# 15-780: Grad AI
# Lecture 6: Optimization

Geoff Gordon

# Admin

- help@cs was unable to retrieve Monday slides from dead hard drive
- I will type up my notes and put on web
- If anyone has questions not answered by notes, email me; if sufficient interest I can schedule a Q&A session outside of class

# Last time, on Grad AI

# FOL

- Quantifiers, variables, scoping
- Models of FOL expressions with quantifiers
- Unification and resolution in FOL

# MGUs

- Someone asked on Mon whether the most general unifier of two first-order expressions is unique
- Yes: MGUs are unique up to renaming of variables

# Planning

- Planning languages like STRIPS
  - operators, preconditions, effects
- Linear planners
  - forward and backward chaining
- Partial-order planning
  - action orderings, open preconditions, guard intervals, plan refinement

# Plan Graphs

# Planning & model search

- For a long time, it was thought that model search (using a logical KB describing a planning domain) was a non-starter as a planning algorithm
- More recently, people have written fast planners that
  - propositionalize the domain
  - turn it into a CSP or SAT problem
  - search for a model

# Plan graph

- Tool for making good CSPs: plan graph
- Encodes a subset of the constraints that plans must satisfy
- Remaining constraints are handled during search (by rejecting solutions that violate them)

# Example

- Start state: have(cake)
- Goal: have(cake) ^ eaten(cake)
- Operators: bake, eat

# Operators

- Bake
  - pre: -have(cake)
  - post: have(cake)
- Eat
  - pre: have(cake)
  - post: -have(cake), eaten(cake)

# Plan graph

have

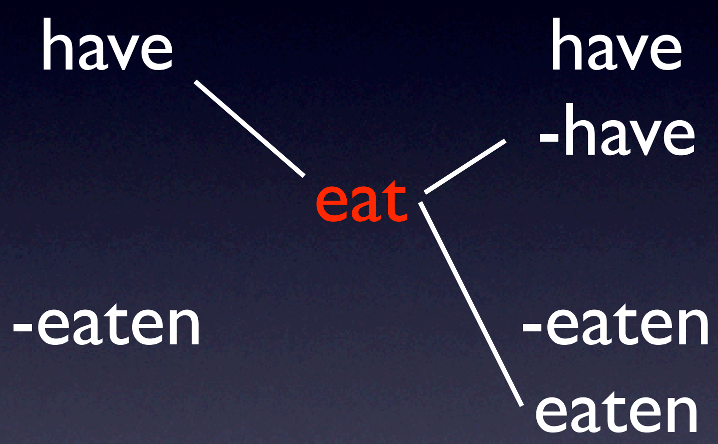
-eaten

- Alternating levels: states and actions
- First level: initial state

# Plan graph

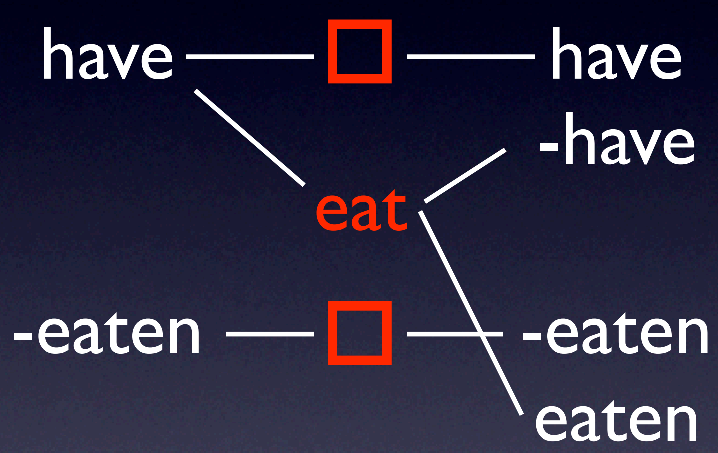

- First action level: all applicable actions
- Linked to their preconditions

# Plan graph



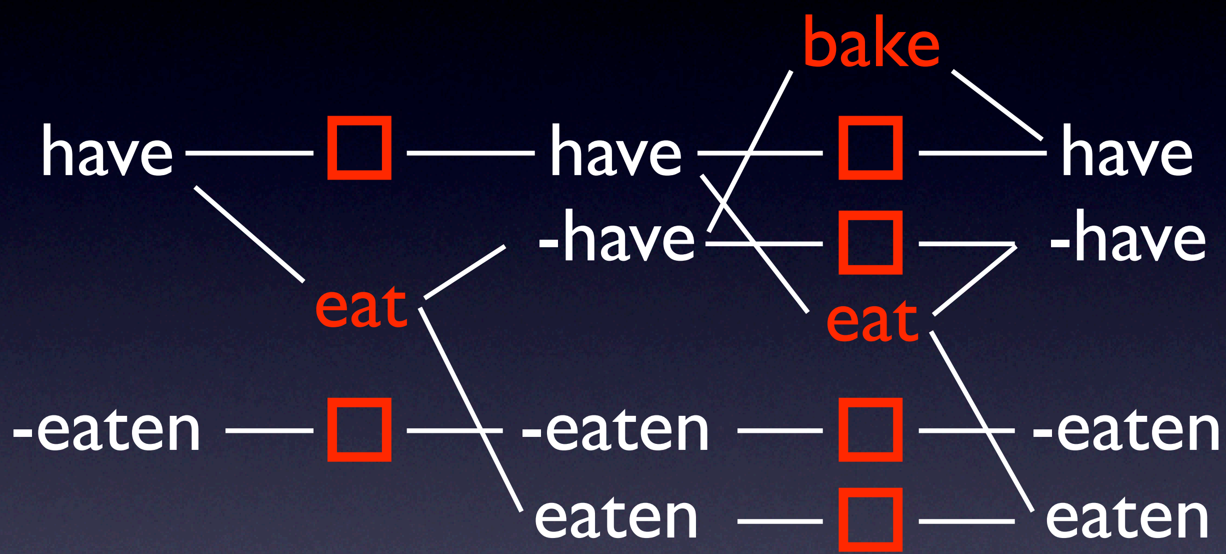

- Second state level: add effects of actions to get literals that could hold at step 2

# Plan graph




- Also add **maintenance actions** to represent effect of doing nothing

# Plan graph

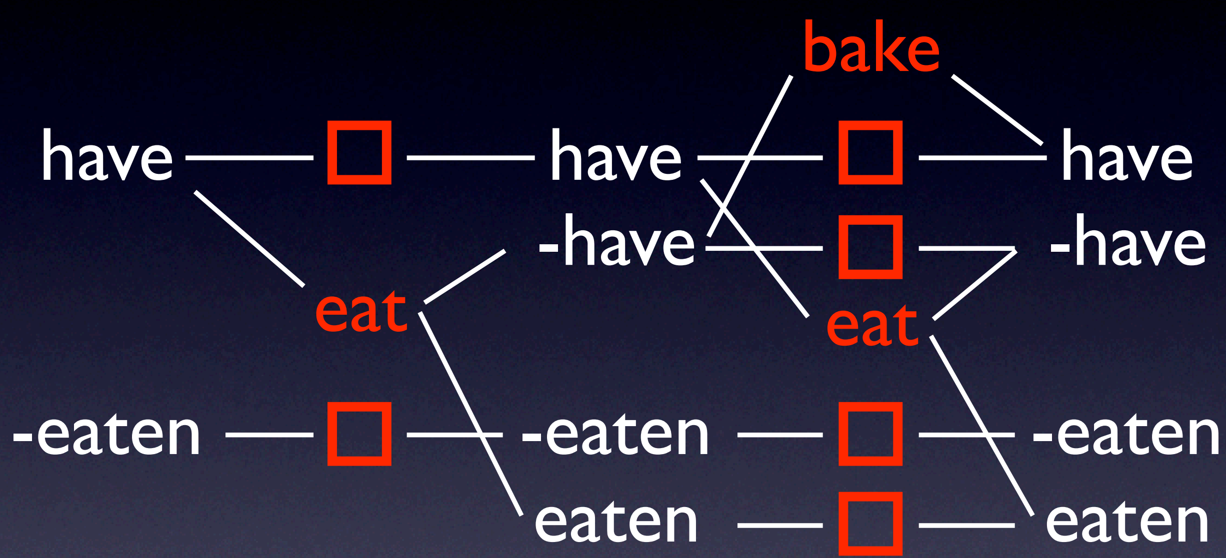



- Extend another pair of levels: now bake is a possible action

# Plan graph

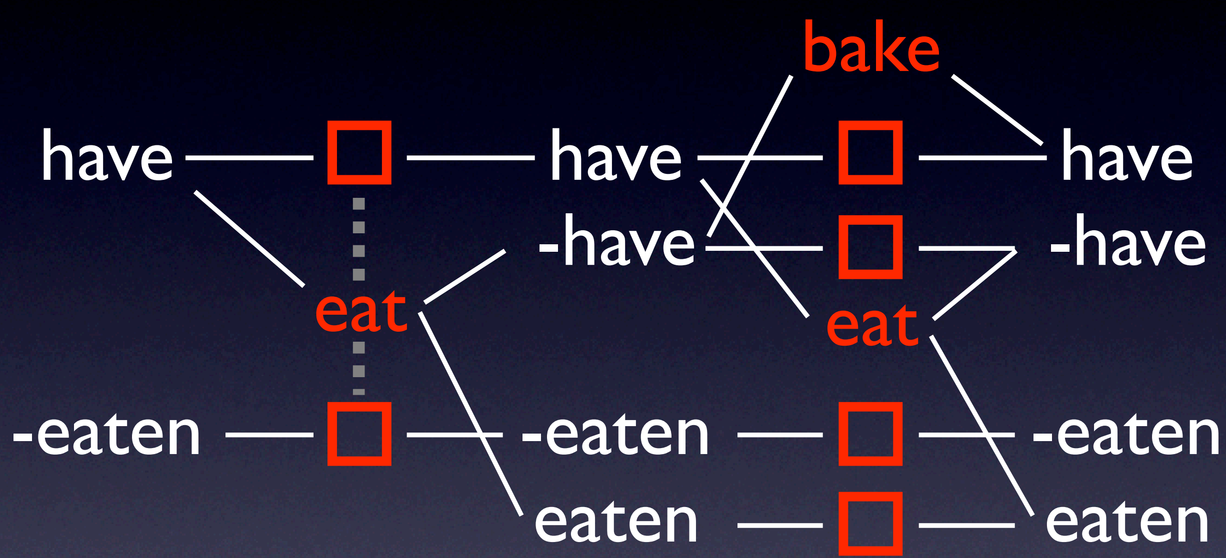
- Can extend as far right as we want
- Plan = subset of the actions at each action level
- Ordering unspecified within a level

