12 Introduction to LP-Duality

A large fraction of the theory of approximation algorithms, as we know it today, is built around linear programming (LP). In Section 12.1 we will review some key concepts from this theory. In Section 12.2 we will show how the LP-duality theorem gives rise to min-max relations which have far-reaching algorithmic significance. Finally, in Section 12.3 we introduce the two fundamental algorithm design techniques of rounding and the primal–dual schema, as well as the method of dual fitting, which yield all the algorithms of Part II of this book.

12.1 The LP-duality theorem

Linear programming is the problem of optimizing (i.e., minimizing or maximizing) a linear function subject to linear inequality constraints. The function being optimized is called the *objective function*. Perhaps the most interesting fact about this problem from our perspective is that it is well-characterized (see definition in Section 1.2). Let us illustrate this through a simple example.

$$\begin{aligned} \text{minimize} \quad & 7x_1 + x_2 + 5x_3 \\ \text{subject to} \quad & x_1 - x_2 + 3x_3 \geq 10 \\ & 5x_1 + 2x_2 - x_3 \geq 6 \\ & x_1, x_2, x_3 \geq 0 \end{aligned}$$

Notice that in this example all constraints are of the kind "$\geq$" and all variables are constrained to be nonnegative. This is the standard form of a minimization linear program; a simple transformation enables one to write any minimization linear program in this manner. The reason for choosing this form will become clear shortly.

Any solution, i.e., a setting for the variables in this linear program, that satisfies all the constraints is said to be a *feasible solution*. Let z^* denote the optimum value of this linear program. Let us consider the question, "Is z^* at most α?" where α is a given rational number. For instance, let us ask whether $z^* \leq 30$. A Yes certificate for this question is simply a feasible solution whose

objective function value is at most 30. For example, $\boldsymbol{x} = (2, 1, 3)$ constitutes such a certificate since it satisfies the two constraints of the problem, and the objective function value for this solution is $7 \cdot 2 + 1 + 5 \cdot 3 = 30$. Thus, any Yes certificate to this question provides an upper bound on $\boldsymbol{z}^*$.

How do we provide a No certificate for such a question? In other words, how do we place a good lower bound on $\boldsymbol{z}^*$? In our example, one such bound is given by the first constraint: since the x_i's are restricted to be nonnegative, term-by-term comparison of coefficients shows that $7x_1 + x_2 + 5x_3 \geq x_1 - x_2 + 3x_3$. Since the right-hand side of the first constraint is 10, the objective function is at least 10 for any feasible solution. A better lower bound can be obtained by taking the sum of the two constraints: for any feasible solution $\boldsymbol{x}$,

$$7x_1 + x_2 + 5x_3 \geq (x_1 - x_2 + 3x_3) + (5x_1 + 2x_2 - x_3) \geq 16.$$

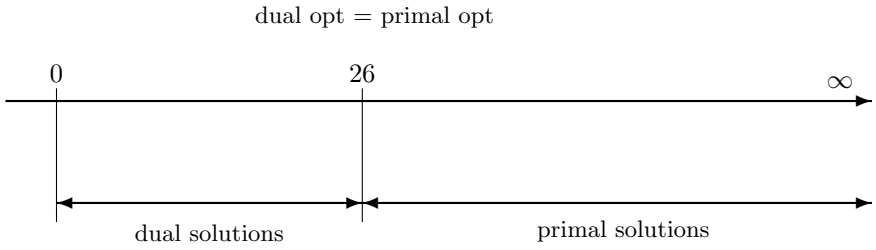
The idea behind this process of placing a lower bound is that we are finding suitable nonnegative multipliers for the constraints so that when we take their sum, the coefficient of each x_i in the sum is dominated by the coefficient in the objective function. Now, the right-hand side of this sum is a lower bound on $\boldsymbol{z}^*$ since any feasible solution has a nonnegative setting for each x_i. Notice the importance of ensuring that the multipliers are nonnegative: they do not reverse the direction of the constraint inequality.

Clearly, the rest of the game lies in choosing the multipliers in such a way that the right-hand side of the sum is as large as possible. Interestingly enough, the problem of finding the best such lower bound can be formulated as a linear program:

$$\begin{array}{lrcl}
\text{maximize} & 10y_1 + 6y_2 & & \\
\text{subject to} & y_1 + 5y_2 & \leq & 7 \\
& -y_1 + 2y_2 & \leq & 1 \\
& 3y_1 - y_2 & \leq & 5 \\
& y_1, y_2 \geq 0 & &
\end{array}$$

Here y_1 and y_2 were chosen to be the nonnegative multipliers for the first and the second constraint, respectively. Let us call the first linear program the *primal program* and the second the *dual program*. There is a systematic way of obtaining the dual of any linear program; one is a minimization problem and the other is a maximization problem. Further, the dual of the dual is the primal program itself (Exercise 12.1). By construction, every feasible solution to the dual program gives a lower bound on the optimum value of the primal. Observe that the reverse also holds. Every feasible solution to the primal program gives an upper bound on the optimal value of the dual. Therefore, if we can find feasible solutions for the dual and the primal with

matching objective function values, then both solutions must be optimal. In our example, $\boldsymbol{x} = (7/4, 0, 11/4)$ and $\boldsymbol{y} = (2, 1)$ both achieve objective function values of 26, and thus both are optimal solutions (see figure below). The reader may wonder whether our example was ingeniously constructed to make this happen. Surprisingly enough, this is not an exception, but the rule! This is the central theorem of linear programming: the *LP-duality theorem.*



In order to state this theorem formally, let us consider the following minimization problem, written in standard form, as the primal program; equivalently, we could have started with a maximization problem as the primal program.

