The Ford-Fulkerson Algorithm and The Max-Flow Min-Cut Theorem

CSCI 4113/6101
Instructor: Norbert Zeh
September 21, 2025

The first class of maximum flow algorithms uses a fairly natural strategy for finding maximum flows: start by sending no flow at all along any edge—this is clearly a feasible flow—and then repeatedly find paths along which we can send more flow from s to t. By sending flow along such a path, we increase, or augment, the amount of flow sent from s to t. Therefore, these algorithms are called **augmenting path algorithms**.

In this topic, we discuss the **Ford-Fulkerson algorithm**, which is the most elementary augmenting path algorithm. All other augmenting path algorithms can be viewed as variants of the Ford-Fulkerson algorithm. The Ford-Fulkerson algorithm has the property that it finds a maximum flow *if it terminates*. Unfortunately, the Fold-Fulkerson algorithm may not always terminate, as we will illustrate using a carefully crafted example. In the next topic, we will discuss the Edmonds-Karp algorithm, whose only difference to the Ford-Fulkerson algorithm is that it sends flow along the *shortest* available augmenting path in each iteration. We will be able to prove that this ensures that the Edmonds-Karp algorithm runs in $O\left(nm^2\right)$ time, where n is the number of vertices, and m is the number of edges of the graph. To prove the correctness of any maximum flow algorithm, we need the **Max-Flow Min-Cut Theorem**. We will prove this theorem in this topic and use it to prove that (any variant of) the Ford-Fulkerson algorithm does compute a maximum flow, if it terminates.

This topic is organized as follows. In § 1, we introduce the concept of a residual network, which is a tool used by all maximum flow algorithms and which we will use here to define what an augmenting path is. Sec. 2 discusses the Ford-Fulkerson algorithm. Sec. 3 presents an example that demonstrates that the Ford-Fulkerson algorithm may not terminate. We prove the correctness of the Ford-Fulkerson algorithm in two parts: Sec. 4 proves that at all times during the execution of the Ford-Fulkerson algorithm, the current flow is a feasible flow. Sec. 6 proves that if the Ford-Fulkerson algorithm terminates, then the flow it returns is a maximum flow. To prove this, we need the Max-Flow Min-Cut Theorem presented in § 5. The final, optional section, § 7, looks at the Max-Flow Min-Cut Theorem through the lens of linear programming and demonstrates that it is in fact simply an application of LP duality.

1 The Residual Network and Augmenting Paths

Assume we are given a network $G = (V, E)$ along with edge capacities $u : E \to \mathbb{R}_0^+$, as well as a feasible flow $f : E \to \mathbb{R}_0^+$ in G. Augmenting path algorithms are based on two key observations:

(i) Assume there exists a path from s to t such that every edge $e \in P$ satisfies $u_e - f_e > 0$. In other words, the flow f does not use any of the edges in P to its full capacity. Then P is an **augmenting**

path in the sense that we can send additional flow along this path: If $\delta = \min\{u_e - f_e \mid e \in P\}$, then the *st*-flow f' defined as

$$f'_e = \begin{cases} f_e + \delta & \text{if } e \in P \\ f_e & \text{otherwise} \end{cases}$$

is feasible and satisfies $F'_s > F_s$. See Figs. 1a,b.

(ii) An augmenting path can also move flow "backwards" along an edge (x, y), from y to x, if $f_{x,y} > 0$. Indeed, this amounts to sending some of the flow currently flowing from x to y back to x, that is, reducing the flow $f_{x,y}$ along the edge (x, y) and diverting the sent-back flow to a different out-neighbour of x. See Figs. 1c,d.

Based on these two observations, the set of edges along which we can try to move more flow from s to t is the set

$$E^f = \{(x, y) \in E \mid f_{x,y} < c_{x,y}\} \cup \{(y, x) \mid (x, y) \in E, f_{x,y} > 0\}.$$

We define the **residual network** of G with respect to f as $G^f = \left(V, E^f\right)$ (see Fig. 1e). The name "residual network" refers to the fact that it tells us how much capacity of an edge remains to be used. The **residual capacity** $u^f_{x,y}$ of an edge $(x, y) \in E^f$ defines how much flow we can move along the edge (x, y). It is defined as

$$u^f_{x,y} = u_{x,y} - f_{x,y} + f_{y,x},$$

that is, as the amount of flow we can still move from x to y along the edge (x, y) plus the amount of flow we can move back from x to y along the edge (y, x). The graph G may contain both edges (x, y) and (y, x). With this definition of the residual network in place, we can define an augmenting path formally:

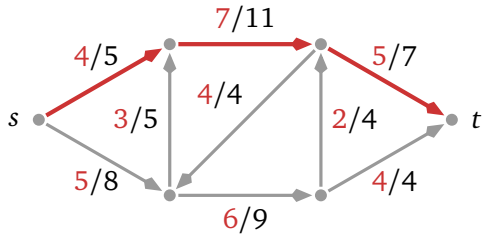
Definition 1 (Augmenting path)**.** Given a feasible *st*-flow f in G, an **augmenting path** P is a path from s to t in the residual network G^f.

Note 2. Here, we defined the residual network with respect to a *feasible* flow, one that respects the edge capacities and flow conservation. It should be obvious that flow conservation is irrelevant to this definition: As long as the flow along every edge is non-negative and does not exceed the capacity of the edge, the residual capacity of every edge is non-negative. This will be important for the discussion of push-relabel algorithms, discussed in an upcoming topic, which use a different strategy to find a maximum flow. As long as such an algorithm has not found a maximum flow yet, the current flow satisfies the capacity constraints but does not satisfy flow conservation.

