INTEGER PROGRAMMING

Integer problems arise when

- the values of (some) variables have to be whole numbers
- set up costs are present
- problems involve yes / no decisions

Decision Problems

Decision problems can be handled by binary (yes/no) variables.

Example:

A motor vehicle manufacturer has to decide where to locate a new factory. If location 1 is chosen, the following constraint must be satisfied:

$$2X1 + X2 \leq 20 \qquad (1).$$

If location 2 is chosen the constraint is

$$3X1 + 4X2 \leq 35 \qquad (2).$$

A binary variable can be used to represent the decision. When this variable is equal to one, location 1 is chosen and when it is equal to zero, location 2 is chosen. Then the constraints can be represented as

$$2X1 + X2 + 10000Y \leq (20 + 10000) \qquad (1).$$

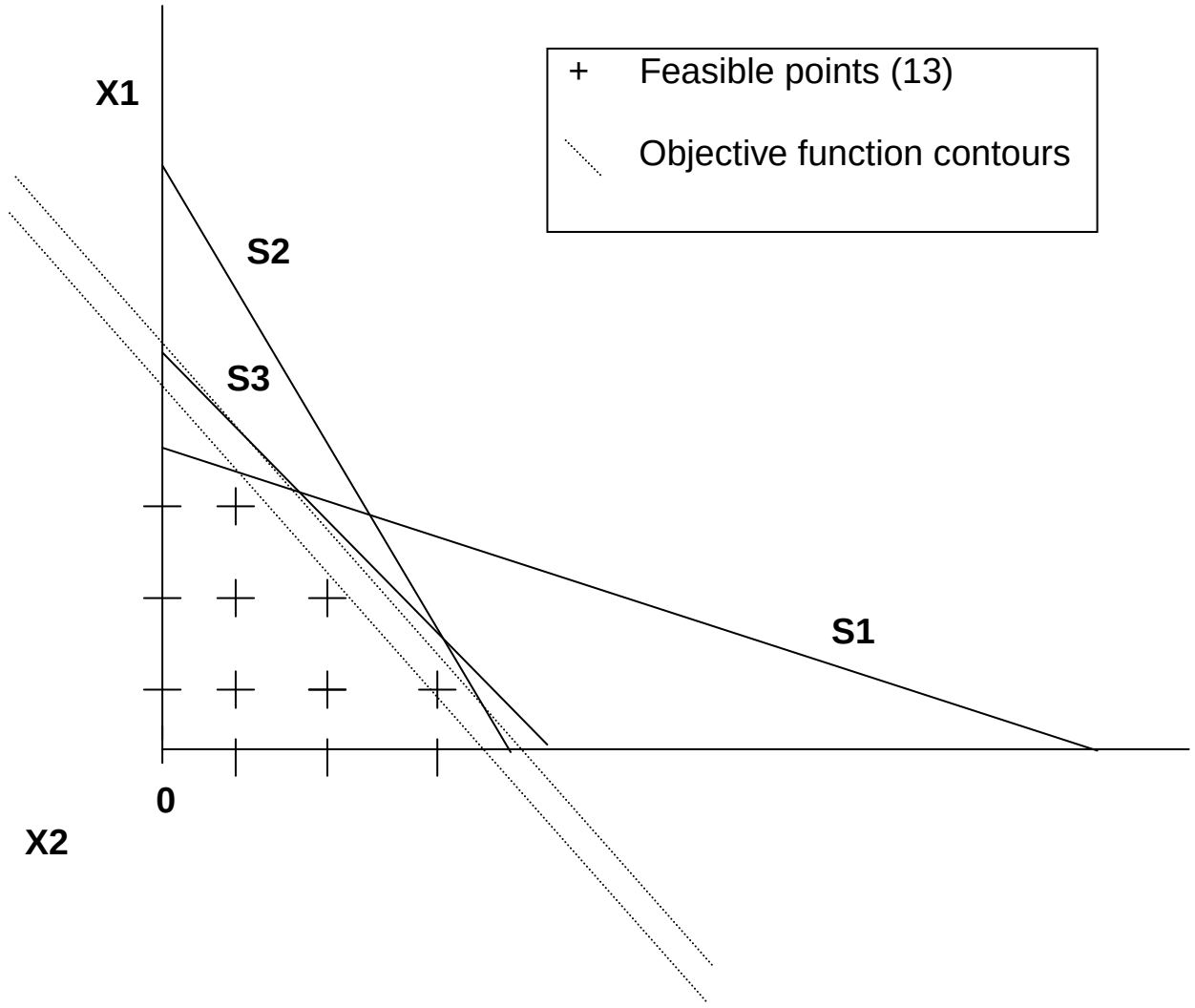
and

$$3X1 + 4X2 - 10000Y \leq 35 \qquad (2).$$

where Y is the binary variable and 10000 is an arbitrary large number which acts to impose a penalty where Y = 1 is selected.

The arbitrary number 10000 is sometimes represented by the letter M.

Graphical Representation
Maximisation Problem: X1 & X2 both integer



Examples of Integer Programming Problems

A) Pure Integer Problems

- The number of training courses to run to maximise profits
- How many computers to buy for the office

B) Binary (0 / 1) Problems

- Choosing whether or not to invest in an investment project
- Deciding which location to move to

C) Mixed (Integer and Non-Integer) Problems

- Fixed cost problems where a cost is incurred or not incurred according to whether or not another variable is non-zero.

Binary Problems

Example:

Choosing between four investment projects A, B, C, D

Let A, B, C, D be represented by *integer* variables with the restrictions

$A \leq 1$, $B \leq 1$, $C \leq 1$, $D \leq 1$. This constrains the variables to 0 / 1.

Binary problems allow the writing of special kinds of restrictions.

Multiple choice restrictions

a) Choose *at least* 2 investment projects

$$A + B + C + D \geq 2$$

b) Choose *at most* 3 projects

$$A + B + C + D \leq 3$$

c) Choose exactly 2 projects

$$A + B + C + D = 2$$

If ... then restrictions

a) If project A is chosen, B must also be chosen

$A \leq B$ or in standard form $A - B \leq 0$

b) Choose project C if and only if A is chosen

$C = A$ or in standard form $C - A = 0$

Integer Programming Solution Methods

