Canonical Form

$$\begin{aligned} & \max_{\mathbf{x} \in \mathbb{R}^d} \mathbf{c}^{\mathbf{T}} \mathbf{x} \\ \text{subject to:} \quad & \mathbf{A}\mathbf{x} \leq \mathbf{b} \\ \text{where} \quad & \mathbf{x} \in \mathbb{R}^d,\ \mathbf{A} \in \mathbb{R}^{m \times d},\ \mathbf{b} \in \mathbb{R}^m,\ \mathbf{c} \in \mathbb{R}^d \end{aligned}$$

Figure:

The polyhedron P, our feasible region, is defined by the set of inequalities Ax b. In two dimensions, i.e. if d = 2, this problem can be represented and solved graphically. The idea is to represent the cost vector c as a level set in the plane. Then, moving this level set towards the highest possible value so that still intersects our feasible region, i.e. the polyhedron P defined by A, yields the optimal solution.

Divisibility

- Divisibility is one of the conventional LP assumptions.
- Divisibility allowed us to consider activities in fractions: We could produce 7.8 units of a product, buy 12500.33 liters of oil, hire 12.123 people for full time, etc.
- Divisibility assumption is very defensible at times but not always.
- We can easily buy 12500.33 liters of oil but can not employ 12.123 people.

Divisibility

- Clearly some activities cannot be done in fractions and must be specified in integers for implementation.
- As soon as some of the activities are set to be integers, we are in **Integer Programming** domain.
- Formally, in an integer program some decision variables are forced to be integers.

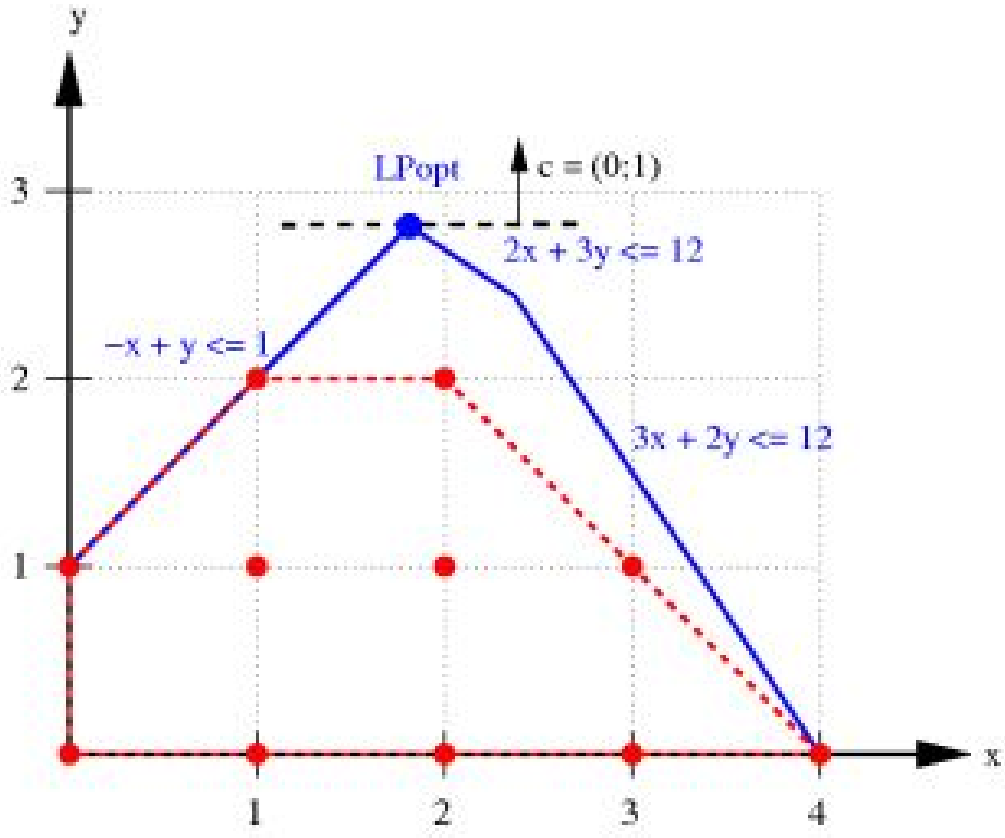

Figure:

