

# INTEGER PROGRAMMING: BRANCH AND BOUND

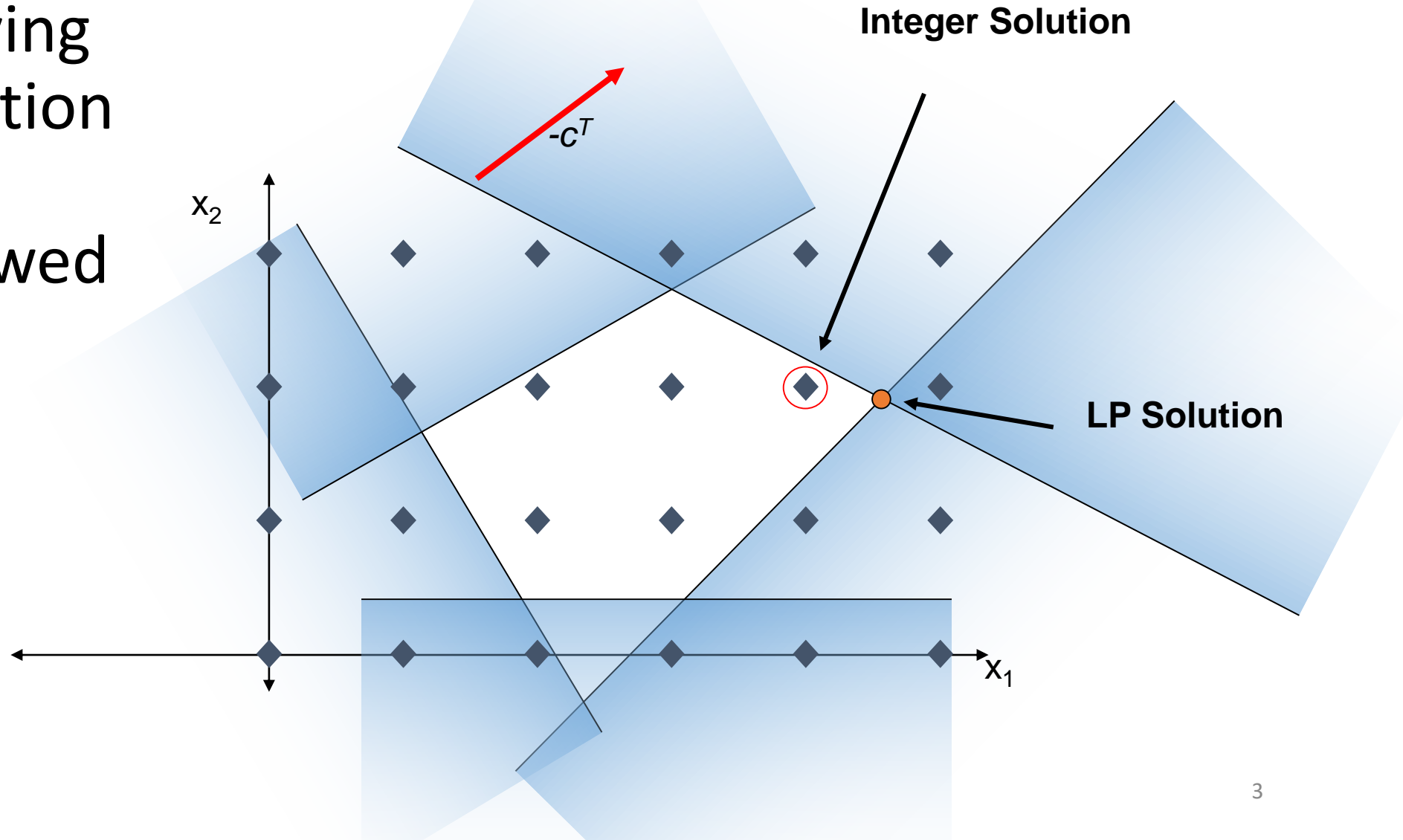
# LP relaxation

- For any IP we can generate an LP (called the *LP relaxation*) from the IP by taking the same objective function and same constraints but with the “integer requirement” on variables relaxed
- That is, the variables are allowed to be continuous and bounded by appropriate constraints

e.g.,  $x_i \in \{0,1\}$  can be replaced by the two continuous constraints  $x_i \geq 0$  and  $x_i \leq 1$

# solving an IP

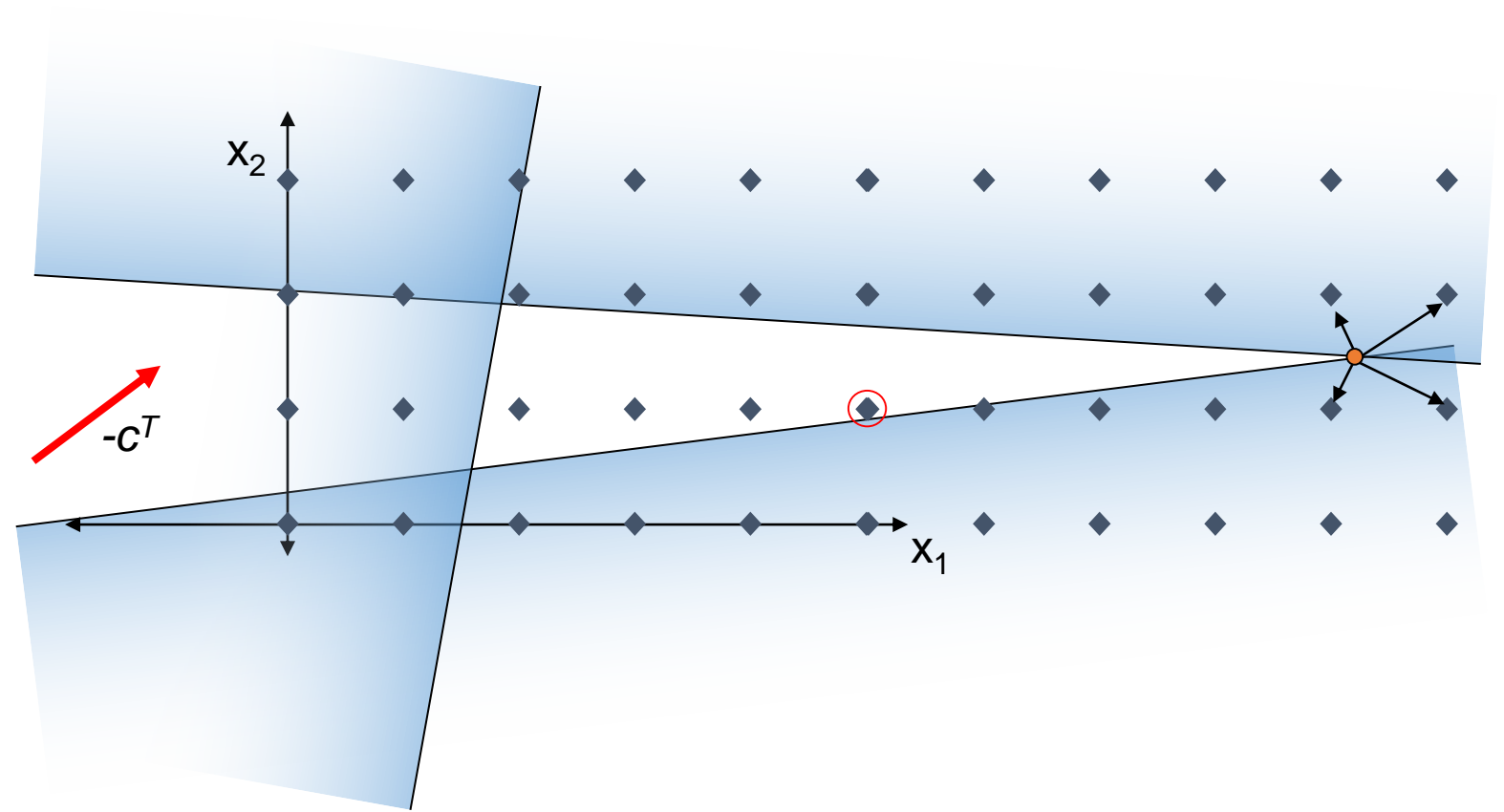
How about solving the IP minimization problem by LP relaxation followed by rounding?



# solving an IP: LP rounding

In general, rounding an LP solution does not work

Rounding can  
be arbitrarily  
far away from  
integer solution



But, the LP provides a lower bound on the IP minimization

# solving an IP

***Complete enumeration***: systematically consider all possible values of the decision variables

e.g., if there are  $n$  binary variables, there are  $2^n$  different combinations of decision variable values

Usual idea: iteratively break the problem in two

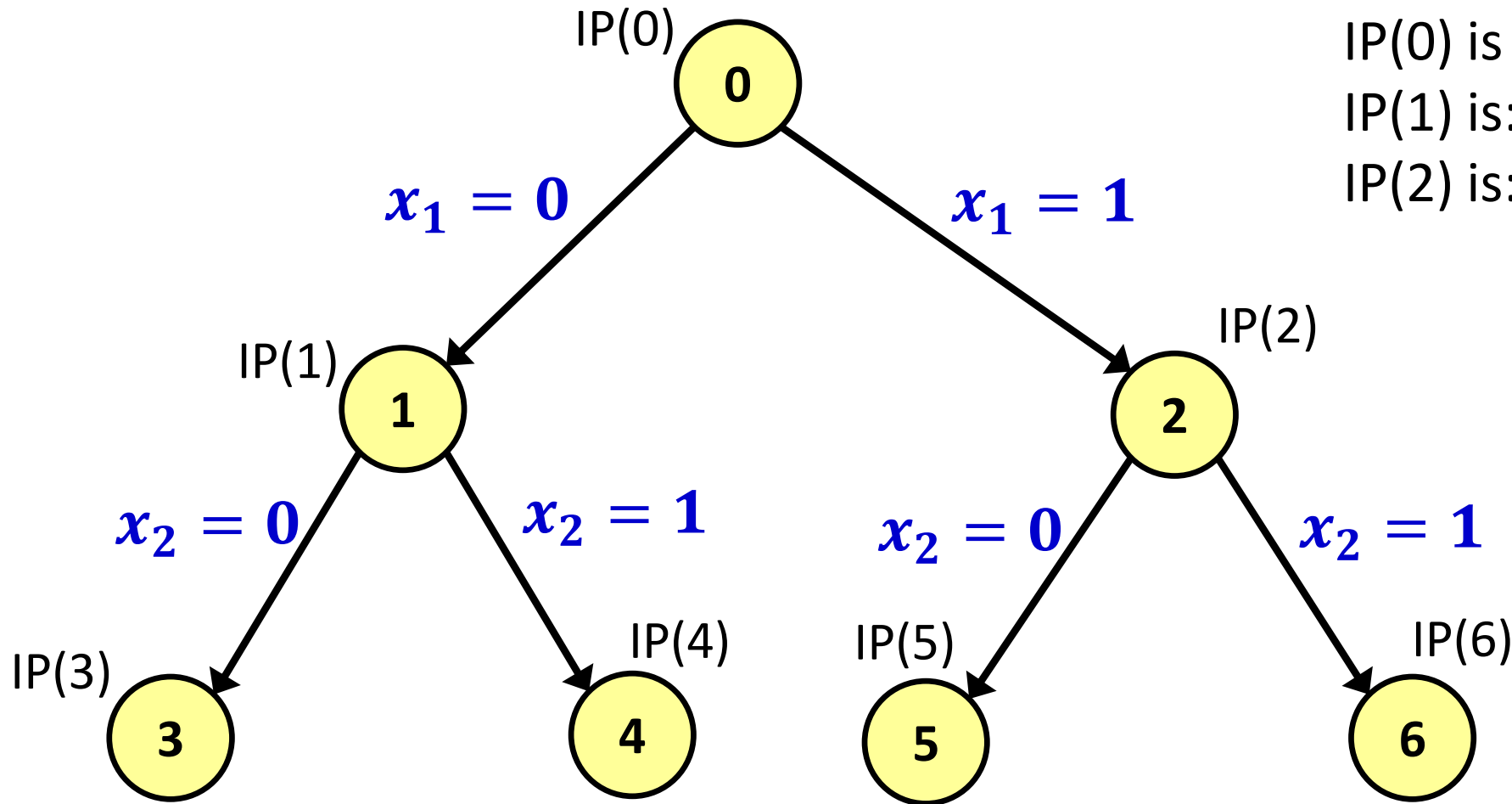
# solving an IP: complete enumeration

Consider a BP problem

→ iteratively break the problem in two:

- consider separately the case that  $x_1 = 0$  and  $x_1 = 1$ 
  - these *sub-problems* are called *nodes*
  - each node represents the original problem plus additional constraints
- the resulting nodes are split again on  $x_2$ , then  $x_3$ , etc.
- the complete series of nodes and constraint *branches* form an *enumeration tree*

# solving an IP: enumeration tree



Note: If there are  $n$  binary variables, there are  $2^n$  leaves in the enumeration tree

# solving an IP: complete enumeration

Suppose that we could evaluate 1 billion solutions per second

Let  $n$  = number of binary variables

Problem size	Approximate solution times
$n = 30$	1 second

This is a problem... complete enumeration is not reasonable.



# solving an IP: complete enumeration

## LP relaxation and rounding

- no guarantee on the quality of the solution
- but, it is fast
- does provide a “lower bound” on IP minimization

## Complete enumeration

- guarantee on quality of solution  $\rightarrow$  *optimal*
- but, it is slow... *really*, slow

# solving an IP: branch-and-bound

However, we can combine the two approaches LP relaxation and enumeration:

- 1. Solve LP Relaxation to get fractional solution**
- 2. Create two branches (sub-problems) by adding constraints**
- 3. Use the “lower bound” property of LP solutions to eliminate parts of the tree**

The basic concept underlying the branch-and-bound technique is to *divide and conquer*.

# solving an IP: branch-and-bound

The dividing (*branching*) is done by partitioning the entire set of feasible solutions into smaller and smaller subsets.

The conquering (*fathoming*) is done partially by:

- i. giving a bound for the best solution in the branch
- ii. discarding the branch if the bound indicates that it can not contain an optimal solution

# branch & bound: terminology

## *Incumbent solution*

- an IP feasible solution that is the best solution so far in the B&B search
- its objective value provides an *upper bound* to the minimization problem

## *Lower bound*

- a solution to an “easier” problem such as LP relaxation
- ideally, you are looking for “tight” lower bounds – those that are closer to the optimal objective value

## *Fathoming*

- dismiss nodes (and thus whole branches of the tree!) if further branching will yield no useful information

# branch & bound

1. Solve LP relaxation for lower bound on cost for current branch
  - If solution exceeds upper bound (incumbent value), branch is terminated
  - Else: if solution is integer, replace incumbent solution and update upper bound
2. Create two branched problems by adding constraints to the problem
  - Select integer variable with fractional LP solution
  - Add constraints for this variable
3. Repeat until no branches remain, return optimal solution

# major steps in B&B algorithm

1. Initialization
2. Check Stopping Condition
3. Explore Active Subproblem (Fathoming)
4. Branching
5. Bounding

# step 1: initialization

- Let the node index  $i = 0$
- Make node 0 (root node) *active*
- If any feasible IP solutions are known, let the best one be the incumbent solution. Otherwise, let  $z^* = \infty$



## step 2: check stopping condition

If there are no active nodes, then: **stop**.

