CS 787: Advanced Algorithms

Linear Programming

Instructor: Dieter van Melkebeek

This lecture covers the notion of a linear program, its dual, and the relationship between the two. We give some examples and survey known algorithms.

1 Notions

A linear program (LP) is the problem of optimizing (maximizing or minimizing) a linear multivariate objective function over the reals under linear equality and inequality ($\leq$) constraints.

Example: Maximize $x_1 + x_2$ given the following constraints:

$$\begin{aligned} 5x_1 - 2x_2 &\geq -2 \\ 4x_1 - x_2 &\leq 8 \\ 2x_1 + x_2 &\leq 10 \\ x_1 &\geq 0 \\ x_2 &\geq 0 \end{aligned}$$

See Figure 1 for a pictorial representation. ⊠

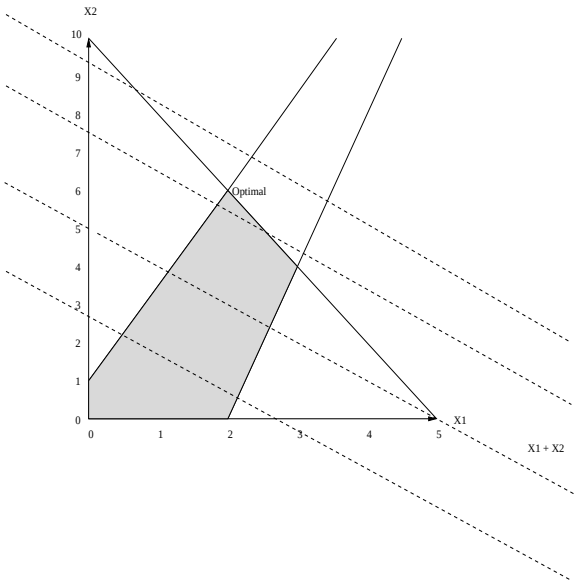

Figure 1: The feasible region and contour lines for our example linear program.

The *feasible region* of a linear program is the set of points satisfying the constraints. It is a *convex polytope* defined by the intersection of the hyperplanes and closed halfspaces given by the linear equality and inequality constraints.

We distinguish between three cases:

1. The constraints cause the feasible region to be the empty set. In this case, the linear program has no solution, and we call it infeasible.
2. The feasible region is non-empty but the objective function is unbounded on the feasible region. In this case, there is no optimal solution to the linear program, and we call the program feasible and unbounded.
3. The feasible region is non-empty and the objective function is bounded on the feasible region. In this case, the optimum is realized at a vertex of the convex polytope (and possibly also at other points), and we call the program feasible and bounded.

Note that the number of vertices of a convex polytope can be exponential in the number of constraints.

As you can see in Figure 1, case three applies to our example, and the optimum is reached at the point $(x_1, x_2) = (6, 2)$.

We defined a linear program as an optimization problem. The corresponding decision version takes the linear program and an additional real t, and asks whether there exists a feasible point whose objective value is at least t (in the case of a maximization problem) or at most t (in the case of a minimization problem). The extra condition on the feasible point is another linear inequality constraint. It follows that the linear programming decision problem is equivalent to the problem of deciding whether the feasible region of an LP is non-empty. The search version of the problem asks for a feasible point if one exists, or reporting that none exists.

All three versions of the problem are computationally equivalent under polynomial-time reductions. The reduction from decision via search to optimization is trivial. The other direction uses a binary search for the optimum value followed by binary searches for the bits of a vertex realizing that value (in case it is finite), and hinges on the fact that the bit-length of the vertices of a polytope is polynomially bounded by the bit-length of the defining constraints. We leave the details as an exercise.

2 Examples

We now give some examples of problems we have already discussed that can be easily cast as linear programs.

2.1 Max Flow

Recall the definition of the network flow problem. We have a directed graph $G(V, E)$ whose edges $e \in E$ have a capacity $c_e \in [0, \infty)$. Two vertices of G are distinguished as source s, and sink t. Introducing a variable x_e for the flow through each edge $e \in E$, the Max Flow problem can be rephrased as a linear program as follows:

$$\max \sum_{e=(s,v)\in E} x_e \quad s.t. \quad \begin{cases} (\forall v \in V \backslash \{s,t\}) & \sum_{e=(u,v)\in E} x_e - \sum_{e=(v,w)\in E} x_e = 0 \\ (\forall e \in E) & x_e \leq c_e \\ (\forall e \in E) & x_e \geq 0 \end{cases} \tag{1}$$

We try to maximize the net outward flow from the source under the constraints that the flow going into any vertex (other than source and sink) is equal to the flow coming out of it, and the flow through any edge is non-negative and less than (or equal to) the capacity of the edge.

