Topics in Computational Sustainability: Integer Linear Programming

Linear Programming

Given integers a_{ij}, b_i, and c_j, find real numbers x_j that satisfy:

$$\begin{array}{rrcl} \min & c^\top x & & \\ \text{s.t.} & Ax & \geq & b \\ & x & \geq & 0 \end{array}$$

$$\begin{array}{rrcll} \min & \sum_{j=1}^{n} c_j x_j & & & \\ \text{s.t.} & \sum_{j=1}^{n} a_{ij} x_j & \geq & b_i & 1 \leq i \leq m \\ & x_j & \geq & 0 & 1 \leq j \leq n \end{array}$$

Linear. No x^2, xy, $\arccos(x)$, $x(1-x)$, etc.

Simplex algorithm. [Dantzig 1947] Can solve LP in practice.

Ellipsoid algorithm. [Khachiyan 1979] Can solve LP in poly-time.

Interior point algorithms. [Karmarkar 1984, Renegar 1988, ...]

Can solve LP both in poly-time and in practice.

Violates Divisibility Assumption of LP

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

When the divisibility assumption is violated then we need to apply "Integer Linear Programming (ILP)!!!"

Integer Linear Programming

Given integers a_{ij}, b_i, and c_j, find integers x_j that satisfy:

$$\begin{array}{lrcl} \min & c^{\mathsf{T}} x & & \\ \text{s.t.} & Ax & \geq & b \\ & x & \geq & 0 \\ & x & & \text{integral} \end{array}$$

$$\begin{array}{lrcll} \min & \sum_{j=1}^{n} c_j x_j & & & \\ \text{s.t.} & \sum_{j=1}^{n} a_{ij} x_j & \geq & b_i & 1 \leq i \leq m \\ & x_j & \geq & 0 & 1 \leq j \leq n \\ & x_j & & \text{integral} & 1 \leq j \leq n \end{array}$$

Why Integer Linear Programming?

- Advantages of restricting the decision 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 (ILP)

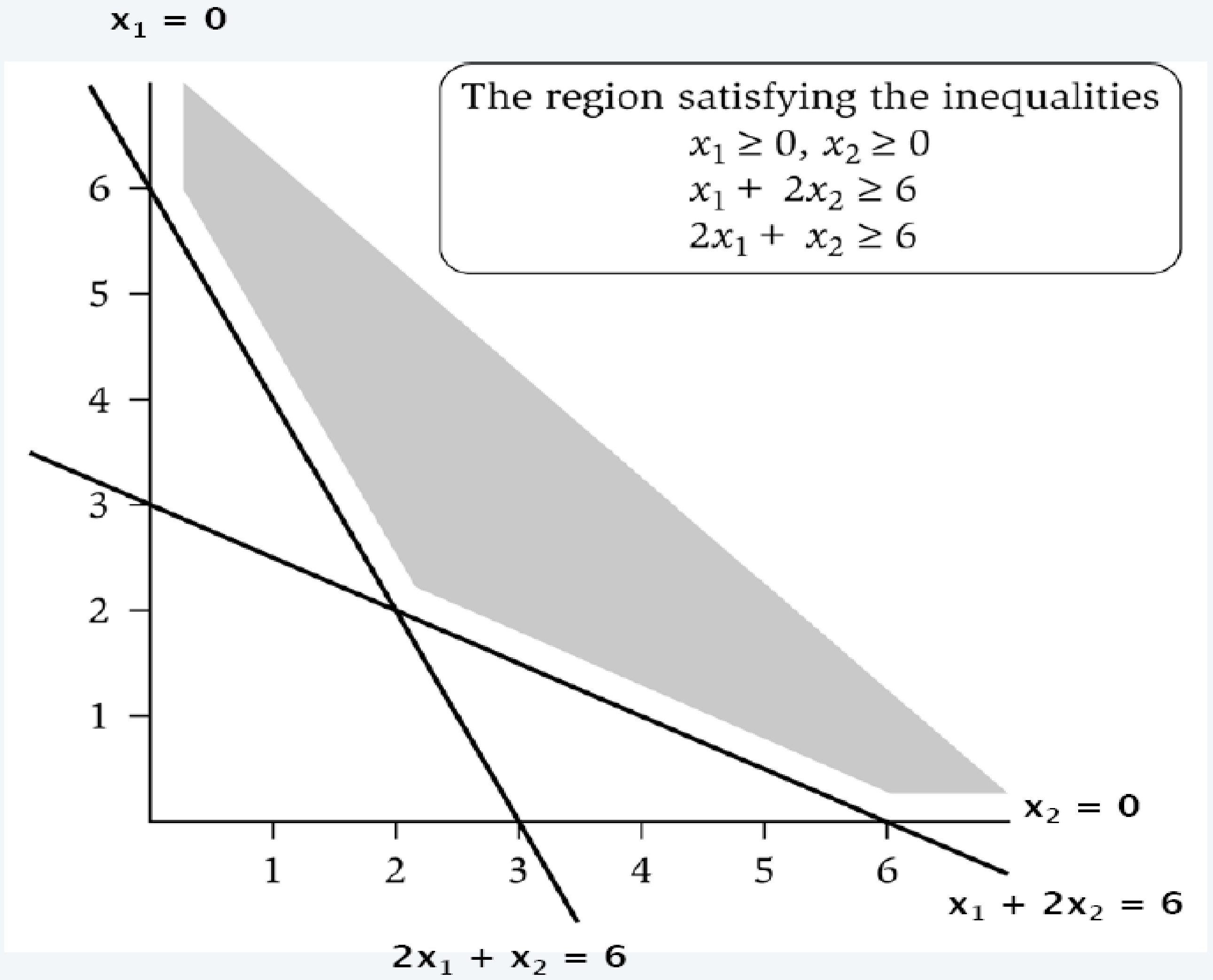
- When are "non-integer" solutions okay?
  - Solution is naturally divisible
    - e.g., Pounds, Hours
  - Solution represents a rate
    - e.g., Units Per Week
- When is rounding okay?
  - When numbers are large
    - e.g., Rounding 114.286 to 114 is probably okay.
- When is rounding not okay?
  - When 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

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

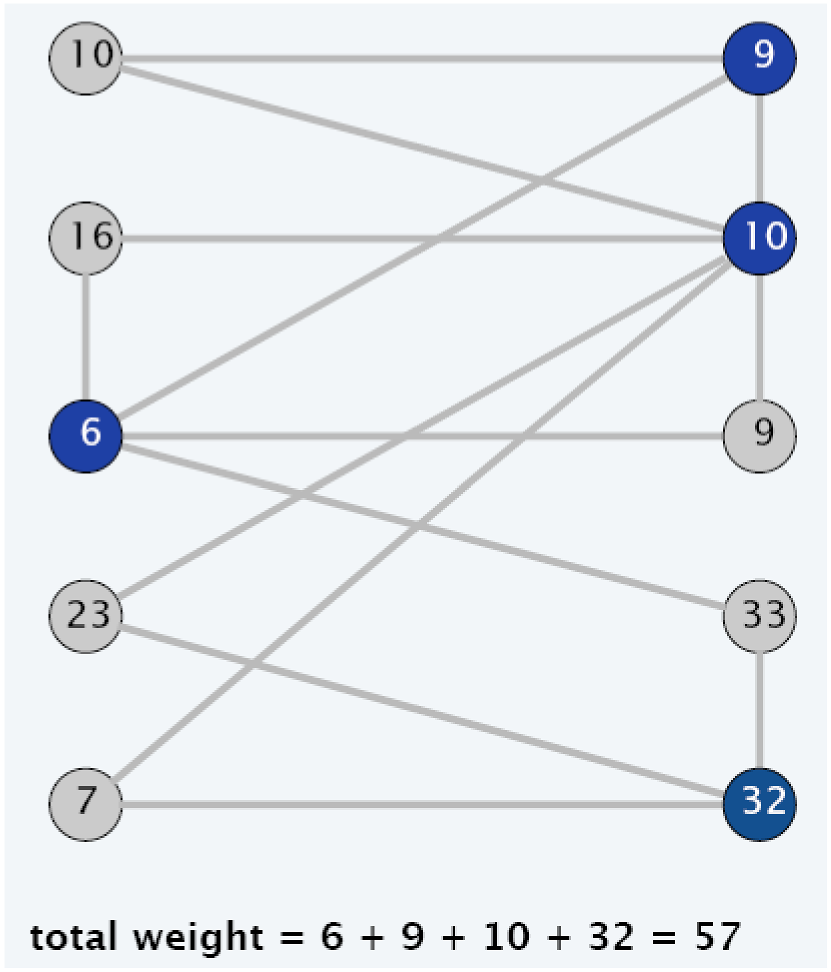
LP feasible region

LP geometry in 2D.

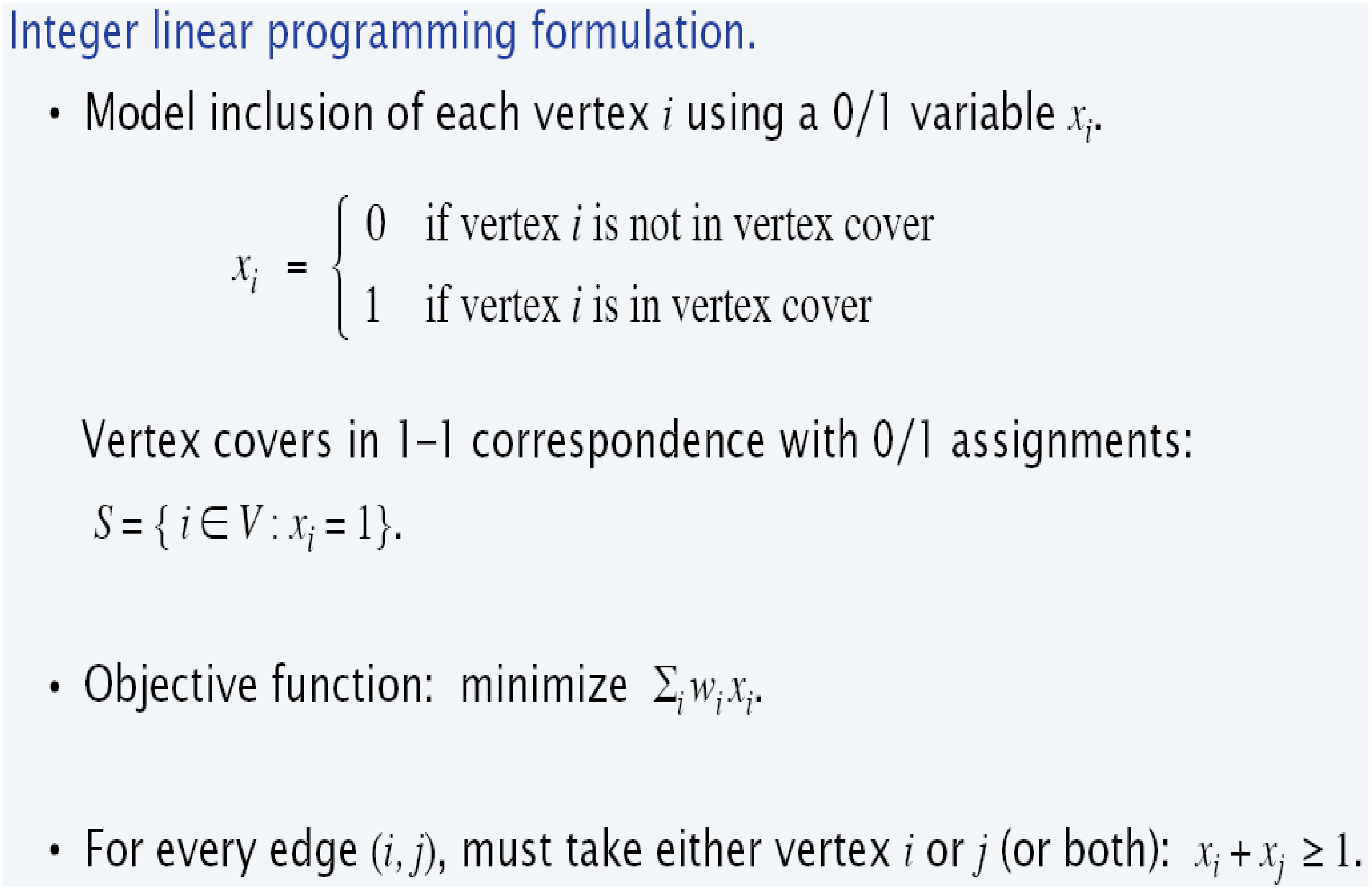


Weighted Vertex Cover: ILP Formulation

Given a graph $G = (V, E)$ with vertex weights $w_i \geq 0$, find a min-weight subset of vertices $S \subseteq V$ such that every edge is incident to at least one vertex in S.



