

Discrete optimization crash course

Content draws on material from Optimization Methods in Management Science from MIT Sloan

An important property of algorithms used in practice is
broad applicability

Example: **Integer programming solvers**

Most popular tool for solving combinatorial (& nonconvex) problems



Routing



Manufacturing



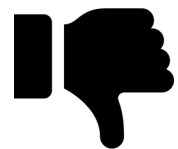
Scheduling



Planning



Finance



...but they can have **unsatisfactory** default performance

Slow runtime, poor solutions, ...

Integer programming (IP)

IP solvers (CPLEX, Gurobi) have a **ton** of parameters

- CPLEX has **170-page** manual describing **172** parameters
- Tuning by hand is notoriously **slow, tedious, and error-prone**

CPX_PARAM_NODEFILEIND 100	CPX_PARAM_TRELIM 160	CPX_PARAM_RANDOMSEED 130	CPXPARAM_MIP_Pool_RelGap 148	CPX_PARAM_FLOWCOVERS 70	CPX_PARAM_BRDIR 39
CPX_PARAM_NODELIM 101	CPX_PARAM_TUNINGDETTILIM 160	CPX_PARAM_REDUCE 131	CPXPARAM_MIP_Pool_Replace 151	CPX_PARAM_FLOWPATHS 71	CPX_PARAM_BTTL 40
CPX_PARAM_NODESEL 102	CPX_PARAM_TUNINGDISPLAY 162	CPX_PARAM_REINV 131	CPXPARAM_MIP_Strategy_Branch 39	CPX_PARAM_FPHEUR 72	CPX_PARAM_CALCQCPCDUALS 41
CPX_PARAM_NUMERICALEMPHASIS 102	CPX_PARAM_TUNINGMEASURE 163	CPX_PARAM_RELAXPREIND 132	CPXPARAM_MIP_Strategy_MIQCPStrat 93	CPX_PARAM_FRACCAND 73	CPX_PARAM_CLIQUES 42
CPX_PARAM_NZREADLIM 103	CPX_PARAM_TUNINGREPEAT 164	CPX_PARAM_RELOBJDIF 133	CPXPARAM_MIP_Strategy_StartAlgorithm 139	CPX_PARAM_FRACCUTS 73	CPX_PARAM_CLOCKTYPE 43
CPX_PARAM_OBJDIF 104	CPX_PARAM_TUNINGTILIM 165	CPX_PARAM_REPAIRTRIES 133	CPXPARAM_MIP_Strategy_VariableSelect 166	CPX_PARAM_FRACPASS 74	CPX_PARAM_CLONELOG 43
CPX_PARAM_OBJLLIM 105	CPX_PARAM_VARSEL 166	CPX_PARAM_REPEATPRESOLVE 134	CPXPARAM_MIP_SubMIP_NodeLimit 155	CPX_PARAM_GUBCOVERS 75	CPX_PARAM_COEREDIND 44
CPX_PARAM_OBJULIM 105	CPX_PARAM_WORKDIR 167	CPX_PARAM_RINSHEUR 135	CPXPARAM_OptimalityTarget 106	CPX_PARAM_HEURFREQ 76	CPX_PARAM_COLREADLIM 45
CPX_PARAM_PARALLELMODE 108	CPX_PARAM_WORKMEM 168	CPX_PARAM_RLT 136	CPXPARAM_Output_WriteLevel 169	CPX_PARAM_IMPLBD 76	CPX_PARAM_CONFLICTDISPLAY 46
CPX_PARAM_PERIND 110	CPX_PARAM_WRITELEVEL 169	CPX_PARAM_ROWREADLIM 141	CPXPARAM_Preprocessing_Aggregator 19	CPX_PARAM_INTSOLFILEPREFIX 78	CPX_PARAM_COVERS 47
CPX_PARAM_PERLIM 111	CPX_PARAM_ZEROHALFCUTS 170	CPX_PARAM_SCAIND 142	CPXPARAM_Preprocessing_Fill 19	CPX_PARAM_INTSOLLIM 79	CPX_PARAM_CPUMASK 48
CPX_PARAM_POLISHAFTERDETTIME 111	CPXPARAM_Benders_Strategy 30	CPX_PARAM_SCRIND 143	CPXPARAM_Preprocessing_Linear 120	CPX_PARAM_ITLIM 80	CPX_PARAM_CRAIN 50
CPX_PARAM_POLISHAFTEREPAGAP 112	CPXPARAM_Benders_Tolerances_feasibilitycut 35	CPX_PARAM_SIFTALG 143	CPXPARAM_Preprocessing_Reduce 131	CPX_PARAM_LANDPCUTS 82	CPX_PARAM_CUTLO 51
CPX_PARAM_POLISHAFTEREPGAP 113	CPXPARAM_Benders_Tolerances_optimalitycut 36	CPX_PARAM_SIFTDISPLAY 144	CPXPARAM_Preprocessing_Symmetry 156	CPX_PARAM_LBHEUR 81	CPX_PARAM_CUTPASS 52
CPX_PARAM_POLISHAFTERINTSOL 114	CPXPARAM_Conflict_Algorithm 46	CPX_PARAM_SIFTTILIM 145	CPXPARAM_Read_DataCheck 54	CPX_PARAM_LPMETHOD 136	CPX_PARAM_CUTSFACTOR 52
CPX_PARAM_POLISHAFTERNODE 115	CPXPARAM_CPUmask 48	CPX_PARAM_SIMDISPLAY 145	CPXPARAM_Read_Scale 142	CPX_PARAM_MFCUTS 82	CPX_PARAM_CUTUP 53
CPX_PARAM_POLISHAFTERTIME 116	CPXPARAM_DistMIP_Rampup_Duration 128	CPX_PARAM_SINGLIM 146	CPXPARAM_ScreenOutput 143	CPX_PARAM_MEMORYEMPHASIS 83	CPXPARAM_DATACHECK 54
CPX_PARAM_POLISHTIME (deprecated) 116	CPXPARAM_LPMethod 136	CPX_PARAM_SOLNPOOLAGAP 146	CPXPARAM_Sifting_Algorithm 143	CPX_PARAM_MIPCBREDLP 84	CPX_PARAM_DEPIND 55
CPX_PARAM_POPULATELIM 117	CPXPARAM_MIP_Cuts_BQP 38	CPX_PARAM_SOLNPOOLCAPACITY 147	CPXPARAM_Sifting_Display 144	CPX_PARAM_MIPDISPLAY 85	CPX_PARAM_DETTILIM 56
CPX_PARAM_PPRIIND 118	CPXPARAM_MIP_Cuts_LocallyImplied 77	CPX_PARAM_SOLNPOOLGAP 148	CPXPARAM_Sifting_Iterations 145	CPX_PARAM_MIPEMPHASIS 87	CPX_PARAM_DISJCUTS 57
CPX_PARAM_PREDUAL 119	CPXPARAM_MIP_Cuts_RLT 136	CPX_PARAM_SOLNPOOLINTENSITY 149	CPXPARAM_Simplex_Display 145	CPX_PARAM_MIPINTERVAL 88	CPX_PARAM_DIVETYPE 58
CPX_PARAM_PREIND 120	CPXPARAM_MIP_Cuts_ZeroHalfCut 170	CPX_PARAM_SOLNPOOLREPLACE 151	CPXPARAM_Simplex_Limits_Singularity 146	CPX_PARAM_MIPKAPPASTATS 89	CPX_PARAM_DPRIIND 59
CPX_PARAM_PRLINEAR 120	CPXPARAM_MIP_Limits_CutsFactor 52	CPX_PARAM_SOLUTIONTARGET (deprecated: see CPXPARAM_OptimalityTarget 106)	CPXPARAM_SolutionType 152	CPX_PARAM_MIPORDIND 90	CPX_PARAM_EACHCUTLIM 60
CPX_PARAM_PREPASS 121	CPXPARAM_MIP_Limits_RampupDetTimeLimit 127	CPXPARAM_SOLUTIONTYPE 152	CPXPARAM_Threads 157	CPX_PARAM_MIPORDTYPE 91	CPX_PARAM_EPAGAP 61
CPX_PARAM_PRESLVND 122	CPXPARAM_MIP_Limits_RampupTimeLimit 128	CPX_PARAM_STARTALG 139	CPXPARAM_TimeLimit 159	CPX_PARAM_MIPSEARCH 92	CPX_PARAM_EPGAP 61
CPX_PARAM_PRICELIM 123	CPXPARAM_MIP_Limits_Solutions 79	CPX_PARAM_STRONGCANDLIM 154	CPXPARAM_Tune_DefTimeLimit 160	CPX_PARAM_MIQCPSTRAT 93	CPX_PARAM_EPINT 62
CPX_PARAM_PROBE 123	CPXPARAM_MIP_Limits_StrongCand 154	CPX_PARAM_STRONGCANDLIM 154	CPXPARAM_Tune_Display 162	CPX_PARAM_MIRCUTS 94	CPX_PARAM_EPMRK 64
CPX_PARAM_PROBEDETTIME 124	CPXPARAM_MIP_Limits_StrongIt 154	CPX_PARAM_STRONGITLIM 154	CPXPARAM_Tune_Measure 163	CPX_PARAM_MPSLONGNUM 94	CPX_PARAM_EPOPT 65
CPX_PARAM_PROBETIME 124	CPXPARAM_MIP_Limits_TreeMemory 160	CPX_PARAM_SUBALG 99	CPXPARAM_Tune_Repeat 164	CPX_PARAM_NETDISPLAY 95	CPX_PARAM_EPPER 65
CPX_PARAM_QPMAKEPSDIND 125	CPXPARAM_MIP_OrderType 91	CPX_PARAM_SUBMIPNODELIMIT 155	CPXPARAM_Tune_TimeLimit 165	CPX_PARAM_NETEPOPT 96	CPX_PARAM_EPRELAX 66
CPX_PARAM_QPMETHOD 138	CPXPARAM_MIP_Pool_AbsGap 146	CPX_PARAM_SYMMETRY 156	CPXPARAM_WorkDir 167	CPX_PARAM_NETEPRHS 96	CPX_PARAM_EPRHS 67
CPX_PARAM_QPNZREADLIM 126	CPXPARAM_MIP_Pool_Capacity 147	CPX_PARAM_THREADS 157	CPXPARAM_WorkMem 168	CPX_PARAM_NETFIND 97	CPX_PARAM_FEASOPTMODE 68
	CPXPARAM_MIP_Pool_Intensity 149	CPX_PARAM_TILIM 159	CraInd 50	CPX_PARAM_NETITLIM 98	CPX_PARAM_FILEENCODING 69
				CPX_PARAM_NETPRIIND 98	

Integer programming (IP)

IP solvers (CPLEX, Gurobi) have a **ton** of parameters

- CPLEX has **170-page** manual describing **172** parameters
- Tuning by hand is notoriously **slow, tedious**, and **error-prone**

What's the best **configuration** for the application at hand?



Best configuration for **routing** problems
likely not suited for **scheduling**



Plan

This class: Overview of how these solvers work

Future classes: How to use ML to optimize these solvers

Outline

- 1. Linear programming**
2. Integer programming
3. SAT solving
4. Next steps

Linear programming

Linear programming (LP) is a central topic in optimization

Provides a powerful tool for modeling many applications

Tons of attention over past two decades due to:

- **Applicability:** Many real-world applications can be modeled via LPs
- **Solvability:** Efficient techniques for solving large-scale problems

Basic components of an LP

Each optimization problem consists of 3 elements:

- **Decision variables**: describe our choices that are under our control;
- **Objective function**: Criterion that we wish to minimize (e.g., cost)
or maximize (e.g., profit)
- **Constraints**: Limitations restricting our choices for decision variables

“Linear programming” refers to an optimization problem where:

- The **objective function** is linear
- Each **constraint** is a linear inequality or equality

An introductory example

A company makes two products (say, P and Q)

- Uses two machines (say, A and B)

Each unit of P that is produced requires:

- 50 minutes processing time on machine A, and
- 30 minutes processing time on machine B

Each unit of Q that is produced requires:

- 24 minutes processing time on machine A, and
- 33 minutes processing time on machine B

An introductory example

- Machine A is going to be available for 40 hours
- Machine B is available for 35 hours
- Profit per unit of P is \$28
- Profit per unit of Q is \$30
- **Goal:** determine production quantity of P and Q such that:
 1. Total profit is maximized
 2. Available resources aren't exceeded
- **Task:** formulate this problem as an LP

Step 1: Defining the decision variables

Decision variables: Describe choices under our control

Goal: determine production quantity of P and Q such that ...

So there are 2 decision variables:

x : the number of units of P

y : the number of units of Q

Step 2: Choosing an objective function

Usually seek a criterion to compare alternative solutions

This yields the objective function

Want to maximize the total profit

- Profit per each unit of product P is \$28
- Profit per each unit of Q is \$30

Total profit is $28x + 30y$ if we produce x units of P & y units of Q

Leads to the following objective function:

$$\max \underline{28x + 30y}$$

Linear

Step 3: Identifying the constraints

Often are limitations that restrict our decisions

Resource, physical, strategic, economical

We describe these limitations using **mathematical constraints**

Step 3: Identifying the constraints

- Each unit of P requires 50 minutes on machine A
- Each unit of Q requires 24 minutes on machine A
- If we produce x units of P and y units of Q:
 - Machine A needs to be used for $50x + 24y$
- Machine A is available for 40 hours = 2400 minutes
 - This imposes the following constraint: $50x + 24y \leq 2400$
- Similarly, amount of time machine B is available means that:
$$30x + 33y \leq 2100$$

Step 3: Identifying the constraints

In most problems, decision variables must be nonnegative

So need to include the following two constraints as well:

$$x \geq 0 \text{ and } y \geq 0$$

In the end, the constraints we're subject to (s.t.) are :

$$50x + 24y \leq 2400, \text{ (machine A time)}$$

$$30x + 33y \leq 2100, \text{ (machine B time)}$$

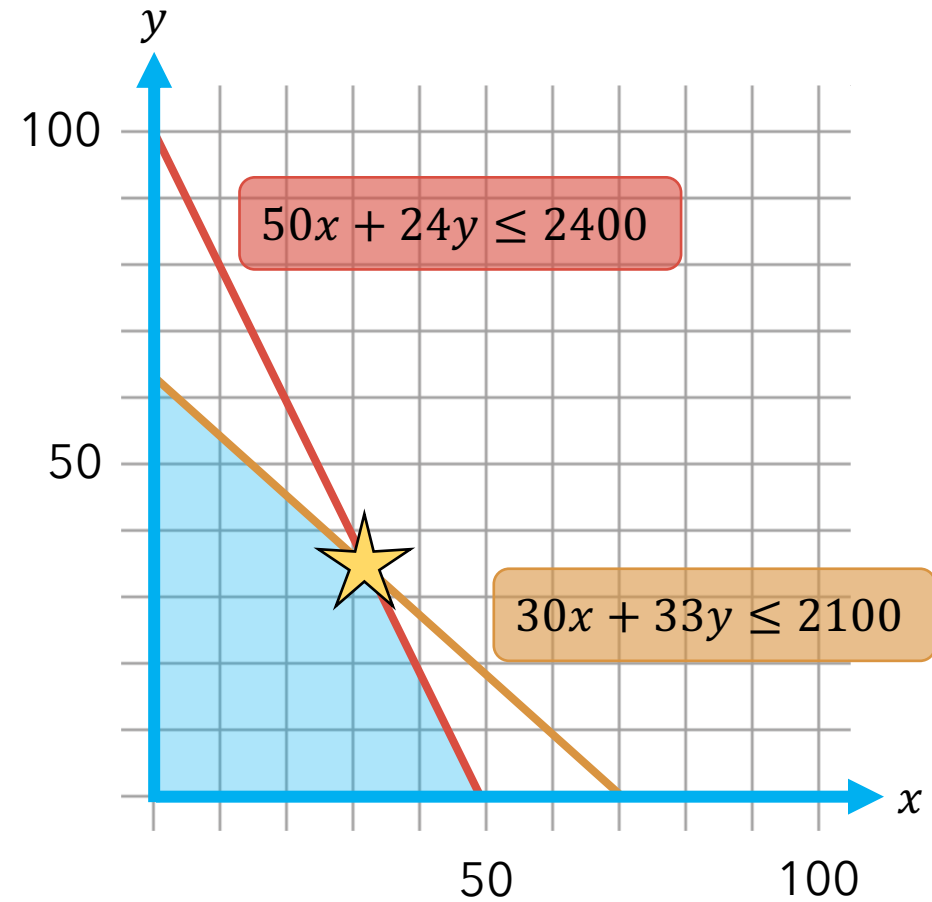
$$x \geq 0,$$

$$y \geq 0$$

LP for the example

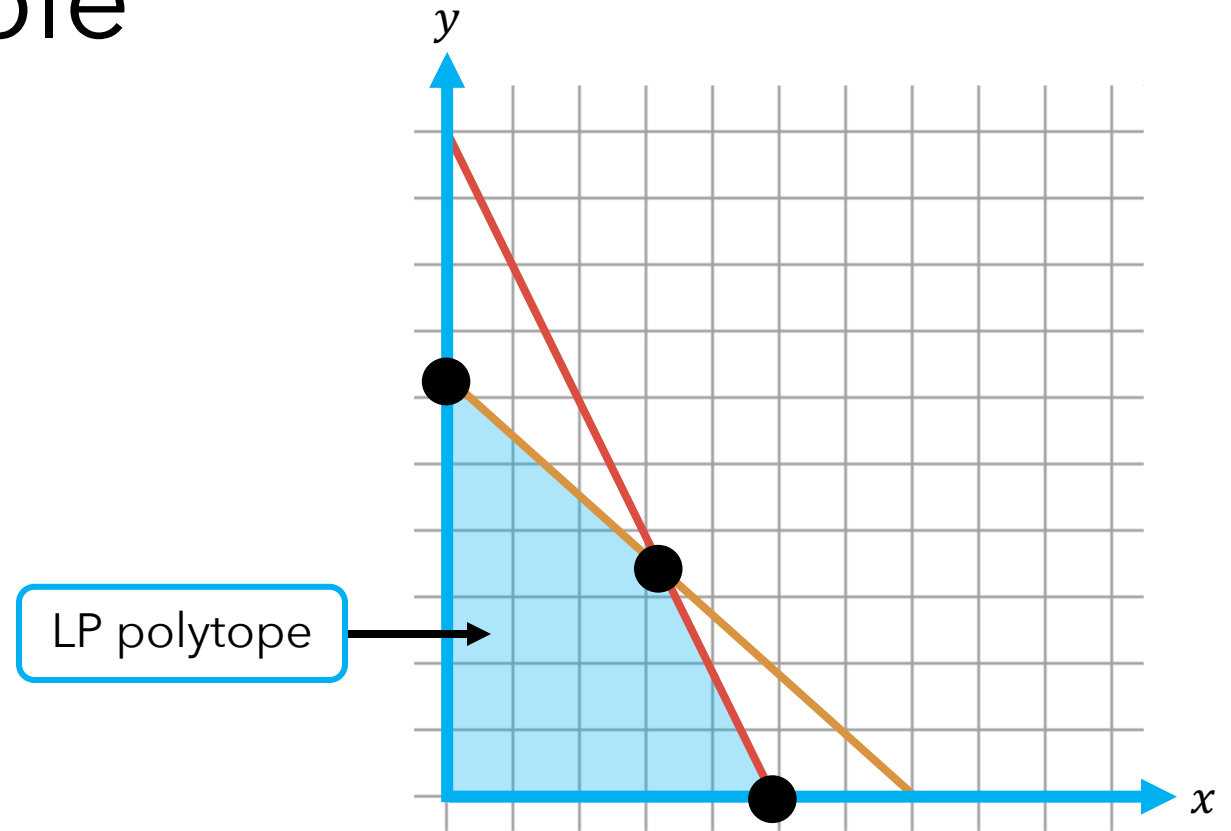
Here is the LP:

$$\begin{aligned} &\text{maximize} && 28x + 30y \\ &\text{subject to} && 50x + 24y \leq 2400 \\ & && 30x + 33y \leq 2100 \\ & && x \geq 0 \text{ and } y \geq 0 \end{aligned}$$



Optimal solution: $x = 30.97, y = 35.48$

LP for the example



Fact: Optimal solution of an LP is always at a vertex

LP algorithms

Simplex algorithm: Practical algorithm for solving LPs

- May run in exponential time in the worst case
- Provable runs in polynomial time on “realistic” LPs
- Used by commercial solvers like CPLEX, Gurobi, ...

