INTEGER LINEAR PROGRAMMING (IP)

IP is the name given to LP problems which have the additional constraint that some or all the variables have to be *integer*.

1. CLASSICAL INTEGER PROGRAMMING PROBLEMS

EXAMPLE 1: CAPITAL BUDGETING

A firm has n projects that it would like to undertake but because of budget limitations not all can be selected. In particular project j is expected to produce a revenue of c_j but requires an investment of a_{ij} in the time period i for i = 1,...m. The capital available in time period i is b_i. The problem of maximising revenue subject to the budget constraints can be formulated as follows: let $x_j = 0$ or 1 correspond to not proceeding or respectively proceeding with project j then we have to

$$\text{max} \quad \sum_{j=1}^{n} c_j x_j$$

$$\text{subject to} \quad \sum_{j=1}^{n} a_{ij} x_j \leq b_i; \; i = 1, \ldots, m$$

$$0 \leq x_j \leq 1 \quad x_j \text{ integer} \quad j = 1, \ldots, n$$

EXAMPLE 2: DEPOT LOCATION

We consider here a simple problem of this type: a company has selected m possible sites for distribution of its products in a certain area. There are n customers in the area and the transport cost of supplying the whole of customer j's requirements over the given planning period from potential site i is c_{ij}. Should site i be developed it will cost f_i to construct a depot there. Which sites should be selected to minimise the total construction plus transport cost?

To do this we introduce m variables $y_1, \ldots y_m$ which can only take values 0 and 1 and correspond to a particular site being not developed or developed respectively. We next define x_{ij} to be the fraction of customer j's requirements supplied from depot i in a given solution. The problem can then be expressed.

$$\text{min} \quad \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij} x_{ij} + \sum_{i=1}^{m} f_i y_i$$

$$\text{subject to} \quad \sum_{i=1}^{m} x_{ij} = 1 \; ; \; x_{ij} \leq y_i \quad i = 1, \ldots, m \; ; \quad j = 1, \ldots, n \qquad (1)$$

$$x_{ij} \geq 0 \; ; \; 0 \leq y_i \leq 1 \; ; \; y_i \text{ integer} \; ; \; i = 1, \ldots, m \; ; \; j = 1, \ldots, n$$

Note that if $y_i = 0$ then $f_i y_i = 0$ and there is no contribution to the total cost. Also $x_{ij} \leq y_i$ implies $x_{ij} = 0$ for j = 1,...n and so no goods are distributed from site i. This corresponds exactly to no depot at site i.

On the other hand, if $y_i = 1$ then $f_i y_i = f_i$ which is the cost of constructing depot i. Also $x_{ij} \leq y_i$ becomes $x_{ij} \leq 1$ which holds anyway from the constraints (1).

THE SET COVERING PROBLEM AND INTEGER PROGRAMMING FORMULATION

Let $S_1, \ldots, S_n$ be a family of subsets of a set $S = \{1,2, \ldots, m\}$. A covering of S is a subfamily S_j for $j \in I$ such that $S = \bigcup_{j \in I} S_j$. Assume that each subset S_j has a cost $c_j > 0$ associated with it. We define the cost of a cover to be the sum of the costs of the subsets included in the cover.

The problem of finding a cover of minimum cost is of particular practical significance. As an integer program it can be specified as follows: define the $m \times n$ matrix $A = \| a_{ij} \|$ by

$$a_{ij} = 1 \text{ if } i \in S_j$$

$$= 0 \text{ otherwise}$$

Let x_j be $0 - 1$ variables with $x_j = 1(0)$ to mean set S_j is included (respectively not included) in the cover. The problem is to

$$\text{Minimise} \quad \sum_{j=1}^{n} c_j x_j$$

$$\text{subject to} \quad \sum_{j=1}^{n} a_{ij} x_j \geq 1 \qquad i = 1,...m \tag{2}$$

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

The m inequality constraints have the following significance: since $x_j = 0$ or 1 and the coefficients a_{ij} are also 0 or 1 we see that $\sum_{j=1}^{n} a_{ij} x_j$ can be zero only if $x_j = 0$ for all j such that $a_{ij} = 1$. In other words only if no set S_j is chosen such that $i \in S_j$. The inequalities are put in to avoid this.

EXAMPLE 3: SET COVERING - AIRLINE CREW SCHEDULING

Consider the following simplified airline crew scheduling problem. An airline has m scheduled flight-legs per week in its current service. A flight-leg being a single flight flown by a single crew e.g. London - Paris leaving Heathrow at 10.30 am. Let S_j, $j = 1,...n$ be the collection of all possible weekly sets of flight-legs that can be flown by a single crew. Such a subset must take account of restrictions like a crew arriving in Paris at 11.30 am. cannot take a flight out of New York at 12.00 pm. and so if c_j is the cost of set S_j of flight-legs then the problem of minimising cost subject to covering all flight-legs is a set covering problem. Note that if crews are not allowed to be passengers on a flight i.e. so that they can be flown to their next flight, then we have to make (2) an equality – the set partitioning problem.

EXAMPLE 4: SET COVERING - BUILDING FIRE STATIONS

There are six cities in region R. The region must determine where to build fire stations. The region wants to build the minimum number of fire stations and ensure that at least one fire station is near 15 minutes of each city. The times (in minutes) required to drive between cities are:

From	To					
	1	2	3	4	5	6
1	0	10	20	30	30	20
2	10	0	25	35	20	10
3	20	25	0	15	30	20
4	30	35	15	0	15	25
5	20	20	30	15	0	14
6	20	10	20	25	14	0

$$x_i = \begin{cases} 1 \text{ if a fire station is built in city i} \\ 0 \text{ otherwise} \end{cases}$$

Objective function: Total number of fire stations to be built

$$x_0 = x_1 + x_2 + x_3 + x_4 + x_5 + x_6$$

Constraints: A fire station within 15 mins of each city.

