Analysis of Algorithms

Network Flows

Hikmat Farhat

May 4, 2020

Maximum Flows

- Imagine having factory that produces materials
- You would like to transport your products to a given destination
- Suppose that there are multiple roads from factory to destination
- Some are congested and some are less some
- What is the maximum number of products you could transport from destination to source?

Flow Networks

- A **flow network** $G =< V, E >$ is a directed graph.
- Each edge $(u, v) \in E$ has a **capacity** $c(u, v) \geq 0$.
- If $(u, v) \notin E$ then we set $c(u, v) = 0$.
- There are two special vertices: **source** $s \in V$ and **sink** $t \in V$.
- We assume that the graph is connected and has no **anti parallel** edges. If $(u, v) \in E$ implies $(v, u) \notin E$.
- A **flow** is a function $f : V \times V \to \mathbf{R}$ with the following constraints:
  1. for all $u, v \in V$ we have $f(u, v) \leq c(u, v)$
  2. for all $u, v \in V$ we have $f(u, v) = -f(v, u)$
  3. for all $u \in V - \{s, t\}$ we have

$$\sum_{v \in V} f(u, v) = 0$$

Example (All examples are taken from the CLRS book)

- Notation: *flow*/*capacity*. if *flow* = 0, e.g $v_2 \to v_1$. then just *capacity*

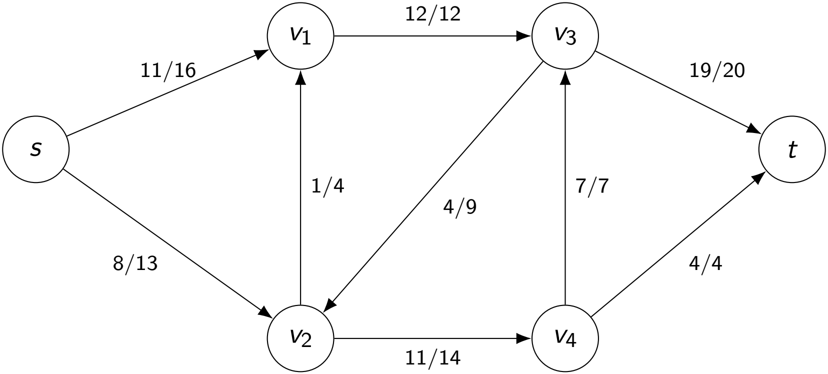


Figure: 1

Residual Networks

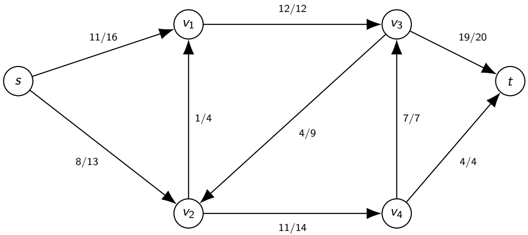
- Given a graph $G =< V, E >$, a flow f in G and a capacity function c.
- Define the residual capacity of an edge (u, v) as

$$c_R(u, v) = c(u, v) - f(u, v)$$

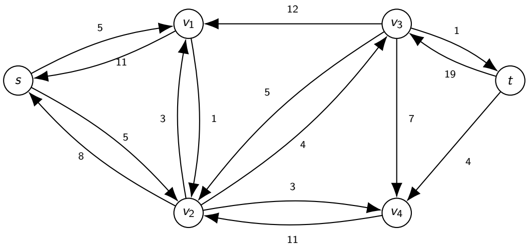
- Intuitively, the residual capacity of an edge is how much more flow can pass through it.
- Note that every $(u, v) \notin E$ also has a residual capacity.
- Since the capacity of such pairs is by definition zero then their residual capacity is

$$\forall (u, v) \notin E \;\; c_R(u, v) = -f(u, v) = f(v, u)$$

Example residual network



(a) Network



(b) Residual

Ford-Fulkerson Method

- Ford-Fulkerson is a general **method** to find a maximum flow in a a network.
- Iteratively find an **augemting path** in the residual network.
- update the residual network until there is no more augemting paths.
- the resulting flow is maximum.
- **Does not** specify how to find an augmenting path.
- For now we will find an augmenting path "visually".

```
FORD-FULKERSON(G,s,t)
```