# Plan graph



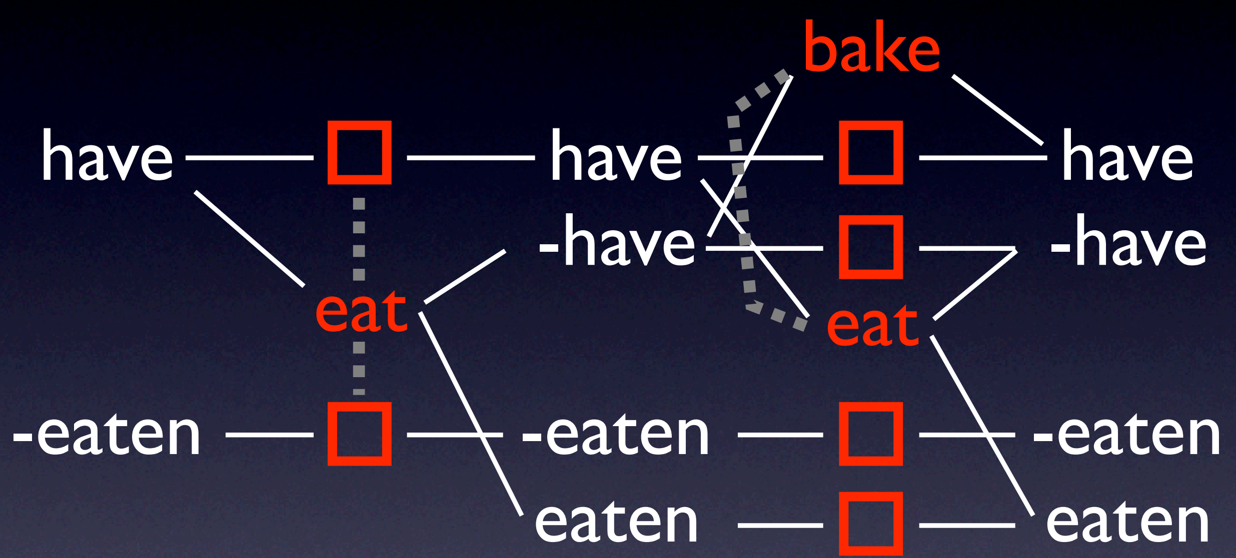

- In addition to the above links, add **mutex** links to indicate mutually exclusive actions or literals

# Plan graph



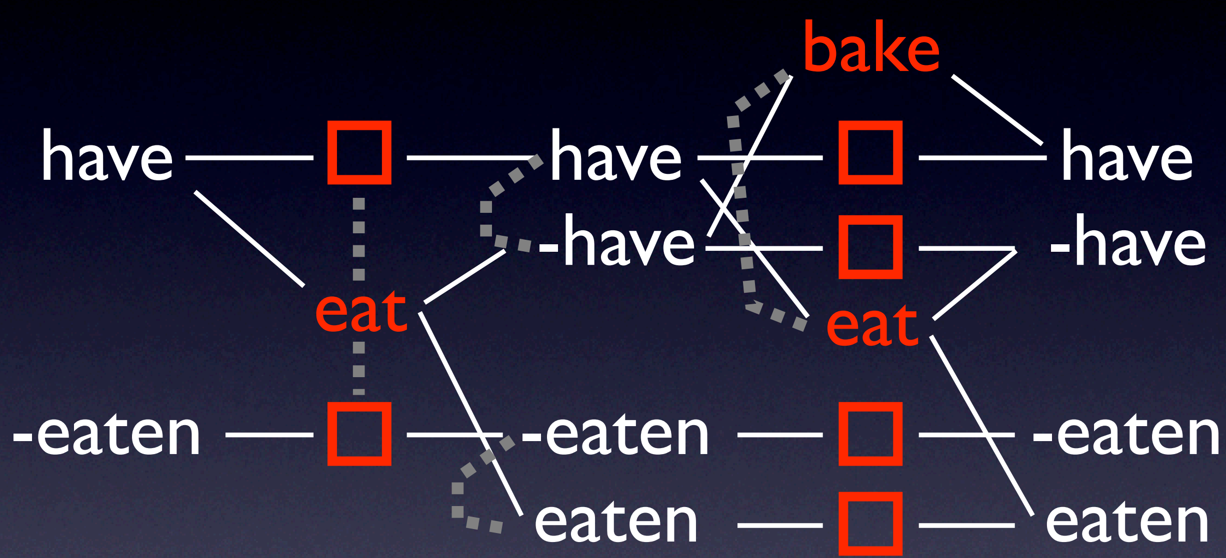

- Actions which assert contradictory literals are mutex

# Plan graph



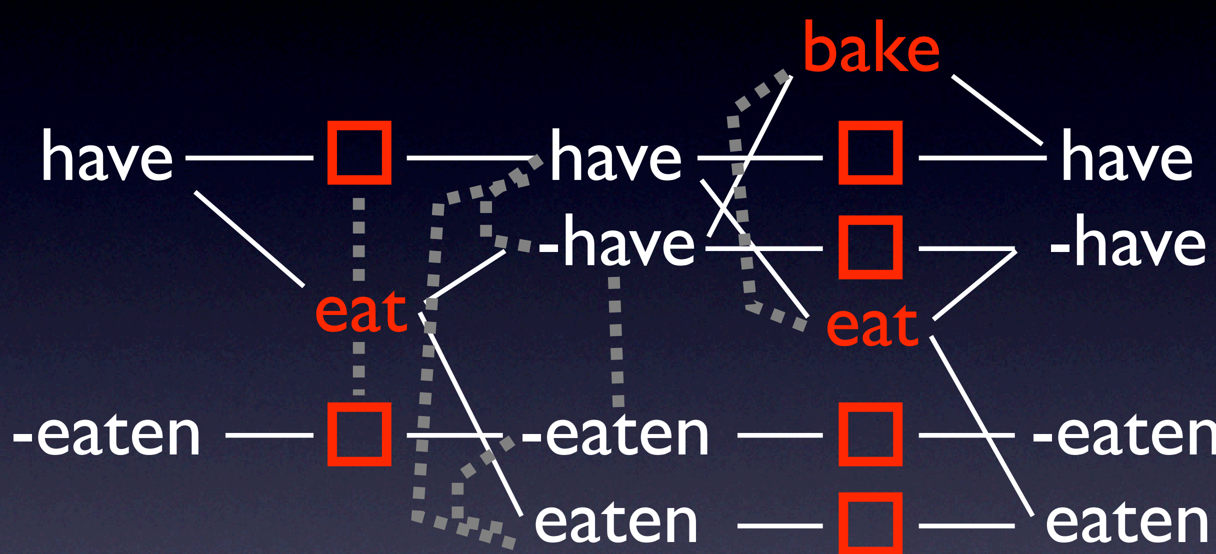

- Actions are also mutex if one deletes a precondition of the other, or if their preconditions are mutex

# Plan graph




- Literals are mutex if they are contradictory

# Plan graph




- Or if there is no non-mutex set of actions that could achieve both

# Getting a plan

- Build the plan graph out to some length k
- Translate to a SAT formula
- Search for a satisfying assignment
- If found, read off the plan
- If not, increment k and try again
- There is a test to see if k is big enough

# Translation to SAT

- One variable for each pair of literals in each state level
- One variable for each action in each action level
- Mutex constraints between actions or literals: add clause $(x \oplus y)$
- Note: mutexes are redundant, but help anyway

# Action constraints

- Each action can only be executed if all of its preconditions are present:

  $act_{t+1} \Rightarrow pre1_t \wedge pre2_t \wedge \ldots$

- If executed, action asserts its postconditions:

  $act_{t+1} \Rightarrow post1_{t+2} \wedge post2_{t+2} \wedge \ldots$

- In order to achieve a literal, we must execute an action that achieves it
  - $post_{t+2} \Rightarrow act1_{t+1} \vee act2_{t+1} \vee \ldots$

# Initial & goal constraints

- Goals must be satisfied at end:

  $goal1_T \wedge goal2_T \wedge \ldots$

- And initial state holds at beginning:

  $init1_I \wedge init2_I \wedge \ldots$

# Optimization and Search

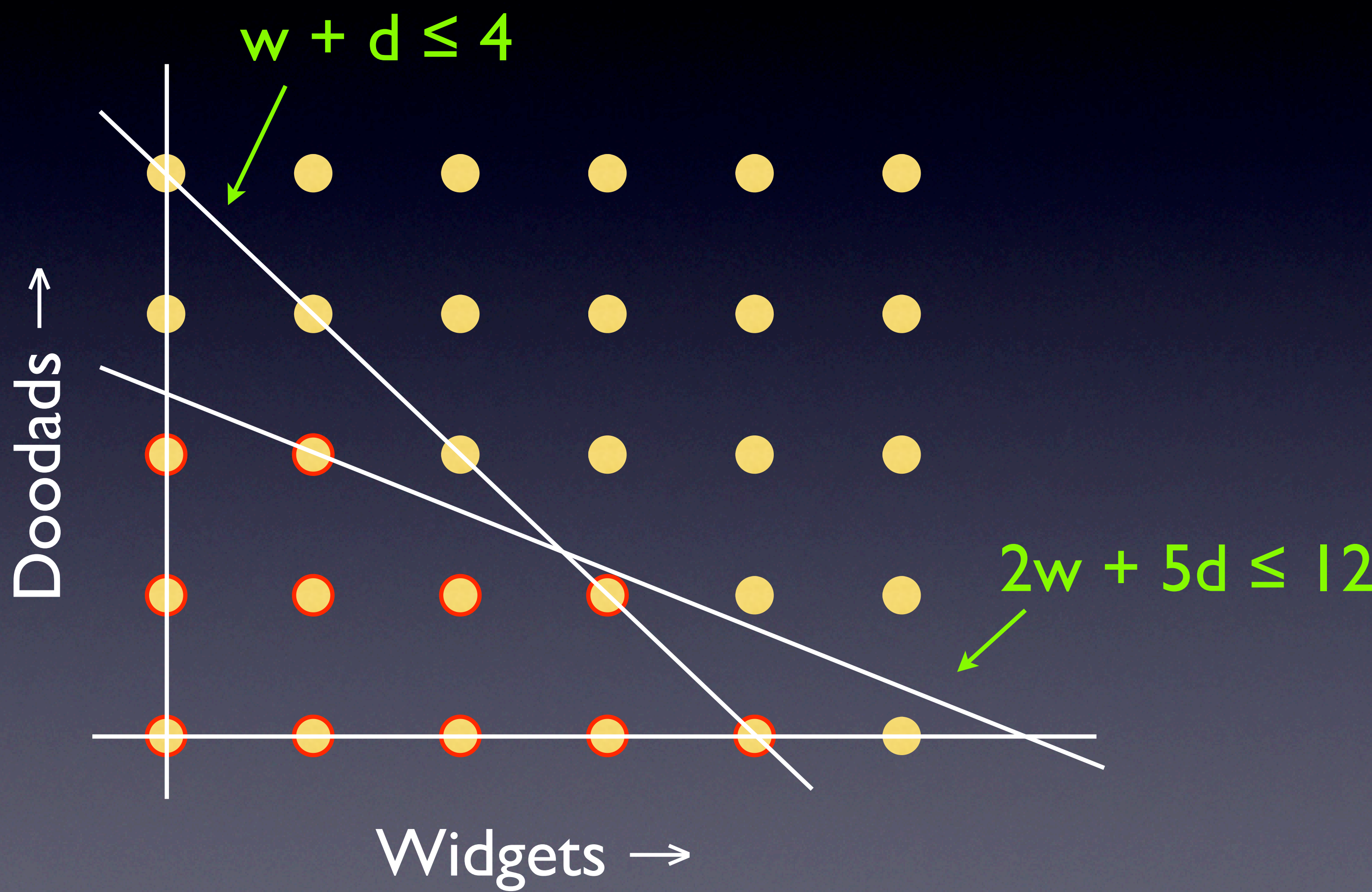
# Search problem

- Typical search problem: CSP or SAT
- Description: variables, domains, constraints
- Find a solution that satisfies constraints
- Any satisfying solution is OK

