

# Getting Started with Gurobi

Abrèmod Training

March 5, 2020



# Overview

- Linear Programming (LP)
- Solving LPs with Gurobi
  - ▶ Interactive Shell (Python)
  - ▶ .NET Interface
- LP Modeling Techniques
- Multi-Objective Optimization
- Integer Programming
- Performance Tuning
- Column Generation
- Convex Quadratic Programming
- Stochastic Programming

# Abrémod

Abrémod specializes in implementing math programming models to solve business problems. Including business analysis, modeling, and implementation.

- Revenue Management
- Assignment/Scheduling Problems
- Network Optimization

# Gurobi

Gurobi is a state-of-the-art solver for mathematical programming. It includes solvers for the following types of models:

- Linear Programming (LP)
- Mixed-Integer Linear Programming (MILP)
- Quadratic Programming (QP)
- Mixed-Integer Quadratic Programming (MIQP)
- Quadratically Constrained Programming (QCP)
- Mixed-Integer Quadratically Constrained Programming (MIQCP)

Here, “program” does not refer to a computer program but rather a schedule.

- The problems Gurobi solves are all special cases of mathematical programs.
- Before we discuss those special cases in detail, we should understand what a math program looks like and define some related terminology.

# What is a Mathematical Program?

A math program consists of three components:

- Decision Variables (what you control)
- Constraints (rules you must follow)
- Objective Function (what you want to minimize/maximize)

# What is a Mathematical Program?

## Definition

$$\begin{array}{ll}\text{minimize/maximize:} & f(x_1, x_2, \dots, x_n) \\ \text{subject to:} & g_i(x_1, x_2, \dots, x_n) \left\{ \begin{array}{l} \leq \\ \geq \\ = \end{array} \right\} b_i, \quad i = 1, \dots, m \\ & x_j \geq 0, \quad j = 1, \dots, n\end{array}$$

# Math Programming Terminology

- $x_j$  are the *decision variables*.
- $g_i(x_1, x_2, \dots, x_n) \begin{cases} \leq \\ \geq \\ = \end{cases} b_i$  are *structural constraints*.
- $x_j \geq 0$  are *nonnegativity constraints*.
- $f(x_1, \dots, x_n)$  is the *objective function*.
- A *feasible solution*,  $\hat{x} = (\hat{x}_1, \dots, \hat{x}_n)$  satisfies all constraints.
- The *feasible region* is the set of all feasible solutions.
- The objective function ranks the feasible solutions.
- The optimal solution  $x^*$  satisfies  $f(x^*) \leq f(\hat{x})$  for all feasible  $\hat{x}$ .
  - ▶  $x^*$  is feasible itself.



# The Linear Program

$$\begin{aligned} z^* = \min_x \quad & c_1x_1 + c_2x_2 + \cdots + c_nx_n \\ \text{subject to:} \quad & a_{i1}x_1 + a_{i2}x_2 + \cdots + a_{in}x_n \begin{cases} \leq \\ \geq \\ = \end{cases} b_i, \quad i = 1, \dots, m \\ & x_1, x_2, \dots, x_n \geq 0 \end{aligned}$$

In standard matrix form:

$$\begin{aligned} z^* = \min_x \quad & c^T x \\ \text{s.t.} \quad & Ax = b \\ & x \geq 0 \end{aligned}$$

# The Linear Program

$$z^* = \min_x \quad c_1x_1 + c_2x_2 + \cdots + c_nx_n$$

$$\text{subject to:} \quad a_{i1}x_1 + a_{i2}x_2 + \cdots + a_{in}x_n \begin{cases} \leq \\ \geq \\ = \end{cases} b_i, \quad i = 1, \dots, m$$

$$x_1, x_2, \dots, x_n \geq 0$$

- $a_{ij}$ ,  $c_j$ , and  $b_i$  are data.
- Find  $x^*$  satisfying  $c_1x_1^* + \cdots + c_nx_n^* \leq c_1\hat{x}_1 + \cdots + c_n\hat{x}_n$  for all feasible  $\hat{x}$ .
- A linear program (LP) is a special type of math program with:
  - ▶  $f(x_1, \dots, x_n) = c_1x_1 + \cdots + c_nx_n$
  - ▶  $g_i(x_1, \dots, x_n) = a_{i1}x_1 + \cdots + a_{in}x_n, \quad i = 1, \dots, m$

# Linear Programming Axioms

$$z^* = \min_x \quad c_1x_1 + c_2x_2 + \cdots + c_nx_n$$

$$\text{subject to:} \quad a_{i1}x_1 + a_{i2}x_2 + \cdots + a_{in}x_n \left\{ \begin{array}{l} \leq \\ \geq \\ = \end{array} \right\} b_i, \quad i = 1, \dots, m$$

$$x_1, x_2, \dots, x_n \geq 0$$

- Additivity
- Proportionality
- Divisibility
- Certainty

# Linear Programming Transformations

Gurobi will automatically perform the following transformations to get into standard form:

- Maximize by minimizing the negative
  - ▶  $\max_x cx \Leftrightarrow \min_x -cx$
- Add a slack or surplus variable to convert inequality into equality
  - ▶  $ax \geq b \Leftrightarrow ax - s = b, s \geq 0$
  - ▶  $ax \leq b \Leftrightarrow ax + s = b, s \geq 0$
- Write a variable that can be positive or negative as the difference of non-negative variables
  - ▶  $x = x^+ - x^-$
  - ▶  $x^+, x^- \geq 0$

# The Diet Problem

- Suppose you are trying to construct a diet out of a given set of foods, each with a different cost and nutritional composition, and wish to meet some minimum requirements of various nutrients.
- How can you find the combination of foods that meets all the nutrient requirements and minimizes cost?

