Graph Cut - I

Anand Mishra

Center for Visual Information Technology
IIIT Hyderabad
http://researchweb.iiit.ac.in/∼anand.mishra/

March 2017

Outline

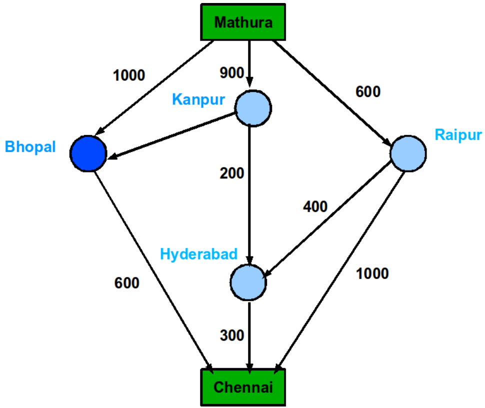
- Motivation behind Maximum Flow problem
- Maximum Flow Problem
- Max-Flow min-cut Theorem
- Ford Flukerson Algorithm
- Push relabel algorithm

Maximum Flow: Motivation

Imagine a oil refinery at Mathura producing oil, and it has a warehouse in Chennai. There are multiple path from the source (Mathura) to destination(Chennai) with each path having some capacity of fluid flow. Such graph are known as flow network.

Maximum Flow: Motivation

Imagine a oil refinery at Mathura producing oil and it has a warehouse in Chennai. There are multiple path from the source (Mathura) to destination(Chennai) with each path having some capacity of fluid flow. Such graph are known as flow network.



Maximum Flow: Motivation

Q: Which problems can be modelled as flow network?

- Liquids following through pipes
- Current through electrical networks
- Information through communication networks
- Vehicles through roads

Maximum Flow: Motivation

In a flow network

1. Each vertex other than source and sink is a conduit junction. They do not store/collect any material.(In context of electrical networks this is the well known rule: Kirchoff's Law)
2. Each edge can be though of a conduit for the material with a predefined capacity. For example: 100 gallon liquid per hour through a pipe or 10 amperes current through a wire.

What Questions can be answered through such flow networks

Q: What is the maximum amount of flow possible in a given flow network?

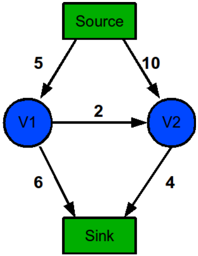
To answer this let us formally define some terms like:

- Flow network
- Flow
- Max-flow problem

Flow Networks

A flow network G = (V,E) is a directed graph in which each edge $(u, v) \in E$ has a non-negative capacity $c(u, v) \geq 0$

In a flow network we distinguish two vertices source(s) and sink(t).



Note: A flow network is always a connected graph thus in any flow network

$$|E| \geq |V| - 1$$

Flow

Let G = (V,E) be a flow network with a capacity function c, let s be the source of the network and t be the sink. Then a **flow** in G is defined as a real valued function $f : V \times V \longrightarrow \mathcal{R}$ that satisfies following three properties:

1. **Capacity constraint:**

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

Flow

Let G = (V,E) be a flow network with a capacity function c, let s be the source of the network and t be the sink. Then a **flow** in G is defined as a real valued function $f : V \times V \longrightarrow \mathcal{R}$ that satisfies following three properties:

1. **Capacity constraint:**
$$f(u,v) \leq c(u,v); \quad \forall u,v \in V$$

2. **Skew Symmetry:**
$$f(u,v) = -f(v,u); \quad \forall u,v \in V$$

Flow

Let G = (V,E) be a flow network with a capacity function c, let s be the source of the network and t be the sink. Then a **flow** in G is defined as a real valued function $f : V \times V \longrightarrow \mathcal{R}$ that satisfies following three properties:

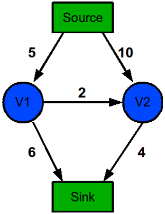
1. **Capacity constraint:**
$$f(u,v) \leq c(u,v); \quad \forall u,v \in V$$
2. **Skew Symmetry:**
$$f(u,v) = -f(v,u); \quad \forall u,v \in V$$
3. **Flow Conservation:**
$$\sum_{u \in V} f(u,v) = 0; \quad \forall u \in V - \{s,t\}$$
or equivalently
$$\sum_{v \in V} f(v,u) = 0; \quad \forall u \in V - \{s,t\}$$

Total Net flow

- Total positive flow entering a vertex v is defined by: $\sum_{u \in V, f(u,v)>0} f(u,v)$
- Similarly, We can define total positive flow leaving a vertex v as: $\sum_{v \in V, f(v,u)>0} f(v,u)$
- Total net flow of vertex v is defined as total positive flow leaving vertex v *minus* total positive flow entering that vertex.

Total Net flow

- Total positive flow entering a vertex v is defined by: $\sum_{u \in V, f(u,v)>0} f(u,v)$
- Similarly, We can define total positive flow leaving a vertex v as: $\sum_{v \in V, f(v,u)>0} f(v,u)$
- Total net flow of vertex v is defined as total positive flow leaving vertex v *minus* total positive flow entering that vertex.



Find total net flow of vertex v1

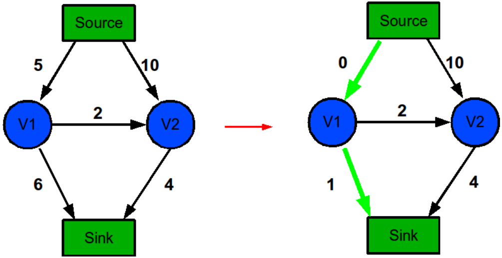
Maximum flow problem

Problem: Given a flow network G with source s and sink t we wish to find a flow of maximum value.

Residual Network: Given a flow network G = (V,E) and a flow f the residual network of G induced by flow f is $G_f = (V, E_f)$ where

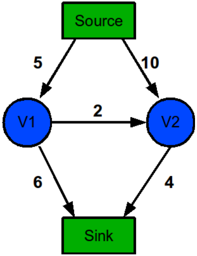
$$E_f = \{(u, v) \in V \times V : C_f(u, v) > 0\}$$

Residual Capacity: The amount of flow we can push from u to v before exceeding the capacity c(u,v) is the residual capacity of c(u,v).



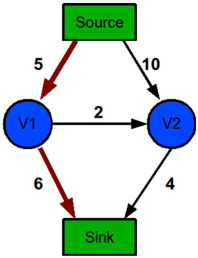
Augmenting path

Given a flow network G=(V,E) and a flow f an augmenting path p is a simple from s to t in the residual network.
Residual capacity of path: is the minimum residual capacity along the path.



Augmenting path

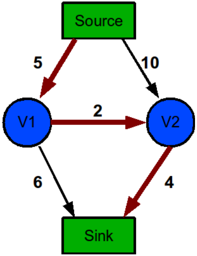
Given a flow network G=(V,E) and a flow f an augmenting path p is a simple from s to t in the residual network.
Residual capacity of path: is the minimum residual capacity along the path.



Residual capacity of path: 5

Augmenting path

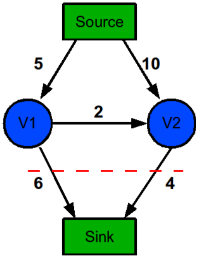
Given a flow network G=(V,E) and a flow f an augmenting path p is a simple from s to t in the residual network.
Residual capacity of path: is the minimum residual capacity along the path.



Residual capacity of path: 2

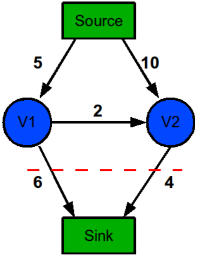
Cut

A cut C(S, T) of flow network G = (V,E) is a partition of set of vertices V into two sets disjoint sets S and T.
Capacity of a cut is the capacity of edges going from vertices belonging to S to vertices belonging to set T.