Weighted Vertex Cover: ILP Formulation



Integer linear programming formulation.

- Model inclusion of each vertex i using a 0/1 variable x_i.

$$x_i = \begin{cases} 0 & \text{if vertex } i \text{ is not in vertex cover} \\ 1 & \text{if vertex } i \text{ is in vertex cover} \end{cases}$$

Vertex covers in 1–1 correspondence with 0/1 assignments:

$S = \{ i \in V : x_i = 1\}$.

- Objective function: minimize $\Sigma_i w_i x_i$.

- For every edge (i, j), must take either vertex i or j (or both): $x_i + x_j \geq 1$.

Weighted Vertex Cover: ILP Formulation

$$
\begin{array}{llrcll}
(ILP) & \min & \sum_{i \in V} w_i \, x_i & & & \\
& \text{s.t.} & x_i + x_j & \geq & 1 & (i,j) \in E \\
& & x_i & \in & \{0,\ 1\} & i \in V
\end{array}
$$

Observation. Vertex cover formulation proves that INTEGER-PROGRAMMING is an **NP**-hard optimization problem.

Observation. If x^* is optimal solution to *ILP*, then $S = \{ i \in V : x_i^* = 1 \}$ is a min-weight vertex cover.

Weighted vertex cover: LP relaxation

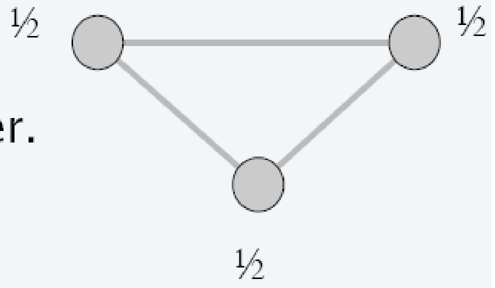
Linear programming relaxation.

$$
\begin{array}{llrcll}
(LP) & \min & \displaystyle\sum_{i \in V} w_i \, x_i & & & \\
& \text{s.t.} & x_i + x_j & \geq & 1 & (i,j) \in E \\
& & x_i & \geq & 0 & i \in V
\end{array}
$$

Observation. Optimal value of *LP* is ≤ optimal value of *ILP*.

Pf. *LP* has fewer constraints.

Note. *LP* solution x^* may not correspond to a vertex cover.
(even if all weights are 1)



Q. How can solving *LP* help us find a low-weight vertex cover?

A. Solve *LP* and round fractional values in x^*.

Weighted vertex cover: LP rounding algorithm

Lemma. If x^* is optimal solution to LP, then $S = \{ i \in V : x_i^* \geq ½ \}$ is a vertex cover whose weight is at most twice the min possible weight.

Pf. [S is a vertex cover]

- Consider an edge $(i, j) \in E$.
- Since $x_i^* + x_j^* \geq 1$, either $x_i^* \geq ½$ or $x_j^* \geq ½$ (or both) $\Rightarrow$ (i, j) covered.

Pf. [S has desired weight]

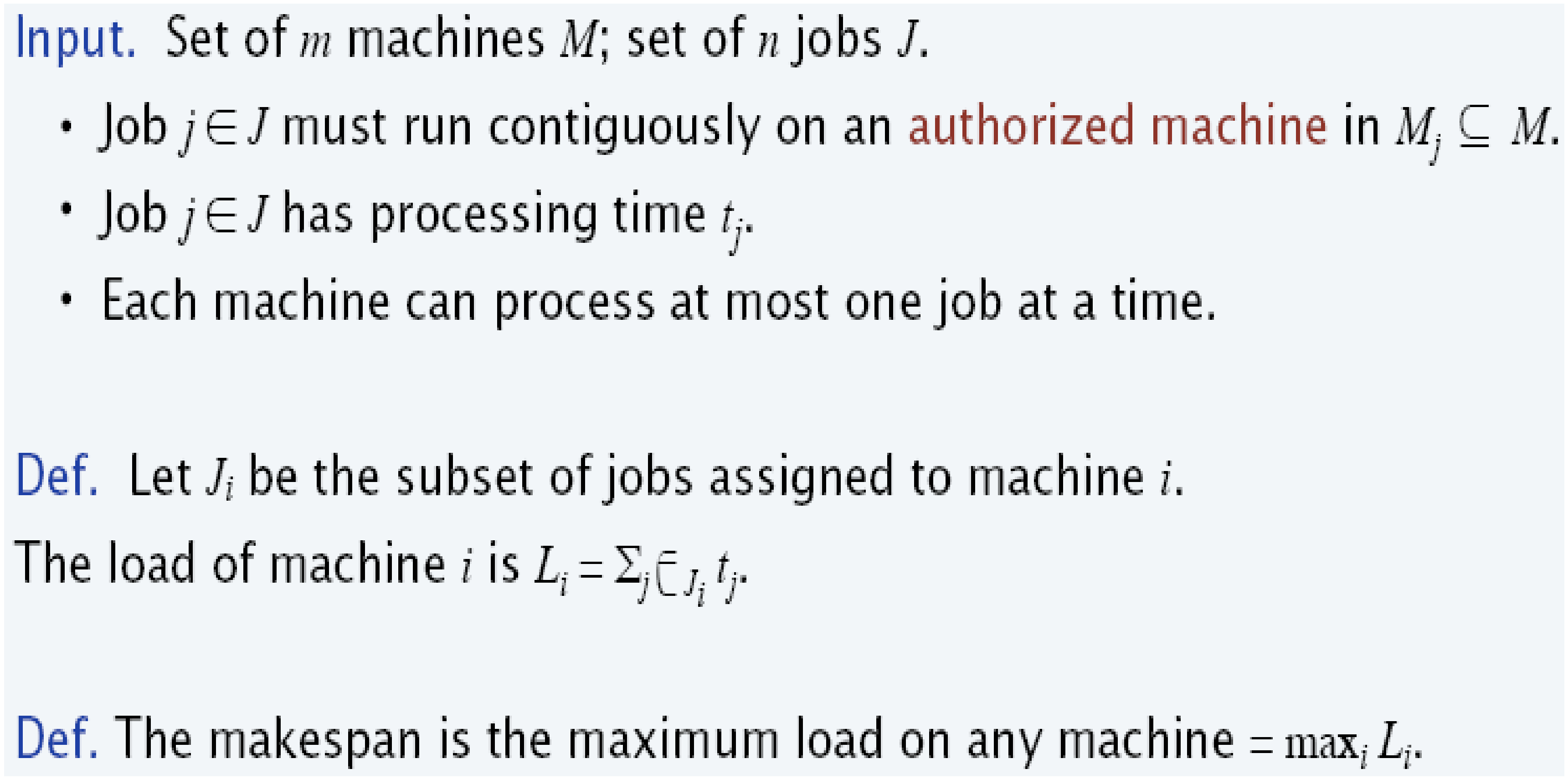
- Let S^* be optimal vertex cover. Then

$$\sum_{i \in S^*} w_i \underset{\substack{\uparrow \\ \text{LP is a relaxation}}}{\geq} \sum_{i \in S} w_i \, x_i^* \underset{\substack{\uparrow \\ x_i^* \ \geq \ ½}}{\geq} \frac{1}{2} \sum_{i \in S} w_i$$

Theorem. The rounding algorithm is a 2-approximation algorithm.
Pf. Lemma + fact that LP can be solved in poly-time.

Generalized Load Balancing

- Assign each job to an authorized machine to minimize the makespan.



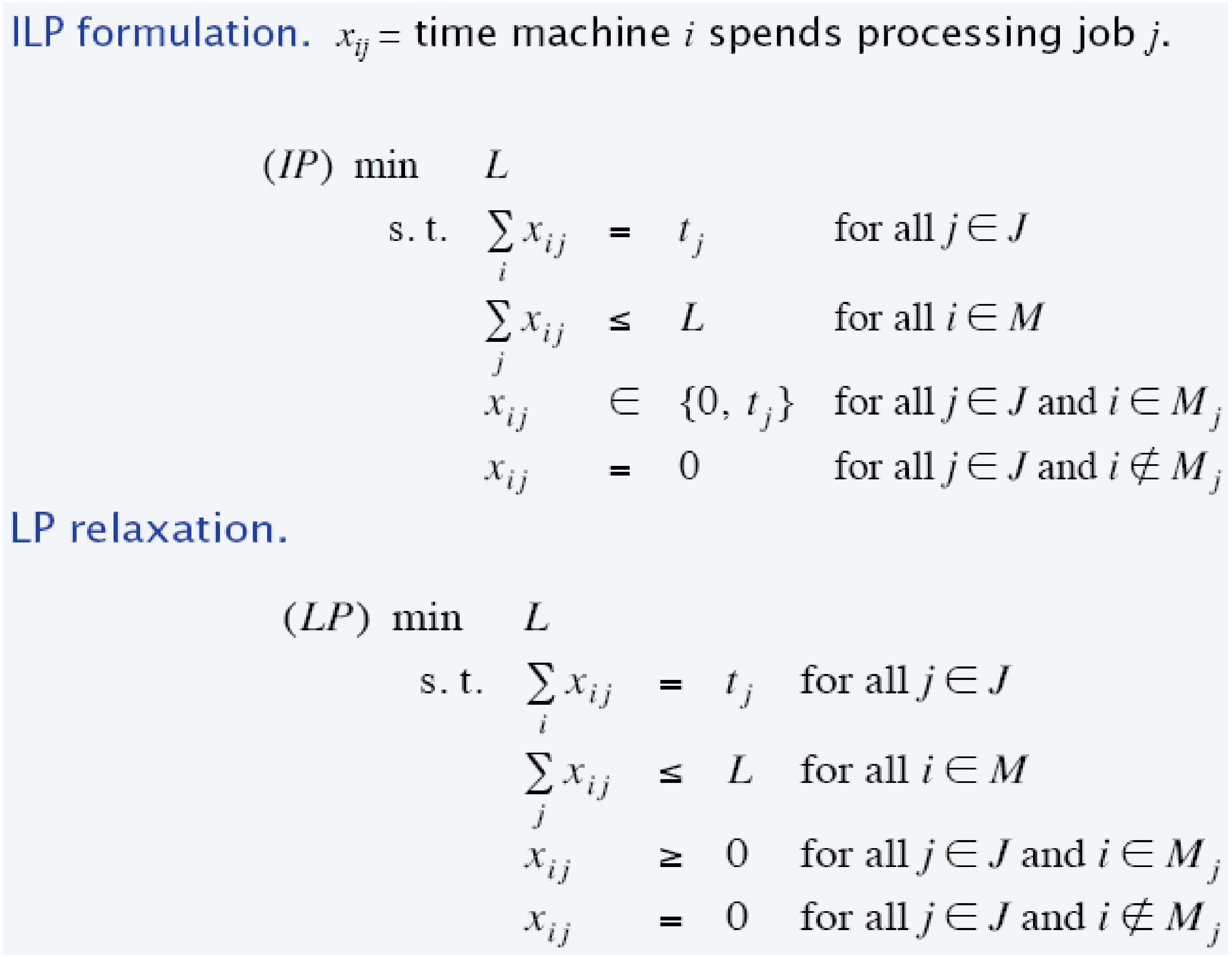
Input. Set of m machines M; set of n jobs J.