Integer Programming is a difficult computational problem. Various solution methods exist. None is as efficient as the simplex method for standard linear programming problems.

Rounding Down

This is not a proper solution method but may be appropriate when the integer values are large.

Complete Enumeration

Go through all permutations and find the one that maximises the objective function. This is feasible in small problems and especially problems with a few binary variables. An example of such a problem is the classic knapsack problem. You are going hiking and have a list of things you would like to take. However, they must all fit in your knapsack. Which items do you take and which leave behind?

Branch and Bound Algorithm

This method partitions the feasible space into smaller subspaces eliminating invalid solutions. Partitioning (branching) continues until we know that no better integer solution can be found.

Example: Suppose variable Y is an integer variable.

Then for any integer value K we know that *either* $Y \leq K$ *or* $Y \geq K+1$.

We can set up two problems imposing one of these 2 constraints. This gives two branches. These eliminate the invalid solutions where Y lies between K and K+1 without ruling out any other valid solution.

We know that adding an extra restriction cannot improve the objective function and in general will reduce its value. This gives a bound on the best that we can do in the branches. The algorithm stops when no future branch can improve on the best integer solution we have found.

The algorithm starts from the unrestricted linear programming solution. This provides the initial upper bound on the best solution.

Steps in the Branch and Bound Algorithm

1. Compute the relaxed (i.e. linear programming) solution. This gives an upper bound on the objective function Z. Rounding down the integer variables gives an initial lower bound.

2. Choose one of the integer variables to branch on.
   e.g. X3 = 7.31

3. Solve the two problems imposing the additional constraints

   a) $X3 \leq 7$ *or* b) $X3 \geq 8$.

   This eliminates the solutions $7 < X3 < 8$ which violate the integer restriction.

4. Choose the branch with the highest objective function Z. This is the new upper bound on subsequent branches.

5. If an integer solution is found with a higher Z than the current lower bound, reset the lower bound. This is the best solution found so far.

6. Iterate while the upper bound is greater than the lower bound
   on all branches.

Network Diagram of Branch and Bound Algorithm