IP Example: Tables and Chairs

- Suppose we consider producing chairs and tables using only 21 m^2 of wood. Each chair (table) requires 6 (7) m^2 of wood.
- Each chair is sold at \$12 (10) and each table is sold at \$13 ($\times$10).
- Let C and T denote the number of tables and chairs produced.
- The IP formulation below maximizes the revenue:

IP Example: Tables and Chairs

$$
\begin{array}{lrlrl}
\text{Maximize :} & 12C & +13T & & \\
\text{Subject to} & & & & \\
& 6C & +7T \leq & 21 & (1) \\
& C & , T \geq & 0 & (2) \\
& C & , T \; int & & (3)
\end{array}
$$

IP Example: Tables and Chairs

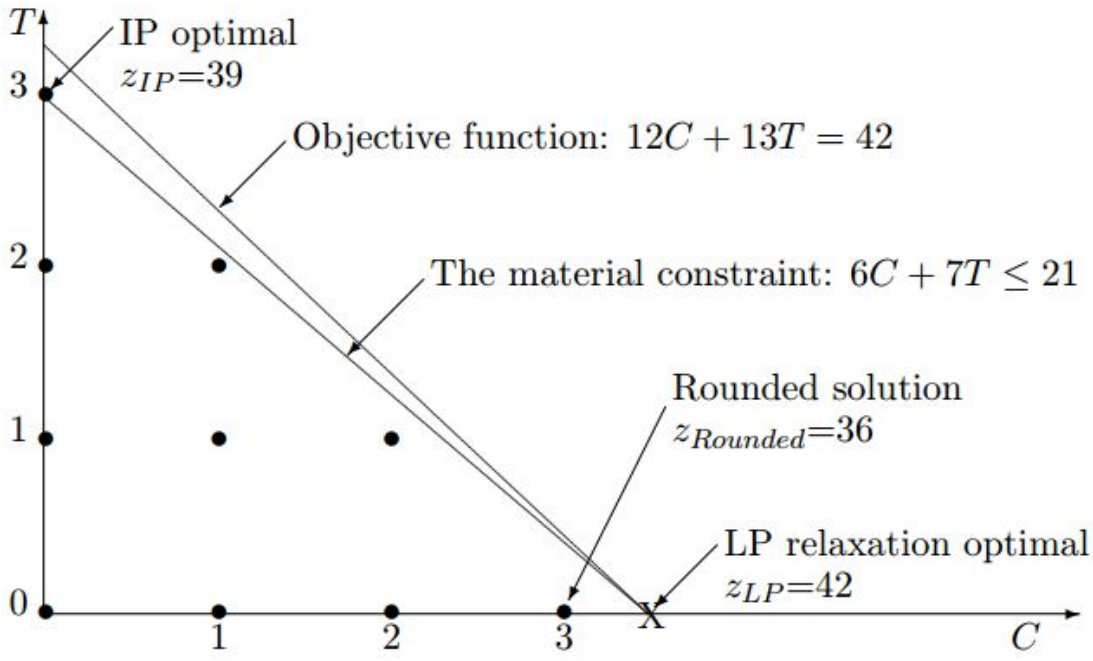

Figure:

LP Relaxation

- Solving an IP can be as straightforward as solving the associated LP and rounding the solution, but only in some cases. (i.e. by coincidence rather than by definition)
- To understand what can wrong with this approach, we will first solve the IP removing constraint (3) and round down (why not to round up?) the optimal values of C and T to satisfy (3).
- When the integer constraints are removed from an IP formulation, we obtain an LP formulation. This LP formulation is called the **LP relaxation**.

LP Relaxation

- LP solution is (7/2,0) and is not integer so we round it down to (3,0).
- The objective value at (3,0) is 36.
- The optimal solution to IP is (0,3) with the objective value 39.
- 3 units of difference between objective value of the IP optimal and the rounded solution can be significantly higher in more complex problems.
- As a summary we cannot use rounded solutions of LP relaxations.

