Lecture 7

Duality Applications (Part II)[*]

In this lecture, we'll look at applications of duality to three problems:

1. Finding *maximum spanning trees* (MST). We know that Kruskal's algorithm finds this, and we'll see a proof of optimality by constructing an LP for MST, and exhibiting a feasible dual solution whose cost is equal to the MST.
2. Finding *minimum cost arborescences*. We'll see an algorithm given independently by Edmonds, Chu & Liu, and Bock, which uses the dual to guide the algorithm, and to give a proof of the optimality of the solution.
3. Finally, we'll look at an LP formulation of *non-bipartite matchings*: this formulation is due to Edmonds, and we'll give a proof (due to Schrijver) that shows the integrality of all vertices of the perfect matching polytope we write down.

7.1 Maximum spanning tree

Given a graph $G = (V, E)$, and edge weights $w_e \geq 0$, the goal is to output a *spanning tree* of maximum weight. To design a linear program for this problem, we use variables $\{x_e\}_{e \in E}$.

Notation 7.1. For a set $S \subseteq V$, we denote by δS the set of edges leaving S. For $A \subseteq E$, define $x(A) = \sum_{e \in A} x_e$.

Consider the following LP.

$$\begin{aligned} \max \quad & \sum_{e \in E} w_e x_e \\ \text{s.t.} \quad & 1 \geq x_e \geq 0 \\ & \sum_{e \in E} x_i = n - 1 \\ & x(\delta S) \geq 1 \quad \forall S \neq \varnothing, V \end{aligned}$$

[*]Lecturer: Anupam Gupta. Scribe: Carol Wang.

Note: In class we left it as an exercise to see whether every vertex of this LP was integral. It is *not*: on the blog we later saw the following counterexample.

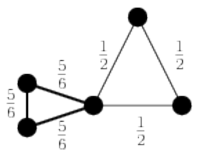


Figure 7.1:

The maximum weight spanning tree has weight 2. However, the LP solution given here (with $x_e = 1/2$ on the thin edges, and $x_e = 5/6$ on the thick ones) has $w^\top x = 3 \cdot 5/6 = 2.5$. It also has $\sum_e x_e = 3 \cdot 1/2 + 3 \cdot 5/6 = |V| - 1$, and you can check it satisfies the cut condition. (In fact, the main gadget that allows us to show this LP has an "integrality gap" is to assign 1/2 to the edges of the thin triangle — much like for the naive non-bipartite matching LP you'll see later in this lecture.)

Well, we tried. Let's consider a slightly different LP. For $S \subseteq V$, let E_S denote all edges between vertices in S. (For simplicity, we will assume in this lecture that all the edge weights are non-negative.)

$$\begin{aligned} \max \quad & \sum_{e \in E} w_e x_e \\ \text{s.t.} \quad & x(E_S) \le |S| - 1 \quad \forall S \subseteq V,\, |S| \ge 1 \\ & x_e \ge 0 \end{aligned} \tag{P}$$

Remark 7.2. Any spanning tree satisfies these constraints. Therefore, opt(P) is at least the weight of the maximum spanning tree.

Recall that Kruskal's algorithm starts with a forest consisting of all the vertices, and iteratively adds the heaviest edge which connects two trees.

Theorem 7.3. *There exists an integer optimum for the LP P, and Kruskal's algorithm finds it.*

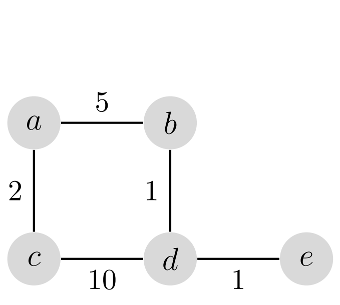
Proof. We will construct a dual solution such that its value is the value of the MST which Kruskal finds. Let's write down the dual.

Notation 7.4. For a set S, write $r(S) := |S| - 1$. (This is the size of a spanning tree on S.)