- Job $j \in J$ must run contiguously on an authorized machine in $M_j \subseteq M$.
- Job $j \in J$ has processing time t_j.
- Each machine can process at most one job at a time.

Def. Let J_i be the subset of jobs assigned to machine i.
The load of machine i is $L_i = \Sigma_{j \in J_i} t_j$.

Def. The makespan is the maximum load on any machine $= \max_i L_i$.

Generalized Load Balancing



ILP formulation. x_{ij} = time machine i spends processing job j.

$$
\begin{array}{llcll}
(IP) \ \min & L & & & \\
\text{s. t.} & \sum_i x_{ij} & = & t_j & \text{for all } j \in J \\
 & \sum_j x_{ij} & \le & L & \text{for all } i \in M \\
 & x_{ij} & \in & \{0, t_j\} & \text{for all } j \in J \text{ and } i \in M_j \\
 & x_{ij} & = & 0 & \text{for all } j \in J \text{ and } i \notin M_j
\end{array}
$$

LP relaxation.

$$
\begin{array}{llcll}
(LP) \ \min & L & & & \\
\text{s. t.} & \sum_i x_{ij} & = & t_j & \text{for all } j \in J \\
 & \sum_j x_{ij} & \le & L & \text{for all } i \in M \\
 & x_{ij} & \ge & 0 & \text{for all } j \in J \text{ and } i \in M_j \\
 & x_{ij} & = & 0 & \text{for all } j \in J \text{ and } i \notin M_j
\end{array}
$$

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 of Dollars)	5	7	4	3	4	6
Total Production (100 KW)	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 resource allocation 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?

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

What are Objective Function and Constraints?

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

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

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

- The previous constraints represent “ economic indivisibilities ” , either a project is selected, or not selected. It is not possible to select a fraction of a project.

- Similarly, integer variables can model the “logical requirements” (for example, 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 the 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 items 4 and 5.

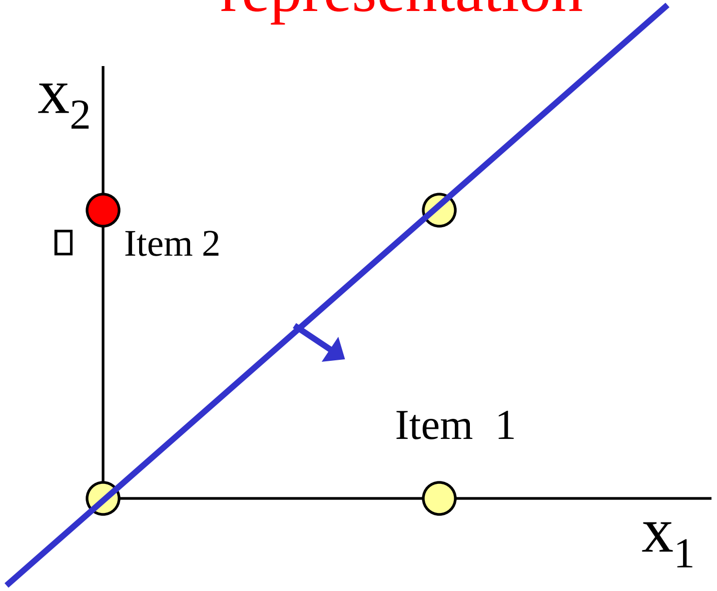
Formulating Constraints

- 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

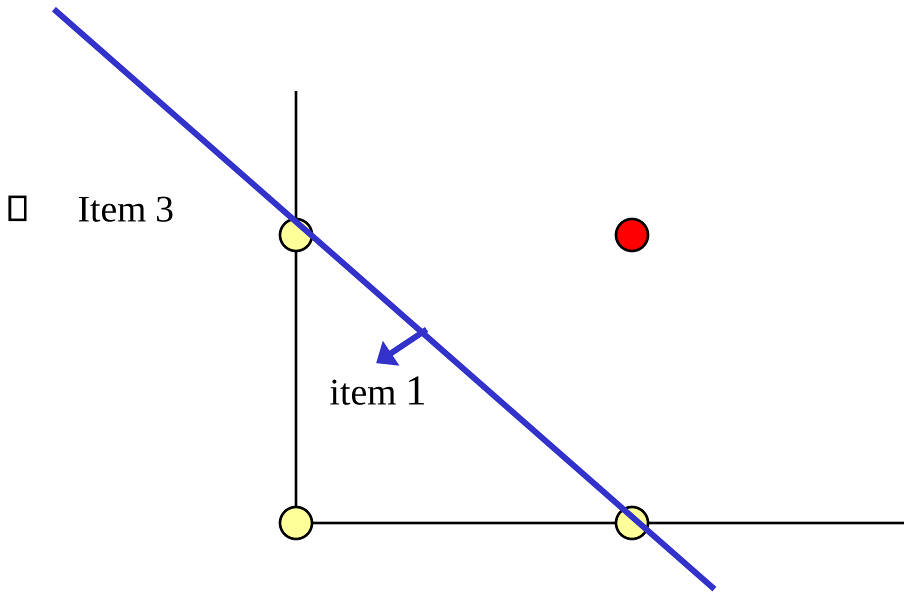


- The integer programming constraint:

$$x_1 \geq x_2$$

If item 1 is selected then item 3 is not selected

- A 2-dimensional representation

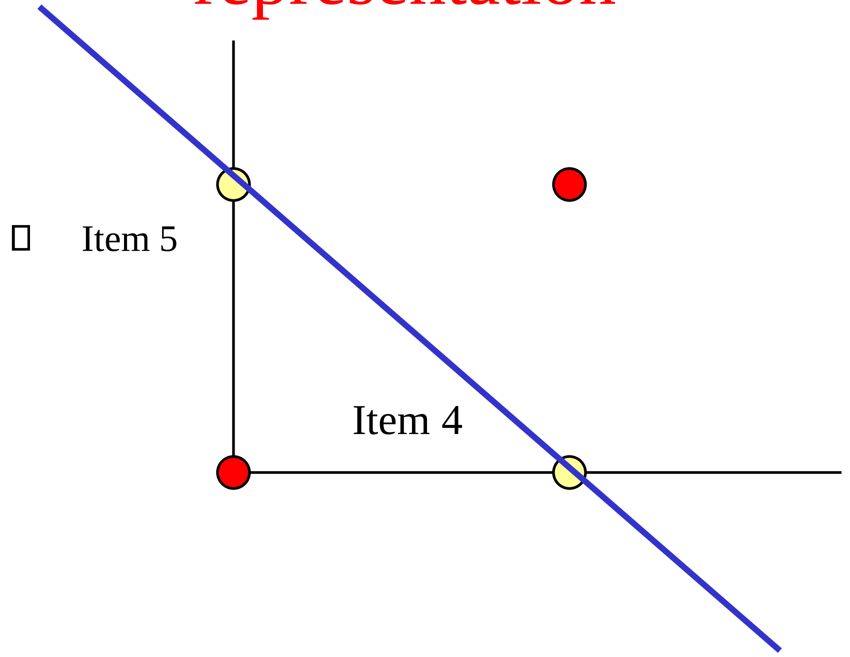


- The integer programming constraint:

$$x_1 + x_3 \leq 1$$

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

- A 2-dimensional representation



- The integer programming constraint:

$$x_4 + x_5 = 1$$

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.

  $x_1 + x_3 \leq 1 \quad \Leftrightarrow \quad x_1 \leq 1 - x_3$
- Either item 4 is selected or item 5 is selected, but not both items. $x_4 + x_5 = 1$

Modeling Fixed Charge Problems

If a product is produced, then 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

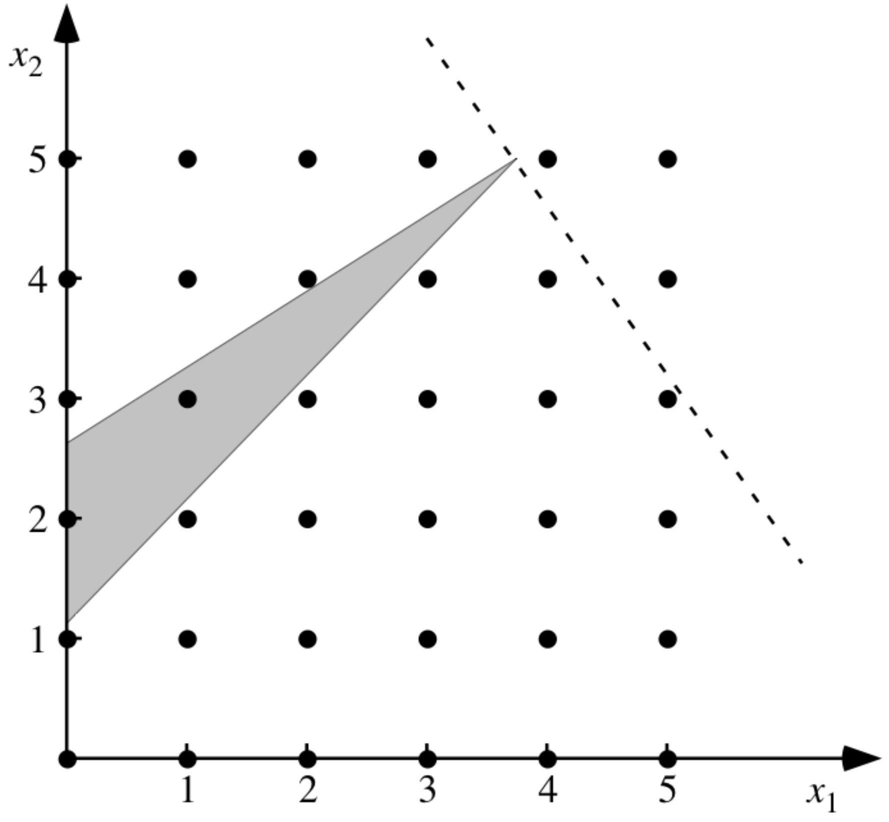
Objective function becomes → $C_1x + C_2y$

Additional Constraint → $x \leq My$ Where, M is a big number

Solving Integer Programs

The Challenges of Rounding

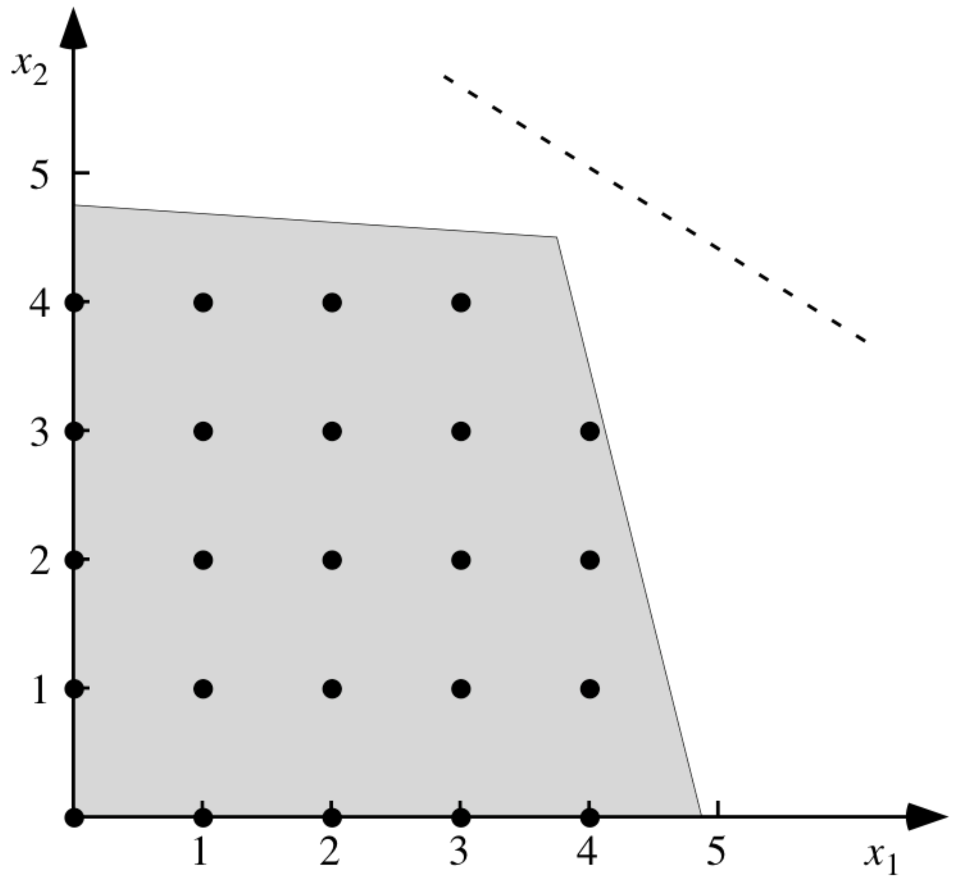
- Rounded Solution may not be feasible solution.
- Rounded solution may not be close to optimal solution.
- There can be *many* rounded solutions.
  - Consider a problem with 30 variables that are non-integer in LP-solution. How many are possible rounded solutions?



