Topics in Computational Sustainability: Integer Linear Programming

Violates Divisibility Assumption of LP

- **Divisibility Assumption of Linear Programming (LP)**: Decision variables in a Linear Programming model are allowed to have *any* values, including *fractional (noninteger)* values, that satisfy the functional and nonnegativity constraints i.e., activities can be run at *fractional levels*..

When divisibility assumption is violated then apply Integer Linear Programming!!!

Why Integer Linear Programming?

- Advantages of restricting the variables to take on integer values
  - More realistic
  - More flexibility

- Disadvantages
  - More difficult to model
  - Can be <u>much</u> more difficult to solve

Integer Linear Programming

- When are “non-integer” solutions okay?
  - Solution is naturally divisible
    - e.g., pounds, hours
  - Solution represents a rate
    - e.g., units per week
- When is rounding okay?
  - When numbers are large
    - e.g., rounding 114.286 to 114 is probably okay.
- When is rounding not okay?
  - When numbers are small
    - e.g., rounding 2.6 to 2 or 3 may be a problem.
  - Binary variables
    - yes-or-no decisions

Graphical Method for Integer Programming

- When a Binary Integer Programming problem has just **two decision variables**, its optimal solution can be found by applying the *Graphical Method for Linear Programming* with just one change at the end.
- Let us begin by graphing the feasible region for the LP relaxation, determining the slope of the objective function lines, and moving a straight edge with this slope through this feasible region in the direction of improving values of the objective function.
- However, rather than stopping at the last instant the straight edge passes through this feasible region, now stop at the last instant the straight edge passes through an integer point that lies within this feasible region.
- This integer point is the optimal solution.

Green Power Company Example: Capital Budgeting Allocation Problem

- A Green Power Company would like to consider 6 investments. The cash required from each investment as well as the total electricity production of the investment is given next. The cash available for the investments is $14M. Danish Green Power wants to maximize electricity production. What is the optimal strategy?
- An investment can be selected or not. One cannot select a fraction of an investment.

Data for the Problem

Investment budget = $14M

Investment	**1**	**2**	**3**	**4**	**5**	**6**
Cost (millions dollars)	5	7	4	3	4	6
Total production (100KW)	16	22	12	8	11	19

Capital Budgeting Allocation Problem (one resource) (Mapped to the Knapsack Problem)

- **Why is a problem with the characteristics of the previous problem called the Knapsack Problem?**
- **It is an abstraction, considering the simple problem:**

A hiker tries to fill knapsack to maximum total value. Each item we consider taking with us has a certain value and a certain weight. An overall weight limitation gives the single constraint.

Practical Applications:

- Project selection and capital budgeting allocation
- Storing a warehouse to maximum value given the indivisibility of goods and space limitations

Integer Programming Formulation

What are the Decision Variables?

$$x_i = \begin{cases} 1, & \text{if you choose item } i = 1,\ldots,6, \\ 0, & \text{else} \end{cases}$$

Objective Function and Constraints?

$$\mathit{Max}\ 16x_1 + 22x_2 + 12x_3 + 8x_4 + 11x_5 + 19x_6$$

$$5x_1 + 7x_2 + 4x_3 + 3x_4 + 4x_5 + 6x_6 \leq 14$$

$$x_j \in \{0,1\} \text{ for each } j = 1 \text{ to } 6$$

- The previous constraints represent "economic indivisibilities", either a project is selected, or not. It is not possible to select a fraction of a project.
- Similarly, integer variables can model logical requirements (e.g., if item 2 is selected, then so is item 1.)

How to model “logical” constraints

- Exactly 3 items are selected.
- If item 2 is selected, then so is item 1.
- If item 1 is selected, then item 3 is not selected.
- Either item 4 is selected or item 5 is selected, but not both.

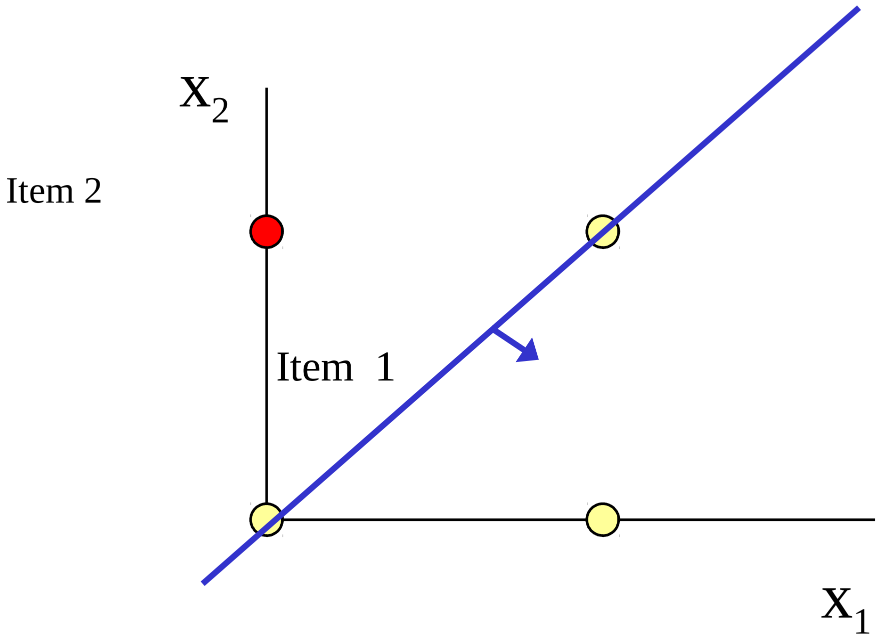
Formulating Constraints

- Exactly 3 items are selected

$$x_1 + x_2 + x_3 + x_4 + x_5 + x_6 = 3$$

If item 2 is selected then so is item 1

A 2-dimensional representation



The integer programming constraint:

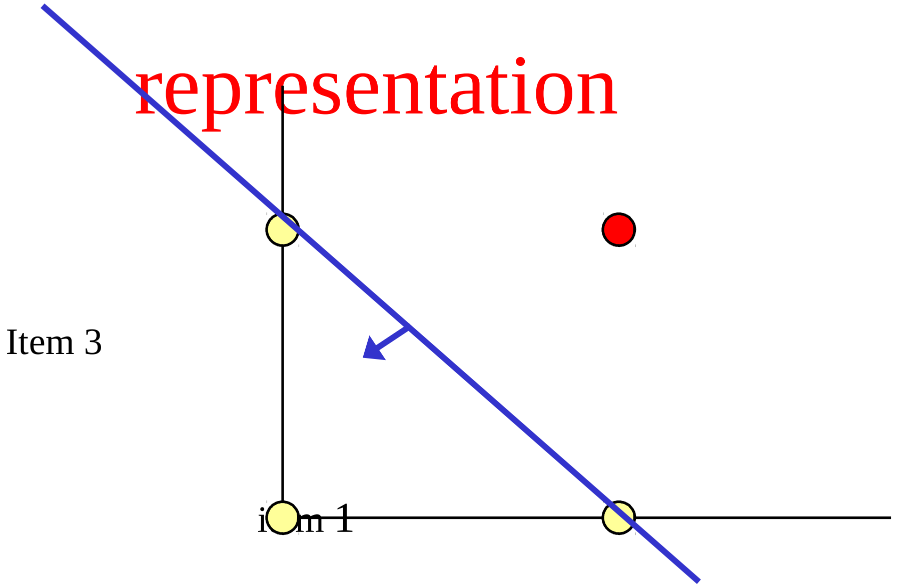
$$\mathbf{x_1 \geq x_2}$$

If item 1 is selected then item 3 is not selected

The integer programming constraint:

A 2-dimensional

representation

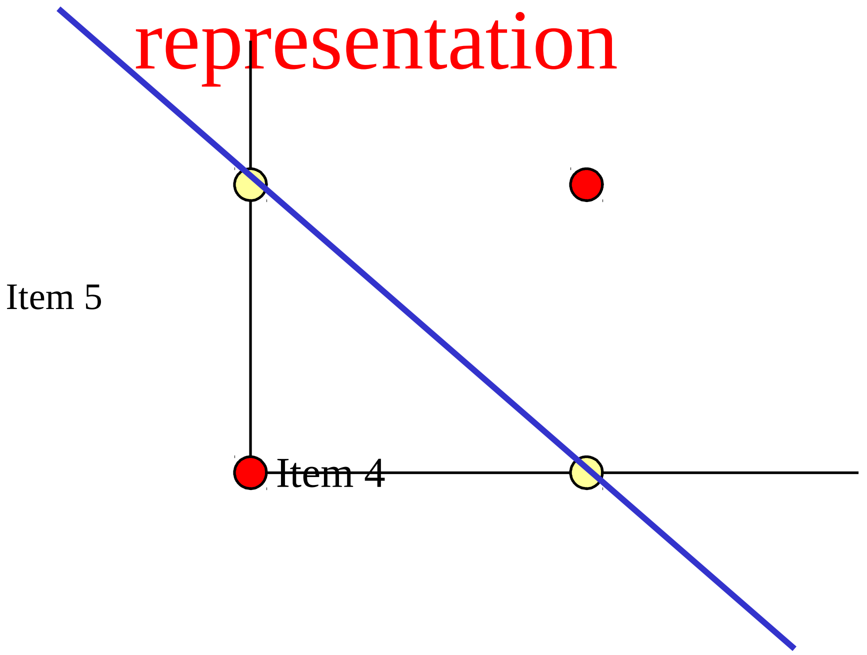


$$x_1 + x_3 \leq 1$$

Either item 4 is selected or item 5 is selected, but not both.

The integer programming constraint:

A 2-dimensional representation



$$\mathbf{x_4 + x_5 = 1}$$

How to model "logical" constraints

- Exactly 3 items are selected. $x_1 + x_2 + x_3 + x_4 + x_5 + x_6 = 3$
- If item 2 is selected, then so is item 1. $x_1 \geq x_2$
- If item 1 is selected, then item 3 is not selected.

$$x_1 + x_3 \leq 1 \quad \Box \; x_1 \leq 1 - x_3$$

- Either item 4 is selected or item 5 is selected, but not both.

$$x_4 + x_5 = 1$$

Modeling Fixed Charge Problems

If a product is produced, a factory is built □ must incur a fixed setup cost.

