Joe Wilck, Integer Programming, *Branch & Bound*
("Operations Research: Applications & Algorithms," 4th E, by W. L. Winston, Duxbury Press, 2003; Chapter 9):

Branch & Bound: A solution procedure for all types of integer programs (i.e., binary, mixed, pure, combinatorial) that enumerates the feasible solutions in a systematic way in order to eliminate feasible solutions while finding the optimal solution.

Key Features of Branch and Bound:
1. Branching Rule: Separating the original (integer) problem into subproblems (and subproblems into more defined subproblems) within a feasible region. The goal is to create subproblems that are associated with a specific portion of the feasible region, and by enumerating all of the subproblems, then you would have considered the entire feasible region.

2. Relaxation: An integer program can be relaxed to form a linear program (LP). This LP's optimal solution is optimal (or *superoptimal*) to the IP's optimal solution. This creates a bound. The solution can then be branched by fixing the integer variables to specific integer values. Some integer problems can be relaxed in different ways. For example, the TSP can be relaxed using the assignment problem (by eliminating the subtour elimination constraints in the relaxation). [Recall, the assignment problem gives integer solutions without LP relaxation.]

3. Bounds: If you have a maximization problem, then the optimal solution to the relaxed problem is greater than or equal to the optimal solution to the integer (original) problem. [If you have a minimization problem, then optimal solution to the relaxed problem is less than or equal to the optimal solution to the integer problem.] Any feasible solution to the integer (original) problem is a lower bound to the original problem (maximization problem). The current best feasible solution is the incumbent solution.

4. Partial Solution: Due to subproblem restrictions, each subproblem is associated with a partial solution. Some variables are fixed (already determined), and the remainder are free (not yet determined). For example, consider the solution: $x = (\#, 1, 0, \#)$. This solution means that: x_2 is fixed at 1, x_3 is fixed at 0, and x_1 & x_4 are free.

5. Subproblem Selection Rules (Two Most Common Rules):
Best Bound (textbook = jumptracking): choose the partial solution with the best objective value
Depth First (textbook = backtracking): choose the partial solution with the most variables fixed
Best Bound uses more computer memory than Depth First. Each is better for certain problems, but neither method is strictly better than the other.

6. Fathoming: A subproblem is fathomed (branch is cutoff) if it resolved, or if no further branching is necessary. This works because the relaxed problem will have a larger (or equal to) feasible region than the original problem (for both minimization and maximization problems). Fathom if: (A) If the subproblem is infeasible. (B) If an optimal (feasible) solution is found for the subproblem. (C) If the current incumbent solution cannot be improved by further branching.

The goal is to branch-and-bound until the incumbent solution is equal to the upper-bound (for a maximization problem). [The goal is to branch-and-bound until the incumbent solution is equal to the lower-bound (for a minimization problem).]

Example (Knapsack Problem):

Original Problem:

Maximize $12x_1 + 16x_2 + 22x_3 + 8x_4$

S.T. $4x_1 + 5x_2 + 7x_3 + 3x_4 \leq 14$

x_i ~ binary for all i

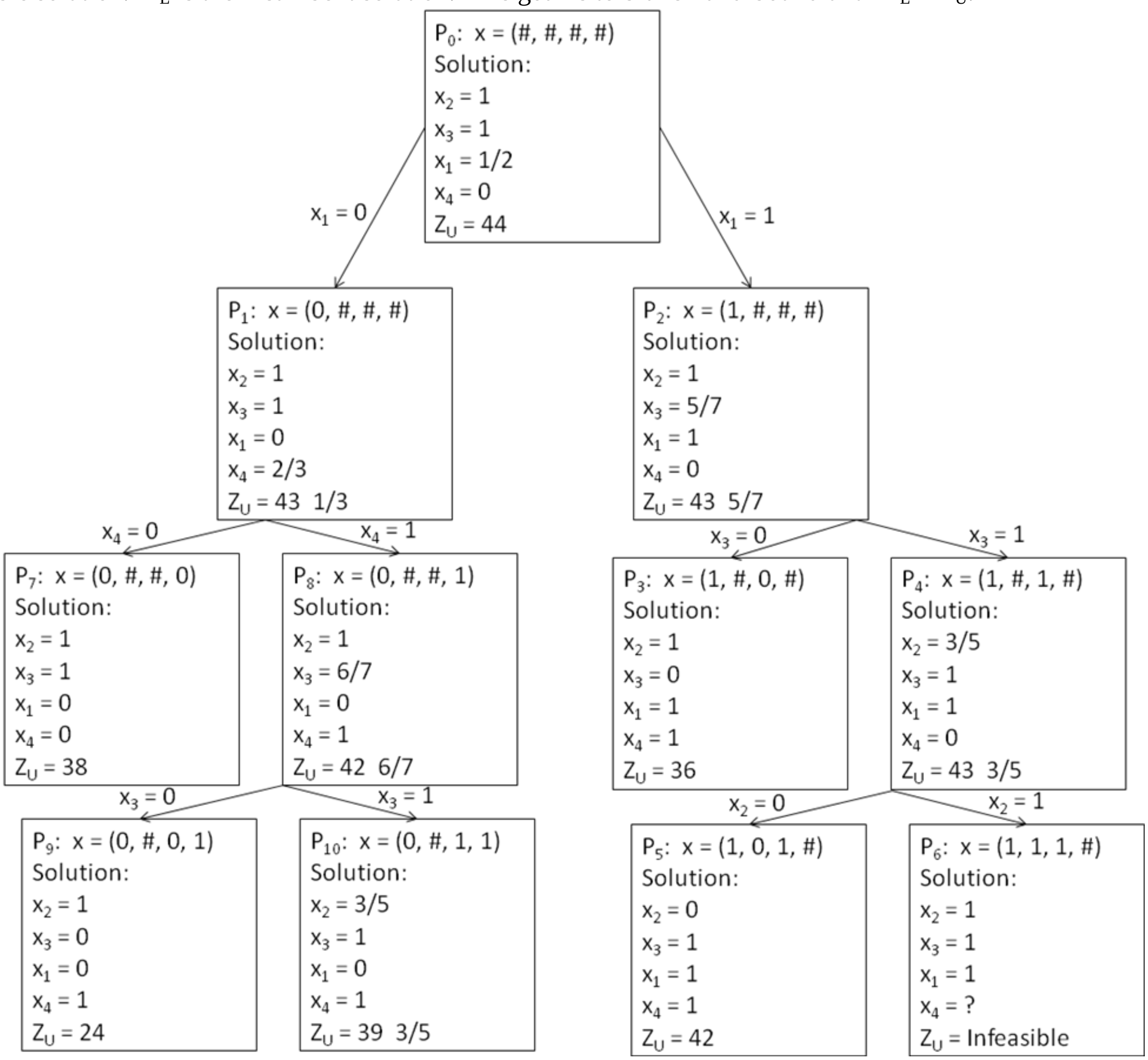
Relaxed Version:

Maximize $12x_1 + 16x_2 + 22x_3 + 8x_4$

S.T. $4x_1 + 5x_2 + 7x_3 + 3x_4 \leq 14$

$0 \leq x_i \leq 1$ for all i

Let P_0 signify initial solution (relaxing binary restrictions); which is the LP optimal solution. Let # signify a free variable. Let Z_U signify an upper-bound (since a maximization problem), which does not have to be an integer feasible solution. Let Z_L signify the best lower-bound (since a maximization problem), and Z_L must be an integer feasible solution. Z_L is the incumbent solution. The goal is to branch-and-bound until $Z_L = Z_U$.



Example:
The mechanics of the Indy Car racing team have six different features that might be added to this year's car to improve its top speed. The following table lists the estimated cost and speed enhancement of each feature.

	Proposed Feature					
	1	2	3	4	5	6
Cost ($)	10200	6000	23800	11100	9800	31600
Speed Increase (mph)	8	3	15	7	10	12

If Indy Car wants to maximize performance gain without exceeding a budget of $35,000, then formulate this problem and solve by branch-and-bound clearly summarizing your solution on a branch-and-bound tree.
Hints:
Formulate as a knapsack problem.
For branch-and-bound pick one method (Best Bound or Depth First) and stick with it.
Looking at the ratio of Speed versus Cost should give you an idea of which variables should be "1" rather than "0".
In other words, variables associated with the highest ratio of Speed/Cost should be "1" if possible.

