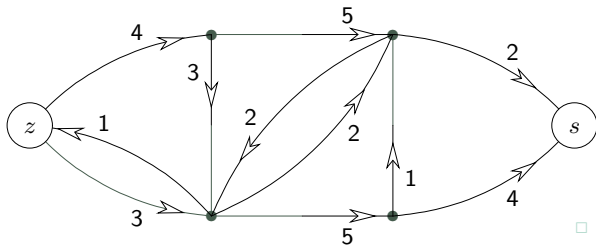


## 4 Network Flow Problems

Yet another area of rich applications of graphs (actually digraphs) deals with so called *networks* and “commodity flows” in them.

This time, the main optimization task is to maximize a *flow* from the designated *source* to the designated *sink*, respecting given constraints – *capacities* of network arcs.



### Brief outline of this lecture

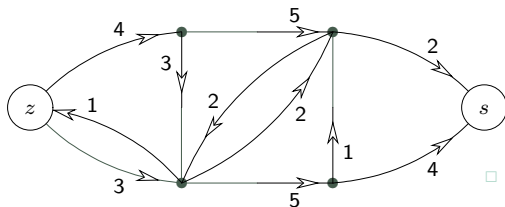
- Networks (weighted directed graphs) and flows in them.
- A network–flow algorithm(s) based on augmenting paths.
- Applications in connectivity, matching, and SDR problems.

## 4.1 Defining a flow network

The underlying structure of a flow network is a directed graph, with its vertices representing network nodes, and arcs representing the (existing or possible) connections between nodes. □

**Definition 4.1.** A **flow network** is a quadruple  $S = (G, z, s, w)$  such that

- $G$  is a digraph,
- the vertices  $z \in V(G)$ ,  $s \in V(G)$  are the **source** and the **sink**, respectively,
- and  $w : E(G) \rightarrow \mathbf{R}^+$  is a positive weighting of the arcs (edges) of  $G$ , these weights are called **edge capacities**.



**Remark:** In reality, more than one source or sink may exist in a flow network, but that is not a problem—we simply create a single artificial source and draw arcs from it to all the real sources (even with source capacities).

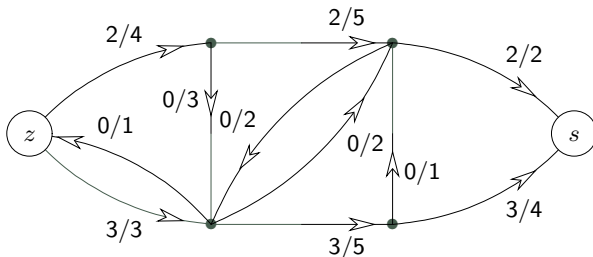
**Notation:** For simplicity, we shall write  $e \rightarrow v$  to mean that an arc  $e$  “comes to” (has its head in) the vertex  $v$ , and  $e \leftarrow v$  analogously for  $e$  “leaving” (having tail in)  $v$ .  $\square$

**Definition 4.2. A network flow**, in a flow network  $S = (G, z, s, w)$ , is an assignment  $f : E(G) \rightarrow \mathbf{R}_0^+$  satisfying (we say  $f$  is *admissible*)

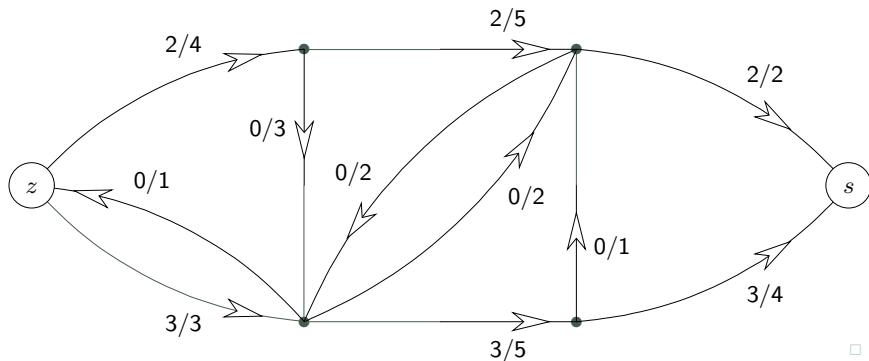
- $\forall e \in E(G) : 0 \leq f(e) \leq w(e)$ , (respecting capacities)
- $\forall v \in V(G), v \neq z, s : \sum_{e \rightarrow v} f(e) = \sum_{e \leftarrow v} f(e)$ . (flow conservation)  $\square$

The *size (value)* of a flow  $f$  is the quantity  $\|f\| = \sum_{e \leftarrow z} f(e) - \sum_{e \rightarrow z} f(e)$ .  $\square$

**Notation:** The flow value  $F$  and the capacity  $C$  of an arc in a picture of a network will be shortly denoted by  $F/C$ , respectively.



