Lecture notes ?: The simplex algorithm

Vincent Conitzer

1 Introduction

We will now discuss the best-known algorithm (really, a family of algorithms) for solving a linear program, the *simplex algorithm*. We will demonstrate it on an example. Consider again the linear program for our (unmodified) painting example:

maximize $3x_1 + 2x_2$
subject to
$4x_1 + 2x_2 \leq 16$
$x_1 + 2x_2 \leq 8$
$x_1 + x_2 \leq 5$
$x_1 \geq 0; x_2 \geq 0$

To run the simplex algorithm, we introduce a slack variable w_i for each constraint i, so that we can rewrite the linear program in equality form, as follows:

maximize $3x_1 + 2x_2$
subject to
$w_1 = 16 - 4x_1 - 2x_2$
$w_2 = 8 - x_1 - 2x_2$
$w_3 = 5 - x_1 - x_2$
$w_1, w_2, w_3, x_1, x_2 \geq 0$

If we set $x_1 = x_2 = 0$, we get a feasible solution to this linear program. (Of course, this is not the case for every linear program, and we will see what to do if this is not the case later on.) Our goal is to improve this solution. If we increase either x_1 or x_2, then the objective value will increase. Let us start by increasing x_1. At some point, one of the constraints will be violated—that is, one of the slack variables will become negative. Specifically, if we increase x_1 to 4, then the first constraint $4x_1 + 2x_2 \leq 16$ will be just barely satisfied, that is, w_1 will be 0, so we cannot increase x_1 further. (The other constraints are still satisfied at this point.) The objective value at this current solution of $x_1 = 4, x_2 = 0$ is 12, a good start but we are not yet at optimality.

The key trick of the simplex algorithm is that at this point, we rewrite the linear program, changing the roles of some of the original and slack variables. After we do so, the current solution will once again correspond to the origin. Specifically, we remove x_1, whose value is no longer 0, from the objective and the right-hand sides of the equalities; we replace it with an expression involving w_1, whose value is now 0. Specifically, from the first constraint, we know that $w_1 = 16 - 4x_1 - 2x_2$, or equivalently, $x_1 = 4 - 0.25w_1 - 0.5x_2$. We replace the first constraint with this new equality. We also rewrite the objective as $3x_1 + 2x_2 = 3(4 - 0.25w_1 - 0.5x_2) + 2x_2 = 12 - 0.75w_1 + 0.5x_2$. We rewrite the second constraint as $w_2 = 8 - x_1 - 2x_2 = 8 - (4 - 0.25w_1 - 0.5x_2) - 2x_2 = 4 + 0.25w_1 - 1.5x_2$, and the third constraint as $w_3 = 5 - x_1 - x_2 = 5 - (4 - 0.25w_1 - 0.5x_2) - x_2 = 1 + 0.25w_1 - 0.5x_2$. This results in the following linear program, which is equivalent to our original linear program:

maximize $12 - 0.75w_1 + 0.5x_2$
subject to
$x_1 = 4 - 0.25w_1 - 0.5x_2$
$w_2 = 4 + 0.25w_1 - 1.5x_2$
$w_3 = 1 + 0.25w_1 - 0.5x_2$
$w_1, w_2, w_3, x_1, x_2 \geq 0$

Our current solution consists of setting $w_1 = 0, x_2 = 0$. Because both of these are 0, the other values are easy to read off: the current objective value is 12, x_1 is 4, w_2 is 4, and w_3 is 1. We call a linear program written in this way a *dictionary*; the left-hand side variables are called the *basic* variables, and the right-hand side variables (which are set to 0 in the current solution) the *nonbasic* variables. When we moved from the first dictionary to the second dictionary, we performed a *pivot*; in this pivot x_1 was the *entering variable* (going from nonbasic to basic) and w_1 was the *leaving variable* (going from basic to nonbasic).