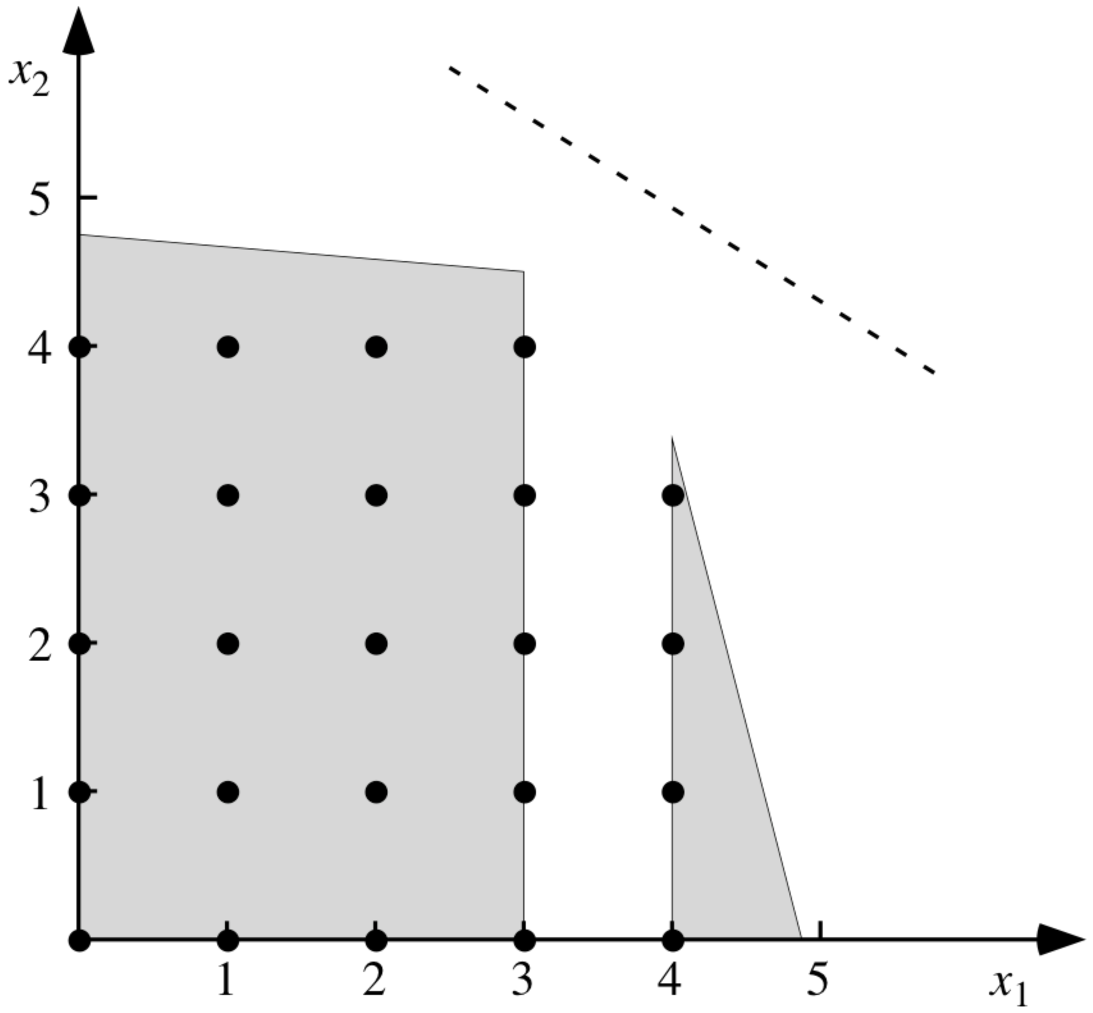
Overview of Techniques for Solving Integer Programs

- Enumeration Techniques
  - Complete Enumeration
    - List all "Solutions" and then Choose the Best Solution
  - 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 the Integer Programs by adding the 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?

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

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

$$\text{Subject to: } 5x_1 + 7x_2 + 4x_3 + 3x_4 + 4x_5 + 6x_6 \leq 14$$

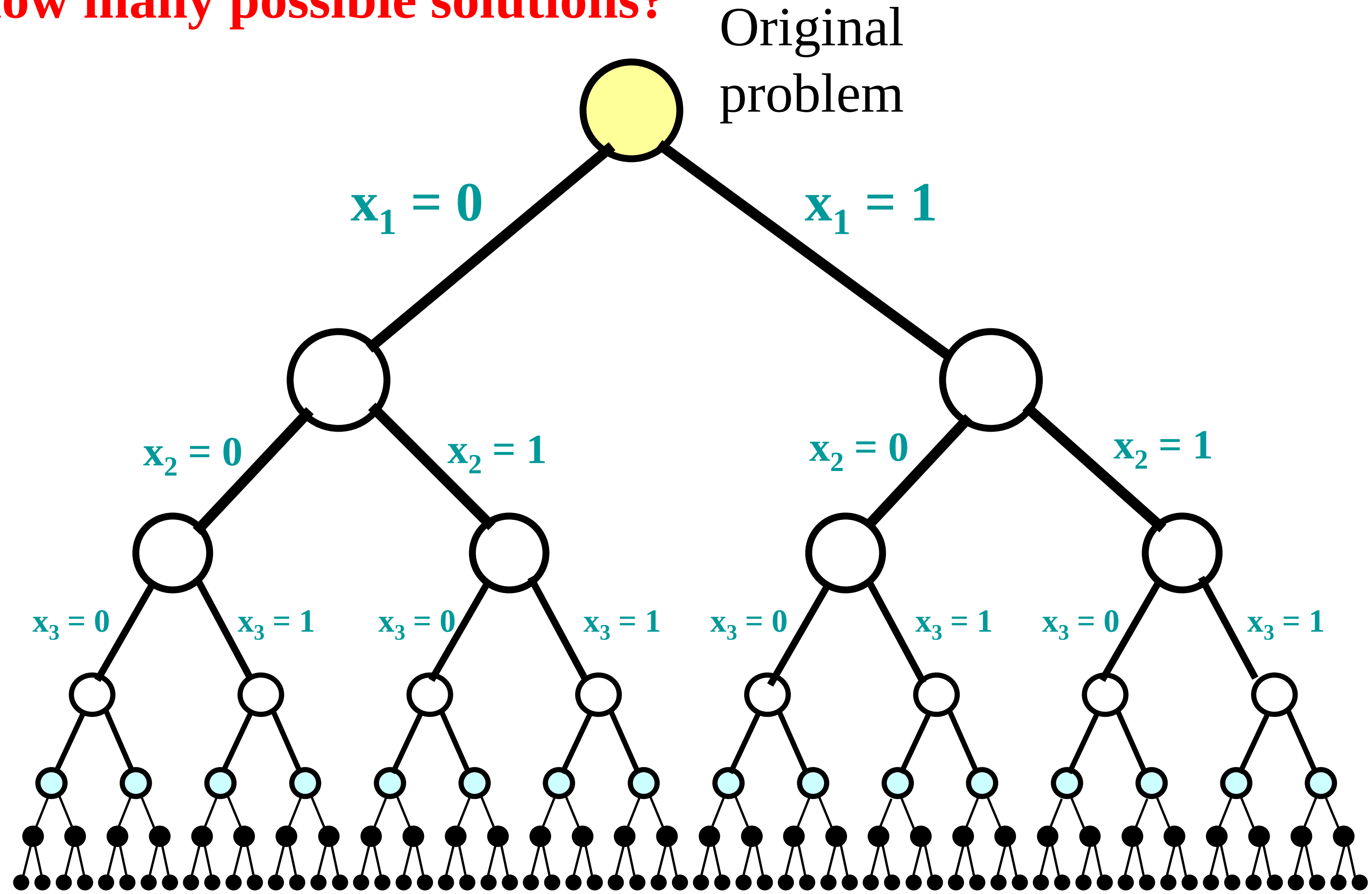
$$x_j\ \varepsilon\ \{0,1\} \text{ for each } j = 1 \text{ to } 6$$

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 these are 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

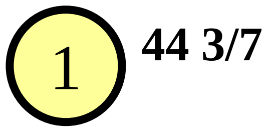
1. Solve the Linear Program (LP) at the node (by relaxing the integrality constraints i.e. LP Relaxation)
2. Eliminate (Prune) the subtree if the Solution is:

(a) Integer (there is no need to go further- why?) or

(b) The best solution in the subtree cannot be as good as the best available solution (the incumbent- how does that happen?) or

(c) Not the feasible solution (There is no feasible solution).

Branch and Bound



Node 1 is the Original LP Relaxation

Maximize $\{Z = 16x_1 + 22x_2 + 12x_3 + 8x_4 + 11x_5 + 19x_6\}$

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

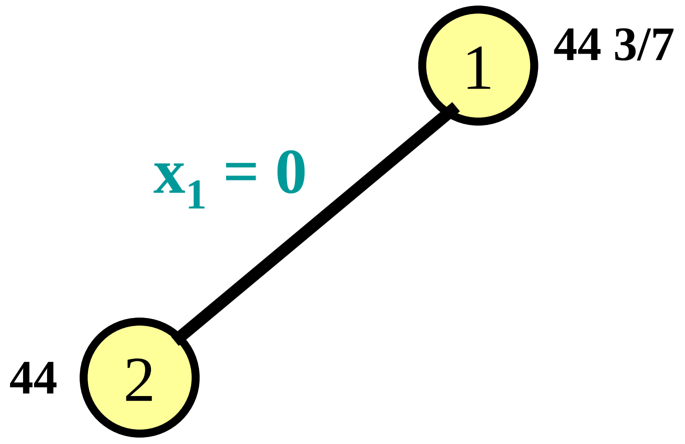
$0 \leq x_j \leq 1$ for $j = 1$ to 6

Solution at Node 1:

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

The IP cannot have the value higher than $Z = 44\ 3/7$.

