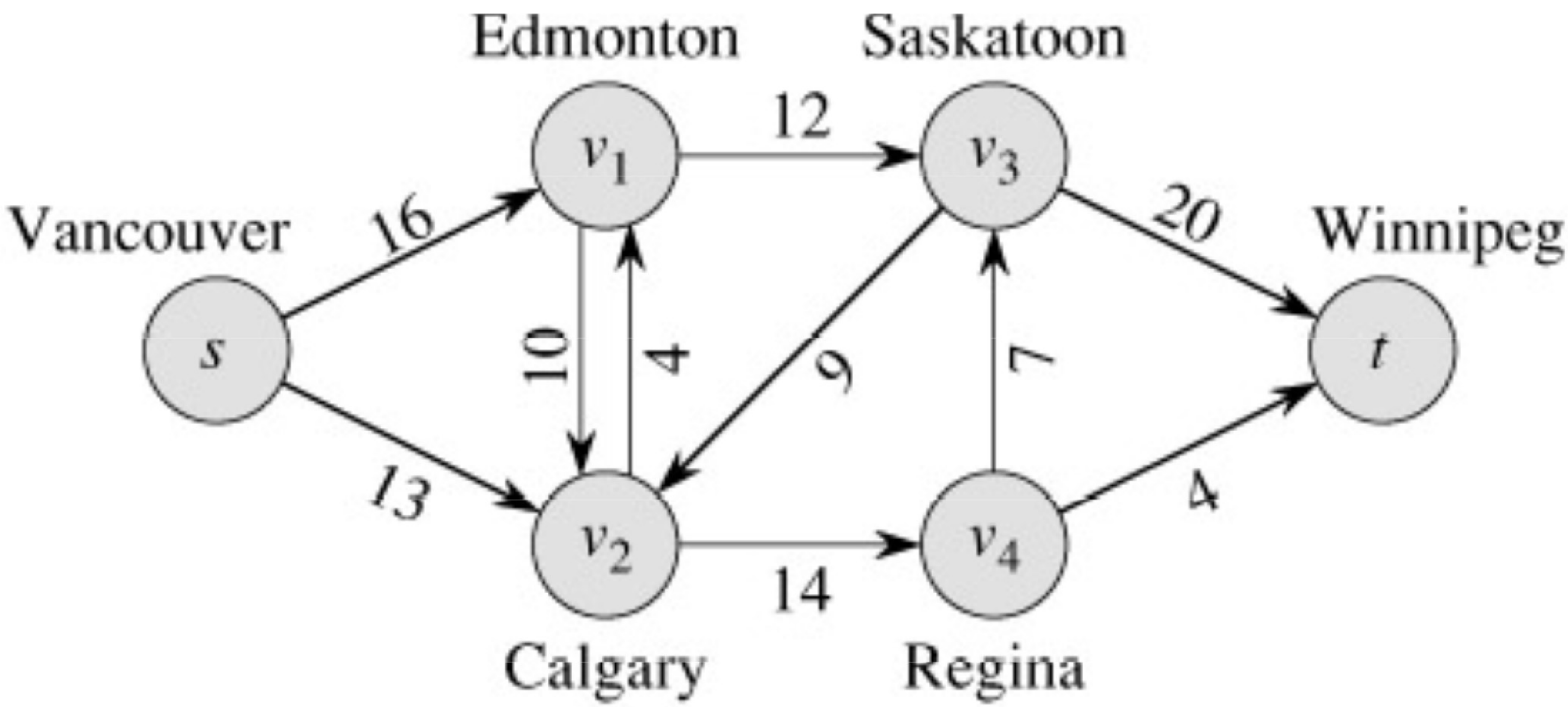
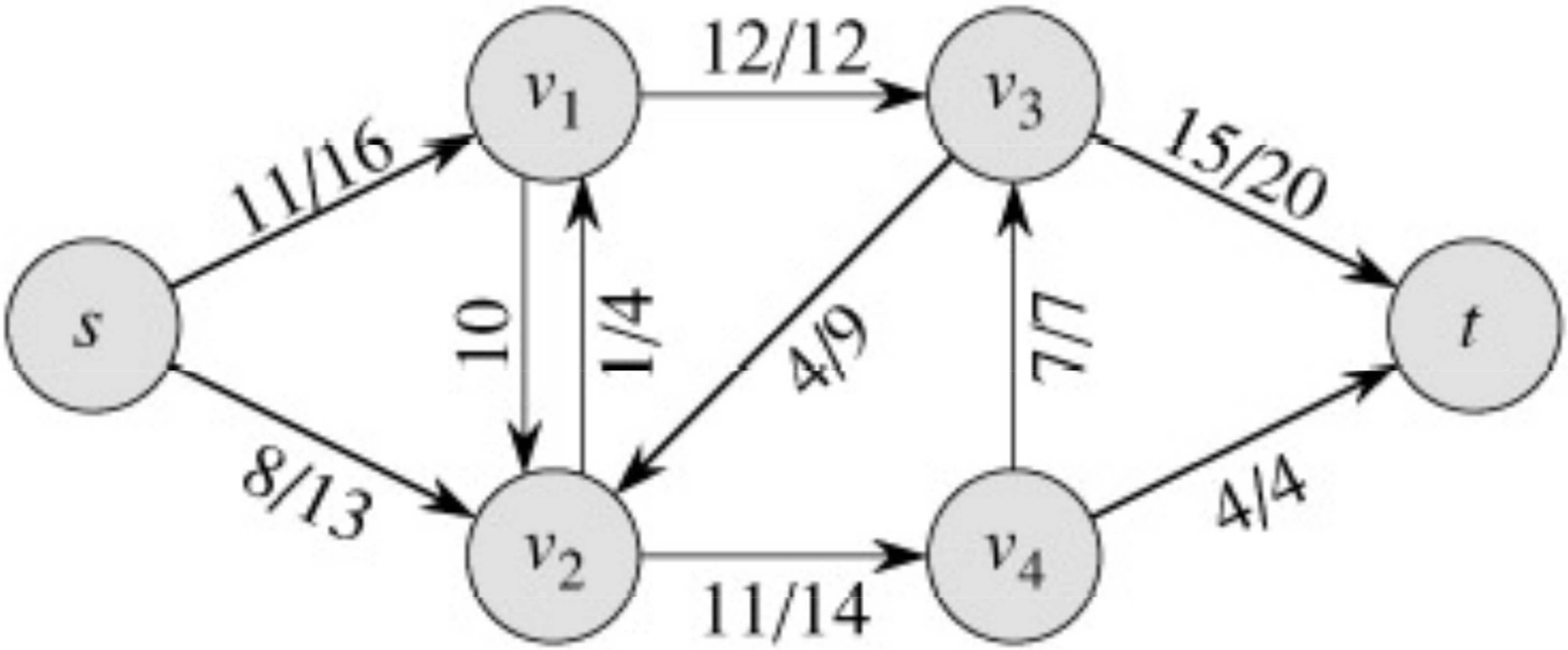
Flux maxim în rețele de transport


Edmonton
Saskatoon
Vancouver
Winnipeg
Calgary
Regina
v_1
v_2
v_3
v_4
s
t
16
13
12
14
10
4
9
7
20
4



v1
v2
v3
v4
s
t
11/16
8/13
10
1/4
12/12
4/9
11/14
7/7
15/20
4/4

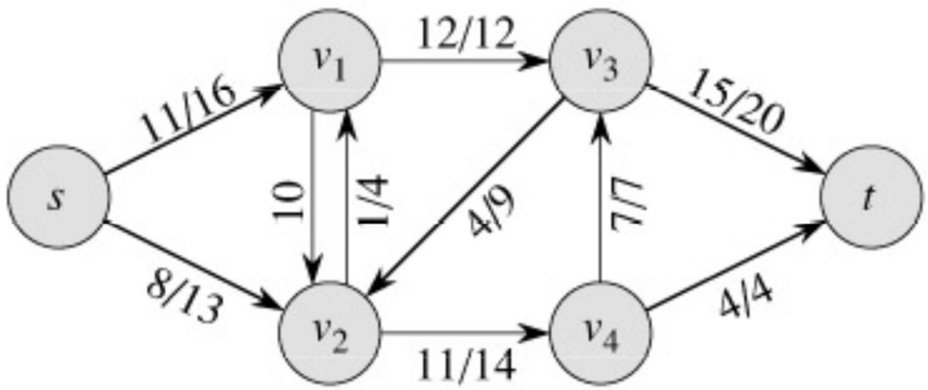
Definiții și proprietăți

- Rețea de transport
  - Graf orientat
  - Capacitate: c(u,v)
  - Sursă și scurgere: s,t
- Flux: f(u,v)
- Proprietăți
  - Restricție de capacitate: f(u,v) $\leq$ c(u,v)
  - Antisimetrie: f(u,v) =- f(v,u)
  - Conservare flux $\sum_{e.in.v} f(e) = \sum_{e.out.v} f(e)$
- Valoarea fluxului

