1 The dual of the max-flow linear problem

We’re working with a network \((N, A)\) with a capacity \(c_{ij} \geq 0\) for every arc \((i, j) \in A\). Let’s assume for simplicity that we’ve already “cleaned up” the network by making sure that there are no arcs going into \(s\) or out of \(t\) with positive capacity (they’d be useless).

Last time, we found that the linear program for finding a maximum flow in a network is

\[
\text{maximize } \sum_{j: (s, j) \in A} x_{sj} \\
\text{subject to } \sum_{i: (i, k) \in A} x_{ik} - \sum_{j: (k, j) \in A} x_{kj} = 0 \quad (k \in N, k \neq s, t) \\
x_{ij} \leq c_{ij} \quad (i, j) \in A \\
x \geq 0
\]

Now, it’s time to find the dual program.

We’ll have two types of dual variables: a dual variable \(u_k\) for every node \(k \in N\) other than \(s\) or \(t\) corresponding to flow conservation at \(k\), and a dual variable \(y_{ij}\) for every arc \((i, j) \in A\) corresponding to the capacity constraint \(x_{ij} \leq c_{ij}\).

Most variables \(x_{ij}\) appear in the primal in three constraints: the flow conservation constraint for \(i\) (with a coefficient of \(-1\)), the flow conservation constraint for \(j\) (with a coefficient of 1), and the capacity constraint for arc \((i, j)\) (with a coefficient of 1). This results in a dual constraint of \(-u_i + u_j + y_{ij} \geq 0\).

There are two exceptional cases. Variables \(x_{sj}\) (for arcs out of \(s\)) don’t have a flow conservation constraint for \(s\), so they only have \(u_j + y_{sj}\) on the left-hand side; also, \(x_{sj}\) appears in the objective function, so the right-hand side is 1. Similarly, variables \(x_{it}\) (for arcs into \(t\)) don’t have a flow conservation constraint for \(t\), so they only have \(-u_i + y_{it}\) on the left-hand side.

Altogether, we get the linear program

\[
\text{minimize } \sum_{(i, j) \in A} c_{ij} y_{ij} \\
\text{subject to } u_j + y_{sj} \geq 1 \quad (x_{sj}) \\
-u_i + u_j + y_{ij} \geq 0 \quad (x_{ij}, i \neq s, j \neq t) \\
-u_i + y_{it} \geq 0 \quad (x_{it}) \\
y \geq 0, u \text{ unrestricted}
\]
But this is just the beginning. We want to clean up this linear program to understand it better.

If we move the $u$ variables to the other side, the generic $(i, j)$ constraints become $y_{ij} \geq u_i - u_j$; the $(s, j)$ constraints become $y_{sj} \geq 1 - u_j$, and the $(i, t)$ constraints become $y_{it} \geq u_i$. This almost follows a universal pattern that works for every arc.

So let’s introduce two “fake” variables: a variable $u_s$ that’s always 1, and a variable $u_t$ that’s always 0. Then for every arc $(i, j)$, we get a constraint $y_{ij} \geq u_i - u_j$, no matter what $i$ and $j$ are. Now we have:

$$\begin{align*}
\text{minimize} & \quad \sum_{(i,j) \in A} c_{ij} y_{ij} \\
\text{subject to} & \quad y_{ij} \geq u_i - u_j \quad (\text{for all } (i,j) \in A) \\
& \quad u_s = 1 \\
& \quad u_t = 0 \\
& \quad y \geq 0, u \text{ unrestricted}
\end{align*}$$

There’s one more thing we can do to simplify this linear program, though it requires adding a not-strictly-speaking-linear constraint. There are two lower bounds on $y_{ij}$: it is at least $u_i - u_j$, and it is at least 0. Since we are minimizing a sum of $y_{ij}$’s with nonnegative coefficients, we want to make $y_{ij}$ as small as possible. So it should be the bigger of these two lower bounds: we should always have $y_{ij} = \max\{u_i - u_j, 0\}$.

