Network Flow Duality and Applications of Network Flows
Overview of Lecture

• Applications of network flows
  • shortest paths
  • maximum flow
  • the assignment problem
  • minimum cost flows

• Linear programming duality in network flows and applications of dual network flow problems
• Applications of network flows
  • shortest paths
  • maximum flow
  • the assignment problem
  • minimum cost flows
Most reliable paths

Let $p_{ij}$ be the probability that an arc is working, and that all arcs are independent.

The probability that a path $P$ is working is $\prod_{(i,j) \in P} p_{ij}$

What is the most reliable path from $s$ to $t$, that is the one that maximizes the probability of working?

$$\max \prod_{(i,j) \in P} p_{ij} \quad \text{s.t.} \quad P \in \mathcal{P}(s,t)$$

$$\min \prod_{(i,j) \in P} \frac{1}{p_{ij}} \quad \text{s.t.} \quad P \in \mathcal{P}(s,t)$$

$$\min \log \left( \prod_{(i,j) \in P} \frac{1}{p_{ij}} \right) = \sum_{(i,j) \in P} \log \left( \frac{1}{p_{ij}} \right) \quad \text{s.t.} \quad P \in \mathcal{P}(s,t)$$

Let $c_{ij} = \log \frac{1}{p_{ij}}$
Suppose that the time it takes to travel in arc \((i, j)\) depends on when one starts. (e.g., rush hour vs. other hours in road networks.)

Let \(c_{ij}(t)\) be the time it takes to travel in \((i, j)\) starting at time \(t\). What is the minimum time it takes to travel from node 1 to node \(n\) starting at 7 AM?

<table>
<thead>
<tr>
<th>Start time</th>
<th>7</th>
<th>7:10</th>
<th>7:20</th>
<th>7:30</th>
<th>7:40</th>
<th>7:50</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1,2)</td>
<td>20</td>
<td>30</td>
<td>30</td>
<td>20</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>(1,3)</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>(2,3)</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>(3,4)</td>
<td>10</td>
<td>20</td>
<td>20</td>
<td>10</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>...</td>
<td>...</td>
<td></td>
</tr>
</tbody>
</table>
What is the minimum time $T$ such that there is a path from node 1 at 7 AM to node $n$ at time $T$?
• Applications of network flows
  • shortest paths
  • maximum flows
  • the assignment problem
  • minimum cost flows
Suppose there are 2 parallel machines. Is there a feasible schedule?

The best (infeasible) schedule without preemption.
A feasible schedule with preemption

Preemption permits a job to be split into two or more parts and scheduled on the same or different machines, but not two machines at the same time. How can one find a feasible schedule when preemption is allowed?

<table>
<thead>
<tr>
<th>Job(j)</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Processing Time</td>
<td>1.5</td>
<td>3</td>
<td>4.5</td>
<td>5</td>
</tr>
<tr>
<td>Release Time</td>
<td>2</td>
<td>0</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>Due Date</td>
<td>5</td>
<td>4</td>
<td>7</td>
<td>9</td>
</tr>
</tbody>
</table>
Time allocation is the thing that is flowing.

- Job $j$ must have $p(j)$ units of time allocated to it.
- Job $j$ can be scheduled for at most one time unit in any period.
- If there are $m$ machines, then at most $m$ different jobs may be scheduled in any period.

$u_{ij} = 1$ for arcs from green to yellow
The optimal allocation and flow
A more efficient transformation

1. Merge two adjacent period nodes if the same set of tasks can be scheduled in both periods. The number of nodes after merging is less than 2 |J|, where J = set of jobs.
Has Tampa already been eliminated from winning in this hypothetical season finale?

<table>
<thead>
<tr>
<th></th>
<th>Games Won</th>
<th>Games Left</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bos</td>
<td>82</td>
<td>8</td>
</tr>
<tr>
<td>NY</td>
<td>77</td>
<td>8</td>
</tr>
<tr>
<td>Balt</td>
<td>80</td>
<td>8</td>
</tr>
<tr>
<td>Tor</td>
<td>79</td>
<td>8</td>
</tr>
<tr>
<td>Tamp</td>
<td>74</td>
<td>9</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>Bos</th>
<th>NY</th>
<th>Balt</th>
<th>Tor</th>
<th>Tamp</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bos</td>
<td>--</td>
<td>1</td>
<td>4</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>NY</td>
<td>1</td>
<td>--</td>
<td>0</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>Balt</td>
<td>4</td>
<td>0</td>
<td>--</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Tor</td>
<td>1</td>
<td>3</td>
<td>1</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>Tamp</td>
<td>2</td>
<td>4</td>
<td>0</td>
<td>3</td>
<td>0</td>
</tr>
</tbody>
</table>
Data assuming that Tampa wins all its remaining games.

