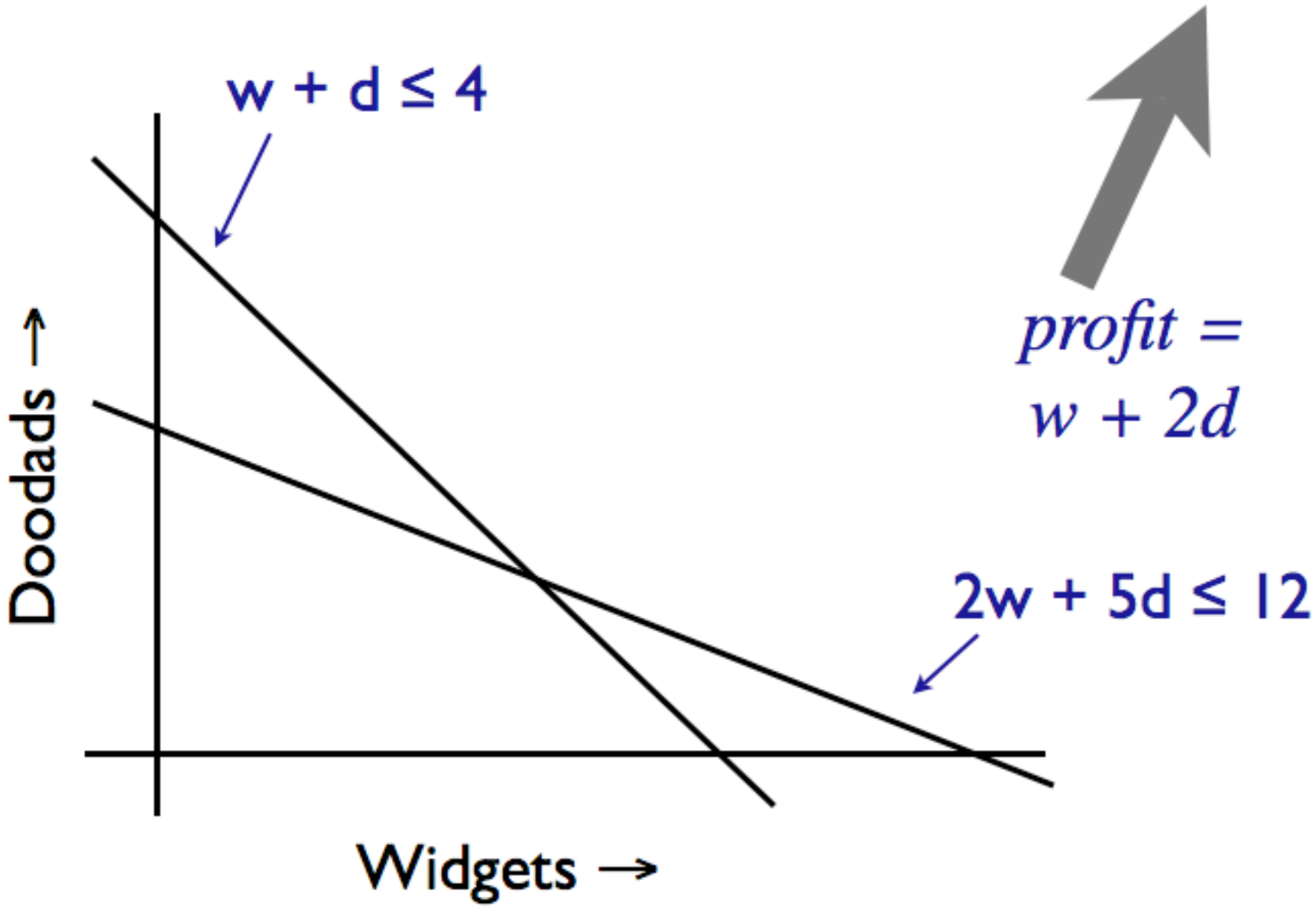
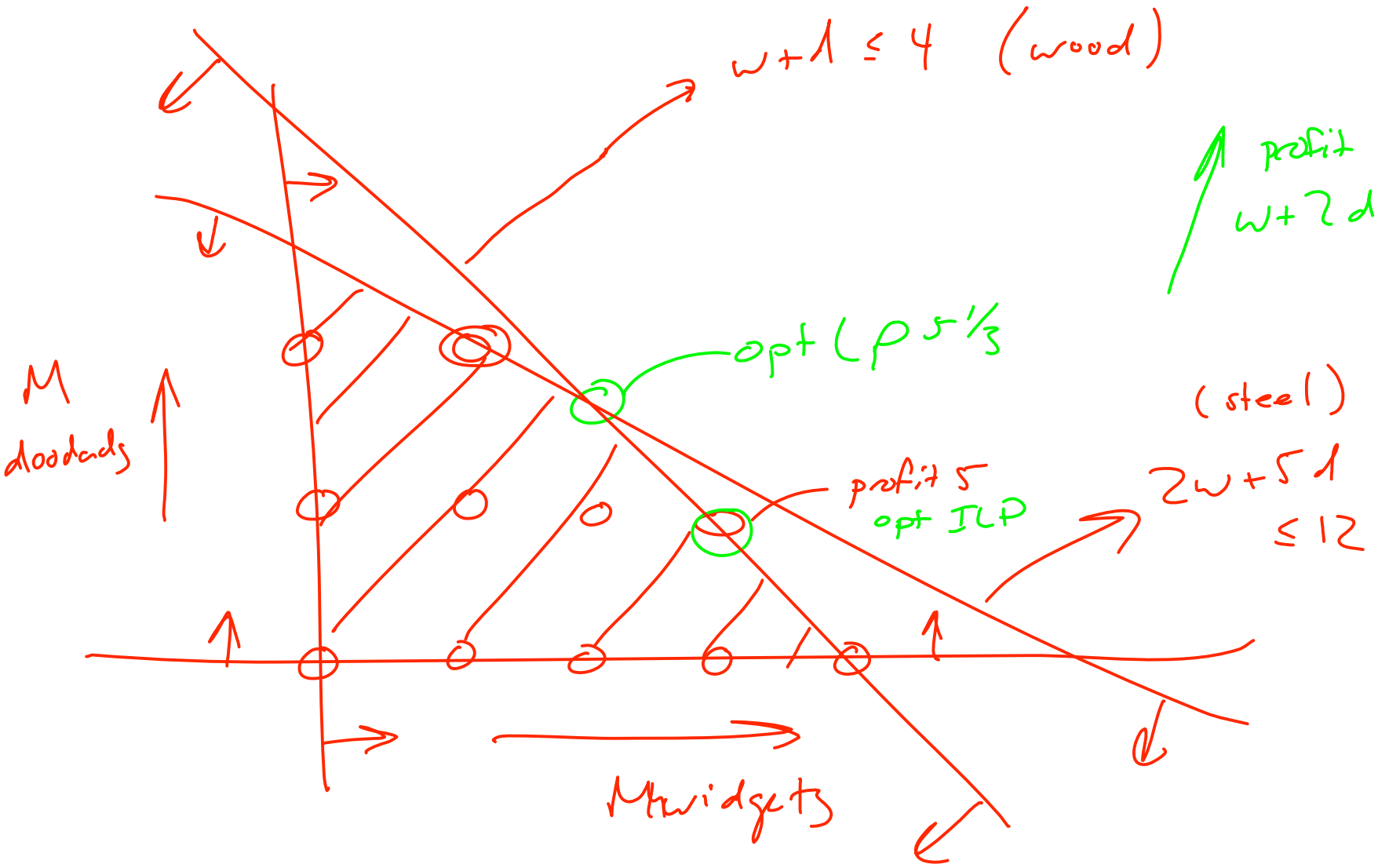
Production planning

