15.082 and 6.855J
February 13, 2003

Flow Decomposition and Transformations

Flow Decomposition and Transformations

- Flow Decomposition
- Removing Lower Bounds
- Removing Upper Bounds
- Node splitting

- Arc flows: an arc flow x is a vector x satisfying:

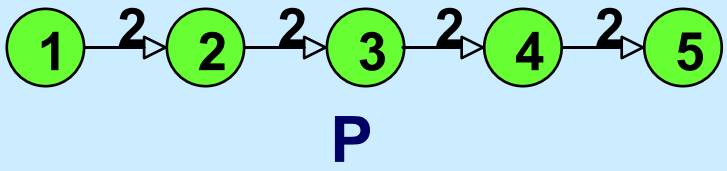
Let $b(i) = \sum_j x_{ij} - \sum_i x_{ji}$

We are not focused on upper and lower bounds on x for now.

Flows along Paths

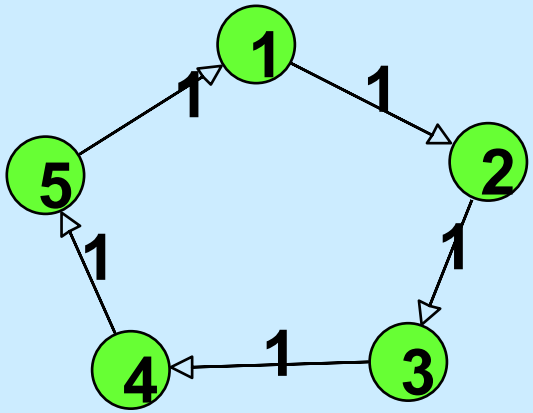
Usual: represent flows in terms of flows in arcs.

Alternative: represent a flow as the sum of flows in paths and cycles.



P

Two units of flow in the path P



C

One unit of flow around the cycle C

Properties of Path Flows

Let P be a directed path.

Let *Flow(δ,P)* be a flow of δ units in each arc of the path P.

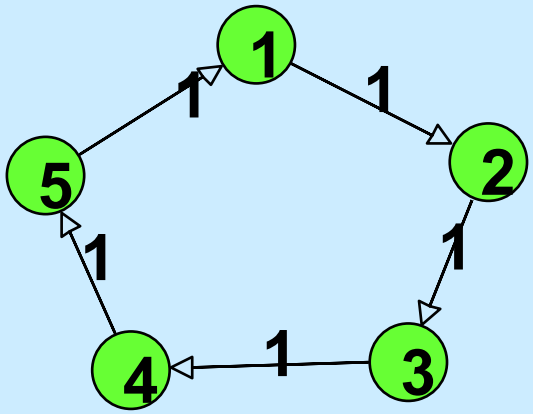
P

Flow(2, P)

Observation. If P is a path from s to t, then Flow(δ,P) sends δ units of flow from s to t, and has conservation of flow at other nodes.

Property of Cycle Flows

- If p is a cycle, then sending one unit of flow along p satisfies conservation of flow everywhere.



Representations as Flows along Paths and Cycles

Let $\mathcal{P}$ be a collection of Paths; let f(P) denote the flow in path P

$\delta_{ij}(P) = 1$ if $(i,j) \in P$

$\delta_{ij}(P) = 0$ if $(i,j) \notin P$

Let $\mathcal{C}$ be a collection of cycles; let f(C) denote the flow in cycle C.

$\delta_{ij}(C) = 1$ if $(i,j) \in C$

$\delta_{ij}(C) = 0$ if $(i,j) \notin C$

Representations as Flows along Paths and Cycles

- **Claim:** one can convert the path and cycle flows into an arc flow x as follows: for each arc $(i,j) \in A$

$$x_{ij} = \sum_{P \in \mathcal{P}} \delta_{ij}(P) f(P) + \sum_{C \in \mathcal{C}} \delta_{ij}(C) f(C)$$

- We next provide an algorithm for converting arc flows to sums of flows around cycles and along paths, where each path is from a supply node wrt x to a demand node wrt x.

Flow Decomposition Animation

Flow Decomposition

Notation:

G = (N, A): the network

x: Initial flow

y: Flow at intermediate stage of the algorithm

A(y) arcs with positive flow in y

N(y) nodes incident to an arc in A(y)

