Lecture Notes – Part 10

Integer Programming (I)

1 Introduction

An *integer program* (IP) is a mathematical programming problem in which some or all of the variables are required to be integers. In most settings (including that in this subject), this term refers to integer linear program, in which the objective function and the constraints (other than the integer constraints) are linear. An IP is called

- a ***pure IP*** if all variables are required to be integers,
- a ***0-1 IP*** or ***binary IP*** if all variables must be 0 or 1, and
- a ***mixed IP*** if only some of the variables are required to be integers.

A *first approximation* to the solution of an IP may be obtained by ignoring the integer requirement and solving the resultant LP. This LP is called the *LP relaxation* of the considered IP. The LP relaxation can be thought of as being less constrained or more relaxed than the IP for the feasible region of the IP lies within that of the LP relaxation. Thus, the optimal objective value of the LP relaxation is no worse, i.e. no less (for max. problems) or no greater (for min. problems) than the corresponding IP.

If the first approximation happens to be an integer solution or partial integer solution as required, then this solution is also an optimal solution to the original IP. Otherwise, we round the components which are not integers but are required to be integers of the first approximation to the nearest feasible integers and obtain a *second approximation*. The procedure will be repeated until the solution to the original IP is yielded.

We introduce two techniques for IPs retaining similar notions – the *branch-and-bound* algorithm and the *cutting-plane* algorithm.

2 Branch-and-Bound Algorithm

The branch-and-bound (B&B) method searches an optimal solution of an IP by efficiently enumerating the extreme points in the feasible regions of subproblems.

2.1 Branching

First we solve the LP relaxation of the considered IP. Assume that the first approximation, $\mathbf{x}^* = (x_1^*, x_2^*, \ldots, x_n^*)^T$, contains a variable whose value is not an integer as required, say x_j^*. Since the band $\left\{\mathbf{x} : \lfloor x_j^* \rfloor < x_j < \lfloor x_j^* \rfloor + 1\right\}$ cannot contain a feasible integer solution, a feasible integer value of x_j must satisfy one of the two conditions $x_j \leq \lfloor x_j^* \rfloor$ or $x_j \geq \lfloor x_j^* \rfloor + 1$. Applying these two conditions respectively to the original LP relaxation yields two new LPs (one LP for each additional constraint). This process, called *branching*, has the effect of shrinking the feasible region.

After the first branching, we arbitrarily choose one of these two

LP subproblems, which have the same objective function as the original IP, to solve. If its optimum is feasible with respect to the considered IP, i.e. an integer solution or partial integer solution as required, then it is recorded as the best one so far available, which is called the *incumbent solution*.[1] If this is not the case, we may need to do the second branching. The subproblem for further branching must be partitioned into two new subproblems by again imposing the integer conditions on one of its variable(s) that currently have fractional values.

After each branching, an 'integer solution' obtained by solving a subproblem is called a *candidate solution*. If a candidate solution has a better objective value than that of the incumbent solution, the incumbent solution has to be updated and replaced with it. Once a candidate solution is obtained, we don't need to perform the further branching. If neither the obtained solution is an integer solution nor the further branching could yield a candidate solution better than the incumbent one[2], it is unnecessary to branch on a subproblem either. This termination is called *fathomed*. The process of branching, where applicable, continues until each subproblem either terminates with a candidate solution or is fathomed. In this case the incumbent solution in hand, if any, is the optimal solution of the considered IP problem.

[1]The objective value of this optimum won't be better than the solution obtained from the LP relaxation.

[2]The step to recognise this situation is called "bounding", which will be introduced later.

2.2 Bounding

The efficiency of the computations in B&B can be enhanced by introducing the concept of *bounding*. Once a candidate solution is found, its associated objective value can be used as a bound to discard inferior subproblems (an upper bound for min. problems and a lower bound for max. problems).

The efficiency of calculations tends to depend directly on the order in which the different subproblems are generated and scanned. The specific problems generated depend on the variable selected for branching. Unfortunately, there is no definite best way for selecting the branching variable or the specific sequence in which subproblems must be scanned. Below are two general approaches which are commonly used to determine which subproblems should be solved next.

- **The LIFO rule**: The most widely used is the *Last-In-First-Out* (LIFO) rule, which chooses to solve the most recently created subproblem. LIFO leads us down one side of the branching tree and quickly finds a candidate solution. Then we backtrack to the top of the other side of the tree. For this reason, the LIFO approach is often called *backtracking*.
- **The Jumptracking rule**: The *jumptracking* approach solves all the problems created by a branching and then branches on the node with the best z-value. Jumptracking often jumps from one side of the tree to the other. The idea behind jumptracking is that moving toward the subproblems with

good z-values should lead us more quickly to the optimal solution. It usually creates more subproblems and requires more computer storage than backtracking, though.