LP Relaxation

Given an integer program (IP), there is an associated Hnear program (LR) called the linear relaxation. It is formed by dropping (relaxing) the integrality restrictions. Since (LR) is less constrained than (IP), the following are immediate:

1. If (IP) is a minimization problem, the optimal objective value of (LR) is less than or equal to the optimal objective value of (IP).
2. If (IP) is a maximization problem, the optimal objective value of (LR) is greater than or equal to the optimal objective value of (IP),
3. If (LR) is infeasible, then so is (IP).
4. If all the variables in an optimal solution of (LR) are integer-valued, then that solution is optimal for (IP) too.

5 If the objective function coefficients are integer-valued, then for minimization problems, the optimal objective value of (IP) is greater than or equal to the ceiling of the optimal objective value of (LR). For maximization problems, the optimal objective value of (IP) is less than or equal to the floor of the optimal objective value of (LR).

- For simple problems one can evaluate all the integer solutions in the feasible region and pick the best.
- However, for real problems this approach will take practically infinite amount of time.
- The solution procedures for IPs are still under development.
- Two approaches are common: Branch and Bound technique, and Cutting planes.
- These techniques are outside the scope of our discussion.

Zero-One Integer Problems

Zero-one linear programming involves problems in which the variables are restricted to be either 0 or 1. Note that any bounded integer variable can be expressed as a combination of binary variables.

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 $i \in 1 \ldots m$.
- The capital available in time period is b_i .

Capital Budgeting

- The problem of maximising revenue subject to i 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

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

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

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

Divide and Conquer

1. Partition the problem into subproblems.

2. Solve the subproblems.

3. Combine the solutions to solve the original one.

Figure:

