Chapter 17

Network Flows

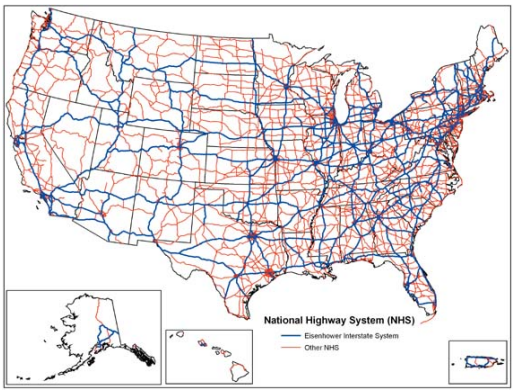
OLD CS 473: Fundamental Algorithms, Spring 2015
March 19, 2015

17.0.0.1 Everything flows

Panta rei – everything flows (literally).
Heraclitus (535–475 BC)

17.1 Network Flows: Introduction and Setup

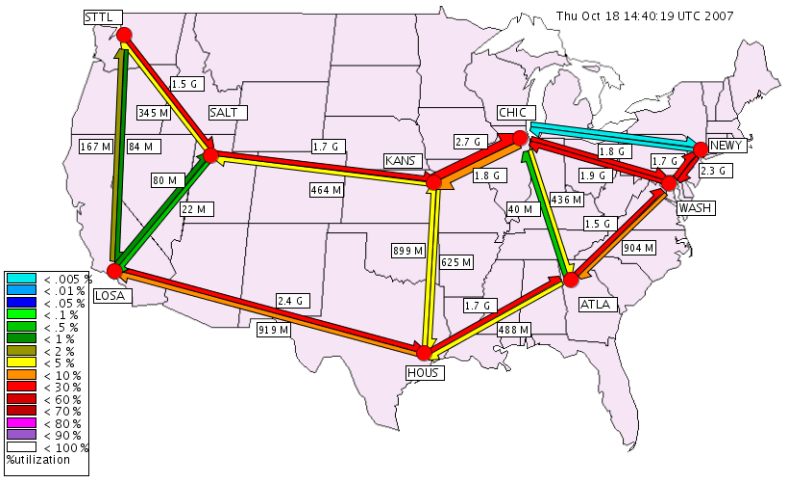
17.1.0.2 Transportation/Road Network



17.1.0.3 Internet Backbone Network

17.1.0.4 Common Features of Flow Networks

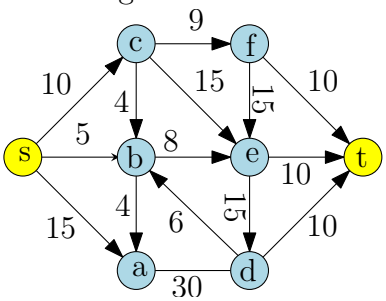
(A) ***Network*** represented by a (directed) *graph* $G = (V, E)$.
(B) Each edge e has a ***capacity*** $c(e) \geq 0$ that limits amount of *traffic* on e.
(C) *Source(s)* of traffic/data.
(D) *Sink(s)* of traffic/data.
(E) Traffic *flows* from sources to sinks.



(F) Traffic is *switched/interchanged* at nodes.
(G) ***Flow*** abstract term to indicate stuff (traffic/data/etc) that *flows* from sources to sinks.

17.1.0.5 Single Source/Single Sink Flows

Simple setting:
(A) Single source s and single sink t.
(B) Every other node v is an ***internal*** node.
(C) Flow originates at s and terminates at t.



(A) Each edge e has a capacity $c(e) \geq 0$.
(B) Sometimes assume:
Source $s \in V$ has no incoming edges, and sink $t \in V$ has no outgoing edges.

(A) **Assumptions:** All capacities are integer, and every vertex has at least one edge incident to it.

17.1.0.6 Definition of Flow

(A) Two ways to define flows...
(B) edge based, or
(C) path based.
(D) Essentially equivalent but have different uses.
(E) Edge based definition is more compact.

17.1.0.7 Edge Based Definition of Flow

Definition 17.1.1. **Flow** *in network* $G = (V, E)$*, is function* $f : E \to \mathbb{R}^{\geq 0}$ *s.t.*

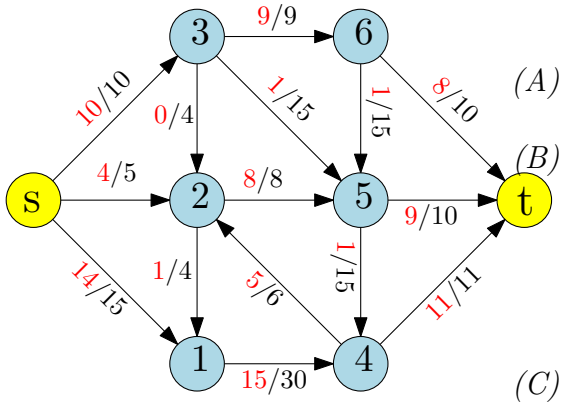


Figure 17.1: Flow with value.