The locations that can reach each city in 15 minutes:

City 1 1, 2 $\Rightarrow x_1 + x_2 \geq 1$ (City 1 constraint)

City 2	1, 2, 6	$\Rightarrow x_1 + x_2 + x_6 \geq 1$	(City 2 constraint)
City 3	3, 4		
City 4	3, 4, 5		
City 5	4, 5, 6		
City 6	2, 5, 6		

$$
\begin{array}{lrcccccccccl}
\text{IP: min} & x_0 = & x_1 & + & x_2 & + & x_3 & + & x_4 & + & x_5 + x_6 & \\
\text{subject to} & & x_1 & + & x_2 & & & & & & & \geq 1 \text{ (City 1)} \\
& & x_1 & + & x_2 & & & & & & + x_6 & \geq 1 \text{ (City 2)} \\
& & & & & & x_3 & + & x_4 & & & \geq 1 \text{ (City 3)} \\
& & & & & & x_3 & + & x_4 + x_5 & & & \geq 1 \text{ (City 4)} \\
& & & & & & & & x_4 + x_5 + x_6 & & & \geq 1 \text{ (City 5)} \\
& & & & x_2 & & & & & + & x_5 + x_6 & \geq 1 \text{ (City 6)}
\end{array}
$$

$$x_i = 1 \text{ or } 0;\ i = 1, \ldots, 6.$$

Optimal solution: $x_0 = 2$; $x_2 = x_4 = 1$, $x_1 = x_3 = x_5 = x_6 = 0$.

In a set covering problem each member of a given set (Set 1) must be "covered" by another member of some set (e.g. Set 2). The objective is to minimize the number of elements in Set 2 that are required to cover Set 1.

2. GENERAL TERMINOLOGY FOR INTEGER PROGRAMMING

The most general problem called the **mixed integer programming problem** can be specified as

$$\begin{aligned} \min \quad & x_0 = c^T x \\ \text{subject to} \quad & A x = b \\ & x_j \geq 0 \qquad j = 1, \ldots, n \\ & x_j \text{ integer for } j \in IN \end{aligned}$$

where IN is some subset of $N_0 = \{0,1,...n\}$.

When $IN = N_0$ we have what is called a **pure integer programming problem**. For such a problem, one generally has all given quantities c_j, a_{ij}, b_i integer. One has to be careful here. Consider for example

$$\begin{aligned} \min \quad & x_0 = -\tfrac{1}{3} x_1 - \tfrac{1}{2} x_2 \\ \text{subject to} \quad & \tfrac{2}{3} x_1 + \tfrac{1}{3} x_2 \leq \tfrac{4}{3} \\ & \tfrac{1}{2} x_1 - \tfrac{3}{2} x_2 \leq \tfrac{2}{3} \\ & x_0, x_1, x_2 \geq 0 \text{ and integer} \end{aligned}$$

As defined this is not a pure problem. For a start x_0 will not necessarily be integer and neither will the slack variables. If we want to use an algorithm for solving pure problems we must scale the objective and constraints to give:

$$\begin{aligned} \min \quad & x_0 = -2x_1 - 3x_2 \\ \text{subject to} \quad & 2x_1 + x_2 + x_3 = 4 \\ & 3x_1 - 9x_2 + x_4 = 4 \\ & x_1, \ldots x_4 \geq 0 \text{ and integer.} \end{aligned}$$

A final class of problems is the **pure 0 – 1 programming problem**

$$\begin{aligned} \max \quad & x_0 = c^T x \\ \text{subject to} \quad & A x \leq b \\ & x_j = 0 \text{ or } 1 \quad \text{for } j = 1, \ldots, n. \end{aligned}$$

3. FURTHER USES OF INTEGER VARIABLES

1. If a variable **x can only take a finite number of values $p_1, \ldots p_m$** we can replace x by the expression

$$p_1 w_1 + ... + p_m w_m$$

where $w_1 + ... + w_m = 1$ and $w_i = 0$ or 1; $1 = 1, ...m$

For example, x might be the output of a plant which can be small p_1, medium p_2 or large p_3. The cost c(x) of the plant could be represented by

$$c_1 w_1 + c_2 w_2 + c_3 w_3$$

where c_1 is the cost of a small plant etc.

2. In LP, one generally considers all constraints to be holding simultaneously. It is **possible that the variables might have to satisfy one or other of a set of constraints**, e.g.

(a) $0 \leq x \leq M$
$0 \leq x \leq 1$ or $x \geq 2$

can be expressed

$$x \leq 1 + (M - 1)\delta$$
$$x \geq 2 + M(\delta - 1)$$
$$x \geq 0 \quad \delta = 0 \text{ or } 1$$

$x \leq M$ is a notional upper bound to make this approach possible.

(b) $x_1 + x_2 \leq 4$
$x_1 \geq 1$ or $x_2 \geq 1$ but not both ≥ 1
$x_1, x_2 \geq 0$

can be expressed

$$x_1 + x_2 \leq 4$$
$$x_1 \geq \delta$$
$$x_2 \geq 1 - \delta$$
$$x_1 \leq (1 - \delta) + 4\delta$$
$$x_2 \leq \delta + 4(1 - \delta)$$
$$\delta = 0 \text{ or } 1$$

Integer programming problems generally take much longer to solve than the corresponding linear program obtained by ignoring integrality. It is wise therefore to consider the possibility of solving as a straight forward LP and then rounding e.g. in the trim – loss problem. This is not always possible for example if x_1 is a $0 - 1$ variable such that $x_1 = 0$ means do not build a plant and $x_1 = 1$ means build a plant rounding $x_1 = 1/2$ is not very satisfactory.