$$\begin{aligned} \text{minimize} \quad & \sum_{j=1}^{n} c_j x_j & & (12.1) \\ \text{subject to} \quad & \sum_{j=1}^{n} a_{ij} x_j \geq b_i, & i = 1, \ldots, m \\ & x_j \geq 0, & j = 1, \ldots, n \end{aligned}$$

where a_{ij}, b_i, and c_j are given rational numbers.
Introducing variables y_i for the ith inequality, we get the dual program:

$$\begin{aligned} \text{maximize} \quad & \sum_{i=1}^{m} b_i y_i & & (12.2) \\ \text{subject to} \quad & \sum_{i=1}^{m} a_{ij} y_i \leq c_j, & j = 1, \ldots, n \\ & y_i \geq 0, & i = 1, \ldots, m \end{aligned}$$

Theorem 12.1 (LP-duality theorem) *The primal program has finite optimum iff its dual has finite optimum. Moreover, if* $\boldsymbol{x}^* = (x_1^*, \ldots, x_n^*)$ *and*

$\boldsymbol{y}^* = (y_1^*, \ldots, y_m^*)$ are optimal solutions for the primal and dual programs, respectively, then

$$\sum_{j=1}^{n} c_j x_j^* = \sum_{i=1}^{m} b_i y_i^*.$$

Notice that the LP-duality theorem is really a min–max relation, since one program is a minimization problem and the other is a maximization problem. A corollary of this theorem is that the linear programming problem is well-characterized. Feasible solutions to the primal (dual) provide Yes (No) certificates to the question, "Is the optimum value less than or equal to α?" Thus, as a corollary of this theorem we get that linear programming is in $\mathbf{NP} \cap$ co-$\mathbf{NP}$.

Going back to our example, by construction, any feasible solution to the dual program gives a lower bound on the optimal value of the primal. In fact, it also gives a lower bound on the objective function value achieved by any feasible solution to the primal. This is the easy half of the LP-duality theorem, sometimes called the *weak duality theorem*. We give a formal proof of this theorem, since some steps in the proof will lead to the next important fact. The design of several exact algorithms have their basis in the LP-duality theorem. In contrast, in approximation algorithms, typically the weak duality theorem suffices.

Theorem 12.2 (Weak duality theorem) *If $\boldsymbol{x} = (x_1, \ldots, x_n)$ and $\boldsymbol{y} = (y_1, \ldots, y_m)$ are feasible solutions for the primal and dual program, respectively, then*

$$\sum_{j=1}^{n} c_j x_j \geq \sum_{i=1}^{m} b_i y_i. \tag{12.3}$$

Proof: Since $\boldsymbol{y}$ is dual feasible and x_j's are nonnegative,

$$\sum_{j=1}^{n} c_j x_j \geq \sum_{j=1}^{n} \left(\sum_{i=1}^{m} a_{ij} y_i \right) x_j. \tag{12.4}$$

Similarly, since $\boldsymbol{x}$ is primal feasible and y_i's are nonnegative,

$$\sum_{i=1}^{m} \left(\sum_{j=1}^{n} a_{ij} x_j \right) y_i \geq \sum_{i=1}^{m} b_i y_i. \tag{12.5}$$

The theorem follows by observing that

$$\sum_{j=1}^{n}\left(\sum_{i=1}^{m} a_{ij}y_i\right)x_j = \sum_{i=1}^{m}\left(\sum_{j=1}^{n} a_{ij}x_j\right)y_i.$$

□

By the LP-duality theorem, $\boldsymbol{x}$ and $\boldsymbol{y}$ are both optimal solutions iff (12.3) holds with equality. Clearly, this happens iff both (12.4) and (12.5) hold with equality. Hence, we get the following result about the structure of optimal solutions:

Theorem 12.3 (Complementary slackness conditions) *Let $\boldsymbol{x}$ and $\boldsymbol{y}$ be primal and dual feasible solutions, respectively. Then, $\boldsymbol{x}$ and $\boldsymbol{y}$ are both optimal iff all of the following conditions are satisfied:*

Primal complementary slackness conditions
For each $1 \leq j \leq n$: either $x_j = 0$ or $\sum_{i=1}^{m} a_{ij}y_i = c_j$; and
Dual complementary slackness conditions
For each $1 \leq i \leq m$: either $y_i = 0$ or $\sum_{j=1}^{n} a_{ij}x_j = b_i$.

The complementary slackness conditions play a vital role in the design of efficient algorithms, both exact and approximation; see Chapter 15 for details. (For a better appreciation of their importance, we recommend that the reader study algorithms for the weighted matching problem, see Section 12.5.)

12.2 Min–max relations and LP-duality

In order to appreciate the role of LP-duality theory in approximation algorithms, it is important to first understand its role in exact algorithms. To do so, we will review some of these ideas in the context of the max-flow min-cut theorem. In particular, we will show how this and other min–max relations follow from the LP-duality theorem. Some of the ideas on cuts and flows developed here will also be used in the study of multicommodity flow in Chapters 18, 20, and 21.

The problem of computing a maximum flow in a network is: given a directed[1] graph, $G = (V, E)$ with two distinguished nodes, *source* s and *sink* t, and positive arc capacities, $c : E \to \mathbf{R}^+$, find the maximum amount of flow that can be sent from s to t subject to

1. *capacity constraint:* for each arc e, the flow sent through e is bounded by its capacity, and

[1] The maximum flow problem in undirected graphs reduces to that in directed graphs: replace each edge (u, v) by two directed edges, $(u \to v)$ and $(v \to u)$, each of the same capacity as (u, v).

2. *flow conservation:* at each node v, other than s and t, the total flow into v should equal the total flow out of v.

An *s–t cut* is defined by a partition of the nodes into two sets X and $\overline{X}$ so that $s \in X$ and $t \in \overline{X}$, and consists of the set of arcs going from X to $\overline{X}$. The *capacity* of this cut, $c(X, \overline{X})$, is defined to be the sum of capacities of these arcs. Because of the capacity constraints on flow, the capacity of any s–t cut is an upper bound on any feasible flow. Thus, if the capacity of an s–t cut, say $(X, \overline{X})$, equals the value of a feasible flow, then $(X, \overline{X})$ must be a minimum s–t cut and the flow must be a maximum flow in G. The max-flow min-cut theorem proves that it is always possible to find a flow and an s–t cut so that equality holds.