Fluxul în rețea

$$|f| = \sum_{v \in V} f(s, v) = \sum_{v \in V} f(v, t)$$

Exemplu |f|



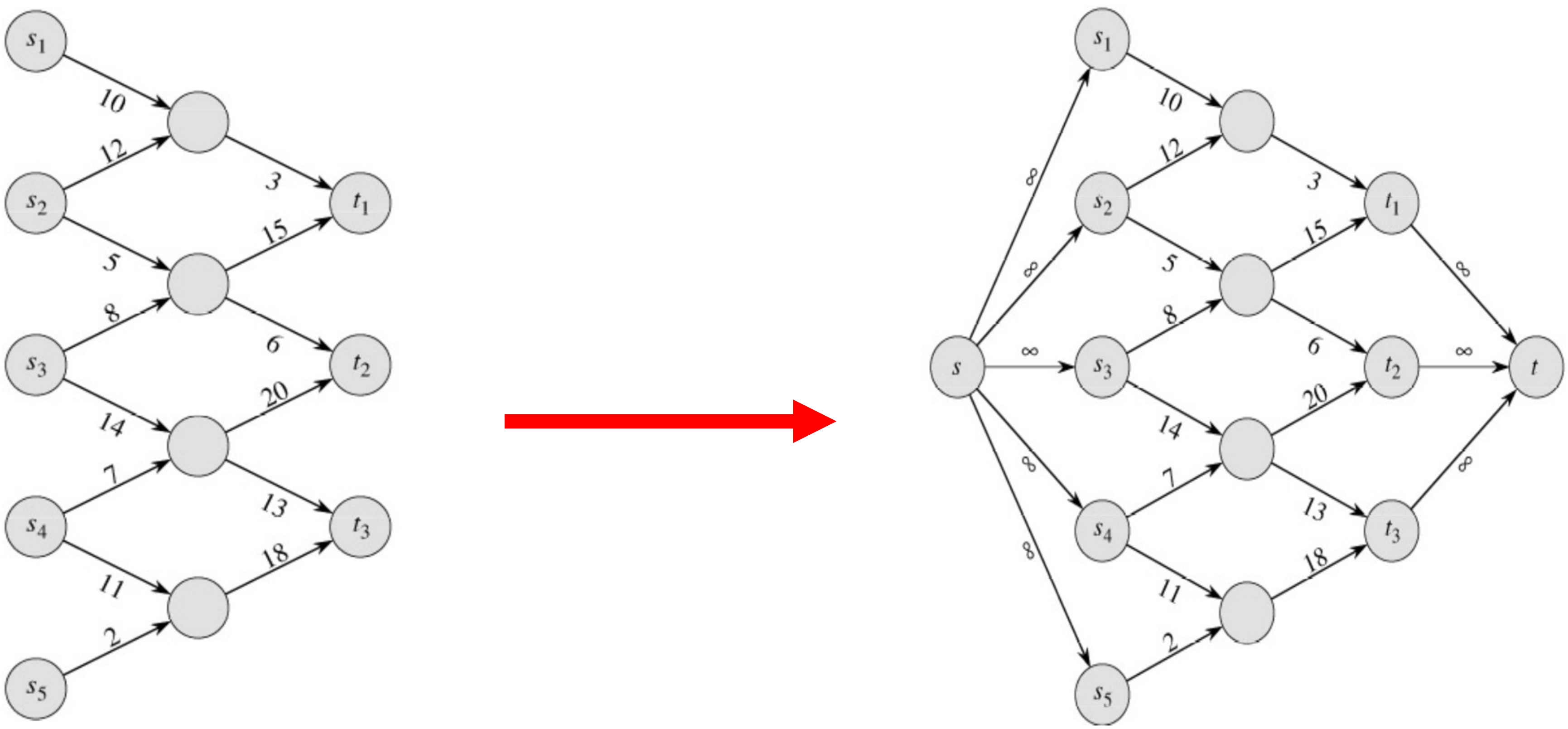
$|f| = f(s, v_1) + f(s, v_2) + f(s, v_3) + f(s, v_4) + f(s, t) =$

$11 + 8 + 0 + 0 + 0 = 19$

$|f| = f(s, t) + f(v_1, t) + f(v_2, t) + f(v_3, t) + f(v_4, t) =$

$0 + 0 + 0 + 15 + 4 = 19$

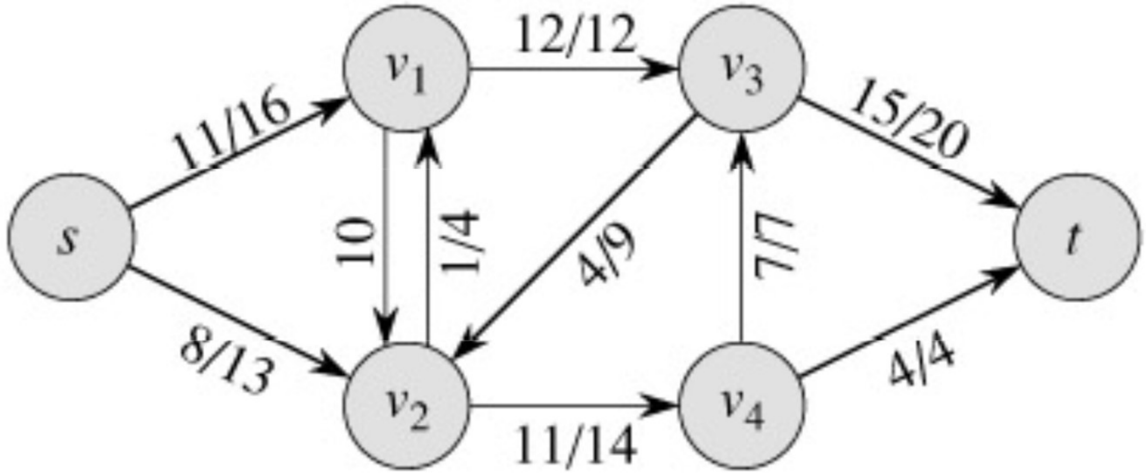
Mai multe surse și destinații

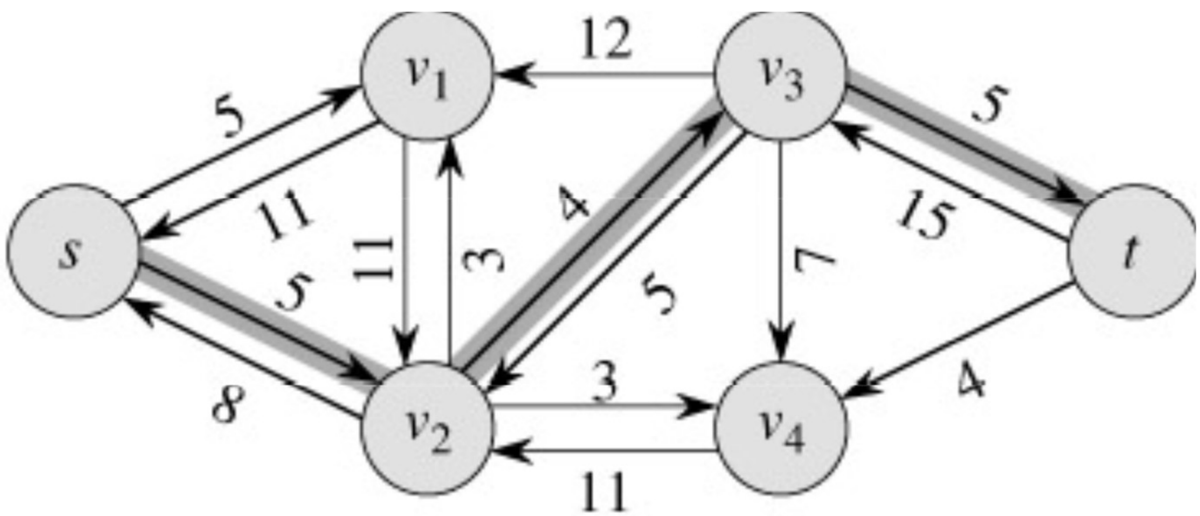


Rețeaua reziduală a lui G indusă de f conține arcele NxN pentru care valoarea de mai jos (capacitatea arcului respectiv) este pozitivă