4. CUTTING PLANE ALGORITHM FOR PURE INTEGER PROGRAMMING

The rationale behind this approach is :

1. Solve the **continuous problem as an LP i.e. ignore integrality**.
2. If by chance the optimal basic variables are all integer then the optimum solution has been found. Otherwise:
3. **Generate a cut** i.e. a constraint which is satisfied by all integer solutions to the problem but not by the current L.P. solution.
4. **Add this new constraint** and go to (1).

It is straight forward to show that if at any stage the current L.P. solution x is integer it is the optimal integer solution. This is because x is optimal over a region containing all feasible integer solutions. The problem is to define cuts that ensure the convergence of the algorithm in a finite number of steps. The first finite algorithm was devised by **Gomory**.

The algorithm is based on the following construction of a 'cutting plane': let

$$a_1 x_1 + ... + a_n x_n = b \qquad (3)$$

be an equation which is to be satisfied by non-negative integers $x_1, \ldots x_n$ and let S be the set of possible solutions.

For a real number ξ we define $\lfloor \xi \rfloor$ **to be the largest integer** $\leq \xi$. Thus, $\xi = \lfloor \xi \rfloor + \epsilon, \quad 0 \leq \epsilon < 1.$

$$\lfloor 6\tfrac{1}{2} \rfloor = 6; \quad \lfloor 3 \rfloor = 3; \quad \lfloor -4\tfrac{1}{2} \rfloor = -5$$

Now let $a_j = \lfloor a_j \rfloor + f_j$ and $b = \lfloor b \rfloor + f$ in (3) then we have

$$\sum_{j=1}^{n} (\lfloor a_j \rfloor + f_j)\, x_j = \lfloor b \rfloor + f$$

and hence

$$\sum_{j=1}^{n} f_j x_j - f = \lfloor b \rfloor - \sum_{j=1}^{n} \lfloor a_j \rfloor x_j \tag{4}$$

Now for $x \in S$ the right hand side of (4) is clearly integer and so $\xi = \Sigma f_j x_j - f$ is integer for $x \in S$. Since $x \geq 0$ for $x \in S$ we also have $\xi \geq -f > -1$ and since ξ is integer we deduce that $\xi \geq 0$ and that

$$\sum_{j=1}^{n} f_j x_j \geq f \qquad \text{for } x \in S$$

Suppose now that one has solved the continuous problem in step 1 of our cutting plane algorithm and the solution is not integer. Therefore there is a basic variable x_i with

$$x_i + \sum_{j \notin I} b_{ij} x_j = b_{i0} \text{where } b_{i0} \text{ is not integer.}$$

Putting $f_j = b_{ij} - \lfloor b_{ij} \rfloor$ and $f = b_{i0} - \lfloor b_{i0} \rfloor$ and we deduce that

$$\sum_{j \notin I} f_j x_j \geq f \tag{5}$$

for all integer solutions to our problem.

Now $f > 0$ since b_{i0} is not integer and so (5) is not satisfied by the current LP solution since $x_j = 0$ for $j \notin I$ and so (5) is a cut.

STATEMENT OF THE CUTTING PLANE ALGORITHM

The initial continuous problem solved by the algorithm is the LP problem obtained by ignoring integrality.

Step 1 Solve current continuous problem.

Step 2 If the solution is integral it is the optimal integer solution, otherwise

Step 3 Choose a basic variable x_i which is currently non-integer. Construct the corresponding constraint (5) and add it to the problem. Go to step 1. (We note that the tableau obtained after adding the cut is dual feasible and so the dual simplex algorithm is used to re-optimise. Nevertheless, we shall not discuss the dual simplex algorithm and confine ourselves to the phase 1 and phase 2 simplex algorithm.)

The basic approach is illustrated in FIGURE 1 below.

