Week 10

Graph Theory and Applications*

10.1 Pseudo Boolean Functions

Pseudo Boolean Function are real valued function of form

$$f : B^n \to R$$

where, B: Boolean domain and n : Arity of function that is non-negative integer.

10.1.1 Minimization of Pseudo Boolean Functions

Method 1: Brute Force Method In this we calculate f(x) for all possible ranges of value that x can take in exponential time complexity. Truth table size is 2^n and increase search cost.
Method 2: Graph Cut Algorithm A graph $G = (V, E)$ is partitioned into disjoint set of two A and B such that $A \cup B = V$ and $A \cap B = 0$ obtained by removing edge connecting A and B.
Degree of dissimilarity Total weight of edges that have been removed.

$$Cut(A, B) = \sum w(u, v) \text{ where } u \in A \text{ and } v \in B$$

This algorithm can be efficiently applied to low level computer vision tasks such as image smoothing and segmentation etc and problems that can be framed as energy minimization.

*Lecturer: Dr. Anand Mishra. Scribe: Sonali.

10.1.2 How to Find Min Cut?

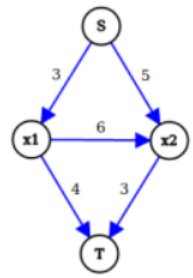

Figure 10.1: Example graph

1. Initialization
   (a) Create a node corresponding to each variable.
   (b) Create two nodes as Source and Sink.
   (c) Assignment of Edges:
      i. Non complementary Variables: Edges will enter nodes with weight.
      ii. Complementary Variables: Edges will move out of the node with assigned weight.
2. Graph Representation obtained after flowing maximum flow of graph.

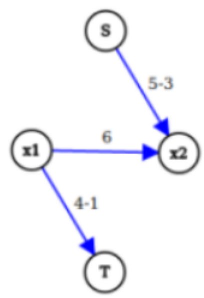

Figure 10.2: Example graph

From path S-x1-T : 3 is exhausted. From path S-x2-T : 3 is exhausted.

3. Initialize set S = x2 and T = x1 on the obtained residual graph.
   Variables on source and target side are initialized as 0 and 1 i.e. x2=0 and x1=1. In f(x), substitute values of the above variables.
   f(x)=3+0+0+3+0=6 that is the minimum value of f(x).

10.1.3 Properties

1. Directed Edge from node i to j has non-negative capacity in the graph.
2. For non-existent arc/edge cap(i,j)=0.

10.1.4 Applications of Graph Cut

Image Segmentation

Figure 10.3: Example of Image Segmentation

Bird (foreground) is segmented from background using Graph Cut.Cost c_i comes from probability estimate i.e, belonging to foreground or background.
This application can be formulated as an Energy minimization problem.This helps in localizing boundaries and objects.

$$E(x) = \sum_i c_i x_i + \sum_{i,j} c_{i,j} x_i (1x_j)$$

Objective is to minimize the above equation or to find global minima of the energy i.e. $x = argmin_x E(x)$

where, x_i : 0 or 1 depending on the background or foreground respectively.
C : Cost associated with pixel

For example, in the 3*3 image shown in Figure 10.4 our goal is to perform image segmentation where pixel closer 255 will be associated with 1 as cost of that particular pixel and cost 0 for pixel closer to 0.

$$\begin{bmatrix} 250 & 240 & 255 \\ 5 & 230 & 9 \\ 6 & 235 & 10 \end{bmatrix} \rightarrow \begin{bmatrix} 1 & 1 & 1 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \end{bmatrix}$$

Figure 10.4: Image matrix

Steps are as follows-

1. Cost associated = For assigning Pixel value 1 : $(255p(x))x_i$ Pixel value 0 : $p(x_i)x_i$

   Substituting values, we get ,

$$f(x) = \sum_{i=1}^{9}(255 - p(x_i))x_i + p(x_i)\overline{x_i}$$

2. Cost also depends on neighboring pixels for consistency. So, $x_i\overline{x_j}$ have some cost if these neighbors have different pixel values.