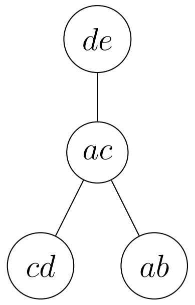
Then the dual of P is

$$\begin{aligned} \min \quad & \sum_{S \neq \varnothing} r(S) y_S \\ \text{s.t.} \quad & \sum_{S : e \in E_S} y_S \ge w_e \, \forall e \in E \\ & y_S \ge 0 \end{aligned} \tag{D}$$

That is, we should assign a value to each nonempty subset of vertices which gives "enough" weight to each edge.



(a) Graph G



(b) Resulting tree

Figure 7.2: Example for Kruskal's algorithm

Primal solution

Kruskal: Pick edges $K = \{e_1, e_2, \ldots, e_{n-1}\}$ with $w(e_1) \geq w(e_2) \geq \cdots \geq w(e_{n-1})$. Then a primal solution is given by

$$x_e = \begin{cases} 1 & e \in K \\ 0 & e \notin K \end{cases}.$$

The value of this solution is $\sum_e w_e x_e$, which is exactly the value of the Kruskal solution.

Dual solution

Suppose that we run Kruskal on our graph. We consider the sequence of components satisfied by the addition of each edge. This naturally induces a tree structure on the edges of the MST, where the parent of a subtree corresponding to some component C is the first edge added to the MST which leaves C.

For example, in Figure 7.2a, choosing the edges (c, d) and (a, b) satisfy the components $\{a, b\}$ and $\{c, d\}$, and adding (a, c) satisfies the entire component $\{a, b, c, d\}$. The final edge (d, e) then satisfies the entire tree.

We will consider the tree induced by Kruskal's algorithm.

Notation 7.5. Let $V(e_j)$ be the set of vertices spanned by the edges in the subtree rooted at e_j.

We will write $T(e_j)$ for this subtree.

Define a dual solution y_S by

$$y_S = \begin{cases} w_{e_j} - w_{\text{parent}(e_j)} & S = V(e_j) \text{ for some } j \\ 0 & \text{else} \end{cases}.$$

Example 7.6. For Figure 7.2b, we have $y_{\{c,d\}} = 10 - 2 = 8$, $y_{\{a,b\}} = 5 - 2 = 3$, $y_{\{a,b,c,d\}} = 2 - 1 = 1$, and $y_{\{a,b,c,d,e\}} = 1 - 0 = 1$.

We will show both that this solution is feasible, and that its value is exactly the value of the maximum spanning tree, proving the theorem.

Lemma 7.7. *y_S is feasible.*

Proof. Kruskal's algorithm is greedy, and the parent of any edge is added after that edge, so $y_S \geq 0$.

To show $\sum_{S:e\in E_S} y_S \geq w_e$ for every edge, fix an edge e. Consider the first time e lies in $T(e_j)$ for some j, and consider a path $p_1 = e_j, \ldots, p_k$ from e_j to the root of our tree. e lies in $V(p_i)$ for each i, and in particular, by nonnegativity,

$$\sum_{S:e\in E_S} y_S \geq \sum_i y_{V(p_i)} = w_{p_1} - w_{p_2} + \cdots + w_{p_k} = w_{p_1} = w_{e_j} \geq w_e,$$

where in the last step we used the fact that if w_e were greater than w_{e_j}, then we would have chosen e rather than e_j. □

Lemma 7.8.

$$\sum_{S\neq\varnothing} r(S) y_S = \sum_{e\in K} w_e.$$

(Recall K is the spanning tree output by Kruskal.)

Proof. We prove by (strong) induction the slightly stronger statement that

$$\sum_{S\subseteq V(e_j)} r(S) y_S = \sum_{e\in T(e_j)} w_e - r\big(V(e_j)\big) w_{\text{parent of } e_j}$$

for every $e_j \in K$.

We induct on the number of nodes in the subtree $T(e_j)$. In the base case, $T(e_j)$ is a leaf, so $|V(e_j)| = 2$ and the claim holds by our definition of y_S.

Therefore, suppose that the claim holds for subtrees of $\leq k$ nodes, and consider $T(e_j)$ of $k+1$ nodes.