- If there is an incumbent solution it is optimal
- If not, the IP is infeasible

Else (if there are any active nodes)

Select active node  $i$  and attempt to solve the associated subproblem  $LP_i$

## step 3: explore active subproblem

If  $LP_i$  is infeasible then

fathom node  $i$  and go to Step 2

else if  $LP_i$  solution is IP feasible then

If  $z_i < z^*$  then

Let  $z^* = z_i$

$LP_i$  solution is the new incumbent solution

Fathom node  $i$  and go to Step 2

else if  $z_i \geq z^*$  then

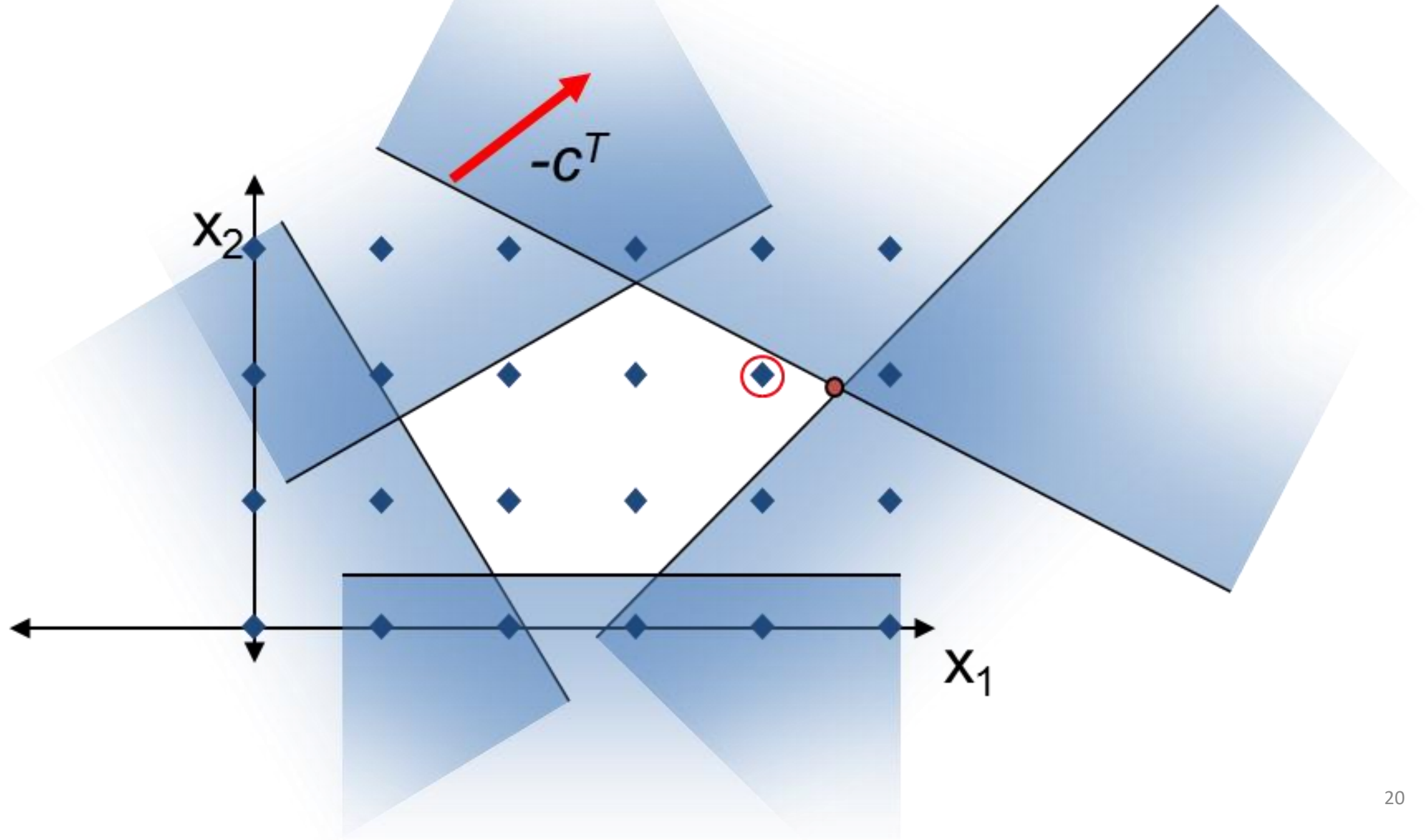
fathom node  $i$  and go to Step 2

else continue to Step 4

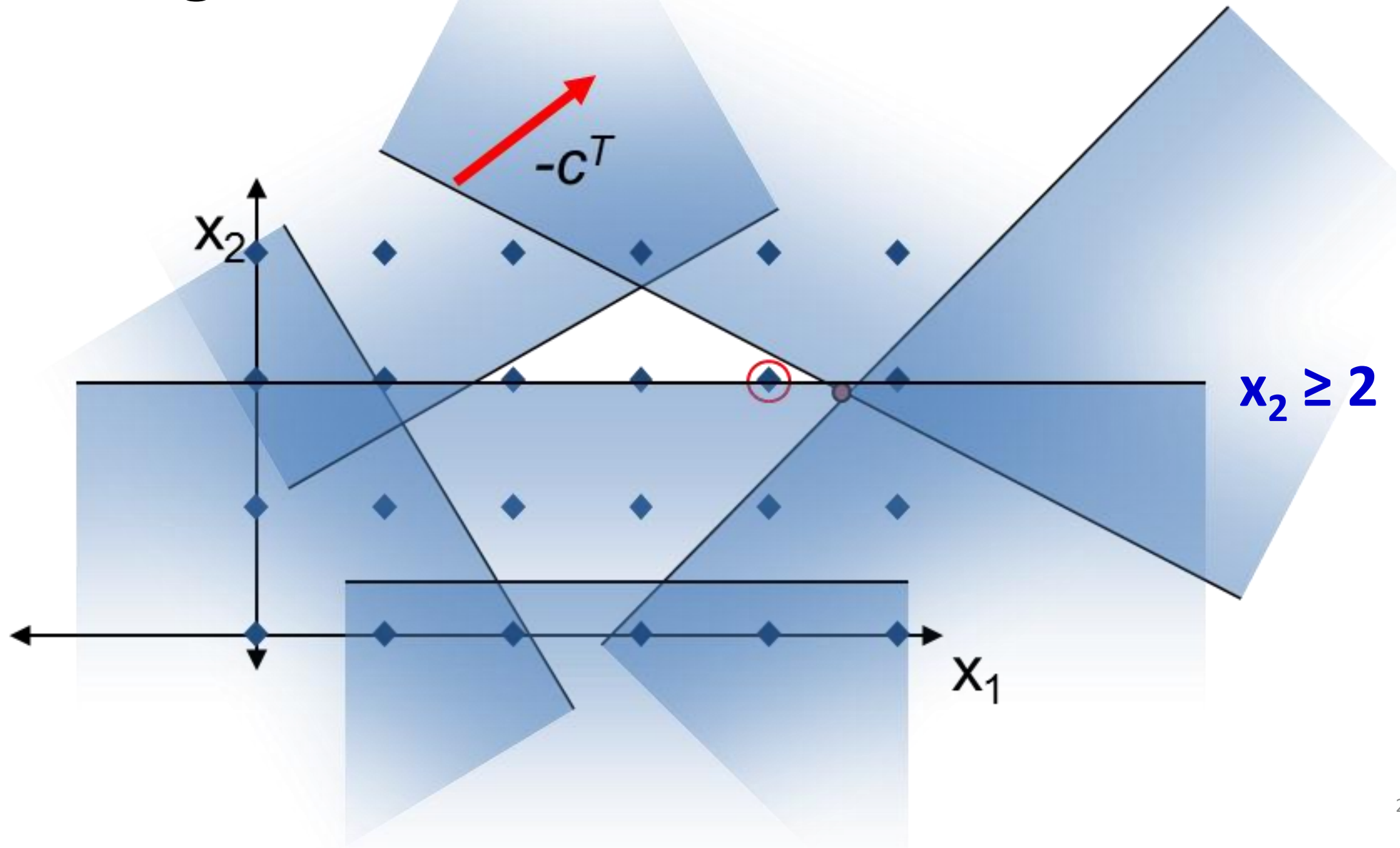
## step 4: branching

- Select a integer variable  $x$ , that has fractional value  $v$  in the optimal solution to  $LP_i$ 
  - Create node  $i + 1$  (down branch)  
i.e., let the associated subproblem  $LP_{i+1}$  be  $LP_i$  subject to  $x \leq \lfloor v \rfloor$
  - Create node  $i + 2$  (up branch):  
i.e., let the associated subproblem  $LP_{i+2}$  be  $LP_i$  subject to  $x \geq \lceil v \rceil$
- Let the lower bound at nodes  $i$ ,  $i + 1$ , and  $i + 2$  be  $z_i$
- Make nodes  $i + 1$  and  $i + 2$  active
- Make node  $i$  inactive
- Go to Step 2

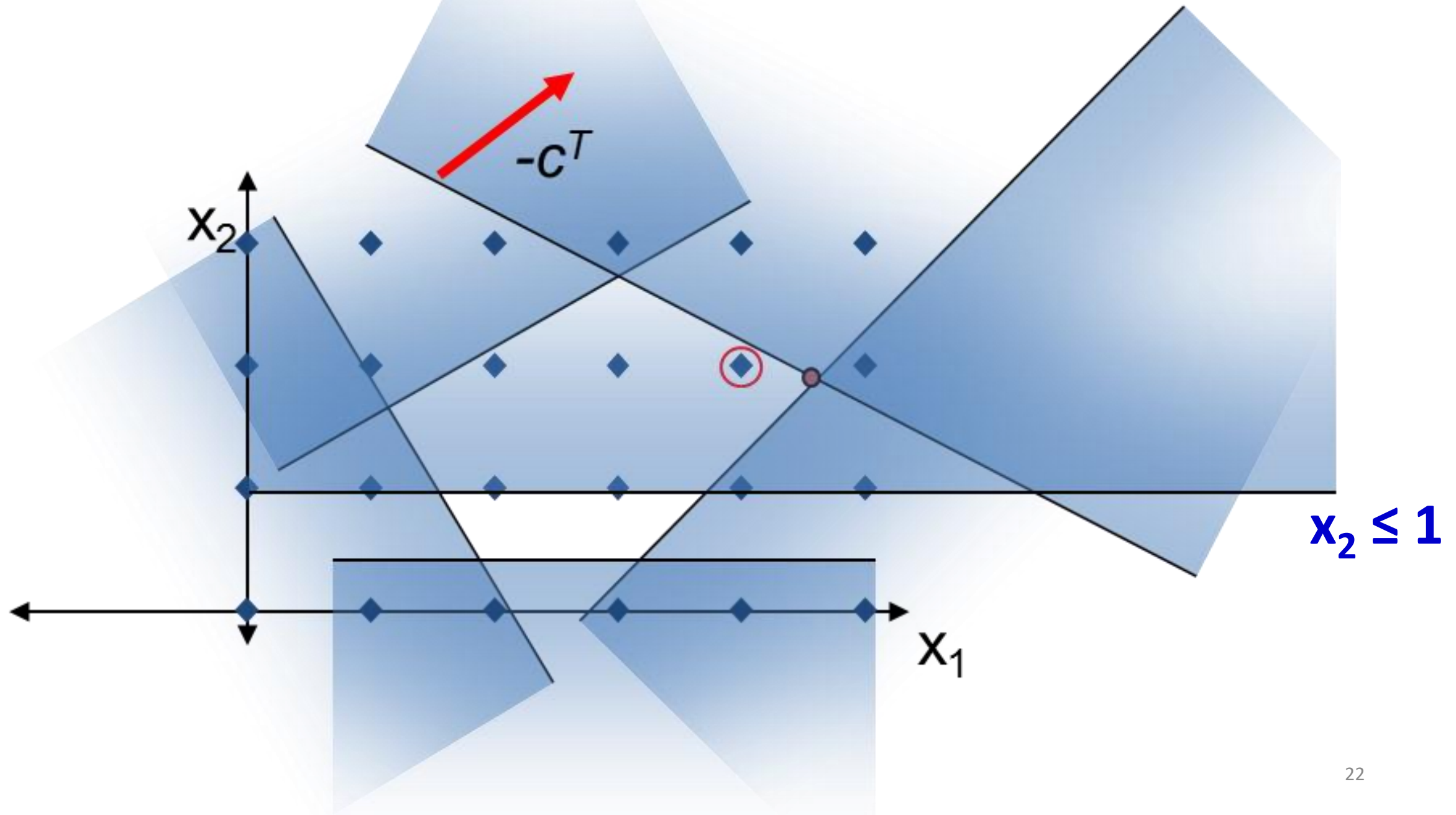
# branching for an IP



# branching for an IP



# branching for an IP



# bounding

- A valid lower bound on the optimal objective function value can be found by taking the minimum lower bound over all active nodes.
- If there are no active nodes and there is an incumbent solution, then the incumbent solution provides both a lower and upper bound.

# INTEGER PROGRAMMING: BRANCH AND BOUND EXAMPLE



# B&B example: An IP problem with 4 integer variables

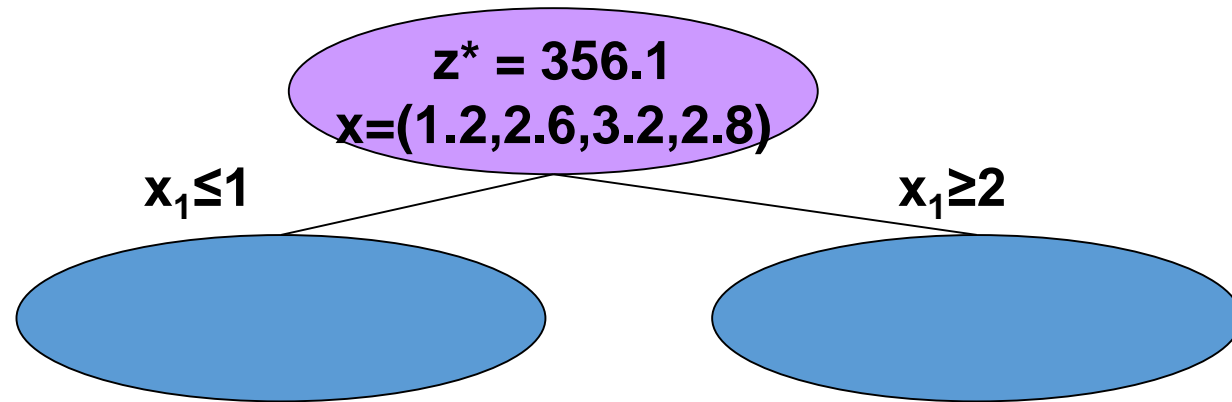
$$x_1, x_2, x_3, x_4 \in \{0,1\}$$

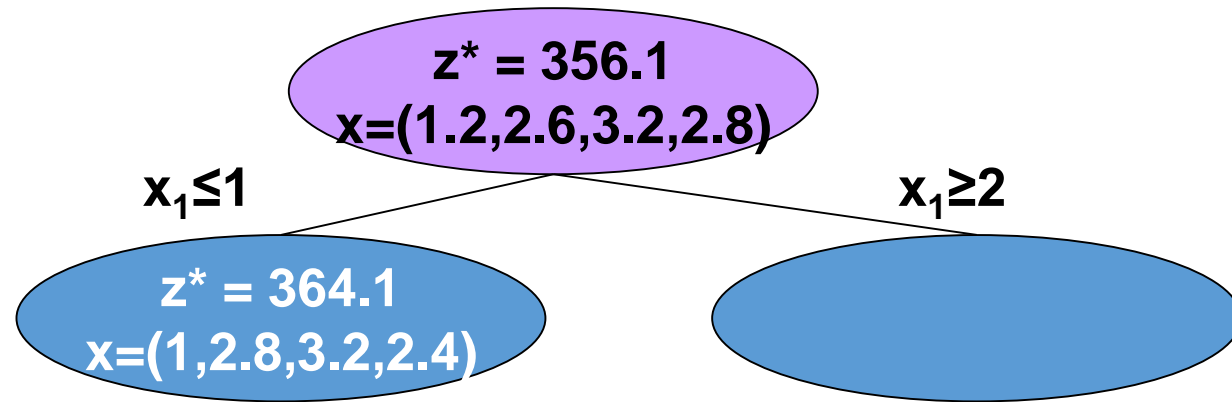
Let  $z_{IP}$  denote the value for which we wish to minimize, i.e.,  $z_{IP} = \min f(x)$