   So, the equation can be optimized as

$$f(x) = \sum_{i=1}^{n}(255 - p(x))x_i + p\left(x_i\right)\overline{x_i} + \sum_{i,j \in N} c_{ij}\overline{x_i}x_j + c_{ji}\overline{x_i}x_j$$

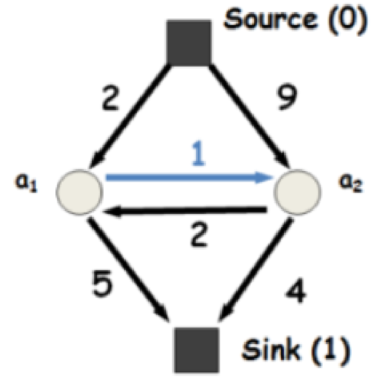

Figure 10.5: Graph

Summarizing by example, this application helps in image segmentation using the above algorithm by construction of graph s_t, any cut corresponds to an assignment of x and cost of cut = E(x).

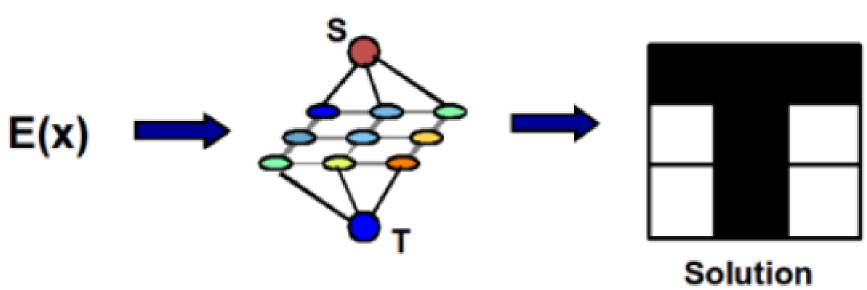

Figure 10.6: Image Segmentation Example

10.1.5 What energy functions can be minimized?

Generally energy functions are NP hard to minimize; we can have approximate solutions for such functions. But, we also have some easy energy functions i.e. sub modular functions can be solved in polynomial time and are graph representable.

Sub Modular Function

It is a function f that is defined over a set of boolean variables $x = x_1, x_2, ..., x_n$.
Properties
By above definition,

1. One variable boolean function is sub modular.

2. Two variable boolean function satisfies sub modular property if

$$f(0,0) + f(1,1) \leq f(0,1) + f(1,0)$$

Generalizing this, a function is sub modular if all its projection to two variables are sub-modular.

10.2 Flow networks

It is digraph G(V,E) with distinguished two vertex sources and a sink t and each edge has a defined capacity c(u,v). For non-existent edges, c(u,v)=0. No edge enter source and no edge comes out of sink.

Some applications that can be formulated as flow networks are liquid and current flowing through pipes and electrical networks etc.

Flow: It is a real valued function $f : V \times V \to R$ obeying flow conservation like Kirchoff's current law. This flow is viewed as a rate, not a quantity.

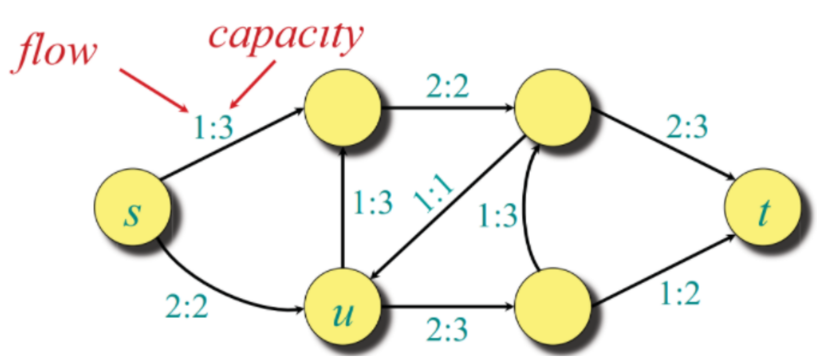

Figure 10.7: Example of Flow Network

For node u :
Flow in : 2 + 1 = 3
Flow out : 1 + 2 = 3

Total net flow = Total positive flow leaving vertex v - Total positive flow entering that vertex v.

