Flow Networks Part-II (DAA, M.Tech + Ph.D.)

By:

Sunil Kumar Singh, PhD
Assistant Professor,
Department of Computer Science and Information Technology

School of Computational Sciences, Information and Communication Technology,
Mahatma Gandhi Central University, Motihari
Bihar, India-845401

Outline

- Network flow problems
- Max-flow minimum cut
- Ford-Fulkerson algorithm
- Conclusion
- References

Ford-Fulkerson Algorithm

- It is a simple and practical max-flow algorithm.

- Main idea: find valid flow paths until there is none left, and add them up

- How do we know if this gives a maximum flow?

✓ Proof sketch: suppose not, take a maximum flow $f^{\star}$ and “subtract” our flow f. It is a valid flow of positive total flow. By the flow decomposition, it can be decomposed into flow paths and circulations. These flow paths must have been found by ford-Fulkerson . Contradiction.

Problem definition and Constraints

- It is not required to maintain the amount of flow on each edge but work with capacity values directly.
- If f amount of flow goes through u →v, then:
  - ✓ Decrease c(u →v) by f
  - ✓ Decrease c(u →v) by f
- Why do we need this?
  - ✓ Sending flow to both directions is equivalent to cancelling flow.

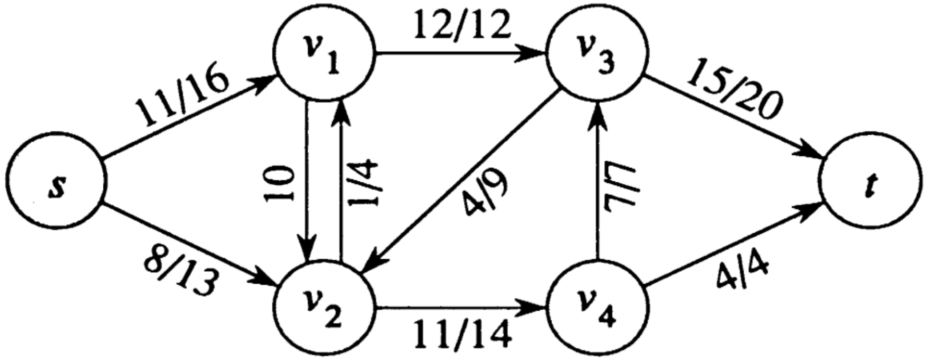
Cont..

- **New ideas**
  - ✓ Residual flow networks – these show where extra capacity might be found
  - ✓ Augmenting paths – the path along which extra capacity is possible
  - ✓ cuts – used to characterize the maximum flow possible in a network

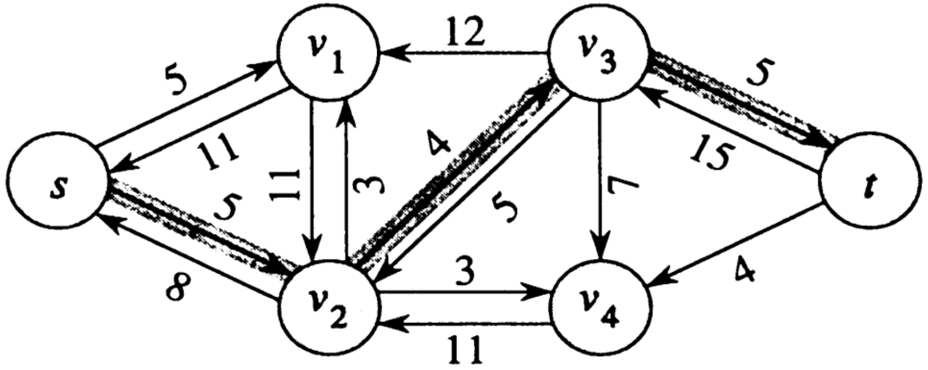
- **Residual Networks**
  - ✓ $c_f(u, v)=c(u,v)-f(u,v)$
  - ✓ The residual network is a graph with the same vertices but the edges are the residual capacities

Cont..

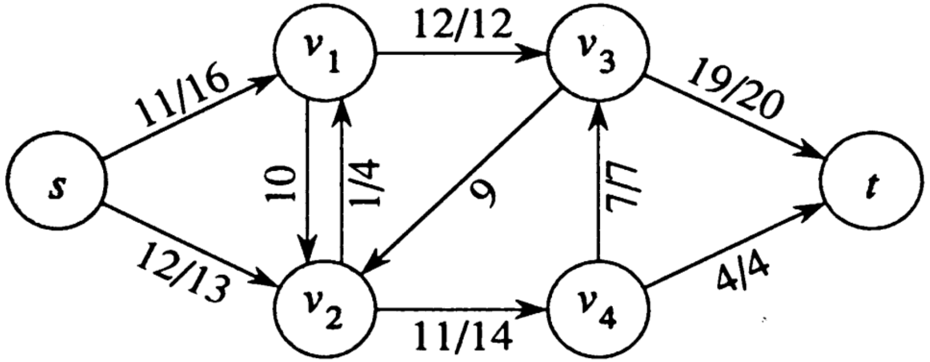
- ✓ $E_f = \{(u,v) \in V \times V : c_f(u,v) \geq 0\}$



