Math 3272: Linear Programming[1] Mikhail Lavrov

Lecture 16: Applications of the dual simplex method

October 11, 2022 *Kennesaw State University*

1 Example: another look at a terrible cube

With the dual simplex method in our toolbox, there are now twice as many linear programs that we can solve in one phase. Suppose we have a linear program with constraints $A\mathbf{x} \leq \mathbf{b}$, for which we decide to write down an initial dictionary where the slack variables are basic. Then:

- Provided $\mathbf{b} \geq \mathbf{0}$, this initial dictionary is feasible, and so we can use the ordinary simplex method.
- Provided we are minimizing $\mathbf{c}^\mathsf{T}\mathbf{x}$ where $\mathbf{c} \geq \mathbf{0}$ (or, equivalently, maximizing $\mathbf{c}^\mathsf{T}\mathbf{x}$ where $\mathbf{c} \leq \mathbf{0}$), this initial dictionary is dual feasible, and so we can use the dual simplex method.

Together, these two cases cover many practical examples, but we've already seen problems that don't fit in either category. For example, take the linear program below, which is the 3-dimensional version of the problem we used to make Bland's rule take exponentially many steps:

$$\begin{array}{ll} \underset{x_1,x_2,x_3\in\mathbb{R}}{\text{maximize}} & x_3 \\ \text{subject to} & 0.1 \leq x_1 \leq 1 - 0.1 \\ & 0.1x_1 \leq x_2 \leq 1 - 0.1x_1 \\ & 0.1x_2 \leq x_3 \leq 1 - 0.1x_2 \\ & x_1, x_2, x_3 \geq 0 \end{array}$$

$$\begin{array}{rrrrr} \max \zeta = & 0 & & & + x_3 \\ \hline w_1 = & -0.1 + & x_1 & & \\ w_1' = & 0.9 - & x_1 & & \\ w_2 = & 0 - 0.1x_1 + & & x_2 & \\ w_2' = & 1 - 0.1x_1 - & & x_2 & \\ w_3 = & 0 & & - 0.1x_2 + x_3 & \\ w_3 = & 1 & & - 0.1x_2 - x_3 & \end{array}$$

If we write each pair of inequalities $\mathsf{LB} \leq x_i \leq \mathsf{UB}$ as $\mathsf{LB} + w_i = x_i$ and $x_i + w_i' = \mathsf{UB}$ (where LB and UB stand in for our various creative lower and upper bounds on x_i) then we can write down an initial dictionary in terms of six slack variables. But this dictionary is neither feasible (since $w_1 < 0$) nor dual feasible (since the reduced cost of x_3 is positive).

We could handle this using a two-phase method where we add an artificial variable. But let's see a different method of accomplishing the same thing.

2 The two-phase dual simplex method

2.1 The plan

Just like our earlier two-phase methods, the two-phase dual simplex method involves solving a phase one problem before we get to the problem we actually want to solve. But this method is much more economical: though it will require an auxiliary objective function, we will not need to add any new variables or constraints.

[1] This document comes from the Math 3272 course webpage: https://facultyweb.kennesaw.edu/mlavrov/courses/3272-fall-2022.php

The logic is this: dual feasibility only depends on the objective function, and not on the constraints. So if we replace our original objective function by an auxiliary objective function, we can choose that auxiliary objective function to make our dictionary dual feasible! Then, we can apply the dual simplex method and solve that phase one problem.

Of course, in the phase one problem, we'll be optimizing something completely unrelated to what we actually want. This doesn't matter: once the phase one problem is solved, we'll have a dictionary that's both feasible and dual feasible. Now, replace the auxiliary objective function by our original objective function. Unless we're very lucky, the resulting dictionary won't be dual feasible—however, it will still be feasible, because we didn't touch the constraints! Therefore we can continue with the ordinary simplex method to solve the problem we actually care about.