10.2.1 Assumptions of Flow network

1. No self loop edges exist.

2. If there exist edge $(u, v) \in E$, then $(v, u) \notin E$.

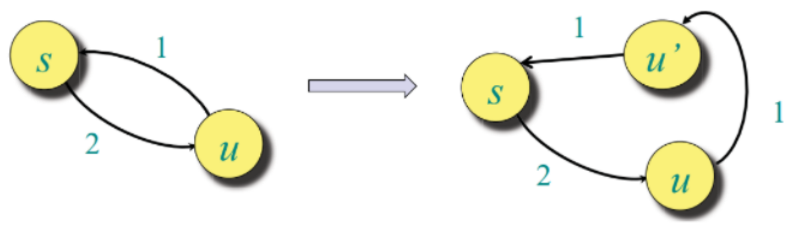

Figure 10.8: Flow network

10.2.2 Flow Constraints

1. For each edge $0 \leq f(e) \leq c$ i.e. Capacity constraint

2. For all $u \in V - \{s, t\}$, $\sum_{v \in V} f(u, v) = 0$ i.e. Flow conservation flow in equals flow out.

3. For all $u, v \in V$, $f(u, v) = -f(v, u)$ i.e. Skew symmetry. This makes it easy to add two flows.

4. For all nodes $\sum_{eintov} f(e) = \sum_{eleavingv} f(e)$ except sink and source i.e. balance constraints.

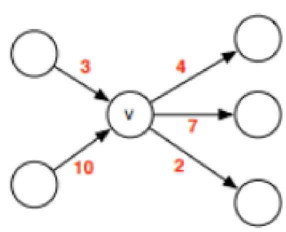

Figure 10.9: Example demonstrating Flow Constraints

Goal is to maximize total flow into sink t and satisfy all above constraints.

10.2.3 Value of Flow

It is an addition of all flows that come out of s i.e. s is able to send out. It can be denoted as $f^{out}(s)$.

$$|f| = \sum f(s, v) = f(s, V)$$

Maximization of this quantity is beneficial. Maximum flow problem is all about finding the maximum value of f.

Flow into sink t: $|f| = f(s, V) = f(V, t) = 4$

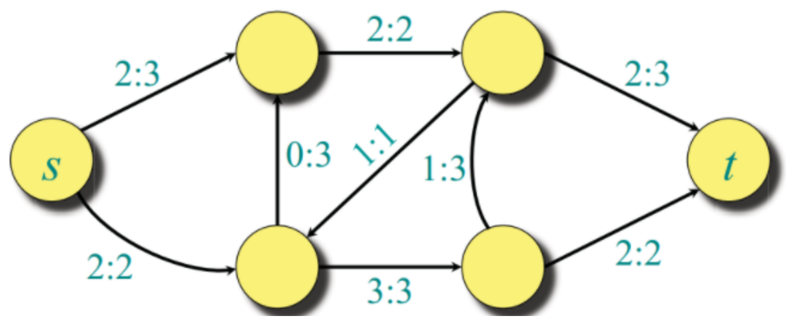

Figure 10.10: Example of Flow Network with capacity and flow

Properties associated with flow

1. f(X, X) = 0
2. f(X,Y) = – f(Y,X)
3. $f(X \cup Y, Z) = f(X, Z) + f(Y, Z)$ if $X \cap Y = \phi$.

Prove $|f| = f(V, t)$
Proof:
$|f| = f(s, V)$
$= f(V, V) - f(V{-}s, V)$
$= f(V, V{-}s)$
$= f(V, t) + f(V, V{-}s{-}t)$
$= f(V, t)$

10.2.4 Cuts

Cut(s,t) is the partition of vertices into A and B sets where $s \in A$ and $t \in B$. Edges going from A to B are edges belonging to cut.and if these edges are removed t gets disconnected from s.

$$\text{f(S,T) = flow across the cut}$$

Capacity of cut: It is the sum of capacity of edges flowing from A to B or in the cut.