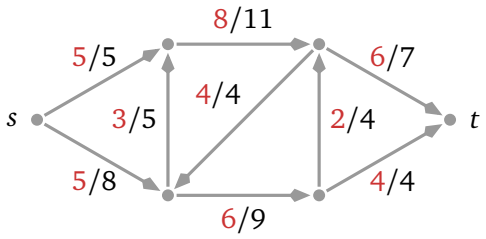
2 The Ford-Fulkerson Algorithm

Every augmenting path algorithm follows the template in the MaxFlow procedure on page 5. The algorithm starts with a feasible *st*-flow f that sends no flow along any edge. Then, as long as there exists an *st*-path P in G^f, the algorithm sends more flow from s to t along this path P. We prove in § 6 that once there is no *st*-path in G^f, f is a maximum *st*-flow in G. So the algorithm terminates and reports f.

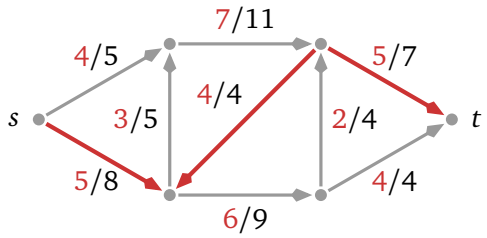
The details of sending flow along an augmenting path P are implemented as a separate procedure Augment. This procedure initializes the augmented flow f' to be the same as the current flow f. Then it inspects all edges in P and finds the minimum residual capacity δ of all edges in P. This is the maximum



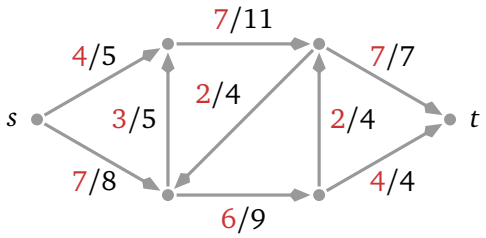
(a) An augmenting path that uses only edges in G



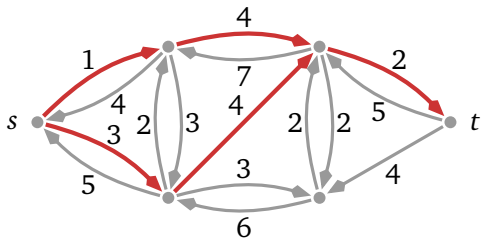
(b) The flow obtained by sending one unit of flow along the path in Fig. 1a



(c) An augmenting path that uses an edge of G in reverse



(d) The flow obtained by sending two units of flow along the path in Fig. 1c



(e) The residual network G^f corresponding to the flow in Figs. 1a,c. The augmenting paths in Figs. 1a,c are directed paths in G^f (red).

Figure 1: Augmenting paths and the residual network

amount of flow we can send along P without violating the capacity constraint of any edge. Finally, AUGMENT sends δ units of flow along P, by reducing the flow of any edge (y,x) such that $(x,y) \in P$ by $\min(\delta, f_{y,x})$ and increasing the flow along any edge (x,y) such that $(x,y) \in P$ by $\max(0, \delta - f_{y,x})$. The resulting flow f' is the new flow to be used in the next iteration of MAXFLOW.

The procedures MAXFLOW and AUGMENT together constitute the Ford-Fulkerson algorithm. For the sake of clarity, the MAXFLOW procedure constructs the residual network G^f from G and f in each iteration of the loop. An actual implementation of the algorithm initializes $G^f = G$ at the beginning of the algorithm (because the flow is zero initially) and then, in each iteration, updates only the edges in G^f involved in the augmenting path P found in the current iteration, since these are the only edges whose residual capacities or presence or absence in G^f are affected by changing the flow along the edges in P.

As we show in the next section, the Ford-Fulkerson algorithm may not terminate if it happens to choose the augmenting path along which to send more flow from s to t poorly. There are various strategies for choosing this path carefully in each iteration, to guarantee that the algorithm terminates, and to ensure that it does so after only a small number of iterations. One such strategy is to send flow along a *shortest* path from s to t in G^f in each iteration. This is the strategy employed by the Edmonds-Karp algorithm, discussed in the next topic.

3 FORD-FULKERSON MAY NOT TERMINATE

The following example shows that the Ford-Fulkerson algorithm may not terminate on some inputs. Consider the network in Fig. 2a, where $\rho = \frac{\sqrt{5}-1}{2}$ and $X > 2$. The red edges are edges used by the maximum flow in this network: we send X units of flow along the left and right path, and one unit of flow along the middle path. Thus, the maximum flow sends $2X + 1$ units of flow from s to t.

Since the initial flow in the Ford-Fulkerson algorithm is 0 along all edges, the network is its own residual network initially. Since the Ford-Fulkerson algorithm may choose to send flow along any augmenting path it finds in G^f, it may choose to send flow along the middle path first. This is shown in Fig. 2b. Sending one unit of flow along this path produces the residual network shown in Fig. 2c for $i = 1$; the capacities of the edges incident to s or t are not shown in Figs. 2c–f because X is large enough that these edges are not the ones that limit the amount of flow sent along any augmenting path used in this example.

Next we show that given the residual network in Fig. 2c, for some integer i, the next four iterations of the Ford-Fulkerson algorithm produce the exact same residual network, only i is increased by 2. Thus, the algorithm produces residual networks whose augmenting paths have smaller and smaller capacities, but there is always an augmenting path left, so the algorithm never terminates:

- The red path in Fig. 2c is an augmenting path of capacity ρ^i. If the Ford-Fulkerson algorithm chooses this path and moves ρ^i units of flow along this path, this produces the residual network in Fig. 2d.
- The red path in Fig. 2d is an augmenting path of capacity ρ^i. If the Ford-Fulkerson algorithm chooses this path and moves ρ^i units of flow along this path, this produces the residual network in Fig. 2e.
- The red path in Fig. 2e is an augmenting path of capacity ρ^{i+1}. If the Ford-Fulkerson algorithm chooses this path and moves ρ^{i+1} units of flow along this path, this produces the residual network in Fig. 2f.