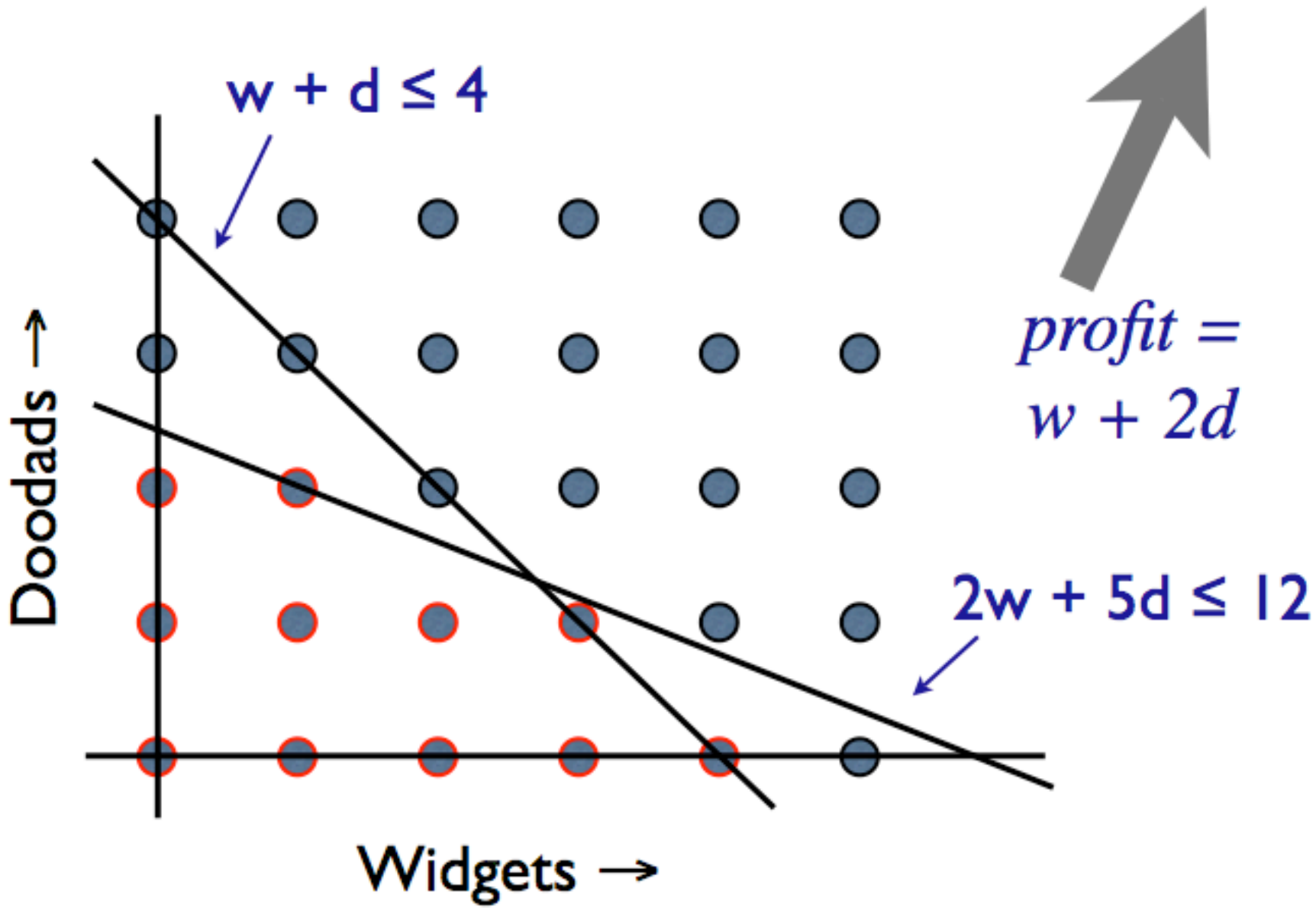
- Big factory, produces widgets, doodads
- Each widget:
  - 1 unit of wood, 2 units steel, profit $1
- Each doodad
  - 1 unit wood, 5 units steel, profit $2
- Have: 4M units wood, 12M units steel

