Integer Programming

Why Use Integer Programming

- Discrete Inputs & Outputs
- Problems with Logical Conditions
- Combinatorial Problems
- Non-linear Problems
- Network Problems

Integer Programming

- **Used to model:**
  - Traveling Salesman Problem
  - Constraint Satisfaction Problem
  - Robotic motion problems
  - Clustering
  - Multiple sequence alignment
  - Haplotype inferencing
  - VLSI circuit design
  - Computer disk read head scheduling
  - Derivation of physical structures of programs
  - Delay-Tolerant Network routing
  - Cellular radio network base station locations
  - Minimum-energy multicast problem in wireless ad hoc networks

Integer Programming definition:

Simply stated, an Integer Programming problem (IP) is a Linear Programming (LP) in which some or all the variables are required to be nonnegative integers.

An integer programming problem in which all variables are required to be integer is called a *pure integer programming problem*.

If some variables are restricted to be integer and some are not then the problem is a *mixed integer programming problem.*

Integer Programming definition:

The case where the integer variables are restricted to be 0 or 1 comes up surprisingly often.

There are two types of models:

Pure (mixed) 0-1 programming problems or *pure (mixed) binary integer programming problems*.

Integer programming problems (IPs) require much more sophisticated mathematical algorithms for solution than do linear programming problems.

Types of Integer Programming Models

- An LP in which all the variables are restricted to be integers is called an all-integer linear program (ILP).
- The LP that results from dropping the integer requirements is called the LP Relaxation of the ILP.
- If only a subset of the variables are restricted to be integers, the problem is called a mixed-integer linear program (MILP).
- Binary variables are variables whose values are restricted to be 0 or 1. If all variables are restricted to be 0 or 1, the problem is called a 0-1 or binary integer linear program.

Types of Integer Programming Models

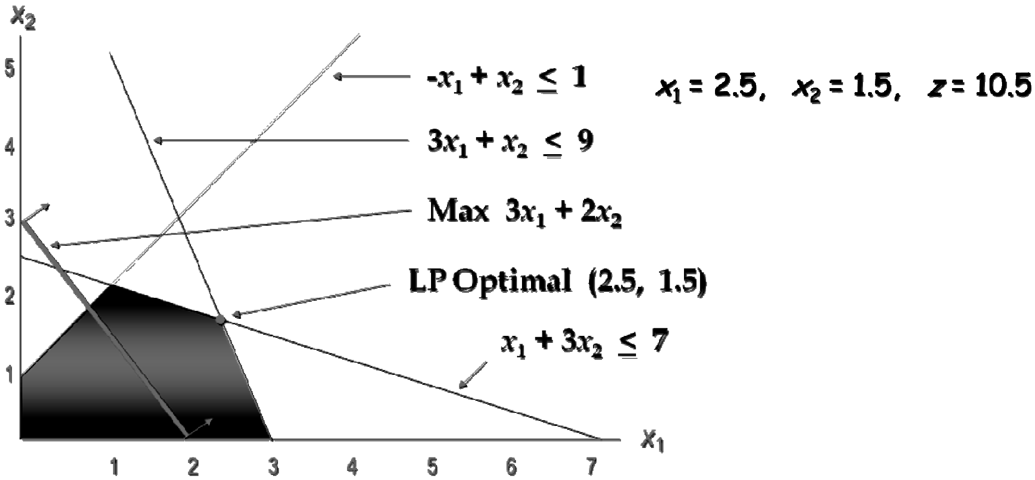
- Consider the following all-integer linear program:

$$\begin{aligned} \text{Max} \quad & 3x_1 + 2x_2 \\ \text{s.t.} \quad & 3x_1 + x_2 \leq 9 \\ & x_1 + 3x_2 \leq 7 \\ & -x_1 + x_2 \leq 1 \\ & x_1, x_2 \geq 0 \text{ and integer} \end{aligned}$$

Types of Integer Programming Models

- LP Relaxation

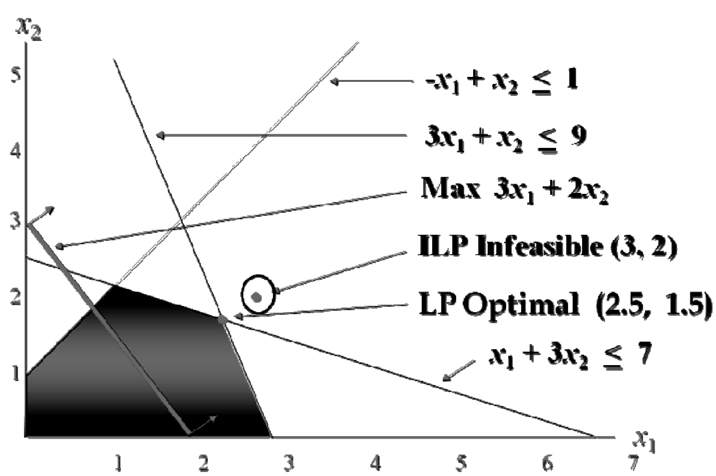
Solving the problem as a linear program ignoring the integer constraints, the optimal solution to the linear program gives fractional values for both x_1 and x_2. From the graph the optimal solution to the linear program is:



Types of Integer Programming Models

- Rounding Up

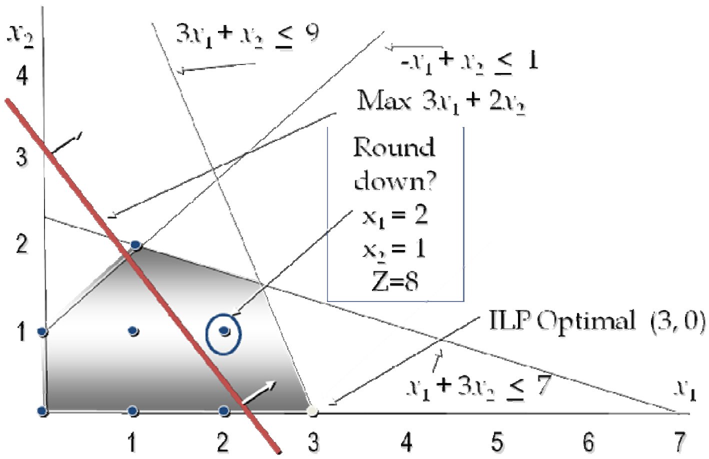
If we round up the fractional solution ($x_1 = 2.5$, $x_2 = 1.5$) to the LP relaxation problem, we get $x_1 = 3$ and $x_2 = 2$. From the graph on the next slide, we see that this point lies outside the feasible region, making this solution infeasible.



Types of Integer Programming Models

- Rounding Down

  By rounding the optimal solution down to $x_1 = 2$, $x_2 = 1$, we see that this solution indeed is an integer solution within the feasible region, and substituting in the objective function, it gives $z = 8$. We have found a feasible all-integer solution, but have we found the OPTIMAL all-integer solution?



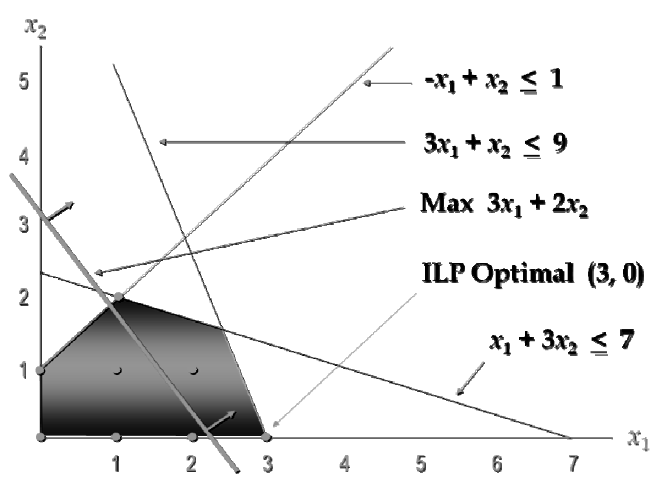
The answer is NO! The optimal solution is $x_1 = 3$ and $x_2 = 0$ giving z = 9, as shown in the next two slides.

Types of Integer Programming Models

Complete Enumeration of Feasible ILP Solutions

There are eight feasible integer solutions to this problem:

	x_1	x_2	z	
1.	0	0	0	
2.	1	0	3	
3.	2	0	6	
4.	3	0	9	<= Optimal solution
5.	0	1	2	
6.	1	1	5	
7.	2	1	8	
8.	1	2	7	



With a fast computer why not the enumeration method…?

- If there are n variables, then there are 2^n different ways to enumerate all possible points,
- Suppose that we could evaluate 1 billion solutions per second.
- Let n = number of binary variables

	Solutions times
n = 30,	1 second
n = 40,	17 minutes
n = 50	11.6 days
n = 60	31 years

Discrete Inputs & Outputs

- Continuous solution from LP is often rounded
- Is effective when answers are sufficiently large (errors will not be serious)
- Often can change the units to generate solution with large numbers
- Common application is when only discrete quantities
  - ie. 40 hrs, 80 hrs or 120 hrs (1,2 or 3 shifts)
  - Cannot use rounding for this
- Strictly IP models with single constraint should be solved with knapsack algorithm

Logical Conditions: Examples

- If product A is made then must make product B or C
- Must choose 1 of a set of possible activities
  - for example product mix application
- If you undertake an activity must do all of it
  - Must include all of ingredients
- Limit the number of ingredients in a blend

Must use ingenuity to formulate these problems.