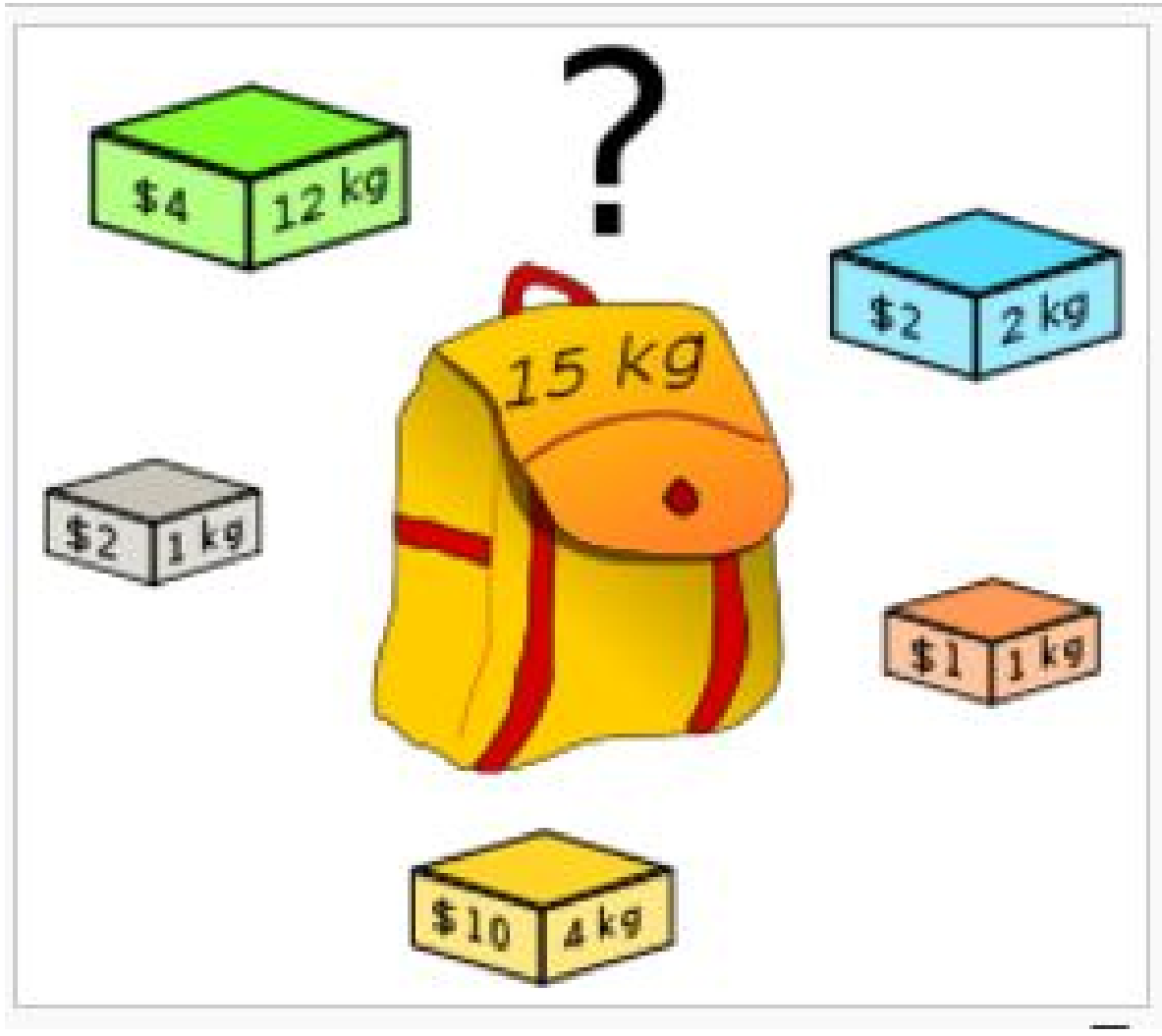

Figure:

The Knapsack Problem

- The knapsack problem or rucksack problem is a problem in combinatorial optimization
- Given a set of items, each with a weight and a value, determine the number of each item to include in a collection so that the total weight is less than or equal to a given limit and the total value is as large as possible.

Knapsack Problems

- Remark: The 0-1 knapsack problems is an optimization problem.
- Brute force: Try all 2^n possible subsets (Recall definition of Power Set) .
- Question: Any solution better than the brute-force?

Knapsack Problems

- The most common problem being solved is the 0-1 knapsack problem, which restricts the number xi of copies of each kind of item to zero or one.
- Given a set of n items numbered from 1 up to n, each with a weight wi and a value vi, along with a maximum weight capacity W,

$$\text{maximize} \sum_{i=1}^{n} v_i x_i$$

$$\text{subject to} \sum_{i=1}^{n} w_i x_i \leq W \text{ and } x_i \in \{0,1\}.$$

Figure:

Knapsack Problems

Here xi represents the number of instances of item i to include in the knapsack. Informally, the problem is to maximize the sum of the values of the items in the knapsack so that the sum of the weights is less than or equal to the knapsack's capacity.

Knapsack Problems

The bounded knapsack problem (BKP) removes the restriction that there is only one of each item, but restricts the number x_i of copies of each kind of item to an integer value c_i:

$$\text{maximize} \sum_{i=1}^{n} v_i x_i$$

$$\text{subject to} \sum_{i=1}^{n} w_i x_i \leq W \text{ and } x_i \in \{0, \ldots, c_i\}$$

Figure:

Knapsack Problems

The unbounded knapsack problem (UKP) places no upper bound on the number of copies of each kind of item and can be formulated as above except for that the only restriction on x_i is that it is a non-negative integer.

$$\text{maximize} \sum_{i=1}^{n} v_i x_i$$

$$\text{subject to} \sum_{i=1}^{n} w_i x_i \leq W \text{ and } x_i \geq 0$$

Figure:

Merrill Lynch is considering investments into 6 projects: A, B, C, D, E and F. Each project has an initial cost, an expected profit rate (one year from now) expressed as a percentage of the initial cost, and an associated risk of failure. These numbers are given in the table below:

	A	B	C	D	E	F
Initial cost (in M)	1.3	0.8	0.6	1.8	1.2	2.4
Profit rate	10%	20%	20%	10%	10%	10%
Failure risk	6%	4%	6%	5%	5%	4%

Figure:

a) Provide a formulation to choose the projects that maximize total expected profit, such that Merrill Lynch does not invest more than 4M dollars and its average failure risk is not over 5%.

   For example, if Merrill Lynch invests only into A and B, it invests only 2.1M dollars and its average failure risk is (6%+4%)/2=5%.

b) Suppose that if A is chosen, B must be chosen. Modify your formulation.

c) Suppose that if C and D are chosen, E must be chosen. Modify your formulation.

Branch and Bound

The branch and bound method is a solution approach that partitions the feasible solution space into smaller subsets of solutions

Depot Location Problem

Cutting Plane Algorithm

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 .

