4. Lecture notes on flows and cuts

4.1 Maximum Flows

Network flows deals with modelling the flow of a commodity (water, electricity, packets, gas, cars, trains, money, or any abstract object) in a network. The links in the network are capacitated and the commodity does not vanish in the network except at specified locations where we can either inject or extract some amount of commodity. The main question is how much can be sent in this network.

Here is a more formal definition of the maximum flow problem. We have a digraph (directed graph) $G = (V, E)$ and two special vertices s and t; s is called the source and t the sink. We have an upper capacity function $u : E \to \mathbb{R}$ and also a lower capacity function $l : E \to \mathbb{R}$ (sometimes chosen to be 0 everywhere). A flow x will be an assignment of values to the arcs (directed edges) so that:

1. for every $e \in E$: $l(e) \leq x_e \leq u(e)$,
2. for every $v \in V \setminus \{s, t\}$:
$$\sum_{e \in \delta^+(u)} x_e - \sum_{e \in \delta^-(u)} x_e = 0. \tag{1}$$

The notation $\delta^+(u)$ represents the set of arcs *leaving* u, while $\delta^-(u)$ represents the set of arcs *entering* u.

Equations (1) are called *flow conservation* constraints. Given a flow x, its *flow value* $|x|$ is the net flow out of s:
$$|x| := \sum_{e \in \delta^+(s)} x_e - \sum_{e \in \delta^-(s)} x_e. \tag{2}$$

One important observation is that $|x|$ is also equal to the net flow into t, or minus the net flow out of t. Indeed, summing (1) over $u \in V \setminus \{s, t\}$ together with (2), we get:

$$\begin{aligned} |x| &= \sum_{v \in V \setminus \{t\}} \left(\sum_{e \in \delta^+(u)} x_e - \sum_{e \in \delta^-(u)} x_e \right) \\ &= \sum_{e \in \delta^-(t)} x_e - \sum_{e \in \delta^+(t)} x_e \end{aligned}$$

by looking at the contribution of every arc in the first summation.

The *maximum flow problem* is the problem of finding a flow x of maximum value $|x|$. This is a linear program:

$$\begin{aligned}
\text{Max} \quad & \sum_{e\in\delta^+(s)} x_e - \sum_{e\in\delta^-(s)} x_e \\
\text{subject to:} \quad & \\
& \sum_{e\in\delta^+(u)} x_e - \sum_{e\in\delta^-(u)} x_e = 0 && u \in V \setminus \{s,t\} \\
& l(e) \le x_e \le u(e) && e \in E.
\end{aligned}$$

We could therefore use algorithms for linear programming to find the maximum flow and duality to derive optimality properties, but we will show that more combinatorial algorithms can be developed and duality translates into statements about *cuts*.

In matrix form, the linear program can be written as:

$$\max\{c^T x : \quad \begin{array}{rcl} Nx & = & 0, \\ Ix & \le & u, \\ -Ix & \le & -l\} \end{array}$$

where N is the (vertex-arc incidence[1]) matrix with rows indexed by $u \in V \setminus \{s,t\}$ and columns indexed by arcs $e = (i,j) \in E$; the entry N_{ue} is:

$$N_{ue} = \begin{cases} 1 & u = i \\ -1 & u = j \\ 0 & u \notin \{i,j\}. \end{cases}$$

The constraints of the linear program are thus: $Ax \lesseqgtr b$ where

$$A = \begin{pmatrix} N \\ - - - \\ I \\ - - - \\ -I \end{pmatrix},$$

and some of the constraints are equalities and some are inequalities.

Lemma 4.1 *A is total unimodular.*

Proof: We could use Theorem 3.14 from the polyhedral chapter, but proving it directly is as easy. Consider any square submatrix of A, and we would like to compute its determinant up to its sign. If there is a row with a single $+1$ or a single -1 in it (in particular, a row coming from either the identity submatrix I or $-I$), we can expand the determinant and

[1] More precisely, part of it as we are not considering vertices s and t

compute the determinant (up to its sign) of a smaller submatrix of A. Repeating this, we now have a square submatrix of N. If there is a column with a single $+1$ or a single -1 then we can expand the determinant along this column and get a smaller submatrix. We are thus left either with an empty submatrix in which case the determinant of the original matrix was $+1$ or -1, or with a square submatrix of N with precisely one $+1$ and one -1 in *every* column. The rows of this submatrix are linearly dependent since their sum is the 0 vector. Thus the determinant is 0. This proves total unimodularity. △

As a corollary, this means that if the right-hand-side (i.e. the upper and lower capacities) are integer-valued then there always exists a maximum flow which takes only integer values.