In this chapter, LIFO rule is assumed to be adopted if it is not stated otherwise.

Example 1. Solve the following pure IP using the B&B algorithm, and illustrate the process with a branching tree diagram.

$$\begin{aligned} \max\ z = \quad & 8x_1 + 5x_2 \\ s.t. \quad & x_1 + x_2 \leq 6 \\ & 9x_1 + 5x_2 \leq 45 \\ & x_1,\, x_2 \geq 0, \text{ integer} \end{aligned}$$

Solution:

- **Subproblem 1**. The subproblem 1 is always the LP relaxation, where the integer restriction is lifted. Adding the slack variables x_3 and x_4, and then choosing them as the initial basis, the Simplex method gives

basis	x_1	x_2	x_3	x_4	rhs	
z	-8	-5	0	0	0	
x_3	1	1	1	0	6	
x_4	$\boxed{9}$	5	0	1	45	
z	0	$-\frac{5}{9}$	0	$\frac{8}{9}$	40	$R_0 \leftarrow R_0 + 8(\text{new})R_2$
x_3	0	$\boxed{\frac{4}{9}}$	1	$-\frac{1}{9}$	1	$R_1 \leftarrow R_1 + \frac{5}{9}(\text{new})R_1$
x_1	1	$\frac{5}{9}$	0	$\frac{1}{9}$	5	$R_2 \leftarrow \frac{1}{9}R_2$ (go first)
z	0	0	$\frac{5}{4}$	$\frac{3}{4}$	$\frac{165}{4}$	$R_0 \leftarrow R_0 + \frac{5}{9}(\text{new})R_1$
x_2	0	1	$\frac{9}{4}$	$-\frac{1}{4}$	$\frac{9}{4}$	$R_1 \leftarrow \frac{9}{4}R_1$ (go first)
x_1	1	0	$-\frac{5}{4}$	$\frac{1}{4}$	$\frac{15}{4}$	$R_2 \leftarrow R_2 - \frac{5}{9}(\text{new})R_1$

The solution of subproblem 1 is $(x_1^*, x_2^*) = (\frac{15}{4}, \frac{9}{4})$ with $z_{\max} = \frac{165}{4}$. Since the values of both decision variables are not integers, we arbitrarily branch the subproblem 1 on x_1 into the following two additional subproblems:

Subproblem 2 = Subproblem 1 + Constraint $x_1 \geq 4$;

Subproblem 3 = Subproblem 1 + Constraint $x_1 \leq 3$.

- **Subproblem 2**. A new constraint $x_1 \geq 4$ is added into Subproblem 1. Introducing a new surplus variable x_5, we append the new constraint $x_1 - x_5 = 4$ to the final tableau of Subproblem 1. Applying the technique for adding a new constraint to an LP[3], we need to express this constraint in terms

[3]Please refer to "Sensitivity Analysis" in Lecture Note – Part 6.

of nonbasic variables. From the final tableau of Subproblem 1, we have

$$-4 = -x_1 + x_5 = -\Big(\frac{15}{4} + \frac{5}{4}x_3 - \frac{1}{4}x_4\Big) + x_5$$