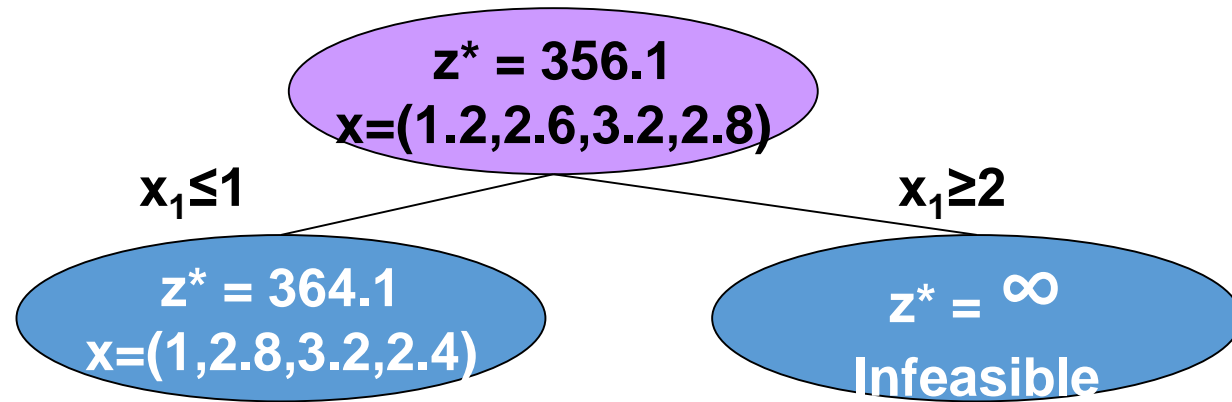
Let  $z^*$  denote the optimal objective value for any LP relaxation of the IP problem or sub-problem

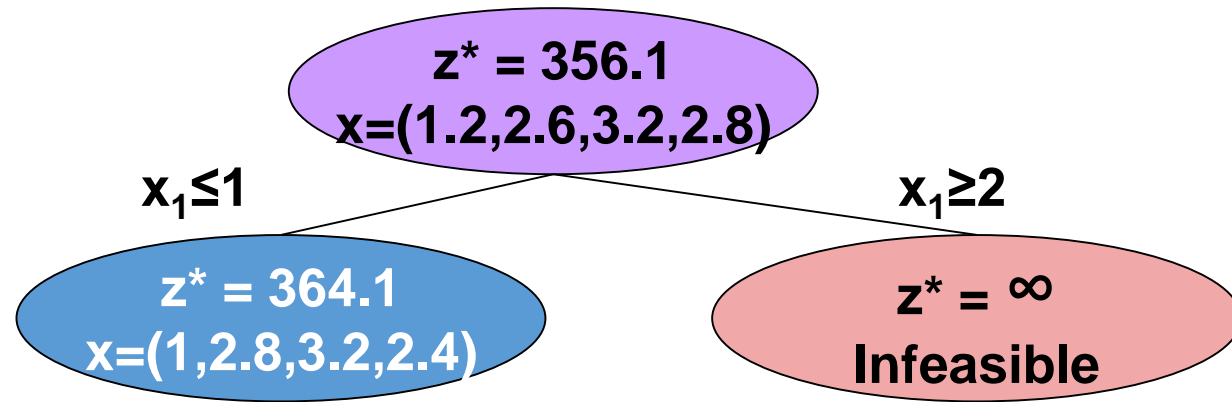
**Initial LP (root node)**

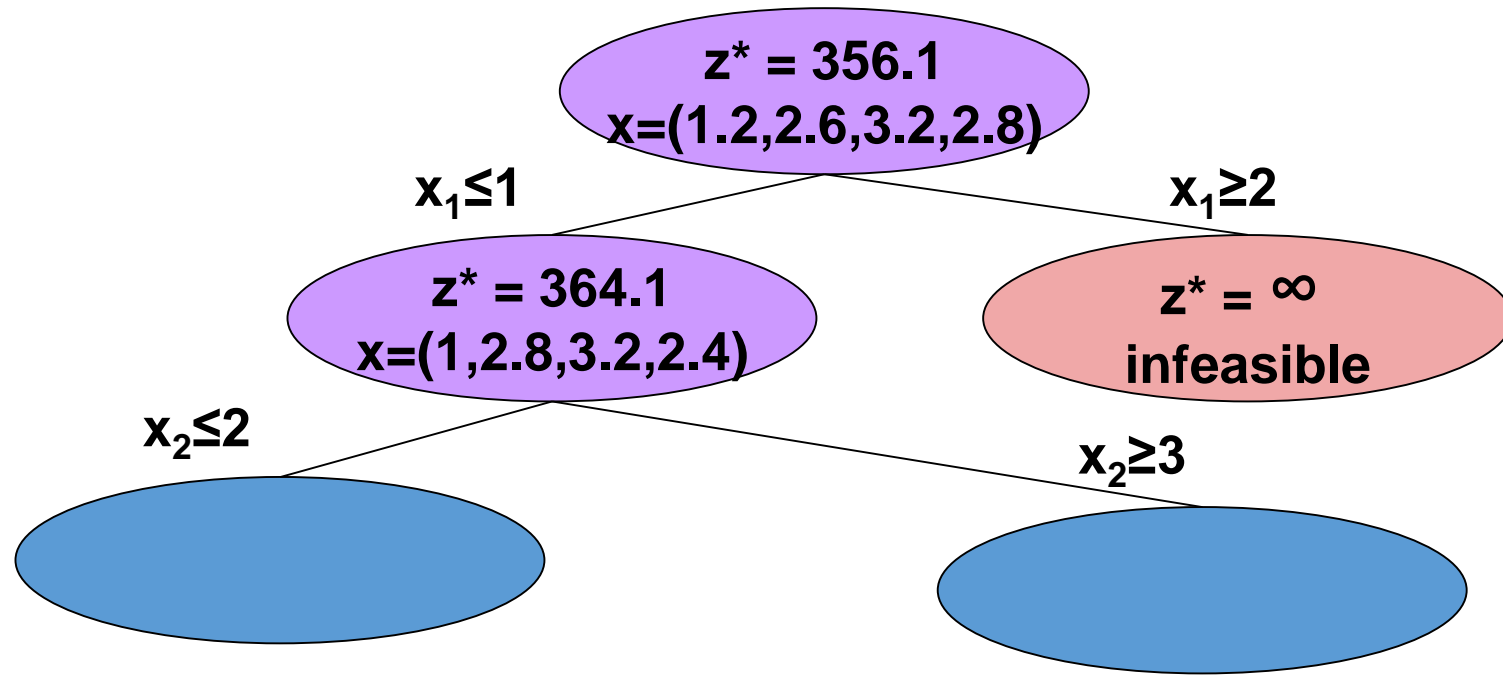
$$z^* = 356.1$$
$$x = (1.2, 2.6, 3.2, 2.8)$$

