More about OPL and Integer Programming

DS 775

Topics

- Optimization Programming Language (OPL)
  - using external data
  - our example: Excel -> R -> OPL
- Integer Programming
  - examples
  - how is different than linear programming?
  - much greater computational complexity

More about OPL and Integer Programming

Topics

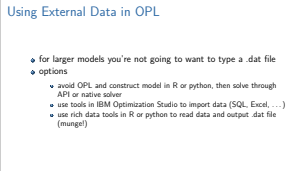
Topics

- Optimization Programming Language (OPL)
  - using external data
  - our example: Excel -> R -> OPL
- Integer Programming
  - examples
  - how is different than linear programming?
  - much greater computational complexity

- audio01.mp3
- continuing on from our discussion of OPL last week,
- we'll look at an example of using external data to build a model in OPL
- our solution isn't elegant, but is typical of the work a data scientist sometimes has to do
- we'll also briefly 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!)



- audio02.mp3
- there are many ways to interact with the solvers in IBM Optimization Studio, but I'd prefer a
- a solution which allows us to continue the paradigm of separating the model and the data.
- we'll use the last option wherein 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 OPL 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

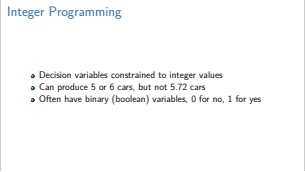
Munging Example

- 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

- audio04.mp3 . . . munging is data science slang for crudely transforming data which is exactly what we are 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 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 and see the data file that is produced.
- you may have to install the XLconnect package first.
- the data file produced should be the same as the transp4.dat file we used in last weeks transportation example

Integer Programming

- Decision variables 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

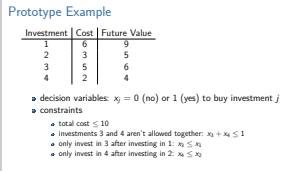


- no audio here

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 ≤ 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: $x_4 \leq x_2$



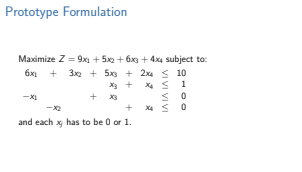
- audio04.mp3
- suppose our company is trying to decide how to spend its budget of 10 unit on 4 investments
- we have four binary decision variables and we want to maximize the future value
- investment 3 is something that only makes sense if we invest in 1 also, the same is true of investments 4 and 2
- investment 3 and 4 shouldn't occur together
- the next slide shows the model formulation

Prototype Formulation

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

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

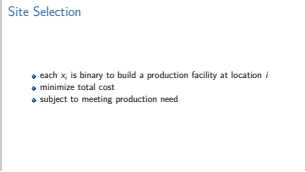
and each x_j has to be 0 or 1.



- no audio here, just leave up for students to study as needed.

Site Selection

- each x_i is binary to build a production facility at location i
- minimize total cost
- subject to meeting production need



- audio05.mp3
- another common example is where to locate factories or something similar
- we could simply minimize the total cost
- but we might also wish to incorporate distribution costs, costs of materials and labor, etc.
- next we'll turn to some different ways in which binary variables can be used in linear and integer programs

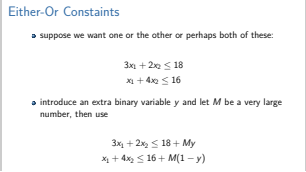
Either-Or Constaints

- suppose we want one or the other or perhaps both of these:

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

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

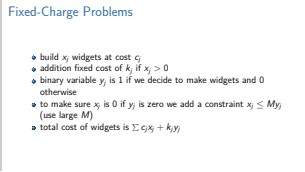
$$3x_1 + 2x_2 \leq 18 + My$$
$$x_1 + 4x_2 \leq 16 + M(1 - y)$$



- audio06.mp3
- if we have two contraints and it is only necessary to enforce at least one of them, then we
- can introduce an auxilliary variable that represents our choice between the two constraints
- let M be a very large positive number, when y $=$ 0 then 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 x_1 and x_2 which satisfy the first constraint will automatically satisfy the second.
- when y $=$ 1 the second contraint is active and the first is not.
- this can be easily generalized to a 3 out of 5 constraints, etc., 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 $\sum c_j x_j + k_j y_j$



- no audio, leave note to students in bottom frame, read more about this example in secction 12.2

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

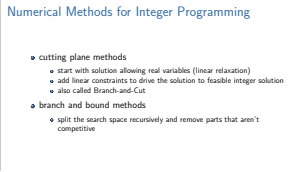
NP-Hard Problems

- 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

- audio07.mp3 . . . 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 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 combminations 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

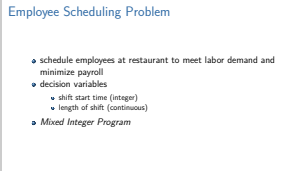
- cutting plane methods
  - start with solution allowing real variables (linear relaxation)
  - add linear constraints to drive the solution to feasible integer solution
  - also called Branch-and-Cut
- branch and bound methods
  - split the search space recursively and remove parts that aren't competitive



- audio08.mp3
- *leave note at bottom for students*: read 12.6 and 12.8 to learn more
- our focus is not on the details of the numerical methods in this course, but if you wish to know more these methdos are briefly described in the book.
- next we'll consider an example of 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*



- audio09.mp3
- this variation of a scheduling problem uses some integer decision variables and some continuous ones
- when that happens we called it a mixed integer program. Most modern software can handle mixed integer programs.
- there are numerous other examples of integer and mixed integer programs you can read about in the text 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)

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)

- no audio

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 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;
```

- no audio here

Integer Programming in Solver

- Dave Reineke is helping with this slide so consider this a placeholder

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

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 method 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