SC1007
Data Structures and Algorithms

Week 12: Matching Problem
Maximum Flow Algorithm

College of Engineering

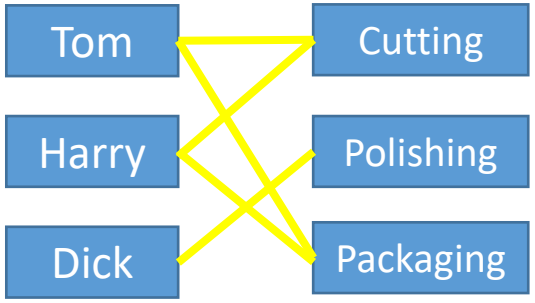
School of Computer Science and Engineering

Instructor: Luu Anh Tuan

Email: anhtuan.luu@ntu.edu.sg

Office: #N4-02b-66

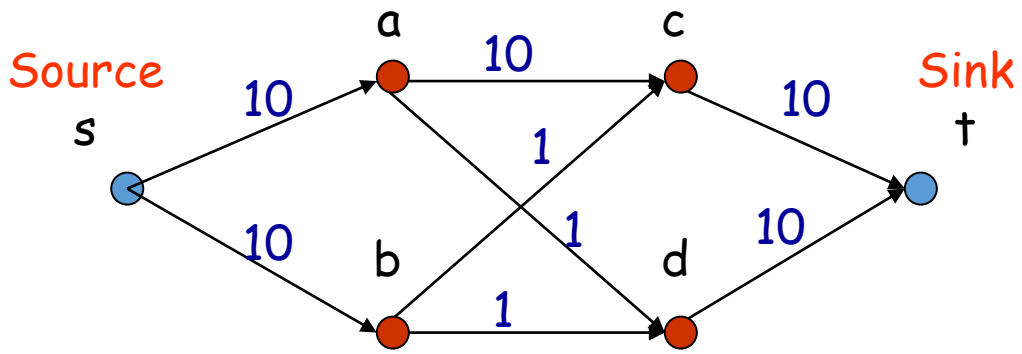
Matching Problem



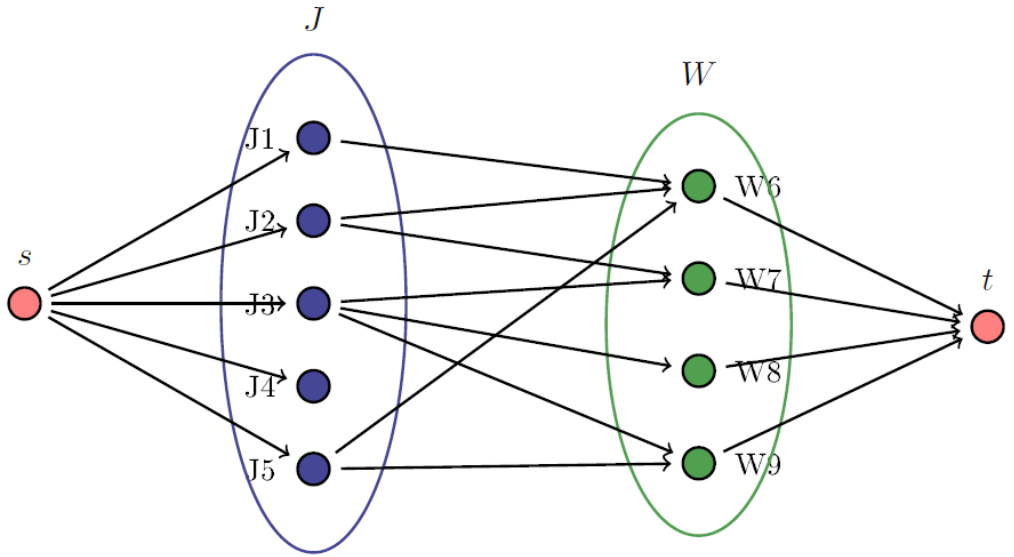
- A graph that vertex V can be partitioned into two subsets, e.g. J and W.
- J and W are two disjoint sets.
- Every edge connects a vertex in J to one in W.
- This graph is known as Bipartite Graph.

- Matching:
  - A subset of edges that are mutually non-adjacent
  - No two edges have an endpoint in common
- Problem: Matching with the maximum number of edges

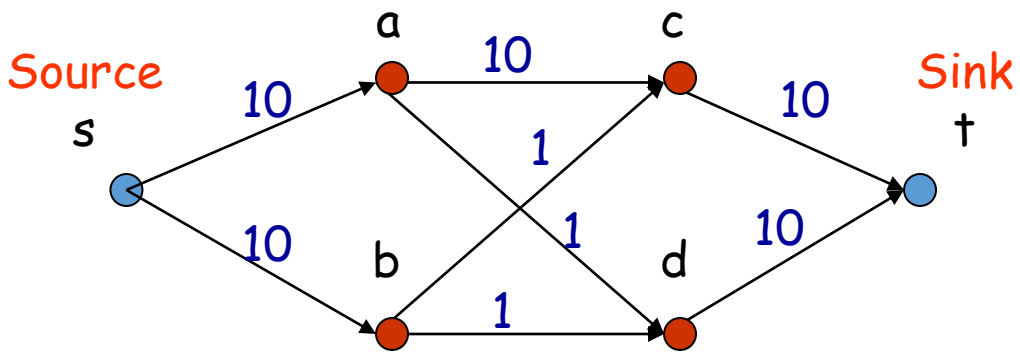
Network flow



- Introduce two vertices: a source *s* and a sink *t*
- A flow network G(V,E) is a directed graph in which each edge (j, w) has a nonnegative capacity.
- Maximum Matching == Maximum Flow
  - The capacity is {0, 1} for matching problem



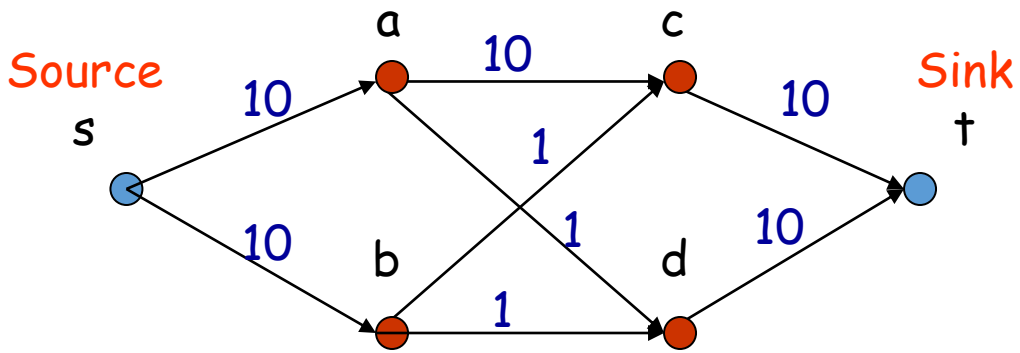
Network Flows



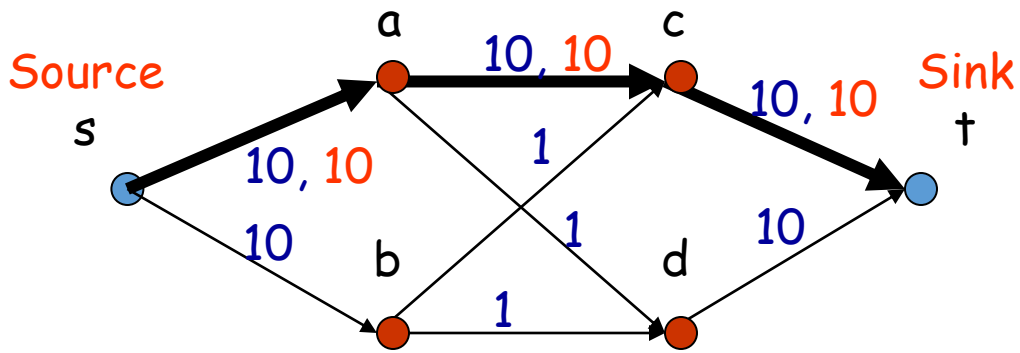
- Let $G = [V,E]$ be a directed graph with capacity $c(v,w)$ on edge $[v,w]$.
- A *flow* is an (integer) function, f, that is chosen for each edge so that f(v,w) <= c(v,w)
- We wish to maximize the flow allocation.

A maximum network flow example

By inspection



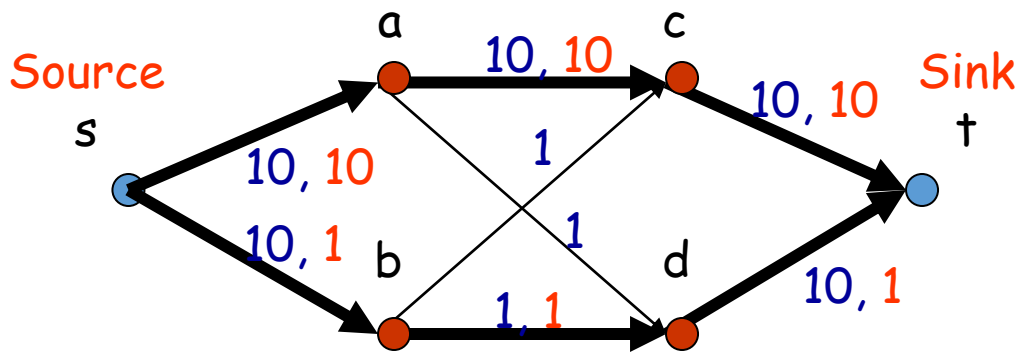
Step 1:



Flow is of size 10

A maximum network flow example

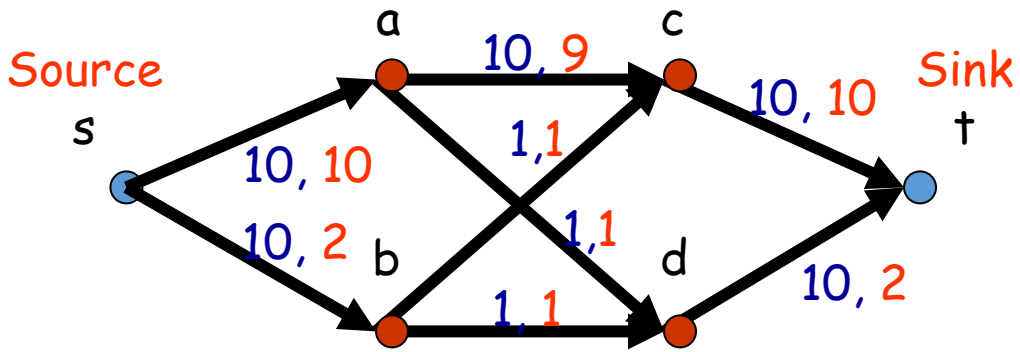
Step 2:



Flow is of size 10+1 = 11

Not obvious

Maximum flow:



Flow is of size 10+2 = 12

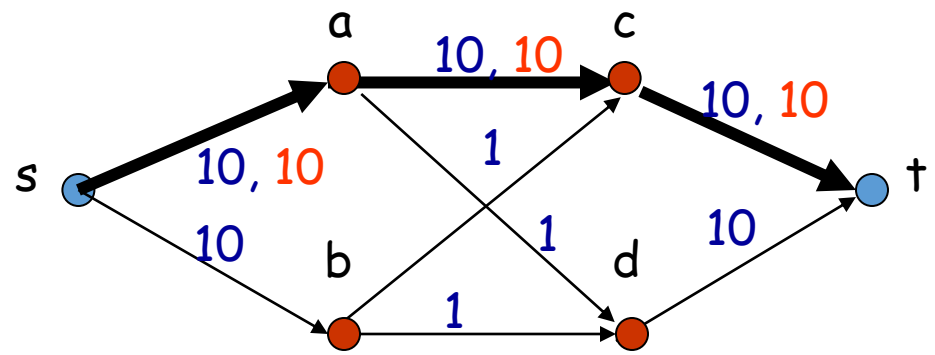
The Ford-Fulkerson Method

- An iterative improvement strategy
- Proposed by L. R. Ford Jr. and D. R. Fulkerson in 1956
- Iteratively find an additional flow (match) in the network
  - A residual network is used to find the available flow
  - $c_f(j, w) = c(j, w) - f(j, w)$

Ford-Fulkerson method of augmenting paths

1. Set $f(v,w) = -f(w,v)$ on all edges.
2. Define a Residual Graph, in which $c_f(v,w) = c(v,w) - f(v,w)$
3. Find paths from s to t for which there is positive residue.
4. Increase the flow along the paths to augment them by the minimum residue along the path.
5. Keep augmenting paths until there are no more to augment.

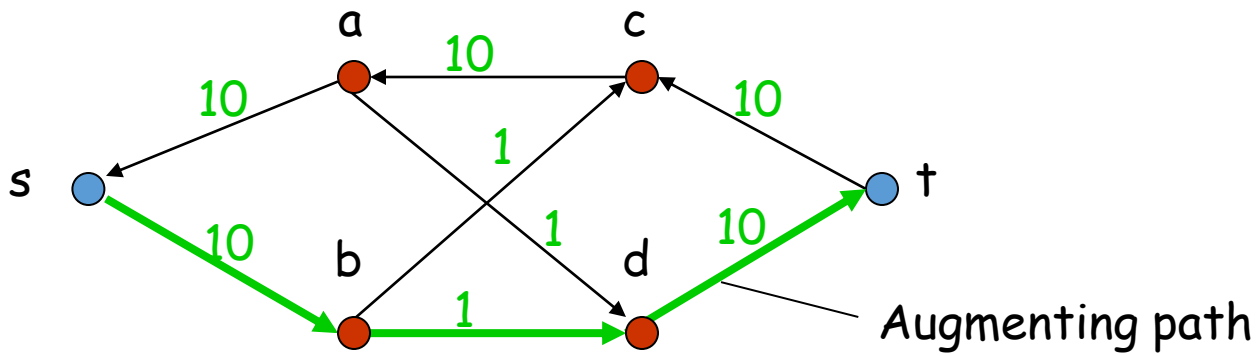
Example of Residual Graph



Flow is of size 10

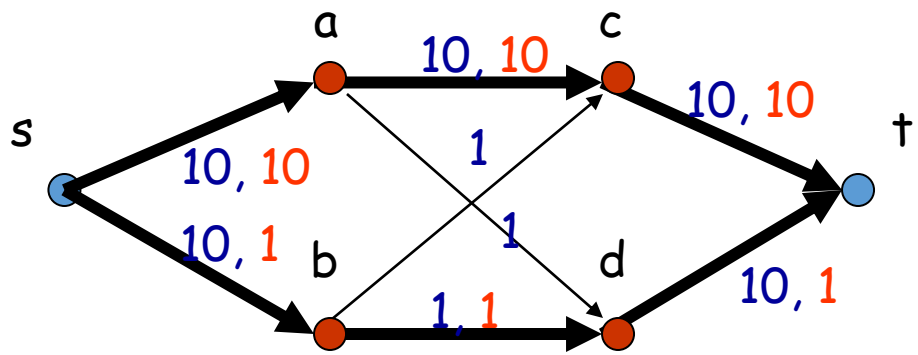
Residual Graph, ***R***

$c_f(v,w) = c(v,w) - f(v,w)$



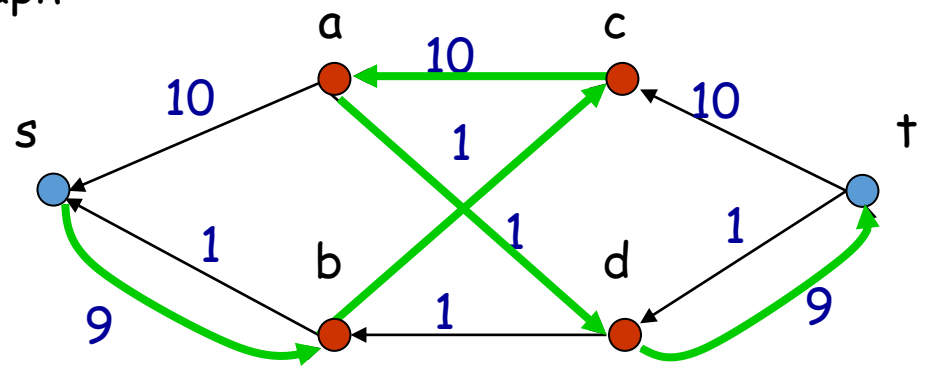
Example of Residual Graph

Step 2:



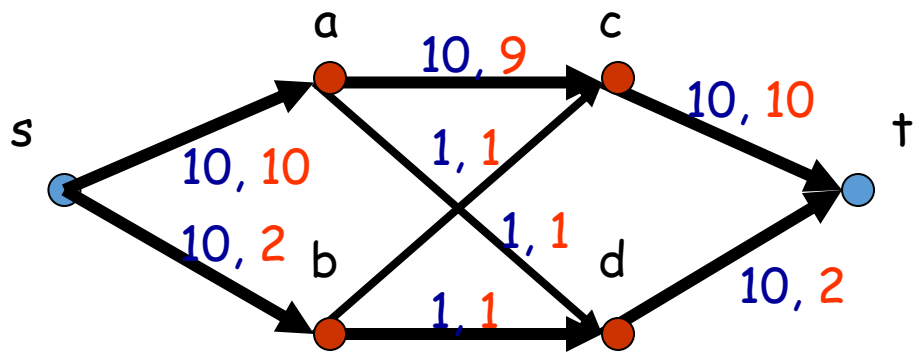
Flow is of size 10+1 = 11

Residual Graph



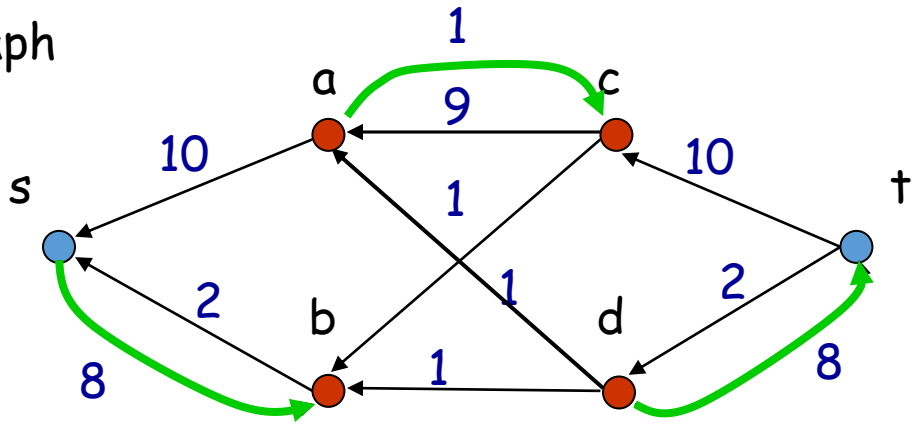
Example of Residual Graph

Step 3:

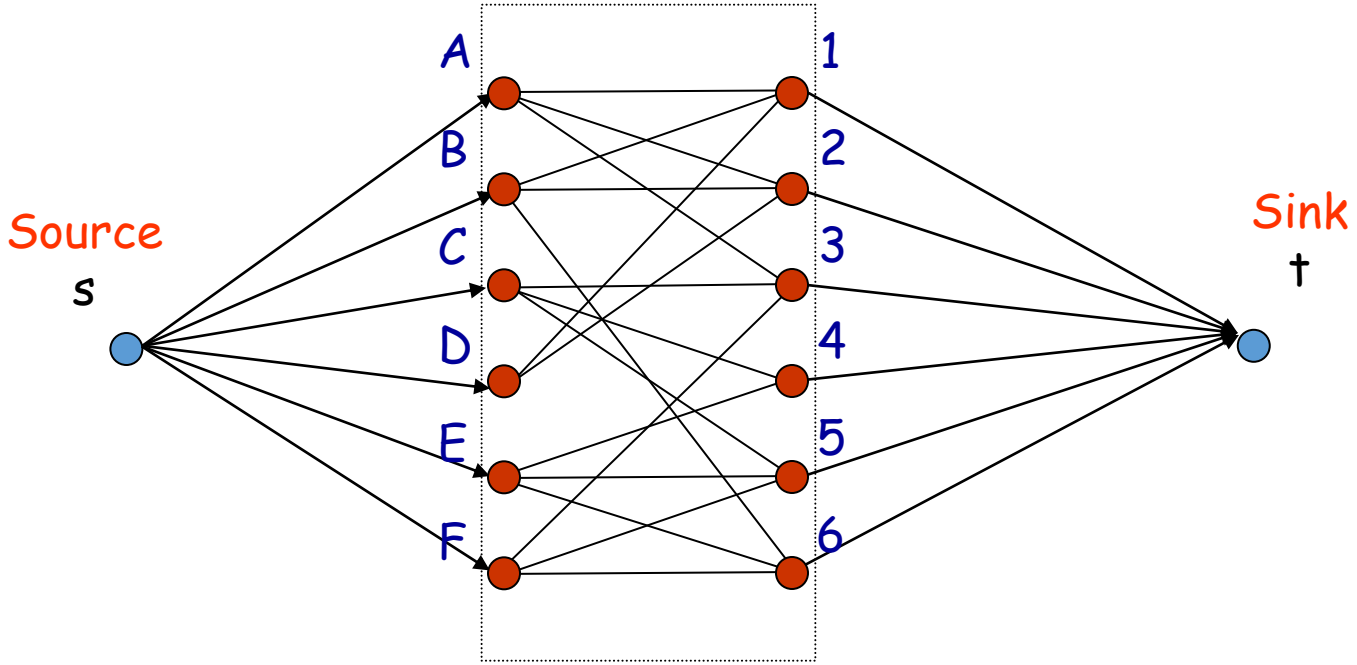


Flow is of size 10+2 = 12

Residual Graph



Network flows and bipartite matching



Finding a maximum size bipartite matching is equivalent to solving a network flow problem with capacities and flows of size 1.