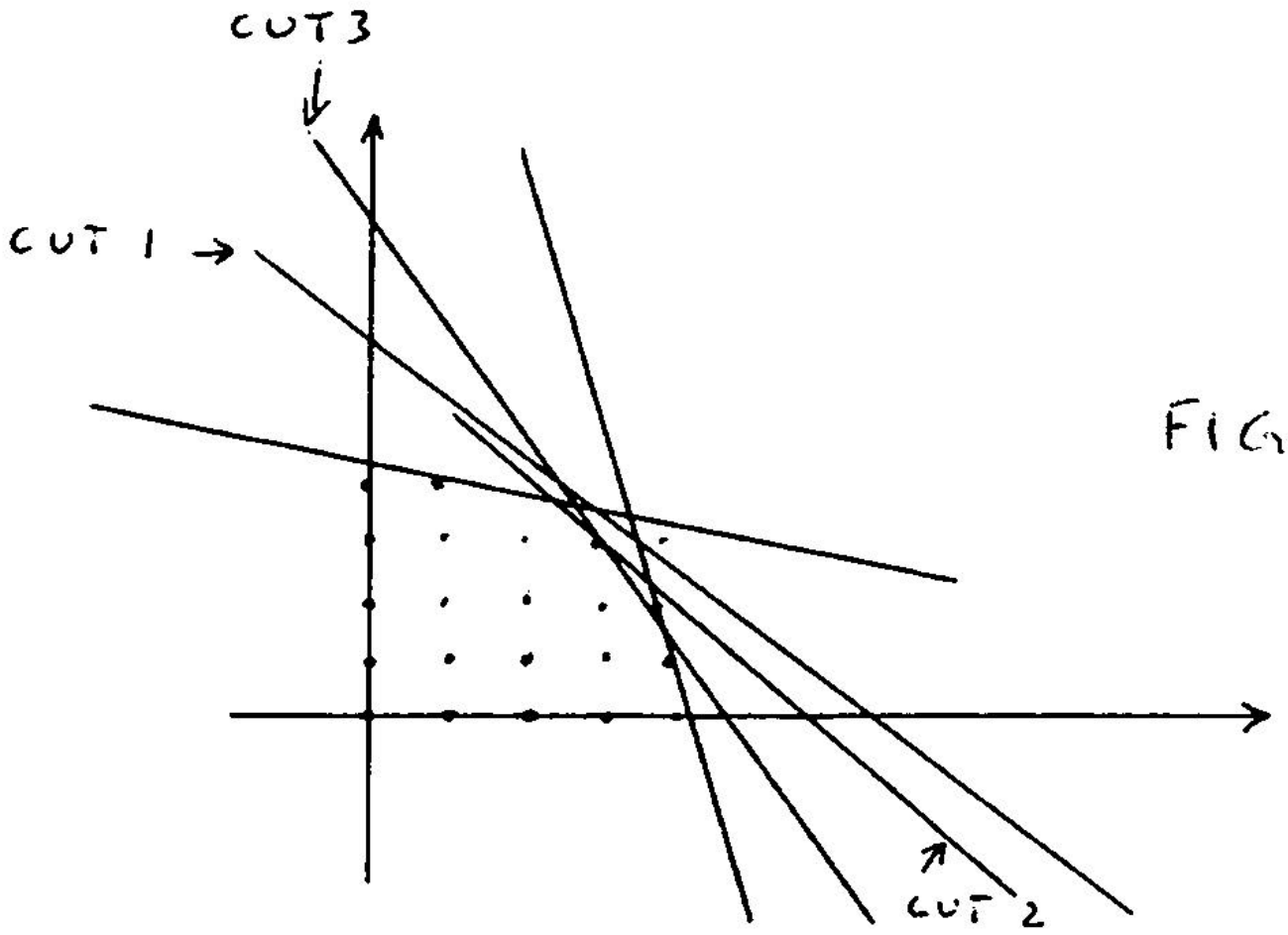


FIG 1

EXAMPLE 5:

$$\begin{aligned} \max \quad & 3x_1 + 4x_2 \\ \text{subject to} \quad & \tfrac{2}{5}x_1 + x_2 \le 3 \\ & \tfrac{2}{5}x_1 - \tfrac{2}{5}x_2 \le 1 \\ & x_1, x_2 \ge 0 \text{ and integer} \end{aligned}$$

to ensure slacks are integer as well, we eliminate the noninteger coefficients

$$\begin{aligned} \max \quad & 3x_1 + 4x_2 \\ \text{subject to} \quad & 2x_1 + 5x_2 \le 15 \\ & 2x_1 - 2x_2 \le 5 \\ & x_1, x_2 \ge 0 \text{ and integer} \end{aligned}$$

$$\begin{aligned} \text{or, subject to} \quad & 2x_1 + 5x_2 + x_3 = 15 \\ & 2x_1 - 2x_2 + x_4 = 5 \\ & x_1, x_2, x_3, x_4 \ge 0 \text{ and integer} \end{aligned}$$

BV	x_1	x_2	x_3	x_4	x_5	x_6	x_7	RHS
x_0	-3	-4						0
x_3	2	$\boxed{5}$	1					15
x_4	2	-2		1				5

↓ 'continuous relaxation'

x_0	$-\frac{7}{5}$		$\frac{4}{5}$					12
x_2	$\frac{2}{5}$	1	$\frac{1}{5}$					3
x_4	$\boxed{\frac{14}{5}}$		$\frac{2}{5}$	1				11

↓ 'continuous relaxation' & first continuous optimum

x_0			1	$\frac{1}{2}$				$\frac{35}{2}$
x_2		1	$\frac{1}{7}$	$-\frac{1}{7}$				$\frac{10}{7}$
x_1	1		$\frac{1}{7}$	$\frac{5}{14}$				$\frac{55}{14}$

cut based on x_1 row

$\frac{1}{7}x_3 + \frac{5}{14}x_4 \ge \frac{13}{14}$, or

$\frac{1}{7}(15-2x_1-5x_2) + \frac{5}{14}(5-2x_1+2x_2) \ge \frac{13}{14}$

$\Rightarrow \quad x_1 \le 3$

$x_5 = -\frac{13}{14} + \frac{1}{7}x_3 + \frac{5}{14}x_4 = 3 - x_1$

add first cut and solve the feasibility problem

x_0			1	$\frac{1}{2}$				$\frac{35}{2}$
x_2		1	$\frac{1}{7}$	$-\frac{1}{7}$				$\frac{10}{7}$
x_1	1		$\frac{1}{7}$	$\frac{5}{14}$				$\frac{55}{14}$
ξ			$\frac{1}{7}$	$\boxed{\frac{5}{14}}$	-1			$\frac{13}{14}$

↑ Current solution infeasible, solve - phase 1 problem.

