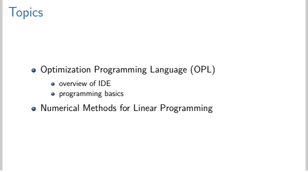
Modeling Languages for Optimization

DS 775

Topics

- Optimization Programming Language (OPL)
    - overview of IDE
    - programming basics
- Numerical Methods for Linear Programming



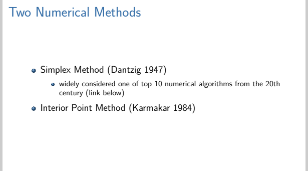
Topics

- Optimization Programming Language (OPL)
  - overview of IDE
  - programming basics
- Numerical Methods for Linear Programming

- audio01.mp3
- Microsoft Excel is great for solving small linear programs, but you probably wouldn't want to specify a model with thousands of variables in a spreadsheet.
- for a large model you could build the model in a good data manipulation enironment such as R or Python, but perhaps even better is to use a modeling language
- we'll focus on the optimization progamming language, or OPL, but there are others such as LINGO or AMPL. We've chosen OPL not only because it's free for academic use and also widely used commercially, but also because OPL supports both linear programming and constraint programming. We'll learn a little about constraint programming later in the course.

Two Numerical Methods

- Simplex Method (Dantzig 1947)
    - widely considered one of top 10 numerical algorithms from the 20th century (link below)
- Interior Point Method (Karmakar 1984)



Two Numerical Methods

- Simplex Method (Dantzig 1947)
  - widely considered one of top 10 numerical algorithms from the 20th century (link below)
- Interior Point Method (Karmakar 1984)

- audio02.mp3
- add the link "The Best of the 20th Century: Editors Name Top 10 Algorithms" as `https://www.siam.org/pdf/news/637.pdf` below the slide
- before we jump into OPL, let's look briefly at the numerical algorithms that are used to solve Linear Programs.
- from an appliied perspective it isn't essential to know the details of the algorithms, but you should know a little so you can choose the appropriate algorithm if necessary
- the simplex method is considered to be one of the most important numerical algorithms of the 20th century because it enabled the fast solution somewhat large linear programs. You might enjoy reading about the top 10 numerical algorithms of the 20th century, see the link below.

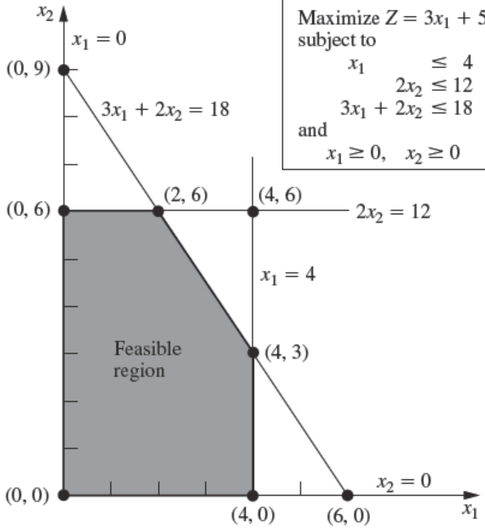
Wyndor Problem - Simplex Method

Maximize $Z = 3x_1 + 5x_2$,
subject to

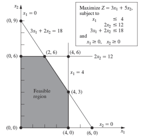
$$\begin{aligned} x_1 \quad\quad &\le 4 \\ 2x_2 &\le 12 \\ 3x_1 + 2x_2 &\le 18 \end{aligned}$$

and

$$x_1 \ge 0, \quad x_2 \ge 0$$



Wyndor Problem - Simplex Method



Maximize $Z = 3x_1 + 5x_2$,
subject to

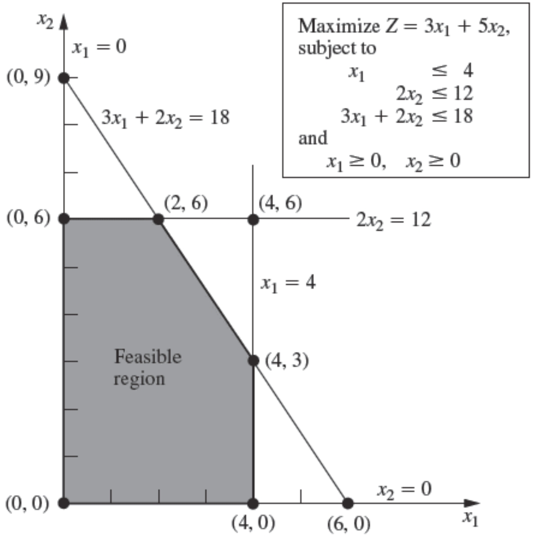
$$\begin{aligned} x_1 &\leq 4 \\ 2x_2 &\leq 12 \\ 3x_1 + 2x_2 &\leq 18 \end{aligned}$$

and

$$x_1 \geq 0, \quad x_2 \geq 0$$

- this slide is a placeholder and should be replaced with Video01.mp4
- use Explain Everything on this slide for video.
- zoom in on problem statement as a reminder of the prototype problem
- zoom in on graph and walk through simplex method (solve algebraic equations to find edges and vertices and move to vertex with higher objective function value)

