Solving Knapsack Problems by the Branch-and-Bound Method

- A knapsack problem is an IP with a single constraint, and each variable must equal 0 or 1.
- So, a knapsack problem may be written as

$$\max z = c_1x_1 + c_2x_2 + \cdots + c_nx_n$$
$$\text{s.t.} \quad a_1x_1 + a_2x_2 + a_nx_n \leq b$$
$$x_i \in \{0,1\} \quad (i = 1,2,...,n).$$

- Recall that c_i is the benefit obtained if item i is chosen, b is the amount of an available resource, and a_i is the amount of the available resource used by item i.
- When knapsack problems are solved by the branch-and-bound method, two aspects of the method greatly simplify.

  (1) Because x_i must equal 0 or 1, branching on x_i will yield an $x_i = 0$ and an $x_i = 1$ branch.

  (2) Also, the LP relaxation (and other subproblems) may be solved by inspection.

  Reason. a_i/c_i may be interpreted as the benefit item i earns for each unit of the resource used by item i. Thus, the best (worst) items have the largest (smallest) values of a_i/c_i. So, one can choose items with large c_i/a_i values.

For (0,1) problem, each time we only allow $x_i \in \{0,1\}$

Example Consider the problem:

$$\max z = 40x_1 + 80x_2 + 10x_3 + 10x_4 + 4x_5 + 20x_6 + 60x_7$$
$$\text{s.t. } 40x_1 + 50x_2 + 30x_3 + 10x_4 + 10x_5 + 40x_6 + 30x_7 \leq 100$$
$$x_i \in \{0,1\} \quad (i = 1,2,...,7).$$

We begin by computing the a_i/c_i ratios and ordering the variables from best to worst.

i	1	2	3	4	5	6	7
c_i/a_i	1	8/5	1/3	1	4/10	1/2	2

Then choose item 7, and $100 - 30 = 70$ units of the resource remain.
Then choose 2, and $70 - 50 = 20$ units of the resource remain.
Then choose item 4 or item 1; we choose item 4, and $20 - 10 = 10$ units of the resource remain.
The best remaining item is item 1. We fill the knapsack with as much of item 1 as we can.
Because only 10 units of the resource remain, we set $x1 = 1/4$ (for the LP relaxation problem).
Thus an optimal solution to the LP relaxation is $z = 80 + 60 + 10 + (1/4)40 = 160$ with
$x_7 = x_2 = x_4 = 1$ and $x_1 = 1/4$.