$$\Rightarrow \qquad -\frac{5}{4}x_3 + \frac{1}{4}x_4 + x_5 = -\frac{1}{4}$$

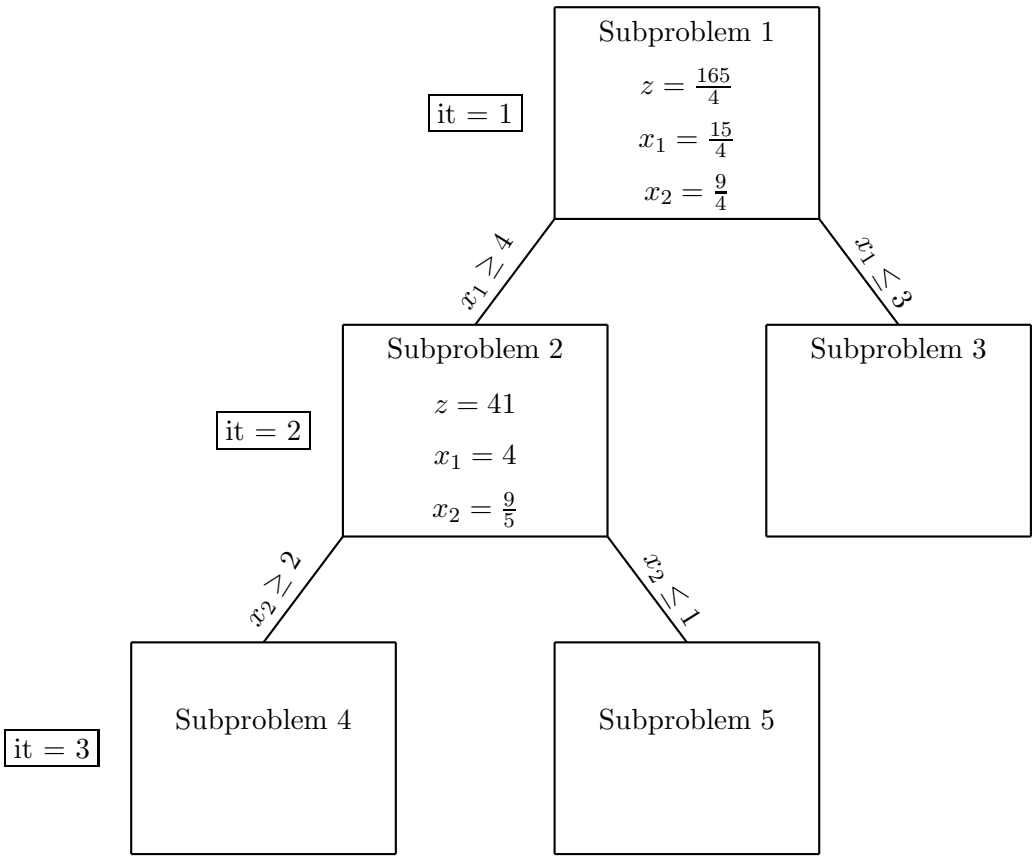
Now this equality is ready for being added to the final tableau of Subproblem 1 with an extra basic variable x_5 and one pivot column for it. Since the rhs of the new constraint is negative, the infeasibility is incurred. To continue solving the problem, the dual Simplex method shall be applied to fix the feasibility.

basis	x_1	x_2	x_3	x_4	x_5	rhs	
z	0	0	$\frac{5}{4}$	$\frac{3}{4}$	0	$\frac{165}{4}$	
x_2	0	1	$\frac{9}{4}$	-$\frac{1}{4}$	0	$\frac{9}{4}$	
x_1	1	0	-$\frac{5}{4}$	$\frac{1}{4}$	0	$\frac{15}{4}$	
x_5	0	0	$\boxed{-\frac{5}{4}}$	$\frac{1}{4}$	1	- $\frac{1}{4}$	
z	0	0	0	1	1	41	$R_0 \leftarrow R_0 + (\text{old})R_3$
x_2	0	1	0	$\frac{1}{5}$	$\frac{9}{5}$	$\frac{9}{5}$	$R_1 \leftarrow R_1 - \frac{9}{4}(\text{new})R_3$
x_1	1	0	0	0	-1	4	$R_2 \leftarrow R_2 - (\text{old})R_3$
x_3	0	0	1	$-\frac{1}{5}$	- $\frac{4}{5}$	$\frac{1}{5}$	$R_3 \leftarrow -\frac{5}{4}R_3$ (go first)

The solution of Subproblem 2 is $(x_1^*, x_2^*) = (4, \frac{9}{5})$ with $z_{\max} = 41$. Since x_2^* in not integer, Subproblem 2 is branched on the variable x_2 into the following two additional subproblems

- Subproblem 4 = Subproblem 2 + Constraint $x_2 \geq 2$,
- Subproblem 5 = Subproblem 2 + Constraint $x_2 \leq 1$