Wyndor Problem - Interior Point Method



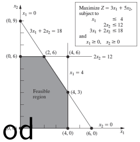
Maximize $Z = 3x_1 + 5x_2$,
subject to

$$\begin{aligned} x_1 \qquad &\le 4 \\ 2x_2 &\le 12 \\ 3x_1 + 2x_2 &\le 18 \end{aligned}$$

and

$$x_1 \ge 0, \quad x_2 \ge 0$$

Wyndor Problem - Interior Point Method



Maximize $Z = 3x_1 + 5x_2$,
subject to

$$\begin{aligned} x_1 &\le 4 \\ 2x_2 &\le 12 \\ 3x_1 + 2x_2 &\le 18 \end{aligned}$$

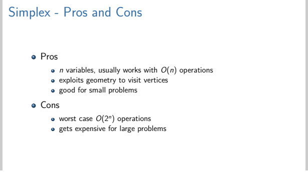
and

$$x_1 \ge 0, \quad x_2 \ge 0$$

- this slide is a placeholder and should be replaced with Video02.mp4
- zoom in on graph and briefly describe method
- build a new objective function that includes information about the constraints and optimize the new function

Simplex - Pros and Cons

- Pros
  - n variables, usually works with $O(n)$ operations
  - exploits geometry to visit vertices
  - good for small problems
- Cons
  - worst case $O(2^n)$ operations
  - gets expensive for large problems



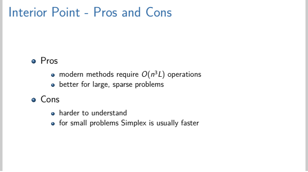
Simplex - Pros and Cons

- Pros
  - n variables, usually works with $O(n)$ operations
  - exploits geometry to visit vertices
  - good for small problems
- Cons
  - worst case $O(2^n)$ operations
  - gets expensive for large problems

- audio03.mp3 goes with this slide
- great for small problems, but there are pathological problems for which the number of operations grows exponentially with the number of variables
- also, for very large problems, some of the numerical linear algebra required by the method becomes quite expensive

Interior Point - Pros and Cons

- Pros
  - modern methods require $O(n^3L)$ operations
  - better for large, sparse problems
- Cons
  - harder to understand
  - for small problems Simplex is usually faster



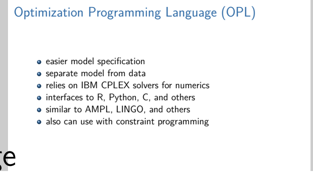
Interior Point - Pros and Cons

- Pros
  - modern methods require $O(n^3 L)$ operations
  - better for large, sparse problems
- Cons
  - harder to understand
  - for small problems Simplex is usually faster

- audio04.mp3 goes with this slide
- again, n represents the number of decision variables and L is the bit length of the data, for example a double precision floating point number is represented by 64 bits of which 53 represent the significant digits. Basically higher precision answers require more operations.
- a sparse problem is one in which each constraint only involves a few of the variables. For example there may be 1000 variables, but each constraint equation only involves 3 or 4 of the variables. We'll learn how to exploit sparsity to write more efficient linear programs a little later.
- from an applied perspective we don't much care that Interior Point methods are harder to understand, just consider selecting an interior point method if you have a very big problem . . . the CPLEX solvers from IBM that we will use can automatically choose between solvers

Optimization Programming Language (OPL)

- easier model specification
- separate model from data
- relies on IBM CPLEX solvers for numerics
- interfaces to R, Python, C, and others
- similar to AMPL, LINGO, and others
- also can use with constraint programming



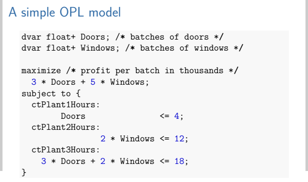
Optimization Programming Language (OPL)

- easier model specification
- separate model from data
- relies on IBM CPLEX solvers for numerics
- interfaces to R, Python, C, and others
- similar to AMPL, LINGO, and others
- also can use with constraint programming

