In the name of God

Network Flows

4. Maximum Flows Problems
4.1 Introduction

Fall 2010

Instructor: Dr. Masoud Yaghini

Introduction

Introduction

- The ***maximum flow problem***
  - In a capacitated network, we wish to send as much flow as possible between two special nodes, a source node *s* and a sink node *t*, without exceeding the capacity of any arc.

Introduction

- ***maximum flow problem*** vs. ***shortest path problem***
  - **Similarities:**
    - They are both pervasive in practice
    - They both arise as subproblems in algorithms for the ***minimum cost flow problem***.
  - ***Differences:***
    - Shortest path problems model arc costs but not arc capacities
    - Maximum flow problems model capacities but not costs.
  - Taken together, the shortest path problem and the maximum flow problem combine all the basic ingredients of network flows.

Introduction

- The algorithms for solving the maximum flow problem are two types:
  - ***Augmenting path algorithms***
    - They maintain mass balance constraints at every node of the network other than the source and sink nodes.
    - These algorithms incrementally augment flow along paths from the source node to the sink node.
  - ***Preflow-push algorithms***
    - They flood the network so that some nodes have excesses (or buildup of flow).
    - These algorithms incrementally relieve flow from nodes with excesses by sending flow from the node forward toward the sink node or backward toward the source node.

Notation and Assumptions

- $G = (N, A)$: a capacitated network with a nonnegative capacity u_{ij} associated with each arc $(i, j) \in A$.
- Let $U = \max\{u_{ij}:(i, j) \in \mathrm{A}\}$.
- the arc adjacency list $A(i) = \{(i, k): (i, k) \in \mathrm{A}\}$ contains all the arcs emanating from node i.
- There are two special nodes in the network G:
  - a source node s
  - a sink node t
- We wish to find the maximum flow from the source node s to the sink node t that satisfies the arc capacities and mass balance constraints at all nodes.

Notation and Assumptions

- We can state the problem formally as follows:

Maximize v

subject to

$$\sum_{\{j:(i,j)\in A\}} x_{ij} - \sum_{\{j:(j,i)\in A\}} x_{ji} = \begin{cases} v & \text{for } i = s, \\ 0 & \text{for all } i \in N - \{s \text{ and } t\} \\ -v & \text{for } i = t \end{cases}$$

$$0 \le x_{ij} \le u_{ij} \qquad \text{for each } (i, j) \in A.$$

  – a vector $x = \{x_{ij}\}$ that satisfying the constraints is ***flow***
  – the scalar variable v : the ***value*** of the flow .

Notation and Assumptions

- We consider the maximum flow problem subject to the following assumptions.
- ***Assumption 1. The network is directed.***
  - we can always fulfill this assumption by transforming any undirected network into a directed network.

Notation and Assumptions

- ***Assumption 2. All capacities are nonnegative integers.***
  - Although it is possible to relax the integrality assumption on arc capacities for some algorithms, this assumption is necessary for others.
  - In reality, the integrality assumption is not a restrictive assumption because all modern computers store capacities as rational numbers and we can always transform rational numbers to integer numbers by multiplying them by a suitably large number.

Notation and Assumptions

- ***Assumption 3. The network does not contain a directed path from node s to node t composed only of infinite capacity arcs.***
  - Whenever every arc on a directed path P from s to t has infinite capacity, we can send an infinite amount of flow along this path, and therefore the maximum flow value is unbounded.

Notation and Assumptions

- ***Assumption 4. Whenever an arc (i, j) belongs to A, arc (j, i) also belongs to A.***
  - This assumption is nonrestrictive because we allow arcs with zero capacity.

Notation and Assumptions

- ***Assumption 5. The network does not contain parallel arcs***
  - i.e., two or more arcs with the same tail and head nodes.
  - This assumption is essentially a notational convenience