<table>
<thead>
<tr>
<th>Team</th>
<th>Games Won</th>
<th>Games Left</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bos</td>
<td>82</td>
<td>6</td>
</tr>
<tr>
<td>NY</td>
<td>77</td>
<td>4</td>
</tr>
<tr>
<td>Balt</td>
<td>80</td>
<td>6</td>
</tr>
<tr>
<td>Tor</td>
<td>79</td>
<td>5</td>
</tr>
<tr>
<td>Tamp</td>
<td>83</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>Bos</th>
<th>NY</th>
<th>Balt</th>
<th>Tor</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bos</td>
<td>--</td>
<td>1</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>NY</td>
<td>1</td>
<td>--</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>Balt</td>
<td>4</td>
<td>0</td>
<td>--</td>
<td>1</td>
</tr>
<tr>
<td>Tor</td>
<td>1</td>
<td>3</td>
<td>1</td>
<td>--</td>
</tr>
</tbody>
</table>

Games remaining

Question: can the remaining games be played so that no team wins more than 83 games?

http://riot.ieor.berkeley.edu/~baseball/
Flow on (i,j) is interpreted as games won.
A Maximum Flow

Red Sox vs. Yankees

Red Sox vs. Orioles

Red Sox vs. Blue Jays

Yankees vs. Blue Jays

Orioles vs. Blue Jays

S

1

1

1

1

1

1

1

1

1

4

3

3

4

4

3

3

1

1

1

1
• Applications of network flows
  • shortest paths
  • maximum flows
  • the assignment problem
  • minimum cost flows
The Assignment Problem

- **n persons**
- **n tasks**
- \( u_{ij} = \text{utility of assigning person } i \text{ to task } j \)

Maximize the sum of the utilities:

\[
\max \sum_{i=1}^{n} \sum_{j=1}^{n} u_{ij} x_{ij}
\]

- Each person gets assigned to a task:
  \[
  \sum_{j=1}^{n} x_{ij} = 1 \quad \forall i
  \]
- Each task has one person assigned to it:
  \[
  \sum_{i=1}^{n} x_{ij} = 1 \quad \forall j
  \]

\( x_j \in \{0, 1\} \quad \forall j \)
Identifying Moving Targets in Space

Suppose that there are moving targets in space.
You can identify each target as a pixel on a radar screen.
Given two successive pictures, identify how the targets have moved.

This is an efficient way of tracking items.
• Applications of network flows
  • shortest paths
  • maximum flows
  • the assignment problem
  • minimum cost flows
A postman (using a postal truck) wants to visit every city street at least once while minimizing the total travel time.

\[\text{x}_{ij} = \text{number of times that the postman traverses arc (i, j)}\]

\[\text{c}_{ij} = \text{length of (i, j)}\]

\[
\begin{align*}
\min & \quad \sum_{(i, j) \in A} c_{ij} x_{ij} \\
\text{s.t.} & \quad \sum_j x_{ij} - \sum_k x_{ki} = 0 \quad \forall i \in N \\
& \quad x_{ij} \geq 1 \quad \forall (i, j) \in A
\end{align*}
\]
Chinese Postman Problem (undirected)

- odd degree
- even degree

\[ d_{ij} = \text{minimum length of a path from node } i \text{ to node } j. \]

Compute \(d_{ij}\) for all nodes \(i, j\) of odd degree.

Add paths joining nodes of odd degree so as to minimize the total \(d\)-length. (This is a nonbipartite matching problem.)
Duality for Network Flow Problems

Each network flow problem has a corresponding problem called the “dual”.

Rest of lecture:

- Review of max flow min cut theorem
- Weak Duality in linear programming
- Strong Duality in linear programming
- Duality for shortest paths plus applications
- Duality for min cost flow plus applications
Theorem. (Weak duality). If $x$ is any s-t flow and if $(S, T)$ is any s-t cut, then the flow out of $s$ is at most the capacity of the cut $(S,T)$.

In this example, the capacities of all arcs is 1.

The max flow consists of the thick arcs.

If $S =$ red nodes, then min cut $(S, N\setminus S)$.

Theorem. (Max-flow Min-Cut). The maximum flow value is the minimum capacity of a cut.
Weak Duality in Linear Programming

**Weak Duality Theorem.**
Suppose that $x$ is feasible for the primal problem and that $\pi$ is feasible for the dual problem. Then $cx \leq \pi b$.