Cut

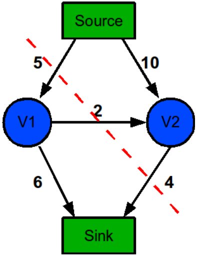
A cut C(S, T) of flow network G = (V,E) is a partition of set of vertices V into two sets disjoint sets S and T.
Capacity of a cut is the capacity of edges going from vertices belonging to S to vertices belonging to set T.



Capacity of this cut = 10

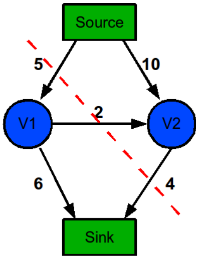
Cut

A cut C(S, T) of flow network G = (V,E) is a partition of set of vertices V into two sets disjoint sets S and T.
Capacity of a cut is the capacity of edges going from vertices belonging to S to vertices belonging to set T.



Cut

A cut C(S, T) of flow network G = (V,E) is a partition of set of vertices V into two sets disjoint sets S and T.
Capacity of a cut is the capacity of edges going from vertices belonging to S to vertices belonging to set T.

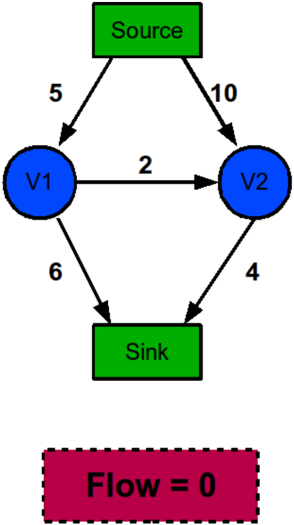


Capacity of this cut = 9

Max flow - min cut theorem

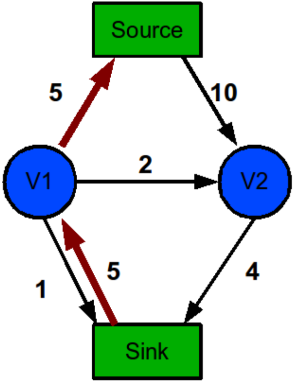
If f is a flow in flow network G = (V, E) with sources s and sink t, then the following conditions are equivalent:

1. f is a max flow in G
2. The residual network G_f contains no augmenting path
3. There exist a cut C(S, T) with capacity f.



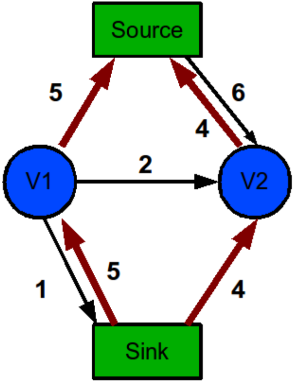
Find an augmenting path p and augment flow f against p

