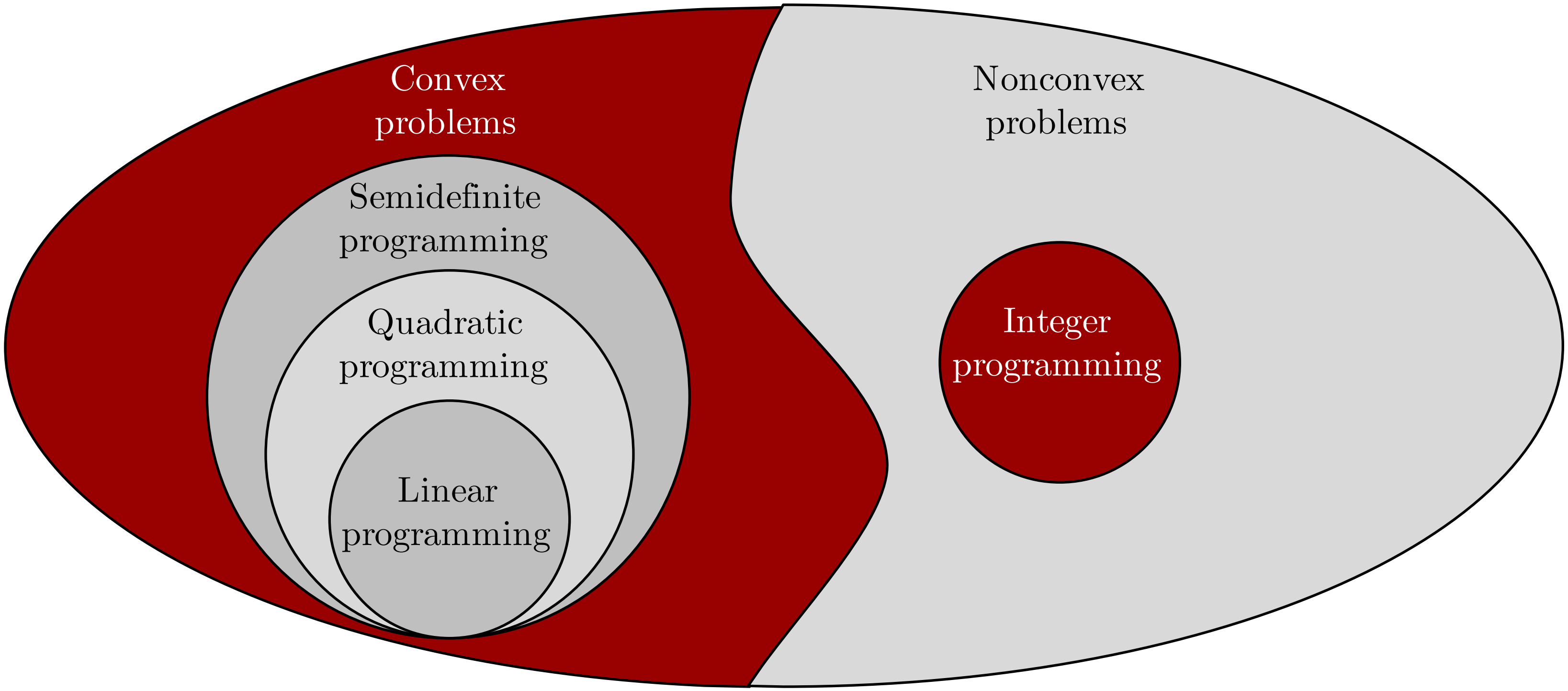
CMU 15-781

Lecture 14: Integer programming

Teacher:
Gianni A. Di Caro

The optimization universe



(Linear) Integer Programming: Feasibility problem

- An integer programming (IP) **feasibility problem**:
  - $a_{ij} \in \mathbb{R}$ for $i \in [k] = \{1, \dots, k\}$, $j \in [\ell] = \{1, \dots, \ell\}$
  - $b_i \in \mathbb{R}$ for $i \in [k]$
  - Decision variables x_j for $j \in [\ell]$

$$
\begin{aligned}
&\text{find } x_1 \dots, x_\ell \\
&\text{s.t. } \forall i \in [k], \quad \textstyle\sum_{j=1}^{\ell} a_{ij} x_j \le b_i \\
&\qquad \forall j \in [\ell], \; x_j \in \mathbb{Z}
\end{aligned}
$$

$$
\begin{aligned}
&\text{find } \boldsymbol{x} \\
&\text{s.t. } A\boldsymbol{x} \le \boldsymbol{b} \\
&\qquad \boldsymbol{x} \in \mathbb{Z}^\ell \\
&A \in \mathbb{R}^{k \times \ell}, \boldsymbol{b} \in \mathbb{R}^k
\end{aligned}
$$

(linear) Integer Programming: Optimization problem

- The canonical formulation optimizes a linear objective function $\boldsymbol{c}^T\boldsymbol{x}$ in the following problem form:

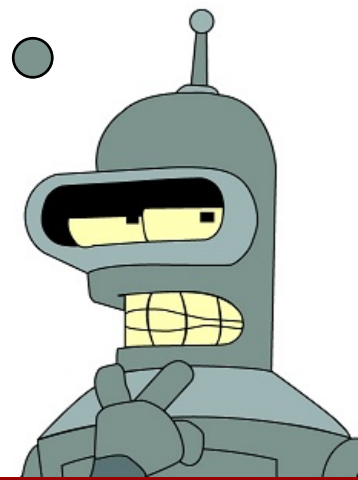
$$\begin{aligned} &\max \textstyle\sum_{j=1}^{\ell} c_j x_j \\ &\text{s.t.} \quad \forall i \in [k],\ \textstyle\sum_{j=1}^{\ell} a_{ij} x_j \le b_i \\ &\qquad\ \ \forall j \in [\ell],\ x_j \in \mathbb{N} \cup \{0\} \end{aligned}$$

Combinatorial Optimization problems (COPs)

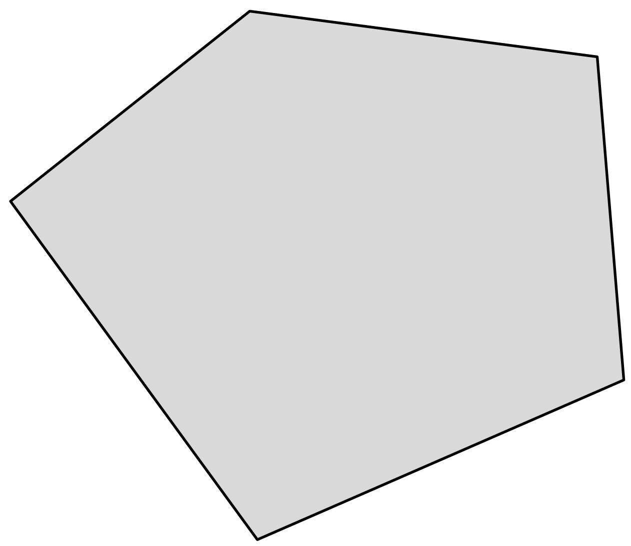
- A COP is an IP optimization problem in which we seek to find a solution in a finite set of solutions
- TSP, VRP, QAP, Set covering, Knapsack, …
- Max or min of an objective function
- A COP can be formulated as a **0-1 integer program,** $\boldsymbol{x} \in \{0,1\}^n$
- Any bounded integer, $0 \leq x \leq u$, can be converted to a set of 0-1 variables, $2^k \leq u \leq 2^{k+1}$
- E.g., $0 \leq x \leq 20$, $x = 2^0 y_0 + 2^1 y_1 + 2^2 y_2 + 2^3 y_3 + 2^4 y_4$

How can we express $\geq$ constraints?
Equality constraints?
Restricted domains?
Min?

$$\geq \to -\leq$$
$$Ax = b \;\leftrightarrow\; Ax \leq b, Ax \;\geq b$$
$$Ax \leq b \;\leftrightarrow\; Ax + s = b$$
$$Ax \geq b \;\leftrightarrow\; Ax - s = b$$
$$Max \;\leftrightarrow\; -Min$$

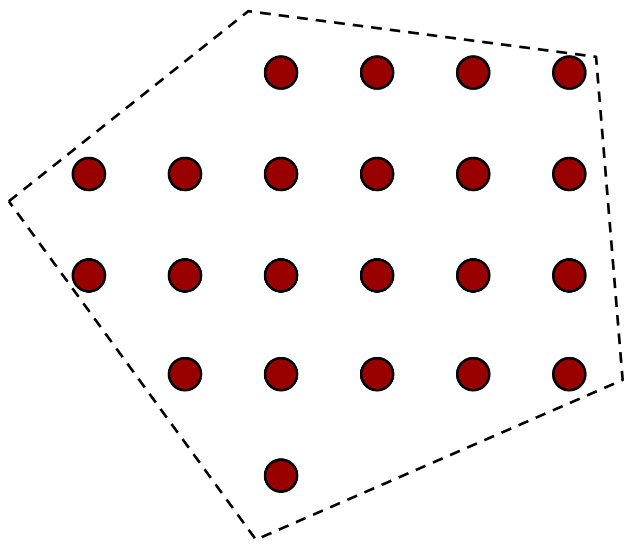


IP Is Not Convex



Linear (Convex) programming

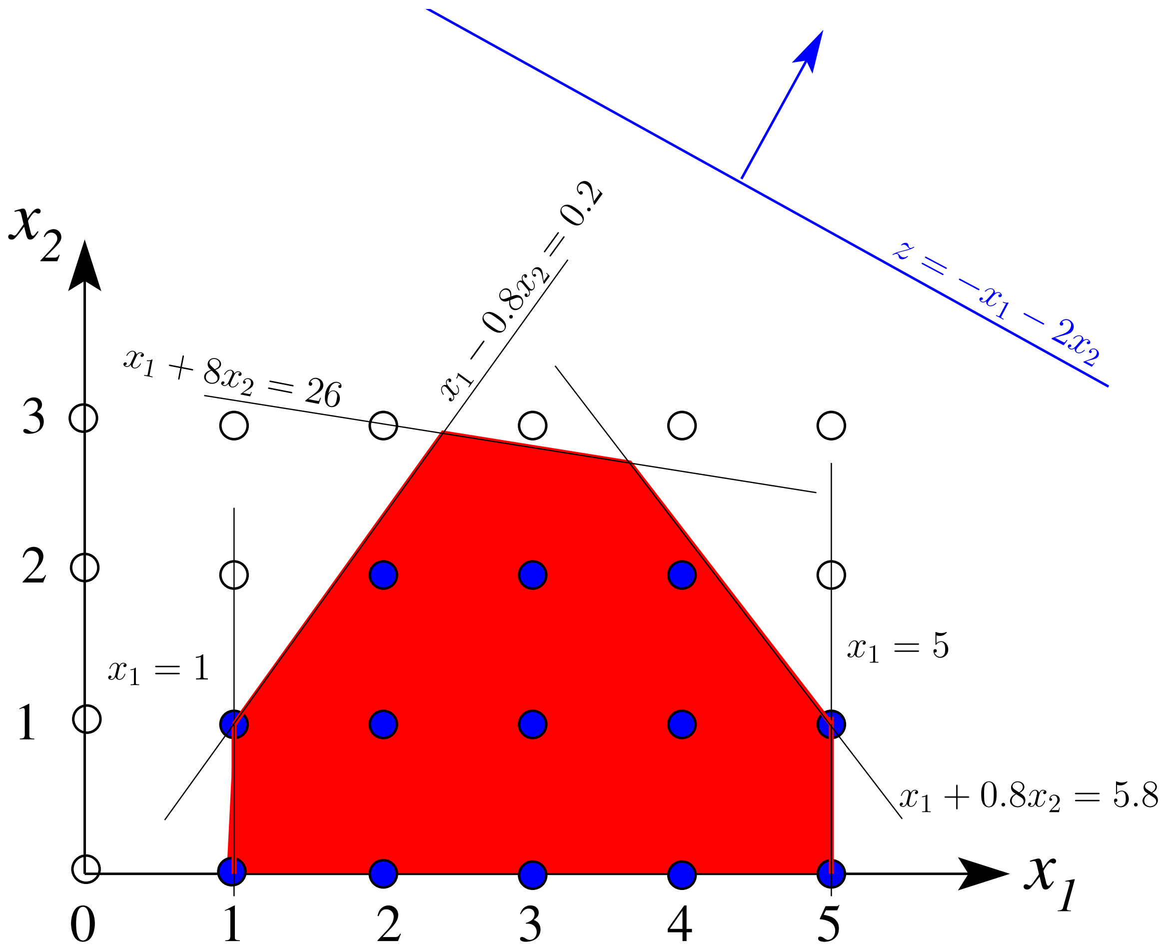
$$\mathcal{F} = \{\boldsymbol{x} \in \mathbb{R}^{\ell} : A\boldsymbol{x} \leq \boldsymbol{b}\}$$
$$A \in \mathbb{R}^{k \times \ell}, \boldsymbol{b} \in \mathbb{R}^{k}$$