What should our auxiliary objective function be? Anything we like, as long as it gives us a dual feasible dictionary. In general, this means minimizing any linear expression with nonnegative coefficients on all the nonbasic (non-slack) variables.

You might be tempted to go with the simplest such linear expression: minimize 0. This is a bad choice, because the value of 0 doesn't change as we pivot from basis to basis. This means that the dual simplex method will constantly be doing "dual degenerate pivots", and once again we have to worry about cycling.

A simple choice that will work as well as any other in general is to minimize the sum of all the nonbasic variables. If you're worried about degeneracy, you could borrow from the lexicographic pivoting rule and decide to minimize $\epsilon_1 x_1 + \epsilon_2 x_2 + \cdots + \epsilon_n x_n$, where $\epsilon_1 \gg \epsilon_2 \gg \cdots \gg \epsilon_n > 0$. We will not bother doing this in our examples.

2.2 Solving the example

Let's try this on our example. Our auxiliary objective will be to minimize $\xi = x_1 + x_2 + x_3$:

$$\begin{array}{rrrrr}
\min \xi = & 0 + & x_1 + & x_2 + x_3 \\
\hline
w_1 = & -0.1 + & x_1 & & \\
w_1' = & 0.9 - & x_1 & & \\
w_2 = & 0 - 0.1x_1 + & x_2 & & \\
w_2' = & 1 - 0.1x_1 - & x_2 & & \\
w_3 = & 0 & - 0.1x_2 + x_3 & & \\
w_3 = & 1 & - 0.1x_2 - x_3 & &
\end{array}$$

Only one basic variable currently has a negative value: $w_1 = -0.1$. The only variable with a positive coefficient in w_1's equation is x_1, so we have no choice in our pivot: x_1 enters and w_1 leaves. We get $x_1 = 0.1 + w_1$ when we solve for x_1, and then we substitute that in the other rows, getting

$$\begin{array}{rrrrr}
\min \xi = & 0.1 + & w_1 + & x_2 + x_3 \\
\hline
x_1 = & 0.1 + & w_1 & & \\
w_1' = & 0.8 - & w_1 & & \\
w_2 = & -0.01 - 0.1w_1 + & x_2 & & \\
w_2' = & 0.99 - 0.1w_1 - & x_2 & & \\
w_3 = & 0 & - 0.1x_2 + x_3 & & \\
w_3 = & 1 & - 0.1x_2 - x_3 & &
\end{array}$$

Again, only one basic variable has a negative value: $w_2 = -0.01$. The only variable with a positive coefficient in w_2's equation is x_2, so we still have no choice in our pivot: x_2 enters and w_2 leaves. We get $x_2 = 0.01 + 0.1w_1 + w_2$ when we solve for x_2, and then we substitute that in the other rows, getting

$$\begin{aligned}
\min \xi = \quad & 0.11 + 1.1w_1 + w_2 + x_3 \\
\hline
x_1 = \quad & 0.1 + w_1 \\
w_1' = \quad & 0.8 - w_1 \\
x_2 = \quad & 0.01 + 0.1w_1 + w_2 \\
w_2' = \quad & 0.98 - 0.2w_1 - w_2 \\
w_3 = \quad & -0.001 - 0.01w_1 - 0.1w_2 + x_3 \\
w_3' = \quad & 0.999 - 0.01w_1 - 0.1w_2 - x_3
\end{aligned}$$

Again, only one basic variable has a (barely) negative value: $w_3 = -0.001$. And yet again, there is only one choice of entering variable to replace w_3 as a leaving variable: x_3. When we pivot, we solve for x_3 and get $x_3 = 0.001 + 0.01w_1 + 0.1w_2 + w_3$, leading us to the following dictionary:

$$\begin{aligned}
\min \xi = \quad & 0.111 + 1.11w_1 + 1.1w_2 + w_3 \\
\hline
x_1 = \quad & 0.1 + w_1 \\
w_1' = \quad & 0.8 - w_1 \\
x_2 = \quad & 0.01 + 0.1w_1 + w_2 \\
w_2' = \quad & 0.98 - 0.2w_1 - w_2 \\
x_3 = \quad & 0.001 + 0.01w_1 + 0.1w_2 + w_3 \\
w_3' = \quad & 0.998 - 0.02w_1 - 0.2w_2 - w_3
\end{aligned}$$