Example 1 Stockco capital budgeting problem

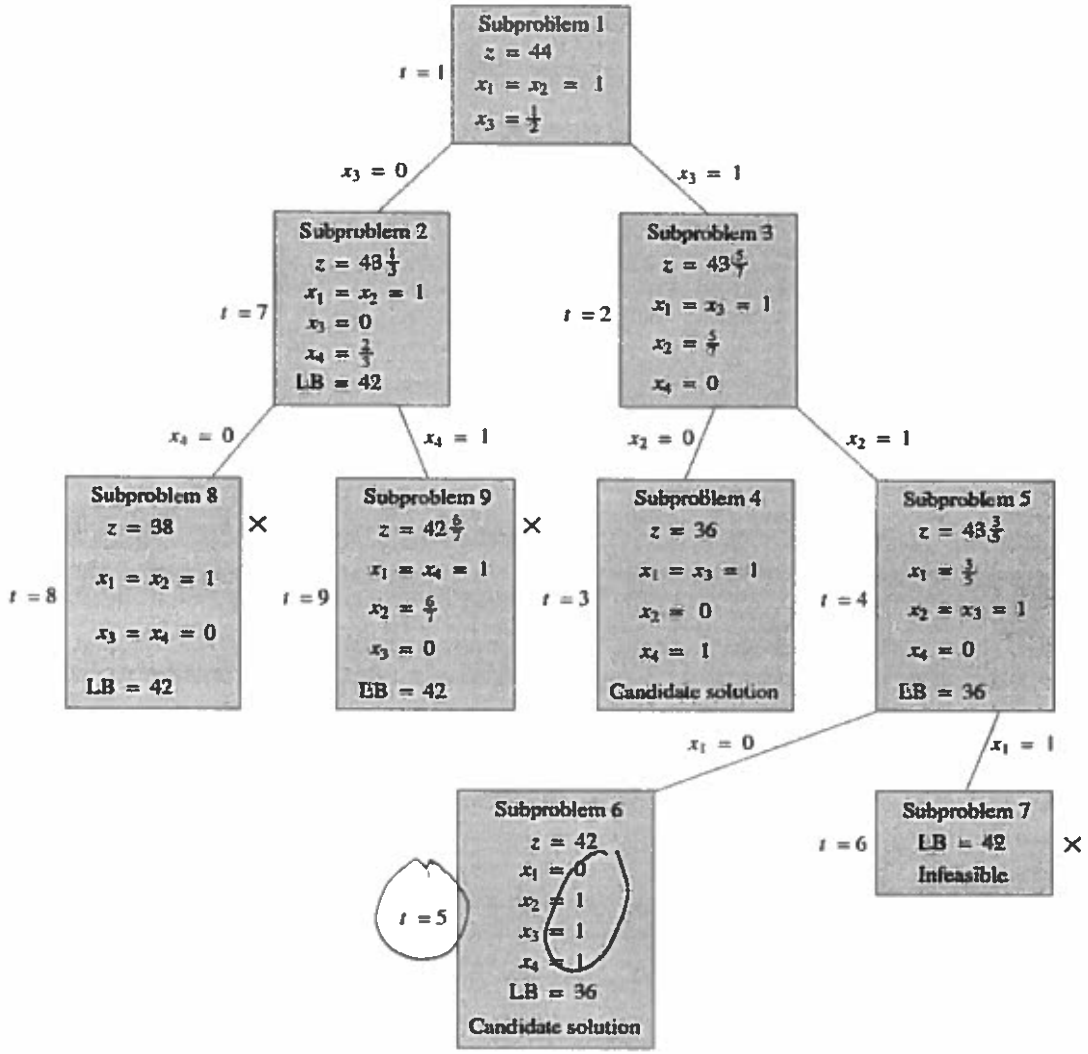
$$\max z = 16x_1 + 22x_2 + 12x_3 + 8x_4$$

$$\text{s.t.} \quad 5x_1 + 7x_2 + 4x_3 + 3x_4 \leq 14$$

$$x_1, \ldots, x_4 \in \{0, 1\}.$$

The branch-and-bound tree for this problem is shown in the following Figure.

The optimal solution is $z = 42$, with $x_1 = 0, x2 = x3 = x4 = 1$.



9.6 Solving Combinatorial Optimization Problems

Branch-and-Bound Machine Scheduling

- Four jobs must be processed on a single machine.
- The time required to process each job and the date the job is due are shown in Table 63.
- The delay of a job is the number of days after the due date that a job is completed.
- In what order should the jobs be processed to minimize the total delay of the four jobs?
- Suppose the jobs are processed in the following order: job 1, job 2, job 3, and job 4.

  By Table 64, the total delay is $0 + 6 + 3 + 7 = 16$ days.
- Consider the branch-and-bound approach.
- Let x_{ij} be 1 if job i is scheduled to take the jth slot to be processed.
- Partition all solutions according to the job that is last processed, i.e., $x_{14} = 1$, $x_{24} = 1$, $x_{34} = 1$, or $x_{41} = 1$.
- We create a node by branching, we obtain a lower bound on the total delay (D) associated with the node.
- So, if $x_{44} = 1$, then job 4 is the last job to be processed, job 4 will be completed at the end of day $6 + 4 + 5 + 8 = 23$ and will be $23 - 16 = 7$ days late.