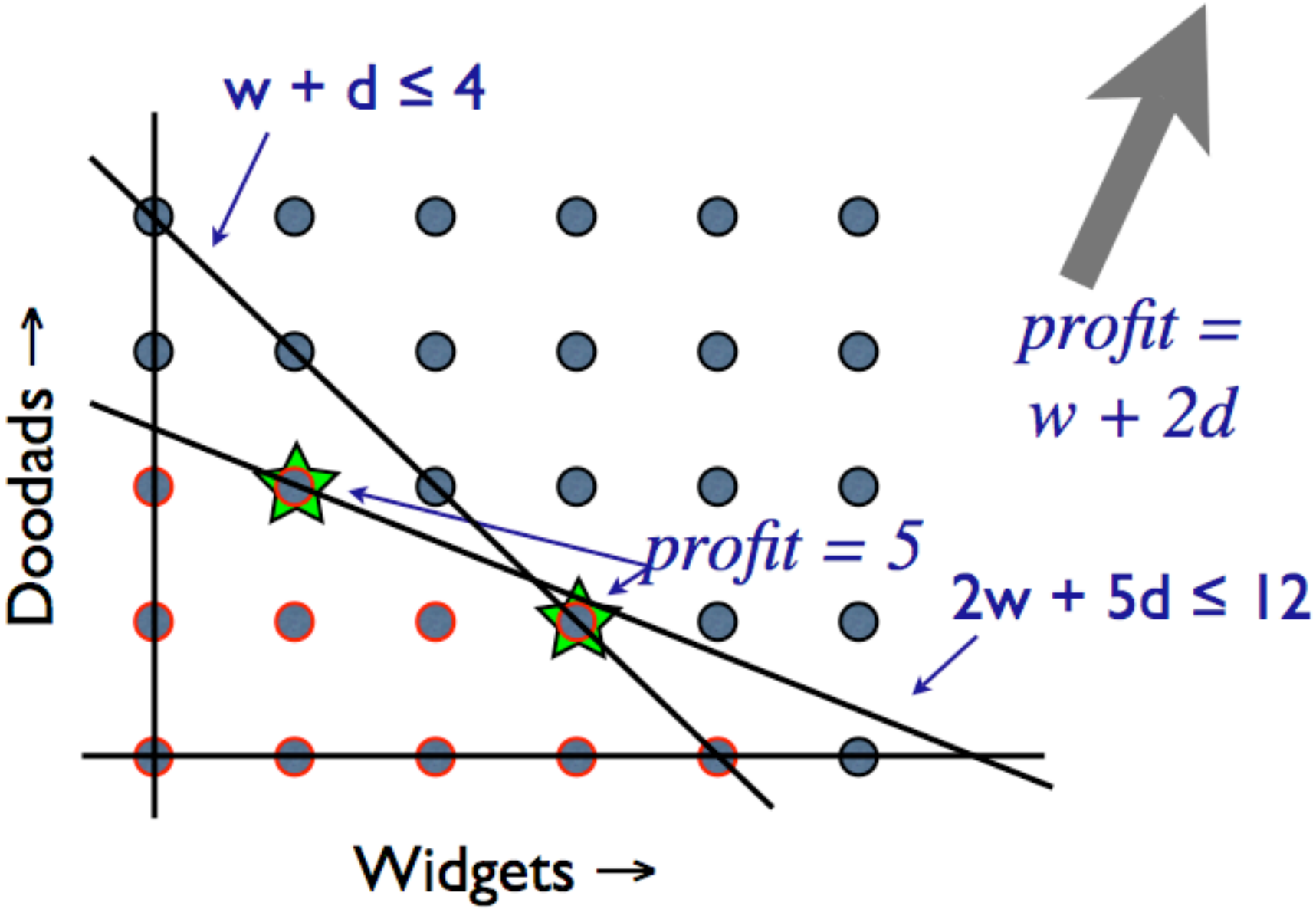

w + d ≤ 4
profit =
w + 2d
2w + 5d ≤ 12
Doodads →
Widgets →


w + 1 ≤ 4 (wood)
profit
w + 2 d
opt LP 5 1/3
M
doodads
(steel)
profit 5
opt ILP
2w + 5 d
≤ 12
Mwidgets