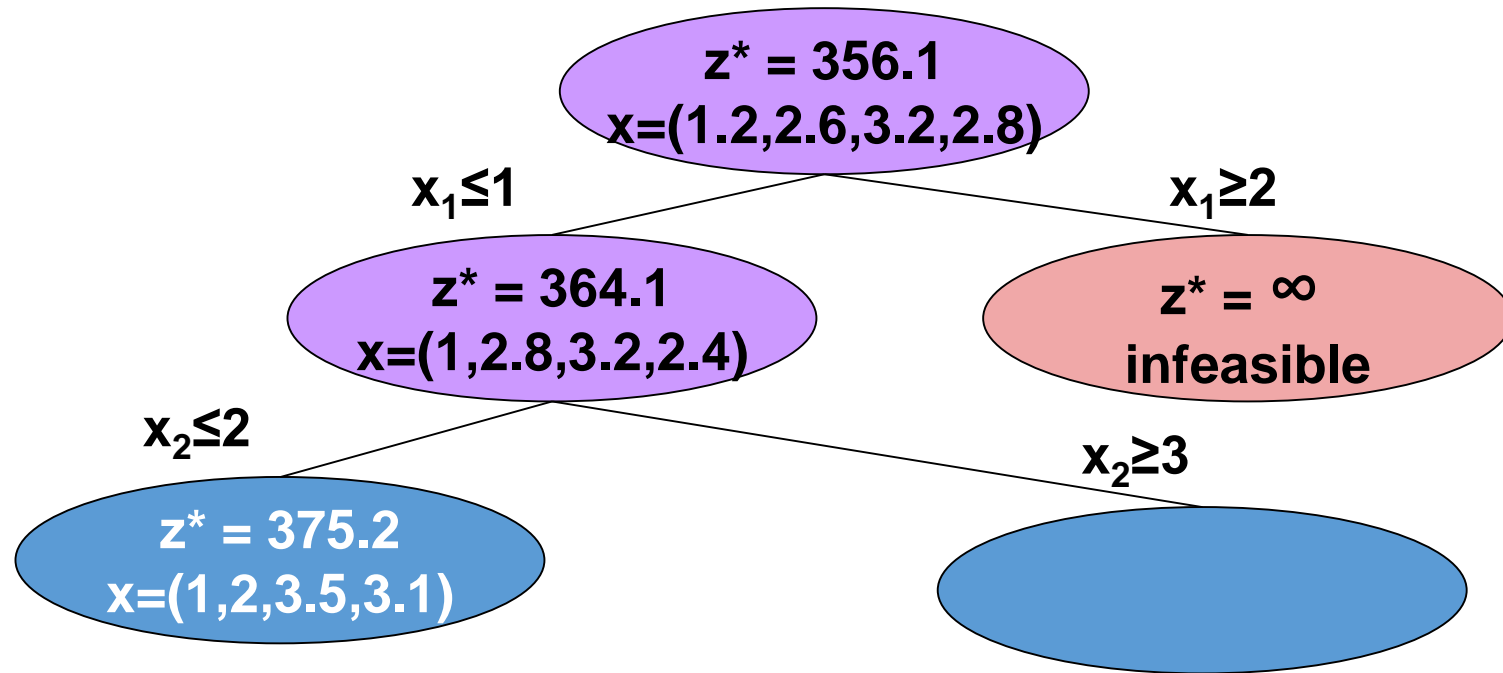




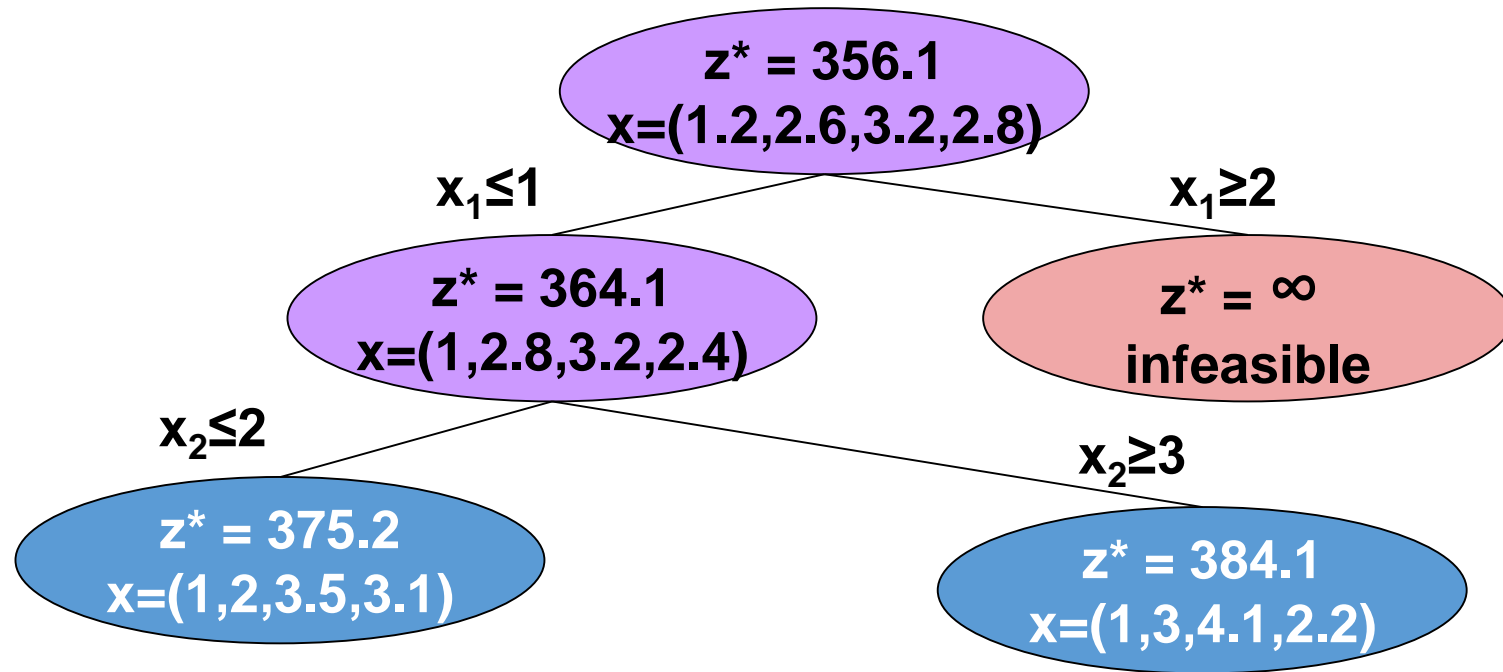


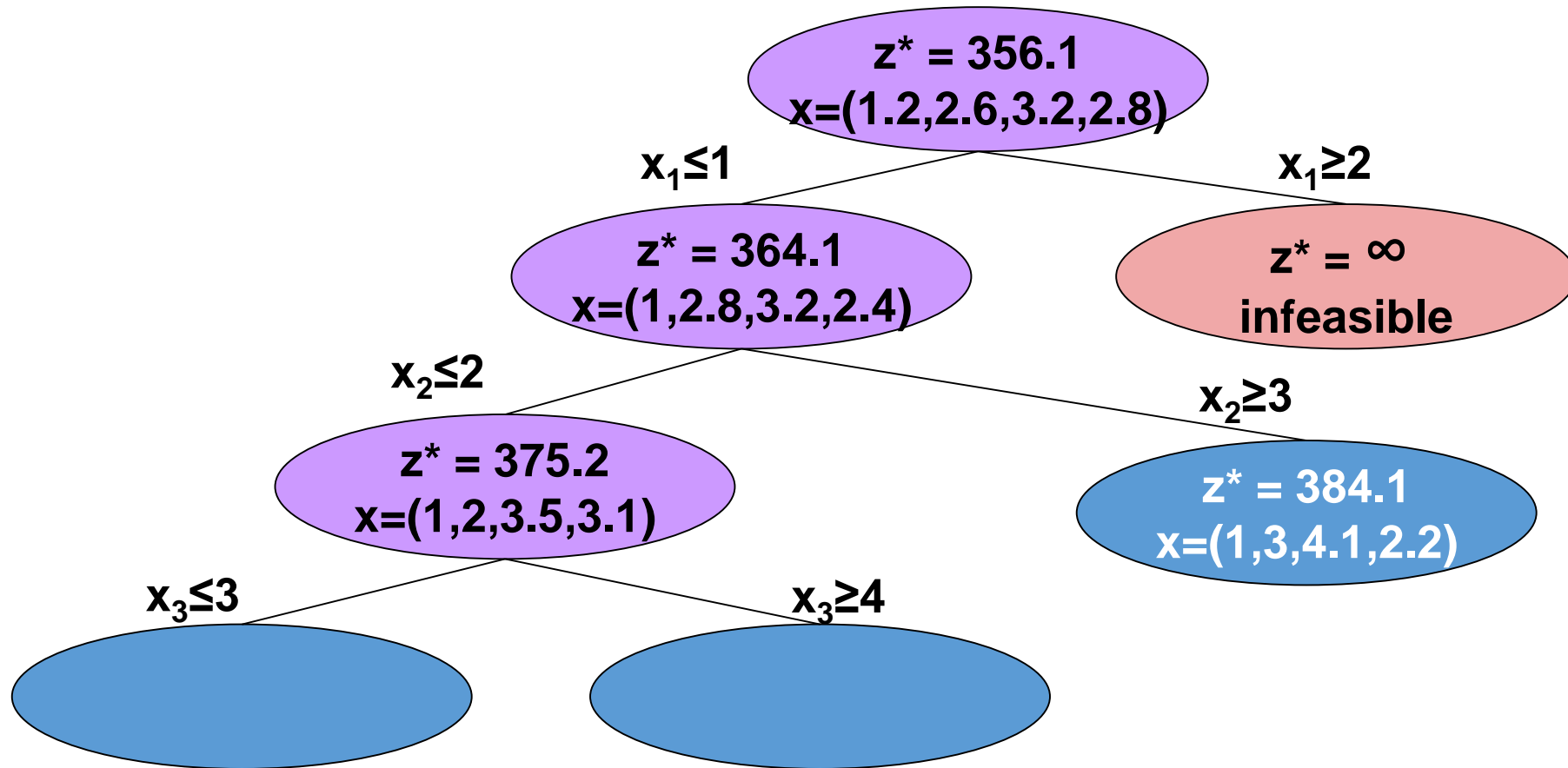


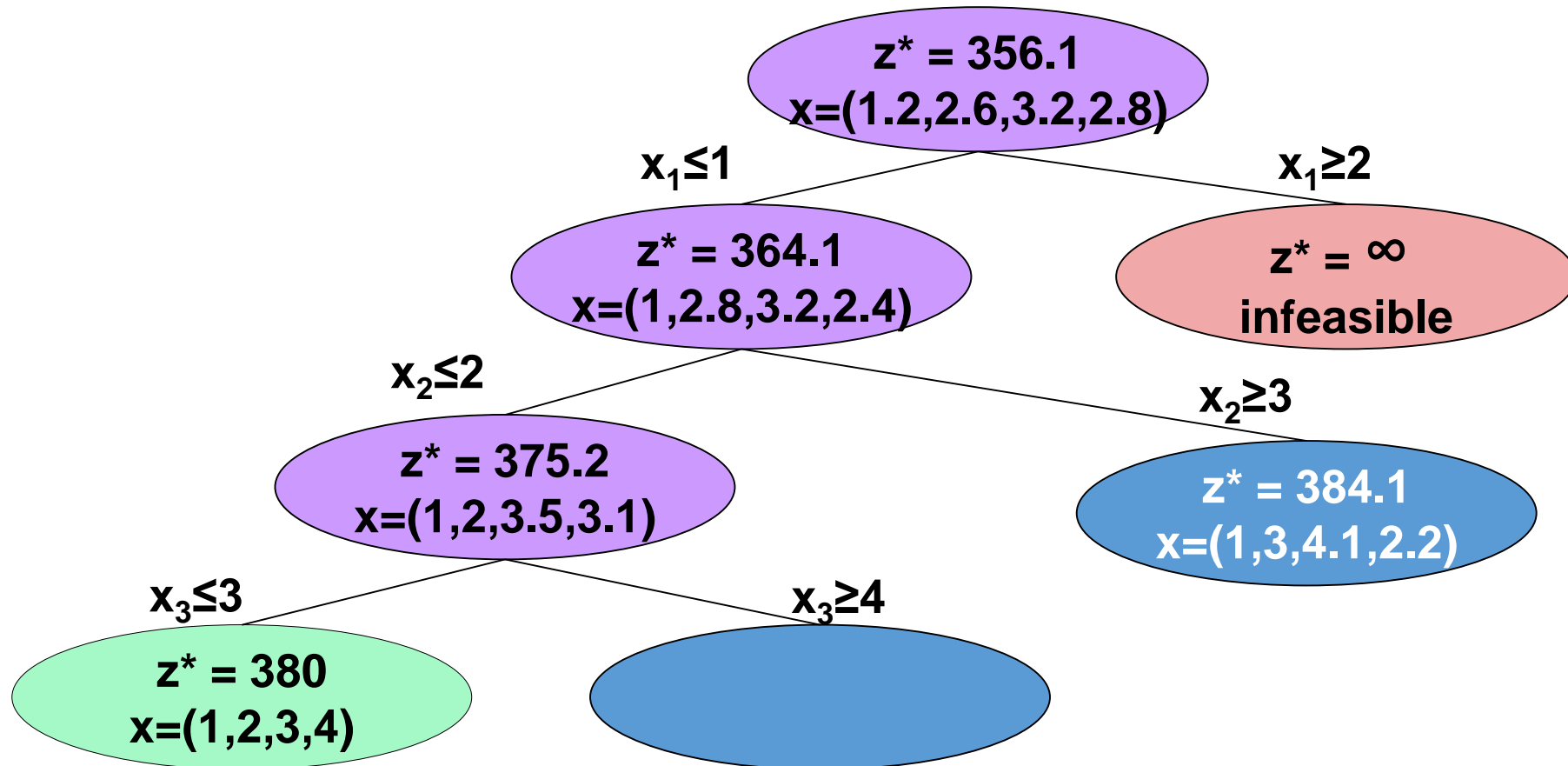


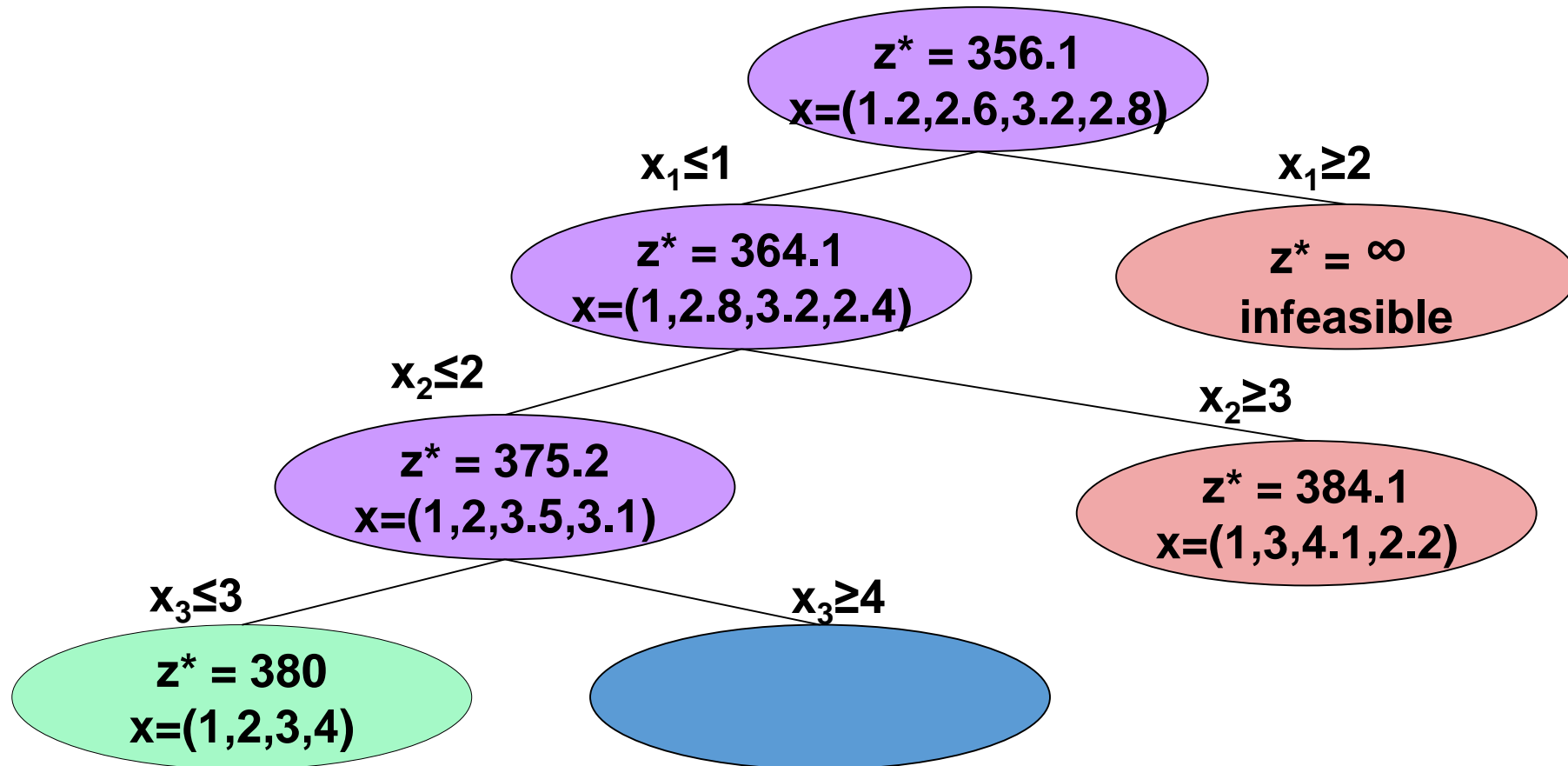


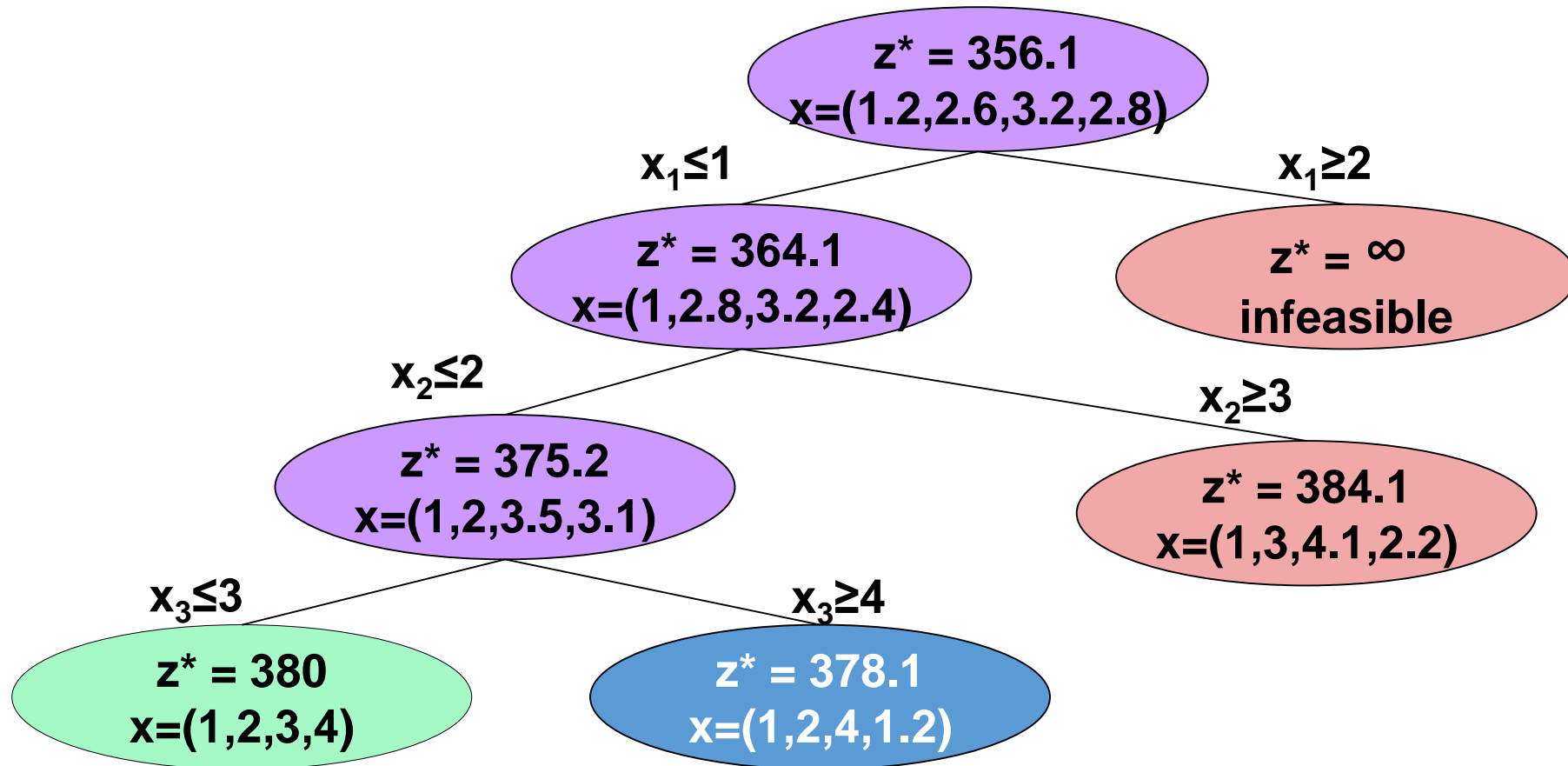


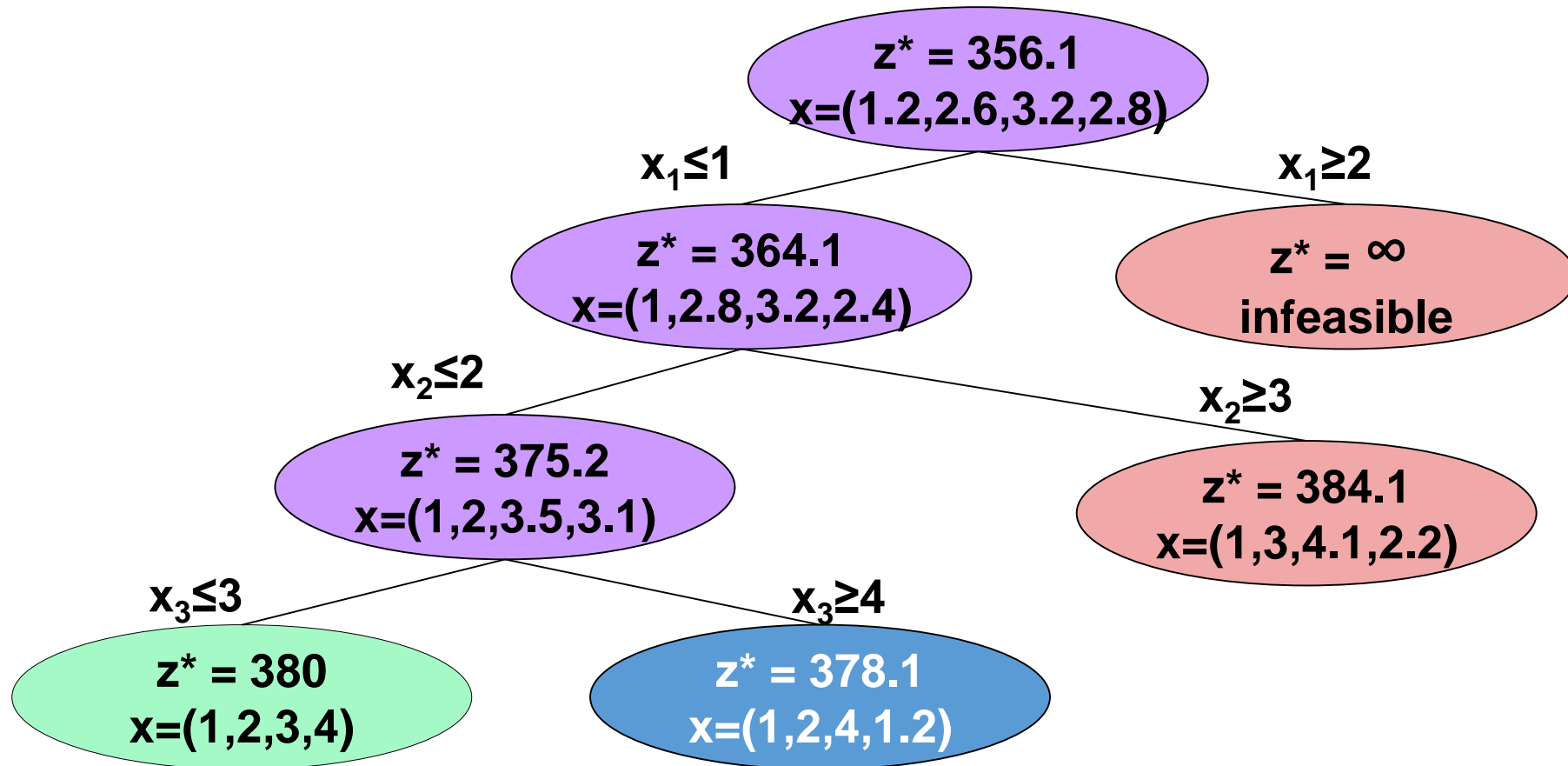


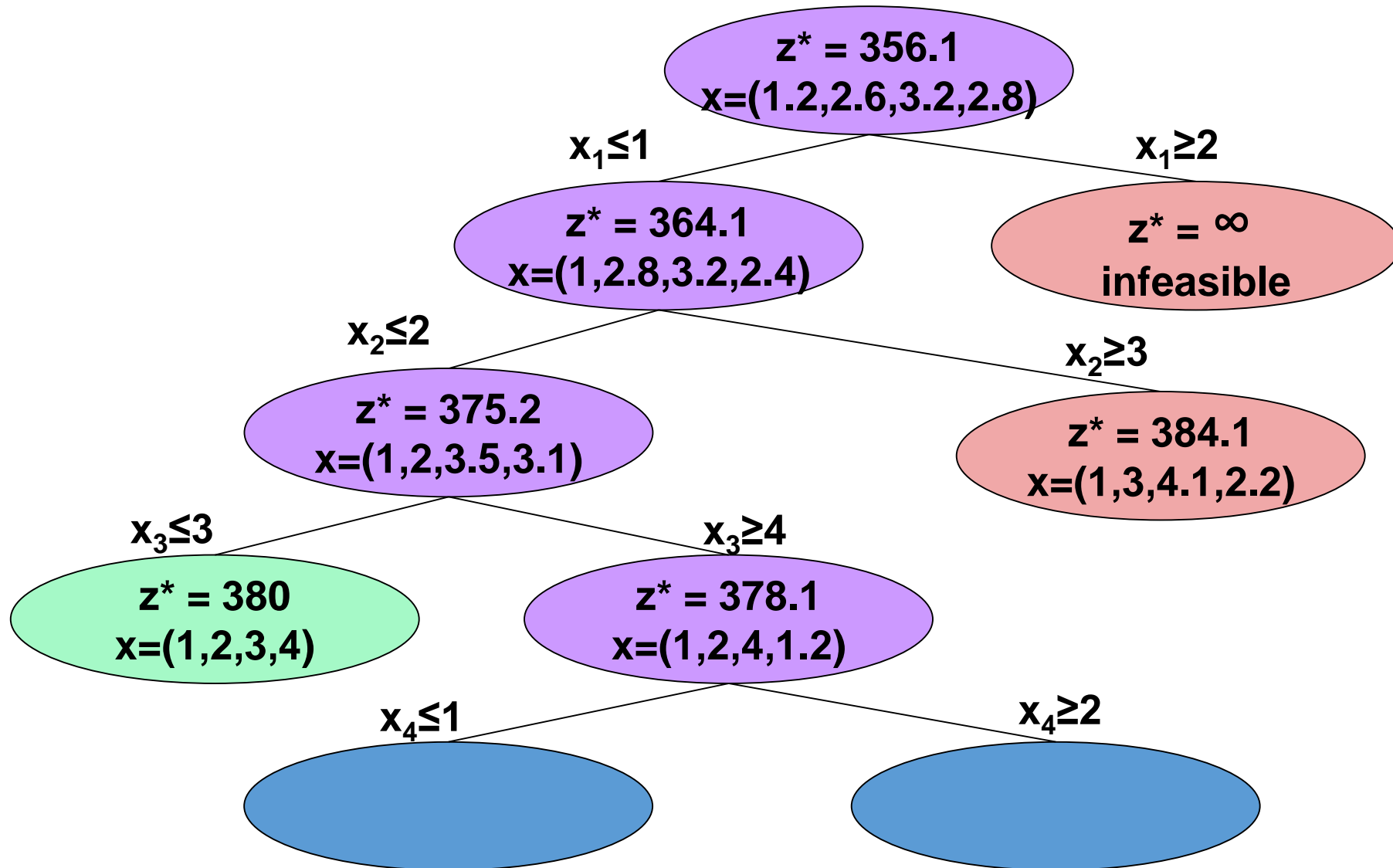


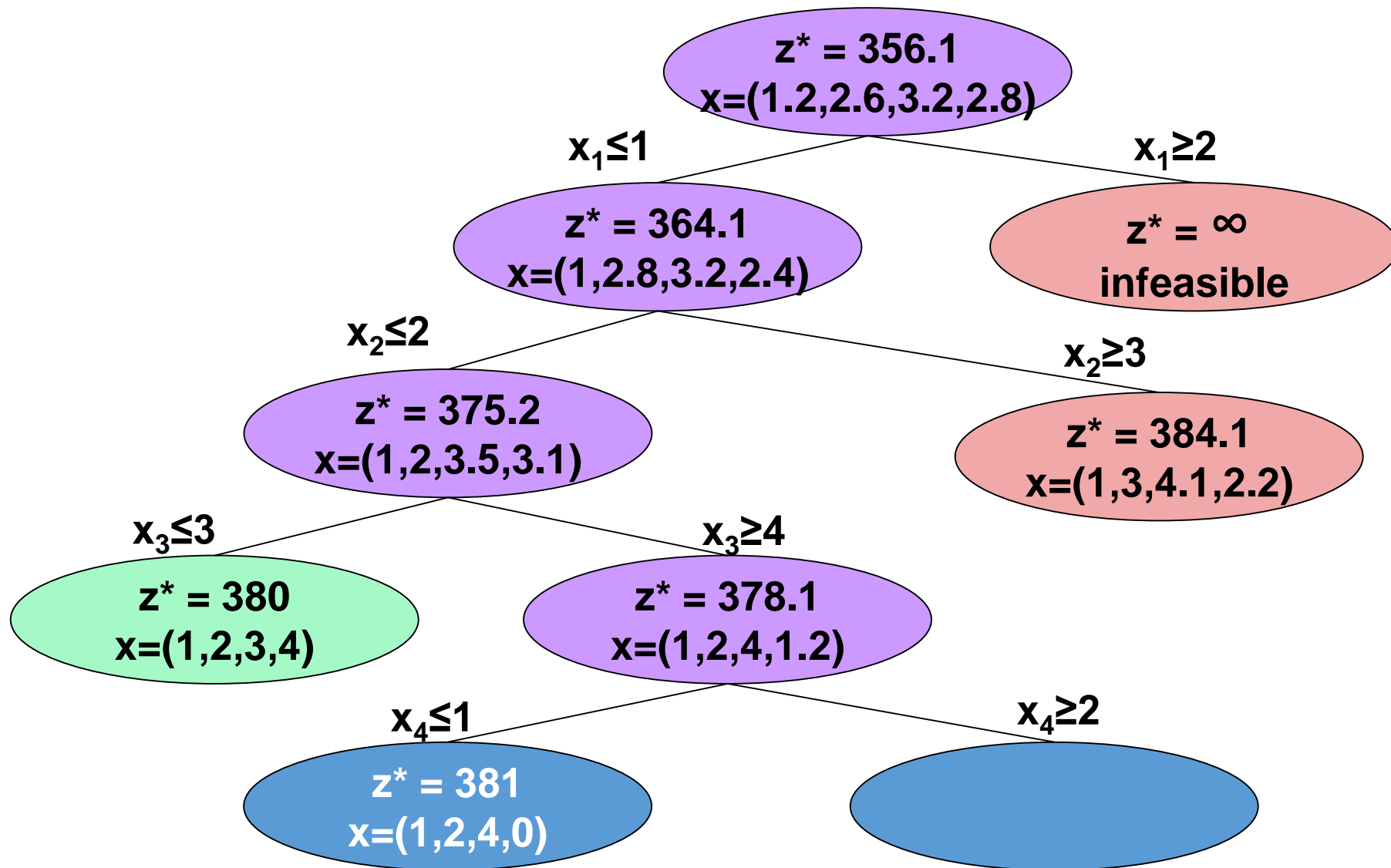




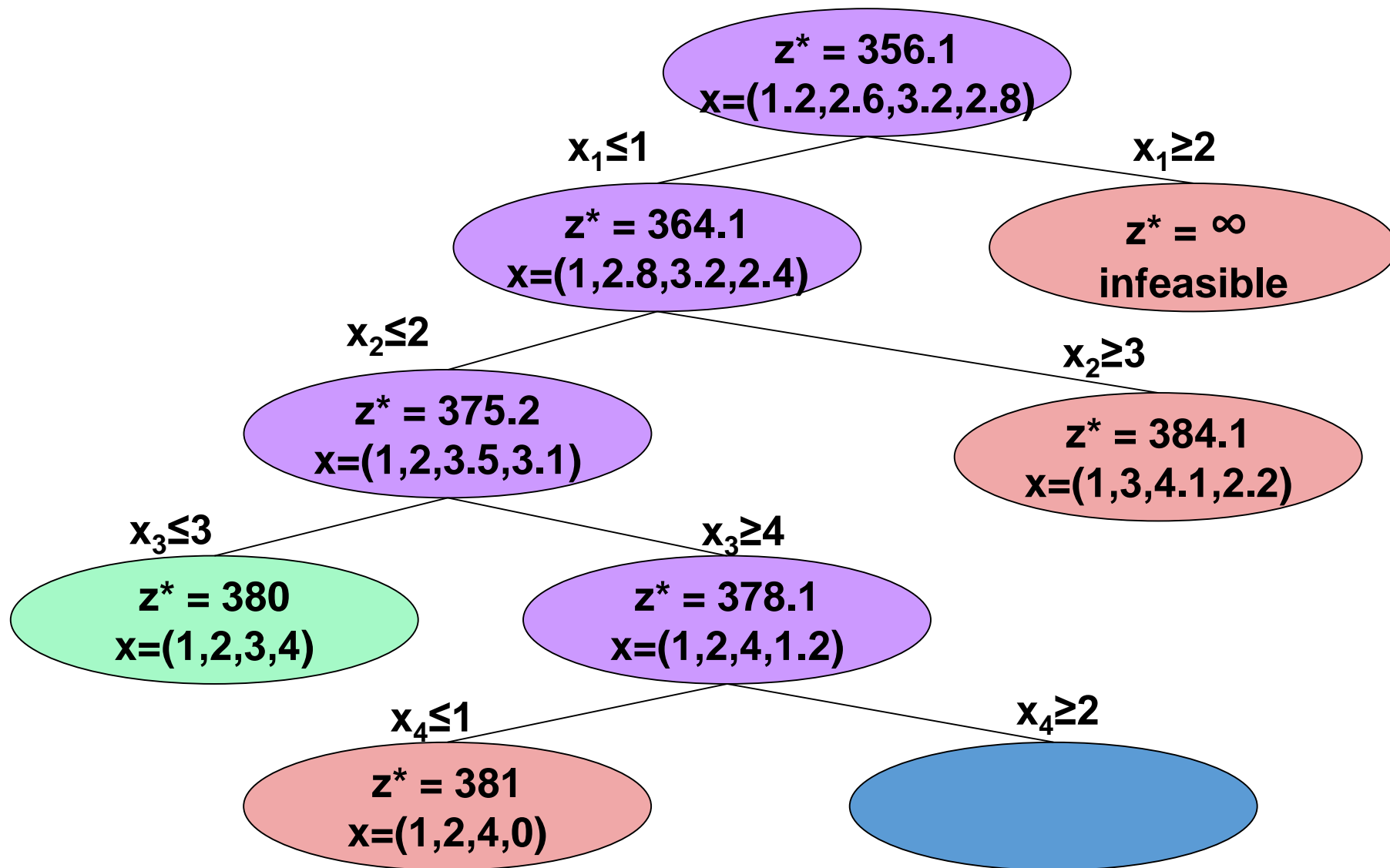


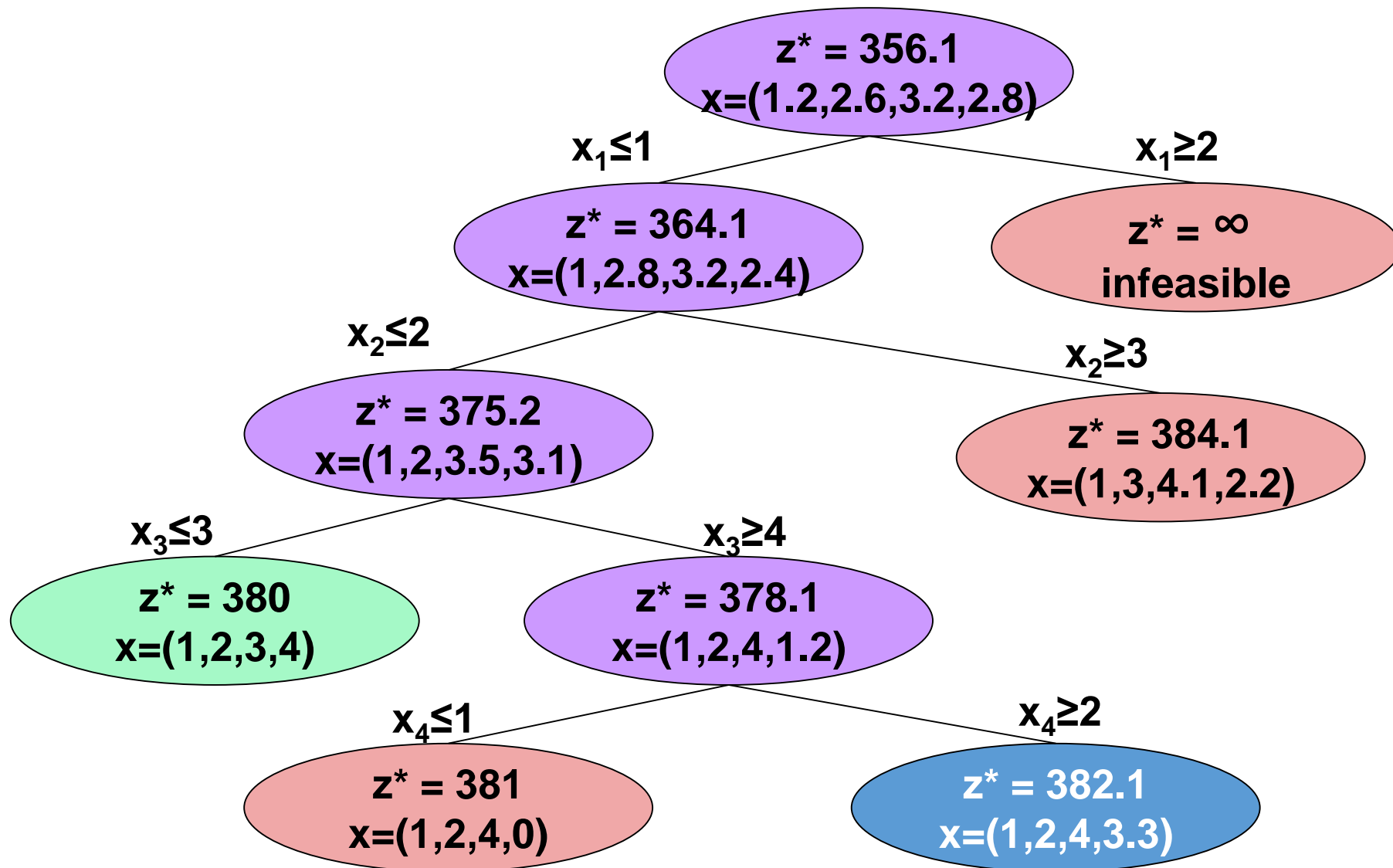


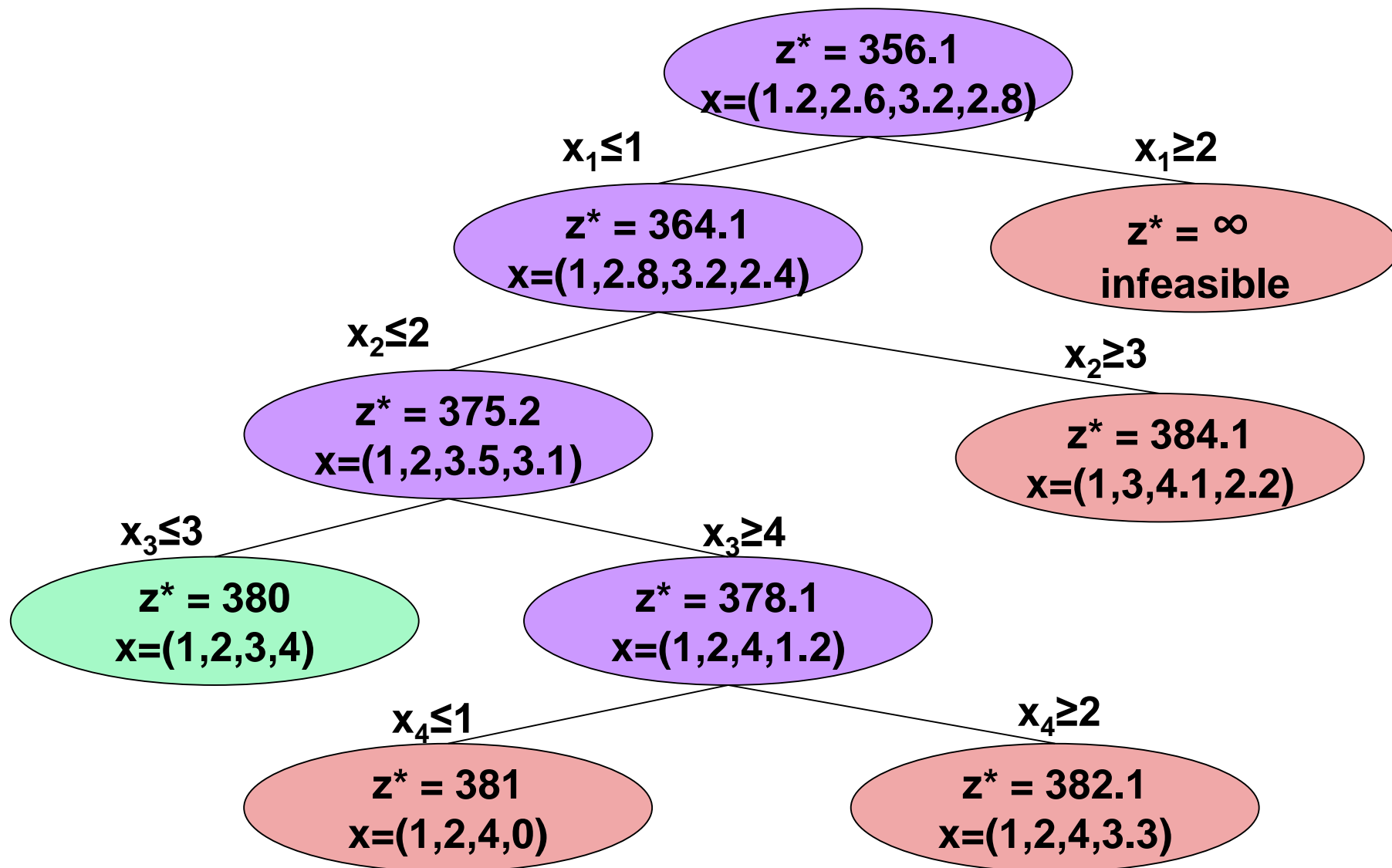












# INTEGER PROGRAMMING: FORMULATIONS AND VALID INEQUALITIES

# solving an IP: complete enumeration

Suppose that we could evaluate **1 trillion** solutions per second, and instantaneously eliminate about **99.99999999%** of all solutions as not worth considering