TABLE 63
Durations and Due Date of Jobs

Job	Days Required to Complete Job	Due Date
1	6	End of day 8
2	4	End of day 4
3	5	End of day 12
4	8	End of day 16

TABLE 64
Delays Incurred If Jobs Are Processed in the Order 1-2-3-4

Job	Completion Time of Job	Delay of Job
1	6	0
2	$6 + 4 = 10$	$10 - 4 = 6$
3	$6 + 4 + 5 = 15$	$15 - 12 = 3$
4	$6 + 4 + 5 + 8 = 23$	$23 - 16 = 7$

TABLE 63
Durations and Due Date of Jobs

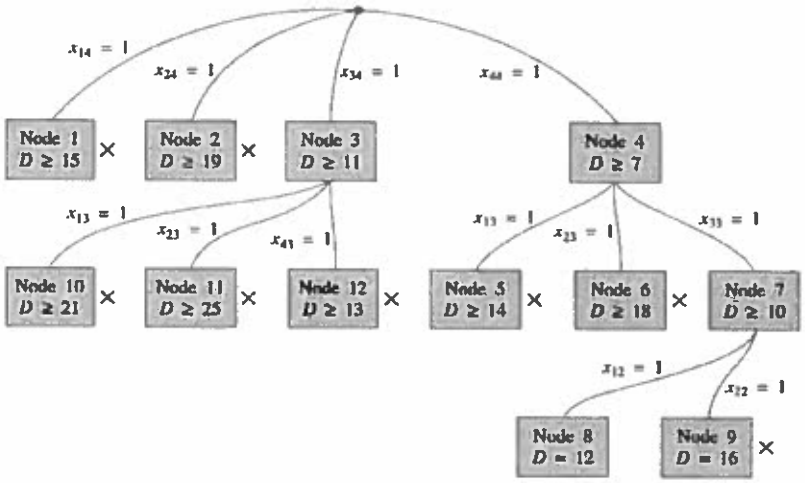
Job	Days Required to Complete Job	Due Date
1	6	End of day 8
2	4	End of day 4
3	5	End of day 12
4	8	End of day 16

TABLE 64
Delays Incurred If Jobs Are Processed in the Order 1-2-3-4

Job	Completion Time of Job	Delay of Job
1	6	0
2	$6 + 4 = 10$	$10 - 4 = 6$
3	$6 + 4 + 5 = 15$	$15 - 12 = 3$
4	$6 + 4 + 5 + 8 = 23$	$23 - 16 = 7$

- Similar reasoning applies to other nodes.

FIGURE 23



- We use the **jumptracking** approach instead of the LIFO (a.k.a. backtracking approach) to branch on the nodes. So, we branch at node 3 and node 4.
- Any job sequence associated with node 4 must have $x_{13} = 1, x_{23} = 1$, or $x_{33} = 1$.
- Branching on node 4 yields nodes 57 in Figure 23.
- For node 7, job 4 will be delayed by 7 days, and job 3 will be the third job processed and completed after $6 + 4 + 5 = 15$ causing a delay of $15 - 12 = 3$ days.
- Any sequence associated with node 7 must have $D \geq 7 + 3 = 10$ days delay.
- Branching at node 7, we arrive at node 8 and 9.
- Now we can eliminate all other nodes using node 8.
- So, the optimal solution is the job sequence $2 - -1 - -3 - -4$ with a total delay of 12 days.

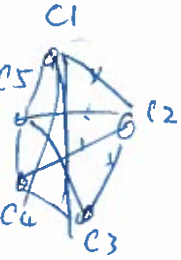
In a network (with or without direction) with n nodes, find a tour visiting each node once in n-steps and back to the original node.

Example 11 Traveling Salesman Problem

- Joe State lives in Gary, Indiana. He owns insurance agencies in Gary, Fort Wayne, Evansville, Terre Haute, and South Bend.
- Each December, he visits each of his insurance agencies.
- The distance between each agency (in miles) is shown in Table 65.