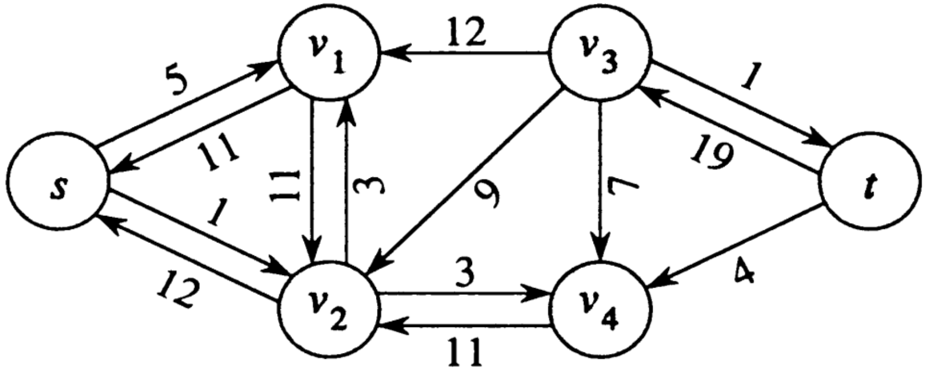
(a)



(b)



(c)



(d)

Augmenting Paths

- An augmenting path is a simple path form s to t in the residual network
  - ✓ The capacity of the augmenting path p is the maximum residual additional flow we can allow along the augmenting path
  - ✓ $c_f(p)=\min\{c_f(u,v):(u,v) \text{ is on } p\}$

Lemma 27.3

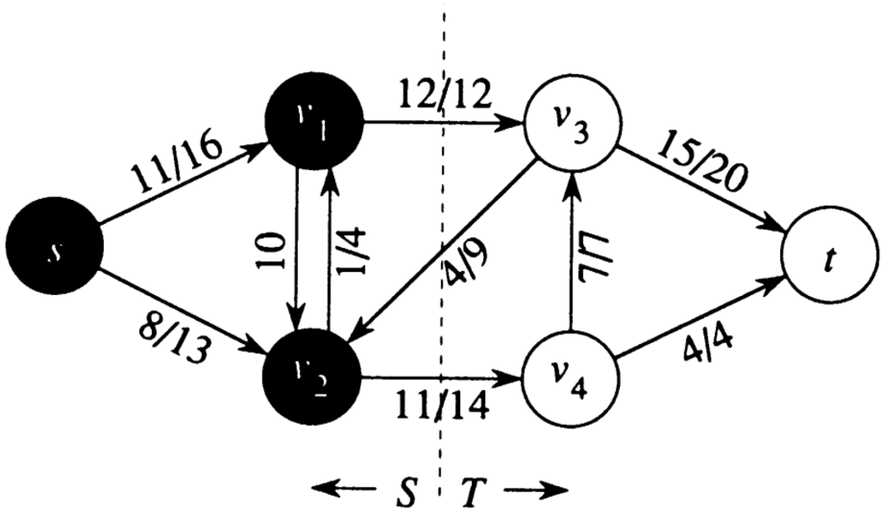
Let $G = (V, E)$ be a flow network, let f be a flow in G, and let p be an augmenting path in G_f. Define a function $f_p : V \times V \to \mathbf{R}$ by

$$f_p(u,v) = \begin{cases} c_f(p) & \text{if } (u,v) \text{ is on } p\ , \\ -c_f(p) & \text{if } (v,u) \text{ is on } p\ , \\ 0 & \text{otherwise}\ . \end{cases} \tag{27.6}$$

Then, f_p is a flow in G_f with value $|f_p| = c_f(p) > 0$. ■

Cuts of Flow Networks

- Definitions
- A cut (S,T) of flow network G=(V,E) is a partition of V into S and T=V-S such that s ∈ S and t ∈ T
- The net flow across a cut is f(S,T), the capacity is c(S,T)



  – f(S,T)=f(v1,v3)+f(v2,v3)+f(v2,v4)=12+(-4)+11=19
  – c(S,T) = c(v1,v3) + c(v2+v4) = 12 + 14 = 26

Ford-Fulkerson method

```
FORD-FULKERSON(G, s, t)
for each edge (u, v) ∈ E[G]
     do f[u, v] ← 0
        f[v, u] ← 0
while there exists a path p from s to t in the residual network G_f
     do c_f(p) ← min {c_f(u, v) : (u, v) is in p}
        for each edge (u, v) in p
            do f[u, v] ← f[u, v] + c_f(p)
               f[v, u] ← −f[u, v]
```