The Ford-Fulkerson Method for matching problem

- Iteratively find an additional flow (match) in the network
  - A residual network is used to find the available flow
  - $c_f(j,w) = c(j,w) - f(j,w)$

- Initially, there is no flow
- Residual network == original network

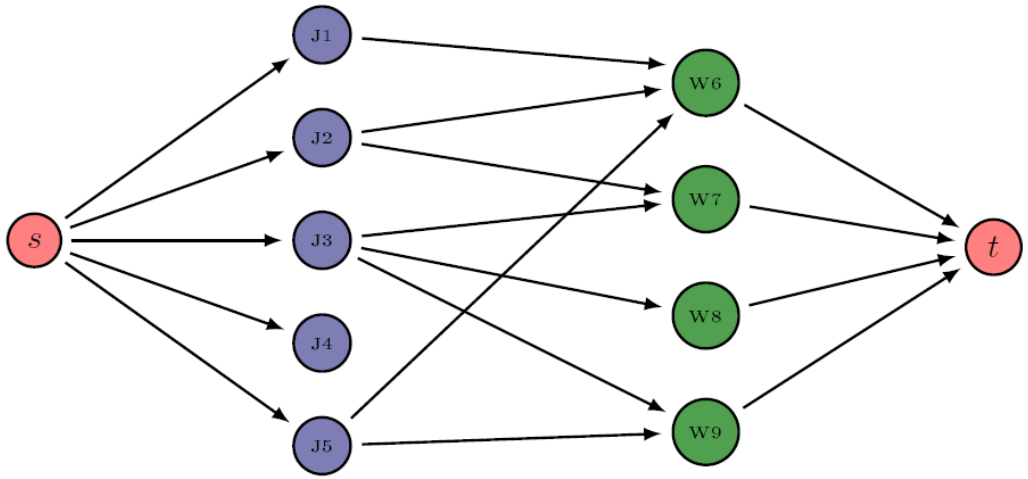
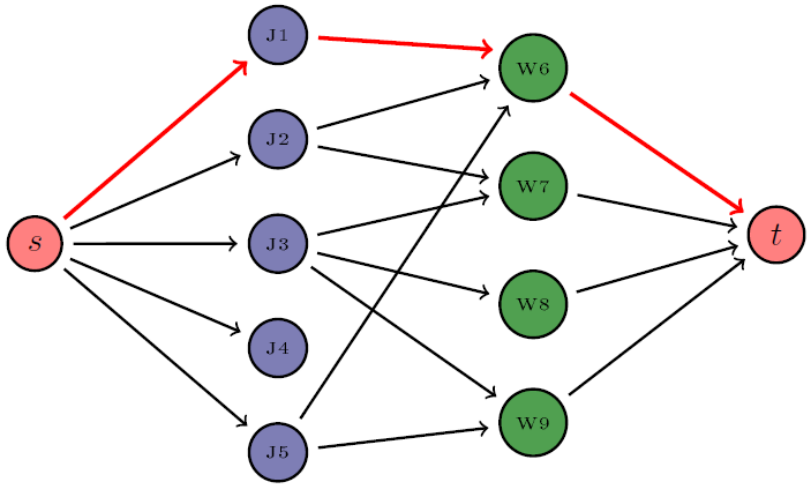


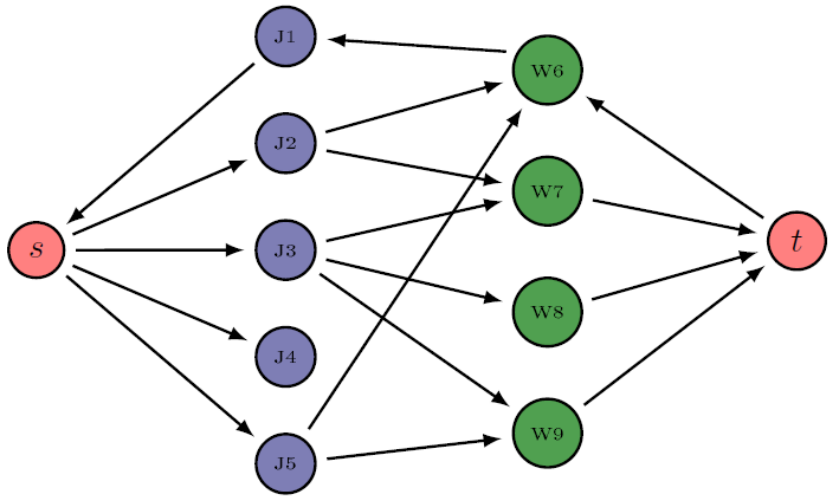
Figure 12.6: Residual Network G_f where $c_f(j,w)$ is 1 $\forall e_f$

The Ford-Fulkerson Method

- In each iteration,
  - Find an augmenting path p from s to t in the residual network G_f
  - Update the original matching network G by adding p
  - Update the residual network G_f

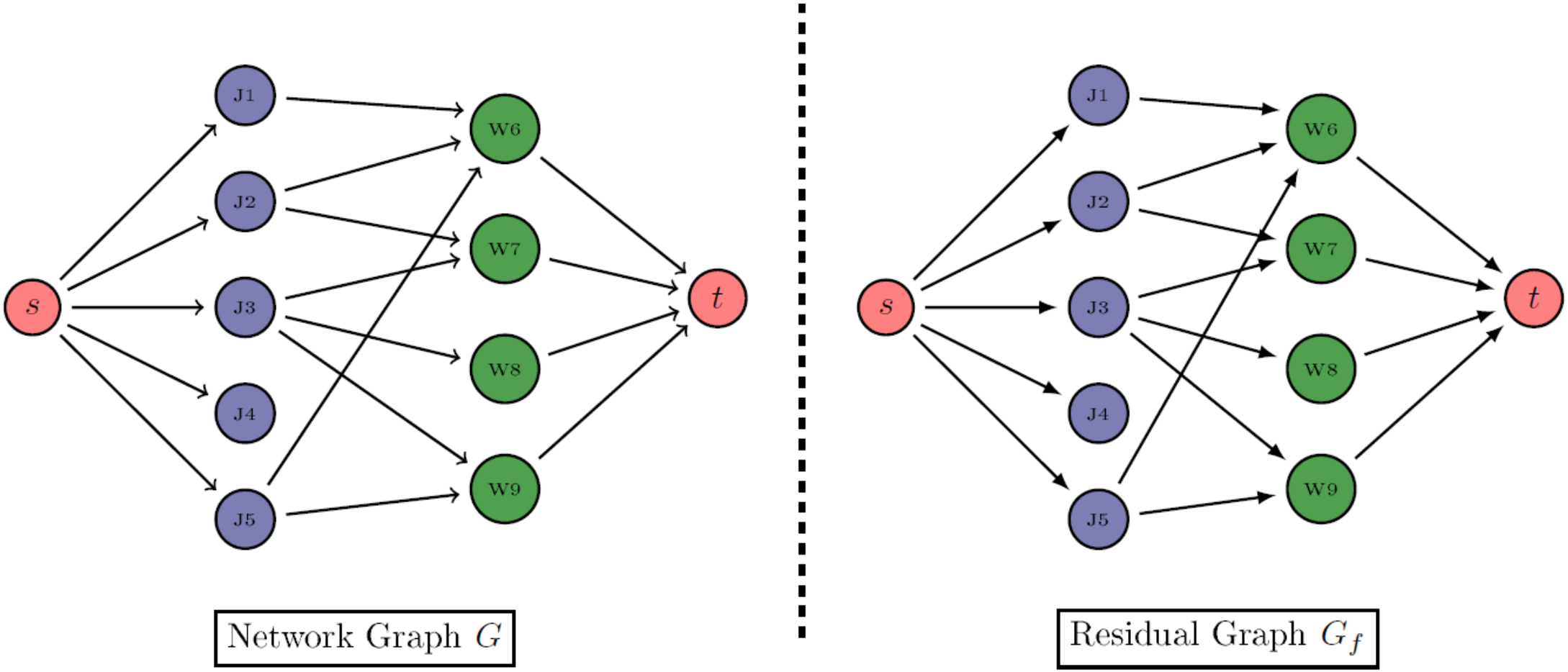


(a) Matching Network G



(b) Residual Network G_f

First …



Network Graph G

Residual Graph G_f

Algorithm 2 Ford-Fulkerson

$$c_f(j,w) = c(j,w) - f(j,w)$$

function FORD-FULKERSON(Graph G, Vertex s, Vertex t)