TABLE 65
Distance between Cities in Traveling Salesperson Problem

City	Gary	Fort Wayne	Evansville	Terre Haute	South Bend
City 1 Gary	0	132	217	164	58
City 2 Fort Wayne	132	0	290	201	79
City 3 Evansville	217	290	0	113	303
City 4 Terre Haute	164	201	113	0	196
City 5 South Bend	58	79	303	196	0



- What order of visiting his agencies will minimize the total distance traveled?
- Joe must determine the order of visiting the five cities that minimizes the total distance traveled.
- For example, Joe could choose to visit the cities in the order 134521. Then he would travel a total of $217 + 113 + 196 + 79 + 132 = 737$ miles.
- Let $x_{ij} = 1$ if Joe leaves the ith city, and go to the jth city.
- Let c_{ij} be the distance from the ith city to the jth city for $i \neq j$, and $c_{ii} = M$ for a big M.
- We want to find a low cost solution with $x_{i_1,i_2}, x_{i_2,i_3}, \ldots, x_{i_5,i_1}$ so that $\{i_1, i_2, i_3, i_4, i_5\} = \{1, 2, 3, 4, 5\}$.
- An itinerary that begins and ends at the same city and visits each city once is called a **tour**.
- One may try to solve an assignment problem to get the solution of the TSL.
- We set it up as subproblem 1.
- If the solution of subproblem 1 is a tour, then it is an optimal solution for the TSL.
- But the solution may not be a tour.
- For example, the optimal solution to the assignment problem might be $x_{15} = x_{21} = x_{34} = x_{43} = x_{52} = 1$.
- We will use a branch and bound method.
- The above solution contains two subtours (1521 and 343).
- We will try to exclude the subtour $3 - 4 - 3$ by considering two subproblems.

$x_{15}\ x_{52}\ x_{21}$ x_{34} or x_{43}

Subproblem 2

Subproblem 1 + ($x_{34} = 0$, or $c_{34} = M$).

Subproblem 3

Subproblem 1 + ($x_{43} = 0$, or $c_{43} = M$).

- Apply the Hungarian method to the cost matrix in Table 67.
- The optimal solution to subproblem 2 is $z = 652$, $x_{14} = x_{25} = x_{31} = x_{43} = x_{52} = 1$.
- This solution includes the subtours 1431 and 252, so this cannot be the optimal solution.
- Now branch on subproblem 2 in an effort to exclude the subtour 252.
- To ensure that either x_{25} or x_{52} equals zero. Thus, we add the following subproblems:

TABLE 66
Cost Matrix for Subproblem 1

	City 1	City 2	City 3	City 4	City 5
City 1	M	132	217	164	58
City 2	132	M	290	201	79
City 3	217	290	M	113	303
City 4	164	201	113	M	196
City 5	58	79	303	196	M

TABLE 67
Cost Matrix for Subproblem 2

	City 1	City 2	City 3	City 4	City 5
City 1	M	132	217	164	58
City 2	132	M	290	201	79
City 3	217	290	M	M	303
City 4	164	201	113	M	196
City 5	58	79	303	196	M

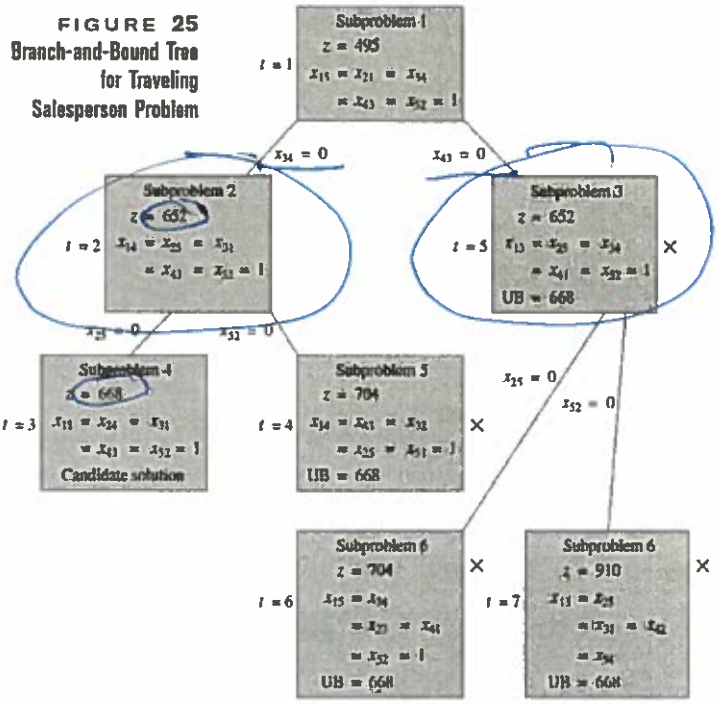


FIGURE 25
Branch-and-Bound Tree for Traveling Salesperson Problem

Subproblem 4

Subproblem 2 $+(x_{25} = 0$, or $c_{25} = M)$.