Let  $n$  = number of binary variables

Problem size	Approximate solution times
$n = 70$	1 second



# B&B plus...

- The basic branch-and-bound algorithm certainly provides a vast improvement on the complete enumeration tree for an IP problem
- However, in practice we are still faced with an *enormous computational challenge!*
- There are few things we can do about this:
  - Branching strategies
  - Improve our IP formulation as best as possible (valid inequalities)
  - Employ “branch-and-cut” with Gomory Cuts
  - Cheat...

# importance of good IP formulation

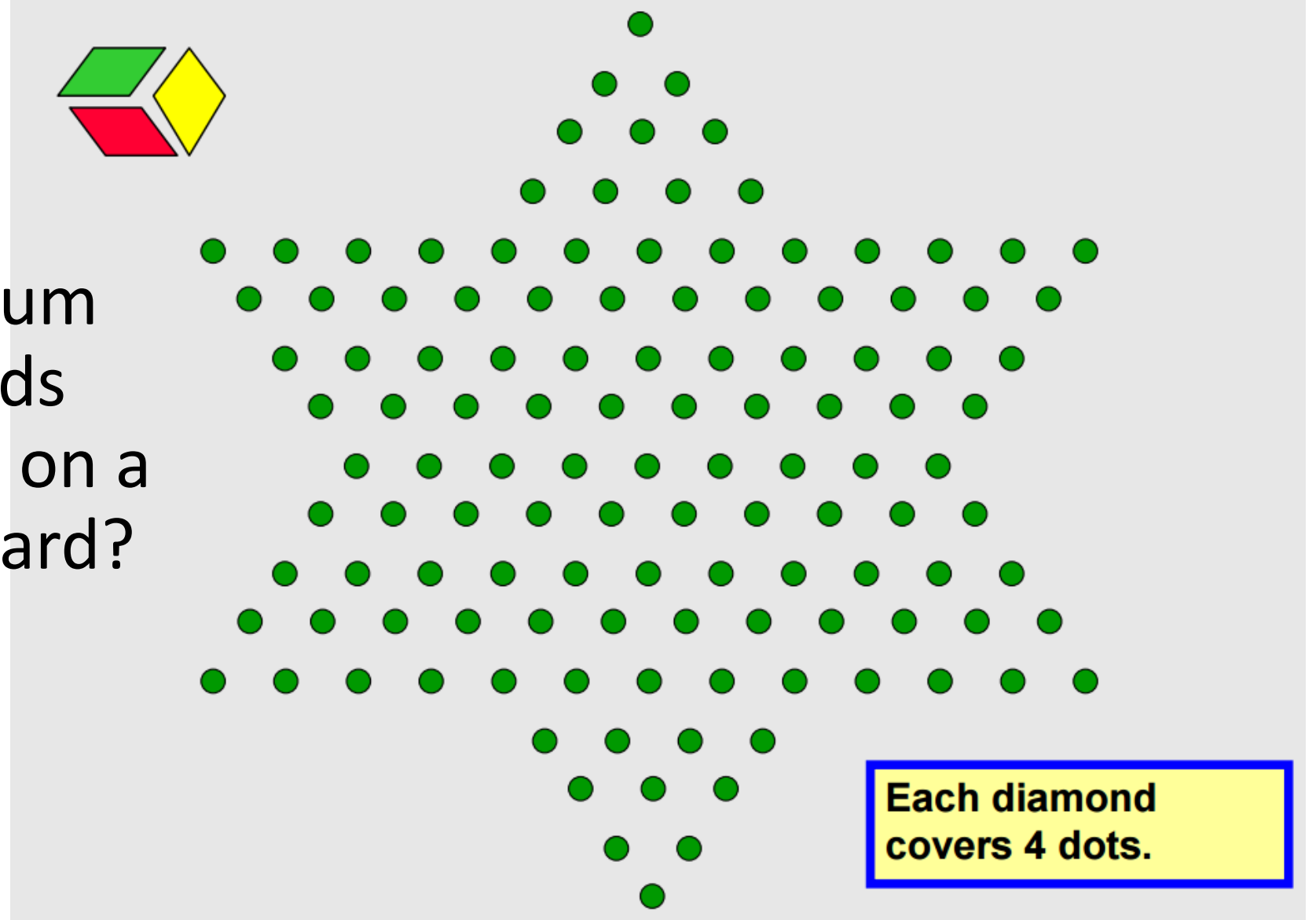
- Computational experiments suggest that the IP formulation *crucially influences* solution time and sometimes solvability