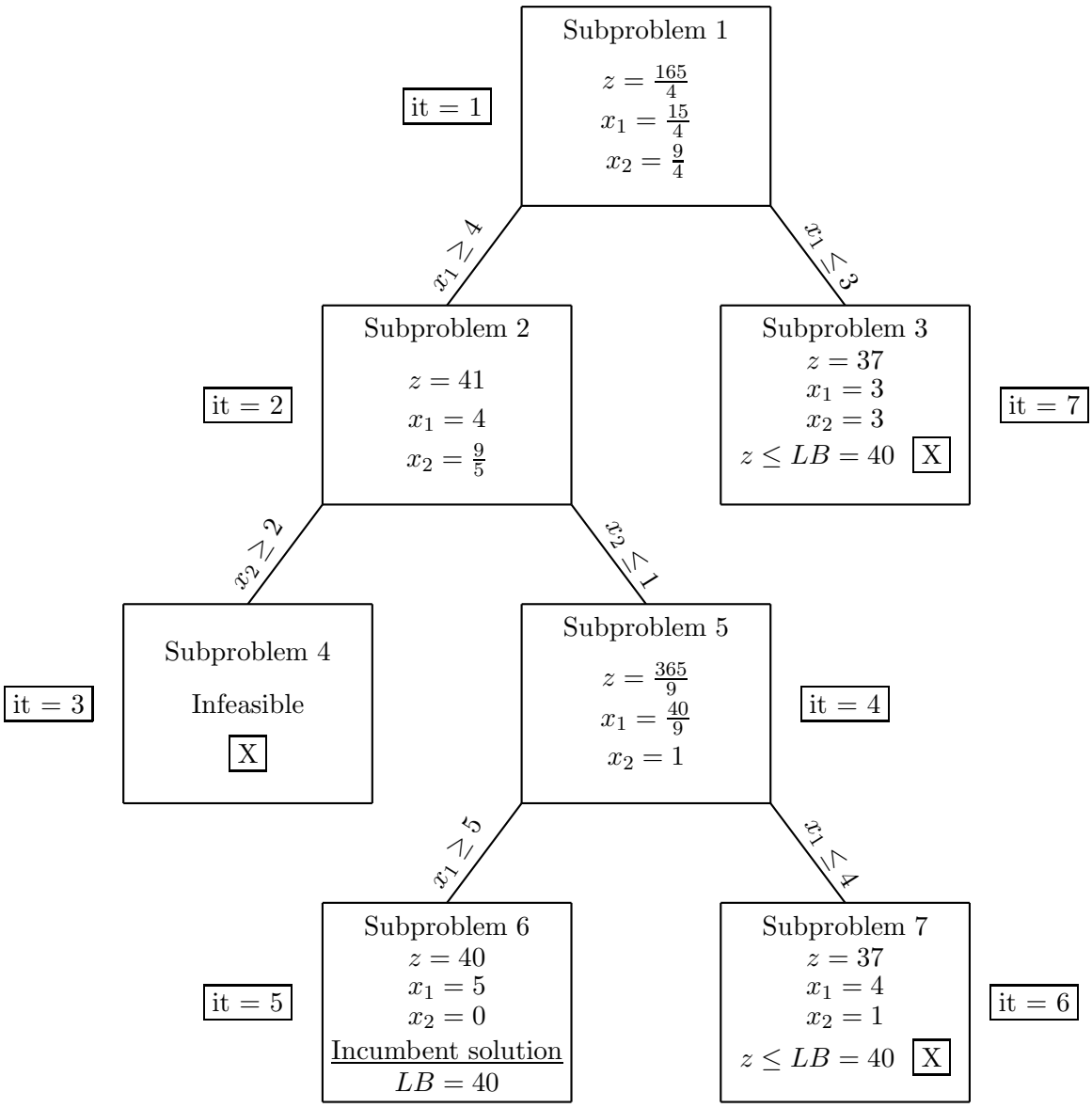
At this point, a branching tree diagram of the solution process is



The unsolved problems are Subproblems 3, 4 and 5. Following the LIFO rule, we should choose Subproblem 4 or Subproblem

5. We arbitrarily choose to solve Subproblem 4. Before trying to solving it with the same approach as previous Subproblem 2, we check its feasibility first. Since $x_1 \geq 4$ and $x_2 \geq 2$ gives $9x_1 + 5x_2 \geq 46$, which contradicts the original constraint $9x_1 + 5x_2 \leq 45$, this subproblem is infeasible and therefore cannot yield an optimal solution to the considered IP. A mark $\boxed{\text{X}}$ is to be placed on Subproblem 4 to indicate that it is fathomed. This branch now is eliminated.

The LIFO rule implies that Subproblem 5 should be solved next. Continuing the procedure, we obtain the following branching tree diagram.