Let us formulate the maximum flow problem as a linear program. First, introduce a fictitious arc of infinite capacity from t to s, thus converting the flow to a circulation; the objective now is to maximize the flow on this arc, denoted by f_{ts}. The advantage of making this modification is that we can now require flow conservation at s and t as well. If f_{ij} denotes the amount of flow sent through arc $(i, j) \in E$, we can formulate the maximum flow problem as follows:

$$\begin{aligned}
\text{maximize} \quad & f_{ts} \\
\text{subject to} \quad & f_{ij} \leq c_{ij}, && (i,j) \in E \\
& \sum_{j:\ (j,i)\in E} f_{ji} \ - \sum_{j:\ (i,j)\in E} f_{ij} \leq 0, && i \in V \\
& f_{ij} \geq 0, && (i,j) \in E
\end{aligned}$$

The second set of inequalities say that for each node i, the total flow into i is at most the total flow out of i. Notice that if this inequality holds at each node, then in fact it must be satisfied with equality at each node, thereby implying flow conservation at each node (this is so because a deficit in flow balance at one node implies a surplus at some other node). With this trick, we get a linear program in standard form.

To obtain the dual program we introduce variables d_{ij} and p_i corresponding to the two types of inequalities in the primal. We will view these variables as distance labels on arcs and potentials on nodes, respectively. The dual program is:

$$\begin{aligned}
\text{minimize} \quad & \sum_{(i,j)\in E} c_{ij} d_{ij} && (12.6) \\
\text{subject to} \quad & d_{ij} - p_i + p_j \geq 0, && (i,j) \in E \\
& p_s - p_t \geq 1 \\
& d_{ij} \geq 0, && (i,j) \in E
\end{aligned}$$

$$p_i \geq 0, \qquad i \in V \tag{12.7}$$

For developing an intuitive understanding of the dual program, it will be best to first transform it into an integer program that seeks 0/1 solutions to the variables:

$$\begin{aligned}
\text{minimize} \quad & \sum_{(i,j)\in E} c_{ij} d_{ij} \\
\text{subject to} \quad & d_{ij} - p_i + p_j \geq 0, \qquad (i,j) \in E \\
& p_s - p_t \geq 1 \\
& d_{ij} \in \{0,1\}, \qquad (i,j) \in E \\
& p_i \in \{0,1\}, \qquad i \in V
\end{aligned}$$

Let $(\boldsymbol{d}^*, \boldsymbol{p}^*)$ be an optimal solution to this integer program. The only way to satisfy the inequality $p_s^* - p_t^* \geq 1$ with a 0/1 substitution is to set $p_s^* = 1$ and $p_t^* = 0$. This solution naturally defines an s–t cut $(X, \overline{X})$, where X is the set of potential 1 nodes, and $\overline{X}$ the set of potential 0 nodes. Consider an arc (i,j) with $i \in X$ and $j \in \overline{X}$. Since $p_i^* = 1$ and $p_j^* = 0$, by the first constraint, $d_{ij}^* \geq 1$. But since we have a 0/1 solution, $d_{ij}^* = 1$. The distance label for each of the remaining arcs can be set to either 0 or 1 without violating the first constraint; however, in order to minimize the objective function value, it must be set to 0. The objective function value must thus be equal to the capacity of the cut $(X, \overline{X})$, and $(X, \overline{X})$ must be a minimum s–t cut.

Thus, the previous integer program is a formulation of the minimum s–t cut problem! What about the dual program? The dual program can be viewed as a relaxation of the integer program where the integrality constraint on the variables is dropped. This leads to the constraints $1 \geq d_{ij} \geq 0$ for $(i,j) \in E$ and $1 \geq p_i \geq 0$ for $i \in V$. Next, we notice that the upper bound constraints on the variables are redundant; their omission cannot give a better solution. Dropping these constraints gives the dual program in the form given above. We will say that this program is the *LP-relaxation* of the integer program.

Consider an s–t cut C. Set C has the property that any path from s to t in G contains at least one edge of C. Using this observation, we can interpret any feasible solution to the dual program as a *fractional s–t cut*: the distance labels it assigns to arcs satisfy the property that on any path from s to t the distance labels add up to at least 1. To see this, consider an s–t path $(s = v_0, v_1, \ldots, v_k = t)$. Now, the sum of the potential differences on the endpoints of arcs on this path is

$$\sum_{i=0}^{k-1} (p_i - p_{i+1}) = p_s - p_t.$$

By the first constraint, the sum of the distance labels on the arcs must add up to at least $p_s - p_t$, which is ≥ 1. Let us define the *capacity* of this fractional s–t cut to be the dual objective function value achieved by it.

In principle, the best fractional s–t cut could have lower capacity than the best integral cut. Surprisingly enough, this does not happen. Consider the polyhedron defining the set of feasible solutions to the dual program. Let us call a feasible solution an *extreme point solution* if it is a vertex of this polyhedron, i.e., it cannot be expressed as a convex combination of two feasible solutions. From linear programming theory we know that for any objective function, i.e., assignment of capacities to the arcs of G, there is an extreme point solution that is optimal (for this discussion let us assume that for the given objective function, an optimal solution exists). Now, it can be proven that each extreme point solution of the polyhedron is integral, with each coordinate being 0 or 1 (see Exercise 12.6). Thus, the dual program always has an integral optimal solution.

By the LP-duality theorem maximum flow in G must equal capacity of a minimum fractional s–t cut. But since the latter equals the capacity of a minimum s–t cut, we get the max-flow min-cut theorem.

