Design and Analysis of Algorithms

CSE 5311

Lecture 23 Maximum Flow

Junzhou Huang, Ph.D.

Department of Computer Science and Engineering

FLOW NETWORKS & FLOWS

- A flow network: a directed graph $G = (V, E)$
  - Two distinguished vertices : a source s and a sink t
  - Each edge has a nonegative capacity $c(u,v) \geq 0$

    (if $(u,v) \notin E$ then $c(u,v) = 0$)
  - for convenience : $\forall\ v \in V - \{ s,t \}, s \leadsto v \leadsto t,$

    i.e., every vertex v lies on some path from *s* to *t*.

FLOW NETWORKS & FLOWS

- *A positive flow p on G: a fn p:V×V* → $R_{\geq 0}$ satisfying
  - capacity constraint: $0 \leq p(u,v) \leq c(u,v)$, $\forall\ u,v \in V$
    - i.e., flow from one vertex to another cannot exceed the capacity
    - note : $p(u,v) > 0 \Rightarrow (u,v) \in E$ with $c(u,v) > 0$
  - flow conservation: Kirschoff's current law

$$\sum_{v \in V} p(u,v) - \sum_{v \in V} p(v,u) = 0 \quad \forall\ u \in V - \{s,t\}$$

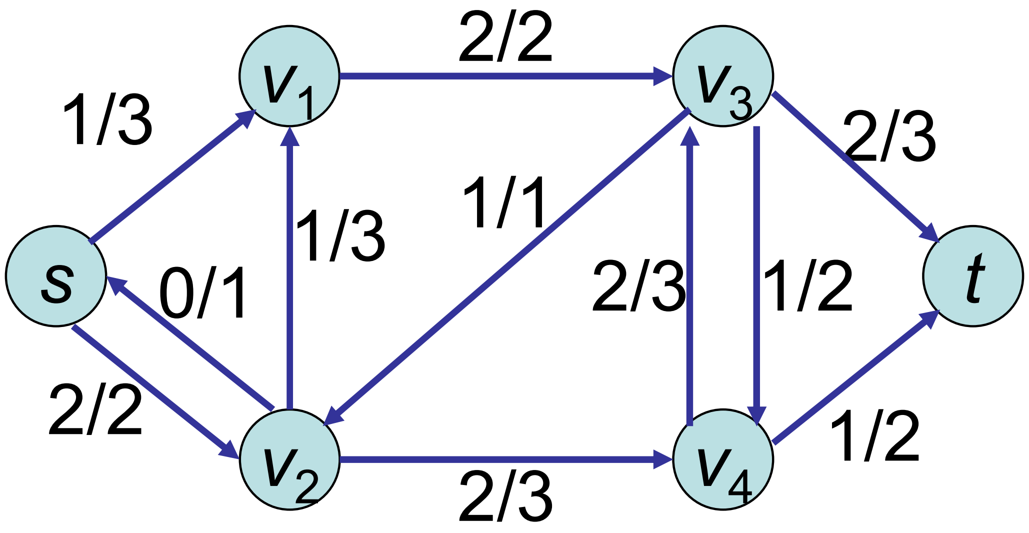
- total positive flow leaving a vertex = total positive flow entering the vertex

FLOW NETWORKS & FLOWS

- value of a positive flow:

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

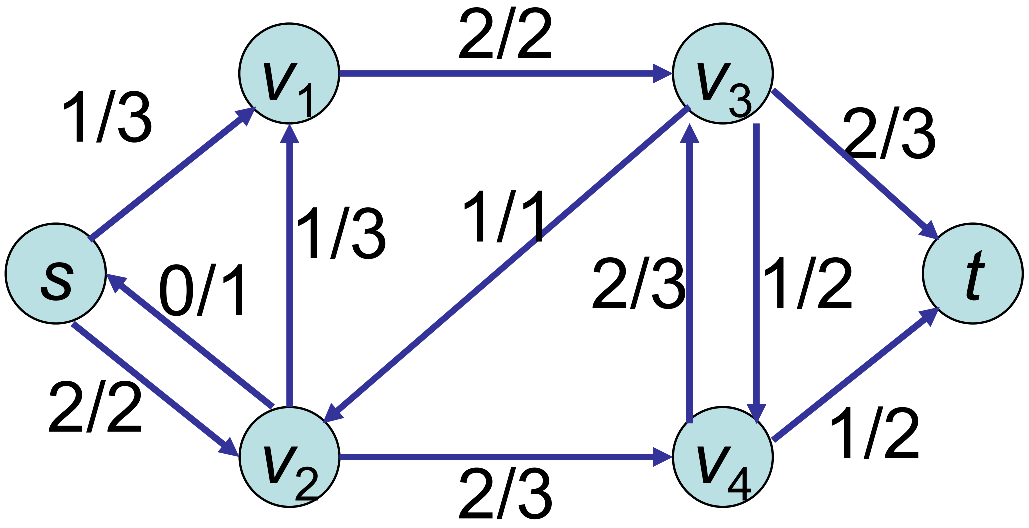
- a sample flow network G and a positive flow p on G: p/c for every edge



note: flow ≤ capacity at every edge

note: flow conservation holds at every vertex (except *s* and *t*)

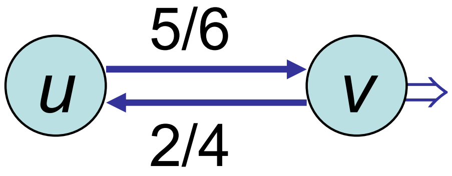
FLOW NETWORKS & FLOWS

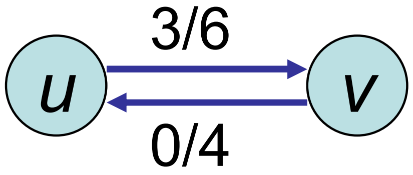


$$|p| = \left(p(s,v_1) + p(s,v_2)\right) - p(v_2,s) = (1+2) - 0 = 3$$

$$|p| = p(v_3,t) + p(v_4,t) = 2 + 1 = 3$$

FLOW NETWORKS & FLOWS

- cancellation : can say positive flow either goes from *u* to *v* or from *v* to *u*, but not both
  - if not true, can transform by cancellation to be true
  - e.g.,  ⇒ 3 units of net positive flow from *u* to *v*

  - ⇒ 

    - ➢ can be obtained by canceling 2 units of flow in each direction

- capacity constraint still satisfied : flows only decrease
- flow conservation still satisfied : flow-in & flow-out both reduced by the same amount

NET FLOW VERSUS POSITIVE FLOW DEFINITIONS

- positive flow is more intuitive
- net flow brings mathematical simplification: half as many summations to write
- A net flow f on G: *a fn* $f: V \times V \to R$ satisfying
  - **capacity constraint:** $\forall\ u,v \in V \quad f(u,v) \leq c(u,v)$
  - **skew symmetry:** $\forall\ u,v \in V \quad f(u,v) = -f(v,u)$
    - thus, $f(u,u) = -f(u,u) \Rightarrow f(u,u) = 0 \Rightarrow$ net flow from a vertex to itself is 0
  - **flow conservation:** $\forall\ u \in V - \{s,t\}$, $\sum_{v \in V} f(u,v) = 0$
    - total net flow into a vertex is 0
- Nonzero net flow from u to $v \Rightarrow (u,v) \in E$, or $(v,u) \in E$, or both.
- value of a net flow : $|f| = \sum_{v \in V} f(s,v) =$ net flow out of the source

NET FLOW VERSUS POSITIVE FLOW DEFINITIONS

- equivalence of net flow and positive flow definitions:
- define net flow in terms of positive flow:
  - f(u,v)= p(u,v) - p(v,u)
    - Given definition of p, this def. of f satisfies (1) capacity constraint, (2) skew symmetry, and (3) flow constraint.

$$(1)\, p(u,v) \le c(u,v) \,\&\, p(v,u) \ge 0 \Rightarrow f(u,v) = p(u,v) - p(v,u) \le c(u,v)$$

$$(2)\, f(u,v) = p(u,v) - p(v,u) = -\big(p(v,u) - p(u,v)\big) = -f(v,u)$$

$$(3)0 = \sum_{v\in V} p(u,v) - \sum_{v\in V} p(v,u) = \sum_{v\in V}\big(p(u,v) - p(v,u)\big) = \sum_{v\in V} f(u,v)$$

NET FLOW VERSUS POSITIVE FLOW DEFINITIONS

- define positive flow in terms of net flow:

$$p(u,v) = \begin{cases} f(u,v), & \text{if } f(u,v) > 0 \\ 0, & \text{if } f(u,v) \leq 0 \end{cases}$$

- Given definition of f, this def. of p satisfies (1) capacity constraint, (2) flow constraint.