Ford-Flukerson Method



Find an augmenting path p and augment flow f against p

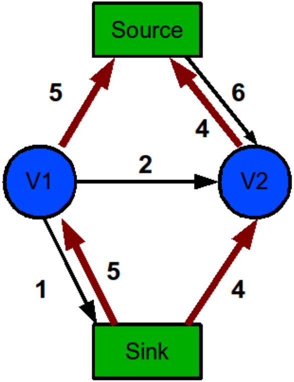
Flow = 5



Find an augmenting path p and augment flow f against p

Flow = 5 + 4

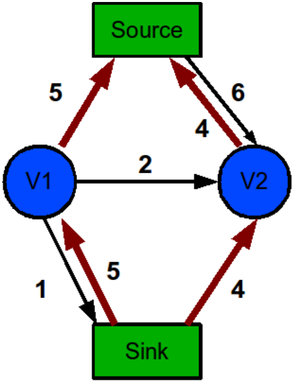
Ford-Flukerson Method

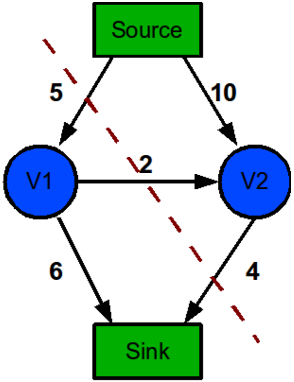


No more augmenting path exists

Flow = 5 + 4

Ford-Flukerson Method





Flow = 5 + 4

Max-flow = 9

Ford-Flukerson Method: Analysis

While finding augmenting path one has to traverse $O(|E|)$ each time. Thus if $max - flow = |f^*|$ then at worst case the complexity of the algorithm based on Ford-Flukerson: $O(|f^*||E|)$

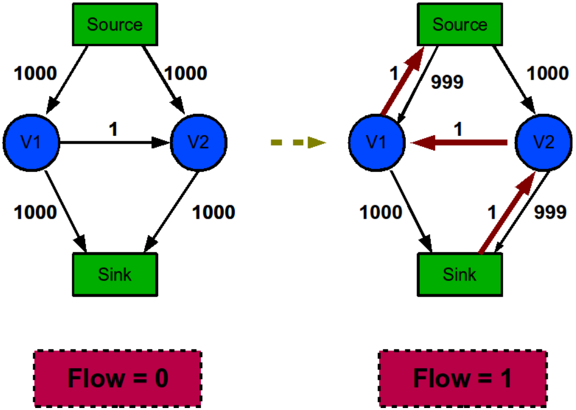
Ford-Flukerson Method: Analysis

While finding augmenting path one has to traverse $O(|E|)$ each time. Thus if $max - flow = |f^*|$ then at worst case the complexity of the algorithm based on Ford-Flukerson: $O(|f^*||E|)$

How worst this complexity can be?

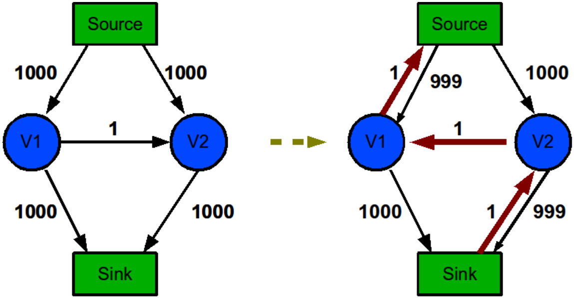
Ford-Flukerson Method: Analysis

Consider following example:



Ford-Flukerson Method: Analysis

Consider following example:



Flow = 0

Flow = 1

If I traverse in this way I have to traverse edges 2000 times

Efficient Ford-Flukerson Method: Edmonds-Karp Algorithm

Edmonds-Karp Algorithm finds the augmenting path with a breadth first search.
and has a complexity of $O(|V||E|^2)$

Push Relabel algorithms

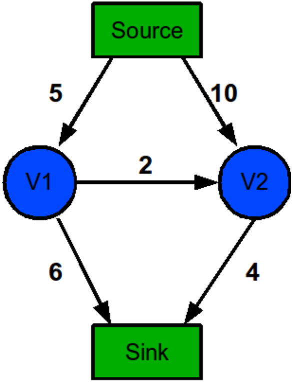
1. The intuition behind push relabel algorithms is the analogy of flow graph and pipes carrying fluids.
2. Each edge corresponds to pipe and each vertex corresponds pipe junctions
3. Each vertex(except source and sink) has an arbitrary large reservoir to accommodate excess flow e. Height of each vertex (except source and sink) h is zero initially and increases with the progress of algorithm.
4. The height of source and sink are fixed to $|V|$ and 0 respectively.
5. Flow can be pushed only downhill.