foreach $(u, v) \in V \times V$ **do**

  $c_r(u, v) \leftarrow c(u, v)$

while $\exists$ *a path p from s to t in* G_f **do**

  $c_r(p) \leftarrow \min\{c_r(u, v) : (u, v) \in p\}$

  foreach $(u, v) \in p$ **do**

    $c_r(u, v) \leftarrow c_r(u, v) - c_r(p)$

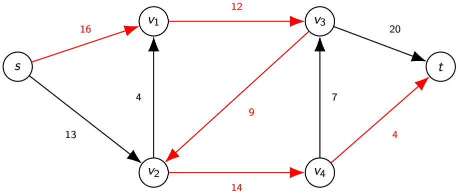
    $c_r(v, u) \leftarrow c_r(v, u) + c_r(p)$

foreach $(u, v) \in E$ **do**

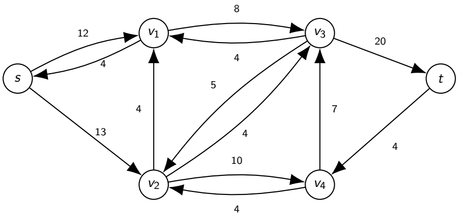
  $f(u, v) \leftarrow c(u, v) - c_r(u, v)$

Example

- Initially there is no flow. Only edges with $c_r > 0$ are shown

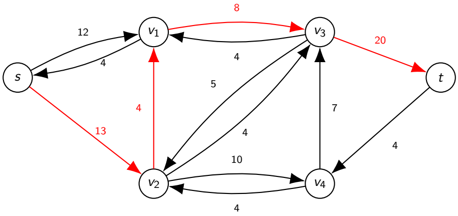


(a) $c_r(p) = 4$

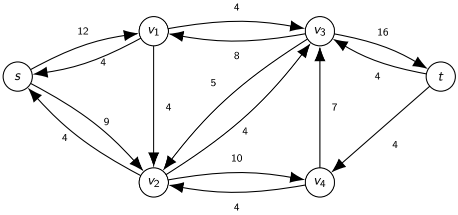


(b) update 1

Example

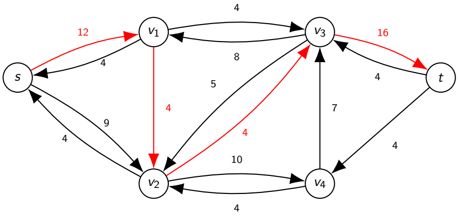


(c) $c_r(p) = 4$

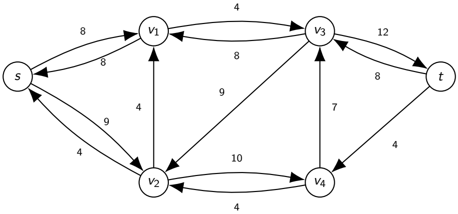


(d) update 2

Example

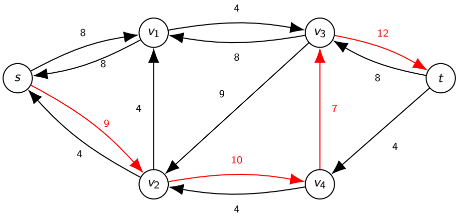


(e) $c_r(p) = 4$

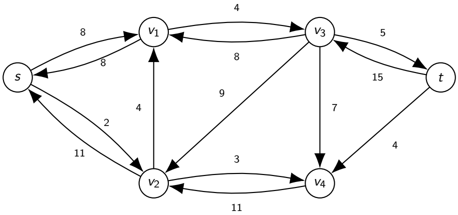


(f) update 3

Example

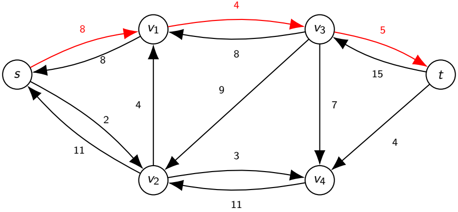


(g) $c_r(p) = 7$

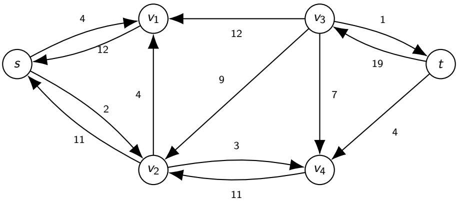