Procedure MAXFLOW(G, u, s, t)

Input: A directed graph $G = (V, E)$, a capacity labelling $u : E \to \mathbb{R}_0^+$ of its edges, and a pair of vertices (s, t)

Output: A maximum st-flow $f : E \to \mathbb{R}_0^+$ in G

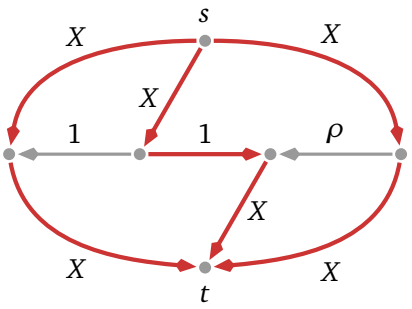
```
1 forall e ∈ E do
2     f_e = 0
3 loop
4     G^f = residual network of G w.r.t. f
5     if there exists an st-path P in G^f then
6         f = AUGMENT(G, u, f, P)
7     else
8         return f
```

Procedure AUGMENT(G, u, f, P)

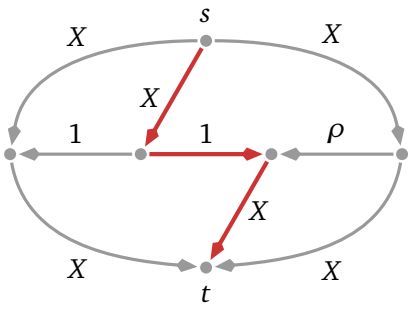
Input: A directed graph $G = (V, E)$, a capacity labelling $u : E \to \mathbb{R}_0^+$, a feasible st-flow $f : E \to \mathbb{R}_0^+$, and an st-path P in G^f

Output: An augmented st-flow f'

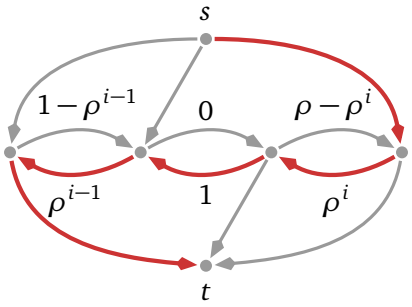
```
1 forall e ∈ E do
2     f'_e = f_e
3 δ = min{ u^f_e | e ∈ W }
4 forall (x, y) ∈ P do
5     f'_{y,x} = f_{y,x} − min(δ, f_{x,y})
6     f'_{x,y} = f_{x,y} + max(0, δ − f_{y,x})
7 return f'
```



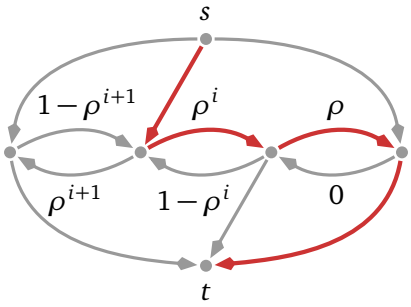
(a) The network and a maximum flow



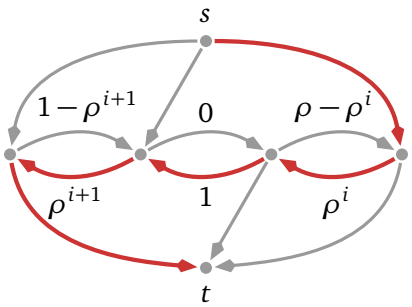
(b) The initial network



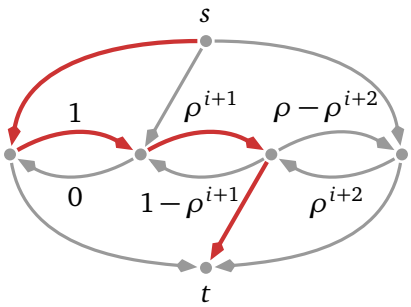
(c) Step 1 of a phase



(d) Step 2 of a phase



(e) Step 3 of a phase



(f) Step 4 of a phase

Figure 2: A non-terminating example for the Ford-Fulkerson algorithm

- The red path in Fig. 2f is an augmenting path of capacity ρ^{i+1}. If the Ford-Fulkerson algorithm chooses this path and moves ρ^{i+1} units of flow along this path, this produces the residual network in Fig. 2c, with i increased by 2.

The Ford-Fulkerson algorithm is an example of an iterative optimization algorithm. These algorithms start with an arbitrary feasible solution and improve this solution in each iteration while maintaining its feasibility. Many algorithms of this type make rapid progress towards an optimal solution in early iterations. As the algorithm progresses, the improvements become smaller and smaller. A good strategy to limit the running time of such an algorithm then is to terminate the algorithm after a fixed number of iterations. The solution the algorithm has found up to this point may not be an optimal solution, but it is often a very good solution, as the early iterations make the greatest progress towards an optimal solution.