- To solve an IP efficiently, we want  $\{x : Ax \leq b\}$  to be close to the convex hull of the feasible integer solutions
  - Finding the exact convex hull is not usually possible\*
- A more tractable task: introduce ***valid inequalities***
  - *Valid inequality for an IP*: any constraint that does not eliminate any feasible integer solutions
  - Valid inequalities are also called ***cutting planes*** or simply ***cuts***

\*unless you can formulate the problem as a minimum cost network flow problem!

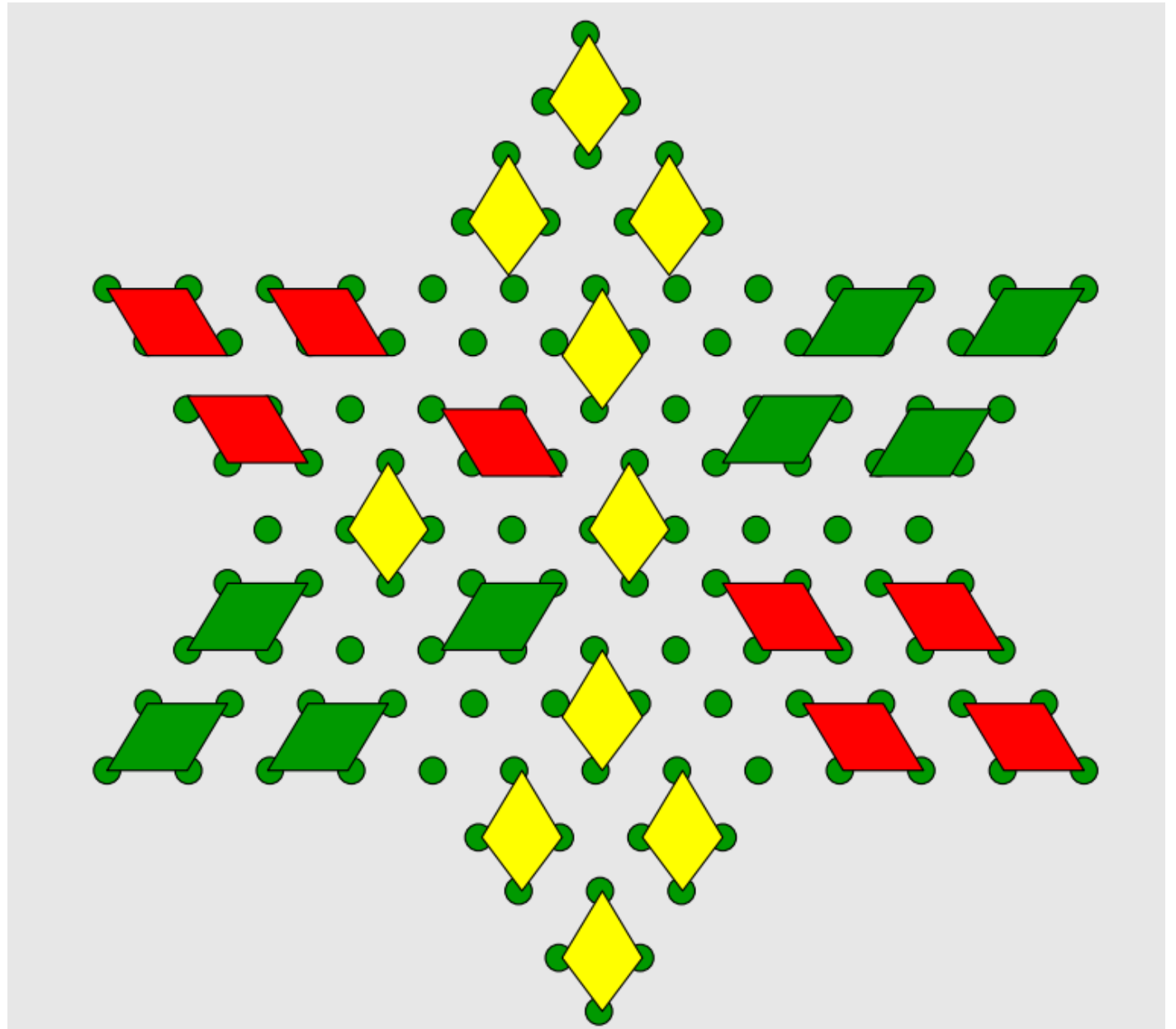


# IP formulation example\* with valid inequalities

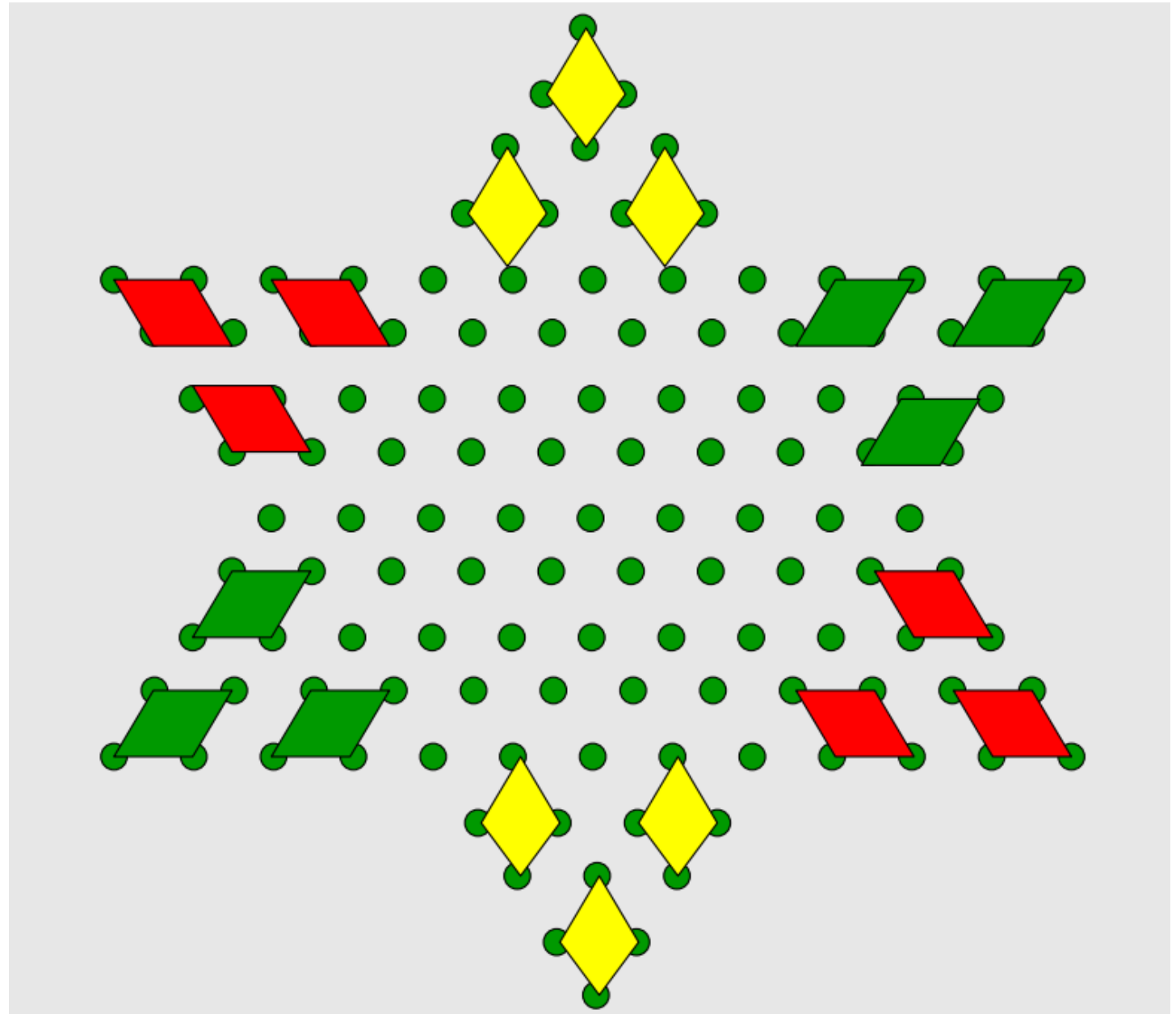
What is the maximum number of diamonds that can be packed on a Chinese checkerboard?



