6 Linear Programming

In Chapters 2 (Greedy Algorithms) and 3 (Dynamic Programming), we saw two general approaches to solving optimization problems that are applicable to a variety of settings (e.g., subsequences, knapsacks, graphs). In this chapter, we study another general technique known as *linear programming*. There are entire courses devoted to linear programming; it is a deep topic with countless applications.

This chapter is a bit different from previous chapters. In Sec. 6.1, we introduce the basic idea behind linear programming and give two examples of applications. But we won't actually study any algorithms that solve linear programs; for our purposes, we can assume that we have access to an LP solver. We illustrate this idea by giving an algorithm for the vertex cover problem in Sec. 6.2. Finally, in Sec. 6.3, we show how linear programming allows us to optimally play two-player zero-sum games.

6.1 The Basics of LPs

A *linear program* (LP) is an optimization problem in which the objective is to maximize (or minimize) a linear function subject to constraints in the form of linear inequalities. For example, here is a linear program that has 2 variables and 1 constraint:[1]

$$\begin{aligned} \max\ & 3x_1 + 2x_2 \\ & x_1 + x_2 \leq 5 \\ & x_1, x_2 \geq 0 \end{aligned}$$

A *solution* to the LP is an assignment of each variable to a real number. It is *feasible* if it satisfies all of the constraints, and it is *optimal* if it is feasible and has the highest (or lowest, for minimization problems) objective value among all feasible solutions. (An LP can have multiple optimal solutions.) In the LP above, the solution $(x_1, x_2) = (1, 2)$ is feasible, $(5, 3)$ and $(-1, 4)$ are not, and $(5, 0)$ is optimal.

We often use vectors and matrices to specify linear programs. If an LP has n variables and m constraints, then we let $x \in \mathbb{R}^n$ denote the vector of variables and $c \in \mathbb{R}^n$ denote the coefficients of the objective function. Furthermore, we let $A \in \mathbb{R}^{m \times n}$ and $b \in \mathbb{R}^m$ specify the constraints. Putting it all together, an LP in *canonical form*[2] looks like this:

$$\begin{aligned} \max\ & c^\top x \\ & Ax \leq b \\ & x \geq 0 \end{aligned}$$

Note that "$x \geq 0$" means each of the n entries of x is at least 0. Similarly, "$Ax \leq b$" means each of the m entries of $Ax \in \mathbb{R}^m$ is at most the corresponding entry in b. So in the first LP, we had $x = (x_1, x_2), c = (3, 2), A = (1, 1)$, and $b = (5)$ (where x, c, b are column vectors). We now look at two well-known applications of linear programming.

[1] The constraint "$x_1, x_2 \geq 0$" is short for "$x_1 \geq 0$ and $x_2 \geq 0$", and these are not usually counted as constraints because they are so ubiquitous in LPs. However, it is not necessary for them to appear in an LP.

[2] Different sources refer to different things as canonical (or sometimes *standard*) form, but all of the forms you might find can be transformed into each other.

The Diet Problem. Suppose a store sells n food items, where each item j has cost c_j per unit. Also, there are m nutrients, and item j provides A_{ij} units of nutrient i. Our diet mandates that we eat at least b_j units of nutrient j. Our goal is to spend the least amount of money on the n food items while satisfying the m nutritional requirements.

We can model this problem as an LP as follows. Let x be an n-dimensional vector; we need to determine x_j for all $j \in \{1, \ldots, n\}$, which represents the amount of food j we buy. Let $c \in \mathbb{R}^n$ denote the vector containing the costs, let A be an $m \times n$ matrix whose (i, j)-th entry is A_{ij}, and let $b \in \mathbb{R}^m$ denote the vector containing the nutritional lower bounds. Our goal is to minimize $c^\top x$ such that $Ax \geq b$ and $x \geq 0$.

The Manufacturing Problem. Now suppose we're the ones doing the selling. Our factory is capable of producing n products, which are made from m different materials. Each unit of product j requires A_{ij} units of material i and yields c_j units of profit. We have b_i units of material i at our disposal, and our goal is to maximize the total profit.

In this case, x_j represents the amount of product j we produce. The n-dimensional vector c contains the profits, and $A \in \mathbb{R}^{m \times n}$ and $b \in \mathbb{R}^m$ encode the material constraints. Our goal is to maximize $c^\top x$ such that $Ax \leq b$ and $x \geq 0$.

Remark. The manufacturing problem LP is in canonical form, but the LP for the diet problem is not. However, if we negate the vector c then the objective becomes a maximization, and if we negative all entries of A and b then the inequality signs flip. In general, the objective can be maximize or minimize, the constraints can be inequalities or equalities, and variables are allowed to be negative.