Subproblem 5

Subproblem 2 $+(x_{52} = 0$, or $c_{52} = M)$.

- Following the LIFO approach, we solve subproblem 4 or subproblem 5.
- Consider subproblem 4 and apply the Hungarian method (Table 68); the optimal solution is $z = 668, x_{15} = x_{24} = x_{31} = x_{43} = x_{52} = 1$.
- This solution contains no subtours and yields the tour 152431.
- Thus, subproblem 4 yields a solution with $z = 668$.
- We then solve subproblem 5 by Hungarian method (Table 69). The optimal solution to subproblem 5 is $z = 704, x_{14} = x_{43} = x_{32} = x_{25} = x_{51} = 1$.
- This solution is a tour, but $z = 704$. The node is fathomed.
- Only subproblem 3 remains; the optimal solution in Table 70 is $x_{13} = x_{25} = x_{34} = x_{41} = x_{52} = 1, z = 652$.
- This solution contains the subtours 1341 and 252. Because $652 < 668$, it is still possible for subproblem 3 to yield a solution with no subtours with $z < 668$.
- Thus, we now branch on subproblem 3 in an effort to exclude the subtours.

TABLE 68
Cost Matrix for Subproblem 4

	City 1	City 2	City 3	City 4	City 5
City 1	M	132	217	164	58
City 2	132	M	290	201	M
City 3	217	290	M	M	303
City 4	164	201	113	M	196
City 5	58	79	303	196	M

TABLE 69
Cost Matrix for Subproblem 5

	City 1	City 2	City 3	City 4	City 5
City 1	M	132	217	164	58
City 2	132	M	290	201	79
City 3	217	290	M	M	303
City 4	164	201	113	M	196
City 5	58	M	303	196	M

TABLE 70
Cost Matrix for Subproblem 3

	City 1	City 2	City 3	City 4	City 5
City 1	M	132	217	164	58
City 2	132	M	290	201	79
City 3	217	290	M	113	303
City 4	164	201	M	M	196
City 5	58	79	303	196	M

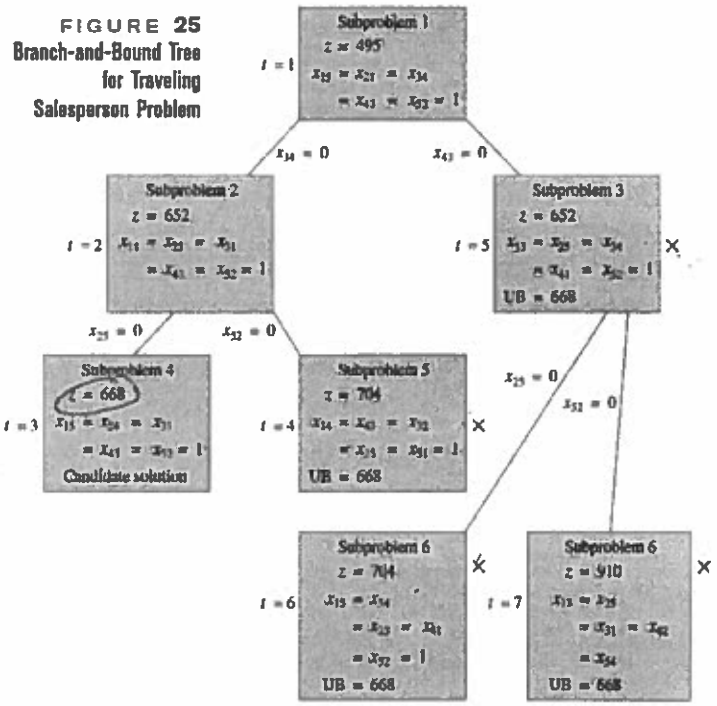


FIGURE 25
Branch-and-Bound Tree for Traveling Salesperson Problem

- Any feasible solution to the traveling salesperson problem that emanates from subproblem 3 must have either $x_{25} = 0$ or $x_{52} = 0$.
- So we create subproblems 6 and 7.