# The Diet Problem

Let's consider the following sample inputs where there are 5 types of food and 2 nutrient requirements.

Units of nutrients and cost per ounce			
Food type	Iron	Calcium	Cost
1	2	0	20
2	0	1	10
3	3	2	31
4	1	2	11
5	2	1	12

Nutrient requirements: 21 units of iron and 12 units of calcium

# Diet Problem Formulation

- Decision Variables

- $x_j = \#$  of ounces of food type  $j = 1, 2, \dots, 5$

- Objective Function

- $\min z = 20x_1 + 10x_2 + 31x_3 + 11x_4 + 12x_5$

- Structural Constraints

- $2x_1 + 0x_2 + 3x_3 + 1x_4 + 2x_5 \geq 21$
- $0x_1 + 1x_2 + 2x_3 + 2x_4 + 1x_5 \geq 12$

- Nonnegativity constraints

- $x_j \geq 0, j = 1, 2, \dots, 5$

Food	Iron	Calcium	Cost
1	2	0	20
2	0	1	10
3	3	2	31
4	1	2	11
5	2	1	12

Nutrient requirements:  
Iron: 21, Calcium: 12

# Solving with the Gurobi Interactive Shell

- Objects and Methods you will need (.NET equivalent)
  - ▶ Model (GRBModel)
    - ★ addVar(lb, ub, obj, vtype, name)
    - ★ addConstr(constr, name)
    - ★ update()
    - ★ optimize()
  - ▶ Var (GRBVar)
    - ★ X
    - ★ RC
  - ▶ Constr (GRBConstr)
    - ★ Pi
    - ★ Slack



# Querying the Solution

- `Model.write()`
  - ▶ `m.write('diet.lp')` outputs the model in human-readable form
  - ▶ `m.write('diet.mps')` outputs a full-precision copy of the model
  - ▶ `m.write('diet.sol')` outputs the solution
- Model object attributes
  - ▶ `m.NumConstrs`
  - ▶ `m.NumVars`
  - ▶ `m.Status` (was the solver able to find an optimal solution?)
  - ▶ `m.ObjVal` (optimal objective value)

# Querying the Solution

- Var object attributes
  - ▶ `x1.X` - optimal value
  - ▶ `x1.RC` - reduced cost, change in objective/change in variable bound
- Constr object attributes
  - ▶ `iron_constraint.Pi` - shadow price, change in objective/change in RHS
  - ▶ `iron_constraint.Slack` - difference between LHS and RHS

# Updating the Model

- Settable Var object attributes
  - ▶ `x1.LB` - lower bound
  - ▶ `x1.UB` - upper bound
  - ▶ `x1.Obj` - objective coefficient
- Settable Constr object attributes
  - ▶ `iron_con.RHS` - right-hand side constant
- `Model.chgCoeff(constr, var, newvalue)` modifies a coefficient in the constraint matrix

# Diet Problem Implemented in .NET

```
GRBEnv env = new GRBEnv();
GRBModel m = new GRBModel(env);
GRBVar x1 = m.AddVar(0, GRB.INFINITY, 20, GRB.CONTINUOUS, "food.1");
GRBVar x2 = m.AddVar(0, GRB.INFINITY, 10, GRB.CONTINUOUS, "food.2");
GRBVar x3 = m.AddVar(0, GRB.INFINITY, 31, GRB.CONTINUOUS, "food.3");
GRBVar x4 = m.AddVar(0, GRB.INFINITY, 11, GRB.CONTINUOUS, "food.4");
GRBVar x5 = m.AddVar(0, GRB.INFINITY, 12, GRB.CONTINUOUS, "food.5");
m.Update();
GRBConstr con1 = m.AddConstr(2 * x1 + 3 * x3 + 1 * x4 + 2 * x5 >= 21, "nutrient.iron");
GRBConstr con2 = m.AddConstr(1 * x2 + 2 * x3 + 2 * x4 + 1 * x5 >= 12, "nutrient.calcium");
m.Optimize();
foreach (GRBVar var in m.GetVars())
{
    Console.WriteLine("{0} = {1}, reduced cost = {2}", var.Get(GRB.StringAttr.VarName),
        var.Get(GRB.DoubleAttr.X), var.Get(GRB.DoubleAttr.RC));
}

foreach (GRBConstr constr in m.GetConstrs())
{
    Console.WriteLine("{0}, slack = {1}, pi = {2}", constr.Get(GRB.StringAttr.ConstrName),
        constr.Get(GRB.DoubleAttr.Slack), constr.Get(GRB.DoubleAttr.Pi));
}

m.Dispose();
env.Dispose();
```

# .NET Objects and Methods So Far

- `GRBEnv()`
- `GRBModel(GRBEnv env)`
- `GRBModel.AddVar(double lb, double ub, double obj, char type, String name)`
- `GRBModel.AddConstr(GRBLinExpr lhs, char sense, GRBLinExpr rhs, String name)`
- Get and Set methods access attributes of `GRBModel`, `GRBVar`, and `GRBConstr` objects
- Attribute names must be preceded by `GRB.DoubleAttr`, `GRB.IntAttr`, `GRB.CharAttr`, or `GRB.StringAttr`
- `GRBModel.GetVars()`, `GRBModel.GetConstrs()`

# Exercises

- How would the optimal cost change if we forced  $x_1 \geq 0.1$ ?
- How would the optimal cost change if we required an extra 0.1 units of iron?