for each edge $(u,v) \in E[G]$ **do** ▷ Initialization of Flows

$f[u,v] \leftarrow 0$

$f[v,u] \leftarrow 0$

end for

while Finding a path from s to t in G_f **do**

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

for each edge $(u,v) \in p$ **do**

if $(u,v) \in E$ **then**

$f[u,v] \leftarrow f[u,v] + c_{min}(p)$ ▷ adding the new flow

else

$f[v,u] \leftarrow f[v,u] - c_{min}(p)$

end if

$c_f(u,v) \leftarrow c_f(u,v) - c_{min}(p)$ ▷ Update Residual Graph, G_f

$c_f(v,u) \leftarrow c_f(v,u) + c_{min}(p)$

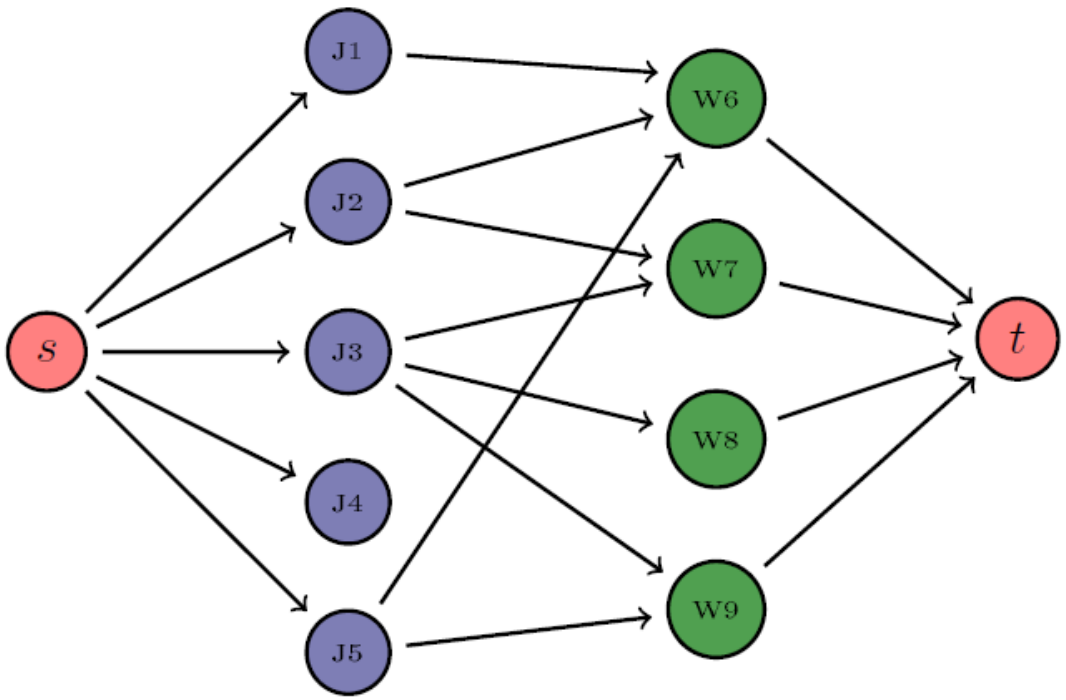
end for

end while

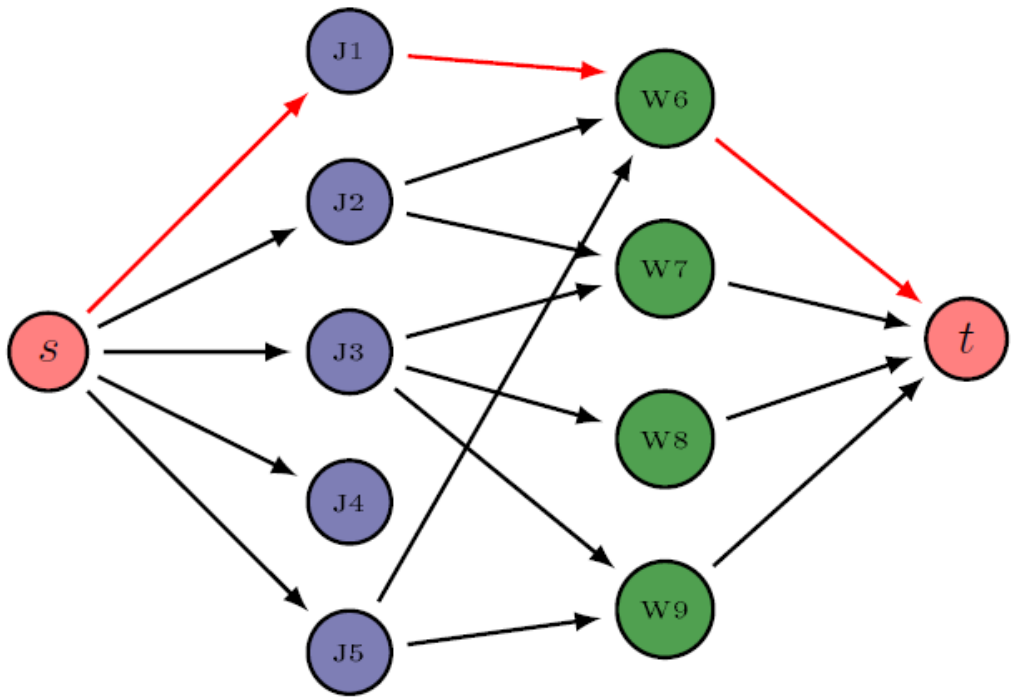
end function

- $c_f(j,w) = c(j,w)$: same as the original network G
- $c_f(j,w)$ is non-zero

Next …



Network Graph G



Residual Graph G_f

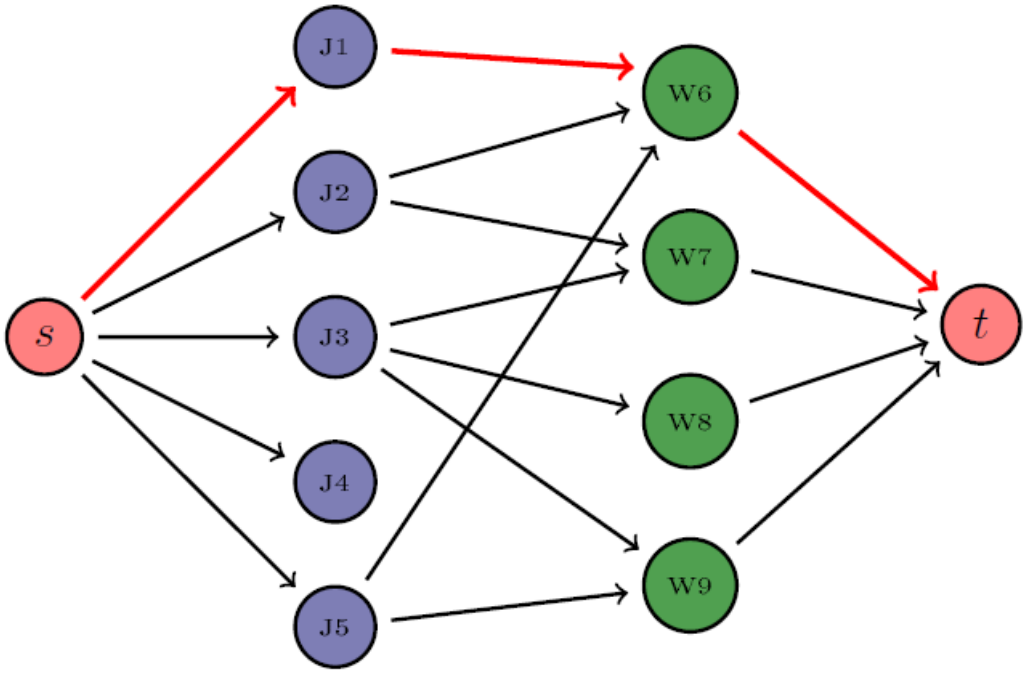
Algorithm 2 Ford-Fulkerson

$c_f(j,w) = c(j,w) - f(j,w)$

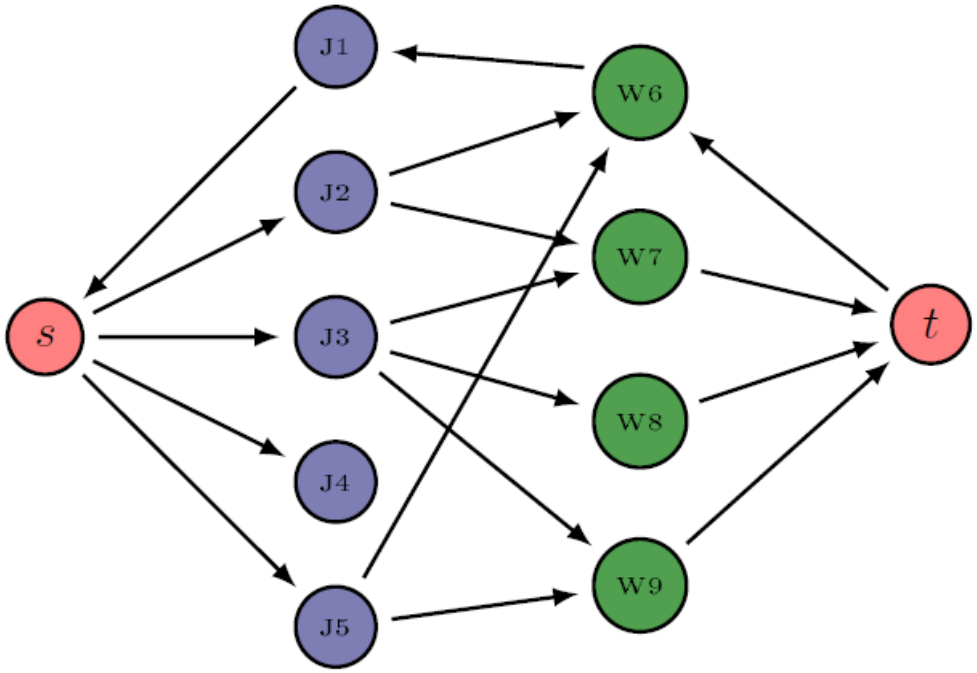
function FORD-FULKERSON(Graph G, Vertex s, Vertex t)
  for each edge $(u,v) \in E[G]$ **do** ▷ Initialization of Flows
    $f[u,v] \leftarrow 0$
    $f[v,u] \leftarrow 0$
  end for
  while Finding a path from s to t in G_f **do**
    $c_{min}(p) \leftarrow \min\{c_f(u,v) : (u,v) \in p\}$
    for each edge $(u,v) \in p$ **do**
      if $(u,v) \in E$ **then**
        $f[u,v] \leftarrow f[u,v] + c_{min}(p)$ ▷ adding the new flow
      else
        $f[v,u] \leftarrow f[v,u] - c_{min}(p)$
      end if
      $c_f(u,v) \leftarrow c_f(u,v) - c_{min}(p)$ ▷ Update Residual Graph, G_f
      $c_f(v,u) \leftarrow c_f(v,u) + c_{min}(p)$
    end for
  end while
end function

- $c_f(j,w) = c(j,w)$: same as the original network G
- $c_f(j,w)$ is non-zero
- $c_{min}(p)$ is always 1 for matching problem here