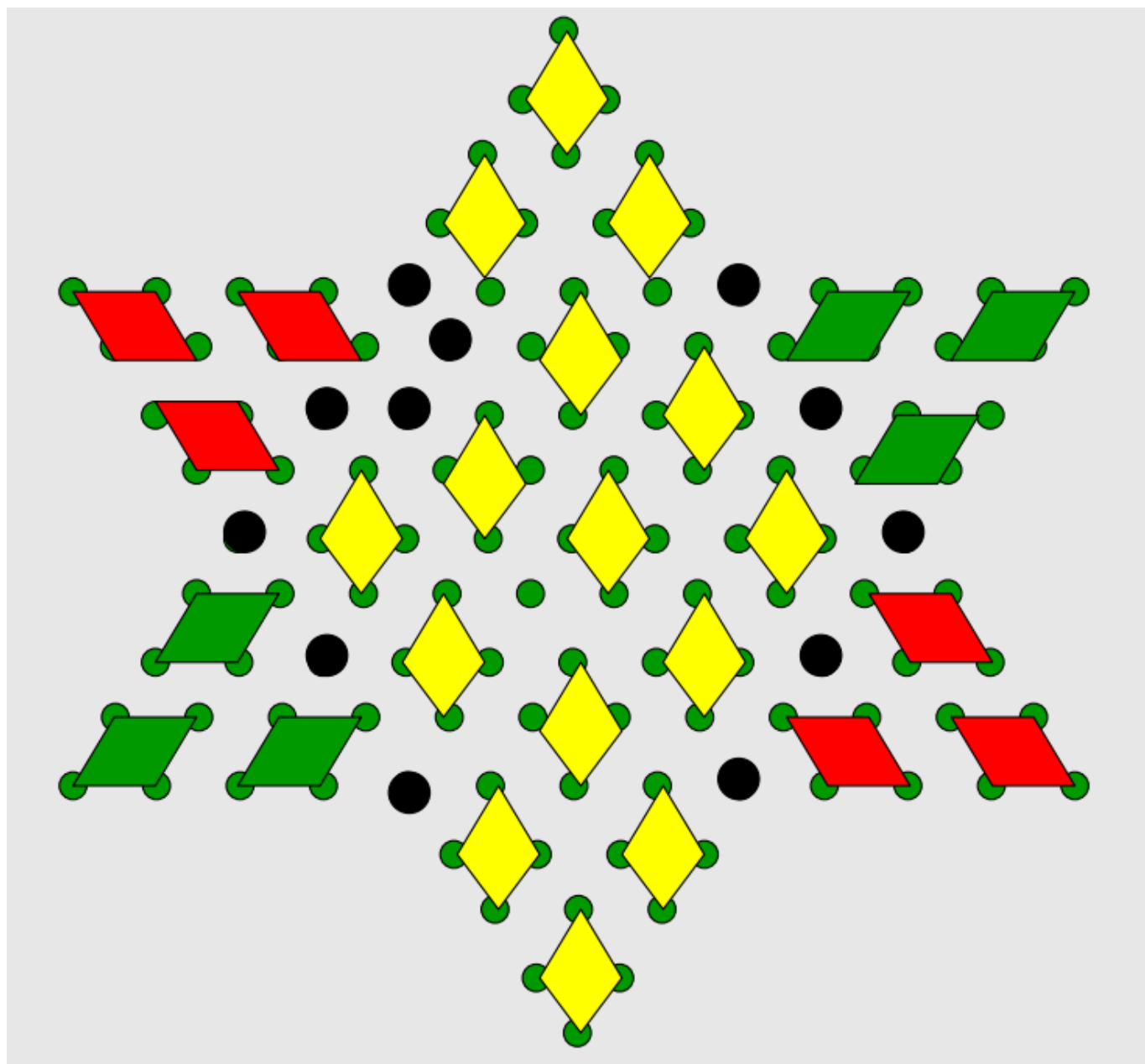
The diamonds are not permitted to overlap, or even to share a single circle.



What is the best  
packing you can find?



Here is one optimal  
packing solution.



# set packing problem

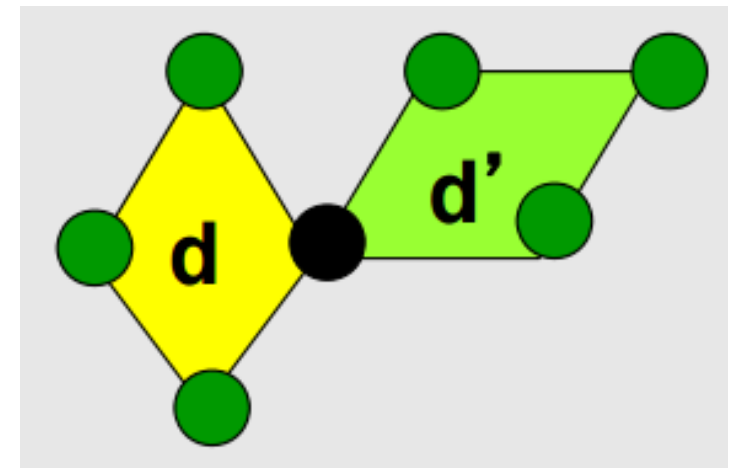
- Let  $D$  be the collection of diamonds

(note: there are 264 possible diamonds, see next slide)

- Decision variables:  $x_d \in \{0,1\} \forall d \in D$

$$x_d = \begin{cases} 1, & \text{if diamond } d \text{ is selected} \\ 0, & \text{if diamond } d \text{ is not selected} \end{cases}$$

- Need to ensure no overlap
- Thus, we want:  $x_d + x_{d'} \leq 1$  for all possible diamond pairs  $d$  and  $d'$  that have at least one point in common
- Let the set  $O$  denote all such pairs



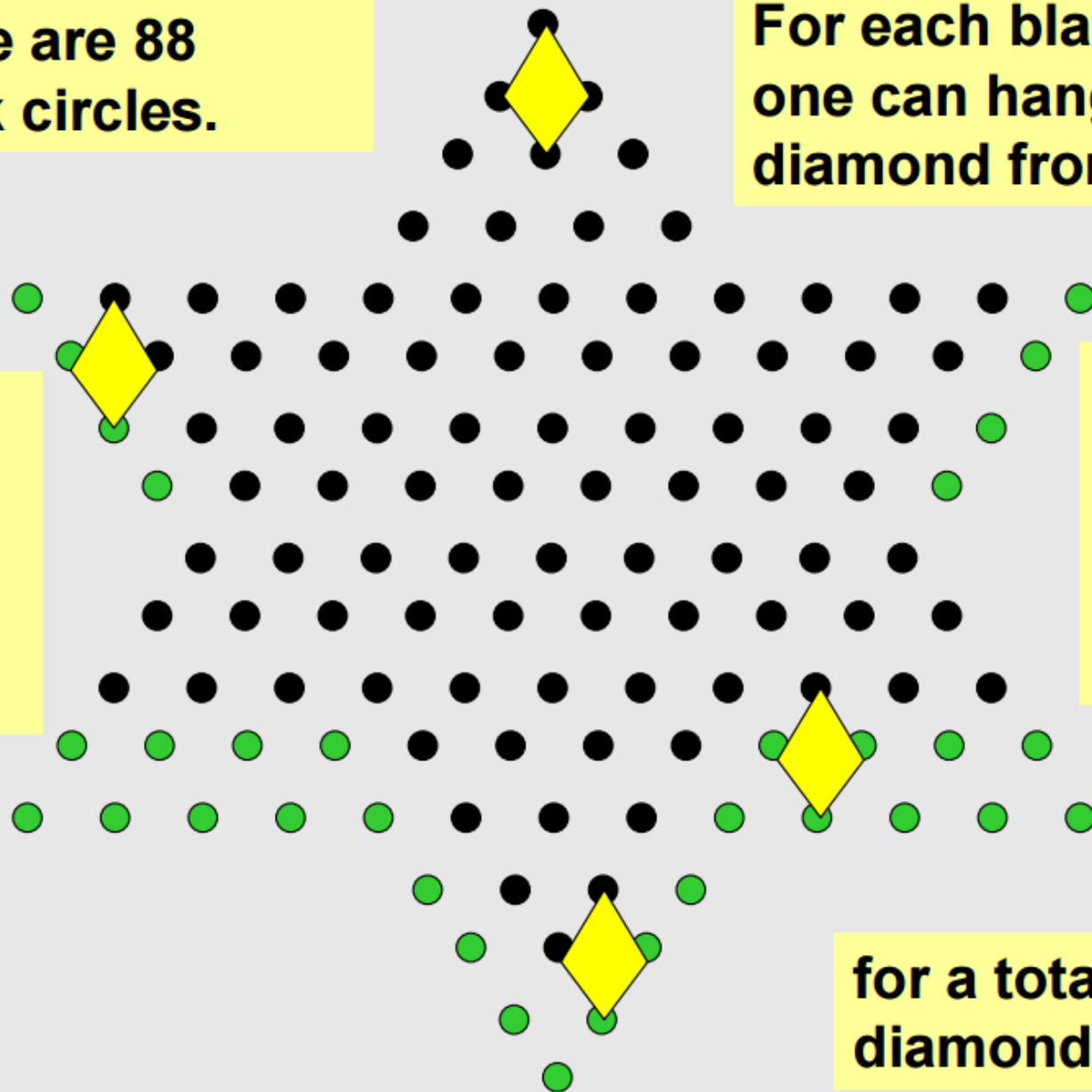
**There are 88  
black circles.**

**For each black circle,  
one can hang a yellow  
diamond from it.**

**So, there  
are 88  
possible  
yellow  
diamonds.**

**And there  
are 88  
green and  
red  
diamonds.**

**for a total of 264  
diamonds.**



# set packing problem: first IP formulation

$$\begin{aligned} \max \quad & \sum_{d \in D} x_d \\ \text{s.t.} \quad & x_d + x_{d'} \leq 1 \quad \forall (d, d') \in O \\ & x_d \in \{0, 1\} \quad \forall d \in D \end{aligned}$$

**This formulation is terrible for B&B!**

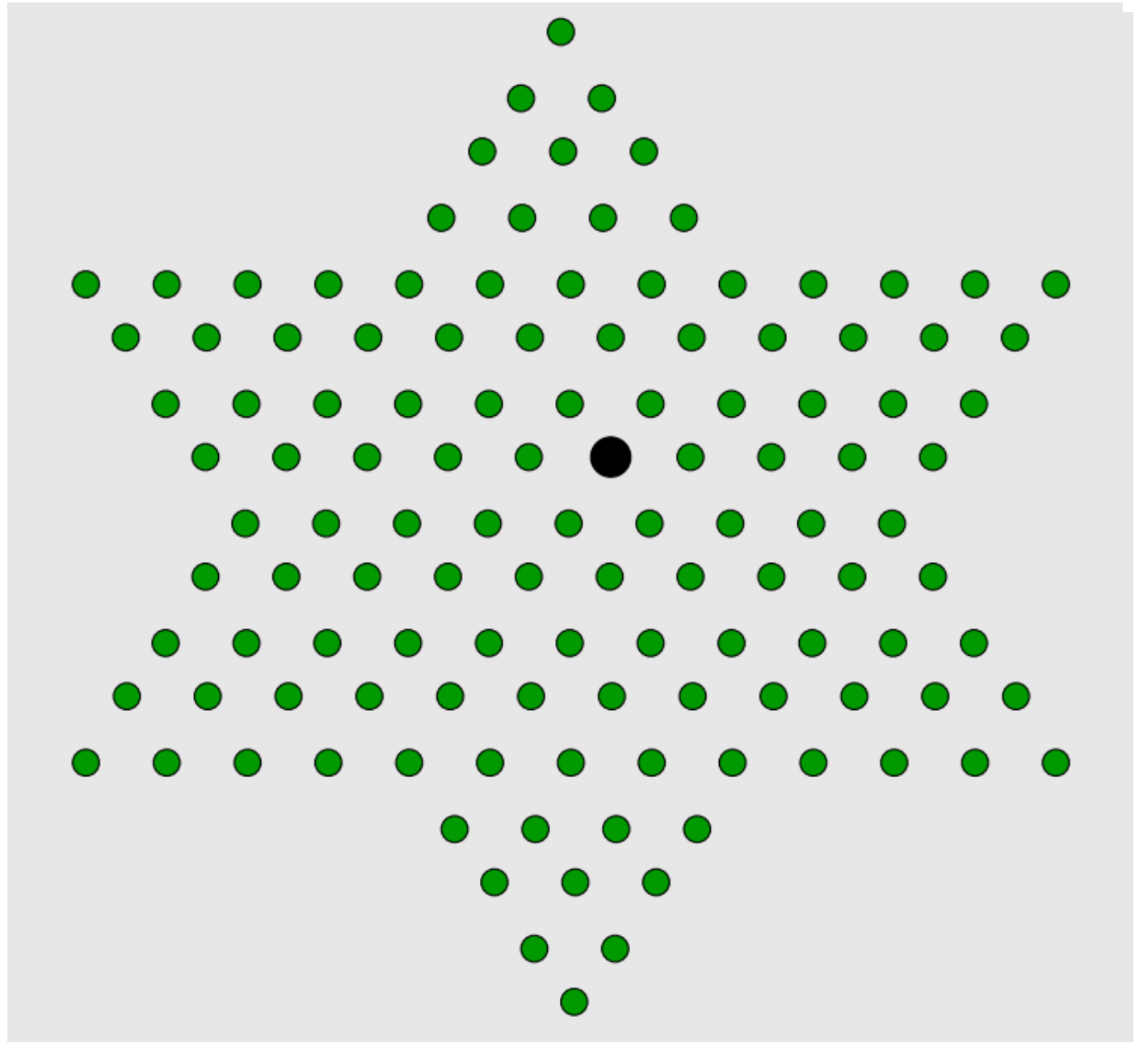
$z_{\text{IP}}^* = 27$ , but  $z_{\text{LP}}^* = 132$

and the LP optimum solution:  $x_d = 1/2 \quad \forall d \in D$

B&B would take much more than **3 billion years** to solve this problem on the fastest computer unless it can add valid inequalities!