Branch and Bound



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

maximize $\{Z = 16x_1 + 22x_2 + 12x_3 + 8x_4 + 11x_5 + 19x_6\}$

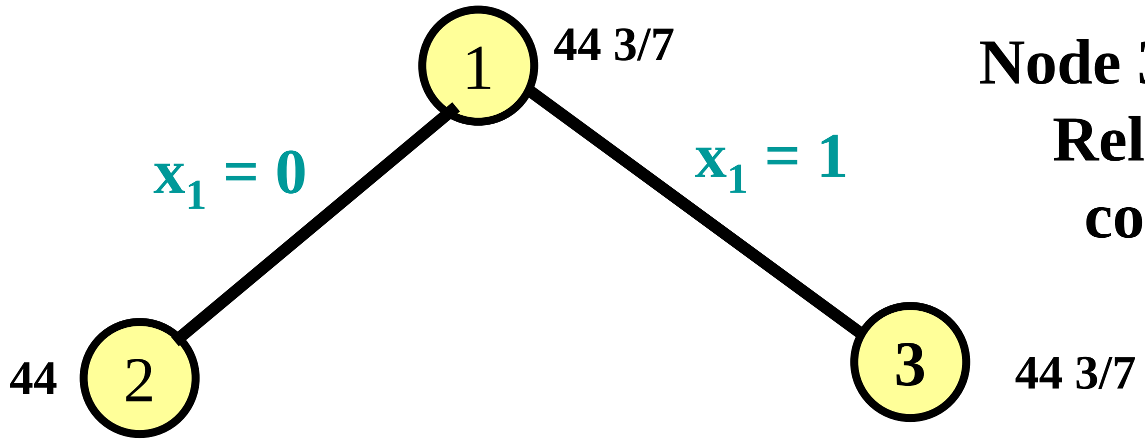
subject to $5x_1 + 7x_2 + 4x_3 + 3x_4 + 4x_5 + 6x_6 \leq 14$

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

Solution at Node 2:

$x_1 = 0$ $x_2 = 1$ $x_3 = 1/4$ $x_4 = x_5 = 0$ $x_6 = 1$ $Z = 44$

Branch and Bound



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

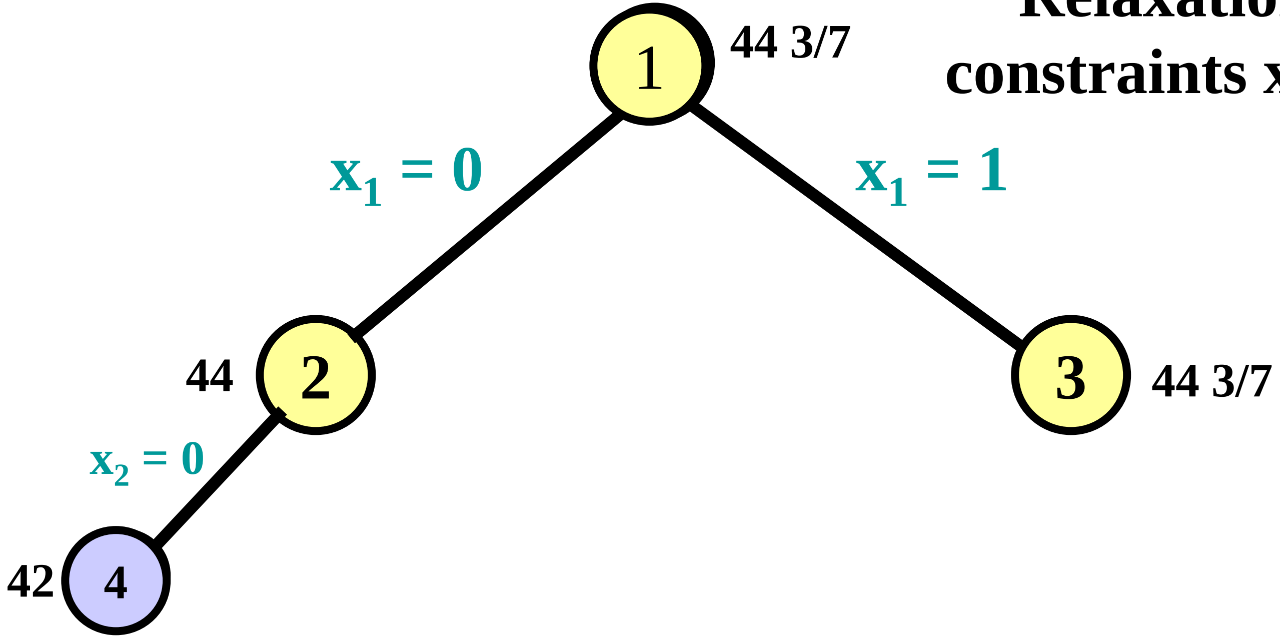
Note: The solution at Node 1 was

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

Note: It was the Best Solution at Node 1 with no constraint on x_1. It is also the Solution for Node 3. (If we add a constraint ($x_1=0$), and the old optimal solution is feasible one, 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: $x_1=0$ $x_2=0$ $x_3=1$ $x_4=0$ $x_5=1$ $x_6=1$ $Z=42$

We got our first ***Incumbent*** Solution!

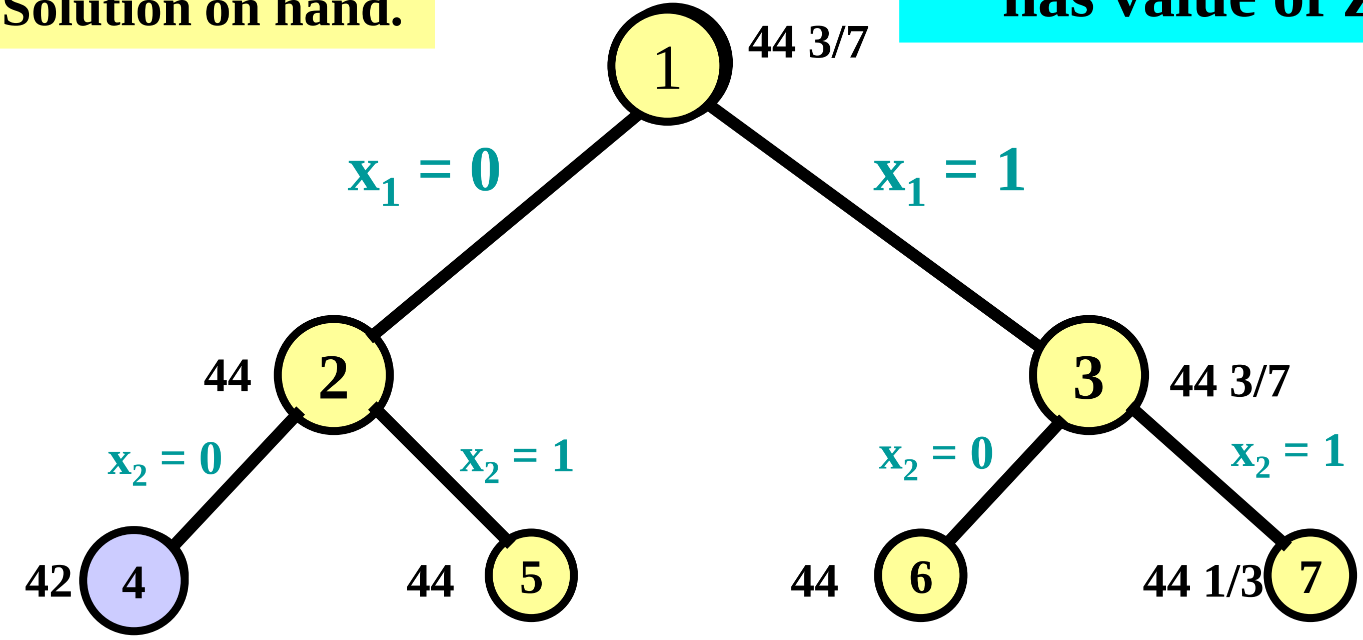
No further searching from Node 4 because there cannot be a better Integer Solution.

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

Branch and Bound

The Incumbent is the Best Solution on hand.

The Incumbent Solution has value of Z = 42



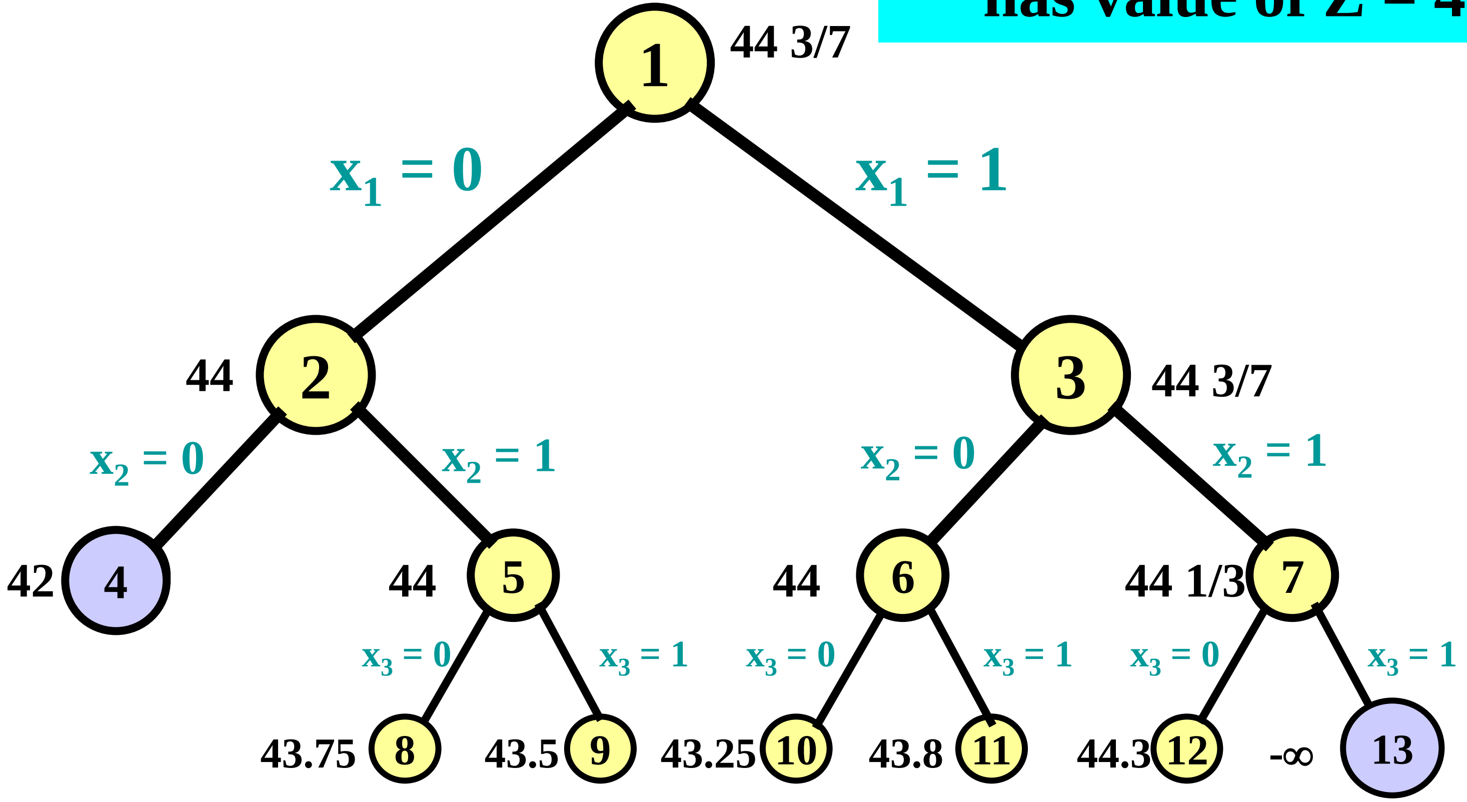
We next solved the LPs associated with Nodes 5, 6, and 7.

