CO 351 Network Flow Theory

Keven Qiu
Instructor: Joseph Cheriyan
Spring 2023

Contents

List of Algorithms

Chapter 1

Graph Theory

1.1 Undirected Graphs

Definition: Graph

A graph $G = (V, E)$ consists of a set of vertices and an unordered pair of elements of V called edges.

Definition: st-Cut

For $S \subseteq V$, the cut induced by S is the set of all edges with one end in S and one end not in S, denoted $\delta(S) = \{(u, v) \in E : u \in S, v \notin S\}$. Given two vertices $s, t \in V$ with $s \in S, t \notin S$, we call $\delta(S)$ an st-cut.

Definition: st-Path

An st-path is a path with starting vertex s and ending vertex t.

Theorem

Let s, t be vertices of a graph $G = (V, E)$. There exists an st-path if and only if every st-cut is non-empty.

Proof. Suppose we have an st-path $P = v_1v_2, v_2v_3, \ldots, v_{k-1}v_k$, where $s = v_1$ and $t = v_k$. Consider any st-cut $\delta(S)$. Choose l largest with $1 \leq l \leq k - 1$ such that $v_l \in S$. Then $v_{l+1} \notin S$ and $v_lv_{l+1} \in E$, hence $\delta(S) \neq \emptyset$. Suppose there is no st-path. Let S be the set of vertices which can be reached from s. By construction, $s \in S$ and $t \notin S$. Moreover, for all $u \in S$ and $v \notin S$, there exists an su-path but no sv-path. Thus $uv \notin E$, hence $\delta(S) = \emptyset$.

Corollary

A graph $G = (V, E)$ is connected if and only if there does not exist S where $S \neq \emptyset$, $S \neq V$, and $\delta(S) = \emptyset$.

1.2 Directed Graphs

Definition: Directed Graph

A digraph $D = (N, A)$ consists of a set of nodes and an ordered pair of elements of N called arcs.
For an arc $(u, v) \in A$, u is called the **tail** and v is called the **head**.
The **out-degree** of node u, denoted $d(u)$ or $d^{out}(u)$, is the number of arcs with tail u.
The **in-degree** of node u, denoted $d(\overline{u})$ or $d^{in}(u)$, is the number of arcs with head u.
A **diwalk** is a sequence of nodes $v_1 v_2 \dots v_k$ where (v_i, v_{i+1}) is an arc.

Definition: Cut

For $S \subseteq N$, the cut induced by S is denoted $\delta(S) = \{(u, v) \in A : u \in S, v \notin S\}$ (sometimes written $\delta^{out}(S)$). We denote the complement of S by $\overline{S}$ and define $\delta(\overline{S}) = \{(u, v) \in A : u \notin S, v \in S\}$ (sometimes written $\delta^{in}(S)$).

Definition: *st*-Cut

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

Theorem

Let $D = (N, A)$ be a digraph and $s, t \in N$. There exists an *st*-dipath if and only if there is no *st*-cut $\delta(S)$ with $\delta(S) = \emptyset$.

Proof. ($\implies$) Suppose there exists an empty *st*-cut $\delta(S)$. This partitions the graph into two sets of nodes S and $N \setminus S$ with $s \in S$ and $t \in N \setminus S$ and no outgoing edges from S to $N \setminus S$. Thus, an *st*-dipath cannot exist.

Proof. ($\impliedby$) Suppose every *st*-cut is non-empty and let S be the set of nodes $v \in A$ where an *sv*-dipath exists. If $t \in S$, then we are done, so suppose $t \notin S$. Then, $\delta(S)$ is an *st*-cut. By assumption, $\delta(S)$ is non-empty and so there is an arc $(x, y) \in \delta(S)$ with $x \in S$ and $y \in N \setminus S$. Since $x \in S$, an *sx*-dipath P exists and since $y \notin S$, there does not exist an *sy*-dipath, but $P + (x, y)$ is one. This is a contradiction.

Definition: Collection

A family of objects where collection is allowed.

Proposition

Let Q be an *st*-diwalk. If $s = t$, then Q can be decomposed into a collection of dicycles.
If $s \neq t$, then Q can be decomposed into an *st*-dipath and a collection of dicycles.

Corollary

If there exists a *uv*-dipath and a *vw*-dipath, then there exists a *uw*-dipath.

Proposition

If every node of D has in-degree at least 1, then D has a directed cycle.

Proof. Let $P = v_1v_2, \dots, v_{k-1}v_k$ be a dipath of D with a maximum number of arcs. Since $d(\overline{v_1}) \geq 1$, there is an arc wv_1. It follows from the choice of P that w is a node of P, i.e. $w = v_i$ for some $i \in \{2, \dots, k\}$. Then $v_1v_2, \dots, v_{i-1}v_i, v_iv_1$ is a dicycle.

1.3 Trees

Definition: Tree

A connected acyclic graph.

Definition: Spanning Tree

A spanning tree is a subgraph that is a tree and has vertex set V.

Theorem (Fundamental Cycles)

Let $T = (V, E)$ be a tree.

- $T + e$ where $e \notin E$ has exactly one cycle C.
- Let e' be any edge of C, then $T + e - e'$ is a tree.

Lemma

If there exists a uv-path and a vw-path, then there exists a vw-path.

Lemma

Let G be a connected graph with a cycle C and let e be an edge of C. Then $G - e$ is connected.

Proof. Let $v_1, v_2 \in G - e$. We need to show there exists a v_1v_2-path P of $G - e$. Since G is connected, there exists a v_1v_2-path P of G. If P does not use e, then we are done.

Otherwise, P implies there exist a v_1u-path P_1 and a wv_2-path P_2, where $uw = e$. Moreover, $C - uw$ is a uw-path. Result follows by applying previous lemma twice.

Lemma

In a tree T, any two vertices s, t are connected by a unique path.

Proof. Let P_1 and P_2 be two different st-paths. P_1 has an edge vw that is not in P_2.

P_1 contains a subpath P_v from v to one of its end nodes and contains a subpath P_w from w to its other end node.

Let $Q = P_v \cup P_w \cup P_2$ be the union of P_v, P_w, P_2. Clearly, Q has a vw-walk that does not contain edge vw. Thus, $P_1 \cup P_2 - vw$ contains a vw-path. Then, $P_1 \cup P_2$ contains a cycle which is a contradiction to the definition of a tree.

Proposition

Let $T = (W, F)$ be a digraph. The following statements are equivalent.

(a) T is a tree.

(b) There is exactly one path between every pair of nodes in T.

(c) T is connected and $|F| = |W| - 1$.

(d) T has no cycles and $|F| = |W| - 1$.

1.4 Incidence Matrix

Definition: Node-Arc Incidence Matrix

A matrix M of $|N|$ rows and $|A|$ columns such that:

- The rows correspond to the nodes of D,
- The columns correspond to the arcs of D,
- The entry for node w and arc (i, j), denoted $m_{w,ij}$, is given by

$$m_{w,ij} = \begin{cases} 0 & \text{if } w \neq i \text{ and } w \neq j \\ +1 & \text{if } w = j \\ -1 & \text{if } w = i \end{cases}$$

Notation

Let $G = (V, E)$ be a digraph and M be its node-arc incidence matrix.

- For a subset of arcs $J \subseteq E$, let M_J be the submatrix of M formed by the columns corresponding to the arcs in J.
- Let $\widetilde{M}_j$ be the submatrix obtained by removing one arbitrary row from the submatrix M_J.

Proposition

Let $G = (V, E)$ be a digraph and let M be its node-arc incidence matrix, and let $J \subseteq E$.

(a) The submatrix M_J of M has linearly independent columns if and only if J contains no cycle.

(b) Suppose that $|E| = |V| - 1$. Then the $(|V| - 1) \times (|V| - 1)$ submatrix $\widetilde{M}_J$ is a nonsingular if and only if J is a spanning tree of G.

Chapter 2

Transshipment Problem

2.1 Introduction and LP Formulation

Transshipment Problem (TP)

Given a digraph $D = (N, A)$ along with $b \in \mathbb{R}^N$ node demands, and $w \in \mathbb{R}^A$ arc costs, find a flow $x \in \mathbb{R}^A$ of minimum cost, where a flow means a vector $x \in \mathbb{R}^A$ such that $Mx = b, x \geq 0$ and the cost of the flow is defined as $w^T x = \sum\limits_{ij \in A} w_{ij} x_{ij}$.

By the definition of a flow $x \in \mathbb{R}^A_+$, we have $Mx = b$. Thus for each node $i \in N$, we have the constraint

$$f_x(i) = row_i(M) \cdot x = \sum_{vi \in A} x_{vi} - \sum_{ij \in A} x_{ij} = x(\delta(\bar{i})) - x(\delta(i))$$

Definition: Flow

A vector $x \in \mathbb{R}^A_+$ that satisfies $Mx = b$.

LP Formulation of TP

$$\begin{aligned} \min \quad & w^T x = \sum_{ij \in A} w_{ij} x_{ij} \\ \text{s.t.} \quad & Mx = \sum_{ij \in A} x_{iv} - \sum_{vk \in A} x_{vk} = b_v, \ \forall v \in N \\ & x \geq 0 \end{aligned}$$

Dual LP of TP

$$\begin{aligned} \max \quad & b^T y = \sum_{v \in N} b_v y_v \\ \text{s.t.} \quad & y_v - y_u \le w_{uv}, \ \forall uv \in A \\ & y \text{ free} \end{aligned}$$

2.2 Complementary Slackness of TP

Theorem (Complementary Slackness Conditions for TP)

x and y are optimal solutions for the primal and dual LP if and only if the following holds for each arc $uv \in A$:

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

Since all constraints in the primal are equality constraints, there is only one condition.

2.3 Infeasibility, Unboundedness, and Basic Solutions

Theorem (Fundamental Theorem of Linear Programming)

Let (P) denote an LP problem.

(i) Either (P) is infeasible, or it is unbounded, or it has an optimal solution.

(ii) If (P) has a feasible solution, then it has a feasible solution that is basic.

(iii) If (P) has an optimal solution, then it has an optimal solution that is basic.

Theorem

An LP $\max\{c^T x : Ax = b, x \ge 0\}$ is infeasible if and only if there exists $y \in \mathbb{R}^m$ such that $y^T A \ge 0$ and $y^T b < 0$.

Theorem

Consider a TP with $b(N) = 0$. The TP is infeasible if and only if $\exists S \subseteq N$ such that $\delta(S) = \emptyset$ and $b(S) < 0$.

Proof. ($\Longleftarrow$) Let $S \subseteq N$ such that $b(S) < 0, \delta(S) = \emptyset$. Suppose $x \in \mathbb{R}^A_+$ is a feasible solution.

$$\begin{aligned} 0 > b(S) &= \sum_{v\in S}(row_v(M)\cdot x) \\ &= x(\delta(\overline{S})) - x(\delta(S)) \\ &= x(\delta(\overline{S})) - 0 \\ &\geq 0 \end{aligned}$$

This is a contradiction.

($\implies$) This proof follows from the infeasibility theorem for Minimum Cost Flows (MCF) or from max-flow min-cut theorem.

Theorem

An LP $\max\{c^T x : Ax = b, x \geq 0\}$ is unbounded if and only if the LP has a feasible solution and $\exists d \in \mathbb{R}^n$ where $d \geq 0$ such that $Ad = 0$ and $c^T d > 0$.

Theorem

Consider a TP with $b(N) = 0$. The TP is unbounded if and only if it has a feasible solution and its digraph has a dicycle of negative cost.

Definition: Tree Solution

Given a spanning tree T of D, the unique solution x of $Mx = b$ such that $x_{ij} = 0$ for all arcs ij not in T is called a tree solution.
A tree solution is a basic solution of (P).

Definition: Tree Flow

Given a spanning tree T of D, a flow $x \in \mathbb{R}^A_+$ such that $x_{ij} = 0$ for all arcs ij not in T is called a tree flow. In other words, a tree solution x that is nonnegative is called a tree flow.
A tree flow is a basic feasible solution of (P).

The node-arc incidence matrix does not have full row rank which is needed by the simplex algorithm. We can delete any one row of M to get the matrix $\tilde{M}$.

$$\text{rank}(\tilde{M}) = \text{rows} - 1$$

Theorem

For any set J of arcs, the submatrix M_J has linearly independent columns if and only if J contains no cycle.

2.4 The Dual

Definition: Node Potential

A vector $y \in \mathbb{R}^N$ is called a node potential.

Definition: Reduced Cost

Given node potentials $y \in \mathbb{R}^N$, the reduced cost of an arc ij is

$$\overline{w}_{ij} = w_{ij} + y_i - y_j$$

Definition: Feasible

A node potential $y \in \mathbb{R}^N$ is called feasible if $\overline{w}_{ij} \geq 0$ for all $ij \in A$.

2.5 Network Simplex Algorithm for TP

We make a few assumptions:

- Node demands sum to zero, i.e. $\sum_{v \in N} b_v = 0$.
- Digraph is connected, i.e. D contains a spanning tree.
- For all node pairs i, j, at most one of the arcs ij or ji is present

Definition: Oriented Cycle

An oriented cycle means a undirected cycle together with a specified orientation.
An arc ij of an oriented cycle Q is called forward if j is the immediate successor of i of the orientation and reverse otherwise.