So what is the value of the depicted flow? 5



**Remark:** Notice the following simple identity

$$0 = \sum_{e \in E} (f(e) - f(e)) = \sum_{v \in V} \left( \sum_{e \leftarrow v} f(e) - \sum_{e \rightarrow v} f(e) \right) = \sum_{v=z, s} \left( \sum_{e \leftarrow v} f(e) - \sum_{e \rightarrow v} f(e) \right).$$

What interesting does it tell us? Briefly, that the (opposite) flow value can be analogically defined at the sink  $s$ , or more generally at any “ $z$ - $s$  cut”.

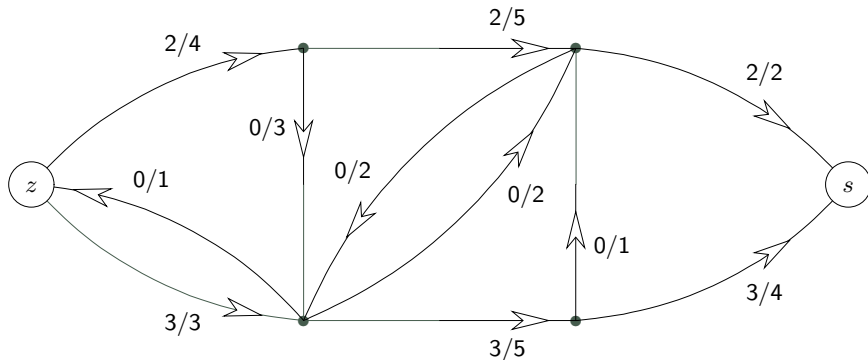
$$\|f\| = \left( \sum_{e \leftarrow z} f(e) - \sum_{e \rightarrow z} f(e) \right) = \left( \sum_{e \rightarrow s} f(e) - \sum_{e \leftarrow s} f(e) \right).$$

## 4.2 Finding the maximum flow value

There exist quite simple and fast algorithms to determine the maximum flow value in a given network.

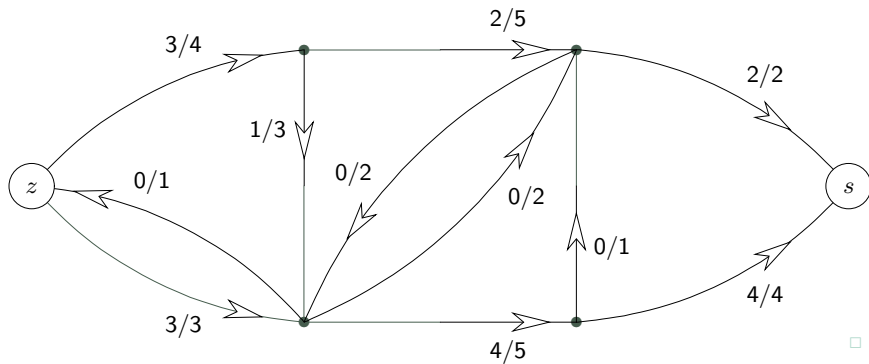
### Problem 4.3. The Max-Flow problem

Given a flow network  $S = (G, z, s, w)$ , the task is to find a flow  $f$  in  $S$  from  $z$  to  $s$  such that the value  $\|f\|$  is maximized (among all admissible flows in  $S$ ).  $\square$



Is the depicted flow really maximum possible?

And what about this flow of value 6?



The main question is; how can we certify that no better flow exists in  $S$ ?

Fortunately, a nice and simply understandable criterion exists there — notice that there is an “arc cut” between  $z$  and  $s$  of total value 6, and hence **obviously** there cannot be a larger flow!