The max-flow min-cut theorem is therefore a special case of the LP-duality theorem; it holds because the dual polyhedron has integral vertices. In fact, most min–max relations in combinatorial optimization hold for a similar reason.

Finally, let us illustrate the usefulness of complementary slackness conditions by utilizing them to derive additional properties of optimal solutions to the flow and cut programs. Let $\boldsymbol{f}^*$ be an optimum solution to the primal LP (i.e., a maximum s–t flow). Also, let $(\boldsymbol{d}^*, \boldsymbol{p}^*)$ be an integral optimum solution to the dual LP, and let $(X, \overline{X})$ be the cut defined by $(\boldsymbol{d}^*, \boldsymbol{p}^*)$. Consider an arc (i, j) such that $i \in X$ and $j \in \overline{X}$. We have proven above that $d^*_{ij} = 1$. Since $d^*_{ij} \neq 0$, by the dual complementary slackness condition, $f^*_{ij} = c_{ij}$. Next, consider an arc (k, l) such that $k \in \overline{X}$ and $l \in X$. Since $p^*_k - p^*_l = -1$, and $d^*_{kl} \in \{0, 1\}$, the constraint $d^*_{kl} - p^*_k + p^*_l \geq 0$ must be satisfied as a strict inequality. By the primal complementary slackness condition, $f^*_{kl} = 0$. Thus, we have proven that arcs going from X to $\overline{X}$ are saturated by $\boldsymbol{f}^*$ and the reverse arcs carry no flow. (Observe that it was not essential to invoke complementary slackness conditions to prove these facts; they also follow from the fact that flow across cut $(X, \overline{X})$ equals its capacity.)

12.3 Two fundamental algorithm design techniques

We can now explain why linear programming is so useful in approximation algorithms. Many combinatorial optimization problems can be stated as integer programs. Once this is done, the linear relaxation of this program provides a natural way of lower bounding the cost of the optimal solution. As stated in Chapter 1, this is typically a key step in the design of an approximation

algorithm. As in the case of the minimum s–t cut problem, a feasible solution to the LP-relaxation can be thought of as a *fractional solution* to the original problem. However, in the case of an **NP**-hard problem, we cannot expect the polyhedron defining the set of feasible solutions to have integer vertices. Thus, our task is not to look for an optimal solution to the LP-relaxation, but rather a near-optimal integral solution.

There are two basic techniques for obtaining approximation algorithms using linear programming. The first, and more obvious, method is to solve the linear program and then convert the fractional solution obtained into an integral solution, trying to ensure that in the process the cost does not increase much. The approximation guarantee is established by comparing the cost of the integral and fractional solutions. This technique is called *LP-rounding* or simply *rounding*.

The second, less obvious and perhaps more sophisticated, method is to use the dual of the LP-relaxation in the design of the algorithm. This technique is called the *primal–dual schema*. Let us call the LP-relaxation the primal program. Under this schema, an integral solution to the primal program and a feasible solution to the dual program are constructed iteratively. Notice that any feasible solution to the dual also provides a lower bound on OPT. The approximation guarantee is established by comparing the two solutions.

Both these techniques have been used extensively to obtain algorithms for many fundamental problems. Fortunately, once again, these techniques can be illustrated in the simple setting of the set cover problem. This is done in Chapters 14 and 15. Later chapters will present ever more sophisticated use of these techniques for solving a variety of problems.

LP-duality theory has also been useful in analyzing combinatorially obtained approximation algorithms, using the *method of dual fitting*. In Chapter 13 we will give an alternative analysis of the greedy set cover algorithm, Algorithm 2.2, using this method. This method has also been used to analyze greedy algorithms for the metric uncapacitated facility location problem (see Exercise 24.12). The method seems quite basic and should find other applications as well.

12.3.1 A comparison of the techniques and the notion of integrality gap

The reader may suspect that from the viewpoint of approximation guarantee, the primal–dual schema is inferior to rounding, since an optimal solution to the primal gives a tighter lower bound than a feasible solution to the dual. It turns out that this is not so. In order to give a formal explanation, we need to introduce the crucial notion of *integrality gap of an LP-relaxation*.

Given an LP-relaxation for a minimization problem Π, let $\mathrm{OPT}_f(I)$ denote the cost of an optimal fractional solution to instance I, i.e., the objective function value of an optimal solution to the LP-relaxation. Define the *inte-*

grality gap, sometimes also called the *integrality ratio*, of the relaxation to be

$$\sup_I \frac{\mathrm{OPT}(I)}{\mathrm{OPT}_f(I)},$$

i.e., the supremum of the ratio of the optimal integral and fractional solutions. In the case of a maximization problem, the integrality gap will be defined to be the infimum of this ratio. As stated in Section 12.2, most min–max relations arise from LP-relaxations that always have integral optimal solutions. Clearly, the integrality gap of such an LP is 1. We will call such an LP-relaxation an *exact relaxation*.

If the cost of the solution found by the algorithm is compared directly with the cost of an optimal fractional solution (or a feasible dual solution), as is done in most algorithms, the best approximation factor we can hope to prove is the integrality gap of the relaxation (see Exercise 12.4). Interestingly enough, for many problems, both techniques have been successful in yielding algorithms having guarantees essentially equal to the integrality gap of the relaxation.

The main difference in performance between the two techniques lies in the running times of the algorithms produced. An LP-rounding algorithm needs to find an optimal solution to the linear programming relaxation. Since linear programming is in **P**, this can be done in polynomial time if the relaxation has polynomially many constraints. Even if the relaxation has exponentially many constraints, this may still be achievable, if a polynomial time *separation oracle* can be constructed, i.e., a polynomial time algorithm that given a point in $\mathbf{R}^n$, where n is the number of variables in the relaxation, either confirms that this point is a feasible solution (i.e., satisfies all constraints), or produces a violated constraint (see the notes in Section 12.5 for references). The running time for both possibilities is high; for the second it may be exorbitant. Let us remark that for certain problems, extreme point solutions have additional structural properties and some LP-rounding algorithms require such a solution to the linear programming relaxation. Such solutions can also be found in polynomial time.