BV	x_1	x_2	x_3	x_4	x_5	x_6	x_7	RHS
x_0			$\frac{4}{5}$		$\frac{7}{5}$			$\frac{81}{5}$
x_2		1	$\frac{1}{5}$		$-\frac{2}{5}$			$\frac{9}{5}$
x_1	1				1			3
x_4			$\frac{2}{5}$	1	$-\frac{14}{5}$			$\frac{13}{5}$

continuous optimum attained with cut
second cut based on x_2 row

$\frac{1}{5}x_3 + \frac{3}{5}x_5 \geq \frac{4}{5}$, or
$\frac{1}{5}(15-2x_1-5x_2) + \frac{3}{5}(3-x_1) \geq \frac{4}{5}$
$\Rightarrow x_1 + x_2 \leq 4$
$x_6 = \frac{1}{5}x_3 + \frac{3}{5}x_5 - \frac{4}{5} = 4 - x_1 - x_2$

x_0			$\frac{4}{5}$		$\frac{7}{5}$			$\frac{81}{5}$
x_2		1	$\frac{1}{5}$		$-\frac{2}{5}$			$\frac{9}{5}$
x_1	1				1			3
x_4			$\frac{2}{5}$	1	$-\frac{14}{5}$			$\frac{13}{5}$
ξ			$\frac{1}{5}$		$\boxed{\frac{3}{5}}$	-1		$\frac{4}{5}$

↑ Current solution infeasible, solve - phase 1 problem.

x_0			$\frac{1}{3}$			$\frac{7}{3}$		$\frac{43}{3}$
x_2		1	$\frac{1}{3}$			$-\frac{2}{3}$		$\frac{7}{3}$
x_1	1		$-\frac{1}{3}$			$\frac{5}{3}$		$\frac{5}{3}$
x_4			$\frac{4}{3}$	1		$-\frac{14}{3}$		$\frac{19}{3}$
x_5			$\frac{1}{3}$		1	$-\frac{5}{3}$		$\frac{4}{3}$

continuous optimum attained with cut
third cut based on x_2 row

$\frac{1}{3}x_3 + \frac{1}{3}x_6 \geq \frac{1}{3}$, or
$\frac{1}{3}(15-2x_1-5x_2) + \frac{1}{3}(4 - x_1 - x_2) \geq \frac{1}{3}$
$\Rightarrow x_1 + 2x_2 \leq 6$
$x_7 = \frac{1}{3}x_3 + \frac{1}{3}x_6 - \frac{1}{3} = 6 - x_1 - 2x_2$

BV	x_1	x_2	x_3	x_4	x_5	x_6	x_7	RHS
x_0			$\frac{1}{3}$			$\frac{7}{3}$		$\frac{43}{3}$
x_2		1	$\frac{1}{3}$			$-\frac{2}{3}$		$\frac{7}{3}$
x_1	1		$-\frac{1}{3}$			$\frac{5}{3}$		$\frac{5}{3}$
x_4			$\frac{4}{3}$	1		$-\frac{14}{3}$		$\frac{19}{3}$
x_5			$\frac{1}{3}$		1	$-\frac{5}{3}$		$\frac{4}{3}$
ξ			$\boxed{\frac{1}{3}}$			$\frac{1}{3}$	-1	$\frac{1}{3}$

x_0						2	1	14
x_2		1				-1	1	2
x_1	1					2	-1	2
x_4				1		-6	4	5
x_5					1	-2	1	1
x_3			1			1	-3	1

EXAMPLE 6:

max $x_1 + 4x_2$ (6)

subject to

$$2x_1 + 4x_2 \leq 7$$

$$10x_1 + 3x_2 \leq 14$$

$$x_1, x_2 \geq 0$$

BV	x_1	x_2	x_3	x_4	x_5	x_6	RHS
x_0	-1	-4					0
x_3	2	$\boxed{4}$	1				7
x_4	10	3		1			14

↓ 'continuous relaxation' solved by one pivot, solution noninteger: add a cut.

x_0	1		1				7
x_2	$\frac{1}{2}$	1	$\frac{1}{4}$				$\frac{7}{4}^{*}$
x_4	$\frac{17}{2}$		$-\frac{3}{4}$	1			$\frac{35}{4}$
$\xi^{(a)}$	$\frac{1}{2}$		$\boxed{\frac{1}{4}}^{(b)}$		-1		$\frac{3}{4}$

↑ current solution infeasible, solve a phase 1 problem.

x_0	-1			4		4
x_2		1		1		1
x_4	$\boxed{10}$		1	-3		11
x_3	2		1	-4		3

↓ 2nd 'continuous' problem solved, but noninteger, add a cut.

x_0				$\frac{1}{10}$	$\frac{37}{10}$		$\frac{51}{10}$
x_2		1			1		1
x_1	1			$\frac{1}{10}$	$-\frac{3}{10}$		$\frac{11}{10}$
x_3			1	$-\frac{1}{5}$	$-\frac{17}{5}$		$\frac{4}{5}$
$\xi^{(c)}$				$\boxed{\frac{1}{10}}$	$\frac{7}{10}$	-1	$\frac{1}{10}$

↑ Current solution infeasible, solve - phase 1 problem.

x_0					3		5
x_2		1			1		1
x_1	1				-1	1	1
x_3			1		-1	-2	1
x_4				1	7	-10	1

(a) Cut generated is $\frac{1}{2}x_1 + \frac{1}{4}x_3 \geq \frac{3}{4}$ or $\frac{1}{2}x_1 + \frac{1}{4}x_3 - x_5 = \frac{3}{4}$. Adding artificial ξ to this, in order to get basic feasible solution to a augmented set of equations, gives $\frac{1}{2}x_1 + \frac{1}{4}x_3 - x_5 + \xi = \frac{3}{4}$.
(b) Choose a pivot that will reduce ξ (pivoting in column 1 is allowable, but the arithmetic is worse).
(c) Cut generated is $\frac{1}{10}x_4 + \frac{7}{10}x_5 - x_6 = \frac{1}{10}$.

It is useful to see what has happened graphically. We first express the cuts in terms of x_1, x_2.

Cut No.1.

$$\tfrac{1}{2}x_1 + \tfrac{1}{4}x_3 \geq \tfrac{3}{4}$$

since

$x_3 = 7 - 2x_1 - 4x_2$ this becomes $x_2 \leq 1$.

Cut No.2.

After re-arranging, this becomes $x_1 + x_2 \leq 2$.

