MATH1326
Advanced Optimisation with Python

Week 2

- Integer Programming Modelling Refresher
- Modelling with Binary variables
- Formulating problems with computational efficiency in mind
- Presolve

Integer Programming Modelling Refresher

- An IP in which all variables are required to be integers is called a **pure integer programming** problem.
- An IP in which only some of the variables are required to be integers is called a **mixed integer programming problem**.
- An integer programming problem in which all the variables must be 0 or 1 is called a 0-1 IP or BP.
- The LP obtained by omitting all integer or 0-1 constraints on variables is called **LP relaxation** of the IP.

Integer Programming Modelling

Stockco is considering four investments. Investment 1 will yield a new present value (NPV) of $16,000; investment 2, an NPV of $22,000; investment 3, an NPV of $12,000; and investment 4, an NPV of $8,000. Each investment requires a certain cash outflow at the present time: investment 1, $5,000; investment 2, $7,000; investment 3, $4,000; and investment 4, $3,000. Currently, $14,000 is available for investment.

Integer Programming Modelling

- Begin by defining a variable for each decision that Stockco must make, i.e. x_i is the money invested in option i
- The objective function is NPV obtained by Stockco;

  Total NPV obtained by Stocko = $16x_1 + 22x_2 + 12x_3 + 8x_4$

- Stockco faces the constraint that at most \$14,000 can be invested.
- Stockco's 0-1 IP is

$$\begin{aligned} \max z &= 16x_1 + 22x_2 + 12x_3 + 8x_4 \\ \text{s.t.} \quad & 5x_1 + 7x_2 + 4x_3 + 3x_4 \leq 14 \\ & x_j = 0 \text{ or } 1 \quad (j = 1,2,3,4) \end{aligned}$$

Integer Programming Modelling

What if we have the following requirements?

1. Stockco can invest in at most two investments

$$x_1 + x_2 + x_3 + x_4 + x_5 \leq 2$$

2. If Stockco invests in investment 2, they must also invent in investment 1.

$$x_2 \leq x_1$$

3. If Stockco invests in investment 2, they cannot invest in investment 4.

$$x_2 + x_4 \leq 1$$

Simple Implications

- If we do A then we must also do B

  $b \geq a$

- If we do A then we must *not* do B

  $a + b \leq 1$

- If we can at most do n out of A, B,…, Z

  $a + b + \ldots + z \leq n$

Implications

- If we do A then we must do B and C

  $b \geq a$ and $c \geq a$

- If we do A then we must do *either* B *or* C

  $b + c \geq a$

- If we do B or C then we must do A

  $a \geq b$ and $a \geq c$

Harder Implications

- If we do both B and C then we must do A

$$2a \geq b + c$$

a	b	c	Check
0, 1	0	0	Ok!
0, 1	0	1	Not!
0, 1	1	0	Not!
1	1	1	Ok!

$$a \geq b + c - 1$$

a	b	c	Check
0, 1	0	0	Ok!
0, 1	0	1	Ok!
0, 1	1	0	Ok!
1	1	1	Ok!

Generalised Implications

- If we do two or more of B,C, D or E then we must do A.

  Try $a \geq b + c + d + e - 1$

  If more than two projects done then $a>1$ which cannot be as a is binary.

  Consider $a \geq (b + c + d + e)/4$

  This works if $b=c=d=e=1$ *but what if* $c=d=e=0$, *f*or $b=1$ and $c=d=e=0$ we get $a \geq 1/4$ *and* since a is binary this implies $a=1$ which is not right.

  $a \geq (b + c + d + e - 1)/3$

  This would work and A will only be forced if two or more of B, C, D, or E is done.

General If-Then Constraints

If $f(x_1, x_2, ..., x_n) > 0$ is satisfied, then the constraint $g(x_1, x_2, ..., x_n) \geq 0$ must be satisfied.

$$-g(x_1, x_2, ..., x_n) \leq My$$

$$f(x_1, x_2, ..., x_n) \leq M(1 - y)$$

$$y = 0 \text{ or } 1$$

M is a large enough number not to limit f(x) or –g(x) in any way, infinity would work, but we need the smallest such number to improve the LP relaxation bound to help the solver.