- audio05.mp3 goes with this slide
- programming languages such as OPL, AMPL, LINGO and many others allow for formulation of optimization programs using syntax that is often very similar to the math notation.
- they also usually have mechanisms for separating the data or coefficients from the mathematical model . . . this allows the model to be easily updated and for multiple different models to be run
- these programming languages often make it easy to exploit sparsity when there are many variables but only a few variables are involved in each constraint
- one of the nice features of OPL is that it is trying to bridge the gap between mathematical programming and constraint programming - constraint programming is something we'll touch on near the end of the course

A simple OPL model

```
dvar float+ Doors; /* batches of doors */
dvar float+ Windows; /* batches of windows */

maximize /* profit per batch in thousands */
  3 * Doors + 5 * Windows;
subject to {
  ctPlant1Hours:
        Doors               <= 4;
  ctPlant2Hours:
                2 * Windows <= 12;
  ctPlant3Hours:
    3 * Doors + 2 * Windows <= 18;
```

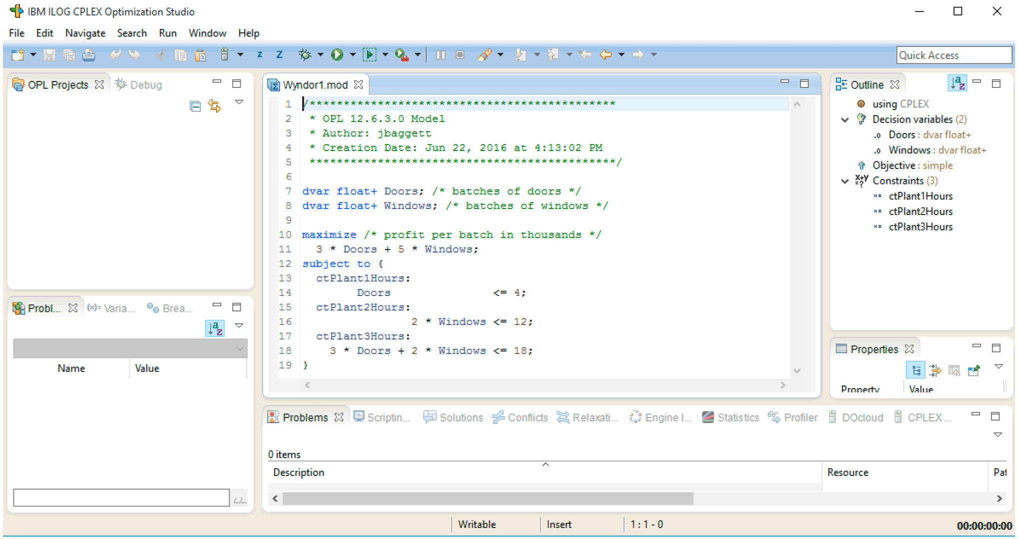


A simple OPL model

```
dvar float+ Doors; /* batches of doors */
dvar float+ Windows; /* batches of windows */

maximize /* profit per batch in thousands */
  3 * Doors + 5 * Windows;
subject to {
  ctPlant1Hours:
        Doors               <= 4;
  ctPlant2Hours:
                2 * Windows <= 12;
  ctPlant3Hours:
    3 * Doors + 2 * Windows <= 18;
}
```

- audio06.mp3 goes with this slide
- the overall structure of an OPL program is straight-forward. Notice that this program pretty much reads the same as the mathematical formulation of the problem.
- each OPL program will start with declaration of decision variables (dvar) and inititialization of parameters, then the objective function is defined, and finally the constraints are specified. Don't worry about studying the syntax here, we'll look at this program more closely in a bit.
- Next we'll look at a series of videos that will demonstrate the IBM Optimization Studio and introduce some of the ideas of programming in OPL. The IBM Optimization Studio is available on the virtual desktop, but you can also install your own copy on a PC. The Mac version does not include OPL or the Development Environment

Introduction to Optimization Studio

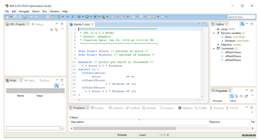


```
/*********************************************
 * OPL 12.6.3.0 Model
 * Author: jbaggett
 * Creation Date: Jun 22, 2016 at 4:13:02 PM
 *********************************************/

dvar float+ Doors; /* batches of doors */
dvar float+ Windows; /* batches of windows */

maximize /* profit per batch in thousands */
  3 * Doors + 5 * Windows;
subject to {
  ctPlant1Hours:
        Doors              <= 4;
  ctPlant2Hours:
                2 * Windows <= 12;
  ctPlant3Hours:
    3 * Doors + 2 * Windows <= 18;
}
```

