Application of Max-flow min-cut theorem for Computer Vision

Hariprasad.P.S (EE11B064), S.R.Manikandasriram (EE11B127)

***Abstract*—This paper reviews the Max-flow min-cut theorem based graph cut algorithms particularly the Ford-Fulkerson algorithm and its applications in Computer Vision and other fields.**

I. Introduction

Graph cut is a well studied concept in Graph Theory. One of the major applications of Graph cuts is in the field of Computer Vision. Many fundamental problems in Computer Vision can be reformulated as a Graph Cut problem and in particular, a Max-flow min-cut problem. In this paper, we will study the Ford-Fulkerson algorithm which is based on Max-flow min-cut theorem. A mathematical formulation of the Max-flow min-cut problem as a general graph cut problem will be presented. The implementation of the Ford-Fulkerson algorithm will be explained in detail and supported by pseudo-code. We will take a brief overview of the various applications of this problem in the field of Computer Vision which will justify the importance of Ford-Fulkerson Algorithm. Finally, a brief analysis of the time complexity of the algorithm will be presented.

According to graph theory, a graph cut is the grouping of nodes in a connected component into two disjoint subsets and the weight of the cut is defined to be equal to the sum of the weights of edges that are present between the disjoint subsets. If we consider the graph as a flow network, then an $s-t$ cut is defined as a graph cut which requires the $source$ and $sink$ nodes to be in different subsets. The weight of an $s-t$ cut is called as the $capacity$ of the cut. For a given graph containing a $source$ and a $sink$ node, there are many possible $s-t$ cuts. Thus a **minimum cut** is defined as that $s-t$ cut whose capacity is less than or equal to every other $s-t$ cut for the given graph.

II. Max-flow min-cut theorem

Given a flow network, the Max-flow min-cut theorem states that the maximum flow between the source and sink nodes equals the minimum capacity over all $s-t$ cuts. While there can be many $s-t$ cuts with the same capacity, consequently there can be multiple ways to assign flows in the network while achieving the same maximum flow.

Consider a network $N=(V,E)$ with node s as source and node t as sink. Let $c(u,v)$ denote the real valued capacity of the edge $(u,v)\in E$ and similarly, let $f(u,v)$ denote the real valued flow between the nodes u and v. The network N must satisfy the following:

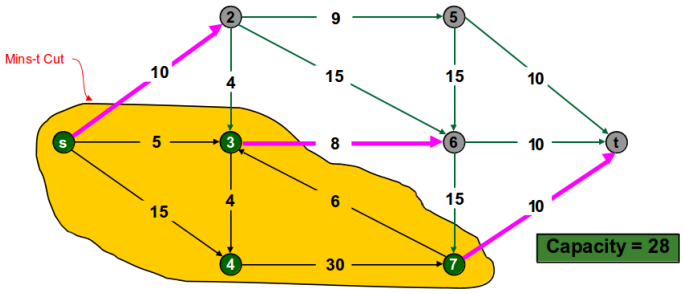

Fig. 1. An example of a flow network showing a min s-t cut. source: Princeton Lecture notes

$$f(u,v)\leq c(u,v)\forall u,v\in V$$

$$f(u,v)=-f(v,u)\forall u,v\in V$$

$$\sum_{v\in V} f(u,v)=0\forall u\in V-\{s,t\}$$

The value of the flow through this network for a given $s-t$ cut is given by:

$$|f|=\sum_{v\in V} f(s,v)=\sum_{v\in V} f(v,t)$$

In order to find the maximal flow, we can apply the Ford-Fulkerson algorithm to determine the minimum $s-t$ cut which produces that maximum flow.

III. Ford-Fulkerson Algorithm

Ford-Fulkerson algorithm can be briefly described as continuously sending flow along paths through the flow network from source to sink until there exists no path which has all edges with non-zero capacity remaining.

The structure of the algorithm is as follows:

- Find augmenting path P in the residual network N_f.
- Find bottleneck capacity of P.
- Augment flow through P.
- Repeat until no augmenting path exists.

Residual Graph

After one iteration of assigning flow to edges constituting a valid path P from s to t, the residual graph N_f is formed by re-computing the edge weights to denote the remaining capacities which must be strictly greater than zero. All residual edges with

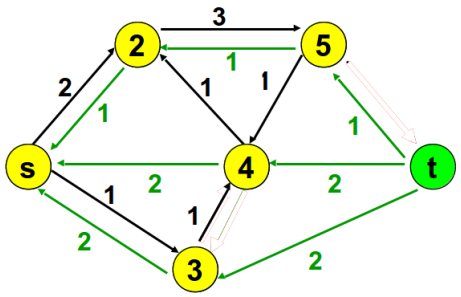

Fig. 2. A flow network implementing Flow Fulkerson algorithm to compute the min s-t cut

weight equal to zero are removed in the residual graph.