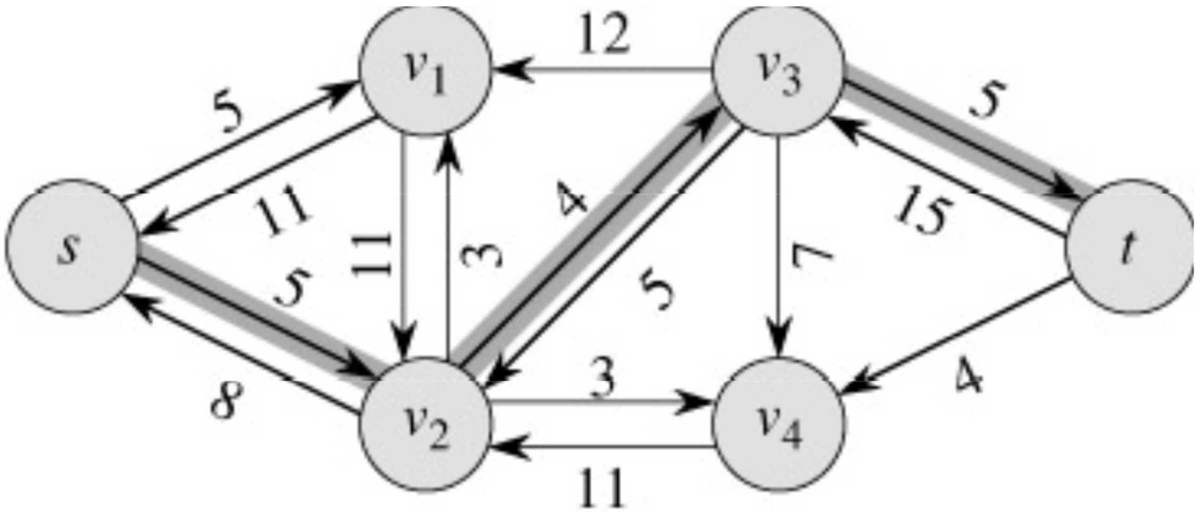
$$c_f(u,v) = c(u,v) - f(u,v)$$

Exemplu de rețea reziduală



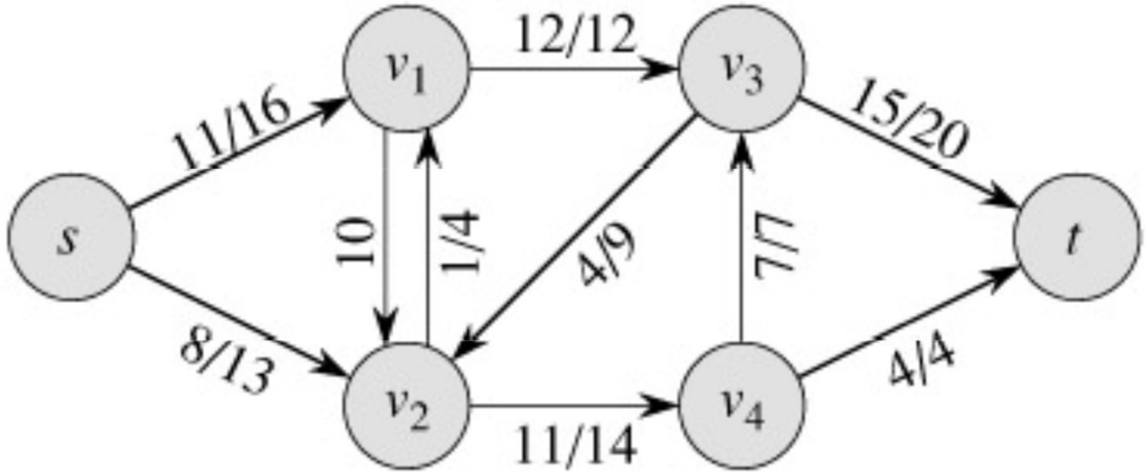


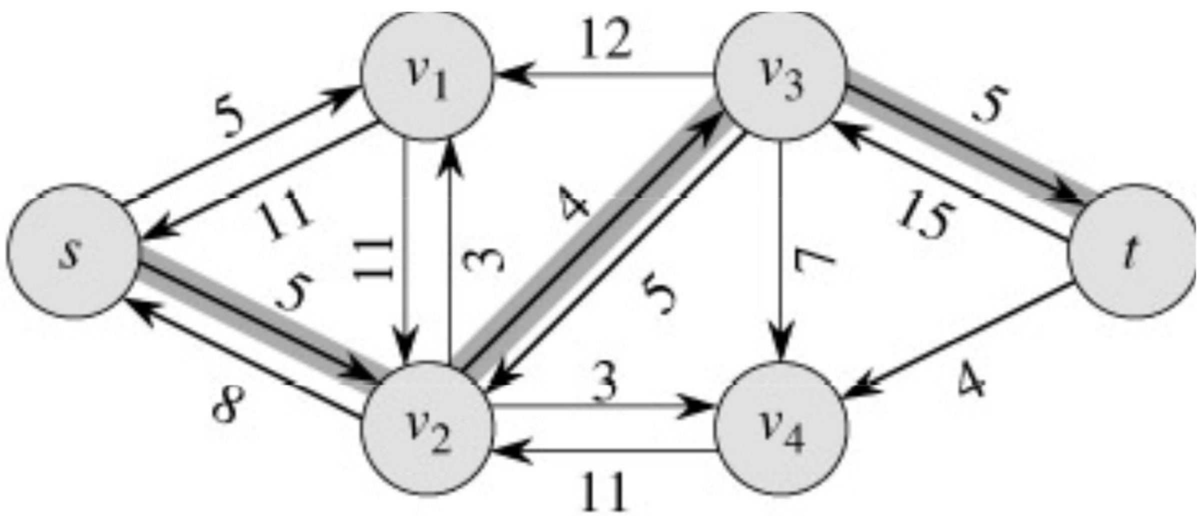
Drum de ameliorare



$$c_f(p) = \min\{c_f(u,v) : (u,v) \text{ este în } p\}$$

Capacitatea drumului de ameliorare=4



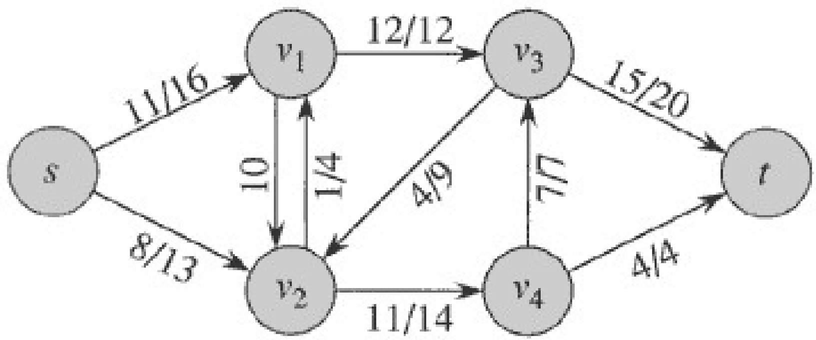


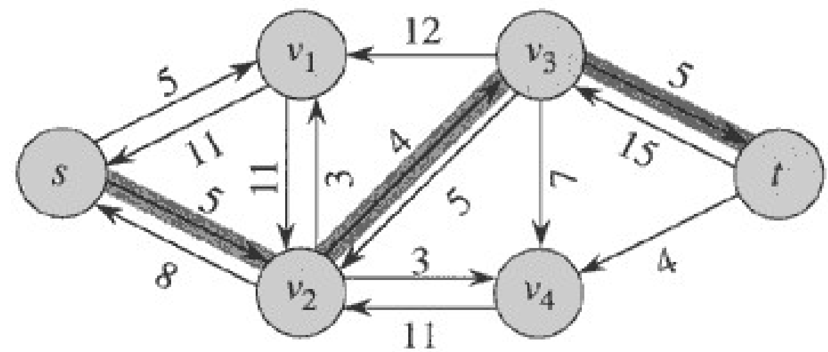
Metoda Ford-Fulkerson

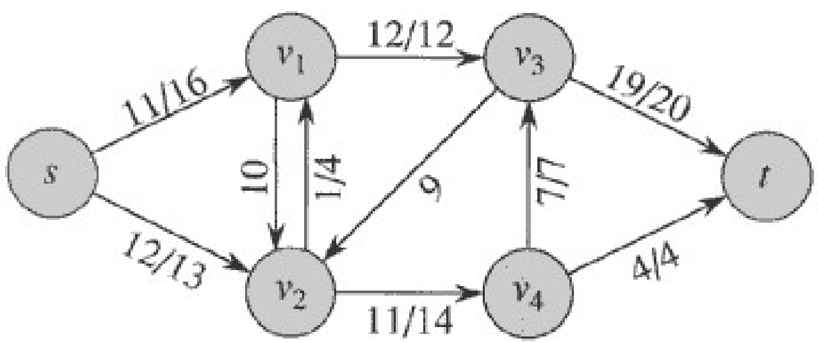
Cât timp există un drum de ameliorare

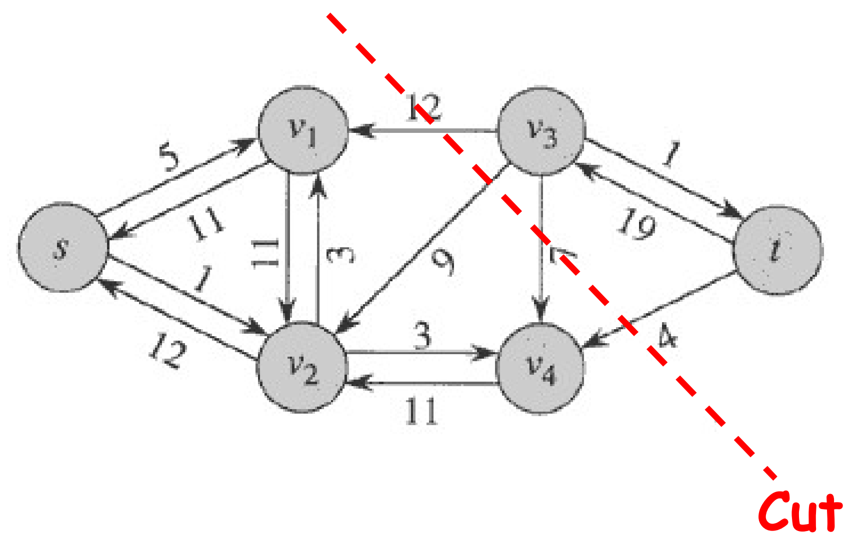
repetă mărește fluxul de-a lungul lui