→ The problem is non-linear.

x – quantity of product to be manufactured

$x = 0$ □ cost =0;

$x > 0$ □ cost = $C_1x + C_2$

→ How to model it? Using an *indicator* variable *y*

y = 1 □ x is produced; y = 0 □ x is not produced

Objective function becomes □ $C_1x + C_2y$

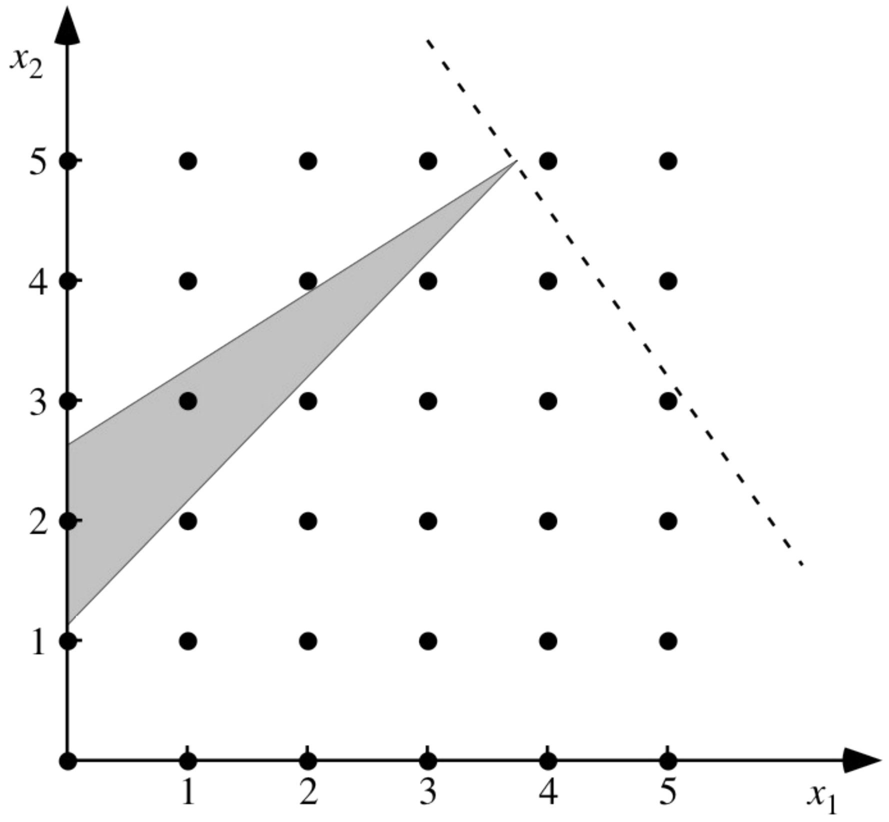
Additional Constraint □ $x \leq My$

M is a big number

Solving Integer Programs

The Challenges of Rounding

- Rounded Solution may not be feasible.
- Rounded solution may not be close to optimal.
- There can be *many* rounded solutions.
  - Example: Consider a problem with 30 variables that are non-integer in the LP-solution. How many possible rounded solutions are there?

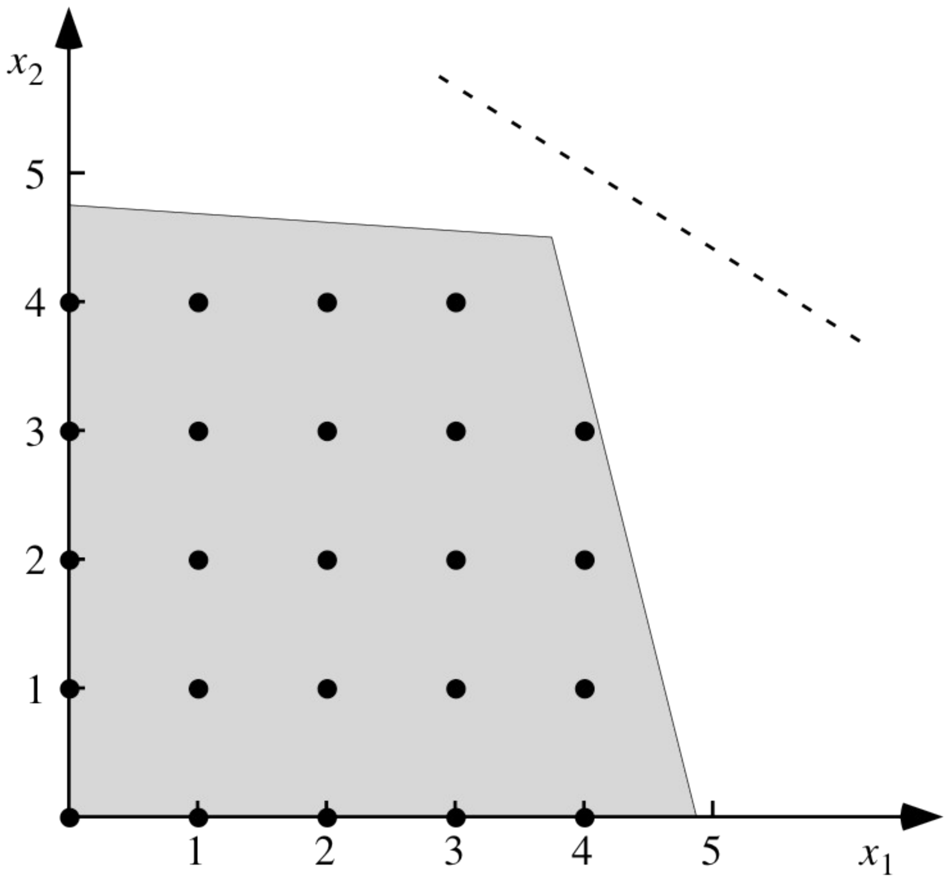


Overview of Techniques for Solving Integer Programs

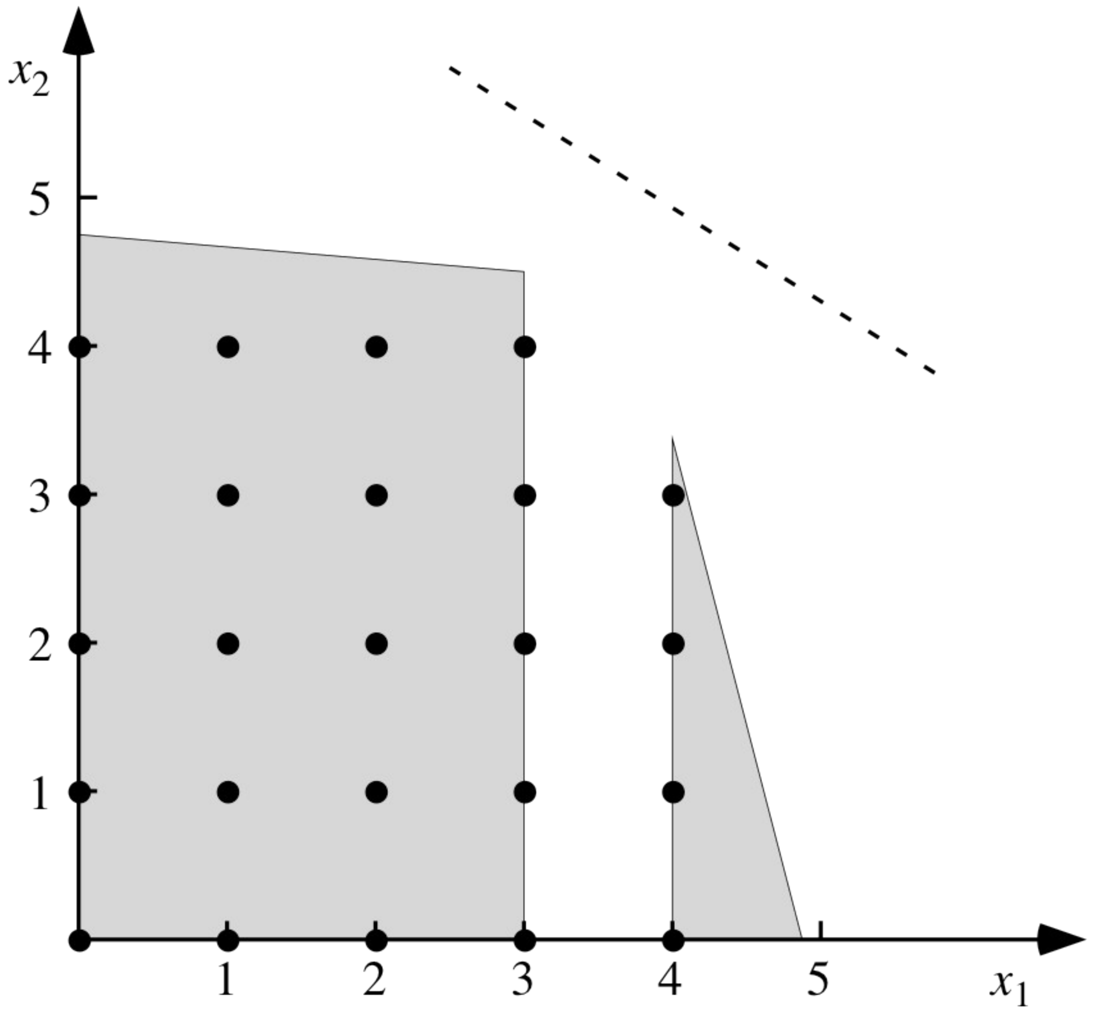
- Enumeration Techniques
  - Complete Enumeration
    - list all “solutions” and choose the best
  - Branch and Bound
    - Implicitly search all solutions, but cleverly eliminate the vast majority before they are even searched

- Cutting Plane Techniques
  - Use Linear Programming (LP) to solve integer programs by adding constraints to eliminate the fractional solutions.
- Hybrid Approaches (e.g., Branch and Cut)

How Integer Programs are Solved: Cuts



How Integer Programs are Solved



Branch and Bound

- Implicit enumeration of all the solutions:
  - It is the starting point for all solution techniques for integer programming
- Lots of research has been carried out over the past 40 years to make it more and more efficient.
- It is an art form to make it efficient (We shall get a sense why).
- Integer programming is intrinsically difficult.