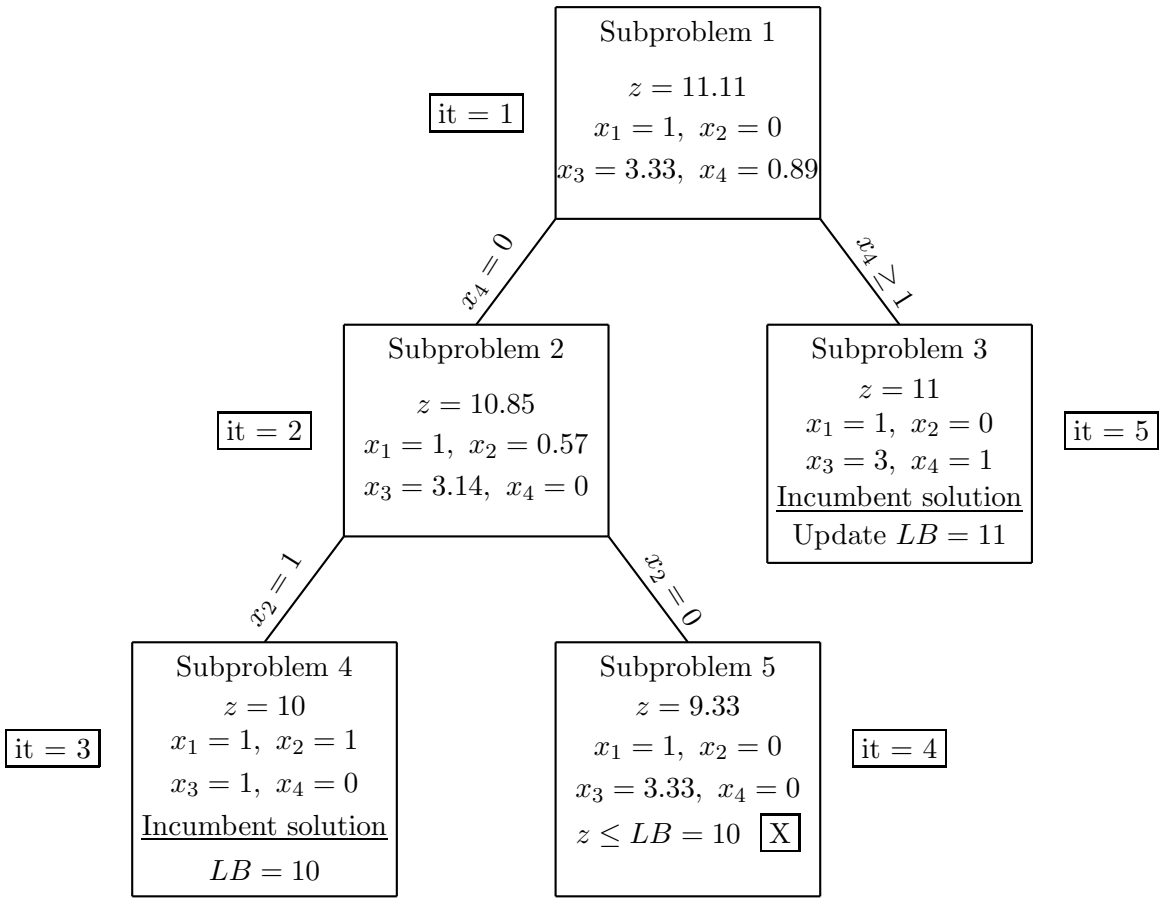
So the optimal solution of the given IP is $(x_1, x_2) = (5, 0)$ with $z_{\max} = 40$.

Example 2. Solve the following IP using the B&B algorithm,

and illustrate the process with a branching tree diagram.

$$
\begin{array}{rrrrrrrrrrl}
\max\ z = & 6x_1 & + & 3x_2 & + & x_3 & + & 2x_4 & & \\
s.t. & x_1 & + & x_2 & + & x_3 & + & x_4 & \leq & 8 \\
& 2x_1 & + & x_2 & + & 3x_3 & & & \leq & 12 \\
& & & 5x_2 & + & x_3 & + & 3x_4 & \leq & 6 \\
& x_1 & & & & & & & \leq & 1 \\
& & & x_2 & & & & & \leq & 1 \\
& & & & & x_3 & & & \leq & 4 \\
& & & & & & & x_4 & \leq & 2 \\
& & & & & & & x_1,x_2,x_3,x_4 & \geq & 0, \text{ integer}
\end{array}
$$

Solution Using the B&B algorithm, we obtain the following branching tree diagram.



So the optimal solution of the given IP is $(x_1, x_2, x_3, x_4) = (1, 0, 3, 1)$ with $z_{\max} = 11$.

Note that the constraints $x_1 \leq 1$ and $x_2 \leq 1$ coupled with nonnegativity requirements mean that x_1 and x_2 are 0-1 variables, i.e. binary variables. Variable x_3 can take five possible values, and variable x_4 can have three possible values. Hence, there are $2 \times 2 \times 5 \times 3 = 60$ possible solutions. A computer can quickly evaluate all 60 possibilities. However, as the problems grow in size, a complete enumeration becomes impractical.

3 Combinatorial Optimisation Problems

This section will introduce the knapsack problem, assignment problem and travelling salesman problem. These three problems are representatives of a larger class of problems known as combinatorial optimisation problems, which have wide applications.

