Maximum Flows

CSCI 4113/6101
Instructor: Norbert Zeh
September 19, 2025

Consider a network of computers. Each network link has a certain bandwidth; some are Gigabit Ethernet links, some may be good old-fashioned 10M links, some may be wireless links. The goal is to design a cooperative data transfer protocol that can send large files between locations as quickly as possible. Clearly, it would be fastest to use a route consisting of only Gigabit Ethernet links when sending lots of data from point A to point B. If there are two edge-disjoint routes consisting of only Gigabit Ethernet links, then the transmission time can be halved by sending half of the file along one route and the other half along the other route. Even though the 10M links by themselves are slow, adding a route of 10M links to the mix nevertheless reduces the load on the two Gigabit routes and reduces the transmission time a little more. Now, everybody likes the fast Gigabit Ethernet links and tries to utilize them to send their data, so it may be the case that only a fraction of the full bandwidth of these links is available for use by the file transfer protocol. Abstractly, the network can be described as a directed graph $G = (V, E)$ whose edges have non-negative **capacities** (bandwidths) u_e. The goal is to determine how much **flow** (data) can be sent from a **source** $s \in V$ to a **sink** (destination) $t \in V$ and how much of it is sent along each edge $e \in E$; the flow sent along edge e is denoted by f_e. This flow has to satisfy two natural constraints (see Fig. 1):

Capacity constraints: The flow sent along an edge e cannot be negative and cannot exceed the edge's capacity:

$$0 \leq f_e \leq u_e \quad \forall e \in E.$$

Flow conservation: No node other than the source or sink generates or absorbs any flow, that is, the amount of flow entering this node must equal the amount of flow leaving this node:

$$\sum_{(x,y)\in E} f_{x,y} = \sum_{(y,z)\in E} f_{y,z} \quad \forall y \in V \setminus \{s, t\}.$$

A flow function f that satisfies these two conditions is called a **feasible *st*-flow**. The total amount of

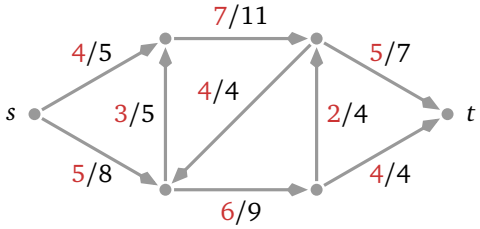

Figure 1: A network G and a feasible st-flow f. Edge capacities are shown in black. The flow across every edge is shown in red.

flow sent from s to t is the net out-flow of s, $\sum_{(s,z)\in E} f_{s,z} - \sum_{(x,s)\in E} f_{x,s}$, which must equal the net in-flow of t, $\sum_{(x,t)\in E} f_{x,t} - \sum_{(t,z)\in E} f_{t,z}$, as a result of flow conservation. It will be convenient to have a notation for the net flow produced by each node:

$$F_y = \sum_{(y,z)\in E} f_{y,z} - \sum_{(x,y)\in E} f_{x,y}. \tag{1}$$

Then flow conservation states that

$$\begin{aligned} F_s &= -F_t \quad \text{and} \\ F_x &= 0 \qquad \forall x \notin \{s,t\}. \end{aligned}$$

A feasible st-flow f is a **maximum *st*-flow** if $F_s \geq F'_s$ for every feasible st-flow f'. This gives the following graph problem, expressed in the language of linear programming introduced in the first part of this book:

Problem 1 (Maximum flow)**.** Given a directed graph $G = (V, E)$, an assignment of **capacities** $u : E \to \mathbb{R}^+$ to its edges, and an ordered pair of vertices (s, t), compute a maximum st-flow f, that is, an assignment $f : E \to \mathbb{R}^+$ of flows to the edges of G such that F_s is maximized and

$$\begin{aligned} 0 \leq f_e &\leq u_e \quad \forall e \in E \\ F_x &= 0 \quad \forall x \in V \setminus \{s,t\}. \end{aligned} \tag{2}$$

Note that we use the term "flow" to refer to the whole function f and to refer to the "flow along an edge e", the value f_e. The meaning will be clear from context.

Given that the problem definition in Prob. 1 is a linear program, it can be solved using the Simplex Algorithm. This approach, but using a more efficient implementation that operates directly on the graph instead of explicitly translating the graph into a set of linear constraints, is known as the **Network Simplex** algorithm and is one of the fastest ways to compute maximum flows in practice. Our focus in the next few topics is on algorithms that solve the maximum flow problem directly, without using linear programming.

Throughout the discussion of maximum flow algorithms, we will refer to an absent edge as an edge of capacity 0; such an edge clearly carries no flow. This allows us to simplify notation, for example, by defining

$$F_x = \sum_{y\in V} \big(f_{x,y} - f_{y,x}\big),$$

which is equivalent to the definition (1) above.