Integer programming

$$\mathcal{F} = \{\boldsymbol{x} \in \mathbb{Z}^{\ell} : A\boldsymbol{x} \leq \boldsymbol{b}\}$$
$$A \in \mathbb{R}^{k \times \ell}, \boldsymbol{b} \in \mathbb{R}^{k}$$

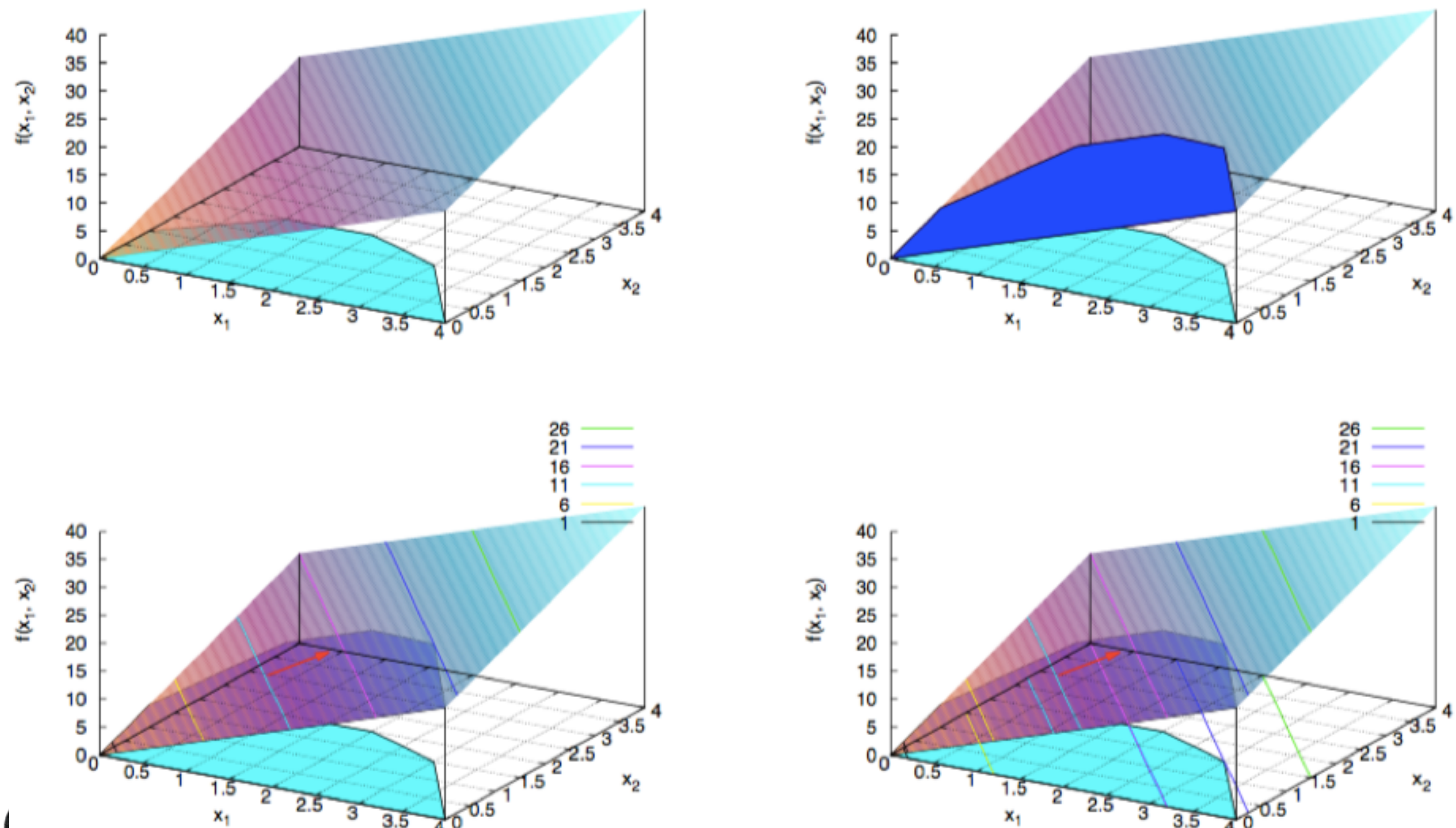
IP Is Not Convex



$$
\begin{array}{llcl}
\min & Z = -x_1 - 2x_2 & & \\
s.t. & x_1 & \geqslant & 1 \\
 & x_1 & \leqslant & 5 \\
 & x_1 + 0.8x_2 & \leqslant & 5.8 \\
 & x_1 - 0.8x_2 & \geqslant & 0.2 \\
 & x_1 + 8x_2 & \leqslant & 26 \\
 & x_1, x_2 \in X & &
\end{array}
$$

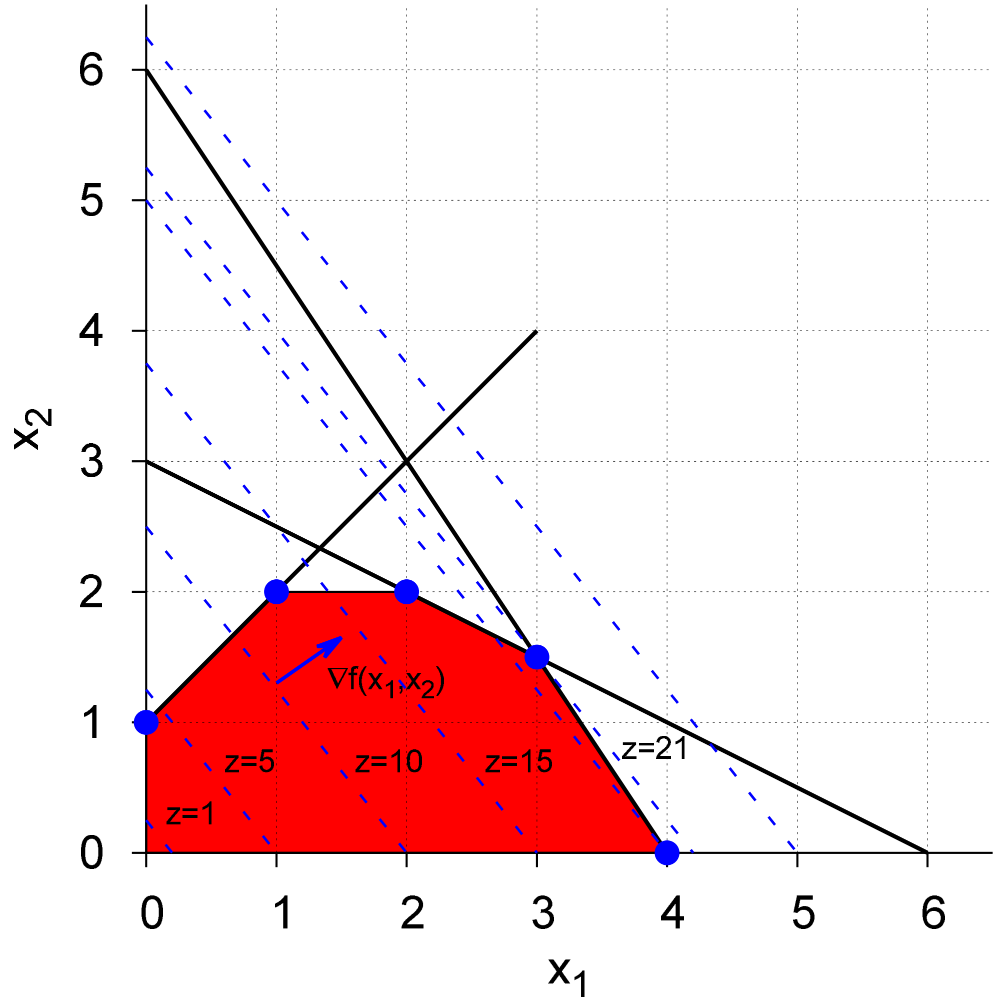
LP Can Exploit Convexity

max $z = f(x_1, x_2) = 5x_1 + 4x_2, \quad x_1, x_2 \in X =$ set of linear constraints

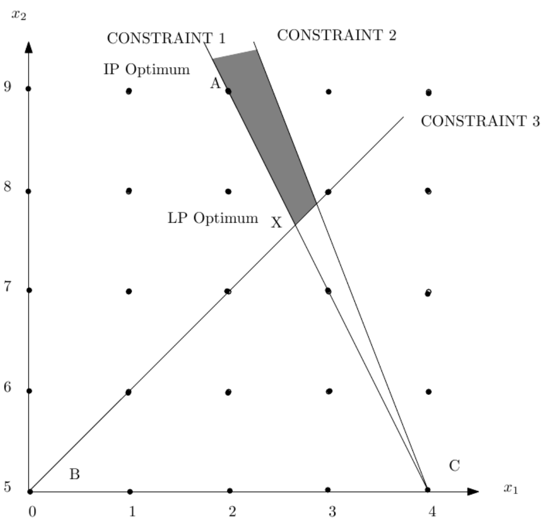


LP Can Exploit Convexity

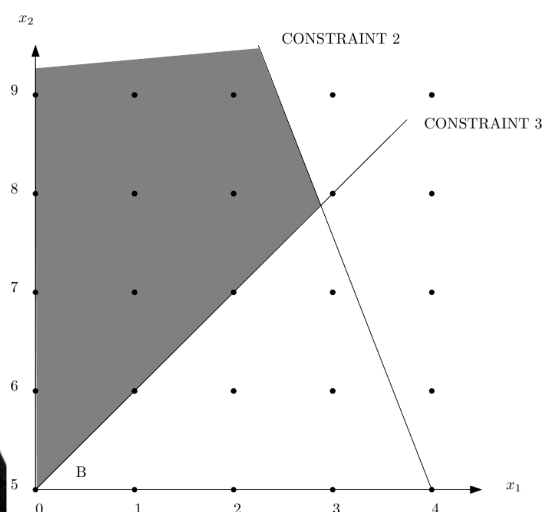
$$\begin{aligned}
\max \quad & z = f(x_1, x_2) = 5x_1 + 4x_2 \\
s.t. \quad & 6x_1 + 4x_2 \leq 24 \\
& x_1 + 2x_2 \leq 6 \\
& -x_1 + x_2 \leq 1 \\
& x_2 \leq 2 \\
& x_1, x_2 \geq 0
\end{aligned}$$