The constraints and cuts are illustrated in FIGURE 2 below where
A is the optimal solution ignoring integrality.

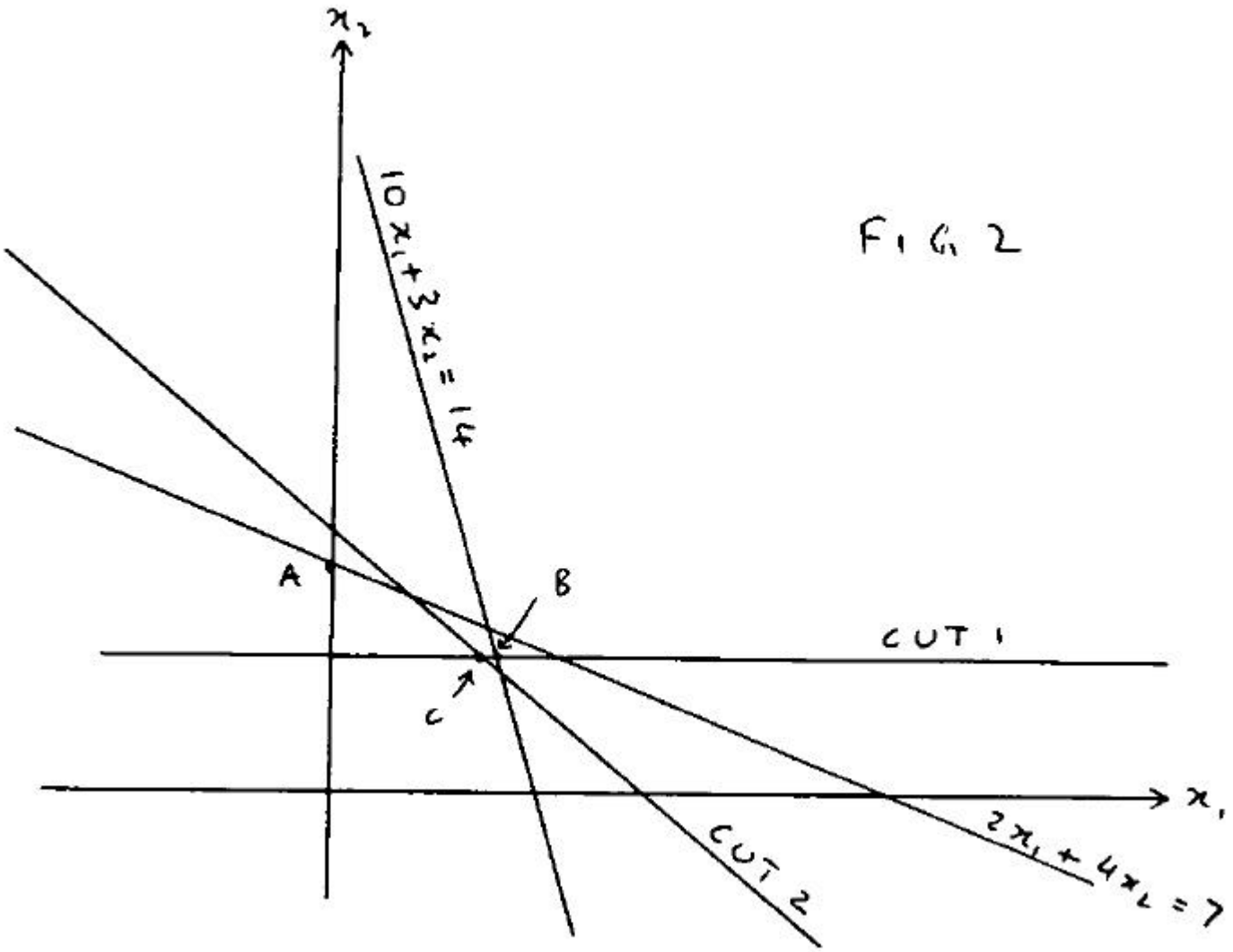


FIG 2

B is the optimal solution after adding cut 1.
C is the optimal integer solution found after adding cut 2.

One can show that the **Gomory** cuts $\Sigma\ f_j\ x_j\ >\ f$ when expressed in terms of the original non-basic variables have the form $\Sigma\ w_j\ x_j\ \leq\ W$ where the w_j, W are integers and the value of $\Sigma\ w_j\ x_j$ after solving the current continuous problem is $w\ +\ \epsilon$ where $0\ <\ \epsilon\ <\ 1$ assuming the current solution non-integer. Thus the cut is obtained by moving a hyperplane parallel to itself to an extent which cannot exclude an integer solution. It is worth noting that the plane can usually be moved further without excluding integer points thus generating deeper cuts. For a discussion on how this can be done refer to specialist books on integer programming.

Further Remarks

1. [As the dual simplex algorithm is not covered in this course, you may choose to ignore this remark.] After adding a cut and carrying out one iteration of the dual simplex algorithm the slack variable corresponding to this cut becomes nonbasic. If during a succeeding iteration this slack variable becomes basic then it may be discarded along with its current row without affecting termination. This means that the tableau never has more than $m + 1$ rows or $m + n$ columns.

2. A valid cut can be generated from any row containing a non-integral variable. One strategy is to choose the variable with the largest fractional part as this helps to produce a 'large' change in the objective value. It is interesting that finiteness of the algorithm has not been proved for this strategy although finiteness has been proved for the strategy of always choosing the 'topmost' row of the tableau with a non-integer variable.

3. The behaviour of this algorithm has been erratic. For example, it has worked well on set covering problems but in other cases the algorithm has to be terminated because of excessive use of computer time. This raises an important point; if the algorithm is stopped prematurely then one does not have a good sub-optimal solution to use. Thus in some sense the algorithm is unreliable.