Next…

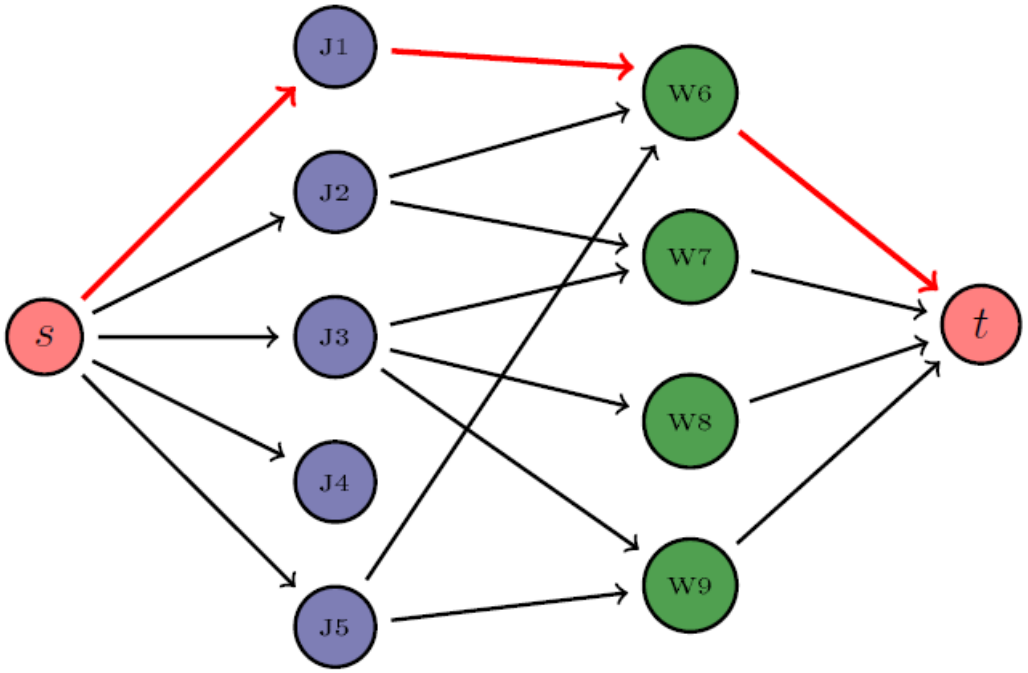


Network Graph G

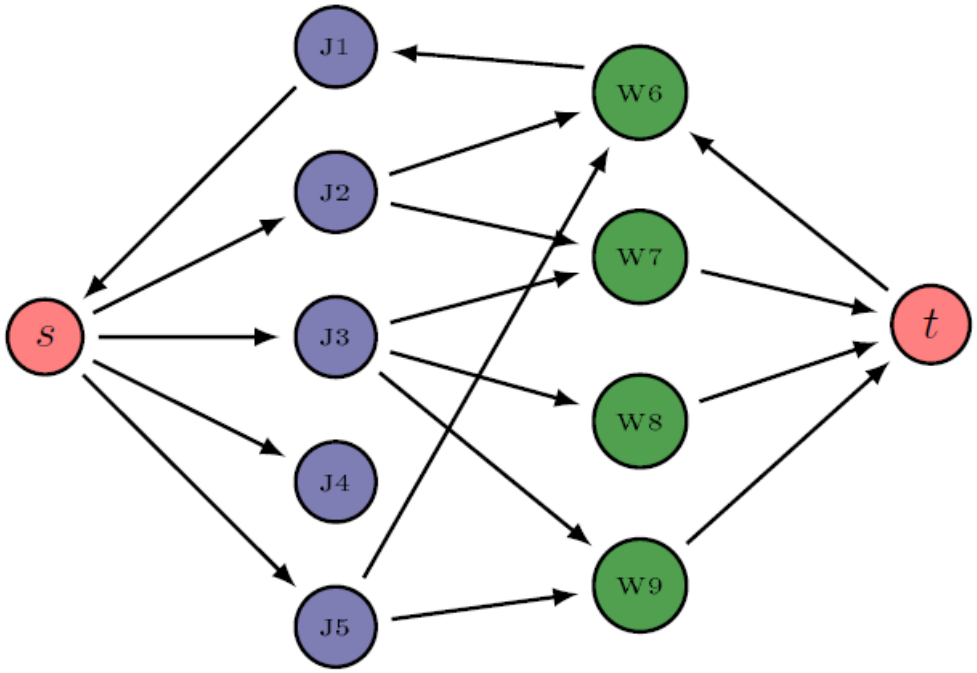


Residual Graph G_f

Next…

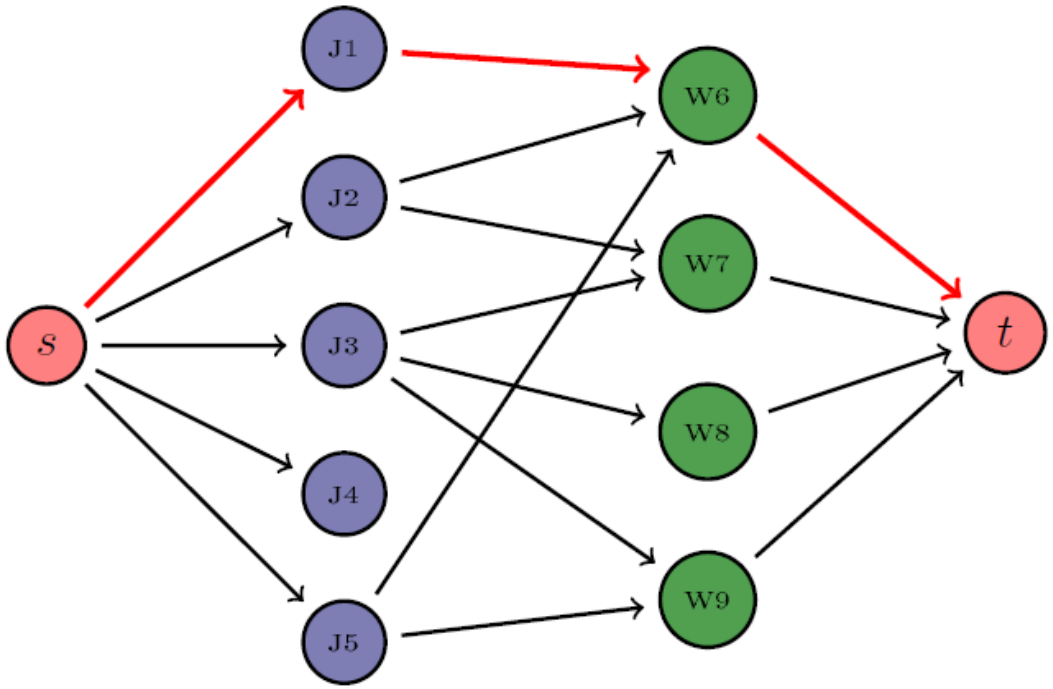


Network Graph G

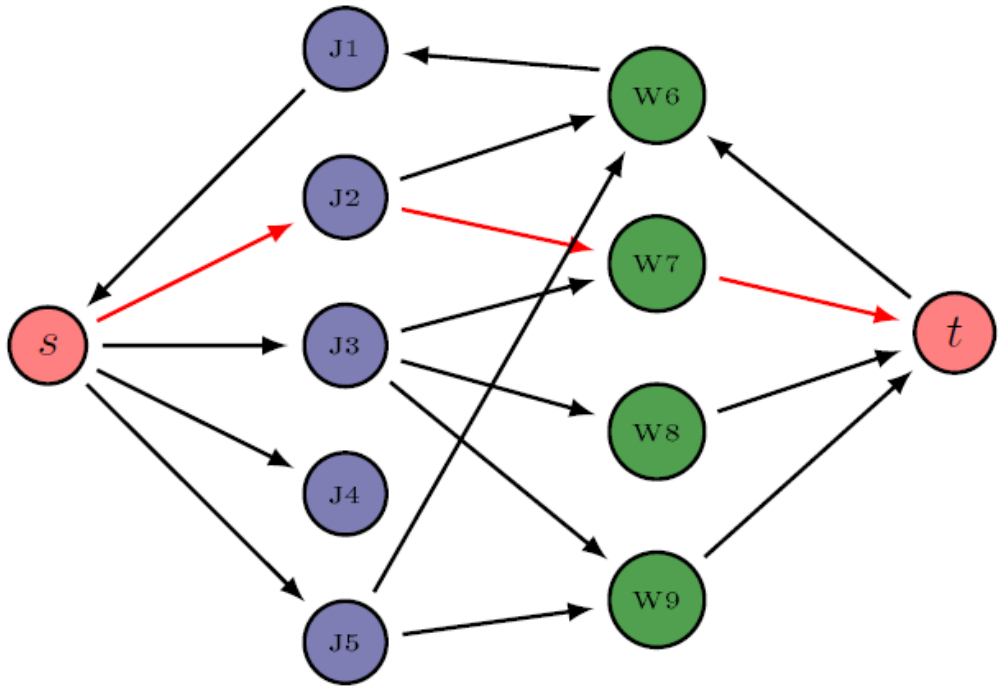


Residual Graph G_f

Next …

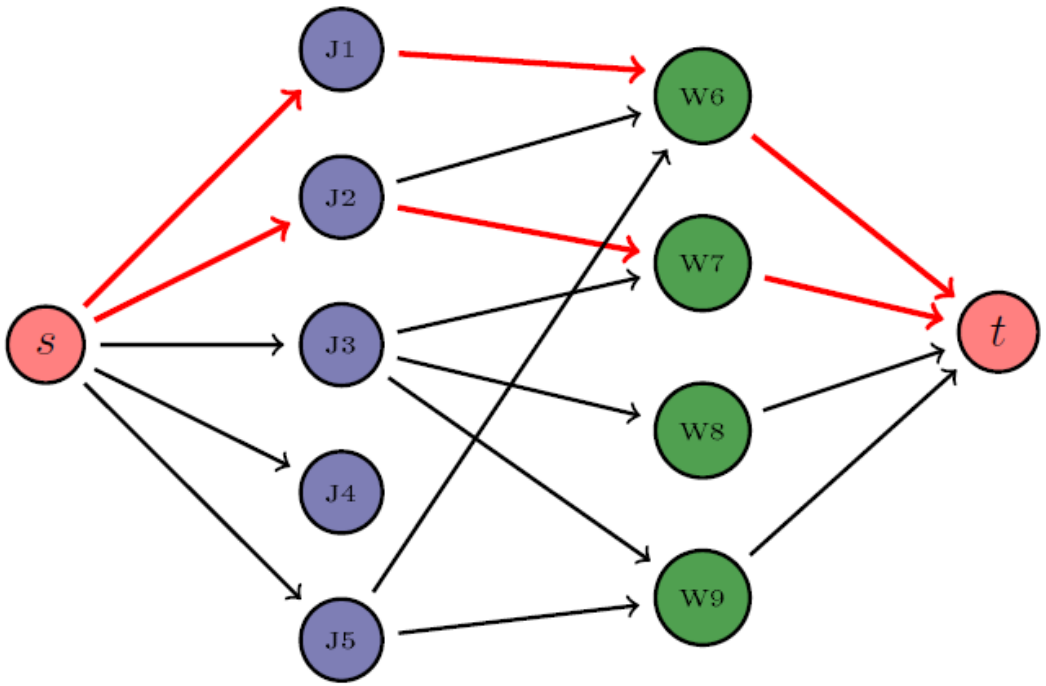


Network Graph G

