Max-Flow Min-Cut Theorem

1 Maximum Cardinality Bipartite Matching

An easily-understood application of max-flow is solving a classic bipartite matching problem.

MAXIMUM CARDINALITY BIPARTITE MATCHING (MCBM): Given a bipartite graph (X, Y, E) where $E \subseteq X \times Y$, find a subset $S \subseteq E$ of edges of maximum size such that no node is incident on more than one edge (i.e., it is a matching).

Theorem 1. *There exists a polynomial time algorithm to solve MCBM.*

Proof. We can reduce MCBM to max-flow, and since we assert there is a polynomial time algorithm to solve max-flow, we can use that algorithm to solve MCBM. We add a source node s connected to all nodes $x \in X$ and a sink node t connected to all nodes $y \in Y$. Let all these generated edges have a capacity of one, and all old edges a capacity of ∞. We also direct all the old edges from $X \to Y$.

This reduction runs in polynomial time. Then, we can find an integer maximum (s, t) flow and read off the optimal matching S by $S := \{e \in E : f_e = 1\}$. □

Lemma 2. *S is a maximum bipartite matching.*

Proof. A two-part proof; (1) S is a matching, (2) S is maximum.

(1). S is a matching.

The incoming flow into every node $x \in X$ is $\leq c_x = 1$, so the outgoing flow is ≤ 1 by conservation. The flow on all edges $e \in S$ is one, so each x participates in at most one edge; similarly for all $y \in Y$.

(2). S is maximum.

Let S^* be the maximum matching. We define the flow f by setting, for each $e = (u, v) \in S^*$, $f_e = 1, f_{(s,u)} = 1, f(v, t) = 1$. This flow obeys conservation: because S^* is a matching, we set $f_{(u,v)} = 1$ for at most one v for any fixed u, and for at most one u for any fixed v. $v(f) = |S^*|$, so the max-flow f^* has $v(f^*) \geq v(f) = |S^*|$. The matching S we found has $v(f) \geq |S^*|$, so it must be maximum. □

2 Max-Flow Min-Cut Theorem

Theorem 3. *The cost of the minimum (s, t) cut with respect to c_e is equal to the maximum (s, t) flow with capacities c_e.*

2.1 Notation

$f^{in}(v) = \sum_{e \text{ into } v} f_e$; $f^{out}(v) = \sum_{e \text{ out of } v} f_e$

$c(S, \bar{S}) = \sum_{e \in (S \times \bar{S}) \cap E} c_e$; $f(S, \bar{S}) = \sum_{e \in (S, \bar{S})} f_e$

$f^{out}(S) = f(S, \bar{S})$; $f^{in}(S) = f(\bar{S}, S)$

2.2 Lemmae

Lemma 4. *For any (s,t) cut $(S,\bar{S})$, $v(f) = f^{out}(S) - f^{in}(S)$.*

Proof. Show that the value of the flow in the graph is bottlenecked by the edges across the cut.

$v(f) = \sum_{e \text{ out of } s} f_e$

$= \sum_{v \in S} f^{out}(v) - f^{in}(v)$ with the inner part equal to 0 for all $v \notin S$ by conservation

$= \sum_{v \in S} (\sum_{e \text{ out of } v} f_e - \sum_{e \text{ into } v} f_e)$

$= \sum_{e \text{ out of } S} f_e - \sum_{e \text{ into } S} f_e$ – the f_e fully inside S cancel out, so we only care about edges crossing $(S,\bar{S})$

$= f^{out}(S) - f^{in}(S)$ □

Corollary 5. *If $(S,\bar{S})$ is any (s,t) cut, then $v(f) \leq c(S,\bar{S})$.*

Proof. By Lemma 4, $v(f) \leq f^{out}(S) = f(S,\bar{S}) \leq c(S,\bar{S})$. □

Corollary 6. *If $f^*$is a maximum (s,t) flow and $(S^*,\bar{S^*})$ a minimum (s,t) cut, then $v(f^*) \leq c(S^*,\bar{S^*})$.*

Corollary 7. *If f is any (s,t) flow and $(S,\bar{S})$ any (s,t) cut, and $v(f) = c(S,\bar{S})$, then f is a maximum (s,t) flow and $(S,\bar{S})$ is a minimum (s,t) cut.*