Fathoming

1. Pruning by Optimality
2. Pruning by Bound
3. Pruning by Infeasibility

Branch-and-bound

- Branch-and-bound is essentially a strategy of divide and conquer. The idea is to partition the feasible region into more manageable subdivisions and then, if required, to further partition the subdivisions.
- In general, there are a number of ways to divide the feasible region, and as a consequence there are a number of branch-and-bound algorithms.
- We shall consider one such technique, for problems with only binary variables, in Section 9.7.
- For historical reasons, the technique that will be described next usually is referred to as the branch-and-bound procedure.

Branch and Bound

Consider pure IP

```
Maximize
obj: x1 + x2
Subject to
c1: -x1 + x2 <= 2
c2: 8 x1 + 2 x2 <= 19
Bounds
x1 x2 >= 0
Integer
x1 x2
End
```

Branch and Bound

LP-relaxation

```
Maximize
obj: x1 + x2
Subject to
c1: -x1 + x2 <= 2
c2: 8 x1 + 2 x2 <= 19
Bounds
x1 x2 >= 0
End
```

Solution of LP-relaxation

- $x_1 = 1.5, x_2 = 3.5$
- Value: $x = 5$

Branch and Bound

LP-relaxation

```
Maximize
obj: x1 + x2
Subject to
c1: -x1 + x2 <= 2
c2: 8 x1 + 2 x2 <= 19
Bounds
x1 x2 >= 0
End
```

Solution of LP-relaxation

- $x_1 = 1.5, x_2 = 3.5$
- Value: $z = 5$

Branch and Bound

Create two sub-problems:

Left sub-problem

```
Maximize
obj: x1 + x2
Subject to
c1: -x1 + x2 <= 2
c2: 8 x1 + 2 x2 <= 19
Bounds
x1 x2 >= 0
x1 <= 1
End
```

Solution of left subproblem

- $x_1 = 1$, $x_2 = 3$ (integral feasible)
- Value: $z = 4$

Branch and Bound

Right sub-problem

```
Maximize
obj: x1 + x2
Subject to
c1: -x1 + x2 <= 2
c2: 8 x1 + 2 x2 <= 19
Bounds
x1 x2 >= 0
x1 >= 2
End
```

Solution of right subproblem

- $x_1 = 2$, $x_2 = 1.5$ (integral infeasible)
- Value: $z = 3.5$

Branch and Bound

- Each integer feasible solution of right sub-problem has value bounded by 3.5.
- Since value of integer feasible solution $x_1 = 1$, $x_2 = 3$ is 4, we can prune the right sub-problem
- Since integer feasible solution $x_1 = 1$, $x_2 = 3$ is also optimal solution of left sub-problem, each integer feasible solution of left-subproblem has value at most 4.
- Thus $x_1 = 1$ and $x_2 = 3$ is optimum solution to integer program.