Computing a tree solution: Given a spanning tree T which defines the basis, we have to find a solution $x \in \mathbb{R}^A$ to the system:

$$\{Mx = b : x_{ij} = 0 \; \forall ij \notin T\}$$

There are two ways to do this: linear algebra or operations on digraphs.

- Linear algebra: Permute the rows and columns of the incidence matrix M_T of T such that deleting the first row gives a truncated incidence matrix $\tilde{M}_T$ that is upper-triangular and has all diagonal entries in $\{\pm 1\}$.

  This property holds because we can number of the nodes of T as $0, 1, \ldots, n-1$ such that the arcs in T can be numbered as $e_1, \ldots, e_{n-1}$ such that arc e_i has one end at i and the other end among $0, \ldots, i-1$.

We solve the resulting triangular system to get the tree solution:

$$x_T = \tilde{M}_T^{-1}\tilde{b}$$

- Examine all the arcs in T in some sequence $e_1, \ldots, e_{n-1}$ such that for $e_k = ij$, one of the end nodes i or j, we have already determined the x value for each of the other arcs incident to the node i or j respectively. Thus, x_{ij} is not determined, but all other variables in the equation constraint for node i (or j), namely $\sum\limits_{vi \in A} x_{vi} - \sum\limits_{ik \in A} x_{ik} = b_i$, have been determined, then we can determine x_{ij} from this equation.

Computing an initial tree flow: We need a tree flow and its spanning tree to start NSM. This is nontrivial and to do this we used the two-phase simplex algorithm to determine an initial BFS.

Start with an auxiliary TP that has additional artificial arcs such that there exists a spanning tree T consisting of artificial arcs. We apply the NSM to the auxiliary TP. In this course, we will assume we are given a spanning tree whose tree solution is ≥ 0 (tree flow).

Computing node potentials y corresponding to a spanning tree T: Consider the problem of finding $y \in \mathbb{R}^N$ such that $y^T M_T = w_T^T$. This is the dual solution for a given basis T. Can do this using linear algebra or operations on digraphs.

- Solving a lower-triangular system where diagonal entries are ± 1.
- 1. Pick any node of T to be the root r
  2. Orient all arcs of T away from r to get an oriented tree $\hat{T}$
     - for each forward arc ij of $\hat{T}$, fix the cost to be w_{ij}
     - for each reverse arc ij of $\hat{T}$, fix the cost to be $-w_{ij}$
  3. For each node v, fix y_v to be the cost of unique rv-path of $\hat{T}$.

Computing Leaving Arc & Updated Tree Flow:

1. Let uv be the entering arc.
2. Let Q^{und} be the unique cycle of $T + uv$ (ignore arc directions).
3. Orient Q^{und} such that uv is forward and let Q bet he resulting oriented cycle.
4. Choose the leaving arc pq to be a reverse arc of Q with smallest x value, thus, $\gamma = x_{pq} = \min\{x_{ij} | ij \text{ reverse in } Q\}$.

   If Q has no reverse arc, stop and report unbounded TP.
5. $T^{new} = (T - pq) + uv$, $x^{new} = x + \gamma \hat{x}^Q$ where $\hat{x}^Q$ is the signed incidence vector of Q.

$$\hat{x}_{ij}^Q = \begin{cases} 0 & ij \notin Q \\ +1 & ij \text{ forward in } Q \\ -1 & ij \text{ reverse in } Q \end{cases}$$

Algorithm 1 Network Simplex Algorithm (TP)

```
1: Find a spanning tree $T$ of $D$ such that the tree solution $x$ is nonnegative
       If no tree can be found, Stop, TP has no feasible solution
       To find the tree solution $x$ for a given $T$, use inspection or solve $M_T x_T = b$
2: Using $T$ and $w$, find $y \in \mathbb{R}^N$ such that each arc $ij \in T$ has $\overline{w}_{ij} = 0$ ($y$ need not be
   feasible for dual)
       To find $y$, use inspection or solve $M_T' y = w_T$, where $M_T'$ denotes the transpose of
   $M_T$
3: while Arc $ij \in D$ with $\overline{w}_{ij} = w_{ij} + y_i - y_j < 0$ do
4:     Find arc $uv$ that $\overline{w}_{uv} < 0$
5:     Let $Q$ be the oriented cycle obtained from $T + uv$ cycle by choosing the orientation
   where $uv$ is a forward arc
6:     if All arcs in $Q$ are forward then
7:         STOP; TP is unbounded
8:     Let $pq$ be reverse arc of $Q$ such that

           $x_{pq} = \min\{x_{ij} | ij \text{ is a reverse arc of } Q\}$

   and let $\gamma = x_{pq}$
9:     Push $\gamma$ units of flow along $Q$, i.e.
         • For each forward arc $ij$ in $Q$, increase $x_{ij}$ by $\gamma$
         • For each reverse arc $ij$ in $Q$, decrease $x_{ij}$ by $\gamma$
10:    $T \leftarrow T + uv - pq$
11:    Recompute $y \in \mathbb{R}^N$ using new $T$
```

2.6 Auxiliary TP

Given a digraph D and node demands $b \in \mathbb{R}^N$, we want to find a feasible basis/spanning tree and a tree flow.

To do this, we can create an auxiliary TP consisting of a digraph $D' = (N', A')$, node demands $b' \in \mathbb{R}^{N'}$, and arc costs $w' \in \mathbb{R}^{A'}$ where

- $N' = N \cup \{z\}$ where z is a new node.
- $A' = A \cup \{vz : v \in N, b_v < 0\} \cup \{zv : v \in N, b_v \geq 0\}$
- $b'_z = 0, b'_v = b_v \ \forall v \in N$
- $w'_e = \begin{cases} 0 & e \in A \\ 1 & e \in A' \setminus A \end{cases}$

We choose the new auxiliary arcs as the spanning tree.

Proposition

The TP has a feasible tree flow if and only if its auxiliary TP has optimal value 0.

2.7 NSM Cycling

Definition: Degenerate Pivot

A pivot (iteration) of the Network Simplex Method is called degenerate if $\gamma = 0$.

Definition: Cycling

If there is a sequence of degenerate pivots such that the first spanning tree and the last spanning tree are the same.

Definition: Strongly Feasible Basis

A spanning tree/basis is called strongly feasible if every arc ij with $x_{ij} = 0$ is directed away from the root.

Theorem (Cunningham's Anti-Cycling Rule)

In each degenerate pivot, the entering arc is directed away from the root in the new spanning tree.

Chapter 3

Shortest Directed Paths

Shortest st-Dipath Problem

Given a digraph $D = (N, A)$ in which each arc $a \in A$ is assigned some real value w_a. We write $w(A')$ for the sum over all arc weights in A'. Let $s, t \in N$. Find a st-dipath of minimum cost length.

3.1 Optimality Conditions

Proposition

Let D be a digraph with weights $w \in \mathbb{R}^A$ which has no negative dicycle. Let Q be an st-diwalk, where $s \neq t$. Then there exists an st-dipath P with $w(P) \leq w(Q)$.

Proof. Q can be decomposed into an st-dipath and a collection of dicycles $C_1, \dots, C_k$. Thus, $w(Q) = w(P) + w(C_1) + \cdots + w(C_k) \geq w(P)$ since there are no negative dicycles.

Proposition

Let D be a digraph with weights $w \in \mathbb{R}^A$ which has no negative dicycle. Let P be the shortest st-dipath and Q be a shortest st-diwalk. Then, $w(P) = w(Q)$.

By previous proposition, $w(P) \leq w(Q)$. Since every st-dipath is an st-diwalk, then $w(Q) \leq w(P)$. So, $w(P) = w(Q)$.

Definition: Node Potential

An entry in the vector $y \in \mathbb{R}^N$.

Definition: Reduced Cost

In vector notation

$$\overline{w} = w - M^T y$$

or in scalar notation

$$\overline{w}_{ij} = w_{ij} + y_i - y_j, \forall ij \in A$$

Definition: Negative Dicycle

A dicycle C of negative cost. i.e. $w(C) = \sum_{ij \in C} w_{ij} < 0$.

Lemma

Let Q be an st-diwalk. Then

$$\overline{w}(Q) = w(Q) + y_s - y_t$$

Proof. Let Q be the st-diwalk $v_1 v_2, \ldots, v_{k-1} v_k$ where $v_1 = s$ and $v_k = t$.

$$\begin{aligned}\overline{w}(Q) &= \sum_{i=1}^{k-1} \overline{w}_{v_i, v_{i+1}} \\ &= \sum_{i=1}^{k-1} (w_{v_i, v_{i+1}} + y_{v_i} - y_{v_{i+1}}) \\ &= w(Q) + y_s - y_t\end{aligned}$$

Corollary

For any two st-dipaths P_1, P_2, we have

$$w(P_1) - w(P_2) = \overline{w}(P_1) - \overline{w}(P_2)$$

Corollary

For every dicycle C, we have $w(C) = \overline{w}(C)$.

Definition: Feasible Node Potentials

Node potentials y are feasible if

$$y_u + w_{uv} \geq y_v \text{ or equivalently } \overline{w} \geq 0$$

for all $uv \in A$.

Definition: Equality Arc

An arc uv for which $y_u + w_{uv} = y_v$ or $\overline{w}_{ij} = 0$.

Lemma

Let $y \in \mathbb{R}^N$ be a feasible potential. Let Q an st-diwalk of D. Then $w(Q) \geq y_t - y_s$. Moreover, $w(Q) = y_t - y_s$ if and only if every arc of Q is an equality arc.

Proof. By previous lemma, $w(Q) + y_s - y_t = \overline{w}(Q) \geq 0$ since y is feasible. Hence, $w(Q) \geq y_t - y_s$.

Suppose $w(Q) = y_t - y_s$, then $\overline{w}(Q) = 0$. Since $\overline{w}(Q) = \sum\{\overline{w}_{ij} : ij \in Q\}$ and each reduced cost is ≥ 0, we have $\overline{w}_{ij} = 0$ for all $ij \in Q$.

Theorem (Optimality for Shortest Dipaths)

Let $D = (N, A)$ be a digraph, let $w \in \mathbb{R}^A$ be weights, and let P be an st-dipath. Then P is a shortest st-dipath if there exists feasible potentials y such that all arcs of P are equality arcs.

Lemma

If D has a negative dicycle, then D has no feasible potentials.

Proof. Let C be a negative dicycle of D. Then for any $y \in \mathbb{R}^n$, $\overline{w}(C) = w(C) < 0$.

Suppose $\overline{y} \in \mathbb{R}^n$ is a feasible potential, then $\overline{w}_{ij} \geq 0$ for all $ij \in A$ with respect to $\overline{y}$. So $\overline{w}(C)$ would be ≥ 0, which is a contradiction.

Lemma

Let D be a digraph with weights $w \in \mathbb{R}^A$ where all nodes can be reached from node s. For every $v \in N$, let y_v be the length of the shortest sv-dipath in D. If D has no negative dicycles, then y are feasible potentials.

Proof. By contradiction, some arc uv violates condition for feasible potentials. Thus, $\overline{w}_{uv} = w_{uv} + y_u - y_v < 0 \implies w_{uv} + y_u < y_v$. Let P be a shortest su-dipath in D. By definition of y, $w(P) = y_u$. Let Q be the sv-diwalk obtained by adding arc uv to the end of P. It follows, by proposition and no negative dicycle, that Q contains an sv-dipath Q' where $w(Q') \leq w(Q)$. But $w(Q) = w(P) + w_{uv} = y_u + w_{uv} < y_v$. Hence, y_v is not the length of the shortest sv-dipath in D, a contradiction.

Theorem

Let $D = (N, A)$ be a digraph with arc costs $w \in \mathbb{R}^A$. Feasible potentials exist if and only if no negative dicycles exist.

Theorem

Let $D = (N, A)$ be a digraph with arc costs $w \in \mathbb{R}^A$ and assume that D has no negative dicycles. P is a shortest st-dipath if and only if there exist $y \in \mathbb{R}^N$ such that y are feasible potentials.
y and all arcs in P are equality arcs is a certificate for a shortest dipath P.

Proof. ($\implies$) Consider feasible potentials defined in previous lemma. Then, $w(P) = y_t = y_t - 0 = y_t - y_s$ and this implies that every arc of P is an equality arc.

3.2 Linear Programming Interpretation

Definition: Characteristic Vector of a Path

Let P be an st-dipath of D. Then we represent P as a vector $x^P \in \mathbb{R}^A$ where $x_a = 1$ if $a \in P$ and $x_a = 0$ otherwise.

Proposition

Let u be a node not in P, then $x^P(\delta(u)) = x^P(\delta(\overline{u})) = 0$.
Let v be a node of P distinct from t, then $x^P(\delta(v)) = 1$.
Let v be a node of P distinct from s, then $x^P(\delta(\overline{u})) = 1$.

It follows that $f_{x^P}(u) := x^P(\delta(\overline{u})) - x^P(\delta(u)) = 0$ for all $u \in N \setminus \{s, t\}$ and $f_{x^P}(t) = 1, f_{x^P}(s) = -1$. Thus, x^P is a feasible solution to the shortest st-dipath problem.