Case 1. e_j has one child, e, in $T(e_j)$. Then $V(e_j) = V(e) \cup \{u\}$ for some vertex $u \notin V(e)$. In particular, $r(V(e)) = r(V(e_j)) - 1$. Then

$$\begin{aligned}\sum_{S\subseteq V(e_j)} r(S) y_S &= \left(\sum_{S\subseteq V(e)} r(S) y_S\right) + r(V(e_j)) y_{V(e_j)} \\ &= \left(\sum_{e\in T(e)} w_e - r(V(e)) w_{e_j}\right) + r(V(e_j)) y_{V(e_j)},\end{aligned}$$

using the inductive hypothesis.

Since $r(V(e)) = r(V(e_j)) - 1$ and $y_{V(e_j)} = w_{e_j} - w_{\text{parent}(e_j)}$, the claim holds.

Case 2. e_j has two children e, e' in $T(e_j)$. Then $V(e_j) = V(e) \cup V(e')$, and $V(e) \cap V(e') = \varnothing$. In particular, $r(V(e_j)) = r(V(e)) + r(V(e')) + 1$. Applying the inductive hypothesis to $T(e)$ and $T(e')$, we can simplify as in Case 1.

Recall that $y_S = 0$ unless $S = V(e)$ for some $e \in K$.

□

Thus the maximum spanning tree LP has an integer optimum given by Kruskal's algorithm.

□

7.2 Minimum cost arborescence

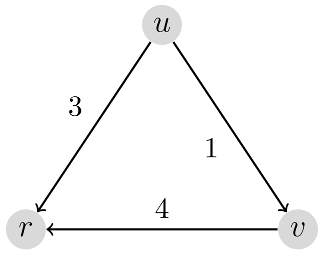
Think of these as spanning trees on *directed* graphs. Given a directed graph $G = (V, E)$ with a root vertex r, an *arborescence* of G is a subgraph $T = (V, E_T)$ such that:

1. Every node is connected to r, and there are no cycles even if we ignore directions.
2. Every node has a directed path to r.

Remark 7.9. One often sees this definition with directed paths from r to other nodes, but we will use this convention.

Remark 7.10. An arborescence may not exist for a given root r, but a certificate of infeasibility is a vertex with no path to r.

Note that it may also not be unique. Furthermore, the following example shows that adapting Prim's algorithm (greedily starting at the root) may not yield an optimal solution.



Notation 7.11. We write $\delta^+ S$ to denote the set of edges leaving S for any $S \subseteq V$.

We will assume $c_e \geq 0$ for every e. The primal LP is

$$\begin{aligned} \min \quad & \sum_{e \in E} c_e x_e \\ \text{s.t.} \quad & x(\delta^+ S) \geq 1 \qquad S \text{ valid} \\ & x_e \geq 0 \end{aligned} \tag{P}$$

and the dual is

$$\begin{aligned} \max \quad & \sum_{S \text{ valid}} y_S \\ \text{s.t.} \quad & \sum_{S : e \in \delta^+ S} y_S \leq c_e \qquad \forall e \\ & y_s \geq 0 \end{aligned} \tag{D}$$

where we will call $S \subseteq V$ valid if S is nonempty and $r \notin S$.

Algorithm for minimum cost arborescence

1. If zero-weight edges connect every $v \neq r$ to r, then we get a (integral) primal solution of value 0 (using depth first search or similar). A matching dual solution sets $y_S = 0$ for every S. In particular, this is optimal.

2. Otherwise, consider the graph restricted to zero-weight edges. Choose a maximal strongly connected component C of this subgraph. Then in the graph with all edges, there are no zero-weight edges out of C. Let $c^* = \min_{e \in \delta^+ C} c_e$. For each $e \in \delta^+ C$, define updated weights $c'_e = c_e - c^*$.

   Run the algorithm recursively on G with C contracted to one vertex $\hat{C}$ and with the updated weights to get optimal primal/dual solutions x_e, y_S for the contracted graph. Inductively, x_e will be integral.

