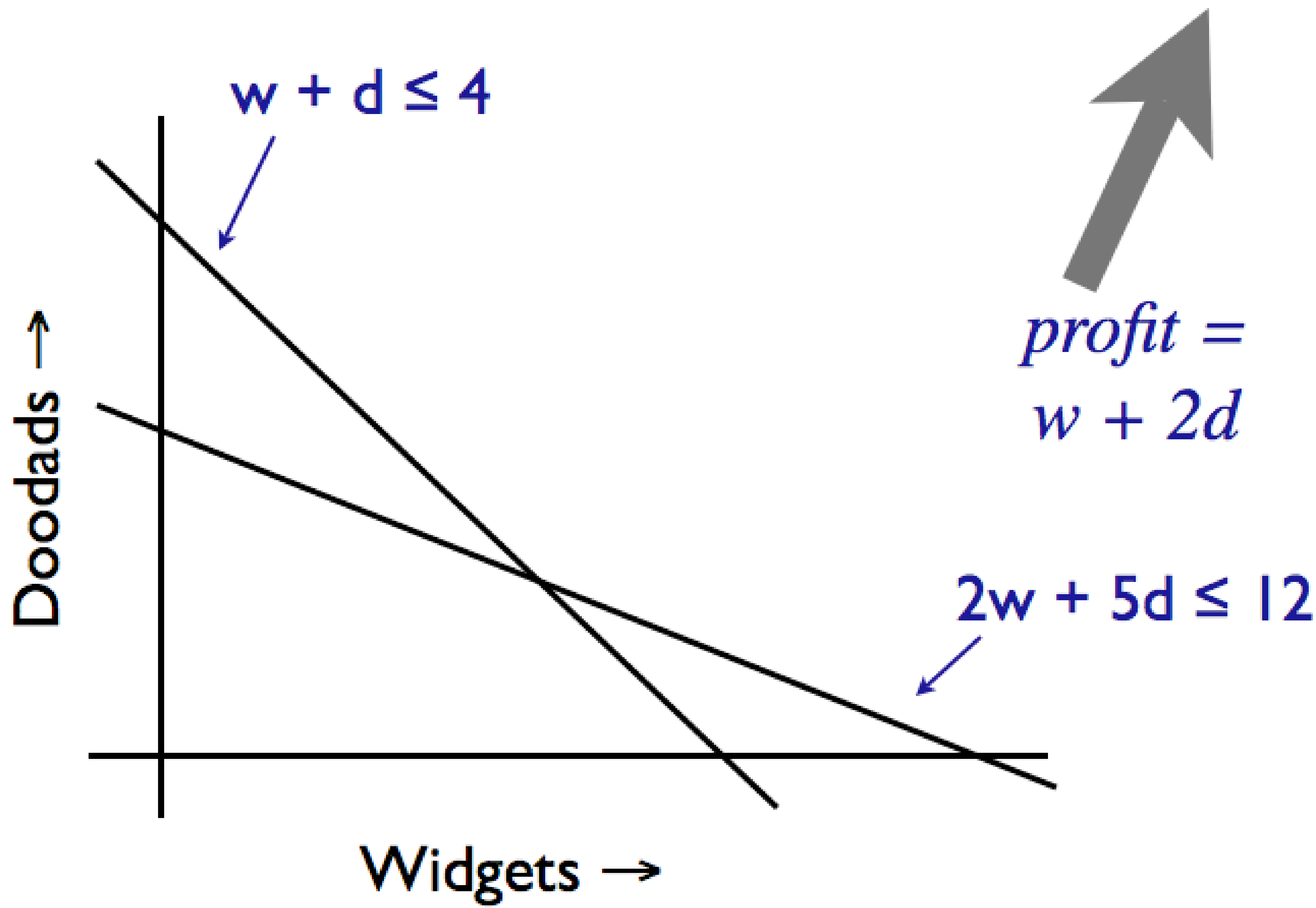