# Example search problem

- You run a factory that makes widgets and doodads
- Each widget takes 1 unit of wood and 2 units of steel to make
- Each doodad uses 1 unit of wood, 5 of steel
- You have 4 units of wood and 12 units of steel; design a feasible production schedule

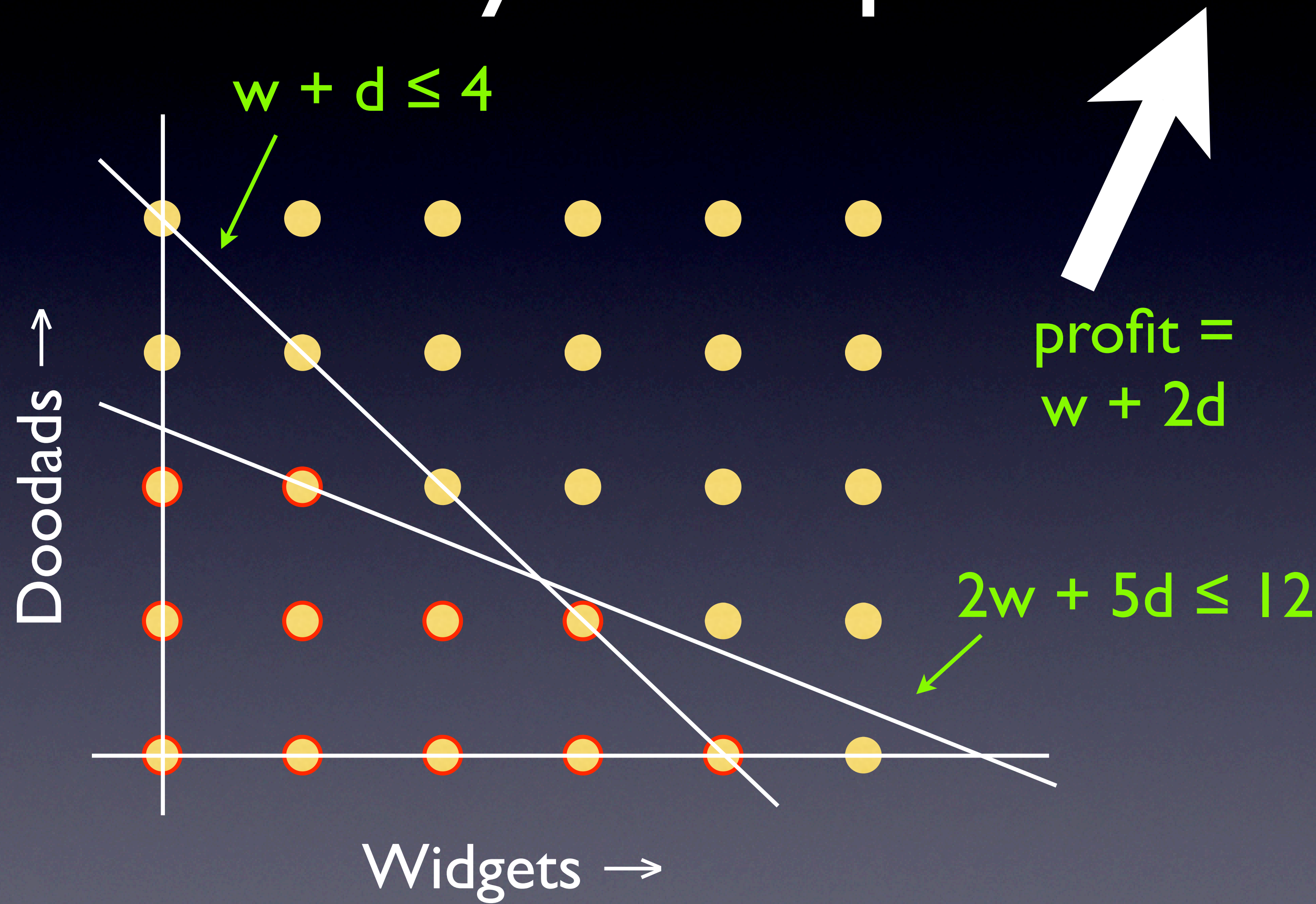
# Factory example



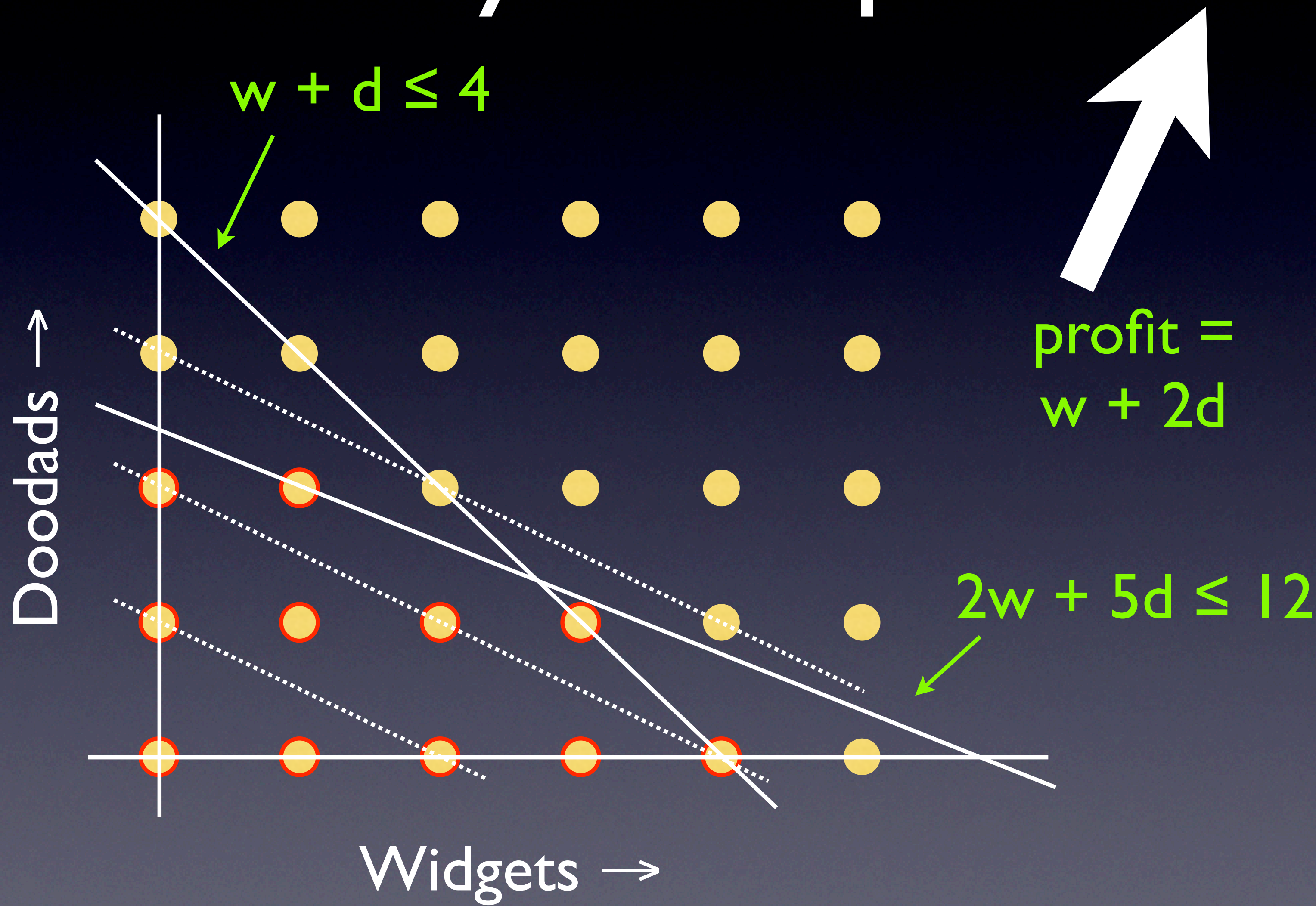
# Optimization

- Not all feasible solutions are equally good
- Within feasible set, want to optimize an objective function
- E.g., maximize profit:
  - Each widget yields a profit of $1
  - Each doodad nets $2