Flows and Cuts

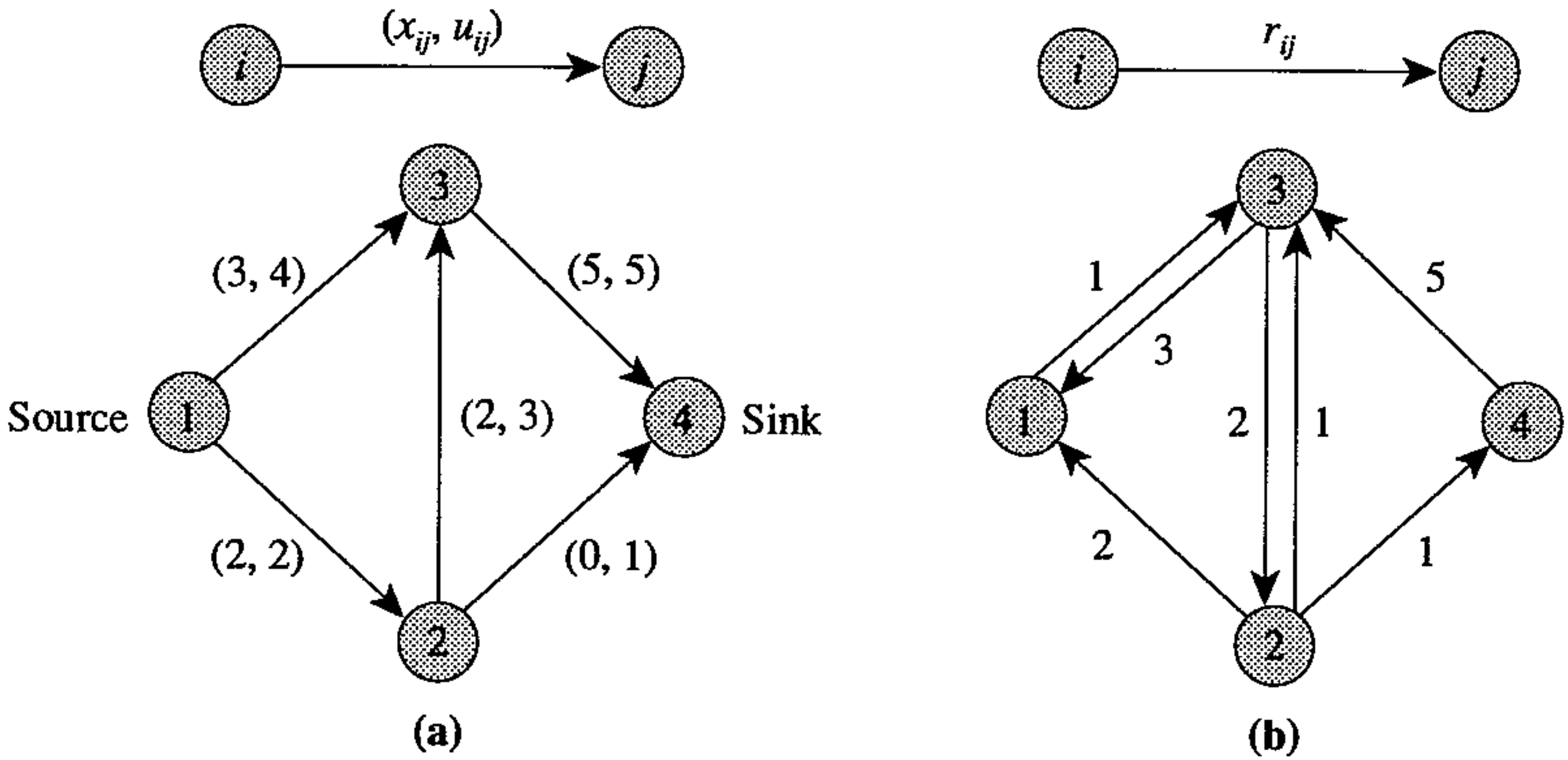
Flows and Cuts

- ***Residual network***
  - The concept of residual network plays a central role in the development of all the maximum flow algorithms we consider.
  - Given a flow x, the ***residual capacity*** r_{ij} of any arc $(i, j) \in A$ is the maximum additional flow that can be sent from node i to node j using the arcs (i, j) and (j, i).
  - The residual capacity r_{ij} has two components:
    - (1) $u_{ij} - x_{ij}$, the unused capacity of arc (i, j), and
    - (2) the current flow x_{ji} on arc (j, i), which we can cancel to increase the flow from node i to node j.

Flows and Cuts

- We refer to the network $G(x)$ consisting of the arcs with positive residual capacities as the **residual network** (with respect to the flow x).