Combinatorial Problems

- Have a very large number of feasible solutions arising from
  - Different orders of carrying out operations (permutations)
  - Allocating resources to different operations
- Two types
  - Sequencing problems (optimal ordering of operations)
  - Allocation problems
- Examples:
  - Depot location problem
  - Assembly line balancing problem
  - Capital budget allocation

Network Problems

- Critical path in a PERT network
- Road location problems

Solution Mechanism

Solution Mechanism

- Cutting Planes Method
  - First solve as an LP
  - If solution is integer – done
  - Else: Keep adding constraints until an integer solution is found or it is found to be infeasible
- Enumerative Methods
- Pseudo-Boolean Methods
- Branch and Bound Methods
- Network Problems

Solution Mechanism

- Branch and Bound Methods
  - Have been very successful for solving
  - Successive variables are chosen for rounding off and the resulting tree in searched
  - Method is continued until an integer solution is found
- Network Problems

Problems with IP

- Solution time and resources
  - Generate many possible solutions
  - May be unsolvable
- Must be very careful when formulating IP problems
- Look for other solutions if possible

Building IP Models

Types of Integer Variables

- Discrete variables
- Logical conditions
- Ordered sets of variables
- Extra conditions applied to LP Models

Discrete Variables

Three Types:

- Discrete quantities
  - Variable must be integer
  - Easy to apply
  - Use infrequently
- Decision variables
  - Usually 0 -1
- Indicator variables
  - 0-1 variables usually linked to continuous variables in the LP

Decision Variables

Indicate number of possible decisions, δ

- Examples:
  - Road built (1) or not built (0)
  - Second shift added (1) or not added (0)
- Usual Implementation
  - Variable is defined as integer
  - Variable has an upper bound of 1
- Not always 0-1
  - Decision might have 3 or more outcomes

Using Indicator Variables to Impose Costs or Restrictions

<u>Indicator Variables</u>

When extra conditions are imposed, 0-1 variables are introduced and linked to some of continues variables

<u>Used for:</u>

- Fixed costs
- Blending or proportion constraints
- Imposing a constraint conditionally

Indicator Variables

Example I – Fixed Costs:

- Cabinet factory can outsource cabinet doors or can produce them in the factory.
- If they produce the doors the have to specially set up the production line at a cost \$500, then it costs \$10 per door to produce them.

How would you model this problem?

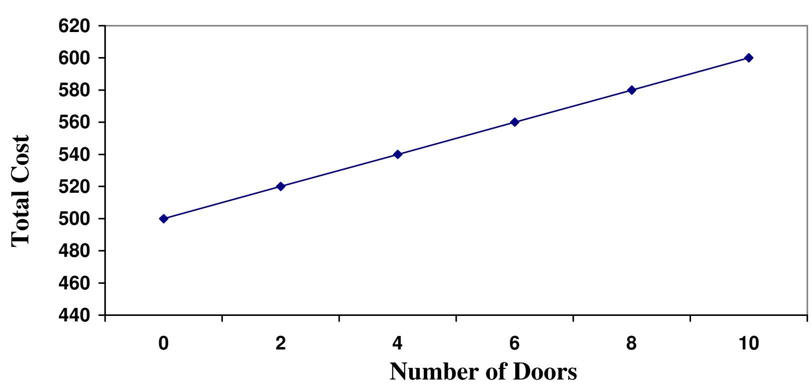
Understanding Fixed Costs

Situation:

Let x = Number of doors produced

If $\mathbf{x = 0}$ total manufacturing cost $\mathbf{= 0}$

If $\mathbf{x > 0}$ total manufacturing cost $\mathbf{= 500 + 10x}$



Formulation

Integer Variable

	Nbr Doors		Prod Indicator	Total Cost		RHS
Upper Bound			1			
Lower Bound			0			
Prod Cost					=	

…

Indicator can take on 2 values. What are they, and why?

Finish this formulation

Formulation

	Nbr Doors		Prod Indicator	Total Cost		RHS
Upper Bound			1			
Lower Bound			0			
Prod Cost	**10**		**500**	-1	=	0

How can you ensure that the indicator is 1 if doors are produced?

Finished Formulation

	Nbr Doors		Prod Indicator	Total Cost		RHS
Upper Bound			1			
Lower Bound			0			
Prod Cost	**10**		**500**	-1	**=**	0
Line Set Up	**1**		**-M**		$\leq$	0

Modeling Fixed Costs

Modeling:

- Introduce 0-1 indicator variable d
- Two constraints in model

$$10x + 500\delta - \text{TotalCost} = 0$$

$$x - M\delta \leq 0$$

Where: M is constant coefficient representing a known upper bound for x

M = very large penalty (999,999)

***Condition Imposed:* x > 0 if δ = 1**

Proportions

Example II - Blending:

A food is manufactured by refining raw oils and blending them together. The food may never be made up of more than three oils in any month. If either VEG1 or VEG2 are used in a month then OIL3 must also be used.