It is easy to see that we have not yet arrived at an optimal solution: the coefficient of x_2 in the objective is positive, meaning that by increasing x_2 we can increase the objective value. In contrast, there is no sense in increasing w_1 because its coeffient in the objective is negative. So, we will increase x_2 as much as we can without violating a constraint. If we increase x_2 to 2, then w_3 will be equal to 0 (and the other two basic variables will still be positive.) So, x_2 is our entering variable, and w_3 is our leaving variable. We know that $w_3 = 1 + 0.25w_1 - 0.5x_2$, or equivalently, $x_2 = 2 + 0.5w_1 - 2w_3$, so we replace the last constraint with this expression. We also use this expression to replace occurrences of x_2 in the objective and the right-hand sides of the constraints. We obtain:

maximize $13 - 0.5w_1 - w_3$
subject to
$x_1 = 3 - 0.5w_1 + w_3$
$w_2 = 1 - 0.5w_1 + 3w_3$
$x_2 = 2 + 0.5w_1 - 2w_3$
$w_1, w_2, w_3, x_1, x_2 \geq 0$

Again, our current solution corresponds to setting the nonbasic variables w_1 and w_3 to 0. We can easily read off that the current value of our solution is 13, and that $x_1 = 3, x_2 = 2$. It is interesting to note that in this pivot, by increasing x_2, we automatically decreased x_1 from 4 to 3. The next step would be to increase either w_1 or w_3 to increase the objective. However, the coefficient on both of these variables in the objective is negative, so there is no point to doing this: it will only decrease the objective. So the simplex algorithm terminates. In fact, this last dictionary gives a *proof* that our current solution is optimal: because w_1 and w_3 must be nonnegative, clearly the objective value can be at most 13, and our current solution achieves this.

2 A richer example

To illustrate some additional phenomena involving the simplex algorithm, we now consider the following richer example, corresponding to the combinatorial auction example with partially acceptable bids from before.

maximize $4x_1 + 5x_2 + 4x_3 + 7x_4 + x_5$
subject to
$x_1 + x_3 + x_4 \leq 1$
$x_1 + x_2 + x_4 \leq 1$
$x_2 + x_3 \leq 1$
$x_4 + x_5 \leq 1$
$x_1, x_2, x_3, x_4, x_5 \geq 0$

We omitted the constraints that for each j, $x_j \leq 1$, but it is easy to see that this is implied by the other constraints. We now write this linear program in equality form:

maximize $4x_1 + 5x_2 + 4x_3 + 7x_4 + x_5$
subject to
$w_1 = 1 - x_1 - x_3 - x_4$
$w_2 = 1 - x_1 - x_2 - x_4$
$w_3 = 1 - x_2 - x_3$
$w_4 = 1 - x_4 - x_5$
$w_1, w_2, w_3, w_4, x_1, x_2, x_3, x_4, x_5 \geq 0$

Again, this is a *feasible dictionary*, in the sense that setting all of the nonbasic variables to 0 corresponds to a feasible solution. Now we need to choose an entering variable. All of the x_j have a positive coefficient in the objective, so we can choose any one of them. One natural heuristic is to choose the one with the greatest coefficient, because we want to improve the objective by as much as possible. So we choose x_4 as the entering variable. Once we increase x_4 to 1, w_1, w_2, and w_4 all *simultaneously* become 0. That means that in this case, we can choose any one of them as the leaving variable. Let us choose w_1 as the leaving variable. The resulting pivot produces the following new dictionary:

maximize $7 - 7w_1 - 3x_1 + 5x_2 - 3x_3 + x_5$
subject to
$x_4 = 1 - w_1 - x_1 - x_3$
$w_2 = w_1 - x_2 + x_3$
$w_3 = 1 - x_2 - x_3$
$w_4 = w_1 + x_1 + x_3 - x_5$
$w_1, w_2, w_3, w_4, x_1, x_2, x_3, x_4, x_5 \geq 0$

We now have a choice between x_2 and x_5 as the next entering variable; because x_2 has a larger coefficient in the objective, let us choose x_2. Now, something strange happens: we cannot increase x_2 at all without violating one of the constraints, because the current value of w_2 is already 0 and we have $-x_2$ on the right-hand side. Still, in some sense, w_2 is the first basic variable that becomes 0, so we choose it as the leaving variable. This results in the following dictionary:

maximize $7 - 2w_1 - 5w_2 - 3x_1 + 2x_3 + x_5$
subject to
$x_4 = 1 - w_1 - x_1 - x_3$
$x_2 = w_1 - w_2 + x_3$
$w_3 = 1 - w_1 + w_2 - 2x_3$
$w_4 = w_1 + x_1 + x_3 - x_5$
$w_1, w_2, w_3, w_4, x_1, x_2, x_3, x_4, x_5 \geq 0$