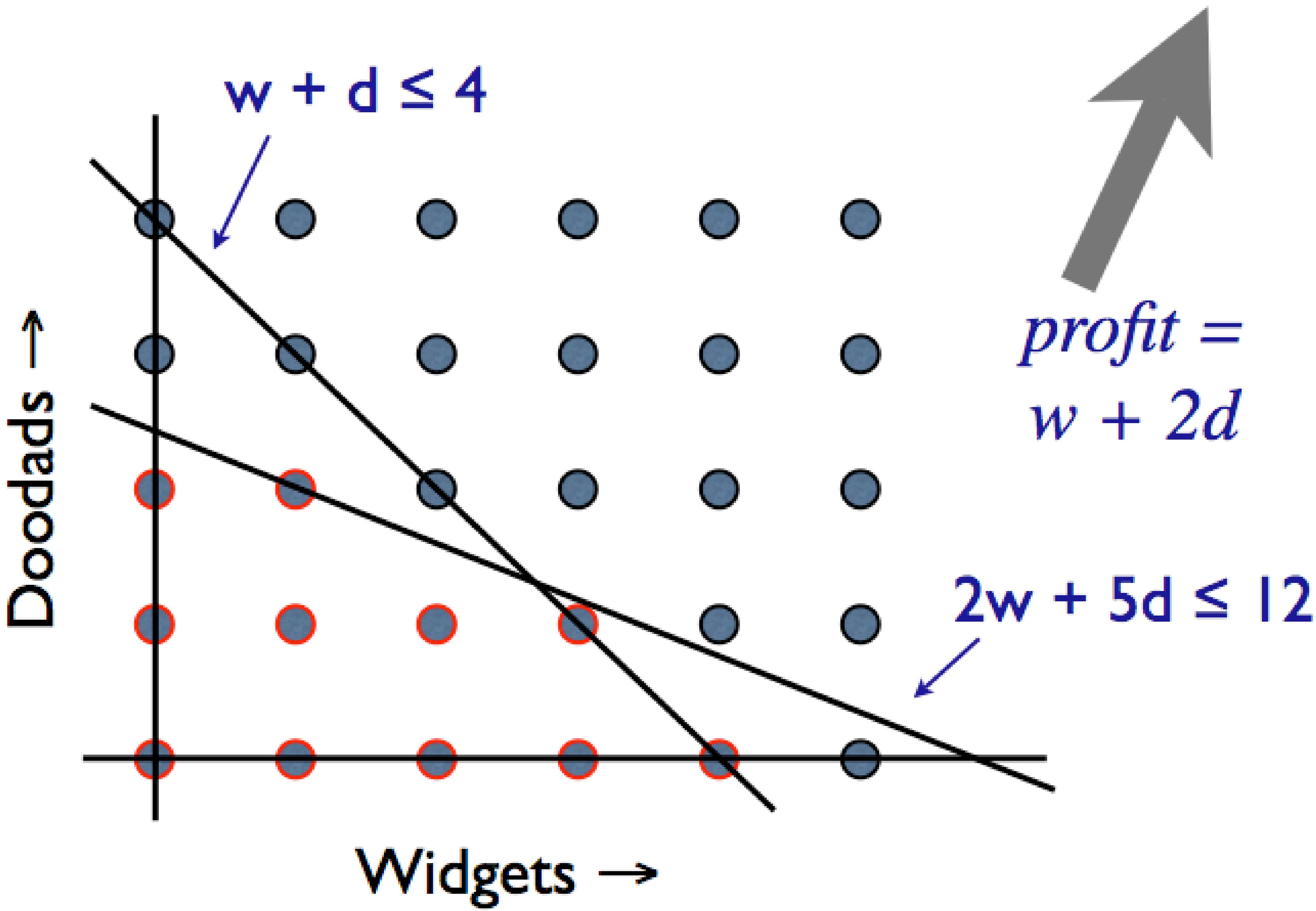
Production planning