If we substitute that into the linear program, we can get a dual just in terms of $u$:

$$\begin{align*}
\text{minimize} & \quad \sum_{(i,j) \in A} c_{ij} \max\{u_i - u_j, 0\} \\
\text{subject to} & \quad u_s = 1 \\
& \quad u_t = 0
\end{align*}$$

Although the variables $u_i$ are unrestricted, we can make some assumptions about their values. We will never want to set a variable $u_k$ smaller than the smallest $u_j$ with an arc $(k,j)$, or larger than the largest $u_i$ with an arc $(i,k)$. Since $u_s$ and $u_t$ are fixed at 1 and 0, we want to put the other $u_i$ somewhere between those, so we can assume that $0 \leq u_i \leq 1$ for all $i$.

If we assume that actually, every variable $u_i$ is either 0 or 1, then we can give this problem a combinatorial interpretation. Let $S = \{i \in N : u_i = 1\}$ and let $T = \{i \in N : u_i = 0\}$. Then we must have $s \in S$ and $t \in T$, and the objective function is just a sum of $c_{ij}$ where $i \in S$ and $j \in T$. So $(S,T)$ is a cut and we are minimizing its capacity: this linear program is a search for the minimum cut.

## 2 The Max-flow min-cut theorem

This duality is halfway to proving the following big result:

**Theorem 2.1.** In any network, the value of a maximum flow is equal to the capacity of a minimum cut.
Strong duality tells us that the max-flow linear program and the min-cut linear program have the same optimal objective value.

However, to know that the min-cut linear program actually has a minimum cut as its optimal solution, we’d need to know that the optimal solution is an integer.

To prove this, we will use the following formulation:

\[
\begin{align*}
\text{minimize} & \quad \sum_{(i,j) \in A} c_{ij} y_{ij} \\
\text{subject to} & \quad y_{ij} - u_i + u_j \geq 0 \quad (\text{for all } (i,j) \in A) \\
& \quad u_s = 1 \\
& \quad u_t = 0 \\
& \quad y \geq 0, u \geq 0.
\end{align*}
\]

We’ve added the \( u \geq 0 \) constraint to fit with our theorem about totally unimodular matrices—it only dealt with nonnegative variables. That theorem actually works for all basic solutions, but just to avoid dealing with the technicality, let’s assume \( u \geq 0 \). This doesn’t change the optimal solutions, because we know negative values of \( u \) will never help.

To understand the constraint matrix, let’s think about its columns, which correspond to variables in the linear program.

- Each \( u_k \) column has a 1 in the rows for edges going into \( k \) (that is, edges of the form \((i,k)\)) and a \(-1\) in the rows for edges going out of \( k \) (that is, edges of the form \((k,j)\)).
- The \( u_s \) and \( u_t \) columns also have a single 1 in the rows for the \( u_s = 1 \) and \( u_t = 0 \) constraints, respectively.
- Each \( y_{ij} \) column is almost entirely made of zeroes. It has a single 1, in the row for the \((i,j)\) edge’s constraint.

Now let’s think about the determinants of \( k \times k \) submatrices.

As with the proof of total unimodularity for the bipartite matching problem, we can eliminate some \( k \times k \) matrices because they can be reduced to smaller cases. In particular, we can reduce to a smaller submatrix whenever we have a row or column with only a single nonzero entry in it.

(\text{Exception: a } 1 \times 1 \text{ submatrix like this doesn’t reduce to anything smaller, but we can check that all entries are } -1, 0, \text{ or } 1, \text{ so } 1 \times 1 \text{ submatrices are all fine.})

This means that in cases that don’t reduce to smaller cases, our submatrix doesn’t use any of the \( y_{ij} \) columns. It doesn’t use the \( u_s = 1 \) row or the \( u_t = 0 \) row. And whenever the row for some \( y_{ij} - u_i + u_j \geq 0 \) constraint is used, both the columns \( u_i \) and \( u_j \) must appear.

But now, every single row of our submatrix \( M \) has two nonzero entries: a \(-1\) and a \(1\). This means that if we multiply \( M \) by the \( k \)-dimensional vector \( 1 = (1,1,\ldots,1) \), we get \( M1 = 0 \). This means that \( \det(M) = 0 \), because \( M \) has a nontrivial null space.

Therefore all \( k \times k \) submatrices have determinant \(-1, 0, \text{ or } 1\). This means that the constraint matrix is totally unimodular. This means that the optimal solution to the min-cut linear program actually represents a minimum cut. And this proves the max-flow min-cut theorem.