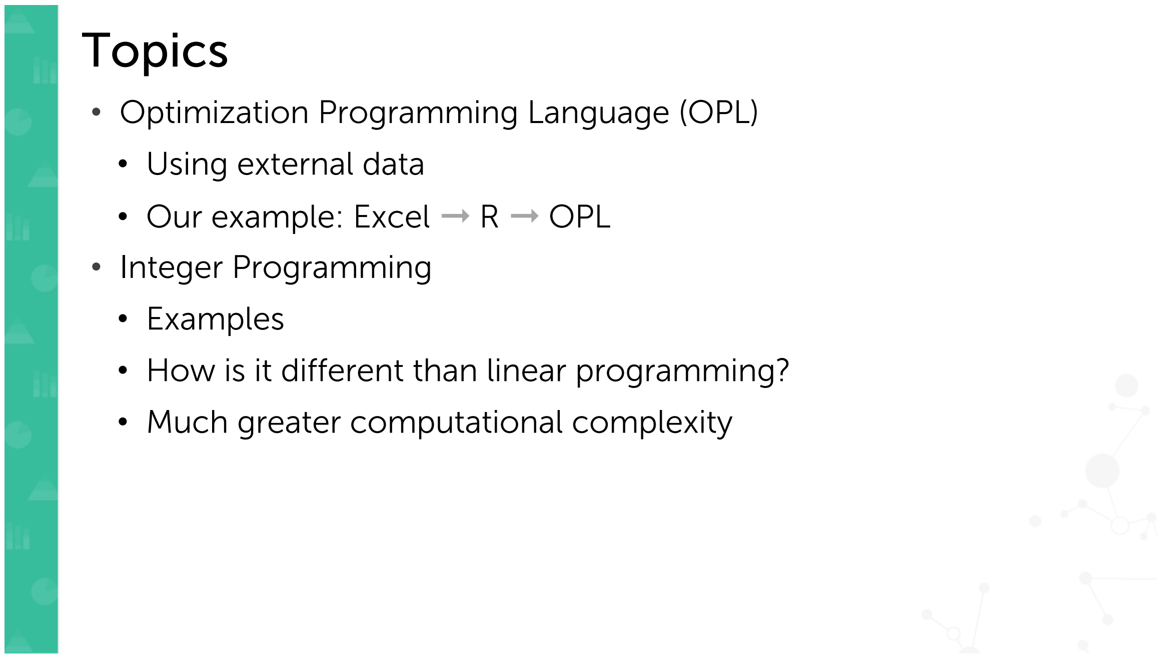
UNIVERSITY OF WISCONSIN
DATA SCIENCE

DS 775

Prescriptive Analytics

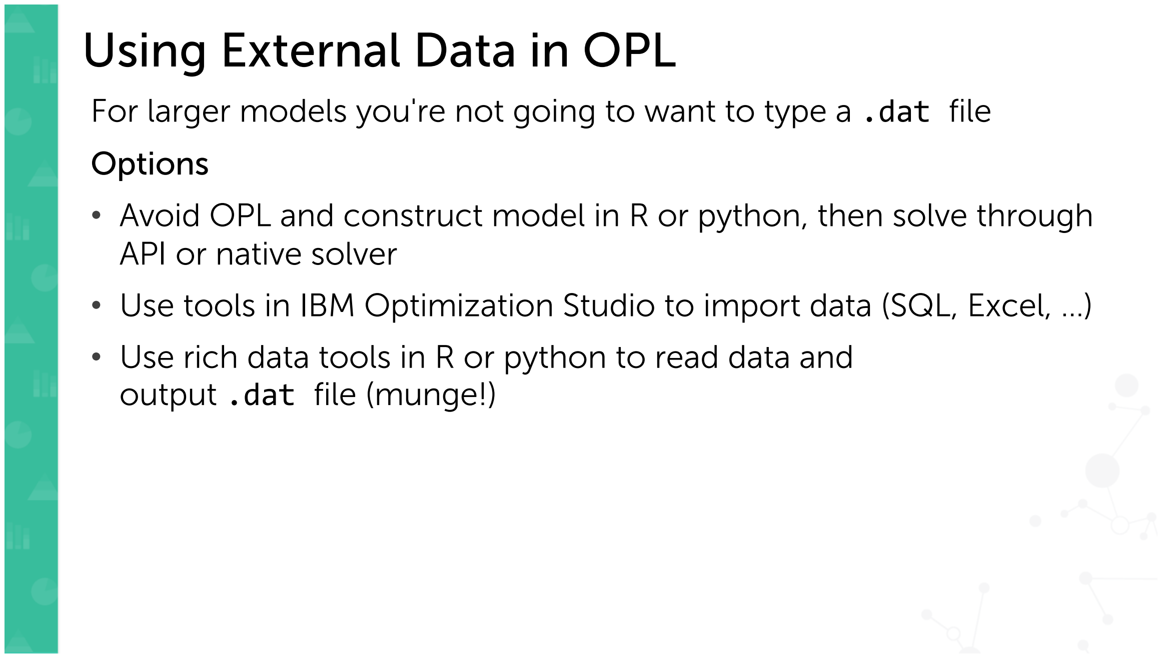
More about OPL and Integer Programming



Topics

- Optimization Programming Language (OPL)
  - Using external data
  - Our example: Excel → R → OPL
- Integer Programming
  - Examples
  - How is it different than linear programming?
  - Much greater computational complexity

Continuing on from our discussion of Opal last week, we'll look at an example of using external data to build a model in Opal. Our solution isn't elegant. But it's typical of the work a data scientist sometimes has to do. We'll also briefly look at integer programming. When the decision variables are required to have integer values instead of continuous values, you might think this would be easier. But integer programming can be much more difficult than linear programming.



Using External Data in OPL

For larger models you're not going to want to type a `.dat` file

Options

- Avoid OPL and construct model in R or python, then solve through API or native solver
- Use tools in IBM Optimization Studio to import data (SQL, Excel, ...)
- Use rich data tools in R or python to read data and output `.dat` file (munge!)

There are many ways to interact with the solvers in IBM Optimization Studio. But I'd prefer a solution which allows us to continue the paradigm of separating the model and the data. We'll use the last option listed on the slide when we will read our model data into R and then use scripting in R to output a data file that can be used by an Opal model.