We note that in this last pivot, the objective remained at 7 (whereas "normally" the objective increases). Such a pivot is called a *degenerate pivot*. Degenerate pivots can cause difficulties in general:

for example, it is possible that a sequence of degenerate pivots returns us to the same dictionary that we started with, resulting in an infinite loop. (This cannot happen with nondegenerate pivots, because the strict increases in the objective make it impossible to return to the same dictionary.) We will see how to deal with this in general later; however, "usually" degenerate pivots do not cause any problems, and, as it turns out, it does not cause any trouble here. Let x_3 be the next entering variable, so that w_3 becomes the next leaving variable. We obtain the following dictionary from this (nondegenerate!) pivot:

maximize $8 - 3w_1 - 4w_2 - w_3 - 3x_1 + x_5$
subject to
$x_4 = 0.5 - 0.5w_1 - 0.5w_2 + 0.5w_3 - x_1$
$x_2 = 0.5 + 0.5w_1 - 0.5w_2 - 0.5w_3$
$x_3 = 0.5 - 0.5w_1 + 0.5w_2 - 0.5w_3$
$w_4 = 0.5 + 0.5w_1 + 0.5w_2 - 0.5w_3 + x_1 - x_5$
$w_1, w_2, w_3, w_4, x_1, x_2, x_3, x_4, x_5 \geq 0$

Finally, we choose x_5, which is the only variable with a positive coefficient in the objective, as the entering variable, so that w_4 becomes the leaving variable and we end up with the following dictionary:

maximize $8.5 - 2.5w_1 - 3.5w_2 - 1.5w_3 - w_4 - 2x_1$
subject to
$x_4 = 0.5 - 0.5w_1 - 0.5w_2 + 0.5w_3 - x_1$
$x_2 = 0.5 + 0.5w_1 - 0.5w_2 - 0.5w_3$
$x_3 = 0.5 - 0.5w_1 + 0.5w_2 - 0.5w_3$
$x_5 = 0.5 + 0.5w_1 + 0.5w_2 - 0.5w_3 - w_4 + x_1$
$w_1, w_2, w_3, w_4, x_1, x_2, x_3, x_4, x_5 \geq 0$

Now, all of the coefficients in the objective are nonnegative, so we know that it is impossible to obtain a better solution than 8.5, and we have arrived at the optimal solution which sets $x_2 = x_3 = x_4 = x_5 = 0.5$.

3 General comments

In general, we can choose any nonbasic variable with positive coefficient in the objective as the entering variable; as the leaving variable, we must choose the basic variable that drops to zero first as the entering variable increases (and if there are multiple basic variables that drop to zero first, we can choose any one of them). (It may be the case that no basic variables will ever drop to zero; in this case, the linear program is unbounded, which we will illustrate with an example later.) Rules for making the above choices are called *pivoting rules*.

In the above examples, we were lucky in the sense that whenever we changed a variable from nonbasic to basic, it never changed back; as a result, in some sense, we traversed the shortest possible path of pivots. Unfortunately, in general, this is not the case. In fact, for the common pivoting rules, there are examples of linear programs where the simplex algorithm goes through a path of exponentially many dictionaries (*Klee-Minty cubes* are a common example of such linear programs). It is not known if there is a pivoting rule that only requires polynomially many pivots on any example; in fact it is not known if there is always a path of only polynomially many pivots to the optimal solution (this is related to a conjecture known as *Hirsch's conjecture*). In practice, however, the simplex algorithm tends to be extremely fast.

Things may be even worse, though: as we mentioned above, if there are degenerate pivots, then

there is the possibility that the simplex algorithm gets stuck in an infinite loop, that is, it *cycles*. This can be taken care of, however, and this is the topic of the next section. (In fact, cycling is extremely rare and hardly ever an issue even if we do not take precautions.)

4 Avoiding cycling

As it turns out, it is possible to avoid cycling in the simplex algorithm. One way to avoid this is to use *Bland's rule* for pivoting. When this rule has a choice among multiple variables as the entering (or leaving) variable, it always chooses the one with the lowest index.