2.2 Min-Cost Flow

A linear program formulation allows us to easily add constraints. For example, we can add to the max flow problem the constraint of finding a max flow of minimum cost. More generally, we can expres the min-cost flow problem, which refers to the problem of finding a flow of minimum cost realizing at least a given target value t. In this case, each edge of the network has a cost associated with it which is linearly proportional to the amount of flow through that edge. We can cast this problem as a linear program as follows:

$$\min \ \sum d_e x_e \qquad s.t. \ \begin{cases} (1) \text{ is satisfied and} \\ \nu(f) \geq t \end{cases}$$

2.3 Shortest Path

We can also convert the shortest path problem on digraphs with arbitrary edge lenghts into a linear program. Our method is based on the Bellman-Ford algorithm for computing shortest paths. As before, we have a graph $G(V, E)$ with edge lengths $\ell : E \to (-\infty, \infty)$, and two vertices - start vertex s and destination t. We introduce a variable x_v for every $v \in V$ and consider the following linear program:

$$\max \ x_t \qquad s.t. \ \begin{cases} (\forall e = (u, v) \in E) \quad x_v \leq x_u + \ell(e) \\ x_s = 0 \end{cases}$$

To argue the equivalence, let us first consider the case where there are negative loops, so the problem is well-defined. The reasoning behind Bellman-Ford shows that if $x_v = d(s, v)$ (the length of a shortest path from s to v, then all constraints of the LP are satisfied. Hence, $x_t \geq d(s, t)$. For the converse, consider any path $(s, v_1, v_2, \ldots, v_k, t)$ of minimum length $d(s, t)$ from s to t. Then, $x_{v_1} \leq \ell(s, v_1)$, $x_{v_2} \leq x_{v_1} + \ell(v_1, v_2) \leq \ell(x, v_1) + \ell(v_1, v_2)$ and so on. So, $x_t \leq \ell(s, v_1) + \ell(v_1, v_2) + \cdots + \ell(v_k, t) = d(s, t)$. So, the maximum value of x_t under the given constraints is equal to the length $d(s, t)$ of a shortest path from s to t.

We leave the remaining cases as an exercise.

3 Algorithms for Linear Programming

There are many (classes of) algorithms for linear programming. We briefly discuss three of them, and refer to dedicated courses for more details.

- *Simplex Algorithm.* This was the first algorithm developed for linear programming problem, and came about in the 50's. It maintains a vertex of the feasible region, and in each step moves to a neighboring vertex (one that can be reached by following an edge) that does deteriorate the objective value. Several criteria (known as pivoting rules) are used to decide which neighbor to move to next. Eventually, we can't move to a neighboring vertex that improves the objective function, which means we are in a local optimum. By the convexity of the problem, a local optimum is also a global optimum, so we are done.

This short description raises a number of issues (how to find the start point, how to detect no further improvement is possible), which can all be handled. A more fundamental problem is the running time of the simplex algorithm - there is no known pivot rule that guarantees a polynomial running time. In fact, all of the pivot rules used in practice have exponential worst-case running times. Nevertheless, they seem to behave very well in practice. One relatively recent theoretical explanation for this phenomenon is the following "smoothed analysis": If small perturbations to the input are allowed, then for every input the running time of the simplex algorithm with the usual pivot rules is polynomially bounded with very high probability. This means that the inputs on which the behavior is bad are pathetic and simply do not happen in practice, where small perturbations automatically occur as measurement errors.

- *Ellipsoid Algorithm.* This was the first LP algorithm that was proven to run in polynomial time on every input. It came about in the 80's. The algorithm finds a feasible point. It maintains an ellipsoid that contains the feasible region, and considers the center of the ellipsoid as the candidate feasible point. If the candidate point violates one of the constraints, we cut the ellipsoid with the parallel hyperplane through the center. The feasible region entirely falls in one of the resulting halves. In the next step we consider the ellipsoid of minimum volume that encloses that half of the previous ellipsoid.

  The calulations for every step can be executed efficiently. One can show that the volume of the ellipsoid decreases by at least some constant factor in every step. It follows that the process ends after a polynomial number of steps (in the bit-length of the input) provided we start with an ellipsoid of exponentially-bounded volume, and the feasible region has positive volume. The provisos can be handled, resulting in an algorithm that is guaranteed to finish in polynomial time. However, the algorithm does not perform well in practice.