Flows and Cuts

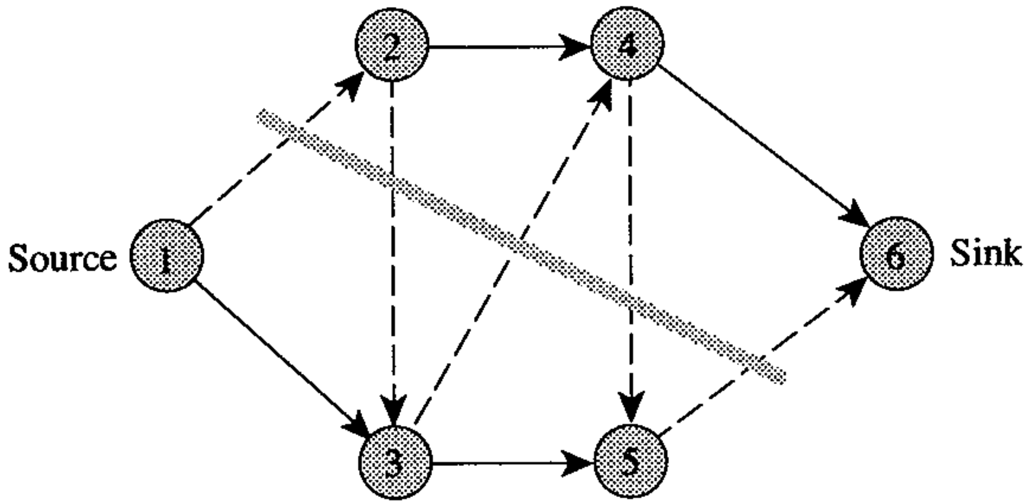


- (a) original network G with a flow x;
- (b) residual network $G(x)$, $\boldsymbol{r_{ij} = u_{ij} - x_{ij}, r_{ji} = x_{ij}}$

Flows and Cuts

- ***s-t cut***
  - A ***cut*** is a partition of the node set N into two subsets S and $\bar{S} = N - S$;
  - We represent this cut using the notation $[S, \bar{S}]$.
  - We can define a cut as the set of arcs whose endpoints belong to the different subsets S and $\bar{S}$.
  - We refer to a cut as an s-t cut if $s \in S$ and $t \in \bar{S}$.
  - ***Forward arc*** of the cut: an arc (i, j) with $i \in S$ and $j \in \bar{S}$
  - ***Backward arc*** of the cut: an arc (i, j) with $i \in \bar{S}$ and $j \in S$
  - $(S, \bar{S})$: denote the set of forward arcs in the cut
  - $(\bar{S}, S)$: denote the set of backward arcs in the cut

Flows and Cuts



- the dashed arcs constitute an *s*-*t* cut.
- $(S, \overline{S}) = \{(1, 2), (3,4), (5, 6)\}$
- $(\overline{S}, S) = \{(2, 3), (4, 5)\}$

Flows and Cuts

- ***Capacity of an s-t cut.***
  - We define the capacity $u[S, \bar{S}]$ of an s-t cut $[S, \bar{S}]$ as the sum of the capacities of the forward arcs in the cut. That is,

$$u[S, \bar{S}] = \sum_{(i,j) \in (S, \bar{S})} u_{ij}.$$

- ***Capacity of a cut***
  - is an upper bound on the maximum amount of flow we can send from the nodes in S to the nodes in $\bar{S}$ while honoring arc flow bounds.

Flows and Cuts

- ***Minimum cut.***
  - We refer to an s-t cut whose capacity is minimum among all s-t cuts as a minimum cut.

- ***Residual capacity of an s-t cut.***
  - We define the residual capacity $r[S, \bar{S}]$ of an s-t cut $[S, \bar{S}]$ as the sum of the residual capacities of forward arcs in the cut.
  - That is,