Another way to avoid cycling is to use the *lexicographic method*, which we present now. The idea of the lexicographic method is that in some sense, we have to be extremely unlucky to have any degenerate pivots at all, because if the parameters of the problem were chosen at random from an interval it would be extremely unlikely that the constant in one of the constraints would be 0 at any point. (Of course, in the real world, this is not how parameters are chosen, and we have already seen from the combinatorial auction example that degenerate pivots do occur in "reasonable" instances.) The idea of the lexiographic method is to slightly perturb the constants so that degeneracy does not occur but the optimal solution is not really affected. While it is possible to use random perturbations, this is not what the lexicographic method does. Rather, it adds ϵ_1 to the first constraint, ϵ_2 to the second, and so on, where the ϵ_i are positive abstract symbols such that each ϵ_i is infinitesimally smaller than the preceding ϵ_{i-1}, and ϵ_1 is infinitesimally smaller than all the "true" parameters of the instance.

For example, with our painting problem instance, we would start out with:

maximize $3x_1 + 2x_2$
subject to
$w_1 = 16 + \epsilon_1 - 4x_1 - 2x_2$
$w_2 = 8 + \epsilon_2 - x_1 - 2x_2$
$w_3 = 5 + \epsilon_3 - x_1 - x_2$
$w_1, w_2, w_3, x_1, x_2 \geq 0$

After the first pivot, we obtain:

maximize $12 + 0.75\epsilon_1 - 0.75w_1 + 0.5x_2$
subject to
$x_1 = 4 + 0.25\epsilon_1 - 0.25w_1 - 0.5x_2$
$w_2 = 4 - 0.25\epsilon_1 + \epsilon_2 + 0.25w_1 - 1.5x_2$
$w_3 = 1 - 0.25\epsilon_1 + \epsilon_3 + 0.25w_1 - 0.5x_2$
$w_1, w_2, w_3, x_1, x_2 \geq 0$

Once we reach an optimal solution, we simply drop all the ϵ_i terms from the solution.

It should be noted that none of this has any effect on our choice of entering variable; that is, we can choose the entering variable however we like. It may, however, affect our choice of leaving variable, because the ϵ_i may break a tie. As it turns out, they *always* break ties, that is, there is never more than one choice for the leaving variable. It also turns out that none of the constants in the constraints ever become zero, which immediately implies that we cannot have any degenerate pivots and hence we cannot cycle. Both of these claims are implied by the following observation. Consider the matrix of the coefficients on the ϵ_i in the constraints. For example, for the first dictionary above, we have the matrix

1 0 0
0 1 0
0 0 1

For the second dictionary, we have:
0.25 0 0
-0.25 1 0
-0.25 0 1

Now, it is not difficult to see that this matrix will always have full rank, that is, rank m, because this is clearly true for the first matrix, and subsequent matrices are obtained by multiplying rows by nonzero constants and adding multiples of other rows to rows. It follows that no row can ever consist only of zeroes, hence there can be no degenerate pivots.

5 Unboundedness

We have skipped over the issue of what happens if the linear program is unbounded. This is best illustrated with an example. Let us consider the following unbounded linear program:

maximize $3x_1 + 2x_2$
subject to
$w_1 = 1 - x_1 + x_2$
$w_2 = 1 + x_1 - x_2$
$w_1, w_2, x_1, x_2 \geq 0$

We first choose x_1 as the entering variable, resulting in w_1 being the leaving variable:

maximize $3 - 3w_1 + 5x_2$
subject to
$x_1 = 1 - w_1 + x_2$
$w_2 = 2 - w_1$
$w_1, w_2, x_1, x_2 \geq 0$

Now we must choose x_2 as the entering variable. However, as we increase x_2, none of the basic variables decrease. Hence, we can increase x_2 forever, indicating that the program is unbounded.

6 Finding a feasible dictionary

So far, we have assumed that we have an initial feasible dictionary. While in many problems, setting all of the variables to zero corresponds to a feasible solution, this is certainly not always the case. If we do not have an initial feasible dictionary, we first need to find one, which can also be done using the simplex algorithm. Finding an initial feasible dictionary is generally referred to as "Phase 1." Going from there to the optimal solution (as we have done above) is called "Phase 2." In this section, we discuss Phase 1.