(h) update 4

Example



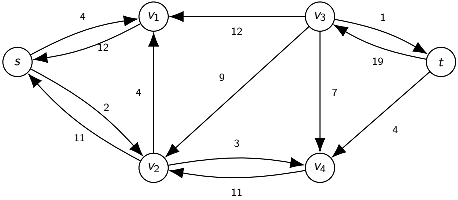
(i) $c_r(p) = 4$



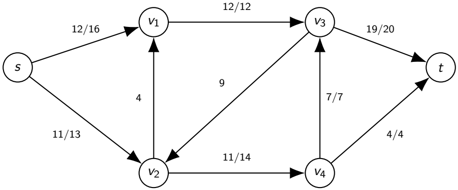
(j) update 5

Compute the flow

- for each $(u, v) \in E$ we have $f(u, v) = c_r(v, u)$. Only edges with $c_r > 0$ are shown



(k) Final Residual Network



(l) Maximum flow

Edmonds-Karp Algorithm

- Uses breadth-first-search (BFS) to find an augmenting path.
- We assign a unit weight for each edge and compute the shortest path from s to t
- Select the shortest path as the augmenting path p.

```
EDMONDS-KARP(G,s,t)
```

foreach $(u, v) \in V \times V$ **do**

  $c_r(u, v) \leftarrow c(u, v)$

while $\exists$ *a* **shortest** *path p from s to t in* G_f **do**

  $c_r(p) \leftarrow \min\{c_r(u, v) : (u, v) \in p\}$

  foreach $(u, v) \in p$ **do**

    $c_r(u, v) \leftarrow c_r(u, v) - c_r(p)$

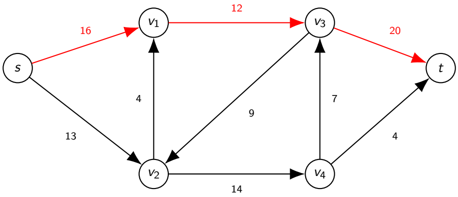
    $c_r(v, u) \leftarrow c_r(v, u) + c_r(p)$

foreach $(u, v) \in E$ **do**

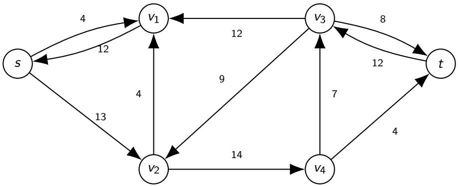
  $f(u, v) \leftarrow c(u, v) - c_r(u, v)$