But no matter what the LP looks like, it can be converted to an equivalent LP in canonical form using variable substitution, negating coefficients, and other small tricks; we leave the details as an exercise. For our purposes, let's assume that we can efficiently find an optimal solution of any linear program, even if it's not in canonical form.[3]

6.2 Vertex Cover

Problem statement. Let $G = (V, E)$ be an undirected graph. A *vertex cover* is a subset of vertices S such that every edge is incident to at least one vertex in S. Our goal is to find a vertex cover of G containing the fewest number of vertices.

Algorithm. Associate each vertex u with a variable $x_u \in \{0, 1\}$. (Intuitively, $x_u = 1$ corresponds to selecting u to be included in the solution, but this is not part of the algorithm.) Then the vertex cover problem is equivalent to the following *integer* LP:

$$\begin{aligned} \min \sum_{u \in V} x_u \\ x_u + x_v \geq 1 \quad \forall \{u, v\} \in E \\ x_u \in \{0, 1\} \quad \forall u \in V. \end{aligned}$$

[3]Again, there are entire courses and books devoted to linear programming and its algorithms; if you are interested in learning about them, I recommend looking at other resources online.

This problem is almost an LP; the only problem is the $x_u \in \{0,1\}$ constraint. Let's replace this constraint with $x_u \geq 0$ and solve the resulting LP to obtain an optimal solution $x^* \in \mathbb{R}^n$. We return $S = \{u : x_u^* \geq 1/2\}$; let $\mathsf{OPT} \subseteq V$ denote an optimal solution.

Theorem 6.1. *The set S is a vertex cover and satisfies $|S| \leq 2 \cdot |\mathsf{OPT}|$.*

Before proving the theorem, we first prove that the optimal value of the linear program is at most the number of vertices in an optimal vertex cover.

Lemma 6.2. *The LP solution x^* and OPT satisfy $\sum_{u \in V} x_u^* \leq |\mathsf{OPT}|$.*

Proof. The integer linear program above (i.e., with the $x_u \in \{0,1\}$ constraints) is equivalent to the vertex cover problem, so its optimal value is equal to $|\mathsf{OPT}|$. By relaxing the constraints $x_u \in \{0,1\}$ to $x_u \geq 0$, the optimal value of the program can only improve, i.e., decrease. Thus, the optimal value of the LP is at most $|\mathsf{OPT}|$, as desired. □

Proof of Theorem 6.1. To show that S is a vertex cover, we must show that every edge $e = \{u, v\}$ is covered. Since x^* is a feasible solution to the linear program, we must have $x_u^* + x_v^* \geq 1$. Thus, at least one the values in $\{x_u^*, x_v^*\}$ must be at least $1/2$, so that corresponding vertex gets included in S, which means it covers the edge e.

Now we bound the size of S. Since every $u \in S$ satisfies $x_u^* \geq 1/2$, we have

$$|S| = \sum_{u \in S} 1 \leq \sum_{u \in S} 2x_u^* \leq 2 \cdot \sum_{u \in V} x_u^* \leq 2 \cdot |\mathsf{OPT}|,$$

where the final inequality follows from Lemma 6.2. □

6.3 Zero-sum Games

Problem statement. Alice and Bob are playing a game; Alice has m options denoted by $[m] = \{1, \ldots, m\}$ while Bob has n options denoted by $[n] = \{1, \ldots, n\}$. If Alice chooses option i and Bob chooses option j, then Alice receives A_{ij} dollars from Bob (or she gives him this amount, if A_{ij} is negative). Of course, Alice wants to maximize her payoff, and Bob wants to minimize it. Our goal is to determine the maximum payoff that Alice can *guarantee* for herself, and her corresponding strategy.[4]

Algorithm. A *pure* strategy is defined by always selecting one option, while a *mixed* strategy is one that specifies a probability distribution over the options. If Alice chooses a pure strategy of option i, then the payoff she can guarantee is $\min_j A_{ij}$, since Bob can pick the strategy j that achieves this minimum value.

Now let's consider Alice's guaranteed payoff if she chooses a mixed strategy, that is, suppose she chooses option i with probability x_i. The key is that Alice can solve for her

[4]This game is known as zero-sum because the sum of Alice's payoff and Bob's payoff is always zero. Alice is happy when her payoff is positive, while Bob is happy when her payoff is negative. There is no result in which both of them are happy, or both of them are sad.

guaranteed payoff value by modeling this problem as a linear program:

$$\max u$$
$$\sum_{i=1}^{n} x_i \cdot A_{ij} \geq u \quad \forall j \in [n]$$
$$\sum_i x_i = 1$$
$$x_i \geq 0 \quad \forall i \in [m]$$

Constraint j says that, if Bob chooses option j, then in expectation (over Alice's options), Alice's payoff must be at least u. Even if Bob uses a mixed strategy, by combining the n inequalities according to his strategy, we can see that Alice still guarantees herself a payoff of u. Although u is technically a variable in the above LP, its value is determined once all of the x_i are determined, and it represents the maximum payoff that Alice can guarantee. The optimal solution of the above LP is known as the *value* of the game.