Subproblem 6

Subproblem 3 $+(x_{25} = 0$, or $c_{25} = M)$.

Subproblem 7 Subproblem 3 $+(x_{52} + 0$, or $c_{52} =$

- Solve subproblem 6. The optimal solution is $x_{15} = x_{34} = x_{23} = x_{41} = x_{52} = 1, z = 704$. This node is fathomed.
- The only remaining subproblem is subproblem The optimal solution is $x_{13} = x_{25} = x_{31} = x_{42}$ This node is fathomed.
- So, subproblem 4 yields the optimal solution: Joe should travel from Gary to South Bend, from South Bend to Fort Wayne, from Fort Wayne to Terre Haute, from Terre Haute to Evansville, and from Evansville to Gary. Joe will travel a total distance of 668 miles.

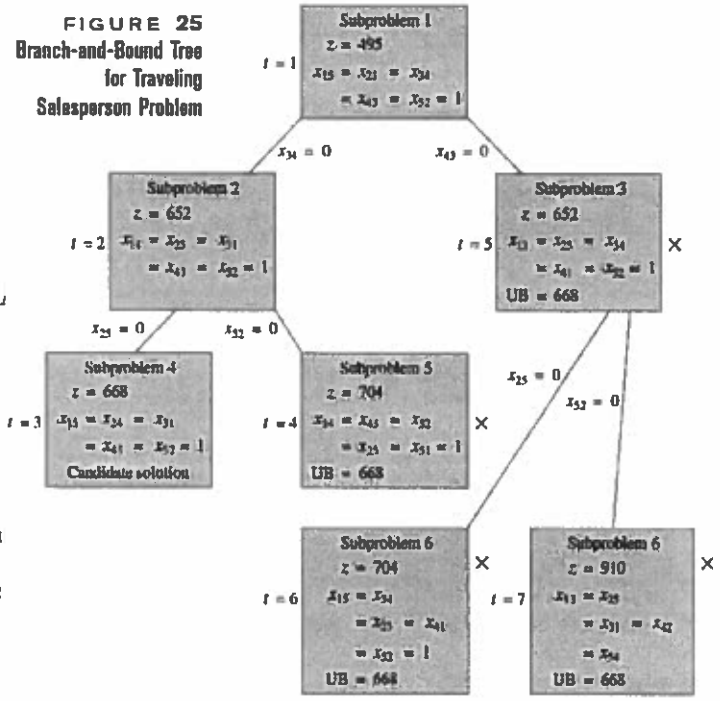


FIGURE 25
Branch-and-Bound Tree for Traveling Salesperson Problem

An integer programming formulation for TSP

We now discuss how to formulate an IP whose solution will solve a TSP. We note, however, that the formulation of this section becomes unwieldy and inefficient for large TSPs. Suppose the TSP consists of cities $1, 2, 3, \ldots, N$. For $i \neq j$ let c_{ij} = distance from city i to city j and let $c_{ii} = M$, where M is a very large number (relative to the actual distances in the problem). Setting $c_{ii} = M$ ensures that we will not go to city i immediately after leaving city i. Also define

$$x_{ij} = \begin{cases} 1 & \text{if the solution to TSP goes from city } i \text{ to city } j \\ 0 & \text{otherwise} \end{cases}$$