Knapsack Problem:
Binary Integer Programming Formulation

What are the decision variables

$$x_i = \begin{cases} 1, & \text{if you choose item } i = 1,\ldots,6, \\ 0, & \text{else} \end{cases}$$

$$\mathit{Max}\ 16x_1 + 22x_2 + 12x_3 + 8x_4 + 11x_5 + 19x_6$$

$$5x_1 + 7x_2 + 4x_3 + 3x_4 + 4x_5 + 6x_6 \leq 14$$

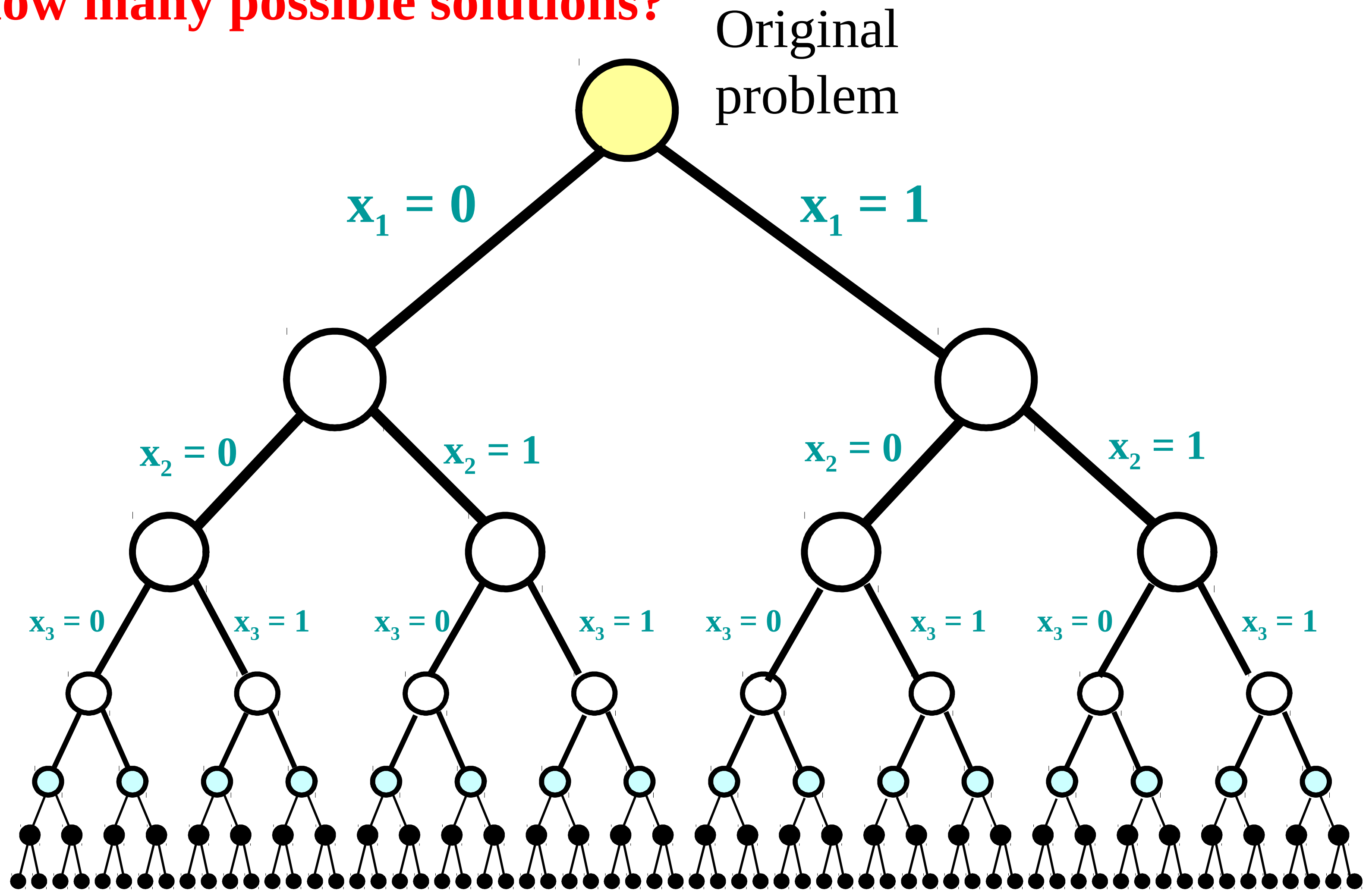
$$x_j \ \varepsilon \ \{0,1\} \text{ for each } j = 1 \text{ to } 6$$

Complete Enumeration

- Systematically considers all possible values of the decision variables.
  - If there are n binary variables, there are 2^n different ways.
- Usual idea: iteratively break the problem in two. At the first iteration, we consider separately the case that $x_1 = 0$ and $x_1 = 1$.

An Enumeration Tree

How many possible solutions?



On Complete Enumeration

- Suppose that we could evaluate 1 billion solutions per second.
- Let n = number of binary variables
- Solutions times (worst case, approx.)
  - n = 30, 1 second
  - n = 40, 18 minutes
  - n = 50 13 days
  - n = 60 31 years

On Complete Enumeration

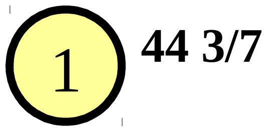
- Suppose that we could evaluate 1 trillion solutions per second, and instantaneously eliminate 99.9999999% of all solutions as not worth considering
- Let n = number of binary variables
- Solutions times
  - n = 70, 1 second
  - n = 80, 17 minutes
  - n = 90 11.6 days
  - n = 100 31 years

Branch and Bound

The essential idea: search the enumeration tree, but at each node

1. Solve the Linear Program (LP) at the node (by relaxing the integrality constraints)
2. Eliminate (Prune) the subtree if
    1. The solution is integer (there is no need to go further- why?) or
    2. The best solution in the subtree cannot be as good as the best available solution (the incumbent- how does that happen?) or
    3. There is no feasible solution

Branch and Bound



Node 1 is the Original LP Relaxation

maximize $16x_1 + 22x_2 + 12x_3 + 8x_4 + 11x_5 + 19x_6$

subject to $5x_1 + 7x_2 + 4x_3 + 3x_4 + 4x_5 + 6x_6 \leq 14$

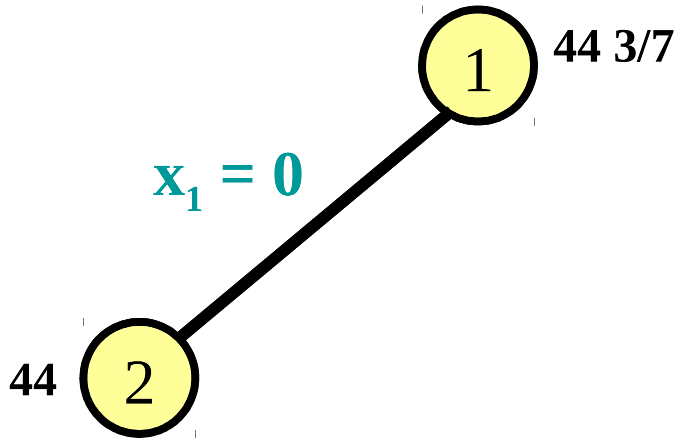
$0 \leq x_j \leq 1$ **for** $j = 1$ **to** 6

Solution at node 1:

$x_1 = 1 \quad x_2 = 3/7 \quad x_3 = x_4 = x_5 = 0 \quad x_6 = 1 \quad z = 44\ 3/7$

The IP cannot have value higher than 44 3/7.

Branch and Bound



Node 2 is the original LP Relaxation plus the constraint $x_1 = 0$.

maximize $16x_1 + 22x_2 + 12x_3 + 8x_4 + 11x_5 + 19x_6$

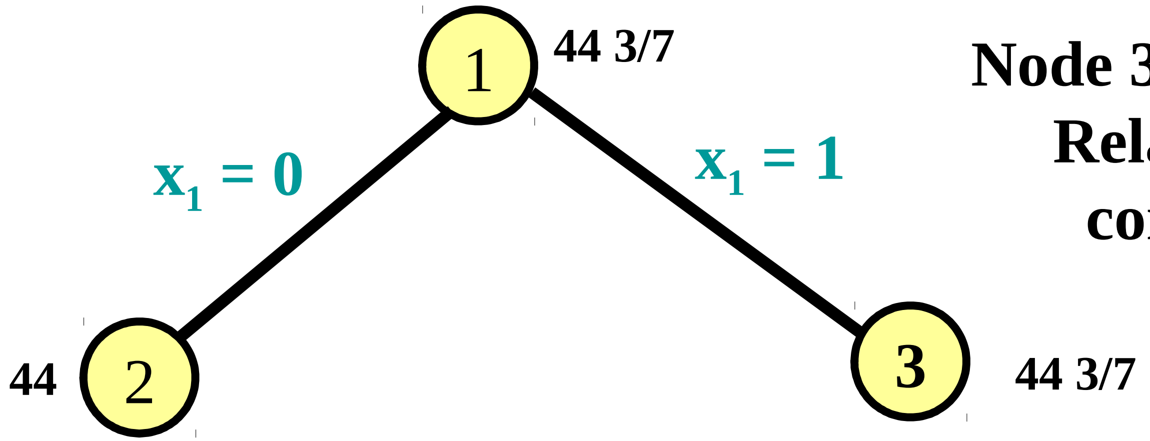
subject to $5x_1 + 7x_2 + 4x_3 + 3x_4 + 4x_5 + 6x_6 \leq 14$

$0 \leq x_j \leq 1$ **for** $j = 1$ **to** 6, $x_1 = 0$

Solution at node 2:

$x_1 = 0 \quad x_2 = 1 \quad x_3 = 1/4 \quad x_4 = x_5 = 0 \quad x_6 = 1 \quad z = 44$

Branch and Bound



Node 3 is the original LP Relaxation plus the constraint $x_1 = 1$.