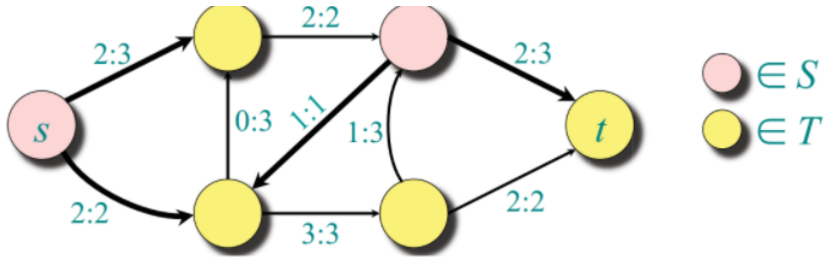

Figure 10.11: Example of Cut In Flow Network

For example,
Flow across the cut $f(S,T) = (2+2) + (-2+1-1+2) = 4$
Capacity of a cut $c(S,T) = (3+2) + (1+3) = 9$

10.2.5 Some Theorem and Lemma

1. **Prove:** $|f| = f(S,T)$ for flow f and any cut (S,T).
   Proof: $f(S,T) = f(S,V) - f(S,S)$
   $= f(S,V)$
   $= f(s,V) + f(S-s,V)$
   $= f(s,V) = |f|$

2. **Prove:** Value of any flow is bounded by the capacity of any cut.
   Proof: $= f(S,T)$
   $= \sum_{u\in S}\sum_{v\in T} f(u,v) \leq \sum_{u\in S}\sum_{v\in T} c(u,v)$
   $= c(S,T)$

10.2.6 Residual and Augmented Flow Network

Residual graph $G_f(V,E_f)$ depends on f(flow) containing the same nodes as G. It signifies how much more flow is allowed in the network graph. G_r may have edges that may not be present in the original graph

Residual capacity=Original edge's capacity - Flow on that edge.

If there is Graph G, there is an edge from s to v1 with capacity=16 and flow=11. So we will have two edges in G_r . Forward edge with residual capacity=5 from s to v1 and backward edge from v1 to s with residual capacity=11.Forward edge signifies additional flow 5 units from s to v1.

Figure 10.12 and 10.13 shows examples of flow network.

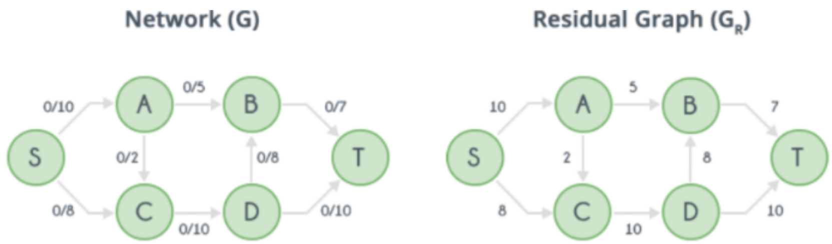

Figure 10.12: Example of Residual Graph

2.

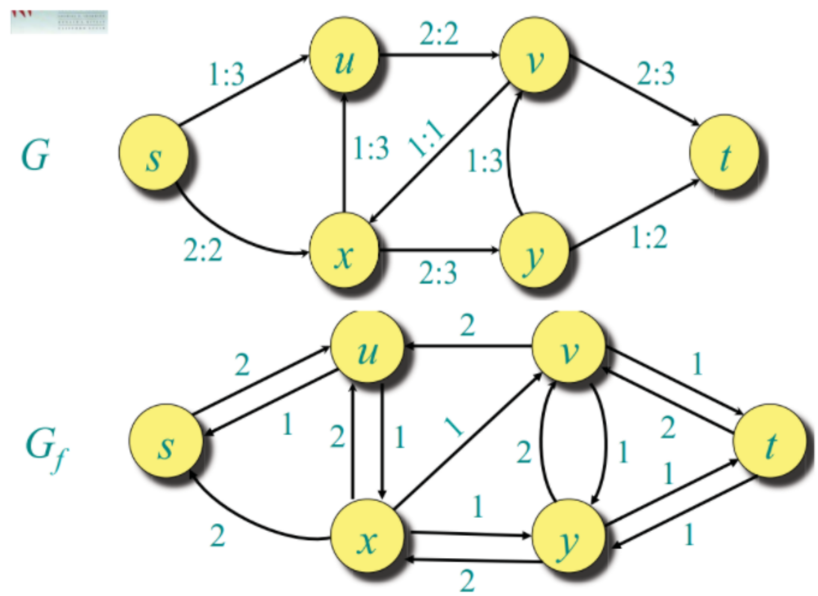

Figure 10.13: Example of Residual Graph

This contains strictly positive residual capacities. $c_f(u,v) = c(u,v) - f(u,v) > 0$. Edges in Residual network $E_f : E_f = (u,v) \in V \times V : C_f(u,v) > 0$ admit more flow. Also, $|E_f| \leq 2|E|$.