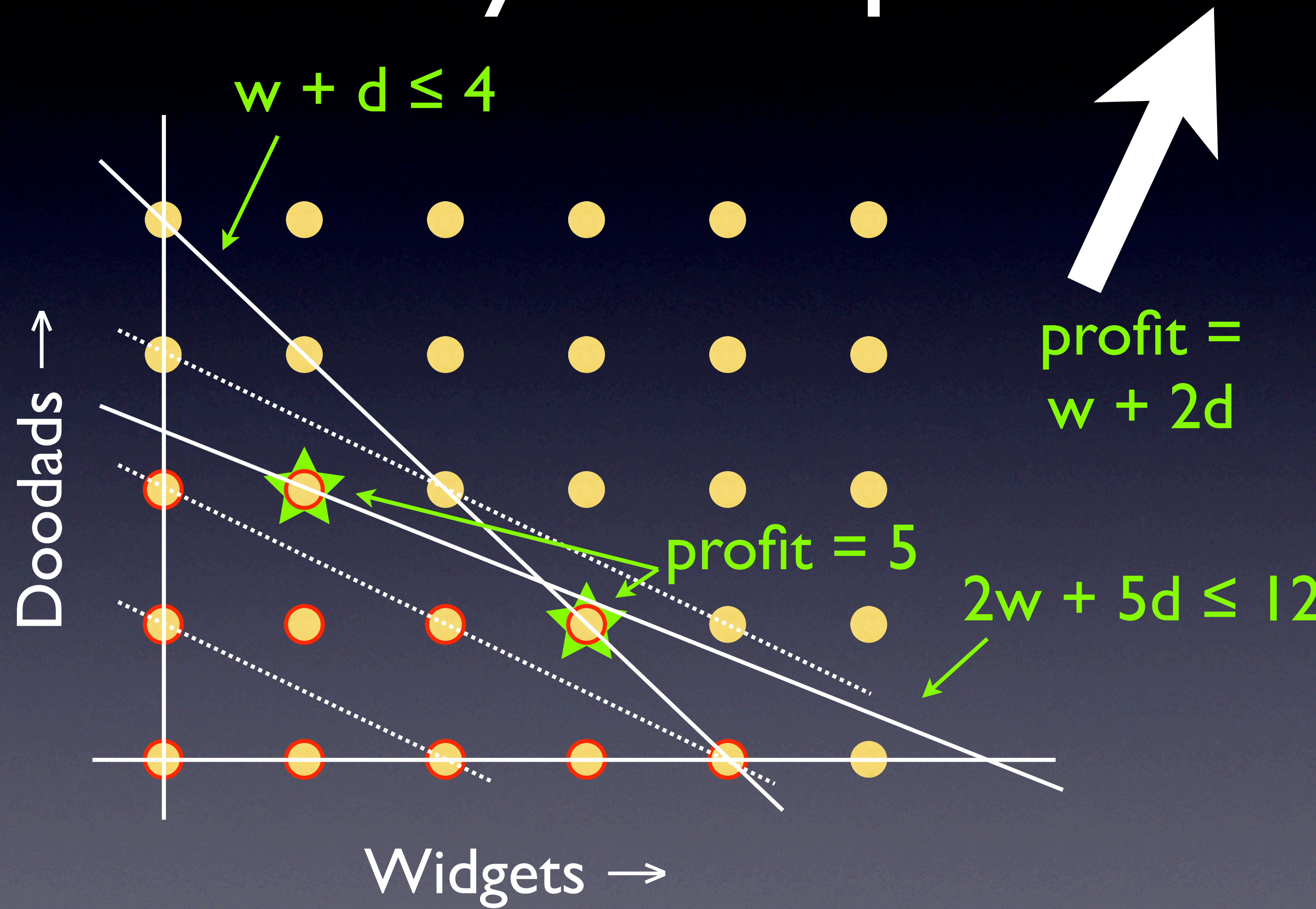
# Factory example



# Factory example



# Factory example



# ILP

- This type of optimization problem is called an **integer linear program**
- Interesting related problems:
  - 0-1 ILP: all variables in {0, 1}
  - SAT: 0-1 ILP with all constraints of form

    $x + (1-y) + (1-z) \geq 1$
  - LP: lift integer restriction, all variables in $\mathbb{R}$
  - MILP: some variables in $\mathbb{R}$

# Search

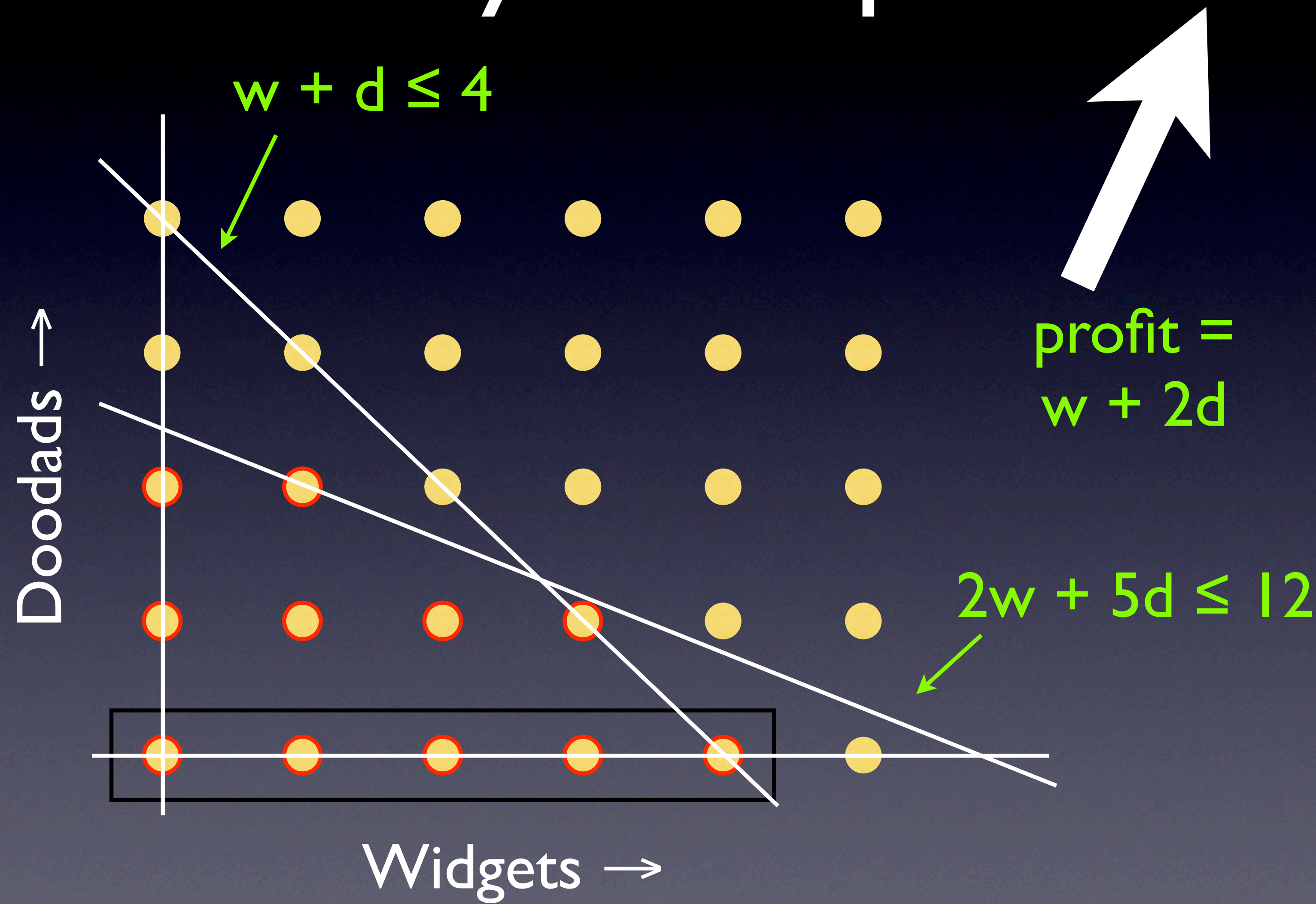
- Can still use search algorithms like DFID for optimization problems
- Just remember the best objective value seen so far
- This is a fine algorithm, but we can often do better!

# Bounds

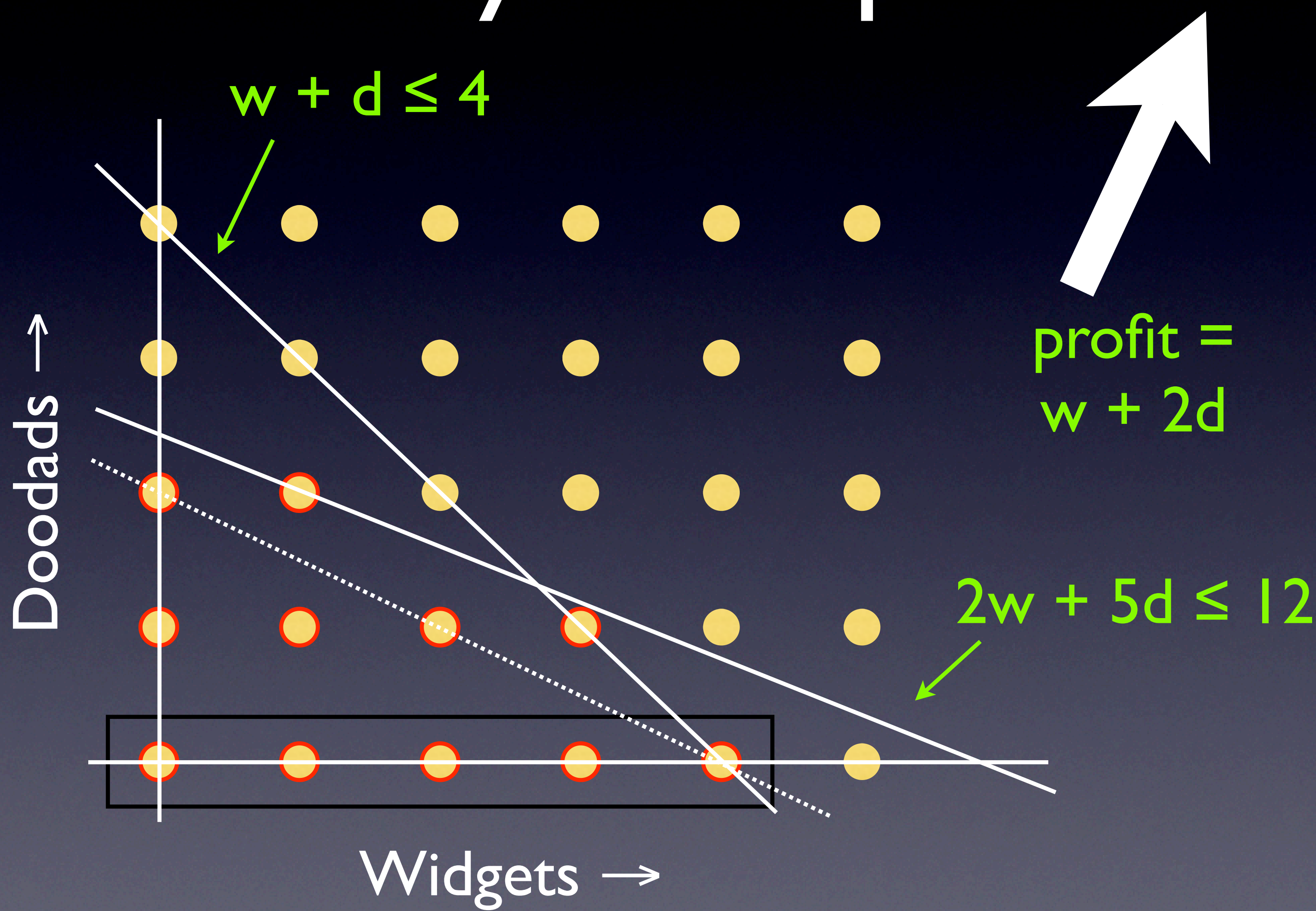
# Smarter algorithms

- We can build smarter algorithms by remembering bounds on optimal value
- First idea: if we have a solution with profit 3, add a constraint "profit ≥ 3"
- If we then find a solution with profit 5, replace constraint with "profit ≥ 5"