Exemplu

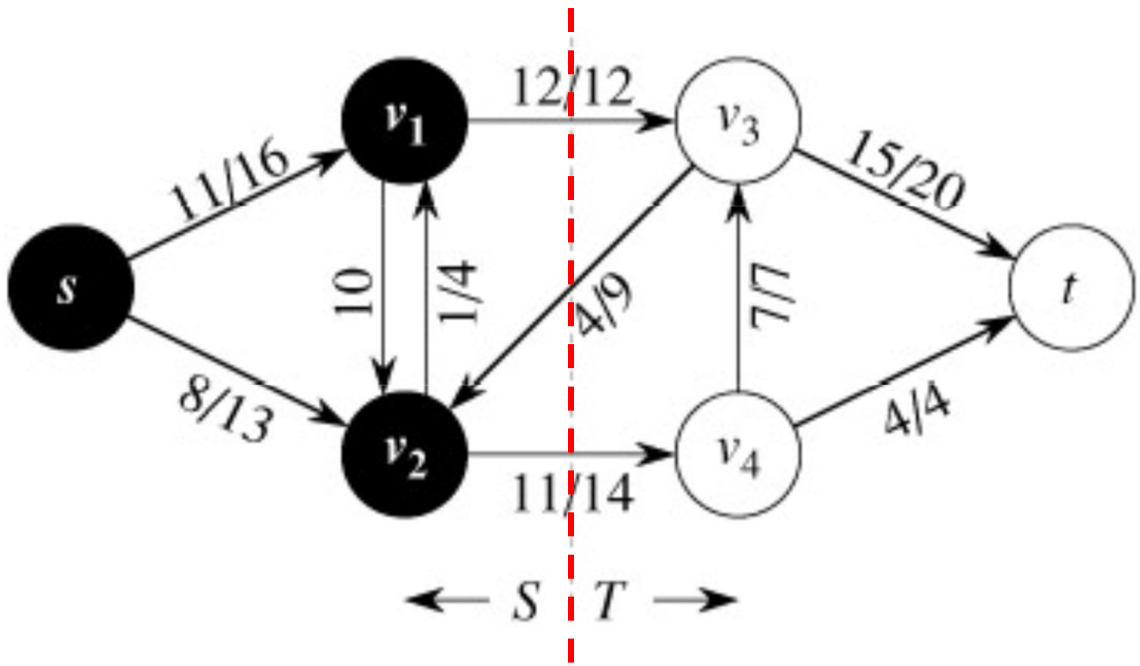

s
v1
v2
v3
v4
t
11/16
8/13
12/12
10
1/4
4/9
7/7
11/14
15/20
4/4


s
v1
v2
v3
v4
t
5
11
5
8
12
11
3
4
5
7
3
11
5
15
4


s
v1
v2
v3
v4
t
11/16
12/13
12/12
10
1/4
9
7/7
11/14
19/20
4/4

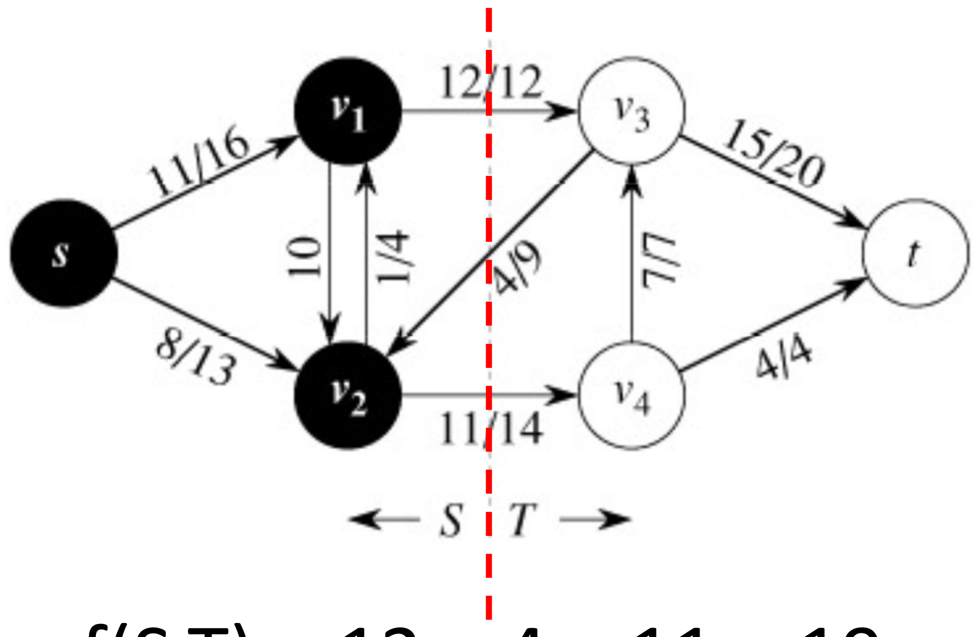

s
v1
v2
v3
v4
t
5
11
1
12
12
11
3
9
7
3
11
1
19
4
Cut

Tăietură



Fluxul prin tăietura (S,T)

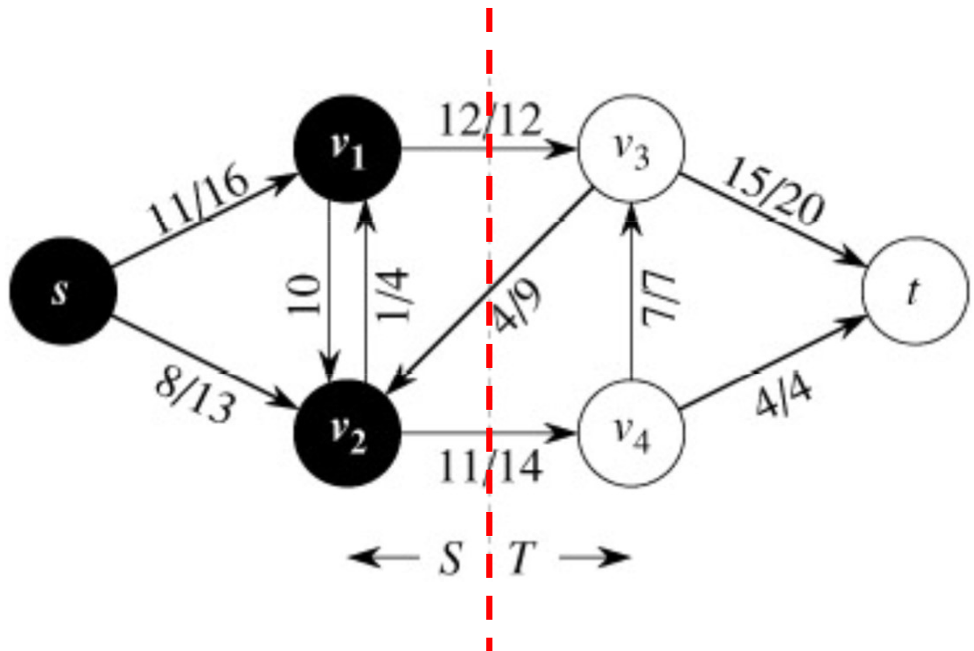
$$f(S,T) = \sum_{u \in S, v \in T} f(u,v)$$



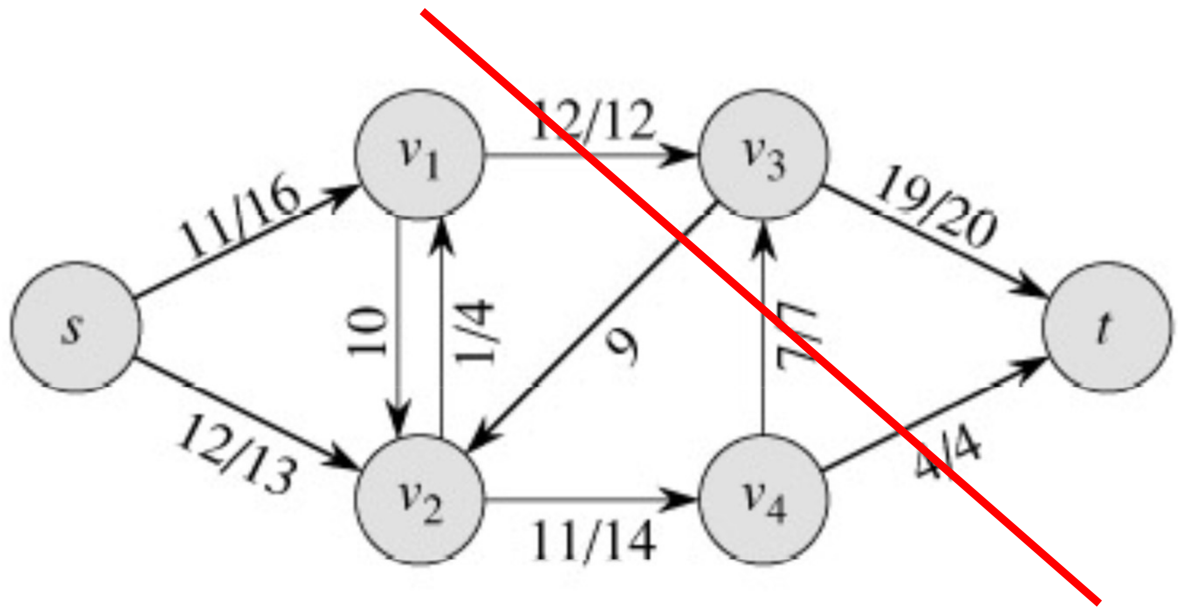
f(S,T) = 12 – 4 + 11 = 19

Capacitatea unei tăieturi (S,T)

