CSE 421

Edmonds-Karp Algorithm, More Applications

Yin Tat Lee
FF may converge to wrong answer

- FF is not polynomial time. It can take $X$ steps for this graph.

- FF may not even converge to a correct flow for irrational capacity.

\[
\phi = \frac{\sqrt{5} - 1}{2}, \quad X = 3.
\]

Augmenting paths along the large figure.
Then,
In limit, it will send $4 + \sqrt{5}$ unit.
But, maxflow = 7.
Edmonds-Karp Algorithm

- Use a shortest augmenting path (via Breadth First Search in residual graph)
- Time: $O(m^2n)$.
Distance to $s$ is non-decreasing.

Let $f$ be a flow, $G_f$ the residual graph, and $P$ a shortest augmenting path. Then no vertex is closer to $s$ after augmentation along $P$.

Proof: Augmentation along $P$ only

- deletes forward edges
  - no new (hence no shorter) path created
- adds back edges that go to previous vertices along $P$
  - BFS is unchanged, since $v$ visited before $(u, v)$ examined
Distance for bottleneck edges

Let $d_f(s, v)$ be the distance from $s$ to $v$ on $G_f$.

Shortest s-t path $P$ in $G_f$

After augmenting along $P$ \[ d_f(s, v) = d_f(s, u) + 1 \]

For $(u, v)$ to be bottleneck again for some flow $f'$

\[ d_{f'}(s, u) = d_{f'}(s, v) + 1 \geq d_f(s, v) + 1 = d_f(s, u) + 2 \]
Theorem

Edmonds-Karp performs $O(mn)$ flow augmentations

Proof:

• Each step, some edge disappear from $G_f$.
  (Note however that some edge may reappear.)

• Any edge $(u, v)$ disappears from $G_f$ at most $n/2$ times.
  (because the distance increased by 2 every disappearance.)

• There are at most $mn/2$ disappearances.

Total time is $O(m^2n)$. 
Maximum flow

Current Best ($U =$ maximum capacity):

- $O((m + nF) \log^{O(1)}(nU))$ [Karger-Levine 02]
- $O(mn)$ [Orlin 13]
- $O(m\sqrt{n} \log^{O(1)}(nU))$ [Lee-Sidford 13]
- $O(m^{\frac{10}{7}}U^{\frac{1}{7}} \log^{O(1)}(nU))$ [Madry 13]
- $O((m + m^{\frac{3}{4}}n^{\frac{1}{4}} \sqrt{F}) \log^{O(1)}(nU))$ [Sidford-Tian 17]
Image Segmentation
Image Segmentation

Given an image we want to separate foreground from background

• Important problem in image processing.
• Divide image into coherent regions.
Foreground / background segmentation

Label each pixel as foreground/background.

- \( V \) = set of pixels, \( E \) = pairs of neighboring pixels.
- \( a_i \) is the original image.
- \( a_i \gg 0 \) means we prefer to label \( i \) in foreground.
- \( p_{i,j} \geq 0 \) is separation penalty for labeling one of \( i \) and \( j \) as foreground, and the other as background.

Goals:

Find partition \((S, \bar{S})\) that minimizes:

\[
- \sum_{i \in S} a_i + \sum_{(i,j) \in E} p_{i,j}
\]

where \( S \) is the foreground.
Min cut Formulation

\[G' = (V', E').\]
Add s to correspond to foreground;  
Add t to correspond to background;  
Use two anti-parallel edges instead of undirected edge.
Consider min cut \((S, \overline{S})\) in \(G'\). \((S = \text{foreground.})\)

\[
cap(S, \overline{S}) = \sum_{i \in S} -a_i \ 1_{a_i<0} + \sum_{i \in \overline{S}} a_i \ 1_{a_i>0} + \sum_{(i,j) \in E} p_{i,j}
\]

\[
= -\sum_{i \in S} a_i + \sum_{i \in S} a_i \ 1_{a_i>0} + \sum_{i \in \overline{S}} a_i \ 1_{a_i>0} + \sum_{i \in S, j \in \overline{S}} p_{i,j}
\]

\[
= -\sum_{i \in S} a_i + \sum_{i} a_i + \sum_{...} p_{i,j}
\]

\[
= -\sum_{i \in S} a_i + \sum_{...} p_{i,j} + \text{constant}
\]