We are done with phase one! We don't really care that we've optimized ξ, but the good news for us is that the dictionary is feasible. When we replace the objective function with $\zeta = x_3 = 0.001 + 0.01w_1 + 0.1w_2 + w_3$, it remains feasible:

$$\begin{aligned}
\max \zeta = \quad & 0.001 + 0.01w_1 + 0.1w_2 + w_3 \\
\hline
x_1 = \quad & 0.1 + w_1 \\
w_1' = \quad & 0.8 - w_1 \\
x_2 = \quad & 0.01 + 0.1w_1 + w_2 \\
w_2' = \quad & 0.98 - 0.2w_1 - w_2 \\
x_3 = \quad & 0.001 + 0.01w_1 + 0.1w_2 + w_3 \\
w_3' = \quad & 0.998 - 0.02w_1 - 0.2w_2 - w_3
\end{aligned}$$

Now we are ready to maximize ζ, and if we like, we can use Bland's rule and take the most ridiculous number of steps possible to do it.

By the way, if you notice, the actual artificial objective function ξ never played a role in our pivoting. This is not guaranteed to happen, but it's not particularly surprising: if we want the point $(x_1, x_2, x_3) = (0, 0, 0)$ to be dual feasible for ξ, then ξ will probably be minimized at some point close to $(0, 0, 0)$. This means that we shouldn't stress out too much about our choice of ξ in problems like this.

2.3 Problems in equational form

This strategy isn't quite enough to deal with constraints of the form $A\mathbf{x} = \mathbf{b}$ where $\mathbf{x} \geq \mathbf{0}$.

For systems like this, we have a three-step procedure:

1. Begin by row-reducing the system: just the usual Gaussian elimination you learn in linear algebra. You will end up solving the equations for some set of basic variables, which is not particularly under your control.

2. Now add an auxiliary objective function to build a dual feasible dictionary out of the basic solution you got. For example, this objective could be to minimize the sum of whichever variables end up nonbasic after step 1.

   Use the dual simplex method to solve this phase one problem.

3. From the optimal dictionary, replace the auxiliary objective function by whatever objective function you originally wanted to optimize. As before, use the ordinary simplex method to solve the phase two problem.

3 Warm starts and row generation

Problem 1. *Once again, you're in charge of a factory that produces gizmos, widgets, and doodads. Let's say you must produce at least* 10 *of each object a day, but you're limited to using at most* 2000 *pounds of iron: it takes* 10 *pounds to make a doodad,* 20 *pounds to make a gizmo, and* 30 *pounds to make a widget. You must maximize the profit:* \$20 *per doodad,* \$30 *per gizmo, and* \$30 *per widget.*

Let's suppose you set up the problem (on the left), find the optimal dictionary (on the right), and bring the optimal solution to your boss:

$$\begin{array}{rl} \underset{x_d,x_g,x_w\in\mathbb{R}}{\text{maximize}} & 20x_d + 30x_g + 40x_w \\ \text{subject to} & \begin{array}{rcl} x_d & \geq & 10 \\ x_g & \geq & 10 \\ x_w & \geq & 10 \\ 10x_d + 20x_g + 30x_w & \leq & 2000 \\ x_d, x_g, x_w & \geq & 0 \end{array} \end{array}$$

$$\begin{array}{rrrrr} \max\zeta = 3700 - & 10w_2 - & 20w_3 - & 2w_4 \\ \hline w_1 = 140 - & 2w_2 - & 3w_3 - & 0.1w_4 \\ x_d = 150 - & 2w_2 - & 3w_3 - & 0.1w_4 \\ x_g = 10 + & w_2 & & \\ x_w = 10 & & + & w_3 \end{array}$$

Your boss takes one look at the printout and says, "No, no, that won't work. What kind of fool doesn't know that the doohickey will overheat if you run it for more than 10 hours a day? The table is right there on the machine: you need the doohickey for 15 minutes per doodad and 5 minutes per gizmo. Go fix this, I need the factory schedule yesterday!"