It is observed that 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 of Z = 42

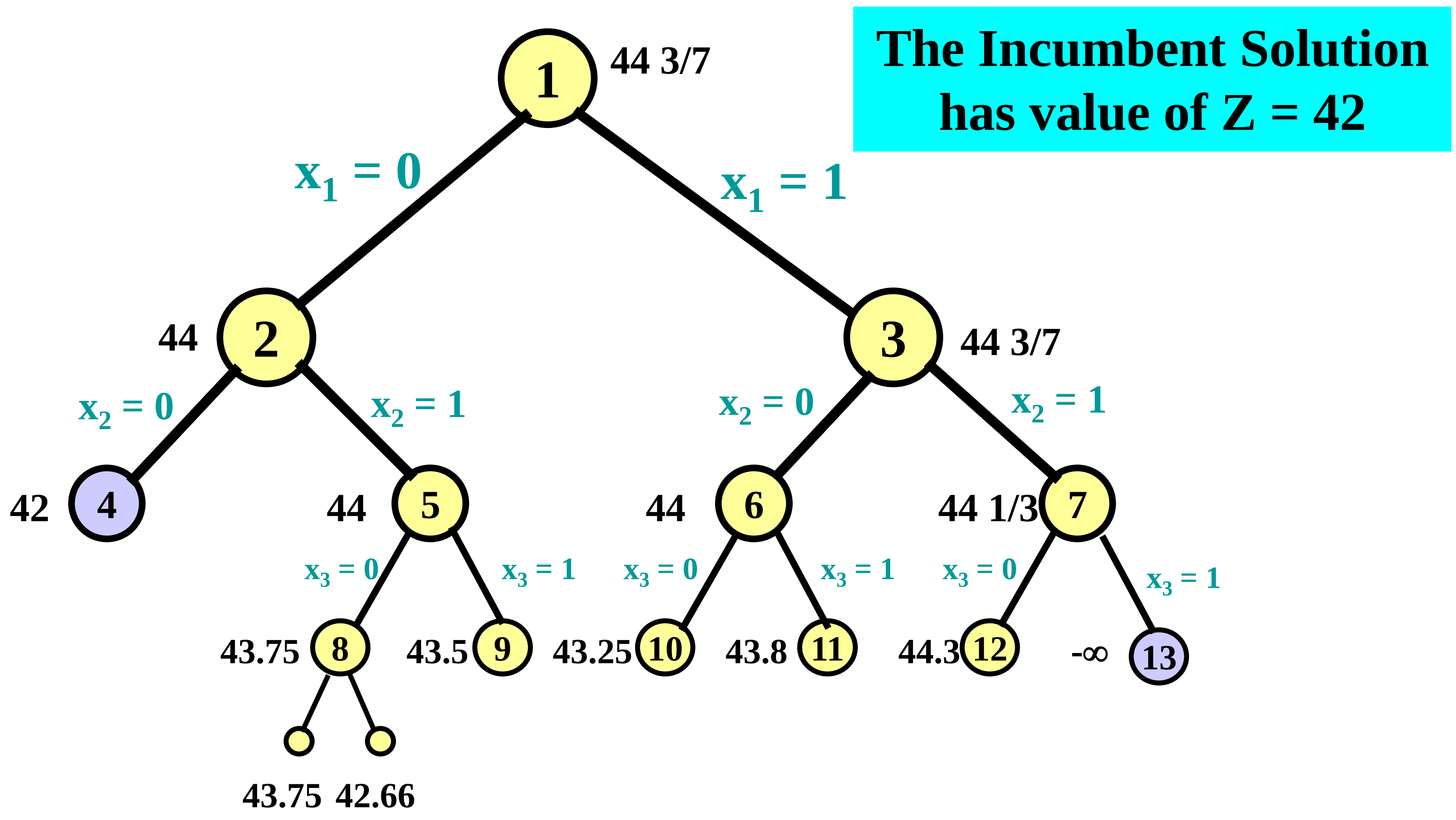


We next solved the LPs associated with Nodes 8-13

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



We next solved the LPs associated with the next nodes.

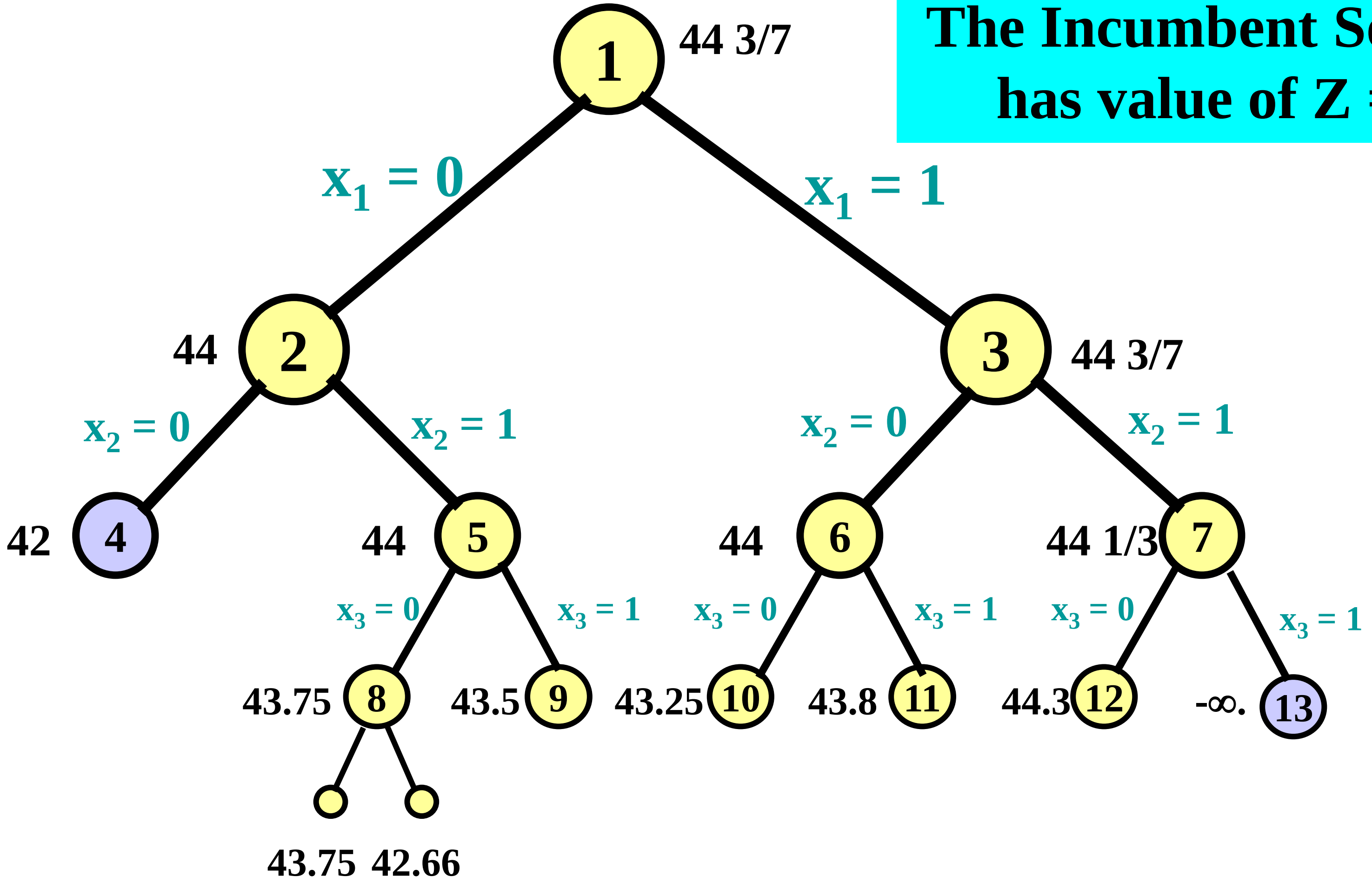
We can prune the node with a value of $Z = 42.66$. Why?

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 Z = 42.66, then the best bound is at most Z = 42.
- Other bounds can also be rounded down.