- *Interior Point Methods.* These algorithms maintain an interior point of the polytope, and move it around in an appropriate way until it is optimal. Their worst-case running time is polynomially bounded, and they can compete with the simplex algorithm in practical usefulness.

An interesting property of the polynomial-time algorithms for linear programming is that limited access they need to the program. They only need a *separation oracle*, which is a procedure that tells whether a given point falls within the feasible region, and if not, produces a constraint that is not satisfied by the point. This allows us to handle certain linear programs with exponentially many constraints but only a polynomial number of variables in polynomial time. We refer to specialized courses on optimization for more details. In the rest of this course we will just use polynomial-time algorithms for linear progamming as a blackbox.

We point out that, in contrast to the special case of max-flow, the existence of a *strongly* polynomial-time algorithm for LP remains open. In all of the known polynomial-time algorithms for LP the number of iterations depends on the bit-length of the input.

4 Weak Duality

We now develop the notion of duality for linear programs. In this section we establish a weak connection between a primal problem and its dual, and strenghten it in the next section. The

results generalize the max-flow min-cut theorem, and are a special case of the duality in convex programming.

In order to facilitate the development, we first introduce the *standard form* for linear programming:

$$\max \;\; c^T x \qquad s.t. \;\; \begin{cases} Ax \leq b \\ x \geq 0 \end{cases} \tag{2}$$

Here $c \in \mathbb{R}^n$ denotes an n-dimensional column-vector, $A\mathbb{R}^{m \times n}$) an $m \times n$ matrix, and b an m-dimensional column vector, and the vector inequalities like $Ax \leq b$ and $x \geq 0$ are interpreted component-wise.

If you are given a linear program that is not in the standard form above, you can transform it into standard form by doing the following:

1. If you are given an inequality in the wrong direction, multiply both sides by -1.
   For example: $3x + 2y \geq 2 \rightarrow -3x - 2y \leq -2$.

2. If you are given an equality constraint, write it as two inequality expressions in each direction.
   For example: $3x + 2y = 2 \rightarrow 3x + 2y \leq 2 \wedge 3x + 2y \geq 2$.

3. If you have an unconstrained variable, write it as the difference of two constrained variables.
   For Example: $3x \leq 5 \rightarrow 3(x_1 - x_2) \leq 5$ with $x_1 \geq 0 \wedge x_2 \geq 0$.

Applying those transformations to our example program yields the following standard form:

$$\begin{aligned} \max \;\; x_1 + x_2 \qquad & s.t. & & \\ -5x_1 + 2x_2 \;\; & \leq \;\; 2 & & (3) \\ 4x_1 - x_2 \;\; & \leq \;\; 8 & & (4) \\ 2x_1 + x_2 \;\; & \leq \;\; 10 & & (5) \\ x_1 \;\; & \geq \;\; 0 & & \\ x_2 \;\; & \geq \;\; 0 & & \end{aligned}$$

As mentioned before, the optimum value in this example is 8.

Note that if we take linear combinations with non-negative coefficients of the constraints, we obtain another linear equality that holds for all points in the feasible region. By choosing the coefficients judiciously, we may be able to get an upper bound on the objective value for points in the feasible region. For instance:

$$\frac{1}{9} \cdot (3) + 0 \cdot (4) + \frac{7}{9} \cdot (5) \Rightarrow x_1 + x_2 \leq 8$$

This gives an upper bound on the objective function which we sought to maximize. In fact, this is a tight upper bound, as we know we already know we can realize the value 8. We will see that this is not a coincidence.

Let us formalize the process of obtaining an upper bound on the optimum value of our linear program. We introduce a non-negative coefficient with each of the three constraints, say y_1, y_2, y_3. Now

$$y_1 \cdot (3) + y_2 \cdot (4) + y_3 \cdot (5) \Rightarrow (-5y_1 + 4y_2 + 2y_3)x_1 + (y_1 - y_2 + y_3)x_2 \leq 2y_1 + 8y_2 + 10y_3$$

The right-hand side is an upper bound for the optimum of the LP whenever the coefficients of the x_j's on the left-hand side are at least as large as in the objective function, i.e., whenever:

$$\begin{aligned} -5y_1 + 4y_2 + 2y_3 &\geq 1 \\ y_1 - y_2 + y_3 &\geq 1 \end{aligned}$$

