Today’s Topics
Introduction
A Maximal Flow Algorithm
The Max Flow, Min Cut Theorem
Matching

NETWORK MODELS
Introduction

• Definition
  – A **transport network** (or more simply **network**) is a simple, weighted, directed graph satisfying:
    (a) A designated vertex, the **source**, has no incoming edges.
    (b) A designated vertex, the **sink**, has no outgoing edges.
    (c) The weight \( C_{ij} \) of the directed edge \((i, j)\), called the **capacity** of \((i, j)\), is a nonnegative number.

• Example
  – the graph of Figure 10.1
Introduction

• Definition
  – Let $G$ be a transport network. Let $C_{ij}$ denote the capacity of the directed edge $(i, j)$. A flow $F$ in $G$ assigns each directed edge $(i, j)$ a nonnegative number $F_{ij}$ such that:
    (a) $F_{ij} \leq C_{ij}$.
    (b) For each vertex $j$, which is neither the source nor the sink,
      $$\sum_i F_{ij} = \sum_i F_{ji} \quad (*) \text{property of the conservation of flow}$$
      – In a sum such as $(*)$, unless specified otherwise, the sum is assumed to be taken over all vertices $i$. Also, if $(i, j)$ is not an edge, we set $F_{ij} = 0$.
  – We call $F_{ij}$ the flow in edge $(i, j)$. For any vertex $j$, we call $\sum_i F_{ij}$ the flow into $j$ and we call $\sum_i F_{ji}$ the flow out of $j$. 
Introduction

• Example
  – Sample flow
    • $F_{ab} = 2$, $F_{bc} = 2$, $F_{cz} = 3$, $F_{ad} = 3$, $F_{dc} = 1$, $F_{de} = 2$, $F_{ez} = 2$
Introduction

• Theorem
  – Given a flow $F$ in a network, the flow out of the source $a$ equals the flow into the sink $z$, that is,
    \[ \sum_i F_{ai} = \sum_i F_{iz}. \]
  – Proof.
    • Let $V$ be the set of vertices.
    • We have
      \[ \sum_{j \in V} (\sum_{i \in V} F_{ij}) = \sum_{j \in V} (\sum_{i \in V} F_{ji}), \]
      since each double sum is $\sum_{e \in E} F_e$, where $E$ is the set of edges.
Introduction

Now, $0 = \sum_{j \in V} (\sum_{i \in V} F_{ij} - \sum_{i \in V} F_{ji})$

$= (\sum_{i \in V} F_{iz} - \sum_{i \in V} F_{zi}) +$

$(\sum_{i \in V} F_{ia} - \sum_{i \in V} F_{ai}) +$

$\sum_{j \in V, j \neq a,z} (\sum_{i \in V} F_{ij} - \sum_{i \in V} F_{ji})$

$= \sum_{i \in V} F_{iz} - \sum_{i \in V} F_{ai}$

since $F_{zi} = 0 = F_{ia}$ for all $i \in V$, and (by definition)

$\sum_{i \in V} F_{ij} - \sum_{i \in V} F_{ji} = 0$ if $j \in V - \{a,z\}$. 
Introduction

• Definition
  – Let $F$ be a flow in a network $G$. The value
    $\sum_i F_{ai} = \sum_i F_{iz}$
  is called the value of the flow $F$.

• Examples
  – the value of the flow in the network of Figure 10.1.2
  – A Pumping Network
    • Figure 10.1.3
      • supersource, supersink
  – A Traffic Flow Network
A Maximal Flow Algorithm

• Note
  – If $G$ is a transport network, a maximal flow in $G$ is a flow with maximal value.
  – Consider the edges of $G$ to be undirected and let
    $P = (v_0, v_1, ..., v_n)$, $v_0 = a$, $v_n = z$
    be a path from $a$ to $z$ in this undirected graph.
    • If an edge $e$ in $P$ is directed from $v_{i-1}$ to $v_i$, we say that
      $e$ is properly oriented (with respect to $P$); otherwise, we say that
      $e$ is improperly oriented (with respect to $P$).
A Maximal Flow Algorithm

• Example
  – the path from $a$ to $z$ in Figure 10.2.2
  – after increasing the flow by 1 (Figure 10.2.3)
  – the four possible orientations of the edges incident on $x$
    • Figure 10.2.4

• Example
  – the path from $a$ to $z$ in Figure 10.2.5
  – after increasing the flow by 1 (Figure 10.2.6)
A Maximal Flow Algorithm

• Theorem
  
  – Let \( P \) be a path from \( a \) to \( z \) in a network \( G \) satisfying the following conditions:
    
    (a) For each properly oriented edge \((i, j)\) in \( P \), \( F_{ij} < C_{ij} \).
    