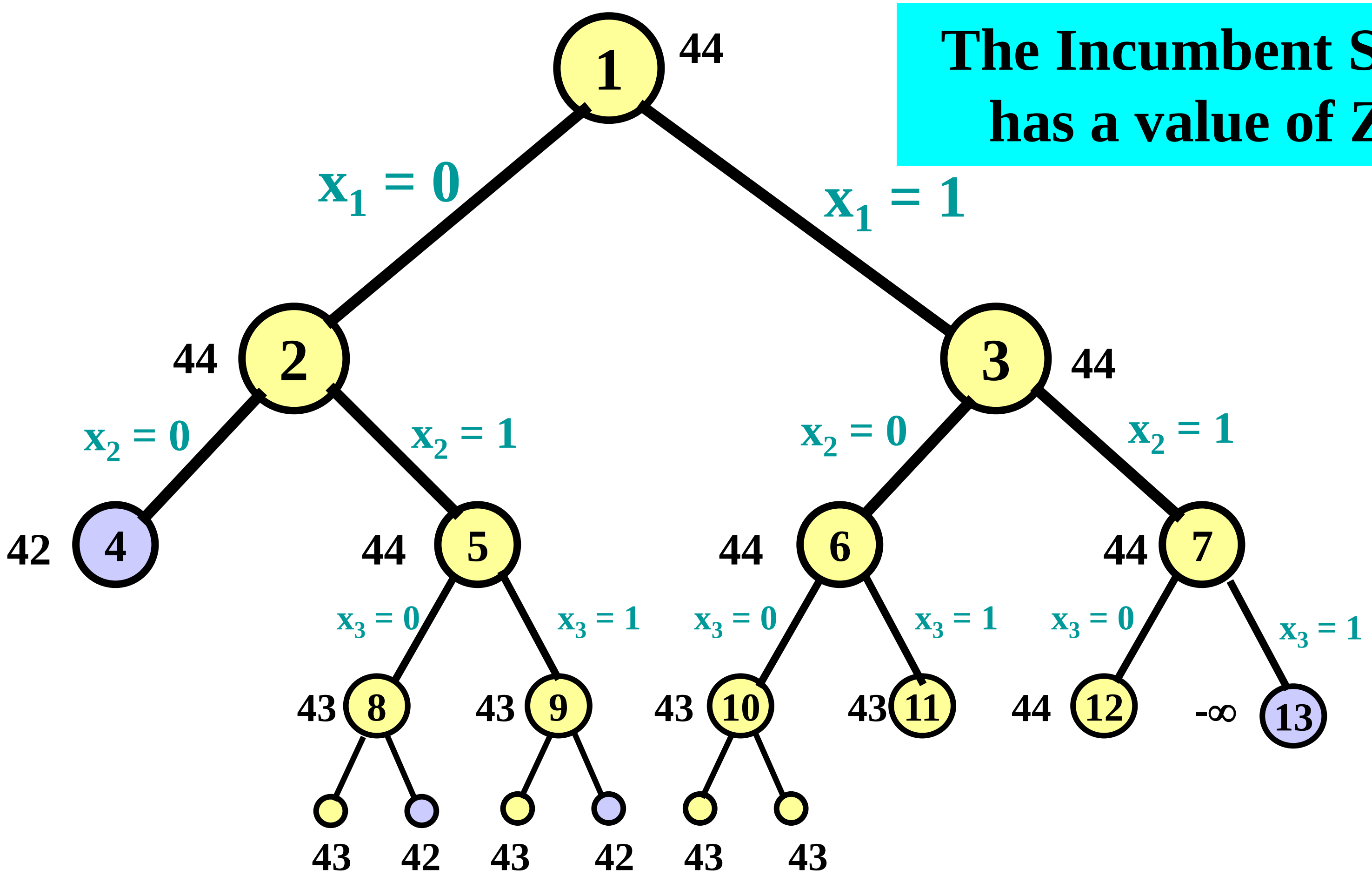
Branch and Bound

The Incumbent Solution has value of Z = 42



Branch and Bound

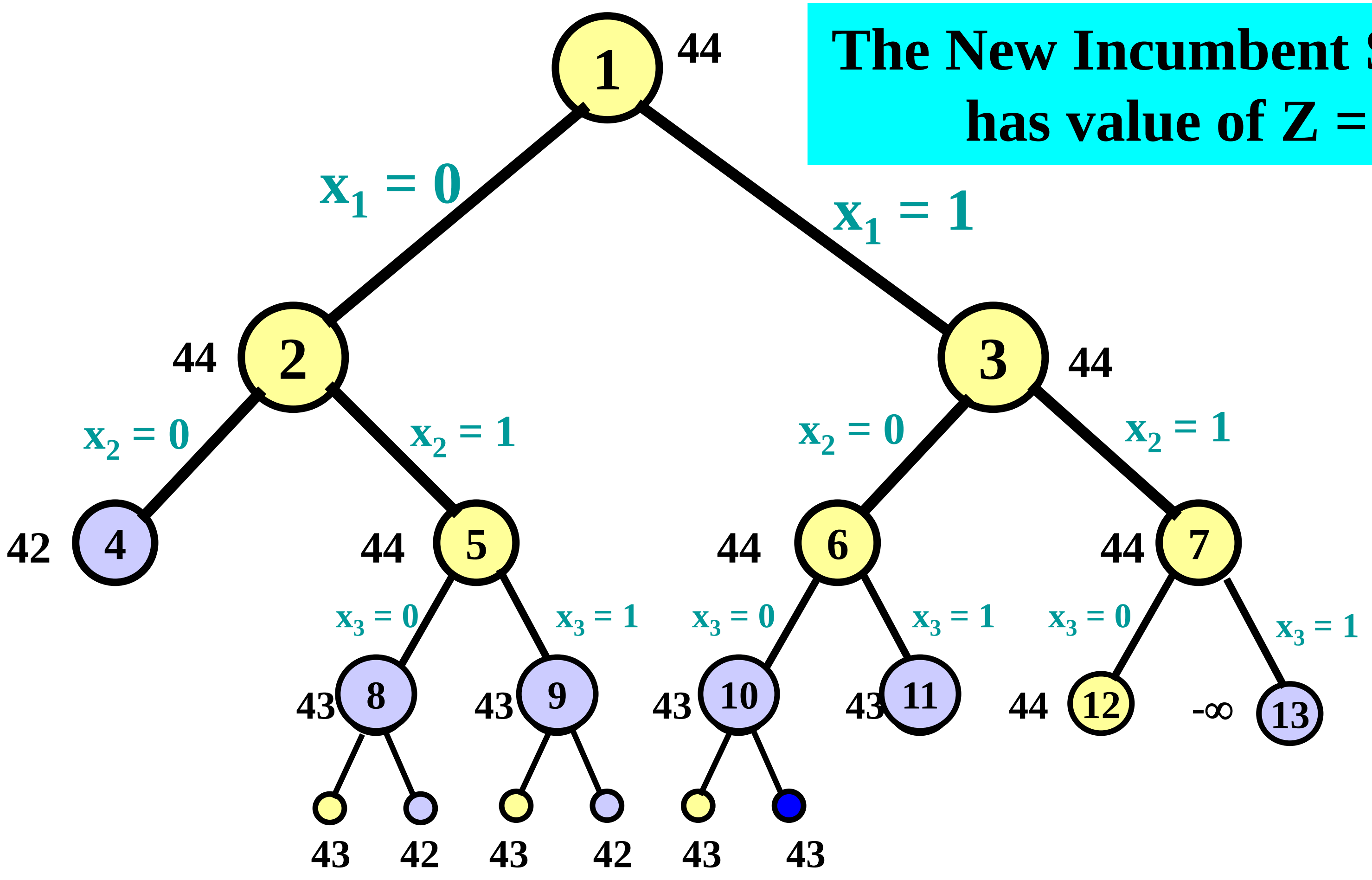
The Incumbent Solution has a value of Z = 42



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

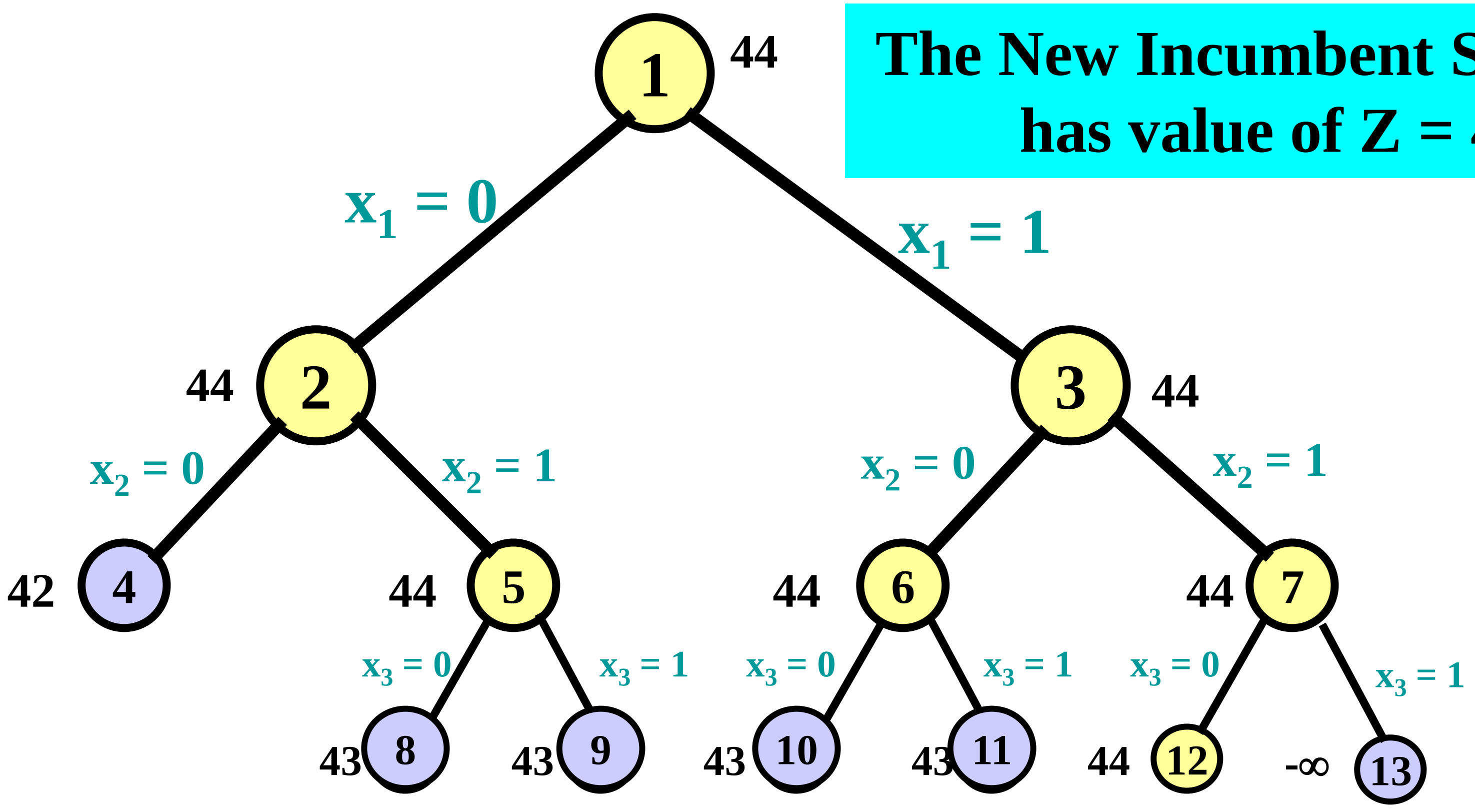


The New Incumbent Solution has value of Z = 43

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

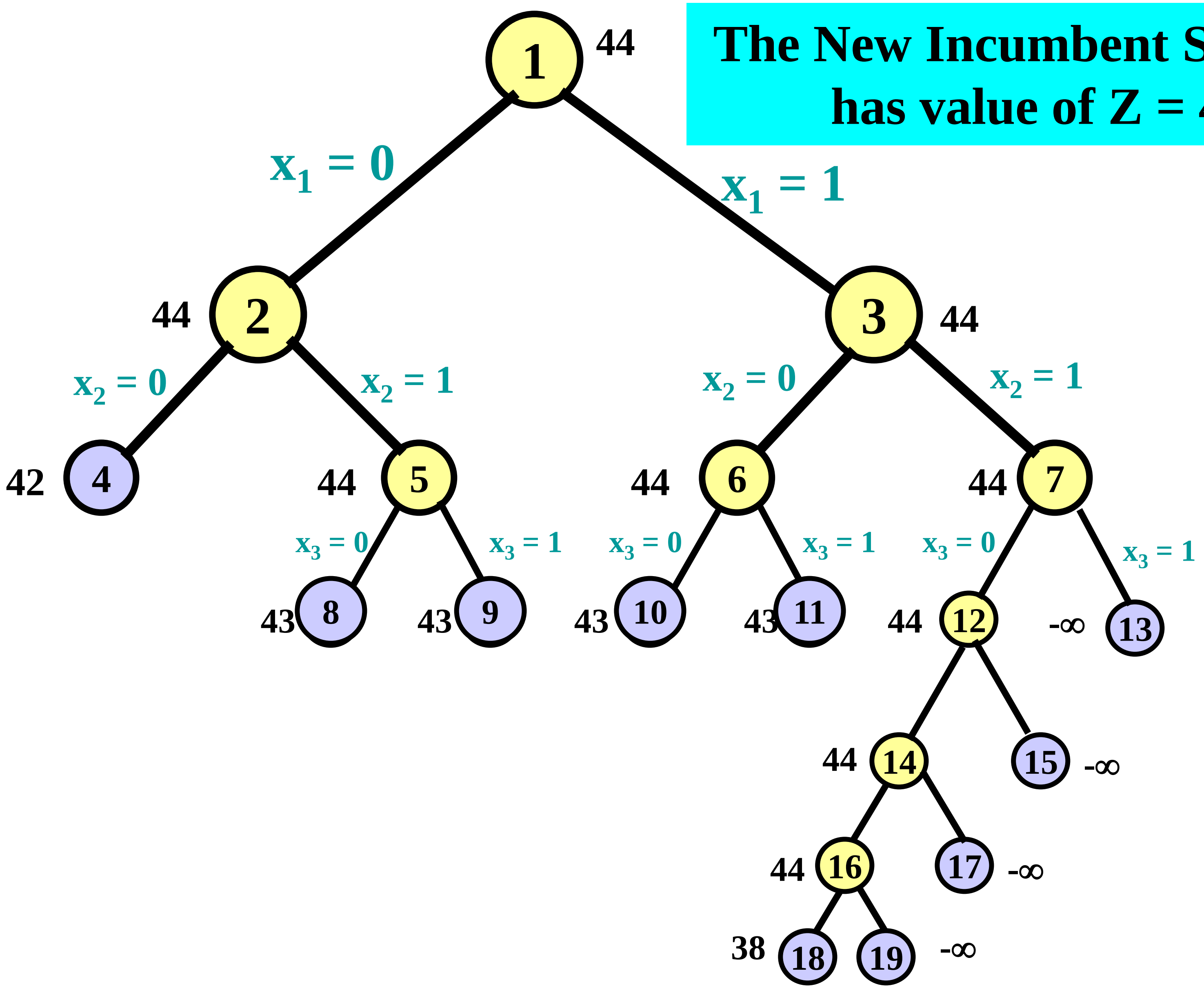


The New Incumbent Solution has value of Z = 43

If we had found this Incumbent Solution earlier, then we could have saved some searching.