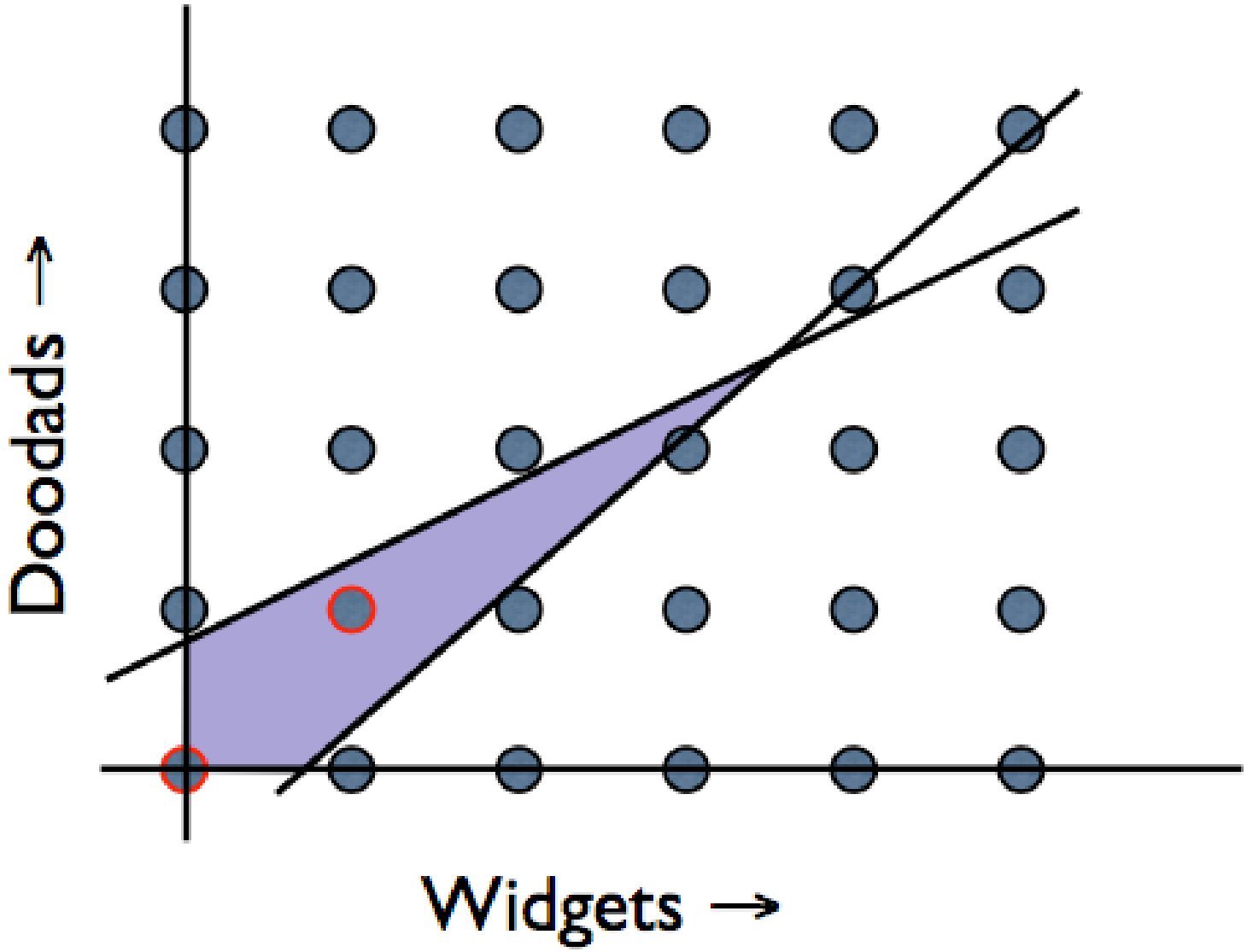
A wrinkle

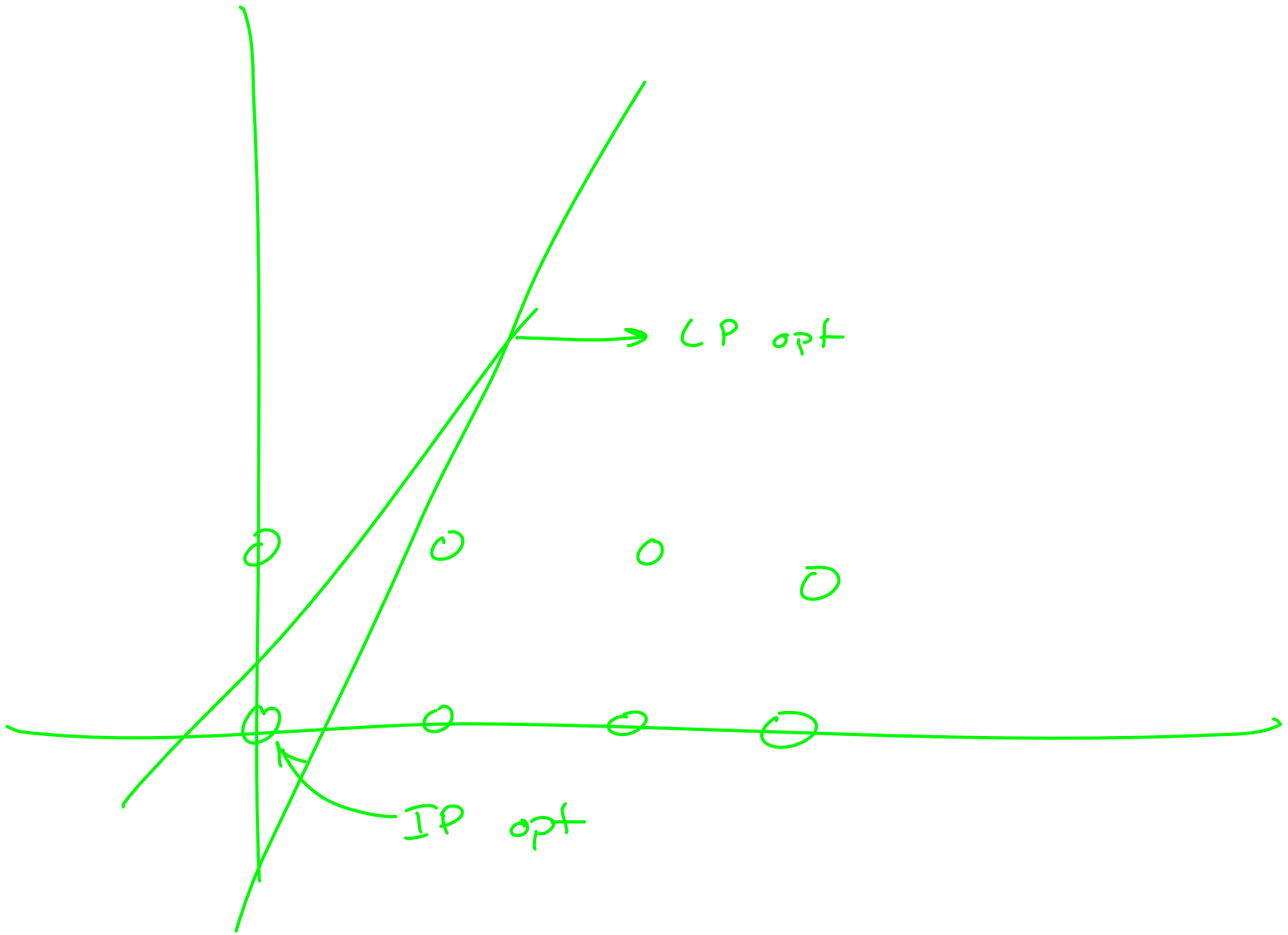
- Due to limitations of machinery, must produce in lots of exactly 1M widgets or 1M doodads


w + d ≤ 4
profit =
w + 2d
2w + 5d ≤ 12
Doodads →
Widgets →


w + d ≤ 4
profit =
w + 2d
Doodads →
profit = 5
2w + 5d ≤ 12
Widgets →

Effect of integrality




LP opt
IP opt

Integer linear programs

- max $c^T x$ s.t.
  $Ax + b \geq 0$
  $Cx + d = 0$
  x integer

E.g., max $w + d \cdot 2$ s.t.