Corollary 4.2 *If $l : E \to \mathbb{Z}$ and $u : E \to \mathbb{Z}$ then there exists a maximum flow x such that $x_e \in \mathbb{Z}$ for all $e \in E$.*

4.1.1 Special cases

Arc-disjoint paths. If $l(e) = 0$ for all $e \in E$ and $u(e) = 1$ for all $e \in E$, any integer flow x will only take values in $\{0, 1\}$. We claim that for an integer flow x, there exist $|x|$ arc-dsjoint (i.e. not having any arcs in common) paths from s to t. Indeed, such paths can be obtained by *flow decomposition*. As long as $|x| > 0$, take an arc out of s with $x_e = 1$. Now follow this arc and whenever we reach a vertex $u \neq t$, by flow conservation we know that there exists an arc leaving u that we haven't traversed yet (this is true even if we reach s again). This process stops when we reach t and we have therefore identified one path from s to t. Removing this path gives us a new flow x' (indeed flow conservation at vertices $\neq s, t$ is maintained) with $|x'| = |x| - 1$. Repeating this process gives us $|x|$ paths from s to t and, by construction, they are arc-disjoint. The paths we get might not be *simple*[2]; one can however make them simple by removing the part of the walk between repeated occurences of the same vertex. Summarizing, if $l(e) = 0$ for all $e \in E$ and $u(e) = 1$ for all $e \in E$, then from a maximum flow of value k, we can extract k arc-disjoint (simple) paths from s to t. Conversely, if the digraph contains k arc-disjoint paths from s to t, it is easy to construct a flow of value k. This means that the maximum flow value from s to t represents the maximum number of arc-disjoint paths between s and t.

Bipartite matchings. One can formulate the maximum matching problem in a bipartite graph as a maximum flow problem. Indeed, consider a bipartite graph $G = (V, E)$ with bipartition $V = A \cup B$. Consider now a directed graph D with vertex set $V \cup \{s, t\}$. In D, there is an arc from s to every vertex of A with $l(e) = 0$ and $u(e) = 1$. There is also an arc from every vertex in B to t with capacities $l(e) = 0$ and $u(e) = 1$. Every edge $(a, b) \in E$ is oriented from $a \in A$ to $b \in B$ and gets a lower capacity of 0 and an upper capacity equal to $+\infty$ (or just 1). One can easily see that from any matching of size k one can construct a flow of value k; similarly to any *integer valued* flow of value k corresponds a matching of size k. Since the capacities are in $\mathbb{Z}$, by Corollary 4.2, this means that a maximum flow in

[2]A simple path is one in which no vertex is repeated.

D has the same value as the maximum size of any matching in G. Observe that the upper capacities for the arcs between A and B do not matter, provided they are ≥ 1.

Orientations. Consider the problem of orienting the edges of an undirected graph $G = (V, E)$ so that the indegree of any vertex v in the resulting digraph is at most $k(v)$. This can be formulated as a maximum flow problem in which we have (i) a vertex for every vertex of G, (ii) a vertex for every edge of G and (iii) 2 additional vertices s and t. Details are left as an exercise.

Exercise 4-1. Suppose you are given an $m \times n$ matrix $A \in \mathbb{R}^{m \times n}$ with row sums $r_1, \cdots, r_m \in \mathbb{Z}$ and column sums $c_1, \cdots, c_n \in \mathbb{Z}$. Some of the entries might not be integral but the row sums and column sums are. Show that there exists a rounded matrix A' with the following properties:

- row sums and column sums of A and A' are identical,
- $a'_{ij} = \lceil a_{ij} \rceil$ or $a'_{ij} = \lfloor a_{ij} \rfloor$ (i.e. a'_{ij} is a_{ij} either rounded up or down.).

By the way, this rounding is useful to the census bureau as they do not want to publish statistics that would give too much information on specific individuals. They want to be able to modify the entries without modifying row and column sums.

4.2 Cuts

In this section, we derive an important duality result for the maximum flow problem, and as usual, this takes the form of a minmax relation.

