15.082J, 6.855J, and ESD.78J
September 21, 2010

Eulerian Walks
Flow Decomposition and Transformations
Eulerian Walks in Directed Graphs in $O(m)$ time.

Step 1. Create a breadth first search tree into node 1. For $j$ not equal to 1, put the arc out of $j$ in $T$ last on the arc list $A(j)$.

Step 2. Create an Eulerian cycle by starting a walk at node 1 and selecting arcs in the order they appear on the arc lists.
Proof of Correctness

Relies on the following observation and invariant:

Observation: The walk will terminate at node 1. Whenever the walk visits node \( j \) for \( j \neq 1 \), the walk has traversed one more arc entering node \( j \) than leaving node \( j \).

Invariant: If the walk has not traversed the tree arc for node \( j \), then there is a path from node \( j \) to node 1 consisting of nontraversed tree arcs.
Eulerian Cycles in undirected graphs

Strategy: reduce to the directed graph problem as follows:

Step 1. Use dfs to partition the arcs into disjoint cycles

Step 2. Orient each arc along its directed cycle. Afterwards, for all i, the number of arcs entering node i is the same as the number of arcs leaving node i.

Step 3. Run the algorithm for finding Eulerian Cycles in directed graphs
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.

Two units of flow in the path $P$

One unit of flow around the cycle $C$
Properties of Path Flows

Let $P$ be a directed path.

Let $\text{Flow}(\delta, P)$ be a flow of $\delta$ units in each arc of the path $P$.

**Observation.** If $P$ is a path from $s$ to $t$, then $\text{Flow}(\delta, P)$ sends units of $\delta$ 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 $P$.

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

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

$$x_{ij} = \sum_{P \ni (i,j)} f(P) + \sum_{C \ni (i,j)} f(C)$$
Flow Decomposition

\( x \): Initial flow

\( y \): updated flow

\( G(y) \): subgraph with arcs \((i, j)\) with \(y_{ij} > 0\) and incident nodes

\( f(P) \): Flow around path \(P\) (during the algorithm)

\( P \): paths with flow in the decomposition

\( C \): cycles with flow in the decomposition

**INVARIANT**

\[
x_{ij} = y_{ij} + \sum_{P \ni (i, j)} f(P) + \sum_{C \ni (i, j)} f(C)
\]

Initially, \( x = y \) and \( f = 0 \).

At end, \( y = 0 \), and \( f \) gives the flow decomposition.
Deficit and Excess Nodes

Let $x$ be a flow (not necessarily feasible)

If the flow out of node $i$ exceeds the flow into node $i$, then node $i$ is a **deficit** node.

Its deficit is $\sum_j x_{ij} - \sum_k x_{ki}$.

If the flow out of node $i$ is less than the flow into node $i$, then node $i$ is an **excess** node.

Its excess is $-\sum_j x_{ij} + \sum_k x_{ki}$.

If the flow out of node $i$ equals the flow into node $i$, then node $i$ is a **balanced** node.
Flow Decomposition Algorithm

Step 0. Initialize: \( y := x; \quad f := 0; \quad \mathcal{P} := \emptyset; \quad \mathcal{C} := \emptyset; \)

Step 1. Select a deficit node \( j \) in \( G(y) \). If no deficit node exists, select a node \( j \) with an incident arc in \( G(y) \);

Step 2. Carry out depth first search from \( j \) in \( G(y) \) until finding a directed cycle \( W \) in \( G(y) \) or a path \( W \) in \( G(y) \) from \( s \) to a node \( t \) with excess in \( G(y) \).

Step 3.
1. Let \( \Delta \) = capacity of \( W \) in \( G(y) \). (See next slide)
2. Add \( W \) to the decomposition with \( f(W) = \Delta \).
3. Update \( y \) (subtract flow in \( W \)) and excesses and deficits
4. If \( y \neq 0 \), then go to Step 1
The capacity of C is
= min arc flow on C
wrt flow y.
capacity = 4

The capacity of P is
denoted as Δ(P, y) =
min[ def(s), excess(t),
min (x_ij : (i,j) ∈ P) ]

χαπαχιτψ = 2

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
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 deficit node has its deficit reduced to 0, or an excess node has its excess reduced to 0.

(Also, there is never a situation with exactly one node whose excess or deficit 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 deficit nodes to excess 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.
Consider a feasible flow where the supply of node 1 is n-1, and the supply of every other node is -1.

\[
\sum_j x_{ij} - \sum_j x_{ji} = \begin{cases} 
n - 1 & \text{if } i = 1 \\
-1 & \text{if } i \neq 1 \end{cases}
\]

Suppose the arcs with positive flow have no cycle. Then the flow can be decomposed into unit flows along paths from node 1 to node j for each j \neq 1.
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.
Application to shortest paths

To find a shortest path from node 1 to each other node in a network, find a minimum cost flow in which \( b(1) = n-1 \) and \( b(j) = -1 \) for \( j \neq 1 \).

The flow decomposition gives the shortest paths.
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
The min cost flow problem (again)

The minimum cost flow problem

\[ u_{ij} = \text{capacity of arc (i,j)}. \]

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

\[ x_{ij} = \text{amount shipped on arc (i,j)} \]

Minimize \[ \sum c_{ij}x_{ij} \]

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

and \[ 0 \leq x_{ij} \leq u_{ij} \quad \text{for all } (i,j) \in A. \]
The model seems very limiting

- The lower bounds are 0.
- The supply/demand constraints must be satisfied exactly
- There are no constraints on the flow entering or leaving a node.

We can model each of these constraints using transformations.

- In addition, we can transform a min cost flow problem into an equivalent problem with no upper bounds.
Eliminating Lower Bound on Arc Flows

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

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

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

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

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

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

$\sum_j (y_{ij} + l_{ij}) - \sum_k (y_{ij} + l_{ij}) = b_i$ for all $i \in N.$

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

Then simplify the expressions.
Allowing inequality constraints

Minimize \( \sum c_{ij}x_{ij} \)
\[
\sum_j x_{ij} - \sum_k x_{ki} \leq b_i \quad \text{for all } i \in N.
\]
and \( l_{ij} \leq x_{ij} \leq u_{ij} \) for all \((i,j) \in A\).

Let \( B = \sum_i b_i \). For feasibility, we need \( B \geq 0 \).

Create a “dummy node” \( n+1 \), with \( b_{n+1} = -B \). Add arcs \((i, n+1)\) for \( i = 1 \) to \( n \), with \( c_{i,n+1} = 0 \). Any feasible solution for the original problem can be transformed into a feasible solution for the new problem by sending excess flow to node \( n+1 \).
Node Splitting

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.
Eliminating Upper Bounds on Arc Flows

The minimum cost flow problem

\[
\text{Min} \quad \sum c_{ij} x_{ij}
\]

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

and \[ 0 \leq x_{ij} \leq u_{ij} \text{ for all } (i,j) \in A. \]
Summary

1. Efficient implementation of finding an eulerian cycle.

2. Flow decomposition theorem

3. Transformations that can be used to incorporate constraints into minimum cost flow problems.