On the other hand, the primal–dual schema leaves enough room to exploit the special combinatorial structure of individual problems and is thereby able to yield algorithms having good running times. It provides only a broad outline of the algorithm – the details have to be designed individually for specific problems. In fact, for many problems, once the algorithm has been designed using the primal–dual schema, the scaffolding of linear programming can be completely dispensed with to get a purely combinatorial algorithm.

This brings us to another advantage of the primal–dual schema – this time not objectively quantifiable. A combinatorial algorithm is more malleable than an algorithm that requires an LP-solver. Once a basic problem is solved using the primal–dual schema, one can also solve variants and generalizations

of the basic problem. Exercises in Chapters 22 and 24 illustrate this point. From a practical standpoint, a combinatorial algorithm is more useful, since it is easier to adapt it to specific applications and fine tune its performance for specific types of inputs.

12.4 Exercises

12.1 Show that the dual of the dual of a linear program is the original program itself.

12.2 Show that any minimization program can be transformed into an equivalent program in standard form, i.e., the form of LP (12.1).

12.3 Change some of the constraints of the primal program (12.1) into equalities, i.e., so they are of the form

$$\sum_{j=1}^{n} a_{ij}x_j = b_i, i \in I.$$

Show that the dual of this program involves modifying program (12.2) so that the corresponding dual variables $y_i, i \in I$ are *unconstrained*, i.e., they are not constrained to be nonnegative. Additionally, if some of the variables $x_j, j \in J$ in program (12.1) are unconstrained, then the corresponding constraints in the dual become equalities.

12.4 Is the following a theorem: An approximation algorithm designed using an LP-relaxation cannot achieve a better approximation guarantee than the integrality gap of the relaxation.
Hint: In principle it may be possible to show, using additional structural properties, that whenever an instance has a bad gap, the cost of the solution found by the algorithm is much less that αOPT, where α is the integrality gap of the relaxation. (Observe that if the instance has a bad gap, the cost of the solution found cannot be much less than αOPT_f.)

12.5 Use the max-flow min-cut theorem to derive Menger's Theorem:

Theorem 12.4 *Let $G = (V, E)$ be a directed graph with $s, t \in V$. Then, the maximum number of edge-disjoint (vertex-disjoint) s–t paths is equal to the minimum number of edges (vertices) whose removal disconnects s from t.*

12.6 Show that each extreme point solution for LP (12.6) is 0/1, and hence represents a valid cut.

Hint: An $n \times m$ matrix A is said to be *totally unimodular* if the determinant of every square submatrix of A is $1, -1$, or 0. Show, by induction, that the constraint matrix of this LP is totally unimodular. Also, use the fact that a feasible solution for a set of linear inequalities in $\mathbf{R}^n$ is an extreme point solution iff it satisfies n linearly independent inequalities with equality.

12.7 This exercise develops a proof of the König-Egerváry Theorem (Theorem 1.6). Let $G = (V, E)$ be a bipartite graph.

1. Show that the following is an exact LP-relaxation (i.e., always has an integral optimal solution) for the maximum matching problem in G.

$$\begin{aligned} \text{maximize} \quad & \sum_{e} x_e && && (12.8)\\ \text{subject to} \quad & \sum_{e:\ e \text{ incident at } v} x_e \le 1, && v \in V \\ & x_e \ge 0, && e \in E \end{aligned}$$

   Hint: Using the technique of Exercise 12.6 show that each extreme point solution for LP (12.8) is 0/1, and hence represents a valid matching.
2. Obtain the dual of this LP and show that it is an exact LP-relaxation for the problem of finding a minimum vertex cover in bipartite graph G.
3. Use the previous result to derive the König-Egerváry Theorem.

12.8 (Edmonds [68])

1. Let $G = (V, E)$ be an undirected graph, with weights w_e on edges. The following is an exact LP-relaxation for the problem of finding a maximum weight matching in G. (By $e : \ e \in S$ we mean edges e that have both endpoints in S.)

$$\begin{aligned} \text{maximize} \quad & \sum_{e} w_e x_e && && (12.9)\\ \text{subject to} \quad & \sum_{e:\ e \text{ incident at } v} x_e \le 1, && v \in V \\ & \sum_{e:\ e \in S} x_e \le \frac{|S|-1}{2}, && S \subset V, |S| \text{ odd} \\ & x_e \ge 0, && e \in E \end{aligned}$$

   Obtain the dual of this LP. If the weight function is integral, the dual is also exact. Observe that Theorem 1.7 follows from these facts.

2. Assume that $|V|$ is even. The following is an exact LP-relaxation for the minimum weight perfect matching problem in G (a matching is *perfect* if it matches all vertices). Obtain the dual of this LP. Use complementary slackness conditions to give conditions satisfied by a pair of optimal primal (integral) and dual solutions for both formulations.

$$\begin{aligned} \text{minimize} \quad & \sum_e w_e x_e & & (12.10)\\ \text{subject to} \quad & \sum_{e:\ e \text{ incident at } v} x_e = 1, & v \in V & \\ & \sum_{e:\ e \in S} x_e \le \frac{|S|-1}{2}, & S \subset V, |S| \text{ odd} & \\ & x_e \ge 0, & e \in E & \end{aligned}$$

12.9 (Edmonds [71]) Show that the following is an integer programming formulation for the minimum spanning tree (MST) problem. Assume we are given graph $G = (V, E)$, $|V| = n$, with cost function $c : E \to \mathbf{Q}^+$. For $A \subseteq E$, we will denote by $\kappa(A)$ the number of connected components in graph $G_A = (V, A)$.

$$\begin{aligned} \text{minimize} \quad & \sum_e c_e x_e & & (12.11)\\ \text{subject to} \quad & \sum_{e \in A} x_e = n - \kappa(A), & A \subset E & \\ & \sum_{e \in E} x_e = n - 1 & & \\ & x_e \in \{0, 1\}, & e \in E & \end{aligned}$$

The rest of this exercise develops a proof that the LP-relaxation of this integer program is exact for the MST problem.