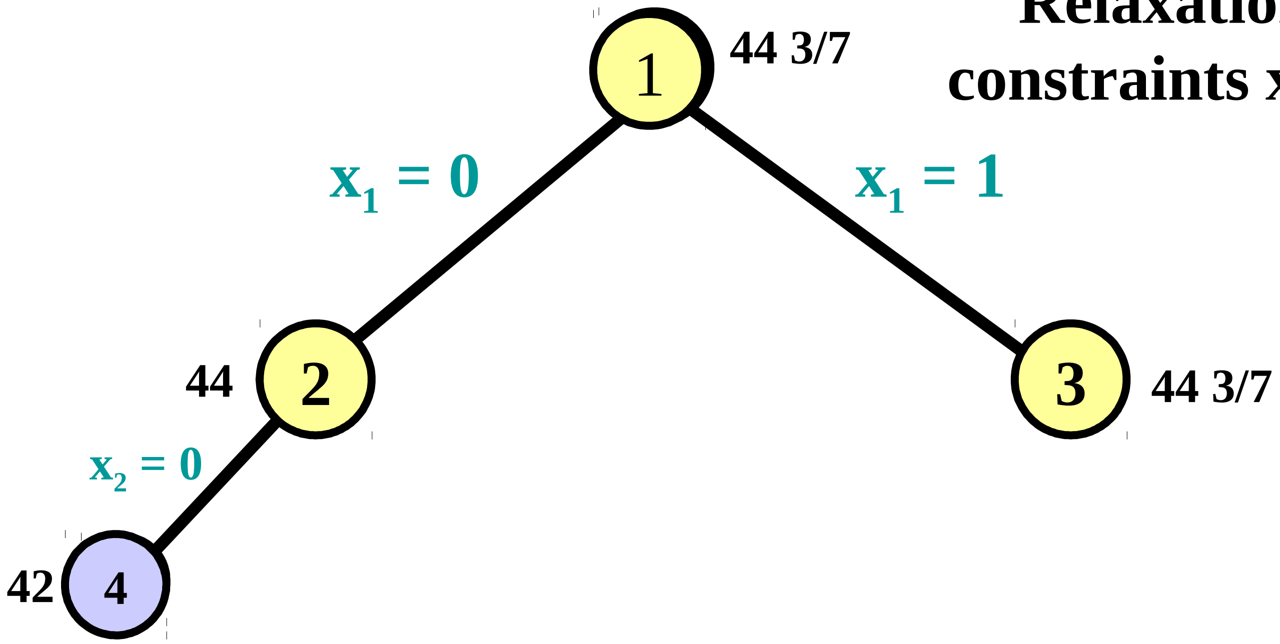
The solution at node 1 was

$x_1 = 1$ $x_2 = 3/7$ $x_3 = x_4 = x_5 = 0$ $x_6 = 1$ $z = 44\ 3/7$

Note: it was the best solution with no constraint on x_1. So, it is also the solution for node 3. (If you add a constraint, and the old optimal solution is feasible, then it is still optimal.)

Branch and Bound

Node 4 is the original LP Relaxation plus the constraints $x_1 = 0$, $x_2 = 0$.



Solution at node 4: 0 0 1 0 1 1 z = 42

Our first ***incumbent*** solution!

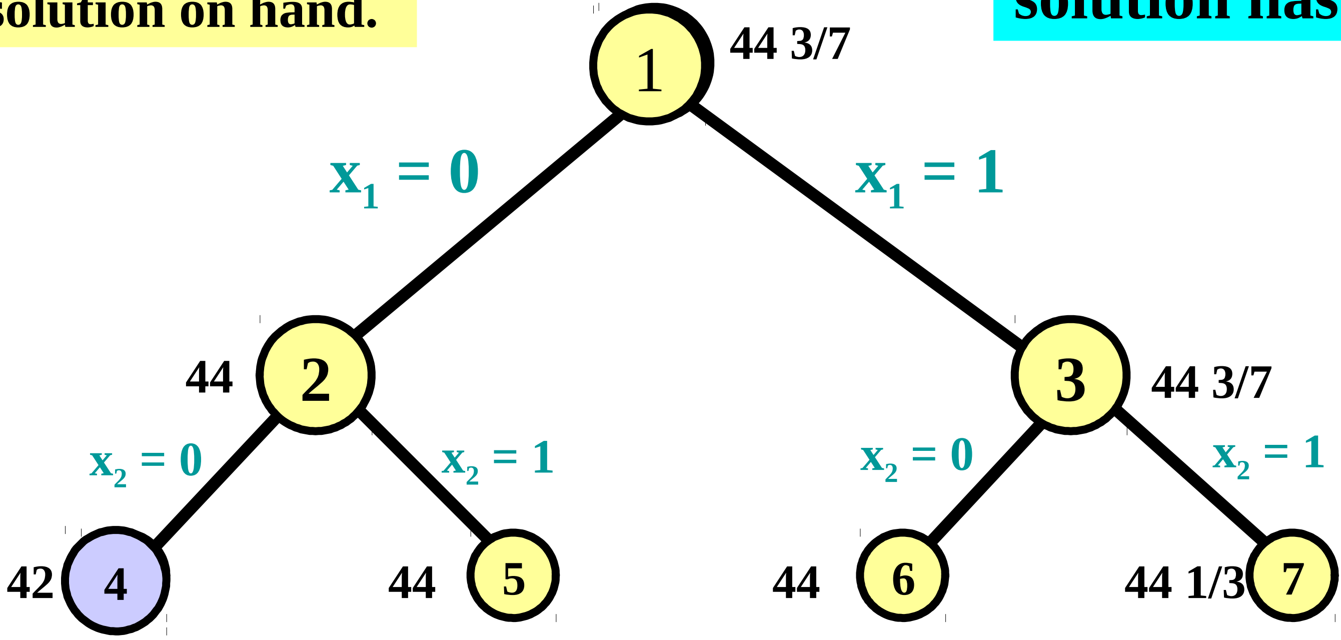
No further searching from node 4 because there cannot be a better integer solution.

No solution in the subtree can have a value better than 42.

Branch and Bound

The incumbent is the best solution on hand.

The incumbent solution has value 42



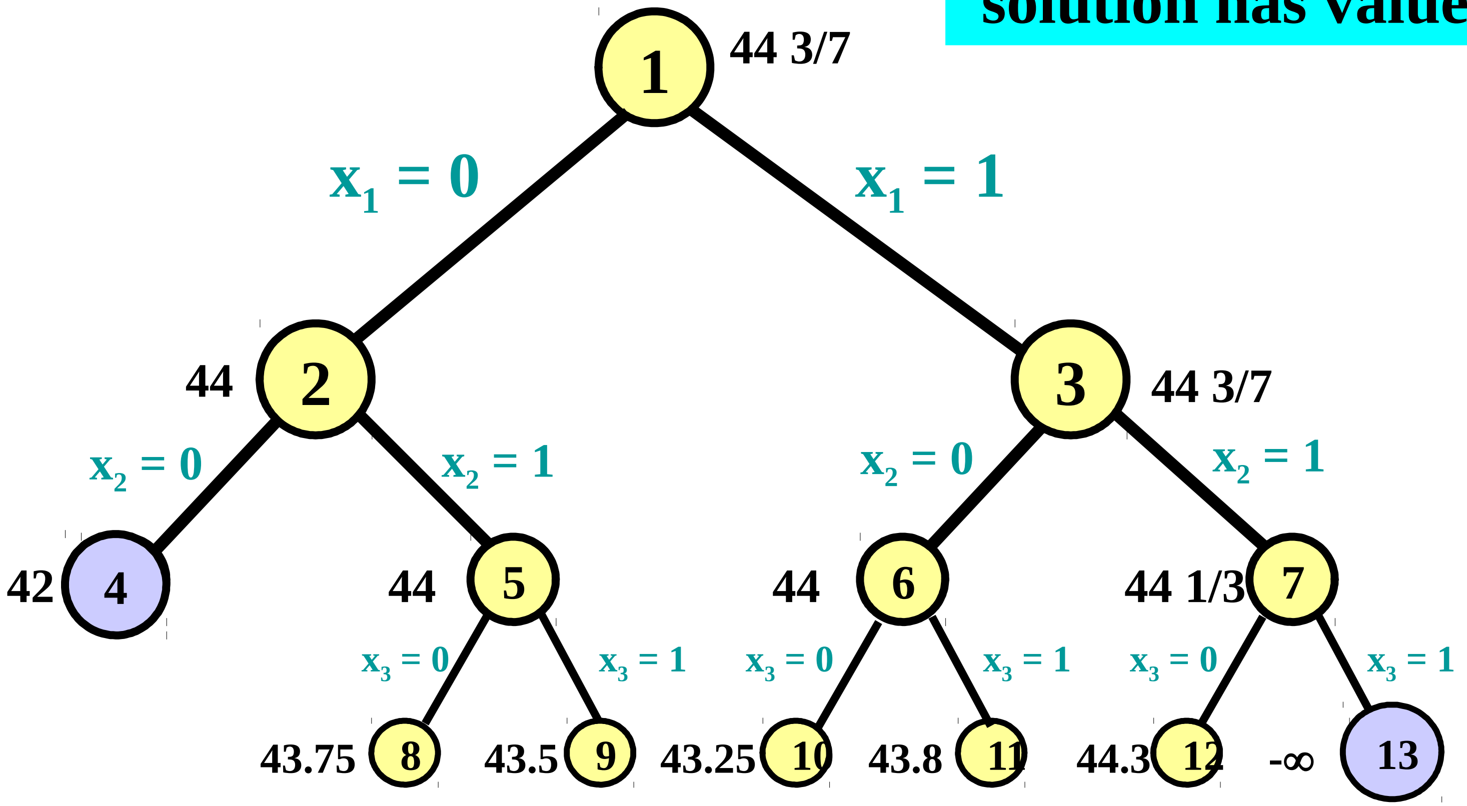
We next solved the LP's associated with nodes 5, 6, and 7.

No new integer solutions were found.

We would eliminate (prune) a subtree if we were guaranteed that no solution in the subtree were better than the incumbent.