3.1 Knapsack Problem

The 0-1 knapsack problem is a particularly simple example of binary IP having only one constraint. Furthermore, the coefficients of this constraint and the objective function are all nonnegative.

Example 3. There is a knapsack with a capacity of 14 units. There are 4 items, each of which retains a size and a value, e.g. item 1 has a size of 5 units and a value of 16 dollars, item 2 has a size of 7 units and a value of 22 dollars, item 3 has a size of 4 units and a value of 12 dollars, and item 4 has a size of 3 units and a value of 8 dollars. The objective is to decide which items shall be packed in the knapsack so as to maximise the total value of items in the knapsack. Then the knapsack problem can be formulated as follows.

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

To solve the associated LP relaxation, it is simply a matter of determining which variable gives the most "bang for the buck." If you take the ratio (the objective coefficient/constraint coefficient)

for each variable, the one with the highest ratio is the best item to place in the knapsack. Then the item with the second highest ratio is put in and so on until we reach an item that cannot fit. At this point, a fractional amount of that item is placed in the knapsack to completely fill it. In our example, the variables are already ordered by the ratio. We would first place item 1 in and then item 2, but item 3 doesn't fit: there are only 2 units left, but item 3 has a size of 4. Therefore, we take half of item 3. The solution $x_1 = 1$, $x_2 = 1$, $x_3 = 0.5$, $x_4 = 0$ is the optimal solution to the LP relaxation.

As we will see shortly, this solution is quite different from the optimal solution to the original knapsack problem. Nevertheless, it will play an important role in the solving with B&B.

Solution. We use the LIFO rule to determine which subproblem should be solved first.

Subproblem 1.

As mentioned above, the solution of Subproblem 1 is $x_1 = 1$, $x_2 = 1$, $x_3 = 0.5$, $x_4 = 0$ with $z_{\max} = 44$. Then the problem is branched into two subproblems by setting $x_3 = 1$ (Subproblem 2), and $x_3 = 0$ (Subproblem 3).

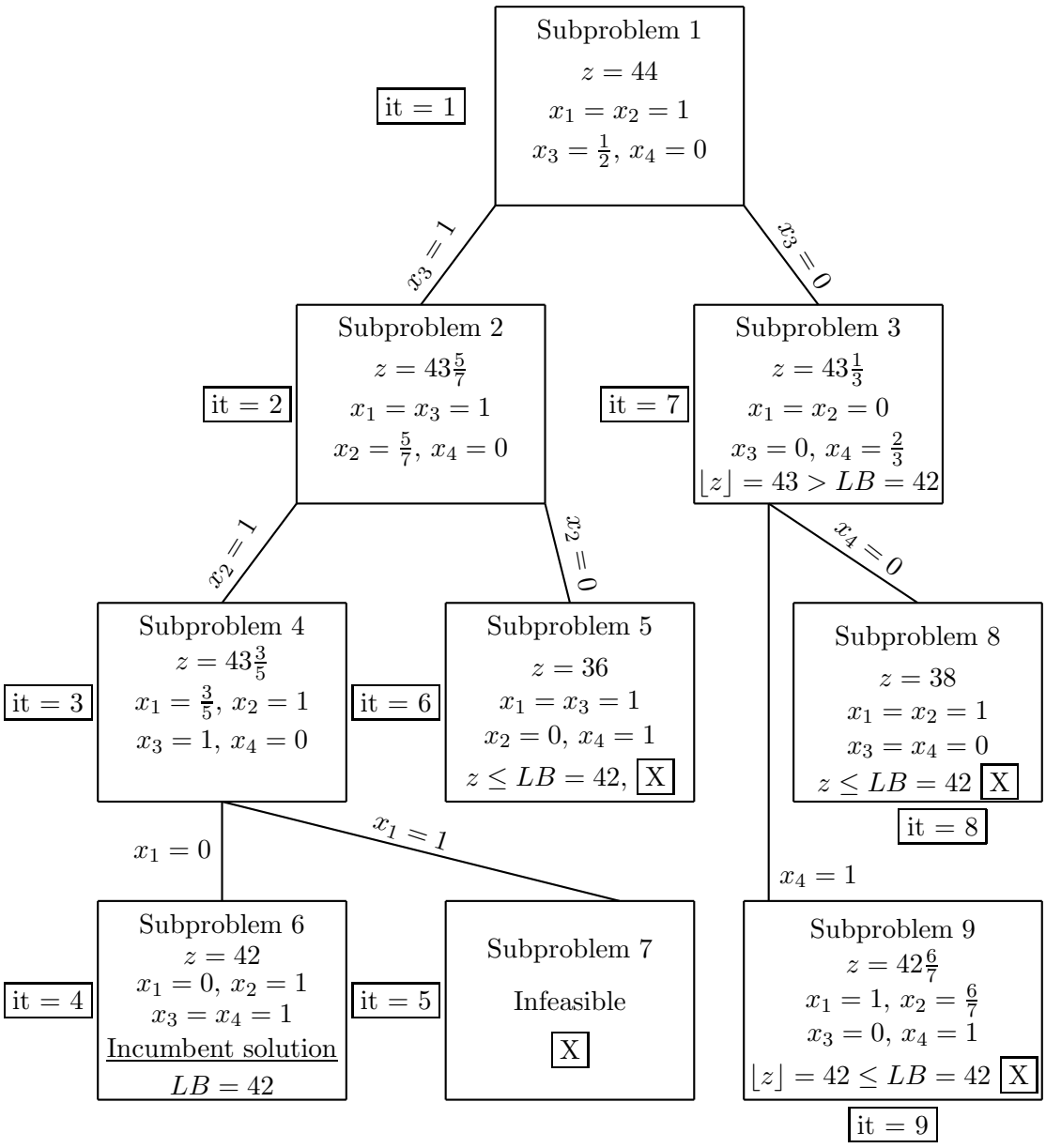
Subproblem 2.