Remark. Let u^* denote the value of the game, and suppose Alice is forced to announce her strategy before playing. By solving the LP, she can guarantee herself a payoff of at least u^*. Can she do better if she hides her strategy? Surprisingly, the answer is no. Due to a concept known as *duality*, there is a mixed strategy for Bob that guarantees Alice's payoff is at most u^*.[5] Thus, if Bob plays this strategy, then even if Alice hides her strategy, her payoff will still be at most u^*.

6.4 Exercises

1. Describe an LP that has infinitely many feasible solutions, but no optimal solution. Then describe an LP that has infinitely many optimal solutions.

2. In Sec. 6.1, we claimed that any linear program can be converted into canonical form. In this exercise, we consider how this can be done.

   (a) Suppose that we want to minimize $c^\top x$ (rather than maximize).

   (b) Suppose we have a constraint of the form $\alpha^\top x \geq \beta$ or $\alpha^\top x = \beta$, where $\alpha, x \in \mathbb{R}^n$ and $\beta \in \mathbb{R}$.

   (c) Suppose we have a variable x_i that is unrestricted in sign, that is, x_i is allowed to be positive or negative (or 0).

   For each situation above, explain how we can modify the LP such that an optimal solution of the new LP yields an optimal solution to the original LP.

[5] A future version of these notes might contain more about the concept of duality. For now, we'll state that the *dual* of an LP in canonical form is minimizing $b^\top y$ subject to $A^\top y \geq c$ and $y \geq 0$, where $y \in \mathbb{R}^m$ is a new vector of variables. The strong duality theorem states that optimal value of an LP is equal to the optimal value of the dual LP.

3. Explain why it might be impossible to optimally solve a linear program if one of its constraints is a strict inequality (e.g., $x_1 + x_2 < 5$).

4. Let $x^*, y^* \in \mathbb{R}^n$ be two distinct optimal solutions of an LP in canonical form. Prove that $(x^* + y^*)/2 \in \mathbb{R}^n$ is also an optimal solution of the LP. Then show that, in fact, the LP has infinitely many optimal solutions.

5. Recall from Sec. 6.2 that the vertex cover problem can be modeled by an LP. However, the LP didn't specify a matrix A, or vectors b, c, x, as shown in Sec. 6.1. For this problem, determine what A, b, c, and x should be.

6. Again, consider the vertex cover LP from Sec. 6.2. Give an example of a graph such that the optimal LP value is less than the size of the optimal vertex cover.

7. In the weighted vertex cover problem, every vertex u has weight $w_u \geq 0$ and our goal is to find a vertex cover of minimum weight. Show how to modify the algorithm from Sec. 6.2 to obtain a solution whose total weight is at most twice the optimal weight.

8. Consider the maximum flow problem from Ch. 5. Model this problem as an LP, including the exact definitions of the matrix A and vectors b, c, x.

9. Consider the minimum spanning tree problem from Sec. 1.3; we shall model this problem using an *integer* LP (ILP, see Sec. 6.2) in two different ways. Both formulations contain a variable $x_e \in \{0, 1\}$ for every edge $e \in E$. The objective of both is to minimize $\sum_{e \in E} w_e x_e$, but they have different constraints.

   (a) For every $S \subset V$, let $\delta(S)$ denote the set of edges with exactly one endpoint in S. Any tree must contain at least one edge in $\delta(S)$ for every S. Express this idea as a set of constraints in the ILP.

   (b) Any tree contains at most $|S| - 1$ edges within any $S \subseteq V$, and any spanning tree contains $n - 1$ total edges. Express these ideas as constraints in the ILP.

   Suppose we have an optimal solution to the first ILP. Show how we can convert this to into a spanning tree, and prove that the result is in fact an MST. Repeat this for the second ILP. Finally, for only the first ILP, suppose we replace $x_e \in \{0, 1\}$ with $0 \leq x \leq 1$. Show that the optimal value of the resulting LP can be less than the weight of an MST.

10. Suppose Alice and Bob are playing a game. Alice has 2 options, and Bob has 3 options (so there are 6 possible outcomes). If Alice plays option 1, then the 3 possible payoffs are $(2, 1, -3)$. If she plays option 2, then the 3 possible payoffs are $(-1, -2, 1)$. What is the value of this game, and what are the optimal strategies for Alice and Bob?[6]

[6] You may want to use an LP solver; many can be found online.