Modeling Languages for Optimization

└Introduction to Optimization Studio

Introduction to Optimization Studio



```
/*********************************************
 * OPL 12.6.3.0 Model
 * Author: jbaggett
 * Creation Date: Jun 22, 2016 at 4:13:02 PM
 *********************************************/

dvar float+ Doors; /* batches of doors */
dvar float+ Windows; /* batches of windows */

maximize /* profit per batch in thousands */
  3 * Doors + 5 * Windows;
subject to {
  ctPlant1Hours:
         Doors              <= 4;
  ctPlant2Hours:
                2 * Windows <= 12;
  ctPlant3Hours:
    3 * Doors + 2 * Windows <= 18;
}
```

- this slide is a placeholder for video03.mp4, use the title above
- [NOTE] include link at bottom: to download Optimization Studio `https://www-01.ibm.com/software/websphere/products/optimization/cplex-studio-community-edition/`
- [NOTE] the file Wyndor1.mod is in the download pack for the week. You can import it into the project by using File -> Copy Files to Project
- use Camtasia to give demo of IDE
- start with fresh IDE screen and show pieces of IDE: project window, editor, output
- create a new project called Wyndor
- add a model file to the project and "type" in wyndor1
- add a run configuration
- run the model
- show the output

Separating the model and data

- this slide is a placeholder for video04.mp4, use the title above
- add the remaining files Wyndor2, ... from the weekly download to this project
- open Wyndor2.mod, Wyndor2.dat
- demonstrate separating data and model
- walk through model, note that we'll later move the data to a distinct file
- show the additional output added as post-processing
- add a run configuration and run it

Indexing using names

- this slide is a placeholder for video05.mp4, use title above
- indexing by other things
- run it

Infeasible constraints

Infeasible constraints

- this slide is a placeholder for video06.mp4, use title above
- use modified version of Wyndor but with tweaks so that there are no feasible solutions

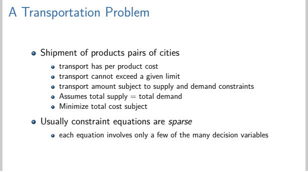
Model and Data - Separate Files

└Model and Data - Separate Files

- this slide is a placeholder for video07.mp4, use title above
- modify Wyndor3.mod to have separate files

A Transportation Problem

- Shipment of products pairs of cities
  - transport has per product cost
  - transport cannot exceed a given limit
  - transport amount subject to supply and demand constraints
  - Assumes total supply $=$ total demand
  - Minimize total cost subject
- Usually constraint equations are *sparse*
  - each equation involves only a few of the many decision variables



A Transportation Problem

- Shipment of products pairs of cities
  - transport has per product cost
  - transport cannot exceed a given limit
  - transport amount subject to supply and demand constraints
  - Assumes total supply = total demand
  - Minimize total cost subject
- Usually constraint equations are *sparse*
  - each equation involves only a few of the many decision variables

- use audio07.mp3 with this slide
- now we'll consider a simple example of a transportation problem
- this particular example is included in the documentation of the IBM Optimization Studio
- typically we'd have a large network of cities and variety of products, but typically each city ships products to only a few of the cities so the model equations will only involve a few of the many variables ... that is the model equations are sparse
- for large problems we want to exploit this sparsity to reduce the number of computations and the amount of memory required
- before we start looking at models we'll have a short video to clarify the problem

A Transportation Problem - a picture

A Transportation Problem - a picture

- this is a placeholder for video08.mp4
- should this be a bipartite graph with suppliers and customers
- This will be a video showing a graph version of the problem.
- start with a few nodes, 5 or 6, and one product
- add supply and demand and cost
- discuss sparsity
- shortly we'll look at two models, a straight-forward model that doesn't exploit sparsity and would be quite inefficient for a large version of this probem, and a second model that uses "tuples" to exploit the sparsity of this problem.

Install Example into Workspace

- this is a placeholder for video09.mp4, please use title above

First Model

- this is a placeholder for video10.mp4, please use title above
- Walk through transp1.mod
- point out the assert statement
- point out the huge .dat file with mostly zeros . . .
- point out where there are opportunities to exploit sparsity
- run the model and inspect the solution

Sparse Model with Tuples

Modeling Languages for Optimization

└─Sparse Model with Tuples

- this is a placeholder for video11.mp4, please use title above
- go through transp4.mod as this seems to be the best sparse implementation
- carefully walk through tuples as there are several examples here
- walk through the assignment notation x:y
- run the model and show output