# Linear Programming Geometry

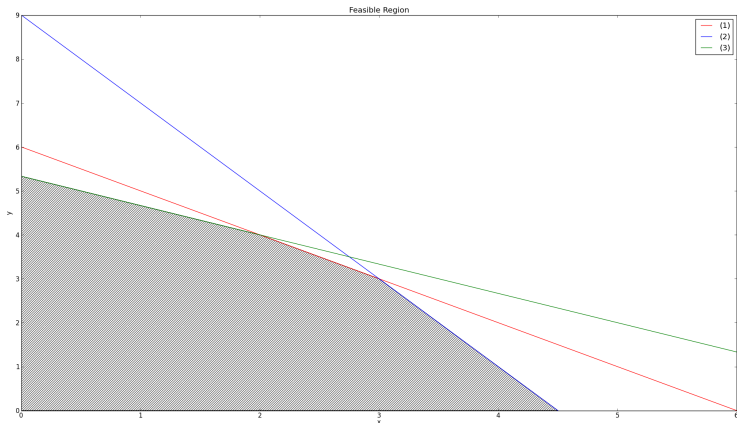
$$\max_{x,y} \quad z = 6x + 4y$$

$$\text{s.t.} \quad x + y \leq 6 \quad (1)$$

$$2x + y \leq 9 \quad (2)$$

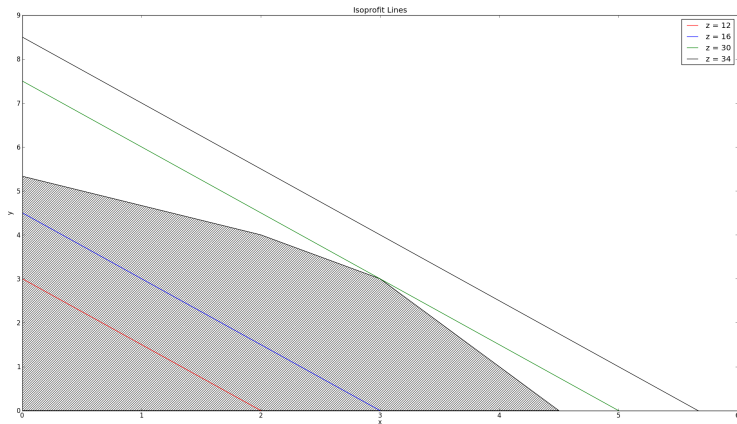
$$2x + 3y \leq 16 \quad (3)$$

$$x, y \geq 0$$



# Linear Programming Geometry

- Plot the objective function  $z = 6x + 4y$  for some fixed values of  $z$ .
- These are the so-called *isoprofit lines* or *objective function contours*.
- Increasing  $z$  results in a parallel shift to the right.





# Observations

- Feasible region of a linear program is always a convex polyhedron
- At least one optimal solution occurs at a corner point (a.k.a. extreme point or vertex) of this polyhedron
- Infinitely-many points in the feasible region, but only finitely many corner points

# Linear Programming Algebra

First, how to solve linear systems of equations?

$$2x_1 + 1x_2 + 1x_3 = 4$$

$$4x_1 - 6x_2 + 0x_3 = 2$$

$$-2x_1 + 7x_2 + 2x_3 = 1$$

Systematically perform row operations to form equivalent systems

$$\text{Row 1} \leftarrow \frac{1}{2} \text{ Row 1}$$

$$\text{Row 2} \leftarrow \text{Row 2} - 4 \text{ Row 1}$$

$$\text{Row 3} \leftarrow \text{Row 3} + 2 \text{ Row 1}$$

$$\vdots$$

Until we arrive at an equivalent system with an obvious solution

$$1x_1 + 0x_2 + 0x_3 = 2$$

$$0x_1 + 1x_2 + 0x_3 = 1$$

$$0x_1 + 0x_2 + 1x_3 = -1$$

# Linear Programming Algebra

- These row operations correspond to multiplying equations by constants and adding the result to other equations.
- “Systematically” performing row operations means picking an equation and solving for a specific variable, then eliminating that variable in all other equations.
- For a square system that has an equal number of variables and equations, it is relatively easy to decide which equation to solve and which variable to solve for.
- If the system has a unique solution, we can solve for the  $i$ th variable in the  $i$ th equation, swapping the order of equations as needed.

# Linear Programming Algebra

Linear programs typically have:

- More variables than equations.
- More than one feasible solution (almost always).
- More than one optimal solution (more often than you might think).

This being said, we can still solve LPs via systematic row operations, but the variable selection step is a little trickier.

# Linear Programming Algebra

$$\begin{array}{ll}\max_{x,y} & 6x + 4y = z \\ \text{s.t.} & x + y \leq 6 \\ & 2x + y \leq 9 \\ & 2x + 3y \leq 16 \\ & x, y \geq 0\end{array}$$

As a system of linear equations:

$$\begin{array}{ll}\max_{x,y,s} & 6x + 4y + 0s_1 + 0s_2 + 0s_3 = z \\ \text{s.t.} & 1x + 1y + 1s_1 + 0s_2 + 0s_3 = 6 \\ & 2x + 1y + 0s_1 + 1s_2 + 0s_3 = 9 \\ & 2x + 3y + 0s_1 + 0s_2 + 1s_3 = 16 \\ & x, y, s_1, s_2, s_3 \geq 0\end{array}$$

Add the objective as the equation  $-z + 6x + 4y = 0$  and write in matrix form:

$$\left[ \begin{array}{cccccc|c} z & x & y & s_1 & s_2 & s_3 & RHS \\ -1 & 6 & 4 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 6 \\ 0 & 2 & 1 & 0 & 1 & 0 & 9 \\ 0 & 2 & 3 & 0 & 0 & 1 & 16 \end{array} \right]$$