We'd like to make this upper bound as small as possible. This leads is captured by the following linear program:

$$\begin{aligned} \min \quad 2y_1 + 8y_2 + 10y_3 \qquad s.t. & \\ -5y_1 + 4y_2 + 2y_3 &\geq 1 \\ y_1 - y_2 + y_3 &\geq 1 \\ y_1 &\geq 0 \\ y_2 &\geq 0 \\ y_3 &\geq 0 \end{aligned}$$

This program is known as the dual of the original program, which is referred to as the primal. More generally, if the *primal* is:

$$\max \quad c^T x \qquad s.t. \quad \begin{cases} Ax \leq b \\ x \geq 0, \end{cases}$$

then the *dual* is:

$$\min \quad b^T y \qquad s.t. \quad \begin{cases} A^T y \geq c \\ y \geq 0 \end{cases}$$

Using the fact that $\min(f) = -\max(-f)$, we can bring the dual into standard form, resulting in the following transformation from primal to dual: $(A, b, c) \to (-A^T, -c, -b)$. If follows that taking the dual of the dual yields the primal again.

Thus, variables in the primal correspond to constraints in the dual, and vice versa. By observing what happens with sign-unconstrained variables in the transformation to standard form and subsequent dualization, one can see that the dual constraint corresponding to a sign-unconstrained primal variable is an equality rather than an inequality. Conversely, the dual variable corresponding to a primal equality constraint is unconstrained in sign.

We set up the dual program in such a way that the objective value at any feasible point is an upper bound for the value of the primal objective function at any feasible point of the primal. This is known as weak duality. We now formally prove it, and analyze necessary and sufficient conditions for equality.

Theorem 1 (Weak Duality Theorem)**.** *For every feasible point x of the primal and feasible point y of the dual, we have that $c^T x \leq b^T y$. Moreover, equality holds iff*

$$(\forall j)\ x_j \cdot (A^T y - c)_j = 0 \quad \wedge \quad (\forall i)\ y_i \cdot (Ax - b)_i = 0. \tag{6}$$

Conditions (6) are referred to as the *complementary slackness* conditions. They express that for every dual constraint that has some slack (i.e., for which equality does not hold), the corresponding primal variable has no slack (i.e., is zero), and the same holds when we swap the words "primal" and "dual".

Proof. Since y is a feasible solution of the dual, we have that $c^T \leq y^T A$. As $x \geq 0$, it follows that $c^T x \leq (y^T A)x$, and equality holds iff the left condition in (6) holds.

Since x is a feasible solution of the primal, we have that $Ax \leq b$. As $y \geq 0$, it follows that $y^T(Ax) \leq y^T b$, and equality holds iff the right condition in (6) holds.

The result follows from the equality $(y^T A)x = y^T(Ax)$. □

Theorem 1 implies the following weak duality of linear programming.

Corollary 1. *If the primal and dual are feasible and bounded with optimum values $MaxPrimal$ and $MinDual$, respectively, then*

$$MaxPrimal \leq MinDual.$$

In fact, equality holds, as we will show next.

5 Strong Duality

A somewhat more general result holds.

Theorem 2 (Strong Duality Theorem)**.** *If the primal is feasible and bounded, then there exists a primal feasible point x and a dual feasible point y such that $c^T x = b^t y$.*

We do not provide a complete formal proof, but give a sketch that captures the main idea. The sketch is complete modulo the following intuitive claim, known as Farkas' Lemma, which is a special case of the KarushKuhnTucker criterion for having a local optimum under constraints.

Lemma 1 (Farkas' Lemma)**.** *A feasible point x of the primal is optimal iff c is in the cone spanned by the outward normals on the binding constraints at x.*

The cone spanned by a collection of vectors are all linear combinations with non-negative coefficients. The outwards normal on a constraint are the normals in the direction of violating the constraint. For a linear inequality constraint of the form $a^T x \leq b$, the outward normal is given by a.

For the intuition behind Farkas' Lemma, consider Figure 2. Let x denote the vertex O in the figure. The binding constraints are L_1 and L_2. The corresponding outer normals are N_1 and N_2. If c falls within the cone spanned by N_1 and N_2 then O is optimal because moving from O to any other point in the feasible region means moving towards (or perpendicular to) the inward normals of the binding constraints at O, and yields a non-positive contribution overall to the inner product with c, which we seek to maximize. Conversely, if c does not fall within that cone, then there is an inward direction along which the inner product with c increases.

We are now ready to prove Theorem 2.

Proof. Since the primal is feasible and bounded, there exists an optimal feasible point. Let x be such a point. We construct a feasible dual point y such that x and y satisfy the complementary slackness conditions from Theorem 1, implying that $c^T x = b^T y$.