In a digraph $G = (V, A)$, we define a *cutset* or more simply a *cut* as the set of arcs $\delta^+(S) = \{(u, v) \in A : u \in S, v \in V \setminus S\}$. Observe that our earlier notation $\delta^+(v)$ for $v \in V$ rather than $\delta^+(\{v\})$ is a slight abuse of notation. Similarly, we define $\delta^-(S)$ as $\delta^+(V \setminus S)$, i.e. the arcs entering the vertex set S. We will typically identify a cutset $\delta^+(S)$ with the corresponding vertex set S. We say that a cut $\delta^+(S)$ is *an $s - t$ cut* (where s and t are vertices) if $s \in S$ and $t \notin S$.

For an undirected graph $G = (V, E)$, $\delta^+(S)$ and $\delta^-(S)$ are identical and will be denoted by $\delta(S) = \{(u, v) \in E : |\{u, v\} \cap S| = 1\}$. Observe that now $\delta(S) = \delta(V \setminus S)$.

For a maximum flow instance on a digraph $G = (V, E)$ and upper and lower capacity functions u and l, we define the capacity $C(S)$ of the cut induced by S as

$$C(S) = \sum_{e \in \delta^+(S)} u(e) - \sum_{e \in \delta^-(S)} l(e) = u(\delta^+(S)) - l(\delta^-(S)).$$

By definition of a flow x, we have that

$$C(S) \geq \sum_{e \in \delta^+(S)} x_e - \sum_{e \in \delta^-(S)} x_e.$$

We have shown earlier that the net flow out of s is equal to the net flow into t. Similarly, we can show that for any S with $s \in S$ and $t \notin S$ (i.e. the cut induced is an $s-t$ cut), we have that the flow value $|x|$ equals:

$$|x| = \sum_{e \in \delta^+(S)} x_e - \sum_{e \in \delta^-(S)} x_e.$$

This is shown by summing (1) over $u \in S \setminus \{s\}$ together with (2). Thus, we get that for any S with $s \in S$ and $t \notin S$ and any $s-t$ flow x, we have:

$$|x| \leq C(S).$$

Therefore, maximizing over the $s-t$ flows and minimizing over the $s-t$ cuts, we get

$$\max |x| \leq \min_{S: s \in S, t \notin S} C(S).$$

This is weak duality, but in fact, one always has equality as stated in the following theorem. Of course, we need the assumption that the maximum flow problem is feasible. For example if there is an edge with $l(e) > u(e)$ then no flow exists (we will show later that a necessary and sufficient condition for the existence of a flow is that (i) $l(e) \leq u(e)$ for every $e \in E$ and (ii) for any $S \subset V$ with $|S \cap \{s,t\}| \neq 1$, we have $u(\delta^+(S)) \geq l(\delta^-(S))$).

Theorem 4.3 (max $s-t$ flow-min $s-t$ cut) *For any maximum flow problem for which a feasible flow exists, we have that that the maximum $s-t$ flow value is equal to the minimum capacity of any $s-t$ cut:*

$$\max_{\text{flow } x} |x| = \min_{S: s \in S, t \notin S} C(S).$$

One way to prove this theorem is by using strong duality of linear programming and show that from any optimum dual solution one can derive an $s-t$ cut of that capacity. Another way, and this is the way we pursue, is to develop an algorithm to find a maximum flow and show that when it terminates we have also a cut whose capacity is equal to the flow we have constructed, therefore proving optimality of the flow and equality in the minmax relation.

Here is an algorithm for finding a maximum flow. Let us assume that we are given a feasible flow x (if $u(e) \geq 0$ and $l(e) \leq 0$ for all e, we could start with $x = 0$). Given a flow x, we define a *residual graph* G_x on the same vertex set V. In G_x, we have an arc (i,j) if (i) $(i,j) \in E$ and $x_{ij} < u((i,j))$ or if (ii) $(j,i) \in E$ and $x_{ji} > l((j,i))$. In case (i), we say that (i,j) is a *forward* arc and in case (ii) it is a *backward* arc. If both (i) and (ii) happen, we introduce two arcs (i,j), one forward and one backward; to be precise, G_x is thus a multigraph. Consider now any directed path P from s to t in the residual graph; such a path is called an *augmenting path*. Let P^+ denote the forward arcs in P, and P^- the backward arcs. We can modify the flow x in the following way:

$$x'_e = \begin{cases} x_e + \epsilon & e \in P^+ \\ x_e - \epsilon & e \in P^- \\ x_e & e \notin P \end{cases}$$

This is known as pushing ϵ units of flow along P, or simply augmenting along P. Oberve that flow conservation at any vertex u still holds when pushing flow along a path. This is trivial if u is not on the path, and if u is on the path, the contributions of the two arcs indicent to u on P cancel each other. To make sure the resulting x' is feasible (satisfies the capacity constraints), we choose

$$\epsilon = \min\left(\min_{e\in P^+}(u(e) - x_e), \min_{e\in P^-} x_e - l(e)\right).$$

By construction of the residual graph we have that $\epsilon > 0$. Thus, pushing ϵ units of flow along an augmenting path provides a new flow x' whose value $|x'|$ satisfy $|x'| = |x| + \epsilon$. Thus the flow x was not maximum.