Ellipsoid method: Impractical but provably runs in poly-time

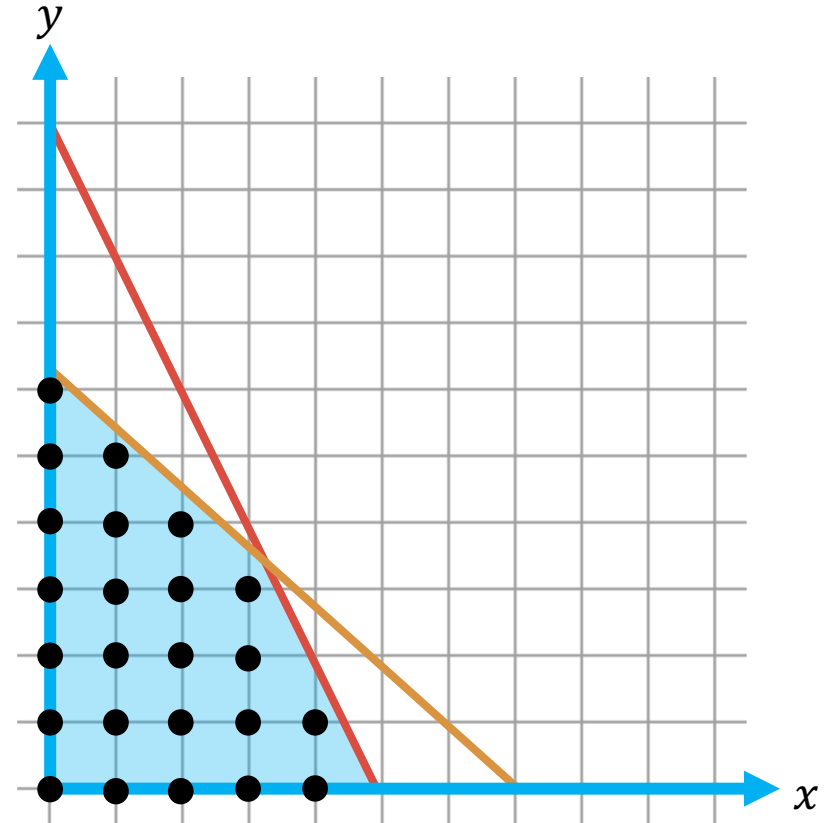
Outline

1. Linear programming
- 2. Integer programming**
3. SAT solving
4. Next steps

Integer programming (IP)

What if the decision variables must be integral?

maximize $28x + 30y$
subject to $50x + 24y \leq 2400$
 $30x + 33y \leq 2100$
 $x \geq 0$ and $y \geq 0$
 $x, y \in \mathbb{Z}$

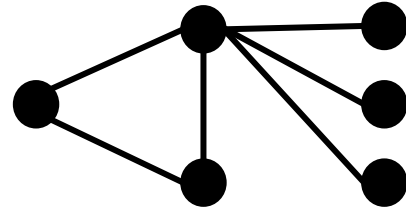


Integer programming is NP-complete

Example: vertex cover

Vertex cover of a graph:

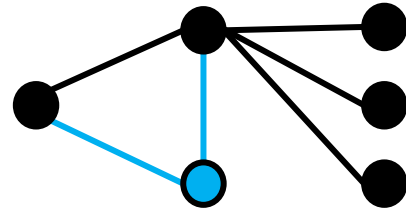
Set of vertices that includes ≥ 1 endpoint of every edge



Example: vertex cover

Vertex cover of a graph:

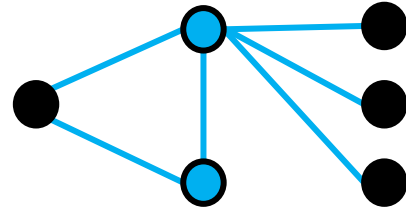
Set of vertices that includes ≥ 1 endpoint of every edge



Example: vertex cover

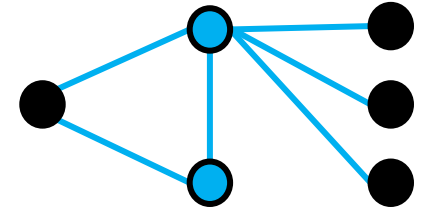
Vertex cover of a graph:

Set of vertices that includes ≥ 1 endpoint of every edge



Goal: Find a vertex cover of minimal size

Vertex cover IP



Input: Graph $G = (V, E)$ with vertex set V , edge set E

1. Decision variables: For each vertex $v \in V$,

$$y_v = \begin{cases} 1 & \text{if } v \text{ in vertex cover} \\ 0 & \text{otherwise} \end{cases}$$

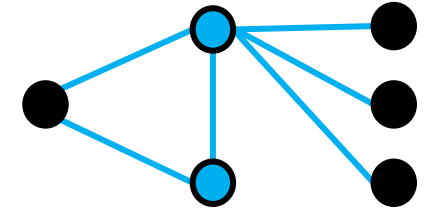
2. Objective function: minimize $\sum_{v \in V} y_v$

3. Constraints:

For every edge $(u, v) \in E$, need $y_u = 1$ and/or $y_v = 1$

In other words: $y_u + y_v \geq 1$

Vertex cover IP

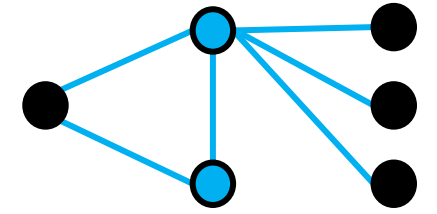


Input: Graph $G = (V, E)$ with vertex set V , edge set E

minimize $\sum_{v \in V} y_v$
subject to $y_u + y_v \geq 1$ for all $(u, v) \in E$
 $y_v \in \{0, 1\}$ for all $v \in V$

Binary integer program

LP relaxations



Input: Graph $G = (V, E)$ with vertex set V , edge set E

$$\begin{aligned} &\text{minimize} && \sum_{v \in V} y_v \\ &\text{subject to} && y_u + y_v \geq 1 \text{ for all } (u, v) \in E \\ & && 0 \leq y_v \leq 1 \\ & && \del{y_v \in \{0, 1\}} \end{aligned}$$

If you remove the integrality constraints,
you obtain the **LP relaxation** of the IP

LP relaxations

Integer program

$$\begin{aligned} \max \quad & \mathbf{c} \cdot \mathbf{x} \\ \text{s.t.} \quad & A\mathbf{x} \leq \mathbf{b} \\ & \mathbf{x} \in \mathbb{Z}^n \end{aligned}$$

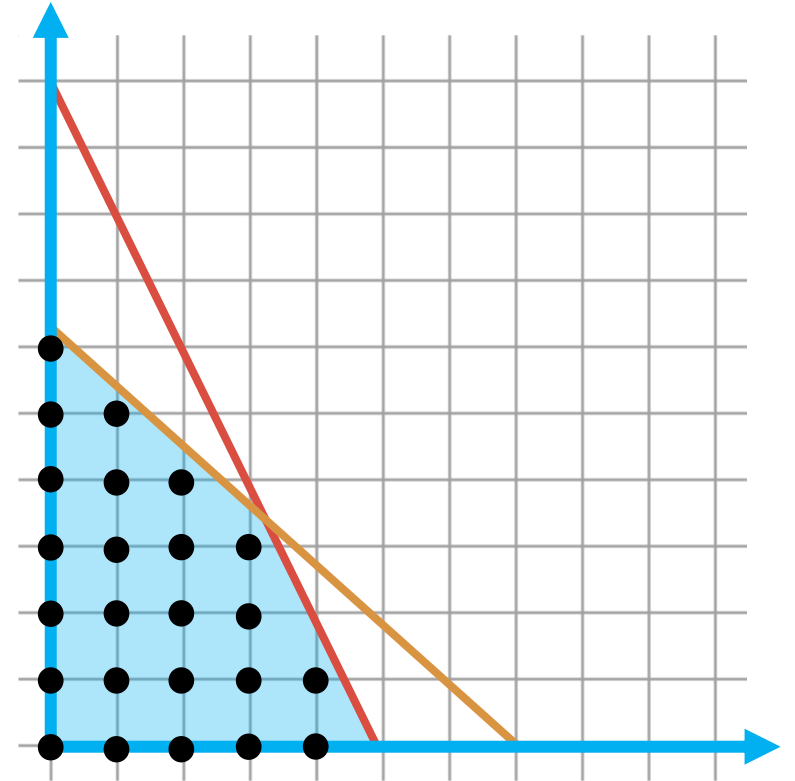
LP relaxation

$$\begin{aligned} \max \quad & \mathbf{c} \cdot \mathbf{x} \\ \text{s.t.} \quad & A\mathbf{x} \leq \mathbf{b} \\ & \cancel{\mathbf{x} \in \mathbb{Z}^n} \end{aligned}$$

\mathbf{x}_{IP}^* = optimal solution to IP

\mathbf{x}_{LP}^* = optimal solution to LP relaxation

Fact: $\mathbf{c} \cdot \mathbf{x}_{IP}^* \leq \mathbf{c} \cdot \mathbf{x}_{LP}^*$



LP relaxations

Integer program

$$\begin{array}{ll} \min & \mathbf{c} \cdot \mathbf{x} \\ \text{s.t.} & A\mathbf{x} \leq \mathbf{b} \\ & \mathbf{x} \in \mathbb{Z}^n \end{array}$$

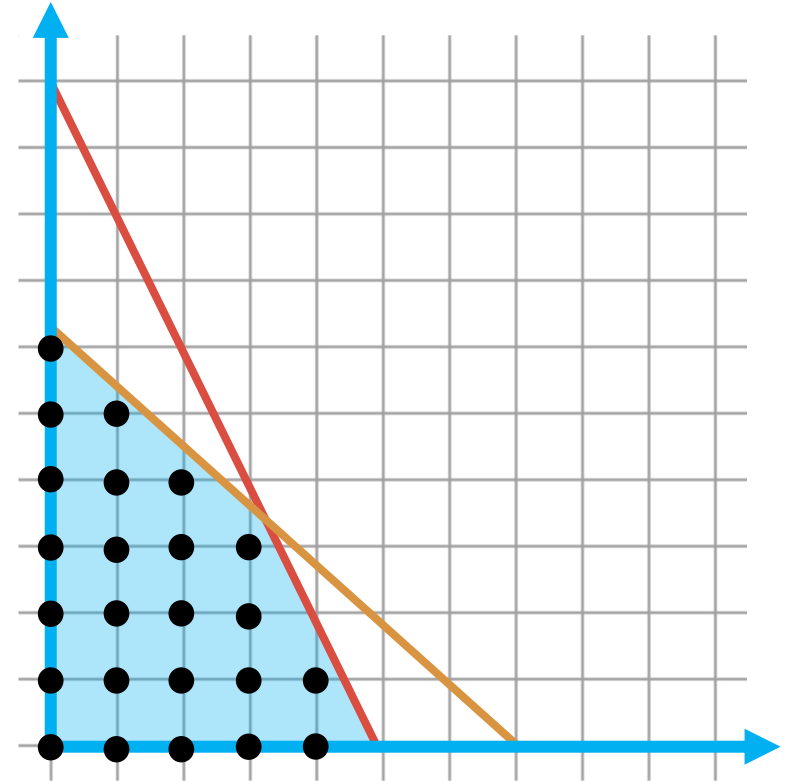
LP relaxation

$$\begin{array}{ll} \min & \mathbf{c} \cdot \mathbf{x} \\ \text{s.t.} & A\mathbf{x} \leq \mathbf{b} \\ & \cancel{\mathbf{x} \in \mathbb{Z}^n} \end{array}$$

\mathbf{x}_{IP}^* = optimal solution to IP

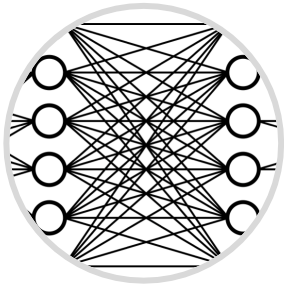
\mathbf{x}_{LP}^* = optimal solution to LP relaxation

Fact: $\mathbf{c} \cdot \mathbf{x}_{IP}^* \geq \mathbf{c} \cdot \mathbf{x}_{LP}^*$



Integer programming solvers

Most popular tool for solving combinatorial problems



Robust ML



Routing



Manufacturing



Scheduling



Planning

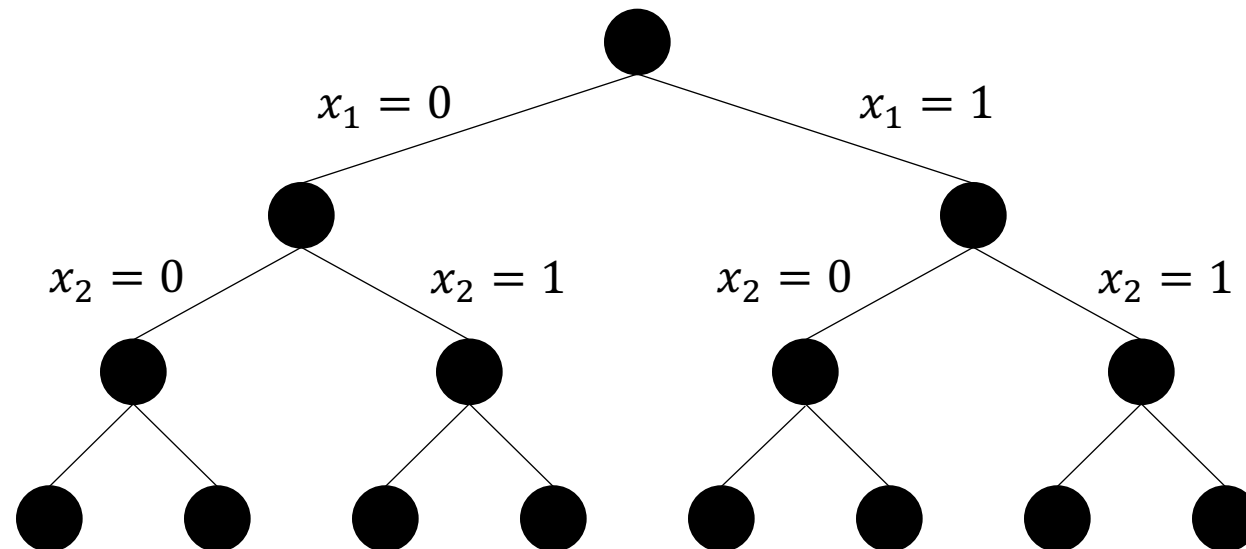
Branch-and-bound

$$\begin{aligned} &\text{maximize} && 15x_1 + 12x_2 + 4x_3 + 2x_4 \\ &\text{subject to} && 8x_1 + 5x_2 + 3x_3 + 2x_4 \leq 10 \\ &&& x_1, x_2, x_3, x_4 \in \{0,1\} \end{aligned}$$

Branch-and-bound

$$\begin{array}{ll} \text{maximize} & 15x_1 + 12x_2 + 4x_3 + 2x_4 \\ \text{subject to} & 8x_1 + 5x_2 + 3x_3 + 2x_4 \leq 10 \\ & x_1, x_2, x_3, x_4 \in \{0,1\} \end{array}$$

- *Enumeration tree*: enumerates all possible solutions of an IP
- At each node, *branch* on an integer variable
 - On each branch, integer variable is restricted to take certain values