How would you model this problem?

> ***Note: the proportionality requirement alone can be handled by LP***

Modeling Proportions

Situation:

Let X_A = proportion of #3 Used

Let X_B = proportion of #1 Used

Condition:

if $X_A > 0$ then $X_B > 0$

Modeling:

> Same conditions as before on δ (integer, ub = 1)

$X_A - \delta \leq 0$

$X_B - 0.01\,\delta \geq 0$

> *m* some proportionate level (say 1/100) below which B as out of blend

> Proportion of X_B allowed before condition imposed (can be very small)

Conditional Constraints

- An indicator variable can be used to indicate whether an inequality holds or doesn't hold (not necessary)
- For example
  - If condition is true then constraint holds
  - If condition is false then constraint holds
- Can be done for
  - $\geq$ constraints
  - $\leq$ constraints
  - $=$ constraints

Conditional Constraints

Example III:

An olive oil company producer is considering which section to harvest. If road Y is built, they must harvest at least 10 000 kg from all of area serviced by that road.

How would you model this problem?

Modeling Conditional Constraints

Modeling:

Let:

Y_i = 0-1 indicator variable
0 – road not built
1 – road built

X_{ij} = Weight harvested from area j serviced by road i

b_i = Weight that must be harvested from area serviced by road i

Situation:

If road Y_i is built

$\Sigma X_{ij} \geq b_i$

Otherwise don't apply this constraint

Modeling Conditional Constraints

Situation:

If road Y_i is built

$\Sigma X_{ij} \geq b_i$

Otherwise don't apply this constraint

This condition can be modeled as:

$\Sigma X_{ij} + m Y_i \geq m + b_i$

Where:
m = penalty (negative)
lower bound on the expression
$\Sigma X_{ij} - b_i$ (so at least $-b_i$)

Modeling Conditional Constraints

In our example:

$b_i = 500$

So set $m = -600$

Constraint is:

$\Sigma X_{ij} + (-600)Y_i \geq (-600) + b_i$

If $Y_i = 0$

Constraint will always be satisfied

Otherwise, constraint holds

Similar Constraints

<u>If conditional constraint is:</u>

$\Sigma X_{ij} \leq b$ if $Y = 1$

Then the constraint is:

$\Sigma X_{ij} + MY_i \geq M + b_i$

Where:

M= penalty (positive)

upper bound on the expression

$\Sigma X_{ij} - b_i$

Similar Constraints

<u>You can also model the following:</u>

If $\Sigma X_{ij} \le b_i$ then $\delta = 1$

$$\rightarrow \Sigma X_{ij} - (m - \varepsilon)\,\delta \ge b_i + \varepsilon$$

If $\Sigma X_{ij} \ge b_i$ then $\delta = 1$

$$\rightarrow \Sigma X_{ij} - (M + \varepsilon)\,\delta \ge b_i - \varepsilon$$

Where:

M = upper bound on $\Sigma X_{ij} - b_i$

m = lower bound on $\Sigma X_{ij} - b_i$

ε = small allowed tolerance on constraint

Formulation with 0/1 (Binary) Variables

$PROJ_i$ = Indicator for "Project" i = $\begin{cases} 1 & \text{accept Project i} \\ 0 & \text{reject Project i} \end{cases}$

- When x_i and x_j represent binary variables designating whether projects i and j have been completed, the following special constraints may be formulated:
  - At most <u>k out of n</u> projects

    will be completed: $Sx_j \le k_j$

    Example: Must Choose at least one of Projects 5, 6, 7

    $PROJ_5 + PROJ_6 + PROJ_7 \ge 1$

  - Prerequisite Projects, j is <u>conditional</u> on project i:

    $x_j - x_i \le 0$

    Example: Project 4 cannot be done unless Project 2 is done

    $PROJ_2 \ge PROJ_4$ or $PROJ_2 - PROJ_4 \ge 0$

Formulation with 0/1 (Binary) Variables

$PROJ_i$ = Indicator for "Project" i = $\begin{cases} 1 & \text{accept Project i} \\ 0 & \text{reject Project i} \end{cases}$

- Co-requisite Projects Constraint

$$x_j - x_i = 0$$

Example: Project 1 and Project 2 have to be selected together

$$PROJ_1 = PROJ_2 \text{ or } PROJ_1 - PROJ_2 = 0$$

- Projects *i* and *j* are mutually exclusive:

$$x_i + x_j = 1$$

Example: Can't select both Project 1 and Project 3

$$PROJ_1 + PROJ_3 = 1$$

Example: Capital Budgeting

Suppose we have 6 projects. Let Y*i* be binary variables set to 1 if project *i* is invested in, and the interactions are as follows:

- Project 3 can only be done if project 2 is also done,
- We must invest in at least one of the first three projects,
- Only one of projects 2, 4 and 6 can be done,
- Exactly two of the last four projects must be invested in, but we do not care which ones are selected.