    (b) For each improperly oriented edge \((i, j)\) in \( P \), \( 0 < F_{ij} \).
  
  – Let \( \Delta = \min X \), where \( X \) consists of the number \( C_{ij} - F_{ij} \) for properly oriented edges \((i, j)\) in \( P \), and \( F_{jj} \) for improperly oriented edges \((i, j)\) in \( P \).
A Maximal Flow Algorithm

– Define

\[ F^*_{ij} = F_{ij} \text{ if } (i, j) \text{ is not in } P, \]
\[ F_{ij} + \Delta \text{ if } (i, j) \text{ is properly oriented in } P, \]

and

\[ F_{ij} - \Delta \text{ if } (i, j) \text{ is not properly oriented in } P. \]

– Then \( F^* \) is a flow whose value is \( \Delta \) greater than the value of \( F \).
A Maximal Flow Algorithm

• Procedure
  – Start with a flow (e.g., the flow in which the flow in each edge is 0).
  – Search for a path satisfying the conditions of the earlier theorem.
    • If no such path exists, stop; the flow is maximal.
  – Increase the flow through the path by $\Delta$, where $\Delta$ is defined as in the earlier theorem, and go to line 2.
A Maximal Flow Algorithm

Algorithm 10.2.4: Finding a Maximal Flow in a Network

Input: A network with source $a$, sink $z$, capacity $C$, vertices $a = v_0, \ldots, v_n = z$, and $n$

Output: A maximal flow $F$

```plaintext
max_flow(a, z, C, v, n) {
    // v's label is (predecessor(v), val(v))
    // start with zero flow
    1. for each edge $(i, j)$
    2. $F_{ij} = 0$
    3. while (true) {
        // remove all labels
        4. for $i = 0$ to $n$ {
            5. predecessor($v_i$) = null
            6. val($v_i$) = null
        }
        // label $a$
        7. predecessor($a$) = —
        8. val($a$) = ∞
        // $U$ is the set of unexamined, labeled vertices
        9. $U = \{a\}$
    }
}
```
A Maximal Flow Algorithm

Algorithm 10.2.4 (continued)

// continue until z is labeled
11. while (val(z) == null) {
12.     if (U == ∅) // flow is maximal
13.         return F
14.     choose v in U
15.     U = U - {v}
16.     Δ = val(v)
17.     for each edge (v, w) with val(w) == null
18.         if (F_{vw} < C_{vw}) {
19.             predecessor(w) = v
20.             val(w) = min{Δ, C_{vw} - F_{vw}}
21.             U = U ∪ {w}
22.         }
23.     for each edge (w, v) with val(w) == null
24.         if (F_{wv} > 0) {
25.             predecessor(w) = v
26.             val(w) = min{Δ, F_{wv}}
27.             U = U ∪ {w}
28.         }
29. } // end while (val(z) == null) loop
A Maximal Flow Algorithm

Algorithm 10.2.4 (continued)

    // find path $P$ from $a$ to $z$ on which to revise flow
30. $w_0 = z$
31. $k = 0$
32. while ($w_k \neq a$) {
33.     $w_{k+1} = \text{predecessor}(w_k)$
34.     $k = k + 1$
35. }
36. $P = (w_{k+1}, w_k, \ldots, w_1, w_0)$
37. $\Delta = \text{val}(z)$
38. for $i = 1$ to $k + 1$ {
39.     $e = (w_i, w_{i-1})$
40.     if ($e$ is properly oriented in $P$)
41.         $F_e = F_e + \Delta$
42.     else
43.         $F_e = F_e - \Delta$
44. }
45. } // end while (true) loop
The Max Flow, Min Cut Theorem

- **Definition**
  - A cut \((P, \sim P)\) in \(G\) consists of a set \(P\) of vertices and the complement \(\sim P\) of \(P\), with \(a \in P\) and \(z \in \sim P\).

- **Examples**
  - the network of Figure 10.3.1
    - \(P = \{a, b, d\}\) and \(\sim P = \{c, e, f, z\}\)
  - the network of Figure 10.3.2
The Max Flow, Min Cut Theorem

• Definition
  – The capacity of the cut \((P, \sim P)\) is the number
    \[ C(P, \sim P) = \sum_{i \in P} \sum_{j \in \sim P} C_{ij}. \]

• Examples
  – the capacity of the cut of Figure 10.3.1
    • \(C_{bc} + C_{de} = 8\)
  – the capacity of the cut of Figure 10.3.2
    • \(C_{bc} + C_{dc} + C_{de} = 6\)
The Max Flow, Min Cut Theorem

