Mathematics for Decision Making: An Introduction

Lecture 2

Matthias Köppe

UC Davis, Mathematics

January 8, 2009

Let's revisit the example from last time

The Notip Table Company sells two models of its patented five-leg tables. The basic version uses a wood top, requires 0.6 hours to assemble, and sells for a profit of $200. The deluxe model takes 1.5 hours to assemble (because of its glass top), and sells for a profit of $350. Over the next week the company has 300 legs, 50 wood tops, 35 glass tops, and 63 hours of assembly available. Notip wishes to determine a maximum profit production plan assuming that everything produced can be sold.

We found a **mathematical model** for this problem. We are using two variables to represent the decision on the production quantities:

x_1 – *number of basic tables,* x_2 – *number of deluxe tables*

$$
\begin{array}{lll}
\max & 200x_1 + 350x_2 & \text{(Total profit)} \\
\text{s.t.} & x_1 \leq 50 & \text{(Wood tops available)} \\
& x_2 \leq 35 & \text{(Glass tops available)} \\
& 5x_1 + 5x_2 \leq 300 & \text{(Legs available)} \\
& 0.6x_1 + 1.5x_2 \leq 63 & \text{(Hours of assembly time available)} \\
& x_1, x_2 \geq 0 & \text{(Non-negative quantities to be produced)} \\
& x_1, x_2 \in \mathbf{Z} & \text{(Integer quantities to be produced)}
\end{array}
$$

How to solve this problem, I: Brute Force

Important observation

The feasible set F is finite.

Because of boundedness ($0 \leq x_1 \leq 50$ and $0 \leq x_2 \leq 35$) and integrality ($x_1, x_2 \in \mathbf{Z}$), we have only a finite number (at most $51 \times 36 = 1836$) feasible solutions.
On a computer, it is easy and super-fast to enumerate all these solutions. We compute the objective function for all these solutions, and pick the best solution!

Theorem ("Bounded Integer Optimization is 'Trivial' ")

Given integer optimization problems of the form

$$\begin{array}{lll} \max & f(\mathbf{x}) & \\ \text{s.t.} & g_i(\mathbf{x}) \leq 0 & \text{for } i = 1, \ldots, m \\ & \mathbf{x} = (x_1, \ldots, x_n) \in X = \mathbf{Z}^n & \end{array}$$

and finite lower and upper bounds for all variables, valid on the feasible region. Then there exists an algorithm that always computes an optimal solution $\mathbf{x}^*$.

Without given bounds, no such general algorithm can exist! (Hilbert's 10th)

The problem with Brute Force

To solve a problem in practice on a computer, it is not enough to know that there exists an algorithm, i.e., a finite procedure.

It easily happens that the brute force method is not useful at all because for most interesting problems it takes too long, even on very fast computers.

Large bounds

If we have three integer variables x_1, x_2, x_3 with bounds $1 \leq x_i \leq 1\,000\,000$, we would have to check 10^{18} solutions!
If a computer with 8 processor cores running at $2.5\,\mathrm{GHz}$ could check one solution in one cycle, it would still take $10^{18}/(8 \cdot 2.5 \cdot 10^9) = 50 \cdot 10^6$ seconds (more than 1.5 years). This may be too long!

High dimension

More impressive examples appear when we have many variables, even if they have low bounds. In a combinatorial problem with 100 binary variables ($x_i = 0$ or $x_i = 1$), we would have to check 2^{100} solutions. A current supercomputer, IBM's Blue Gene/P, has 884,736 processors running at 850 MHz. It would take $1.6 \cdot 10^{15}$ years. The age of the universe is only $(13.7 \pm 0.1) \cdot 10^9$ years (Wikipedia).

How to solve this problem, II: Graphing

An important observation is that we can gain insight by graphing the example problem – because it only has 2 variables.

How to solve this problem, III: Using optimization software

We will be using three pieces of software in this lecture. (All of them are free at least for noncommercial and academic use. ZIMPL is free (open source) software.)

SCIP – "Solving Constraint and Integer Programs"

SCIP is a very good solver for **Mixed Integer Linear Optimization Problems**, i.e.,

$$\begin{aligned} \max \quad & f(\mathbf{x}) \\ \text{s.t.} \quad & g_i(\mathbf{x}) \leq 0 \qquad \text{for } i = 1, \ldots, m \\ & \mathbf{x} = (x_1, \ldots, x_n) \in X \end{aligned}$$

where f and g_i are linear functions, and $X = \mathbf{R}^n$, or $X = \mathbf{Z}^n$, or $X = \mathbf{R}^{n_1} \times \mathbf{Z}^{n_2}$.

SoPLEX – a solver for linear optimization problems

A super-fast solver for linear optimization problems ($X = \mathbf{R}^n$) is the basic building block of optimization and operations research technology.

ZIMPL – an algebraic modeling language for optimization

ZIMPL provides a convenient way to write down optimization problems in the computer.

SCIP uses both SoPLEX and ZIMPL internally.

Our example problem in ZIMPL

ZIMPL input files are plain text files; use a text editor (such as *Emacs* or *gedit*) to create them, not a word processor. This is file `notip-1.zpl`:

```
var basic_tables integer;
var deluxe_tables integer;

maximize profit:
  200 * basic_tables + 350 * deluxe_tables;

subto legs_constraint:
  5 * (basic_tables + deluxe_tables) <= 300;

subto wood_tops_constraint:
  basic_tables <= 50;

subto glass_tops_constraint:
  deluxe_tables <= 35;

subto assembly_hours_constraint:
  0.6 * basic_tables + 1.5 * deluxe_tables <= 63;
```

The ZIMPL language: Variable definitions

A complete description is found in section 4.4 in ZIMPL User Guide.

Variable type:

`var NAME integer;`	Defines an integer variable
`var NAME binary;`	Defines a binary (0/1) variable
`var NAME real;`	Defines a real variable
`var NAME;`	Also defines a real variable

All variables (except binary) have a lower bound of 0 and no upper bound. If different upper bounds are needed, change the variable definition:

`var NAME integer >= -5 <= 5`	Defines an integer variable that can take values from -5 to 5
`var NAME real >= -infinity <= infinity`	Defines a free variable that can take arbitrary real values

The ZIMPL language: Objective and constraints

The objective function is written as:

```
maximize NAME: TERM;
minimize NAME: TERM;
```

where TERM is an arbitrary (linear) expression.

Each of the constraints is written as:

```
subto NAME: TERM SENSE TERM
```

Here SENSE is one of $<=$, $>=$, $==$.

(The ZIMPL language has a much greater power than this; we'll dive into its complexity gradually.)

Homework

For Thursday Jan 15:

- By rolling dice, decide randomly for one of the exercises 2-2, 2-3, 2-4, and 2-5 from the handout.
  Then do this exercise, and prepare a 5-minute presentation of the problem and its solution.