1. First, it will be convenient to change the objective function of IP (12.11) to $\max \sum_e -c_e x_e$. Obtain the LP-relaxation and dual of this modified formulation.
2. Consider the primal solution produced by Kruskal's algorithm. Let $e_1, \ldots, e_m$ be the edges sorted by increasing cost, $|E| = m$. This algorithm greedily picks a maximal acyclic subgraph from this sorted list. Obtain a suitable dual feasible solution so that all complementary slackness conditions are satisfied.
   Hint: Let $A_t = \{e_1, \ldots, e_t\}$. Set $y_{A_t} = e_{t+1} - e_t$, for $1 \le t < m$, and $y_E = -c_m$, where y is the dual variable.

3. Show that $\boldsymbol{x}$ is a feasible solution to the above-stated primal program iff it is a feasible solution to the following LP. That is, prove that this is also an exact relaxation for the MST problem.

$$\begin{aligned} \text{minimize} \quad & \sum_e c_e x_e && (12.12)\\ \text{subject to} \quad & \sum_{e \in S} x_e = |S| - 1, \qquad S \subset V \\ & \sum_{e \in E} x_e = n - 1 \\ & x_e \geq 0, \qquad e \in E \end{aligned}$$

12.10 In this exercise, you will derive von Neumann's minimax theorem in game theory from the LP-duality theorem. A finite two-person zero-sum game is specified by an $m \times n$ matrix $\boldsymbol{A}$ with real entries. In each round, the row player, R, selects a row, say i; simultaneously, the column player, C, selects a column, say j. The *payoff* to R at the end of this round is a_{ij}. Thus, $|a_{ij}|$ is the amount that C pays R (R pays C) if a_{ij} is positive (a_{ij} is negative); no money is exchanged if a_{ij} is zero. *Zero-sum game* refers to the fact that the total amount of money possessed by R and C together is conserved.

The *strategy* of each player is specified by a vector whose entries are nonnegative and add up to one, giving the probabilities with which the player picks each row or column. Let R's strategy be given by m-dimensional vector $\boldsymbol{x}$, and C's strategy be given by n-dimensional vector $\boldsymbol{y}$. Then, the expected payoff to R in a round is $\boldsymbol{x}^T \boldsymbol{A}\boldsymbol{y}$. The job of each player is to pick a strategy that *guarantees* maximum possible expected winnings (equivalently, minimum possible expected losses), regardless of the strategy chosen by the other player. If R chooses strategy $\boldsymbol{x}$, he can be sure of winning only $\min_{\boldsymbol{y}} \boldsymbol{x}^T \boldsymbol{A}\boldsymbol{y}$, where the minimum is taken over all possible strategies of C. Thus, the optimal choice for R is given by $\max_{\boldsymbol{x}} \min_{\boldsymbol{y}} \boldsymbol{x}^T \boldsymbol{A}\boldsymbol{y}$. Similarly, C will minimize her losses by choosing the strategy given by $\min_{\boldsymbol{y}} \max_{\boldsymbol{x}} \boldsymbol{x}^T \boldsymbol{A}\boldsymbol{y}$. The minimax theorem states that for every matrix $\boldsymbol{A}$, $\max_{\boldsymbol{x}} \min_{\boldsymbol{y}} \boldsymbol{x}^T \boldsymbol{A}\boldsymbol{y} = \min_{\boldsymbol{y}} \max_{\boldsymbol{x}} \boldsymbol{x}^T \boldsymbol{A}\boldsymbol{y}$.

Let us say that a strategy is *pure* if it picks a single row or column, i.e., the vector corresponding to it consists of one 1 and the rest 0's. A key observation is that for any strategy $\boldsymbol{x}$ of R, $\min_{\boldsymbol{y}} \boldsymbol{x}^T \boldsymbol{A}\boldsymbol{y}$ is attained for a pure strategy of C: Suppose the minimum is attained for strategy $\boldsymbol{y}$. Consider the pure strategy corresponding to any nonzero component of $\boldsymbol{y}$. The fact that the components of $\boldsymbol{y}$ are nonnegative and add up to one leads to an easy proof that this pure strategy attains the same minimum. Thus, R's optimum strategy is given by $\max_{\boldsymbol{x}} \min_j \sum_{i=1}^m a_{ij} x_i$. The second critical observation

is that the problem of computing R's optimal strategy can be expressed as a linear program:

$$\begin{array}{lll} \text{maximize} & z & \\ \text{subject to} & z - \sum_{i=1}^{m} a_{ij} x_i \leq 0, & j = 1, \ldots, n \\ & \sum_{i=1}^{m} x_i = 1 & \\ & x_i \geq 0, & i = 1, \ldots, m \end{array}$$

Find the dual of this LP and show that it computes the optimal strategy for C. (Use the fact that for any strategy $\boldsymbol{y}$ of C, $\max_{\boldsymbol{x}} \boldsymbol{x}^T \boldsymbol{A}\boldsymbol{y}$ is attained for a pure strategy of R.) Hence, prove the minimax theorem using the LP-duality theorem.

12.5 Notes

For a good introduction to theory of linear programming, see Chvátal [49]. There are numerous other books on the topic, e.g., Dantzig [58], Karloff [160], Nemhauser and Wolsey [212], and Schrijver [237]. Linear programming has been extensively used in combinatorial optimization, see Cook, Cunningham, Pulleyblank, and Schrijver [52], Grötschel, Lovász, and Schrijver [117], Lovász [194], Lovász and Plummer [195], and Papadimitriou and Steiglitz [217]. For a good explanation of Edmonds' weighted matching algorithm, see Lovász and Plummer [195]. For algorithms for finding a solution to an LP, given a separation oracle, see Grötschel, Lovász, and Schrijver [116, 117] and Schrijver [237].