Then the solution to a TSP can be found by solving

$$\min z = \sum_i \sum_j c_{ij} x_{ij} \tag{40}$$

$$\text{s.t.} \quad \sum_{i=1}^{N} x_{ij} = 1 \quad (\text{for } j = 1, 2, \ldots, N) \tag{41}$$

$$\sum_{j=1}^{N} x_{ij} = 1 \quad (\text{for } i = 1, 2, \ldots, N) \tag{42}$$

$$u_i - u_j + N x_{ij} \leq N - 1 \quad (\text{for } i \neq j; i = 2, 3, \ldots, N; j = 2, 3, \ldots, N) \tag{43}$$

$$\text{All } x_{ij} = 0 \text{ or } 1, \text{ All } u_j \geq 0$$

- The objective function (40) gives the total length of the arcs included in a tour.
- The constraints in (41) ensure that we arrive once at each city.
- The constraints in (42) ensure that we leave each city once.
- The constraints in (43) are the key to the formulation. They ensure the following:

1) Any set of x_{ij}s containing a subtour will violate (43).

2) Any set of x_{ij}s that forms a tour will be feasible, i.e., there is a set of u_js satisfying (43).

Example Consider the assignment of the five vertex network that contains the subtours $1 - 5 - 2 - 1$ and $3 - 4 - 3$.

We obtain $u_3 - u_4 + 5x_{34} \leq 4$ and $u_4 - u_3 + 5x_{43} \leq 4$.

Adding these constraints yields $5(x_{34} + x_{43}) \leq 8$.

Clearly, this rules out the possibility that $x_{43} + x_{34} = 1$, or any subtour.

Next, we show that if the assignment does not contain a subtour then it is feasible.

Consider the tour $1 - 3 - 4 - 5 - 2 - 1$.

Then we choose $u_1 = 1, u_3 = 2, u_4 = 3, u_5 = 4, u_2 = 5$.

Consider any constraint corresponding to an arc having $x_{ij} = 1$.

For example, the constraint corresponding to x_{52} is $u_5 - u_2 + 5x_{52} \leq 4$.

Because $u_5 - u_2 = -1$, the constraint for x_{52} in (43) reduces to $-1 + 5 \leq 4$.

Now consider a constraint corresponding to an $x_{ij} = 0$, say, x_{32}.

We obtain the constraint $4 \geq u_3 - u_2 + 5x_{32} = u_3 - u_2$, which is at most $5 - 1 = 4$.

9.7 Implicit Evaluation

- Note that an integer constraint on $x < 2^{n+1}$ by setting $x = 2^n u_n + \cdots + u_0$ with $u_i \in \{0, 1\}$.
- We will see that 0-1 problems are easier to solve.
- The tree used in the implicit enumeration method is similar to those used to solve 0-1 knapsack problems.
- At each node, the values of some of the variables are specified.
- For instance, suppose a 0-1 problem has variables $x_1, x_2, x_3, x_4, x_5, x_6$.
- Part of the tree looks like Figure 26.
- At node 4, the values of x_3, x_4, and x_2 are specified.
- These variables are referred to as **fixed** variables.
- All variables whose values are unspecified at a node are called free variables.
- Thus, at node 4, x_1, x_5, and x_6 are **free** variables.
- For any node, a specification of the values of all the free variables is called a **completion** of the node.
- Thus $x_1 = 1, x_5 = 1, x_6 = 0$ is a completion of node 4.

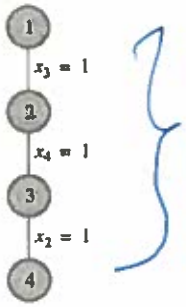


FIGURE 26

Some general ideas:

1. At a node, find a completion of the free variables that optimize the objective function.
   If it is feasible, then we have the optimal solution at the node and no branching is needed.
   For example, $\max z = 4x_1 + 2x_2 - x_3 + 2x_4$
   s.t. $x_1 + 3x_2 - x_3 - 2x_4 \geq 1$,
   $x_i \in \{0, 1\}, \quad i = 1, 2, 3, 4.$
   If $(x_1, x_2, x_3, x_4) = (0, 1, 0, 1)$ is given, we see that the solution is feasible. The node is fathomed with $z = 4$.
2. If an optimal completion is not feasible, then we have an upper bound for the completion of the rest of the node.
3. If an optimal completion is worse than a feasible solution found before, the node can be removed.
4. If there is a constraint such that the most feasible completion fails, then there is no completion that are feasible.