FLOW NETWORKS & MAXIMUM FLOW PROBLEM

- maximum flow problem : given a flow network G with source *s* and sink *t*
  - find a flow of maximum value from *s* to *t*
- flow network with multiple sources and sinks :
  - a flow network G with m sources $\{s_1, s_2, \ldots, s_m\} = S_\mathrm{m}$ and n sinks $\{t_1, t_2, \ldots, t_n\} = T_\mathrm{n}$
  - Max flow problem : find a flow of max value from *m* sources to *n* sinks
  - Can reduce to an ordinary max-flow problem with a single source & a single sink

FLOW NETWORKS & MAXIMUM FLOW PROBLEM

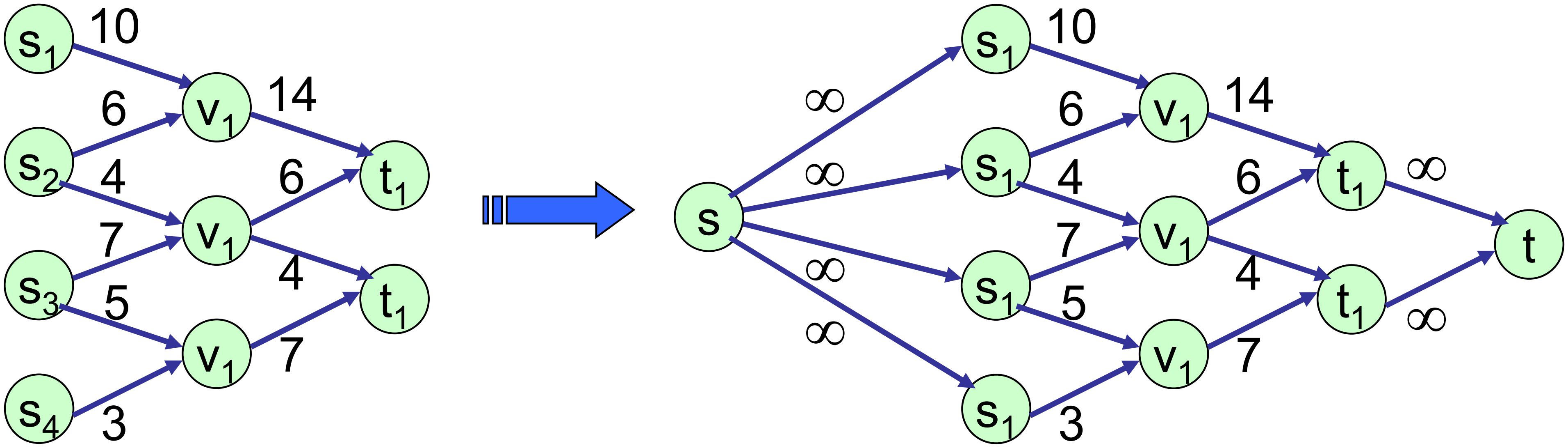
- add a supersource s and a supersink t such that
  - Add a directed edge (s,s_i) with capacity $c(s,s_i)=\infty$

    for $i=1,2,\ldots,m$
  - Add a directed edge (t_i, t) with capacity $c(t,t_i)=\infty$

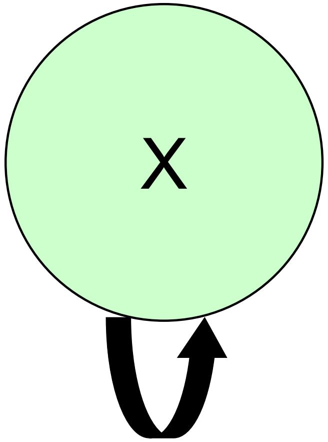
    for $i=1,2,\ldots,n$
  - i.e., $\hat{V} = V \cup \{s,t\}$; $\hat{E} = E \cup \{(s,s_i) \text{ with } c(s,s_i)=\infty : \forall s_i\} \cup \{(t,t_i) \text{ with } c(t,t_i)=\infty : \forall t_i\}$

FLOW NETWORKS & MAXIMUM FLOW PROBLEM

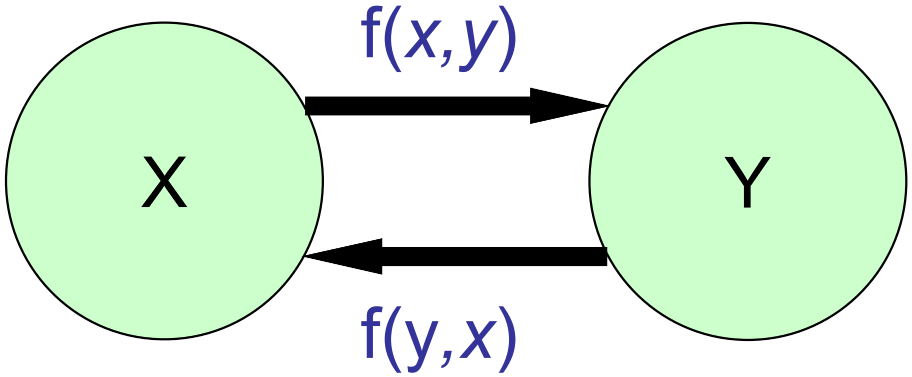
Example: A flow network with multiple sources and sinks



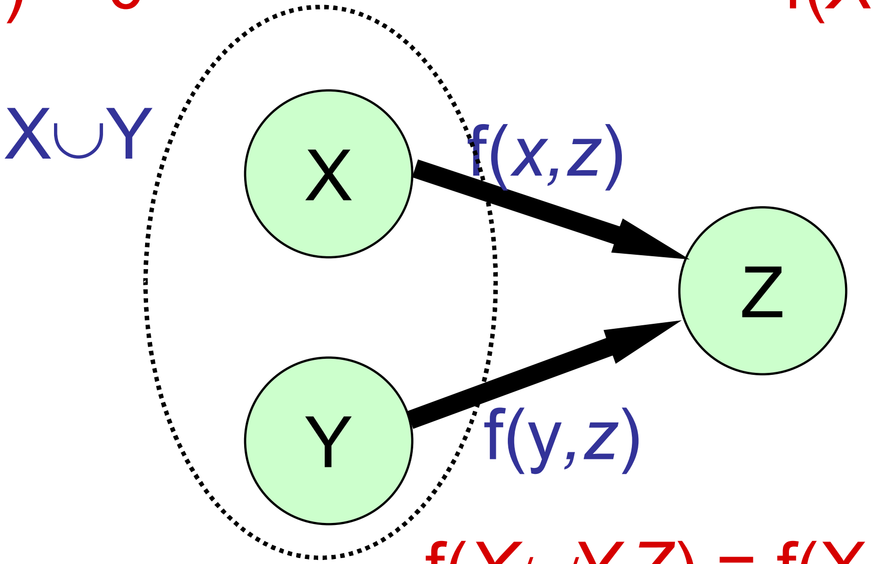
IMPLICIT SUMMATION NOTATION



$f(X,X) = 0$



$f(X,Y) = – f(Y,X)$



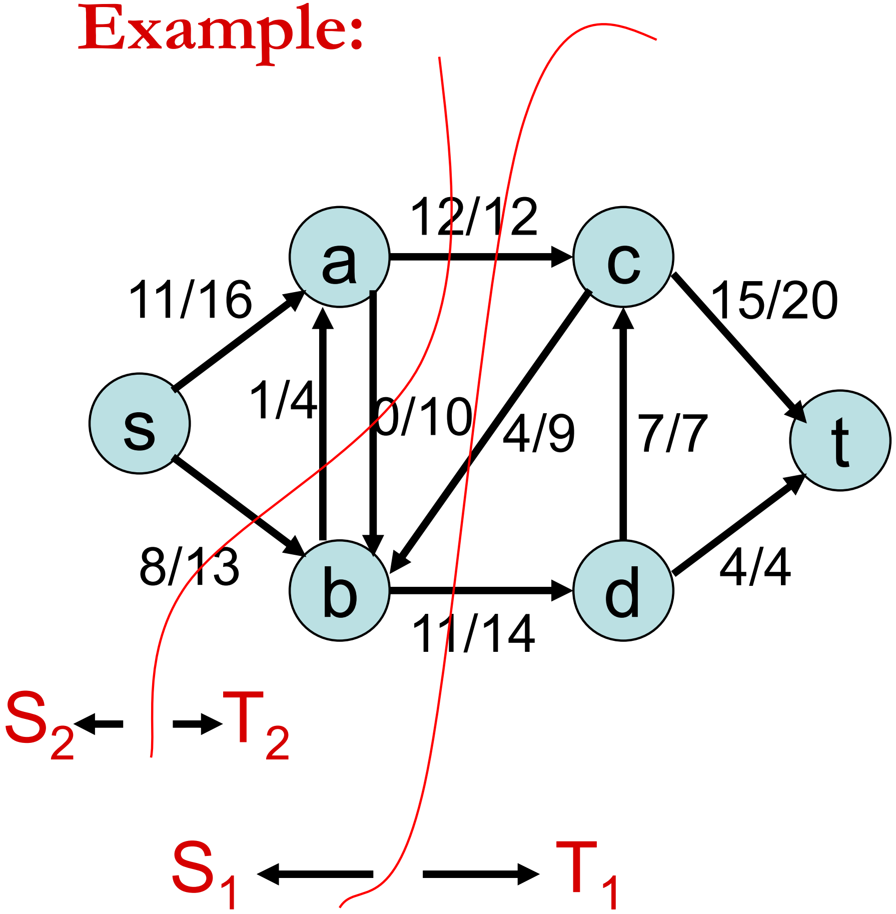
$f(X \cup Y,Z) = f(X,Z) + f(Y,Z)$

CUTS OF FLOW NETWORKS & UPPER BOUND ON MAX FLOW

- def: A cut (S, T) of a flow network is a partition of V into S and $T = V\text{-}S$ such that $s \in S$ and $t \in T$
  - Similar to the cut definition given for MST
  - Differences: G is a directed graph here & we insist that $s \in S$ and $t \in T$
- def: $f(S, T)$ is the net flow across the cut (S, T) of G for a flow f on G
  - Add all edges $S \rightarrow T$ and negative of all edges $T \rightarrow S$ due to skew symmetry
- def: $c(S, T)$ is the capacity across the cut (S, T) of G
  - not like flow because no skew symmetry; just add edges $S \rightarrow T$ (no neg. values)