• Theorem
  – Let $F$ be a flow in $G$ and let $(P, \sim P)$ be a cut in $G$. Then the capacity of $(P, \sim P)$ is greater than or equal to the value of $F$, that is,
    \[ \sum_{i \in P} \sum_{j \in \sim P} C_{ij} \geq \sum_i F_{ai}. \]
  – The notation $\sum_i$ means the sum over all vertices $i$.
  – Proof.
    • Exercise.
The Max Flow, Min Cut Theorem

• Example
  – the value of the flow of Figure 10.3.1
    • 5
  – the capacity of the cut of Figure 10.3.1
    • 8

• Note
  – A minimal cut is a cut having minimum capacity.
The Max Flow, Min Cut Theorem

• Theorem (Max Flow, Min Cut Theorem)
  – Let \( F \) be a flow in \( G \) and let \((P, \sim P)\) be a cut in \( G \). If equality holds in the previous theorem, then the flow is maximal and the cut is minimal. Moreover, equality holds in the the previous theorem if and only if
    (a) \( F_{ij} = C_{ij} \) for \( i \in P, j \in \sim P \) and
    (b) \( F_{ij} = 0 \) for \( i \in \sim P, j \in P \).
  – Proof.
    • Exercise.

• Example
  – the flow of Figure 10.3.2
The Max Flow, Min Cut Theorem

• Theorem
  – At termination, the algorithm of finding a maximal flow in a network produces a maximal flow. Moreover, if \( P \) (respectively, \( \sim P \)) is the set of labeled (respectively, unlabeled) vertices at the termination of the algorithm, the cut \((P, \sim P)\) is minimal.
Matching

• Example
  – Suppose that four persons $A$, $B$, $C$, and $D$ apply for five jobs $J_1$, $J_2$, $J_3$, $J_4$ and $J_5$.
  – Suppose that applicant $A$ is qualified for jobs $J_2$ and $J_5$; applicant $B$ is qualified for jobs $J_2$ and $J_5$; applicant $C$ is qualified for jobs $J_1$, $J_3$, $J_4$ and $J_5$; and applicant $D$ is qualified for jobs $J_2$ and $J_5$.
  – Suppose finally that each job takes only one person.
  – Is it possible to find a job for each applicant?
Matching

• Definition
  – Let $G$ be a directed, bipartite graph with disjoint vertex sets $V$ and $W$ in which the edges are directed from vertices in $V$ to vertices in $W$. (Any vertex in $G$ is either in $V$ or in $W$.)
  – A matching for $G$ is a set of edges $E$ with no vertices in common.
  – A maximal matching for $G$ is a matching $E$ in which $E$ contains the maximum number of edges.
  – A complete matching for $G$ is a matching $E$ having the property that if $v \in V$, then $(v,w) \in E$ for some $w \in W$. 
Matching

• Examples
  – the matching for the graph of Figure 10.4.2
  – A Matching Network (Figure 10.4.3)
Matching

• Theorem
  – Let $G$ be a directed, bipartite graph with disjoint vertex sets $V$ and $W$ in which the edges are directed from vertices in $V$ to vertices in $W$. (Any vertex in $G$ is either in $V$ or in $W$.)
    (a) A flow in the matching network gives a matching in $G$. The vertex $v \in V$ is matched with the vertex $w \in W$ if and only if the flow in edge $(v, w)$ is 1.
    (b) A maximal flow corresponds to a maximal matching.
    (c) A flow whose value is $|V|$ corresponds to a complete matching.
Matching

• Example
  – the matching of Figure 10.4.1 as a flow in Figure 10.4.3
Matching

- Note
  - If $S \subseteq V$ for a bipartite graph $G$ with vertex sets $V$ and $W$, we let
    \[
    R(S) = \{w \in W \mid v \in S \text{ and } (v,w) \text{ is an edge in } G\}.
    \]
  - Suppose that $G$ has a complete matching. If $S \subseteq V$, we must have
    \[
    |S| \leq |R(S)|.
    \]
Matching

• Theorem (Hall’s Marriage Theorem)
  – Let $G$ be a directed, bipartite graph with disjoint vertex sets $V$ and $W$ in which the edges are directed from vertices in $V$ to vertices in $W$. (Any vertex in $G$ is either in $V$ or in $W$.)
  – There exists a complete matching in $G$ if and only if $|S| \leq |R(S)|$ for all $S \subseteq V$
    where $R(S) = \{ w \in W | v \in S \text{ and } (v, w) \text{ is an edge in } G \}$.
  – Proof.
    • Exercise.
Matching

• Examples
  – the graph of Figure 10.4.1
  – computers and disk drives
Summary

- Introduction
- A Maximal Flow Algorithm
- The Max Flow, Min Cut Theorem
- Matching