Branch-and-bound

$$\begin{array}{ll} \text{maximize} & 15x_1 + 12x_2 + 4x_3 + 2x_4 \\ \text{subject to} & 8x_1 + 5x_2 + 3x_3 + 2x_4 \leq 10 \\ & x_1, x_2, x_3, x_4 \in \{0,1\} \end{array}$$

- *Enumeration tree*: enumerates all possible solutions of an IP
- If we can enumerate all solutions with the tree, why not compute objective for each solution and pick the best one?
- Would work, but # possible solutions explodes exponentially

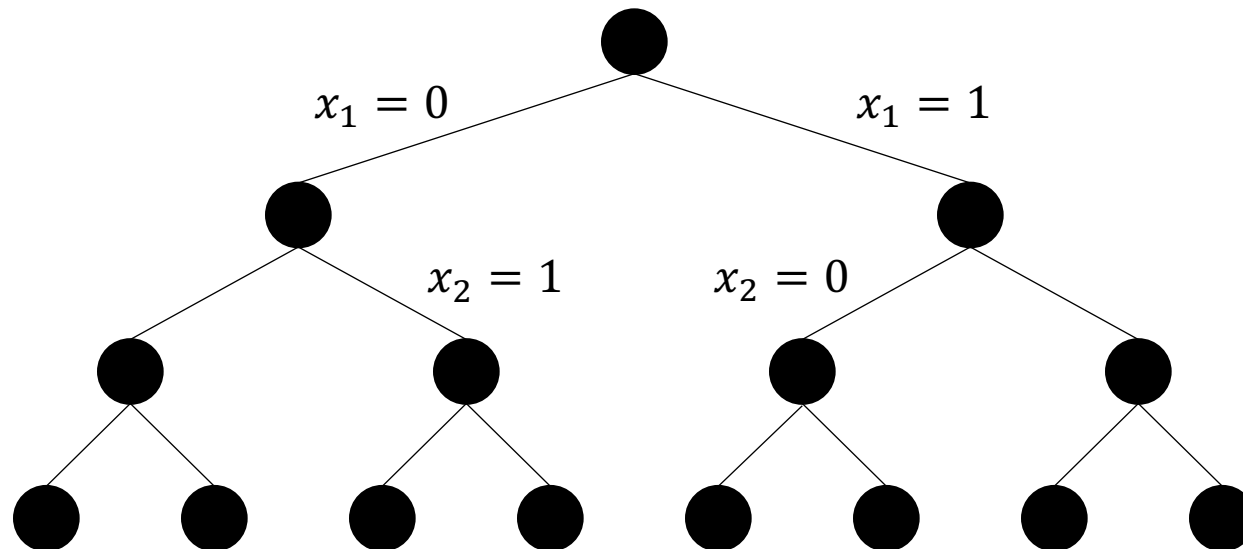
Branch-and-bound

maximize
subject to

$$15x_1 + 12x_2 + 4x_3 + 2x_4$$
$$8x_1 + 5x_2 + 3x_3 + 2x_4 \leq 10$$
$$x_1, x_2, x_3, x_4 \in \{0,1\}$$

Key idea of branch-and-bound (B&B):

- Using **LP relaxations**, bound the optimal integer solutions in **subtrees** of the enumeration tree



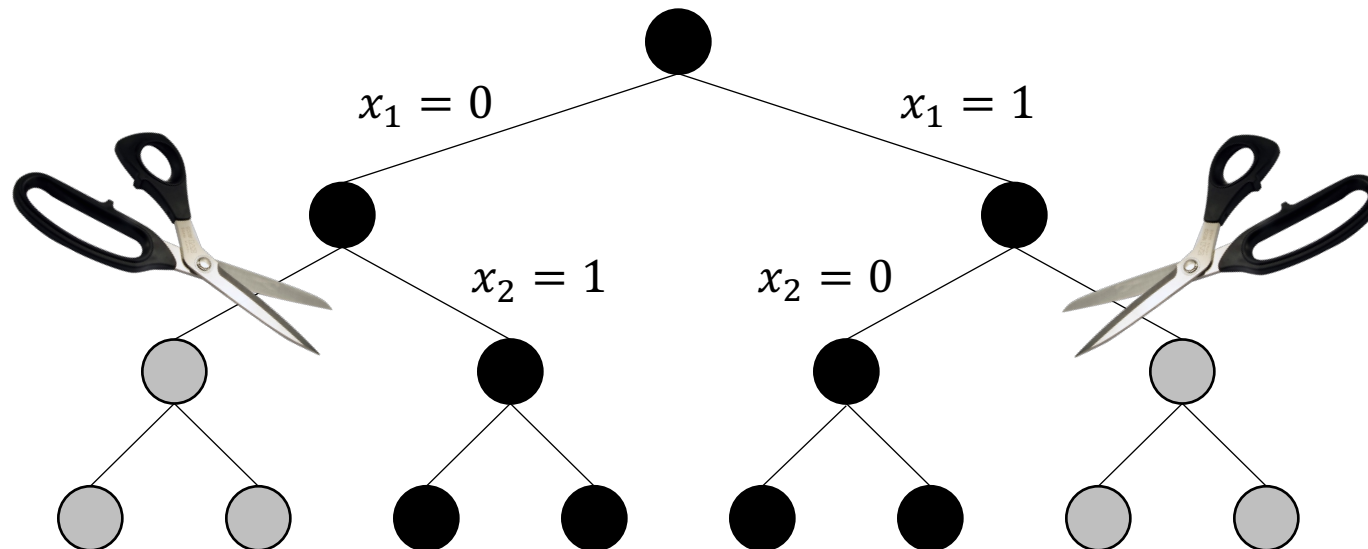
Branch-and-bound

maximize
subject to

$$\begin{aligned} &15x_1 + 12x_2 + 4x_3 + 2x_4 \\ &8x_1 + 5x_2 + 3x_3 + 2x_4 \leq 10 \\ &x_1, x_2, x_3, x_4 \in \{0,1\} \end{aligned}$$

Key idea of branch-and-bound (B&B):

- Using **LP relaxations**, bound the optimal integer solutions in **subtrees** of the enumeration tree
- Allows us to eliminate a lot of the enumeration tree



Branch-and-bound

$$\begin{array}{ll} \text{maximize} & 15x_1 + 12x_2 + 4x_3 + 2x_4 \\ \text{subject to} & 8x_1 + 5x_2 + 3x_3 + 2x_4 \leq 10 \\ & x_1, x_2, x_3, x_4 \in \{0,1\} \end{array}$$

To start, we assume we have a feasible solution \mathbf{x}^*

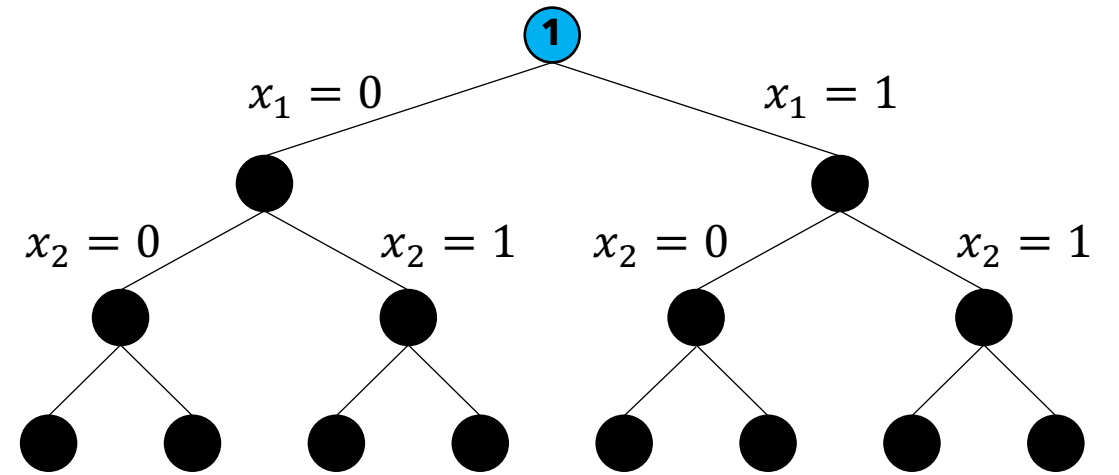
- E.g., $\mathbf{x}^* = (0,0,0,0)$

At each iteration of B&B:

- \mathbf{x}^* is the *incumbent* solution
- Its objective value z^* is the *incumbent objective*

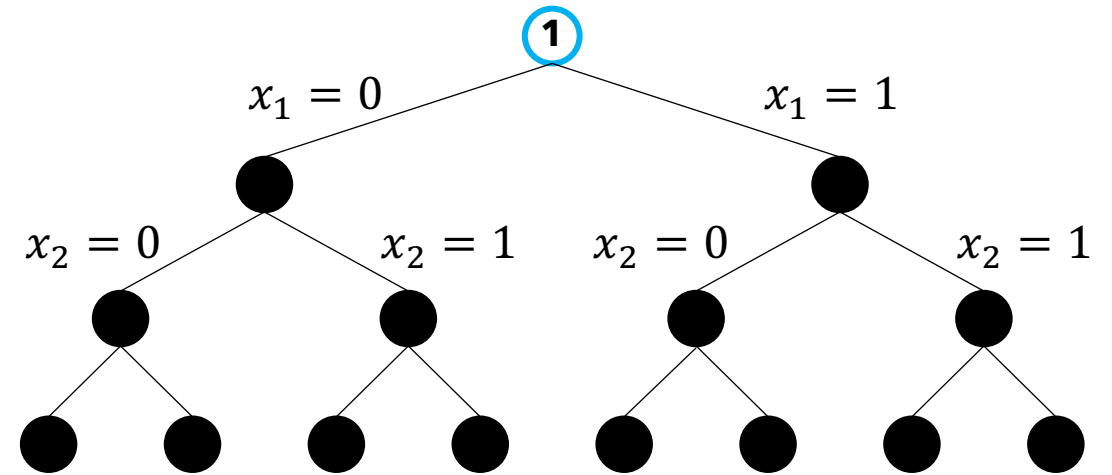
Here, incumbent means "best so far"

Branch-and-bound



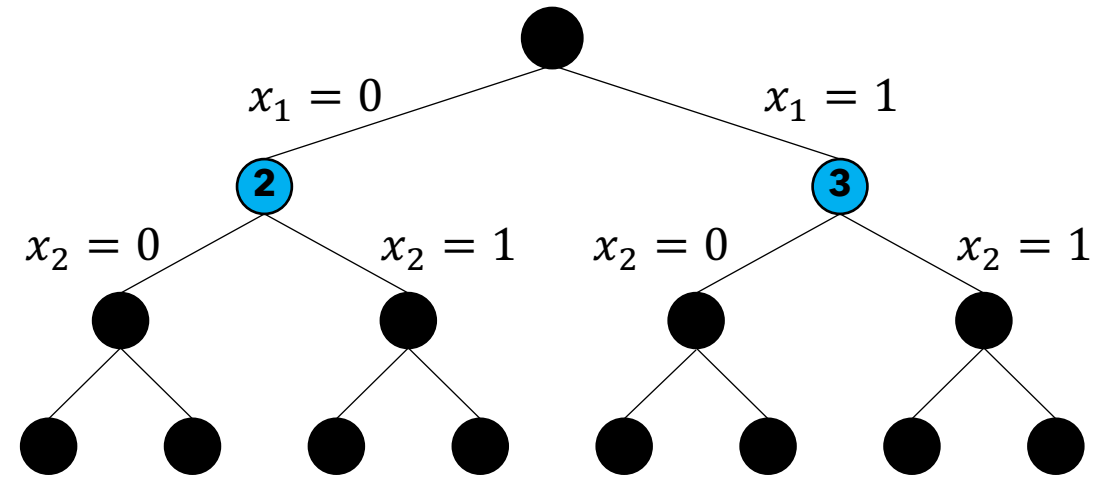
1. Mark the root node as active
2. While there remain active nodes:

Branch-and-bound



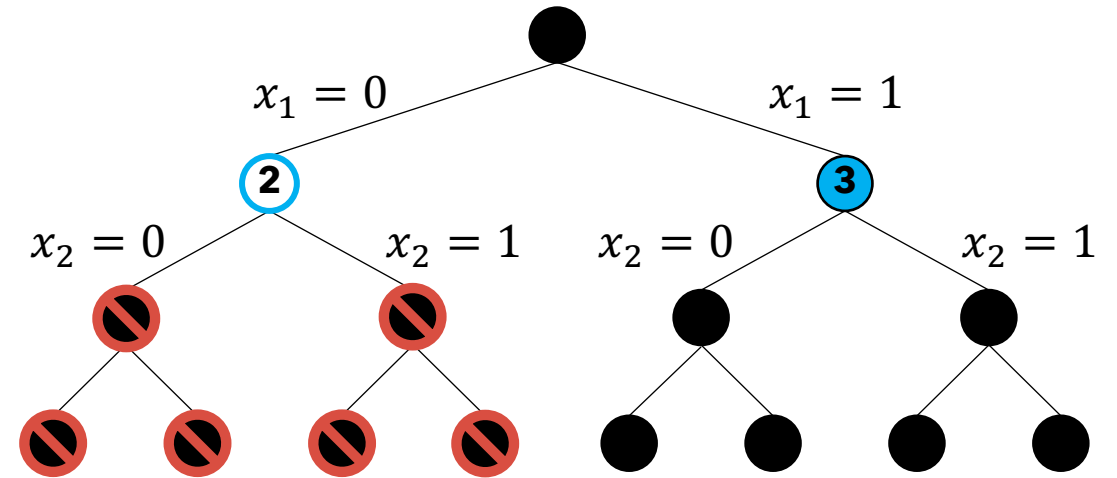
1. Mark the root node as active
2. While there remain active nodes:
 - i. Select an active node j and mark it as inactive
 - ii. $\mathbf{x}(j)$ = optimal solution of LP relaxation of Problem(j)
 - iii. $z(j)$ = objective value of $\mathbf{x}(j)$
 - iv. Case 1: If $z^* < z(j)$ and $\mathbf{x}(j)$ isn't feasible for IP then
Mark the direct descendants of node j as active
Possible to find a better incumbent solution among j 's descendants

Branch-and-bound



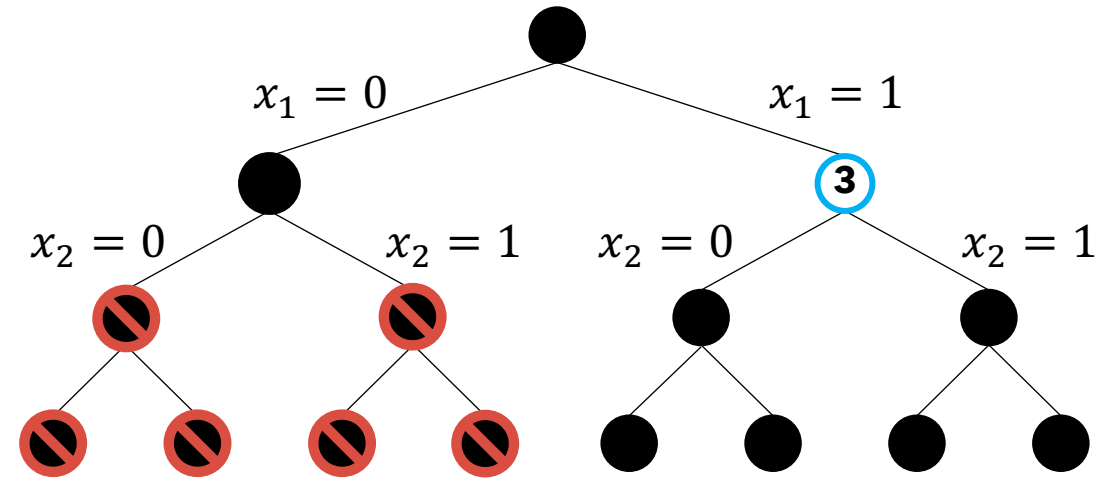
1. Mark the root node as active
2. While there remain active nodes:
 - i. Select an active node j and mark it as inactive
 - ii. $\mathbf{x}(j)$ = optimal solution of LP relaxation of Problem(j)
 - iii. $z(j)$ = objective value of $\mathbf{x}(j)$
 - iv. Case 1: If $z^* < z(j)$ and $\mathbf{x}(j)$ isn't feasible for IP then
 - Mark the direct descendants of node j as active
 - Possible to find a better incumbent solution among j 's descendants*

Branch-and-bound



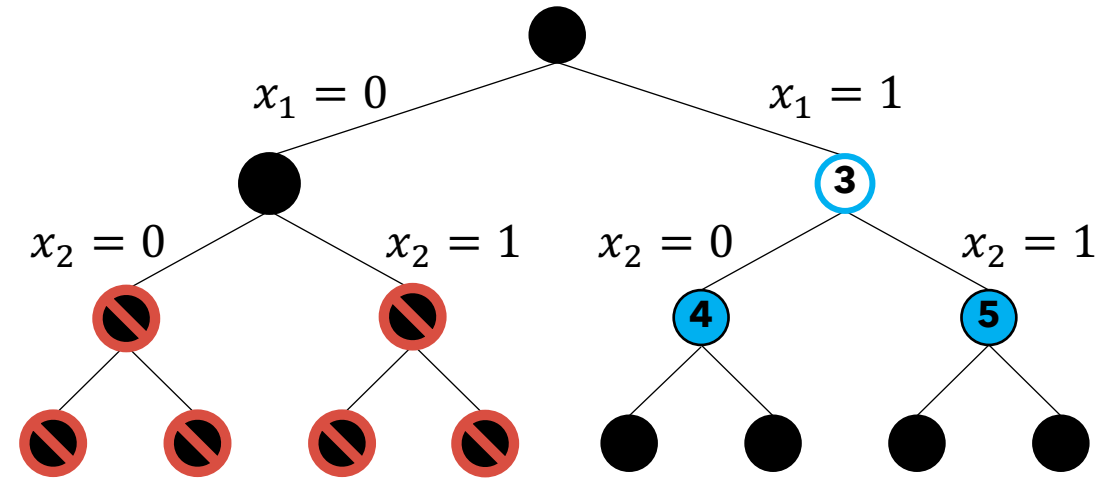
1. Mark the root node as active
2. While there remain active nodes:
 - i. Select an active node j and mark it as inactive
 - ii. $\mathbf{x}(j)$ = optimal solution of LP relaxation of Problem(j)
 - iii. $z(j)$ = objective value of $\mathbf{x}(j)$
 - iv. Case 1: If $z^* < z(j)$ and $\mathbf{x}(j)$ isn't feasible for IP then
Mark the direct descendants of node j as active
 - v. Case 2: If $z^* < z(j)$ and $\mathbf{x}(j)$ is feasible for IP then
Replace the incumbent by $\mathbf{x}(j)$ and prune node j
Could be the optimal solution!

