Maximum Flow

Chapter 26

Flow Graph

• A common scenario is to use a graph to represent a “flow network” and use it to answer questions about material flows
• Flow is the rate that material moves through the network
• Each directed edge is a conduit for the material with some stated capacity
• Vertices are connection points but do not collect material
  – Flow into a vertex must equal the flow leaving the vertex, flow conservation
Sample Networks

<table>
<thead>
<tr>
<th>Network</th>
<th>Nodes</th>
<th>Arcs</th>
<th>Flow</th>
</tr>
</thead>
<tbody>
<tr>
<td>communication</td>
<td>telephone exchanges, computers, satellites</td>
<td>cables, fiber optics, microwave relays</td>
<td>voice, video, packets</td>
</tr>
<tr>
<td>circuits</td>
<td>gates, registers, processors</td>
<td>wires</td>
<td>current</td>
</tr>
<tr>
<td>mechanical</td>
<td>joints</td>
<td>rods, beams, springs</td>
<td>heat, energy</td>
</tr>
<tr>
<td>hydraulic</td>
<td>reservoirs, pumping stations, lakes</td>
<td>pipelines</td>
<td>fluid, oil</td>
</tr>
<tr>
<td>financial</td>
<td>stocks, companies</td>
<td>transactions</td>
<td>money</td>
</tr>
<tr>
<td>transportation</td>
<td>airports, rail yards, street intersections</td>
<td>highways, railbeds, airway routes</td>
<td>freight, vehicles, passengers</td>
</tr>
<tr>
<td>chemical</td>
<td>sites</td>
<td>bonds</td>
<td>energy</td>
</tr>
</tbody>
</table>

Flow Concepts

- Source vertex s
  - where material is produced
- Sink vertex t
  - where material is consumed
- For all other vertices – what goes in must go out
  - Flow conservation
- **Goal:** determine maximum rate of material flow from source to sink
Formal Max Flow Problem

– Graph \( G=(V,E) \) – a flow network
  - Directed, each edge has capacity \( c(u,v) \geq 0 \)
  - Two special vertices: source \( s \), and sink \( t \)
  - For any other vertex \( v \), there is a path \( s \rightarrow \ldots \rightarrow v \rightarrow \ldots \rightarrow t \)

– Flow – a function \( f: V \times V \rightarrow R \)
  - Capacity constraint: For all \( u, v \in V \): \( f(u,v) \leq c(u,v) \)
  - Skew symmetry: For all \( u, v \in V \): \( f(u,v) = -f(v,u) \)
  - Flow conservation: For all \( u \in V \) – \( \{s,t\} \):

\[
\sum_{v \in V} f(u,v) = f(u,V) = 0, \text{ or } \\
\sum_{v \in V} f(v,u) = f(V,u) = 0
\]

Cancellation of flows

• We would like to avoid two positive flows in opposite directions between the same pair of vertices
  – Such flows cancel (maybe partially) each other due to skew symmetry
Max Flow

• We want to find a flow of maximum value from the source to the sink
  – Denoted by $|f|$

Lucky Puck Distribution Network

Max Flow, $|f| = 19$
Or is it?
Best we can do?

Ford-Fulkerson method

• Contains several algorithms:
  – Residue networks
  – Augmenting paths
    • Find a path $p$ from $s$ to $t$ (augmenting path), such that there is some value $x > 0$, and for each edge $(u,v)$ in $p$ we can add $x$ units of flow
      – $f(u,v) + x \leq c(u,v)$

FORD-FULKERSON-METHOD($G, s, t$)
1. initialize flow $f$ to 0
2. while there exists an augmenting path $p$
3. do augment flow $f$ along $p$
4. return $f$
Residual Network

• To find augmenting path we can find any path in the **residual network**:
  - Residual capacities: \( c_f(u,v) = c(u,v) - f(u,v) \)
    - i.e. the actual capacity minus the net flow from \( u \) to \( v \)
    - Net flow may be negative
  - Residual network: \( G_f = (V,E_f) \), where
    \[ E_f = \{(u,v) \in V \times V : c_f(u,v) > 0\} \]
  - Observation – edges in \( E_f \) are either edges in \( E \) or their reversals: \(|E_f| \leq 2|E|\)

Residual Graph

• Compute the residual graph of the graph with the following flow:
Residual Capacity and Augmenting Path

• Finding an Augmenting Path
  – Find a path from $s$ to $t$ in the residual graph
  – The residual capacity of a path $p$ in $G_f$:
    
    $c_f(p) = \min\{c_f(u,v) : (u,v) \text{ is in } p\}$
    
    • i.e. find the minimum capacity along $p$
  – Doing augmentation: for all $(u,v)$ in $p$, we just add this $c_f(p)$ to $f(u,v)$ (and subtract it from $f(v,u)$)
  – Resulting flow is a valid flow with a larger value.