Solution

- **Project 3 can only be done if 2 is also done:** Add constraint: $y3 \leq y2$
- **We must invest in at least one of the first three projects:** Add constraint: $y1 + y2 + y3 \geq 1$
- **Only one of projects 2, 4 and 6 can be done:** Add constraint: $y2 + y4 + y6 \leq 1$
- **Exactly two of the last four projects must be invested in, but we do not care which ones:** Add constraint: $y3 + y4 + y5 + y6 = 2$

Modeling Logical Conditions with 0-1 Variables

Logical Conditions

Examples:

- If product A is manufactured then Product B or one of products C and D must be made
- Operation A must be finished before operation B starts
- If road is not built then it is not possible to harvest from a district
- No more than 10 products may be produced at any one time

Boolean Algebra Operators

$\vee$ or

$\cdot$ and

$\sim$ not

$\rightarrow$ if … then

$\leftrightarrow$ if and only if

Example From Manufacturing

- If one of products A or B are manufactured, then at least one of C, D, or E must also be manufactured.
- The logical condition is:

  $(X_A \vee X_B) \rightarrow (X_C \vee X_D \vee X_E)$
- The is accomplished using indicator variables
  - $\delta_{\mathbf{i}} = 1$ if product i is manufactured
  - $\delta = 1$ if the proposition $(X_A \vee X_B)$ holds

Logical Conditions

<u>1 st - We wish to impose the condition:</u>

$\delta_A + \delta_B \geq 1 \rightarrow \delta = 1$

This is done as follows:

$\delta_A + \delta_B - 2\delta \leq 0$ (1)

<u>2 nd - We wish to impose the condition:</u>

$\delta = 1 \rightarrow \delta_C + \delta_D + \delta_E \geq 1$

This is done as follows:

$-\delta_C - \delta_D - \delta_E + \delta \leq 0$ (2)

Modeling this with the LP

1. Introduce δA δB δC δD δE and link to original continuous LP variables as follows:

   $X_i - M\,\delta_i \leq 0$ for A, B, C, D, E

   (one constraint for each product)

2. Add additional constraints 1 and 2 on previous slide

   $\delta_A + \delta_B \quad -2\,\delta \leq 0$

   $-\delta_C - \delta_D - \delta_E + \delta \leq 0$

Modeling Polynomial Constraints

Suppose we want to model:

$\delta_1 \bullet \delta_2$

Do this with a new variable: δ_3

This can be modeled with 3 constraints:

$$-\delta_1 + \quad + \delta_3 \leq 0$$

$$-\delta_2 \quad + \delta_3 \leq 0$$

$$\delta_1 + \delta_2 \quad - \delta_3 \leq 1$$

New 0-1 variable δ_3 added

Summary of Logical Conditions

- Many modelers are unaware that logical conditions can be modeled with 0-1 variables
- It requires a bit of thought to set it up properly
- It's easy to model a restriction incorrectly. Use Boolean algebra and check to make sure the restriction works.

Special Ordered Sets of Variables

- Very common type of restriction
- Two types:

SOS1

- A set of variables in which exactly one must be non-zero

SOS2

- A set of variables in which at most 2 can be non-zero
- The two must be adjacent in the ordering given to the set

Examples

1. A depot can be sited at any one of three locations. Only one can be built.
2. The capacity of a plant can be extended only in discrete amounts

Example I: Depot Sitting

A Wine company is planning to build a depot to service customers from their western district.

The depot can be built at Location A, B, C, D, or E. Only one depot can be built.

How would you model this problem?

Modeling the Condition

Create 5 new 0-1 variables:

$\delta_i = 1$ if depot is located at site i

The set of variables can be considered a SOS1 set. A constraint is added:

$\delta_1 + \delta_2 + \delta_3 + \delta_4 + \delta_5 = 1$

Each δ_i is connected to a continuous variable in the usual way:

$X_{ij} - M \delta_i \leq 0$

where: X_{ij} is volume from factory j shipped to depot i

Example 2

The capacity of a plant can be increased in discrete amounts by increasing levels of investment.

How much capacity should the plant purchase?

A strict LP would allow continuously increasing the levels of capacity

How would you model this problem to deal with step levels?