Branch and Bound

The incumbent solution has value 42

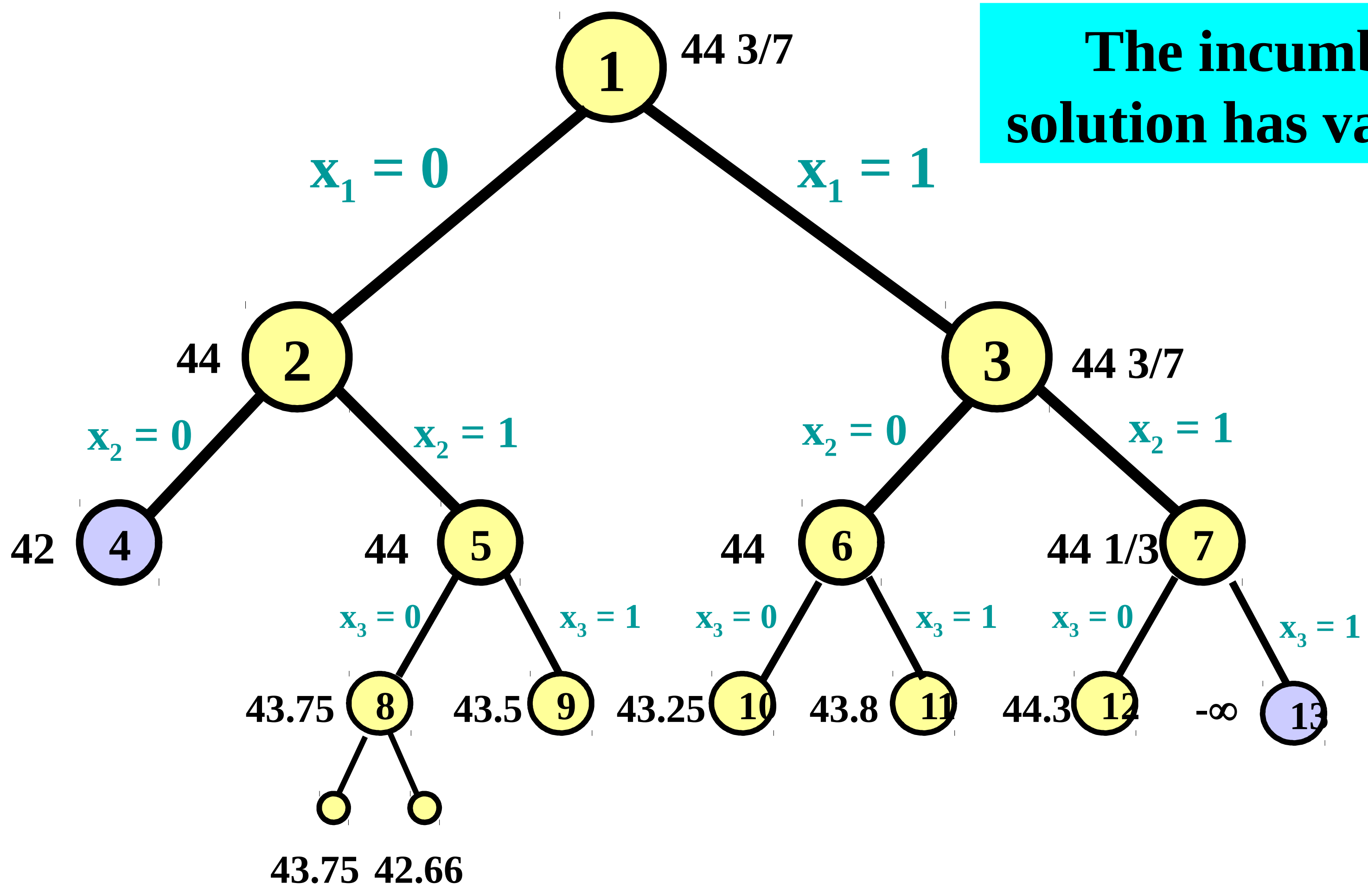


We next solved the LPs associated with nodes 8-13

Summary so far

- We have solved 13 different linear programs so far.
  - One integer solution was found
  - One subtree pruned because the solution was integer (node 4)
  - One subtree pruned because the solution was infeasible (node 13)
  - No subtrees pruned because of the bound

Branch and Bound



The incumbent solution has value 42

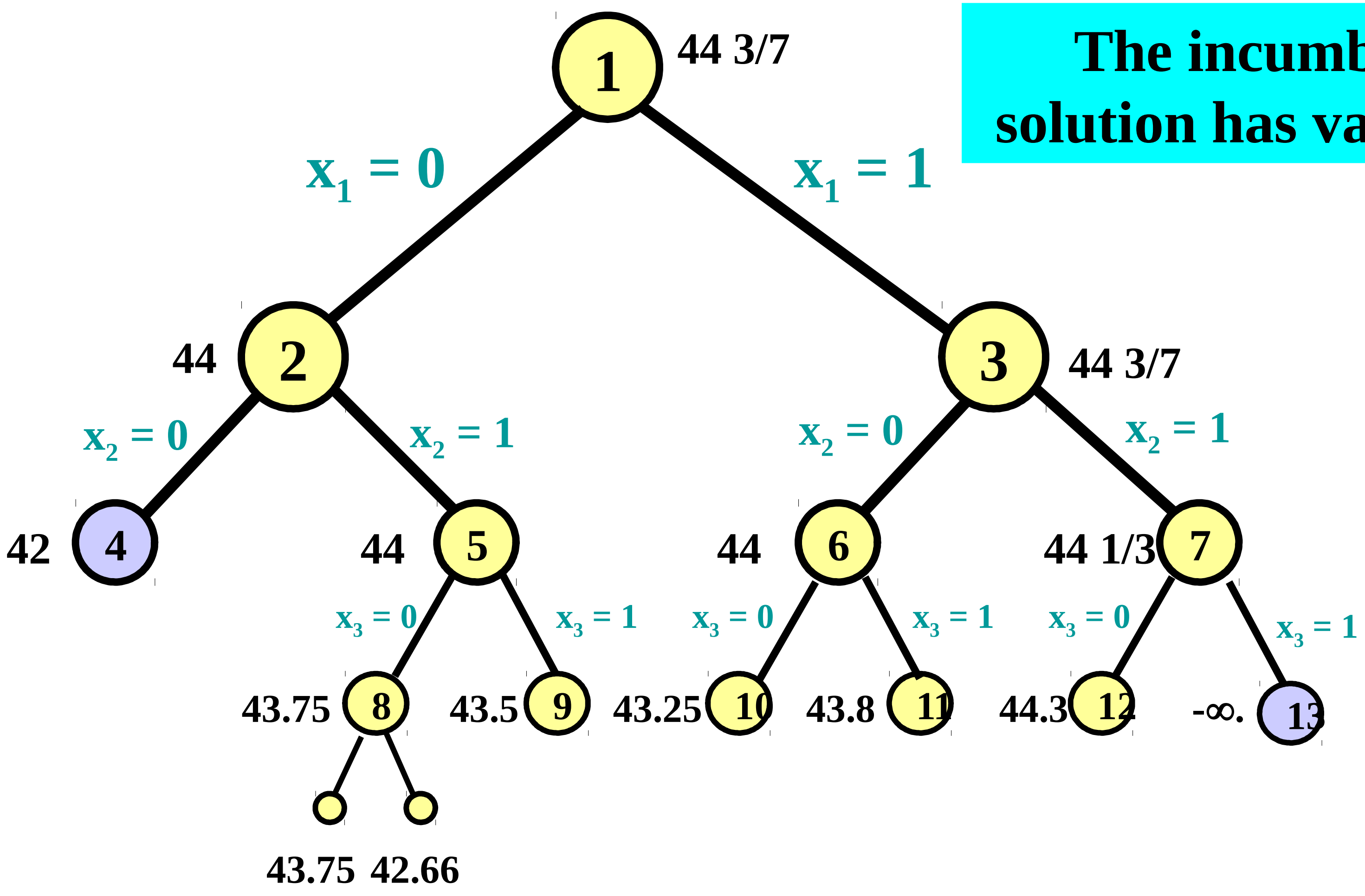
We next solved the LPs associated with the next nodes.

We can prune the node with z = 42.66. Why?

Getting a Better Bound

- The bound at each node is obtained by solving an LP.
- But all costs are integer, and so the objective value of each integer solution is integer. So, the best integer solution has an integer objective value.
- If the best integer valued solution for a node is at most 42.66, then we know the best bound is at most 42.
- Other bounds can also be rounded down.

