http://www.cs.unh.edu/~ruml/cs758
Applications of Cuts and Flows
Max-Flow Min-Cut Theorem

value of a flow = flow across any cut

any flow value \leq \text{capacity of cut}

Theorem: these are the same:
1. \( f \) is a maximum flow
2. the residual network \( G_f \) contains no augmenting paths
3. there exists a cut whose capacity is the value of \( f \)

1=2: FF is correct; 1=3: FF also finds minimum cuts
Image Segmentation

an image as a graph!

maximize

\[
\sum_{i \in A} a_i + \sum_{i \in B} b_i - \sum_{e \text{ cut by } A} p_{i,j}
\]

minimize

\[
\sum_{i \in A} b_i + \sum_{i \in B} a_i + \sum_{e \text{ cut by } A} p_{i,j}
\]

cut crosses three types of edges: \( s_i, t_i, \) and \( p_{i,j} \)
Maximum Matching

Applications
- Max-Flow Thm
- Segmentation
- Matching
- Scheduling
- Break
- LPs

bipartite graphs: jobs/machines, classes/instructors, ...
bipartite graphs: jobs/machines, classes/instructors, ...

unit capacities

flow = matching

FF guarantees integer flow

running time? (hint: bound $|f^*|$)
does a feasible schedule exist using only 3 machines (allowing preemption)?

<table>
<thead>
<tr>
<th>job</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>1.25</td>
<td>2.1</td>
<td>3.6</td>
</tr>
<tr>
<td>release date</td>
<td>3</td>
<td>1</td>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td>due date</td>
<td>5</td>
<td>4</td>
<td>7</td>
<td>9</td>
</tr>
</tbody>
</table>
does a feasible schedule exist using only 3 machines (allowing preemption)?

<table>
<thead>
<tr>
<th>job</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>1.25</td>
<td>2.1</td>
<td>3.6</td>
</tr>
<tr>
<td>release date</td>
<td>3</td>
<td>1</td>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td>due date</td>
<td>5</td>
<td>4</td>
<td>7</td>
<td>9</td>
</tr>
</tbody>
</table>

arcs from $s$ to jobs labeled with job size

intervals: 1–3–4–5–7–9

arcs from job to feasible intervals labeled with length of interval

arcs from interval to $t$ labeled with total achievable work (num machines times length of interval)
<table>
<thead>
<tr>
<th>Applications</th>
</tr>
</thead>
<tbody>
<tr>
<td>Max-Flow Thm</td>
</tr>
<tr>
<td>Segmentation</td>
</tr>
<tr>
<td>Matching</td>
</tr>
<tr>
<td>Scheduling</td>
</tr>
<tr>
<td>Break</td>
</tr>
</tbody>
</table>

### asst 11
Linear Programming
real variables, linear constraints, linear objective

eg, cheapest diet that meets nutrition guidelines

\[ y_{vitaminA} = 2.3x_{broccoli} + 1.7x_{carrots} \ldots \]
\[ cost = 4.99x_{broccoli} + 2.67x_{carrots} \ldots \]

minimize cost

subject to \( y_{vitaminA} > 500 \ldots \)

polynomial time (ellipsoid, Karmarkar's), but simplex method is popular

CPLEX, Gurobi, lpsolve
Example LPs

- shortest paths
- max flow obeying capacities
- min-cost flow meeting demand
- multicommodity flow meeting demands
- earliest finish time subject to job durations
Example: Max Flow

maximize $\sum_v f(s, v) - \sum_v f(v, s)$
subject to

- $f(u, v) \geq 0$ for every edge $(u, v)$
- $\sum_v f(u, v) = \sum_v f(v, u)$ for every vertex $u$
- $f(u, v) \leq c(u, v)$ for every edge $(u, v)$
Example: Shortest Path

maximize $d_t$

subject to

$$d_v \leq d_u + w(u, v) \text{ for every edge } (u, v)$$

$$d_s = 0$$
Min-cost Flow Meeting Demand

minimize $\sum_{(u,v)} a(u, v)f(u, v)$
subject to

$f(u, v) \geq 0$ for every edge $(u, v)$

$\sum_v f(u, v) = \sum_v f(v, u)$ for every vertex $u$

$f(u, v) \leq c(u, v)$ for every edge $(u, v)$

$\sum_v f(s, v) - \sum_v f(v, s) = d$
minimize 0
subject to

\[ \sum_{v} f_i(u, v) - \sum_{v} f_i(v, u) = 0 \]

\[ f_i(u, v) \geq 0 \]

\[ \sum_{i=1}^{k} f_i(u, v) \leq c(u, v) \]

\[ \sum_{v} f_i(s_i, v) - \sum_{v} f_i(v, s_i) = d_i \]
Beyond Linear Programming

**convex programming**: constraints and objective are convex polynomial time

**quadratic programming**: constraints and objective are quadratic

some forms are polynomial time

**integer linear programming**: integer variables, linear constraints and objective

NP-complete

**0-1 ILP**: 0-1 variables, linear constraints, linear objective

(BIP) NP-complete

**combinatorial optimization**: variables are discrete
selection

multicommodity flow is NP-hard for integer flows. Use LP for fractional flows.
For example:

- What’s still confusing?
- What question didn’t you get to ask today?
- What would you like to hear more about?

Please write down your most pressing question about algorithms and put it in the box on your way out.

*Thanks!*