# Linear Programming Algebra

$$\left[ \begin{array}{cccccc|c} z & x & y & s_1 & s_2 & s_3 & RHS \\ -1 & 6 & 4 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 6 \\ 0 & 2 & 1 & 0 & 1 & 0 & 9 \\ 0 & 2 & 3 & 0 & 0 & 1 & 16 \end{array} \right]$$

Systematically perform row operations

$$\left[ \begin{array}{cccccc|c} z & x & y & s_1 & s_2 & s_3 & RHS \\ -1 & 0 & 1 & 0 & -3 & 0 & -27 \\ 0 & 0 & 1/2 & 1 & -1/2 & 0 & 3/2 \\ 0 & 1 & 1/2 & 0 & 1/2 & 0 & 9/2 \\ 0 & 0 & 2 & 0 & -1 & 1 & 7 \end{array} \right]$$

Until the solution is obvious

$$\left[ \begin{array}{cccccc|c} z & x & y & s_1 & s_2 & s_3 & RHS \\ -1 & 0 & 0 & -2 & -2 & 0 & -30 \\ 0 & 0 & 1 & 2 & -1 & 0 & 3 \\ 0 & 1 & 0 & -1 & 1 & 0 & 3 \\ 0 & 0 & 0 & -4 & 1 & 1 & 1 \end{array} \right]$$

What is the obvious solution? This is the equivalent LP:

$$\begin{array}{ll} \max_{x,y,s} & 0x + 0y - 2s_1 - 2s_2 + 0s_3 + 30 = z \\ \text{s.t.} & 0x + 1y + 2s_1 - 1s_2 + 0s_3 = 3 \\ & 1x + 0y - 1s_1 + 1s_2 + 0s_3 = 3 \\ & 0x + 0y - 4s_1 + 1s_2 + 1s_3 = 1 \\ & x, y, s_1, s_2, s_3 \geq 0 \end{array}$$

The optimal solution to this transformed LP is  $(x, y, s_1, s_2, s_3) = (3, 3, 0, 0, 1)$ ,  $z^* = 30$

# Linear Programming Algebra

In more detail

$$\left[ \begin{array}{cccccc|c} z & x & y & s_1 & s_2 & s_3 & RHS \\ -1 & 6 & 4 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 6 \\ 0 & 2 & 1 & 0 & 1 & 0 & 9 \\ 0 & 2 & 3 & 0 & 0 & 1 & 16 \end{array} \right]$$

Feasible solution  $(x, y, s_1, s_2, s_3) = (0, 0, 6, 9, 16)$ ,  $z = 0$ . Increase  $x$  since it has a positive coefficient.

$$\left[ \begin{array}{cccccc|c} z & x & y & s_1 & s_2 & s_3 & RHS \\ -1 & 0 & 1 & 0 & -3 & 0 & -27 \\ 0 & 0 & 1/2 & 1 & -1/2 & 0 & 3/2 \\ 0 & 1 & 1/2 & 0 & 1/2 & 0 & 9/2 \\ 0 & 0 & 2 & 0 & -1 & 1 & 7 \end{array} \right]$$

Feasible solution  $(x, y, s_1, s_2, s_3) = (9/2, 0, 3/2, 0, 7)$ ,  $z = 27$ . Increase  $y$  since it has a positive coefficient.

$$\left[ \begin{array}{cccccc|c} z & x & y & s_1 & s_2 & s_3 & RHS \\ -1 & 0 & 0 & -2 & -2 & 0 & -30 \\ 0 & 0 & 1 & 2 & -1 & 0 & 3 \\ 0 & 1 & 0 & -1 & 1 & 0 & 3 \\ 0 & 0 & 0 & -4 & 1 & 1 & 1 \end{array} \right]$$

Feasible solution  $(x, y, s_1, s_2, s_3) = (3, 3, 0, 0, 1)$ ,  $z = 30$ .

Transformed objective has no positive coefficients.

Transformed constraints have an "obvious" solution in which all variables with a negative objective coefficient are zero.

This is a provably optimal solution.

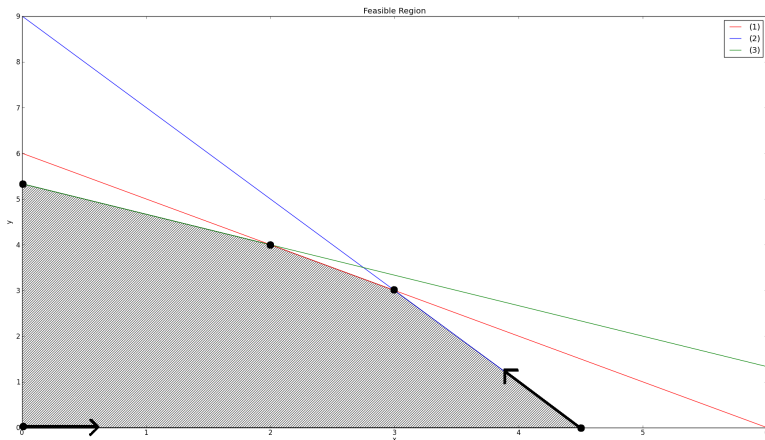
# Observations

- Each iteration maintained exactly 3 positive decision variables (one for each of the original structural constraints).
- The set of positive variables is *basis* (and the associated solution is a *basic feasible solution*).
- Each iteration adds a new variable to a basis, and kicks an old variable out.
- The objective improves at each iteration.
- Proof of optimality: transformed objective coefficients are zero for basic variables, non-positive for non-basic variables.
- What would change if we perturbed the original
  - ▶ objective function coefficients?
  - ▶ right-hand sides?