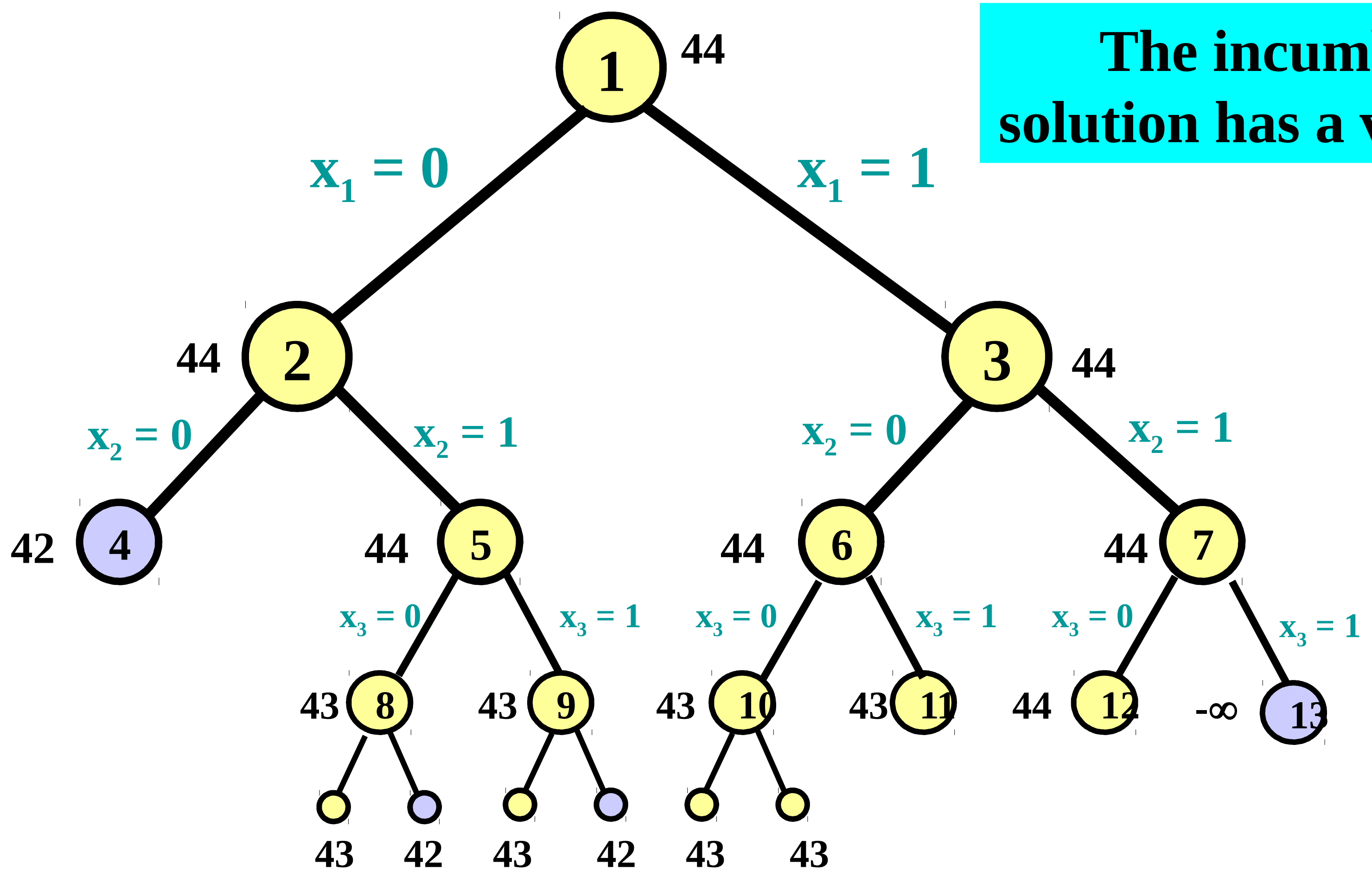
Branch and Bound



The incumbent solution has value 42

Branch and Bound

The incumbent solution has a value 42

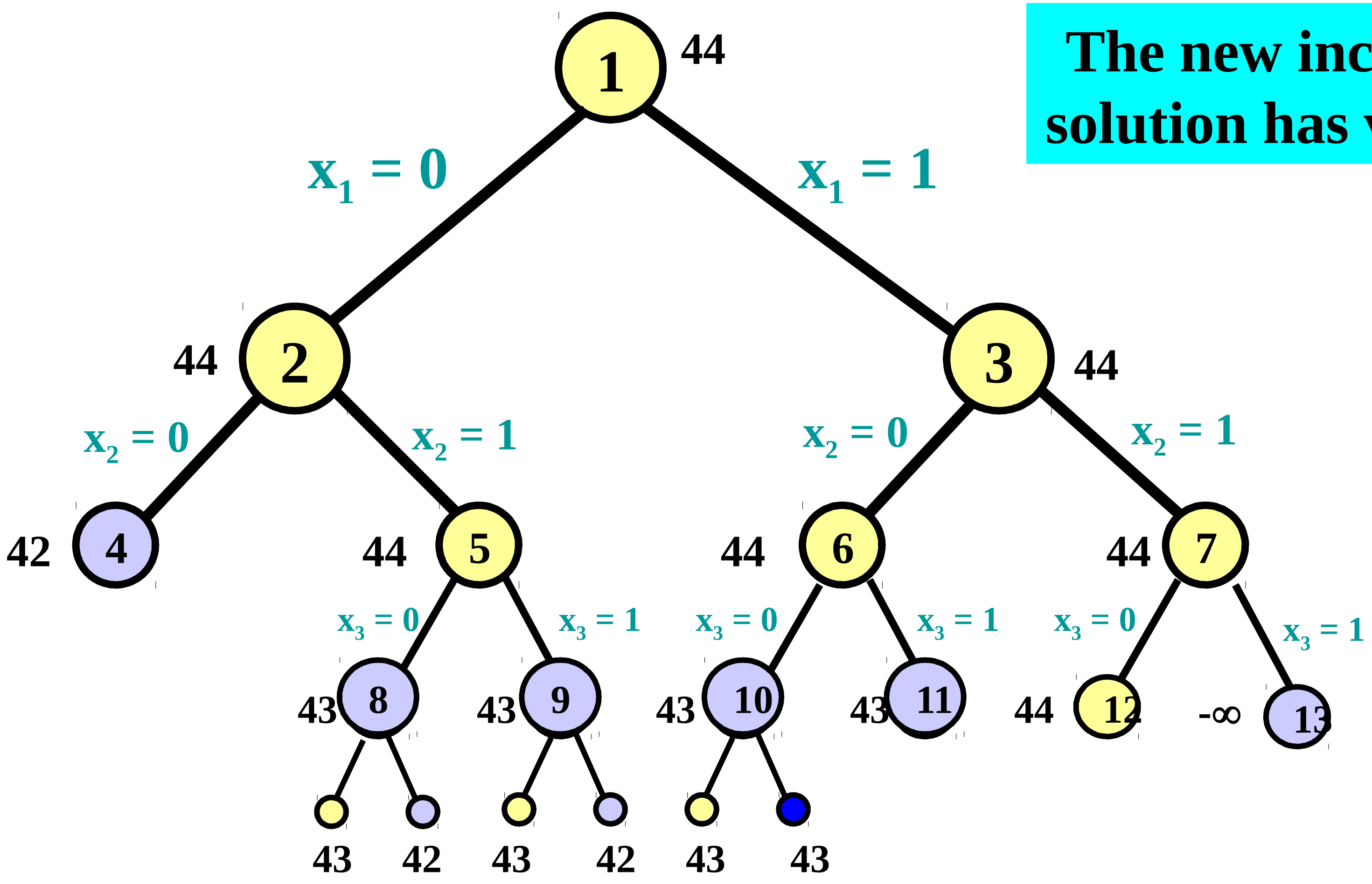


We found a new incumbent solution!

$x_1 = 1, x_2 = x_3 = 0, x_4 = 1, x_5 = 0, x_6 = 1 \quad z = 43$

Branch and Bound

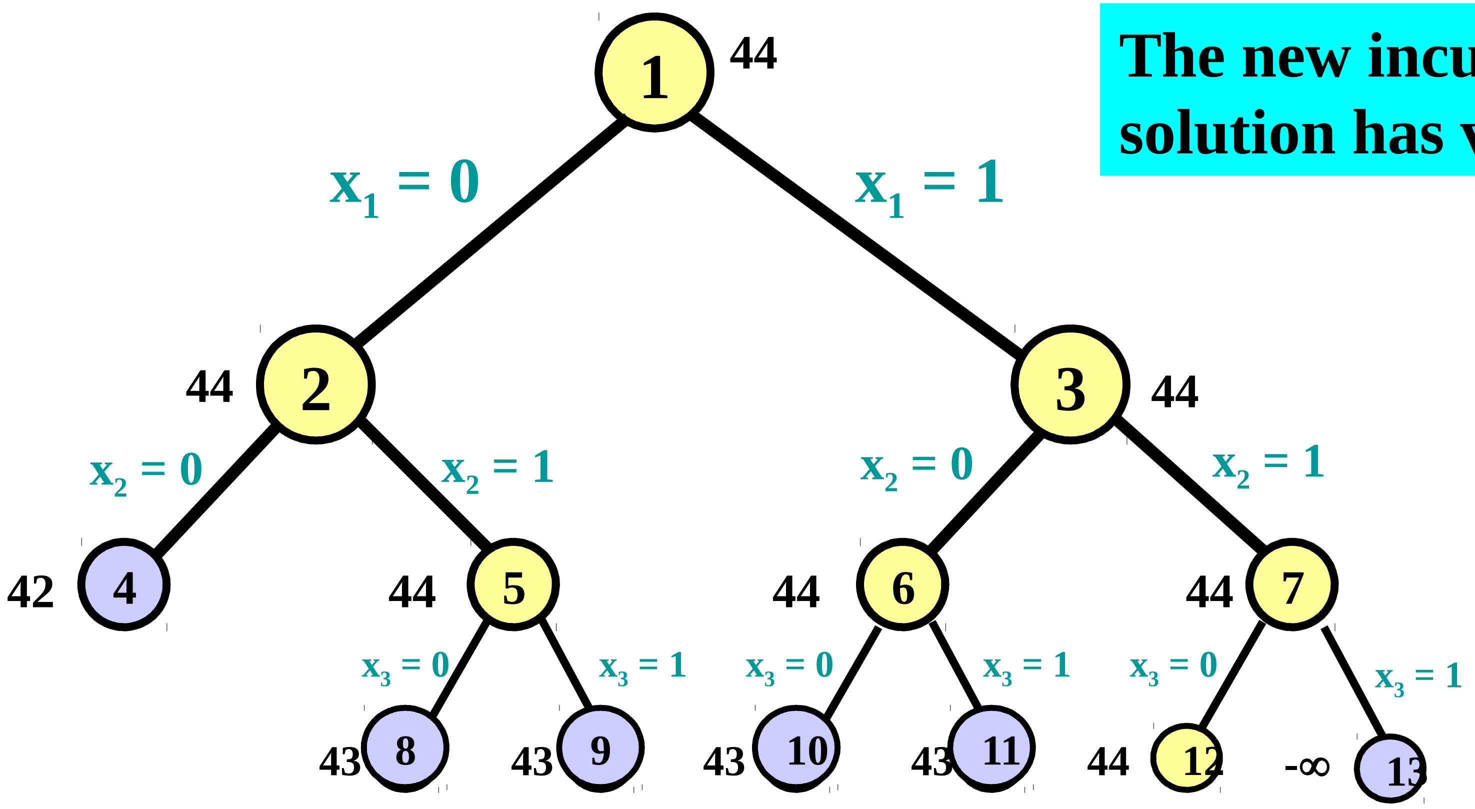
The new incumbent solution has value 43



We found a new incumbent solution!