(A) ***Capacity Constraint:*** *For each edge e, $f(e) \leq c(e)$.*

(B) ***Conservation Constraint:*** *For each vertex $v \neq s, t$.*

$$\sum_{e \text{ into } v} f(e) = \sum_{e \text{ out of } v} f(e)$$

(C) ***Value of flow*** *= (total flow out of source) − (total flow in to source).*

17.1.0.8 Flow...

Conservation of flow law is also known as ***Kirchhoff's law***.

17.1.0.9 More Definitions and Notation

Notation

(A) The inflow into a vertex v is $f^{\text{in}}(v) = \sum_{e \text{ into } v} f(e)$ and the outflow is $f^{\text{out}}(v) = \sum_{e \text{ out of } v} f(e)$

(B) For a set of vertices A, $f^{\text{in}}(A) = \sum_{e \text{ into } A} f(e)$. Outflow $f^{\text{out}}(A)$ is defined analogously

Definition 17.1.2. *For a network $G = (V, E)$ with source s, the* **value** *of flow f is defined as $v(f) = f^{\text{out}}(s) - f^{\text{in}}(s)$.*

17.1.0.10 A Path Based Definition of Flow

Intuition: Flow goes from source s to sink t along a path.

$\mathcal{P}$: set of all paths from s to t. $|\mathcal{P}|$ can be *exponential* in n.

Definition 17.1.3 (Flow by paths.). *A* **flow** *in network $G = (V, E)$, is function $f : \mathcal{P} \to \mathbb{R}^{\geq 0}$ s.t.*

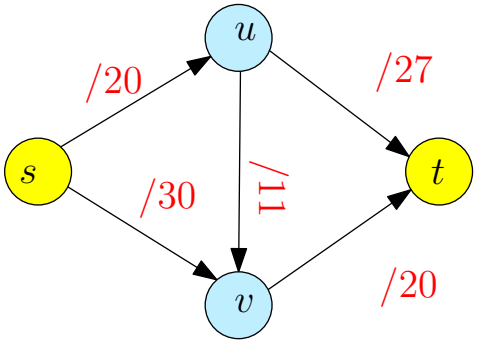
(A) ***Capacity Constraint:*** *For each edge e, total flow on e is $\leq c(e)$.*

$$\sum_{p \in \mathcal{P}: e \in p} f(p) \leq c(e)$$

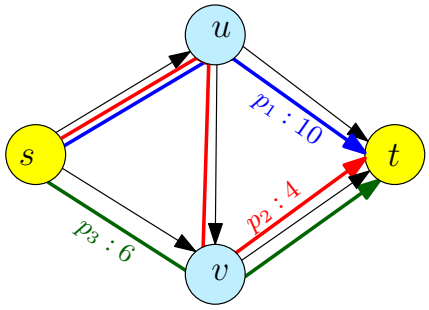
(B) ***Conservation Constraint:*** *No need! Automatic.*

Value of flow: $\sum_{p \in \mathcal{P}} f(p)$.

17.1.0.11 Example



$\mathcal{P} = \{p_1, p_2, p_3\}$
$p_1 : s \to u \to t$
$p_2 : s \to u \to v \to t$
$p_3 : s \to v \to t$
$f(p_1) = 10, f(p_2) = 4, f(p_3) = 6$

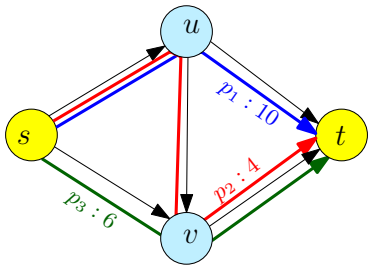


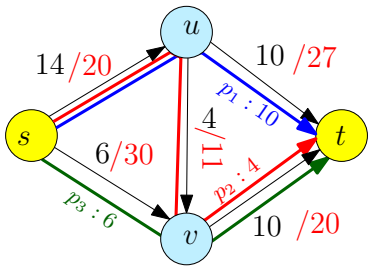
17.1.0.12 Path based flow implies edge based flow

Lemma 17.1.4. *Given a path based flow $f : \mathcal{P} \to \mathbb{R}^{\geq 0}$ there is an edge based flow $f' : E \to \mathbb{R}^{\geq 0}$ of the same value.*