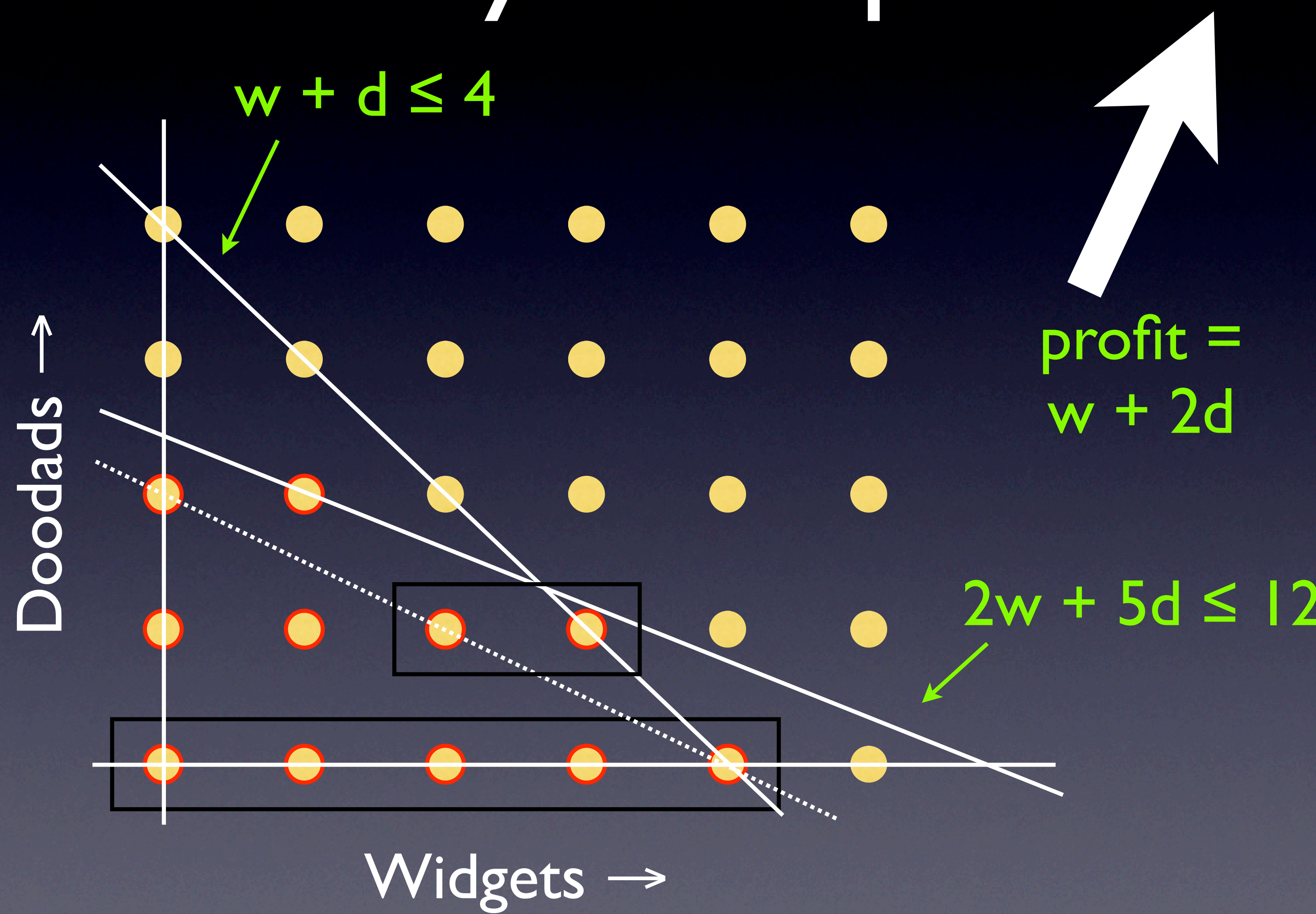
# Factory example



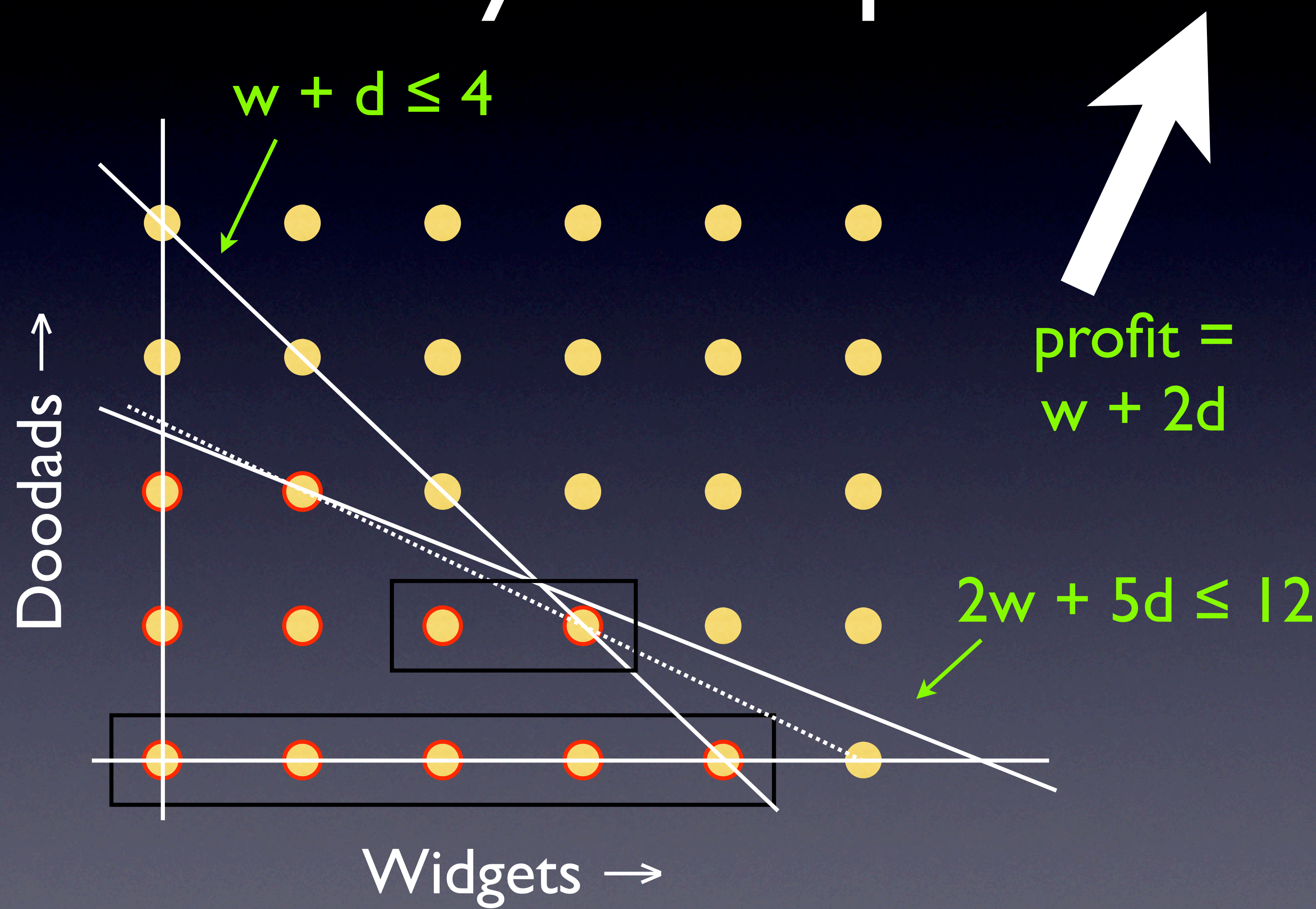
# Factory example



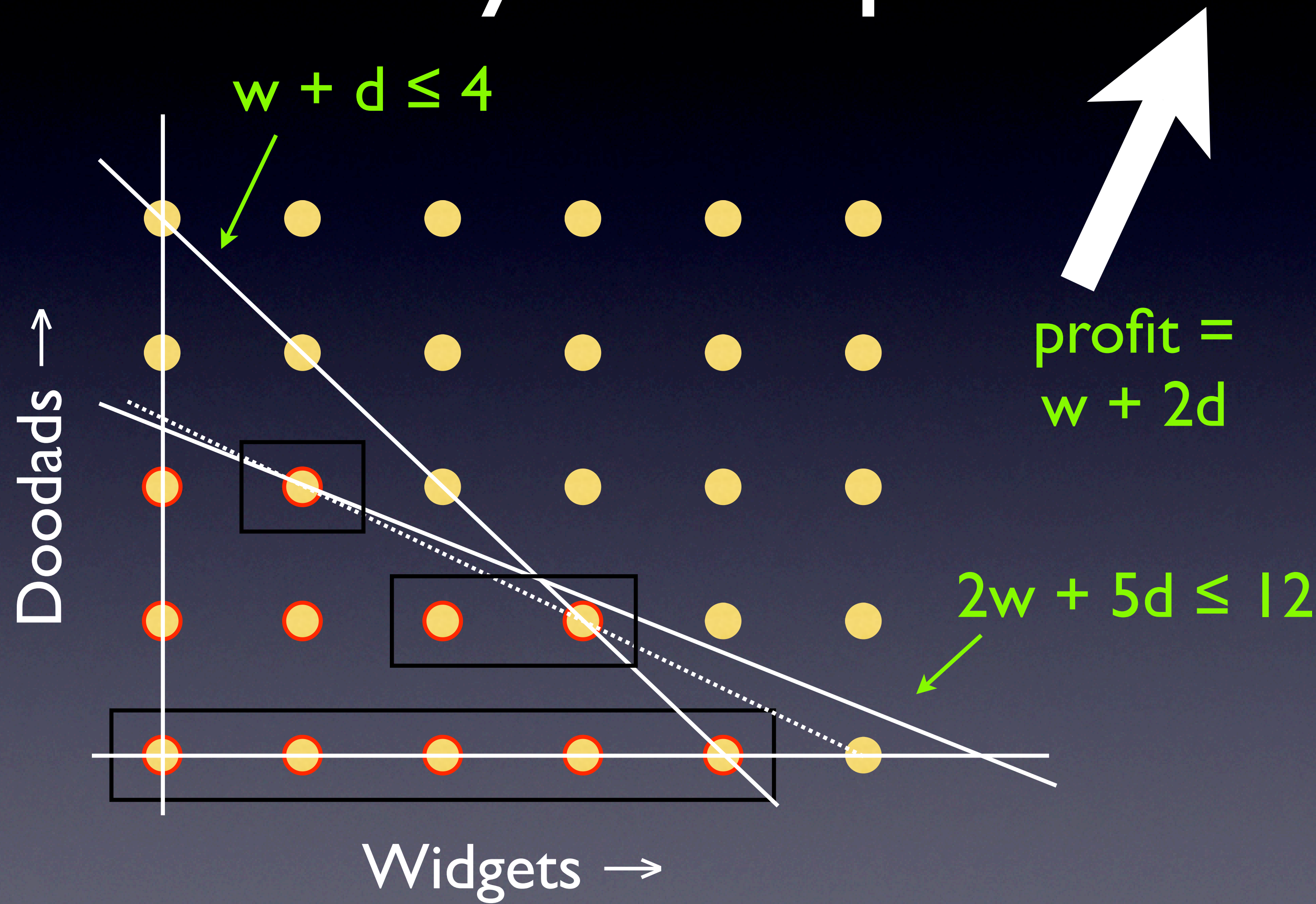
# Factory example



# Factory example



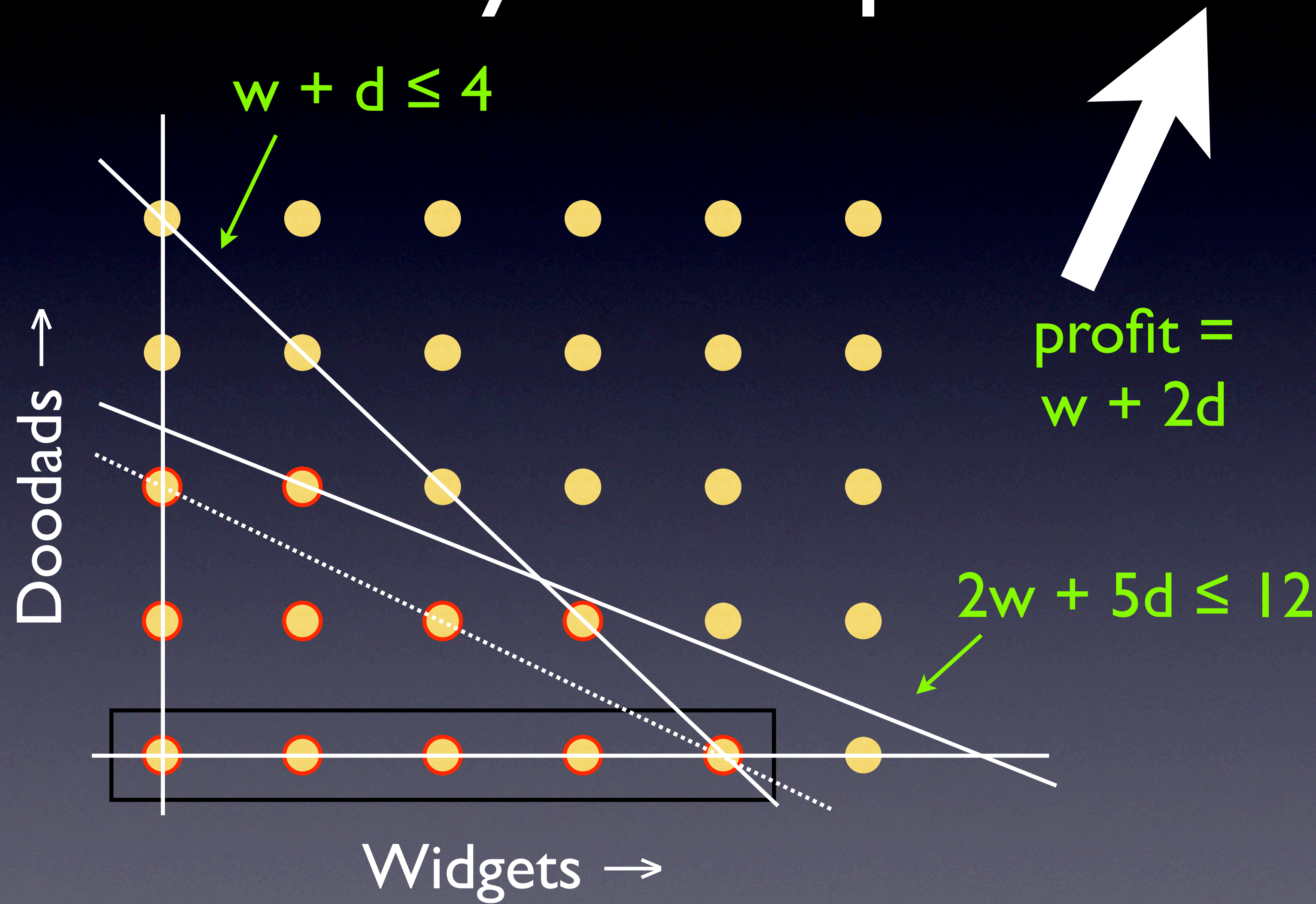
# Factory example



# Upper bounds

- Suppose we're partway finished: we've examined a few nodes and found a solution of profit $4

# Factory example



# Upper bounds

- We have a solution of profit \$4
- How much profit would we lose by stopping now?