$x_1 = 1, x_2 = x_3 = 0, x_4 = 1, x_5 = 0, x_6 = 1 \quad z = 43$

Branch and Bound

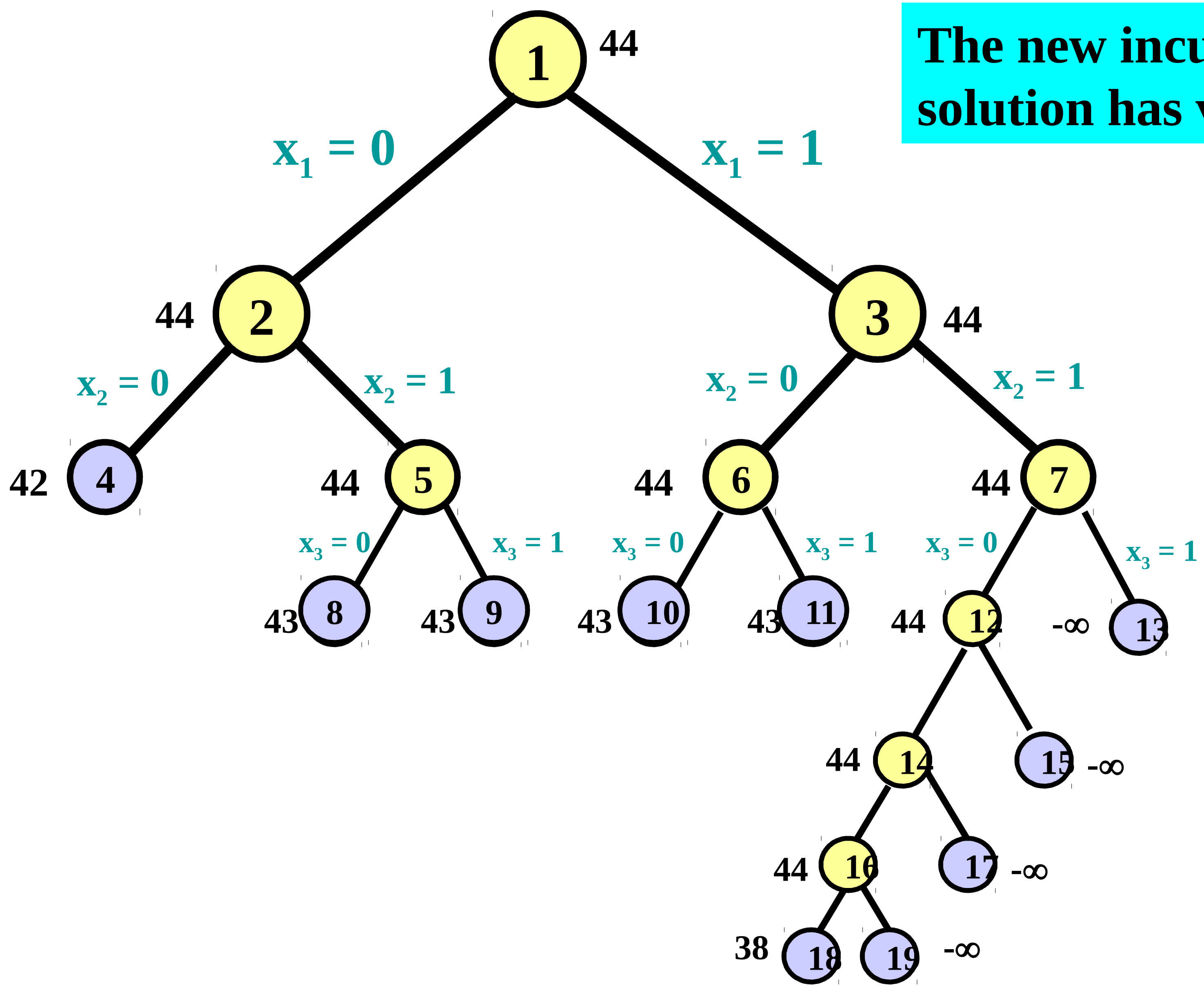


The new incumbent solution has value 43

If we had found this incumbent solution earlier, we could have saved some searching.

Finishing Up

The new incumbent solution has value 43



Key Observations

- Branch and Bound can speed up the search process
  - Only 25 nodes (linear programs) were evaluated
  - Other nodes were pruned
- Obtaining a good incumbent earlier can be valuable
  - only 19 nodes would have been evaluated.
- Solve linear programs faster, because we start with an excellent or optimal solution
- Obtaining better bounds can be valuable.
  - We sometimes use properties that are obvious to us, such as the fact that integer solutions have integer solution values

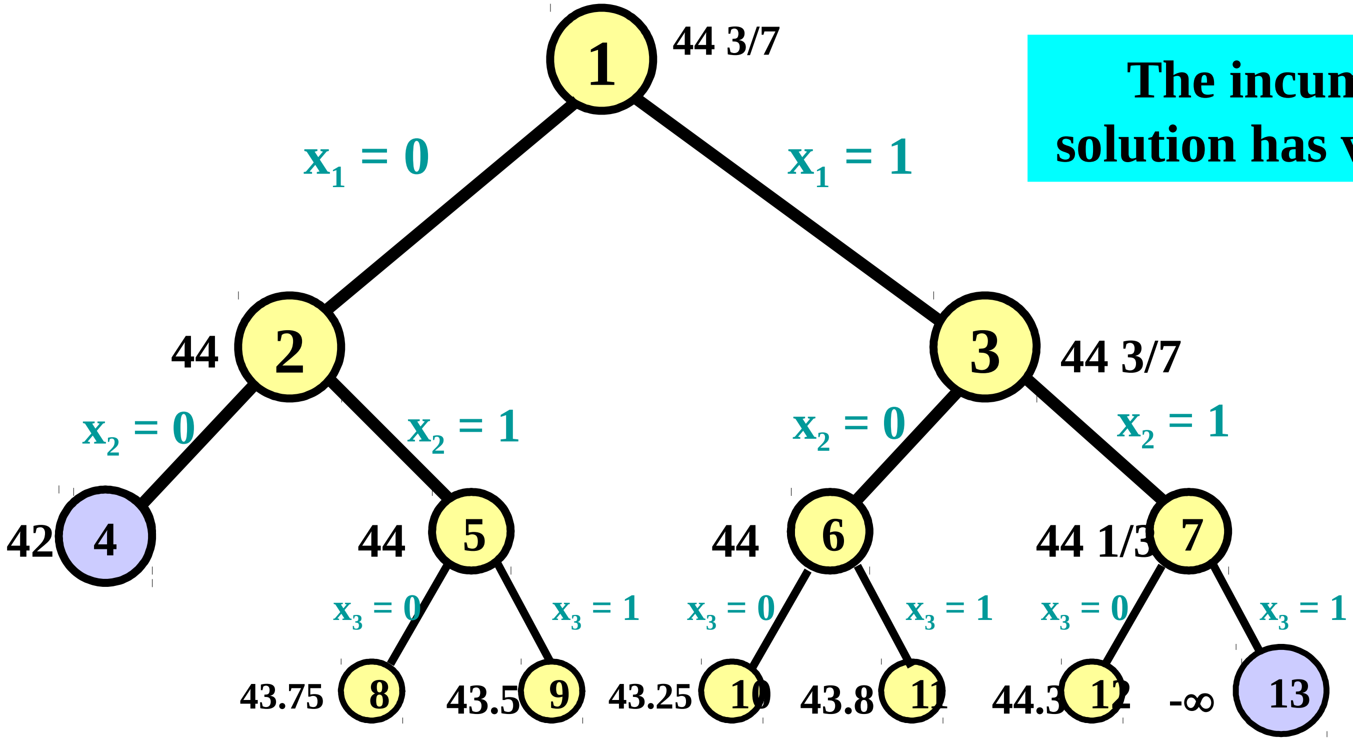
Branch and Bound

Notations:

- z^* = optimal integer solution value
- Subdivision: a node of the B&B Tree
- Incumbent: the best solution on hand
- z^I: value of the incumbent
- z^{LP}:   value of the LP relaxation of the current node
- Children of a node: the two problems created for a node, e.g., by saying $x_j = 1$ or $x_j = 0$.
- LIST: the collection of active (not fathomed) nodes, with no active children.

NOTE: $z^I \leq z^*$

Illustrating the Definitions

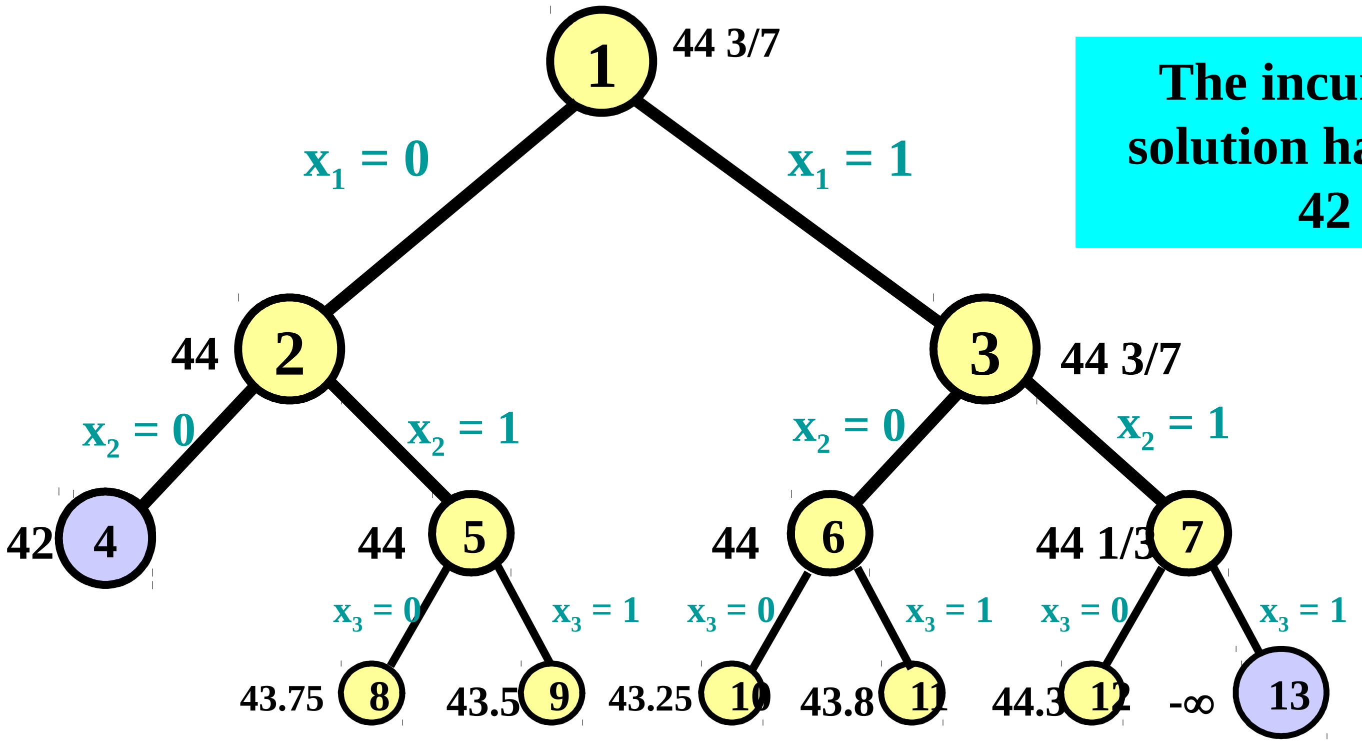


The incumbent solution has value 42

z* = 43 = optimal integer solution value. (We found it later in the search)

Incumbent is 0 0 1 0 1 1 $z^I = 42$.
It is the optimal solution for the subdivision 4.
The z^{LP} values for each subdivision are next to the nodes.