Unfortunately, as this example shows, this strategy works poorly for the Ford-Fulkerson algorithm: The initial iteration of the algorithm in this example moves one unit of flow from s to t. Each group of four iterations after this initial iteration moves $2(\rho^i + \rho^{i+1})$ units of flow from s to t, for $i = 1, 3, 5, \ldots$. Thus, the total amount of flow moved from s to t converges to

$$1 + \sum_{i=1}^{\infty} 2\rho^i = \sum_{i=0}^{\infty} 2\rho^i - 1 = \frac{4}{3-\sqrt{5}} - 1 < 5$$

while the maximum st-flow shown in Fig. 2a has the value $2X + 1 > 5$. By using a large value of X, the gap between the maximum st-flow and the limit of the st-flows produced by the algorithm can be made arbitrarily large.

4 Ford-Fulkerson Maintains a Feasible Flow

The first half of the correctness proof of the Ford-Fulkerson algorithm is to show that the flow f maintained by the algorithm is a feasible flow. Since the initial flow satisfies $f_e = 0$, for all $e \in G$, it is certainly a feasible flow. Thus, to prove this claim, it suffices to prove that the flow returned by AUGMENT is feasible if the flow given as input to AUGMENT is feasible.

Lemma 3. *If f is a feasible st-flow in G, then so is the flow f' returned by* AUGMENT.

Proof. Let $f^\delta = f' - f$, that is, for every pair of vertices $x, y \in V$, we have $f^\delta_{x,y} = f'_{x,y} - f_{x,y}$. It's okay if theses values are negative. There is no expectation that f^δ is a valid flow. Then

$$f^\delta_{x,y} = \begin{cases} \max\bigl(0, \delta - f_{y,x}\bigr) & \text{if } (x,y) \in P \\ -\min\bigl(\delta, f_{x,y}\bigr) & \text{if } (y,x) \in P \\ 0 & \text{otherwise.} \end{cases}$$

Since f is a valid flow, and thus satisfies flow conservation, and $f' = f + f^\delta$, f' if satisfies flow conservation if f^δ does. In other words, we need to prove that

$$\sum_{y \in V} \left(f^\delta_{x,y} - f^\delta_{y,x} \right) = 0 \quad \forall x \in V \setminus \{s, t\}. \tag{1}$$

To prove (1) for all $x \notin P$, observe that $f^{\delta}_{x,y} = f^{\delta}_{y,x} = 0$ for all $x \notin P$ and all $y \in V$ because $(x,y),(y,x) \notin P$ if $x \notin P$.

To prove (1) for all $x \in P \setminus \{s,t\}$, let y be x's predecessor in P and let z be x's successor in P. Then

$$\begin{aligned}
f^{\delta}_{y,x} &= \max\bigl(0, \delta - f_{x,y}\bigr),\\
f^{\delta}_{x,y} &= -\min\bigl(\delta, f_{x,y}\bigr),\\
f^{\delta}_{x,z} &= \max\bigl(0, \delta - f_{z,x}\bigr),\\
f^{\delta}_{z,x} &= -\min\bigl(\delta, f_{z,x}\bigr), \text{ and}\\
f^{\delta}_{x,v} &= f^{\delta}_{v,x} = 0 \quad \forall v \notin \{y,z\}.
\end{aligned}$$

Thus,

$$\begin{aligned}
\sum_{v \in V}\left(f^{\delta}_{x,v} - f^{\delta}_{v,x}\right) &= f^{\delta}_{x,y} + f^{\delta}_{x,z} - f^{\delta}_{y,x} - f^{\delta}_{z,x}\\
&= -\min\bigl(\delta, f_{x,y}\bigr) - \max\bigl(0, \delta - f_{x,y}\bigr) + \max\bigl(0, \delta - f_{z,x}\bigr) + \min\bigl(\delta, f_{z,x}\bigr)\\
&= -\delta + \delta\\
&= 0.
\end{aligned}$$

To prove that f' respects the capacity constraints, consider any pair of vertices (x,y).

If $(x,y),(y,x) \notin P$, then $f'_{x,y} = f_{x,y}$. Since $0 \le f_{x,y} \le u_{x,y}$, this implies that $0 \le f'_{x,y} \le u_{x,y}$.

If $(x,y) \in P$, then

$$0 \le f_{y,x} - \min\bigl(\delta, f_{y,x}\bigr) \le f_{y,x} \le u_{y,x}$$

and

$$\begin{aligned}
0 \le f_{x,y} &\le f_{x,y} + \max\bigl(0, \delta - f_{y,x}\bigr)\\
&\le \max\bigl(f_{x,y}, f_{x,y} + u_{x,y} - f_{x,y} + f_{y,x} - f_{y,x}\bigr)\\
&= \max\bigl(f_{x,y}, u_{x,y}\bigr)\\
&= u_{x,y}
\end{aligned}$$

because f is a feasible st-flow and $0 < \delta \le u^{f}_{x,y} = u_{x,y} - f_{x,y} + f_{y,x}$. Since $f'_{x,y} = f_{x,y} + \max\bigl(0, \delta - f_{y,x}\bigr)$ and $f'_{y,x} = f_{y,x} - \min\bigl(\delta, f_{y,x}\bigr)$, this shows that $0 \le f'_{x,y} \le u_{x,y}$ and $0 \le f'_{y,x} \le u_{y,x}$. Thus, f' satisfies the capacity constraints. Since it also satisfies flow conservation, f' is a feasible st-flow. □

Since the st-flow f' returned by Augment is feasible and sends more flow from s to t than f—$F'_s > F_s$—we conclude that as long as there exists an augmenting path in G^f, f cannot be a maximum flow. Here is the contrapositive of this statement:

COROLLARY 4. *If f is a maximum st-flow in G, then there exists no augmenting path in G^f.*

5 THE MAX-FLOW MIN-CUT THEOREM

Assume now that the Ford-Fulkerson algorithm happens to terminate, or that we use a variant of the Ford-Fulkerson algorithm that is guaranteed to terminate, such as the Edmonds-Karp algorithm. How do we prove that the flow it outputs is a maximum flow? We have shown already that at the very least, the returned flow is a feasible flow. In this section, we develop the Max-Flow Min-Cut Theorem, which characterizes when a feasible flow is a maximum flow. We use this theorem in the next section to prove that, if the Ford-Fulkerson algorithm terminates, then the flow it returns is a maximum flow.

