Week 10

GRAPH CUT

10.1 Last Week Recap

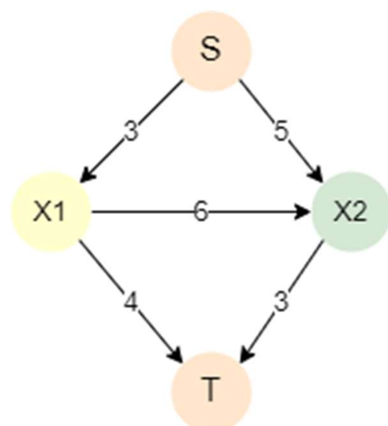
PSEUDO BOOLEAN FUNCTION:

f : {0,1} → R

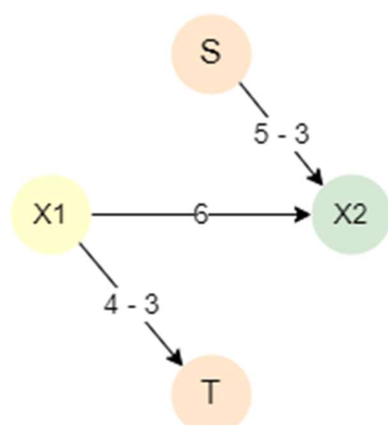
$$f(x) = 3x_1 + 4\overline{x_1} + 5x_2 + 3\overline{x_2} + 6\overline{x_1}\, x_2$$

where x1, x2 € {0, 1}
Goal : min f (x)
Bruit force method cannot be used here. So, we will use the graph cut method.
Construct a graph as follows:

- Every cut of the graph corresponds to some assignments to the variables.
- Min cut = minimum cost assignment.



Now the task is to find the min-cut of this graph:



So, from above figure we can say 6 is the min cut.

$X_2 = 0$ and $X_1 = 1$

So, the above equation will become:

f(x) = 3 + 0 + 0 + 3 + 0 = 6

10.2 Graph Cut

Motivation:

Consider a problem of image segmentation in energy minimization framework. We are considering the energy as object. This is very common issue in Computer Vision domain.

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

The goal is to find the global minima of energy function.

$$x^* = \mathrm{argmin}_x E(x)$$

Example: let say there is an image of scale of {0, 255}

250	240	255
5	230	9
6	235	10

→

1	1	1
0	1	0
0	1	0

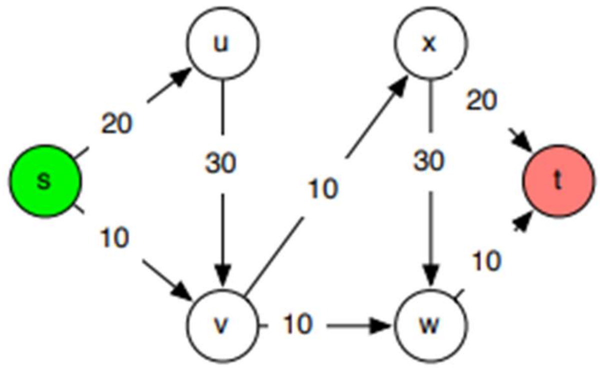
$X_1, X_2, \ldots X_9$

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

Above function is for single assignment/pixel.

10.3 FLOW NETWORKS

- Only for directed graphs. G (v, e) where v is vertex and e are edges.
- Two distinguished vertices where source is S and sink is T. The basic idea here is some flow coming from S with capacities of edges in order to make a way to T.
- Each edge e has a non-negative, integer capacity C_e
- A single source s ∈ V.
- A single sink t ∈ V.
- No edge enters the source, and no edge leaves the sink.



Assumptions Flow Network:

- Capacities are integers
- Every node has one edge adjacent to it.
- No edge enters the source, and no edge leaves the sink.
- No self-loop allowed.

Theorem: | f | = f (V, t)

Proof:

| f | = f (s, V) –> definition of cardinality

| f | = f (V, V) – f (V-s, V)

These are clearly disjoints.

f (V, V) = 0

So above equation will become:

| f | = f (V, V-s)

= f(V, t) + f(V, V-s-t)

= f(V, t) - f(V-s-t, V)

Using flow conservation, for any u that's neither s not t but in V, the sum has to be 0.

f(V-s-t, V) = 0

So, | f | = f(V, t)   hence proved.

This can be done using implicit summation notation.

10.4 CUTS in FLOW NETWORKS

A cut (S, T) of a flow network $G = (V, E)$ is a partition of V such that s ∈ S and t ∈ T.

