LP Duality (3/6)

Learning Objectives:

- Describe the mathematical definition of shadow prices. (Analyze)
- Interpret the meaning of the shadow prices given by a LP solver. (Analyze)
- Describe the basic commands to implement a LP in Guorbi. (Code)

Textbook readings: DMD 7.5.

Clarifying unclear points

The following points were explained in class. (See lecture recording for more information.)

- When we have integer decision variables, we no longer have a linear program (LP), but a mixed-integer program (MIP).
- When solving a 2-D LP graphically (see last session), we find the direction of the objective as follows:
- 1) Transform the linear objective into a maximum. (If it's currently min, then do max of its negative.)
- 2) Read the coefficient in front of every variable. For example if the expression is $30y - 20x$, then the coefficient for x is -20 and for y is 30.
- 3) Plot a vector from the origin (0 in all coordinates) to the vector of coefficients. In the above example, the vector is $(-20, 30)$. This is the direction of the objective. The LP is finding points in the feasible solution that are furthest along this direction.
- Binding constraints are helpful for:
- 1) identifying bottlenecks. (The magnitude of bottlenecks are given by the shadow prices.) Non-binding constraints always have zero shadow costs.
- 2) solving for the optimal solution, as it gives us a system of linear equations that can be used to solve.

Shadow Prices (aka Dual Values)

The rigorous definition of the shadow price of a constraint is "the rate of change in the value of the optimal objective when the right hand side (RHS) of the constraint is increased by an infitessimal amount." If you know calculus, this is the partial derivative of the optimal objective function with respect to the right hand side of the constraint, evaluated at the current value. For example, consider the LP from last class.

$$\begin{array}{rr}
\text{maximize} & 20X + 10Y \\
\text{subject to:} & \\
\text{(Material 1)} & 4X + Y \leq 60 \\
\text{(Material 2)} & 2Y \leq 48 \\
\text{(Labor)} & X + Y \leq 30 \\
\text{(Non-negativity)} & X, Y \geq 0
\end{array}$$

The value of the optimal objective is 400. The shadow price of the "Material 1" constraint is defined to be the following ratio with a sufficiently small $\delta > 0$,

$$\frac{(\text{Optimal objective when we replace RHS by } 60+\delta) - 400}{\delta}$$

The theory of shadow price states that in a region with sufficiently small δ, the above ratio would be constant, and is defined to be the shadow price. For example, with a small $\delta = 0.001$, we solve the LP,

$$\begin{aligned}
\text{maximize} \quad & 20X + 10Y \\
\text{subject to:} \quad & \\
\text{(Material 1)} \quad & 4X + Y \leq 60 + 0.001 \\
\text{(Material 2)} \quad & 2Y \leq 48 \\
\text{(Labor)} \quad & X + Y \leq 30 \\
\text{(Non-negativity)} \quad & X, Y \geq 0
\end{aligned}$$

And find that the optimal objective value is $400.0033333\ldots$. Hence, the shadow price is $(400.0033333 - 400)/(0.001) \approx 3.33$.

Moreover, if we repeate the above exercise with $\delta = 1$, we get the optimal objective value of 403.33333, using which we also calculate a ratio of $(403.3333 - 400)/(1) \approx 3.33$, which is the same as before. In fact, in this case we will get the same answer using a δ anywhere between -12 and 60. (So RHS of the constraint is between 48 and 120.) This is called the **allowable range** at which the shadow price is valid. Beyond this range, we do not know from the shadow price alone how the objective value would change with respect to the RHS of the constraint.

Since the allowable range often includes $\delta = 1$, **an approximately correct definition of shadow price (given by DMD) is "the change in the optimal objective value when the RHS of the constraint is increased by one unit."**

Note: You will never be expected to calculate shadow prices by hand. The purpose of the above formula is so that you know what shadow price means, not to expect you to calculate shadow prices using it. The shadow prices are always given by the LP solver. In fact, the Simplex algorithm that Gurobi uses to solve LPs automatically obtains shadow prices for all constraints as part of an intermediate computation.

Interpretation of Shadow Price

The shadow price reveals how incremental changes to the constraint affects the optimal solution. In a production planning LP like the one above, the interpretation is how much benefit is there in one unit of each additional resource. (Hence, it's called a "price" because it's your maximum willingness to pay for an additional resource.)

For example, the shadow prices in the above LP are

	Material 1	Material 2	Labor
Shadow Price	3.33	0	6.67

Since the objective is profit. The interpretations are:

- Additional amounts of material 1 yields additional profit at a rate of \$3.33/unit. (So 1 unit additional would bring \$3.33.)
- Additional amounts of material 2 is not helpful for improving profit. (Since it's not a binding constraint.)
- Additional amounts of labor yields additional profit at a rate of \$6.67/hour.

A non-binding constraint always have a shadow price of zero, because it's not helpful to obtain additional resources if you are not even using what you have already.

In-Class Exercise

There are 2 production plants, A and B, with capacities 20 and 15 respectively. There are 3 demand centers, 1, 2, 3, with demand of 10 each. The cost of transporting each unit of good from each plant to each demand center is shown below.

	1	2	3
A	3	7	5
B	5	3	3

The following LP minimizes total transportation cost subject to satisfying demand at all three centers and not exceeding the capacity of each plant.