Augmenting Path

Given a residual graph N_f, an augmenting path P is a valid path from s to t which can contribute towards increasing the overall flow from s to t.

Pseudo Code

Inputs: Given a Network $G = (V, E)$ with flow capacity c, a source node s, and a sink node t

Output: Compute a flow f from s to t of maximum value

We define the residual network $G_f(V, E_f)$ to be the network with capacity $c_f(u, v) = c(u, v) - f(u, v)$ and no flow

1. $f(u, v) \leftarrow 0$ for all edges (u, v)

2. While there is a path p from s to t in G_f, such that $c_f(u, v) > 0$ for all edges $(u, v) \in p$:

1. Find $c_f(p)$ = min$\{c_f(u, v) : (u, v) \in p\}$

2. For each edge $(u, v) \in p$

1. $f(u, v) \leftarrow f(u, v) + c_f(p)$

2. $f(v, u) \leftarrow f(v, u) - c_f(p)$

Complexity Analysis

When the capacities are integers, the runtime of Ford-Fulkerson is bounded by $O(Ef)$, where E is the number of edges in the graph and f is the maximum flow in the graph. However if the graph has irrational values, the flow might not converge towards the maximum value and the algorithm runs forever

IV. Real World Applications

There are numerous real world applications of maximum flow and minimum cut. A common question about any network is "what is the maximum flow rate between some node in that network and any other node?". For example, traffic engineers may want to know the maximum flow rate of vehicles from the downtown car park to the freeway because this will influence their decisions on whether to widen the roadways. Another example might be the maximum number of simultaneous telephone calls between two cities via the various land-lines, satellites, and microwave towers operated by a telephone company. Airline scheduling and Baseball elimination are other examples which use this algorithm.

The Ford-Fulkerson algorithm particularly has a lot of applications in Image Processing and Computer Vision. Some of them are image segmentation, optical flow estimation, stereo correspondence, etc. where the given problem is transformed into a maximum flow minimum cut problem and then solved using the Ford-Fulkerson algorithm.

In the image segmentation problem, there are n pixels, where each pixel i can be assigned a foreground value f_i or a background value b_i. There is a penalty of p_{ij} if pixels i, j are adjacent and have different assignments. The problem is to assign pixels to background or foreground such that the sum of their values minus the penalties is maximum.

Let P be the set of pixels assigned to foreground and Q be the set of points assigned to background, then the problem can be formulated as,

$$\max(g) = \sum_{i \in P} f_i + \sum_{i \in Q} b_i - \sum_{i \in P, j \in Q \cup j \in P, i \in Q} p_{ij} \quad (1)$$

This maximization problem can be formulated as a minimization problem instead, that is,

$$\min(g') = \sum_{i \in P} f_i + \sum_{i \in Q} b_i + \sum_{i \in P, j \in Q \cup j \in P, i \in Q} p_{ij} \quad (2)$$

The above equations can be formulated as a maximum flow minimum cut problem by constructing a network where the source is connected to all the pixels with capacity f_i, and the sink is connected by all the pixels with capacity b_i. Two edges (i, j) and (j, i) with p_{ij} capacity are added between two adjacent pixels. The $s - t$ cut-set then represents the pixels assigned to the foreground in P and pixels assigned to background in Q.

References

[1] Ford, L. R.; Fulkerson, D. R. (1956). "Maximal flow through a network". Canadian Journal of Mathematics 8: 399–404, MR 0079251

[2] Gass, Saul I.; Assad, Arjang (2005), "1954 Max-flow min-cut theorem", An annotated timeline of operations research: an informal history, International series in operations research and management science 75, Springer-Verlag, p. 96, ISBN 978-1-4020-8112-5

[3] George T. Heineman, Gary Pollice, and Stanley Selkow (2008). "Chapter 8:Network Flow Algorithms". Algorithms in a Nutshell. Oreilly Media. pp. 226250. ISBN 978-0-596-51624-6

[4] http://www.cs.princeton.edu/courses/archive/spr04/cos226/lectures/maxflow.4up.pdf