Example 2

Modeling Discrete Capacity

Let: set of variables $\delta_1, \delta_2, \delta_3, \delta_4, \delta_5$ is regarded as an SOS1

then

$C_{\mathbf{i}}$ = Capacity at level i
$I_{\mathbf{i}}$ = Investment required for capacity i
δ_i = 1 if capacity level **I** is chosen

Modeling

$$C_{\mathbf{1}}\delta_1 + C_{\mathbf{2}}\delta_2 + C_{\mathbf{3}}\delta_3 + C_{\mathbf{4}}\delta_4 + C_{\mathbf{5}}\delta_5 = \mathbf{C}$$
$$I_{\mathbf{1}}\delta_1 + I_{\mathbf{2}}\delta_2 + I_{\mathbf{3}}\delta_3 + I_{\mathbf{4}}\delta_4 + I_{\mathbf{5}}\delta_5 = \mathbf{I}$$
$$\delta_1 + \delta_2 + \delta_3 + \delta_4 + \delta_5 = \mathbf{1}$$

Extra Conditions Applied to LP Models

Extra Conditions

- Requiring a subset of constraints to hold conditionally
- Limiting the number of variables in a solution

  $\boldsymbol{\delta_1 + \delta_2 + \delta_3 + \delta_4 + \delta_5 \leq k}$

  k is the permissible number of variables
- Sequentially dependent decisions
  - Used in multiple period models
  - For example decision to permanently close plant affects all other periods

Formulating IP Models

Good Modeling Practices

- IP models can be very difficult and time consuming to solve
- Solution time should always be considered when formulating the model
- It is common to build a model and then find out it cannot be solved
- Difficulty is a function of
  - Number of IP variables
  - Number of constraints

Number of IP Variables

- If a model has *n,* 0-1 variables there are 2^{n} possible solutions
- This gets very large with small values of n
- However, branch & bound solutions mechanisms surprising efficient

Number of Constraints

- Solution time of an LP is largely a function of the number of constraints
- However, IP or Mixed IP problems solve faster with *more* constraints

Restricting Feasible Region

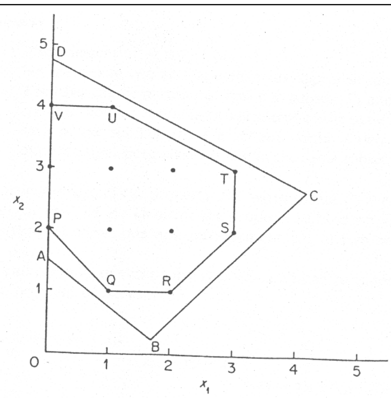


Figure 10.1

LP Feasible Region
IP Feasible Region
(all interior points in the solution space)

Fastest solution would be to restrict LP constraints to coincide with the IP Feasible Region

Enumeration Method

Unlike LP in IP's (where all variables are integers) each variable can only take a finite number of discrete (integer) values.

Hence the obvious solution approach is simply to *enumerate* all these possibilities - calculating the value of the objective function at each one and choosing the (feasible) one with the optimal value.

When variable size is small, problem has easy solution.

But it becomes very hard quickly as the problem size increases.

Example: Capital budgeting problem

There are four possible projects, which each run for 3 years with some characteristics. Which projects would you choose in order to maximise the total return?

Enumeration Method

Solution:

For example for the capital budgeting problem considered above there are $2^4=16$ possible solutions. These are:

x_1 x_2 x_3 x_4
0 0 0 0 do no projects

0 0 0 1 do one project
0 0 1 0
0 1 0 0
1 0 0 0

0 0 1 1 do two projects
0 1 0 1
1 0 0 1
0 1 1 0
1 0 1 0
1 1 0 0

x_1 x_2 x_3 x_4
1 1 1 0 do three projects
1 1 0 1
1 0 1 1
0 1 1 1

1 1 1 1 do four projects

Branch and Bound Method

Let's explore how to solve an integer programming problem (IP). Here are some terms that you should get to know:

Branching – dividing up the problem into different subsets, or make the problem into a decision tree

Bounding – using a *relaxed* LP solution to place a bound on the integer solutions for a given branch of the problem

Fathoming – eliminating a branch of the problem from further consideration

3 ways to fathom:

Fathom if bound is worse than the best integer solution found so far

Fathom if its relaxed LP has no feasible solutions

Fathom if optimal relaxed LP solution is integer (**all** variables integer)

Incumbent – the best integer solution seen thus far

Branch and Bound Method

Here are the steps of the algorithm:

1. **Initialize:** set incumbent solution to negative infinity (maximizing) or positive infinity (minimizing)
2. **Solve Root Node:** solve the linear programming relaxation of the original problem
3. **Branch:** pick a variable having non-integer value in the current solution
4. **Bound:** solve the relaxed LP for both branches just created to bound each of these subproblems.
5. **Fathom:** Check the bounds created in step 4 to see if you can eliminate any branches.
6. **Optimality Test:** Any remaining subproblems? If so, go to step 3. If not, current incumbent is optimal.