Branch-and-bound



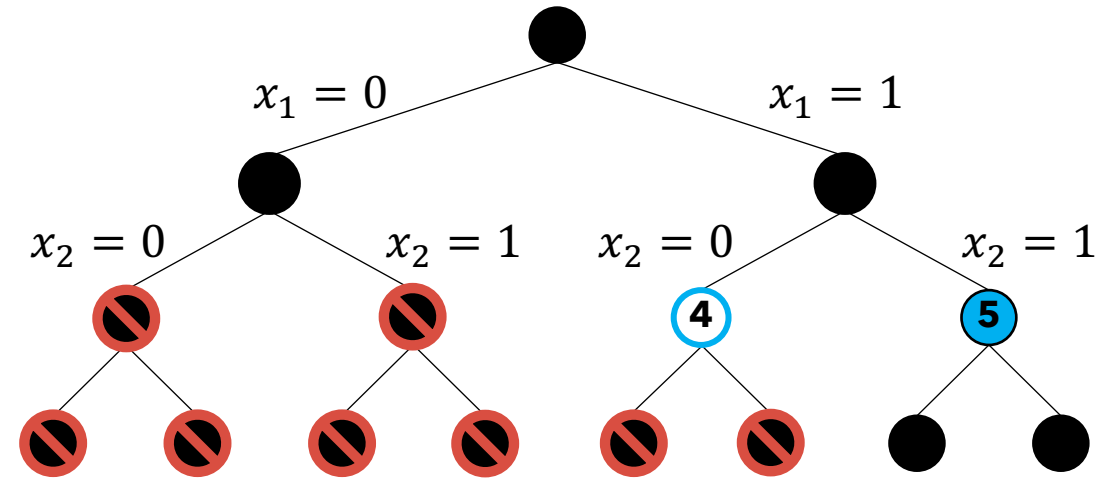
1. Mark the root node as active
2. While there remain active nodes:
 - i. Select an active node j and mark it as inactive
 - ii. $\mathbf{x}(j)$ = optimal solution of LP relaxation of Problem(j)
 - iii. $z(j)$ = objective value of $\mathbf{x}(j)$
 - iv. Case 1: If $z^* < z(j)$ and $\mathbf{x}(j)$ isn't feasible for IP then
Mark the direct descendants of node j as active
 - v. Case 2: If $z^* < z(j)$ and $\mathbf{x}(j)$ is feasible for IP then
Replace the incumbent by $\mathbf{x}(j)$ and prune node j

Branch-and-bound



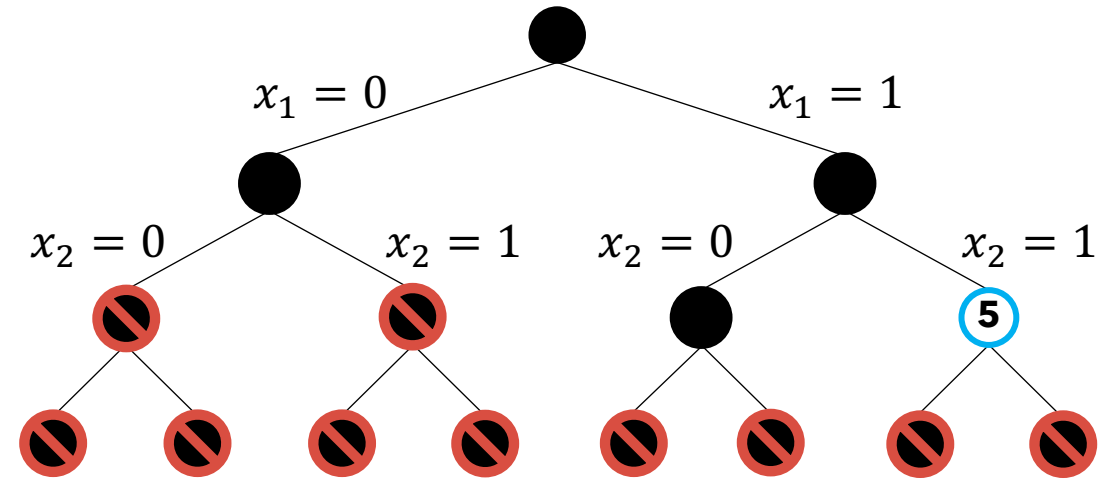
1. Mark the root node as active
2. While there remain active nodes:
 - i. Select an active node j and mark it as inactive
 - ii. $\mathbf{x}(j)$ = optimal solution of LP relaxation of Problem(j)
 - iii. $z(j)$ = objective value of $\mathbf{x}(j)$
 - iv. Case 1: If $z^* < z(j)$ and $\mathbf{x}(j)$ isn't feasible for IP then
Mark the direct descendants of node j as active
 - v. Case 2: If $z^* < z(j)$ and $\mathbf{x}(j)$ is feasible for IP then
Replace the incumbent by $\mathbf{x}(j)$ and prune node j

Branch-and-bound



1. Mark the root node as active
2. While there remain active nodes:
 - i. Select an active node j and mark it as inactive
 - ii. $\mathbf{x}(j)$ = optimal solution of LP relaxation of Problem(j)
 - iii. $z(j)$ = objective value of $\mathbf{x}(j)$
 - iv. Case 1: If $z^* < z(j)$ and $\mathbf{x}(j)$ isn't feasible for IP then
Mark the direct descendants of node j as active
 - v. Case 2: If $z^* < z(j)$ and $\mathbf{x}(j)$ is feasible for IP then
Replace the incumbent by $\mathbf{x}(j)$ and prune node j
 - vi. Case 3: If LP is infeasible or $z^* \geq z(j)$ then prune node j

Branch-and-bound



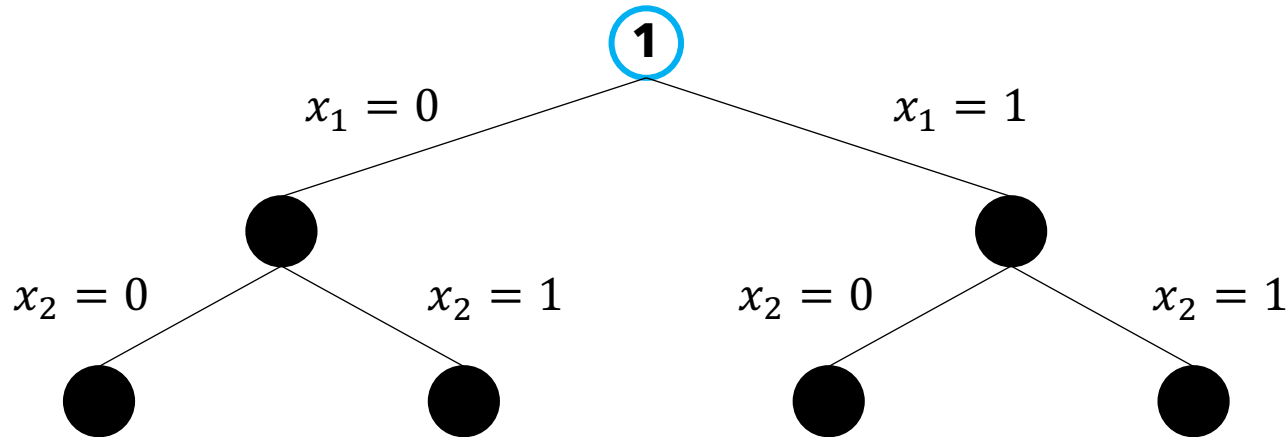
1. Mark the root node as active
2. While there remain active nodes:
 - i. Select an active node j and mark it as inactive
 - ii. $\mathbf{x}(j)$ = optimal solution of LP relaxation of Problem(j)
 - iii. $z(j)$ = objective value of $\mathbf{x}(j)$
 - iv. Case 1: If $z^* < z(j)$ and $\mathbf{x}(j)$ isn't feasible for IP then
Mark the direct descendants of node j as active
 - v. Case 2: If $z^* < z(j)$ and $\mathbf{x}(j)$ is feasible for IP then
Replace the incumbent by $\mathbf{x}(j)$ and prune node j
 - vi. Case 3: If LP is infeasible or $z^* \geq z(j)$ then prune node j

Branch-and-bound

maximize
subject to

$$\begin{aligned} &15x_1 + 12x_2 + 4x_3 + 2x_4 \\ &8x_1 + 5x_2 + 3x_3 + 2x_4 \leq 10 \\ &x_1, x_2, x_3, x_4 \in \{0,1\} \end{aligned}$$

Incumbent: $\mathbf{x}^* = (0,0,0,0)$
 $z^* = 0$



Case 1: If $z^* < z(j)$ and $\mathbf{x}(j)$ isn't feasible for IP then
Mark the direct descendants of node j as active

Problem(1):

$$\begin{aligned} \max & 15x_1 + 12x_2 + 4x_3 + 2x_4 \\ \text{s.t} & 8x_1 + 5x_2 + 3x_3 + 2x_4 \leq 10 \\ & x_1, x_2, x_3, x_4 \in [0,1] \end{aligned}$$

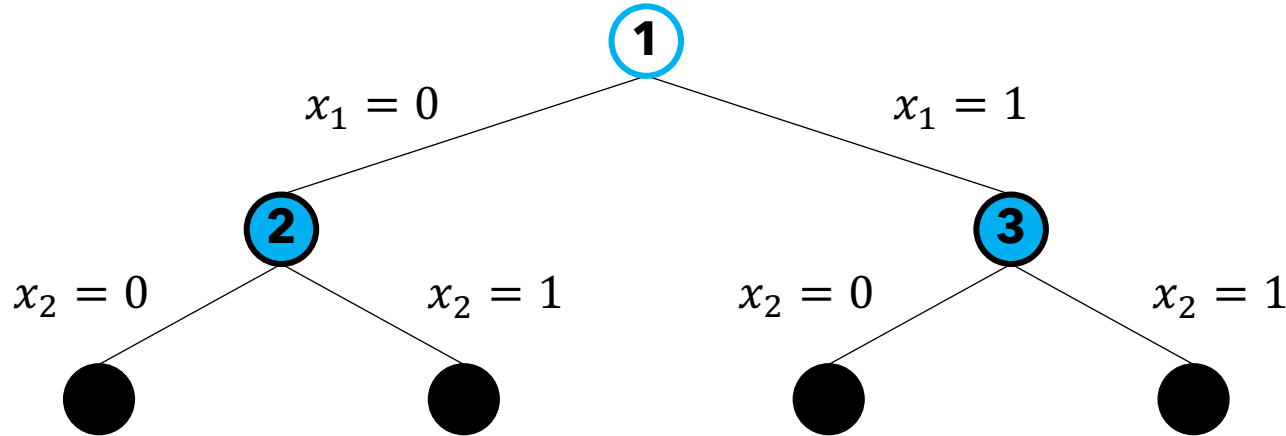
$$\begin{aligned} \mathbf{x}(1) &= \left(\frac{5}{8}, 1, 0, 0\right) \\ z(1) &= 21.38 \end{aligned}$$

Branch-and-bound

maximize
subject to

$$\begin{aligned} &15x_1 + 12x_2 + 4x_3 + 2x_4 \\ &8x_1 + 5x_2 + 3x_3 + 2x_4 \leq 10 \\ &x_1, x_2, x_3, x_4 \in \{0,1\} \end{aligned}$$

Incumbent: $\mathbf{x}^* = (0,0,0,0)$
 $z^* = 0$



Case 1: If $z^* < z(j)$ and $\mathbf{x}(j)$ isn't feasible for IP then
Mark the direct descendants of node j as active

Problem(1):

$$\begin{aligned} \max & 15x_1 + 12x_2 + 4x_3 + 2x_4 \\ \text{s.t} & 8x_1 + 5x_2 + 3x_3 + 2x_4 \leq 10 \\ & x_1, x_2, x_3, x_4 \in [0,1] \end{aligned}$$

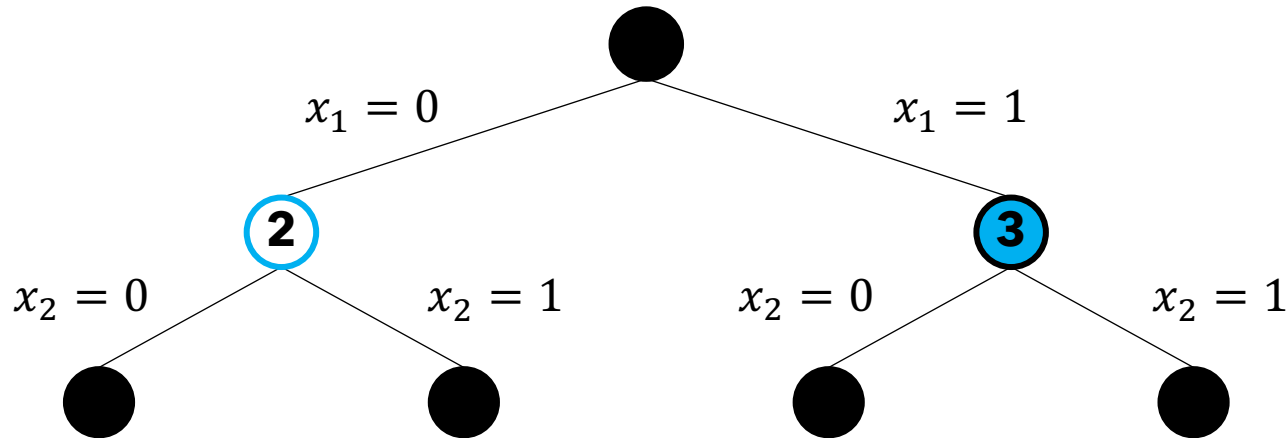
$$\begin{aligned} \mathbf{x}(1) &= \left(\frac{5}{8}, 1, 0, 0\right) \\ z(1) &= 21.38 \end{aligned}$$

Branch-and-bound

maximize
subject to

$$\begin{aligned} &15x_1 + 12x_2 + 4x_3 + 2x_4 \\ &8x_1 + 5x_2 + 3x_3 + 2x_4 \leq 10 \\ &x_1, x_2, x_3, x_4 \in \{0,1\} \end{aligned}$$

Incumbent: $\mathbf{x}^* = (0,0,0,0)$
 $z^* = 0$



Case 2: If $z^* < z(j)$ and $\mathbf{x}(j)$ is feasible for IP then
Replace the incumbent by $\mathbf{x}(j)$ and prune node j

Problem(2):

$$\begin{aligned} \max & 15x_1 + 12x_2 + 4x_3 + 2x_4 \\ \text{s.t} & 8x_1 + 5x_2 + 3x_3 + 2x_4 \leq 10 \\ & x_1 = 0 \\ & x_2, x_3, x_4 \in [0,1] \end{aligned}$$

$$\begin{aligned} \mathbf{x}(2) &= (0,1,1,1) \\ z(2) &= 18 \end{aligned}$$

Branch-and-bound

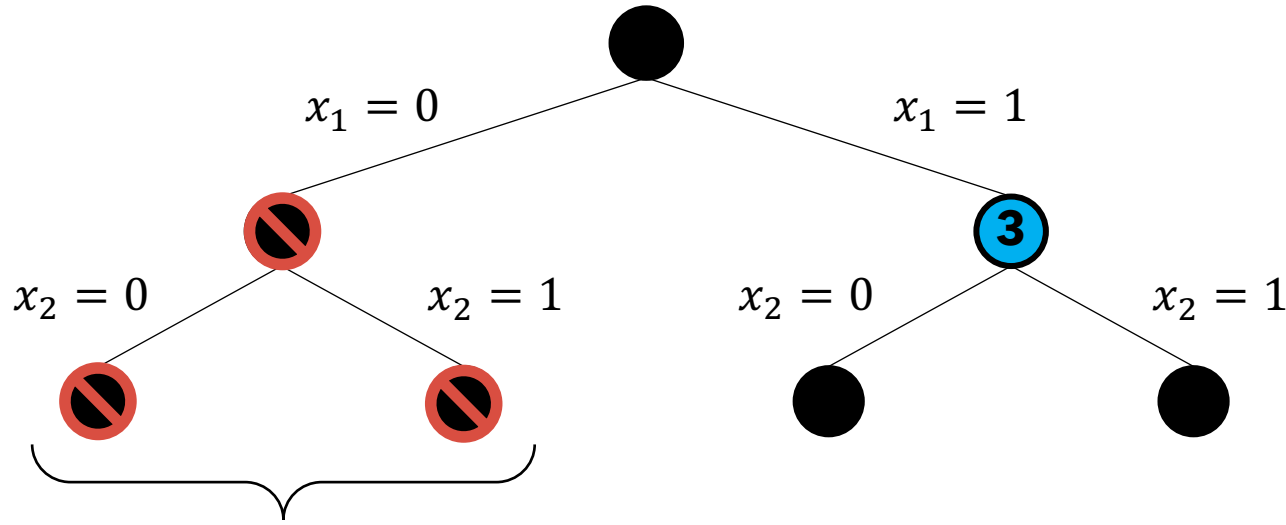
maximize
subject to

$$15x_1 + 12x_2 + 4x_3 + 2x_4$$

$$8x_1 + 5x_2 + 3x_3 + 2x_4 \leq 10$$

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

Incumbent: $\mathbf{x}^* = (0,1,1,1)$
 $z^* = 18$



Can't find better feasible solution in this subtree

Case 2: If $z^* < z(j)$ and $\mathbf{x}(j)$ is feasible for IP then
Replace the incumbent by $\mathbf{x}(j)$ and prune node j