   Let A be an arborescence on C (of zero cost). Define primal/dual solutions $\hat{x}_e, \hat{y}_S$ for the uncontracted graph by

$$\hat{x}_e = \begin{cases} x_e & e \notin C \\ 1 & e \in A \\ 0 & e \in C \setminus A \end{cases} \qquad \hat{y}_S = \begin{cases} y_S & C \not\subseteq S \\ y_{\hat{C}} + c^* & S = C \\ y_{S \setminus C \cup \{\hat{C}\}} & C \subsetneq S \end{cases}.$$

Remark 7.12. If $\sum c'_e x_e = \sum y_S$ (i.e., if the primal and dual solutions mutually certify optimality), then $\sum c_e \hat{x}_e = \sum \hat{y}_S$. This holds since $\sum y_S + c^* = \sum \hat{y}_S$. Furthermore, in any minimal arborescence on the contracted graph, exactly one edge from $\delta^+ C$ will be chosen.

Lemma 7.13. *$\hat{x}_e$ and $\hat{y}_S$ are feasible.*

Proof. $\hat{x}_e$ is feasible because x_e is feasible (and clearly the arborescence A satisfies the conditions). To show $\hat{y}_S$ is feasible, we only need to check $\sum_{S: e \in \delta^+ S} \hat{y}_S \leq c_e$ for $e \in \delta^+ C$. It is easy to see that this holds by definition of $\hat{y}_S$, since $\sum y_S \leq c_e - c^*$. □

Note: This LP we wrote for arborascences is very similar to the one we first wrote for spanning trees, but that one did not work, whereas this one does! Interesting. Indeed, this LP can be used to give an algorithm for MSTs on undirected graphs.

Indeed, take an undirected graph and replace each undirected edge by two directed edges of the same weight, pointing in opposite directions. Now the max-weight arborescence in this digraph has the same weight as the maximum spanning tree in the original graph. So an LP that looks pretty similar to the failed undirected one (namely $\max\{w^\top x \mid x(\partial v) = 1, x(\partial S) \geq 1, x \geq 0\}$) on that specially constructed *directed* graph gives an arborescence that corresponds to an integral maximum spanning tree on the original undirected graph.

7.3 Minimum cost perfect matching

We saw how to do this in the bipartite case, but suppose we just have a general graph.

Here is the LP we were using before.

$$\begin{aligned} \min \quad & \sum_e c_e x_e \\ \text{s.t.} \quad & \sum_{e \in \delta v} x_e = 1 \quad v \in V \\ & x_e \geq 0 \end{aligned}$$

This does not necessarily have integer vertices: consider an unweighted triangle. By assigning each edge 1/2, we get a solution of value 1.5, but the maximum integral matching has value 1.

But suppose we added the constraint $x(\delta S) \geq 1$ for every odd set S. Now our LP is

$$\begin{aligned} \min \quad & \sum_e c_e x_e \\ \text{s.t.} \quad & \sum_{e \in \delta v} x_e = 1 \quad v \in V \\ & x(\delta S) \geq 1 \quad 2 \nmid |S| \\ & x_e \geq 0 \end{aligned} \tag{P}$$

Note that in the second set of constraints, we can just consider S of size at least 3 and at most $|V|-3$: if $|S| = 1$ or $|V|-1$, then the first set of constraints already implies $x(\partial S) = 1$. So just focus on

$$\begin{aligned} x(\partial v) &= 1 && \forall v \in V \\ x(\partial S) &\geq 1 && \forall S \subset V, |S| \text{ odd }, 3 \leq |S| \leq |V| - 3 \\ x &\geq 0 \end{aligned}$$

Let us call this perfect matching polytope PM. We'll call the first set of equalities the *vertex constraints*, and the second set the *odd-set inequalities.*

Remark 7.14. The odd-set inequalities are satisfied by any perfect integral matching, because at least one vertex in an odd set must be matched to a vertex outside the set.

Theorem 7.15 (Edmonds)**.** *Every vertex of P is integral.*

Proof. This was proven on the course blog. For completeness, the proof is copied here.

Suppose not, and suppose there exists graphs for which there is a fractional vertex. Consider a minimal counterexample $G = (V, E)$ (minimizing the sum of $|V| + |E|$, say), and some vertex solution x that is not integral. Clearly, $|V|$ must be even, else it will not satisfy the odd set constraint for $S = V$. First, the claim is that G cannot have a vertex of degree 1, or be disconnected (else we'll get a smaller counterexample) or be just an even cycle (where we know this LP is indeed integral). Being connected implies that $|E| \geq |V| - 1$, and neither being a cycle nor having a degree-1 vertex implies that $|E| \neq |V|$. So $|E| > |V|$.