Proof: For each edge e define $f'(e) = \sum_{p:e\in p} f(p)$.
Exercise: Verify capacity and conservation constraints for f'.
Exercise: Verify that value of f and f' are equal ■

17.1.0.13 Example





$\mathcal{P} = \{p_1, p_2, p_3\}$
$p_1 : s \to u \to t$
$p_2 : s \to u \to v \to t$
$p_3 : s \to v \to t$
$f(p_1) = 10, f(p_2) = 4, f(p_3) = 6$
$f'(s \to u) = 14$
$f'(u \to v) = 4$
$f'(s \to v) = 6$
$f'(u \to t) = 10$
$f'(v \to t) = 10$

17.1.1 Flow Decomposition

17.1.1.1 Edge based flow to Path based Flow

Lemma 17.1.5. *Given an edge based flow $f_1 : E \to \mathbb{R}^{\geq 0}$, there is a path based flow $f : \mathcal{P} \to \mathbb{R}^{\geq 0}$ of same value. Moreover, f assigns non-negative flow to at most m paths where $|E| = m$ and $|V| = n$. Given f_1, the path based flow can be computed in $O(mn)$ time.*

17.1.2 Flow Decomposition

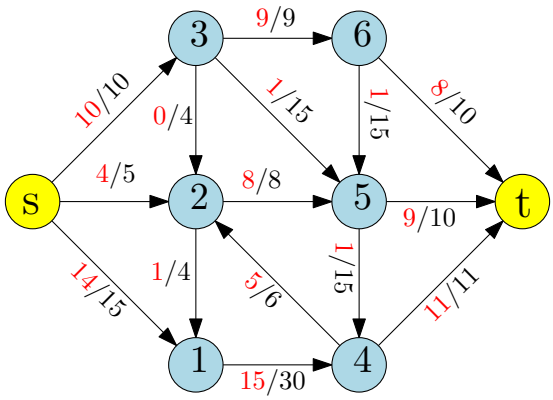
17.1.2.1 Edge based flow to Path based Flow

Proof:[Proof Idea]

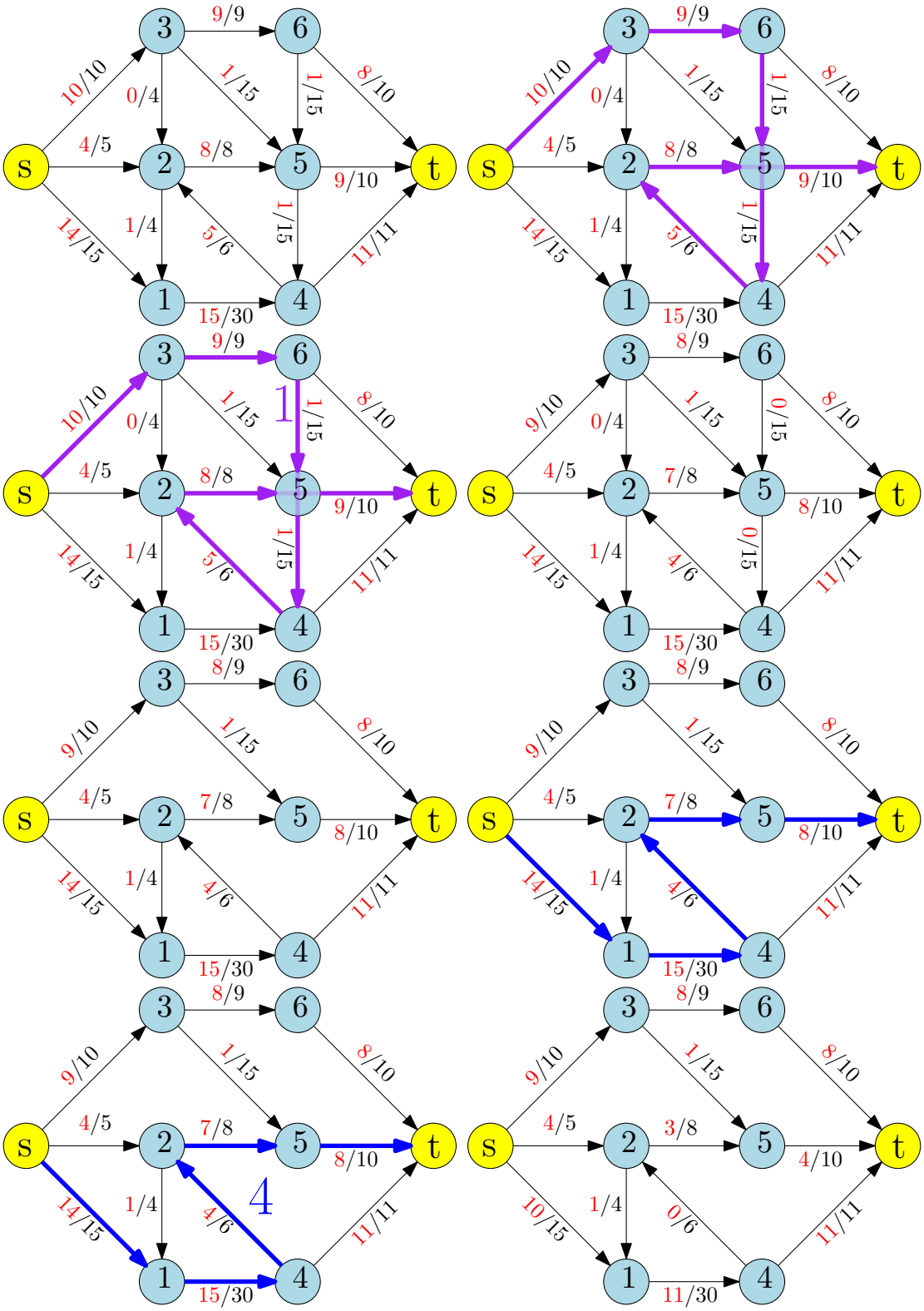
(A) Remove all edges with $f_1(e) = 0$.
(B) Find a path p from s to t.
(C) Assign $f(p)$ to be $\min_{e \in p} f_1(e)$.
(D) Reduce $f_1(e)$ for all $e \in p$ by $f(p)$.
(E) Repeat until no path from s to t.
(F) In each iteration at least on edge has flow reduced to zero.
(G) Hence, at most m iterations. Can be implemented in $O(m(m+n))$ time. $O(mn)$ time requires care.