Residual network and augmenting path

Figure 26.3  (a) The flow network $G$ and flow $f$ of Figure 26.1(b). (b) The residual network $G_f$ with augmenting path $p$ shaded; its residual capacity is $c_f(p) = c_f(v_2, v_3) = 4$. (c) The flow in $G$ that results from augmenting along path $p$ by its residual capacity 4. (d) The residual network induced by the flow in (c).
The Ford-Fulkerson method

\textbf{Ford-Fulkerson}(G,s,t)
\begin{algorithmic}
  \STATE \textbf{for} each edge \((u,v)\) in \(G\) \textbf{do}
  \STATE \hspace{1em} \(f(u,v) \leftarrow f(v,u) \leftarrow 0\)
  \STATE \textbf{while} there exists a path \(p\) from \(s\) to \(t\) in residual network \(G_f\) \textbf{do}
  \STATE \hspace{1em} \(c_f = \min\{c_{rf}(u,v) : (u,v) \text{ is in } p\}\)
  \STATE \hspace{1em} \textbf{for} each edge \((u,v)\) in \(p\) \textbf{do}
  \STATE \hspace{2em} \(f(u,v) \leftarrow f(u,v) + c_f\)
  \STATE \hspace{1em} \(f(v,u) \leftarrow -f(u,v)\)
  \STATE \textbf{return} \(f\)
\end{algorithmic}

The algorithms based on this method differ in how they choose \(p\) in step 3. If chosen poorly the algorithm might not terminate.

\section*{Execution of Ford-Fulkerson (1)}

\begin{figure}[h]
\centering
\begin{subfigure}{0.45\textwidth}
\includegraphics[width=\textwidth]{residual_graph_1}
\caption{Residual Graph}
\end{subfigure} \quad
\begin{subfigure}{0.45\textwidth}
\includegraphics[width=\textwidth]{augmented_flow_1}
\caption{Augmented Flow}
\end{subfigure}
\end{figure}
Execution of Ford-Fulkerson (2)

Left Side = Residual Graph  Right Side = Augmented Flow

Cuts

• Does the method find the minimum flow?
  – Yes, if we get to the point where the residual graph has no path from \( s \) to \( t \)
  – A cut is a partition of \( V \) into \( S \) and \( T = V - S \), such that \( s \in S \) and \( t \in T \)
  – The net flow \( f(S,T) \) through the cut is the sum of flows \( f(u,v) \), where \( s \in S \) and \( t \in T \)
    • Includes negative flows back from \( T \) to \( S \)
  – The capacity \( c(S,T) \) of the cut is the sum of capacities \( c(u,v) \), where \( s \in S \) and \( t \in T \)
    • The sum of positive capacities
  – Minimum cut – a cut with the smallest capacity of all cuts.
    \( |f| = f(S,T) \) i.e. the value of a max flow is equal to the capacity of a min cut.

Cut capacity = 24  Min Cut capacity = 21
Max Flow / Min Cut Theorem

1. Since $|f| \leq c(S,T)$ for all cuts of $(S,T)$ then if $|f| = c(S,T)$ then $c(S,T)$ must be the min cut of $G$
2. This implies that $f$ is a maximum flow of $G$
3. This implies that the residual network $G_f$ contains no augmenting paths.
   - If there were augmenting paths this would contradict that we found the maximum flow of $G$
   - $1 \rightarrow 2 \rightarrow 3 \rightarrow 1$ … and from $2 \rightarrow 3$ we have that the Ford Fulkerson method finds the maximum flow if the residual graph has no augmenting paths.

Worst Case Running Time

- Assuming integer flow
- Each augmentation increases the value of the flow by some positive amount.
- Augmentation can be done in $O(E)$.
- Total worst-case running time $O(E|f^*|)$, where $f^*$ is the max-flow found by the algorithm.
- Example of worst case:
Edmonds Karp

• Take **shortest path** (in terms of number of edges) as an augmenting path – Edmonds-Karp algorithm
  – How do we find such a shortest path?
  – Running time $O(VE^2)$, because the number of augmentations is $O(VE)$
  – Skipping the proof here

  – Even better method: push-relabel, $O(V^2E)$ runtime

Multiple Sources or Sinks

• What if you have a problem with more than one source and more than one sink?
• Modify the graph to create a single supersource and supersink
Application – Bipartite Matching

• Example – given a community with $n$ men and $m$ women
• Assume we have a way to determine which couples (man/woman) are compatible for marriage
  – E.g. (Joe, Susan) or (Fred, Susan) but not (Frank, Susan)
• Problem: Maximize the number of marriages
  – No polygamy allowed
  – Can solve this problem by creating a flow network out of a bipartite graph

Bipartite Graph

• A bipartite graph is an undirected graph $G=(V,E)$ in which $V$ can be partitioned into two sets $V_1$ and $V_2$ such that $(u,v) \in E$ implies either $u \in V_1$ and $v \in V_{12}$ or vice versa.
• That is, all edges go between the two sets $V_1$ and $V_2$ and not within $V_1$ and $V_2$. 
Model for Matching Problem

- Men on leftmost set, women on rightmost set, edges if they are compatible

Solution Using Max Flow

- Add a supersource, supersink, make each undirected edge directed with a flow of 1

Since the input is 1, flow conservation prevents multiple matchings