15 Set Cover via the Primal–Dual Schema

As noted in Section 12.3, the primal–dual schema is the method of choice for designing approximation algorithms since it yields combinatorial algorithms with good approximation factors and good running times. We will first present the central ideas behind this schema and then use it to design a simple f factor algorithm for set cover, where f is the frequency of the most frequent element.

The primal–dual schema has its origins in the design of exact algorithms. In that setting, this schema yielded the most efficient algorithms for some of the cornerstone problems in **P**, including matching, network flow, and shortest paths. These problems have the property that their LP-relaxations have integral optimal solutions. By Theorem 12.3 we know that optimal solutions to linear programs are characterized by fact that they satisfy all the complementary slackness conditions. In fact, the primal–dual schema for exact algorithms is driven by these conditions. Starting with initial feasible solutions to the primal and dual programs, it iteratively starts satisfying complementary slackness conditions. When they are all satisfied, both solutions must be optimal. During the iterations, the primal is always modified integrally, so that eventually we get an integral optimal solution.

Consider an LP-relaxation for an **NP**-hard problem. In general, the relaxation will not have an optimal solution that is integral. Does this rule out a complementary slackness condition driven approach? Interestingly enough, the answer is 'no'. It turns out that the algorithm can be driven by a suitable relaxation of these conditions! This is the most commonly used way of designing primal–dual based approximation algorithms – but not the only way.

15.1 Overview of the schema

Let us consider the following primal program, written in standard form.

$$\text{minimize} \quad \sum_{j=1}^{n} c_j x_j$$

$$\begin{aligned} \text{subject to} \quad & \sum_{j=1}^{n} a_{ij} x_j \geq b_i, \qquad i = 1, \ldots, m \\ & x_j \geq 0, \qquad\qquad\quad j = 1, \ldots, n \end{aligned}$$

where a_{ij}, b_i, and c_j are specified in the input. The dual program is:

$$\begin{aligned} \text{maximize} \quad & \sum_{i=1}^{m} b_i y_i \\ \text{subject to} \quad & \sum_{i=1}^{m} a_{ij} y_i \leq c_j, \qquad j = 1, \ldots, n \\ & y_i \geq 0, \qquad\qquad\quad i = 1, \ldots, m \end{aligned}$$

Most known approximation algorithms using the primal–dual schema run by ensuring one set of conditions and suitably relaxing the other. In the following description we capture both situations by relaxing both conditions. Eventually, if primal conditions are ensured, we set $\alpha = 1$, and if dual conditions are ensured, we set $\beta = 1$.

Primal complementary slackness conditions

Let $\alpha \geq 1$.

For each $1 \leq j \leq n$: either $x_j = 0$ or $c_j/\alpha \leq \sum_{i=1}^{m} a_{ij} y_i \leq c_j$.

Dual complementary slackness conditions

Let $\beta \geq 1$.

For each $1 \leq i \leq m$: either $y_i = 0$ or $b_i \leq \sum_{j=1}^{n} a_{ij} x_j \leq \beta \cdot b_i$,

Proposition 15.1 *If $\boldsymbol{x}$ and $\boldsymbol{y}$ are primal and dual feasible solutions satisfying the conditions stated above then*

$$\sum_{j=1}^{n} c_j x_j \leq \alpha \cdot \beta \cdot \sum_{i=1}^{m} b_i y_i.$$

Proof:

$$\begin{aligned} \sum_{j=1}^{n} c_j x_j &\leq \alpha \sum_{j=1}^{n} \left(\sum_{i=1}^{m} a_{ij} y_i \right) x_j = \alpha \sum_{i=1}^{m} \left(\sum_{j=1}^{n} a_{ij} x_j \right) y_i \\ &\leq \alpha\beta \sum_{i=1}^{m} b_i y_i \ . \end{aligned} \tag{15.1}$$

The first and second inequalities follow from the primal and dual conditions, respectively. The equality follows by simply changing the order of summation. □

The algorithm starts with a primal infeasible solution and a dual feasible solution; these are usually the trivial solutions $\boldsymbol{x} = \mathbf{0}$ and $\boldsymbol{y} = \mathbf{0}$. It iteratively improves the feasibility of the primal solution, and the optimality of the dual solution, ensuring that in the end a primal feasible solution is obtained and all conditions stated above, with a suitable choice of α and β, are satisfied. The primal solution is always extended integrally, thus ensuring that the final solution is integral. The improvements to the primal and the dual go hand-in-hand: the current primal solution is used to determine the improvement to the dual, and vice versa. Finally, the cost of the dual solution is used as a lower bound on OPT, and by Proposition 15.1, the approximation guarantee of the algorithm is $\alpha\beta$.

15.2 Primal–dual schema applied to set cover

Let us obtain a factor f algorithm for the set cover problem using the primal–dual schema. For this algorithm, we will choose $\alpha = 1$ and $\beta = f$. We will work with the primal and dual pair of LP's given in (13.2) and (13.3), respectively. The complementary slackness conditions are:

Primal conditions

$$\forall S \in \mathcal{S} : x_S \neq 0 \Rightarrow \sum_{e:\ e \in S} y_e = c(S).$$

Set S will be said to be *tight* if $\sum_{e:\ e \in S} y_e = c(S)$. Since we will increment the primal variables integrally, we can state the conditions as: *Pick only tight sets in the cover.*
Clearly, in order to maintain dual feasibility, we are not allowed to overpack any set.

Dual conditions

$$\forall e : y_e \neq 0 \Rightarrow \sum_{S:\ e \in S} x_S \leq f$$

Since we will find a 0/1 solution for $\boldsymbol{x}$, these conditions are equivalent to: *Each element having a nonzero dual value can be covered at most f times.* Since each element is in at most f sets, this condition is trivially satisfied for all elements.