Conversely, assume that the residual graph G_x does not contain any directed path from s to t. Let $S = \{u \in V : \text{there exists a directed path in } G_x \text{ from } s \text{ to } u\}$. By definition, $s \in S$ and $t \notin S$ (otherwise there would be an augmenting path). Also, by definition, there is no arc in G_x from S to $V \setminus S$. This means that, for $e \in E$, if $e \in \delta^+(S)$ then $x_e = u(e)$ and if $e \in \delta^-(S)$ then $x_e = l(e)$. This implies that

$$C(S) = \sum_{e\in\delta^+(S)} u(e) - \sum_{e\in\delta^-(S)} l(e) = u(\delta^+(S)) - l(\delta^-(S)) = \sum_{e\in\delta^+(S)} x_e - \sum_{e\in\delta^-(S)} x_e = |x|.$$

This shows that the flow x is maximum and there exists an $s-t$ cut of the same capacity as $|x|$.

This almost proves Theorem 4.3. Indeed, as long as there exists an augmenting path, we can push flow along it, update the residual graph and continue. Whenever this algorithm stops, *if it stops*, we have a maximum flow and a corresponding minimum cut. But maybe this algorithm never stops; this can actually happen if the capacities might be irrational and the "wrong" augmenting paths are chosen at every iteration. To complete the proof of the max flow min cut theorem, we can simply use the linear programming formulation of the maximum flow problem and this shows that a maximum flow exists (in a linear program, the max is a real maximum (as it is achieved by a vertex) and not just a supremum which may not be attained). Starting from that flow x and constructing its residual graph G_x, we get that there exists a corresponding minimum $s-t$ cut of the same value.

4.2.1 Interpretation of max flow min cut

The max $s-t$ flow min $s-t$ cut theorem together with integrality of the maximum flow allows to derive several combinatorial min-max relations.

Bipartite matchings. Consider for example the maximum bipartite matching problem and its formulation as a maximum flow problem given in section 4.1.1. We said that for the arcs between A and B we had flexibility on how we choose $u(e)$; here, let us assume we have set them to be equal to $+\infty$ (or any sufficiently large integer). Consider any set

$S \subseteq (\{s\} \cup A \cup B)$ with $s \in S$ (and $t \notin S$). For $C(S)$ to be finite there cannot be any edge $(i,j) \in E$ between $i \in A \cap S$ and $j \in B \setminus S$. In other words, $N(A \cap S) \subseteq B \cap S$, i.e. if we set $C = (A \setminus S) \cup (B \cap S)$ we have that C is a vertex cover. What is the capacity $C(S)$ of the corresponding cut? It is precisely $C(S) = |A \setminus S| + |B \cap S|$, the first term corresponding to the arcs from s to $A \setminus S$ and the second term correspodning to the arcs between $B \cap S$ and t. The max $s-t$ flow min $s-t$ cut theorem therefore implies that there exists a vertex cover C whose cardinality equals the size of the maximum matching. We have thus rederived König's theorem. We could also derive Hall's theorem about the existence of a perfect matching.

Arc-disjoint paths. For the problem of teh maximum number of arc-disjoint paths between s and t, the max $s-t$ flow min $s-t$ cut theorem can be interpreted as Menger's theorem:

Theorem 4.4 *In a directed graph $G = (V, A)$, there are k arc-disjoint paths between s and t if and only if for all $S \subseteq V \setminus \{t\}$ with $s \in S$, we have $|\delta^+(S)| \geq k$.*

Exercise 4-2. At some point during baseball season, each of n teams of the American League has already played several games. Suppose team i has won w_i games so far, and $g_{ij} = g_{ji}$ is the number of games that teams i and j have yet to play. No game ends in a tie, so each game gives one point to either team and 0 to the other. You would like to decide if your favorite team, say team n, can still win. In other words, you would like to determine whether there exists an outcome to the games to be played (remember, with no ties) such that team n has at least as many victories as all the other teams (we allow team n to be tied for first place with other teams).