SupplyNodes(y) nodes with supply wrt y

DemandNodes(y) nodes with demand wrt y

$\mathcal{P}$: paths with flow in the decomposition

$\mathcal{C}$: cycles with flow in the decomposition

Flow Decomposition Algorithm

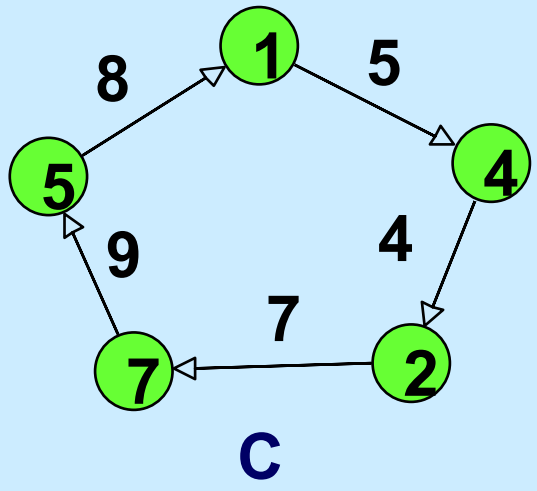
Initialize
begin
$\mathcal{P} := \emptyset$;
$\mathcal{C} := \emptyset$;
end

Search(s, y)

Carry out a depth first search starting with node s until finding a directed cycle C in G(y) or a path P in G(y) from s to a node t in DemandNodes(y).

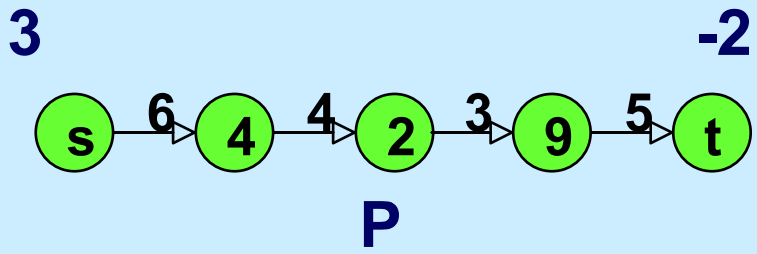
Select(s, y)
begin
if SupplyNodes(y) = $\emptyset$**, then s** $\in$ **SupplyNodes(y);**
else s $\in$ **N(y)**
end

Capacities of Paths and Cycles



The capacity of C is denoted as $\Delta(C, y)$ = min arc flow on C wrt flow y.

$\Delta(C) = 4$



$\Delta(P) = 2$

Let P be a path from s to t. The capacity of P is denoted as $\Delta(P, y)$ = min[b(s), -b(t), min (x_{ij} : (i,j) $\in$ P)]

The decomposition algorithm

```
begin
    Initialize
    while y ≠ ∅ do
    begin
        Select(s, y)
        Search(s, y)
        if a cycle C  is found then do
        begin
            let Δ = Capacity(C, y)
            Add Flow(Δ, C) to cycle flows
            Subtract Flow(Δ, C) from y.
        end
        if a path P is found then do
        begin
            let Δ = Capacity(P, y)
            Add Flow(Δ, P) to path flows
            Subtract Flow(Δ, P) from y.
        end
end
```

Flow Decomposition Animation

Complexity Analysis

- *Select initial node:*
  - O(1) per path or cycle, assuming that we maintain a set of supply nodes and a set of balanced nodes incident to a positive flow arc
- *Find cycle or path*
  - O(n) per path or cycle since finding the next arc in depth first search takes O(1) steps.
- *Update step*
  - O(n) per path or cycle

Complexity Analysis (continued)

Lemma. The number of paths and cycles found in the flow decomposition is at most m + n – 1.

Proof. In the update step for a cycle, at least one of the arcs has its capacity reduced to 0, and the arc is eliminated.

In an update step for a path, either an arc is eliminated, or a supply node has its supply reduced to 0, or a demand node has its demand reduced to 0.

(Also, there is never a situation with exactly one node whose supply is non-zero).

Conclusion

Flow Decomposition Theorem. **Any non-negative feasible flow x can be decomposed into the following:**

i. **the sum of flows in paths directed from supply nodes to demand nodes, plus**

ii. **the sum of flows around directed cycles.**

It will always have at most n + m paths and cycles.

Remark. The decomposition usually is not unique.

Corollary

A *circulation* is a flow with the property that the flow in is the flow out for each node.

Flow Decomposition Theorem for circulations. Any non-negative feasible flow x can be decomposed into the sum of flows around directed cycles.

It will always have at most m cycles.

An application of Flow Decomposition

Consider the problem of finding a shortest path from node 1 to all other nodes. Assume that all arc costs are non-negative.