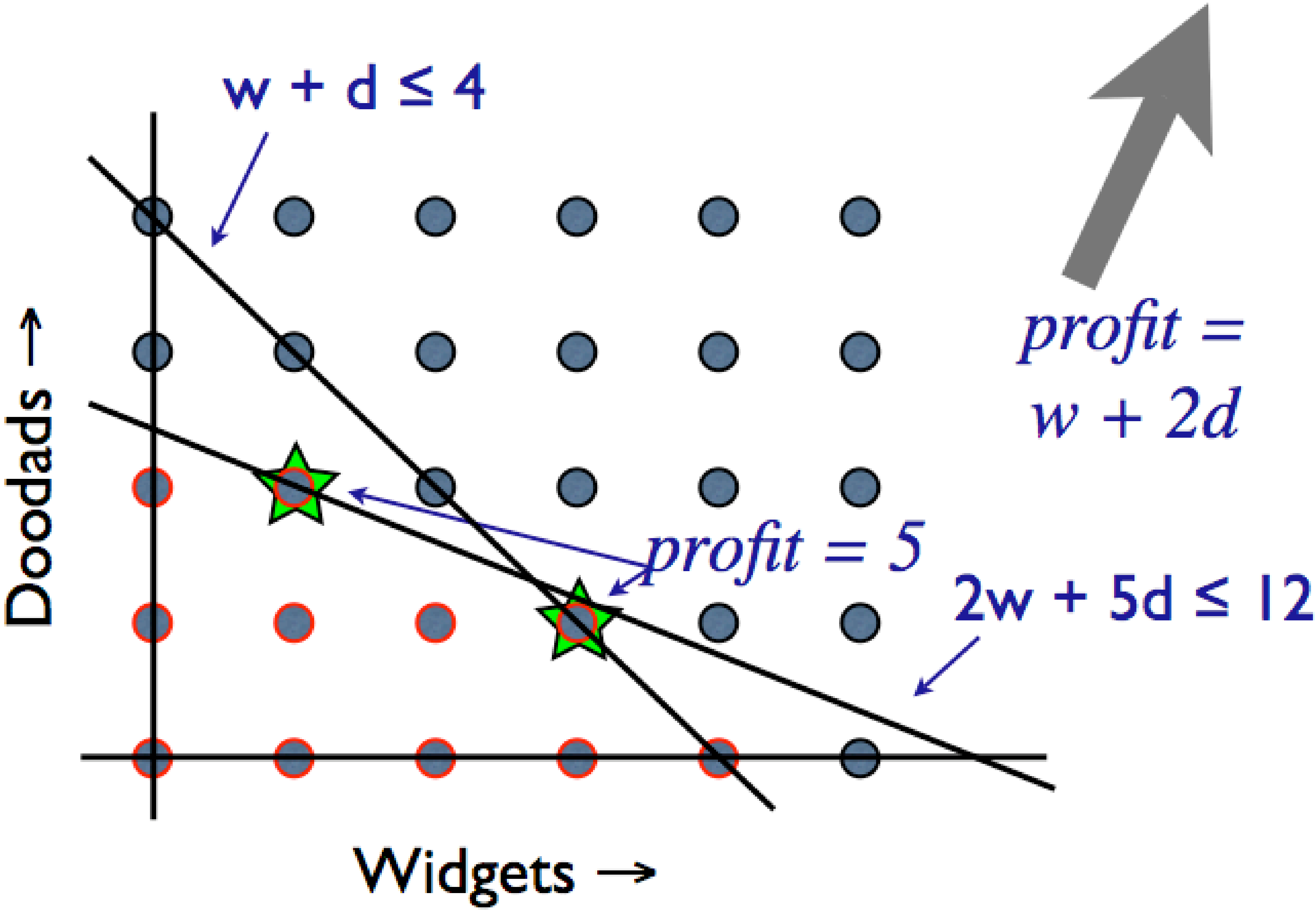
- Big factory, produces widgets, doodads
- Each widget:
  - 1 unit of wood, 2 units steel, profit \$1
- Each doodad
  - 1 unit wood, 5 units steel, profit \$2
- Have: 4M units wood, 12M units steel