Either-or constraints

We are given two constraints of the form

$$f(x_1, x_2, ..., x_n) \leq 0$$

$$g(x_1, x_2, ..., x_n) \leq 0$$

We want to ensure at least one of them is satisfied

$$f(x_1, x_2, ..., x_n) \leq My$$

$$g(x_1, x_2, ..., x_n) \leq M(1 - y)$$

$$y = 0 \text{ or } 1$$

M is a large enough number not to limit f(x) or g(x) in any way, infinity would work, but we need the smallest such number to improve the LP relaxation bound to help the solver.

Either-or example

$$\text{Either } 2x_1 + x_2 \geq 6$$

$$\text{or } x_1 + 2x_2 \geq 7$$

$$Consider$$

$$2x_1 + x_2 \geq 6y$$

$$x_1 + 2x_2 \geq 7(1 - y)$$

Formulating problems with computational efficiency in mind

Computational effort

n binary variables have 2^n possible settings and hence

2^n possible nodes at the bottom of the

Branch and Bound solution tree.

eg $2^{100} = 10^{30}$ (or $million^5$)

In practice B&B more efficient than this but still important to reduce binary variables where possible.
Where applicable make use of Special Ordered Sets (SOS1 and SOS2). See discussion and example in Book 3.4.4 (SOS1) and 3.4.5 (SOS2).

Symmetry Example

$$b_{ij} = \begin{cases} 1 \text{ if truck } i \text{ sent on trip } j \\ 0 \text{ otherwise} \end{cases}$$

where $i = 1,2,3$ and $j = 1,2$

If the trucks are identical in terms of running costs, capacities, etc then for each integer solution eg

$b_{11} = 1 \;\; b_{22} = 1 \;\; b_{32} = 1 \qquad (1)$

there will be symmetric integer solutions, eg

$b_{21} = 1 \;\; b_{12} = 1 \;\; b_{32} = 1 \qquad (2)$

Symmetric solution will be obtained at different nodes of the B&B costing us valuable computational time and effort.

Better:

$n_i =$ number of trucks sent on trip j, then (1) and (2)

$\mathrm{n}_1 = 1$ and $n_2 = 2$

Presolve / Tightening Bounds

$$\begin{aligned} &Min \quad 5b_1 + 7b_2 + 10b_3 + 3b_4 + b_5 \\ &\text{s.t.} \\ &b_1 - 3b_2 + 5b_3 + b_4 - b_5 \geq 2 && (1) \\ &-2b_1 + 6b_2 - 3b_3 - 2b_4 + 2b_5 \geq 0 && (2) \\ &-b_2 + 2b_3 - 2b_4 - b_5 \geq 1 && (3) \\ &b_i \; i = 1,2,3,4,5 \text{ binary} \end{aligned}$$

by (3) $2b_3 \geq 1 + b_2 + 2b_4 + b_5 \geq 1$

$\Rightarrow b_3 \geq \frac{1}{2}$. Thus $b_3 = 1$ and can be removed from the problem.

Presolve / Tightening bounds (cont)

Constraint (2) now becomes:

$$6b_2 \geq 3 + 2b_1 + 2b_4 - 2b_5 \geq 1$$

Thus $b_2 \geq \dfrac{1}{6} \Rightarrow b_2 = 1$

Presolve / Tightening bounds (cont)

Substituting known values in constraint (3)

$$-1 + 2 - 2b_4 - b_5 \geq 1$$

$$\text{thus} \quad - 2b_4 - b_5 \geq 0$$

$$\text{giving} \quad b_4 = b_5 = 0$$

Substituting known values into constraint (1) gives $b_1 \geq 0$

Looking now at the objective function, we see that it will be at minimum when $b_1 = 0$

So we have managed to solve the whole problem simply by analysing the constraints.