Let us again take our painting problem instance; we will add a constraint to it that we must produce at least two reproductions of paintings, that is, $x_1 + x_2 \geq 2$ or equivalently, $-x_1 - x_2 \leq -2$. (Of course, this will not affect the optimal solution, because we know that even without this constraint, we create at least 2 reproductions.) This results in the following linear program:

maximize $3x_1 + 2x_2$
subject to
$4x_1 + 2x_2 \leq 16$
$x_1 + 2x_2 \leq 8$
$x_1 + x_2 \leq 5$
$-x_1 - x_2 \leq -2$
$x_1 \geq 0; x_2 \geq 0$

Now, setting $x_1 = x_2 = 0$ is no longer a feasible solution. To find a feasible solution, we can temporarily forget about the objective. Instead, we add an auxiliary variable x_0, and we require that each of the original constraints is violated by at most x_o. This results in the following linear program:

maximize $-x_0$
subject to
$4x_1 + 2x_2 - x_0 \leq 16$
$x_1 + 2x_2 - x_0 \leq 8$
$x_1 + x_2 - x_0 \leq 5$
$-x_1 - x_2 - x_0 \leq -2$
$x_0, x_1, x_2 \geq 0$

This linear program has an optimal solution with objective value 0 if and only if there is a feasible solution to the original linear program. In equality form, we have:

maximize $-x_0$
subject to
$w_1 = 16 - 4x_1 - 2x_2 + x_0$
$w_2 = 8 - x_1 - 2x_2 + x_0$
$w_3 = 5 - x_1 - x_2 + x_0$
$w_4 = -2 + x_1 + x_2 + x_0$
$w_1, w_2, w_3, w_4, x_0, x_1, x_2 \geq 0$

This is still not a feasible dictionary, but we can transform it into a feasible dictionary by choosing x_0 as the entering variable and the most negative basic variable, w_4, as the leaving variable. This results in the following dictionary:

maximize $-2 - w_4 + x_1 + x_2$
subject to
$w_1 = 18 + w_4 - 5x_1 - 3x_2$
$w_2 = 10 + w_4 - 2x_1 - 3x_2$
$w_3 = 7 + w_4 - 2x_1 - 2x_2$
$x_0 = 2 + w_4 - x_1 - x_2$
$w_1, w_2, w_3, w_4, x_0, x_1, x_2 \geq 0$

Next, we choose (say) x_1 as the entering variable, so that x_0 is the leaving variable, resulting in:

maximize $-x_0$
subject to
$w_1 = 8 - 4w_4 + 2x_2 + 5x_0$
$w_2 = 6 - w_4 - x_2 + 2x_0$
$w_3 = 3 - w_4 + 2x_0$
$x_1 = 2 + w_4 - x_2 - x_0$
$w_1, w_2, w_3, w_4, x_0, x_1, x_2 \geq 0$

Now, all the coefficients in the objective are nonpositive, so we have found the optimal solution to the auxiliary problem. We transform this into a feasible dictionary for the original problem by simply dropping x_0 everywhere, and replacing the objective with the original one:

maximize $3x_1 + 2x_2$
subject to
$w_1 = 8 - 4w_4 + 2x_2$
$w_2 = 6 - w_4 - x_2$
$w_3 = 3 - w_4$
$x_1 = 2 + w_4 - x_2$
$w_1, w_2, w_3, w_4, x_1, x_2 \geq 0$

Because we now have a feasible dictionary, we can start Phase 2.

7 The simplex algorithm and duality

As it turns out, as the simplex solves the primal, it simultaneously solves the dual as well. We illustrate this with our standard example.

maximize $3x_1 + 2x_2$
subject to
$4x_1 + 2x_2 \leq 16$
$x_1 + 2x_2 \leq 8$
$x_1 + x_2 \leq 5$
$x_1 \geq 0; x_2 \geq 0$

The dual (written as a maximization problem instance) is:

maximize $-16y_1 - 8y_2 - 5y_3$
subject to
$4y_1 + y_2 + y_3 \geq 3$
$2y_1 + 2y_2 + y_3 \geq 2$
$y_1 \geq 0; y_2 \geq 0; y_3 \geq 0$

In equality form, the primal is:

maximize $3x_1 + 2x_2$
subject to
$w_1 = 16 - 4x_1 - 2x_2$
$w_2 = 8 - x_1 - 2x_2$
$w_3 = 5 - x_1 - x_2$
$w_1, w_2, w_3, x_1, x_2 \geq 0$

We can similarly write the dual in equality form:

maximize $-16y_1 - 8y_2 - 5y_3$
subject to
$z_1 = -3 + 4y_1 + y_2 + y_3$
$z_2 = -2 + 2y_1 + 2y_2 + y_3$
$z_1, z_2, y_1, y_2, y_3 \geq 0$

It should be noted that this is not a feasible dictionary for the dual.