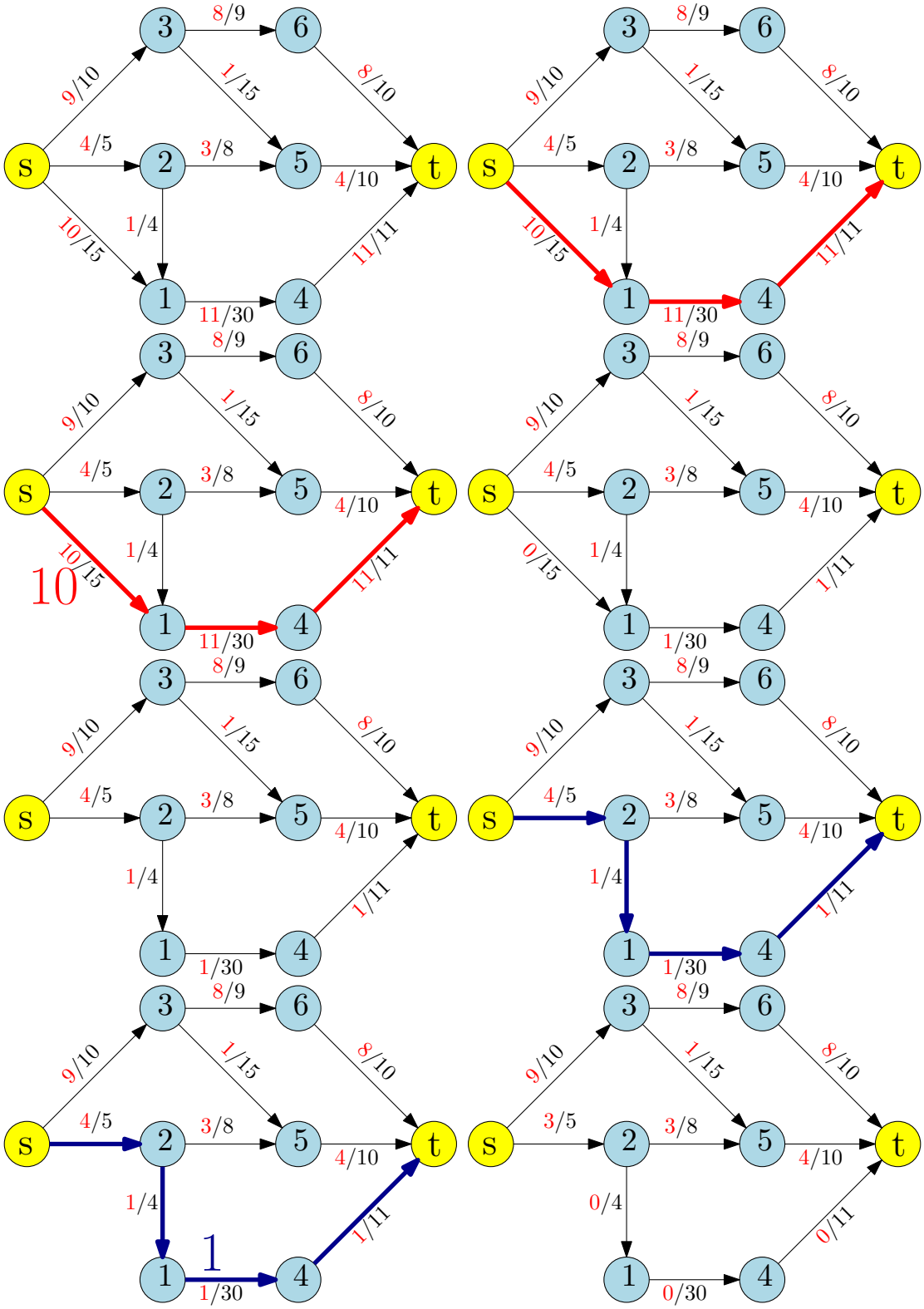
■

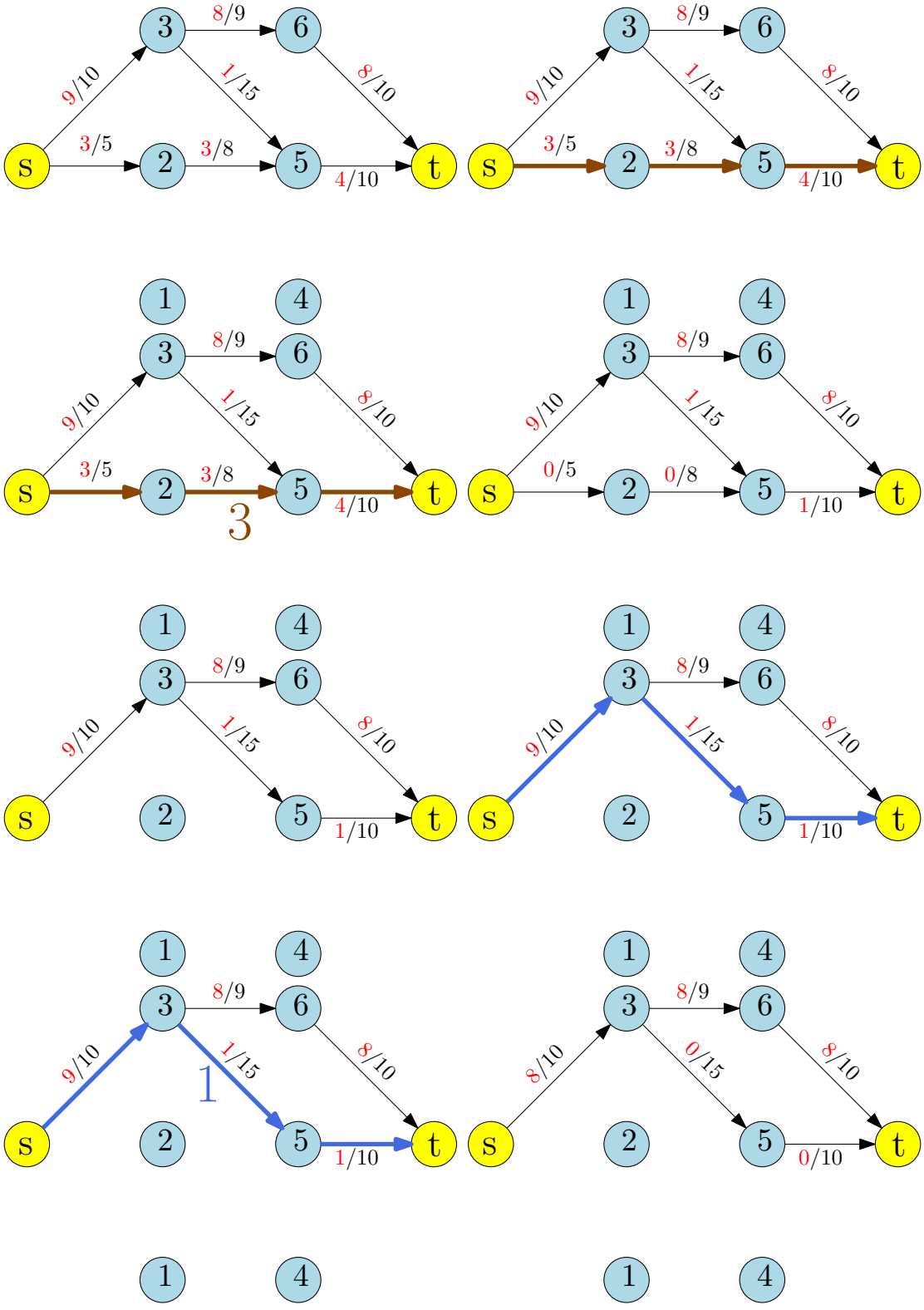
17.1.2.2 Example

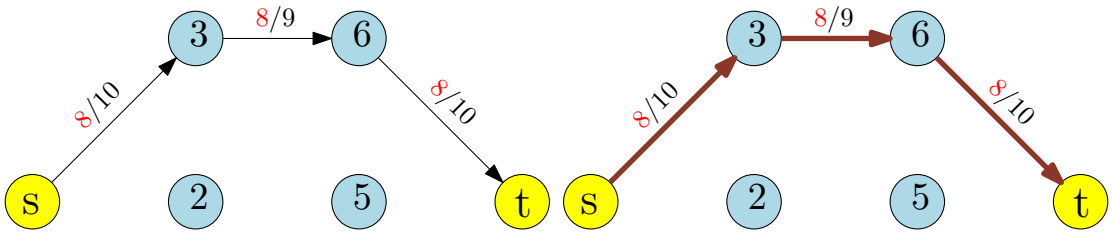


17.1.2.3 Example