$w + d \leq 4$

$2w + 5d \leq 12$

$w, d \geq 0$

w, d integers

Variations

- Mixed integer linear program (MILP)

  Some vars $\mathbb{R}$, others $\mathbb{Z}$

- Integer quadratic program

  quadratic objective

- 0-1 ILP vars are binary $\{0,1\}$

Combinatorial optimization

- Roughly, any optimization problem whose decision version is in NP
- ILP, MILP, 0/1 IP: complete for this class

Applications

- SAT $(a \vee b \vee c) \wedge (\bar{b} \vee c \vee \bar{d}) \wedge \dots$

  $a + b + c \geq 1$

  $(1-b) + c + (1-d) \geq 1$

  $\dots$

  ILF

  $a, b, c, d, \dots \in \{0, 1\}$

  feasibility problem

- MAXSAT

  $a + b + c \geq 1 - s_1$

  $(1-b) + c + (1-d) \geq 1 - s_2$

  $\dots$

  (M) ILP

  $\min \sum_i w_i \cdot s_i$ ← weight for clause i

  $a, b, c, d, \dots, s_1, s_2, \dots \in \{0, 1\}$

Applications

- Facility location problem:
  - have n stores, at positions $y_1, y_2, \ldots$ $\in \mathbb{R}^d$ const
  - can build m warehouses, at $x_1, x_2, \ldots$ $\in \mathbb{R}^d$ vars
  - minimize distance from each store to nearest warehouse
- Extra vars: $z_{ij} \in \{0,1\}$ store i uses warehouse j

$$d_{ij} \begin{cases} 0 & \text{if } z_{ij} = 0 \\ \|x_i - y_j\| & \text{o/w} \end{cases}$$

- min $\sum_{ij} d_{ij}$ s.t.

$$\|x_i - y_j\| \le d_{ij} + M(1 - z_{ij})$$

big M const

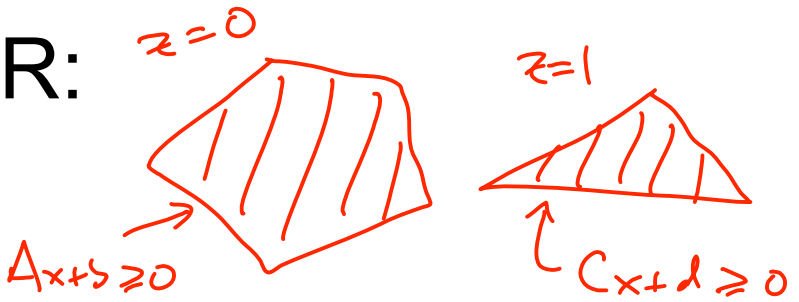
$$d_{ij} \ge 0$$
$$z_{ij} \in \{0,1\}$$
$$\forall i: \sum_j z_{ij} \ge 1$$

MISOCP

Embedding logic in MILPs

- $z \in \{0,1\} \implies a^T x + b \geq 0 \longrightarrow a^T x + b + M(1-z) \geq 0$

- OR:



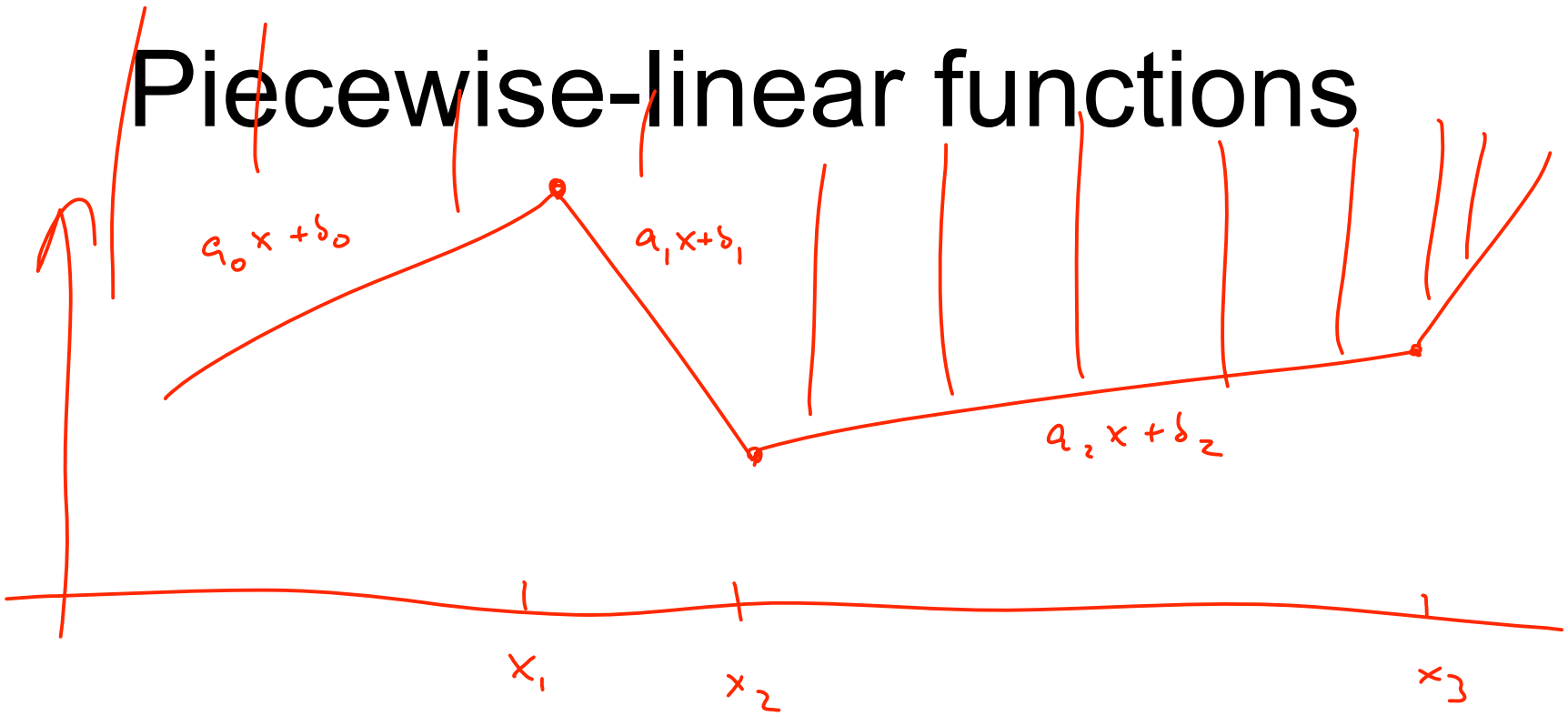
big const
all 1s vector

$$Ax + b + Mz\vec{1} \geq 0$$
$$Cx + d + M(1-z)\vec{1} \geq 0$$

- k-of-n:

$$A_i x + b_i + M(1-z_i)\vec{1} \geq 0 \quad i \in 1 \ldots n$$
$$\sum_i z_i \geq k$$