**Primal Problem**

\[
\begin{align*}
\text{max} & \quad cx \\
\text{s.t.} & \quad Ax = b \\
& \quad x \geq 0
\end{align*}
\]

**Dual Problem**

\[
\begin{align*}
\text{min} & \quad \pi b \\
\text{s.t.} & \quad \pi A \geq c
\end{align*}
\]

**Proof**

\[
\begin{align*}
\pi A \geq c \quad \text{and} \quad x \geq 0 & \implies \pi Ax \geq cx \\
Ax = b & \implies \pi Ax = \pi b
\end{align*}
\]

Therefore,

\[
\pi b \geq cx.
\]
**Strong Duality in Linear Programming**

**Primal Problem**

\[
\begin{align*}
\text{max} & \quad cx \\
\text{s.t.} & \quad Ax = b \\
& \quad x \geq 0
\end{align*}
\]

**Dual Problem**

\[
\begin{align*}
\text{min} & \quad \pi b \\
\text{s.t.} & \quad \pi A \geq c
\end{align*}
\]

**Strong Duality Theorem.**

Suppose that the primal problem has an optimal solution \(x^*\).

Then the dual problem also has an optimal solution, say \(\pi^*\), and the two optimum objective values are the same. That is, \(cx^* = \pi^*b\).

**Note:** it is not obvious that max-flow min-cut is a special case of LP duality.
Duality for the shortest path problem

Let $G = (N, A)$ be a network, and let $c_{ij}$ be the length or cost of arc $(i, j)$. The shortest path problem is to find the path of shortest length from node 1 to node $n$.

We say that a distance vector $d(\ )$ is dual feasible for the shortest path problem if

1. $d(1) = 0$
2. $d(j) \leq d(i) + c_{ij}$ for all $(i, j) \in A$.

The dual shortest path problem is to maximize $d(n)$ subject to the vector $d(\ )$ being dual feasible.
Duality Theorem for Shortest Path Problem

Let $G = (N, A)$ be a network, and let $c_{ij}$ be the length or cost of arc $(i, j)$. If there is no negative cost cycle, then the minimum length of a path from node 1 to node $n$ is the maximum value of $d(n)$ subject to $d(\ )$ being dual feasible.
TeX optimally decomposes paragraphs by selecting the breakpoints for each line optimally. It has a subroutine that computes the attractiveness $F(i,j)$ of a line that begins at word $i$ and ends at word $j-1$. How can one use $F(i,j)$ to create a shortest path problem whose solution will solve the paragraph problem?

The paragraph layout problem can be modeled as a shortest path problem or the dual of a shortest path problem.
Let $d^*(j)$ be the value of laying out words 1 to $j-1$ most attractively. $d^*$ can be computed as follows.

$$\min d(n+1)$$

subject to

$$d(j) \geq d(i) + F(i, j) \quad \forall (i,j) \in A$$

$$d(i) \leq d(j) - F(i, j)$$

$$d(1) = 0$$

For any feasible vector $d$, $d(j)$ is an upper bound on the beauty of laying out words 1 to $j-1$.

The most accurate upper bound gives the optimum beauty.

There is a close connection to dynamic programming.
Application: project scheduling

<table>
<thead>
<tr>
<th>Activity</th>
<th>Predecessor</th>
<th>Duration</th>
</tr>
</thead>
<tbody>
<tr>
<td>A (Train workers)</td>
<td>None</td>
<td>6</td>
</tr>
<tr>
<td>B (Purchase raw materials)</td>
<td>None</td>
<td>9</td>
</tr>
<tr>
<td>C (Make subassembly 1)</td>
<td>A, B</td>
<td>8</td>
</tr>
<tr>
<td>D (Make subassembly 2)</td>
<td>B</td>
<td>7</td>
</tr>
<tr>
<td>E (Inspect subassembly 2)</td>
<td>D</td>
<td>10</td>
</tr>
<tr>
<td>F (Assemble subassemblies)</td>
<td>C, E</td>
<td>12</td>
</tr>
</tbody>
</table>
Application: project scheduling

<table>
<thead>
<tr>
<th>Activity</th>
<th>Predecessor</th>
<th>Duration</th>
</tr>
</thead>
<tbody>
<tr>
<td>A (Train workers)</td>
<td>None</td>
<td>6</td>
</tr>
<tr>
<td>B (Purchase raw materials)</td>
<td>None</td>
<td>9</td>
</tr>
<tr>
<td>C (Make subassembly 1)</td>
<td>A, B</td>
<td>8</td>
</tr>
<tr>
<td>D (Make subassembly 2)</td>
<td>B</td>
<td>7</td>
</tr>
<tr>
<td>E (Inspect subassembly 2)</td>
<td>D</td>
<td>10</td>
</tr>
<tr>
<td>F (Assemble subassemblies)</td>
<td>C, E</td>
<td>12</td>
</tr>
</tbody>
</table>

Let $s(i)$ be the start time of task $i$.
Let $f(i)$ be the finish time of task $i$.
Let $p(i)$ be the processing time of task $i$.

minimize $f(n)$
subject to $s(0) = 0$
$f(j) = s(j) + p(j)$ for all $j \neq 0$ or $n$
$s(j) \geq f(i)$ if $i$ precedes $j$.