CUTS OF FLOW NETWORKS & UPPER BOUND ON MAX FLOW

Example:



(S_1, T_1) = ({s, a, b}, {c, d, t})

- *f(S_1, T_1)* = *f(a, c) + f(b, c) + f(b, d)*

  = 12 + (-4) + 11 = 19

- *c(S_1, T_1) = c(a, c) + c(b, d)*

  = 12 + 14 = 26

(S_2, T_2) = ({s, a}, {b, c, d, t})

- *f(S_2, T_2)* = *f(s, b) + f(a, b) + f(a, c)*

  = 8 + (-1) + 12 = 19

- *c(S_2, T_2) = c(s, b) + c(a, b) + c(a, c)*

  = 13 + 10 + 12 = 35

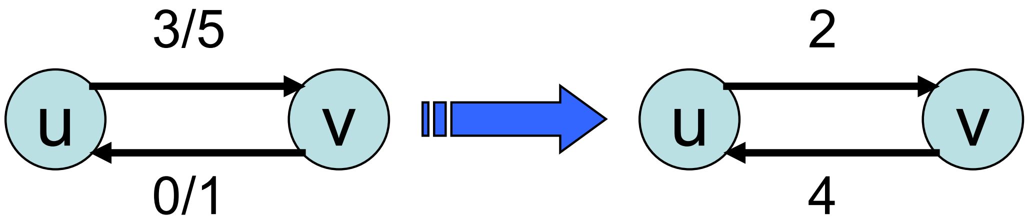
RESIDUAL NETWORKS

- *intuitively*: the residual network G_f of a flow network G with a flow f
  - Consists of edges that can admit more flow
- *def*: given a flow network $G = (V, E)$ with a flow f
  - **residual capacity** of (u, v): $c_f(u, v) = c(u, v) - f(u, v)$ $\forall u,v \in V$
    - $c_f(u, v)$: additional flow we can push from u to v without exceeding $c(u, v)$
  - **residual network** of G induced by f is the graph $G_f = (V, E_f)$ with
    - Strictly positive residual capacities: $E_f = \{(u, v) \in V \times V: c_f(u, v) > 0\}$

RESIDUAL NETWORKS

- ***recall***: if both $(u, v) \in E$ and $(v, u) \in E$ then
  - – Transform such that either $f(u, v) = 0$ or $f(v, u) = 0$ by cancellation
- ***examples:***

(1) *both $(u, v) \in E$ and $(v, u) \in E$*

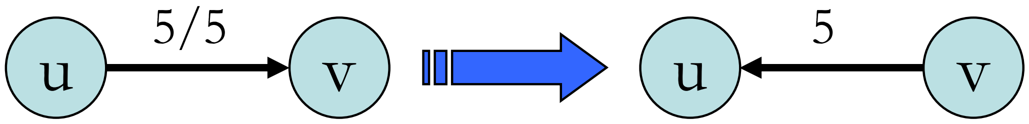


$$c_f(u, v) = c(u, v) - f(u, v) = 5 - 3 = 2 > 0$$

$$c_f(v, u) = c(v, u) - f(v, u) = 1 - (-3) = 4 > 0$$

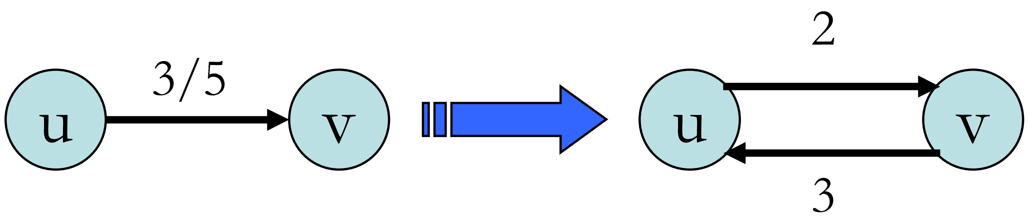
RESIDUAL NETWORKS

(2) $(u, v) \in E$ *but* $(v, u) \notin E$ *and* $f(u, v) \geq 0$: (v, u) *becomes an edge of* E_f



$c_f\ (u, v) = 5 - 5 = 0 \leq 0$ ✗

$c_f\ (v, u) = 0 - (-5)\ = 5 > 0$ √



$c_f\ (u, v) = 5 - 3 = 2 > 0$ √

$c_f\ (v, u) = 0 - (-3) = 3 > 0$ √

- **$|E_f| \leq 2|E|$, since $(u,v) \in E_f$ only if at least one of (u,v) and (v,u) is in E**
- ***note:* $c_f(u, v) + c_f(v, u) = c(u, v) + c(v, u)$**

AUGMENTING PATHS

- For a flow f on G
  - augmenting path p is a simple path from s to t in G_f
- $c_f(p)$: residual capacity of a path p = $\min_{(u,v)\in p}\{c_f(u,v)\}$
  - i.e., $c_f(p)$ = max. amount of the flow we can ship along edges of p on G_f

AUGMENTING PATHS

- *L6:* let f be a flow on G, and let p be an augmenting path in G_f. Let f_p be a flow on G_f with value $| f_p | = c_f(p) > 0$ defined as

$$f_p(u,v) = \begin{cases} c_f(p) \text{ if } (u,v) \in p \text{ in } G_f \\ 0 \qquad \text{otherwise} \end{cases}$$

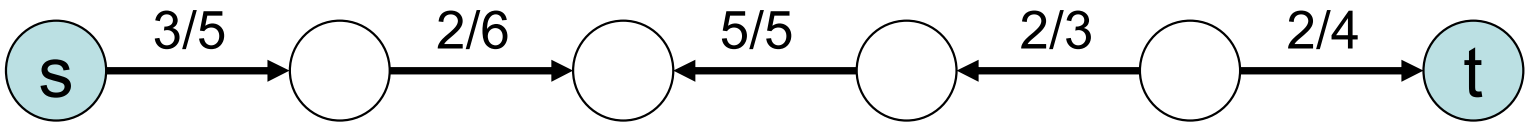
then, $f' = f + f_p$ is a flow on G with value

$$| f' | = | f | + | f_p | > | f |$$

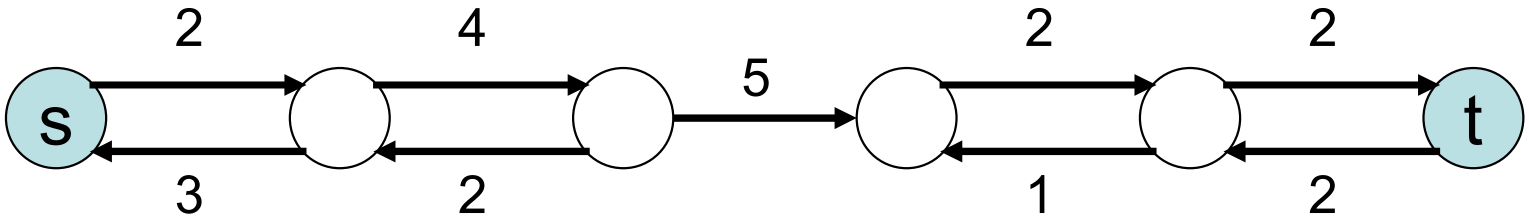
AUGMENTING PATHS

- *example:* a single path p on G
  - can easily say that it is not all of G since flow is not conserved on p

p on G with f:

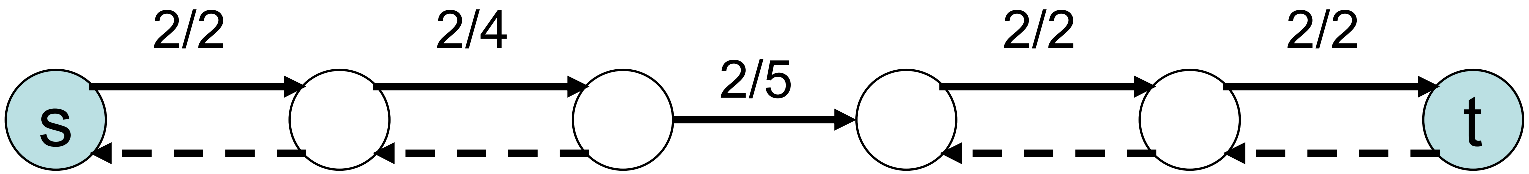


p on G_f:



- define flow f_p on G_f with $c_f(p) = min\{2, 4, 5, 2, 2, 2\} = 2$

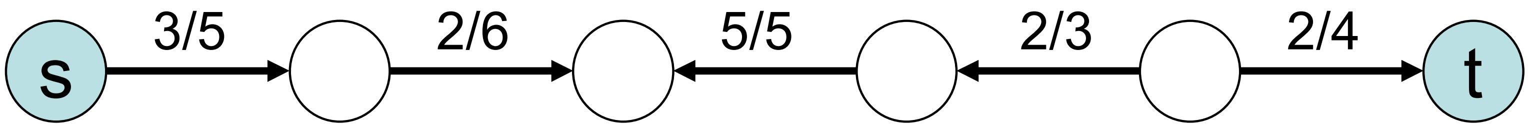
f_p on G_f:



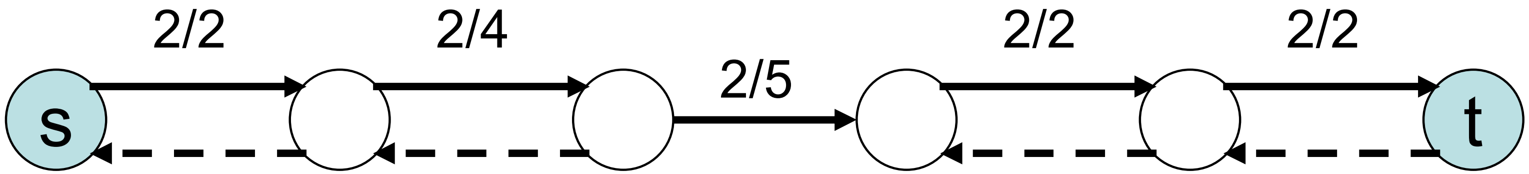
AUGMENTING PATHS

- ***example (cont.):*** **a single path *p* on *G***

p on *G* with *f*:

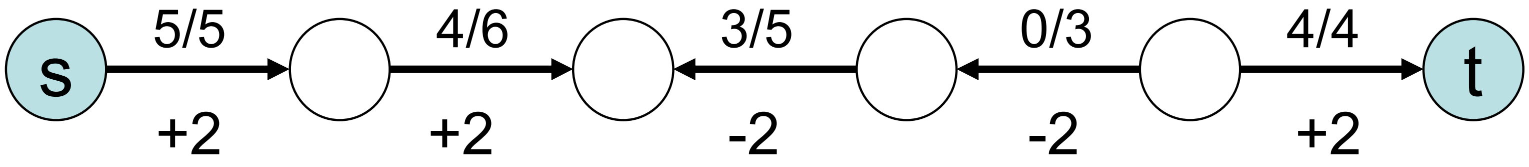


f_p on G_f :

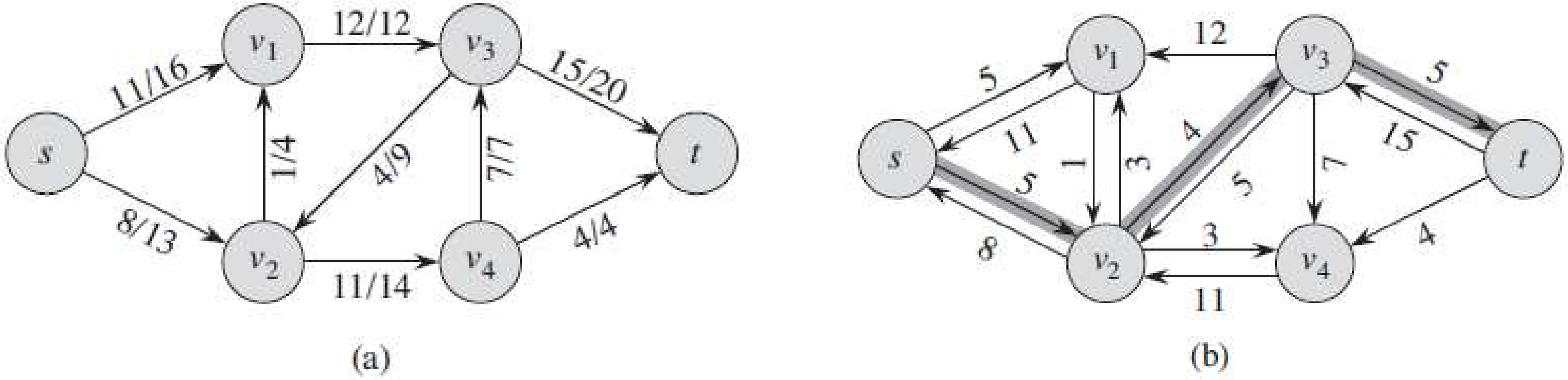


– flow on *p* in *G* that results from augmenting along path *p*:

f + f_p *on p* of *G*:



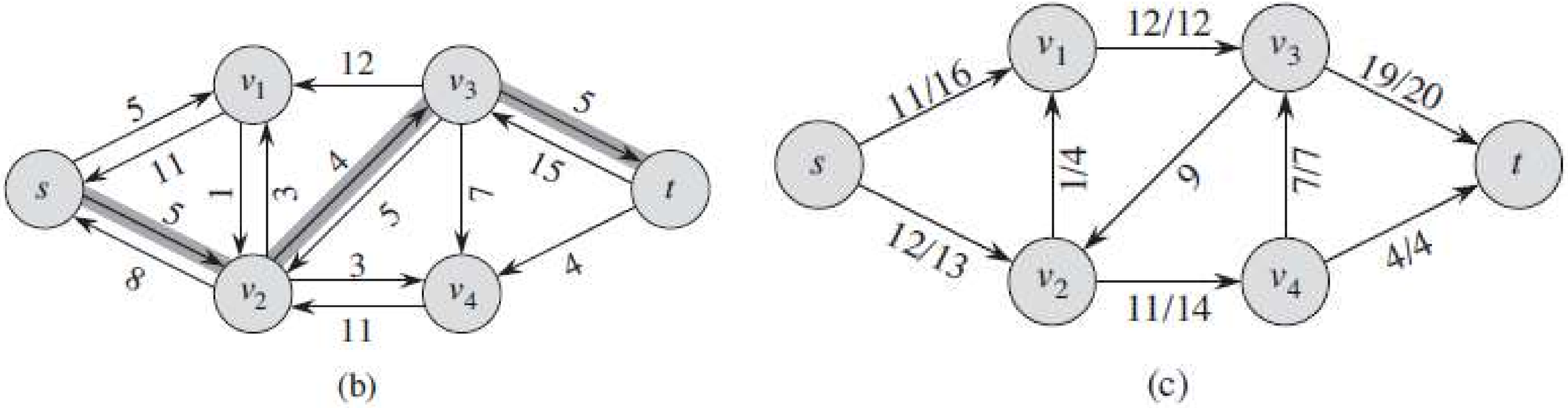
EXAMPLE



(a) The flow network G and flow f.