Recall the definition of a cut in a graph. This was simply a partition of the vertex set into two non-empty subsets S and T. Since $T = V \setminus S$ in this case, the cut is fully characterized by specifying S, which must be a non-empty proper subset of V for both S and T to be non-empty. Thus, we defined a cut to be simply a subset S of vertices that satisfies $\emptyset \subset S \subset V$.

An ***st*-cut** is a cut S with $s \in S$ and $t \notin S$. We define the capacity of this cut as the total capacity of the edges that cross from S to $T = V \setminus S$:

$$U_S = \sum_{x \in S, y \notin S} u_{x,y}$$

Similarly, the flow from S to T is the total amount of flow across all edges from S to T minus the total amount of flow across the edges back from T to S:

$$F_S = \sum_{x \in S, y \notin S} \left(f_{x,y} - f_{y,x} \right)$$

These definitions are illustrated in Fig. 3. Note that the definition of F_S considers the flow across the "backward edges" from T to S, but the definition of U_S considers only the capacities of the "forward edges" from S to T.

A **minimum *st*-cut** is an st-cut of minimum capacity. First, it is helpful to understand how st-cuts interact with flows from s to t.

LEMMA 5. *Let S be an st-cut of G and let f be a feasible st-flow in G. Then $F_s = F_S$.*

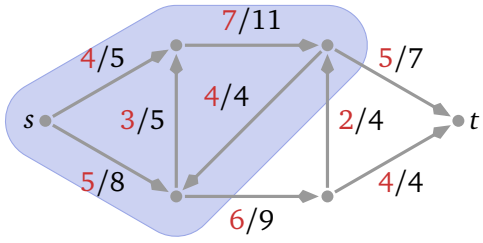
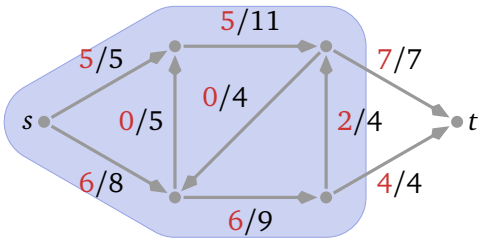
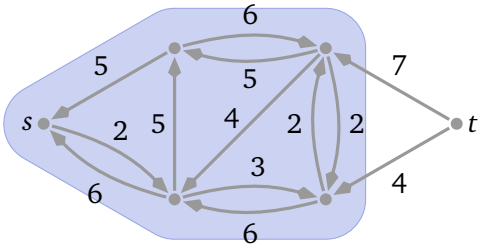


Figure 3: An st-flow in a network G and an st-cut S consisting of the vertices in the shaded region. This cut has capacity $U_S = 16$. The flow across this cut is $F_S = 9$.



(a) A maximum flow f and a minimum st-cut S (shaded) that satisfy $F_s = U_S = 11$.



(b) The residual network G^f. S is the set of vertices reachable from s in G^f.

Figure 4: Illustration of the proof of Thm. 6

Proof. The lemma follows directly from flow conservation:

$$\begin{aligned} F_s &= \sum_{y \in V} \left(f_{s,y} - f_{y,s}\right) \\ &= \sum_{y \in V} \left(f_{s,y} - f_{y,s}\right) + \sum_{x \in S \setminus \{s\}} \sum_{y \in V} \left(f_{x,y} - f_{y,x}\right) \end{aligned}$$

because $\sum_{y \in V} \left(f_{x,y} - f_{y,x}\right) = 0$ for all $x \in S \setminus \{s\}$ due to flow conservation (x is neither s, because $x \in S \setminus \{s\}$, nor t, because $t \notin S$). Thus,

$$\begin{aligned} F_s &= \sum_{x \in S} \sum_{y \in V} \left(f_{x,y} - f_{y,x}\right) \\ &= \sum_{x \in S} \sum_{y \in S} \left(f_{x,y} - f_{y,x}\right) + \sum_{x \in S} \sum_{y \notin S} \left(f_{x,y} - f_{y,x}\right). \end{aligned}$$

The first sum in this last equation is 0 because the flow $f_{x,y}$ across every edge (x, y) with $x, y \in S$ occurs twice in this sum, once with positive sign and once with negative sign. The second sum is F_S. Thus, $F_s = F_S$. □

The following theorem now characterizes maximum flows:

Theorem 6 (Max-Flow Min-Cut Theorem)**.** *Let f be a feasible st-flow in G and let S be an st-cut of G. Then f is a maximum st-flow and S is a minimum st-cut if and only if $F_s = U_S$.*

Proof. Let f^* be a maximum st-flow and let S^* be a minimum st-cut. By Lem. 5, we have

$$F_s \leq F_s^* = F_{S^*}^* = \sum_{x \in S^*, y \notin S^*} \left(f_{x,y}^* - f_{y,x}^*\right) \leq \sum_{x \in S^*, y \notin S^*} f_{x,y}^* \leq \sum_{u \in S^*, v \notin S^*} u_{x,y} = U_{S^*} \leq U_S.$$

Thus, if $F_s = U_S$, then $F_s = F_s^* = U_{S^*} = U_S$, that is, f is a maximum st-flow ($F_s = F_s^*$) and S is a minimum st-cut ($U_{S^*} = U_S$).