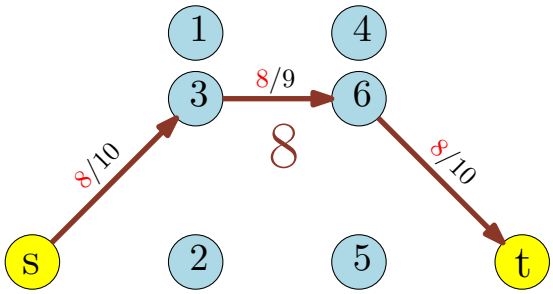


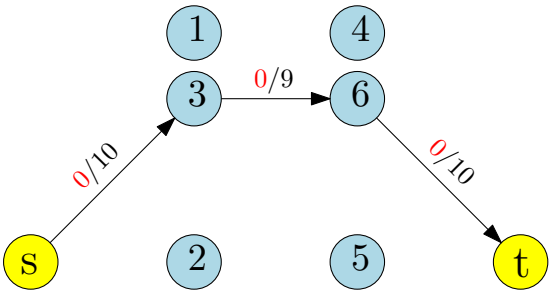

s
1
2
3
4
5
6
t
9/10
4/5
10/15
8/9
1/15
3/8
1/4
11/30
8/10
4/10
11/11
10
1
1/30
0/15
1/11
3/5
0/4
0/30
0/11