Finishing Up

The New Incumbent Solution has value of Z = 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 the linear programs faster, such that we can start with an excellent or optimal solution
- Obtaining better bounds can be valuable.

  - Use properties that are obvious to us, such as the fact that integer solutions have integer solution values.

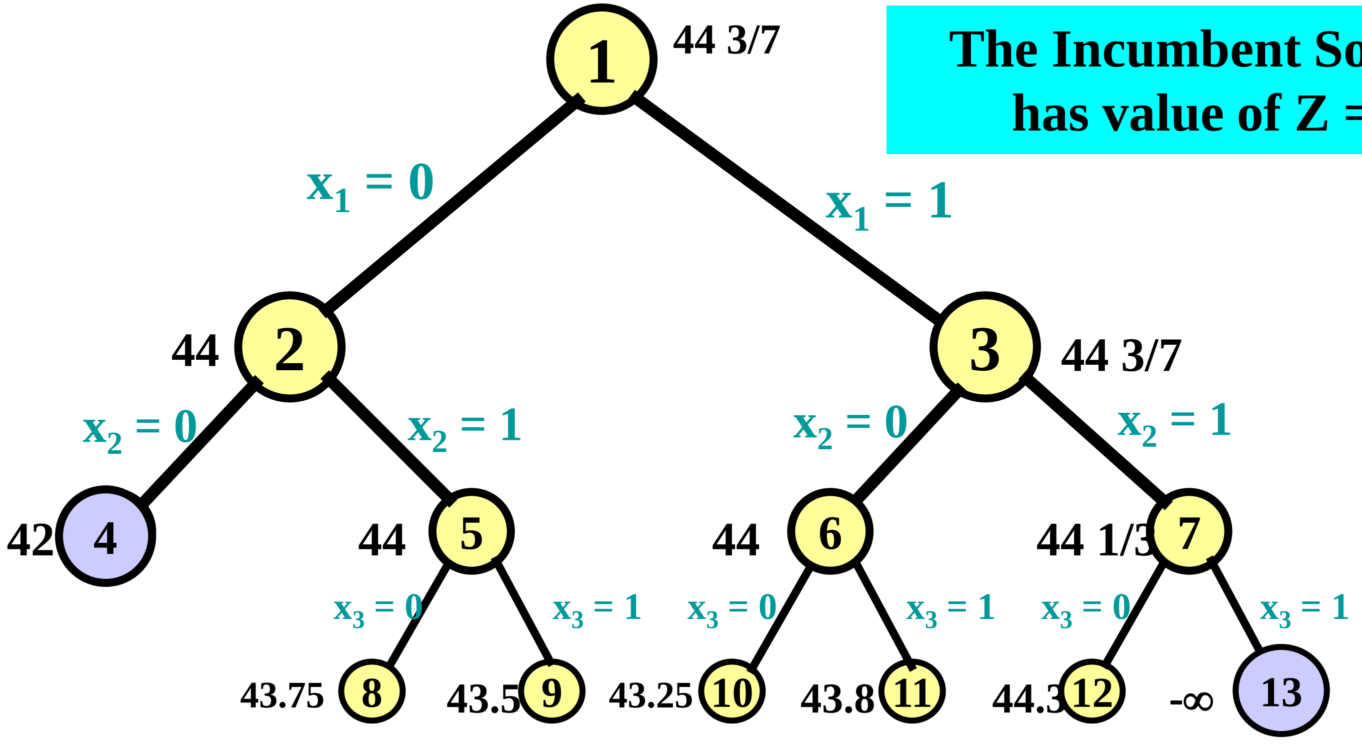
Branch and Bound

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



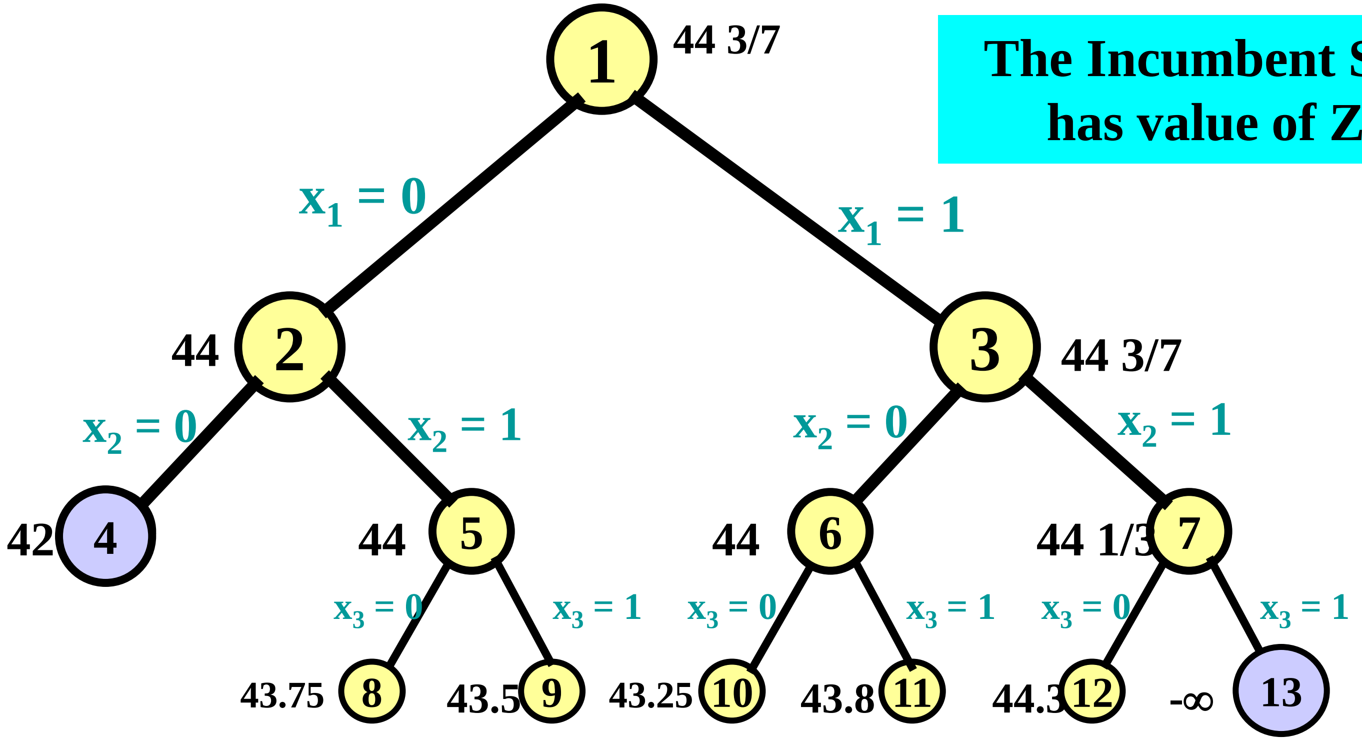
The Incumbent Solution has value of Z = 42

Z* = 43 = An Optimal Integer Solution Value. (We found it later during the search process).

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 of Z = 42

The children of Node 1 (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} **44 3/7** **1**

e.g., S = {13} **-∞** 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

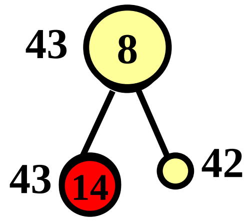
Let Z^{LP} = its Objective Value

CASE 2. $-\infty < Z^{LP} \leq Z^{I}$.

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 of Z = 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.

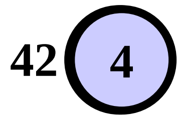
That is, the LP solution is integral and dominates the incumbent.

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.

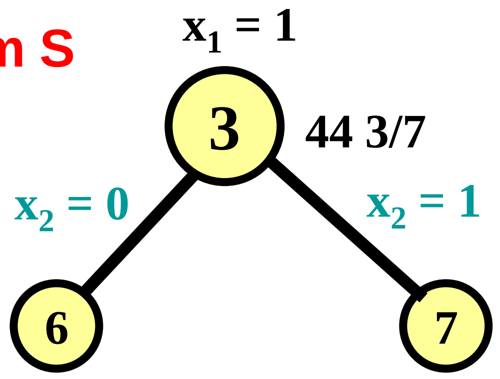
CASE 4. $Z^I < Z^{LP}$ and x^{LP} is not integral.

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}

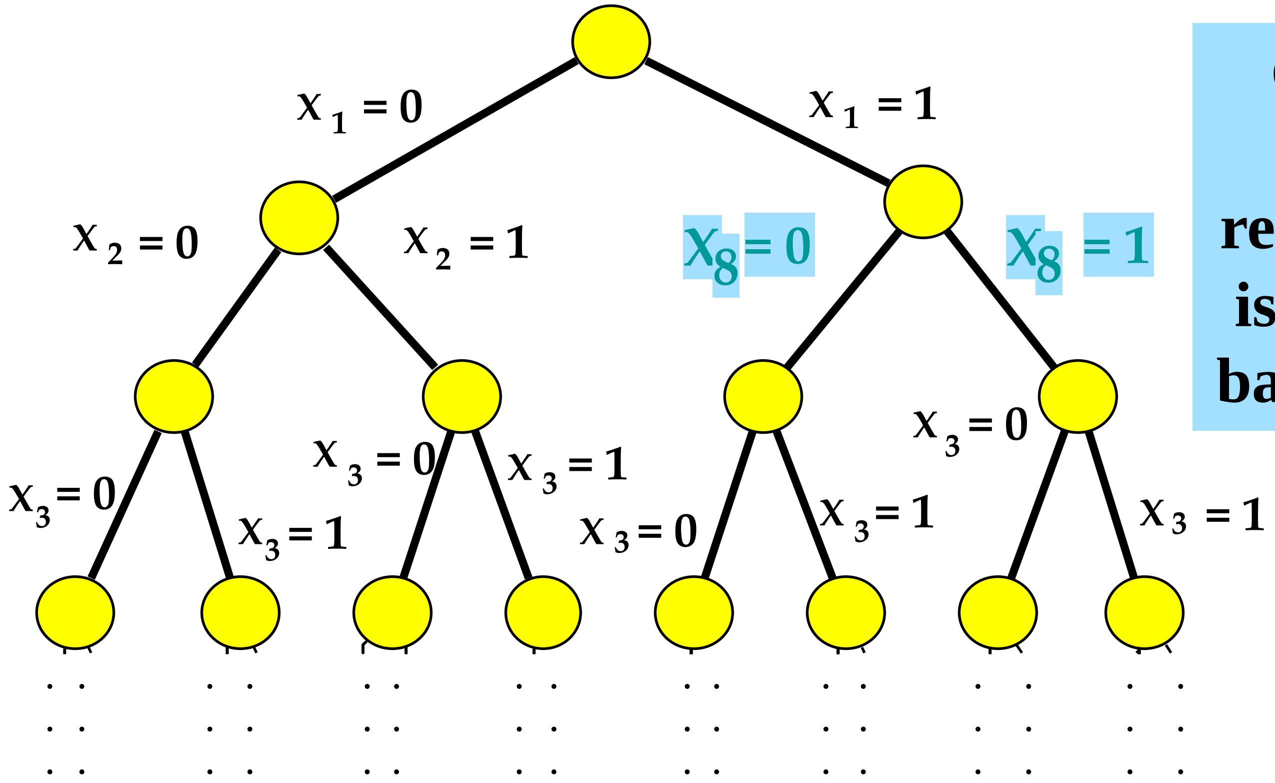
Possible Selection Rules

- 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 the 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 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 the 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 the Integer Linear Programs (ILPs) optimallly.
- There is an art to making it work well in practice.
- It is built into state-of-the-art solvers such as CPLEX. The CPLEX Interactive Optimizer is an executable program that can **read a problem interactively or from the files in certain standard formats, solve the problem, and deliver the solution interactively or into the text files**. The program consists of the file cplex.exe on Windows or cplex on UNIX platforms.