Solution:
General Formulation:
Index Sets:
j: proposed car feature j (j=1, …, J)

Data:
C_j: cost of feature j ($)
S_j: speed increase of feature j (mph)
B: budget ($)

Variables:
X_j: 1 if feature j is used; 0 otherwise

Objective:

Maximize Speed Increase = Z = $\sum_{j=1}^{J} X_j * S_j$

Constraints:

$\sum_{j=1}^{J} X_j * C_j \le B$ (maintain budget)

X_j binary $\forall$ j (variable restrictions)

Specifics:
Index Sets: j = 1, 2, 3, 4, 5, 6
Data:
C_j = 10200, 6000, 23800, 11100, 9800, 31600
S_j = 8, 3, 15, 7, 10, 12
B = 35,000
Per Unit Cost Increase

j	S (mph)	C ($000)	S/C	Rank
1	8	10.2	0.7843	2
2	3	6	0.5000	5
3	15	23.8	0.6303	4
4	7	11.1	0.6306	3
5	10	9.8	1.0204	1
6	12	31.6	0.3797	6

Therefore, X_5 would enter first, followed by X_1, …, until X_6 enters or until the budget is used.

Branch-And-Bound Tree (Using Best Bound Method):

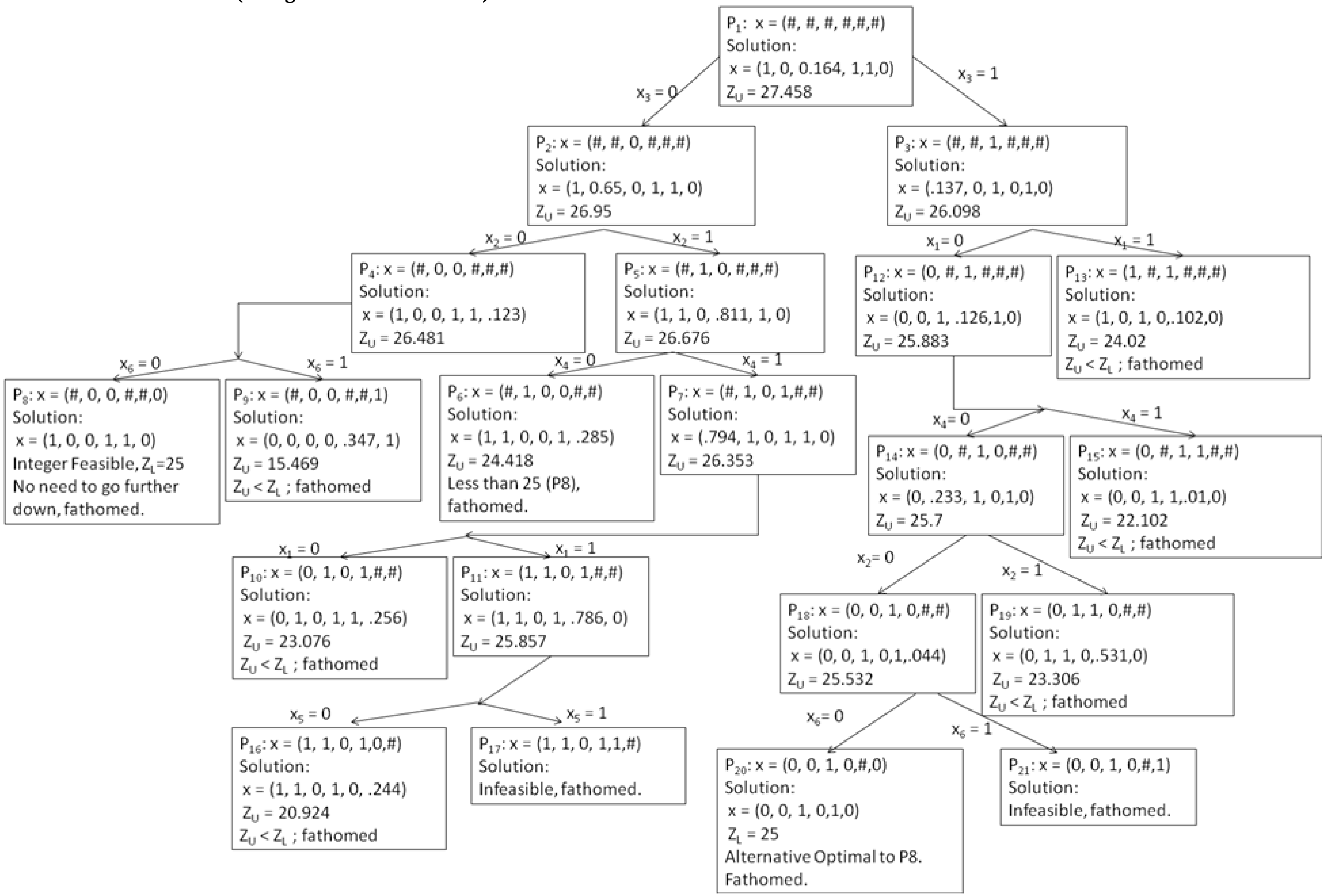


Table of Nodes and LP Relaxation with Optimal Solutions
(Node P's correspond to Branch and Bound Tree)
Z is given in $1000's.

Node	**Free/Fixed**						**Relaxed Solution**						**Relaxed**	**Relaxed**
P	**x1**	**x2**	**x3**	**x4**	**x5**	**x6**	**x1**	**x2**	**x3**	**x4**	**x5**	**x6**	**Z**	**Cost**
1	#	#	#	#	#	#	1.000	0.000	0.164	1.000	1.000	0.000	27.458	35
2	#	#	0	#	#	#	1.000	0.650	0.000	1.000	1.000	0.000	26.950	35
3	#	#	1	#	#	#	0.137	0.000	1.000	0.000	1.000	0.000	26.098	35
4	#	0	0	#	#	#	1.000	0.000	0.000	1.000	1.000	0.123	26.481	35
5	#	1	0	#	#	#	1.000	1.000	0.000	0.811	1.000	0.000	26.676	35