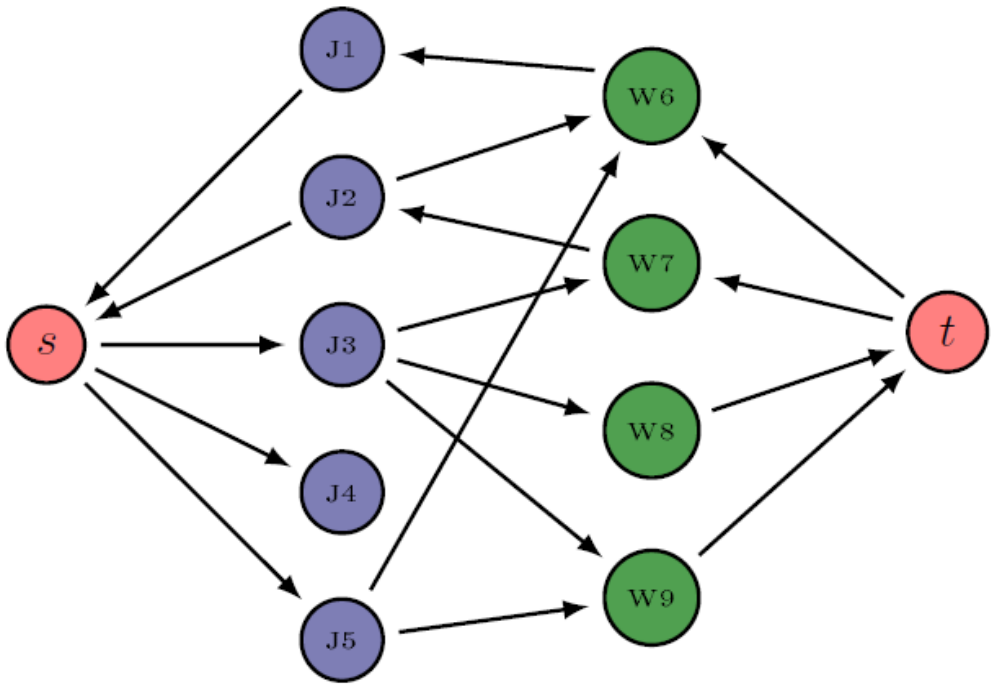


Residual Graph G_f

Next …

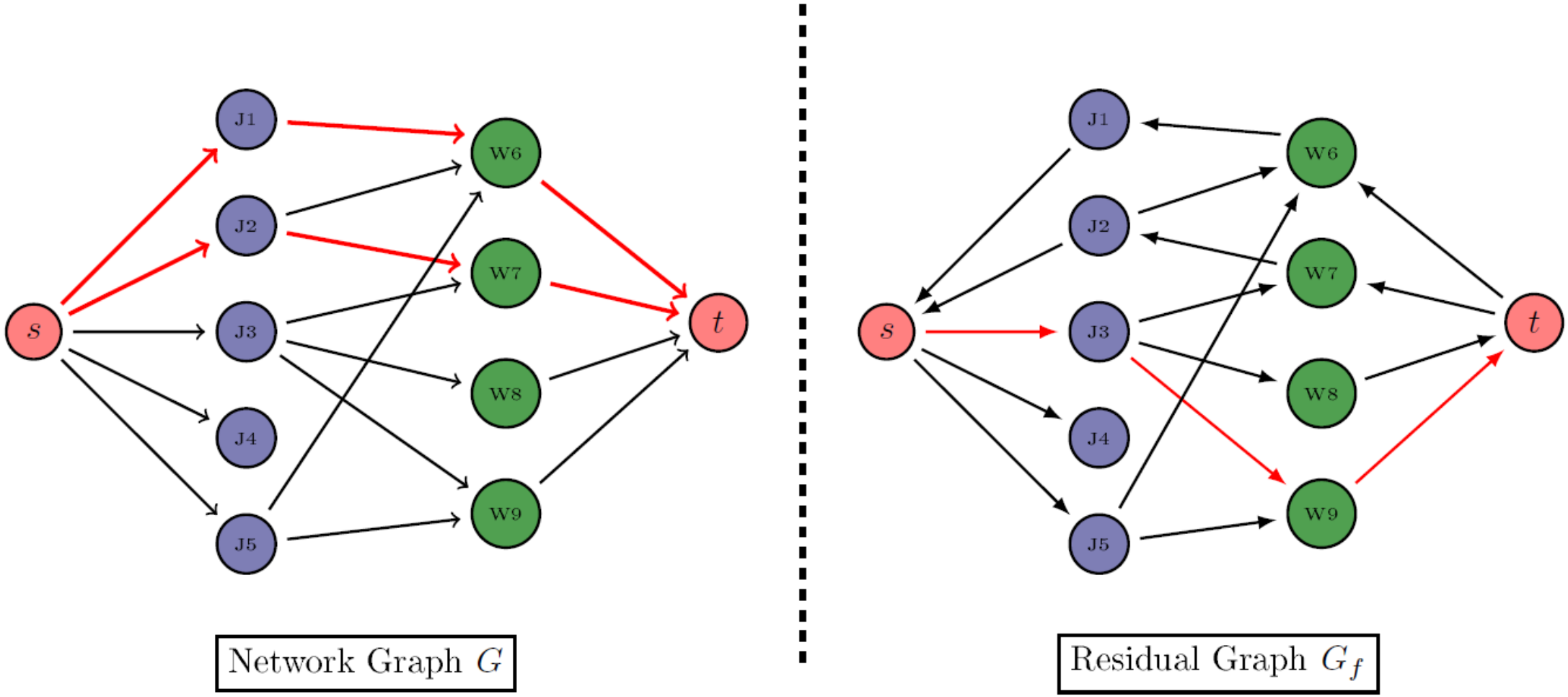


Network Graph G

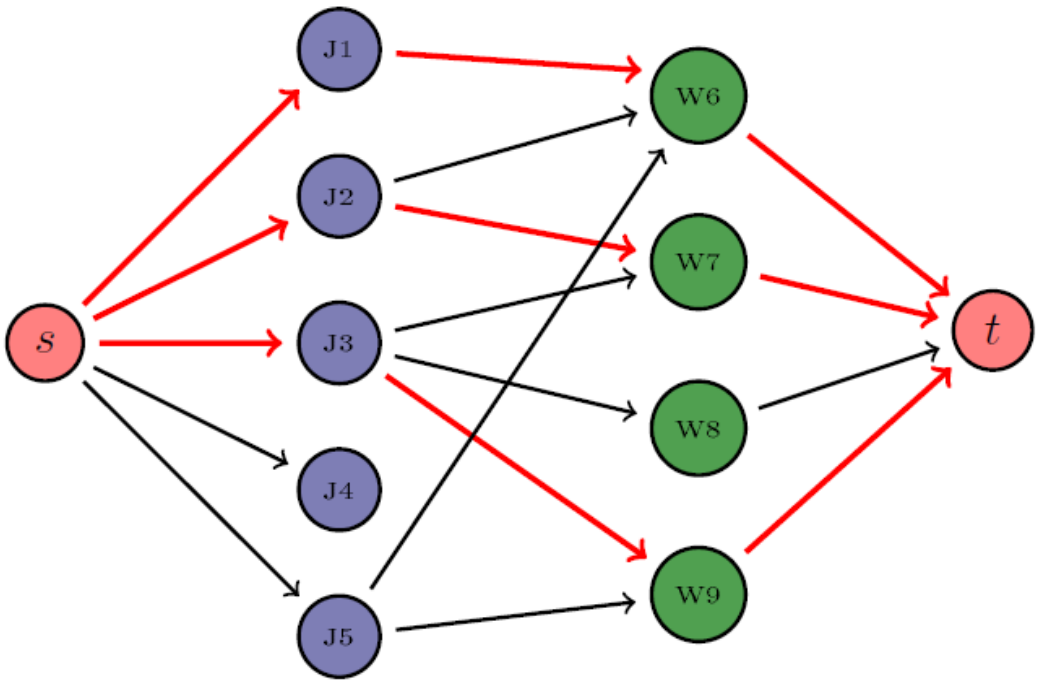


Residual Graph G_f

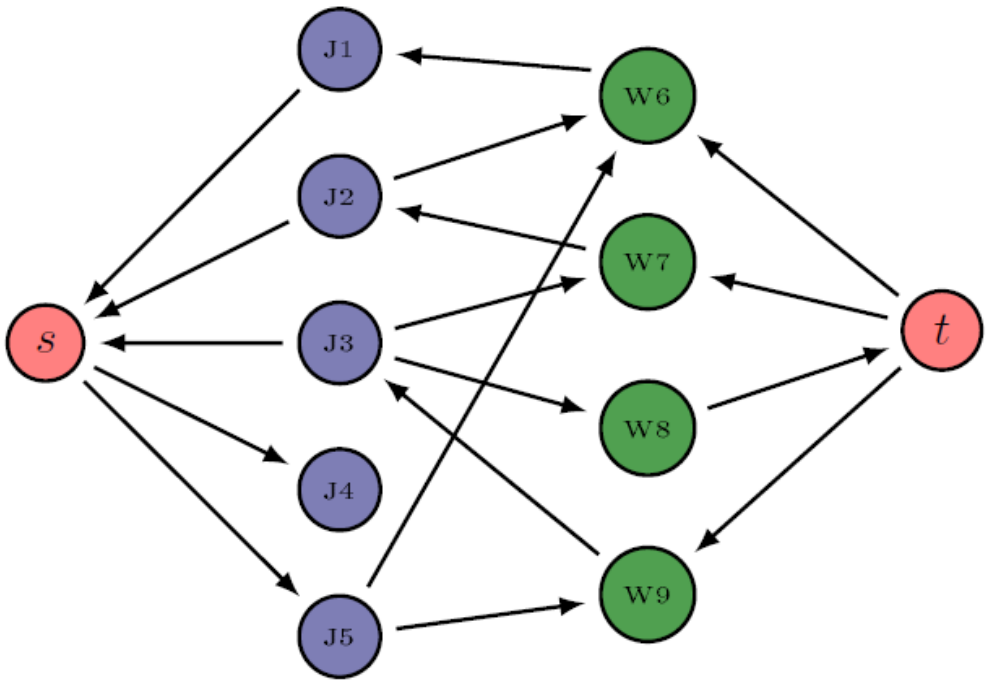
Next …


J1
J2
J3
J4
J5
W6
W7
W8
W9
s
t
Network Graph G
J1
J2
J3
J4
J5
W6
W7
W8
W9
s
t
Residual Graph G_f