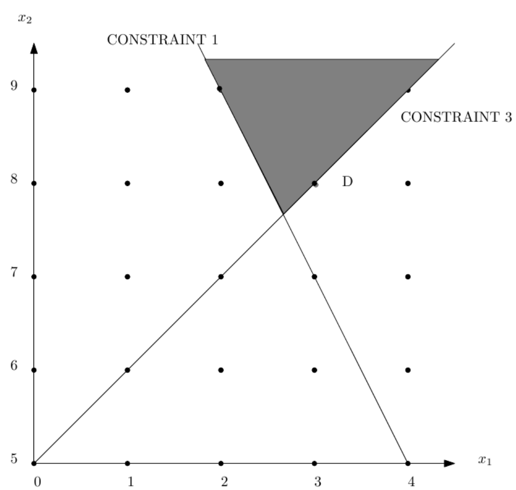


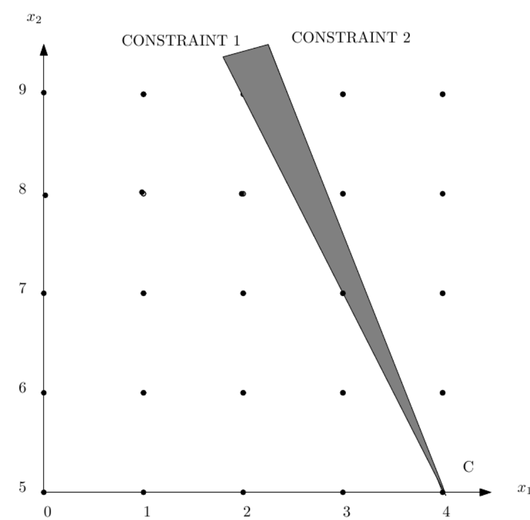
Geometry of IP



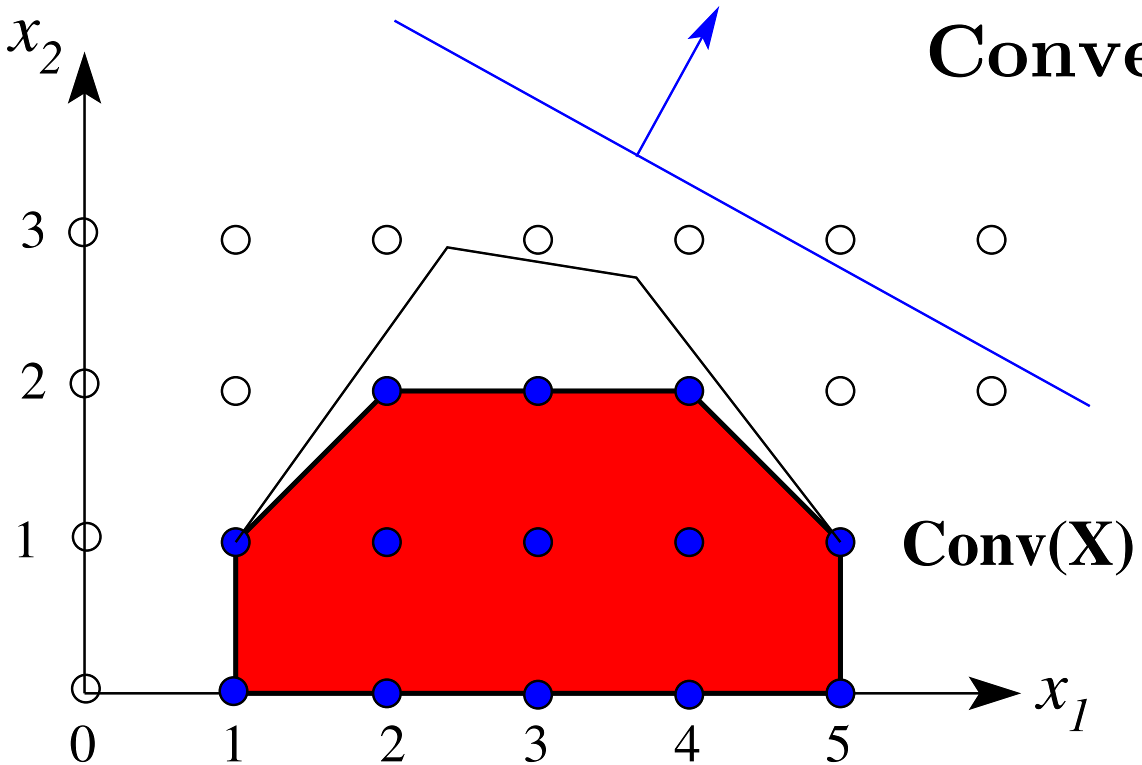
$$
\begin{aligned}
\min \quad & Z_{ILP} = x_2 \\
s.t. \quad & 2x_1 + \ \ x_2 \geqslant 13 \\
& 5x_1 + 2x_2 \leqslant 30 \\
& -x_1 + \ \ x_2 \geqslant 5 \\
& x_1,\ x_2 \in \mathbb{Z}^+
\end{aligned}
$$

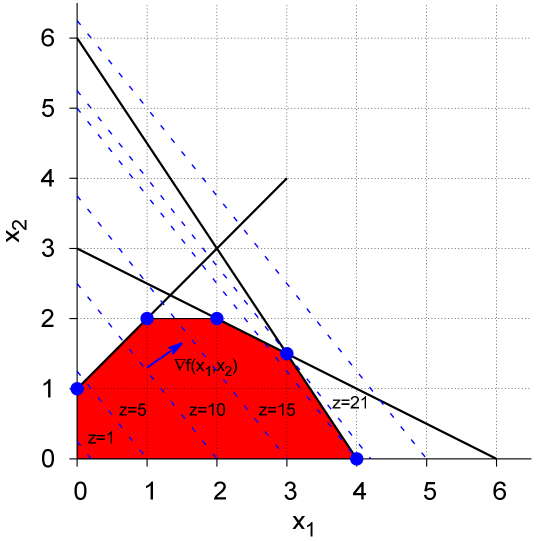






Tight formulations: IP can enjoy convexity!





Example: Sudoku

8			4		6			7
						4		
	1					6	5	
5		9		3		7	8	
				7				
	4	8		2		1		3
	5	2					9	
		1						
3			9		2			5

Example: Sudoku

$$x_{ijk} = \begin{cases} 1, & \text{if element } (i,j) \text{ of the } n \times n \text{ Sudoku matrix contains the integer } k \\ 0, & \text{otherwise.} \end{cases}$$

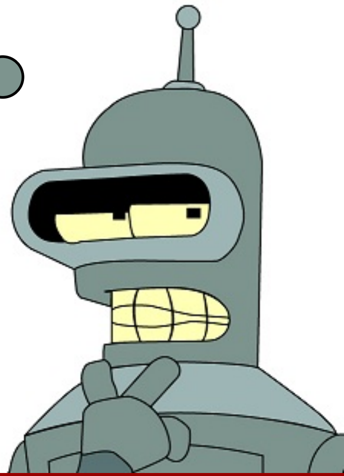
$$\begin{array}{lll}
\min & \mathbf{0}^T \mathbf{x} & \\
s.t. & \sum_{i=1}^{n} x_{ijk} = 1, \quad j{=}1{:}n,\ k{=}1{:}n & \text{(only one } k \text{ in each column)} \\
 & \sum_{j=1}^{n} x_{ijk} = 1, \quad i{=}1{:}n,\ k{=}1{:}n & \text{(only one } k \text{ in each row)} \\
 & \sum_{j=mq-m+1}^{mq} \sum_{i=mp-m+1}^{mp} x_{ijk} = 1, \quad k{=}1{:}n,\ p{=}1{:}m,\ q{=}1{:}m & \text{(only one } k \text{ in each submatrix)} \\
 & \sum_{k=1}^{n} x_{ijk} = 1 \quad i{=}1{:}n,\ j{=}1{:}n & \text{(every position in matrix must be filled)} \\
 & x_{ijk} = 1 \quad \forall\,(i,j,k) \in G & \text{(given elements } G \text{ in matrix are set “on”)} \\
 & x_{ijk} \in \{0,1\} &
\end{array}$$

Example: Sudoku (from Ariel)

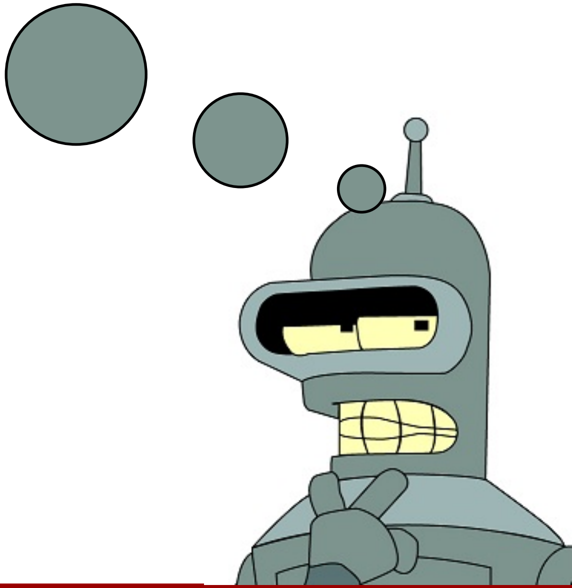
- For each $i, j, k \in [9]$, binary variable x_k^{ij} s.t. $x_k^{ij} = 1$ iff we put k in entry (i, j)
- For $\mathrm{t} = 1, \ldots, 27$, S_t is a row, column, or 3×3 square

$$
\begin{array}{ll}
\text{find} & x_1^{11}, \ldots, x_9^{99} \\
\text{s.t.} & \forall t \in [27], \forall k \in [9], \sum_{(i,j)\in S_t} x_k^{ij} = 1 \\
 & \forall i, j \in [9], \sum_{k\in[9]} x_k^{ij} = 1 \\
 & \forall i, j, k \in [9], x_k^{ij} \in \{0,1\}
\end{array}
$$

If you have a hard time expressing something as an IP, try using binary variables

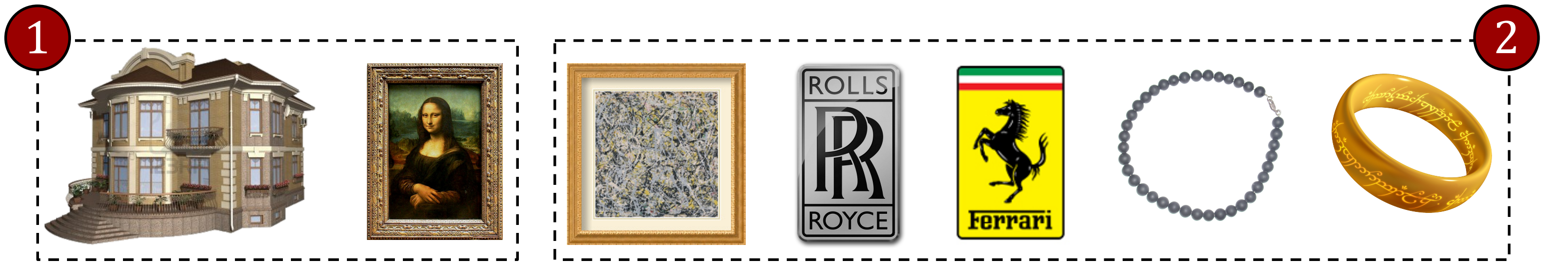


SUDOKU is NP-complete, so we "proved" that IP feasibility is NP-complete!



