



# CMU 15-781

Lecture 14:

Integer Programming

Teachers:

Emma Brunskill

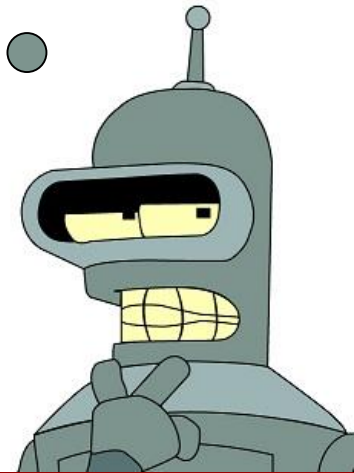
Ariel Procaccia (this time)

# INTEGER PROGRAMMING

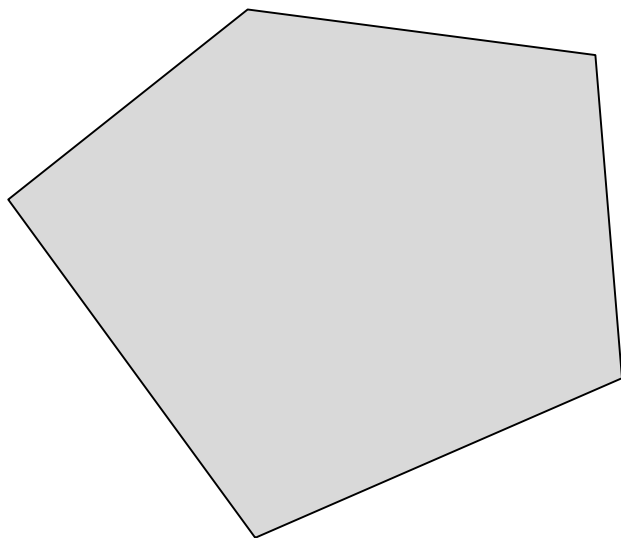
- An integer programming (IP) problem:
  - $a_{ij} \in \mathbb{R}$  for  $i \in [k] = \{1, \dots, k\}, j \in [\ell]$
  - $b_i \in \mathbb{R}$  for  $i \in [k]$
  - Variables  $x_j$  for  $j \in [\ell]$
- The (feasibility) problem is:

$$\begin{array}{ll} \text{find} & x_1 \dots, x_\ell \\ \text{s.t.} & \forall i \in [k], \sum_{j=1}^{\ell} a_{ij} x_j \leq b_i \\ & \forall j \in [\ell], x_j \in \mathbb{Z} \end{array}$$

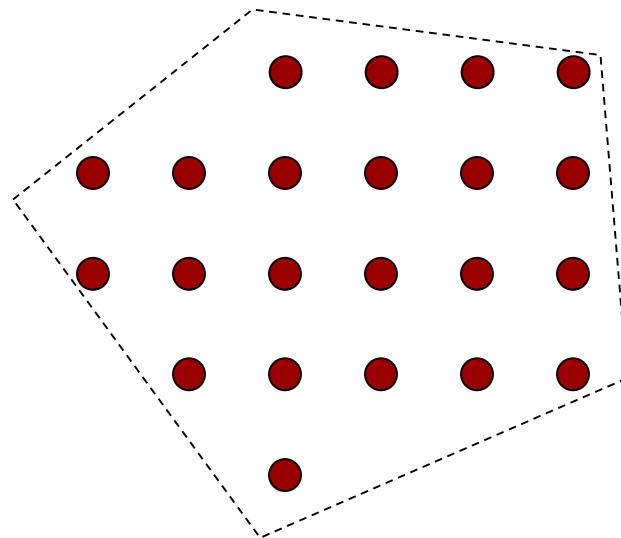
How can we express  
 $\geq$  constraints?  
Equality constraints?  
Restricted domains?