LP Formulation of Shortest st-Dipath

$$\begin{aligned} \min \quad & w^T x \\ \text{s.t.} \quad & f_x(u) = x(\delta(\overline{u})) - x(\delta(u)) = \begin{cases} +1 & u = t \\ -1 & u = s \\ 0 & \text{otherwise} \end{cases} \\ & x \geq 0 \end{aligned}$$

Note that not all solutions to the LP correspond to an st-dipath.

Dual LP of Shortest st-Dipath

$$\begin{aligned} \max \quad & y_t - y_s \\ \text{s.t.} \quad & y_v - y_u \leq w_{uv}, \ \forall uv \in A \\ & y_e \geq 0 \end{aligned}$$

Theorem (Complementary Slackness Conditions)

For all $uv \in A$, if $x_{uv} > 0$, then
$$y_v - y_u = w_{uv}$$

3.2.1 Reduced Cost Interpretation

We know if Q is an st-diwalk, then $\overline{w}(Q) = w(Q) + y_s - y_t$. For dicycle C, $\overline{w}(C) = w(C)$. This also implies that for two st-dipaths P, P', $w(P) \le w(P')$ if and only if $\overline{w}(P) \le \overline{w}(P')$.

Proposition

D has a negative dicycle with costs w if and only if D has a negative dicycle with costs $\overline{w}$.
P is a shortest st-dipath with costs w if and only if P is a shortest st-dipath with costs $\overline{w}$.

3.3 Shortest Dipaths From a Fixed Node

Definition: Rooted Tree from s

Let $s \in N$, we say that T is a tree rooted from s if for every node v, there exists an sv-dipath.

Lemma

Let $D = (N, A)$ be a digraph with node s. Then D has a spanning tree rooted from s if and only if every node can be reached from s.

Proof. If there is a spanning tree T rooted at s, then every node is reachable from s in T and hence every node is reachable from s.

Suppose every node can be reached from s. Choose a tree $T = (N', A')$ rooted at s and contained in D with as many nodes as possible. Such a tree exists since we can choose $N' = \{s\}$ and $A' = \emptyset$. We may assume $N' \subset N$, else T is already a spanning tree.

Let $t \in N \setminus N'$. By assumption, t is reachable from s. There exists an arc $uv \in \delta(N')$. But then $T + uv$ is a tree with more nodes than T, a contradiction to choice of T.

Theorem (Optimality for Tree of Shortest Dipaths)

Let $D = (N, A)$ be a digraph with weights $w \in \mathbb{R}^A$ and let T be a spanning tree rooted from s. Then T is a tree of shortest dipaths if there exists feasible potentials y such that all arcs of T are equality arcs.

Proof. Suppose we have feasible potentials y such that all arcs T are equality arcs. Then, apply optimality of shortest dipaths theorem to each su-dipath in T and deduce T is a tree of shortest dipaths.

Theorem (Existence of Tree of Shortest Dipaths)

Let $D = (N, A)$ be a digraph with weights $w \in \mathbb{R}^A$ and $s \in N$. Suppose all nodes are reachable from s and that D has no negative dicycle. Then D has a tree of shortest dipaths T rooted from s.

Proof. For every node v, fix y_v to be cost of a shortest sv-dipath of D. y is a feasible potential. Focus on the set of equality arcs $A^=$ with respect to y.

For any node v, every shortest sv-dipath is contained in $A^=$. Thus, every node is reachable from s in the digraph $(N, A^=)$. There exists a tree rooted from s in $A^=$, call it $T^=$.

For all nodes v, the sv-dipath of $T^=$ is a shortest sv-dipath of D.

3.4 Dijkstra's Algorithm

Consider a digraph $D = (N, A)$ with nonnegative weights $w \in \mathbb{R}^A_+$ and a node s. Dijkstra's algorithm finds a tree of shortest dipaths rooted from s.

Algorithm 2 Dijkstra's Algorithm

```
 1: Input: Digraph $D = (N, A)$ with arcs costs $w_a \geq 0$ and node $s$
 2: Output: Tree $T$ of shortest dipaths rooted from $s$
        $y_u : u \in N$ are the lengths of the shortest $su$-dipaths.
 3: $y_u = 0$ for all $u \in N$
 4: while True do
 5:     $A^=$ be the set of equality arcs
 6:     $D^= = (N, A^=)$
 7:     $S$ be the set of nodes reachable from $s$ in $D^=$
 8:     if $S = N$ then break
 9:     Increase uniformly the potential of each node $v \in N - S$ until some arc $uv \in A - A^=$
    becomes an equality arc:
10:     $\epsilon = \min\{\overline{w}_{ij} : ij \in \delta_D(S)\}$
11:     $y_v = y_v + \epsilon$ for all $v \in N \setminus S$
12:     $S = S \cup \{v\}$
13: return $T$ be any spanning tree rooted at $s$ in $D^=$
```

Proposition

Suppose at the start of Dijkstra's algorithm, y is a feasible potential. Then at the end of every iteration (except the last), at least one arc in $\delta(S)$ becomes an equality arc and y^{new} is a feasible potential.

Proof. Consider any arc ij and the change to $\overline{w}_{ij}$ when we update the vector y to y^{new} (by increasing $y_j, \forall j \in N - S$ by ϵ).

Case 1: $i \in S, j \in S$
$\overline{w}_{ij}$ stays the same for y and y^{new}

Case 2: $i \notin S, j \notin S$
$\overline{w}_{ij}$ stays the same for y and y^{new}

Case 3: $i \notin S, j \in S$
$\overline{w}_{ij}$ increases by ϵ since y_i^{new} increases by ϵ and y_j^{new} stays the same.

Case 4: $i \in S, j \notin S$
ij is in the cut $\delta(S)$, so $\overline{w}_{ij}$ decreases by ϵ since y_i^{new} stays the same and y_j^{new} increases. Since we fix ϵ to be $\min\{\overline{w}_{ij} : ij \in \delta(S)\}$,

$$\overline{w}_{ij}^{new} \geq 0$$

for all $ij \in \delta(S)$, and is equal to zero for at least one arc ij.

3.5 Acyclic Digraphs

3.5.1 Topological Ordering

Definition: Topological Ordering

D admits a topological ordering if we can label the nodes of D from 1 to $n := |N|$ such that for every arc $ij \in A$, we have $i < j$.

Theorem

A digraph D is acyclic if and only if it admits a topological ordering.

Algorithm 3 Topological Ordering

```
1: **Input**: Digraph $D = (N, A)$
2: **Output**: If $D$ has no dicycle, finds a topological ordering of the nodes
3: $i = 1$
4: while $\exists v$ with $\delta(\overline{v}) = \emptyset$ do
5:     $v = i$
6:     $i = i + 1$
7:     Delete $v$ and all arcs with tail $v$
8: if No nodes left then
9:     return Ordering $i_1, \dots, i_n$
10: else
11:     return $\exists$ dicycle
```

3.5.2 Shortest Dipaths in Acyclic Digraphs

Algorithm 4 Acyclic Shortest Dipath Algorithm

```
1: Input: Digraph $D$ with nodes $1, \dots, n$ from topological ordering
2: Output: Finds tree $T$ of shortest dipaths from 1
       $y_i : i = 1, \dots, n$ are the lengths of the shortest $1i$-dipaths
3: $y_1 = 0, T = (\{1\}, \emptyset)$
4: for $j = 2$ to $n$ do
5:     $y_j := \min\{y_i + w_{ij} : 1 \leq i \leq j - 1, ij \in A\}$
6:     Let $i^*$ such that $y_j = y_{i^*} + w_{i^*j}$
7:     $T = T + i^*j$
```

Proposition

To find the longest dipaths from s in an acyclic digraph, define $w'_{uv} = -w_{uv}$ for every arc uv and run the shortest dipath algorithm with w' to find y. $-y_u$ is the length of the longest su-dipath.
We can also run the above algorithm with max instead of min.

3.5.3 Project Evaluation and Review Technique (PERT)

Definition: Makespan

The earliest time by which all jobs can be completed while respecting the precedence constraints.

Definition: PERT

A method for managing large projects consisting of many tasks, having varying duration, subject to precedence constraints.
A set J of jobs where each job $j \in J$ has length l_j, which is the time to perform j and a possibly empty set $P(j)$ of predecessor jobs that must all be completed prior to the start of j. Assume no resource restrictions, i.e., we can perform as many tasks as we wish in parallel, obeying precedence constraints.
The goal is to find a feasible schedule (a list of completion times of the jobs), respecting the precedence constraints that achieves the project makespan.

We can represent the project by a digraph D which is the digraph representing the precedence constraints appended with a source s and sink t. We have a start node s, end node t, and a node j for each job $j \in J$.

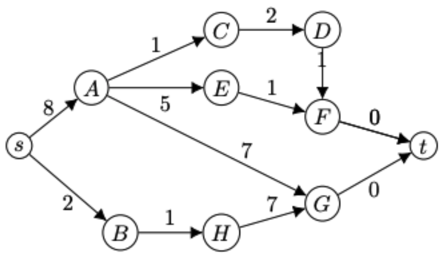
s and t represent the time points when the project starts and ends. Each job node j represents the time point when j is completed. We have an arc ij from job i to j whenever $i \in P(j)$. Since j may only complete l_j time units after i completes, we give arc ij a weight of l_j.

Observe the first job to start in any feasible schedule must be a job with no predecessors. So we have arc sj for every job j with $P(j) = \emptyset$ and weight l_j on these arcs to encode that j can only complete l_j time units after the project starts. Similarly, the last job to complete in any feasible schedule does not have any successors. So for any arc kt for every job k such that $k \notin P(j)$. We give 0 weight to arc kt since if k is the last job to complete, then the project finishes when k is completed.

D must be acyclic because if there was a dicycle, it would only contain job nodes which represents an impossible situation where every job in the dicycle is waiting for another job in the dicycle to complete, so no job can ever be started.

Example:

Task	Duration	Predecessors
A	8 hours	-
B	2 hours	-
C	1 hour	A
D	2 hours	C
E	5 hours	A
F	1 hour	D, E
G	7 hours	A, H
H	1 hour	B



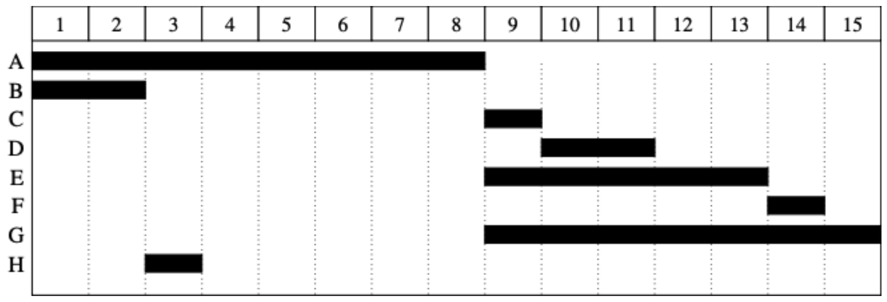
Definition: Earliest Scheduling Rule

Let $d(u,v)$ denote the length of the longest uv-dipath. A longest st-dipath computation find an optimal schedule. Consider the rule

1. Find $d(s,u)$ for every job node u.
2. Complete job u at time $d(s,u)$; that is, start u at time $d(s,u) - l_u$.

The resulting schedule is called an **early schedule**.

Node u	s	A	B	H	G	C	D	E	F	t
Topological ordering	1	2	3	4	5	6	7	8	9	10
$d(s,u)$	0	8	2	3	15	9	11	13	14	15



Claim

Let v be a job node and P be any sv-dipath. Then, in any feasible schedule, the completion time of v is at least $w(P)$.

Proof. Let $P = s_1, v_1, \dots, v_{k-1}, v_k$ where $k \geq 1$ and $v_k = v$. All arcs $v_i v_{i+1}$ for $i = 1, \dots, k-1$ denote precedence constraints. So v's completion time in any feasible schedule is at least $\sum_{i=1}^{k} l_{v_i} = w(P)$.

Proposition

The earliest scheduling rule gives a feasible and optimal schedule.

Proof. To prove feasibility, we show that we satisfy all precedence constraints. So whenever $i \in P(j)$, we need to argue that $d(s,j) - d(s,i) \geq l_j$. Let P be a longest si-dipath. Note

$ij \in A$ with weight l_j. Appending ij to P yields an sj-dipath (since D is acyclic) and so by definition, $d(s, j) \geq w(P) + l_j = d(s, i) + l_j$.

To prove optimality, we note that all jobs complete by time $d(s, t)$. $d(s, t)$ is at most the makespan. It follows that the makespan is $d(s, t)$ and the early scheduling rule produces an optimal schedule.

3.6 Dantzig's Algorithm

This algorithm is for general digraphs and general weights. Assume that every node is reachable from s. There is a spanning tree T for every step. An arc in D not in T is called a non-tree arc. Denote P_u to be the unique su-dipath in T.