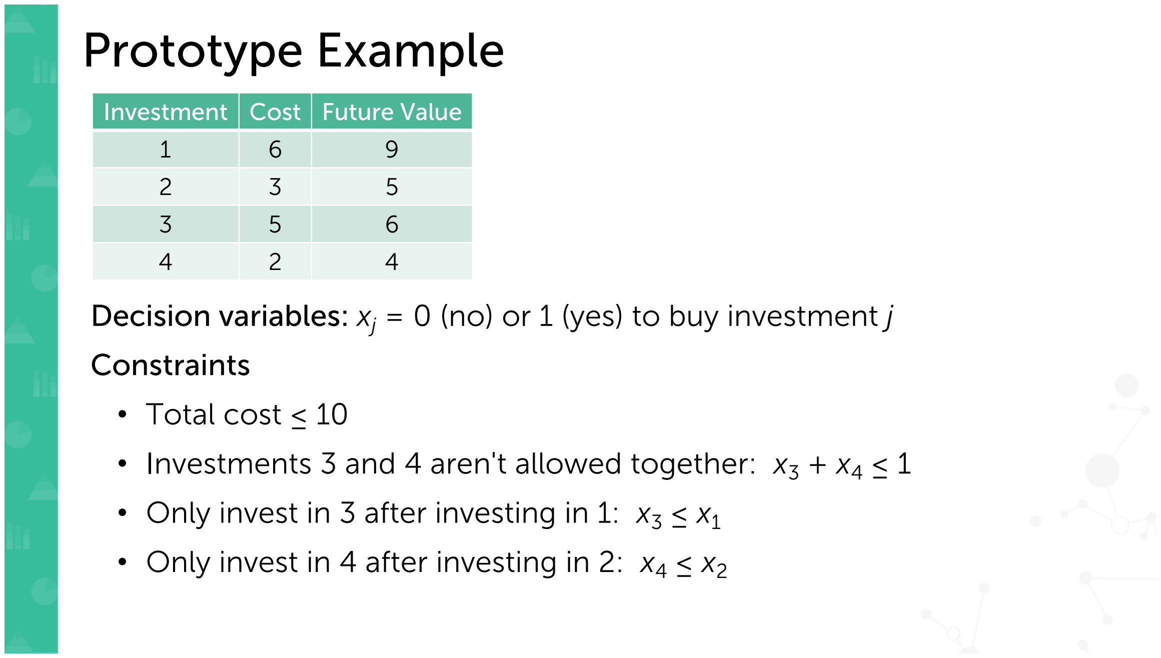
Munging Example

- Use R package XLconnect to read data from Excel spreadsheet into dataframes
- Use R to write text `.dat` file
- Run in IBM optimization studio
- Example write `transp4.dat` file we saw last week
  - See `transpsheet.xlsx` and `transpWrite.R` that from this week's download folder

Munging is data science slang for crudely transforming data which is exactly what we're doing here. I sometimes think that 90% of data analysis is preparing the data. You'll need to do something similar for your homework. The easiest way to study this example is to create a working directory for the download files and point R to that directory. You'll have to modify the path that is set at the top of the R file. Run the R script in R and see the data file that is produced. You may have to install the Excel Connect package first. The data file produced should be the same as the [? transpf4.dat ?] file we used in last week's transportation example.

Integer Programming