Example: Advertising cost

Dorian auto manufactures luxury cars and trucks. The company believes that its most likely customers are high-income women and men. To reach these groups, Dorian auto has embarked on an ambitious TV advertising campaign and has decided to purchase 1-minute commercial spots on two types of programs: comedy shows and football games. Each comedy commercial is seen by 7 million high income women and 2 million high income men. Each football commercial is seen by 2 million high-income women and 12 million high income men. A 1-minute comedy ad costs $50000 and a 1-minute football ad costs $100000. Dorian would like the commercials to be seen by at least 28 million high income women and 24 million high income men. Use IP to determine how Dorian Auto can meet its advertising requirements at min cost.

Solution:

Decision variables

x1: number of 1-minute comedy ads purchased

x2: number of 1-minute football ads purchased

Objective function

Total advertising cost = cost of comedy ads. + cost of football ads.

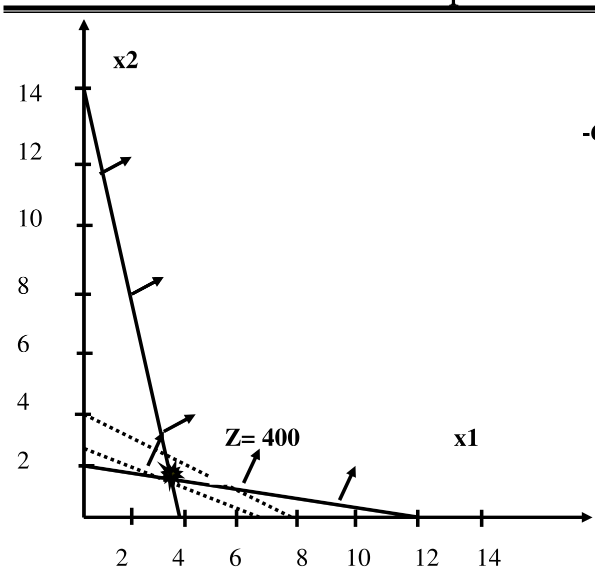
MIN **Z =** 50 x1 + 100 x2

S.T.: 7 x1 + 2 x2 ≥ 28

2 x1 + 12 x2 ≥ 24

x1, x2 ≥ 0

Example: Advertising cost



$$\left.\begin{aligned} -6\ /\ 7x1 + 2\ x2 &= 28 \\ 2x1 + 12\ x2 &= 24 \\ -40\ x1 &= 144 \end{aligned}\right\}$$

x1 = 3,6

x2 = 1,4

Z= 50 • 3.6 + 100 •1.4 = 320

Problem is solved firstly using Linear Programming approach and found an initial point for integer programming with branch and bound approach.