After the first pivot on the primal, for which x_1 is the entering variable and w_1 the leaving variable, we get:

maximize $12 - 0.75w_1 + 0.5x_2$
subject to
$x_1 = 4 - 0.25w_1 - 0.5x_2$
$w_2 = 4 + 0.25w_1 - 1.5x_2$
$w_3 = 1 + 0.25w_1 - 0.5x_2$
$w_1, w_2, w_3, x_1, x_2 \geq 0$

We can perform the analogous pivot on the dual dictionary (it is possible to perform pivots on infeasible dictionaries, as we saw in our discussion of Phase 1). We know that x_1 corresponds to z_1 and w_1 to y_1 (recall the complementary slackness theorem), so in the analogous pivot on the dual, those are the entering and leaving variables. This results in the following dictionary for the dual.

maximize $-12 - 4z_1 - 4y_2 - 1y_3$
subject to
$y_1 = 0.75 + 0.25z_1 - 0.25y_2 - 0.25y_3$
$z_2 = -0.5 + 0.5z_1 + 1.5y_2 + 0.5y_3$
$z_1, z_2, y_1, y_2, y_3 \geq 0$

This dictionary is still not feasible, but it seems to be getting closer. Can you see the relationship betweeen the current primal dictionary and the current dual dictionary? Consider, in each case, the matrix of all the constants and coefficients in the objective and right-hand sides of the constraints, ignoring the variables. For the primal, this is a 4×3 matrix of rational numbers, and for the dual, a 3×4 matrix. In fact, the dual matrix is the *negative transpose* of the primal matrix. This was also the case in the original dictionaries. In fact, the simplex method preserves this property in each pivot (as long as the pivot on the dual uses entering and leaving variables corresponding to those used for the primal). The current dual dictionary is always the dictionary that would result directly from taking the dual of the current primal dictionary.

We have one more pivot to go until we reach optimality. The resulting primal dictionary is:

maximize $13 - 0.5w_1 - w_3$
subject to
$x_1 = 3 - 0.5w_1 + w_3$
$x_2 = 2 + 0.5w_1 - 2w_3$
$w_2 = 1 - 0.5w_1 + 3w_3$
$w_1, w_2, w_3, x_1, x_2 \geq 0$

It should be noted that we reordered the constraints to keep the basic variables sorted. In the dual, the entering variable is y_3 (corresponding to the leaving variable w_3 in the primal), and the leaving variable is z_2 (corresponding to the entering variable x_2 in the primal), resulting in the dictionary:

maximize $-13 - 3z_1 - 2z_2 - 1y_2$
subject to
$y_1 = 0.5 + 0.5z_1 - 0.5z_2 + 0.5y_2$
$y_3 = 1 - 1z_1 + 2z_2 - 3y_2 \;\; z_1, z_2, y_1, y_2, y_3 \geq 0$

Again, the negative transpose property is preserved. The dual dictionary is now feasible because all the constants in the constraints (the current values for the basic variables) are nonnegative. This is equivalent to the coefficients in the objective all being nonpositive, which is the termination condition for the simplex algorithm. Thus, once the simplex algorithm terminates on the primal, the corresponding dual dictionary is feasible. Moreover, the objective value for the final dual dictionary must be the negative of that for the optimal primal dictionary, by the negative transpose property; if we convert the dual back to a minimization problem, then the objectives must be the same. Therefore, the final dual dictionary must in fact be optimal. (In effect, this is a proof of the strong duality theorem.) So we have found the optimal dual solution $y_1 = 0.5, y_3 = 1$. Actually, by the negative transpose property, we could have read these values off directly from the objective coefficients in the final primal dictionary. In effect, because of the negative transpose property, there is no real reason to keep both the primal and the dual dictionaries around: one can easily be constructed from the other. By this reasoning, for any problem, we can also run the simplex algorithm on the dual, and from the result we can easily read off the solution to the primal. This is known as the *dual simplex method*.