Precisely, what we want to minimize.
Reality

- The main difficulty is to come up with a good model.
- May want to have human interaction.
- Segmentation may be real-valued instead of \( \{0,1\} \).
- There are many more than 1 objects.
- May need labeling.
- Augmenting path is not great for GPU.
Project Selection

- **Given** a DAG $G = (V, E)$ representing precedence constraints on tasks
  (a task points to its *predecessors*)
  a profit value $p(v)$ for each task $v \in V$
  (may be positive or negative)

- **Find** a set $A \subseteq V$ of tasks that is closed under predecessors,
  (i.e. if $(u, v) \in E$ and $u \in A$ then $v \in A$)
  that maximizes $\text{Profit}(A) = \sum_{v \in A} p(v)$

Each task points to its predecessor tasks
Extended Graph
Extended Graph $G'$

For each vertex $v$

If $p(v) \geq 0$ add $(s, v)$ edge
  with capacity $p(v)$

If $p(v) < 0$ add $(v, t)$ edge
  with capacity $-p(v)$
Extended Graph $G'$

- **Want:** Set capacities on edges of $G$ so that for minimum $s$-$t$-cut $(S, \overline{S})$ in $G'$, the set $A = S - \{s\}$
  - satisfies precedence constraints
  - has maximum possible profit in $G$

- Cut capacity with $S = \{s\}$ is just $C = \sum_{v: p(v) \geq 0} p(v)$
  \[ \text{Profit}(A) \leq C \] for any set $A$

- To satisfy constraints, don’t want any original edges of $G$ going forward across the minimum cut
  That would correspond to a task in $A = S - \{s\}$ that had a predecessor not in $A = S - \{s\}$

- Set capacity of each of the edges of $G$ to $+\infty$. 

Extended Graph $G'$

Capacity $C = \sum_{v: p(v) \geq 0} p(v)$
Extended Graph $G'$

Cut value
$= 13 + 3 + 2 + 3 + 4$
$= 13 + 3$
$+ C-4-8-10-11-12-14$
Project Selection

• **Claim** Any s-t-cut \((S, \overline{S})\) in \(G'\) such that \(A = S - \{s\}\) satisfies
  
  • precedence constraints and
  • has capacity \(c(S, T) = C - \sum_{v \in A} p(v) = C - \text{Profit}(A)\)

• **Corollary** A minimum cut \((S, \overline{S})\) in \(G'\) yields an optimal solution \(A = S - \{s\}\) to the project selection problem

• **Algorithm** Compute maximum flow \(f\) in \(G'\), find the set \(S\) of nodes reachable from \(s\) in \(G'_{f}\) and return \(S - \{s\}\)
Proof of Claim

- $A = S - \{s\}$ satisfies precedence constraints
  - No edge of $G$ crosses forward out of $A$ since those edges have capacity $+\infty$
  - Only forward edges cut are of the form $(v,t)$ for $v \in A$ or $(s,v)$ for $v \notin A$

The $(v,t)$ edges for $v \in A$ contribute

$$\sum_{v \in A: p(v) < 0} -p(v) = -\sum_{v \in A: p(v) < 0} p(v)$$

The $(s,v)$ edges for $v \notin A$ contribute

$$\sum_{v \notin A: p(v) \geq 0} p(v) = C - \sum_{v \in A: p(v) \geq 0} p(v)$$

Therefore the total capacity of the cut is

$$c(S,T) = C - \sum_{v \in A} p(v) = C - \text{Profit}(A)$$