Problem(2):

$$\max \quad 15x_1 + 12x_2 + 4x_3 + 2x_4$$

$$\text{s.t} \quad 8x_1 + 5x_2 + 3x_3 + 2x_4 \leq 10$$

$$x_1 = 0$$

$$x_2, x_3, x_4 \in [0,1]$$

$$\mathbf{x}(2) = (0,1,1,1)$$

$$z(2) = 18$$

Branch-and-bound

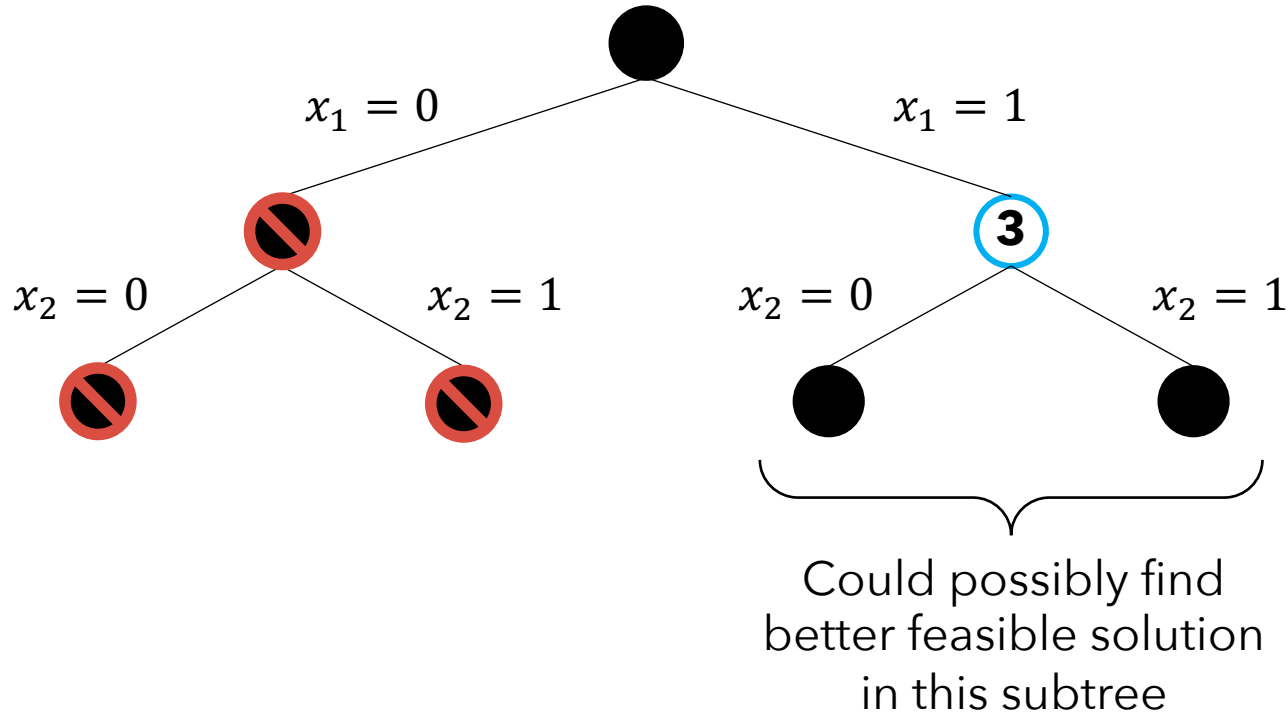
maximize
subject to

$$15x_1 + 12x_2 + 4x_3 + 2x_4$$

$$8x_1 + 5x_2 + 3x_3 + 2x_4 \leq 10$$

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

Incumbent: $\mathbf{x}^* = (0,1,1,1)$
 $z^* = 18$



Case 1: If $z^* < z(j)$ and $\mathbf{x}(j)$ isn't feasible for IP then
Mark the direct descendants of node j as active

Problem(3):

$$\max \quad 15x_1 + 12x_2 + 4x_3 + 2x_4$$

$$\text{s.t} \quad 8x_1 + 5x_2 + 3x_3 + 2x_4 \leq 10$$

$$x_1 = 1$$

$$x_2, x_3, x_4 \in [0,1]$$

$$\mathbf{x}(3) = \left(\frac{5}{8}, 1, 0, 0\right)$$

$$z(3) = 21.38$$

Branch-and-bound

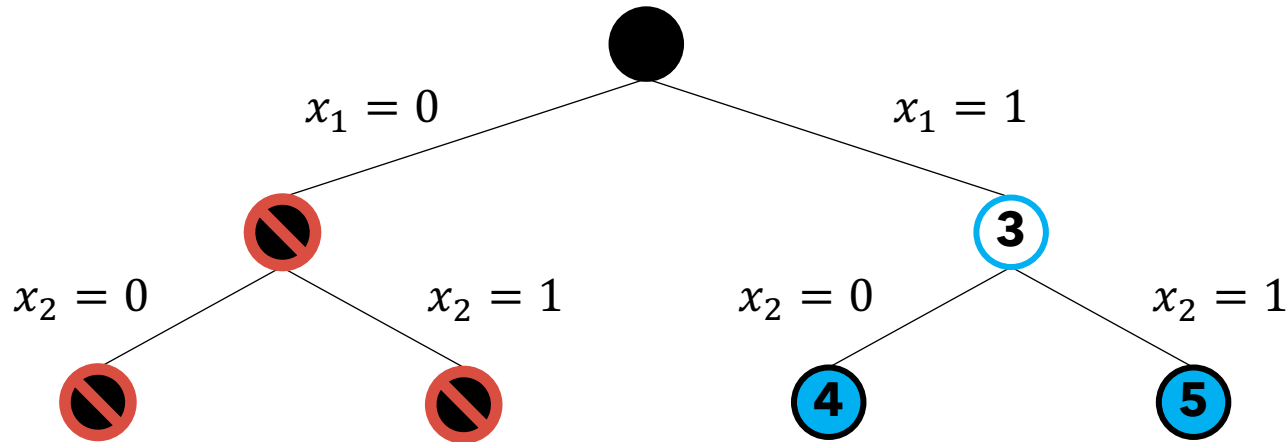
maximize
subject to

$$15x_1 + 12x_2 + 4x_3 + 2x_4$$

$$8x_1 + 5x_2 + 3x_3 + 2x_4 \leq 10$$

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

Incumbent: $\mathbf{x}^* = (0,1,1,1)$
 $z^* = 18$



Case 1: If $z^* < z(j)$ and $\mathbf{x}(j)$ isn't feasible for IP then
Mark the direct descendants of node j as active

Problem(3):

$$\max \quad 15x_1 + 12x_2 + 4x_3 + 2x_4$$

$$\text{s.t} \quad 8x_1 + 5x_2 + 3x_3 + 2x_4 \leq 10$$

$$x_1 = 1$$

$$x_2, x_3, x_4 \in [0,1]$$

$$\mathbf{x}(3) = \left(\frac{5}{8}, 1, 0, 0\right)$$

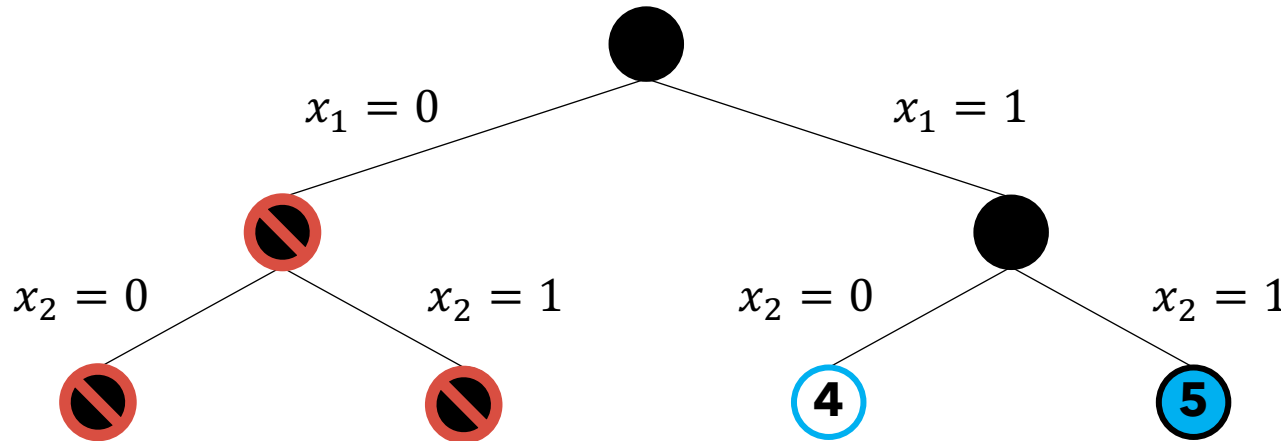
$$z(3) = 21.38$$

Branch-and-bound

maximize
subject to

$$\begin{aligned} & 15x_1 + 12x_2 + 4x_3 + 2x_4 \\ & 8x_1 + 5x_2 + 3x_3 + 2x_4 \leq 10 \\ & x_1, x_2, x_3, x_4 \in \{0,1\} \end{aligned}$$

Incumbent: $\mathbf{x}^* = (0,1,1,1)$
 $z^* = 18$



Problem(4):

$$\begin{aligned} \max & 15x_1 + 12x_2 + 4x_3 + 2x_4 \\ \text{s.t} & 8x_1 + 5x_2 + 3x_3 + 2x_4 \leq 10 \\ & x_1 = 1 \\ & x_2 = 0 \\ & x_3, x_4 \in [0,1] \end{aligned}$$

$$\begin{aligned} \mathbf{x}(4) &= \left(1, 0, \frac{2}{3}, 0\right) \\ z(4) &= 17.66 \end{aligned}$$

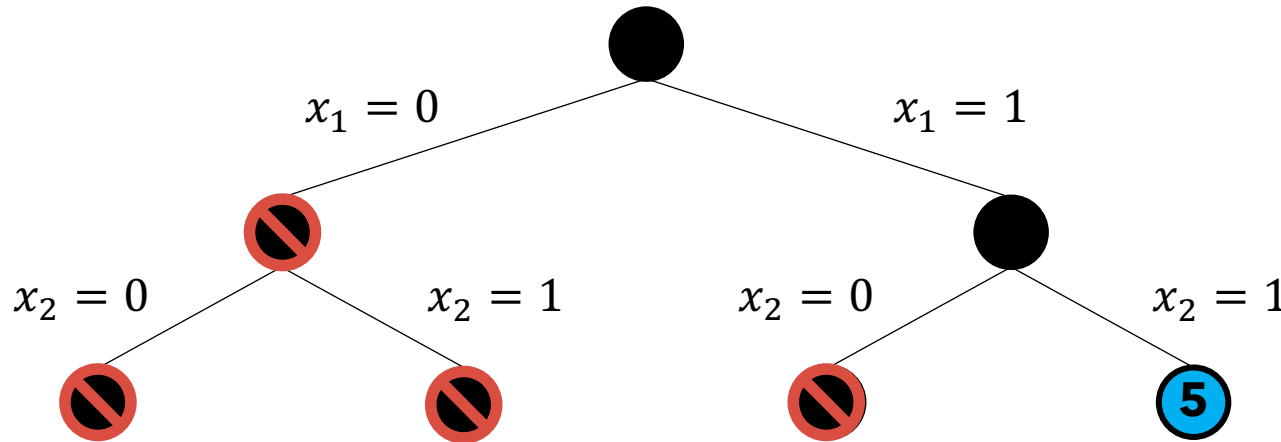
Case 3: If LP is infeasible or $z^* \geq z(j)$ then prune node j

Branch-and-bound

maximize
subject to

$$\begin{aligned} &15x_1 + 12x_2 + 4x_3 + 2x_4 \\ &8x_1 + 5x_2 + 3x_3 + 2x_4 \leq 10 \\ &x_1, x_2, x_3, x_4 \in \{0,1\} \end{aligned}$$

Incumbent: $\mathbf{x}^* = (0,1,1,1)$
 $z^* = 18$



Problem(4):

$$\begin{aligned} \max & 15x_1 + 12x_2 + 4x_3 + 2x_4 \\ \text{s.t} & 8x_1 + 5x_2 + 3x_3 + 2x_4 \leq 10 \\ & x_1 = 1 \\ & x_2 = 0 \\ & x_3, x_4 \in [0,1] \end{aligned}$$

$$\mathbf{x}(4) = \left(1, 0, \frac{2}{3}, 0\right)$$

$$z(4) = 17.66$$

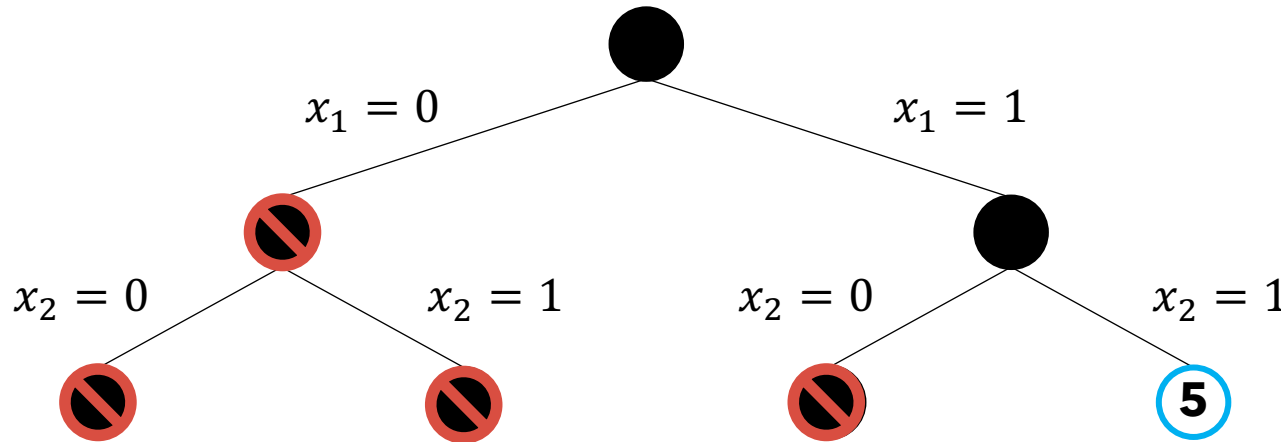
Case 3: If LP is infeasible or $z^* \geq z(j)$ then prune node j

Branch-and-bound

maximize
subject to

$$\begin{aligned} & 15x_1 + 12x_2 + 4x_3 + 2x_4 \\ & 8x_1 + 5x_2 + 3x_3 + 2x_4 \leq 10 \\ & x_1, x_2, x_3, x_4 \in \{0,1\} \end{aligned}$$

Incumbent: $\mathbf{x}^* = (0,1,1,1)$
 $z^* = 18$



Problem(5):

$$\begin{aligned} \max & 15x_1 + 12x_2 + 4x_3 + 2x_4 \\ \text{s.t} & 8x_1 + 5x_2 + 3x_3 + 2x_4 \leq 10 \\ & x_1 = 1 \\ & x_2 = 1 \\ & x_3, x_4 \in [0,1] \end{aligned}$$

$\mathbf{x}(5) = \text{infeasible}$

$z(5) = \text{infeasible}$

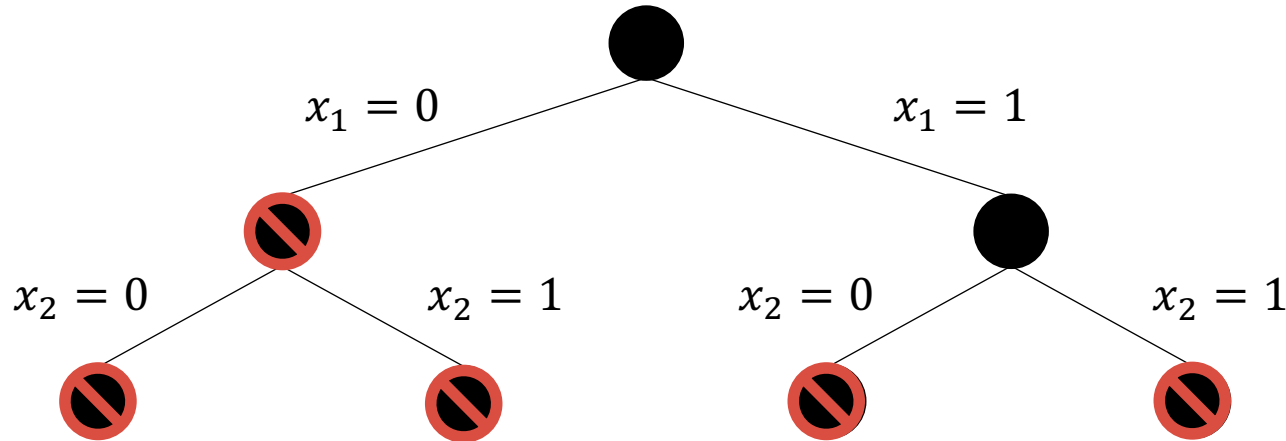
Case 3: If LP is infeasible or $z^* \geq z(j)$ then prune node j

Branch-and-bound

maximize
subject to

$$\begin{aligned} & 15x_1 + 12x_2 + 4x_3 + 2x_4 \\ & 8x_1 + 5x_2 + 3x_3 + 2x_4 \leq 10 \\ & x_1, x_2, x_3, x_4 \in \{0,1\} \end{aligned}$$