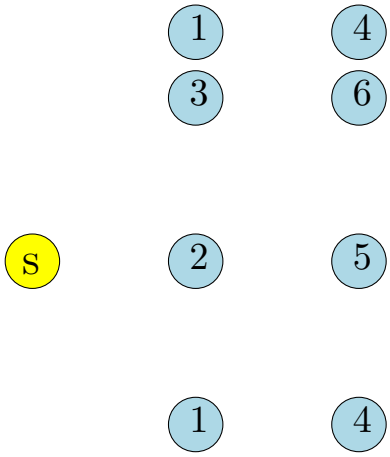

s
2
3
5
6
t
1
4
9/10
3/5
8/9
1/15
8/10
3/8
4/10
0/5
0/8
1/10
0/15
0/10
3
1

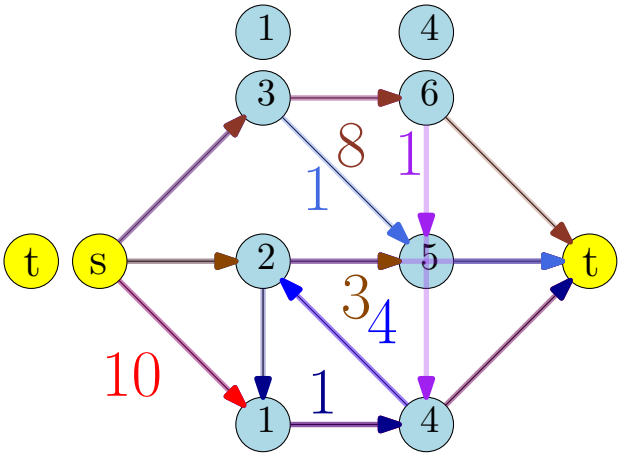

s
2
5
t
3
6
8/10
8/9
8/10

s
2
5
t
3
6
8/10
8/9
8/10


s
2
5
t
1
3
4
6
8/10
8/9
8
8/10

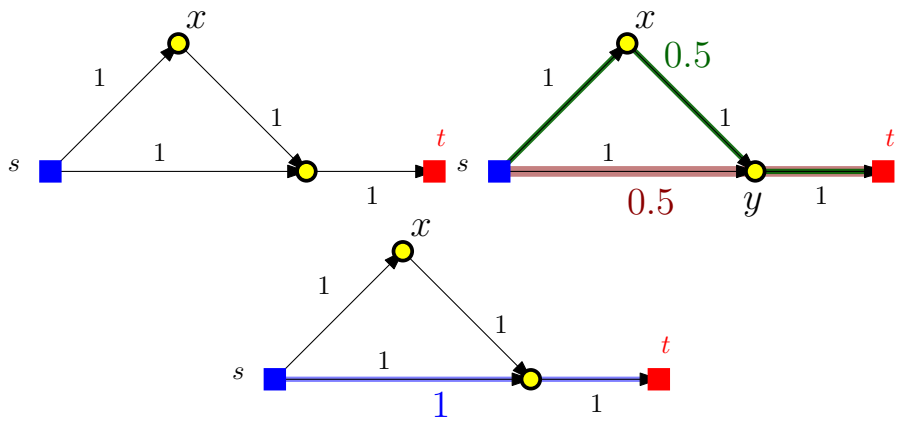

s
2
5
t
1
3
4
6
0/10
0/9
0/10


s
2
5
1
3
4
6


s
2
5
t
1
3
4
6
8
1
1
3
4
10
1

17.1.3 Path flow decomposition

17.1.3.1 Do not have to be efficient...



17.1.3.2 Edge vs Path based Definitions of Flow

(A) Edge based flows:
  (A) *compact* representation, only m values to be specified, and
  (B) need to check flow conservation explicitly at each internal node.

(B) Path flows:
  (A) in some applications, paths more natural,
  (B) not compact,
  (C) no need to check flow conservation constraints.

(C) Equivalence shows that we can go back and forth easily.