Next …

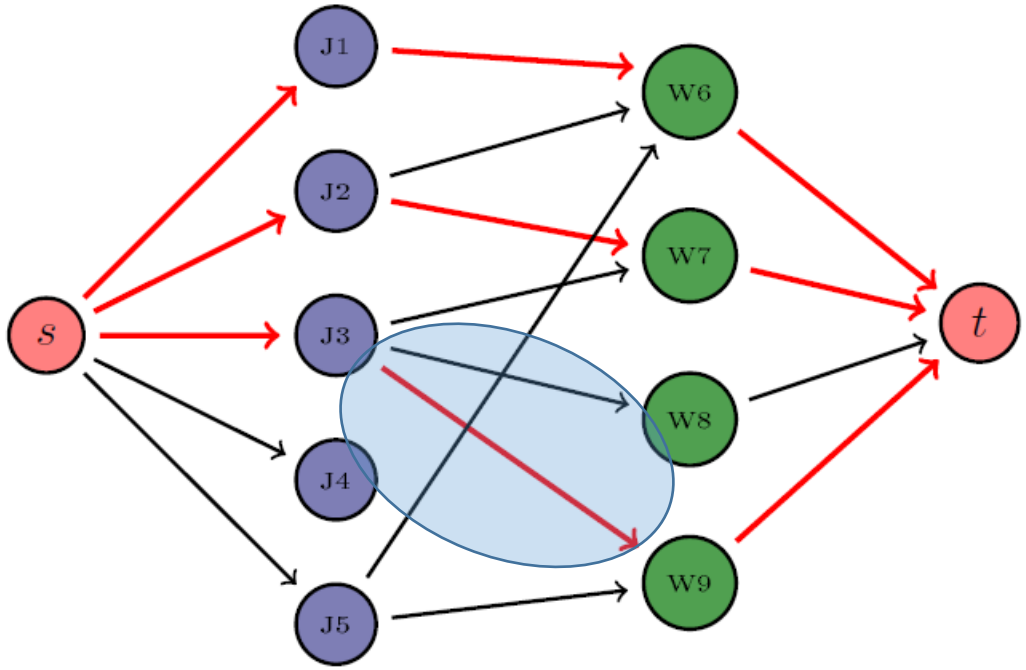


Network Graph G

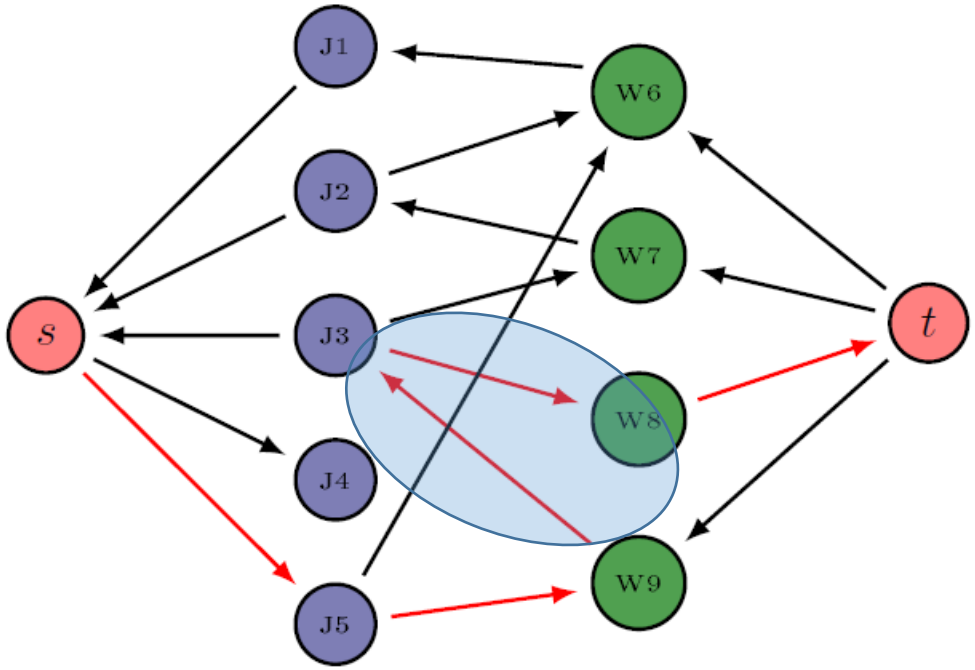


Residual Graph G_f

Next… Why does it work?



Network Graph G



Residual Graph G_f

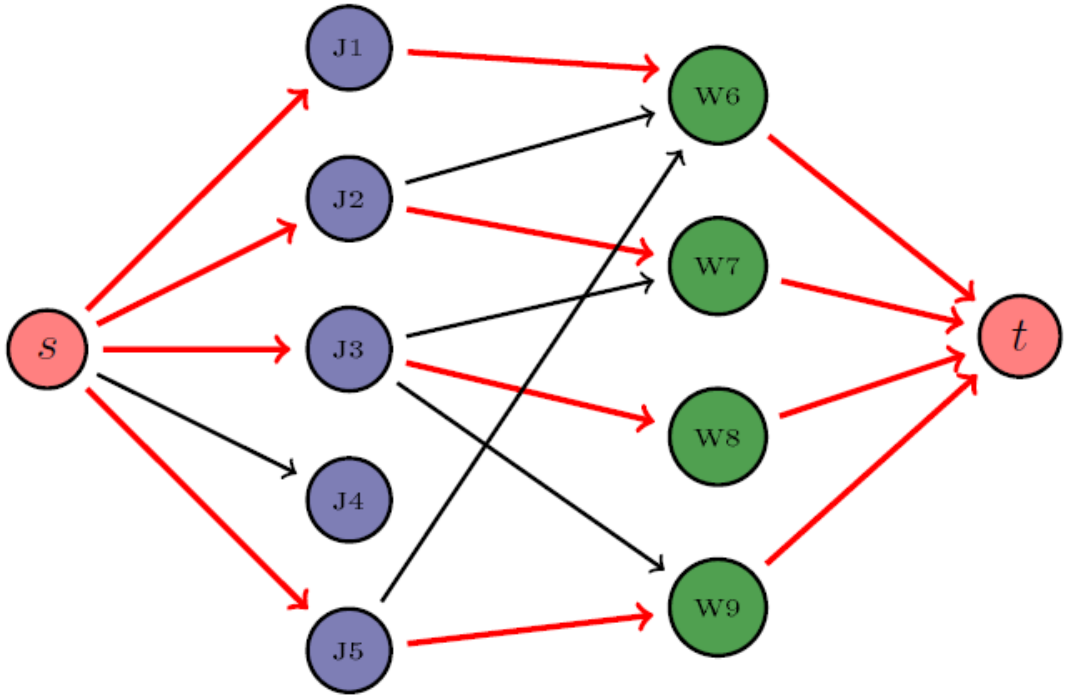
Algorithm 2 Ford-Fulkerson

$$c_f(j,w) = c(j,w) - f(j,w)$$

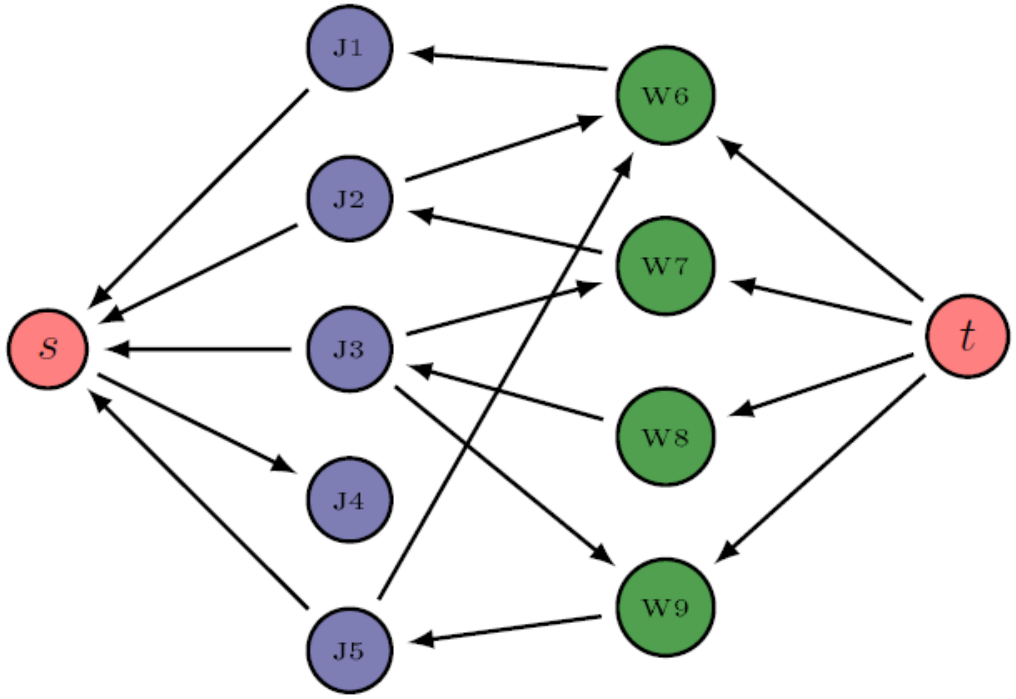
function FORD-FULKERSON(Graph G, Vertex s, Vertex t)
  for each edge $(u,v) \in E[G]$ **do** ▷ Initialization of Flows
    $f[u,v] \leftarrow 0$
    $f[v,u] \leftarrow 0$
  end for
  while Finding a path from s to t in G_f **do**
    $c_{min}(p) \leftarrow \min\{c_f(u,v) : (u,v) \in p\}$
    for each edge $(u,v) \in p$ **do**
      if $(u,v) \in E$ **then**
        $f[u,v] \leftarrow f[u,v] + c_{min}(p)$ ▷ adding the new flow
      else
        $f[v,u] \leftarrow f[v,u] - c_{min}(p)$
      end if
      $c_f(u,v) \leftarrow c_f(u,v) - c_{min}(p)$ ▷ Update Residual Graph, G_f
      $c_f(v,u) \leftarrow c_f(v,u) + c_{min}(p)$
    end for
  end while
end function

- $c_f(j,w) = c(j,w)$: same as the original network G
- $c_f(j,w)$ is non-zero
- $c_{min}(p)$ is always 1 for matching problem here
- Remove a flow in opposite direction

Next…No new flow. Done!