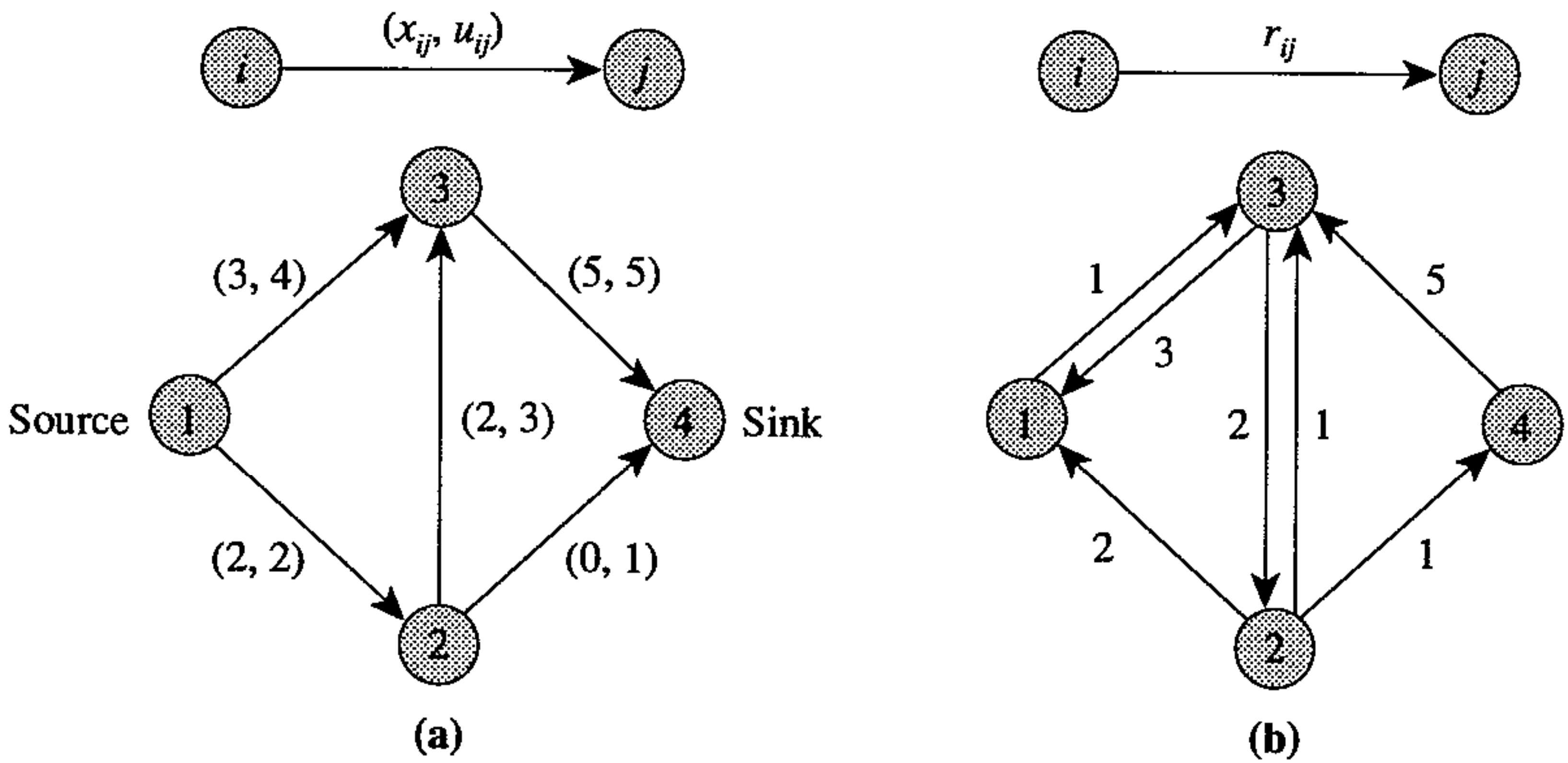
$$r[S, \bar{S}] = \sum_{(i,j)\in(S,\bar{S})} r_{ij}.$$

Generic Augmenting Path Algorithm

Generic Augmenting Path Algorithm

- ***Augmenting path***
  - a directed path from the source to the sink in the residual network
- ***Residual capacity of an augmenting path***
  - the minimum residual capacity of any arc in the path.

Generic Augmenting Path Algorithm



– the residual network contains exactly one augmenting path 1-3-2-4

– the residual capacity of this path is $\delta = \min\{r_{13}, r_{32}, r_{24}\} = \min\{1, 2, 1\} = 1$.

Generic Augmenting Path Algorithm

- The capacity δ of an augmenting path is always positive.
  - Consequently, whenever the network contains an augmenting path, we can send additional flow from the source to the sink.
- The generic augmenting path algorithm is essentially based on this simple observation.
- The algorithm proceeds by identifying augmenting paths and augmenting flows on these paths until the network contains no such path.

Generic Augmenting Path Algorithm

- ***Generic augmenting path algorithm***

algorithm *augmenting path*;
begin
    $x := 0$;
    while $G(x)$ contains a directed path from node s to node t **do**
    begin
        identify an augmenting path P from node s to node t;
        $\delta := \min\{r_{ij} : (i, j) \in P\}$;
        augment δ units of flow along P and update $G(x)$;
    end;
end;

- ***<Animation>***

The End