Algorithm 5 Dantzig's Algorithm

```
1: **Input**: Digraph $D$ with arc costs $w \in \mathbb{R}^A$
2: **Output**: Tree $T$ of shortest dipaths from $s$; if $D$ has a negative dicycle, then it outputs
   this cycle
3: Find initial spanning tree $T$ rooted at $s$
4: For every $u \in N$, denote $P_u$ be the $su$-dipath and $y_u = w(P_u)$
5: Find a non-tree arc $uv$ such that $\overline{w}_{uv} = w_{uv} + y_u - y_v < 0$
6: **if** No arc $uv$ exists **then**
7:     **return** $T$
8: **if** $v \in P_u$ **then**
9:     **return** $T + uv$ has negative dicycle
10: $zv$ be last arc of $P_v$
11: $T = T + uv - zv$
```

Suppose we want to find the shortest dipath between every pair of nodes. Instead of applying Dantzig's algorithm n times, the idea is to run Dantzig's algorithm once from s. Then, we know from previous proposition that since y is feasible, the shortest dipath is the shortest if and only if it is the shortest with reduced cost weights, so we can work with reduced costs. We can use Dijkstra's algorithm to find the shortest dipath starting from every other node.

3.7 Bellman-Ford Algorithm

Idea: Set node potentials (infeasible). If an arc violates the dual constraint ($\overline{w}_{uv} < 0$), then try to fix it.

To fix it, change $y_v = y_u + w_{uv}$.

When we fix an arc, it becomes a tight arc, so it is eligible to be part of any shortest st-dipath.

Definition: Predecessor

In a spanning tree T rooted at s, for each node $v \in N \setminus \{s\}$, its predecessor is the unique node u so that uv is an arc in T. This is denoted $p_v = u$.

Definition: Predecessor Digraph

$D_p = (N, \{p_v v : v \in N, p_v \text{ is defined}\})$.

We put all arcs in a fixed sequence. In each iteration, we look at all arcs once in sequence and fix arcs. Do this until we get a feasible potential or we have completed $|N| - 1$ iterations

Algorithm 6 Bellman-Ford Algorithm

```
1: **Input**: Digraph $D = (N, A)$, $w \in \mathbb{R}^A$, $s \in N$
2: $y_s = 0$
3: $y_v = \infty$ for all $v \in N \setminus \{s\}$
4: $p_v =$ undef for all $v \in N$
5: $i = 0$
6: while $i \leq |N| - 1$ and $y$ is not a feasible potential do
7:     $i = i + 1$
8:     for $uv \in A$ do
9:         if $\overline{w}_{uv} < 0$ then
10:            $y_v = y_u + w_{uv}$ (if $y_u = y_v = \infty$, do not fix $uv$)
11:            $p_v = u$
12: if $i = |N| - 1$ and $y$ is not feasible then
13:     return $D$ has a negative dicycle
14: else
15:     return $D_p$
```

3.7.1 Analysis of Bellman-Ford

Assume all nodes are reachable from s and outcomes predecessor digraph D_p is either a tree of shortest dipaths rooted from s or D_p contains a negative dicycle.

Claim

For each node v, y_v does not increase.

Proof. By each step of the algorithm.

Claim

At termination of the algorithm, y_v is finite for each node v and p_v is defined for each node $v \neq s$.

Proof. By assumption.

Claim

At termination, either D_p is a tree rooted from s or D_p contains a dicycle.

Claim

Let uv be an arc where $u = p_v$ at termination of the algorithm. If D has no negative dicycle, then $a_u < a_v$ and uv is an equality arc with respect to node potentials $y^{(n-1)}$.

Claim

$\overline{w}_{uv} \leq 0$ for all arcs uv in D_p.

Proof. Sketch: At the step when we add uv to D_p, we have $y_v = w_{uv} + y_u \iff \overline{w}_{uv} = 0$. At a later step, if we update y_v, then the same holds. By the first claim, y_u decreases, then $\overline{w}_{uv}$ becomes negative.

Claim

If D_p contains a dicycle Q, then Q is a negative dicycle.

Recall: $w(Q) = \overline{W}(Q)$.

Claim

$a_u \geq 1$ for all nodes $u \neq s$.
Also, $a_s = 0$ if D has no negative dicycles.

Theorem

Bellman-Ford algorithm correctly determines if D has negative dicycles. If D has no negative dicycles, then the tree T returns is a shortest path tree from s, the potentials $y^{(n-1)}$ are feasible, and all arcs of T are equality arcs with respect to $y^{(n-1)}$.

Chapter 4

Maximum Flow

Given a digraph D with capacity constraints, how many arc-disjoint st-paths are there? We can create a digraph D' where we create c_{uv} copies of the arc uv. This is the motivating problem.

Maximum st-Flow Problem

Given a digraph $D = (N, A)$ where each arc uv has capacity $c_{uv} \in \mathbb{R}, c_{uv} \geq 0$. We have a *source* node s and a *sink* node t. The goal is to push as much flow from s to t.

The amount of flow leaving u is $x(\delta(u))$, the amount of flow entering u is $x(\delta(\overline{u}))$, and the net amount of flow entering u is $f_x(u) = x(\delta(\overline{u})) - x(\delta(u))$.

For every node $u \in N \setminus \{s,t\}$, $f_x(u) = 0$, these constraints are called flow conservation constraints. The net flow entering t is equal to the net flow leaving s, so $f_x(t) = -f_x(s)$.

LP Formulation of Maximum st-Flow

$$\begin{aligned} \max \quad & f_x(t) \\ \text{s.t.} \quad & f_x(u) = 0, \ \forall u \in N \setminus \{s,t\} \\ & 0 \leq x_{uv} \leq c_{uv}, \ \forall uv \in A \end{aligned}$$

Definition: st-Flow

A feasible solution x to the maximum st-flow LP.

Definition: Value of st-Flow

$$f_x(t)$$

We can add an arc ts with cost -1 and assign other arcs with cost 0, which converts the problem into a minimum cost flow problem or transshipment problem with capacity constraints. The constraints can now be written as $Mx = b$.

4.1 Augmenting Paths

Definition: Forward Arc

We say an arc uv of an st-path P is a forward arc if uv is directed from s to t.

Definition: Backward Arc

If uv is not a forward arc, then uv is a backward arc.

Definition: Residual Capacity

Denoted $\gamma(P)$, of the st-path P is the minimum value in

$\gamma(P) = \min\{c_{uv} - x_{uv} : uv \text{ is a forward arc of } P\} \cup \{x_{uv} : uv \text{ is a backward arc of } P\}$

Definition: Incrementing Path

We say a path P is an incrementing path if P is an st-path and $\gamma(P) > 0$.

Definition: Push Flow Along P

$x' \in \mathbb{R}^A$ is obtained from x by pushing flow along P if for all $uv \in A$,

$$x'_{uv} = \begin{cases} x_{uv} + \gamma(P) & \text{if } uv \text{ forward arc of } P \\ x_{uv} - \gamma(P) & \text{if } uv \text{ backward arc of } P \\ x_{uv} & \text{otherwise} \end{cases}$$

Lemma

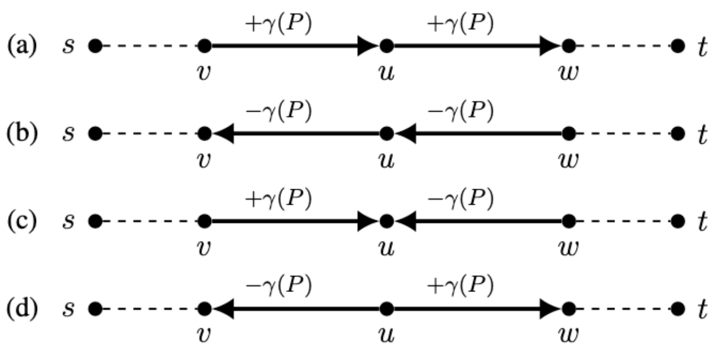
Let x be an st-flow and let x' be obtained by pushing flow along an incrementing path P. Then x' is an st-flow and $f_{x'}(t) = f_x(t) + \gamma(P)$.
In particular, x is not maximum.

Proof. Let uv be a forward arc of P. Then $x'_{uv} = x_{uv} + \gamma(P) \leq x_{uv} + (c_{uv} - x_{uv}) \leq c_{uv}$. Let uv be a backward arc of P. Then $x'_{uv} = x_{uv} - \gamma(P) \geq x_{uv} - x_{uv} \geq 0$. Thus, the capacity and nonnegativity constraints are satisfied for x'. Otherwise, let v be the node preceding u in P and let w be the node following u in P.

- For (a), $x'(\delta(u)) = x(\delta(u)) + \gamma(P)$ and $x'(\delta(\overline{u})) = x(\delta(\overline{u})) + \gamma(P)$.
- For (b), $x'(\delta(u)) = x(\delta(u)) - \gamma(P)$ and $x'(\delta(\overline{u})) = x(\delta(\overline{u})) - \gamma(P)$.
- For (c) and (d), $x'(\delta(u)) = x(\delta(u))$ and $x'(\delta(\overline{u})) = x(\delta(\overline{u}))$.

In all cases, $f_{x'}(u) = f_x(u) = 0$. Hence, x' is an st-flow. Let vt be the last arc of P. Then

$$f_{x'}(t) = f_x(t) + (x'_{vt} - x_{vt}) = f_x(t) + \gamma(P)$$



Definition: Residual Digraph

$D' = (N, A')$ of D where $uv \in A'$ if either

- $uv \in A$ and $x_{uv} < c_{uv}$, *or*
- $vu \in A$ and $x_{vu} > 0$.

Definition: Augmenting Path

An st-dipath in the residual digraph D'. There is an augmenting path if and only if there is an incrementing path in D

Use augmenting path as a synonym to incrementing path.

Lemma

Let D be a digraph with capacities c and an st-flow x. If the residual digraph D' has an st-dipath, then x is not a maximum flow.

4.2 Max-Flow Min-Cut

Lemma

Let $D = (N, A)$ be a digraph, $U \subseteq N$, and $x \in \mathbb{R}^A$. Then

$$\sum_{v \in U} f_x(v) = x(\delta(\overline{U})) - x(\delta(U))$$

Lemma

Let x be an st-flow and let $\delta(S)$ be an st-cut.

$$f_x(t) = x(\delta(S)) - x(\delta(\overline{S}))$$

Proof. Apply previous lemma for the case where $U = \overline{S}$. Then $\sum_{v \in \overline{S}} f_x(v) = x(\delta(S)) -$

$x(\delta(\overline{S}))$. But for all $u \in \overline{S} \setminus \{t\}$, $f_x(u) = 0$, thus $\sum_{v \in \overline{S}} f_x(v) = f_x(t)$.

Definition: Capacity of st-Cut

$$c(\delta(S)) = \sum_{a \in \delta(S)} c_a$$

Lemma

Let x be an st-flow and let $\delta(S)$ be an st-cut. Then $f_x(t) \leq c(\delta(S))$.
Moreover, $f_x(t) = c(\delta(S))$ if and only if $x_{uv} = c_{uv}$ for all arcs $uv \in \delta(S)$ and $x_{uv} = 0$ for all arcs $uv \in \delta(\overline{S})$.

Proof. By previous lemma,

$$f_x(t) = x(\delta(S)) - x(\delta(\overline{S})) \leq x(\delta(S)) \leq c(\delta(S))$$

The first inequality follows from $x_{uv} \geq 0$. It is strict if and only if $x_{uv} > 0$ for some $uv \in \delta(\overline{S})$. The second inequality follows from $x_{uv} \leq c_{uv}$. It is strict if and only if $x_{uv} < c_{uv}$ for some $uv \in \delta(S)$.

We cannot get equality if any one of the arcs gives a strict inequality. Thus, $x_{uv} = c_{uv}$ if $uv \in \delta(S)$ and $x_{uv} = 0$ if $uv \in \delta(\overline{S})$.

This shows that the value of an st-flow can never exceed the capacity of any st-cut. If the value of an st-flow x is f and the capacity of an st-cut $\delta(S)$ is f, then x is a maximum st-flow and $\delta(S)$ is an st-cut of minimum capacity.

Lemma (Key Lemma)

Consider a digraph $D = (N, A)$ with and st-flow x and suppose D has no augmenting path. Then there exists an st-cut $\delta(S)$ such that $f_x(t) = c(\delta(S))$.

Proof. Since there is no augmenting path in D, there is no st-dipath in the residual digraph $D' = (N, A')$. From previous theorem, there exists an st-cut $\delta_{D'}(S)$ where $\delta_{D'}(S) = \emptyset$. It follows from the definition of the residual digraph D' that if $uv \in A$, then $x_{uv} = c_{uv}$ and if $vu \in A$, then $x_{vu} = 0$. Now previous lemma implies that $f_x(t) = c(\delta(S))$.

Theorem (Augmenting Path)

An st-flow x is maximum if and only if there is no augmenting path.

Proof. If there is an augmenting path, then x is not maximum by a previous lemma.

Conversely, suppose there is no augmenting path. Previous lemma states that there is an st-cut $\delta(S)$ with $f_x(t) = c(\delta(S))$. But then this implies x is a maximum st-flow.

Theorem (Max-Flow Min-Cut)

The maximum value of an st-flow is equal to the minimum capacity of an st-cut.

Proof. The maximum st-flow problem can be formulated as an integer program. Because the value of the maximum st-flow is bounded, it follows from LP theory that there exists a maximum st-flow x. There are no more augmenting paths for x. It follows from the Key lemma that there is an st-cut $\delta(S)$ with $c(\delta(S)) = f_x(t)$. The value of an st-flow can never exceed the capacity of any st-cut. Thus, $\delta(S)$ is a minimum st-cut.