# IP IS NOT CONVEX



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



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

# 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

- 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  $t = 1, \dots, 27$ ,  $S_t$  is a row, column, or  $3 \times 3$  square

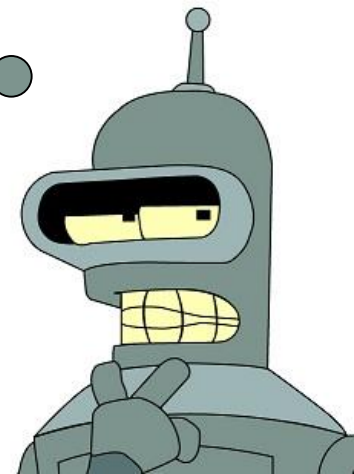
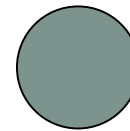
find  $x_1^{11}, \dots, x_9^{99}$

s.t.  $\forall t \in [27], \forall k \in [9], \sum_{(i,j) \in S_t} x_k^{ij} = 1$

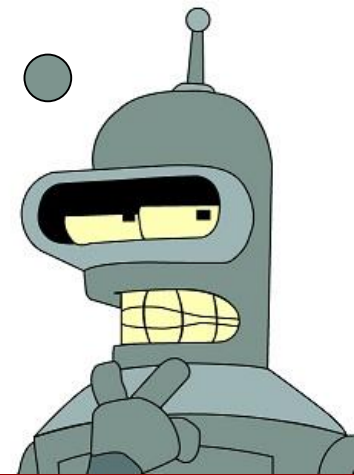
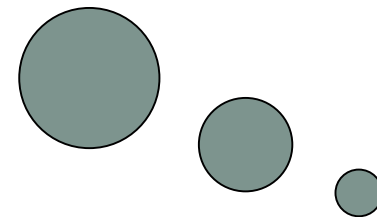
$\forall i, j \in [9], \sum_{k \in [9]} x_k^{ij} = 1$

$\forall i, j, k \in [9], x_k^{ij} \in \{0, 1\}$

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










SUDOKU is NP-  
complete, so we just  
proved that integer  
programming is  
NP-complete!





# EXAMPLE: FAIR DIVISION

- **Players**  $N = \{1, \dots, n\}$  and **items**  $M = \{1, \dots, m\}$
- Player  $i$  has value  $v_{ij}$  for item  $j$
- Partition items to bundles  $A_1, \dots, A_n$
- $A_1, \dots, A_n$  is **envy-free** iff  $\forall i, i', \sum_{j \in A_i} v_{ij} \geq \sum_{j \in A_{i'}} v_{ij}$

	1	2	3	4	5	6	7
1							
1	\$30	\$50	\$2	\$5	\$5	\$3	\$5
2	\$2	\$10	\$5	\$20	\$20	\$3	\$40

# EXAMPLE: FAIR DIVISION

- Variables:  $x_{ij} \in \{0,1\}$ ,  $x_{ij} = 1$  iff  $j \in A_i$
- ENVY-FREE as an IP:

find  $x_{11}, \dots, x_{nm}$

s.t.  $\forall i \in N, \forall i' \in N, \sum_{j \in M} v_{ij} x_{ij} \geq \sum_{j \in M} v_{ij} x_{i'j}$

$\forall j \in M, \sum_{i \in N} x_{ij} = 1$

$\forall i \in N, j \in M, x_{ij} \in \{0,1\}$



# APPLICATION: SPLIDDIT



[DIVIDE](#) [RENT](#) [FARE](#) [CREDIT](#) [GOODS](#) [TASKS](#) | [ABOUT](#) [FEEDBACK](#)

## PROVABLY FAIR SOLUTIONS.

Spliddit offers quick, free solutions to everyday fair division problems, using methods that provide indisputable fairness guarantees and build on decades of research in economics, mathematics, and computer science.