5. BRANCH AND BOUND ALGORITHM FOR INTEGER PROGRAMMING

The method to be described in this section constitutes the most successful method applied to date. The idea is quite general and has been applied to many other discrete optimisation problems (e.g. travelling salesman, job shop scheduling).

Let us assume we are trying to solve the mixed integer problem. Let us call this problem P_0. The first step is to solve the 'continuous' LP problem obtained by ignoring the integrality constraints. If in the optimal solution, one or more of the integer variables turn out to be non-integer, we choose one such variable and use it to split the given problem P_0 into two 'sub-problems' P_1 and P_2. Suppose the variable chosen is y_j and it takes the non-integral value β_j in the continuous optimum.
Then P_1 and P_2 are defined as follows:

$$\mathbf{P_1\ \equiv\ P_0\ with\ the\ added\ constraint\ y_j\ \leq\ \lfloor \beta_j \rfloor}$$

$$\mathbf{P_2\ \equiv\ P_0\ with\ the\ added\ constraint\ y_j\ \geq\ \lfloor \beta_j \rfloor\ +\ 1}$$

Now any solution to P_0 is either a solution of P_1 or P_2 and so P_0 can be solved by solving P_1 and P_2. We continue by solving the LP problems associated with P_1 and P_2. We then choose one of the
problems and if necessary split it into two sub-problems as was done with P_0. The principle of branch and bound is illustrared in FIGURE 3 below.

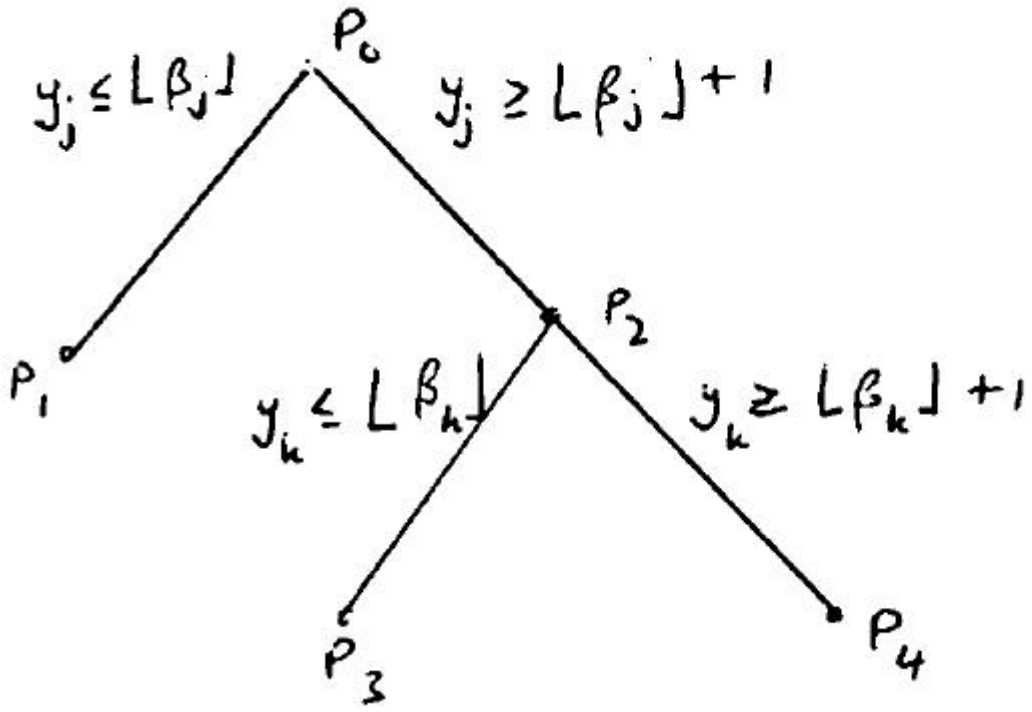

P_0
$y_j \le \lfloor \beta_j \rfloor$
$y_j \ge \lfloor \beta_j \rfloor + 1$
P_1
P_2
$y_k \le \lfloor \beta_k \rfloor$
$y_k \ge \lfloor \beta_k \rfloor + 1$
P_3
P_4

FIG. 3

This process can be viewed as the construction of a binary tree of sub-problems whose terminal (pendant) nodes correspond to the problems that remain to be solved.

In an actual computation one keeps a list of the unsolved problems into which the main problem has been split. One also keeps a note of the objective value MIN of the best integer solution found so far.

Step 0: Initially the list consists of the initial problem P_0. Put MIN equal to either the value of some known integer solution, or if one is not given equal to some upper bound calculable from initial data, if neither possibility is possible put MIN $= \infty$.

Solve the L.P. problem associated with P_0. If the solution has integer values for all integer variables terminate, otherwise

Step 1: Remove a problem P from the list whose optimal continuous objective function value x_0 is greater than MIN. If there are no such problems terminate. The best integer solution found so far is optimal. If none have been found the problem is infeasible.

Step 2: Amongst the integer variables in problem P with non-integer values in the optimal continuous solution for P select one for branching. Let this variable be y_p and let its value in the continuous solution be β.

Step 3: Create two new problems P' and P" by adding the extra restrictions $y_p \leq \lfloor \beta \rfloor$ and $y_p \geq \lfloor \beta \rfloor + 1$ respectively. Solve the LP problems associated with P' and P" and add these problems to the list. If a new and improved integer solution is found store it and update MIN. The new LP problems do not have to be solved from scratch but can be re-optimised using the dual algorithm (or parametrically altering the bound on y_p). If during the re-optimisation of either LP problem the value of the objective function exceeds MIN this problem may be abandoned. Go to step 1.