Theorem (Maximum Integral Flow)

Suppose all capacities are integer. There exists a maximum st-flow which is integer.

Proof. There exists an integer st-flow (just use 0 for all arcs). Let x be a maximum st-flow. Suppose there is a fractional st-flow with larger value. Then Augmenting Path theorem implies there is an augmenting path. Let x' be obtained by pushing flow along this path. Since c and x are integer, so is x'. Moreover, $f_{x'}(t) > f_x(t)$. But then x' contradicts our choice of x.

4.3 Ford-Fulkerson Algorithm

Algorithm 7 Ford-Fulkerson Algorithm

```
1: Input: Digraph $D = (N, A)$ with capacities $c$ and nodes $s, t$
2: Output: Maximum $st$-flow $x$ and minimum $st$-cut $\delta(S)$ with $f_x(t) = c(\delta(S))$
3: $x_{uv} = 0$ for all $uv \in A$
4: while Residual digraph $D'$ has an augmenting path ($st$-dipath) do
5:     Let $P'$ be the augmenting path/$st$-dipath in $D'$ (Edmonds-Karp: choose shortest $st$-dipath)
6:     Let $P$ be the corresponding incrementing path in $D$
7:     Push $\gamma(P)$ flow along $P$
8: Let $S$ be the set of nodes reachable from $s$ in $D'$
9: Flow $x$ is maximum and $st$-cut $\delta(S)$ is minimum
```

4.3.1 Efficiency and Convergence

Theorem (Edmonds & Karp)

If each flow augmentation is made along an augmenting path containing the minimum possible number of arcs, then a maximal flow is obtained after at most $|N|\,|A|$ augmentations, i.e. choose the shortest augmenting path.

Definition: Level Graph

Given a digraph $D = (N, A)$ with source s, its level graph is defined by:

- $l(v)$ = number of edges in shortest sv-dipath.
- $L_D = (N, A_D)$ is the subgraph of D that contains only those arcs with $l(w) = l(v) + 1$.

Proposition

P is a shortest sv-dipath in D if and only if P is an sv-dipath in L_D.

Lemma

The length of a shortest augmenting path never decreases.

Proof. Let f, f' be the flow before and after a shortest path augmentation. Let L_G and $L_{G'}$ be level graphs of G_f and $G_{f'}$ (these are residual digraphs). We only add back arcs to G_f since any st-dipath that uses a back arc is longer than previous length.

Lemma

After at most m shortest path augmentations, the length of a shortest augmenting path strictly increases.

Lemma

Let x be a feasible st-flow and let x' be the flow obtained from x by pushing flow on a shortest augmenting path. Then for each $v \in N$, $d_x(s, v) \le d_{x'}(s, v)$.

4.4 Max-Flow With Lower Bounds

Generalizing maximum st-flow problem to impose a lower bound l_{uv} on each arc uv.

LP Formulation of Lower Bounded Maximum st-Flow

$$\begin{aligned} \max \quad & f_x(t) \\ \text{s.t.} \quad & f_x(u) = 0, \ \forall u \in N \setminus \{s, t\} \\ & l_{uv} \le x_{uv} \le c_{uv}, \ \forall uv \in A \end{aligned}$$

Definition: Residual Capacity

Let P be an st-path, then

$\gamma(P) = \min\{c_{uv} - x_{uv} : uv \text{ is forward arc of } P\} \cup \{x_{uv} - l_{uv} : uv \text{ is backward arc of } P\}$

Lemma

Let x be an st-flow and let $\delta(S)$ be an st-cut. Then

$$f_x(t) \leq c(\delta(S)) - l(\delta(\overline{S}))$$

Proof.

$$f_x(t) = x(\delta(S)) - x(\delta(\overline{S})) \leq c(\delta(S)) - l(\delta(\overline{S}))$$

where $x_{uv} \geq l_{uv}$ and $x_{uv} \leq c_{uv}$. It is only strict if and only if $x_{uv} > l_{uv}$ for some $uv \in \delta(\overline{S})$ or $x_{uv} < c_{uv}$ for some $uv \in \delta(S)$.

Lemma (Key Lemma)

Consider a digraph $D = (N, A)$ with an st-flow x and suppose D has no augmenting path. Then there exists an st-cut $\delta(S)$ such that $f_x(t) = c(\delta(S)) - l(\delta(\overline{S}))$.

Theorem (Generalized Max-Flow Min-Cut)

The maximum value of an st-flow with lower bounds is equal to the minimum over all st-cuts $\delta(S)$ of $c(\delta(S)) - l(\delta(\overline{S}))$.

4.5 Applications

4.5.1 Matrix Rounding

Matrix Rounding Problem

Given a $p \times q$ matrix of real numbers $D = \{d_{ij}\}$ with row sums α_i and column sums β_j. We can round any real number a to $\lfloor a \rfloor$ or $\lceil a \rceil$. Round the matrix so that the sum of each rounded row is equal to the rounded row sum and the sum of each rounded column is equal to the rounded column sum.

Definition: Consistent Rounding

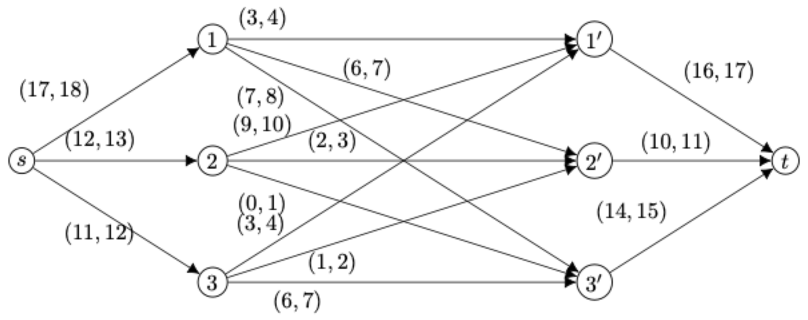
The solution to the the Matrix Rounding problem.

Example: Let the matrix be

	C1	C2	C3	Row sum
R1	3.1	6.8	7.3	17.2
R2	9.6	2.4	0.7	12.7
R3	3.6	1.2	6.5	11.3
Column sum	16.3	10.4	14.5	

Flow Network Construction

The digraph D contains a node i for each row i and a node j' for each column j, an arc ij' for each matrix element d_{ij}, an arc si for each row sum, and an arc $j't$ for each column sum. The lower and upper bounds for each arc ij' are $\lfloor d_{ij} \rfloor$ and $\lceil d_{ij} \rceil$, for each si are $\lfloor \alpha_{ij} \rfloor$ and $\lceil \alpha_{ij} \rceil$, and for each arc it are $\lfloor \beta_{ij} \rfloor$ and $\lceil \beta_{ij} \rceil$.



To solve the problem, we find a feasible integral st-flow to the digraph.

4.5.2 Optimum Closure

Let $D = (N, A)$ be a digraph with node-weights $w \in \mathbb{R}^N$.

Definition: Closure

A set $X \subseteq N$ where there is no arc $uv \in A$ where $u \in X$ and $v \in N - X$.

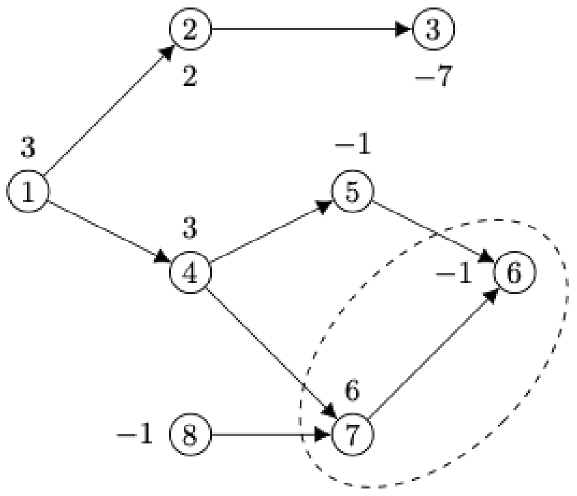
Maximum Closure Problem

Find a closure X maximizing $w(X) := \sum_{u \in X} w_u$.

Open Mining Problem

We associate a value w_v with region v which represents the difference between the value of material that can be extracted from v and the cost of digging out v. For every pair $u, v \in N$, we have an arc $uv \in A$ if region v needs to be dug out before u can be.

Example: The closure $\{6, 7\}$ has value $-1 + 6 = 5$.



Flow Network Construction

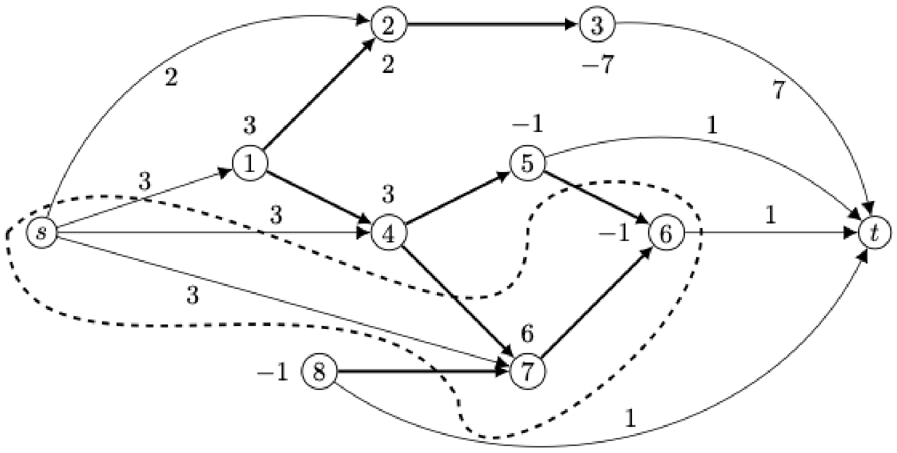
Define $N^+ := \{v \in N : w_v > 0\}$ and $N^- := \{v \in N : w_v \leq 0\}$. Define $\hat{D} = (\hat{N}, \hat{A})$ such that $\hat{N} = N \cup \{s, t\}$ and $uv \in \hat{A}$ if either $uv \in A$, $u = s$ and $v \in N^+$, or $u \in N^-$ and $v = t$. Define arc capacities $c \in \mathbb{R}^A$ such that

$$c_{uv} = \begin{cases} \infty & uv \in A \\ w_v & u = s \\ |w_u| & v = t \end{cases}$$

Remark

Let $\delta(\hat{S})$ be an st-cut of $\hat{D}$. Then $c(\delta(\hat{S}))$ is finite if and only if $\hat{S} - \{s\}$ is a closure of D.

Example: Digraph $\hat{D}$ with arc capacities c, bold arcs uv have capacity ∞. Let $S = \{s, 6, 7\}$, then $\delta(S)$ is an st-cut. Since $c(\delta(S))$ is finite, $S - \{s\} = \{6, 7\}$ is a closure.



Proposition

Let $\delta(\hat{S})$ be an st-cut in $\hat{D}$ and let $S := \hat{S} - \{s\}$. If S is a closure in D, then

$$c(\delta(\hat{S})) = w(N^+) - w(S)$$

Proof.

$$c(\delta(S)) = \sum_{i \in N^+ - S} c_{si} + \sum_{j \in N^- \cap S} c_{jt} = \sum_{i \in N^+ - S} w_i + \sum_{j \in N^- \cap S} -w_j$$

Since $\sum_{i \in N^+ \cap S} w_i = \sum_{i \in N^+ \cap S} -w_i$, we can add it to the RHS above and get

$$c(\delta(S)) = \sum_{i \in N^+ - S} w_i + \sum_{j \in N^- \cap S} -w_j + \sum_{i \in N^+ \cap S} w_i + \sum_{i \in N^+ \cap S} -w_i = w(N^+) - w(S)$$

Corollary

If $\delta(\hat{S})$ is a minimum st-cut in $\hat{D}$, then $\hat{S} - \{s\}$ is a maximum closure.

4.6 Combinatorial Implications

4.6.1 Flow Decomposition

Definition: Circulation

A vector $x \in \mathbb{R}^A$ such that $f_x(u) = 0$ for all $u \in N$ and $x_{uv} \geq 0$ for all $uv \in A$.

Lemma

Let C be a dicycle and let x^C be the characteristic vector of C. Then x^C is a circulation.

Lemma

If x is the sum of dicycles, then x is an integer circulation.

Theorem (Circulation Decomposition)

If x is an integer circulation, then x is a sum of dicycles.

Proof. We prove by induction on $x(A)$. If $x(A) = 0$, then x is the empty sum of dicycles. Now assume that the result holds for every circulation x' where $x'(A) < x(A)$. We may assume D has no arcs with zero flow (or isolated nodes), as we may delete them.

Let $u \in N$. Suppose for a contradiction that $x(\delta(\overline{u})) = 0$. Then since x is a circulation, $x(\delta(u)) = 0$. Since D has no zero flow arcs, u must be an isolated node, which is a contradiction.