- Might we find a node with profit \$73 if we kept looking?

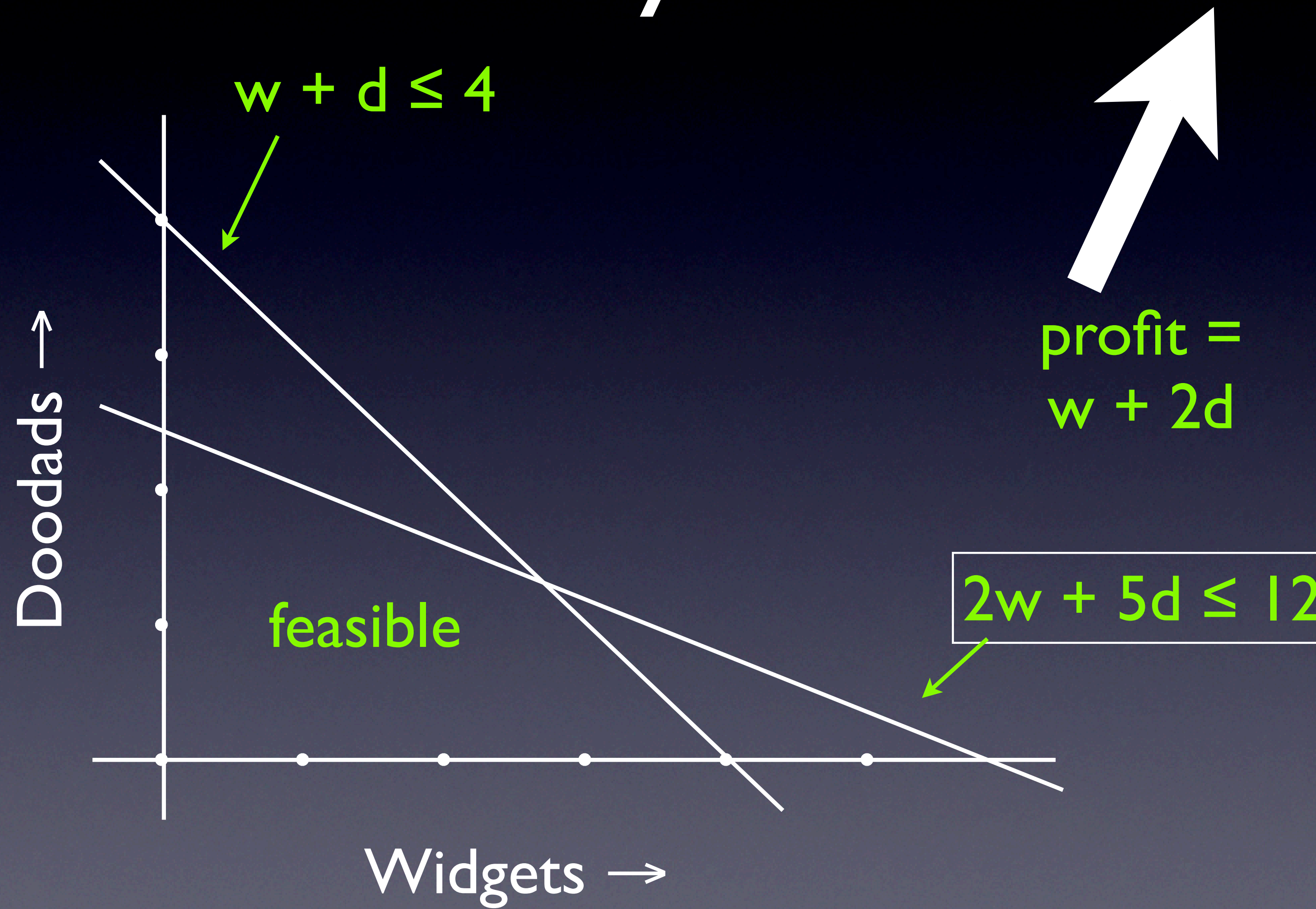
# Relaxation

- First idea: what if we solve an easier version of the problem?
- If we make feasible region bigger, objective value can only get better
- So, value of relaxed problem is an upper bound on value of original problem

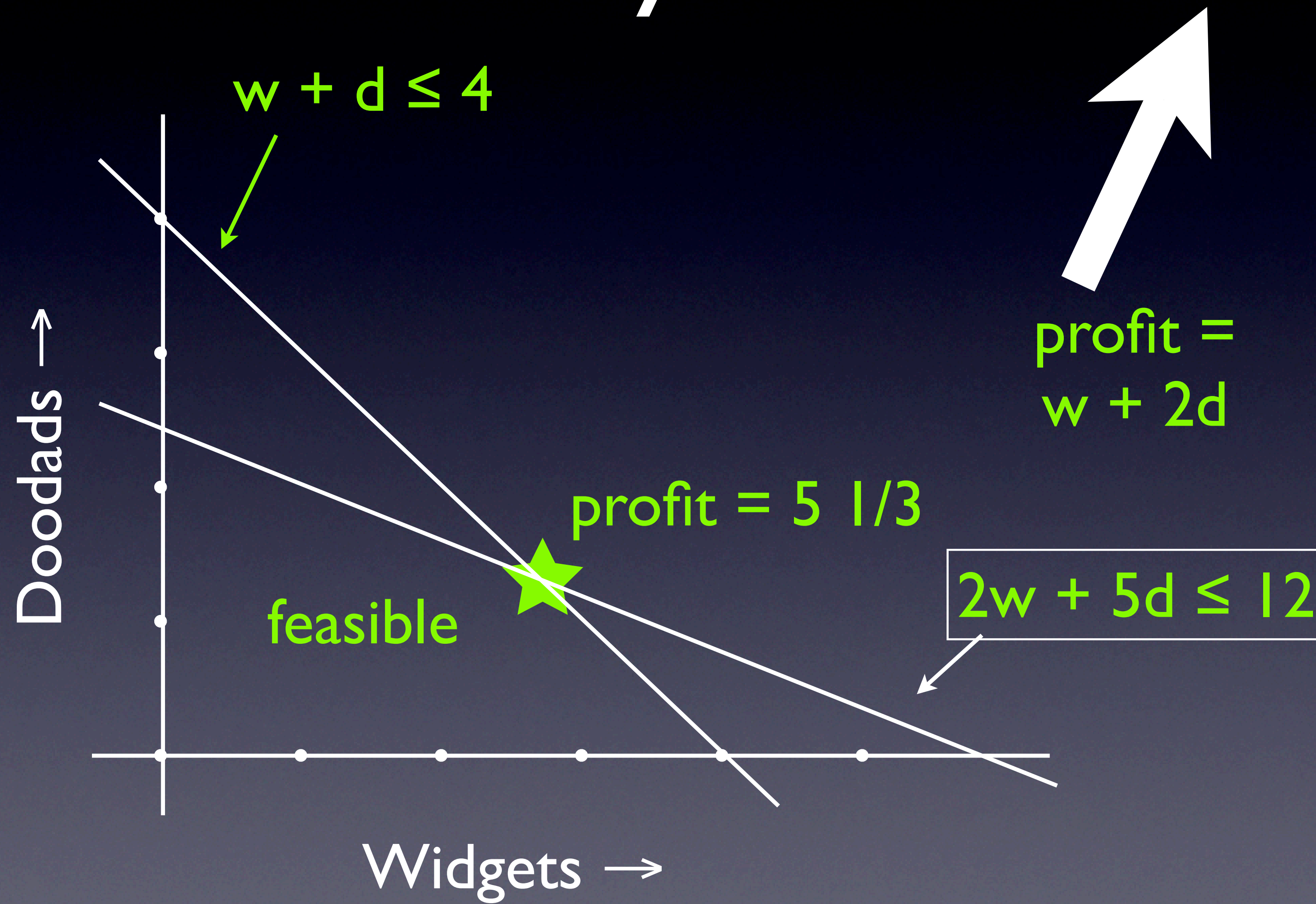
# LP relaxation

- Nice way of making feasible region bigger: drop integrality constraints
- Called the **LP relaxation** of our problem
- LPs are efficiently solvable (see below)