Piecewise-linear functions



$z_i \in \{0,1\}$ indicate whether in segment i

$\rightarrow \quad z_i \Rightarrow x_i \leq x \leq x_{i+1} \quad \rightarrow$ def $x_0 = -\infty$, $x_{k+1} = +\infty$

$y \geq a_i x + b_i - M(1 - z_i)$

Applications

- Planning (e.g., STRIPS)

initial state: have(cake), ¬eaten(cake)

goal predicates: have(cake) ∧ eaten(cake)

eat(cake). pre: have(cake)
eff: ¬have(cake), eaten(cake)

bake(cake): pre: ¬have(cake)
eff: have(cake)

Have cake & eat it as CNF

$$(\overline{h}_1 \vee \overline{M}_1) \wedge (h_1 \vee M_1) \wedge (\overline{e}_1 \vee M_1) \wedge (e_1 \vee \overline{M}_1)$$
$$\wedge (\overline{M}_1 \vee \overline{h}_1) \wedge (\overline{M}_1 \vee e_1) \wedge (\overline{M}_2 \vee h_1) \wedge (\overline{M}_2 \vee \overline{h}_2)$$
$$\wedge (\overline{M}_2 \vee e_2) \wedge (\overline{B}_2 \vee \overline{h}_1) \wedge (\overline{B}_2 \vee h_2) \wedge (\overline{h}_2 \vee h_1 \vee B_2)$$
$$\wedge (\overline{h}_2 \vee \overline{M}_2 \vee B_2) \wedge (h_2 \vee \overline{h}_1 \vee M_2)$$
$$\wedge (h_2 \vee \overline{B}_2 \vee M_2) \wedge (\overline{e}_2 \vee e_1 \vee M_2)$$
$$\wedge (e_2 \vee \overline{e}_1) \wedge (e_2 \vee \overline{M}_2) \wedge (\overline{M}_2 \vee \overline{B}_2) \wedge (h_2) \wedge (e_2)$$

h_i : have cake @ i

e_i : eaten @ i

B_i : bake @ i

M_i : "munch" @ i

$M_2 \Rightarrow h_1$

$M_1 \Rightarrow \neg h_2$

$h_2 \Rightarrow (h_1 \wedge \overline{M}_2) \vee B_2$

Applications

- Scheduling (e.g. airline crews)

crews $(1 \dots m)$ flights $(1 \dots n)$ cities $(1 \dots k)$

s_i, t_i : start, stop times for crew i $\quad t_i \le s_i + 10\text{hrs}$

$z_{ij} \in \{0,1\}$: does crew i serve flight j

$q_{i,j_1,j_2} \in \{0,1\}$: does crew i stay @ dest(j_1) until j_2 leaves

$z_{ij} \Rightarrow s_i \le \text{depart}(j) - 1\text{ hr}$

$z_{ij_0} \Rightarrow \text{home}(i) = \text{origin}(j_1) \vee \exists j_2 : z_{i,j_2} \wedge q_{i,j_2,j_1}$

Solving combinatorial optimization problems

- Two basic strategies

  Search

  approximate

- And, combinations

Basic search

- Schema: if n 0/1 variables, $\{0, 1, *\}^n$
- E.g., 1 0 * * 1

- Schema is **full** if no *s: e.g., 1 0 1 0 1
- Notation: schema/(variable→value)
- E.g., 10**1/(x_3→1) = 1 0 1 * 1

Basic search

[schema, value] = search(F, sch)

- If full(sch): return [sch, F(sch)]
- pick a variable x_i
- $[sch^{(0)}, v^{(0)}] = search(F, sch/(x_i \rightarrow 0))$
- $[sch^{(1)}, v^{(1)}] = search(F, sch/(x_i \rightarrow 1))$
- if $v^{(0)} \leq v^{(1)}$ return $[sch^{(0)}, v^{(0)}]$