Example: Advertising cost

Integer Programming with Branch-Bound Approach

LPO	**Min**

X1 = 3.6 X2 = 1,4 Z =320

7X1 + 2 X2 ≥ 28
2X1 + 12 X2 ≥ 24
X1 ≤3

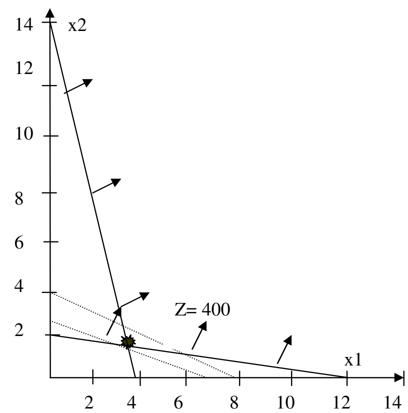
X1≤ 3

LP1
X1 = 3 , X2 = 1.5 ,Z =300
Z = 50 X1 + 100 X2

X1≥ 4

LP2
X1 = 4 , X2 = 1.3 , Z =330
Z = 50 X1 + 100 X2

7X1 + 2 X2 ≥ 28
2X1 + 12 X2 ≥ 24
X1 ≥4



X2≤ 1

Infeasible

7X1 + 2 X2 ≥ 28
2X1 + 12 X2 ≥ 24
X1 ≤3
X2 ≤ 1

X2 ≥ 2

Infeasible

7X1 + 2 X2 ≥ 28
2X1 + 12 X2 ≥ 24
X1 ≤3
X2 ≥ 2

Example: Advertising cost

Integer Programming with Branch-Bound Approach

LPO	Min

x1 = 3.6 x2 = 1,4 Z =320

7x1 + 2 x2 ≥ 28
2x1 + 12 x2 ≥ 24
x1 ≤3

x1≤ 3

LP1
X1 = 3 , x2 = 1.5 , Z =300

x2≤ 1

Infeasible

7x1 + 2 x2 ≥ 28
2x1 + 12 x2 ≥ 24
x1 ≤3
x2 ≤ 1

x2 ≥ 2

Infeasible

7x1 + 2 x2 ≥ 28
2x1 + 12 x2 ≥ 24
x1 ≤3
x2 ≥ 2

x1≥ 4

7x1 + 2 x2 ≥ 28
2x1 + 12 x2 ≥ 24
x1 ≥4

LP2
x1 = 4 , x2 = 1.3 , Z =330

x2≤ 1

LP3
x1 = 6 , x2 = 1 , Z =400

7x1 + 2 x2 ≥ 28
2x1 + 12 x2 ≥ 24
x1 ≥4
x2 ≤1

x2≥ 2

7x1 + 2 x2 ≥ 28
2x1 + 12 x2 ≥ 24
x1 ≥4
x2 ≥ 2

LP4
x1 = 4 , x2 = 12 , Z =400

The outcomes showed that LP solution for this problem is:
x1 = 3,6
x2 = 1,4
Z= 320
When we solved the problem with integer programming, the outcomes are:

x1 = 6
x2= 1
Z= 400

OR

x1 = 4
x2=2
Z= 400

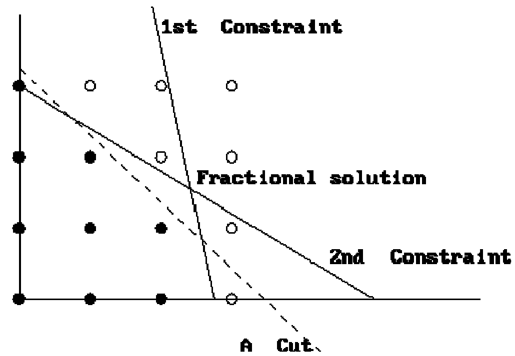
The z value increases a little with the integer programming aproach from 320 to 400.

Cutting Plane Techniques

A cutting planes Techniques is an alternative to branch and bound method which is also used to solve integer programs.

The fundamental idea behind cutting planes is to add constraints to a linear program until the optimal basic feasible solution takes on integer values.

1. every feasible integer solution is feasible for the cut, and
2. the current fractional solution is not feasible for the cut.



Cutting Plane Techniques

There are two ways to generate cuts.

1. **General Cutting Planes**
2. **Cuts for Special Structure**

General Cutting Planes : This is called Gomory cuts, generates cuts from any linear programming tableau. This has the advantage of ``solving'' any problem but has the disadvantage that the method can be very slow.

If we have a constraint $x_k + \sum_i a_i x_i = b$

with b not an integer, we can write each, $a_i = [a_i] + a'_i$ for some $0 \le a'_i \le 1$ and $b_i = [b_i] + b'_i$ for some $0 \le b'_i \le 1$

Using the same steps we get: $x_k + \sum_i [a_i] x_i - [b_i] = b' - \sum_i a' x_i$ to get the cut

$$b' - \sum_i a' x_i \le 0$$

This cut can then be added to the linear program and the problem resolved. The problem is guaranteed not to get the same solution. This method can be shown to guarantee finding the optimal integer solution.

Cutting Plane Techniques

Cuts for Special Structure
This second approach is to use the structure of the problem to generate very good cuts. The approach needs a problem-by-problem analysis, but can provide very efficient solution techniques.
.

Warehouse location problem

- **n** warehouses
- cost $\mathbf{f_j}$ of opening warehouse j
- m customers
- customer i has a "demand" of $\mathbf{d_i}$
- unit shipping cost c_{ij} of serving customer i via warehouse j.

Variables:

let $y_j = 1$ if warehouse j is opened

Let x_{ij} = amount of demand for customer i satisfied at warehouse j.

yj = 1 for j in S,
yj = 0 for j not in S.
Suppose you knew which <u>warehouses</u> were <u>open</u>. S = set of open warehouses

$$\textbf{Minimize} \quad \sum_{i,j} c_{ij}x_{ij} + \sum_{j\in S} f_j$$

<u>**Subject to:**</u>

customers get their demand satisfied $\quad \mathbf{x_{ij} \le d_i \quad \text{if } y_j = 1}$

no shipments are made from an empty warehouse $\quad \mathbf{x_{ij} = 0 \quad \text{if } y_j = 0}$

and $\mathbf{x \ge 0}$

Warehouse location problem

y_j = 1 if warehouse i is opened   M**inimize** $\sum_{i,j} c_{ij}x_{ij} + \sum_j f_i y_i$

y_j = 0 otherwise

x_{ij} = flow from i to j

subject to:

customers get their demand satisfied $\sum_i x_{ij} = d_j$

each warehouse is either opened or it is not (no partial openings) $0 \leq y_j \leq 1$

no shipments are made from an empty warehouse

(We do not allow shipping from warehouse j if it is not opened) $x_{ij} \leq d_j y_i$ **for all i,j**