Incumbent: $\mathbf{x}^* = (0,1,1,1)$
 $z^* = 18$



Problem(5):

$$\begin{aligned} \max & 15x_1 + 12x_2 + 4x_3 + 2x_4 \\ \text{s.t} & 8x_1 + 5x_2 + 3x_3 + 2x_4 \leq 10 \\ & x_1 = 1 \\ & x_2 = 1 \\ & x_3, x_4 \in [0,1] \end{aligned}$$

$\mathbf{x}(5) = \text{infeasible}$

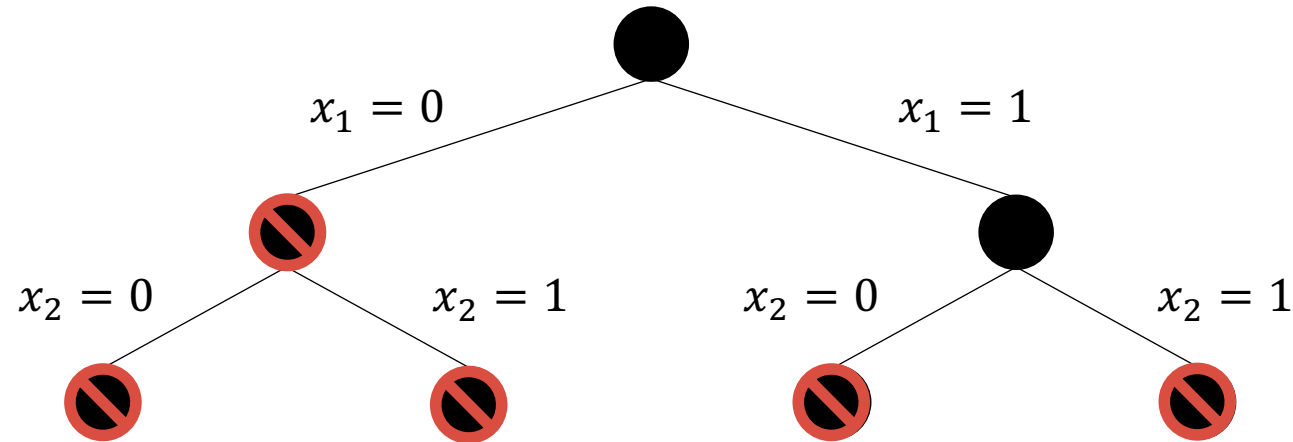
$z(5) = \text{infeasible}$

Case 3: If LP is infeasible or $z^* \geq z(j)$ then prune node j

Branch-and-bound

maximize
subject to

$$\begin{aligned} &15x_1 + 12x_2 + 4x_3 + 2x_4 \\ &8x_1 + 5x_2 + 3x_3 + 2x_4 \leq 10 \\ &x_1, x_2, x_3, x_4 \in \{0,1\} \end{aligned}$$



Incumbent: $x^* = (0,1,1,1)$
 $z^* = 18$

Optimal solution

Major challenge of using B&B

Many different ways to configure/optimize this algorithm, e.g.:

- Node selection policy
- Variable selection policy
- ...

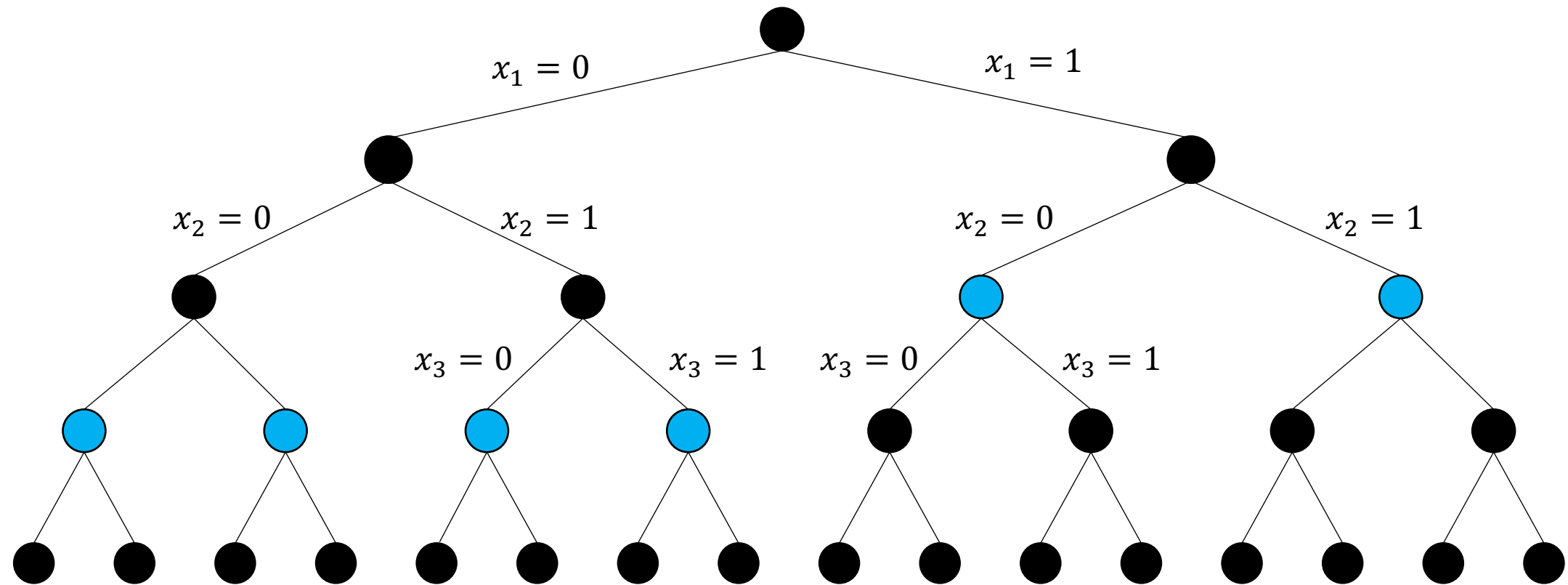
What's the best **configuration** for the application at hand?



Best configuration for **routing** problems
likely not suited for **scheduling**



Node selection policy

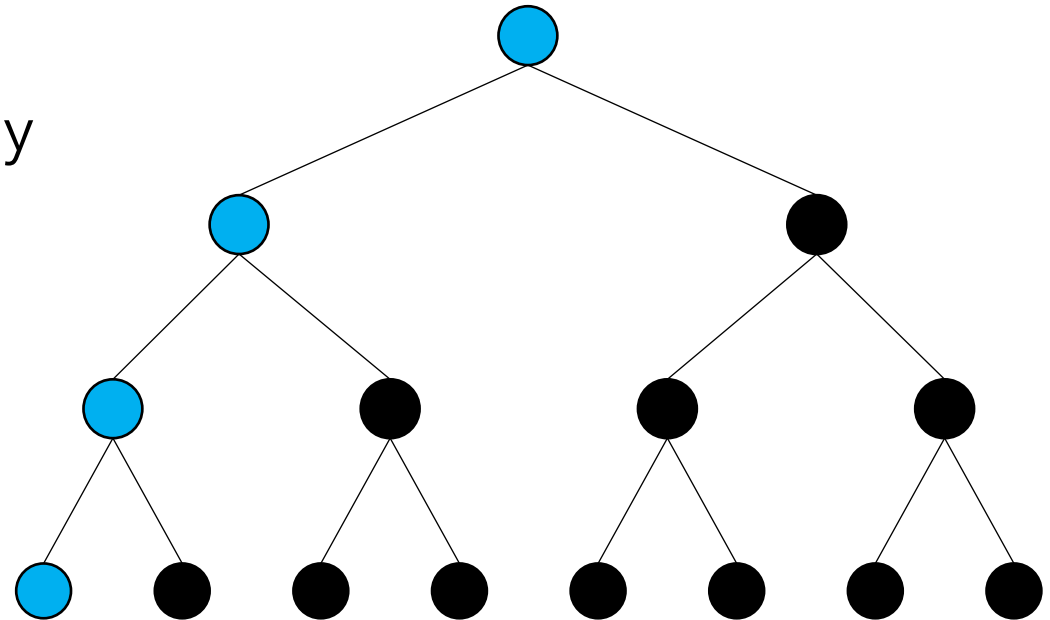


Among many active nodes, which to explore next?

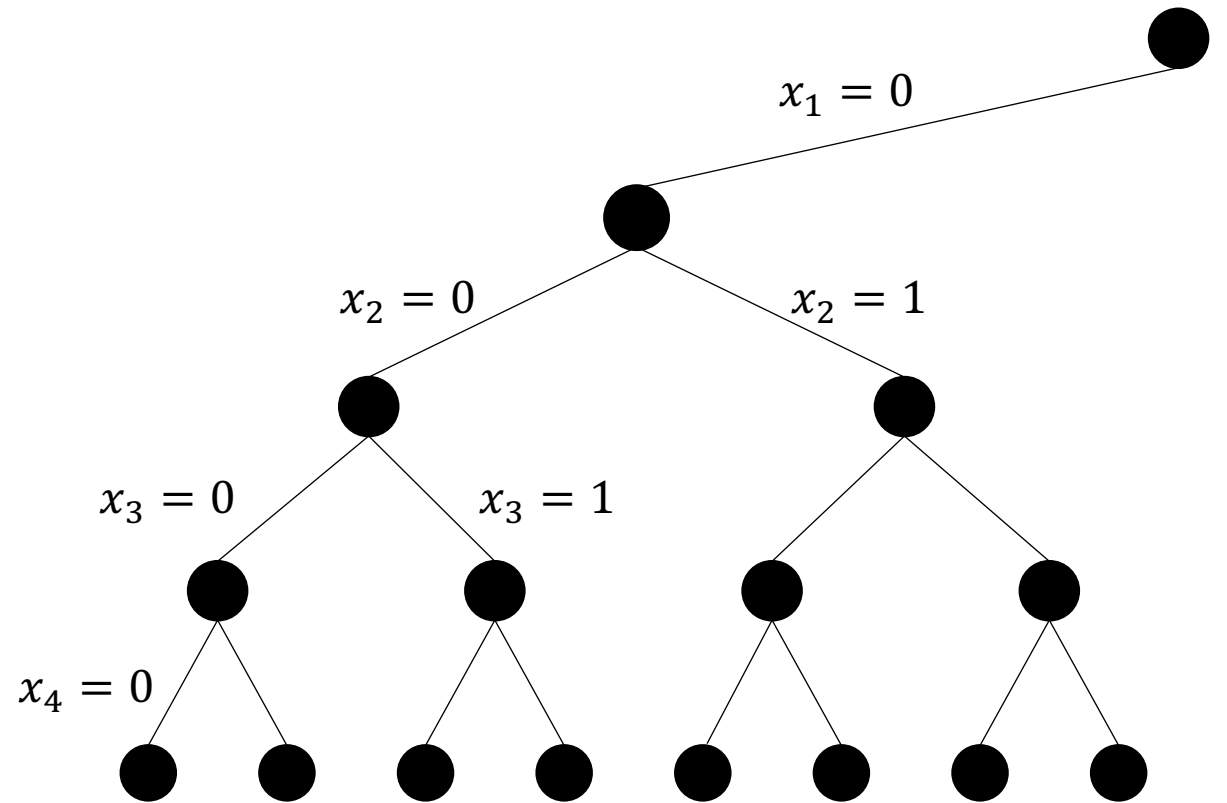
Node selection policy

Among many active nodes, which to explore next?

- *Depth-first search (DFS)*
 - Finds incumbent solutions quickly
- *Best-first search (BFS)*
 - Explore node with highest LP objective value
 - The "most promising" nodes
- *BFS with plunging*
 - Mix of BFS and DFS

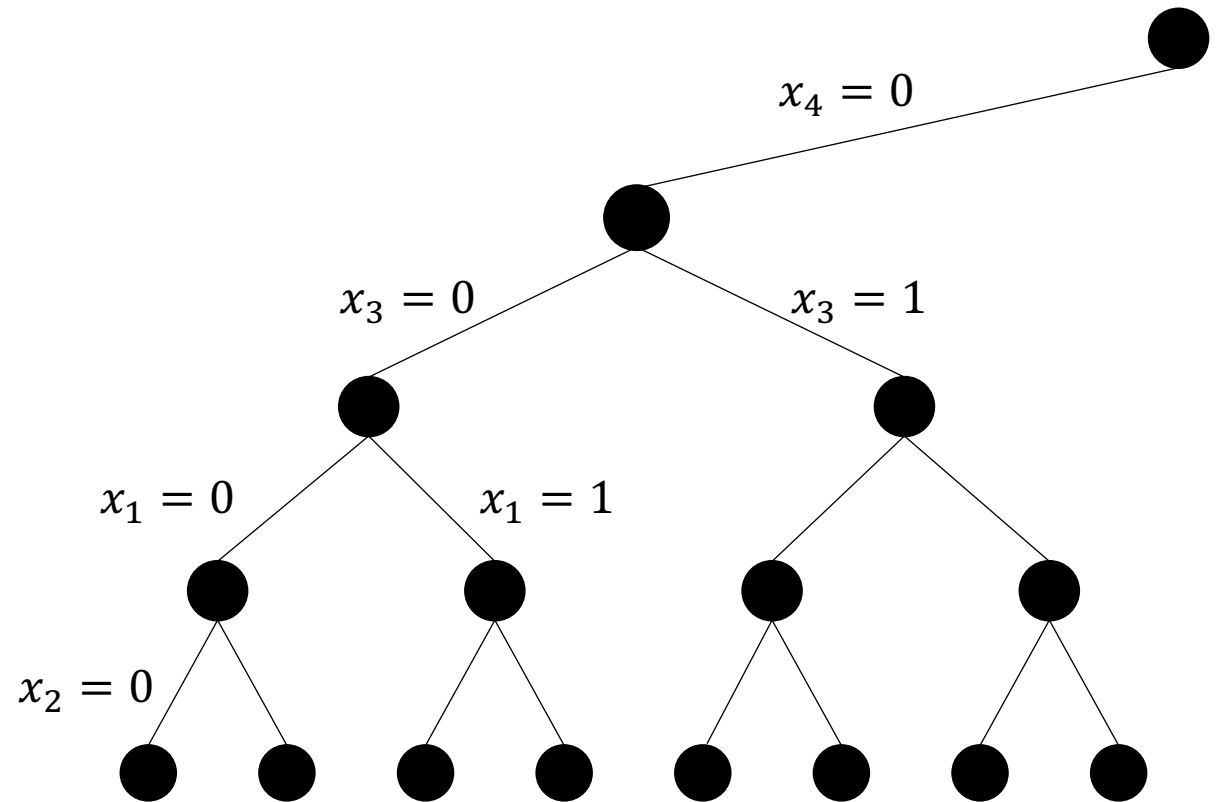


Variable selection policy



Better branching order than x_1, x_2, x_3, x_4 ?

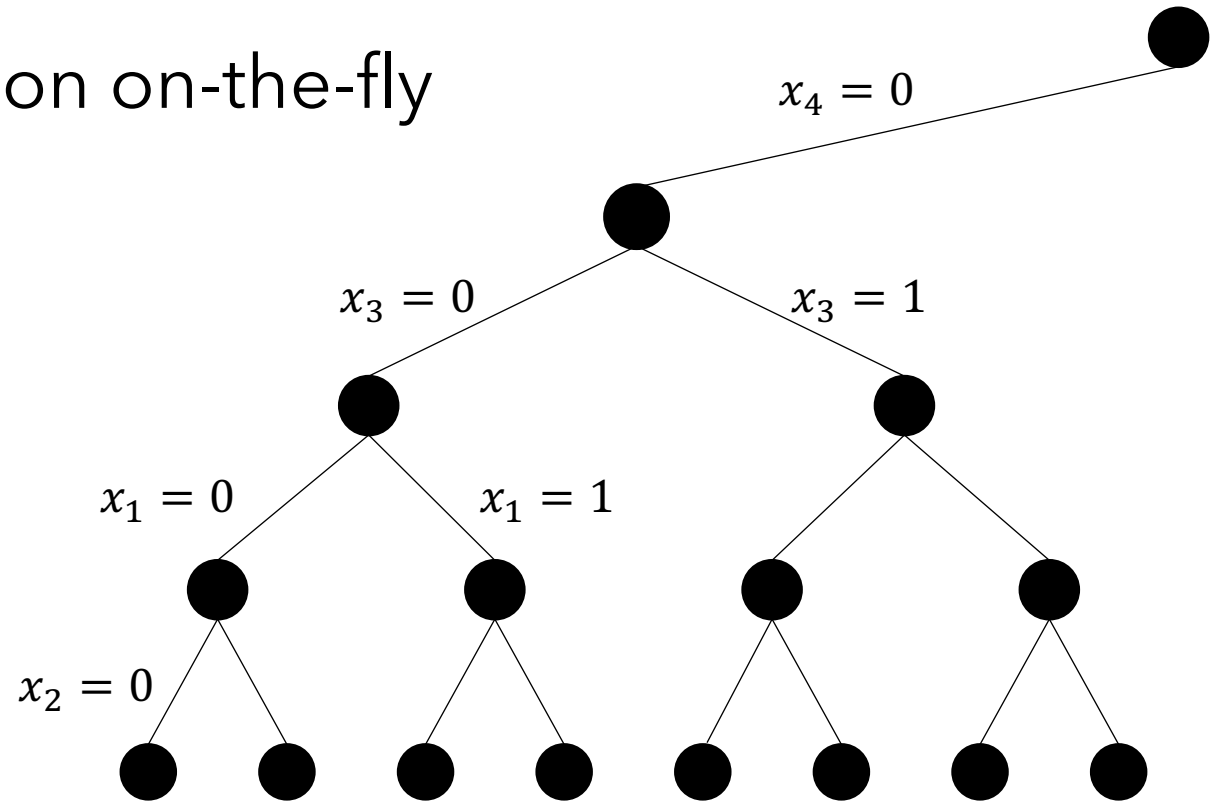
Variable selection policy



Better branching order than x_1, x_2, x_3, x_4 ? E.g., x_4, x_3, x_1, x_2

Variable selection policy

Chooses variables to branch on on-the-fly
Rather than pre-defined order



Variable selection policy

On Problem(j) with LP objective value $z(j)$:

- Let $z_i^+(j)$ be the LP objective value after setting $x_i = 1$
- Let $z_i^-(j)$ be the LP objective value after setting $x_i = 0$
- Branch on the variable x_i that maximizes

$$\max\{z(j) - z_i^+(j), z(j) - z_i^-(j)\}$$

Maximal change in objective value

Variable selection policy

On Problem(j) with LP objective value $z(j)$:

- Let $z_i^+(j)$ be the LP objective value after setting $x_i = 1$
- Let $z_i^-(j)$ be the LP objective value after setting $x_i = 0$
- Branch on the variable x_i that maximizes

$$\min\{z(j) - z_i^+(j), z(j) - z_i^-(j)\}$$

Minimal change in objective value

Variable selection policy

On Problem(j) with LP objective value $z(j)$:

- Let $z_i^+(j)$ be the LP objective value after setting $x_i = 1$
- Let $z_i^-(j)$ be the LP objective value after setting $x_i = 0$
- Branch on the variable x_i that maximizes

$$\mu \cdot \min\{z(j) - z_i^+(j), z(j) - z_i^-(j)\} + (1 - \mu) \cdot \max\{z(j) - z_i^+(j), z(j) - z_i^-(j)\}$$

For some IPs, it's better to choose μ closer to 0...