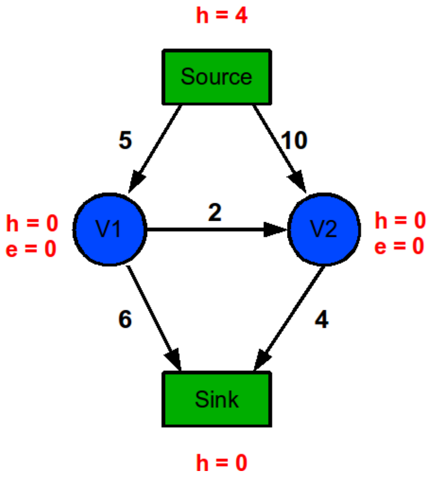
Push Relabel algorithm

The algorithm works as follows

1. Push as much fluid possible from source (towards sink)
2. Increase the height of receiver vertex.
3. Continue Pushing fluids downhill and increasing height of receiver vertex until excess flow of all vertex become zero.
4. If at any stage excess fluid can not be pushed downhill relabel the vertex (increase its height) so that excess fluid can be pushed.

Push Relabel algorithm: Example

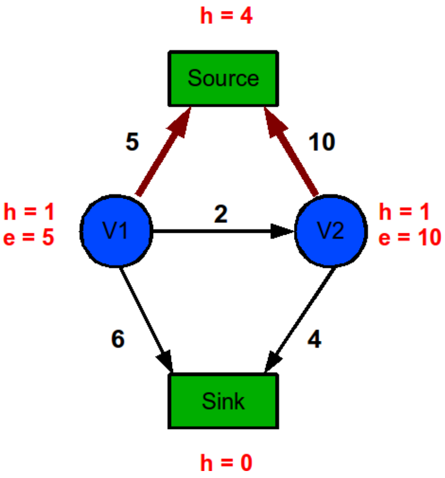




Intialization:

1. h = |v| for source
2. h = 0 for other nodes
3. e = 0 for nodes except source and sink

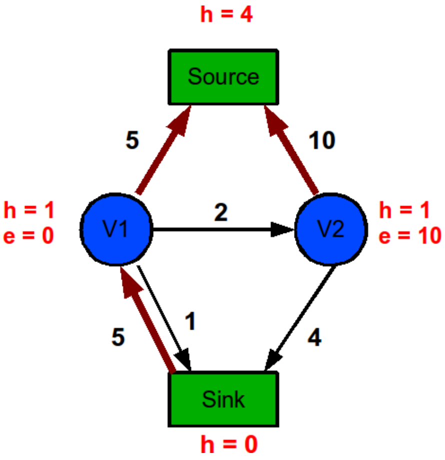
Push Relabel algorithm: Example



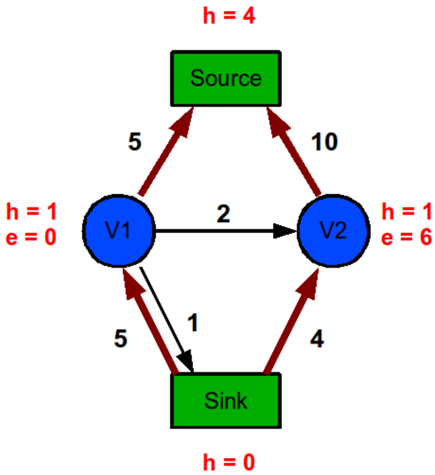
1. Push as much as possible from source to all adjacent nodes