Recall there are $|E|$ variables. So any vertex/BFS is defined by $|E|$ tight constraints. If any of these tight constraints are the non-negativity constraints $x_e \geq 0$, then we could drop that edge e and get a smaller counterexample. And since at most $|V|$ tight constraints come from the vertex constraints. So at least one odd-set constraint should be tight. Say this tight odd-set constraint is for the odd set $W \subseteq V$ with $|W| \geq 3$: i.e.,

$$x(\partial W) = 1$$

Now consider the two graphs G/W and $G/\overline{W}$ obtained by contracting W and $\overline{W}$ to a single new vertex respectively, and removing the edges lying within the contracted set. Since both W and $\overline{W}$ have at least 3 vertices, both are smaller graphs.

Now x naturally extends to feasible solutions y and z for these new graphs. E.g., to get y, set $y_e = x_e$ for all edges $e \in E \setminus \binom{W}{2}$. Note that if the set W got contracted to new vertex $\widehat{w}$ in G/W, then the fact that $x(\partial W) = 1$ implies that $y(\partial \widehat{w}) = 1$, and hence y is a feasible solution to the perfect matching polytope for graph G/W. Similarly, z is a feasible solution to the perfect matching polytope for graph $G/\overline{W}$.

By minimality of G, it follows that the perfect matching LP is integral for both G/W and $G/\overline{W}$: i.e., the vertices of the perfect matching polytope for these smaller graphs all correspond to perfect matchings. And that means that

$$y = \sum_i \lambda_i \cdot \chi_{M_i},$$

where χ_{M_i} is the natural vector representation of the perfect matching M_i in G/W, for values $\lambda_i \geq 0, \sum_i \lambda_i = 1$. Also, λ_i's can be taken to be rational, since y is rational, as are χ_{M_i}. Similarly, we have a rational convex combination

$$z = \sum_i \mu_i \cdot \chi_{N_i},$$

where N_i are perfect matchings in $G/\overline{W}$. Since λ_i, μ_i are rationals, we could have repeated the matchings and instead written

$$y = \frac{1}{k} \sum_i \chi_{M_i}$$
$$z = \frac{1}{k} \sum_i \chi_{N_i}$$

Finally, we claim that we can combine these to get

$$x = \frac{1}{k} \sum_i \chi_{O_i}$$

where O_i's are perfect matchings in G. How? Well, focus on edge $e = \{l, r\} \in \partial W$, with $l \in W$. Note that $y_e = z_e = x_e$. If we look at k matchings M_i in the sum for y: exactly x_e fraction of these matchings M_i – that is, kx_e matchings – contain e. Similarly, exactly kx_e

of the matchings N_i in the sum for x contain e. Now we can pair such matchings (which share a common edge in ∂W) up in the obvious way: apart from the edge e, such an M_i contains edges only within $\overline{W}$ and matches up all the vertices in $\overline{W}$ except vertex r, and N_i contains edges only within W and matches up all the vertices in $W \setminus \{l\}$. And e matches up $\{l, r\}$. Hence putting together these perfect matchings M_i and N_i in G/W and $G/\overline{W}$ gets us a perfect matching O_i for G.

So x can be written as a convex combination of perfect matchings of G. Hence, for x to be an extreme point (vertex) itself, it must be itself a perfect matching, and integral. This gives us the contradiction. $\square$

Max-Weight Matchings

We didn't get to this, but suppose you want to write an LP whose vertices are precisely (integral) matchings in G, not just the perfect matchings. Here is the polytope Edmonds defined.

$$\begin{aligned} x(\partial v) &\leq 1 && \forall v \in V \\ \textstyle\sum_{e \in \binom{S}{2}} x_e &\leq \tfrac{|S|-1}{2} && \forall S \subset V, |S| \text{ odd} \\ x &\geq 0 \end{aligned}$$

Clearly, all matchings in G are feasible for this LP. Moreover, one can use the Perfect Matching Theorem above to show that every vertex of this polytope is also integral.