10.2.7 Augmenting Paths

It is a simple path from s to t in G_r.This helps in increasing flow on certain edges that increases the overall flow. It is not necessarily true that it will only increase flow . The flow value can be increased $c_f(p) = min c_f(u,v)$ by augmenting path p.

Augmenting Paths in above Figure:

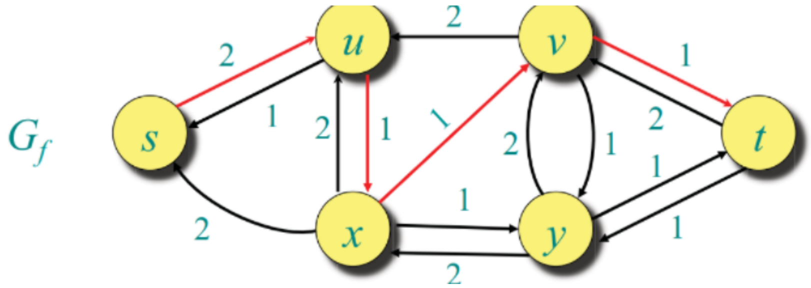

Figure 10.14: Example of Augmenting Paths

This helps in finding maximum flow using Ford-Fulkerson algorithm.

10.3 Max-flow and Min-Cut Theorem

Maximum Flow: The maximum amount of flow passing from s to t = total edge weight in a minimum cut.

This theorem states that:

1. —f— = c(S,T) for some cut (S,T)
2. f have no augmenting paths
3. f is a maximum flow

10.4 Ford-Fulkerson max-flow algorithm

For finding maximum flow this algorithm uses augmenting paths.

$$f[u,v] \leftarrow \forall (u,v) \in V$$

while an augmenting path p in G with respect to f exists, then do augment f by $c_f(p)$. This guarantees maximum flow because of the max-flow min-cut theorem explained above.

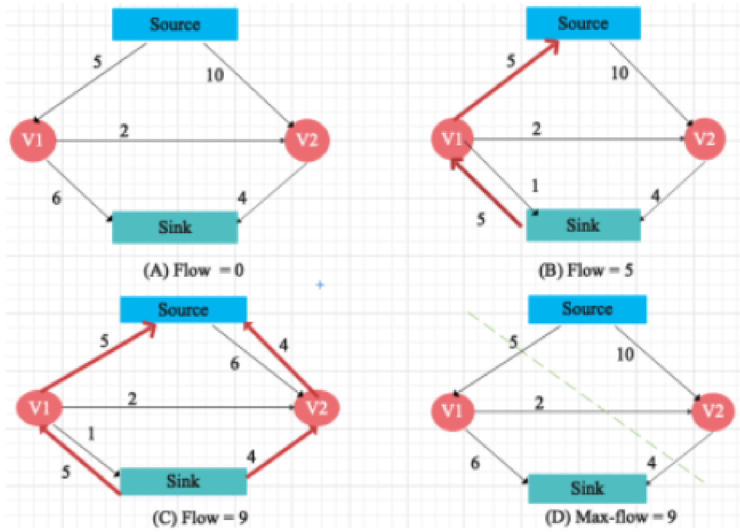

Figure 10.15: Example of Max Flow

Time Complexity: Choose the shortest path with available capacity as our augmenting path in each iteration.Worst case, we may add 1 unit flow in every iteration so it becomes O(max-flow * E).

Efficiency: Using BFS for finding augmenting path we can obtain $O(|VkE|^2)$.

References

1. Avrim Lectures
2. Princeton Kleiberg-tardos Network Flow
3. Augmenting path
4. MIT course
5. CMU lectures Netflow