# Factory LP



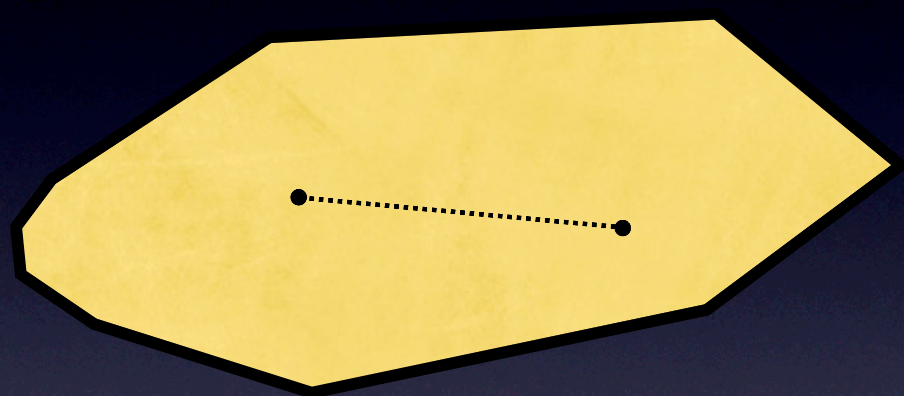
# Factory LP



# Complexity

- It is NP-complete to test whether it is possible to achieve objective ≥ k in an MILP
- But LPs can be solved in poly time
  - rough estimate: solving an LP with n variables and m constraints ~50–200x as expensive as n x m linear regression
- The difference from ILP: convexity

# Convex sets

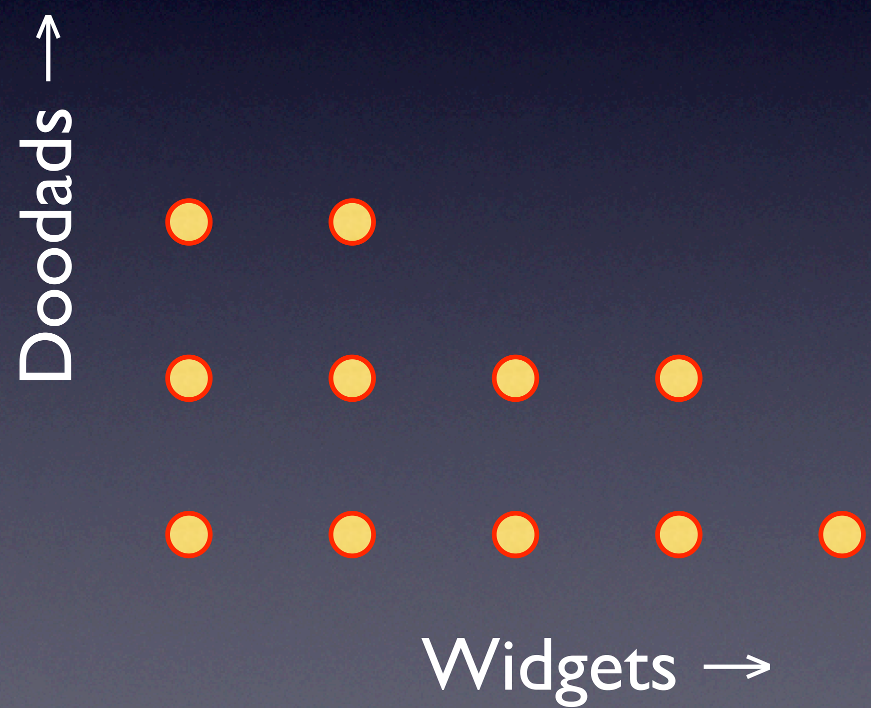


- Convex set C: if a, b in C, then C contains line segment ab

# Convex functions

- Convex function: **epigraph** is convex
- Epigraph = { (x, y) | y ≥ f(x) }
- Implies **level sets** convex: { x | f(x) ≤ k }

# ILP feasible region



# Convex optimization

- LP: minimize a linear objective over a polyhedral convex region
- Convex program: minimize a convex objective over a convex region
- Both are poly-time solvable (LP exactly, CP to within $\varepsilon$, poly in $1/\varepsilon$)

# Algorithms

- For LP
  - simplex: first algorithm, not always poly time
  - ellipsoid: first poly-time algorithm, but often slower than simplex
  - barrier methods: poly-time, fast in practice
- For CP: ellipsoid or barrier

# More bounds

# What if we're lazy?

- It was a lot of work to get that bound: had to solve the LP and find its exact optimum
- Can we do less work—perhaps find a suboptimal solution to LP?
- Sadly, a non-optimal feasible point in the LP relaxation gives us no useful bound

# A simple bound

- Recall:
  - constraint $w + d \leq 4$ (limit on wood use)
  - profit $w + 2d$
- Since $w, d \geq 0$,
  - profit $= w + 2d \leq 2w + 2d$
- And, doubling both sides of constraint,
  - $2w + 2d \leq 8$

# The same trick works twice

- Try other constraint (steel use)
  - $2w + 5d \leq 12$
- $2 * \text{profit} = 2w + 4d \leq 2w + 5d \leq 12$
- So $\text{profit} \leq 6$

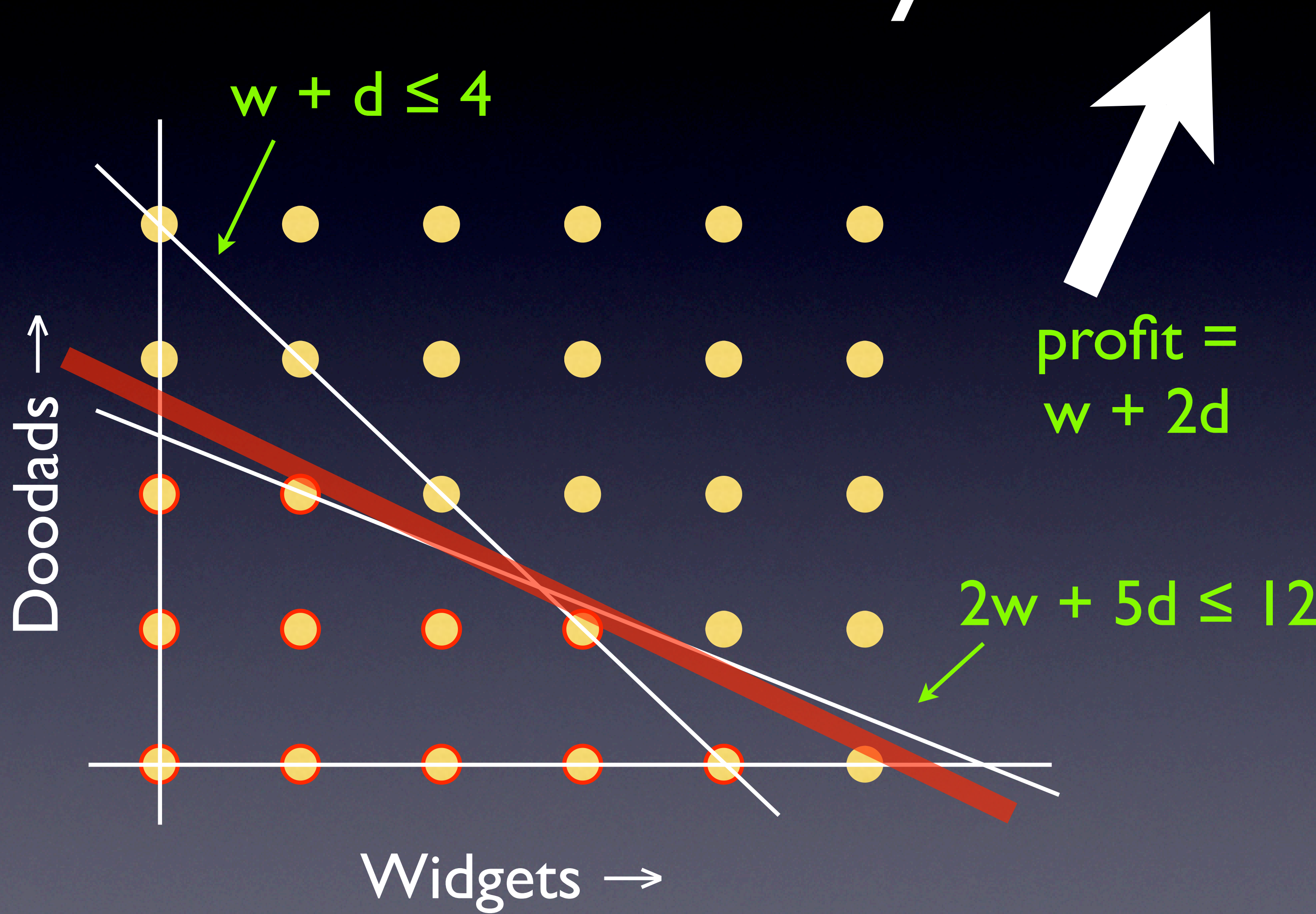
# In fact it works infinitely many times

- We could take any positive linear combination of our constraints (negative weights would flip sign)