- Decision variable constrained to integer values
- Can produce 5 or 6 cars, but not 5.72 cars
- Often have binary (boolean) variables
  - `0` for no
  - `1` for yes



Prototype Example

Investment	Cost	Future Value
1	6	9
2	3	5
3	5	6
4	2	4

Decision variables: x_j = 0 (no) or 1 (yes) to buy investment j

Constraints

- Total cost $\leq$ 10
- Investments 3 and 4 aren't allowed together: $x_3 + x_4 \leq 1$
- Only invest in 3 after investing in 1: $x_3 \leq x_1$
- Only invest in 4 after investing in 2: $x_4 \leq x_2$

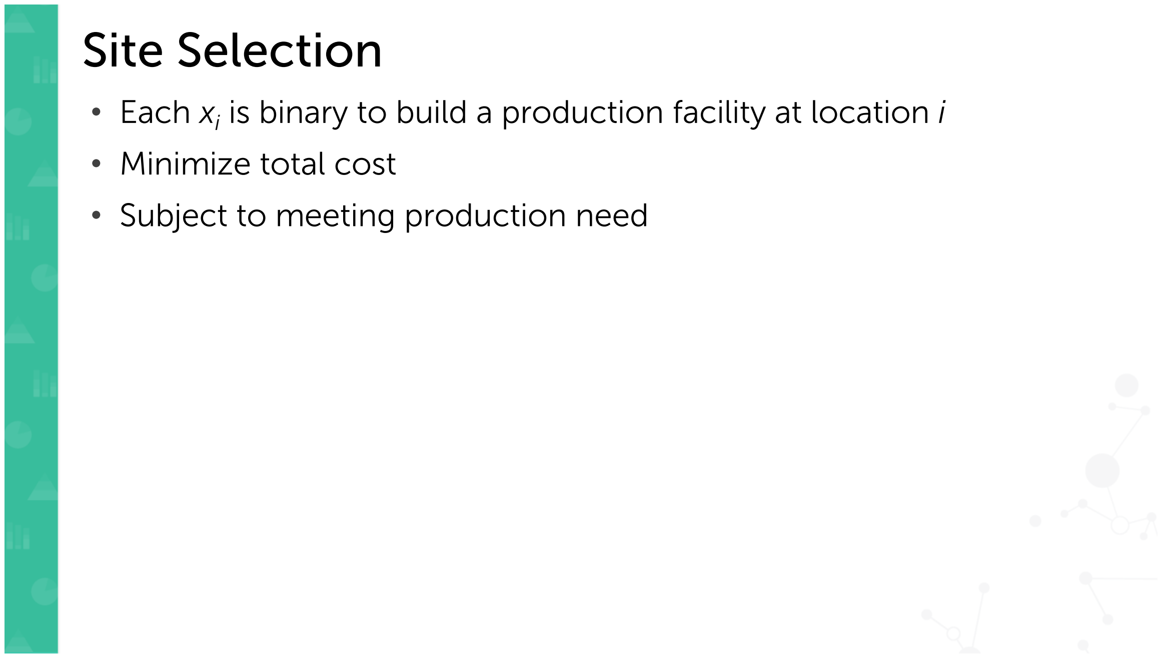
Suppose your company is trying to decide how to spend its budget of 10 units on four investments. We have four binary decision variables. And we want to maximize the future value. Investment three is something that only makes sense if we invest in one also. The same is true of investments four and two. Investments three and four should not occur together. You can read more about this prototype example in the textbook. The next slide shows the mathematical formulation.