2.3 Residual Flow

Given a graph $\mathcal{G}$ and a flow f on $\mathcal{G}$, the residual flow graph $\mathcal{G}_f$ has edges $(u,v) \in E$ with $c'_e = c_e - f_e$ (forward edges) and (v,u) with $c'_{(v,u)} = f_{(u,v)}$ (backward edges). c'_e are called **residual capacities**; for forward edges, the residual capacity is how much more flow can fit on that edge, and for backward edges, the residual capacity is how much flow you can "undo" from the current flow. An **augmenting path** P is an (s,t) path such that each $e \in P$ has $c'_e > 0$.

2.4 Ford-Fulkerson Algorithm

Algorithm 1 Ford-Fulkerson Max-Flow Algorithm

```
1: f = 0
2: while G_f contains an augmenting path P do
3:    Select one such P
4:    E = min_{e∈P} c'_e
5:    for all e ∈ P do
6:       if e is a forward edge then
7:          f'_e = f_e + E
8:       else if e = (u,v) is a backward edge then
9:          f'_(v,u) = f_(v,u) − E
10:      end if
11:   end for
12: end while
```

2.5 Proof of Correctness

Lemma 8. *$f'$is a flow.*

Proof. A 3-part proof; (1) f' is not negative, (2) f' obeys the capacity constraint, (3) f' obeys the conservation constraint.

(1) f' is not negative.

We only need to prove this for backward edges $e = (u, v)$. For those, $c'_e = f_{(v,u)}$ and $\mathcal{E} \leq c'_e \leq f_{(v,u)}$, so it updates to $f'_{(v,u)} = f_{(v,u)} - \mathcal{E} \geq 0$.

(2) f' obeys the capacity constraint.

We only need to prove this for forward edges e. For those, $c'_e = c_e - f_e$, so $f'_e = f_e + \mathcal{E} \leq f_e + c'_e = c_e$.

(3) f' obeys the conservation constraint.

Consider a node u on an augmenting path P with in-edge e and out-edge e'.

(i). e, e' both forward, then inflow and outflow increased by $\mathcal{E}$, so conservation holds.

(ii). e, e' both backward, then inflow and outflow decreased by $\mathcal{E}$, so conservation holds.

(iii). e forward and e' backward, then inflow on e increased by $\mathcal{E}$ and outflow on e' decreased by $\mathcal{E}$ which cancels out, so conservation holds.

(iv). e backward and e' forward, then outflow on e decreased by $\mathcal{E}$ and inflow on e' increased by $\mathcal{E}$ which cancels out, so conservation holds. □

Lemma 9. *If all $c_e \in \mathbb{N}$, then in each iteration all $f_e \in \mathbb{N}$.*

Proof. By induction, and $\mathcal{E} = \min_{e \in P} c'_e \in \mathbb{N}$. □

Lemma 10. *The Ford-Fulkerson Algorithm terminates in at most C^* iterations where $C^* = \sum_e c_e$ or $C^* = \sum_{e \text{ out of } S} c_e$.*

Proof. $v(f)$ increases by at least one each iteration, and is bounded by $c(\{s\}, V \setminus \{s\})$. Each iteration takes $\mathcal{O}(m + n)$ via BFS to find P, so the runtime is $\mathcal{O}(C^*(m + n))$ which is pseudo-polynomial (next class, we will go over an improvement to the algorithm which will reduce it to polynomial time). □

Theorem 11. *The cost of the minimum (s, t) cut with respect to c_e is equal to the maximum (s, t) flow with capacities c_e.*

Proof. At termination, there is no more augmenting path in $\mathcal{G}_f$. Let $S = \{v | \mathcal{G}_f \text{ contains an } (s, v) \text{ path with } c'_e > 0\}$. By termination, we know $t \notin S$, so $(S, \bar{S})$ is an (s, t) cut. By Lemma 4, $v(f) = f^{out}(S) - f^{in}(S)$. Because all forward edges are saturated out of S, $f^{out}(S) = c(S, \bar{S})$. Because there are no backward edges out of S, $f^{in}(S) = 0$. Thus, $v(f) = c(S, \bar{S})$. By Corollary 7, f is a max-flow and $(S, \bar{S})$ is a min-cut. □