  else return $[sch^{(1)}, v^{(1)}]$

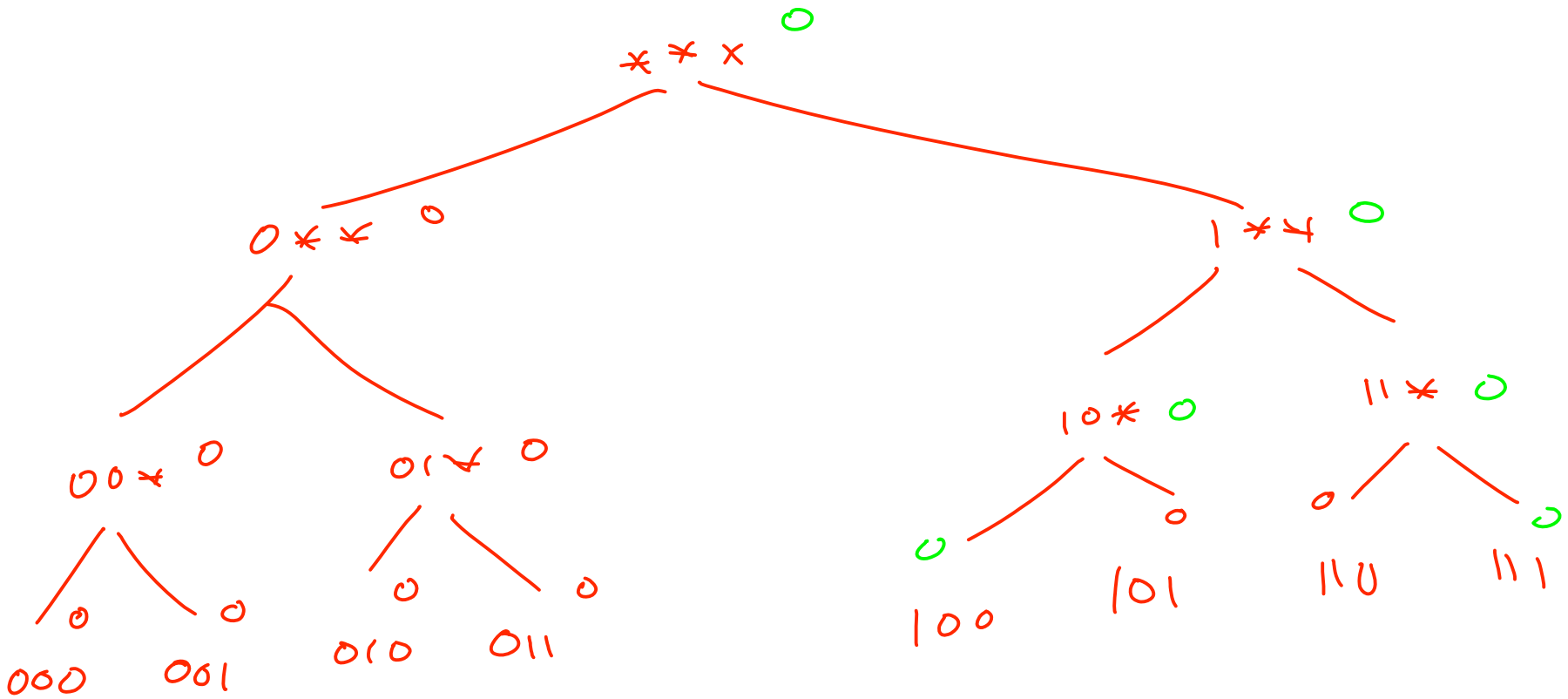
Exercise

- SAT problem w/ XOR constraints:

  search(a ⊕ b ^ b ⊕ c ^ c ⊕ a, ***)

- How many total calls to search()?

search(a ⊕ b ^ b ⊕ c ^ c ⊕ a)



Constraint propagation

$a \oplus b \wedge b \oplus c \wedge c \oplus a$

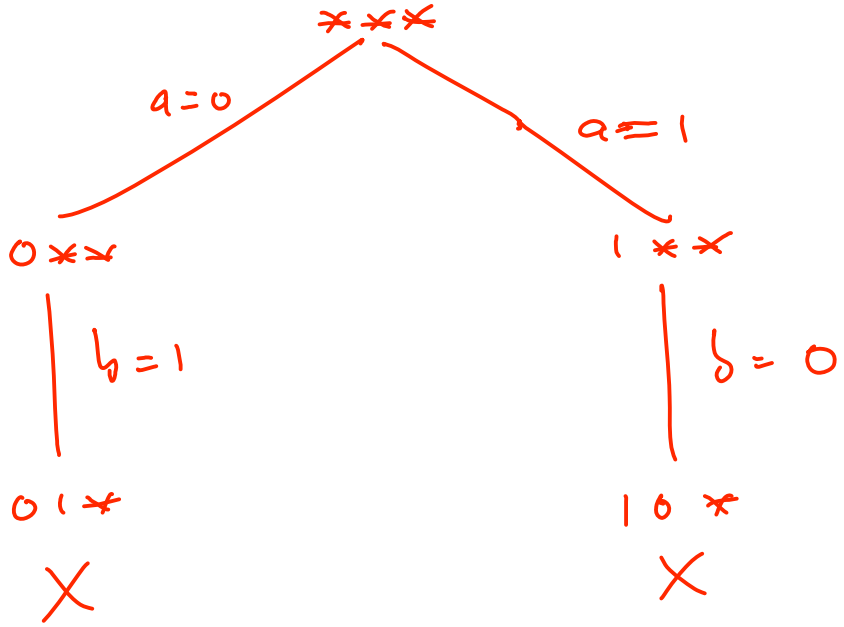
- Start w/ table of feasible values

a: 0 1 b: ~~0~~ 1 c: ~~0 1~~

- When we set a var, look at its constrs
  - e.g., set a = 0: $a \oplus b$ $c \oplus a$
  - remove infeasible values from domains of neighbors
- If a domain becomes empty: $\Rightarrow$ backtrack
- If one becomes singleton: $\Rightarrow$ "branch" on that var next

e.g., $b = 1 \Rightarrow c \neq 1$

Search tree



In SAT
"unit resolution"

In ILP
more vals in each table but single constr can remove lots

Relaxation

- min f(x) s.t. $x \in S_1$ (*)
- min g(y) s.t. $y \in S_2$ (**)
- (*) is a **relaxation** of (**) if:
  - $S_1 \subseteq S_2$
  - $f(x) = g(x) \quad \forall x \in S_1$
- Example: widget LP relaxes widget ILP

Why are relaxations useful?

- Relaxation may be easier to solve (e.g. convex)
- From solution of relaxed problem: can sometimes get "good" solution to original problem called "rounding"
- Suppose x*, y* are optimal solutions to original / relaxed problems
  - we know: $g(y^*) \leq f(x^*)$

Example: have & eat LP

$$\ldots \wedge (h_2 \vee \overline{B}_2 \vee M_2) \wedge (\overline{e}_2 \vee e_1 \vee M_2)$$
$$\wedge (e_2 \vee \overline{e}_1) \wedge (e_2 \vee \overline{M}_2) \wedge (\overline{M}_2 \vee \overline{B}_2) \wedge (h_2) \wedge (e_2)$$

min $5s_1 + 5s_2 + \ldots + 5s_{19} + s_{20} + 2s_{21}$ s.t.