We arbitrarily choose to solve Subproblem 2 before subproblem 3. To solve problem 2 we set $x_3 = 1$ and then solve the resulting knapsack problem.

$$\begin{array}{lrll} \max\ z = & 16x_1 + 22x_2 + 8x_4 + 12 & & \\ s.t. & 5x_1 + 7x_2 + 3x_4 & \leq & 14 - 4 = 10 \\ & x_1, x_2, x_4 & = & 0 \text{ or } 1. \end{array}$$

Applying the technique used to solve the LP relaxation of a knapsack problem gives the optimal solution $x_3 = 1$, $x_1 = 1$, $x_2 = \frac{5}{7}$ with $z_{\max} = \frac{306}{7}$.

Other subproblems can be solved with the same way. Below is the branching tree diagram of the problem.



The optimal solution is $x_1 = 0$, $x_2 = x_3 = x_4 = 1$ with $z_{\max} = 42$. Comparing with the LP relaxation, the highest priority item, i.e. variable x_1, is surprisingly not used.

3.2 Assignment Problem

The assignment problem is a combinatorial optimisation problem of assigning a set of m workers to a set of m jobs with a one-to-one matching of individuals to tasks for minimising the sum of the corresponding assignment costs. Given a nonnegative $m \times m$ matrix $\mathbf{C} = (c_{ij})$, where c_{ij}, the element in the i^{th} row and j^{th} column, represents the cost of assigning the i^{th} worker to the j^{th} job, we aim to find an assignment that has the minimum of total cost. The matrix $\mathbf{C}$ is called the cost matrix.

Binary IP for Assignment Problem

For each $i = 1, \ldots, m$ and each $j = 1, \ldots, m$, let

$$x_{ij} = \begin{cases} 1, & \text{if worker } i \text{ is assigned to complete job } j, \\ 0, & \text{otherwise.} \end{cases}$$

We obtain the binary IP below

$$\begin{aligned} \min\ z = \quad & \textstyle\sum_{i=1}^{m} \sum_{j=1}^{m} c_{ij} x_{ij} & & \\ s.t. \quad & \textstyle\sum_{j=1}^{m} x_{ij} & = & \ 1, \text{ for each } i = 1, \ldots, m, \\ & \textstyle\sum_{i=1}^{m} x_{ij} & = & \ 1 \text{ for each } j = 1, \ldots, m, \\ & x_{ij} & = & \ 0 \text{ or } 1. \end{aligned}$$

The first m constraints ensure that each worker i is employed to process one and only one job. The last m constraints ensure that each job j is assigned to one and only one worker.

The Hungarian Method

The Hungarian method is a combinatorial optimisation algorithm which solves the assignment problem in polynomial time.

It was developed and published by Harold Kuhn in 1955, who gave the name "Hungarian method" because the algorithm was largely based on the earlier works of two Hungarian mathematicians: Denes Konig and Jeno Egervary. The method consists of three steps below:

1. Find the minimum element in each row of the cost matrix. Construct a new matrix by subtracting from each cost the minimum cost in its row. For this new matrix, find the minimum cost in each column. Construct another new matrix (called the *reduced cost matrix*) by subtracting from each cost the minimum cost in its column.