Share Rent



Split Fare



Assign Credit



Divide Goods



Distribute Tasks



Suggest an App

# PHASE TRANSITION

- Imagine the  $v_{ij}$  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

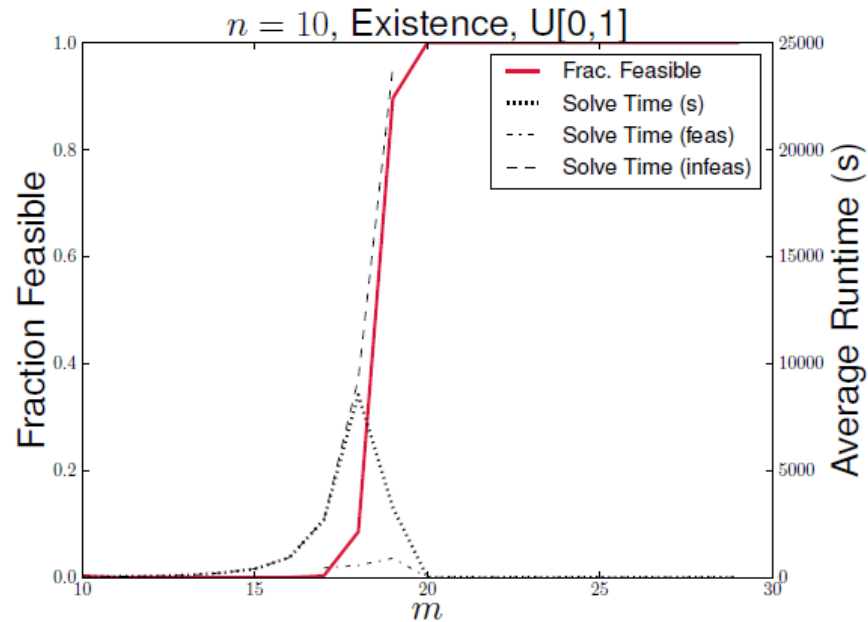


# PHASE TRANSITION

- Imagine the  $v_{ij}$  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

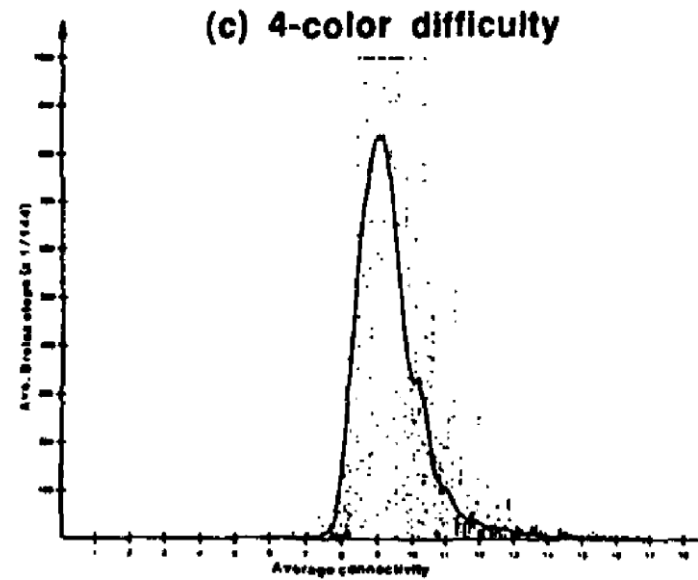
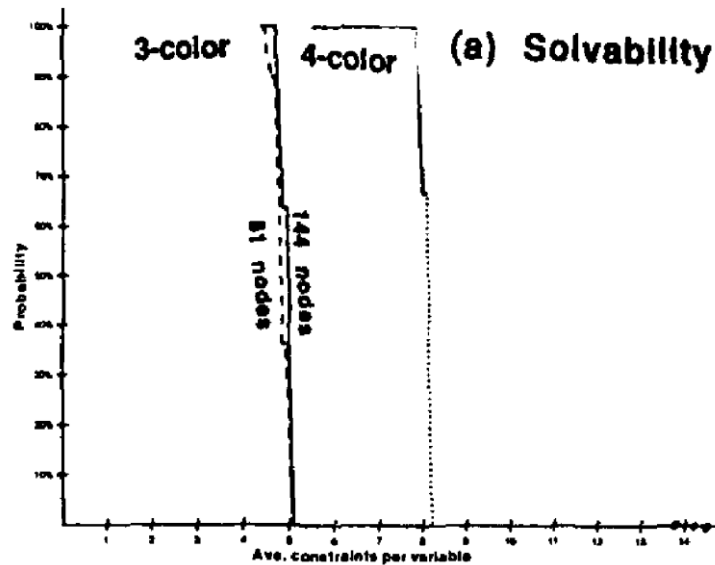