Corresponds to a longest path problem. We can make it a shortest path problem by letting $g(j) = -f(j)$ for all $j$. 


Let $s(i)$ be the start time of task $i$.
Let $f(i)$ be the finish time of task $i$.
Let $p(i)$ be the processing time of task $i$.

\[
\begin{align*}
\text{minimize} & \quad f(n) \\
\text{subject to} & \quad s(0) = 0 \\
& \quad f(j) = s(j) + p(j) \quad \text{for all } j \neq 0 \text{ or } n \\
& \quad s(j) \geq f(i) \quad \text{if } i \text{ precedes } j \\
& \quad s(j) \leq f(i) + h(i, j) \quad \text{or } f(i) \geq s(j) - h(i, j)
\end{align*}
\]

Suppose that for some tasks $i$ and $j$, task $j$ must be started within $h(i, j)$ time units of task $i$ finishing.
Duality for minimum cost flows

Uncapacitated min cost flow problem

min \[ \sum_{(i,j) \in A} c_{ij} x_{ij} \]

s.t. \[ \sum_j x_{ij} - \sum_k x_{ki} = b_i \quad \forall i \in N \]

\[ x_{ij} \geq 0 \quad \forall (i, j) \in A \]

Dual of the uncapacitated MCF problem.

max \[ \sum_{i \in N} \pi_i b_i \]

s.t. \[ c_{ij} - \pi_i + \pi_j \geq 0 \quad \forall (i, j) \in A \]

\[ \pi_i - \pi_j \leq c_{ij} \]

Theorem. Suppose that \( x \) is feasible for the uncapacitated MCF problem, and \( \pi \) is feasible for the dual problem. Then \( cx \geq \pi b \).

If one of the problems has a finite optimum, then so does the other, and the two values are the same.
More on Duality

1. One can solve the dual problem using an algorithm for solving the uncapacitated MCF problem.

2. Any linear programming problem in which every constraint is either a lower bound on a variable or an upper bound on a variable or of the form “$y_i - y_j \leq c_{ij}$” is the dual of a minimum cost flow problem.
Maximum Weight Closure of a Graph

Let $G = (N, A)$. Let $w_i$ be the weight of node $i$.

A subset $S \subseteq N$ is called a closure if there are no arcs leaving the subset. That is, if $i \in S$ and if $(i, j) \in A$, then $j \in S$.

The maximum weight closure problem is to find a closure of maximum weight. It is the dual of a minimum cost flow problem.

$$\max \sum_{i \in N} w_i y_i$$

subject to

$$y_i - y_j \leq 0 \quad \forall (i, j) \in A$$

$$0 \leq y_i \leq 1 \quad \forall i \in N$$
Open Pit Mining

Suppose an open pit mine is subdivided into blocks. We create a graph $G = (N, A)$ as follows:

1. There is a node for each block
2. If block $j$ must be removed before block $i$, then $(i, j) \in A$.
3. The net revenue from block $i$ is $w_i$.

Special case of the closure problem.
Project management with “crashing”

Suppose that one can reduce the time at which task \( j \) is completed for each \( j \). The cost of reducing the time for task \( i \) is \( c_i \) per unit of time.

What is the least cost schedule that completes all tasks by time \( T \)?

Let \( s(i) \) be the start time of task \( i \).
Let \( f(i) \) be the finish time of task \( i \).
Let \( p(i) \) be the original processing time of task \( i \).

\[
\min \sum_i c_i \left( p(i) + s(i) - f(i) \right)
\]
\[
s.t. \quad f(j) - s(i) \leq 0 \quad \forall (i, j) \in A
\]
\[
\quad f(i) - s(i) \leq p(i) \quad \forall i \in N
\]
\[
\quad s(0) = 0; \quad f(n) \leq T
\]

The above LP is the dual of a minimum cost flow problem.
Summary

There are hundreds of direct applications of the minimum cost flow problem or its dual.

Even more common, min cost flow problems arise as subproblems of a larger and more complex problem. We will see more of this in a few lectures from now.

Next Lecture: the simplex algorithm for the min cost flow problem.