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

A wrinkle

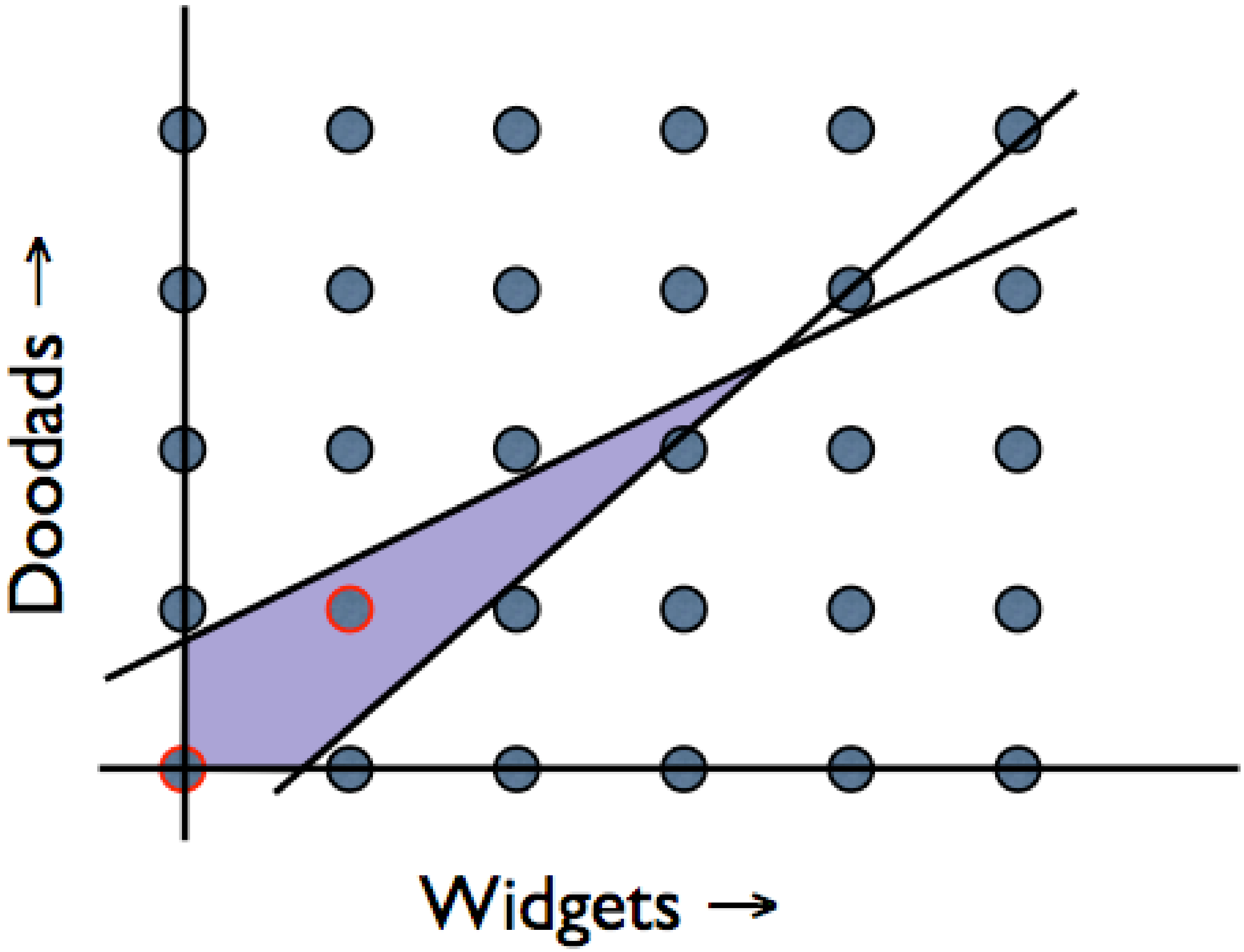
- Due to limitations of machinery, must produce in lots of exactly 1M widgets or 1M doodads


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



Integer linear programs

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

E.g., max s.t.

Variations

- Mixed integer linear program (MILP)
- Integer quadratic program
- 0-1 ILP

Combinatorial optimization

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

Applications

- SAT

- MAXSAT

Applications

- Facility location problem:
  - have n stores, at positions y_1, y_2, …
  - can build m warehouses, at x_1, x_2, …
  - minimize distance from each store to nearest warehouse
- Extra vars:
- min s.t.

Embedding logic in MILPs

- $z \implies a^T x + b \geq 0$
- OR:
- k-of-n:

Piecewise-linear functions

Applications

- Planning (e.g., STRIPS)

Have cake & eat it as CNF

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

Applications

- Scheduling

Solving combinatorial optimization problems

- Two basic strategies

- And,

Basic search

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

- Schema is **full** if no *s: e.g.,
- Notation: schema/(variable→value)
- E.g., 10**1/(x_3→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 $v^{(0)} \geq v^{(1)}$ return $[sch^{(0)}, v^{(0)}]$
  else return $[sch^{(1)}, v^{(1)}]$

Exercise

- SAT problem w/ XOR constraints:

  search($a \oplus b \wedge b \oplus c \wedge c \oplus a$, ***)

- How many total calls to search()?

$$\text{search}(a \oplus b \wedge b \oplus c \wedge c \oplus a)$$

Constraint propagation

- Start w/ table of feasible values
- When we set a var, look at its constrs
  - e.g., set a = 0:
  -
- If a domain becomes empty:
- If one becomes singleton:

Search tree

Relaxation

- min f(x) s.t. $x \in S_1$ (*)
- min g(y) s.t. $y \in S_2$ (**)
- (*) is a **relaxation** of (**) if:
  –
  –
- Example:

Why are relaxations useful?

- Relaxation may be
- From solution of relaxed problem:
- Suppose x*, y* are optimal solutions to original / relaxed problems
  - we know:

Example: have & eat LP

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

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.,
- Substitute in fixed variables
- Relax integrality constraints for *s
- Solve relaxation
- Quiz: in min problem, lower value for relaxation w/ or ?
  - 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 [$sch^{(1)}$, $v^{(1)}$]