# SHARP TRANSITION



[Dickerson et al., AAAI 2014]

# SHARP TRANSITION



[Cheeseman et al., IJCAI 1993]

# INTEGER PROGRAMMING, REVISITED

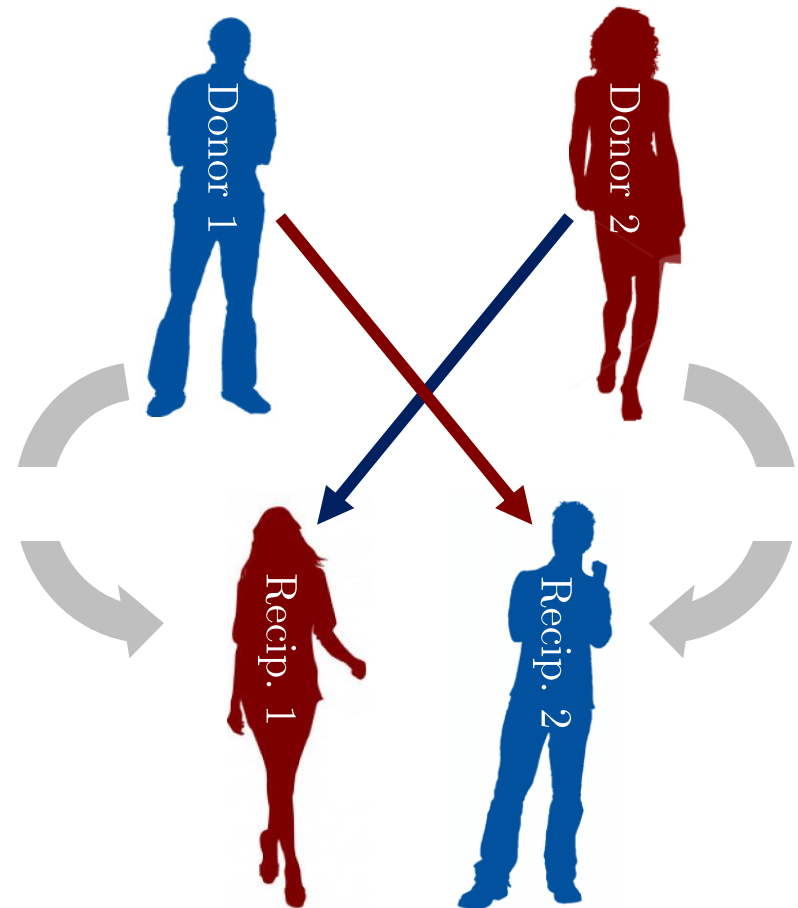
- The standard formulation optimizes a linear objective function  $\mathbf{c}^T \mathbf{x}$
- The problem is:

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



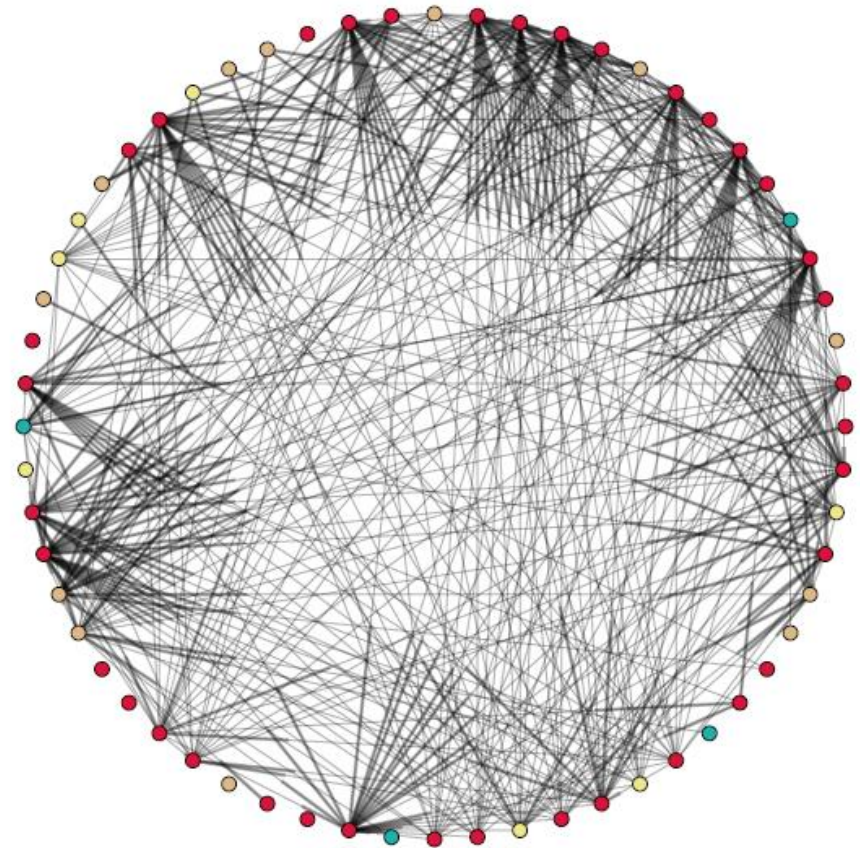
# EXAMPLE: KIDNEY EXCHANGE

- Kidney donations from live donors are common
- But some donors are incompatible with their patients
- Kidney exchange enables swaps



# EXAMPLE: KIDNEY EXCHANGE

- **CYCLE-COVER:** Given a directed graph  $G$  and  $L \in \mathbb{N}$ , find a collection of cycles of length  $\leq L$  in  $G$  that maximizes the number of covered vertices
- The problem is:
  - Easy for  $L = 2$  (why?)
  - Easy for unbounded  $L$
  - NP-hard for a constant  $L \geq 3$



UNOS pool, Dec 2010

[Courtesy John Dickerson, CMU]

# EXAMPLE: KIDNEY EXCHANGE

- Variables: For each cycle  $c$  of length  $\ell_c \leq L$ , variable  $x_c \in \{0,1\}$ ,  $x_c = 1$  iff cycle  $c$  is included in the cover
- CYCLE-COVER as an IP:

$$\begin{aligned} \max \quad & \sum_c x_c \ell_c \\ \text{s.t.} \quad & \forall v \in V, \sum_{c:v \in c} x_c \leq 1 \\ & \forall c, x_c \in \{0,1\} \end{aligned}$$



# APPLICATION: UNOS



**UNITED NETWORK FOR ORGAN SHARING**

# IP vs. LP, REVISITED

- Denote the optimal solutions of the two programs by  $\text{OPT}_{IP}$  and  $\text{OPT}_{LP}$
- **Poll 3:** Which statement is true?