Thus, $x(\delta(\overline{u})) \neq 0$ and that u has in-degree at least 1. Since this is true for every node, D has a dicycle C. Let x^C be the characteristic vector of C and let $x' := x - x^C$. It follows that $f_{x'}(u) = f_x(u) - f_{x^C}(u)$. Since x and x^C are circulations, then $f_x(u) = f_{x^C}(u) = 0$. Since D has no zero flow arcs and x is integer, then for every arc $uv \in A$, $x_{uv} \geq 1$. Thus, $x' \geq 1$ and x' is a circulation.

Since $x'(A) < x(A)$, it follows by induction that $x' = x^{C_1} + \cdots + x^{C_t}$ for dicycles $C_1, \ldots, C_t$. Hence, $x = x^C + x^{C_1} + \cdots + x^{C_t}$.

Theorem (st-Flow Decomposition)

If x is an integer st-flow, then x is the sum of st-dipaths and dicycles.

Proof. Let D' be obtained from D by adding an arc ts and x' be obtained by setting $x'_{uv} = x_{uv}$ for all $uv \in A(D)$ and $x'_{ts} = f_x(t)$. If $u \in N - \{s, t\}$, then $f_{x'}(u) = f_x(u) = 0$. By construction, $f_{x'}(t) = f_x(t) - x_{ts} = 0$. Similarly, $f_{x'}(s) = 0$. Thus, x' is a circulation.

By Circulation Decomposition theorem, we have that $x' = x^{C_1} + \cdots + x^{C_r}$ for dicycles $C_1, \ldots, C_r$. We may assume that there is a q where $1 \leq q \leq r$ such that $C_1, \ldots, C_q$ use arc ts and $C_{q+1}, \ldots, C_r$ do not use ts. Let $P_1, \ldots, P_q$ be the st-dipaths $C_1 - ts, \ldots, C_q - ts$. Then, x is the sum of $P_1, \ldots, P_q$ and $C_{q+1}, \ldots, C_r$.

4.6.2 Disjoint Dipaths

Definition: Arc-Disjoint

A set of st-dipaths are arc-disjoint if no two dipaths in the set share an arc.

Theorem

Let $D = (N, A)$ be a digraph where all $uv \in A$ have capacity $c_{uv} = 1$. Then there exists an st-flow of value k if and only if there exists k arc-disjoint st-dipaths.

Theorem (Menger – Arc Version)

The size of the largest set of arc-disjoint st-dipaths is equal to the size of the minimum set of arcs which disconnects t from s.

Definition: Node-st-Cut

A set $N' \subseteq N - \{s, t\}$ such that there are no st-dipaths in the digraph $D' = (N - N', A')$ where $A' := \{uv \in A : u \notin N', v \notin N'\}$.

Definition: Internally Disjoint

A set of st-dipaths are internally disjoint if no two dipaths share a node distinct from s or t.

Theorem (Menger – Node Version)

If st is not an arc, then the size of the largest set of internally disjoint st-dipaths is equal to the size of the minimum set of nodes which disconnects t from s.

4.6.3 Bipartite Matching and Vertex Cover

Definition: Bipartite Graph

An undirected graph $G = (V, E)$ is bipartite if we can partition $V = V_1 \cup V_2$ so that each edge has one endpoint in V_1 and one endpoint in V_2.

Definition: Matching

A set $M \subseteq E$ such that each $v \in V$ is incident with at most one edge of M.

Flow Network Construction

Construct a digraph $D = (N, A)$ where $N := V \cup \{s, t\}$ and $A := \{uv : u \in V_1, v \in V_2, uv \in E\} \cup \{su : u \in V_1\} \cup \{vt : v \in V_2\}$. Define capacities $c_{uv} = 1$ if $u = s$ or $v = t$ and $c_{uv} = \infty$ otherwise for all $uv \in A$.

Lemma

Let G be a bipartite graph with associated digraph D and capacities c. Let x be an integer st-flow in D. Then $M := \{uv \in E : u \in V_1, v \in V_2, x_{uv} = 1\}$ is a matching. Moreover, $|M| = f_x(t)$.

Definition: Neighbourhood

Let $S \subseteq V$, then $N(S)$ is the neighbourhood, which is the set of vertices adjacent to all vertices in S.

Theorem (Hall's Marriage Theorem)

Let $G = (V_1 \cup V_2, E)$ be a bipartite graph with $|V_1| = |V_2|$. Then, G has a perfect matching if and only if $|N(S)| \geq |S|$ for all $S \subseteq V_1$.

Proof. ($\implies$) Each node in S has to be matched to a different vertex in $N(S)$.

($\impliedby$) Suppose G does not have a perfect matching. We formulate this as a max-flow problem and let $\delta(S)$ be a minimum cut in G'. By the Max-Flow Min-Cut theorem, $c(\delta(S)) = \left|V_1 \cap \overline{S}\right| + |V_2 \cap S|$.

$$\left|V_1 \cap \overline{S}\right| + |V_2 \cap S| < |V_1 \cap S| + \left|V_2 \cap \overline{S}\right| \implies |V_2 \cap S| < |V_1 \cap S|$$

Definition: Vertex Cover

A set $W \subseteq V$ such that each edge of E is incident with at least one $v \in W$.

Lemma

Let G be a bipartite graph with associated digraph D and capacities c. Let $\delta(S)$ be a finite st-cut of D. Then $W := \{u \in V : su \in \delta(S) \text{ or } ut \in \delta(S)\}$ is a cover. Moreover, $c(\delta(S)) = |W|$.

Theorem (König's Theorem)

Let $G = (V, E)$ be a bipartite graph. The size of the largest matching of G is equal to the size of the smallest vertex cover of G.

Proof. Construct a flow network and compute the max-flow x^*. The size of the maximum matching is equal to $f_{x^*}(t)$. By Max-Flow Min-Cut, the max-flow is equal to the capacity of the min-cut $c(\delta(S)) = \left|V_1 \cap \overline{S}\right| + |V_2 \cap S|$. Every edge of the bipartite graph has an end node in $(V_1 \cap \overline{S}) \cup (V_2 \cap S)$.

4.7 Further Applications

4.7.1 Sports Team Elimination

Sports Team Elimination Problem

Several sports teams play a series of games. Each game is played by two teams (assuming no ties). A series champ is a team who wins the most games.

Definition: Eliminated

Given data from the middle of the series, a team is eliminated if no matter the outcome of the remaining games, it cannot be series champ.

Notation: Given data from the middle of series, our goal is to find out if a team, say A, is eliminated.

- T: set of teams other than A
- w_i: number of wins for team i
- r_{ij}: number of games remaining between teams i and j
- P: set of all unordered pairs in T

- M: number of wins for team A at the end of the season, assuming they win all remaining games, thus $M = w_A + \sum_{j \in T} r_{A,j}$

Consider $R \subseteq T$, the total number of wins for teams in R at the end of the season is at least

$$\sum_{i \in R} w_i + \sum_{\substack{\{i,j\} \in P \\ i,j \in R}} r_{ij}$$

since any game between two teams in R is won by one of them. If this number is $> M\,|R|$, then the average number of wins for teams in R is $> M$, so there must be a team in R who wins $> M$ games.

Theorem

Team A is eliminated if and only if there exists $R \subseteq T$ such that

$$\sum_{i \in R} w_i + \sum_{\substack{\{i,j\} \in P \\ i,j \in R}} r_{ij} > M\,|R|$$

Definition: Sum of Capacities

The sum of the capacities of arcs with head t, $\hat{C}$, where $\hat{C} = c(\delta(V - \{t\}))$.

Flow Network Construction

The flow network $G = (V, E)$ is built as follows:

- $V = T \cup P \cup \{s, t\}$.
- For each team $i \in T$, there is an arc si with capacity $c_{si} = M - w_i$.
- For each team pair $\{i, j\} \in P$, there is an arc $\{i, j\}t$ with capacity $c_{\{i,j\}t} = r_{ij}$.
- For each team i and pair $\{k, l\} \in P$, if $i = k$ or $i = l$, there is an arc $i\{k, l\}$ with $c_{i\{k,l\}} = \infty$.

Let $\delta(S^*)$ be the min st-cut of the flow network and $R = T - S^*$.

Proposition

If $c(\delta(S^*)) < \hat{C} = c(\delta(V - \{t\}))$, then at the end of the season, the total number of wins among teams in $R = T - S^*$ is $> M\,|R|$, i.e.

$$\sum_{i \in R} w_i + \sum_{\substack{\{i,j\} \in P \\ i,j \in R}} r_{ij} > M\,|R|$$

Proof. Let $\delta(S^*)$ be a minimum st-cut.

$$\begin{aligned}
c(\delta(S^*)) &= \sum_{v\notin S^*} c_{sv} + \sum_{v\in S^*} c_{vt} \\
&= \sum_{i\in R} c_{si} + \sum_{\{i,j\}\in S^*} c_{\{i,j\}t} \\
&= \sum_{i\in R}(M-w_i) + \sum_{\{i,j\}\in S^*} r_{ij} \\
&= M\,|R| - \sum_{i\in R} w_i + \sum_{\{i,j\}\in S^*} r_{ij}
\end{aligned}$$

Since $\hat{C} = c(\delta(V-\{t\})) = \sum_{\{i,j\}\in P} r_{ij}$ and that $c(\delta(S^*)) < \hat{C}$,

$$\begin{aligned}
M\,|R| &< \sum_{i\in R} w_i + \sum_{\{i,j\}\in P} r_{ij} - \sum_{\{i,j\}\in S^*} r_{ij} \\
&\leq \sum_{i\in R} w_i + \sum_{\substack{\{i,j\}\in P \\ i,j\in R}} r_{ij}
\end{aligned}$$

where the last inequality holds because if both i and j are in $V-S^*$, then we may assume that $\{i,j\}$ is not in S^*, otherwise by moving $\{i,j\}$ from S^* to $V-S^*$, we decrease the capacity of $\delta(S^*)$ by r_{ij} and $r_{ij}\geq 0$.

4.8 Preflow-Push/Push-Relabel Algorithm

Definition: st-Preflow

A vector $x\in\mathbb{R}^A$ such that $0\leq x_{uv}\leq c_{uv}$ for all $uv\in A$ and $x(\delta(\overline{u})) - x(\delta(u)) \geq 0$ for all $u\in N\setminus\{s\}$.

Definition: Excess

$$e_x(u) = x(\delta(\overline{u})) - x(\delta(u))$$

Definition: Residual Digraph

$D'_x = (N, A'_x)$.

Definition: Height

A vector h where $h(u)$ or h_u is the height of node u.

The Preflow-Push algorithm maintains a preflow and attempts to convert it into a feasible flow. It maintains a nonnegative height h_u for each $u\in N$ and pushes flow on an arc uv only if uv is a downward arc, i.e. if v's height is lower than u's height.

Definition: Source-Sink Conditions

$$h(s) = |N|, h(t) = 0$$

Definition: Steepness Conditions

$$h(v) \geq h(u) - 1, \ \forall uv \in A'_x$$

Definition: Compatible

A preflow x and height h are compatible if the Source-Sink conditions and Steepness conditions are satisfied.

Lemma

If $x \in \mathbb{R}^A$ is an st-preflow and $h \in \mathbb{R}^N$ are compatible height labels, then D'_x has no st-dipath.

Corollary

If $x \in \mathbb{R}^A$ is a feasible st-flow and $h \in \mathbb{R}^N$ are compatible height labels, then x is a maximum st-flow.

The main goal of the algorithm is to convert a preflow x into a feasible st-flow. To do this, we ensure none of the nodes in $N \setminus \{s, t\}$ have positive excess. This is done by pushing existing excess downhill on available arcs in the residual digraph.

Assume that $e_x(u) > 0$ and uv be a downward arc in D'_x (compatibility implies that $h_v = h_u - 1$. The algorithm pushes excess from u to v.

Algorithm 8 Push(x, h, uv)

```
1: Require: $e_x(u) > 0, uv \in A'_x, h_v = h_u - 1$
2: if $uv$ is a forward arc then
3:     $x_{uv} = x_{uv} + \min\{e_x(u), c_{uv} - x_{uv}\}$
4: else
5:     $x_{vu} = x_{vu} - \min\{e_x(u), x_{vu}\}$
```

If no such arc exists, then $h_v \geq h_u$ for all arcs $uv \in \delta_{D'_x}(u)$. In this case, we can increase the height of u by 1, without violating compatibility.

Algorithm 9 Relabel(x, h, v)

```
1: Require: $e_x(u) > 0$ and $h_v \geq h_u$ for all $uv \in A'_x$
2: $h_u = h_u + 1$
```

Assumption: $\delta(\overline{s}) = \emptyset$.

Example:

Algorithm 10 Preflow-Push/Push-Relabel Algorithm

```
1:  $h_s = |N|$
2:  $h_v = 0$ for all $v \in N \setminus \{s\}$
3:  $x_a = \begin{cases} c_a & \forall a \in \delta(s) \\ 0 & \text{otherwise} \end{cases}$
4:
5:  while $\exists u \in N \setminus \{t\}$ with $e_x(u) > 0$ do
6:      if $\exists uv \in D'_x$ with $h_v = h_u - 1$ then
7:          Push$(x, h, uv)$
8:      else
9:          Relabel$(x, h, u)$
10: return $x$
```

4.8.1 Analysis of Preflow-Push/Push-Relabel Algorithm

Lemma

If $x \in \mathbb{R}^A$ is an st-preflow and $h \in \mathbb{R}^N$ are compatible height labels, then D'_x has no st-dipath.