# Connecting the Algebra to the Geometry

We iterated over basic feasible solutions  $(0, 0, 6, 9, 16)$ ,  $(9/2, 0, 3/2, 0, 7)$ , and  $(3, 3, 0, 0, 1)$ . Plotting these points in  $(x, y)$  space...



# Exercises

- Solve the preceding model with Gurobi.
  - ▶ Note: Set the model attribute ModelSense to -1 in order to maximize
- Which constraints are tight at the optimal solution?
- Besides the origin, the feasible region has three other extreme points that are suboptimal under the current objective function. How might you change the objective function coefficients so that:
  - ▶ the extreme point at  $(0, \frac{16}{3})$  is optimal?
  - ▶ the extreme point at  $(\frac{9}{2}, 0)$  is optimal?
  - ▶ all points between  $(2, 4)$  and  $(3, 3)$  are optimal?

# Possible Outcomes of an LP

- Infeasible - feasible region is empty, i.e.,  $x_1 \geq 0$ ,  $x_1 \leq -1$
- Unbounded - no finite optimum, i.e.  $\max 15x_1$  subject to  $x_1 \geq 0$
- Multiple optima, i.e.  $\max 3x_1 + 3x_2$  subject to  $x_1 + x_2 \leq 1$  and non-negative
- Unique optimal solution (as in previous example)

# Linear Programming Solvers

- Finds **an** optimal solution if feasible region is non-empty and objective function is bounded
- Might be multiple optimal solutions
  - ▶ Which optimal solution returned is not defined.
- Constraints are “hard”
  - ▶ Won't violate constraints <sup>1</sup> even if it helps the objective.
  - ▶ Will tell if relaxing a constraint would help the objective.
- Solver makes no apologies for this behavior!
- Includes proof of optimality

---

<sup>1</sup>by more than the numerical tolerance

# Real Linear Programming Solvers

## Strengths

- Can Solve very large problems in Practice
  - ▶ Tuned for real-world business problems
  - ▶ Routinely solve problems with  $10^7$  variables and constraints
  - ▶ Multiple Algorithms (controlled by parameter Method)

## Weaknesses

- Don't guarantee a time to a solution
- Might not reach optimality (but will tell you if it doesn't)
- Work with floating point values
  - ▶ Might violate constraints by a small tolerance (controlled by parameter FeasibilityTol)
  - ▶ Might return a solution that is within some numerical tolerance of optimal (controlled by parameter OptimalityTol)

The better the linear programming solver, the less of an issue you will have with these realities.

# Duality

The diet problem, revisited:

- Let's take the perspective of a supplement vendor, who has pills that contain a single unit of iron or calcium that can be used to replace meals.
- This vendor will attempt sell these pills to a dieter, and must determine the appropriate price to offer.
- We'll assume the dieter knows how to solve the diet problem and will replace his optimal diet with pills but only if his cost does not increase.
- How does the vendor determine pill prices that maximize revenue and are competitive with the food types?

# Duality

Food	Iron	Calcium	Cost
1	2	0	20
2	0	1	10
3	3	2	31
4	1	2	11
5	2	1	12

Nutrient requirements:  
Iron: 21, Calcium: 12

- Let  $\pi_i, \pi_c$  be the price to be charged for an iron, calcium pill.
- We wish to maximize total revenue of  $v = 21\pi_i + 12\pi_c$ .
- We must charge prices that are competitive with the prices of the five food types.
  - ▶  $2\pi_i \leq 20$
  - ▶  $\pi_c \leq 10$
  - ▶  $3\pi_i + 2\pi_c \leq 31$
  - ▶  $\pi_i + 2\pi_c \leq 11$
  - ▶  $2\pi_i + \pi_c \leq 12$

# Duality

The diet problem:

$$\begin{aligned} z^* = \min_x \quad & 20x_1 + 10x_2 + 31x_3 + 11x_4 + 12x_5 \\ \text{s.t.} \quad & 2x_1 + 0x_2 + 3x_3 + 1x_4 + 2x_5 \geq 21 \\ & 0x_1 + 1x_2 + 2x_3 + 2x_4 + 1x_5 \geq 12 \\ & x_j \geq 0, \quad j = 1, 2, \dots, 5 \end{aligned}$$

The “dual” problem:

$$\begin{aligned} v^* = \max_{\pi} \quad & 21\pi_i + 12\pi_c \\ \text{s.t.} \quad & 2\pi_i + 0\pi_c \leq 20 \\ & 0\pi_i + 1\pi_c \leq 10 \\ & 3\pi_i + 2\pi_c \leq 31 \\ & 1\pi_i + 2\pi_c \leq 11 \\ & 2\pi_i + 1\pi_c \leq 12 \\ & \pi_i, \pi_c \geq 0 \end{aligned}$$



# Exercises

- Intuitively, is it possible for  $v^* > z^*$ ?
- Solve the dual of the diet problem with Gurobi. (Maintain a copy of the Model object for the original diet problem for comparison purposes.)
- How are  $z^*$  and  $v^*$  related?
- How are  $\pi_i^*$  and  $\pi_c^*$  related to solution of the original diet problem?
- Multiply the iron constraint in the original diet problem by  $\pi_i^*$ , the calcium constraint by  $\pi_c^*$ , and add the results.
  - ▶ What is the resulting inequality?
  - ▶ How can this inequality be used to prove optimality?