Analysis

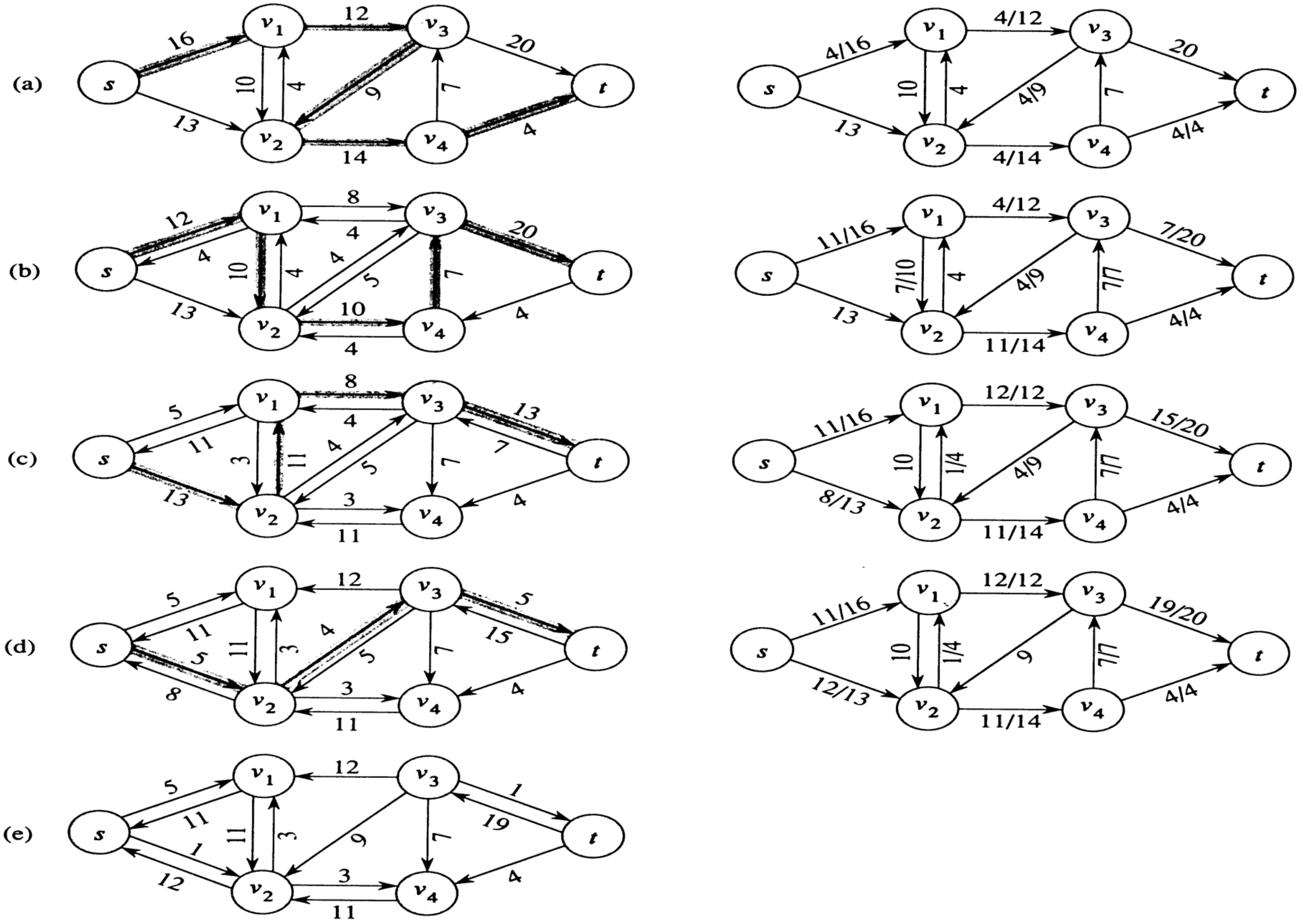
- Assumption: capacities are integer-valued

- Finding a flow path takes $\Theta(n + m)$ time

- We send at least 1 unit of flow through the path

- If the max-flow is f^*, the time complexity is $O((n + m)\, f^*)$
  - ✓ "Bad" in that it depends on the output of the algorithm
  - ✓ Nonetheless, easy to code and works well in practice

Example of Execution



Cont..

- To the left are successive iterations of the while loop
- To the right are the residual graphs
- The residual network at the last while loop test, it has no augmenting paths, and the flow f shown in figure (d) is therefore a maximum flow

References

1. Cormen, Thomas H., Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to algorithms*. MIT press, 2009.
2. Cormen, Thomas H., Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. "Introduction to algorithms second edition." *The Knuth-Morris-Pratt Algorithm, year* (2001).
3. Seaver, Nick. "Knowing algorithms." (2014): 1441587647177.

Thank You