Decision variables: x_{A1} is the amount to be shipped from plant A to region 1, x_{A2} is from plant B to region 2, etc.

$$\begin{aligned}
&\text{minimize:} && 3x_{A1} + 7x_{A2} + 5x_{A3} + 5x_{B1} + 3x_{B2} + 3x_{B3} \\
&\text{subject to:} && \\
&\text{(Capacity A)} && x_{A1} + x_{A2} + x_{A3} \leq 20 \\
&\text{(Capacity B)} && x_{B1} + x_{B2} + x_{B3} \leq 15 \\
&\text{(Demand 1)} && x_{A1} + x_{B1} \geq 10 \\
&\text{(Demand 2)} && x_{A2} + x_{B2} \geq 10 \\
&\text{(Demand 3)} && x_{A3} + x_{B3} \geq 10 \\
&\text{(Non-negativity)} && x_{ij} \geq 0 \quad \text{for all } i \in \{A, B\}, j \in \{1, 2, 3\}
\end{aligned}$$

The optimal solution and shadow prices are as follows:

	1	2	3
A	10	0	5
B	0	10	5

	Capacity A	Capacity B	Demand 1	Demand 2	Demand 3
Shadow Price	0	-2	3	5	5

Exercise 1: Write a sentence to interpret the shadow price of each constraint.
Solution:

- Increasing capacity at plant A is not helpful for reducing transportation cost.
- Increasing capacity at plant B would save transportation cost at a rate of 2 per unit of capacity added.
- An increase in demand at center 1 would increase the transportation cost at a rate of 3 per unit of demand added.
- An increase in demand at center 2 would increase the transportation cost at a rate of 5 per unit of demand added.
- An increase in demand at center 3 would increase the transportation cost at a rate of 5 per unit of demand added.

Basic Gurobi Commands for Linear Programming

- `import gurobipy as grb` imports the module and give it a nickname `grb`. (Similar to `import numpy as np`.)
- `mod=grb.Model()` creates an object called `mod` of type Model. This object stores the entire linear program formulation.
- `mod.addVar(lb=0,name='product Y')` creates a new decision variable in the Model `mod`. The optional argument `lb` specifies a lower bound (non-negativity). The `name` argument is for display purposes if you choose to write the model to a file using the command `mod.write('filename.lp')`. For ease of later referencing this decision variable, you can assign it to another variable, as in `Y=mod.addVar(lb=0,name='product Y')`. The type of this variable `Y` is "Var."
- `mod.addConstr(4*X+Y<=60,name='Capacity 1')` creates a new constraint in the Model `mod`. The optional argument `name` is for display purposes as above. For ease of later referencing this constraint (for example to obtain the shadow price), you can assign it to another variable, as in `C=mod.addConstr(4*X+Y<=60,name='Capacity 1')`. The type of this variable `C` is "Constr."
- `mod.setObjective(20*X+10*Y,sense=grb.GRB.MAXIMIZE)` sets the objective. To minimize, replace the `grb.GRB.MAXIMIZE` with `grb.GRB.MINIMIZE`. Note that Python is case sensitive.
- `mod.optimize()` solves the Model `mod`, given the decision variables, constraints, and objectives you gave it.

After running `mod.optimize()`, the outputs can be retrieved if you have references to the variable and constraint objects (this is why it is convenient to assign these to variables, in addition to adding to the Model `mod`).

- `mod.ObjVal` stores the value of the optimal objective for the Model `mod`.
- `Y.VarName` stores the name of the variable you assigned. In the above, it is "product Y."
- `Y.x` stores the optimal value of decision variable `Y`. It's always `.x` regardless of the name of the variable.
- `C.ConstrName` stores the name of the constraint.
- `C.PI` stores the shadow price of constraint `C`.
- `C.SARHSLow` and `C.SARHSUp` stores the allowable range of the RHS of the constraint in which the shadow price is valid. (In the first example of this note, the allowable range is `SARHSLow` would be 48 and `SARHSUp` would be 120.)

There are other attributes you can obtain to get information about the solution. A list of all attributes is here: http://www.gurobi.com/documentation/7.0/refman/attributes.html

Exercise 2: Write the LP formulation corresponding to the following code.

```
[ ]: # Explicitly constructing a simple production planning LP
     import gurobipy as grb
     mod=grb.Model()

     X=mod.addVar(lb=0)
     Y=mod.addVar(lb=0)

     mod.setObjective(30*X+40*Y,sense=grb.GRB.MAXIMIZE)

     mat1=mod.addConstr(2*X+3*Y <=100)
     mat2=mod.addConstr(3*Y<=75)
     labor=mod.addConstr(X+Y<=30)
```

```
# Do not print anything when calling mod.optimize()
mod.setParam('OutputFlag',False)
mod.optimize()

print('Optimal objective: {0:.2f}'.format(mod.ObjVal))
print('Optimal solution:')
print('\tX= {0:.2f}'.format(X.x))
print('\tY= {0:.2f}'.format(Y.x))
print('Shadow prices:')
print('\tMaterial 1: {0:.2f}'.format(mat1.PI))
print('\tMaterial 2: {0:.2f}'.format(mat2.PI))
print('\tLabor: {0:.2f}'.format(labor.PI))
```

Exercise 3: Solve the LP in exercise 1 using Gurobi. (Homework 6, posted on Blackboard, contains more exercises for writing Gurobi code.)