Conversely, assume that f is a maximum st-flow and S is a minimum st-cut. By Cor. 4, there exists no augmenting path in G^f. Thus, if we choose S' to be the set of all vertices reachable from s in G^f, then $s \in S'$ and $t \notin S'$, that is, S' is an st-cut; see Fig. 4. Since S is a minimum st-cut, we have $F_s = F_S \leq U_S \leq U_{S'}$, where $F_s = F_S$ holds by Lem. 5. We prove that $F_s = U_{S'}$. Thus, $F_s = U_S = U_{S'}$.

For every edge $(x,y) \in E$ with $x \in S'$ and $y \notin S'$, $f_{x,y} = u_{x,y}$ because otherwise, the edge (x,y) would be an edge of G^f and thus, since $x \in S'$, y would also be in S'. Similarly, if $x \notin S'$ and $y \in S'$, then $f_{x,y} = 0$ because otherwise, the edge (y,x) would belong to G^f and thus $x \in S'$. Thus,

$$F_s = F_{S'} = \sum_{x \in S', y \notin S'} \big(f_{x,y} - f_{y,x}\big) = \sum_{x \in S', y \notin S'} u_{x,y} = U_{S'}. \qquad \square$$

6 The Ford-Fulkerson Algorithm Computes a Maximum Flow

It remains to prove that

Theorem 7. *If the Ford-Fulkerson algorithm terminates, then it returns a maximum flow.*

Proof. By Lem. 3, each iteration of the algorithm maintains that f is a feasible flow. Thus, the flow returned by Ford-Fulkerson is a feasible flow.

Once the algorithm terminates, there is no more augmenting path in G^f. As shown in the proof of Thm. 6, this implies that there exists an st-cut in G whose capacity equals F_s and, hence, f is a maximum st-flow, by Thm. 6. $\square$

7 The Max-Flow Min-Cut Theorem via LP Duality*

The Max-Flow Min-Cut Theorem can be viewed as a duality result: Cuts are duals of flows in the sense that an st-flow can never exceed the capacity of an st-cut, and Thm. 6 even shows that the maximum flow equals the capacity of a minimum cut. We will see more such duality results throughout this course. Not all duality results are applications of LP duality, but all duality results in this course are. In this section, we discuss an alternative proof of the Max-Flow Min-Cut Theorem via LP duality. To this end, we need LP formulations of the maximum flow and minimum cut problems and demonstrate that they are each other's dual. The Max-Flow Min-Cut Theorem then follows immediately from strong LP duality.

We already formulated the maximum flow problem as an LP. However, we will need an ever so slightly different LP formulation of it. Yes, many optimization problems have more than one LP formulation, some more natural than others. The LP formulation of the maximum flow problem we need here is still rather natural. To arrive at it, it is best to start with an *integer* LP formulation of the minimum cut problem. Every subset $S \subseteq V$ can be characterized by associating a value $\hat{x}_v \in \{0,1\}$ with every vertex $v \in V$ and defining $S = \{v \in V \mid \hat{x}_v = 1\}$. To ensure that S is an st-cut, we need $s \in S$ and $t \notin S$, that is, we need to impose the constraints that $\hat{x}_s = 1$ and $\hat{x}_t = 0$. Given such a cut S, the edges from S to T are exactly those edges (u,v) with $\hat{x}_u = 1$ and $\hat{x}_v = 0$. Those are the edges that contribute to the capacity of the cut. All other edges do not contribute to the capacity of the cut, not even negatively. This gives the following ILP formulation of the minimum cut problem:

$$\begin{aligned} \text{Minimize} \sum_{v,w \in V} u_{v,w} z_{v,w} & \\ \text{s.t. } z_{v,w} - x_v + x_w \geq 0 & \quad \forall v, w \in V \\ x_s = 1 & \\ x_t = 0 & \\ x_v \in \{0,1\} & \quad \forall v \in V \setminus \{s,t\} \\ z_{v,w} \geq 0 & \quad \forall v, w \in V \end{aligned} \tag{2}$$

The constraints on the variables x_v, for all $v \in V$, ensure that for every feasible solution $(\hat{x}, \hat{z})$ of this ILP, the set $S = \{v \in V \mid \hat{x}_v = 1\}$ is an *st*-cut. The non-negativity constraints on the variables $z_{v,w}$ ensure that no edge, not even the ones from $V \setminus S$ to S, makes a negative contribution to the objective function. The constraint $z_{v,w} - x_v + x_w \geq 0$ ensures that $z_{v,w} = 1$—that is, that the capacity of the edge (v,w) is added to the capacity of the cut—whenever $v \in S$ ($x_v = 1$) and $w \notin S$ ($x_w = 0$). (Technically, the constraint ensures only that $z_{v,w} \geq 1$, but an optimal solution minimizes the objective function value, which it achieves by choosing the smallest value for each variable $z_{v,w}$ it can get away with. Thus, it ends up setting $z_{v,w} = 1$.)

By the following lemma, we can drop the constraints that $x_v \leq 1$, for all $v \in V \setminus \{s,t\}$, which gives the following as the ILP formulation of the minimum cut problem we will use:

$$\begin{aligned} \text{Minimize} \sum_{v,w \in V} u_{v,w} z_{v,w} & \\ \text{s.t. } z_{v,w} - x_v + x_w \geq 0 & \quad \forall v, w \in V \\ x_s = 1 & \\ x_t = 0 & \\ x_v \geq 0 & \quad \forall v \in V \setminus \{s,t\} \\ x_v \in \mathbb{Z} & \quad \forall v \in V \setminus \{s,t\} \\ z_{v,w} \geq 0 & \quad \forall v, w \in V \end{aligned} \tag{3}$$

Lemma 8. *For every feasible solution $(\hat{x}, \hat{z})$ of* (4)*, there exists a feasible solution $(\tilde{x}, \tilde{z})$ of* **??** *with the same objective function value.*

Proof. We define $(\tilde{x}, \tilde{z})$ by setting $\tilde{z} = \hat{z}$ and $\tilde{x}_v = \min(\hat{x}_v, 1)$, for all $v \in V$. Since $\tilde{z} = \hat{z}$, both solutions clearly have the same objective function value. We have to prove that $\tilde{x}$ is integral and that $(\tilde{x}, \tilde{y})$ is feasible.