Sample example but using shortest path

- Initially there is no flow. Only edges with $c_r > 0$ are shown



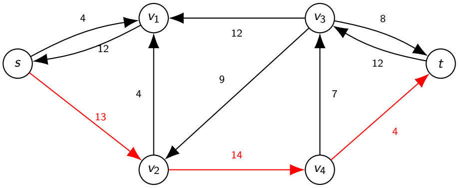
(a) $c_r(p) = 12$



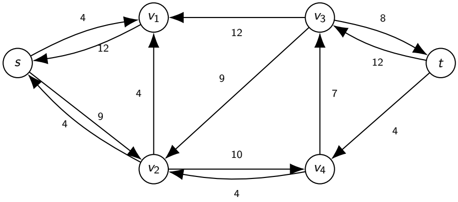
(b) update 1

Sample example but using shortest path

- Initially there is no flow. Only edges with $c_r > 0$ are shown



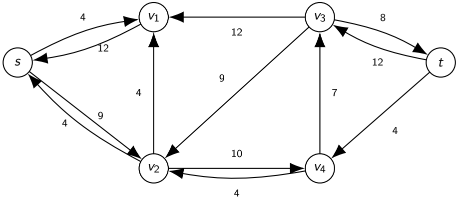
(c) $c_r(p) = 4$



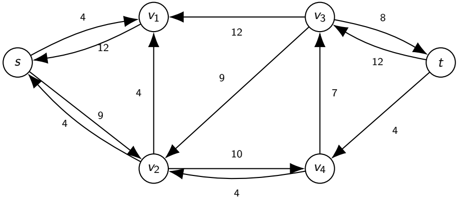
(d) update 1

Sample example but using shortest path

- Initially there is no flow. Only edges with $c_r > 0$ are shown



(e) $c_r(p) = 4$

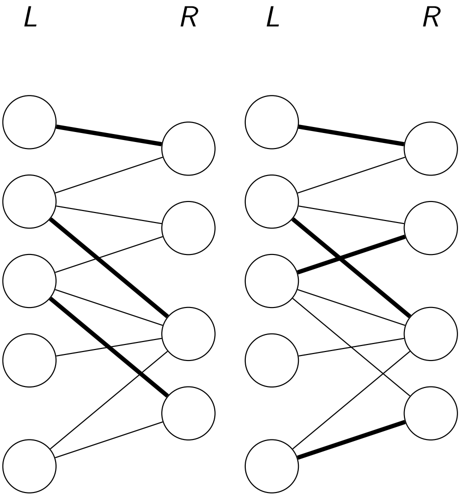


(f) update 1

Bipartite matching

- Given a graph $G =< V, E >$ a **matching** is a set of edges $M \subseteq E$ such that for all $v \in V$ at **most** one edge in M is incident on v.
- $v \in V$ is **matched** if $\exists (u, v) \in M$ for some $u \in V$.
- M is said to be a maximum matching if for all matching M' we have $|M'| \leq |M|$
- A graph $G =< V, E >$ is said to be bipartite if it can be partitioned $V = L \cup R$ where $L \cap R = \emptyset$ and for all $(u, v) \in E$, $u \in L$ and $v \in R$.

Example: Matching in bipartite graphs



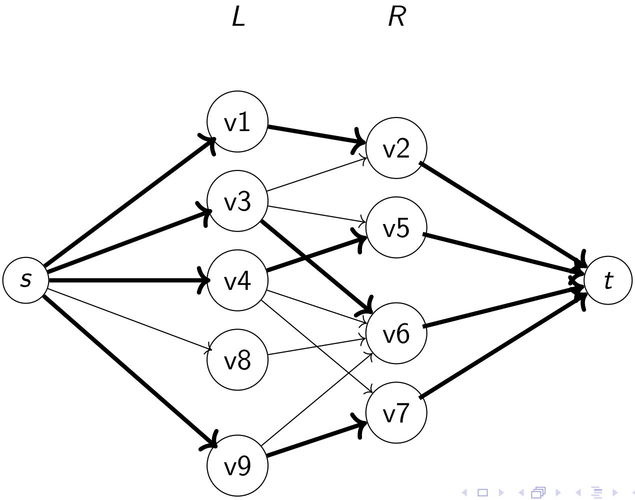
matching maximum matching

Constructing an equivalent flow network

- Given a bipartite graph $G =< V, E >$ we construct a new (flow) graph $G' =< V', E' >$ as follows:
- $V' = V \cup s, t$. With $s \in L$ and $t \in R$.
- $E' = E \cup \{(s, u) \mid u \in L)\} \cup \{(u, t) \mid u \in R\}$
- Also every edge $(u, v) \in E$ is made a direct edge from L to R.
- Finally the capacity of every edge in E' is set to 1.

Example

- From the maximum matching in the previous example we construct the flow network shown below where the maximum matching corresponds to the maximum flow.



Example

- two windows and one linux machine

Edge disjoing paths

- Given a graph $G =< V, E >$ and set of paths is said to be edge disjoint if each $(u, v) \in E$ appears in at most one path.
- The problem to be solved is as follows
  1. Given a directed graph $G =< V, E >$ and two nodes s and t
  2. Find the maximum number of edge disjoint paths from s to t