We construct y using Farkas' Lemma. The lemma tells us that we can write c as

$$c = \sum_{i:A_{i,\cdot}x=b_i} y_i^* (A_{i,\cdot})^T - \sum_{j:x_j=0} z_j^* e_j, \tag{7}$$

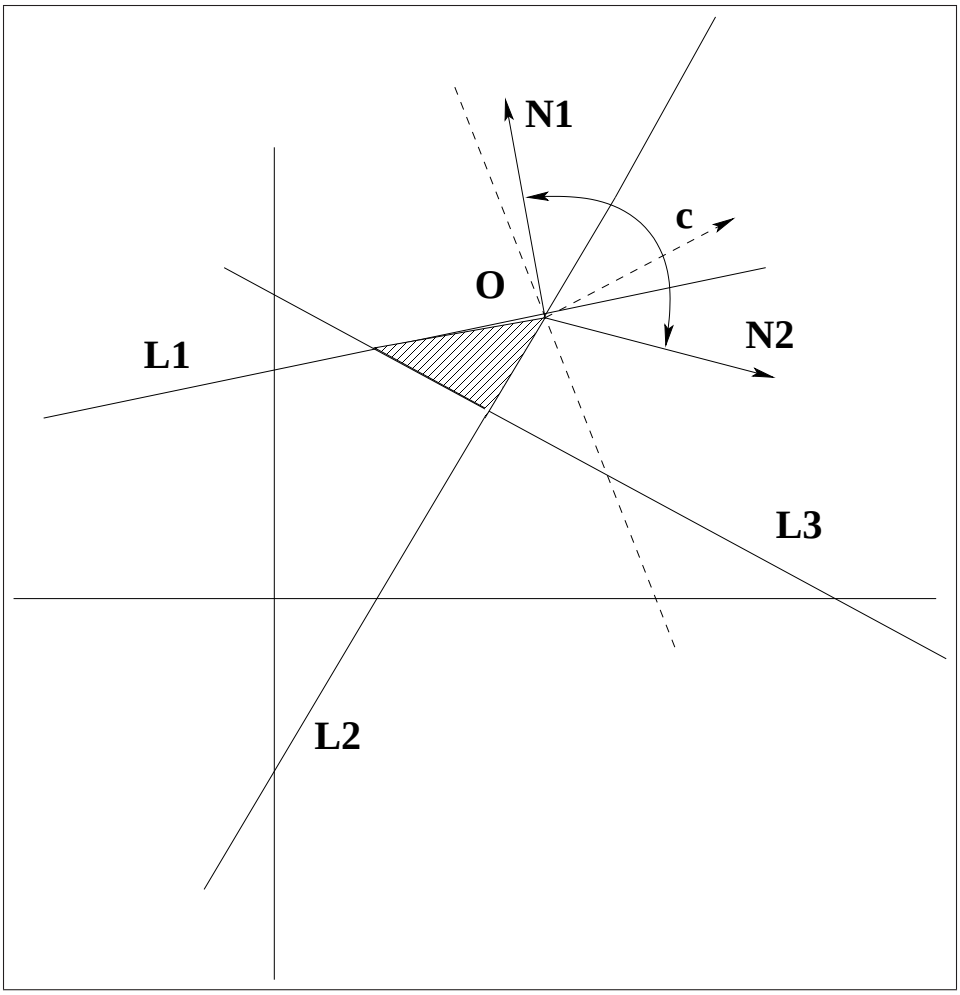

Figure 2: Outward Normals at the optimal vertex and c

where each y_i^* and z_j^* is non-negative, and e_j denotes the unit vector in the direction of x_j. The first term comes from the contribution of the outward normals on the binding constraints in $Ax \leq b$, and the second one comes from the binding non-negativity constraints $x \geq 0$. For the latter, we need to consider positive contributions in the direction of the outward normals, which are the $-e_j$s.

We define $y_i = y_i^*$ if $A_{i,}.x = b_i$, and $y_i = 0$ otherwise. By construction, we have that $y \geq 0$ and

$$c = \sum_{i:A_{i,}.x=b_i} y_i^*(A_{i,\cdot})^T - \sum_{j:x_j=0} z_j^* e_j = A^T y - \sum_{j:x_j=0} z_j^* e_j \leq A^T y,$$

so y is a feasible point for the dual.