## Cut in a network

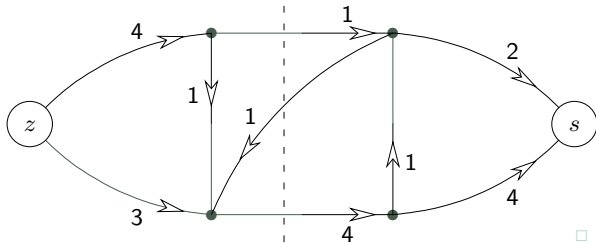
**Definition 4.4.** A **cut** in a flow network  $S = (G, z, s, w)$

is a subset of edges (arcs)  $C \subset E(G)$  such that there is **no**  $z \rightarrow s$  directed path in  $G$  completely avoiding  $C$  (i.e. no directed  $z \rightarrow s$  path existing in  $G - C$ ).

The **size** of a cut  $C$  is the sum of the capacities of arcs in  $C$ , i.e.  $\|C\| = \sum_{e \in C} w(e)$ .  $\square$

**Theorem 4.5.** The maximum (admissible) flow value in a network  $S$  equals the minimum cut size in  $S$ .

See the following example with a cut of size 5 (in the middle), and so the flow value 5 is maximum.



**Remark:** Theorem 4.5 is an example of a so called **good characterization** of a property — not having a larger flow can be certified by showing an **obvious constraint**, i.e. a cut of the corresponding size.

## Residual and augmenting paths

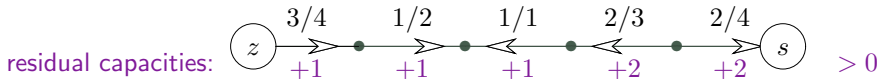
**Definition:** Consider a flow network  $S$  and a flow  $f$  in it. A *residual  $z$ - $x$  path* (in  $S$  w.r.t.  $f$  and source  $z$ ) is an **undirected** path in  $G$  from the source  $z$  to any vertex  $x$ , i.e. a sequence of adjacent edges  $e_1, e_2, \dots, e_m$ ;

- such that  $f(e_i) < w(e_i)$  if  $e_i$  is directed from  $z$ ,
- and  $f(e_i) > 0$  if  $e_i$  is directed from  $s$ .  $\square$

The quantity  $w(e_i) - f(e_i)$ , or  $f(e_i)$ , resp., is called the *residual capacity* of edge  $e_i$ .

$\square$

A residual path is that of strictly positive residual capacities...



$\square$

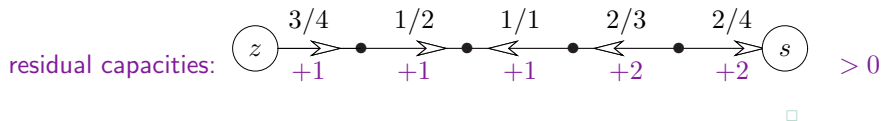
### Method 4.6. Maximizing flow via residual paths.

The overall idea is *really simple*;

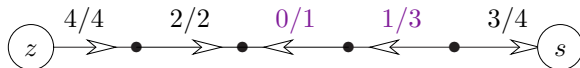
one should repeatedly enlarge the flow by adding to it along residual paths...



How to augment a residual path...

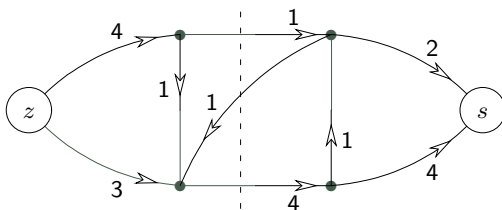


min. residual capacity  $r = 1 > 0 \rightsquigarrow$  augmenting the current flow by +1: □



**Fact:** Augmenting (enlarging) an admissible flow by the minimal residual capacity of a residual  $z-s$  path results in an admissible flow, again.

## A residual path algorithm



### Algorithm 4.7. Ford–Fulkerson’s for network flows.

input  $<$  a flow network  $S = (G, z, s, w)$ ;

flow  $f \equiv 0$ ;

repeat {

    Search (BFS) the graph  $G$  to find the set  $U$  of those vertices accessible from the source  $z$  along residual paths;

    if ( $s \in U$ ) {

$P$  = any residual  $z$ – $s$  path in  $S$  (this  $P$  then called an *augmenting path*);  
        Augment (“enlarge”)  $f$  by the minimal residual capacity along  $P$ ;

    }

until ( $s \notin U$ );

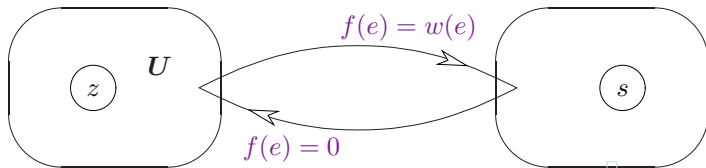
output  $>$  a maximum flow  $f$  in  $S$ ;

output  $>$  a minimum cut in  $S$  from  $U$  to  $V(G) - U$ .