$$c(S,T) = \sum_{u \in S, v \in T} c(u,v)$$

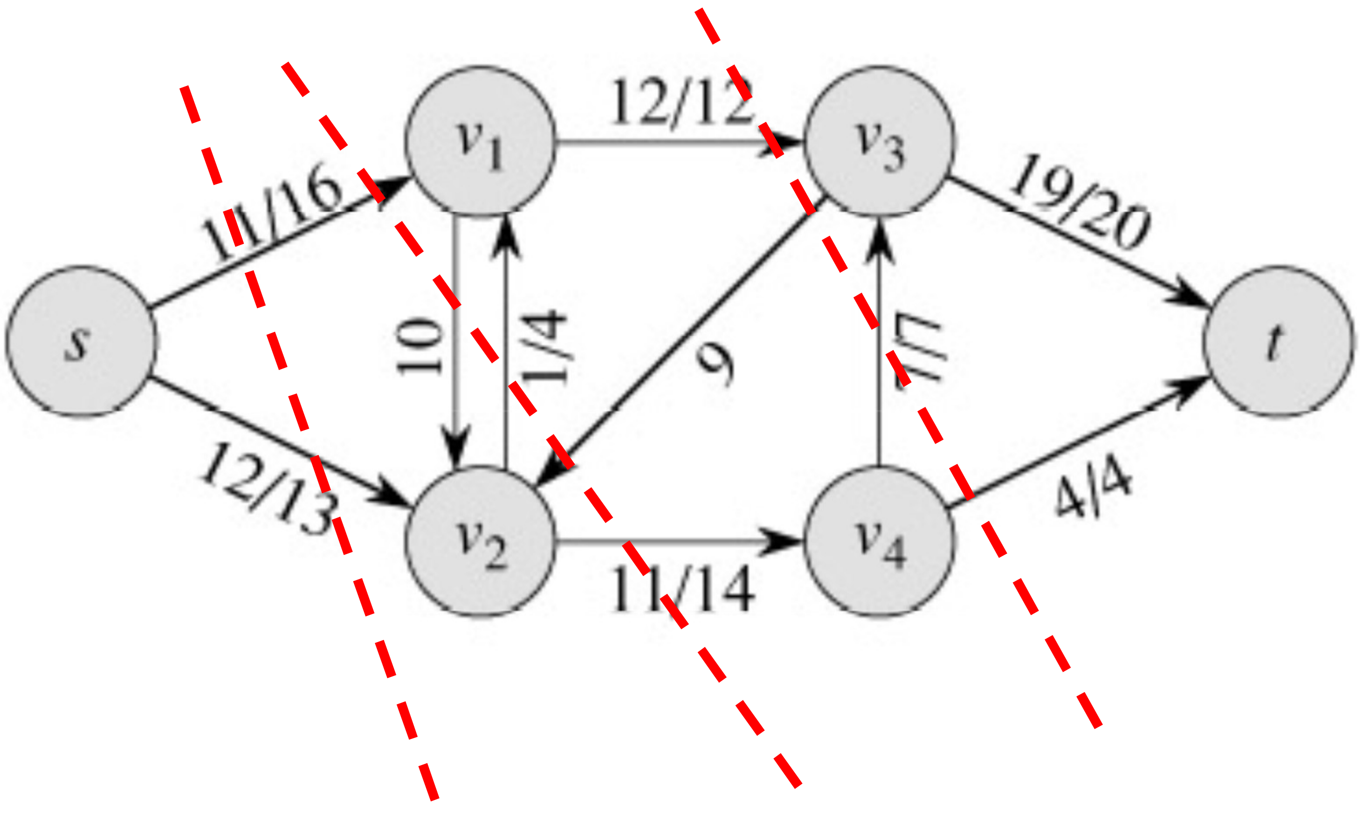


c(S,T)= 12+ 0 + 14 = 26



Capacitatea maximă a rețelei este de cel mult 12 + 7 + 4 = 23 (tăietura minimă)

Fluxul

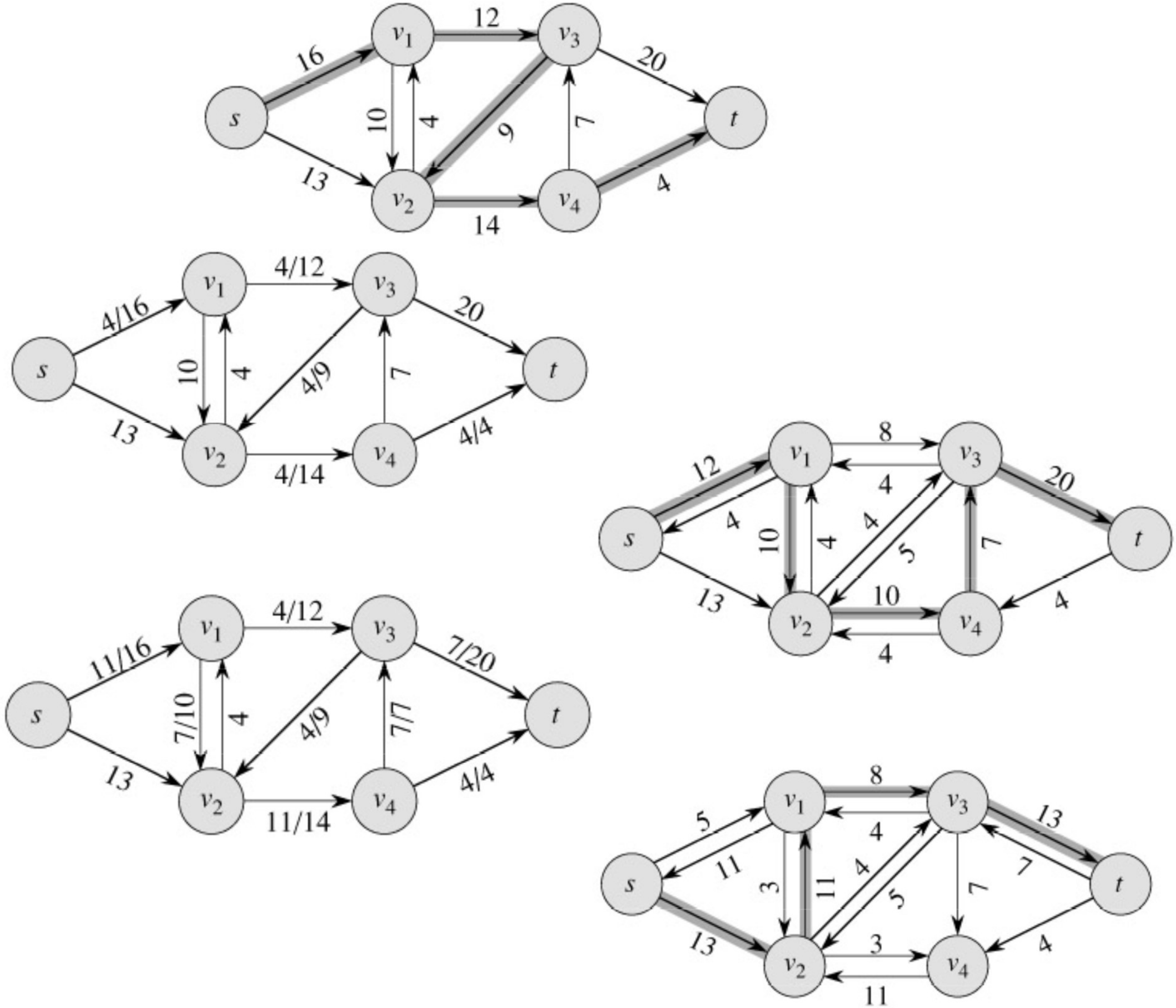


Teorema de flux maxim, tăietură minimă

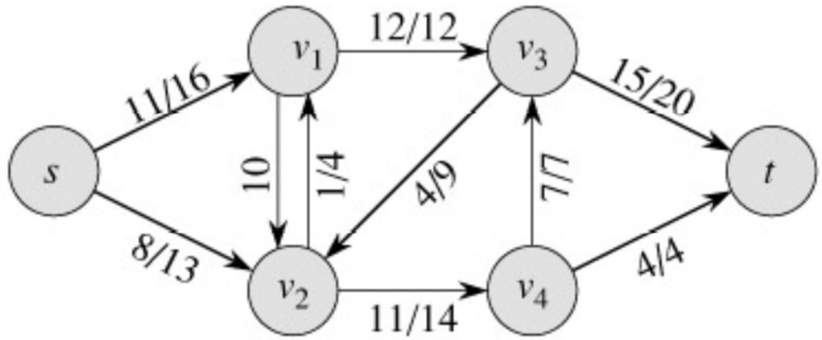
- Dacă *f* este un flux în *G=(N,A)*, cu sursă *s* și scurgere *t*, atunci condițiile sunt echivalente:
  1. *f* este un flux maxim în *G*.
  2. Rețeaua reziduală nu conține căi de ameliorare.
  3. $|f| = c(S,T)$ pentru o tăietură *(S,T)* (minimă).

Algoritmul Ford-Fulkerson

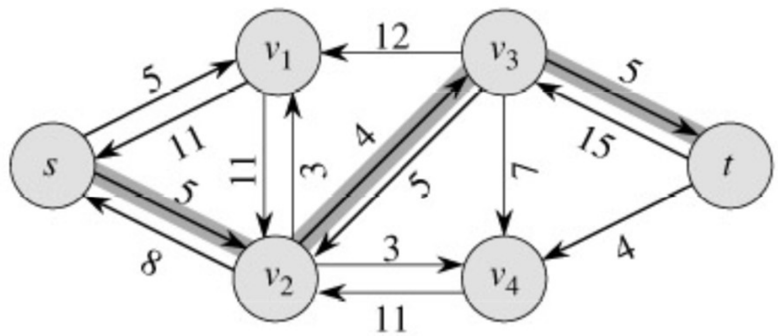
```
Algoritm Ford-Fulkerson(G,s,t)
    pentru fiecare (u,v) din A
        f[u,v]←0
        f[v,u]←0
    cât timp există o cale p de la s la t în rețeaua reziduală G_f
        repetă
            c_f(p)←min{c_f(u,v) : (u,v) în p}
            pentru fiecare (u,v) în p repetă
                f[u,v]←f[u,v]+c_f(p)
                f[v,u]←- f[u,v]
```



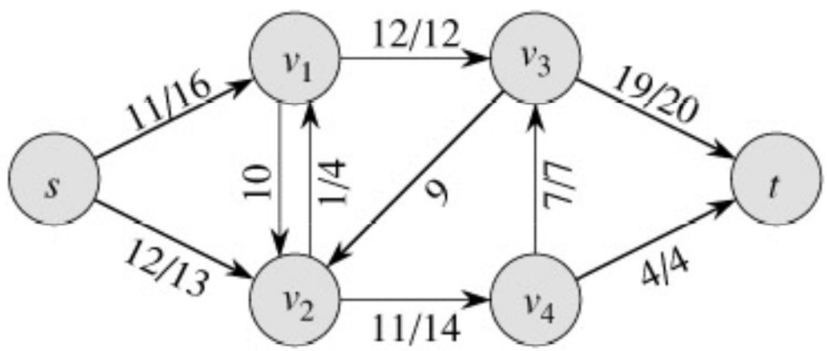
s
v1
v2
v3
v4
t
16
13
10
4
12
9
14
7
20
4
4/16
13
10
4
4/12
4/9
4/14
7
20
4/4
12
4
13
10
4
8
4
4
5
10
4
7
20
4
11/16
13
7/10
4
4/12
4/9
11/14
7/7
7/20
4/4
5
11
13
3
11
8
4
4
5
3
11
7
13
7
4



s
v1
v2
v3
v4
t
11/16
8/13
10
1/4
12/12
4/9
7/7
11/14
15/20
4/4



s
v1
v2
v3
v4
t
5
11
5
8
11
3
12
4
5
7
3
11
5
15
4



s
v1
v2
v3
v4
t
11/16
12/13
10
1/4
12/12
9
7/7
11/14
19/20
4/4

Analiză

Algoritm Ford-Fulkerson(G,s,t)

    pentru fiecare (u,v) din A

        f[u,v]←0

        f[v,u]←0

O(E)