Enumeration Tree

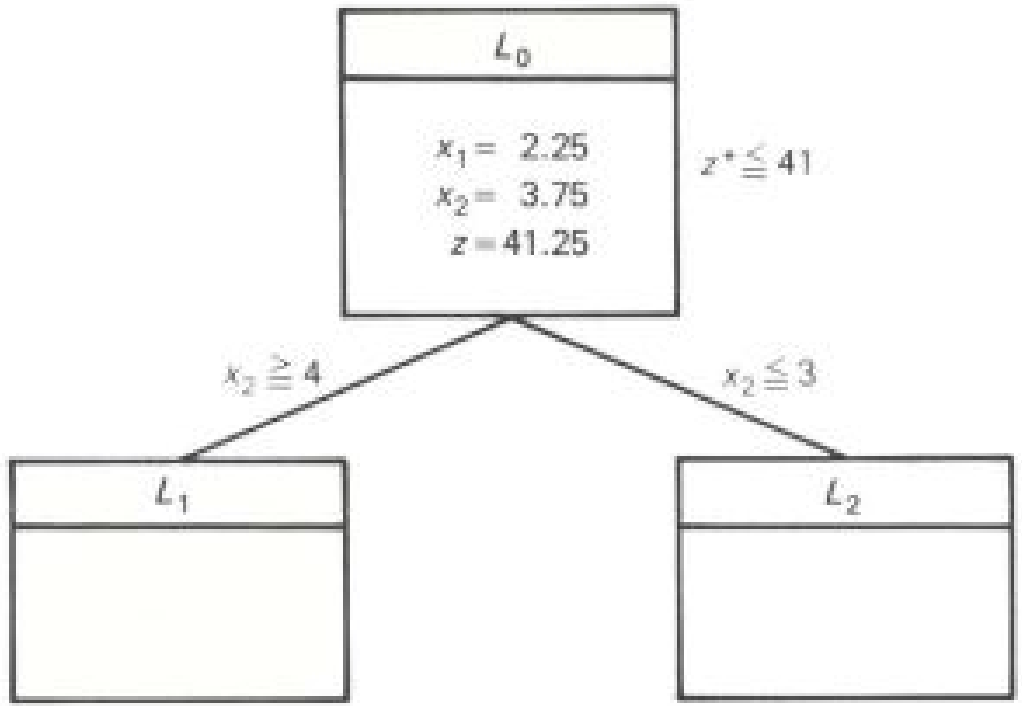

Figure:

Enumeration Tree

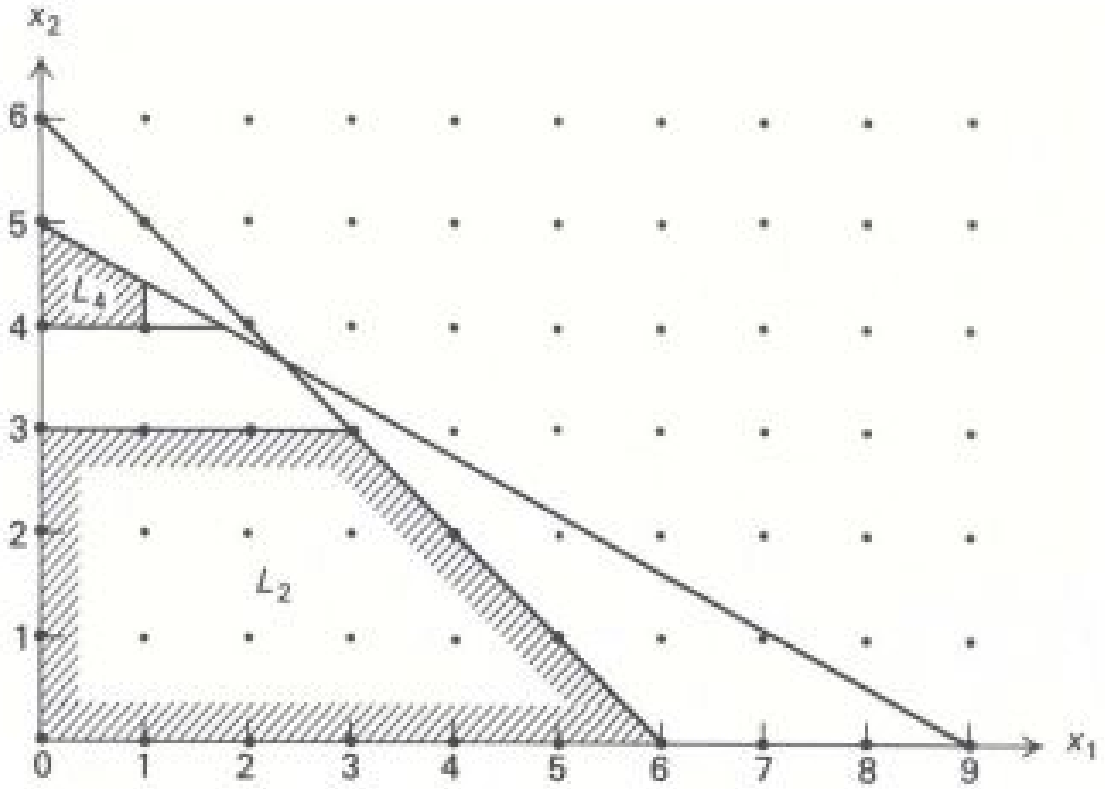

Figure:

It is not always the case that branch and bound quickly solves integer programs. In particular, it is possible that the bounding aspects of branch and bound are not invoked, and the branch and bound algorithm can then generate a huge number of subproblems. In the worst case, a problem with n binary variables (variables that have to take on the value 0 or 1) can have 2^n subproblems. This exponential growth is inherent in any algorithm for integer programming,