**Proof** of Algorithm 4.7:

For any flow  $f$  and any cut  $C$  in  $S$ , it holds  $\|f\| \leq \|C\|$ . If the algorithm stops with a flow  $f$  in  $S$  and a cut  $C$  such that  $\|C\| = \|f\|$ , then it is clear that  $f$  is a maximum flow in  $S$ . (We have, however, not proved yet that the algorithm stops!)  $\square$

So to prove that whenever the algorithm stops with  $f, C$ , then  $\|f\| = \|C\|$ , we use the following schematic picture (in which  $s$  does not belong to the “accessible” set  $U$ ):



Since no further vertex than  $U$  is accessible along residual paths, every arc  $e$  leaving  $U$  has full flow  $f(e) = w(e)$ , and every arc  $e$  entering  $U$  has zero flow  $f(e) = 0$ . Therefore;

$$\sum_{e \leftarrow U} f(e) - \sum_{e \rightarrow U} f(e) = \sum_{e \leftarrow U} f(e) = \sum_{e \in C} w(e) = \|C\| .$$

That is nice, and it remains to argue that  $\|C\| = \sum_{e \leftarrow U} f(e) - \sum_{e \rightarrow U} f(e) = \|f\|$ , finishing the proof.  $\square$

The proof of Algorithm 4.7 shows several interesting things:

**Fact:** If we can prove that Algorithm 4.7 stops, we prove also Theorem 4.5.  $\square$

**Fact:** If the edge capacities in  $S$  are integral, then Algorithm 4.7 always stops.

**Corollary 4.8.** *If the edge capacities in  $S$  are integral, then Algorithm 4.7 outputs an integral flow.*  $\square$

## Improved flow algorithms

Generally, one cannot guarantee that Algorithm 4.7 stops, since counterexamples exist with real capacities, but we can do better as follows.  $\square$

### Algorithm 4.9. Edmonds–Karp’s for network flows.

As in Algorithm 4.7, we always enlarge one of the *shortest* residual paths in  $G$  (i.e. find the path  $P$  using BFS, cf. Corollary 3.4).

This implementation is guaranteed to stop after  $O(|V(G)| \cdot |E(G)|)$  iterations, so in total computing time  $O(|V(G)| \cdot |E(G)|^2)$ .

## Improved flow algorithms, II

Even better, we can use the following “clever” algorithms.

### Algorithm 4.10. Dinitz’s for network flows (a sketch).

We modify Algorithm 4.7 with the following iteration:

- Using BFS, we find *all* the shortest residual paths in  $S$ , creating a “layered” residual network.
- The layered network is then completely saturated in one run.

This implementation makes only  $O(|V(G)|)$  iterations of the main cycle. Total computing time now is  $O(|V(G)|^2 \cdot |E(G)|)$ .  $\square$

### Algorithm 4.11. MPM “Three Indians” (a sketch).

Same as Algorithm 4.10, except that a layered network is saturated faster, and the total computing time is  $O(|V(G)|^3)$ .

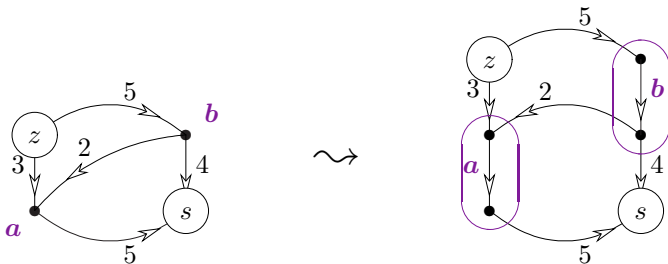
## 4.3 Generalized network flow settings

### Networks with vertex capacities

In a flow network with *vertex capacities* (of course, retaining edge capacities as well), the weight function is  $w : E(G) \cup V(G) \rightarrow \mathbf{R}^+$ .

The meaning for admissible flows is that the total sum of incoming flow to any vertex  $x$  is not more than  $w(x)$ . (Differently applicable to the source or the sink, though. . . )  $\square$

**Fact.** Such a generalized flow network can easily be translated to an ordinary network via “doubling” the capacitated vertices (replacing them with new arcs between the two copies), as follows.



## Networks with lower capacities

In a flow network with *lower capacities*, in addition to the weight function  $w$ , there is another weight function  $\ell : E(G) \rightarrow \mathbf{R}_0^+$  giving the lower edge capacities.