?

    cât timp există o cale p de la s la t în rețeaua reziduală G_f **repetă**

        $c_f(p)$←min{$c_f(u,v)$: (u,v) în p}

        pentru fiecare (u,v) în p **repetă**

            f[u,v]←f[u,v]+$c_f(p)$

            f[v,u]←- f[u,v]

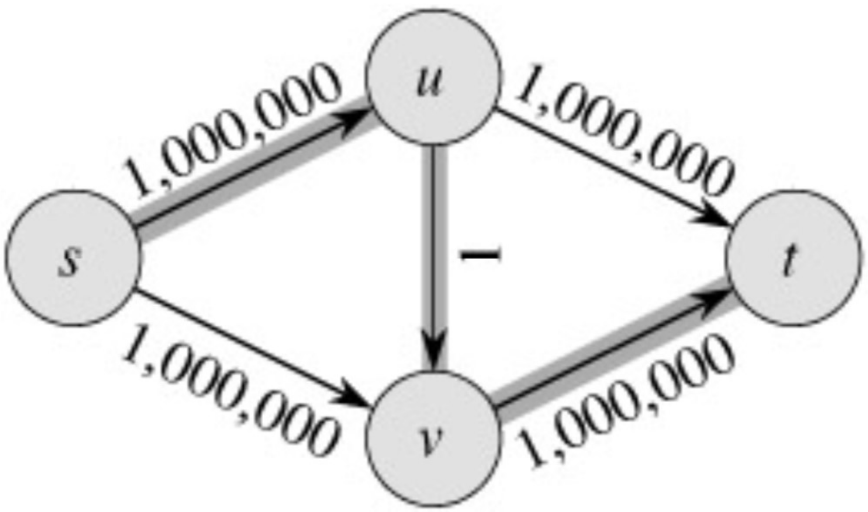
O(E)

Analiză

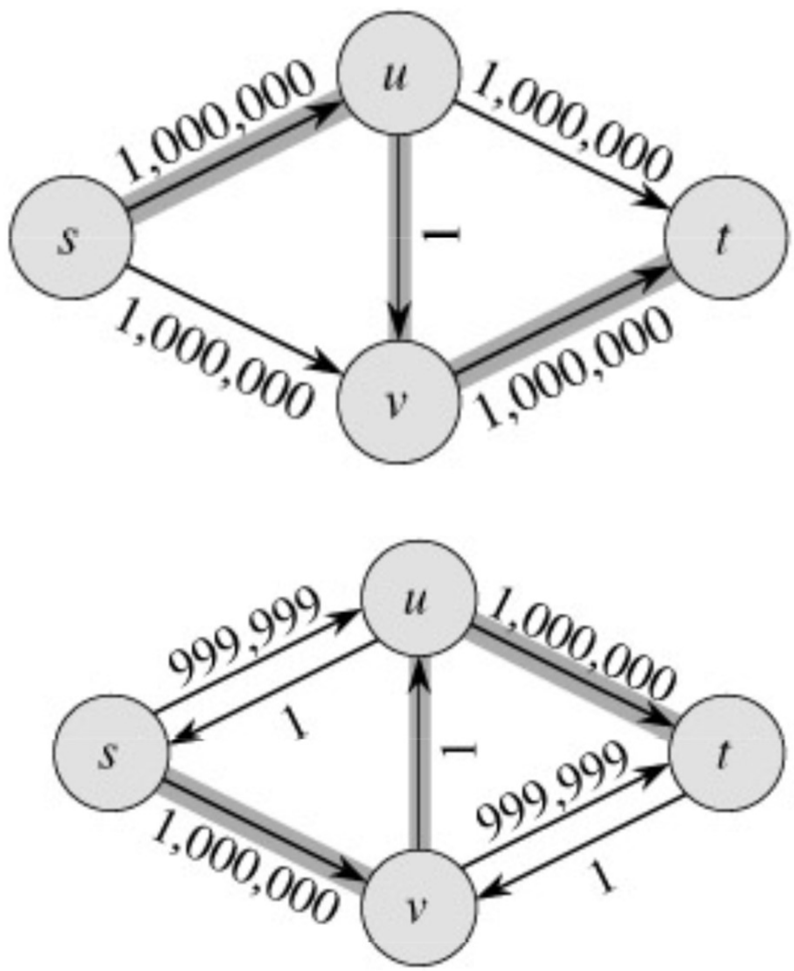
- Capacitati intregi – marire a |f| cu ≥ 1.
- Daca fluxul max e f*, atunci ≤ |f*| iteratii
  → O(a|f*|).
- Observati ca timpul de rulare este **nepolinomial** fata de marimea intrarii. Depinde de |f*|, care nu e o functie de |n| sau |a|.
- Daca sunt capacitati rationale, se pot scala la intregi
- Daca sunt irationale, FORD-FULKERSON s-ar putea sa nu se termine!

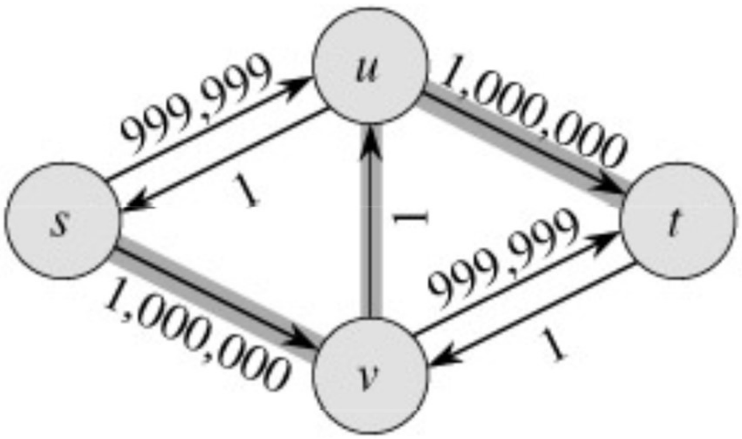
Ford-Fulkerson

- O (a |f*|)
- Nepolinomial

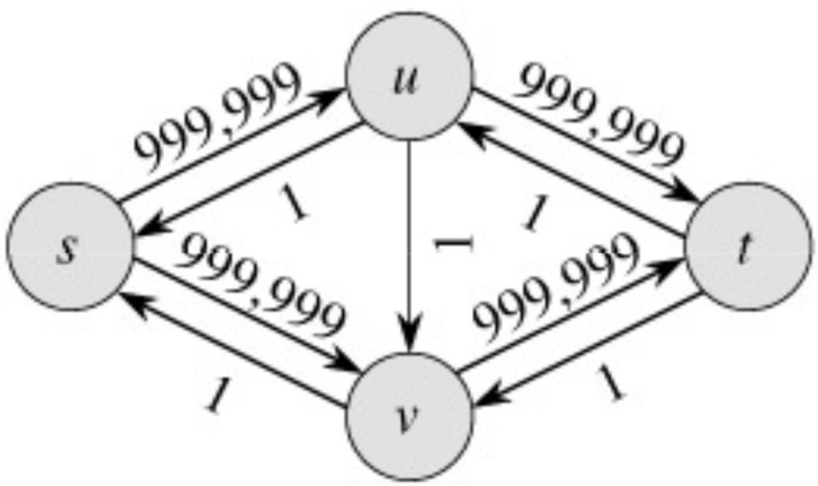


|f*|=2,000,000

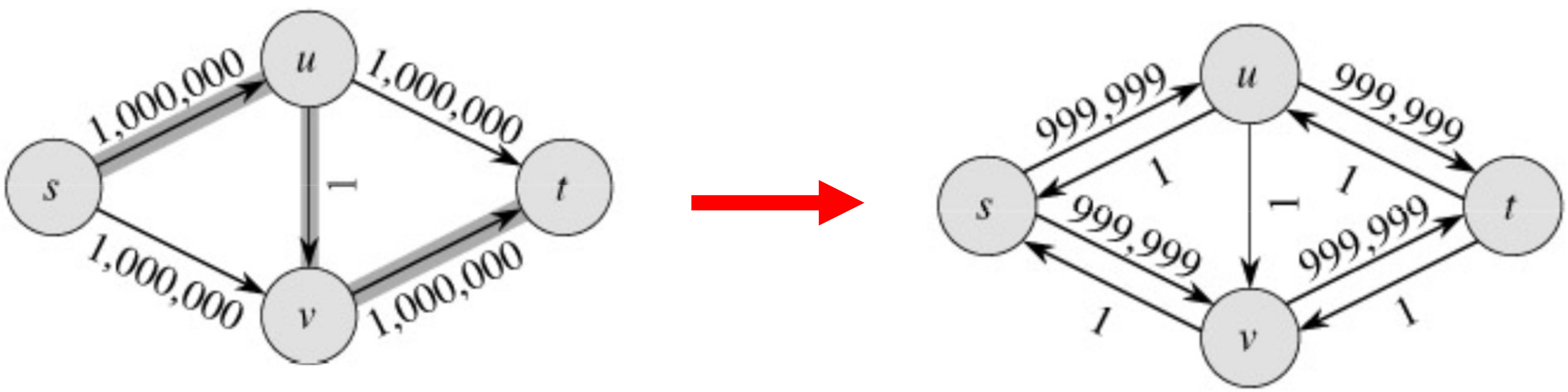

u
s
t
v
1,000,000
1,000,000
1
1,000,000
1,000,000
u
s
t
v
999,999
1
1,000,000
1
999,999
1,000,000
1