1.  $\text{OPT}_{IP} \leq \text{OPT}_{LP}$
2.  $\text{OPT}_{IP} \geq \text{OPT}_{LP}$
3.  $\text{OPT}_{IP} = \text{OPT}_{LP}$
4.  $\text{OPT}_{IP} \parallel \text{OPT}_{LP}$

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

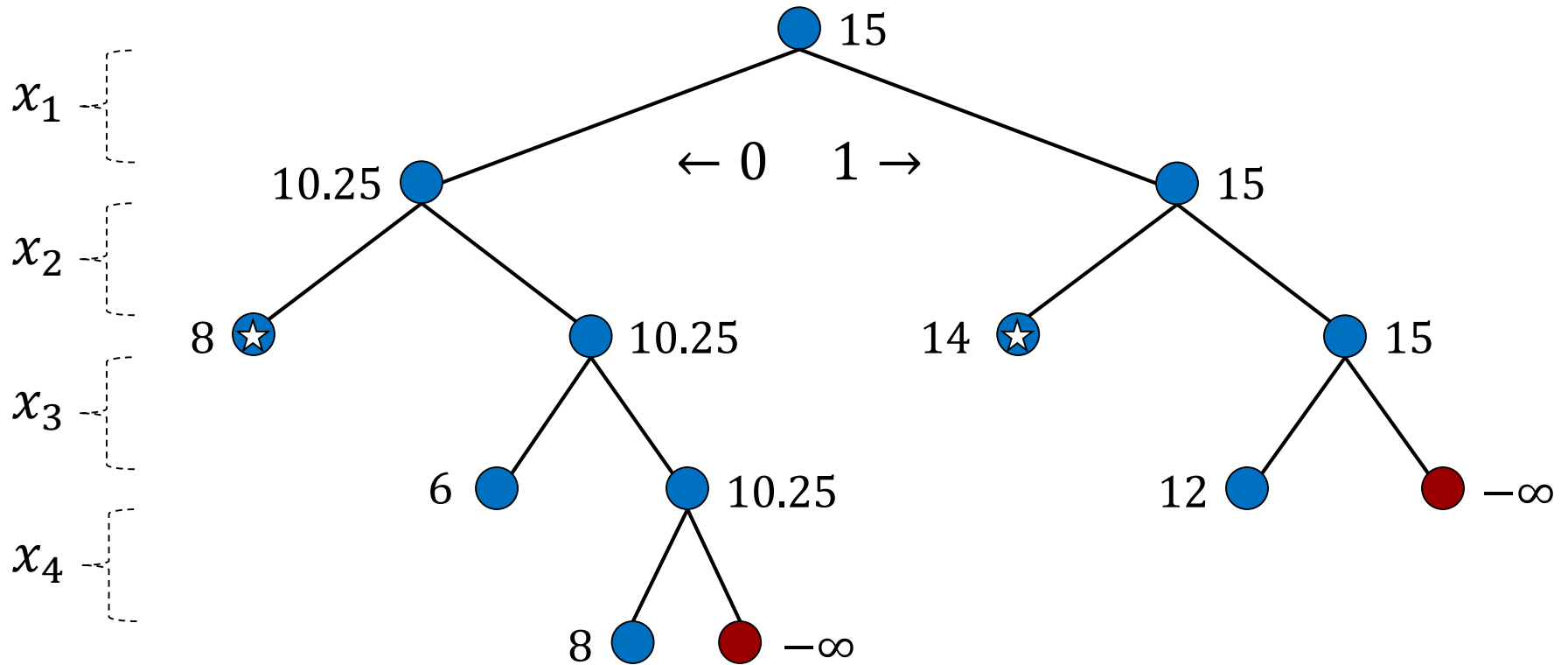
$$\begin{aligned} \max \quad & \sum_{j=1}^{\ell} c_j x_j && \text{LP} \\ \text{s.t.} \quad & \forall i \in [k], \sum_{j=1}^{\ell} a_{ij} x_j \leq b_i \\ & \forall j \in [\ell], x_j \in [0,1] \end{aligned}$$

# BRANCH AND BOUND

- The linear program (LP) relaxation gives an “admissible” heuristic!
- LPs can be solved in polynomial time!
- Branch and bound:
  - Use a search tree to assign the variables one by one
  - At each node, solve the LP relaxation
  - Prune the branch if there is no solution, or if the solution is worse than best known



# BRANCH AND BOUND



# 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  $\Leftrightarrow$  complexity
  - LP as an “admissible” heuristic

