Discrete Mathematics

CS204: Spring, 2008

Jong C. Park
Computer Science Department
KAIST

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,
      $\Sigma_i F_{ij} = \Sigma_i F_{ji}$ (* 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 $\Sigma_i F_{ij}$ the flow into j and we call $\Sigma_i F_{ji}$ 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,

$$\Sigma_i F_{ai} = \Sigma_i F_{iz}.$$

  - Proof.
    - Let V be the set of vertices.
    - We have

$$\Sigma_{j\in V}(\Sigma_{i\in V} F_{ij}) = \Sigma_{j\in V}(\Sigma_{i\in V} F_{ji}),$$

since each double sum is $\Sigma_{e\in E} F_e$, where E is the set of edges.

Introduction

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

$$= (\Sigma_{i \in V} F_{iz} - \Sigma_{i \in V} F_{zi}) + (\Sigma_{i \in V} F_{ia} - \Sigma_{i \in V} F_{ai}) + \Sigma_{j \in V, j \neq a,z}(\Sigma_{i \in V} F_{ij} - \Sigma_{i \in V} F_{ji})$$

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

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

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

Introduction

- Definition
  - Let F be a flow in a network G. The value

    $$\Sigma_i F_{ai} = \Sigma_i F_{iz}$$

    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

    $$P = (v_0, v_1, \ldots, v_n),\ v_0 = a,\ v_n = z$$

    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_{ij} for improperly oriented edges (i, j) in P.

A Maximal Flow Algorithm

– Define

$F^*_{ij} =$ F_{ij} if (i, j) is not in P,

$F_{ij} + \Delta$ if (i, j) is properly oriented in P, and

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

– Then F^* is a flow whose value is Δ 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 Δ, where Δ 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

```
max_flow(a, z, C, v, n) {
  // v's label is (predecessor(v), val(v))
  // start with zero flow
  for each edge (i, j)
    F_ij = 0
  while (true) {
    // remove all labels
    for i = 0 to n {
      predecessor(v_i) = null
      val(v_i) = null
    }
    // label a
    predecessor(a) = —
    val(a) = ∞
    // U is the set of unexamined, labeled vertices
    U = {a}
```

A Maximal Flow Algorithm

Algorithm 10.2.4 (continued)

```
// continue until z is labeled
while (val(z) == null) {
   if (U == ∅) // flow is maximal
      return F
   choose v in U
   U = U − {v}
   Δ = val(v)
   for each edge (v, w) with val(w) == null
      if (F_vw < C_vw) {
         predecessor(w) = v
         val(w) = min{Δ, C_vw − F_vw}
         U = U ∪ {w}
      }
   for each edge (w, v) with val(w) == null
      if (F_wv > 0) {
         predecessor(w) = v
         val(w) = min{Δ, F_wv}
         U = U ∪ {w}
      }
} // 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
        w_0 = z
        k = 0
        while (w_k ¬= a) {
            w_{k+1} = predecessor(w_k)
            k = k + 1
        }
        P = (w_{k+1}, w_k, ..., w_1, w_0)
        Δ = val(z)
        for i = 1 to k + 1 {
            e = (w_i, w_{i-1})
            if (e is properly oriented in P)
                F_e = F_e + Δ
            else
                F_e = F_e − Δ
        }
    } // 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

$$C(P, \sim P) = \Sigma_{i \in P} \Sigma_{j \in \sim P} C_{ij}.$$

- 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,

$$\Sigma_{i \in P} \Sigma_{j \in \sim P} C_{ij} \geq \Sigma_i F_{ai}.$$

  - The notation Σ_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 V or in W.)
  - A matching for G is a set of edges E <u>with no vertices in common</u>.
  - 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

    $$|S| \leq |R(S)| \text{ for all } S \subseteq V$$

    where

    $$R(S) = \{w \in W \mid v \in S \text{ and } (v, w) \text{ is an edge in } G\}.$$
  - 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