Prototype Formulation

Maximize $Z = 9x_1 + 5x_2 + 6x_3 + 4x_4$ subject to:

$$\begin{array}{rcrcrcrcr}
6x_1 & + & 3x_2 & + & 5x_3 & + & 2x_4 & \leq & 10 \\
 & & & & x_3 & + & x_4 & \leq & 1 \\
-x_1 & & & + & x_3 & & & \leq & 0 \\
 & & -x_2 & & & + & x_4 & \leq & 0
\end{array}$$

and each x_j has to be 0 or 1.



Site Selection

- Each x_i is binary to build a production facility at location i
- Minimize total cost
- Subject to meeting production need

Another common example of integer programming is where to locate factories or something similar. We could simply minimize the total cost. But we might also wish to incorporate distribution costs, cost of materials, and labor, et cetera. Next we'll turn to some different ways in which binary variables can be used in linear and integer programs.

Either-or Constraints

Suppose we want one or the other or perhaps both of these:

$$3x_1 + 2x_2 \leq 18$$
$$x_1 + 4x_2 \leq 16$$

Introdue an extra binary variable y and let M be a very large number, then use:

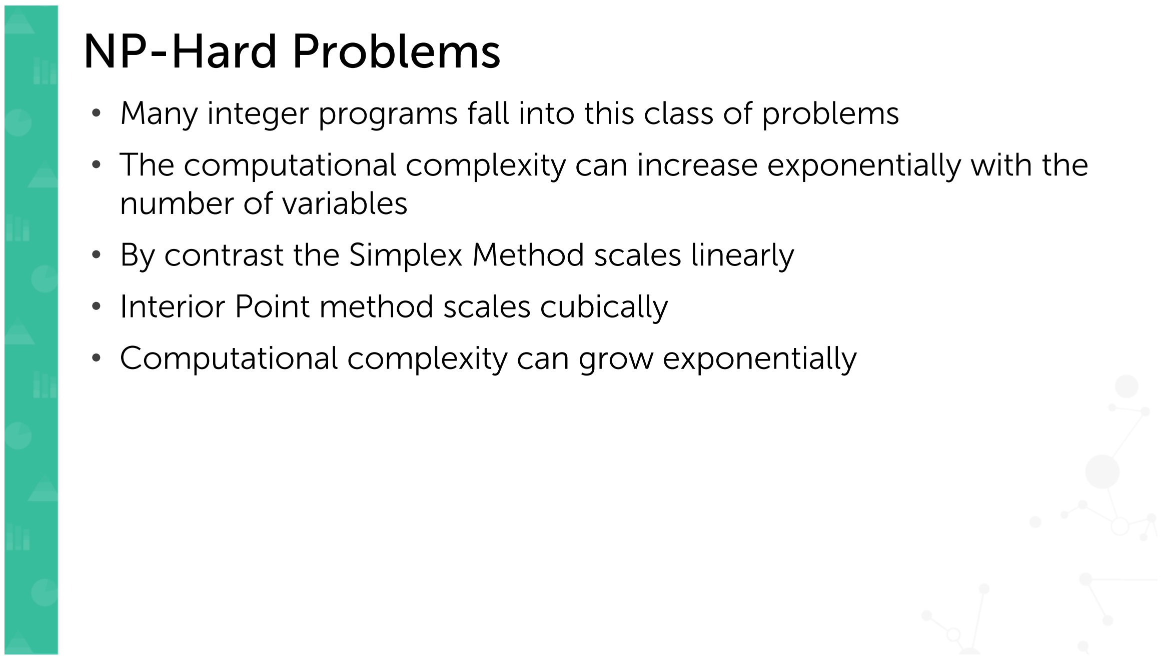
$$3x_1 + 2x_2 \leq 18 + My$$
$$x_1 + 4x_2 \leq 16 + M(1 - y)$$