Proof. Suppose D'_x has an st-dipath $P = v_1v_2, \ldots, v_{k-1}v_k$ with $s = v_1, t = v_k$. P has $\leq |N| - 1$ arcs implying $k \leq |N|$. By compatibility, $h(v_1) = |N|$ and $h(v_k) = 0$. By repeatedly applying Steepness conditions,

$$h_{v_k} \geq h_{v_{k-1}-1} - 1 \geq \cdots \geq h_{v_1} - (k-1) = |N| - k + 1 \geq 1$$

This is a contradiction since $h_{v_k} = 0$.

Corollary

If $x \in \mathbb{R}^A$ is a feasible st-flow and $h \in \mathbb{R}^N$ are compatible height labels, then x is a maximum st-flow.

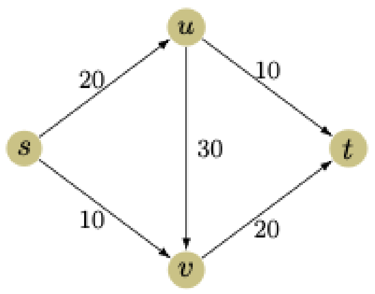
Lemma

Throughout the Preflow-Push algorithm, we have

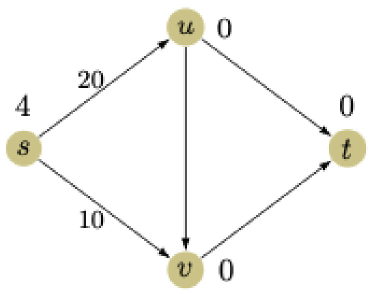
- for all $v \in N$, the heights h_v are nonnegative integers.
- x is a preflow and if the capacities are integral, then x is integral.
- preflow x and heights h are compatible.

Lemma

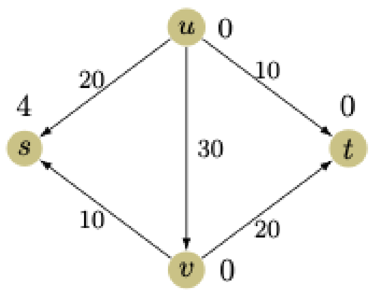
Let x be the current preflow at some point during the execution of the algorithm. If some node $v \in N$ has $e_x(v) > 0$, then the residual digraph D'_x contains a vs-dipath.



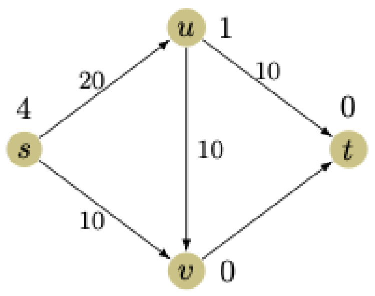
(0)



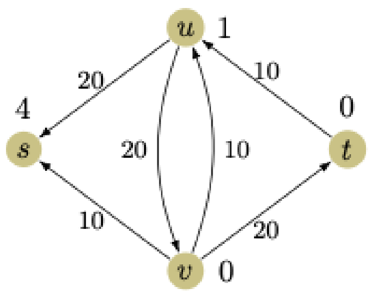
(1a)



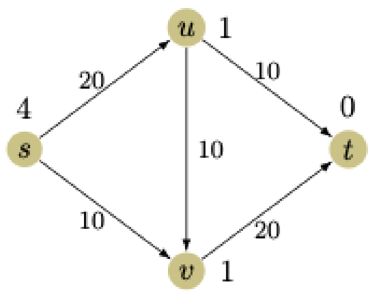
(1b)



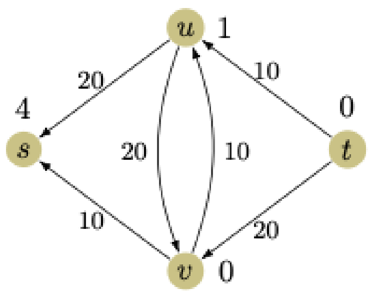
(2a)



(2b)



(3a)



(3b)

Proof. Let $S = \{v \in N : D'_x \text{ has a } vs\text{-dipath}\}$.

Claim: $x_{ij} = 0$ for all arcs $ij \in \delta(S)$. Otherwise, suppose an arc $ij \in \delta(S)$ has $x_{ij} > 0$. Then, arc ji is in D'_x. Then D'_x has a js-dipath, since $i \in S$, so j would be in S. But, $j \notin S$.

By the preflow property,

$$0 \leq \sum_{v \in \overline{S}} e_x(v) = \sum_{v \in \overline{S}} (x(\delta(\overline{v})) - x(\delta(v))) = x(\delta(S)) - x(\delta(\overline{S})) \leq 0$$

since $x(\delta(S)) = 0$.

Corollary

For all nodes $v \in N$, the height of v never exceeds $2\,|N| - 1$ throughout the execution of Preflow-Push.

Proof. Assume for the sake of contradiction that there is a node v whose height is at least $2|N|$ at some point during execution. Consider when v is relabelled and its height reaches $2|N|$. Let x be the preflow at this time. Note that the algorithm relabels nodes only if they have excess, thus by lemma, D'_x has a vs-dipath. This dipath has at most $|N|-1$ arcs, and using compatibility, we conclude that

$$h(s) \geq 2|N| - (|N|-1) = |N|+1$$

which contradicts the Source-Sink conditions.

Corollary

The number of relabel operations in the execution of Preflow-Push is at most $2|N|^2$.

Definition: Saturating Push

A push on an arc $uv \in A'_x$ such that the arc uv is not contained in the residual digraph after the push, i.e. a push such that either uv is a forward arc in D'_x and $c_{uv} - x_{uv}$ flow is pushed or uv is a backward arc in D'_x and x_{vu} flow is pushed.

Definition: Non-Saturating Push

A push operation that is not a saturating push.

Lemma

There are at most $2|N||A|$ saturating pushes throughout the execution of Preflow-Push.

Proof. Consider any arc uv of D'_x. After a saturating push, uv is absent from D'_x. Note that $h_u = h_v + 1$.

Arc uv must be present in D'_x before the next saturating push, if any, can occur. Hence a push from v to u must occur. Then, the height of v increases by ≥ 2. Over the whole execution, the height of v increases by two at most $|N|-1$ times. Hence, the number of saturating push operations is at most $|N|$.

Lemma

The total number of non-saturating pushes is bounded by $4|N|^2|A|$.

Theorem

The Preflow-Push algorithm terminates after $O(|N|^2|A|)$ push and relabel operations.

Theorem

If at each step of Preflow-Push, we choose an excess node of maximum height, then the number of non-saturating pushes is bounded by $4|N|^3$.
The total number of push and relabel operations of the algorithm is $O(|N|^3)$.

Chapter 5

Global Minimum Cuts

Global Minimum Cut Problem

Given a connected, undirected graph $G = (V, E)$, find a cut (A, B) of minimum capacity.

5.1 Karger's Randomized Algorithm

Definition: Edge Contraction

A contraction on edge $e = uv$ is removing e and merging u, v into one vertex x_{uv} and the edges incident to u or v are now incident to x_{uv}.

Algorithm 11 Karger's Algorithm

```
1: while $|V| > 2$ do
2:     Pick an edge $uv$ with probability $c_{uv} / \sum_{e \in E} c_e$
3:     Contract $uv$
4: Let $(S, V \setminus S)$ be the vertex sets corresponding to the 2 vertices remaining
5: return $\delta(S)$
```

Theorem

Let $\delta(S)$ be a specific minimum cut. The probability that the algorithm produces $\delta(S)$ is at least $\frac{1}{\binom{|V|}{2}} = \frac{2}{n(n-1)}$.

Proof. Let E_j be the event that an edge in the minimum cut is not contracted in the jth iteration.

$$\begin{aligned} P[E_1 \cap \cdots \cap E_{n-2}] &= P[E_1] \cdot P[E_2|E_1] \cdots P[E_{n-2}|E_1 \cap \cdots \cap E_{n-3}] \\ &\geq (1 - 2/n)(1 - 2/(n-1)) \cdots (1 - 2/3) \\ &= \frac{2}{n(n-1)} \end{aligned}$$

Chapter 6

Minimum Cost Flow

Minimum Cost Flow Problem (MCFP)

Consider a digraph $D = (N, A)$ with minimum flow capacity ℓ_{uv} and maximum flow capacity c_{uv} for all $uv \in A$, where each node u is either a demand node ($b_u > 0$), supply node ($b_u < 0$), or transshipment node ($b_u = 0$). For each arc uv, we have a cost w_{uv} which is the cost of a unit of flow.
Find a minimum cost flow that satisfies all node demands.

Minimum Cost Flow LP Formulation

$$\begin{array}{rl} \min & w^T x \\ \text{s.t.} & f_x(u) = b_u, \ \forall u \in N \\ & \ell_{uv} \leq x_{uv} \leq c_{uv}, \ \forall uv \in A \end{array}$$

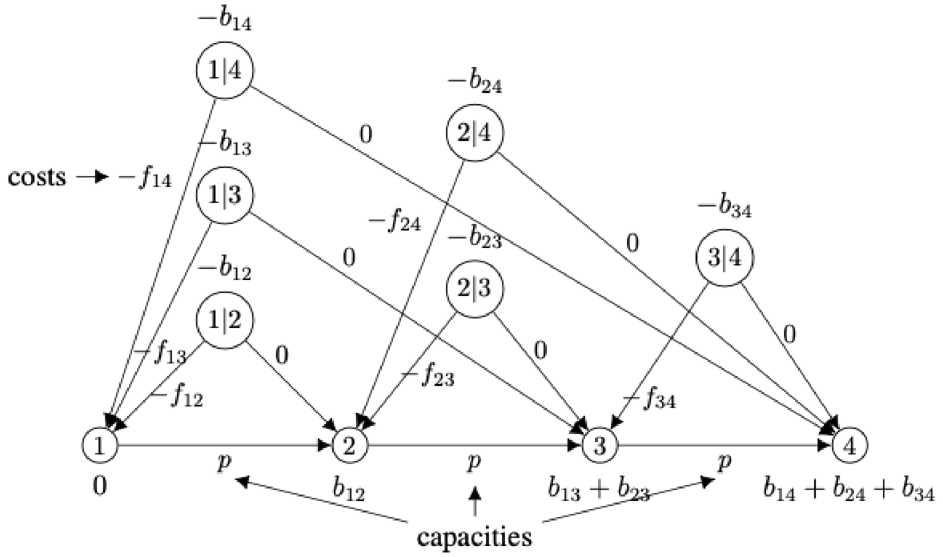
6.1 Applications

6.1.1 Airline Scheduling

Airline Scheduling Problem

A plane has a maximum seating capacity p. The plane will visit the cities $1, 2, \ldots, n$ in fixed order. The plane can pick up passengers at any city i and drop them off at any city j where $j > i$.
b_{ij} is the number of passengers at city i wishing to go to city j. f_{ij} is the fare from city i to j.
Determine the number of passengers to carry between various origins and destinations to maximize the total fare per trip while never exceeding the plane capacity.

Example: Let $n = 4$ cities. Introduce nodes $1, 2, 3, 4$ for each city and arcs $12, 23, 34$ for each flight leg. The flow on arc $(i, i+1)$ for $i = 1, 2, 3$ represents the number of passengers travelling on that flight leg. The cost of these arcs is zero and the capacity is p.



We have a node $[i|j]$ for every pair i, j with $i < j$. $b_{[i|j]} = -b_{ij}$. Node j for $j = 1, 2, 3, 4$ has demand $\sum_{k=1}^{j-1} b_{kj}$ which represents the total number of passengers which can potentially arrive to city j from a city preceding j. Thus, $b_1 = 0, b_2 = b_{12}, b_3 = b_{13} + b_{23}, b_4 = b_{14} + b_{24} + b_{34}$.

We have infinite capacity from node $[i|j]$ to nodes i and j and $\text{cost}([i|j], i) = -f_{ij}$ and $\text{cost}([i|j], j) = 0$.

Each unit of flow on arc $([i|j], i)$ corresponds to a passenger flying from i to j. It contributes $-f_{ij}$ to the objective function and contributes one unit of flow for arcs corresponding to the flight legs from i to j.

Each unit of flow on arc $([i|j], j)$ corresponds to a passenger who is not boarding the plane, and contributes zero to the objective function. It does not contribute to the flow on any other arc.

6.2 Lower Bounds

Let $D = (N, A)$ be a digraph, $\ell, c \in \mathbb{R}^A$, and $b \in \mathbb{R}^N$. Suppose x satisfies: $f_x(u) = b_u$ for all $u \in N$ and $\ell_{uv} \le x_{uv} \le c_{uv}$ for all $uv \in A$.

Let $x' = x - \ell$ and consider any $u \in N$. Then, $f_{x'}(u) = f_{x-\ell}(u) = f_x(u) - f_\ell(u)$. For all $u \in N$, define $b'_u = b_u - f_\ell(u)$ and for all $uv \in A$, define $c'_{uv} = c_{uv} - \ell_{uv}$.