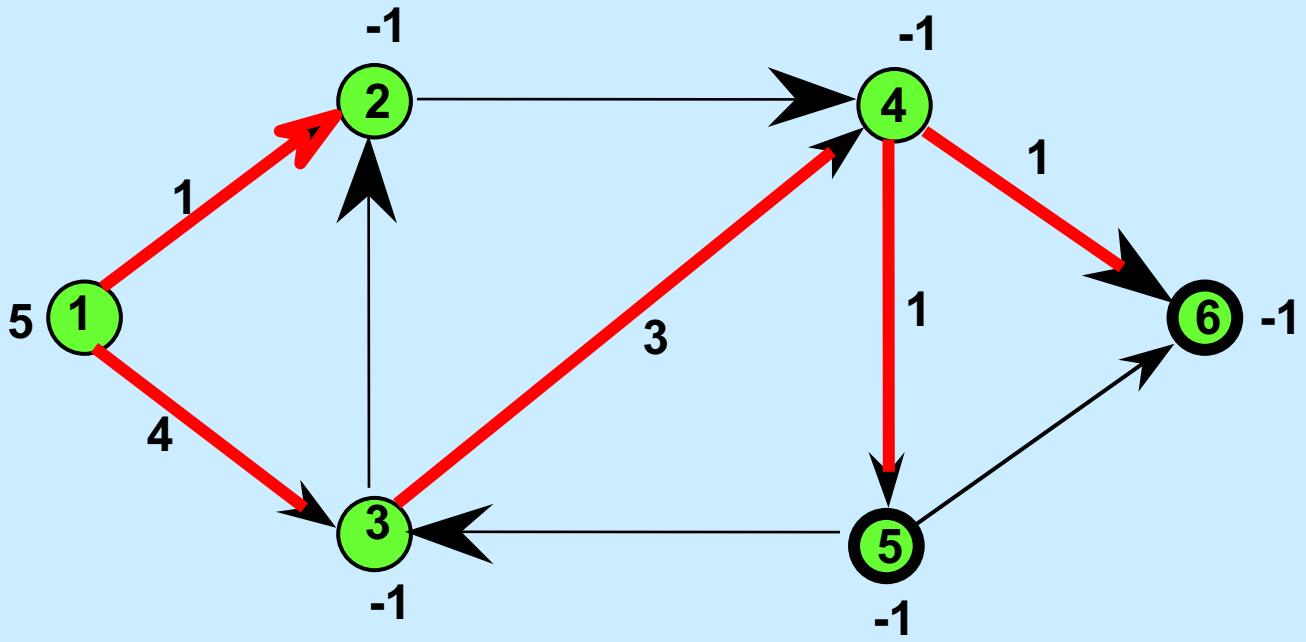
The model:

Minimize $\sum_{(i,j)\in A} c_{ij}x_{ij}$

Subject to $\sum_j x_{ij} - \sum_j x_{ji} = \begin{cases} n-1 & \text{if } i=1 \\ -1 & \text{if } i\neq 1 \end{cases}$

- **Let x* denote the minimum cost flow. Then x* is the sum of flows of paths from node 1 to each other node, plus the sum of flows around cycles.**

A shortest path tree



The decomposition of flows yields the paths:

1-2, 1-3, 1-3-4

1-3-4-5 and 1-3-4-6.

There are no cycles in the decomposition.

Trees have no cycles.

Other Applications of Flow Decomposition

- Reformulations of Problems.
  - There are network flow models that use path and cycle based formulations.
  - Multicommodity Flows
- Used in proving theorems
- Can be used in developing algorithms

Transformations

The minimum cost flow problem

u_{ij} = capacity of arc (i,j).

c_{ij} = unit cost of flow sent on (i,j).

x_{ij} = amount shipped on arc (i,j)

Minimize $\sum c_{ij}x_{ij}$

$$\sum_j x_{ij} - \sum_k x_{ki} = b_i \quad \text{for all } i \in N.$$

$$\text{and } 0 \le x_{ij} \le u_{ij} \quad \text{for all } (i,j) \in A.$$

If a constraint matrix (ignoring upper bounds) has at most one 1 and at most one -1 in each column, it corresponds to a network flow problem.

Comment on Transformations

Some transformations work directly with the LP formulations (system of equalities and inequalities)

Others work directly on the networks

Eliminating Lower Bound on Arc Flows

Suppose that there is a lower bound l_{ij} on the arc flow in (i,j)

Then let $y_{ij} = x_{ij} - l_{ij}$. Then $x_{ij} = y_{ij} + l_{ij}$

Minimize $\Sigma\ c_{ij}x_{ij}$

$$\Sigma_j\ x_{ij} - \Sigma_k\ x_{ki} = b_i \quad \text{for all } i \in N.$$

$$\text{and } l_{ij} \le x_{ij} \le u_{ij} \quad \text{for all } (i,j) \in A.$$

Minimize $\Sigma\ c_{ij}(y_{ij} + l_{ij})$

$$\Sigma_j\ (y_{ij} + l_{ij}) - \Sigma_k\ (y_{ij} + l_{ij}) = b_i \quad \text{for all } i \in N.$$

$$\text{and } l_{ij} \le (y_{ij} + l_{ij}) \le u_{ij} \quad \text{for all } (i,j) \in A.$$

Then simplify the expressions.