Example: Fair Division

- Players $P = \{1, \dots, n\}$ and items $I = \{1, \dots, m\}$
- Player p has value v_{pi} for item i
- Partition items to bundles $A_1, \dots, A_n$
- $A_1, \dots, A_n$ is envy-free iff $\forall p, p', \sum_{i \in A_p} v_{pi} \geq \sum_{i \in A_{p'}} v_{pi}$



1	\$30	\$50	\$2	\$5	\$5	\$3	\$5
2	\$2	\$10	\$5	\$20	\$20	\$3	\$40

Example: Fair Division

- Variables: $x_{pi} \in \{0,1\}$, $x_{pi} = 1$ iff i $\in A_p$
- Envy-Free as an IP:

$$
\begin{aligned}
&\text{find } x_{11}, \ldots, x_{nm} \\
&\text{s.t.} \quad \forall p \in N, \forall p' \in N, \textstyle\sum_{i \in M} v_{pi} x_{pi} \geq \sum_{i \in M} v_{pi} x_{p'i} \\
&\qquad \forall i \in M, \ \textstyle\sum_{p \in N} x_{pi} = 1 \\
&\qquad \forall p \in N, i \in M, x_{pi} \in \{0,1\}
\end{aligned}
$$

(Ariel) Application: Spliddit

Phase Transition

- Imagine the v_{pi} are drawn independently and uniformly at random from $[0,1]$
- Poll 1: If $m = n/2$, what is the probability that an envy-free allocation exists?
    1. 0
    2. $2/n$
    3. $1/2$
    4. 1

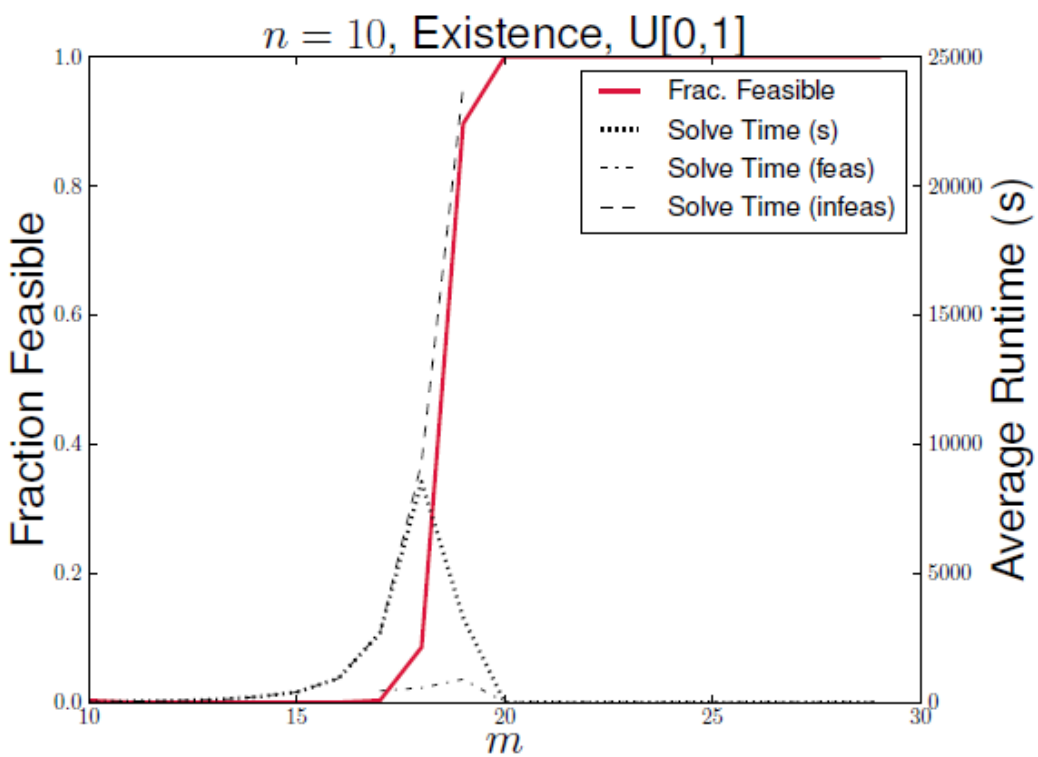
Phase Transition

- Imagine the v_{pi} are drawn independently and uniformly at random from [0,1]
- Poll 2: If $m \gg n$, what is the probability that an envy-free allocation exists?
  1. Close to 0
  2. Close to 1/3
  3. Close to 1/2
  4. Close to 1

Sharp Transition

Given an instance,
the probability of getting
an envy-free allocation?

Depends on a single parameter:
n/m

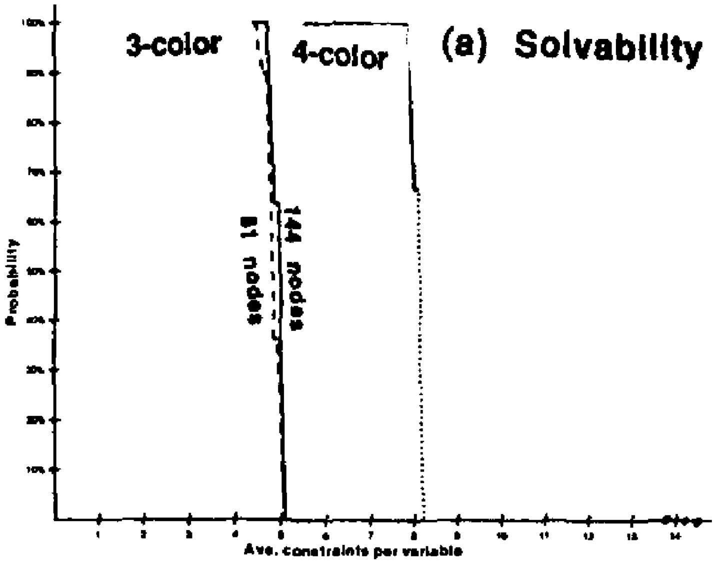


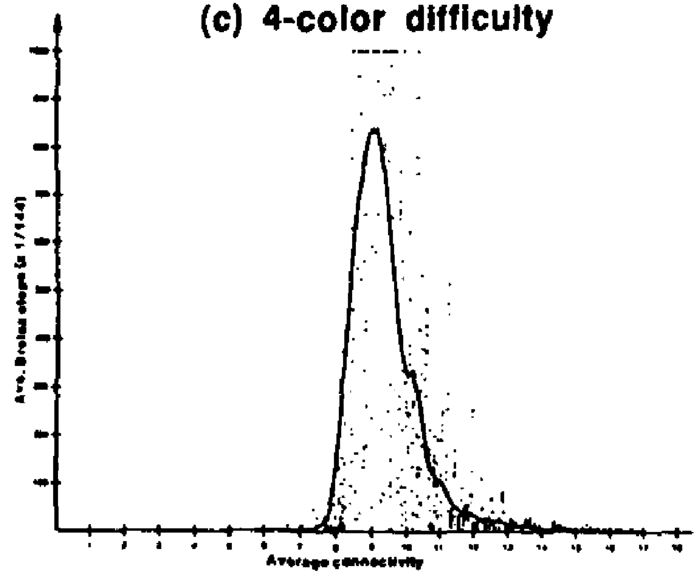
[Dickerson et al., AAAI 2014]

Sharp Transition

Graph coloring

Critical parameter: average degree in the graph



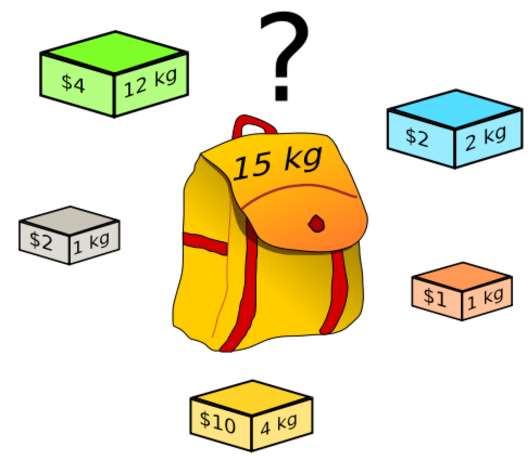


[Cheeseman et al., IJCAI 1993]

IP Optimization: Knapsack*

*Optional slide, IP example, not required for the course

Are given n objects and one container of limited capacity W. Each object i has a value p_i and uses a capacity w_i. The goal is to select the subset of objects that maximize the sum of the values while no exceeding the capacity of the container.



$$
\begin{aligned}
\max \quad & Z = p_1 x_1 + p_2 x_2 + \ldots + p_n x_n \\
s.t. \quad & w_1 x_1 + w_2 x_2 + \ldots + w_n x_n \leq W \\
& x_1,\ x_2,\ \ldots,\ x_n \ \in \{0, 1\}
\end{aligned}
$$

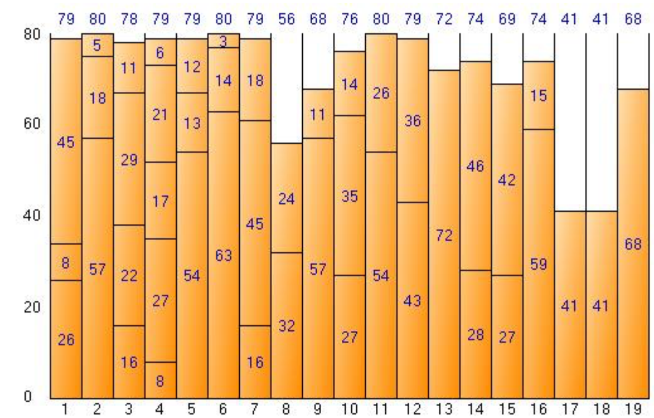
Multiple containers

$$
\begin{aligned}
\max \quad & Z = \sum_{i=1}^{n} \sum_{j=1}^{m} p_i x_{ij} \\
s.t. \quad & \sum_{i=1}^{n} w_i x_{ij} \leq W_j \qquad & j = 1, 2, \ldots, m \\
& \sum_{j=1}^{m} x_{ij} \leq 1 \qquad & i = 1, 2, \ldots, n \\
& x_{ij} \in \{0, 1\}, \qquad & i = 1, 2, \ldots, n,\ j = 1, 2, \ldots, m
\end{aligned}
$$