## An improved IP formulation

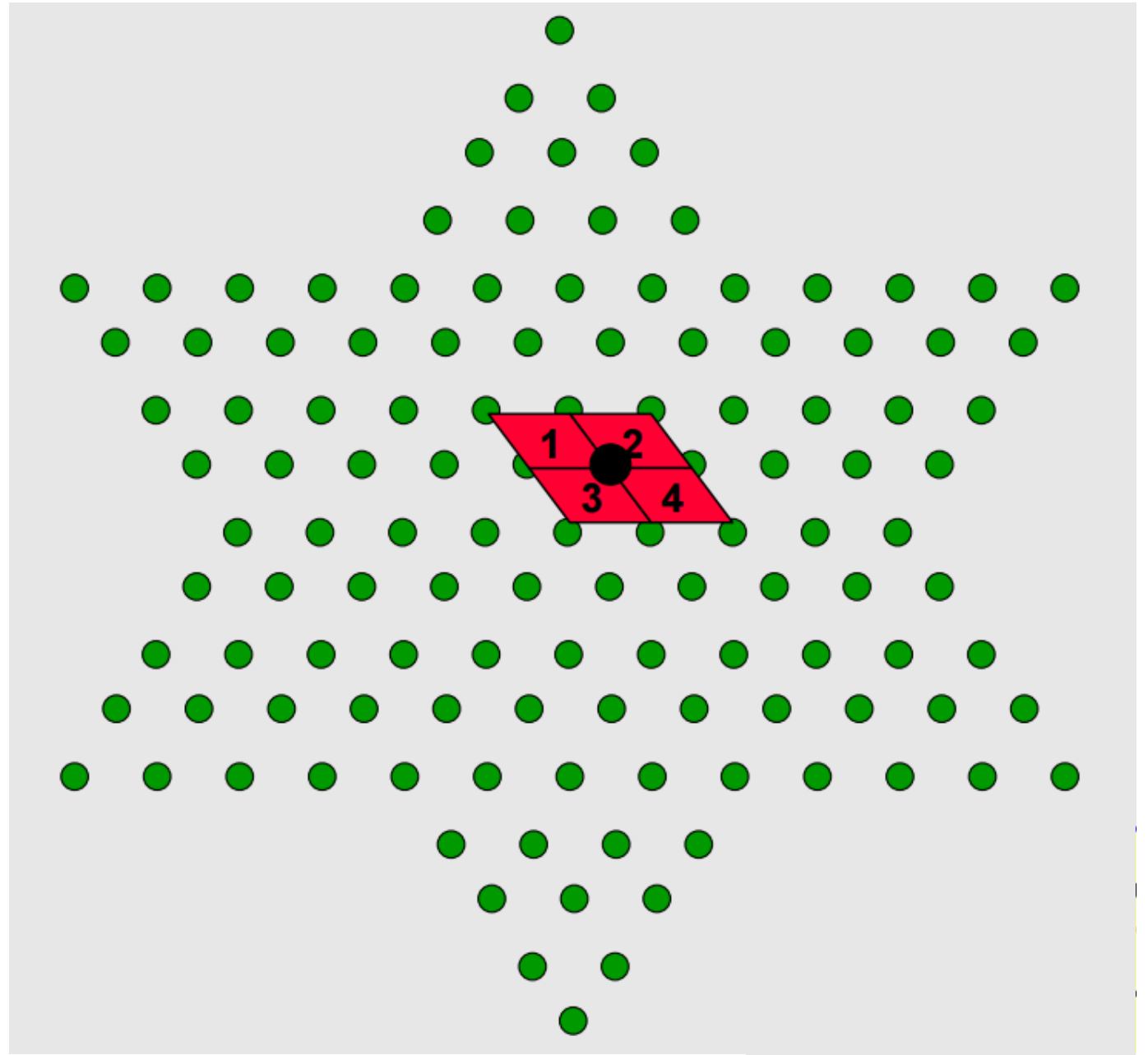
For every dot, choose at most one of the diamonds containing the dot.





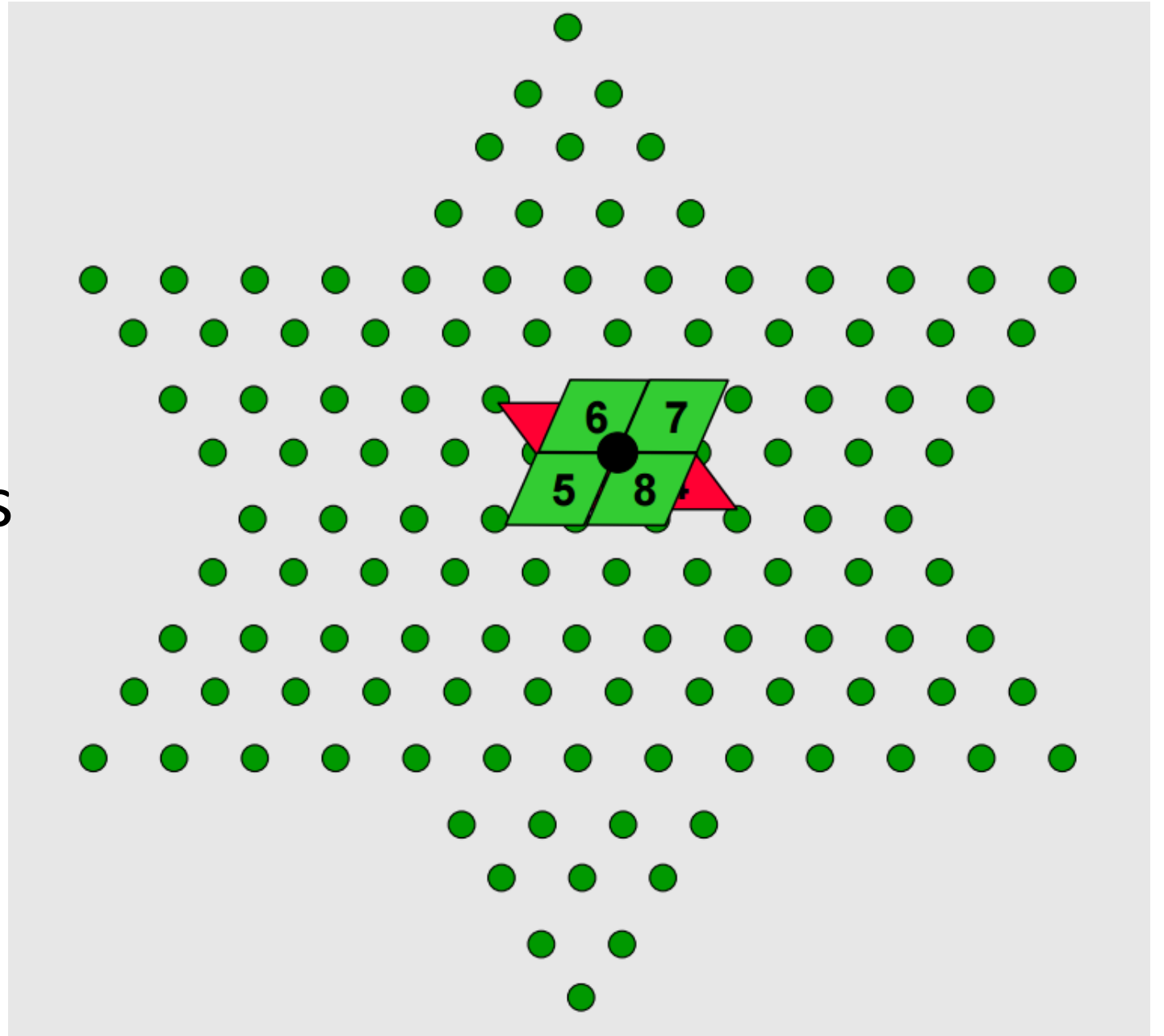
## An improved IP formulation

For every dot, choose at most one of the diamonds containing the dot.



## An improved IP formulation

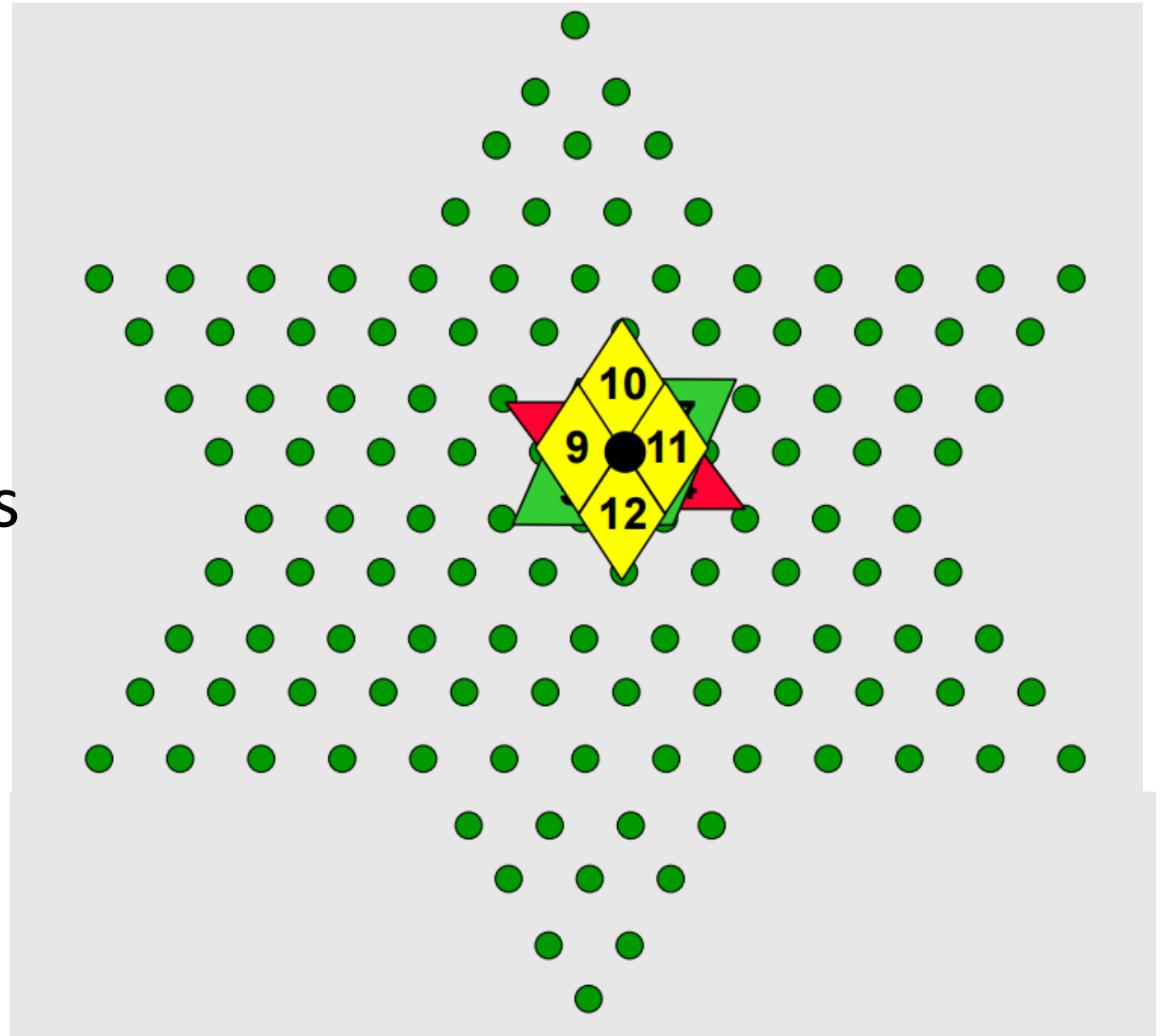
For every dot, choose at most one of the diamonds containing the dot.



## An improved IP formulation

For every dot, choose at most one of the diamonds containing the dot.

At most 1 of the 12 diamonds may be selected.



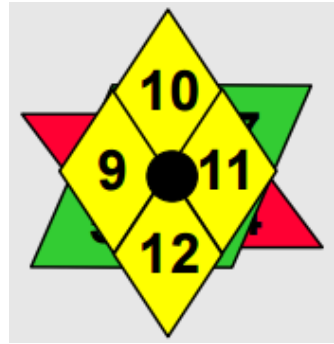
# set packing problem: an improved IP constraint

Let  $D(c)$  be the set of diamonds that include circle  $c$ .

$$\sum_{d \in D(c)} x_d \leq 1 \quad \text{for each circle } c$$

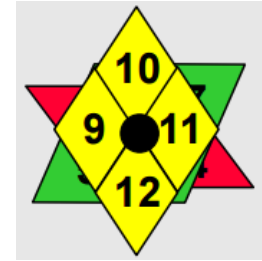
Example constraint for one circle (in the interior):

$$x_1 + x_2 + x_3 + \dots + x_{12} \leq 1$$



# set packing problem: an improved IP constraint

$$x_1 + x_2 + x_3 + \dots + x_{12} \leq 1$$



Note:  $x_j = 1/12$  would be LP feasible, but  $x_j = 1/2$  would not be feasible for all such circles!

In this constraint, we've combined 66 different constraints from our previous formulation:

$x_1 + x_2 \leq 1$	$x_1 + x_3 \leq 1$	$x_1 + x_4 \leq 1$
$x_1 + x_5 \leq 1$	$x_1 + x_6 \leq 1$	$x_1 + x_7 \leq 1$
$x_1 + x_8 \leq 1$	$\dots$	$x_{11} + x_{12} \leq 1$

# set packing problem: an improved IP formulation

$$\begin{aligned} \max \quad & \sum_{d \in D} x_d \\ \text{s.t.} \quad & \sum_{d \in D(c)} x_d \leq 1 \quad \text{for each circle } c \\ & x_d \in \{0, 1\} \quad \forall d \in D \end{aligned}$$

$$\mathbf{z}_{\text{LP}}^* = 27.5 \rightarrow \mathbf{z}_{\text{IP}}^* = 27$$

B&B solution time: **0.001 seconds!** (which, FYI, is considerably better than 3 billion years)

# valid inequalities question

$$\begin{aligned} \max \quad & 22x_1 + 19x_2 + 16x_3 + 12x_4 + 11x_5 + 8x_6 \\ \text{s.t.} \quad & 7x_1 + 6x_2 + 5x_3 + 4x_4 + 4x_5 + 3x_6 \leq 14 \\ & x_j \in \{0, 1\} \quad \text{for } j = 1, \dots, 6 \end{aligned}$$

1. Why can at most two of  $x_1, x_2, x_3$  be set to 1?
2. How can you write this as a valid inequality?
3. What are some more valid inequalities you could come up with?

# valid inequalities question

$$\begin{aligned} \max \quad & 22x_1 + 19x_2 + 16x_3 + 12x_4 + 11x_5 + 8x_6 \\ \text{s.t.} \quad & 7x_1 + 6x_2 + 5x_3 + 4x_4 + 4x_5 + 3x_6 \leq 14 \\ & x_j \in \{0, 1\} \quad \text{for } j = 1, \dots, 6 \end{aligned}$$

## Some valid inequalities:

$$\begin{array}{lll} x_1 + x_2 + x_3 \leq 2 & x_1 + x_2 + x_5 \leq 2 & x_1 + x_3 + x_4 \leq 2 \\ x_1 + x_2 + x_4 \leq 2 & x_1 + x_2 + x_6 \leq 2 & \text{etc.} \end{array}$$

**A really good constraint:**  $x_1 + x_2 + x_3 + x_4 \leq 2$



# integer programming and B&B summary

- **IP dramatically improves the modeling capability**
  - yes/no decisions, contingent decisions
  - logical constraints
  - fixed costs and piecewise linear cost functions
- **Not as easy to model, and not as easy to solve**
- **Branch and Bound**
  - general technique based on divide and conquer
  - “implicit enumeration”
- **Cutting planes (valid inequalities)**
  - used to help improve B&B
  - there are “automatic” and “logical” ways of implementing
  - Key takeaway: a good IP formulation is crucial for practical efficiency