runs $\binom{|V|}{2} \approx |V|^2$ times.

**Corollary 8.1.1.1**
The probability that the agorithm produces $\delta(S^*)$ after $k|V|^2$ runs is at least

$$1 - e^{-2k}, k \geq 1$$

**Proof**
We use the simple bound

$$1 - x \leq e^{-x}$$

The probability of failure is at most

$$\begin{aligned}\left(1-\frac{1}{\binom{|V|}{2}}\right)^{k|V|^2} &\leq \left(1-\frac{2}{|V|^2}\right)^{k|V|^2} \\ &\leq \left(e^{-\frac{2}{|V|^2}}\right)^{k|V|^2} \\ &= e^{-2k}\end{aligned}$$

and thus the probability of success is at least

$$1 - e^{-2k}$$

Recall that if we wish to use the brute force algorithm to check every single cut, it takes exponential time with Karger's Algorithm gives us a polynomial time solution with very high chance.