Show that this problem can be solved as a maximum flow problem. Give a necessary and sufficient condition on the g_{ij}'s so that team n can still win.

Exercise 4-3. Consider the following orientation problem. We are given an undirected graph $G = (V, E)$ and integer values $p(v)$ for every vertex $v \in V$. we would like to know if we can orient the edges of G such that the directed graph we obtain has at most $p(v)$ arcs incoming to v (the "indegree requirements"). In other words, for each edge $\{u, v\}$, we have to decide whether to orient it as (u, v) or as (v, u), and we would like at most $p(v)$ arcs oriented towards v.

1. Show that the problem can be formulated as a maximum flow problem. That is, show how to create a maximum flow problem such that, from its solution, you can decide whether or not the graph can be oriented and if so, it also gives the orientation.
2. Consider the case that the graph cannot be oriented and meet the indegree requirements. Prove from the max-flow min-cut theorem that there must exist a set $S \subseteq V$ such that $|E(S)| > \sum_{v \in S} p(v)$, where as usual $E(S)$ denotes the set of edges with both endpoints within S.

4.3 Efficiency of Maximum Flow Algorithm

The proof of the max $s-t$ flow min $s-t$ cut theorem suggests a simple augmenting path algorithm for finding the maximum flow. Start from any feasible flow and keep pushing flow along an augmenting in the residual graph until no such augmenting path exists. The main question we address now is how many iterations does this algorithm need before terminating.

As mentioned earlier, if the capacities are irrational, this algorithm may never terminate. In the case of integral capacities, if we start from an integral flow, it is easy to see that we always maintain an integral flow and we will always be pushing an integral amount of flow. Therefore, the number of iterations is bounded by the maximum difference between the values of two flows, which is at most $\sum_{e\in\delta(s)}(u(e)-l(e))$. This is finite, but not polynomial in the size of the input (which depends only logarithmically on the capacities u and l).

Shortest augmenting path variant. Edmonds and Karp proposed a variant of the augmenting path algorithm which is guaranteed to terminate in a polynomial number of iterations depending only on $n=|V|$ and $m=|E|$. No assumptions on the capacities are made, and the algorithm is even correct and terminates for irrational capacities.

The idea of Edmonds and Karp is to always find in the residual graph a *shortest* augmenting path, i.e. one with the fewer number of arcs. Given a flow x, consider the residual graph G_x. For any vertex v, let $d(v)$ denote the distance (number of arcs) from s to v in G_x. The shortest augmenting path algorithm is to select a path $v_0-v_1-\cdots-v_k$ in the residual graph where $v_0=s$, $v_k=t$ and $d(v_i)=i$.

The analysis of the algorithm proceeds as follows. Let P be a shotest augmenting path from s to t in G_x and let x' be the resulting flow after pushing as much flow as possible along P. Let d' be the distance labels corresponding to $G_{x'}$. Observe that only reverse arcs (i,j) along P (thus satisfying $d(i)=d(j)+1$) may get introduced in $G_{x'}$. Therefore, after augmentation, we have that $d(j)-d(i)\leq 1$ for every arc $(i,j)\in E_{x'}$. Summing these inqualities along the edges of any path P' in $G_{x'}$ from s to $j\in V$, we get that $d(j)\leq d'(j)$ for any $j\in V$. In particular, we have that $d(t)\leq d'(t)$. As distance labels can never become greater than $n-1$, we have that the distance to t can only increase at most $n-1$ times. But $d'(t)$ can also be equal to $d(t)$. In this case though, the fact that an arc of P is saturated means that there is one fewer arc (i,j) with $d(j)=d(i)+1$ in $G_{x'}$ than in G_x. Thus after at most m such iterations, we must have a strict increase in the distance label of t. Summarizing, this means that the number of augmentations is at most $m(n-1)$. The time it takes to build the residual graph and to find an augmenting path in it is at most $O(m)$ time. This means that the total running time of the shortest augmenting path algorithm is at most $O(m^2n)$. This can be further improved but this is not the focus of these notes.

4.4 Minimum cuts

From now on, we assume that we have only upper capacities u and no lower capacities l ($l(e)=0$ for all e). The minimum $s-t$ cut problem that we have solved so far corresponds