17.1.3.3 The Maximum-Flow Problem

(A) The ***network flow problem***: Problem

Input A network G with capacity c and source s and sink t.

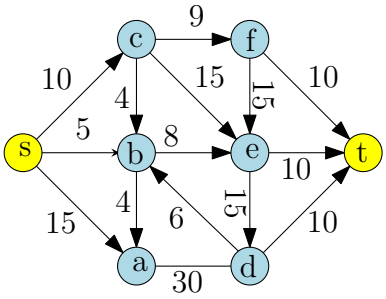
Goal Find flow of *maximum* value.

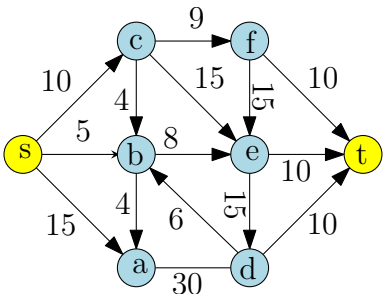
(B) **Question:** Given a flow network, what is an *upper bound* on the maximum flow between source and sink?

17.1.3.4 Cuts

Definition 17.1.6 (s-t cut). *Given a flow network an* **s-t cut** *is a set of edges $E' \subset E$ such that removing E'* disconnects *s from t: in other words there is no directed $s \to t$ path in $E - E'$.*

The **capacity** *of a cut E' is $c(E') = \sum_{e \in E'} c(e)$.*

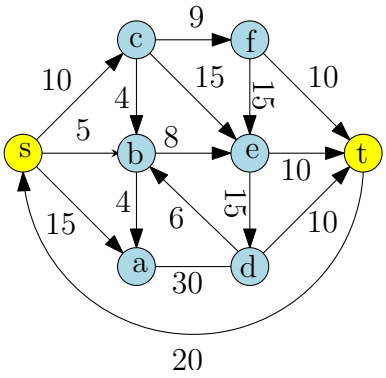


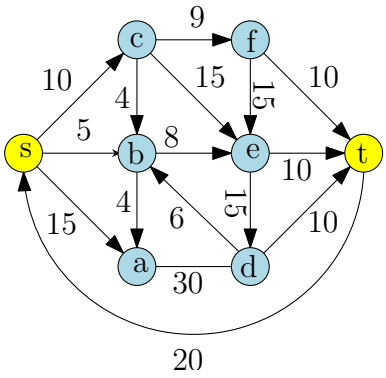


Caution:
(A) Cut may leave $t \to s$ paths!
(B) There might be many s-t cuts.

