

## Min Cut Algorithm

Before we design approximation algorithms for cut problems, let us review a simple case that has a polynomial-time algorithm.

**Min Cut:** Input a directed graph  $G = (V, E)$ , two vertices  $s$  and  $t$  (with no edges coming into  $s$  nor coming out of  $t$ .) Output a partition  $V = L \cup R$ . Objective: minimize the number of edges that go from  $L$  to  $R$ .

**Max Flow:** Input a directed graph  $G = (V, E)$ , two vertices  $s$  and  $t$  (with no edges coming into  $s$  nor coming out of  $t$ .) Output a flow  $f_e \in \{0, 1\}$  for each edges  $e$ , such that for every vertex  $v \neq s, t$ , we have flow conservation:  $\sum_e \text{into } v f_e = \sum_e \text{out of } v f_e$ . Objective: maximum the flow out of  $s$ ,  $\sum_e \text{out of } s f_e$ .

**Lemma 1** *Let  $(L, R)$  be a cut. Then the value of  $f$  equals the net amount of flow from  $L$  to  $R$ .*

Proof sketch: Induction on  $|L|$ .

**Lemma 2** *Let  $(L, R)$  be a cut. Then the value of  $f$  is less than or equal to the cost of cut  $(L, R)$ .*

MIN CUT( $G, s, t$ )

```

1  { ← 0
2   $G_f = G$ 
3  while  $G_f$  has a path  $p$  from  $s$  to  $t$ 
4      do
5          Update  $f$ :
6               $f'_{uv} = 1$  if  $(u, v) \in p, f_{uv} = 0$ 
7               $f'_{vu} = 0$  if  $(u, v) \in p, f_{vu} = 1$ 
8               $f'_{uv} = f_{uv}$  otherwise
9               $f' \leftarrow f$ 
10         Update  $G_f = (V, E_f)$ :
11              $(u, v) \in E_f$  if  $(u, v) \in E, f_{uv} = 0$  or
12                  $(v, u) \in E, f_{vu} = 1$ .
13 return  $(L, V \setminus L)$  with  $L = \{\text{vertices reachable from } s \text{ in } G_f\}$ 

```

Correctness: by stopping condition, the output is a valid cut. By induction over time,  $f$  is always a valid flow. It is easy to check that the value of the last flow  $f$  equals the cost of the output cut, hence both are optimal and we have found a min cut.

Termination: By induction over time,  $f$  is always a valid flow. By inspection, the value of  $f$  increases by 1 at every iteration, so the number of iterations is finite.

Running time: Define  $p$  as the shortest path from  $s$  to  $t$  in  $G_f$  (smallest number of edges.) Then  $\sum_v \text{dist}_{G_f}(s, v)$  increases at each iterations, hence a polynomial bound on the number of iterations.

Extension: extends to graphs with non-negative edge-weights.