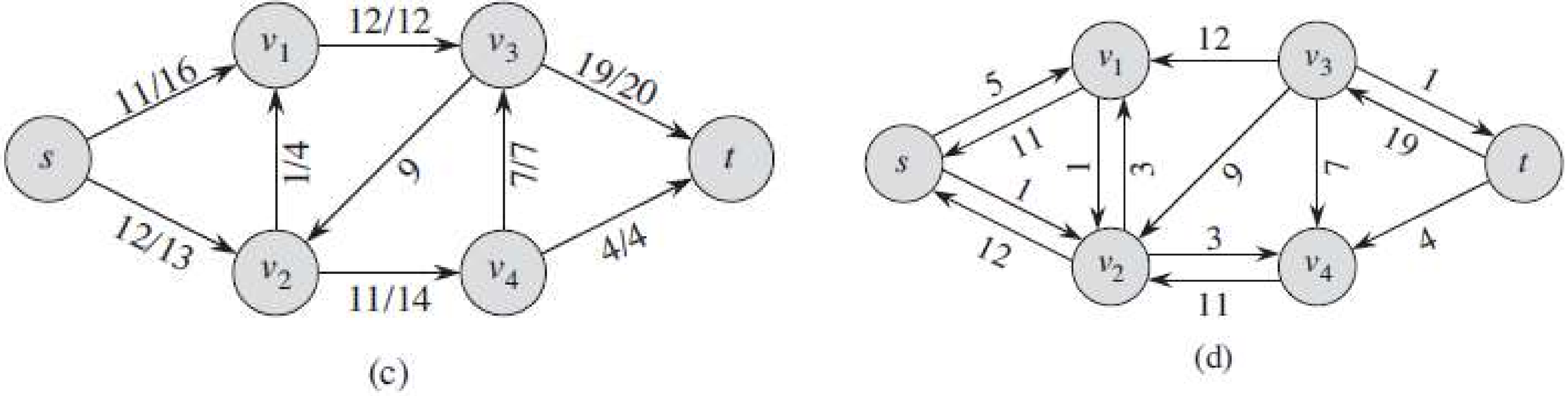
(b) The residual network G_f with augmenting path p shaded; its residual capacity is c_f (p)=$c_f(v_2, v_3)$. Edges with residual capacity equal to 0, such as (v_1, v_3) are not shown

EXAMPLE



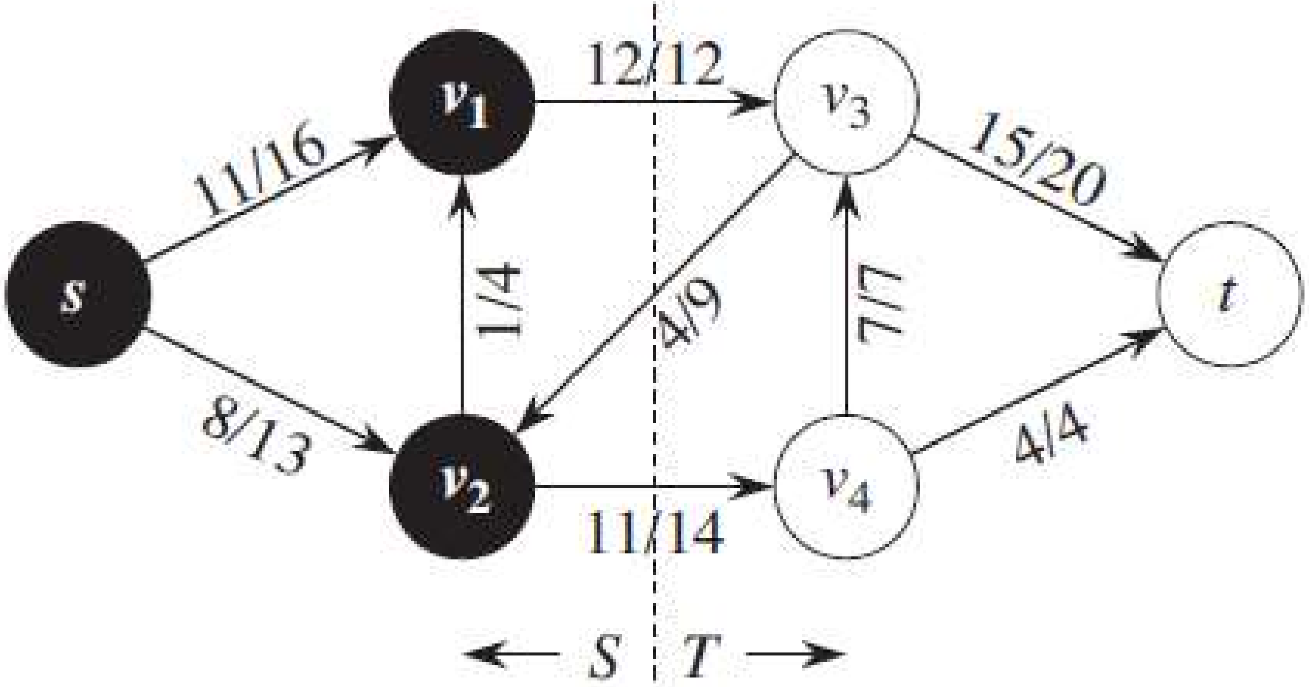
(c) The flow in G that results from augmenting along path p by its residual capacity 4. Edges carrying no flow, such as (v_3, v_2) are labeled only by their capacity, another convention we follow throughout.

EXAMPLE



(d) The residual network induced by the flow in (c).

EXAMPLE



A cut (S, T) in the flow network, where S={s, v_1, v_2 } and T ={v_3, v_4 , t }. The vertices in S are black, and the vertices in T are white. The net flow across (S, T) is f (S,T)=19, and the capacity is c(S, T)= 26.

MAX-FLOW MIN-CUT THEOREM

- *Thm (max-flow min-cut):* the following are equivalent for a flow f on G
  - (1) f is a maximum flow
  - (2) G_f contains no augmenting paths
  - (3) $| f | = c(S, T)$ for some cut (S, T) of G

FORD-FULKERSON METHOD

- iterative algorithm: start with initial flow f=[0] with $|f| = 0$
  - at each iteration, increase $| f |$ by finding an augmenting path
  - repeat this process until no augmenting path can be found
  - by max-flow min-cut theorem: *upon termination* this process yields a max flow

```
FORD-FULKERSON-METHOD(G, s, t)
    initialize flow f to 0
    while    an augmenting path p do
            augment flow f along path p
    return f
```

FORD-FULKERSON ALGORITHM

- basic Ford-Fulkerson Algorithm: data structures
  - note (u,v) E_f only if (u,v) E or (v,u) E
  - maintain an adj-list representation of directed graph $G' = (V', E')$, where
    - $E' = \{(u,v)\text{: } (u,v)$ E *or* (v,u) $E\}$, i.e.,
  - for each v $Adj[u]$ in G' maintain the record

v	$f(u,v)$	$c(u,v)$	$c_f\,(u,v)$

  - note: G' used to represent both G and G_f, i.e., for any edge (u,v) E'
    - $c[u,v] > 0 \Rightarrow$ (u,v) E and $c_f\,[u,v] > 0 \Rightarrow$ (u,v) E_f

FORD-FULKERSON ALGORITHM

```
FORD-FULKERSON ($G'$, $s$, $t$)
    for each edge $(u,v)$      $E'$ do
        $f[u,v] \leftarrow 0$
        $c_f[u,v] \leftarrow 0$
    $G_f \leftarrow$ COMPUTE-GF($G'$, $f$)
    while      an $s \rightsquigarrow t$ path $p$ in $G_f$ do
        $c_f(p) \leftarrow \min\{c_f[u,v]: (u,v)$      $p\}$
        for each edge $(u,v)$      $p$ do
            $f[u,v] \leftarrow f[u,v] + c_f(p)$
            CANCEL($G'$, $u$, $v$)
        $G_f \leftarrow$ COMPUTE-GF($G'$, $f$)
```

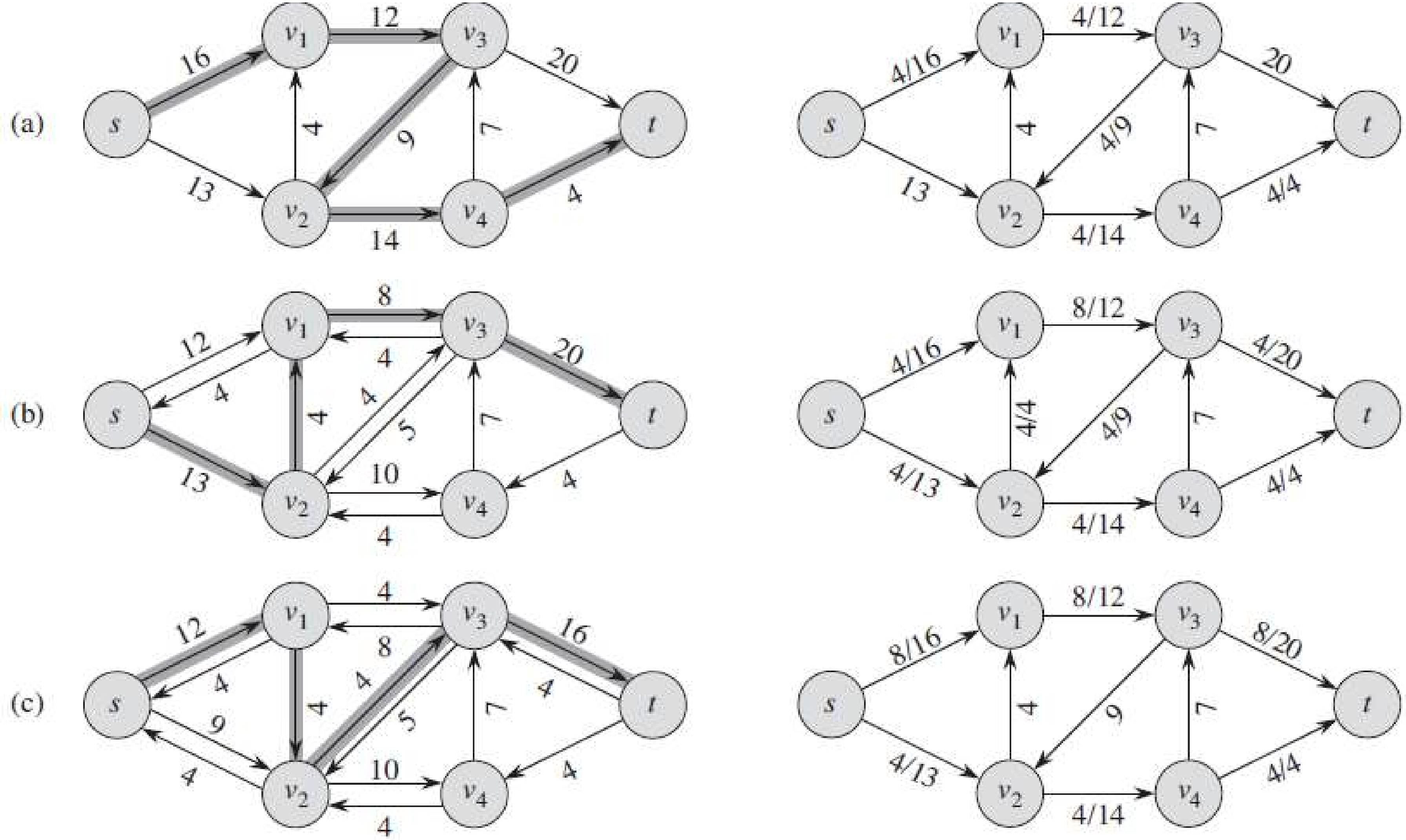
```
COMPUTE-GF ($G'$, $f$)
    for each edge $(u,v)$      $E'$ do
        if $c[u,v] - f[u,v] > 0$ then
            $c_f[u,v] \leftarrow c[u,v] - f[u,v]$
        else
            $c_f[u,v] \leftarrow 0$
    return $G'$
```

```
CANCEL ($G'$, $u$, $v$)
    $\min \leftarrow \{f[u,v], f[v,u]\}$
    $f[u,v] \leftarrow f[u,v] - \min$
    $f[v,u] \leftarrow f[v,u] - \min$
```