If we have two constraints and it is only necessary to enforce at least one of them, then we can introduce an auxiliary variable that represents our choice between the two constraints. We'll let m be a very large positive number. When y equals zero, the first constraint is active and the second one is inactive because the right-hand side of the second constraint will be very large so that any values of the variables x1 and x2 with satisfy the first constraint will automatically satisfy the second. And in effect, we've activated the second constraint.

When y equals 1, the second constraint is active and the first is not. I encourage you to pause and think this through to make sure you understand it. This can be easily generalized to three out of five constraints or k out of n constraints. You can read more about the generalization in the book.

Fixed-Charge Problems

- Build x_j widgets at cost c_j
- Addition fixed cost of k_j if $x_j > 0$
- Binary variable y_j is 1 if we decide to make widgets and 0 otherwise
- To make sure x_j is 0 if y_j is zero we add a constraint $x_j \leq My_j$ (use large M)
- Total cost of widgets is $\sum c_j x_j + k_j y_j$

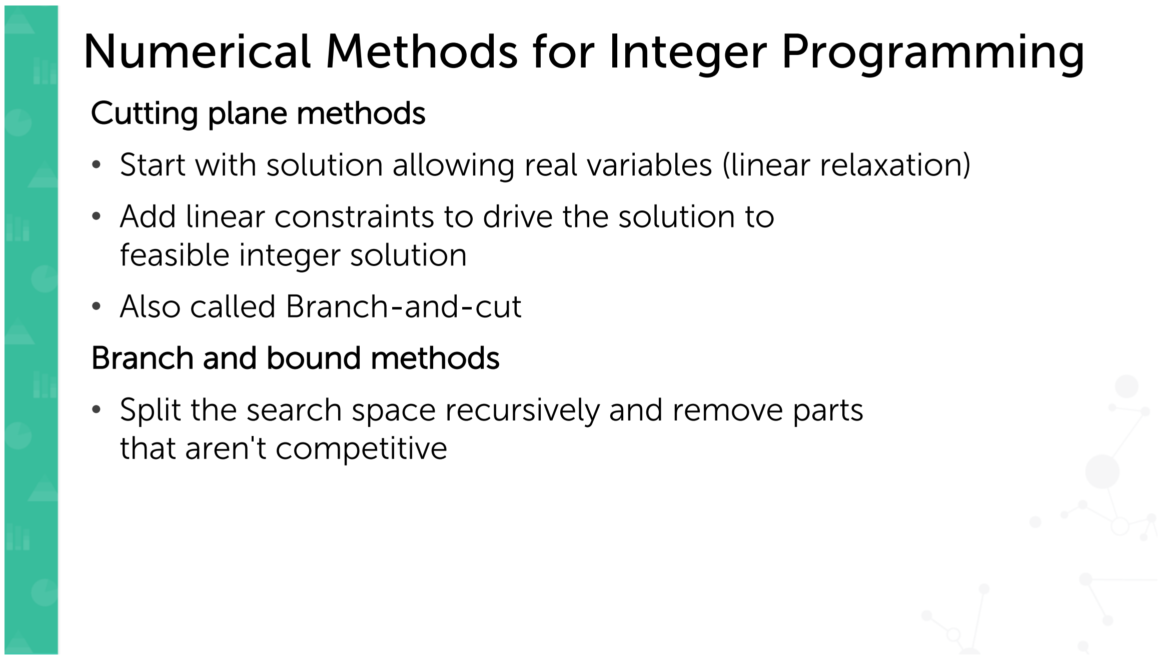


NP-Hard Problems

- Many integer programs fall into this class of problems
- The computational complexity can increase exponentially with the number of variables
- By contrast the Simplex Method scales linearly
- Interior Point method scales cubically
- Computational complexity can grow exponentially