Network Graph G



Residual Graph G_f

Algorithm 2 Ford-Fulkerson

$$c_f(j,w) = c(j,w) - f(j,w)$$

```
function FORD-FULKERSON(Graph G, Vertex s, Vertex t)
    for each edge (u, v) ∈ E[G] do                          ▷ Initialization of Flows
        f[u, v] ← 0
        f[v, u] ← 0
    end for
    while Finding a path from s to t in G_f do
        c_min(p) ← min{c_f(u, v) : (u, v) ∈ p}
        for each edge (u, v) ∈ p do
            if (u, v) ∈ E then
                f[u, v] ← f[u, v] + c_min(p)                ▷ adding the new flow
            else
                f[v, u] ← f[v, u] − c_min(p)
            end if
            c_f(u, v) ← c_f(u, v) − c_min(p)                ▷ Update Residual Graph, G_f
            c_f(v, u) ← c_f(v, u) + c_min(p)
        end for
    end while
end function
```

- $c_f(j,w) = c(j,w)$: same as the original network G
- $c_f(j,w)$ is non-zero
- Remove a flow in opposite direction

Summary

- Ford Fulkerson Method
  - Maximum Flow Problem
  - Maximum Matching Problem
- Finding the augmenting paths
  - BFS or DFS or any graph traversal

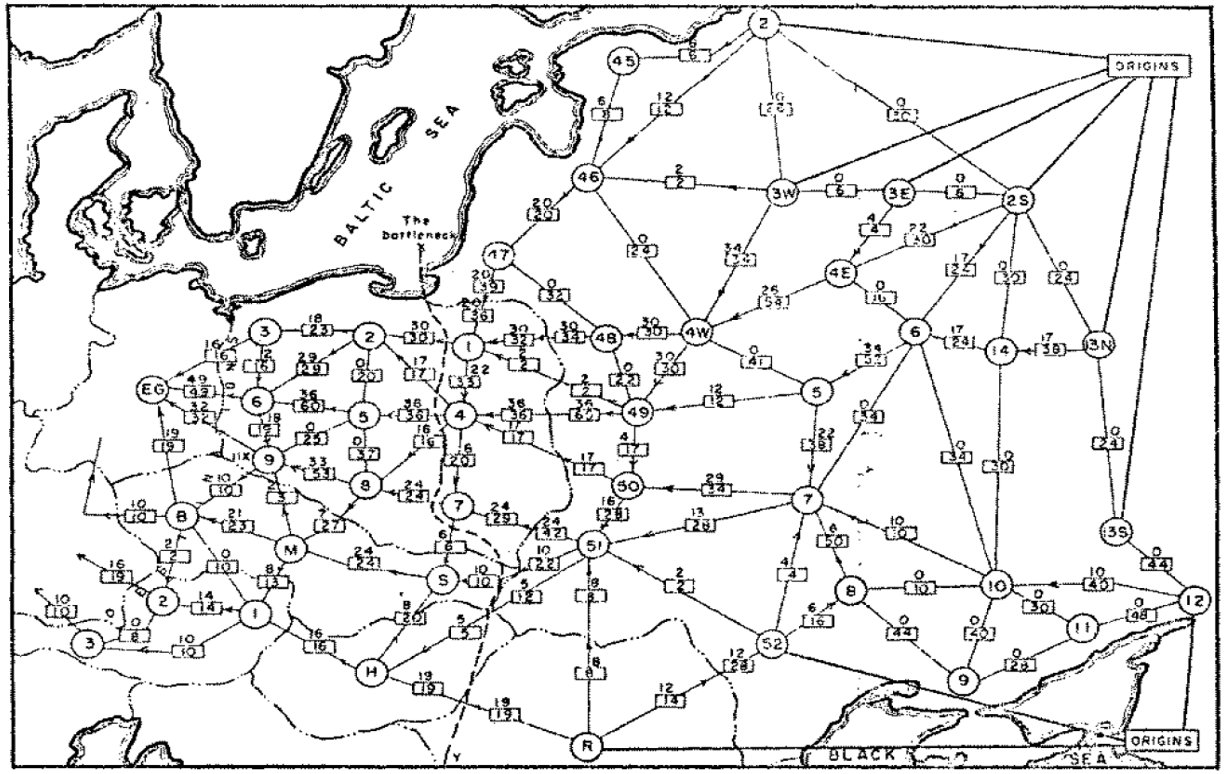
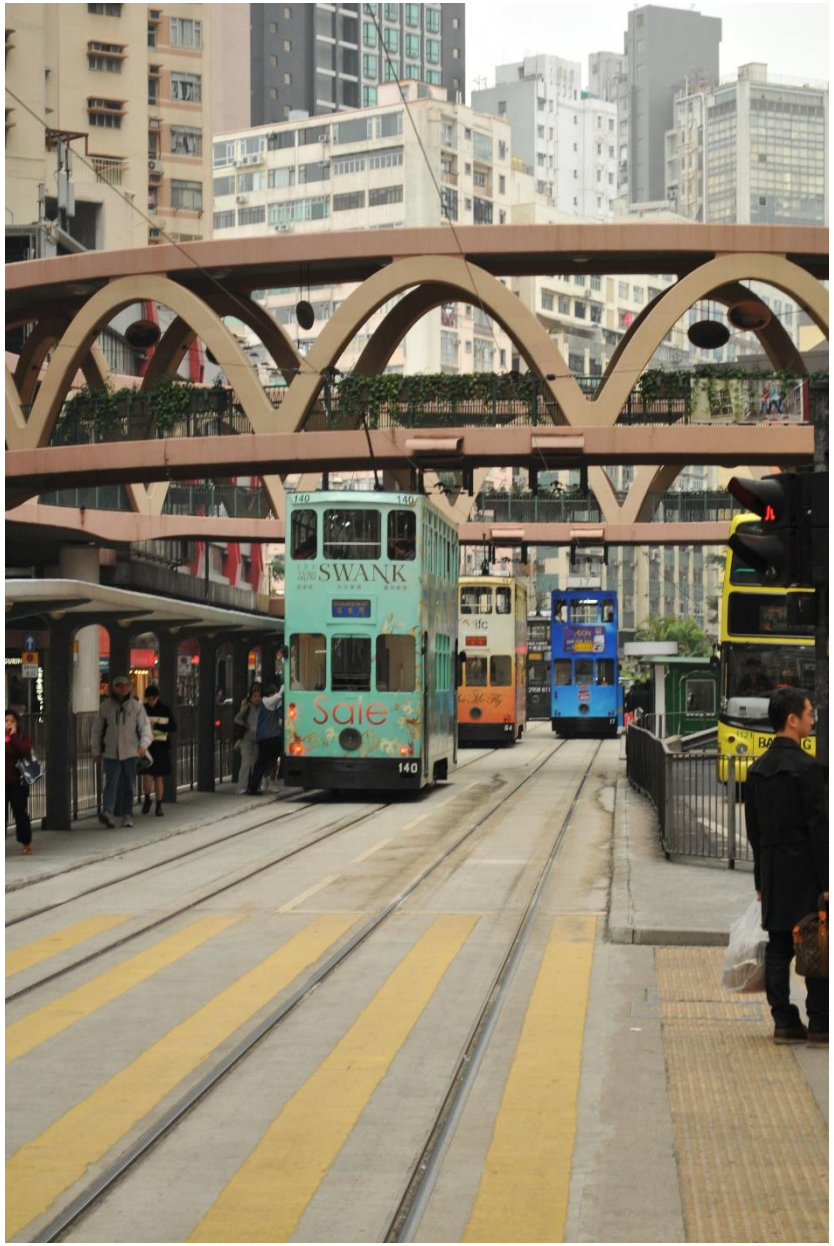


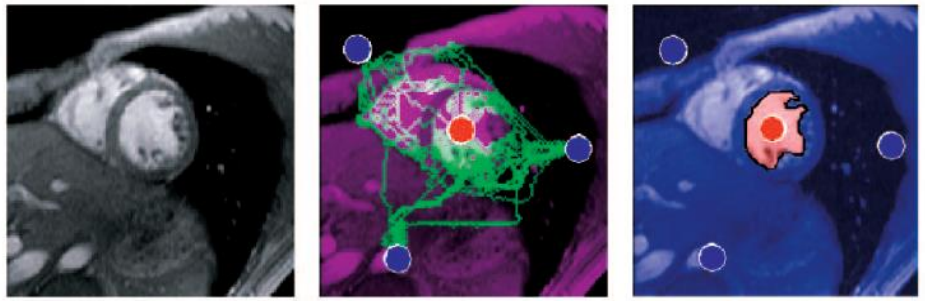
Figure 2

From Harris and Ross [1955]: Schematic diagram of the railway network of the Western Soviet Union and Eastern European countries, with a maximum flow of value 163,000 tons from Russia to Eastern Europe, and a cut of capacity 163,000 tons indicated as "The bottleneck".

- Matching problem is a {0,1} weighted graph or an unweighted graph
  - Implementation is easier

Source: On the history of the transportation and maximum flow problems. Alexandar Schrijver in Math Programming, 91:3,2002





(a) (b) (c)

Fig. 3. Graph cut/flow example in the context of image segmentation in Section 4.4. Red and blue seeds are "hard-wired" to the source s and the sink t, correspondingly. As usual, the cost of edges between the pixels (graph nodes) is set to low values in places with high intensity contrast. Thus, cuts along object boundaries in the image should be cheaper. Weak edges also work as "bottlenecks" for a flow. In (b), we show a maximum flow from s to t. In fact, it saturates graph edges corresponding to a minimum cut boundary in (c). (a) Original image. (b) A maximum flow. (c) A minimum cut.

Source: An Experimental Comparison of Min-Cut/ Max-Flow Algorithms for Energy Minimization in Vision
Yuri Boykov and Vladimir Kolmogorov, TPAMI, Vol. 26 No. 9, 2004


Q&A