IP Optimization: Bin Packing*

*Optional slide of IP example, not required for the course

Given n objects, each using a capacity p_i , i=1,..,n, and m containers (bins) of limited capacity q_j, j=1,..,m, the goal is to group all the n objects minimizing the number of bins that are used out of the m available ones, and respecting their capacity limits



$$
\begin{aligned}
\min \quad & Z = \sum_{j=1}^{m} b_j \\
s.t. \quad & \sum_{i=1}^{n} p_i x_{ij} \leqslant q_j b_j, \quad j = 1, \dots, m \\
& \sum_{j=1}^{m} x_{ij} = 1, \quad i = 1, \dots, n \\
& x_{ij} \in \{0, 1\}, \quad i = 1, \dots, n, \; j = 1, \dots, m \\
& b_j \in \{0, 1\}, \quad j = 1, \dots, m
\end{aligned}
$$

IP Optimization: Set Covering*

*Optional slide, IP example, not required for the course

$$\min \quad Z = \sum_{j=1}^{k} c_j x_j$$

$$s.t. \quad \sum_{j=1}^{k} a_{ij} x_j \geqslant 1, \quad \forall i = 1, \ldots, m$$

$$x_j \in \{0,1\}, \quad \forall j = 1, \ldots, k$$

Are given a set of k "activities" A, and a set of m "requirements" R. Each activity j can "cover" one or more requirements with a cost c_j. Select a subset of the activities such that *all* requirements are covered by *at least one activity* and the total cost is minimized

$$
\begin{array}{llllllll}
\min & Z = x_1 + x_2 + x_3 + x_4 + x_5 + x_6 \\
s.t. & x_1 + & x_2 + & & & x_5 & & \geqslant 1 \\
& x_1 + & & x_3 & & & & \geqslant 1 \\
& & x_2 + & & x_4 & & & \geqslant 1 \\
& & & x_3 + & & & x_6 & \geqslant 1 \\
& & x_2 + & x_3 + & & & x_6 & \geqslant 1 \\
& x_1,\ x_2,\ x_3,\ x_4,\ x_5\, x_6 \in \{0,1\}
\end{array}
$$

		A					
		1	2	3	4	5	6
R	1	X	X			X	
	2	X		X			
	3		X		X		
	4			X			X
	5		X	X			X

IP vs. LP

- Denote the optimal solutions of the two programs by OPT_{IP} and OPT_{LP}
- Poll 3: Which statement is true?
  1. $\mathrm{OPT}_{IP} \le \mathrm{OPT}_{LP}$
  2. $\mathrm{OPT}_{IP} \ge \mathrm{OPT}_{LP}$
  3. $\mathrm{OPT}_{IP} = \mathrm{OPT}_{LP}$
  4. $\mathrm{OPT}_{IP} \,||\, \mathrm{OPT}_{LP}$

IP

$$\max \sum_{j=1}^{\ell} c_j x_j$$

$$\text{s.t.} \quad \forall i \in [k],\ \sum_{j=1}^{\ell} a_{ij} x_j \le b_i$$

$$\forall j \in [\ell],\ x_j \in \{0,1\}$$

LP

$$\max \sum_{j=1}^{\ell} c_j x_j$$

$$\text{s.t.} \quad \forall i \in [k],\ \sum_{j=1}^{\ell} a_{ij} x_j \le b_i$$

$$\forall j \in [\ell],\ x_j \in [0,1]$$

LP IS A *RELAXATION* OF IP

Original IP Problem (Primal, P):

$$\begin{aligned} \max \quad & Z_P = \boldsymbol{c}^T \boldsymbol{x} \\ s.t. \quad & A\boldsymbol{x} \leq \boldsymbol{b} \\ & \boldsymbol{x} \in \mathbb{Z}_0^{n+} \end{aligned}$$

IP

Relaxed LP Problem (RLP):

$$\begin{aligned} \max \quad & Z_{RLP} = \boldsymbol{c}^T \boldsymbol{x} \\ s.t. \quad & A\boldsymbol{x} \leq \boldsymbol{b} \\ & \boldsymbol{x} \in \mathbb{R}_0^{n+} \end{aligned}$$

LP

Easier to solve!

RLP provides an **UPPER BOUND** (UB) on the optimal value of P

$$Z_P^* \leq Z_{RLP}^*$$

If the problem is a *min* one, then RLP provides a **LOWER BOUND** (LB) on the optimal value of P

$$Z_P^* \geq Z_{RLP}^*$$

Cases for LP solutions vs. IP

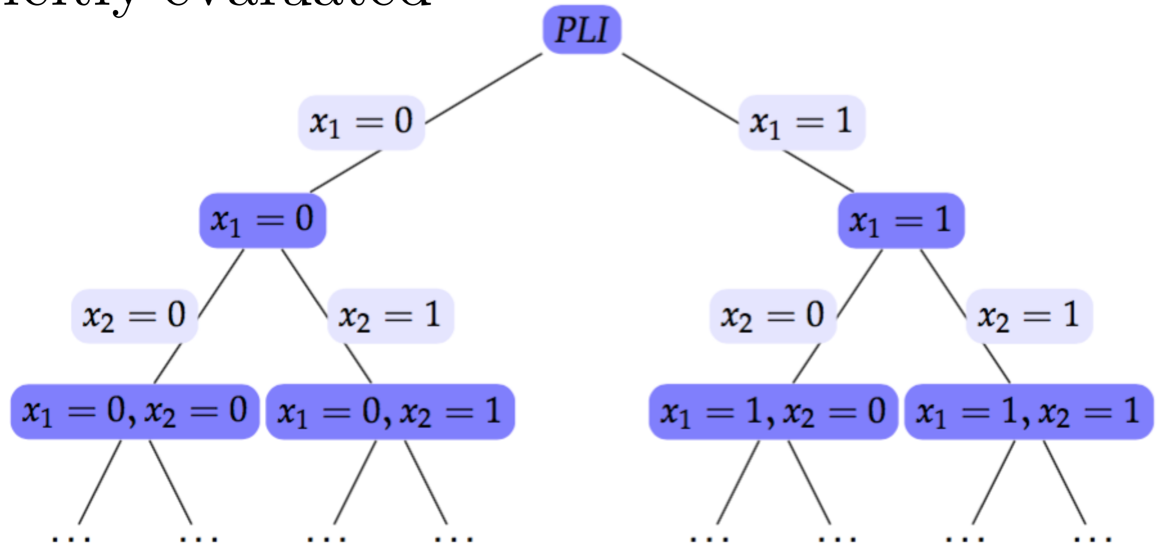
1. **UB:** *RLP* has an optimal solution of the form $\boldsymbol{x}^*_{RLP} = (x_1, x_2, \ldots, x_n)$ such that, for at least one $k \in \{1,..n\}$, $x_k \in \mathbb{R} \Rightarrow \boldsymbol{x}^*_{RLP}$ is *not feasible* for P, but $\boldsymbol{x}^*_{RLP} \geq \boldsymbol{x}^*_P$
2. **Feasible solution:** *RLP* has an optimal solution of the form $\boldsymbol{x}^*_{RLP} = (x_1, x_2, \ldots, x_n)$ such that $\forall\, k \in \{1,..n\}$, $x_k \in \mathbb{Z}_0^+$ $\Rightarrow \boldsymbol{x}^*_{RLP}$ is *feasible* for P, and $\boldsymbol{x}^*_{RLP} = \boldsymbol{x}^*_P$
3. **No solution (unfeasible):** *RLP* does *not* have a solution $\rightarrow$ also P has no solution
4. **No solution (unbounded):** *RLP* is *unbounded* $(+\infty)$ $\rightarrow$ P is either unfeasible or unbounded $\rightarrow$ does *not* have a finite solution ("almost" true)

Solving IPs

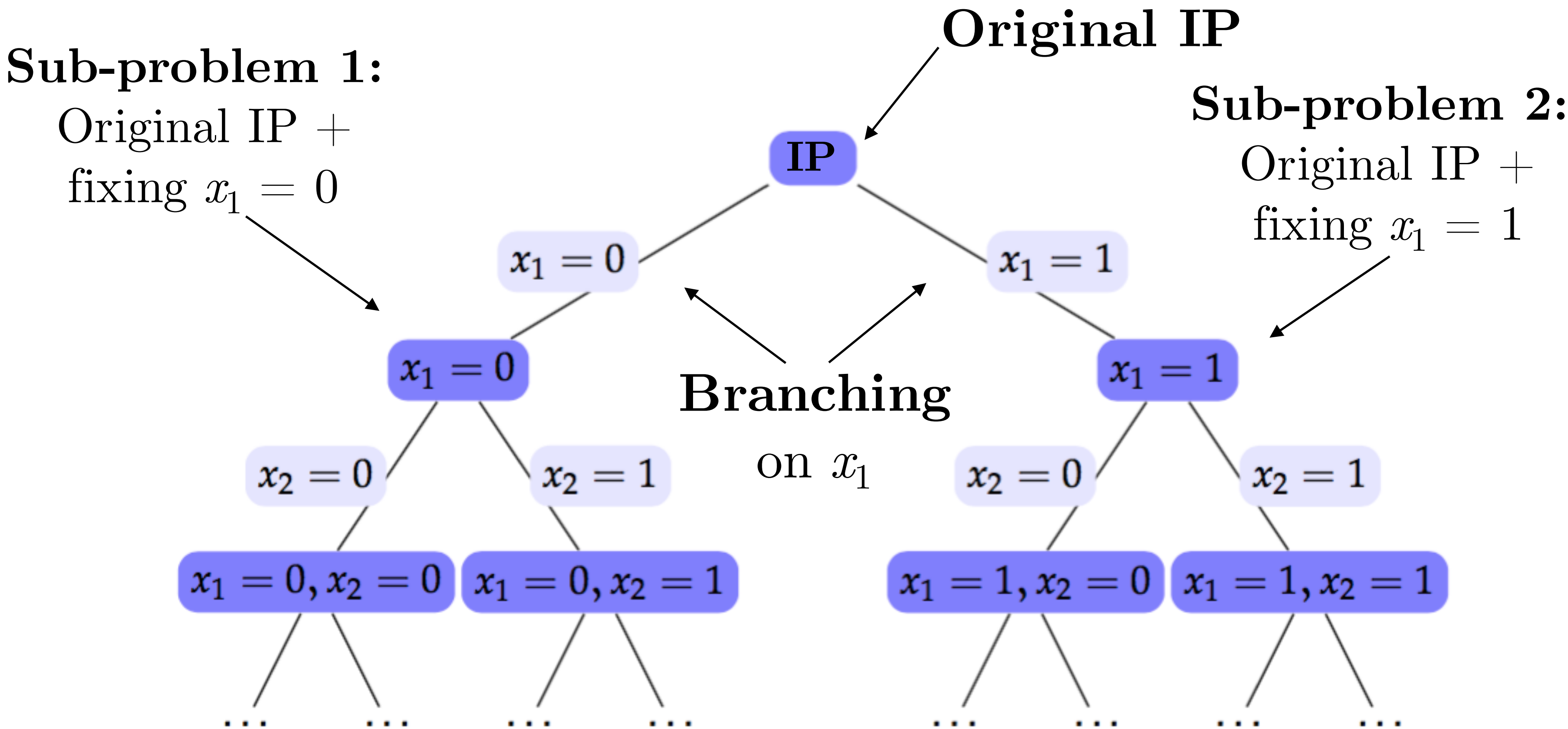
- n integer variables each taking m values: $O(m^n)$ solutions $\rightarrow$ Complete enumeration cannot be afforded (IP is NP-complete/hard)
- *Implicit (intelligent) enumeration:* cover all possible solutions by explicitly evaluating only a small subset of them
- **Divide and conquer** $\rightarrow$ *Branch and bound*