For others, it's better to choose μ closer to 1

Often, $\mu = \frac{5}{6}$ works well [Achterberg, '09]

Variable selection policy

On Problem(j) with LP objective value $z(j)$:

- Let $z_i^+(j)$ be the LP objective value after setting $x_i = 1$
- Let $z_i^-(j)$ be the LP objective value after setting $x_i = 0$
- Branch on the variable x_i that maximizes

$$\mu \cdot \min\{z(j) - z_i^+(j), z(j) - z_i^-(j)\} + (1 - \mu) \cdot \max\{z(j) - z_i^+(j), z(j) - z_i^-(j)\}$$

Challenge: Computing $z_i^-(j), z_i^+(j)$ requires solving a lot of LPs
Computing all of these LP relaxations referred to as “*strong-branching*”

Pseudo-cost branching: only use estimates of these LP values

Variable selection policy

On Problem(j) with LP objective value $z(j)$:

- Let $z_i^+(j)$ be the LP objective value after setting $x_i = 1$
- Let $z_i^-(j)$ be the LP objective value after setting $x_i = 0$
- Branch on the variable x_i that maximizes

$$\mu \cdot \min\{z(j) - z_i^+(j), z(j) - z_i^-(j)\} + (1 - \mu) \cdot \max\{z(j) - z_i^+(j), z(j) - z_i^-(j)\}$$

Many other possible variable selection policies!

See, e.g., [Achterberg, '09]

Major challenge of using B&B

How to choose the best $\{node, variable, \dots\}$ -selection policy?

Little theory about which to use when

This course:

Use machine learning to optimize B&B's performance

Outline

1. Linear programming
2. Integer programming
- 3. SAT solving** (SAT solvers also use tree search)
4. Next steps

SAT refresher

- $x_1, x_2, x_3, \dots \in \{0,1\}$
- \bar{x}_1 means **Not** x_1
 - If $x_1 = 1$ then $\bar{x}_1 = 0$; if $x_1 = 0$ then $\bar{x}_1 = 1$
- \vee means **Or**
 - $x_1 \vee x_2$ evaluates to **True** if $x_1 = 1$ **or** $x_2 = 1$
 - $x_1 \vee \bar{x}_2$ evaluates to **True** if $x_1 = 1$ **or** $x_2 = 0$
- \wedge means **And**
 - $x_1 \wedge x_2$ evaluates to **True** if $x_1 = 1$ **and** $x_2 = 1$
 - $x_1 \wedge \bar{x}_2$ evaluates to **True** if $x_1 = 1$ **and** $x_2 = 0$
- $(x_1 \vee x_2) \wedge (x_2 \vee \bar{x}_3)$ evaluates to **True** for $(x_1, x_2, x_3) = (1,0,0)$

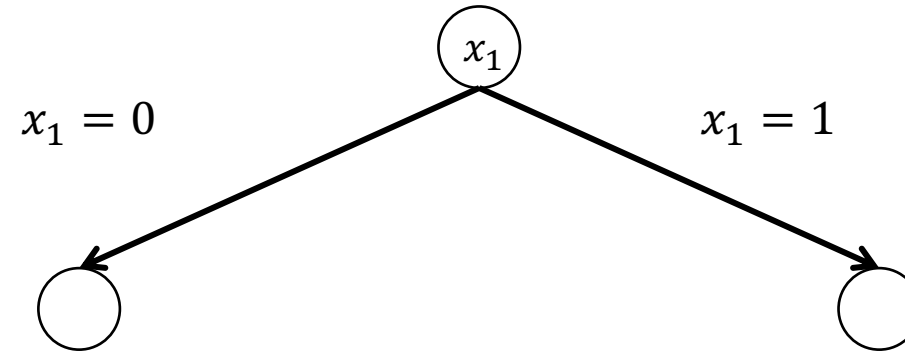
SAT refresher

$$\begin{aligned} & (x_1 \vee x_4) \\ \wedge & (x_1 \vee \bar{x}_3 \vee \bar{x}_8) \\ \wedge & (x_1 \vee x_8 \vee x_{12}) \\ \wedge & (x_2 \vee x_{11}) \\ \wedge & (\bar{x}_7 \vee \bar{x}_3 \vee x_9) \\ \wedge & (\bar{x}_7 \vee x_8 \vee \bar{x}_9) \\ \wedge & (x_7 \vee x_8 \vee \bar{x}_{10}) \\ \wedge & (x_7 \vee x_{10} \vee \bar{x}_{12}) \end{aligned}$$

SAT: Is there an assignment of $x_1, \dots, x_{12} \in \{0,1\}$ such that this formula evaluates to **True**?

SAT tree search

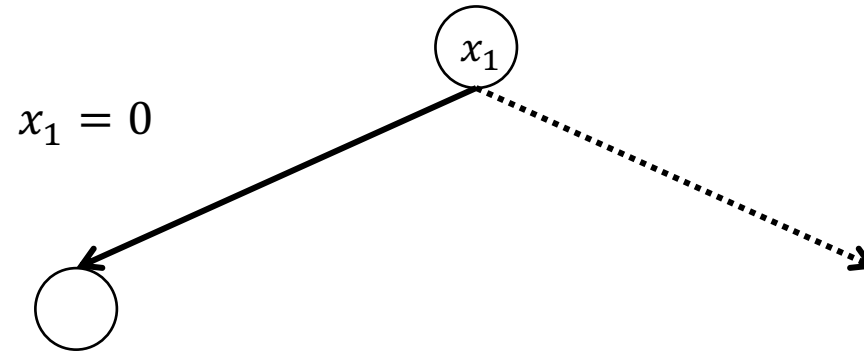
$(x_1 \vee x_4)$
 $\wedge (x_1 \vee \bar{x}_3 \vee \bar{x}_8)$
 $\wedge (x_1 \vee x_8 \vee x_{12})$
 $\wedge (x_2 \vee x_{11})$
 $\wedge (\bar{x}_7 \vee \bar{x}_3 \vee x_9)$
 $\wedge (\bar{x}_7 \vee x_8 \vee \bar{x}_9)$
 $\wedge (x_7 \vee x_8 \vee \bar{x}_{10})$
 $\wedge (x_7 \vee x_{10} \vee \bar{x}_{12})$



How to prune if there's
no objective function?

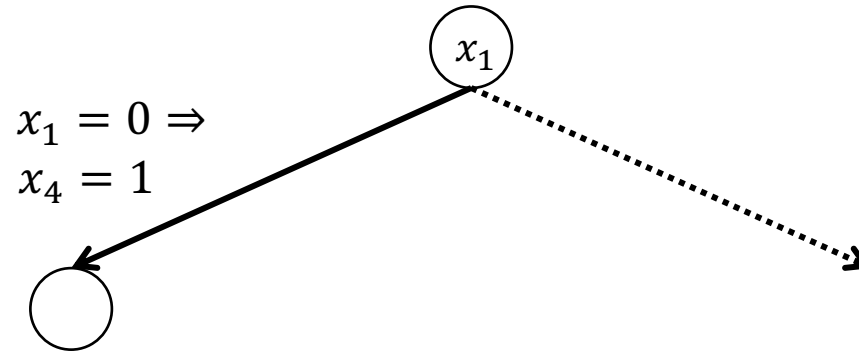
SAT tree search

$$\begin{aligned} & (x_1 \vee x_4) \\ \wedge & (x_1 \vee \bar{x}_3 \vee \bar{x}_8) \\ \wedge & (x_1 \vee x_8 \vee x_{12}) \\ \wedge & (x_2 \vee x_{11}) \\ \wedge & (\bar{x}_7 \vee \bar{x}_3 \vee x_9) \\ \wedge & (\bar{x}_7 \vee x_8 \vee \bar{x}_9) \\ \wedge & (x_7 \vee x_8 \vee \bar{x}_{10}) \\ \wedge & (x_7 \vee x_{10} \vee \bar{x}_{12}) \end{aligned}$$



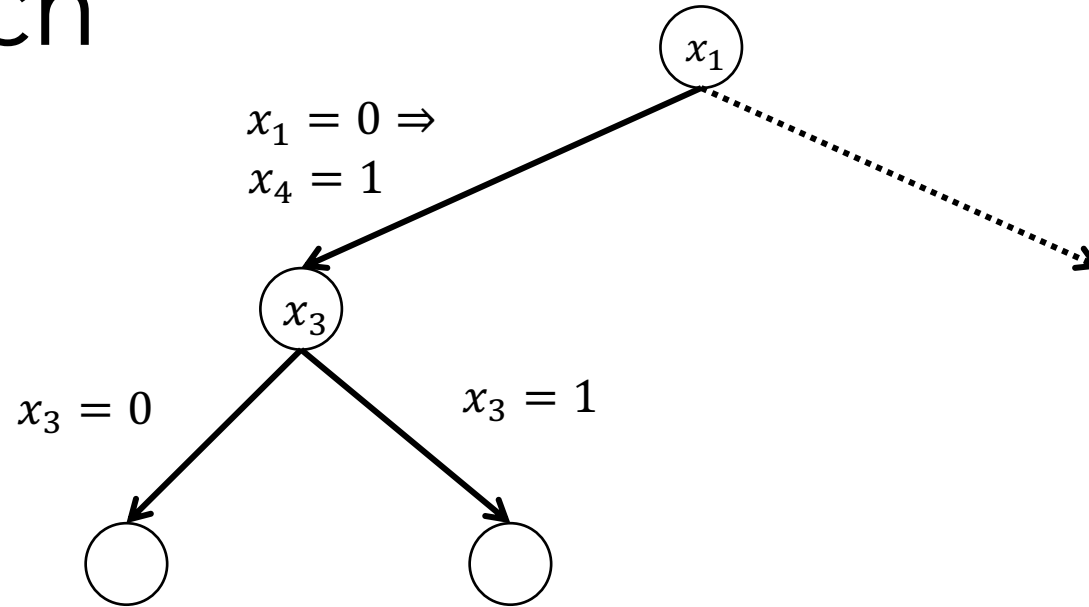
SAT tree search

$$\begin{aligned} & (x_1 \vee x_4) \\ \wedge & (x_1 \vee \bar{x}_3 \vee \bar{x}_8) \\ \wedge & (x_1 \vee x_8 \vee x_{12}) \\ \wedge & (x_2 \vee x_{11}) \\ \wedge & (\bar{x}_7 \vee \bar{x}_3 \vee x_9) \\ \wedge & (\bar{x}_7 \vee x_8 \vee \bar{x}_9) \\ \wedge & (x_7 \vee x_8 \vee \bar{x}_{10}) \\ \wedge & (x_7 \vee x_{10} \vee \bar{x}_{12}) \end{aligned}$$



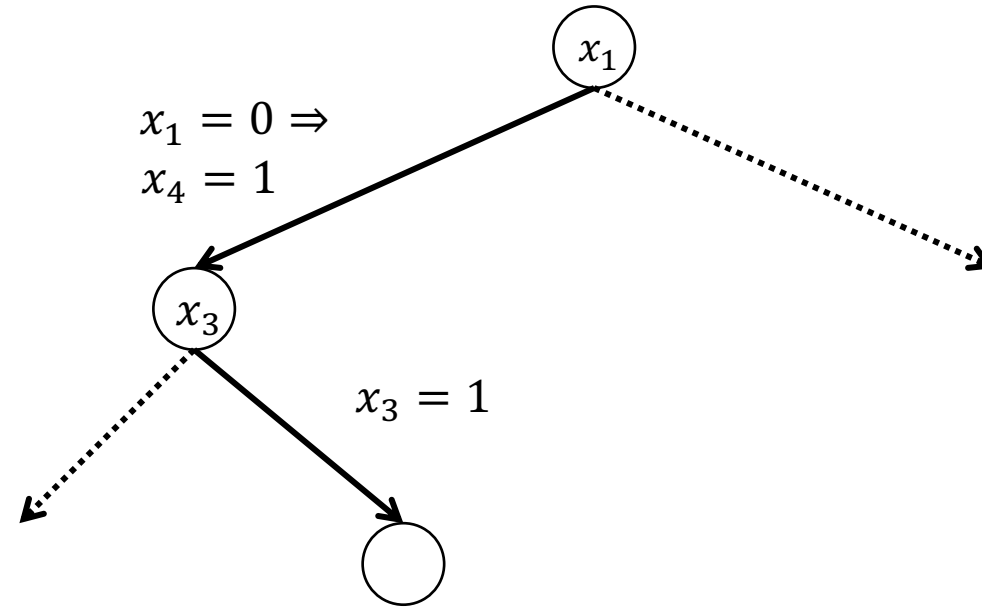
SAT tree search

$(x_1 \vee x_4)$
 $\wedge (x_1 \vee \bar{x}_3 \vee \bar{x}_8)$
 $\wedge (x_1 \vee x_8 \vee x_{12})$
 $\wedge (x_2 \vee x_{11})$
 $\wedge (\bar{x}_7 \vee \bar{x}_3 \vee x_9)$
 $\wedge (\bar{x}_7 \vee x_8 \vee \bar{x}_9)$
 $\wedge (x_7 \vee x_8 \vee \bar{x}_{10})$
 $\wedge (x_7 \vee x_{10} \vee \bar{x}_{12})$



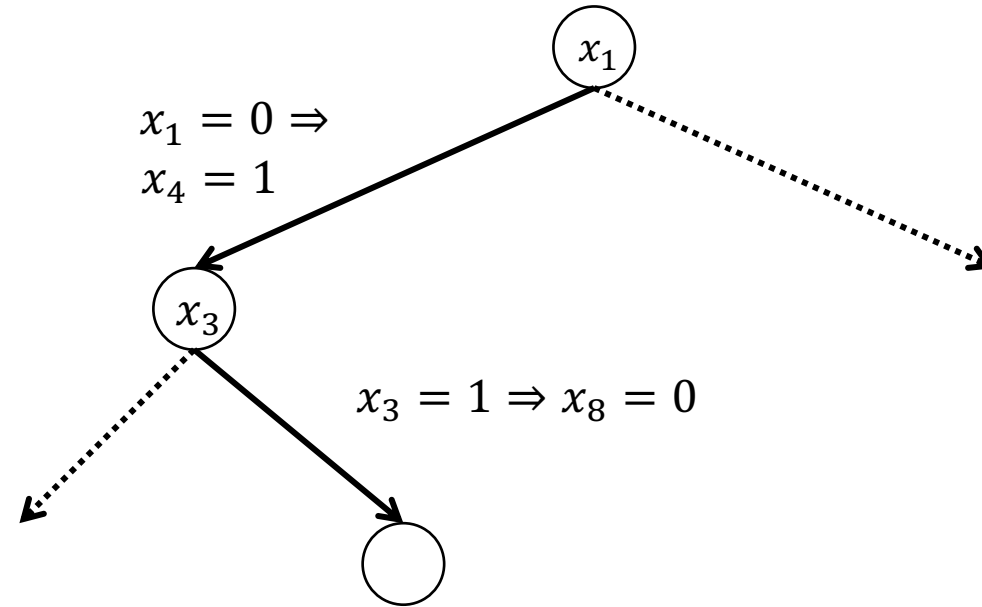
SAT tree search

$$\begin{aligned} & (x_1 \vee x_4) \\ \wedge & (x_1 \vee \bar{x}_3 \vee \bar{x}_8) \\ \wedge & (x_1 \vee x_8 \vee x_{12}) \\ \wedge & (x_2 \vee x_{11}) \\ \wedge & (\bar{x}_7 \vee \bar{x}_3 \vee x_9) \\ \wedge & (\bar{x}_7 \vee x_8 \vee \bar{x}_9) \\ \wedge & (x_7 \vee x_8 \vee \bar{x}_{10}) \\ \wedge & (x_7 \vee x_{10} \vee \bar{x}_{12}) \end{aligned}$$