Note that by (7), the only way there can be some slack in a dual constraint, say $c_j < (A^T y)_j = \sum_{i:A_{i,}.x=b_i} y_i^* A_{i,j}$, is if $z_j^* > 0$, which can only happen when $x_j = 0$. Thus, the left complementary slackness condition (6) holds. The right complementary slackness condition holds by the construction of y out of y^*. Theorem 1 then implies that $c^T x = b^T y$. □

Theorem 2 immediately yields the following strengthening to Corollary 1:

Corollary 2. *If the primal and dual are feasible and bounded with optimum values $MaxPrimal$ and $MinDual$, respectively, then*

$$MaxPrimal = MinDual.$$

Morever, we can say the following about the cases that can arise.

Corollary 3. *One of the following holds:*

1. *Both primal and dual are feasible and bounded.*
2. *The primal is unbounded and the dual is infeasible.*
3. *The primal is infeasible and the dual is unbounded.*
4. *Both primal and dual are infeasible.*

Proof. Theorem 1 implies that whenever the primal is unbounded, the dual is infeasible, and whenever the dual is unbounded, then the primal is infeasible. Thus, the only claim that remains to argue is that if the primal is feasible and bounded then so is the dual, and vice versa. This follows from Theorem 2. □

We leave it as an exercise to show that each of the four cases in Corollary 3 can actually occur.

6 Duality Examples

We now show that the max-flow min-cut theorem is an instantiation of the strong duality of linear programming.

We first compute the dual of the linear program from Section 2.1 for max-flow. We introduce the dual variable y_v for $v \in V \setminus \{s, t\}$ for the conservation constraints, and the dual variable z_e for $e \in E$ for the capacity constraints. We obtain:

$$\min \sum_{e \in E} c_e z_e \qquad s.t. \quad (\forall\, e = (u,v) \in E) \begin{cases} y_v - y_u + z_e \geq 0 & \text{if } u \neq s, v \neq t \\ y_v + z_e \geq 1 & \text{if } u = s, v \neq t \\ -y_u + z_e \geq 0 & \text{if } u \neq s, v = t \\ z_e \geq 1 & \text{if } u = s, v = t \\ z_e \geq 0 & \end{cases}$$

If we add two new variables, y_s and y_t, with constraints that $y_s = 1$ and $y_t = 0$, we can rewrite the dual as follows:

$$\min \sum_{e \in E} c_e z_e \qquad s.t. \quad (\forall\, e = (u,v) \in E) \begin{cases} z_e \geq y_u - y_v \\ z_e \geq 0 \\ y_s = 1 \\ y_t = 0 \end{cases}$$

We claim that this is exactly the min-cut problem. Let us denote the optimum value of the dual by OPT, and the optimum value of the min-cut problem by Min-Cut.

Claim 1. *$OPT \leq$ Min-Cut.*

Proof. Given any cut (S, T), we set z_e to be 1 if $e \in S \times T$, and zero otherwise. Also, y_v is set to 1 if v is in S and zero otherwise. Then all the constraints of the dual are satisfied, and the value of the objective equals $c(S, T)$. The claim follows. □

The construction in the proof of the claim can be reversed provided the feasible point of the dual only uses the values 0 and 1. To handle arbitrary feasible points y, intuitively, we treat "large" y_is like the 1-values above, and "small" ones as 0-values in the above construction, and argue the existence of a threshold θ between the two for which the capacity of the resulting cut can be upper bounded by the value of the dual objective at the given y. We use the probabilistic method to establish the existence of a good threshold.

Claim 2. *Min-Cut* $\leq$ *OPT.*

Proof. Consider any feasible solution y, z. Pick $\theta \in (0, 1]$ uniformly at random. Define

$$S = \{v \in V | y_v \geq \theta\}.$$

Note that this is a valid cut: a partition of the vertices into (S, T) such that $s \in S$ (because $y_s = 1 \leq \theta$) and $t \in T$ (because $y_t = 0 < \theta$). Now, for any fixed edge $e \in E$ we have

$$\Pr[e \in S \times T] = \Pr(\theta \in (y_v, y_u]) = \max(y_u - y_v, 0) \leq z_e.$$

Hence,

$$\mathbb{E}[c(S, T)] = \sum_{e \in E} c_e \Pr[e \in S \times T] \leq \sum_{e \in E} c_e z_e.$$

Thus, there must be a cut (S, T) of capacity less that or equal to $\sum_{e \in E} c_e z_e$. The claims follows. □

We leave it as an exercise to figure out a natural interpretation of the duals of the linear programs we set up for minimum cost flow, and for the shortest path problem.