2. Draw the minimum number of lines (horizontal, vertical, or both) that are needed to cover all the zeros in the reduced cost matrix. If m lines are required, then an optimal solution is available among the covered zeros in the matrix. If fewer than m lines are needed, then go to Step 3.

3. Among all elements in the reduced cost matrix that are uncovered by the lines drawn in Step 2, find the smallest nonzero element, say c_0. Construct a new reduced cost matrix by

   - subtracting c_0 from each uncovered element of the reduced cost matrix;
   - adding c_0 to each element of the reduced cost matrix that is covered by two lines.

   Then go to Step 2.

Example 4.
A manufacturing factory has four machines which can process four sorts of tasks to be completed. The time required to set up each machine for completing each job is shown in the table below. Any machine which has been assigned to process a task cannot be reassigned to another task. The factory manager aims to minimise the total setup time needed to complete four jobs with these four machines. Please solve this assignment problem via Hungarian method.

	Time (Hours)			
Machine	Task 1	Task 2	Task 3	Task 4
1	14	5	8	7
2	2	12	6	5
3	7	8	3	9
4	2	4	6	10

Solution.
For each $i = 1, \ldots, 4$ and $j = 1, \ldots, 4$, let

$$x_{ij} = \begin{cases} 1, & \text{if machine } i \text{ is assigned to task } j; \\ 0, & \text{otherwise.} \end{cases}$$

The problem can be formulated by the following binary IP.

$$\text{minimise} \quad \sum_{i=1}^{4} \sum_{j=1}^{4} c_{ij} x_{ij}$$

$$\text{subject to} \quad \sum_{j=1}^{4} x_{ij} = 1, \quad \forall i = 1, \ldots, 4,$$

$$\sum_{i=1}^{4} x_{ij} = 1, \quad \forall j = 1, \ldots, 4,$$
$$x_{ij} \in \{0, 1\}, \quad \forall i, j = 1, \ldots, 4.$$

Denote the cost matrix by

$$\mathbf{c} = [c_{ij}] = \begin{pmatrix} 14 & 5 & 8 & 7 \\ 2 & 12 & 6 & 5 \\ 7 & 8 & 3 & 9 \\ 2 & 4 & 6 & 10 \end{pmatrix}$$

Then we solve this assignment problem by Hungarian method as shown below.

Step 1. We have the cost matrix with the row minimum as shown below.

				Row min.
14	5	8	7	5
2	12	6	5	2
7	8	3	9	3
2	4	6	10	2

For each row, we subtract the row minimum from each element in the row, obtaining a new matrix with the column minimum as shown below.

	9	0	3	2
	0	10	4	3
	4	5	0	6
	0	2	4	8
Column min.	0	0	0	2

Then we subtract 2 from each cost in column 4, obtaining the reduced cost matrix shown as follows.

9	0	3	0
0	10	4	1
4	5	0	4
0	2	4	6

Step 2. After drawing the lines through row 1, row 3, and column 1, we cover all the zeros in the reduced cost matrix with the minimum number of lines, i.e. 3, which is smaller than $m = 4$. Then we proceed to Step 3.

~~9~~	~~0~~	~~3~~	~~0~~
0	10	4	1
~~4~~	~~5~~	~~0~~	~~4~~
0	2	4	6

Step 3. The smallest uncovered element equals 1. Subtracting 1 from each uncovered element and adding 1 to each twice-covered element give the new reduced cost matrix as shown below.

10	0	3	0
0	9	3	0
5	5	0	4
0	1	3	5

As shown below, four lines are now required to cover all zeros. Thus, an optimal solution exists among the covered zeros in this matrix.

~~10~~	~~0~~	~~3~~	~~0~~
0	9	3	0
~~5~~	~~5~~	~~0~~	~~4~~
0	1	3	5

To find an optimal solution we observe that the only zero in row 3 is cell $(3, 3)$, so we must have $x_{3,3} = 1$. Similarly, the only zero in row 4 is cell $(4, 1)$, so we set $x_{4,1} = 1$. Then the zero(s) in either column 1 or column 3 shall not be considered anymore. Now the only available zero in row 2 is cell $(2, 4)$, so we choose $x_{2,4} = 1$, which excludes the zero(s) in column 4 from further consideration. Finally, the only available choice for row 1 is $x_{1,2} = 1$.

Thus, the optimal solution is $x_{1,2} = x_{2,4} = x_{3,3} = x_{4,1} = 1$ with the minimum total setup time of $5 + 5 + 3 + 2 = 15$ hours.

Further reading: Section 9.1–9.5 and 7.5 in the reference book "Operations Research: Applications and Algorithms" (Winston, 2004)