2. Change the height of the adjacent nodes to source

Push Relabel algorithm: Example

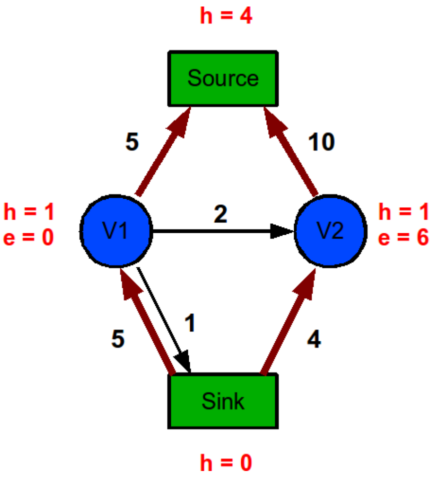


1. Push as much as possible from every node until e = 0 for all nodes



1. Push as much as possible from every node until e = 0 for all nodes

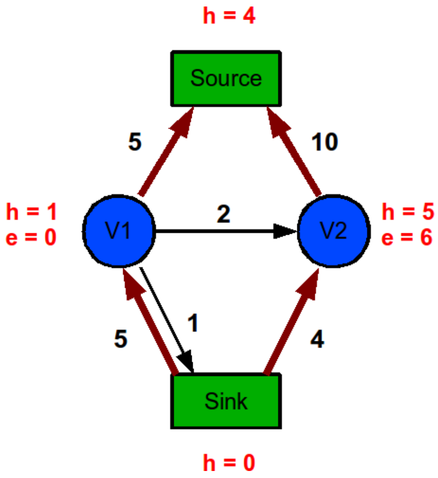
Push Relabel algorithm: Example



1. Push as much as possible from every node until e = 0 for all nodes

2. Relable the node if required

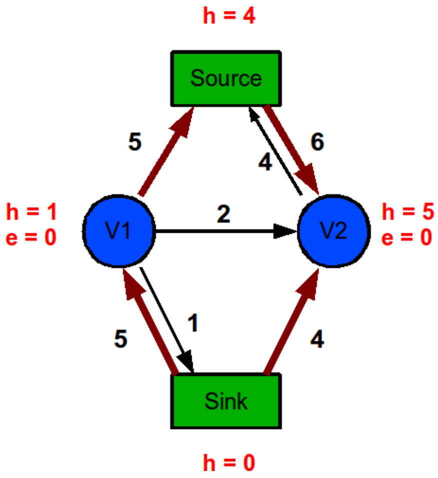
Push Relabel algorithm: Example



1. Push as much as possible from every node until e = 0 for all nodes

2. Relable the node if required
"V2 is relabled"

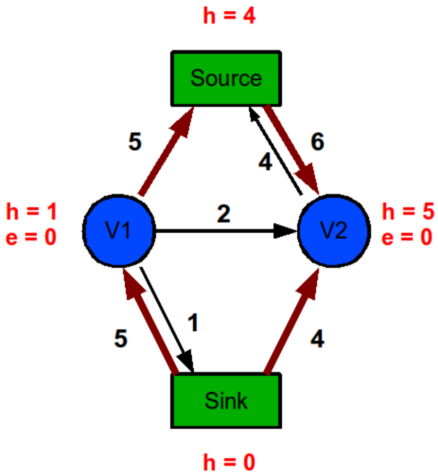
Push Relabel algorithm: Example



1. Push as much as possible from every node until e = 0 for all nodes

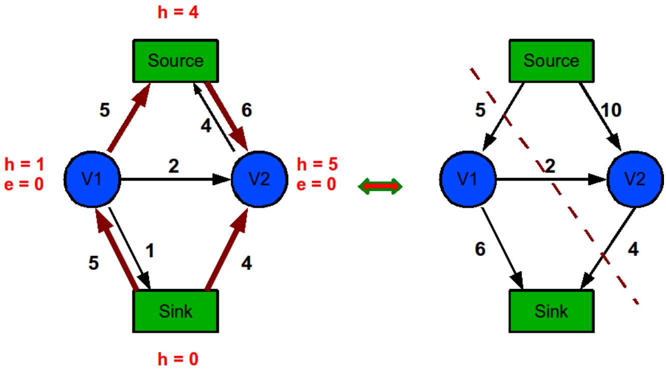
2. Relable the node if required
"V2 is relabled"

Push Relabel algorithm: Example



1. Now e = 0 for all nodes Thus we can stop the algorithm

Push Relabel algorithm: Example



Max flow = 9

Push Relabel algorithm: Analysis

Each of the basic operation Relabels, saturating pushes and bounded separately. Complexity of push relabel algorithm is $O(|V|^2|E|)$ Most efficient implementation of max flow are push relabel methods.

References

- Cormen et al, Introductions to algorithms, PHI: 2nd Edition, Chapter -26