Solving IPs

- **Divide and conquer**: *Divide* the (finite) problem domain into a series of *easier to solve sub-problems* that are systematically generated by branching on variables, and are *fathomed* (understood and closed) (*conquer*).
- The procedure can be applied recursively until all the solutions are implicitly evaluated



Solving IPs



Every parent node "contains" the solutions of all its children

Solving IPs

- In principle every sub-problem needs to be solved
- If the original IP has n variables, the sub-problems at the level k of the search tree have n-k free variables, and k fixed ones
- Sub-problems are progressively "easier" compared to the original IP since they involve less variables to assign, but still they can be too difficult
- ***Branch and bound*** effectively prunes the search tree by exploiting the properties of a relaxation

Branch and Bound: Ideas

1. ***Branch*** on variables to systematically generate new sub-problems
2. Solve sub-problems using a ***relaxation*** (e.g., LP!), which is "easy" (polynomial time) and therefore can be applied to solve a large number of sub-problems
3. The result from a relaxed sub-problem allows to define a ***bound*** on the quality of the solutions that can be obtained by expanding the sub-problem.
4. Bounds are used to ***prune*** the tree, removing all the potential children of a fathomed sub-problem

Sub-problem Fathoming and BB Tree Pruning

A sub-problem *SP* is **fathomed**, and its children branch in the tree can be **pruned,** if one of the following conditions is satisfied (according to the fact that *SP* “contains” all its children solutions):

1. *SP* is *unfeasible* (has no solution) $\rightarrow$ Fathomed by unfeasibility
2. *SP*’s optimal solution is *IP feasible* $\rightarrow$ Fathomed by integrality
3. The bound (*UB/LB*) obtained from solving *SP* shows that the solutions that can be obtained from expanding *SP* cannot be better than *the best incumbent solution for P* (or the bound *estimated* from external heuristics) $\rightarrow$ Fathomed by bound

INCUMBENT SOLUTION $\leftrightarrow$ DYNAMICALLY UPDATED LB (UB)

- The value of all *IP-feasible* solutions obtained by solving the *SP*s is recorded while the BB search progresses
- At each step k, the *best* of the IP-feasible solutions generated so far by *SP*s (or generated by external heuristics) is the current **incumbent solution** $\tilde{\boldsymbol{Z}}_k$ to the IP (i.e., the current candidate to be the optimal solution)
- At step k, the incumbent solution is the ***best known LB*** for the (max) IP (i.e., we know that Z^*_{IP} *at least* must be equal to $\tilde{\boldsymbol{Z}}_k$)

Dynamically updated UB (max)

- The objective value Z^*_{SP0} of the root node establishes an upper bound on the optimal objective, Z^*_{IP}, because the feasible region of *SP0* contains all integer feasible solutions to *IP*
- Child node objective values are no better than those of their parent, since a child consist of (parent + fix-a-variable-value)
- → Objective values keep decreasing from Z^*_{SP0} along the branches
- It is apparent that, at every step, a better integer solution can only be produced by the children of currently unexplored nodes
- → At every step, Z^*_{IP} can be no better than the objective value (the UB) of best unexpanded sub-problem (the tree leafs)
- → **The best objective of leaf nodes dynamically defines an UB!**

Integrality GAP and Optimality

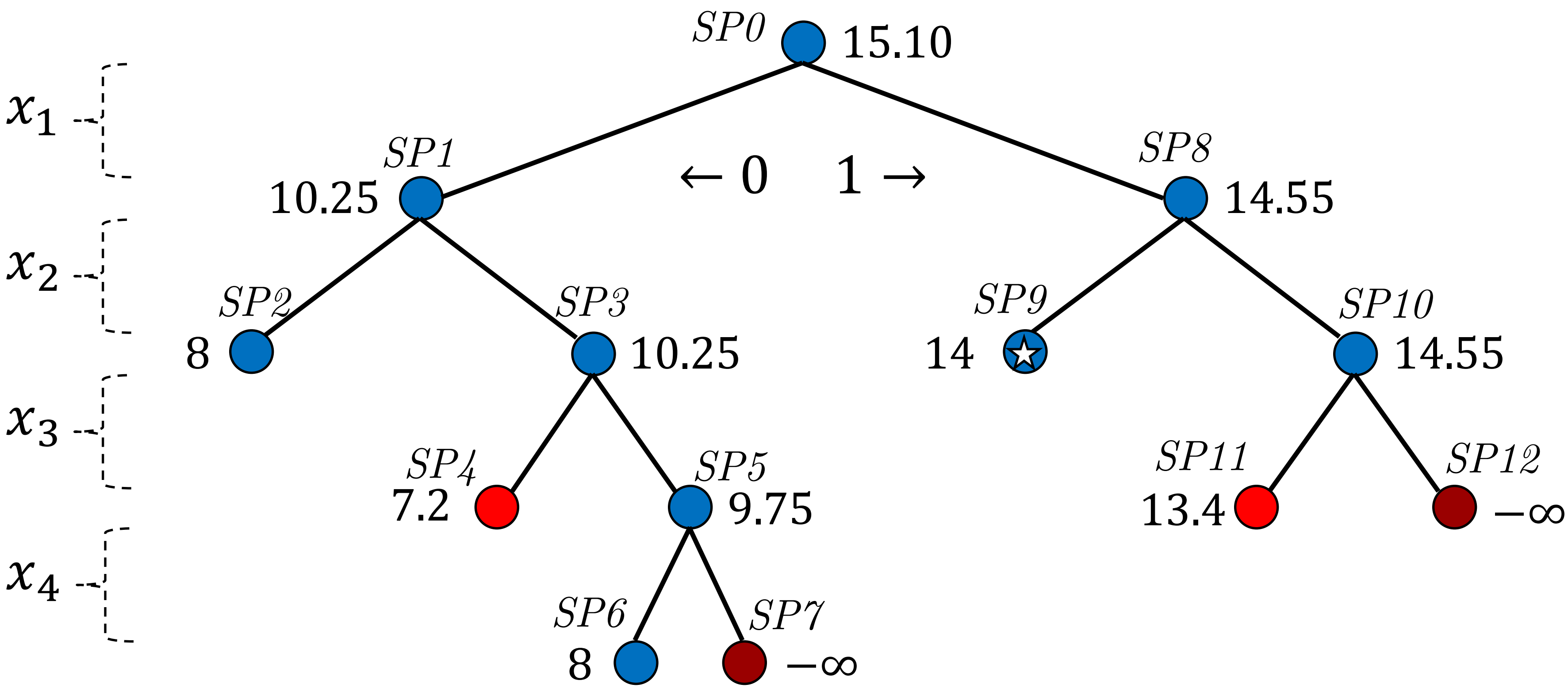
- Each new incumbent defines a better LB to *IP* (max problem)
- The best UB from current leaf nodes defines a better UB to *IP*
- This progresses monotonically: the distance (or the ratio) between the LB and the UB defines the current relative (MIP) *gap*

$$(\max)\ \mathrm{Gap}_k = \frac{|ObjBound - IncumbentVal\,|_k}{|IncumbentVal|_k} = \frac{|UB - LB\,|_k}{|LB|_k}$$

$$(\min)\ \mathrm{Gap}_k = \frac{|ObjBound - IncumbentVal\,|_k}{|IncumbentVal|_k} = \frac{|LB - UB\,|_k}{|UB|_k}$$

- At optimality $\rightarrow$ Gap $= 0$
- **Branch and bound guarantees to find the optimal solution**
- BB finishes when (Gap $=$0) $\vee$ (All nodes are fathomed)

Branch and Bound

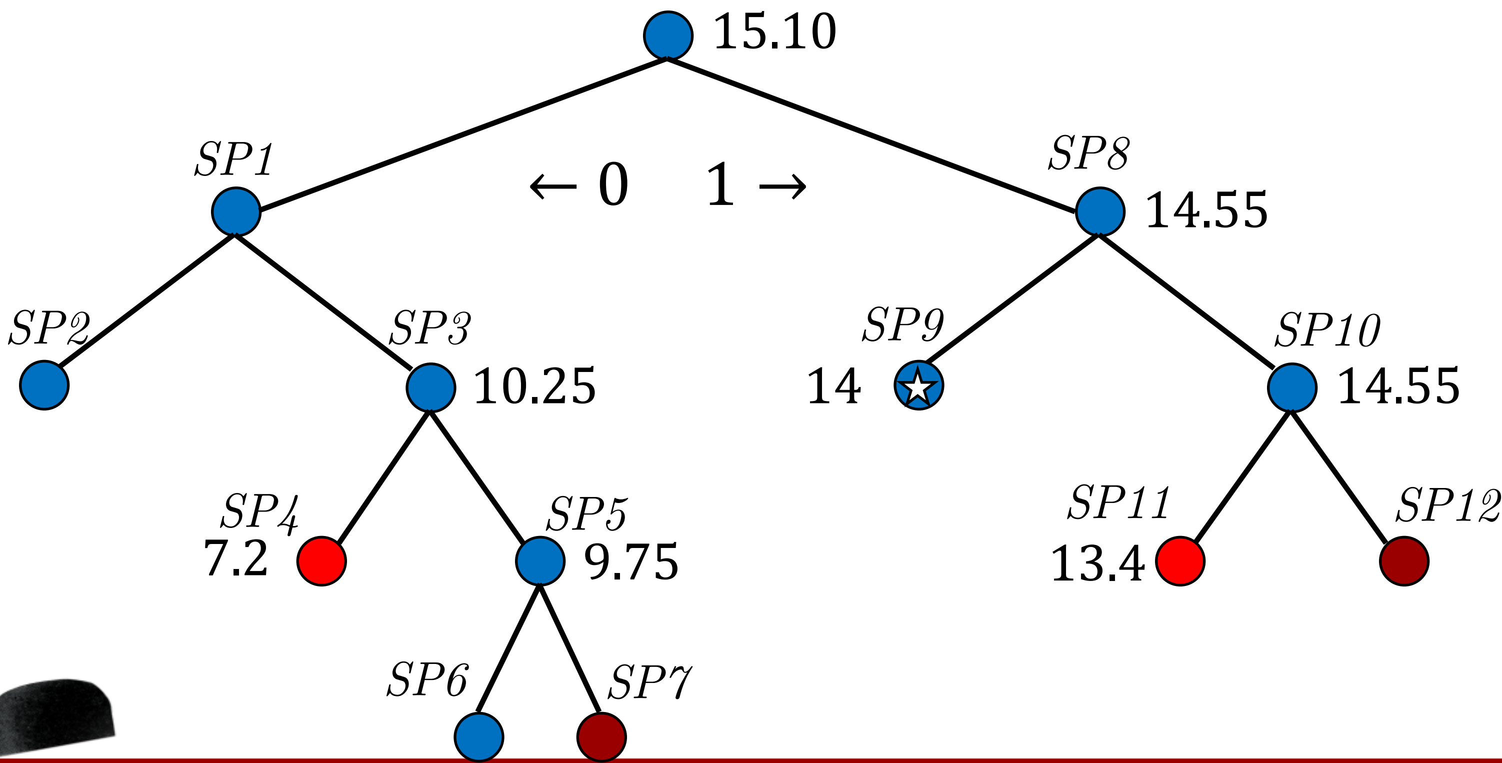


SP 2,6,9 fathomed by integrality
SP 4, 11 fathomed by bound
SP 7, 12 fathomed by infeasibility