# Computing Shadow Prices

Gurobi computes  $\pi$  for us even when we solve the primal. Recall that in the optimal solution to the diet problem example, only  $x_4$  and  $x_5$  were non-zero. Letting  $b_i$  and  $b_c$  be nutrient requirements, we have

$$x_4 + 2x_5 = b_i$$

$$2x_4 + x_5 = b_c$$

We can solve for  $x_4$  and  $x_5$  as

$$x_4 = -1/3b_i + 2/3b_c$$

$$x_5 = 2/3b_i - 1/3b_c$$

Plugging into the objective, we get

$$\begin{aligned} z &= 11x_4 + 12x_5 \\ &= 11(-1/3b_i + 2/3b_c) + 12(2/3b_i - 1/3b_c) \\ &= 13/3b_i + 10/3b_c \end{aligned}$$

So,  $\pi_i = 13/3$  and  $\pi_c = 10/3$ .

# Computing Reduced Costs

- How are the reduced costs related to the shadow prices?
- Consider food 3, which costs 31 per ounce and provides 3 units of iron and 2 units of calcium.
- Iron is priced at  $13/3$  per unit, calcium at  $10/3$  per unit.
- If we discount the cost of food 3 by the value of the nutrients that it provides, we get  $31 - 3 * (13/3) - 2 * (10/3) = 34/3$ , which is exactly the reduced cost.
- What is the reduced cost for food 4?
- How cheap would food type 3 need to be in order for it to be in our diet?
- Suppose we introduce a new food that costs 20 per ounce and provides 2 units of iron and 3 units of calcium. Should we include this new food in our diet? Do we need to reoptimize?

# Notation

Linear Programming involves exclusively addition and multiplication by constants. The symbols  $\in$  and  $\notin$  is read as “in” and “not in”. If  $I = \{1, 2, 3\}$ .

$$\sum_{i \in I} a_i x_i$$

$$a_1 x_1 + a_2 x_2 + a_3 x_3$$

$$x_i \leq b_i \quad \forall i \in I$$

$$x_1 \leq b_1$$

$$x_2 \leq b_2$$

$$x_3 \leq b_3$$

# Knapsack Problem

- Set of items  $I = \{1, \dots, n\}$ .
- Each item has value  $v_i$  and weight  $w_i$ .
- Have a knapsack that with capacity  $b$ .
- Can take part of any item.
- What is the highest valued collection of items (and partial items) that can go into the knapsack?

$$\begin{aligned} \max_x \quad & \sum_{i \in I} v_i x_i \\ \text{s.t.} \quad & \sum_{i \in I} w_i x_i \leq b \\ & 0 \leq x_i \leq 1, \quad i \in I \end{aligned}$$

- Use a GRBLinExpr to build up the sum in the constraint.

# .NET Implementation

```
int capacity = ...;int[] weights = ...;int[] profits = ...;
int nItems = weights.Length;

GRBVar[] itemSelected = new GRBVar[nItems];
GRBLinExpr totalKnapsackWeight = new GRBLinExpr();
for (int i = 0; i < nItems; ++i) {
    itemSelected[i] = m.AddVar(0, 1, profits[i], GRB.CONTINUOUS, "item_selected." + i);
    totalKnapsackWeight.AddTerm(weights[i], itemSelected[i]);
}
m.Update();
GRBConstr con = m.AddConstr(totalKnapsackWeight, 'L', capacity, "knapsack");
m.Set(GRB.IntAttr.ModelSense, GRB.MAXIMIZE);

m.Update();
m.Write("knapsack.lp");
m.Optimize();

if (m.get(GRB.IntAttr.Status) == GRB.OPTIMAL) {
    foreach (GRBVar var in m.GetVars()) {
        Console.WriteLine("{0} = {1}, reduced cost = {2}", var.Get(GRB.StringAttr.VarName),
            var.Get(GRB.DoubleAttr.X), var.Get(GRB.DoubleAttr.RC));
    }
    foreach (GRBConstr constr in m.GetConstrs()) {
        Console.WriteLine("{0}, slack = {1}, pi = {2}", constr.get(GRB.StringAttr.ConstrName),
            constr.get(GRB.DoubleAttr.Slack), constr.get(GRB.DoubleAttr.Pi));
    }
}
```

# The Diet Problem Generalized

- Sets and Indices
  - ▶  $i \in I$ : nutrients
  - ▶  $j \in J$ : food types
- Data
  - ▶  $c_j$ : per ounce cost of food type  $j$
  - ▶  $a_{ij}$ : quantity of nutrient  $i$  per ounce of food type  $j$
  - ▶  $l_i, u_i$ : min, max daily requirements for nutrient  $i$
- Decision Variables
  - ▶  $x_j$ : the number of ounces to consume of food type  $j$ .
- Formulation?
- Let  $\pi_i$  be the shadow price for nutrient  $i$ . What is the reduced cost of food type  $j$ , in terms of  $\pi$ ?