Illustrating the Definitions



The incumbent solution has value 42

The children of node (subdivision) 1 are nodes 2 and 3.
The children of node 3 are nodes 6 and 7.

LIST = { 8, 9, 10, 11, 12 } = unpruned nodes with no active children

Branch and Bound Algorithm

INITIALIZE LIST = {original problem}

Incumbent: = ∅

$z^I = -\infty$

SELECT:

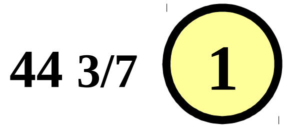
If LIST = ∅, then the Incumbent is optimal if it exists, and the problem is infeasible if no incumbent exists;

else, let S be a node (subdivision) from LIST.

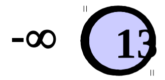
Let x^{LP} be the optimal solution to S

Let z^{LP} = its objective value

e.g., S = {1}



e.g., S = {13}



CASE 1. $z^{LP} = -\infty$ (the LP is infeasible)

Remove S from LIST (prune it)

Return to SELECT

Branch and Bound Algorithm

INITIALIZE

SELECT:

If LIST = ∅, then the Incumbent is optimal (if it exists), and the problem is infeasible if no incumbent exists;

else, let S be a node from LIST.

Let x^{LP} be the optimal solution to S

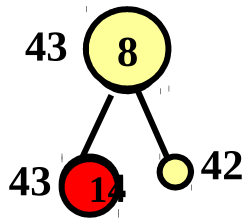
Let z^{LP} = its objective value

CASE 2. $-\infty < z^{LP} \le z^I$.

That is, the LP is dominated by the incumbent.

Then remove S from LIST (fathom it)

Return to SELECT



e.g., the incumbent has value 43, and node 14 is selected. $z^{LP} = 43$.

Branch and Bound Algorithm

INITIALIZE

SELECT:

If LIST = ∅, then the Incumbent is optimal (if it exists), and the problem is infeasible if no incumbent exists;

else, let S be a subdivision from LIST.

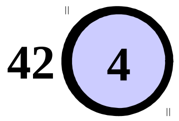
Let x^{LP} be the optimal solution to S

Let z^{LP} = its objective value

CASE 3. $z^I < z^{LP}$ and x^{LP} is integral.

That is, the LP solution is integral and dominates the incumbent.

e.g., node 4 was selected, and the solution to the LP was integer-valued.



Then Incumbent := x^{LP};

$z^I := z^{LP}$

Remove S from LIST (fathomed by integrality)

Return to SELECT

Branch and Bound Algorithm

INITIALIZE

SELECT:

If LIST = ⊘, then the Incumbent is optimal (if it exists), and the problem is infeasible if no incumbent exists;

else, let S be a subdivision from LIST.

Let x^{LP} be the optimal solution to S

Let z^{LP} = its objective value

e.g., select node 3.

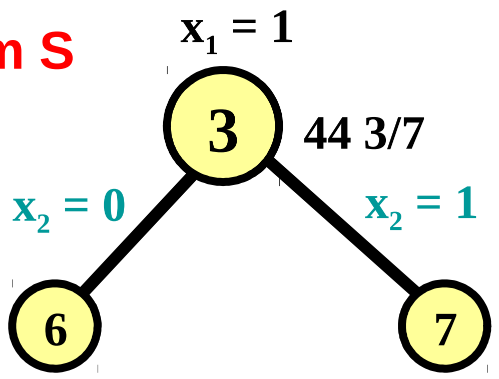
CASE 4. $z^I < z^{LP}$ and x^{LP} is not integral.

There is not enough information to fathom S

Remove S from LIST

Add the children of S to LIST

Return to SELECT



List := List – 3 + {6,7}

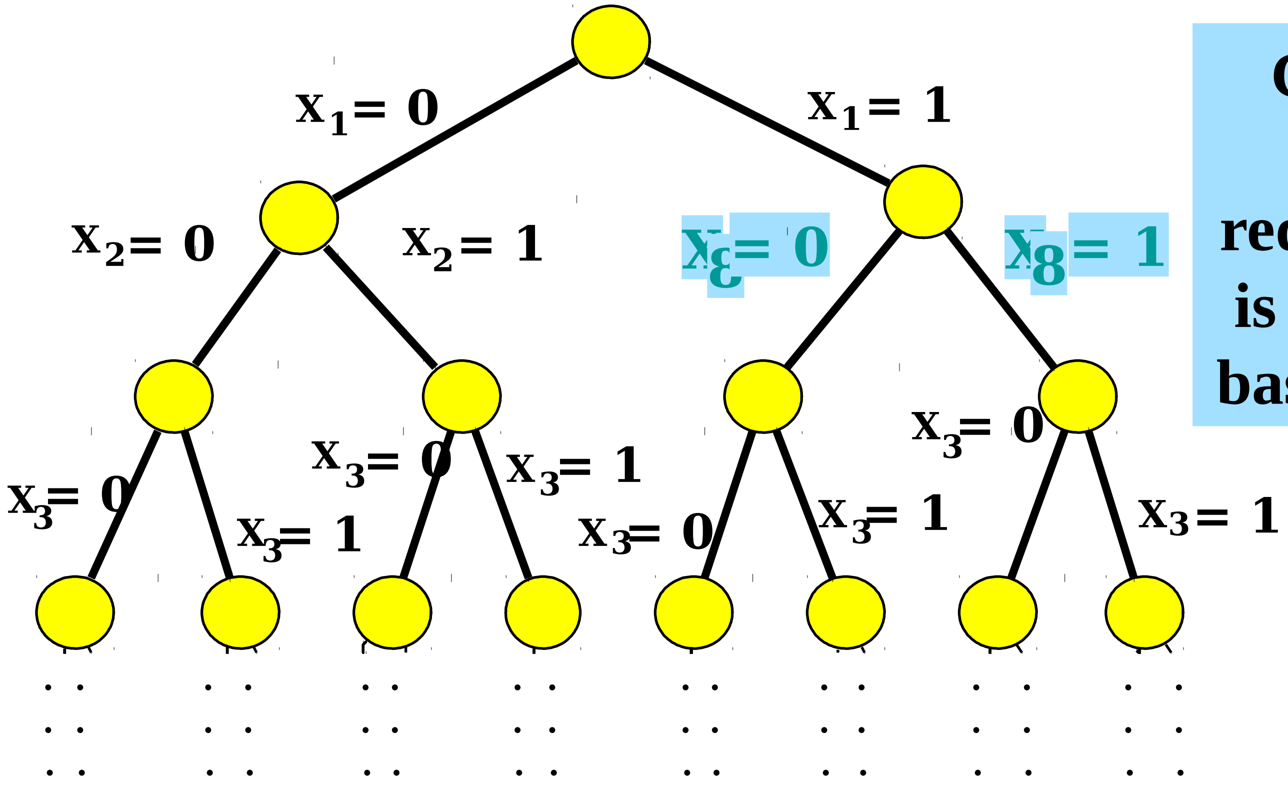
Possible Selection Rules

- Rule of Thumb 1: Don’t let LIST get too big (the solutions must be stored). So, prefer nodes that are further down in the tree.

- Rule of Thumb 2: Pick a node of LIST that is likely to lead to an improved incumbent. Sometimes special heuristics are used to come up with a good incumbent.

Branching

One does not have to have the B&B tree be symmetric, and one does not select subtrees by considering variables in order.



Choosing how to branch so as to reduce running time is largely “art” and based on experience.

Possible Branching Rules

- Branching: determining children for a node. There are many choices (Rules of Thumb).
- Rule 1: if it appears clear that $x_j = 1$ in an optimal solution, it is often good to branch on $x_j = 0$ vs $x_j = 1$.
  - The hope is that a subdivision with $x_j = 0$ can be pruned.
- Rule 2: branching on important variables is worthwhile

Possible Bounding Techniques

- We use the bound obtained by dropping the integrality constraints (LP relaxation). There are other choices.
- Key tradeoff for bounds: time to obtain a bound vs quality of the bound.
- If one can obtain a bound much quicker, sometimes we would be willing to get a bound that is worse
- It usually is worthwhile to get a bound that is better, so long as it does not take too long.

What if the Variables are General Integer Variables?

- One can choose children as follows:
  - child 1: $x_1 \leq 3$ (or $x_j \leq k$)
  - child 2 $x_1 \geq 4$ (or $x_j \geq k+1$)

- How would one choose the variable j and the value k
  - A common choice would be to take a fractional value from x^{LP}. e.g., if $x_7 = 5.62$, then we may branch on $x_7 \leq 5$ and $x_7 \geq 6$.
  - Other choices are also possible.

Summary

- Branch and Bound is the standard way of solving IPs to optimality.
- There is art to making it work well in practice.
- Much of the art is built into state-of-the-art solvers such as CPLEX.