

## Lecture 4: Max Flow/Min Cut

In this lecture, we will examine the Ford-Fulkerson (FF) algorithm for computing maxflow within a network. We will also present related theorems and proofs, most importantly, the Maxflow/Mincut Theorem. We then analyze the computational complexity of FF and consider an improved algorithm due to Edmonds and Karp.

### 1 Ford-Fulkerson

In this section we discuss the Ford-Fulkerson method and analyze its computational complexity.

#### 1.1 The Ford-Fulkerson Method

The Ford-Fulkerson Method is as follows:

- Initialize  $f$  to 0
- while there is an augmenting path  $p$  in  $G_f$ 
  - augment  $f$  along  $p$  in  $G$
  - end
- Return  $f$

The first few iterations of the Ford-Fulkerson algorithm are shown in figure 1. In the figure, (a) shows the initial network with flow 12, having chosen the augmenting path along the top of the graph, and (b) is the residual network induced by that flow. Within (b), the next augmenting path is shown

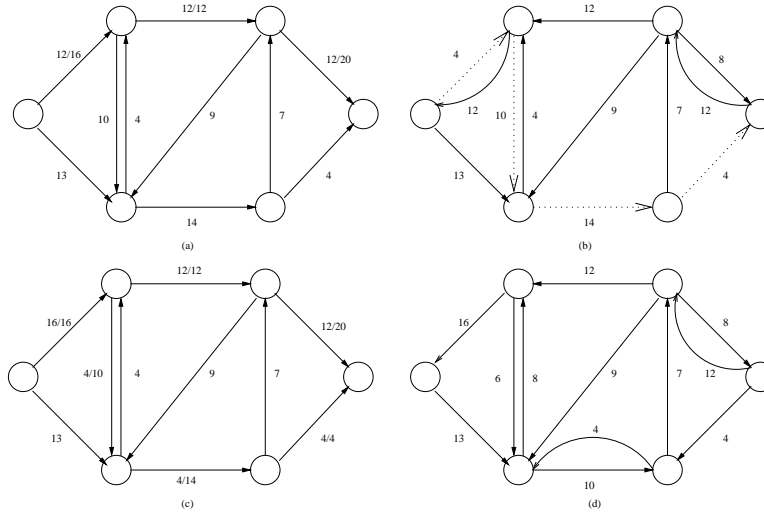


Figure 1: Flows in  $G$  and its residual network.

in dotted lines, and (c) shows the original network augmented with this path. Finally, (d) shows the residual network corresponding to (c), from which a subsequent augmenting path may be selected. The algorithm terminates when there are no more augmenting paths.

**Theorem 1.1.1 (Maxflow/Mincut Theorem)** *Given a network,  $G$ , and a flow,  $f$ , the following are equivalent:*

1.  $f$  is a maxflow in  $G$ ;
2. The residual network,  $G_f$ , contains no augmenting paths;
3.  $|f| = c(S, T)$  for some cut  $(S, T)$  of  $G$ .

**Proof (by cycle of implication).**

(1)  $\Rightarrow$  (2) Assume  $G_f$  contains an augmenting path. Then we can improve  $f$ , so  $f$  is not a maxflow, which contradicts (1).

(2)  $\Rightarrow$  (3) Assume  $G_f$  has no augmenting path. Let  $S = \{v \in V \mid v \text{ is reachable from } s \text{ in } G_f\}$  and  $T = V \setminus S$ . Obviously,  $s \in S$ ; and  $t \notin S$ , since  $t$

is not reachable in  $G_f$  ( $G_f$  has no augmenting path, by assumption). Hence,  $(S, T)$  is some cut of  $G$ .

**Claim 1.1.2**  $\forall u \in S, v \in T, f(u, v) = c(u, v)$ .

**Proof.** Assume  $f(u, v) < c(u, v)$ . It follows that  $(u, v) \in E_f \Rightarrow v \in S$ , which contradicts that  $(S, T)$  is a cut of  $G$ .

Now by earlier lemma<sup>1</sup>,  $|f| = f(S, T) = \sum_{u \in S} \sum_{v \in T} f(u, v)$ . Together with the above claim, we may now write:

$$|f| = f(S, T) = \sum_{u \in S} \sum_{v \in T} f(u, v) = \sum_{u \in S} \sum_{v \in T} c(u, v) = c(S, T).$$

(3)  $\Rightarrow$  (1) Assume  $|f| = c(S, T)$ . Then  $|f| \leq |f^*| \leq c(S, T) \Rightarrow |f| = |f^*| \Rightarrow f = f^*$ .

## 1.2 Computational Complexity

Initialization of  $f$  and the construction of  $G_f$  are both  $\mathcal{O}(|E|)$ , as is the process of finding an augmenting path. The **while** loop is executed at most  $|f^*|$  times, since the augmenting path increases flow by at least one unit each iteration. It is therefore easily seen that worst case bounds are  $\mathcal{O}(|f^*||E|)$ .

Now, consider the flow in figure 2. In (a), a flow of 1 unit follows the path of dotted lines, and (b) shows the resulting residual network. Now, suppose the next augmenting path is that highlighted in (b); once again, an increase in flow of 1 unit will result, and this alternating pattern can continue  $|f^*|$  times, exhibiting the worst case performance. For improved performance, we next sketch the Edmonds-Karp algorithm.

## 2 Edmonds-Karp

The EK algorithm employs a breadth-first search in choosing the shortest augmenting path from  $s$  to  $t$ , where *shortest augmenting path* refers to the

---

<sup>1</sup>See scribe notes dated February 2, 2004.

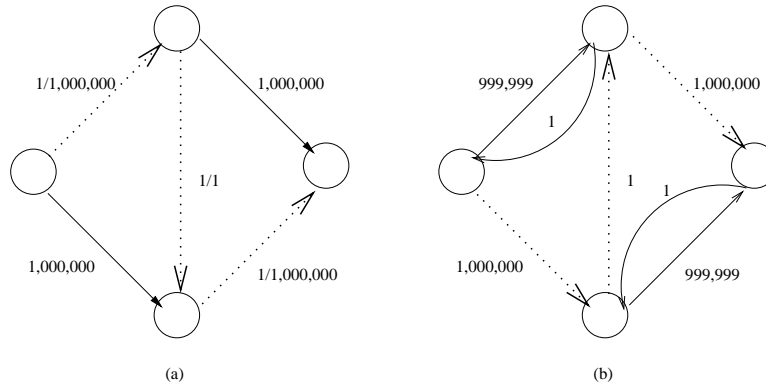


Figure 2: Worst Case Flow in Ford-Fulkerson.

augmenting path with the *fewest number of edges*. Referring to figure 2, choosing paths with the fewest number of edges (namely the paths of two edges of 1,000,000 along the top and bottom of the graph) would have readily achieved the 2,000,000 capacity maximum flow by avoiding the extra edge of capacity 1 (the bottleneck). Moreover, maximum flow would then have been realized in 2 iterations of the **while** loop instead of 2,000,000 by Ford-Fulkerson! Note that a breadth-first search requires  $\mathcal{O}(|V| + |E|) = \mathcal{O}(|E|)$ . This leads to the following worst-case bounds for the EK algorithm:

**Claim 2.0.1** *Edmonds-Karp is  $\mathcal{O}(|V||E|^2)$ .*

## References

- [Cormen2001] Cormen, T., Leiserson, C., Rivest, R. and C. Stein. “Introduction to Algorithms,” Second Edition, MIT Press, Cambridge, MA, 2001, pp. 651-663.