Proposition

If x and $x' = x - \ell$ is a flow, then

$$f_x(u) = b_u, \forall u \in N, \ell_{uv} \leq x_{uv} \leq c_{uv}, \forall uv \in A$$

if and only if

$$f_{x'}(u) = b'_u, \forall u \in N, 0 \leq x'_{uv} \leq c'_{uv}, \forall uv \in A$$

Moreover, $w^T x' = w^T(x - \ell) = w^T x - w^T \ell$.

Thus, $w^T x$ and $w^T x'$ differ by a constant. Hence, we will assume that $\ell_{uv} = 0$ for the MCFP.

6.3 Flow Feasibility

Flow Network Construction

Define $N^+ := \{u \in N : b_u \geq 0\}$, $N^- := \{u \in N : b_u < 0\}$, and $\hat{D} = (\hat{N}, \hat{A})$ where $\hat{N} = N \cup \{s, t\}$ and $uv \in \hat{A}$ if either $uv \in A$, $u = s$ and $v \in N^-$, or $u \in N^+$ and $v = t$. For each $uv \in \hat{A}$, define

$$\hat{c}_{uv} = \begin{cases} c_{uv} & \text{if } uv \in A \\ -b_v & \text{if } u = s, v \in N^- \\ b_u & \text{if } u \in N^+, v = t \end{cases}$$

Theorem

There exists a flow if and only if $b(N) = 0$ and for all $S \subseteq N$, $b(S) \leq c(\delta(\overline{S}))$.

Proof. ($\implies$) Flow x^* exists, so $Mx^* = b$ holds. Sum up these $|N|$ equations to get $0 = b(N)$. Consider any set of nodes S and sum of equations $row_v(M) \cdot x = b_v$ for all nodes $v \in S$.

$$\begin{aligned} \sum_{v \in S} f_x(v) &= \sum_{v \in S} b_v \\ x(\delta(\overline{S})) - x(\delta(S)) &= b(S) \\ c(\delta(\overline{S})) - 0 &\geq b(S) \end{aligned}$$

So, $b(S) \leq c(\delta(\overline{S}))$.

($\impliedby$) Assume $b(N) = 0$ and for all $S \subseteq N$, $b(S) \leq c(\delta(\overline{S}))$. Construct the st-flow network from above. Let $\beta = c(\delta(\overline{t})) = c(\delta(s))$.

Lemma: x^* is a flow for the minimum cost flow network if and only if $\hat{x}$ is an st-flow of value β.

Fix

$$\hat{x}_{uv} = \begin{cases} x^*_{uv} & \text{if } uv \in A \\ -[x(\delta(\overline{v})) - x(\delta(v))] & \text{if } uv = sv \\ x(\delta(\overline{u})) - x(\delta(u)) & \text{if } uv = ut \end{cases}$$

Now we need to show every st-cut $\delta(S)$ of the st-flow network has capacity $\geq \beta$.

$$\begin{aligned} c(\delta(S)) &= \sum_{v \notin S} c_{sv} + \sum_{u \in S} c_{ut} + \sum_{u \in S, v \notin S} c_{uv} \\ &= \sum_{\substack{v \notin S \\ v \in N^-}} (-b_v) + \sum_{\substack{u \in S \\ u \in N^+}} b_u + c(\delta(S \setminus \{s\})) \\ &= \sum_{\substack{v \notin S \\ v \in N^-}} (-b_v) + \beta - \sum_{\substack{u \in S \\ u \in N^+}} b_u + c(\delta(S \setminus \{s\})) \\ &= \beta + c(\delta(S \setminus \{s\})) - \sum_{x \notin S} b_x \\ &= c(\delta(S \setminus \{s\})) - b(N - (S \setminus \{s\})) + \beta \end{aligned}$$

Since by the assumption $b(S) \leq c(\delta(\overline{S}))$, then $c(\delta(S \setminus \{s\})) - b(N - (S \setminus \{s\})) + \beta \geq \beta$. By max-flow min-cut, there is an st-flow of value β, so then by the lemma, we have x^* is a flow for the minimum cost flow problem.

Lemma

A flow exists for (D, b, c) if and only if $b(N) = 0$ and there exists an st-flow $\hat{x}$ in $\hat{D}$ with capacities $\hat{c}$ of value $b(N^+)$.
Moreover, we can use the max-flow min-cut algorithm to check if a flow exists or to prove that none does.

Proof. Used in previous theorem.

Corollary

If (D, b, c) has a flow and b, c integer, then (D, b, c) has an integer flow.

6.4 Incrementing Cycles

Definition: Orientation

An orientation of an undirected cycle C is a order in which we visit the nodes of C such that two nodes visited consecutively correspond to the endpoints of an arc.

Define C^+ to be the set of forward arcs of C and C^- to be the set of backward arcs of C.

Definition: Oriented Cycle

A cycle with an orientation.

Definition: Residual Capacity

Let C be a cycle, then

$$\gamma(C) = \min\{c_{uv} - x_{uv} : uv \text{ forward arc of } C\} \cup \{x_{uv} : uv \text{ backward arc of } C\}$$

Definition: Incrementing Cycle

C is an oriented cycle with $\gamma(C) > 0$.

Definition: Pushing Flow Along C

$$x'_{uv} = \begin{cases} x_{uv} + \gamma(C) & uv \text{ forward arc of } C \\ x_{uv} - \gamma(C) & uv \text{ backward arc of } C \\ x_{uv} & \text{otherwise} \end{cases}$$

Lemma

Let (D, b, c, w) define a minimum cost flow problem and let x be a flow. Let C be an oriented cycle. Let x' be obtained by pushing flow along C. Then x' is a flow and $w^T x' = w^T x + \gamma(C)[w(C^+) - w(C^-)]$.
In particular, if $w(C^+) - w(C^-) < 0$ and C is an incrementing cycle, then x is not maximum.

Definition: Residual Digraph

Let x be a flow. The residual digraph $D' = (N, A', w')$ where $uv \in A'$ with weight w'_{uv} if either

- $uv \in A$ and $x_{uv} < c_{uv}$ with weight w_{uv}, *or*
- $vu \in A$ and $x_{vu} > 0$ with weight $-w_{uv}$.

Lemma

Let (D, b, c, w) define a minimum cost flow problem and let x be a flow. Let D' be the residual digraph with costs w'. If D' has a negative dicycle, then x is not a minimum cost flow.

6.5 Optimality

6.5.1 Reduced Costs Optimality

Lemma

Let (D, b, c, w) define a minimum cost flow problem and let x be a flow. Let $\overline{w}$ be the reduced costs, then $\overline{w}^T x = w^T x - \sum_{u \in N} b_u y_u$.

Proof. (sketch) Recall $\overline{w}^T = w^T - y^T M$. Hence,

$$\overline{w}^T x = (w^T - y^T M)x = w^T x - y^T M x = w^T x - y^T b$$

Corollary

x is a minimum flow with costs w if and only if x is a minimum flow with reduced costs $\overline{w}$.

Definition: Optimality Conditions

Let (D, b, c, w) define a minimum cost flow problem and let x be a flow. For each $uv \in A$,

- if $\overline{w}_{uv} > 0$, then $x_{uv} = 0$.
- if $\overline{w}_{uv} < 0$, then $x_{uv} = c_{uv}$.

Economic interpretation: avoid arcs with positive reduced cost, and saturate arcs with negative reduced cost.

Lemma

If x satisfies the optimality conditions, then x is a minimum cost flow.

Lemma (Key Lemma)

Let (D, b, c, w) define a minimum cost flow problem and let x be a flow. Let y be any feasible potentials for the residual digraph $D'_x = (N, A')$ with costs w'. Then, the reduced costs $\overline{w}$ of D satisfy the optimality conditions.

Proof. Recall feasible potentials: $y \in \mathbb{R}^N$ such that $w'_{uv} + y_u - y_v \geq 0$ for all $uv \in A'$.

Case 1: $uv \in A'$ with $x_{uv} < c_{uv}$.
Then, uv is present in D'_x and $w'_{uv} + y_u - y_v \geq 0$ or equivalently in the original digraph, $w_{uv} + y_u - y_v \geq 0$. Thus, optimality condition 2 holds.

Case 2: $vu \in A'$ with $x_{uv} > 0$ and $w'_{vu} = -w_{uv}$.
Since y are feasible potentials for D'_x, w', $w'_{vu} + y_v - y_u \geq 0$ or equivalently in the original digraph, $-w_{uv} - y_i + y_j \geq 0 \implies \overline{w}_{uv} \leq 0$. Thus, optimality condition 1 holds.

Theorem (Min-Cost Flow)

The following are equivalent for a flow x of (D, b, c, w):

- x is a minimum cost flow.
- There is no negative dicycle in the residual digraph D' with costs w'.
- There exist potentials y such that the optimality conditions hold.

Theorem (Minimum Cost Integral Flow)

If b and c are integral and there exists a minimum cost flow x, then there exists an integer minimum cost flow.

6.5.2 Linear Programming Optimality

Recall the minimum cost flow problem LP

$$\begin{aligned} \min \quad & w^T x \\ \text{s.t.} \quad & f_x(u) = b_u, \ \forall u \in N \\ & -x_{uv} \geq -c_{uv}, \ \forall uv \in A \\ & x \geq 0 \end{aligned}$$

Definition: Dual LP of MCFP

$$\begin{aligned} \max \quad & \sum_{u \in N} b_u y_u - \sum_{uv \in A} c_{uv} z_{uv} \\ \text{s.t.} \quad & -y_u + y_v - z_{uv} \leq w_{uv} \iff z_{uv} \geq -\overline{w}_{uv}, \ \forall uv \in A \\ & z \geq 0 \end{aligned}$$

We can assume $z_{uv} := \max\{0, -\overline{w}_{uv}\}$ for all $uv \in A$.

Theorem (Complementary Slackness Conditions)

1. If $z_{uv} = \max\{0, -\overline{w}_{uv}\} > 0$, then $x_{uv} = c_{uv}$.
2. If $x_{uv} > 0$, then $\overline{w}_{uv} = -\max\{0, -\overline{w}_{uv}\} = z_{uv}$.

Proof. $\max\{0, -\overline{w}_{uv}\} > 0$ if and only if $\overline{w}_{uv} < 0$. This corresponds to the second condition from the previous section.

$\overline{w}_{uv} = -\max\{0, -\overline{w}_{uv}\}$ if and only if $\overline{w}_{uv} \leq 0$. Thus, if $x_{uv} > 0$, then $\overline{w}_{uv} \leq 0$ or equivalently, if $\overline{w}_{uv} > 0$, then $x_{uv} = 0$, which corresponds to the first condition from the previous section.

Computing flow x from potentials y:

1. Delete any arc uv with $\overline{w}_{uv} > 0$.
2. Delete any arc uv with $\overline{w}_{uv} < 0$ and push c_{uv} flow to node v from node u.
3. Find any feasible flow in the resulting digraph.

6.5.3 Incrementing Cycle Algorithm

Algorithm 12 Incrementing Cycle Algorithm for MCFP

```
1: Input: digraph $D = (N, A)$ with $c, w \in \mathbb{R}^A$ and $b \in \mathbb{R}^N$
2: Output: minimum cost flow $x$ (if one exists)
3: Find feasible flow $x$ by solving maximum $st$-flow problem or prove none exists
4: Find residual digraph $D'$ and costs $w'$
5: while $D'$ has negative dicycles do
6: Let $C'$ be negative dicycle
7:     Let $C$ be incrementing cycle corresponding to negative dicycle $C'$ in $D'$
8:     Push flow along $C$ in direction of $C'$
9: return $x$ is minimum flow
```

This algorithm is not efficient.

6.6 Network Simplex Algorithm

Definition: Tree Flow

x is a tree flow if there exists a spanning tree T of D such that for all $uv \notin A(T)$, either $x_{uv} = 0$ or $x_{uv} = c_{uv}$.

Theorem

If there exists a minimum cost flow for (D, b, c, w), then there exists a minimum cost flow which is a tree flow.

Lemma

Let G be a connected graph and H be a subgraph of G which has no cycles. Then there exists a spanning tree T of G which contains all edges of H.

Algorithm 13 Network Simplex Algorithm (MCFP)

```
1: Input: digraph $D = (N, A)$ with $c, w \in \mathbb{R}^A$ and $b \in \mathbb{R}^N$
2: Output: minimum cost flow $x$ (if one exists)
3: Find initial tree flow $x$ with spanning tree $T$
4: Find potentials $y$ such that $\overline{w}_{uv} = 0$ for all $uv \in T$
5: while all non-tree arcs do not satisfy optimality conditions do
6:     There exists a non-tree arc $uv$ where either
         (1) $\overline{w}_{uv} < 0$ and $x_{uv} = 0$, or
         (2) $\overline{w}_{uv} > 0$ and $x_{uv} = c_{uv}$
7:     $C \in T + uv$
8:     if (1) occurs then
9:         Orient $C$ in direction of $uv$
10:    else
11:        Orient $C$ in opposite direction of $uv$
12:    Push $\gamma$ flow along $C$
13:    $hz \in C$ such that $x_{hz} = 0$ or $x_{hz} = c_{hz}$
14:    $T = T + uv - hz$
15:    Recompute potentials $y$
16: return $x$
```