## .NET Implementation

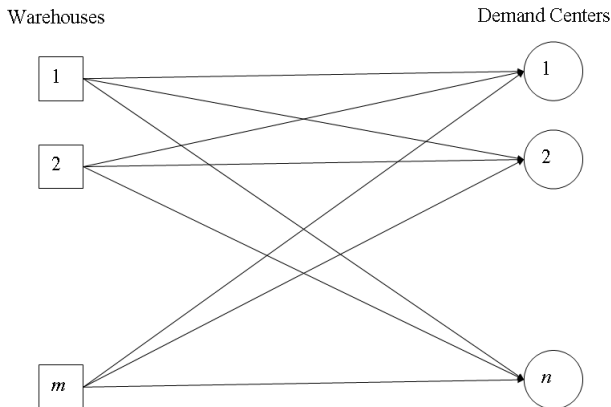
Add ranged constraints to GRBModel via `GRBModel.addRange(expr, lower, upper, name)`

For example,  $l_j \leq \sum_{i \in I} a_{ij} x_i \leq u_j, \quad j \in J$  can be created via:

```
for (int j = 0; j < J; ++j) {  
    GRBLinExpr nutrientConsumption = new GRBLinExpr();  
    for (int i = 0; i < I; ++i)  
        nutrientConsumption.AddTerm(a[i][j], x[i]);  
    model.AddRange(nutrientConsumption, l[j], u[j],  
        "nutrient_consumption." + j);  
}
```



# Transportation Problem



Input:

Warehouse capacity  $u_i$  (widgets)

Customer demand  $d_j$  (widgets)

Shipping cost  $c_{ij}$  (\$/widget)

# Transportation Problem

- Sets and Indices

- ▶  $i \in I$ : Warehouses
- ▶  $j \in J$ : Customers

- Data

- ▶  $u_i$ : capacity for warehouse  $i$  (widgets)
- ▶  $d_j$ : demand at demand center  $j$  (widgets)
- ▶  $c_{ij}$ : shipping cost from warehouse  $i$  to customer  $j$  (\$/widget)

- Decision Variables

- ▶  $x_{ij}$ : number of widgets to ship from warehouse  $i$  to customer  $j$

# LP Formulation

$$\min_x \quad \sum_{i \in I} \sum_{j \in J} c_{ij} x_{ij} \quad (\text{minimize shipping costs})$$

$$\text{s.t.} \quad \sum_{i \in I} x_{ij} = d_j, \quad j \in J \quad (\text{satisfy demand})$$

$$\sum_{j \in J} x_{ij} \leq u_i, \quad i \in I \quad (\text{don't exceed capacity})$$

$$x_{ij} \geq 0, \quad i \in I, j \in J \quad (\text{ship nonnegative quantities})$$

# .NET Implementation

Use GRBLinExpr to build up the summations in the constraints.  
Assume  $x$  is a 2d array of GRBVar and has already been populated.  
Demand constraints  $\sum_{i \in I} x_{ij} = d_j$ ,  $j \in J$  are built via:

```
for (int j = 0; j < J; ++j) {  
    GRBLinExpr unitsShippedToJ = new GRBLinExpr();  
    for (int i = 0; i < I; ++i)  
        unitsShippedToJ.AddTerm(1.0, x[i][j]);  
    model.AddConstr(unitsShippedToJ, GRB.EQUAL, Demand[j], "Demand." + j);  
}
```

Capacity constraints  $\sum_{j \in J} x_{ij} \leq u_i$ ,  $i \in I$  are built via:

```
for (int i = 0; i < I; ++i) {  
    GRBLinExpr unitsShippedFromI = new GRBLinExpr();  
    for (int j = 0; j < J; ++j)  
        unitsShippedFromI.AddTerm(1.0, x[i][j]);  
    model.AddConstr(unitsShippedFromI, GRB.LESS_EQUAL, Capacity[i], "Capacity." + i);  
}
```

# Diagnosing Infeasibility

- After optimization, always check the Model attribute Status to determine whether the model was solved to optimality.
- If Status is `GRB.Status.Infeasible`, how to diagnose infeasibility?
- `GRBModel.ComputeIIS()` computes an irreducible inconsistent subsystem.
- Pass `GRBModel.Write()` a filename with suffix `.ilp` to write the IIS to a file.
- `GRBVar` attributes `IISLB`, `IISUB` indicate which variable bounds participate in the IIS.
- `GRBConstr` attributes `IISConstr` indicate which constraints participate in the IIS.

# Exercise

- Implement the transportation model with Gurobi.
- Under what conditions would the problem become infeasible?
- Create an infeasible instance, compute an IIS, and write it to a file.

# Elasticizing Constraints

To extend the transportation problem allow demand to go unsatisfied at a per-unit penalty of  $\rho$  replace the demand constraint with

$$\sum_{i \in I} x_{ij} = d_j - y_j, \quad j \in J,$$

where  $y_j \geq 0$ , and add  $\rho \sum_{j \in J} y_j$  to the objective.

# Piecewise Linear Penalties

Penalize the first 20% of demand shortfall at a rate  $\rho$ , and any additional demand shortfall at a rate  $1.5\rho$ .

$$\sum_{i \in I} x_{ij} = d_j - y_j^1 - y_j^2, \quad j \in J$$

$$0 \leq y_j^1 \leq 0.2d_j, \quad j \in J$$

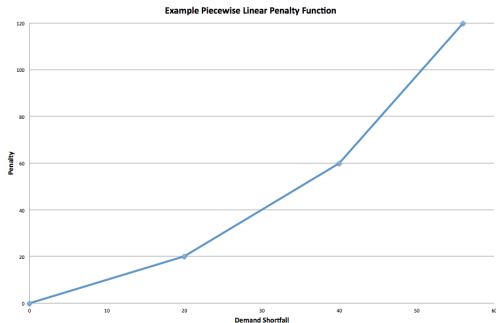
$$0 \leq y_j^2, \quad j \in J,$$

and add  $\rho \sum_{j \in J} (y_j^1 + 1.5y_j^2)$  to the objective.



# Piecewise Linear Penalties

- Starting with 6.0, Gurobi provides a method `Model.setPWLObj`.



- The above penalty function can be created in one call:
  - `model.setPWLObj(var, [0, 20, 40, 56], [0, 20, 60, 120])`
- No auxiliary variables are required.
- Note: If the objective function is not convex, the resulting model will be an Integer Program.