6	#	1	0	0	#	#	1.000	1.000	0.000	0.000	1.000	0.285	24.418	35
7	#	1	0	1	#	#	0.794	1.000	0.000	1.000	1.000	0.000	26.353	35
8	#	0	0	#	#	0	1.000	0.000	0.000	1.000	1.000	0.000	**25.000**	31.1
9	#	0	0	#	#	1	0.000	0.000	0.000	0.000	0.347	1.000	15.469	35
10	0	1	0	1	#	#	0.000	1.000	0.000	1.000	1.000	0.256	23.076	35
11	1	1	0	1	#	#	1.000	1.000	0.000	1.000	0.786	0.000	25.857	35
12	0	#	1	#	#	#	0.000	0.000	1.000	0.126	1.000	0.000	25.883	35
13	1	#	1	#	#	#	1.000	0.000	1.000	0.000	0.102	0.000	24.020	35
14	0	#	1	0	#	#	0.000	0.233	1.000	0.000	1.000	0.000	25.700	35
15	0	#	1	1	#	#	0.000	0.000	1.000	1.000	0.010	0.000	22.102	35
16	1	1	0	1	0	#	1.000	1.000	0.000	1.000	0.000	0.244	20.924	35
17	1	1	0	1	1	#	1.000	1.000	0.000	1.000	1.000	0.000	28.000	**37.1**
18	0	0	1	0	#	#	0.000	0.000	1.000	0.000	1.000	0.044	25.532	35
19	0	1	1	0	#	#	0.000	1.000	1.000	0.000	0.531	0.000	23.306	35
20	0	0	1	0	#	0	0.000	0.000	1.000	0.000	1.000	0.000	**25.000**	33.6
21	0	0	1	0	#	1	0.000	0.000	1.000	0.000	0.000	1.000	27.000	**55.4**

* Best Bound Method Used
* Yellow Means that those are fixed.
* Red means that those are infeasible

There are two integer feasible optimal solutions (Nodes P8 and P20) with a maximum speed increase of 25 mph. However, Node P8 is recommended since it is the least cost solution of the two optimal solutions, its total cost is $31,100. Furthermore, node P8 was the first integer feasible optimal solution found. The P8 node corresponds with using features 1, 4, and 5.