If one assumes that each integer variable in P_0 has a finite upper bound (equal to some large number for notionally unbounded variable) then the algorithm must terminate eventually, because as one proceeds further down the tree of problems the bounds on the variables become tighter and tighter, and these would eventually become exact if the LP solutions were never integer.

EXAMPLE 7:

As an example we show a possible tree for solving

Minimise $20 - 3x_1 - 4x_2$

Subject to

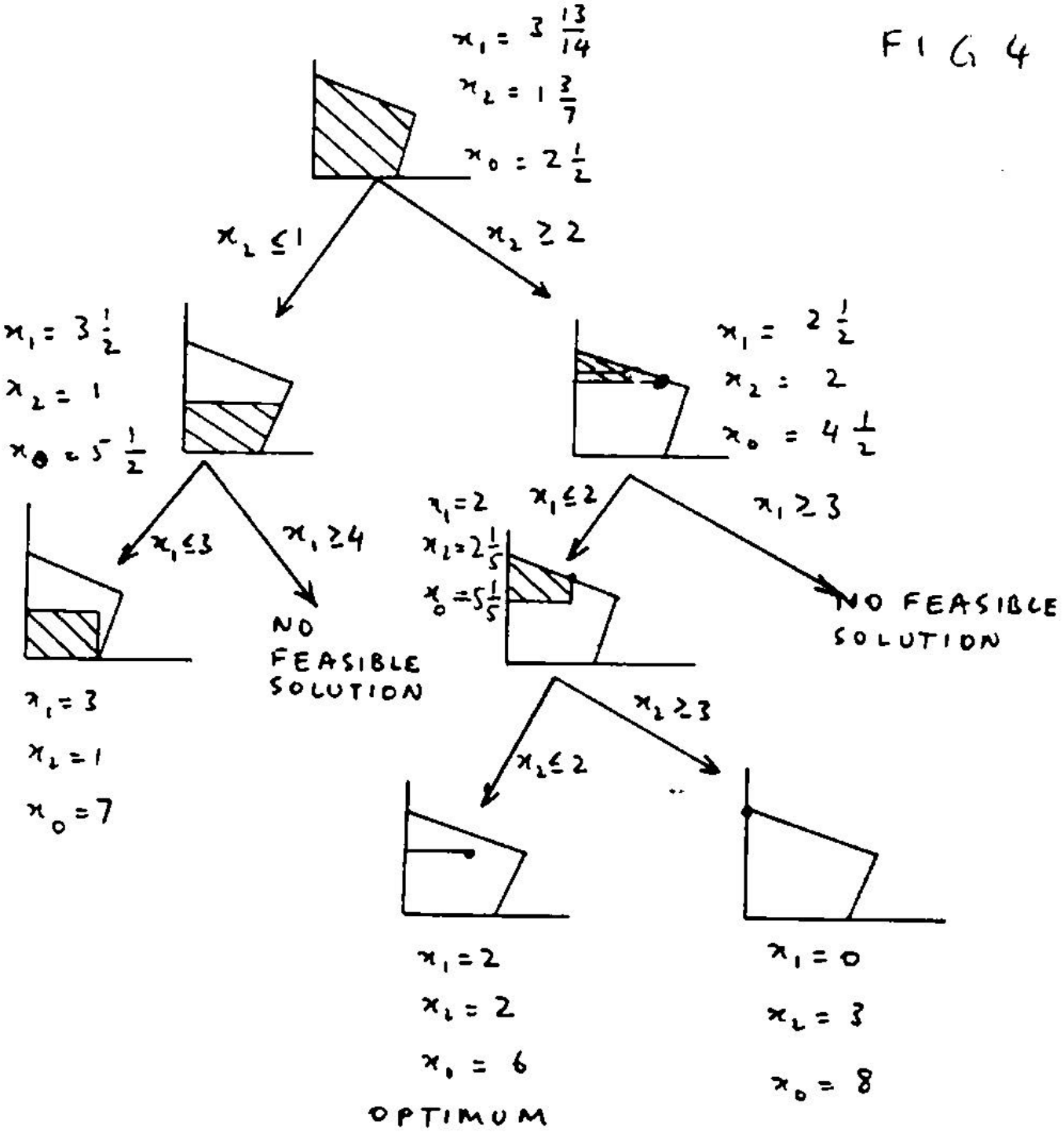
$$\tfrac{2}{5} x_1 + x_2 \leq 3$$

$$\tfrac{2}{5} x_1 - \tfrac{2}{5} x_2 \leq 1$$

$x_1, x_2 \geq 0$ and x_1, x_2 integer

The application of branch and bound to this problem is described in FIGURE 4 below.

FIG 4


$x_1 = 3\frac{13}{14}$
$x_2 = 1\frac{3}{7}$
$x_0 = 21\frac{1}{2}$
$x_2 \leq 1$
$x_2 \geq 2$
$x_1 = 3\frac{1}{2}$
$x_2 = 1$
$x_0 = 5\frac{1}{2}$
$x_1 = 2\frac{1}{2}$
$x_2 = 2$
$x_0 = 4\frac{1}{2}$
$x_1 \leq 3$
$x_1 \geq 4$
NO FEASIBLE SOLUTION
$x_1 = 2$
$x_2 = 2\frac{1}{5}$
$x_0 = 5\frac{1}{5}$
$x_1 \leq 2$
$x_1 \geq 3$
NO FEASIBLE SOLUTION
$x_1 = 3$
$x_2 = 1$
$x_0 = 7$
$x_2 \geq 3$
$x_2 \leq 2$
$x_1 = 2$
$x_2 = 2$
$x_0 = 6$
OPTIMUM
$x_1 = 0$
$x_2 = 3$
$x_0 = 8$