…

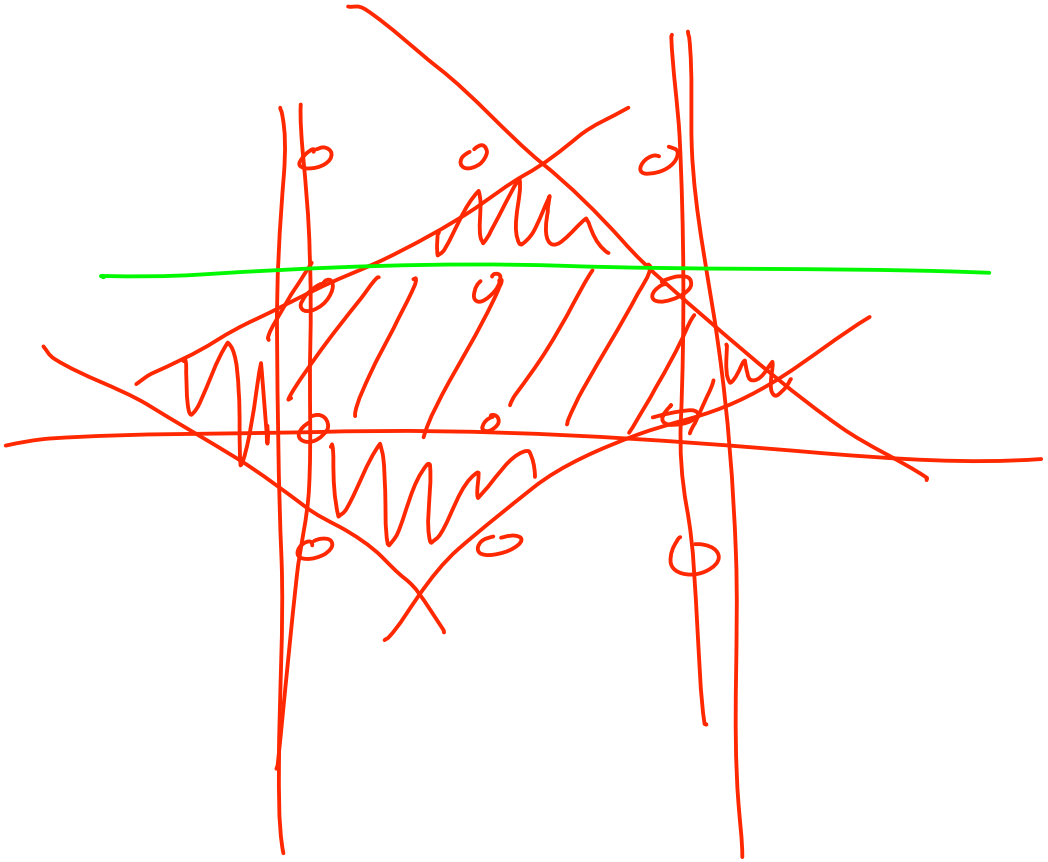
$h_2 + (1–B_2) + (1–M_2) \geq 1–s_{15}$

$(1–e_2) + e_1 + M_2 \geq 1–s_{16}$

…

$0 \leq M_1, h_1, e_1, M_2, B_2, h_2, e_2, s_1, \ldots, s_{21} \leq 1$

How should we relax?



Combine relaxation w/ search

- Given a schema: e.g., *10*0
- Substitute in fixed variables x_1, x_4 remain
- Relax integrality constraints for *s ⇒ smaller problem
- Solve relaxation gives lower bound ⇒ maybe prune!
- Quiz: in min problem, lower value for relaxation w/ ***** or *10*0 ?
  - A: *****

A random 3-CNF formula

$$
\begin{aligned}
&(x_5 \vee x_1 \vee x_2) \wedge (x_7 \vee x_2 \vee \bar{x}_4) \wedge (x_5 \vee x_2 \vee \bar{x}_8) \wedge (\bar{x}_6 \vee \bar{x}_1 \vee \bar{x}_7) \\
\wedge\, &(x_1 \vee x_3 \vee x_5) \wedge (\bar{x}_7 \vee x_1 \vee \bar{x}_6) \wedge (x_8 \vee x_5 \vee x_7) \wedge (\bar{x}_4 \vee \bar{x}_6 \vee \bar{x}_7) \\
\wedge\, &(\bar{x}_7 \vee x_2 \vee x_1) \wedge (\bar{x}_6 \vee x_4 \vee \bar{x}_4) \wedge (\bar{x}_2 \vee x_3 \vee \bar{x}_2) \wedge (x_4 \vee x_2 \vee \bar{x}_1) \\
\wedge\, &(x_1 \vee \bar{x}_6 \vee x_6) \wedge (x_7 \vee \bar{x}_8 \vee \bar{x}_3) \wedge (x_3 \vee \bar{x}_4 \vee x_4) \wedge (\bar{x}_4 \vee x_7 \vee \bar{x}_3) \\
\wedge\, &(x_2 \vee x_4 \vee x_1) \wedge (\bar{x}_6 \vee \bar{x}_7 \vee x_5) \wedge (\bar{x}_2 \vee x_7 \vee \bar{x}_4) \wedge (\bar{x}_5 \vee x_6 \vee x_3) \\
\wedge\, &(x_7 \vee \bar{x}_1 \vee x_6) \wedge (x_7 \vee x_4 \vee x_7) \wedge (\bar{x}_5 \vee \bar{x}_6 \vee x_5) \wedge (x_7 \vee x_8 \vee \bar{x}_1) \\
\wedge\, &(\bar{x}_1 \vee \bar{x}_1 \vee x_3) \wedge (\bar{x}_8 \vee x_3 \vee \bar{x}_3) \wedge (x_5 \vee x_4 \vee \bar{x}_6) \wedge (x_4 \vee \bar{x}_1 \vee x_4) \\
\wedge\, &(\bar{x}_8 \vee x_4 \vee x_4) \wedge (\bar{x}_4 \vee \bar{x}_4 \vee \bar{x}_1) \wedge (\bar{x}_8 \vee x_7 \vee x_7) \wedge (\bar{x}_2 \vee x_8 \vee \bar{x}_8) \\
&\wedge (x_1 \vee x_2 \vee x_6) \wedge (\bar{x}_5 \vee \bar{x}_2 \vee x_1)
\end{aligned}
$$

Example search tree

Branch & bound

[schema, value] = bb(F, sch, bnd)

- $[v_{rx}, rsch] = relax(F, sch)$
- if integer(rsch): return [rsch, v_{rx}, bnd]
- if $v_{rx} \geq bnd$: return [sch, v_{rx}, bnd]
- Pick variable x_i
- $[sch^{(0)}, v^{(0)}] = bb(F, sch/(x_i \rightarrow 0), bnd)$
- $[sch^{(1)}, v^{(1)}] = bb(F, sch/(x_i \rightarrow 1), \min(bnd, v^{(0)}))$
- if $(v^{(0)} \leq v^{(1)})$: return $[sch^{(0)}, v^{(0)}]$
- else: return $[sch^{(1)}, v^{(1)}]$