Final Project

Siddharth Pant
CSCI 5454

Graduate Student

Computer Science

University of Colorado Boulder

April 2016

Introduction

Published by L.R. Ford Jr and D.R.Fulkerson, the Ford-Fulkerson algorithm is used to compute the maximum flow in a network.

In graph theory, a directed graph where each edge has a capacity and each edge receives a flow is called a flow network. The edge of a network restricts the flow across that edge. Unless a node is a source node which has only flow going outwards or a sink node which has only incoming flow, for every other node the total flow that enters a node must be equal to flow that leaves the node.

Definition:

Graph $G = (V, E)$ has edge $(u, v) \in E$ has a non-negative capacity.
Capacity between 2 edges is denoted as $c(u, v)$ and is always positive.
Source vertex is s and sink node is t.
A flow in a flow network is a real function $f : V$ x $V \rightarrow R$ having the below mentioned properties:

i) $f(u, v) \leq c(u, v) \Rightarrow$ the flow of an edge cannot exceed that edge's capacity

ii) $f(u, v) = -f(v, u) \Rightarrow$ the flow from one node to another should be the reverse of the flow from the other edge to this edge

iii) $\sum_{w \in V} f(u, w) = 0 \Rightarrow$ the net flow to a node is 0, unless that edge is a source or sink node.

iv) $\sum_{(s,u) \in E} f(s, u) = \sum_{(v,t) \in E} f(v, t) \Rightarrow$ the total flow leaving from s must be equal to the total flow arriving at t. The Residual capacity of an edge is the difference between the capacity of that edge and the flow through that edge i.e., $c_f(u, v) = c(u, v) - f(u, v)$.

Hence, a residual network comprises of edges marked with the residual capacities. $G_f(V, E_f))$ indicates how much capacity is left.

Applications

1) Image Processing and Computer Vision are examples of fields where the Ford-Fulkerson Algorithm is particularly useful. Some of the areas where it can be used include optical flow estimation, stereo correspondence, image segmentation etc. These problems can be transformed into a max-flow min-cut problem before being solved by this method.

Consider Image segmentation where a digitalized image is divided into segments or sets of pixels.
There are n pixels where each pixel i is given a foreground value f_i or a background value b_i. If adjascent pixels i, j have different assignments, then there is a penalty p_{ij}. The Problem is of assigning pixels in such a way that the difference between the sum of their values and penalties is maximized.

The set of pixels assigned to foreground is P and to the background is Q. Therefore,
$max(g) = \sum_{i \in P} f_i + \sum_{i \in Q} b_i - \sum_{i \in P, j \in Q \cup j \in P, i \in Q} p_{i,j}$

This can be thought of as a max-flow problem where source node is connected to all pixels with capacity f_i and sink to all pixels with capacity b_i. Edges (i, j) and (j, i) can be added between two adjascent pixels with capacity $p_{i,j}$. The $s - t$ can represent pixels assigned to P and Q respectively.

2) Another application of maximum flow is in computing the max-size matching in a bipartite graph.
A matching is a subgraph where every vertex has a degree of atmost 1 OR a collection of edges such that no two edges share a vertex.

Let G be a bipartite graph with vertex set $U \cap V$ where every edge connects some vertex in U to some vertex in V. Now add two more vertices 's' and 't' such that there is an edge from s to every edge in U and there is any edge from each vertex in V to t. Also each edge between U and V is directed from U to V. Let this new graph be G'.

For each edge uv push one unit of flow along $s \to u \to v \to t$. Consider any flow from $s \to t$. Each edge has unit capacity. Since each edge capacity is an integer, the Ford-Fulkerson algorithm assigns an integer flow to every edge. Hence, flow along each edge is either 1 or 0. If it is 0 then that edge is avoided. Also, each vertex in U can accept one unit of flow and each vertex in V can have unit of flow flowing out.

Thus, the value of the maximum flow in G' is the same as maximum matching in G and actual max-flow can be used for maximum matching. The maximum flow has a value of at most min$|U|, |V|$=$O(Vertices)$.

Therefore, The ford-Fulkerson algorithm runs in $O(VE)$.
Note: This 'V' denotes vertices and is different from the V used in the example.

INSERT IMAGE!!!!!!!!!!!!!!!!!!!!!!!!!!!