u
999,999
1,000,000
1
s
1
t
999,999
1,000,000
v
1



u
999,999
999,999
1
1
s
999,999
1
t
999,999
1
v
1



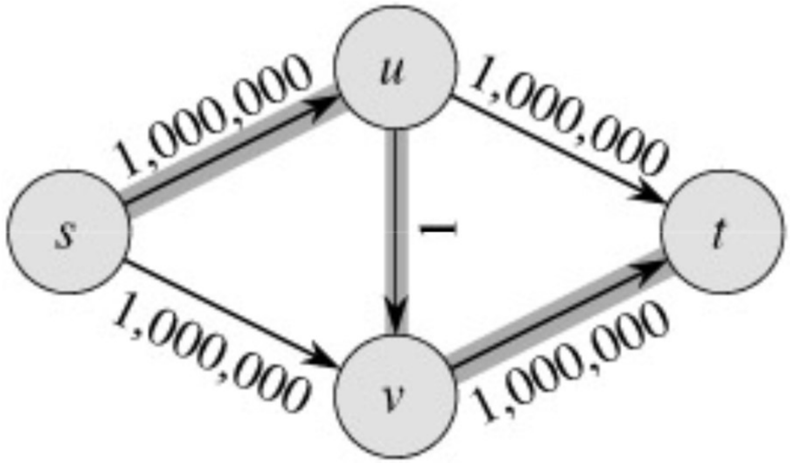
- Repetă de 999,999 ori

Edmonds-Karp

- Căutare în lățime pe rețeaua reziduală →Calea de ameliorare este cea mai scurtă cale în rețeaua reziduală
- $O(na^2)$

Algoritm Ford-Fulkerson(G,s,t)
  pentru fiecare (u,v) din A
    f[u,v]←0
    f[v,u]←0
  cât timp există o cale p de la s la t în rețeaua reziduală G_f
    repetă
      $c_f(p) \leftarrow \min\{c_f(u,v) : (u,v) \text{ în } p\}$
      pentru fiecare (u,v) în p **repetă**
        $f[u,v] \leftarrow f[u,v]+c_f(p)$
        f[v,u]←- f[u,v]

Edmonds-Karp



- 2 iterații

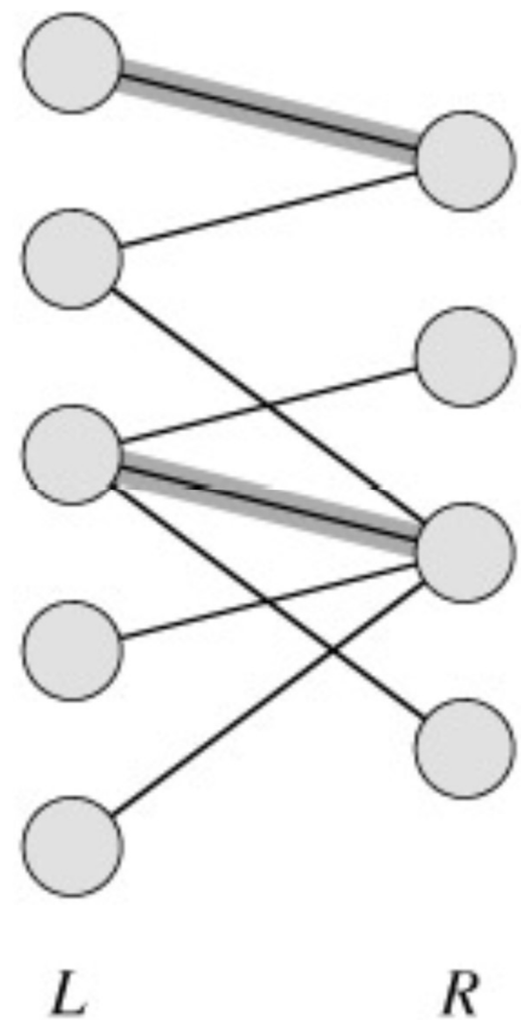
Alte imbunatatiri

- Push-relabel algorithm ([CLRS, 26.4]) – $O(V^2 E)$.
- Relabel-to-front algorithm ([CLRS, 26.5) – $O(V^3)$.

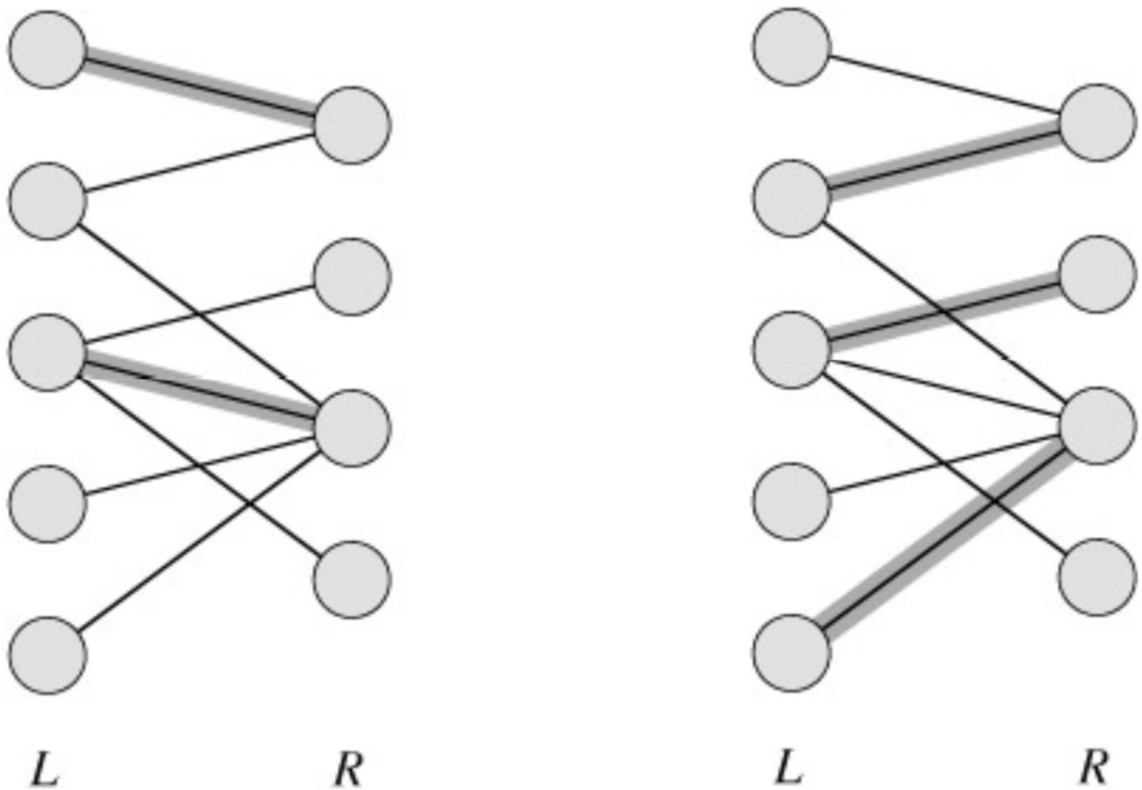
Maximum Bipartite Matching

- A bipartite graph is a graph *G=(V,E)* in which *V* can be divided into two parts *L* and *R* such that every edge in *E* is between a vertex in *L* and a vertex in *R*.

- e.g. vertices in *L* represent skilled workers and vertices in *R* represent jobs. An edge connects workers to jobs they can perform.

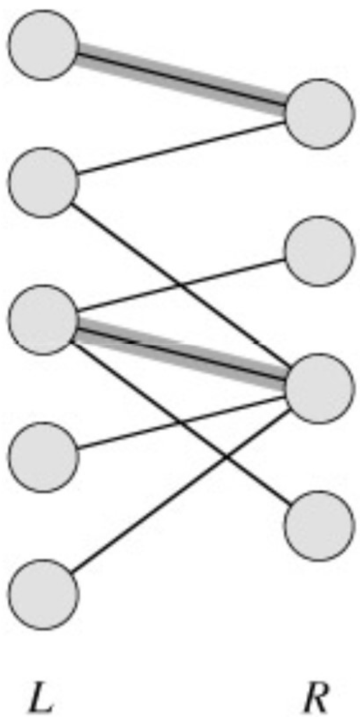


- A matching in a graph is a subset M of E, such that for all vertices v in V, at most one edge of M is incident on v.