For all $v \in V \setminus \{s,t\}$, $\hat{x}_v$ is an integer, so $\tilde{x}_v = \min(\hat{x}_v, 1)$ is also an integer.

The non-negativity constraints hold because $\tilde{z} = \hat{z} \geq 0$ and, for all $v \in V \setminus \{s,t\}$, $\tilde{x}_v = \min(\hat{x}_v, 1) \geq \min(0,1) = 0$.

Finally consider any of the constraints $z_{v,w} - x_v + x_w \geq 0$. If $\tilde{x}_v - \tilde{x}_w \leq 0$, then $\tilde{z}_{v,w} - \tilde{x}_v + \tilde{x}_w \geq 0$ beccause $\tilde{z}_{v,w} = \hat{z}_{v,w} \geq 0$. If $\tilde{x}_v - \tilde{x}_w > 0$, then $\tilde{x}_v = 1$ and $\tilde{x}_w = 0$. Thus, $\hat{x}_v \geq 1$ and $\hat{x}_w = 0$, that is, $\hat{x}_v - \hat{x}_w \geq \tilde{x}_v - \tilde{x}_w$. Therefore, $\tilde{z}_{v,w} - \tilde{x}_v + \tilde{w} \geq \hat{z}_{v,w} - \hat{x}_v + \hat{x}_w \geq 0$. Since this holds for all $v, w \in V$, $(\tilde{x}, \tilde{z})$ is feasible. □

The LP relaxation of (3) is

$$\begin{aligned} \text{Minimize } & \sum_{v,w \in V} u_{v,w} z_{v,w} \\ \text{s.t. } & z_{v,w} - x_v + x_w \geq 0 \quad \forall v, w \in V \\ & x_s = 1 \\ & x_t = 0 \\ & x_v \geq 0 \quad \forall v \in V \setminus \{s, t\} \\ & z_{v,w} \geq 0 \quad \forall v, w \in V \end{aligned} \tag{4}$$

The dual of (4) associates a variable $f_{v,w}$ with every constraint in the first group, a variable F_s with the constraint $x_s = 1$, and a variable F_t with the constraint $x_t = 0$. The remaining constraints are non-negativity constraints and do not have any associated variables in the dual. The naming of the variables was chosen on purpose because $f_{v,w}$ ends up being exactly the flow along the edge (v, w) in a feasible solution of the dual, F_s ends up being the net out-flow of s, and F_t ends up being the net out-flow of t. Here is the dual of (4):

$$\begin{aligned} \text{Maximize } & F_s \\ \text{s.t. } & f_{v,w} \leq u_{v,w} \quad \forall v, w \in V \\ & \sum_{w \in V} (f_{w,s} - f_{s,w}) + F_s = 0 \\ & \sum_{w \in V} (f_{w,t} - f_{t,w}) + F_t = 0 \\ & \sum_{w \in V} (f_{w,v} - f_{v,w}) \leq 0 \qquad \forall v \in V \setminus \{s, t\} \\ & f_{v,w} \geq 0 \qquad \forall v, w \in V \end{aligned}$$

It is helpful to rearrange the constraints slightly:

$$\begin{aligned} \text{Maximize } & F_s \\ \text{s.t. } & f_{v,w} \leq u_{v,w} \quad \forall v, w \in V \\ & \sum_{w \in V} (f_{s,w} - f_{w,s}) = F_s \\ & \sum_{w \in V} (f_{t,w} - f_{w,t}) = F_t \\ & \sum_{w \in V} (f_{v,w} - f_{w,v}) \geq 0 \qquad \forall v \in V \setminus \{s, t\} \\ & f_{v,w} \geq 0 \qquad \forall v, w \in V \end{aligned} \tag{5}$$

Note that this is almost the LP formulation of the maximum flow problem we developed earlier:

- The constraints $f_{v,w} \geq 0$ and $f_{v,w} \leq u_{v,w}$ together just express the capacity constraints of all edges of G.
- The equality constraints F_s and F_t simply enforce that F_s and F_t are the net out-flows of s and t. Therefore, the objective just states that we want to maximize the total out-flow of s, which is

exactly the aim of the maximum flow problem.

- If the constraints $\sum_{w\in V}(f_{v,w}-f_{w,v})\geq 0$ were equality constraints, they would simply express that a feasible flow must satisfy flow conservation. The weaker constraints in (5) require only that the total out-flow of every vertex is no less than its total in-flow, that is, that no vertex other than t absorbs any flow. It is not hard to prove that for an optimal solution of (5), these constraints are tight, that is, an optimal solution does indeed satisfy flow conservation. We won't do this here because we will prove the Max-Flow Min-Cut Theorem in a more direct manner. We will need the following observation though:

OBSERVATION 9. *For every feasible flow $\hat{f}$, the solution $(\hat{f},\hat{F}_s,\hat{F}_t)$ is a feasible solution of* (5) *with objective function value $\hat{F}_s$.*

To prove the Max-Flow Min-Cut Theorem now, let us implicitly extend any feasible flow $\hat{f}$ to a feasible solution $(\hat{f},\hat{F}_s,\hat{F}_t)$ of (5). This allows us to consider every feasible flow $\hat{f}$ to be a feasible solution of (5). Similarly, every st-cut $\hat{S}$ will be treated to be interchangeable with its corresponding feasible solution $(\hat{x},\hat{z})$ of (3) defined as

$$\begin{aligned}\hat{x}_v &= \left\{\begin{matrix}1 & \text{if } v\in S\\ 0 & \text{otherwise}\end{matrix}\right\} \quad \forall v\in V,\\ \hat{z}_{v,w} &= \max(0,x_v-x_w) \quad \forall v,w\in V.\end{aligned} \tag{6}$$

Thus, we can treat every st-cut as a feasible solution of (3).