Eliminating Upper Bounds on Arc Flows

The minimum cost flow problem

$$\text{Min} \sum c_{ij}x_{ij}$$

$$\text{s.t.} \sum_j x_i - \sum_k x_{ki} = b_i \text{ for all } i \in N.$$

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

Subtract constraint 3 from constraint 1

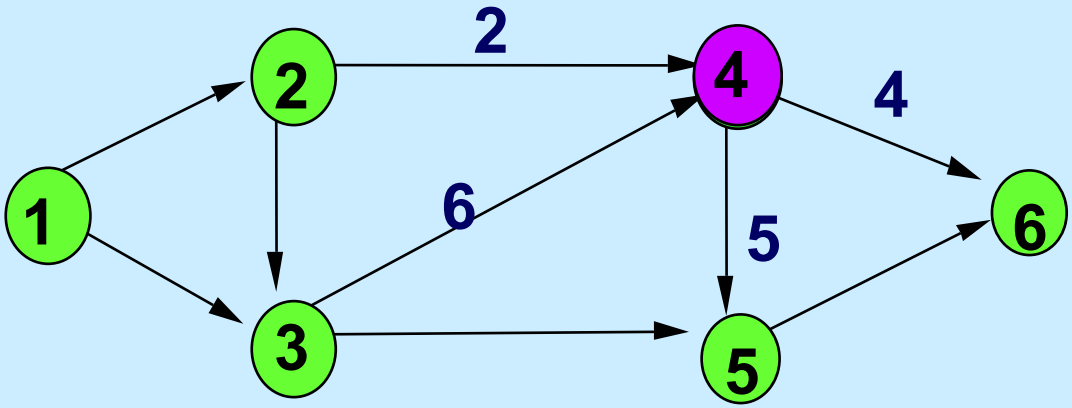
Before

$$\begin{array}{c} \\ i \\ j \\ u_{ij} \end{array} \begin{array}{c} \begin{array}{cc} x_{ij} & s_{ij} \end{array} \\ \begin{bmatrix} 1 & 0 \\ -1 & 0 \\ 1 & 1 \end{bmatrix} \end{array} \begin{array}{l} \\ = b_i \\ = b_j \\ = u_{ij} \end{array}$$

After

$$\begin{array}{c} \\ i \\ j \\ u_{ij} \end{array} \begin{array}{c} \begin{array}{cc} x_{ij} & s_{ij} \end{array} \\ \begin{bmatrix} 0 & -1 \\ -1 & 0 \\ 1 & 1 \end{bmatrix} \end{array} \begin{array}{l} \\ = b_i - u_{ij} \\ = b_j \\ = u_{ij} \end{array}$$

Node Splitting

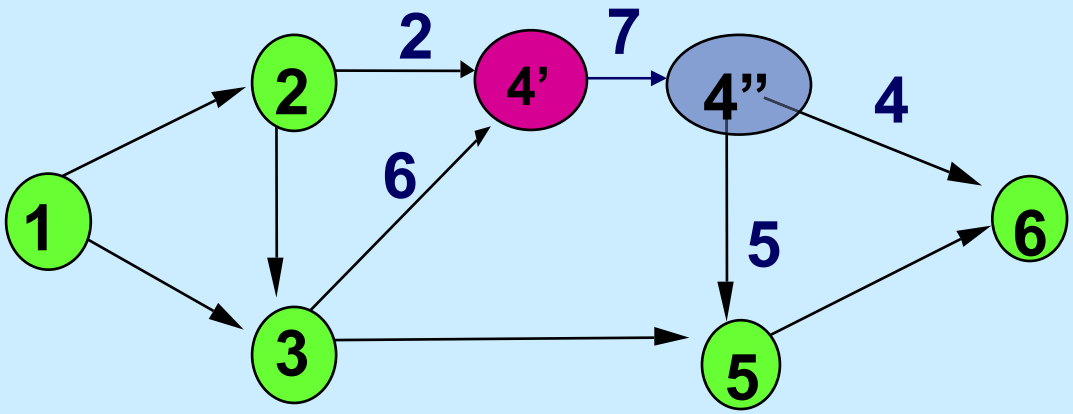


Flow x

Arc numbers are capacities

Suppose that we want to add the constraint that the flow into node 4 is at most 7.

Method: split node 4 into two nodes, say 4' and 4"



Flow x' can be obtained from flow x, and vice versa.

Summary

1. Any feasible flow can be decomposed into the sum of flows around cycles plus the sum of flows from nodes with supply to nodes with demand.

2. Lower bounds that are not 0 can be replaced (via a simple transformation) with lower bounds that are 0.

3. Each upper bound on an arc can be eliminated, but it is at the expense of creating an additional node.

4. One can model constraints on flow into or out of a node by splitting the node