Do you have to start from scratch? No. Let's take the new constraint and insert it into our final dictionary. Okay, this takes a bit of work: the constraint is $15x_d + 5x_g \leq 600$, which we write as $w_5 = 600 - 15x_d - 5x_g$ for a new slack variable w_5. But x_d and x_g are also basic, so we substitute their equations in:

$$w_5 = 600 - 15(150 - 2w_2 - 3w_3 - 0.1w_4) - 5(10 + w_2)$$

which simplifies to $w_5 = -1700 + 25w_2 + 45w_3 + 1.5w_4$. Now we can add that to our dictionary.

The new dictionary is

$$\begin{aligned}
\max \zeta = \quad & 3700 - 10w_2 - 20w_3 - \quad 2w_4 \\
\hline
w_1 = \quad & 140 - \quad 2w_2 - \quad 3w_3 - 0.1w_4 \\
x_d = \quad & 150 - \quad 2w_2 - \quad 3w_3 - 0.1w_4 \\
x_g = \quad & 10 + \quad w_2 \\
x_w = \quad & 10 + \qquad\qquad w_3 \\
w_5 = \quad & -1700 + 25w_2 + 45w_3 + 1.5w_4
\end{aligned}$$

which, of course, is no longer feasible: the doohickey constraint is not satisfied. (We're trying to use it for over 38 hours per day. Poor doohickey!) But the dictionary is still dual feasible, because it started out dual feasible before we added the new constraint. So we can try to fix it with the dual simplex method.

In this case, only a single step is required. The leaving variable must be w_5. All three nonbasic variables have positive coefficients, so we compare the ratios: $\frac{10}{25}$, $\frac{20}{45}$, and $\frac{2}{1.5}$. The smallest ratio is $\frac{10}{25}$, so w_2 is the entering variable. After one pivot step, we get:

$$\begin{aligned}
\max \zeta = 3020 - \quad & 2w_3 - \quad 1.4w_4 - \quad 0.4w_5 \\
\hline
w_1 = \quad & 4 + 0.6w_3 + 0.02w_4 - 0.08w_5 \\
x_d = \quad & 14 + 0.6w_3 + 0.02w_4 - 0.08w_5 \\
x_g = \quad & 78 - 1.8w_3 - 0.06w_4 + 0.04w_5 \\
x_w = \quad & 10 + \qquad w_3 \\
w_5 = \quad & 68 - 1.8w_3 - 0.06w_4 + 0.04w_5
\end{aligned}$$

The point $(x_d, x_g, x_w) = (14, 78, 10)$ is our new optimal solution that takes doohickey usage into account.

This is called solving the problem from a **warm start**: we start from a dictionary that was optimal for a closely-related problem. There is no guarantee that a warm start will be faster than solving the problem from scratch, and in particular, there is no guarantee that a single pivot step will be enough, like it was here. However, in practice, it often seems to work well.

The situation where we forget about a constraint is contrived. Sometimes this situation occurs when you have to solve a linear program for many similar problems. Suppose that the factory constraints change a little every day; then always doing a warm start from the previous day's optimal solution might be a good idea.

A related concept is **row generation**,[2] which we'll see several times later this semester. Here, we know in advance that our set of constraints is incomplete, but we don't know which of many possible constraints we'll need. So we solve our linear program, then look at an optimal solution to see if it violates any constraints we left out. (The way this is done depends on the exact application, and often involves techniques outside linear programming.) Then, we add some missing constraint and use the dual simplex method to continue.

[2]Note: the term "row generation" is often used specifically for a technique called Benders decomposition, which is one particular example of the general idea.