$Z^*_{IP} = 14$

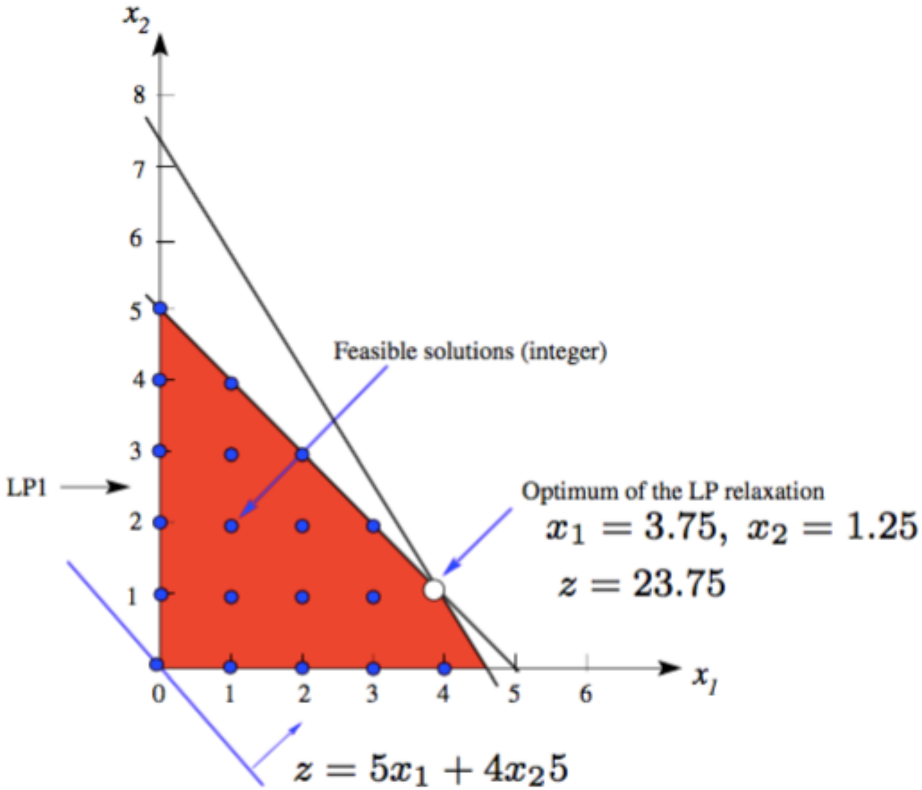
Gap evolution

SP	0	1	2	3	4	5	6	7	8	9	10	11	12
LB	-∞	-∞	8	8	8	8	8	8	8	14	14	14	14
UB	15.1	15.1	15.1	15.1	15.1	15.1	15.1	15.1	14.55	14.55	14.55	14.55	14.55

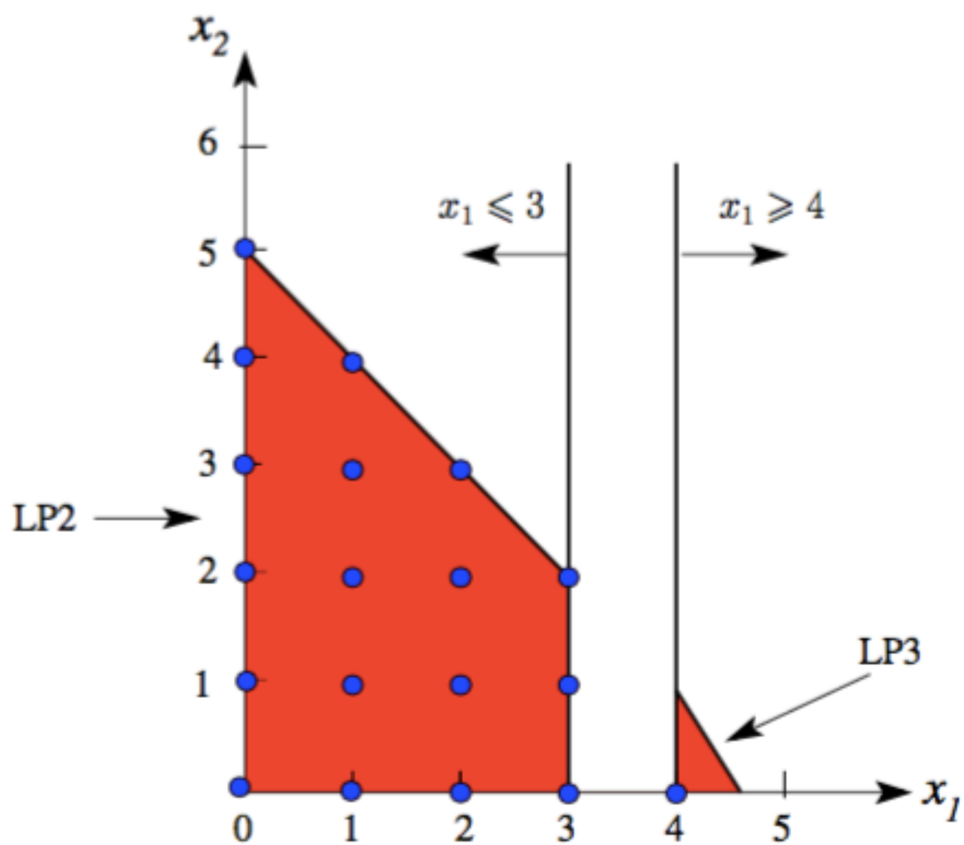


BEYOND 0-1 VARIABLES: A GENERAL INTEGER PROBLEM

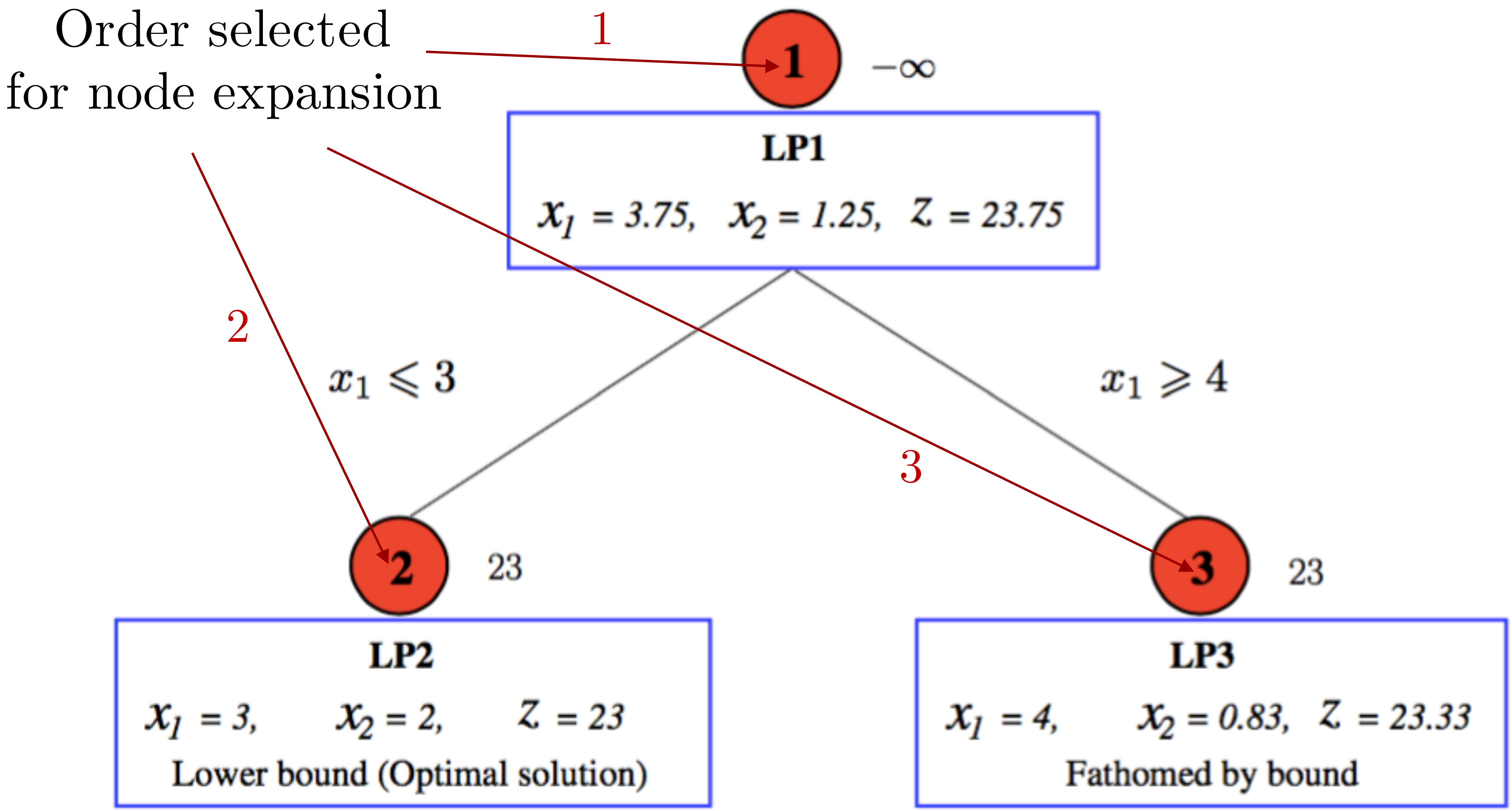
$$\begin{array}{llcl} \max & Z = 5x_1 + 4x_2 & & \\ s.t. & x_1 + \ \ x_2 & \leqslant & 5 \\ & 10x_1 + 6x_2 & \leqslant & 45 \\ & x_1,\ x_2 \in \mathbb{Z}_+ & & \end{array}$$