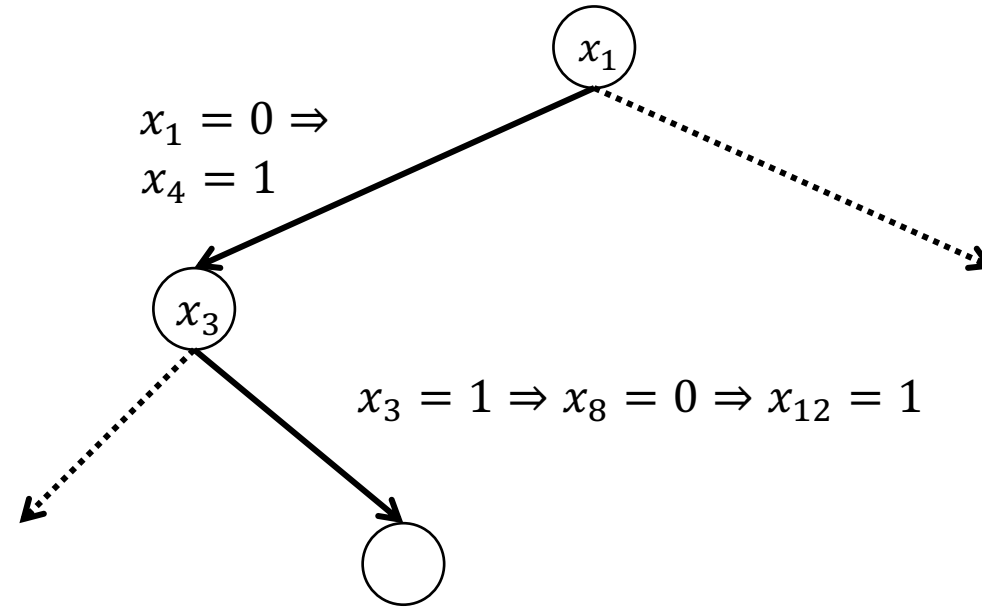
SAT tree search

$$\begin{aligned} & (x_1 \vee x_4) \\ \wedge & (x_1 \vee \bar{x}_3 \vee \bar{x}_8) \\ \wedge & (x_1 \vee x_8 \vee x_{12}) \\ \wedge & (x_2 \vee x_{11}) \\ \wedge & (\bar{x}_7 \vee \bar{x}_3 \vee x_9) \\ \wedge & (\bar{x}_7 \vee x_8 \vee \bar{x}_9) \\ \wedge & (x_7 \vee x_8 \vee \bar{x}_{10}) \\ \wedge & (x_7 \vee x_{10} \vee \bar{x}_{12}) \end{aligned}$$



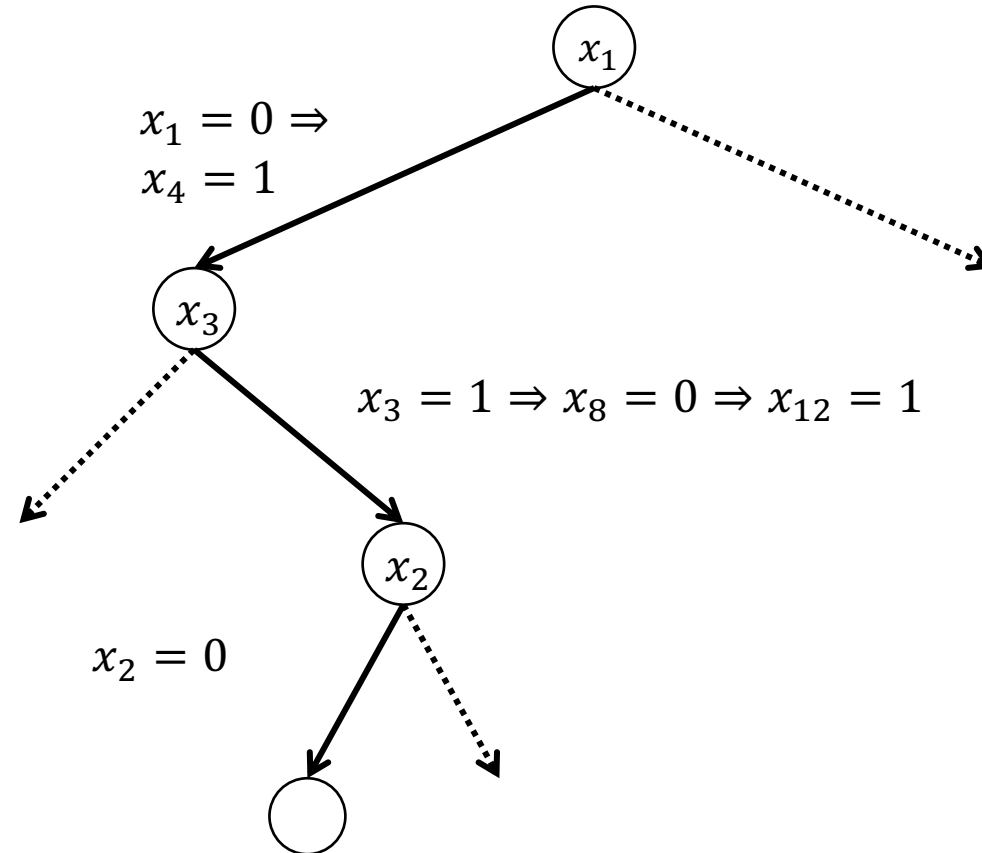
SAT tree search

$(x_1 \vee x_4)$
 $\wedge (x_1 \vee \bar{x}_3 \vee \bar{x}_8)$
 $\wedge (x_1 \vee x_8 \vee x_{12})$
 $\wedge (x_2 \vee x_{11})$
 $\wedge (\bar{x}_7 \vee \bar{x}_3 \vee x_9)$
 $\wedge (\bar{x}_7 \vee x_8 \vee \bar{x}_9)$
 $\wedge (x_7 \vee x_8 \vee \bar{x}_{10})$
 $\wedge (x_7 \vee x_{10} \vee \bar{x}_{12})$



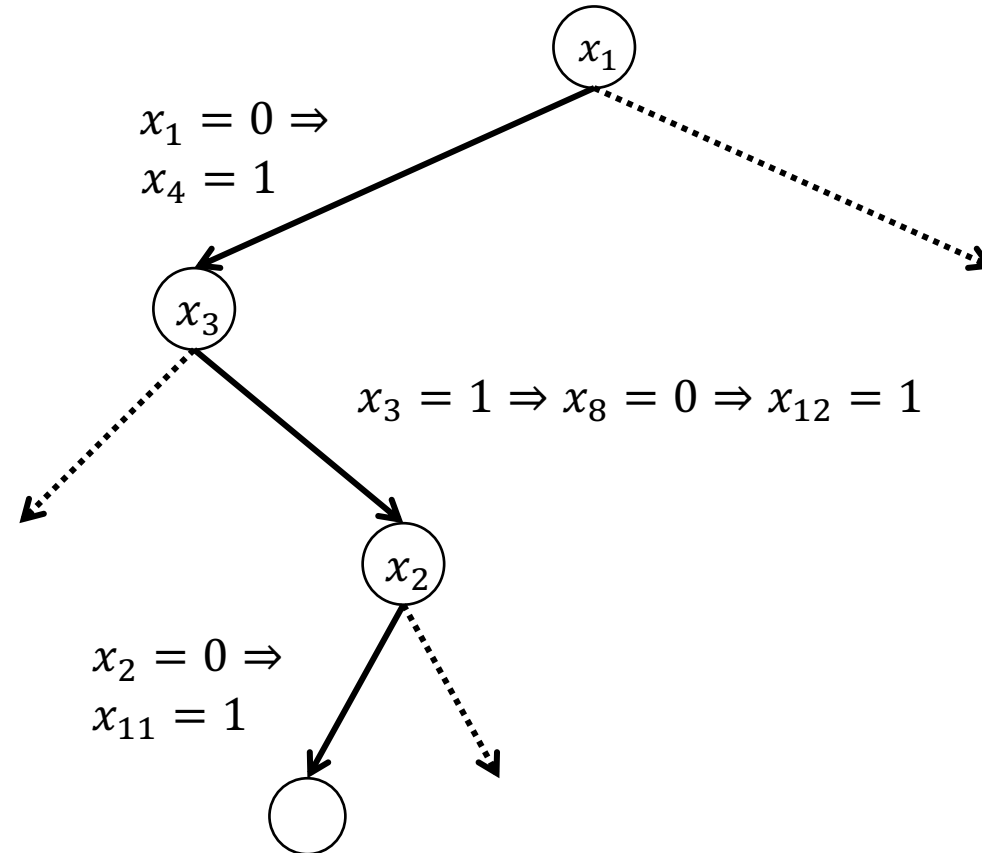
SAT tree search

$(x_1 \vee x_4)$
 $\wedge (x_1 \vee \bar{x}_3 \vee \bar{x}_8)$
 $\wedge (x_1 \vee x_8 \vee x_{12})$
 $\wedge (x_2 \vee x_{11})$
 $\wedge (\bar{x}_7 \vee \bar{x}_3 \vee x_9)$
 $\wedge (\bar{x}_7 \vee x_8 \vee \bar{x}_9)$
 $\wedge (x_7 \vee x_8 \vee \bar{x}_{10})$
 $\wedge (x_7 \vee x_{10} \vee \bar{x}_{12})$



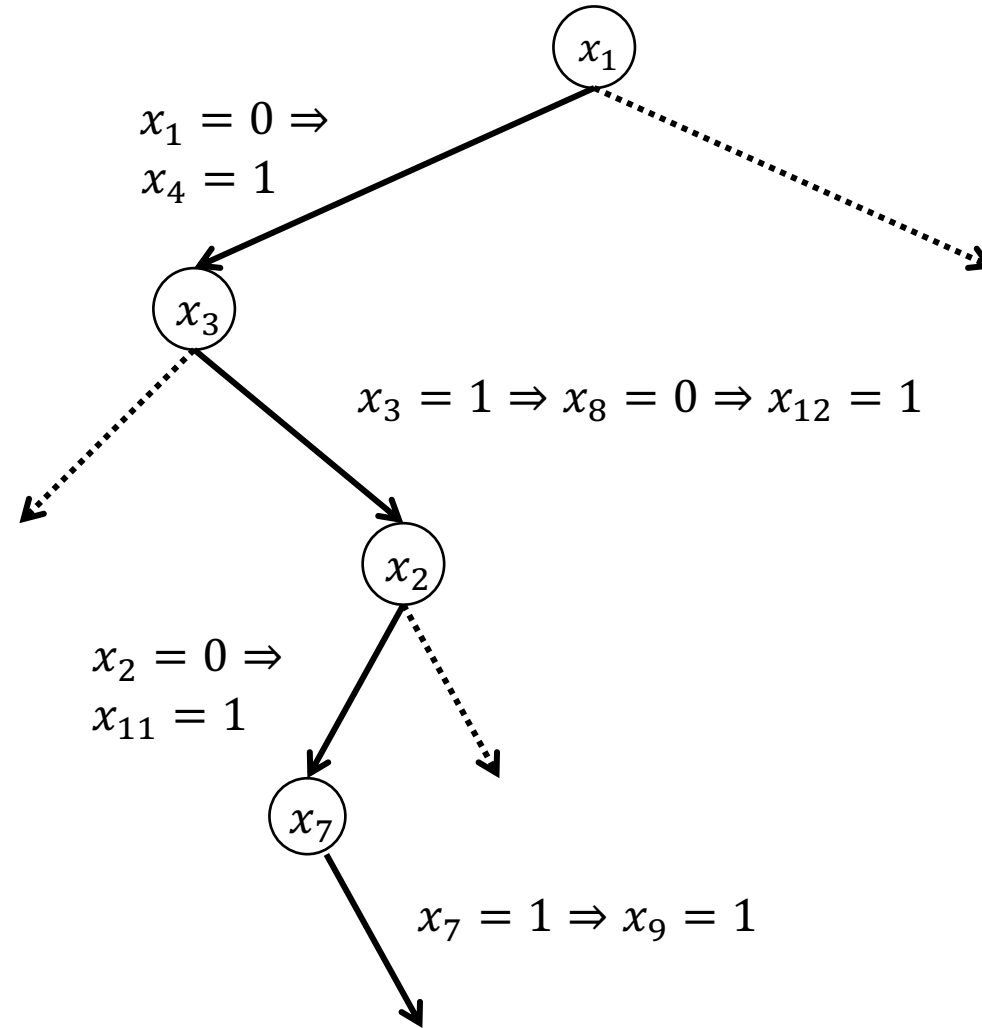
SAT tree search

$$\begin{aligned} & (x_1 \vee x_4) \\ \wedge & (x_1 \vee \bar{x}_3 \vee \bar{x}_8) \\ \wedge & (x_1 \vee x_8 \vee x_{12}) \\ \wedge & (x_2 \vee x_{11}) \\ \wedge & (\bar{x}_7 \vee \bar{x}_3 \vee x_9) \\ \wedge & (\bar{x}_7 \vee x_8 \vee \bar{x}_9) \\ \wedge & (x_7 \vee x_8 \vee \bar{x}_{10}) \\ \wedge & (x_7 \vee x_{10} \vee \bar{x}_{12}) \end{aligned}$$




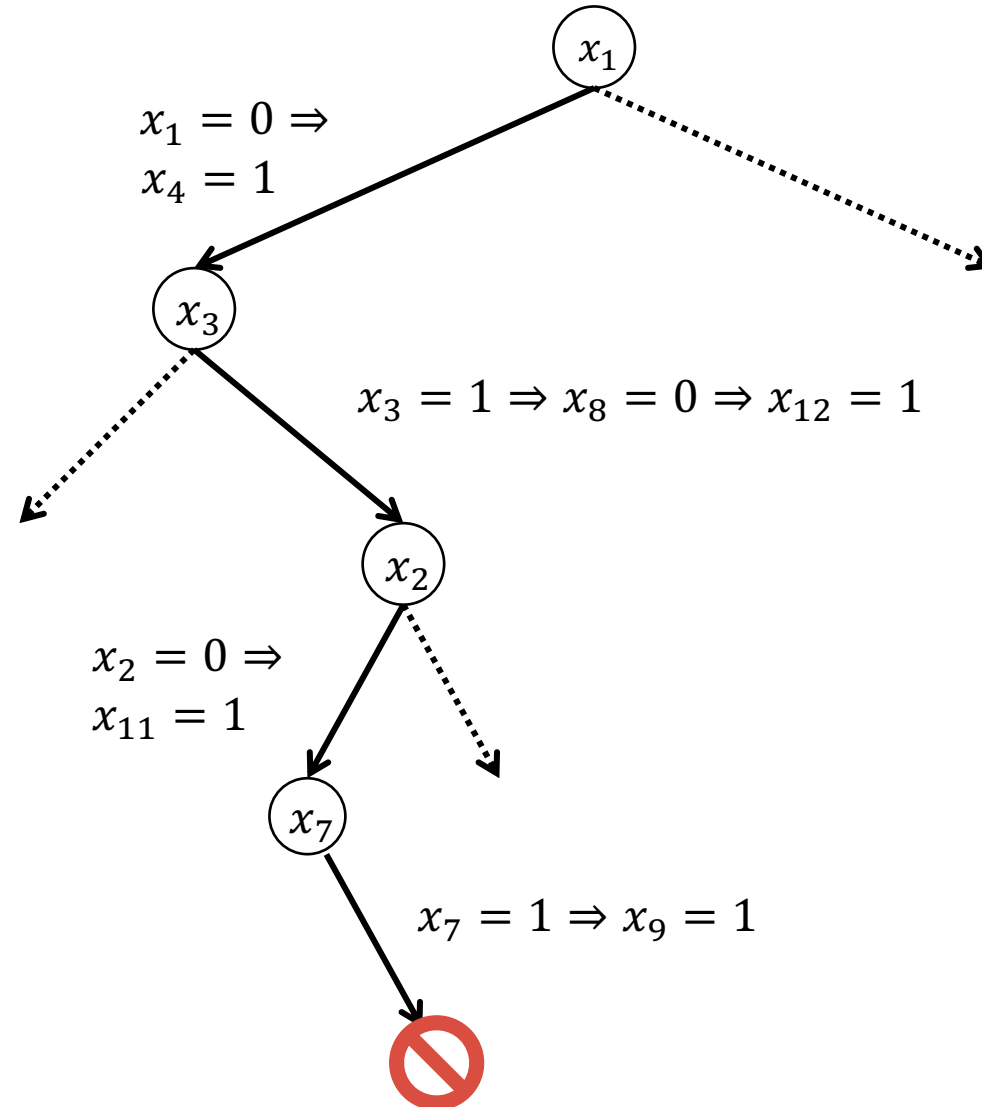
SAT tree search

$$\begin{aligned} & (x_1 \vee x_4) \\ \wedge & (x_1 \vee \bar{x}_3 \vee \bar{x}_8) \\ \wedge & (x_1 \vee x_8 \vee x_{12}) \\ \wedge & (x_2 \vee x_{11}) \\ \wedge & (\bar{x}_7 \vee \bar{x}_3 \vee x_9) \\ \wedge & (\bar{x}_7 \vee x_8 \vee \bar{x}_9) \\ \wedge & (x_7 \vee x_8 \vee \bar{x}_{10}) \\ \wedge & (x_7 \vee x_{10} \vee \bar{x}_{12}) \end{aligned}$$



SAT tree search

$(x_1 \vee x_4)$
 $\wedge (x_1 \vee \bar{x}_3 \vee \bar{x}_8)$
 $\wedge (x_1 \vee x_8 \vee x_{12})$
 $\wedge (x_2 \vee x_{11})$
 $\wedge (\bar{x}_7 \vee \bar{x}_3 \vee x_9)$
 $\wedge (\bar{x}_7 \vee x_8 \vee \bar{x}_9)$ 
 $\wedge (x_7 \vee x_8 \vee \bar{x}_{10})$
 $\wedge (x_7 \vee x_{10} \vee \bar{x}_{12})$



SAT tree search

Many similar design choices as IP tree search

Variable selection: branch on variable that leads to the **largest number of deductions** on other variables

Outline

1. Linear programming
2. Integer programming
3. SAT solving
- 4. Next steps**

Next steps

- **Today:** Saw many ways solvers can be optimized & configured
- With a deft configuration:
Can quickly solve extremely challenging real-world problems



Routing



Manufacturing



Scheduling



Planning



Finance

- **Future classes:** How to use ML to optimize these solvers

Plan for the next few classes

This Thursday: GNNs overview

Tu 4/18, Th 4/20: GNN paper discussions

Tu 4/25, Th 4/27: IP/SAT paper discussions