FORD-FULKERSON ALGORITHM

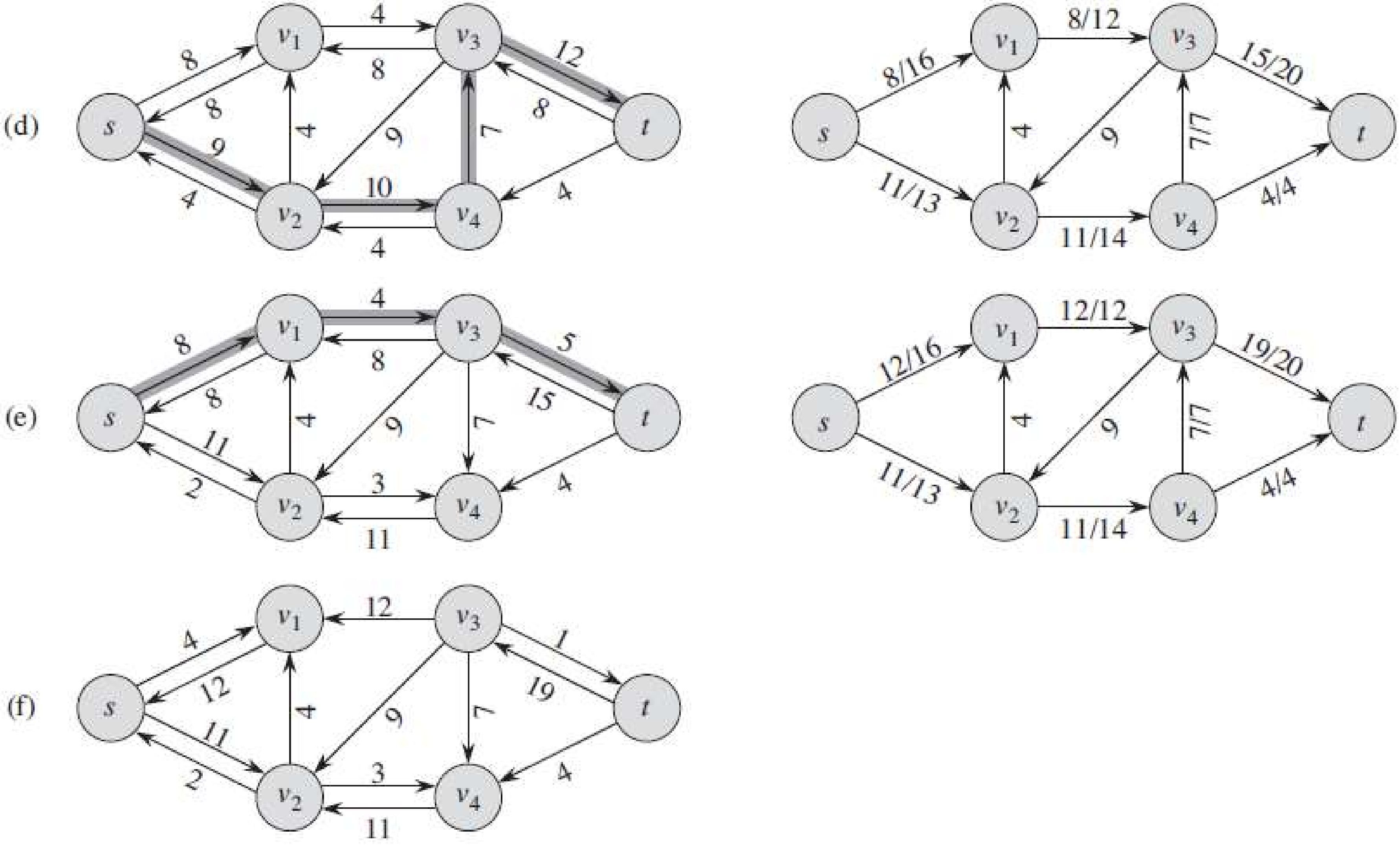
- augmenting path in G_f is chosen arbitrarily
- performance if capacities are integers: $O(E\ |f^*|)$
  - while-loop: time to find $s \rightsquigarrow t$ path in $G_f = O(E') = O(E)$
  - # of while-loop iterations: $\leq |f^*|$, where f^* = max flow
- so, running time is good if capacities are integers and $|f^*|$ is small

FORD-FULKERSON ALGORITHM



The left side of each part shows the residual network Gf from line 3 with a shaded augmenting path p. The right side of each part shows the new flow f that results from augmenting f by fp.

FORD-FULKERSON ALGORITHM



(f) The residual network at the last while loop test. It has no augmenting paths, and the flow f shown in (e) is therefore a maximum flow. The value of the maximum flow found is 23.

FORD-FULKERSON ALGORITHM

- might never terminate for non-integer capacities
- efficient algorithms:
  - augment along max-capacity path in G_f: not mentioned in textbook
  - augment along breadth-first path in G_f: Edmonds-Karp algorithm

    $\Rightarrow O(VE^2)$

EDMONDS-KARP ALGORITHM

- def: $\delta_f(s,v)$ = shortest path distance from s to v in G_f
  - unit edge weights in $G_f \Rightarrow \delta_f(s,v)$ = breadth-first distance from s to v in G_f
- L7: $\forall\ v \qquad V - \{s,t\}$; $\delta\,(s,v)$ in G_f's increases monotonically with each augmentation
- proof: suppose
  - (i) a flow f on G induces G_f
  - (ii) f_p along an augmenting path in G_f produces $f' = f + f_p$ on G
  - (iii) f' on G induces $G_{f'}$
- notation: $\delta(s,v) = \delta_f(s,v)$ and $\delta'(s,v) = \delta_{f'}(s,v)$

Maximum Bipartite Matching Problem

- many combinatorial optimization problems can be reduced to a max-flow problem
- maximum bipartite matching problem is a typical example