# Minimize the Maximum Demand Shortfall

- Recall:  $\sum_{i \in I} x_{ij} = d_j - y_j, \quad j \in J$
- $y_j$  is the demand shortfall at demand center  $j$ .
- Suppose we want to control  $z = \max\{y_1, y_2, \dots, y_n\}$ .
- Let  $z \geq y_j, \quad j \in J$ .
- Penalize  $z$  in the objective, or put an upper bound on  $z$ .
- Only works if we are trying to minimize  $z$ , otherwise we require integer variables.
- Extends to any problem involving minimization (maximization) of the maximum (minimum) of several linear functions.
- $y = |x|$  can be linearized as  $y \geq x, y \geq -x$  (assuming minimization of  $y$ ).

# Multiple Objectives

$$\begin{aligned} \min_x \quad & \sum_{i \in I} \sum_{j \in J} c_{ij} x_{ij} + \rho \sum_{j \in J} y_j \\ \text{s.t.} \quad & \sum_{i \in I} x_{ij} = d_j - y_j, \quad j \in J \\ & \sum_{j \in J} x_{ij} \leq u_i, \quad i \in I \\ & x_{ij} \geq 0, \quad i \in I, j \in J \\ & y_j \geq 0, \quad j \in J \end{aligned}$$

- Objectives:
  - ▶ Minimize transportation cost  $\sum_{i \in I} \sum_{j \in J} c_{ij} x_{ij}$
  - ▶ Minimize demand shortfall  $\sum_{j \in J} y_j$
- How do we choose  $\rho$ ?

# Multiple Objectives

- Generate an *efficient frontier* of solutions.
- Let  $z = \sum_{j \in J} y_j$ .
- Add  $\rho z$  to the objective.
- Optimize with  $\rho = 0$ .
- Query GRBVar attribute SAObjUp.
- Reoptimize with  $\rho = (1 + \epsilon)\text{SAObjUp}$ .
- LPs are cheap to reoptimize!

# Exercise

Generate the efficient frontier of the transportation problem.

# Primary and Secondary Objectives

- Optimize primary objective first.
- Constrain primary objective to be within some tolerance of optimal.
- Optimize secondary objective.

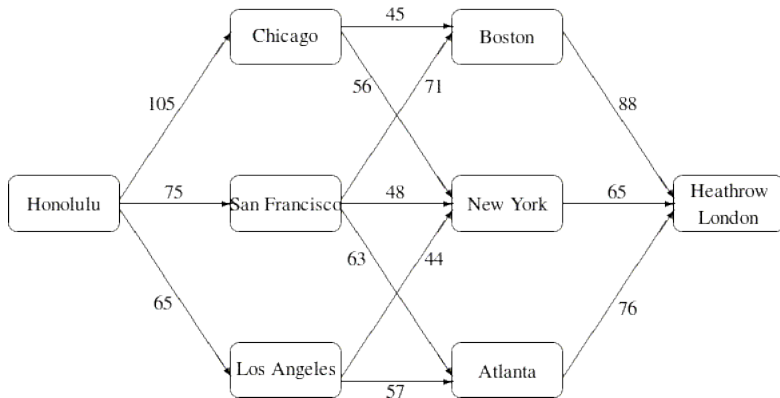
```
GRBLinExpr primary = new GRBLinExpr();  
GRBLinExpr secondary = new GRBLinExpr();  
// build up objective functions  
m.SetObjective(primary);  
m.Optimize()  
double objValue = model.Get(GRB.DoubleAttr.ObjVal);  
m.AddConstr(primary, 'L', (1 + EPS)*objValue, "PrimaryObj");  
m.SetObjective(secondary);  
m.Optimize();
```

# Best Modeling Practices

- Allow objectives to become constraints and vice versa
- Multiple optima are the norm, not the exception
- Exploit this with tie-breakers (secondary, tertiary objectives)
- Elasticize constraints

# Shortest Path

- Let  $N$  be a set of cities.
- Let  $A$  be a set of arcs between cities.
- Let  $d_{ij}$  be the distance between city  $i \in N$  and city  $j \in N$ .
- What is the shortest distance between a given origin city  $s \in N$  and destination city  $t \in N$ ?





# Shortest Path

- Let  $\pi_j$  be the length of the shortest path from the origin to city  $j$ .
- Can write  $\pi_j = \min_{(i,j) \in A} d_{ij} + \pi_i$  for every  $j \in N$ .
- Linearize, and maximize  $\pi_t$  to compute length of shortest path.
- Tight constraints indicate arcs on the shortest path.

$$\begin{array}{ll}\max_{\pi} & \pi_t \\ \text{s.t.} & \pi_j \leq \pi_i + d_{ij}, \quad (i,j) \in A \\ & \pi_s = 0\end{array}$$

# Shortest Path

Let  $x_{ij} = 1$  if arc  $(i, j)$  is traversed.

$$\begin{aligned} \min_x \quad & \sum_{(i,j) \in A} d_{ij} x_{ij} \\ \text{s.t.} \quad & \sum_{j \in RS(i)} x_{ji} - \sum_{j \in FS(i)} x_{ij} = b_i, \quad i \in N \\ & 0 \leq x_{ij} \leq 1, \quad (i, j) \in A, \end{aligned}$$

where  $FS(i) = \{j | (i, j) \in A\}$ ,  $RS(i) = \{j | (j, i) \in A\}$ ,  $b_s = -1$ ,  $b_t = 1$ , and  $b_i = 0$  for  $i \neq s, t$ .

# Exercise

Implement both shortest path formulations. How are the two formulations related?