With this convention, the Max-Flow Min-Cut Theorem states that $\hat{f}$ is a maximum flow, and $\hat{S}$ is a minimum cut if and only if

$$\hat{F}_s=\sum_{v,w\in V}u_{v,w}\hat{z}_{v,w}. \tag{7}$$

The "if" direction is easy to prove: By Obs. 9, $\hat{f}$ is a feasible solution of (5) with objective function value $\hat{F}_s$. Since (4) is the LP relaxation of (3), $(\hat{x},\hat{z})$ is a feasible solution of (4). Thus, by weak LP duality, there cannot be any maximum flow $\tilde{f}$ with $\tilde{F}_s>\hat{F}_s$ nor any st-cut $\tilde{S}$ with with $\sum_{v,w\in V}u_{v,w}\tilde{z}_{v,w}<\sum_{v,w\in V}u_{v,w}\hat{z}_{v,w}$. Thus, $\hat{f}$ is a maximum flow and $\hat{S}$ is a minimum cut.

For the "only-if" direction, assume that $\hat{f}$ is a maximum flow, and $\hat{S}$ is a minimum st-cut. Then we construct from $\hat{f}$ another st-cut $\tilde{S}$. such that $\hat{f}$ and $\tilde{S}$ satisfy the complementary slackness conditions of (4) and (5). This implies that $\hat{f}$ and $\tilde{S}$ are optimal solutions of (5) and (4), respectively. By strong LP duality, this implies that they satisfy (7). However, since $\hat{S}$ is a minimum st-cut, we have

$$\sum_{v,w\in V}u_{v,w}\hat{z}_{v,w}\leq\sum_{v,w}y_{v,w}\tilde{z}_{v,w}$$

and, by weak LP duality,

$$\hat{F}_s\leq\sum_{v,w\in V}u_{v,w}\hat{z}_{v,w}.$$

Since

$$\hat{F}_s=\sum_{v,w\in V}u_{v,w}\tilde{z}_{v,w},$$

this implies that

$$\hat{F}_s=\sum_{v,w\in V}u_{v,w}\hat{z}_{v,w},$$

that is, $\hat{f}$ and $\hat{S}$ satisfy (7).

It remains to show how to construct the *st*-cut $\tilde{S}$ from a maximum flow $\hat{f}$ and to prove that $\hat{f}$ and $\tilde{S}$ satisfy the complementary slackness conditions of (4) and (5).

Proposition 10. *Let $\hat{f}$ be a maximum flow, let $(\hat{f}, \hat{F}_s, \hat{F}_t)$ be the corresponding solution of* (5) *defined in Obs. 9, let $\tilde{S}$ be the set of vertices reachable from s in $G^{\hat{f}}$, and let $(\tilde{x}, \tilde{z})$ be the corresponding feasible solution of* (3) *defined according to* (6)*. Then $(\hat{f}, \hat{F}_s, \hat{F}_t)$ and $(\tilde{x}, \tilde{z})$ satisfy the complementary slackness conditions of* (4) *and* (5)*.*

Proof. First note that $\tilde{S}$ is an *st*-cut: Clearly, $s \in \tilde{S}$. Since $\hat{f}$ is a maximum flow, Cor. 4 states that there is no path from s to t in $G^{\hat{f}}$, so $t \notin \tilde{S}$.

It remains to verify the complementary slackness conditions. First, we verify primal complementary slackness, taking (5) to be the primal LP, and (4) to be the dual LP.

The dual constraint corresponding to each variable $f_{v,w}$ is $z_{v,w} - x_v + x_w \geq 0$. This constraint is tight for $(\tilde{x}, \tilde{z})$ unless $\tilde{x}_v = 0$, $\tilde{x}_w = 1$, and $\tilde{z}_{v,w} = 0$. Therefore, if this constraint is not tight, then $v \notin \tilde{S}$ and $w \in \tilde{S}$. This implies that $(w, v) \notin G^{\hat{f}}$ and, therefore, that $\hat{f}_{v,w} = 0$.

The dual complementary slackness conditions are equally easy to verify: Since $\hat{f}$ satisfies flow conservation, we have

$$\sum_{w \in V} \left(\hat{f}_{v,w} - \hat{f}_{w,v} \right) = 0,$$

for all $v \in V \setminus \{s, t\}$, that is, the dual complementary slackness condition corresponding to every variable x_v, $v \in V \setminus \{s, t\}$, is satisfied.

The primal constraint corresponding to each variable $z_{v,w}$ is $f_{v,w} \leq u_{v,w}$. If $\tilde{z}_{v,w} > 0$, then we have $\tilde{x}_v = 1$ and $\tilde{x}_w = 0$, so $v \in \tilde{S}$ and $w \notin \tilde{S}$. Therefore, $(v, w) \notin G^{\hat{f}}$ and, thus, $\hat{f}_{v,w} = u_{v,w}$. □