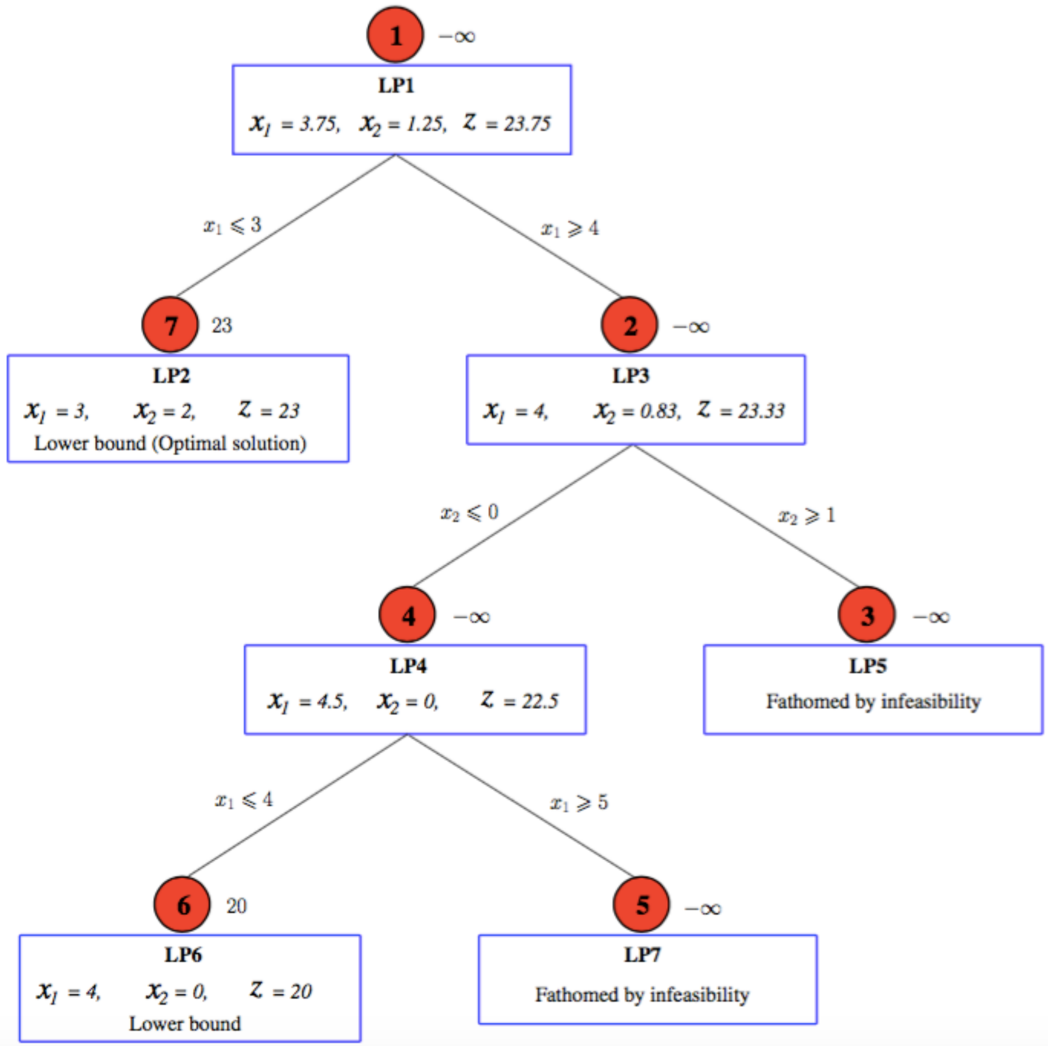
Branching on a variable taking fractional values



BB Tree


Order selected for node expansion
1
2
3
1
$-\infty$
LP1
$x_1 = 3.75, \quad x_2 = 1.25, \quad z = 23.75$
$x_1 \leqslant 3$
$x_1 \geqslant 4$
2
23
3
23
LP2
$x_1 = 3, \quad x_2 = 2, \quad z = 23$
Lower bound (Optimal solution)
LP3
$x_1 = 4, \quad x_2 = 0.83, \quad z = 23.33$
Fathomed by bound

Different node selections



Design choices

- BB is guaranteed to find the optimal solution, but in an *exponential worst-case time*! Design choices matters...
- **Branching rules:** which variable select next for branching?
- **Bound calculation:** how to evaluate the solutions of a sub-problem? Which relaxation should be used?
- **Tree exploration strategy**: how to select the sub-problem to solve next?
- **Availability of feasible solutions for fathoming**: use of external heuristics to get useful bound? When do apply such heuristics during the BB process?
- **Termination criteria**: which quality/time/convergence criteria?

Tree exploration strategies

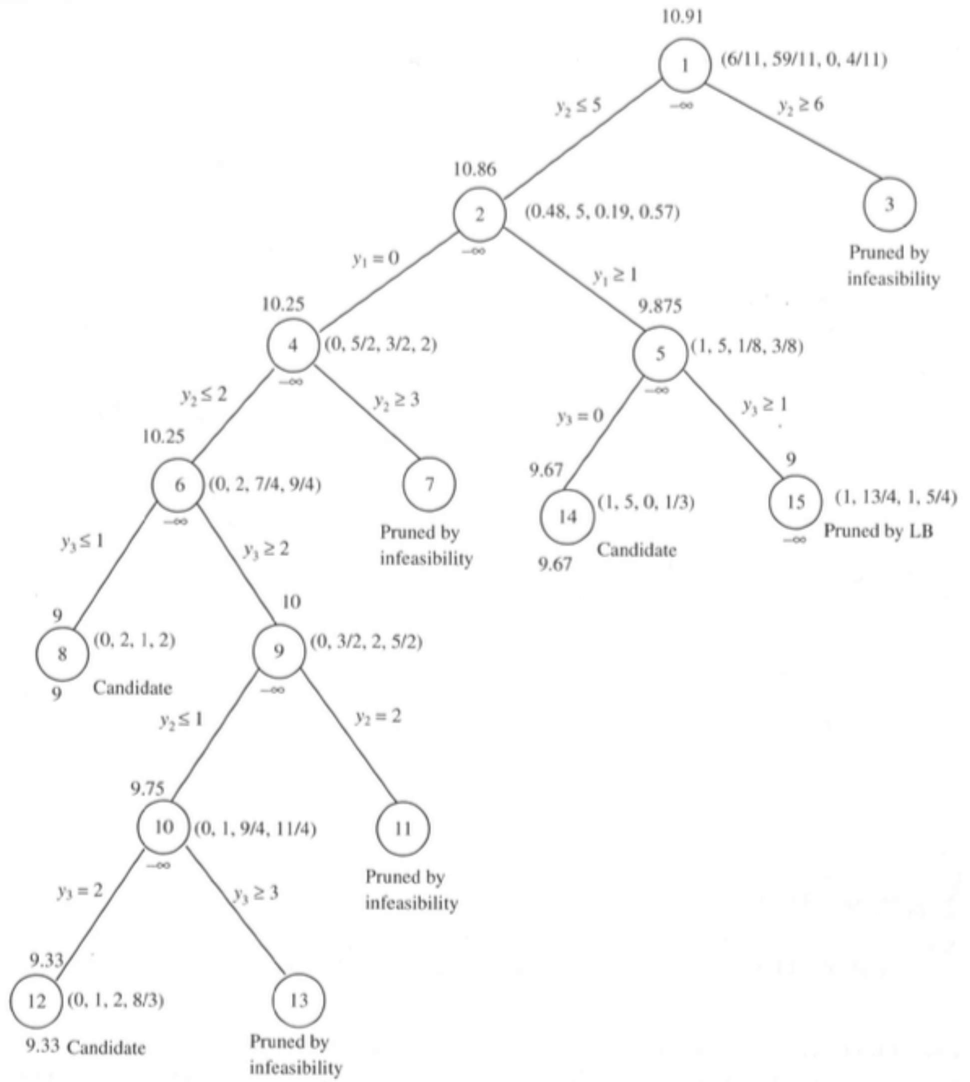
- **Depth First:** LIFO, solve the most recently generated node first, then backtrack and plunge into another depth first
  - Fast to reach feasible solutions (rapidly reaching leafs)
  - Optimize memory (many nodes closed by feasibility)
  - Risk: fully explore sub-trees with low quality solutions
- **Breadth First:** generate and solve all the nodes at the same level before going to the next level
  - Require a queue data structure
  - # of open nodes grows exponentially with search depth
  - Used rarely in practice

Tree exploration strategies

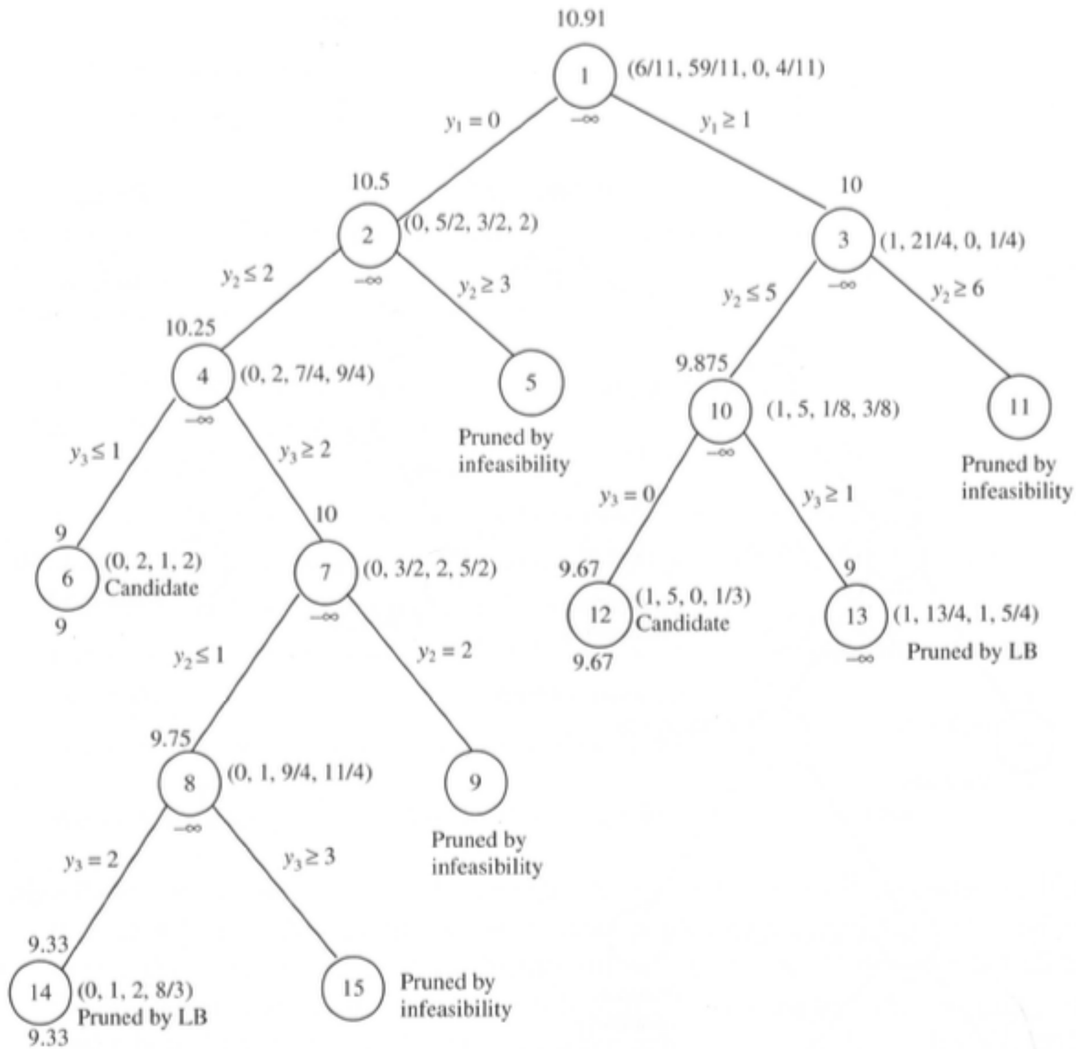
- **Best Bound First:** the most promising node is chosen, that is, the node with the best bound (for max, the higher UB)
  - Usually, it allows to reduce the number of expanded nodes compared to DF
  - It doesn't really dive into the tree, since it tends to stay where problems are less constrained (i.e., bounds are more promising)
  - It's hard to quickly find good feasible solutions for fathoming, such that the number of open problems is usually quite large

- **Hybrid combinations:** DF at the beginning to obtain good feasible solutions, and after a mixture of BBF and DF

EXAMPLE USING DF



Example Using BBF



Commercial IP Solvers

IBM ILOG CPLEX

Gurobi

Other IPs: Coming Soon

Dodgson's
voting rule

Stackelberg
security games

Summary

- Terminology:
  - Integer programs / linear programs
- Big ideas:
  - IP representation leads to "efficient" solutions
  - Phase transition ⇔ complexity
  - LP as an "admissible" heuristic
  - Intelligent implicit enumeration: Branch and bound