Maximum Bipartite Matching Problem

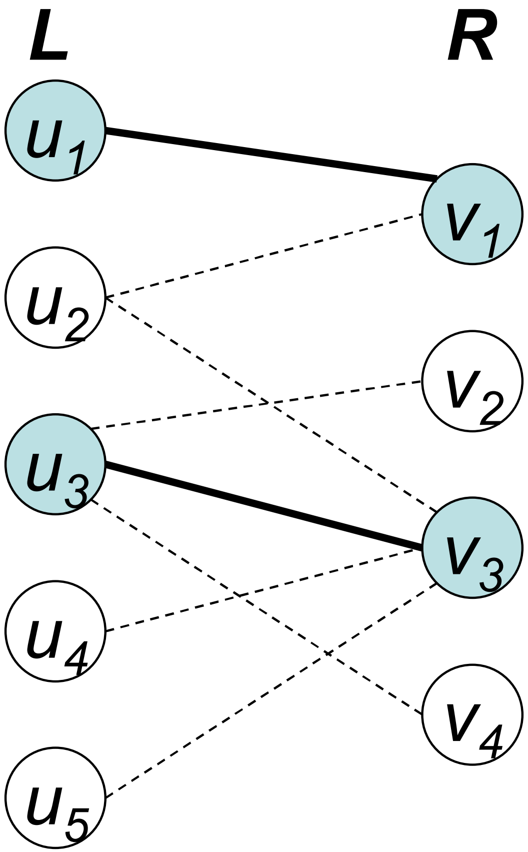
- given an undirected graph $G = (V, E)$
- *def:* a matching is a subset of edges $M \subseteq E$ such that
  $\forall\ v \quad V$, at most one edge of M is incident to v
- *def:* a vertex $v \quad V$ is matched by a matching M if some edge M is incident to v, otherwise v is unmatched
- *def:* a maximum matching M^* is a matching M of maximum cardinality, i.e., $|M^*| \geq |M|$ for any matching M
- *def:* $G=(V,E)$ is a bipartite graph if $V=L \cup R$ where $L \cap R = \emptyset$ such that $E=\{(u,v)\colon u \quad L$ and $v \quad R\}$

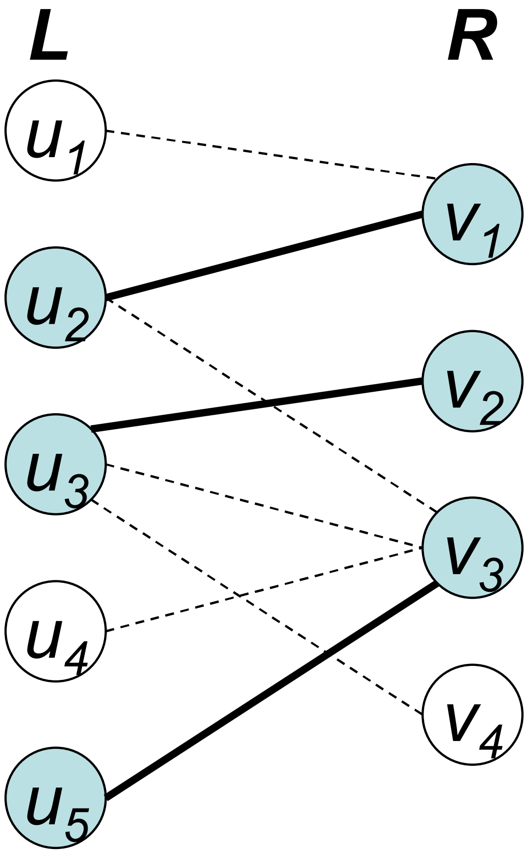
Maximum Bipartite Matching Problem

- *applications:* job task assignment problem
  - Assigning a set L of tasks to a set R of machines
  - (u,v) $E \Rightarrow$ task u L can be performed on a machine v R
  - a max matching provides work for as many machines as possible

Maximum Bipartite Matching Problem

- *example:* two matchings M_1 & M_2 on a sample graph with $|M_1| = 2$ & $|M_2| = 3$





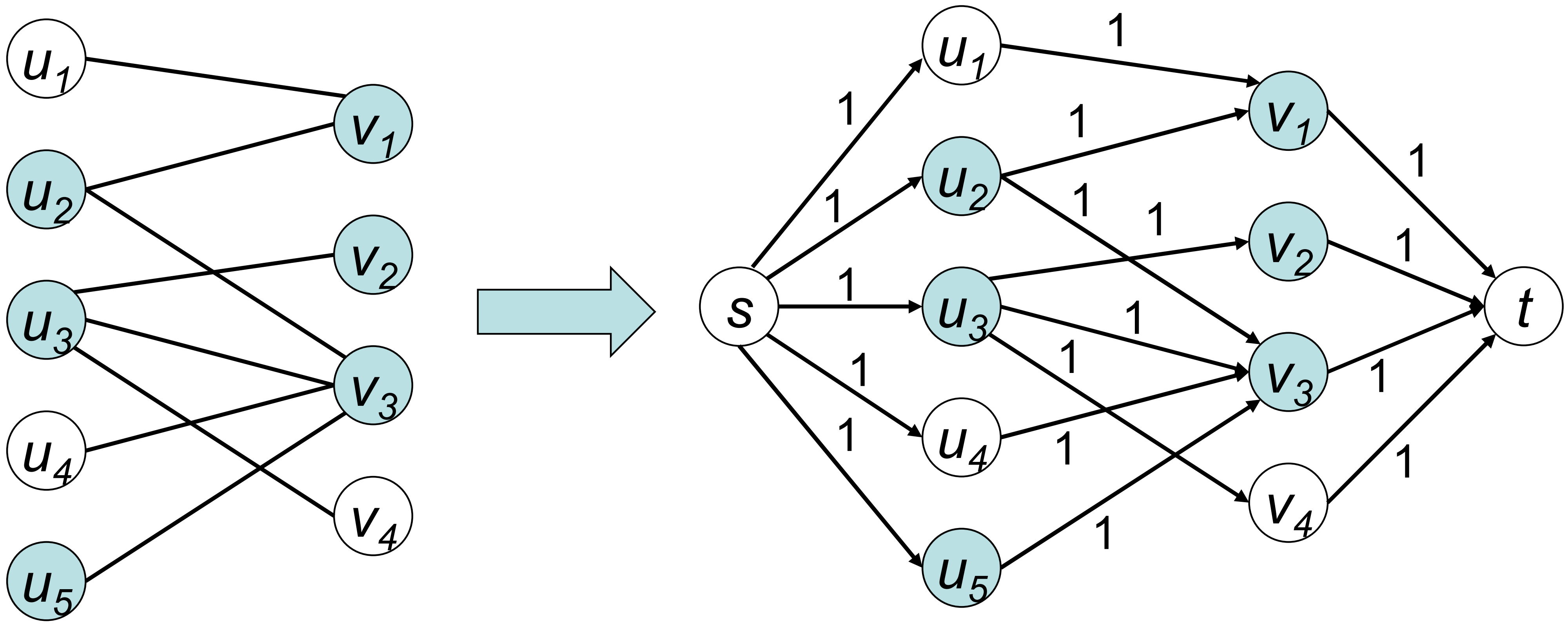
$M_1 = \{(u_1,v_1), (u_3,v_3)\}$ $M_2 = \{(u_2,v_1), (u_3,v_2) , (u_5,v_3)\}$

Finding a Maximum Bipartite Matching

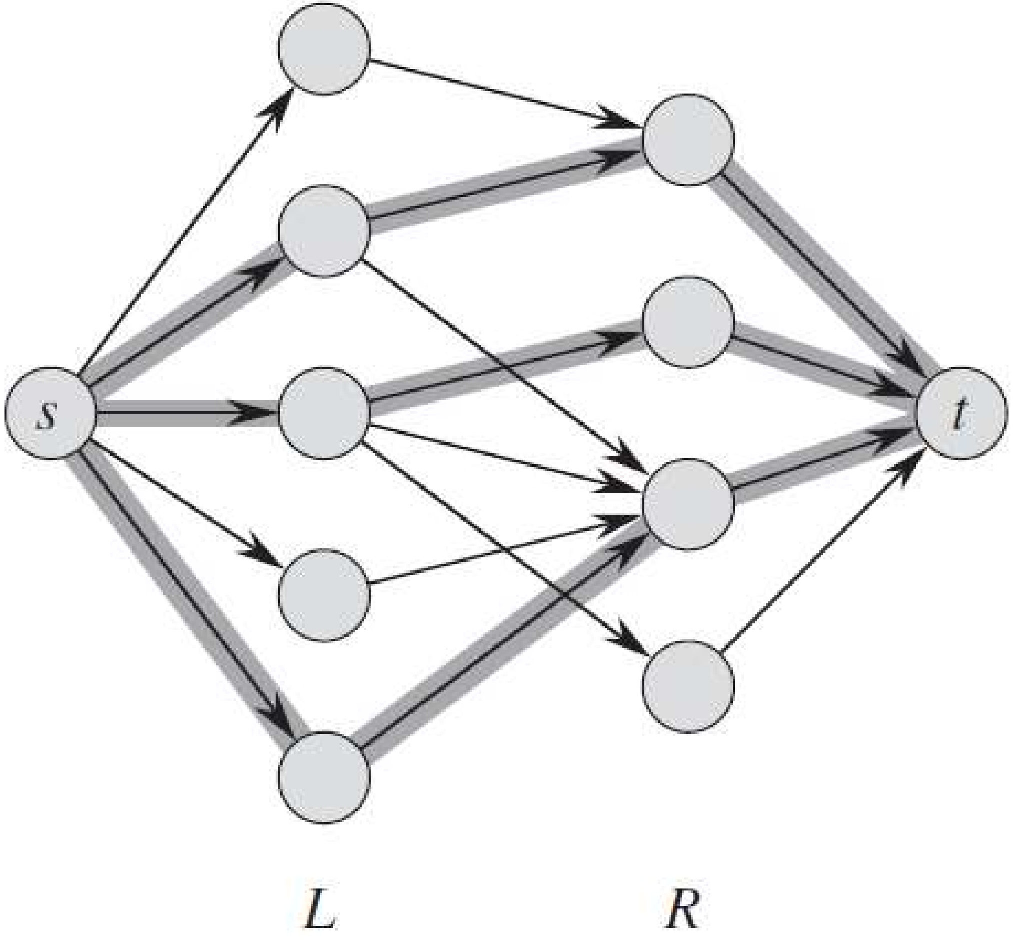
- *idea:* construct a flow network in which flows correspond to matchings