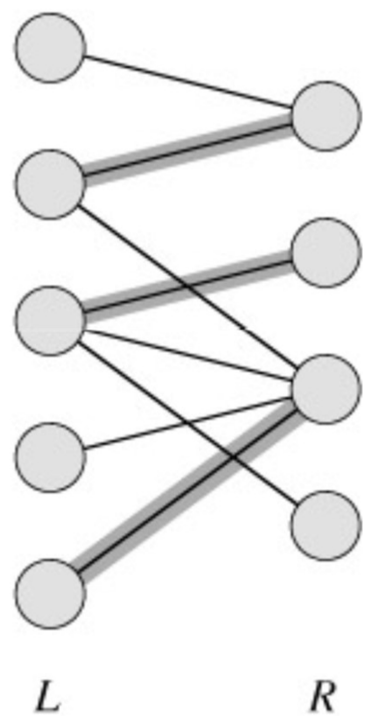


- A **maximum matching** is a matching of maximum cardinality (maximum number of edges).

not maximum

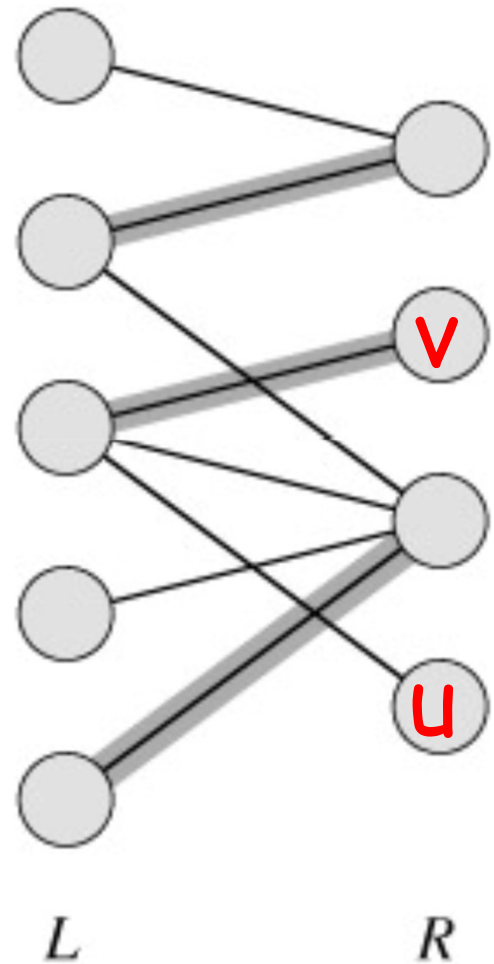


maximum



A Maximum Matching

- No matching of cardinality 4, because only one of v and u can be matched.
- In the workers-jobs example a max-matching provides work for as many people as possible.

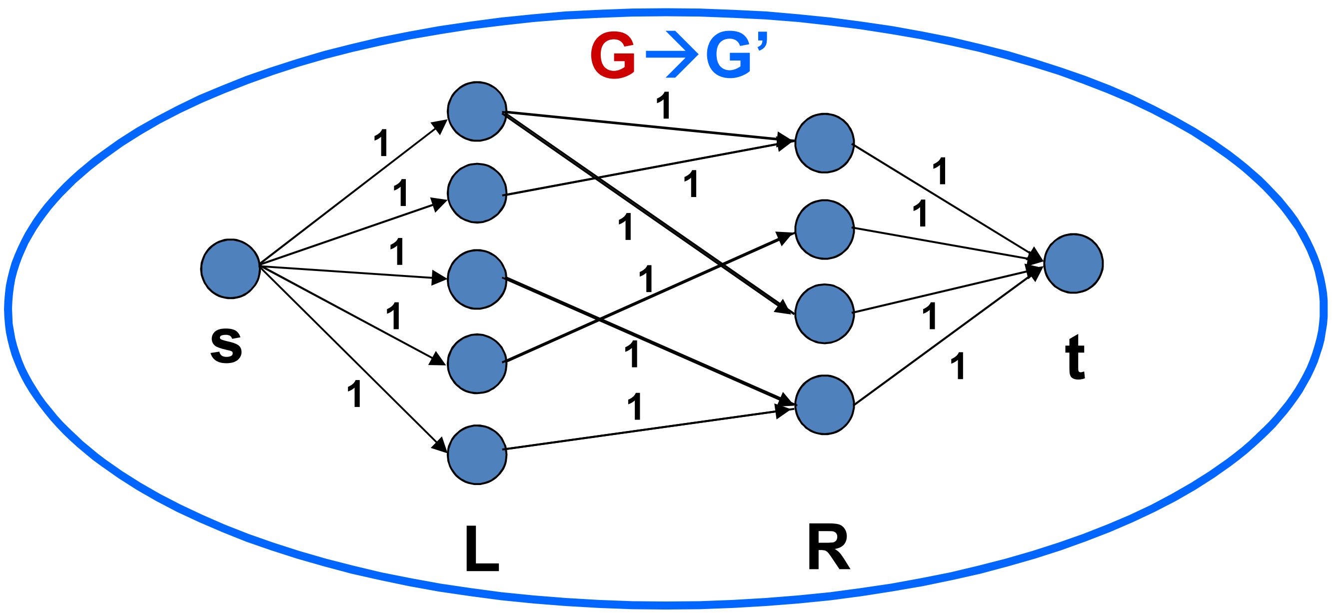


Solving the Maximum Bipartite Matching Problem

- Reduce the maximum bipartite matching problem on graph **G** to the max-flow problem on a corresponding flow network **G'**.
- Solve using Ford-Fulkerson method.

Corresponding Flow Network

- To form the corresponding flow network **G'** of the bipartite graph **G**:
  - Add a source vertex s and edges from s to L.
  - Direct the edges in E from L to R.
  - Add a sink vertex t and edges from R to t.
  - Assign a capacity of 1 to all edges.
- Claim: max-flow in **G'** corresponds to a max-bipartite-matching on **G**.



Solving Bipartite Matching as Max Flow

Let $G = (V, E)$ be a bipartite graph with vertex partition $V = L \cup R$.

Let $G' = (V', E')$ be its corresponding flow network.

If M is a matching in G,

then there is an integer-valued flow f in G' with value $|f| = |M|$.

Conversely if f is an integer-valued flow in G',

then there is a matching M in G with cardinality $|M| = |f|$.

Thus $\max |M| = \max(\text{integer } |f|)$

Does this mean that max $|f|$ = max $|M|$?

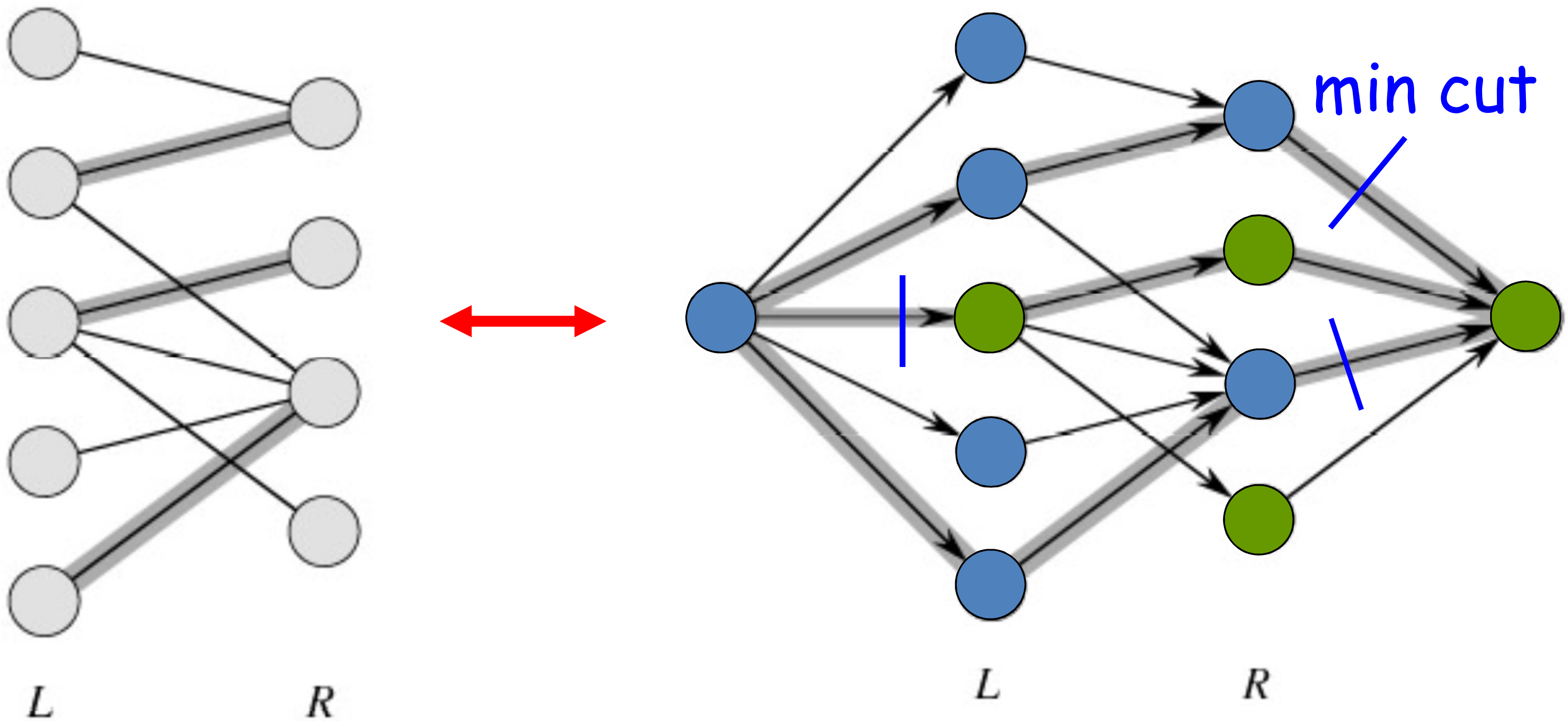
- Problem: we haven't shown that the max flow f(u,v) is necessarily integer-valued.

Integrality Theorem

- If the capacity function c takes on only integral values, then:
    1. The maximum flow f produced by the Ford-Fulkerson method has the property that |f| is integer-valued.
    2. For all vertices u and v the value f(u,v) of the flow is an integer.

- So max|M| = max |f|

Example



$|M| = 3 \quad \leftrightarrow \quad$ max flow $= |f| = 3$

Conclusion

- Network flow algorithms allow us to find the maximum bipartite matching fairly easily.

- Similar techniques are applicable in other combinatorial design problems.

Example

- In a department there are n courses and m instructors.
- Every instructor has a list of courses he or she can teach.
- Every instructor can teach at most 3 courses during a year.

- The goal: find an allocation of courses to the instructors subject to these constraints.