17.1.4 $s-t$ cuts

17.1.4.1 A death by a thousand cuts

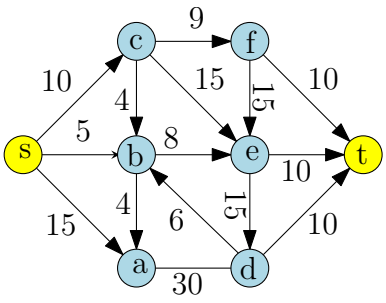




17.1.4.2 Minimal Cut

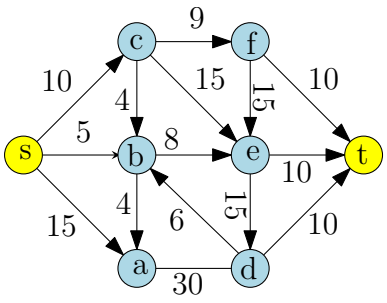
Definition 17.1.7 (Minimal *s-t* cut.). *Given a s-t flow network $G = (V, E)$, $E' \subseteq E$ is a* **minimal cut** *if for all $e \in E'$, if $E' \setminus \{e\}$ is not a cut.*

Observation: given a cut E', can check efficiently whether E' is a minimal cut or not. How?



17.1.4.3 Cuts as Vertex Partitions

(A) Let $A \subset V$ such that
  (A) $s \in A$, $t \notin A$, and
  (B) $B = V \setminus A$ (hence $t \in B$).
(B) The ***cut*** (A, B) is the set of edges $(A, B) = \{(u, v) \in E \mid u \in A, v \in B\}$.
  Cut (A, B) is set of edges leaving A.



Claim 17.1.8. *(A, B) is an s-t cut.*

Proof: Let P be any $s \to t$ path in G. Since t is not in A, P has to leave A via some edge (u, v) in (A, B). ■

17.1.4.4 Cuts as Vertex Partitions

Lemma 17.1.9. *Suppose E' is an s-t cut. Then there is a cut (A, B) such that $(A, B) \subseteq E'$.*

Proof: E' is an s-t cut implies no path from s to t in $(V, E - E')$.
(A) Let A be set of all nodes reachable by s in $(V, E - E')$.
(B) Since E' is a cut, $t \notin A$.
(C) $(A, B) \subseteq E'$. Why?If some edge $(u, v) \in (A, B)$ is not in E' then v will be reachable by s and should be in A, hence a contradiction.

■

Corollary 17.1.10. *Every* minimal *s-t cut E' is a cut of the form (A, B).*

17.1.4.5 Minimum Cut

Definition 17.1.11. *Given a flow network an s-$t* **minimum** *cut is a cut E' of smallest capacity among all s-t cuts.*

Observation: exponential number of s-t cuts and no "easy" algorithm to find a minimum cut.

17.1.4.6 The Minimum-Cut Problem

Problem

Input A flow network G

Goal Find the capacity of a *minimum* s-t cut

17.1.4.7 Flows and Cuts

Lemma 17.1.12. *For any s-t cut E',* **maximum** *s-t flow $\leq$ capacity of E'.*

Proof:
(A) Formal proof easier with path based definition of flow.
(B) Suppose $f : \mathcal{P} \to \mathbb{R}^{\geq 0}$ is a max-flow.
(C) Every path $p \in \mathcal{P}$ contains an edge $e \in E'$. Why?
(D) Assign each path $p \in \mathcal{P}$ to exactly one edge $e \in E'$.
(E) Let $\mathcal{P}_e$ be paths assigned to $e \in E'$. Then

$$v(f) = \sum_{p \in \mathcal{P}} f(p) = \sum_{e \in E'} \sum_{p \in \mathcal{P}_e} f(p) \leq \sum_{e \in E'} c(e).$$

■

17.1.4.8 Flows and Cuts

Lemma 17.1.13. *For any s-t cut E',* **maximum** *s-t flow $\leq$ capacity of E'.*

Corollary 17.1.14. *Maximum s-t flow $\leq$ minimum s-t cut.*

17.1.4.9 Max-Flow Min-Cut Theorem

Theorem 17.1.15. *In any flow network:*

$$\Big(\textit{value of maximum } s\text{-}t \textit{ flow}\Big) = \Big(\textit{capacity of minimum } s\text{-}t \textit{ cut}\Big).$$

(A) Can compute minimum-cut from maximum flow and vice-versa!
(B) Proof coming shortly.
(C) Many applications:
  (A) optimization
  (B) graph theory
  (C) combinatorics

17.1.4.10 The Maximum-Flow Problem

Problem

Input A network G with capacity c and source s and sink t.

Goal Find flow of *maximum* value from s to t.

Exercise: Given G, s, t as above, show that one can remove all edges into s and all edges out of t without affecting the flow value between s and t.

Bibliography

Dinic, E. A. (1970). Algorithm for solution of a problem of maximum flow in a network with power estimation. *Soviet Math. Doklady*, 11:1277–1280.

Edmonds, J. and Karp, R. M. (1972). Theoretical improvements in algorithmic efficiency for network flow problems. *J. Assoc. Comput. Mach.*, 19(2):248–264.