- define the corresponding flow network $G'=(V', E')$ for the bipartite graph as
  - $V' = V \cup \{s\} \cup \{t\}$ $\quad s, t \quad V$
  - $E' = \{(s,u): \forall\ u \quad L\} \cup \{(u,v): u \quad L,\ v \quad R,\ (u,v) \quad E\}$ $\cup \{(v,t): \forall\ v \quad R\}$
  - assign unit capacity to each edge of E'

Finding a Maximum Bipartite Matching



Finding a Maximum Bipartite Matching



Finding a Maximum Bipartite Matching

- *def:* a flow f on a flow network is integer-valued if $f(u,v)$ is integer $\forall u,v \quad V$
- *L8:* **(*a*)** IF M is a matching in G, THEN ∃ an integer-valued f on G' with $|f| = |M|$

  (*b*) IF f is an integer-valued f on G', THEN ∃ a matching M in G with $|M| = |f|$

Finding a Maximum Bipartite Matching

- *proof L8 (a):* let M be a matching in G
  - define the corresponding flow f on G' as
    - ➢ $\forall\, u,v \quad M; f(s,u)=f(u,v)=f(v,t) = 1$ & $f(u,v) = 0$ for all other edges
- *first show that f is a flow on G':*
  - 1 unit of flow passes thru the path $s \rightarrow u \rightarrow v \rightarrow t$ for each u,v M
    - ➢ these paths are disjoint $s \rightsquigarrow t$ paths, i.e., no common intermediate vertices
  - f is a flow on G' satisfying capacity constraint, skew symmetry & flow conservation
    - ➢ because f can be obtained by flow augmentation along these $s \rightsquigarrow t$ disjoint paths

Finding a Maximum Bipartite Matching

- *second show that* $|f| = |M|$*:*
  - net flow accross the cut $(\{s\} \cup L, R \cup \{t\}) = |f|$ by *L3*
  - $|f| = f(s \cup L, R \cup t) = f(s, R \cup t) + f(L, R \cup t)$

    $= f(s, R \cup t) + f(L, R) + f(L, t)$

    $= 0 + f(L, R) + 0;$ $f(s, R \cup t) = f(L, t)$ since $\exists$ no such edges

    $= f(L, R) = |M|$ since $f(u,v) = 1\ \forall u \quad L, v \quad R$ & (u,v)

    M

Finding a Maximum Bipartite Matching

- *proof L8 (b):* let f be a integer-valued flow in G'
  - define $M=\{(u,v): u \quad L, v \quad R, \textit{and } f(u,v) > 0\}$
- first show that M is a matching in G: i.e., all edges in M are vertex disjoint
  - let $p_e(u)$ / $p_l(u)$ = positive net flow entering / leaving vertex u, $\forall u \quad V$
  - each $u \quad L$ has exactly one incoming edge (s,u) with $c(s,u)=1 \Rightarrow p_e(u) \leq 1 \ \forall u \quad L$

Finding a Maximum Bipartite Matching

- since f is integer-valued; $\forall u \quad L, p_e(u) = 1 \Leftrightarrow p_l(u) = 1$ due to flow conservation

  $\Rightarrow \forall u \quad L, p_e(u)=1 \Leftrightarrow \exists$ exactly one vertex $v \quad R \ni f(u,v) = 1$ to make $p_l(u) = 1$

- thus, at most one edge leaving each vertex $u \quad L$ carries positive flow = 1
- a symmetric argument holds for each vertex $v \quad R$
- therefore, M is a matching

Finding a Maximum Bipartite Matching

- second show that $|M| = |f|$:
  - $|M| = f(L, R)$ by above def for M since $f(u,v)$ is either 0 or 1

$$= f(L, V' - s - L - t)$$

$$= f(L,V') - f(L,s) - f(L,L) - f(L,t)$$

$$= 0 - f(L,s) - 0 - 0$$

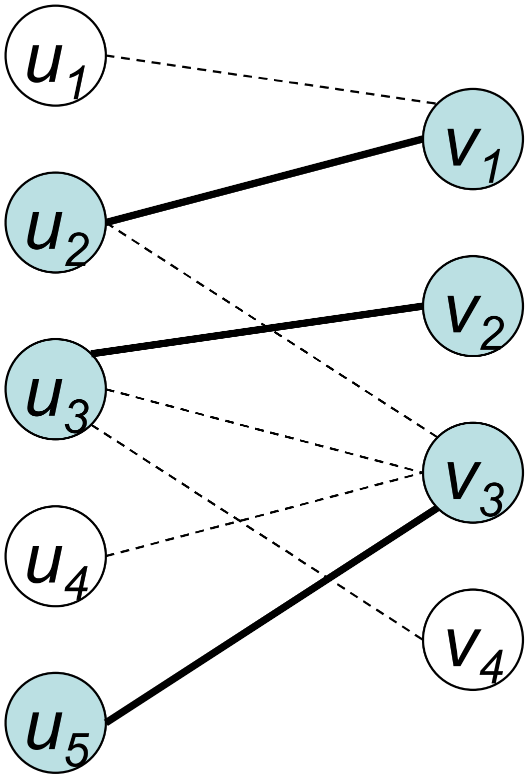
$$= -f(L,s) = f(s,L) = |f|$$ due to skew symmetry & then def.

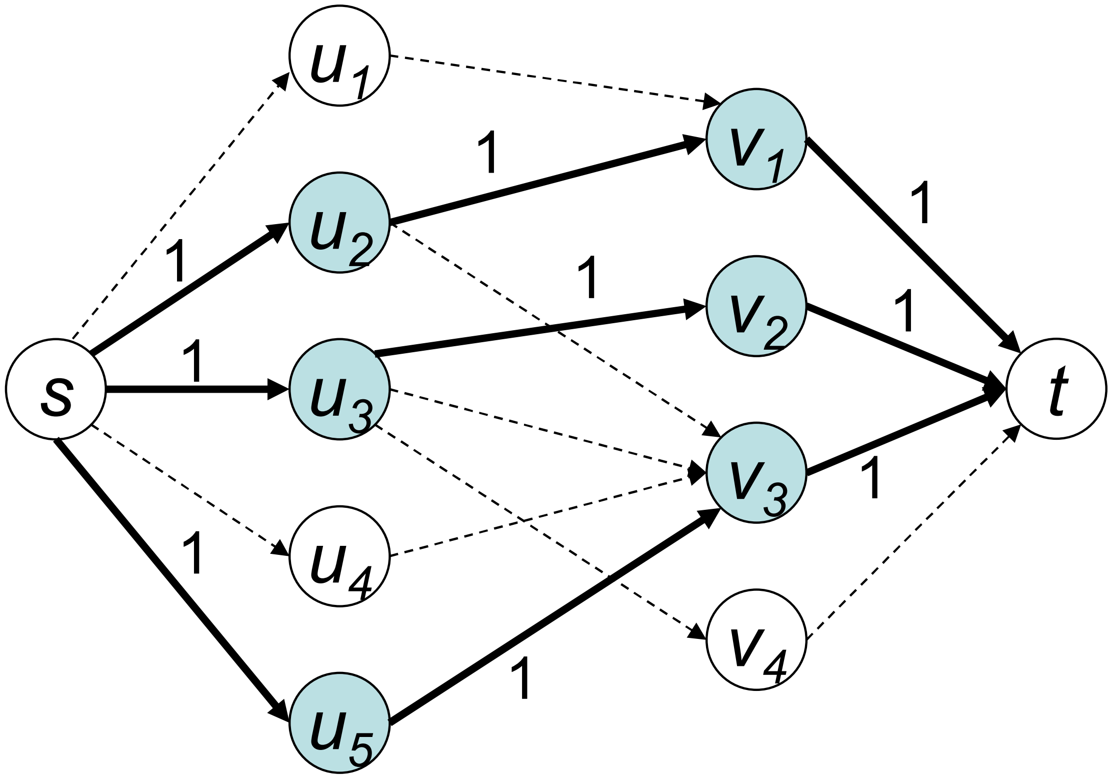
$f(L,V') = f(L,V') =$ 0 due to flow cons.

$f(L,t) = 0$ since no edges from L to t

Finding a Maximum Bipartite Matching

- *example:* a matching M with $|M| = 3$ & a f on the corresponding G' with $|f| = 3$





Thank you!