

DS 775

# Prescriptive Analytics

Modeling Languages for Optimization



## Topics

- Numerical Methods of Linear Programming
- Optimization Programming Language (OPL)
  - Overview of IDE
  - Programming Basics



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 environment, such as R or Python. Perhaps even better is to use a modeling language.

We'll focus on the optimization programming 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 the top 10 numerical algorithms from the 20<sup>th</sup> century. ([link below](#))

### Interior Point Method (Karmakar 1984)



Before we jump into OPL, let's look briefly at the numerical algorithms that are used to solve linear programs. From an applied perspective, it is not 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 of 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

- Video01.mp4

## Simplex Method – 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

The simplex method is 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.

## Wybodor Problem – Interior Point Method

- Video02.mp4



## Interior Point Method – 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



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 other variables. For example, there may be 1,000 variables, but each constraint equation only involves three or four of the variables. We'll learn how to exploit sparsity to write more efficient 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



Programming languages such as OPL, AMPL, Lingo and many others allow for formulation of optimization problems 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 of the 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, something we'll learn about it later in 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;
```

The overall structure of an OPL program is straightforward. 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, the dvar statements you see there, and initialization 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 problem 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 desktop environment.



# Introduction to Optimization Studio

- Video03.mp4



## Separating the Model and Data

- Video04.mp4





## Indexing Using Names


- Video05.mp4



## Infeasible Constraints

- Video06.mp4





## Model and Data – Separate Files

- Video07.mp4



## 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



Now we'll consider a simple example of a transportation problem. This particular example is included in the documentation of the IBM Optimization Studio, so we'll see how to import it from the documentation shortly.

Typically, we'd have a large network of cities and a 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 the 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.

## Transportation Problem – A Picture

- Video08.mp4







## Install Example Into Workspace

- Video09.mp4





## First Model

- Video10.mp4



## Sparse Model with Tuples

- Video11.mp4