A flow  $f$  is admissible in such a network if  $\ell(e) \leq f(e) \leq w(e)$  for every edge  $e$  of the network. □ Notice that an **admissible flow may not exist** in such a lower-capacitated network. □

### Algorithm 4.12. Flows in lower-capacitated network

*For this kind of a flow network, the solution has two steps.*

- *First, an admissible **circulation** is found (with a “back-arc”  $sz$ ), respecting both the lower and upper bounds  $\ell, w$ . This is done by finding a maximum flow in an artificial network modelling the “surplus” of lower capacities at every vertex. . .*
- *Second, this admissible circulation is enlarged by a maximum possible **excessive** flow from  $z$  to  $s$  (the capacities are now  $w(e) - r(e)$  where  $r$  is the circulation found above). □*

## Multicommodity flows

This is a difficult problem setting reaching beyond the scope of our lecture.

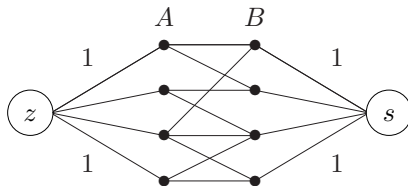
## 4.4 Other applications of network flows

### Bipartite matching

**Definition:** A *matching* in a graph  $G$  (bipartite here) is a subset of edges  $M \subset E(G)$  such that no two edges from  $M$  share a vertex.  $\square$

#### Algorithm 4.13. Finding maximum matching in a bipartite graph

Given a bipartite graph  $G$  with the vertex parts  $A, B$ , we construct the following flow network  $S$ :



All the edges are implicitly directed from the source towards the sink, and all capacities are 1. We run Algorithm 4.7, and form the resulting matching  $M$  by those edges of  $G$  having nonzero flow at the end.  $\square$

**Proof** of Algorithm 4.13: By Corollary 4.8, the maximum flow found by our algorithm is integral, which now means that each flow unit is 0 or 1. Hence no two edges of  $M$  could share a vertex. Conversely, any matching  $M'$  gives an admissible flow in  $S$ .  $\square$



## Higher graph connectivity

Let us consider a graph  $G$  as a generalized (symmetrically-oriented) network with all vertex capacities equal to 1. Then the network flow theorem immediately says:

**Corollary 4.14.** *Let  $u, v$  be two vertices of  $G$  and  $k > 0$  be an integer. Then there exists at least  $k$  internally disjoint  $u$ - $v$  paths in  $G$  if, and only if, removing any subset of at most  $k - 1$  vertices of  $G$  (other than  $u, v$ ) does not disconnect  $u$  from  $v$ .  $\square$*

This statement immediately implies Theorem 2.7 (Menger's)!

## Systems of distinct representatives (SDR)

**Definition:** Let  $M_1, M_2, \dots, M_k$  be a collection of nonempty sets. A *system of distinct representatives (SDR)* of the set family  $\{M_1, M_2, \dots, M_k\}$  is a sequence of pairwise *distinct* elements  $(x_1, x_2, \dots, x_k)$  such that  $x_i \in M_i$  for  $i = 1, 2, \dots, k$ .  $\square$

**Theorem 4.15.** (Hall) *Let  $\{M_1, M_2, \dots, M_k\}$  be a family of nonempty sets. Then there exists a system of its distinct representatives if, and only if,*

$$\forall J \subset \{1, 2, \dots, k\} : \left| \bigcup_{j \in J} M_j \right| \geq |J|,$$

*i.e., the union of any subfamily of these sets has at least that many elements as the number of sets in it.*

Necessity of Hall's condition in this theorem is obvious, and its sufficiency can be proved by an application of network flows again.

## Flows in image segmentation

Yet another profitable application of flow networks lies in the computer vision area:

The basic *image segmentation* problem asks for decomposing a given image into a foreground and a background.

- Let the input consists of a pixel matrix, each pixel carrying two values of likelihood to be in the foreground and in the background. Additionally, separation penalties are assigned to the neighbouring pairs of pixels. □
- The network is constructed by introducing a source  $z$  and a sink  $s$  such that the arcs from  $z$  to the pixels have their capacities equal to the foreground likelihood, and the arcs from the pixels to  $s$  have their capacities equal to the background likelihood. Additional bidirectional arcs join the neighbouring pixel pairs.
- A minimal cut in this network then defines the separation between foreground and background parts.