It may be counter-intuitive. But by requiring the decision variables to be integers, we have made the problem much harder. For general problems, we are no longer guaranteed that the optimal solution is a corner feasible point. And we may have to try every possible combination of integer values. If, for instance, there were 30 binary variables, then there would be 2 to the 30th power or about a trillion possible combinations to be checked. Imagine if there were hundreds of binary variables.

Smart numerical methods can systematically remove many of the possible combinations to drastically reduce the computational complexity. But generally integer programs are restricted to a smaller number of variables than linear programs with continuous variables.



Numerical Methods for Integer Programming

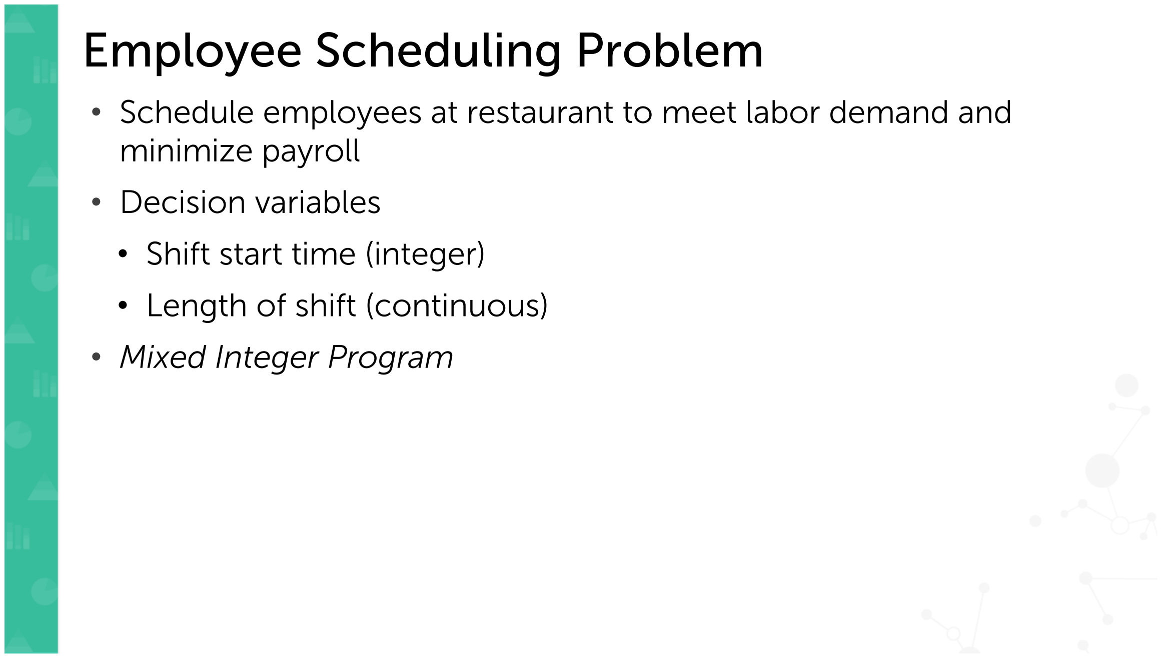
Cutting plane methods

- Start with solution allowing real variables (linear relaxation)
- Add linear constraints to drive the solution to feasible integer solution
- Also called Branch-and-cut

Branch and bound methods

- Split the search space recursively and remove parts that aren't competitive

Our focus is not on the details of the numerical methods in this course. But if you wish to know more about these methods, they are briefly described in the book. Next we'll consider an example of what's called a mixed integer program.