The two sets of conditions naturally suggest the following algorithm:

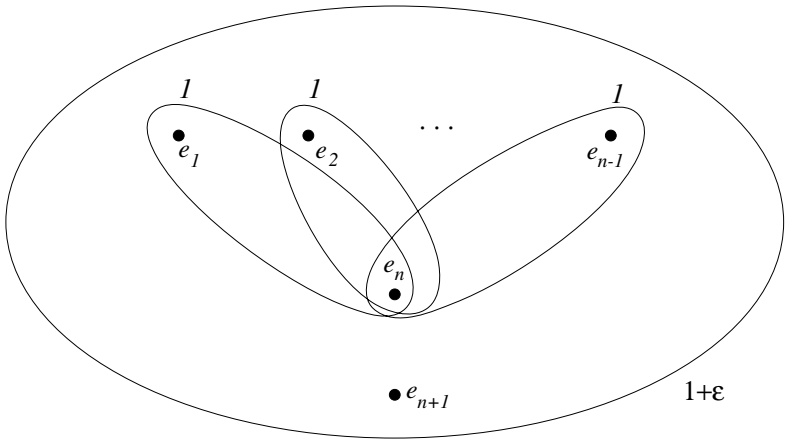
Algorithm 15.2 (Set cover – factor f)

1. **Initialization:** $\boldsymbol{x} \leftarrow \mathbf{0}$; $\boldsymbol{y} \leftarrow \mathbf{0}$
2. Until all elements are covered, do:
   Pick an uncovered element, say e, and raise y_e until some set goes tight.
   Pick all tight sets in the cover and update $\boldsymbol{x}$.
   Declare all the elements occurring in these sets as "covered".
3. Output the set cover $\boldsymbol{x}$.

Theorem 15.3 *Algorithm 15.2 achieves an approximation factor of f.*

Proof: Clearly there will be no uncovered elements and no overpacked sets at the end of the algorithm. Thus, the primal and dual solutions will both be feasible. Since they satisfy the relaxed complementary slackness conditions with $\alpha = f$, by Proposition 15.1 the approximation factor is f. □

Example 15.4 A tight example for this algorithm is provided by the following set system:



Here, $\mathcal{S}$ consists of $n-1$ sets of cost 1, $\{e_1, e_n\}, \ldots, \{e_{n-1}, e_n\}$, and one set of cost $1+\varepsilon$, $\{e_1, \ldots, e_{n+1}\}$, for a small $\varepsilon > 0$. Since e_n appears in all n sets, this set system has $f = n$.

Suppose the algorithm raises y_{e_n} in the first iteration. When y_{e_n} is raised to 1, all sets $\{e_i, e_n\}$, $i = 1, \ldots, n-1$ go tight. They are all picked in the cover, thus covering the elements $e_1, \ldots, e_n$. In the second iteration, $y_{e_{n+1}}$ is raised to ε and the set $\{e_1, \ldots, e_{n+1}\}$ goes tight. The resulting set cover has a cost of $n + \varepsilon$, whereas the optimum cover has cost $1 + \varepsilon$. □

15.3 Exercises

15.1 How is the algorithm given in Exercise 2.11 for the weighted vertex cover problem related to Algorithm 15.2 for the case $f = 2$?

15.2 Remove the scaffolding of linear programming from Algorithm 15.2 to obtain a purely combinatorial factor f algorithm for set cover.
Hint: See the algorithm in Exercise 2.11.

15.3 Let k be a fixed constant, and consider instances of set cover whose maximum frequency, f, is bounded by k. Algorithm 15.2 shows that the integrality gap of LP (13.2) is upper bounded by k for these instances. Provide examples to show that this bound is essentially tight.
Hint: Consider a regular hypergraph, G, on n vertices which has a hyperedge corresponding to each choice of k of the n vertices. Construct the set system as follows. It has an element corresponding to each hyperedge and a set corresponding to each vertex, with incidence defining inclusion.

15.4 The following LP-relaxation is exact for the maximum weight matching problem (see definition in Exercise 12.8) in bipartite graphs but not in general graphs. Give a primal–dual algorithm, relaxing complementary slackness conditions appropriately, to show that the integrality gap of this LP is $\geq 1/2$. What is the best upper bound you can place on the integrality gap?

$$\begin{aligned} \text{maximize} \quad & \sum_e w_e x_e & & (15.2)\\ \text{subject to} \quad & \sum_{e:\ e \text{ incident at } v} x_e \leq 1, & v \in V \\ & x_e \geq 0, & e \in E \end{aligned}$$

15.5 (Chudak, Goemans, Hochbaum, and Williamson [46]) Interpret the layering-based algorithms obtained for set cover and feedback vertex set problems in Chapters 2 and 6 as primal–dual schema based algorithms. How are the complementary slackness conditions being relaxed?

15.4 Notes

Kuhn [179] gave the first primal–dual algorithm – for the weighted bipartite matching problem – however, he used the name "Hungarian Method" to describe his algorithm. Dantzig, Ford, and Fulkerson [60] used this method

for giving another means of solving linear programs and called it the primal–dual method. Although the schema was not very successful for solving linear programs, it soon found widespread use in combinatorial optimization.

Algorithm 15.2 is due to Bar-Yehuda and Even [20]. Although it was not originally stated as a primal–dual algorithm, in retrospect, this was the first use of the schema in approximation algorithms. The works of Agrawal, Klein, and Ravi [1] and Goemans and Williamson [105] revived the use of this schema in the latter setting, and introduced the powerful idea of growing duals in a synchronized manner (see Chapter 22). The mechanism of relaxing complementary slackness conditions was first formalized in Williamson, Goemans, Mihail, and Vazirani [258]. For further historical information, see Goemans and Williamson [107].