If f is a flow on G, then the flow across the cut is $f(S, T)$.

The minimum cut of a weighted graph is defined as the sum of the weights of edges that divide the graph into two sets when they are removed.

Max-Flow Min-Cut Theorem:

This states that the lowest total of a cut is exactly equal to the maximum flow via any network from a given source to a given sink. The Ford-Fulkerson algorithm can be used to prove this theorem. The maximum flow of a network or graph is discovered using this approach.

- Run Ford-Fulkerson algorithm and consider the final residual graph.
- Find the set of vertices that are reachable from the source in the residual graph.
- All edges which are from a reachable vertex to non-reachable vertex are minimum cut edges.

Residual Network:

The residual network's intuition is that it allows us to cancel an already assigned flow. For example, if we've already allocated two units of flow from A to B, passing one unit of flow from B to A is understood as cancelling one unit of the original flow from A to B.

Actual network G(V, E)

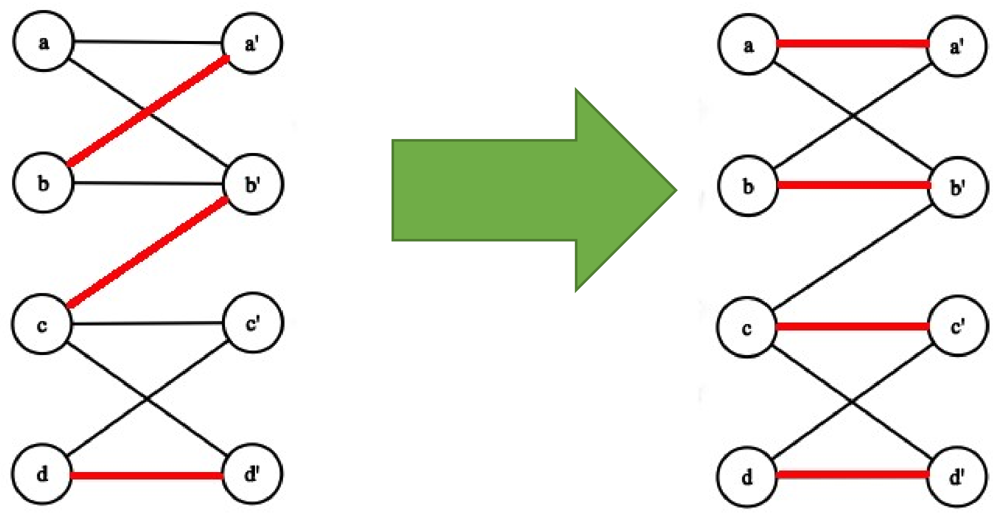
Residual Network $G_f(V, E_f)$, strictly positive residual capacities.

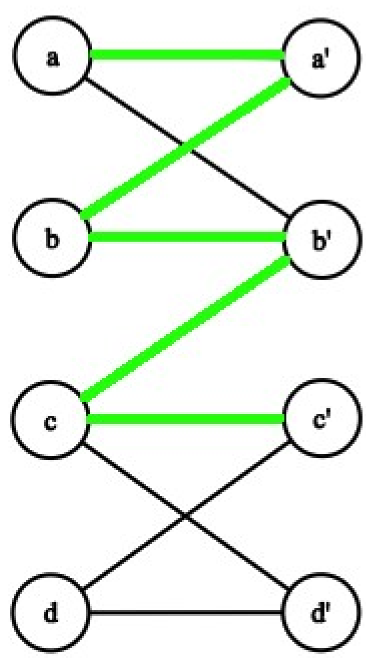
$C_f(u,v) = C(u,v) – f(u,v) > 0$

Augmenting Path:

A path in G which starts at an unmatched vertex and then contains, alternately, edges from E-M and M, is an alternating path with respect to M. We call an alternating path that ends in an unmatched vertex an augmenting path.

We can use an augmenting path P to turn a matching M into a larger matching by taking the symmetric difference of M with the edges of P. In other words, we remove edges from M which are in both P and M. We add/keep all other edges. We can best illustrate this by an example:





Here we used the augmenting path a-a'-b-b'-c-c' (shown in green) to augment the initial matching M to obtain a bigger matching M'. The edges ba' and cb' were removed from M and the edges aa', bb', and cc' were added to M. We also kept the edge dd' in M. We can prove that if we start with any matching and repeatedly augment it by augmenting paths, we'll always eventually obtain matching of optimal size, i.e., a maximum matching.