Employee Scheduling Problem

- Schedule employees at restaurant to meet labor demand and minimize payroll
- Decision variables
  - Shift start time (integer)
  - Length of shift (continuous)
- *Mixed Integer Program*

This variation of a scheduling problem uses some integer decision variables and some continuous ones. When that happens, we call it a mixed integer program. Most modern software can handle mixed integer programs. You just have to declare the various variables to be of the right types, floats or integers. There are numerous other examples of integer and mixed integer programs you can read about in the book. A few are mentioned on the next slide.

Applications of (Mixed) Integer Programming

- **Production planning** (maximize production)
- **Scheduling** (find feasible solution, minimize cost)
- **Telecomm network planning** (minimizing cost)
- **Cellular networks** (allocating frequencies)
- **Traveling salesman** (minimize cost)

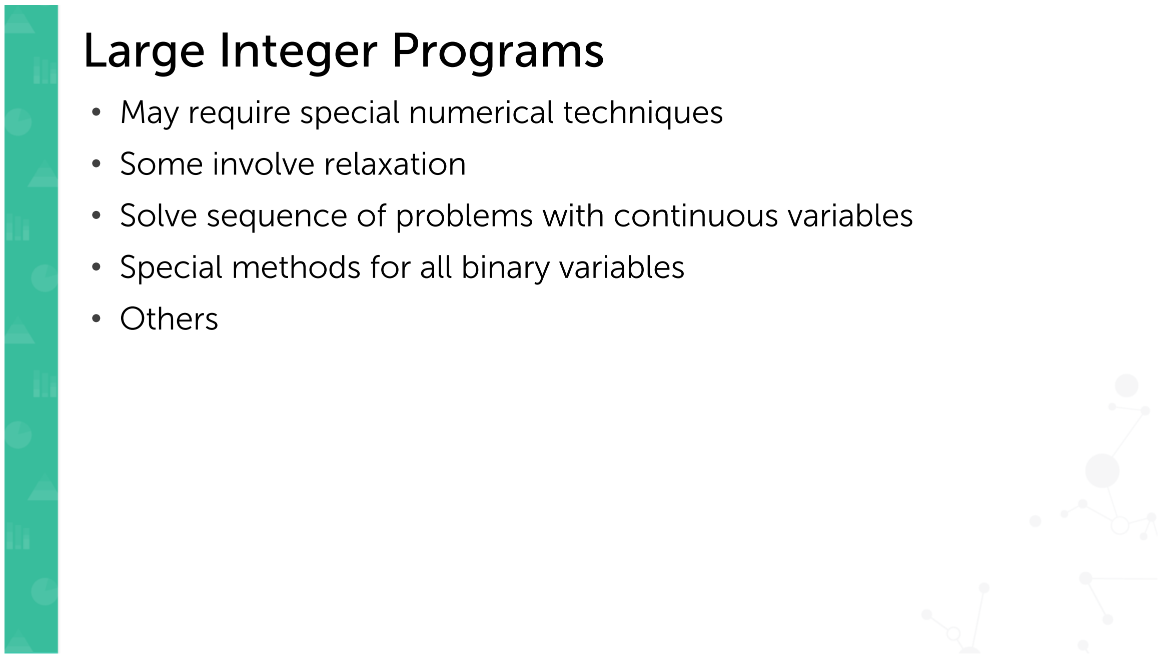
Integer Programming in OPL

- Just declare the relevant variables to be integer type and specify the range
- For binary variables the last two statements are equivalent

```
dvar int x in 0..maxint;
dvar int y in 0..5;
dvar z in 0..1;
dvar boolean z;
```

Integer Programming in Solver

- Video>Excel_Integer_Prog.mp4



Large Integer Programs

- May require special numerical techniques
- Some involve relaxation
- Solve sequence of problems with continuous variables
- Special methods for all binary variables
- Others

If you have a very large integer program, you may not be able to solve it with a canned commercial solver. There are advanced numerical methods which may help. But they are beyond the scope of our class. Sometimes the problem may have to be reformulated. Remember that we can typically solve much larger problems with continuous variables than with integer variables.