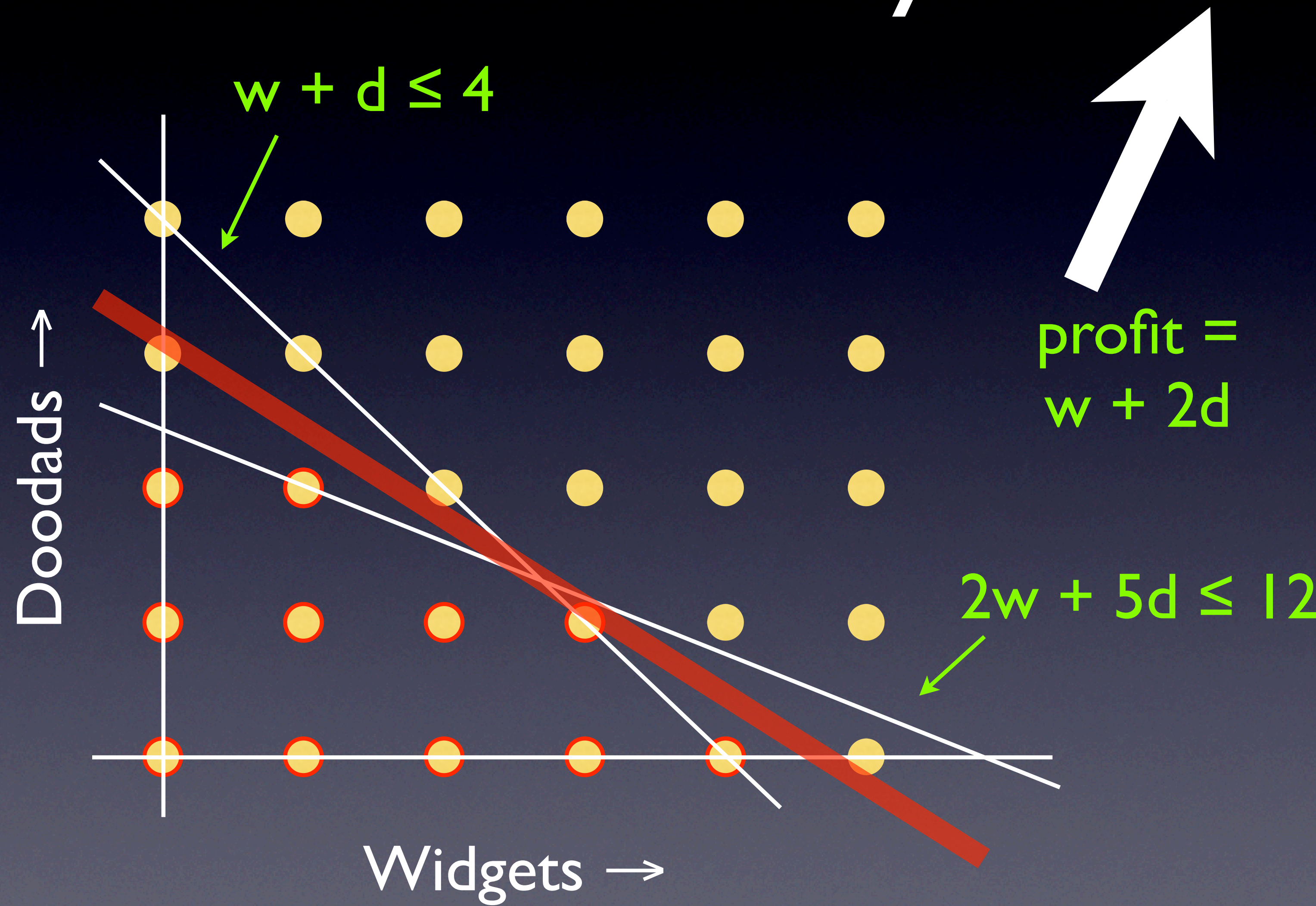
$$a\,(w + d - 4) + b\,(2w + 5d - 12) \le 0$$

$$(a + 2b)\,w + (a + 5b)\,d \le 4a + 12b$$

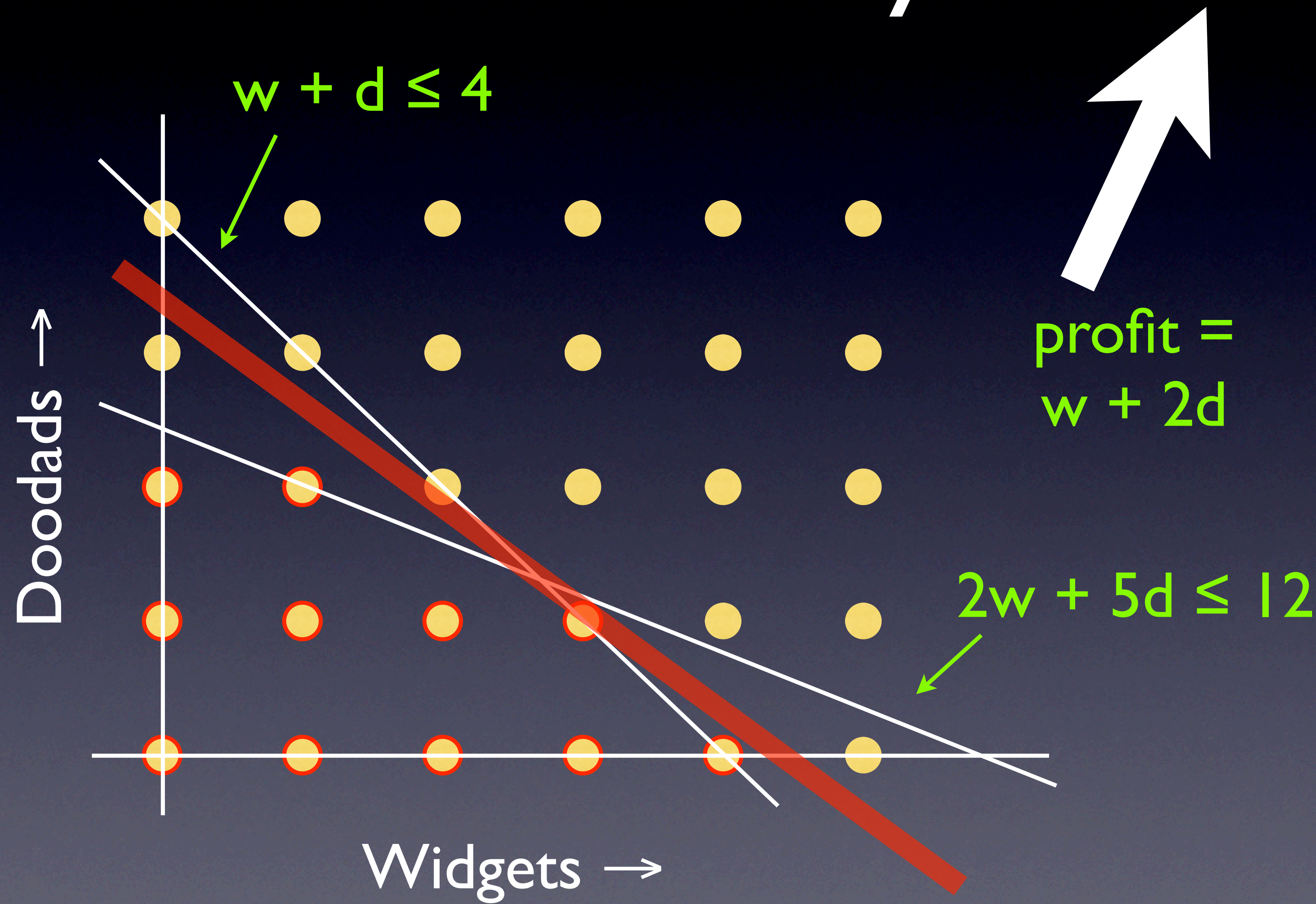
# Geometrically



# Geometrically



# Geometrically



# Bound

- $(a + 2b)\,w + (a + 5b)\,d \leq 4a + 12b$
- profit = $1w + 2d$
- So, if we pick $(a + 2b) \geq 1$ and $(a + 5b) \geq 2$, we will have profit $\leq 4a + 12b$
- Equivalently, could have picked $(a + 2b) \geq 2$ and $(a + 5b) \geq 4$ to bound 2*profit

# The best bound

- If we search for the tightest bound, we have an LP:

  minimize $4a + 12b$ such that

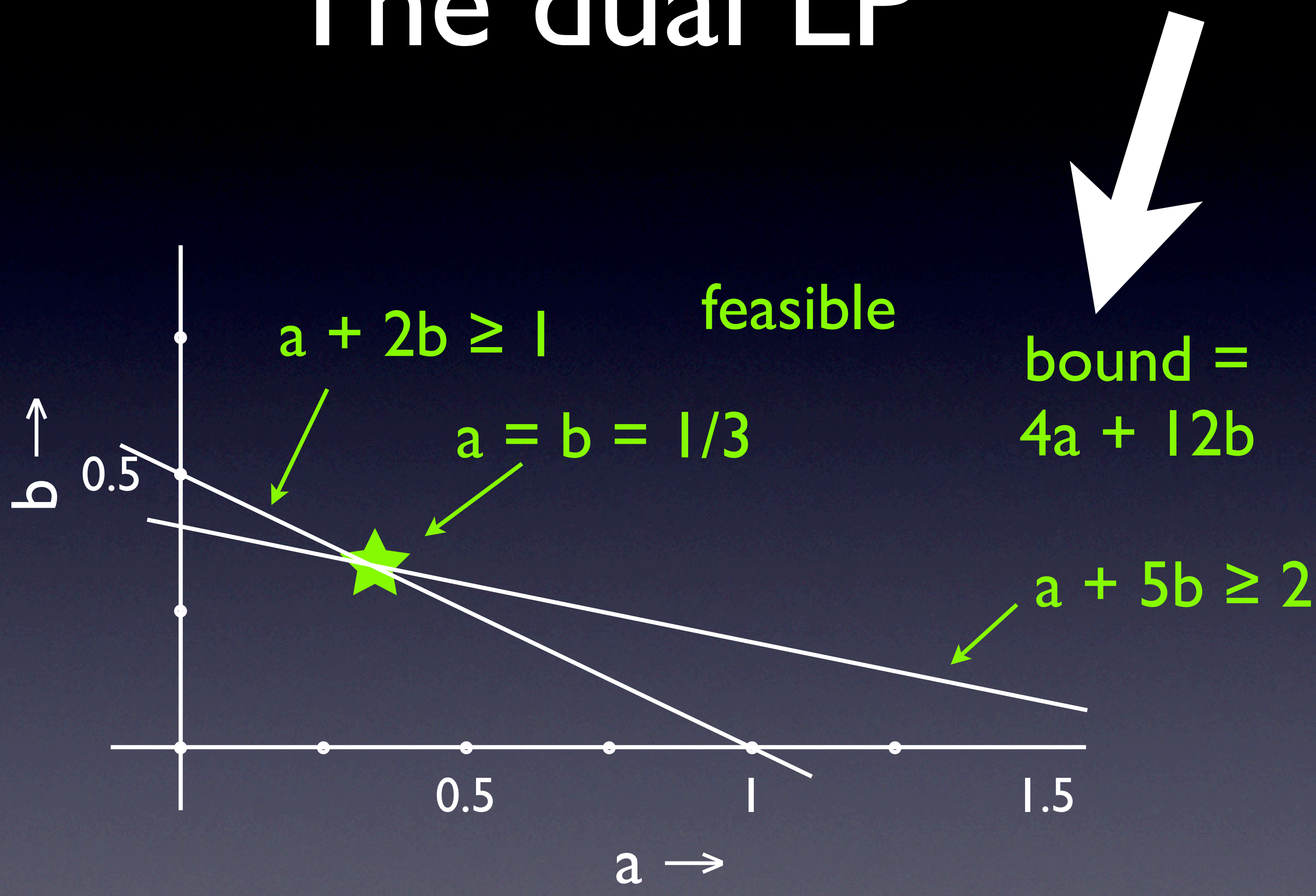
  $a + 2b \geq 1$

  $a + 5b \geq 2$

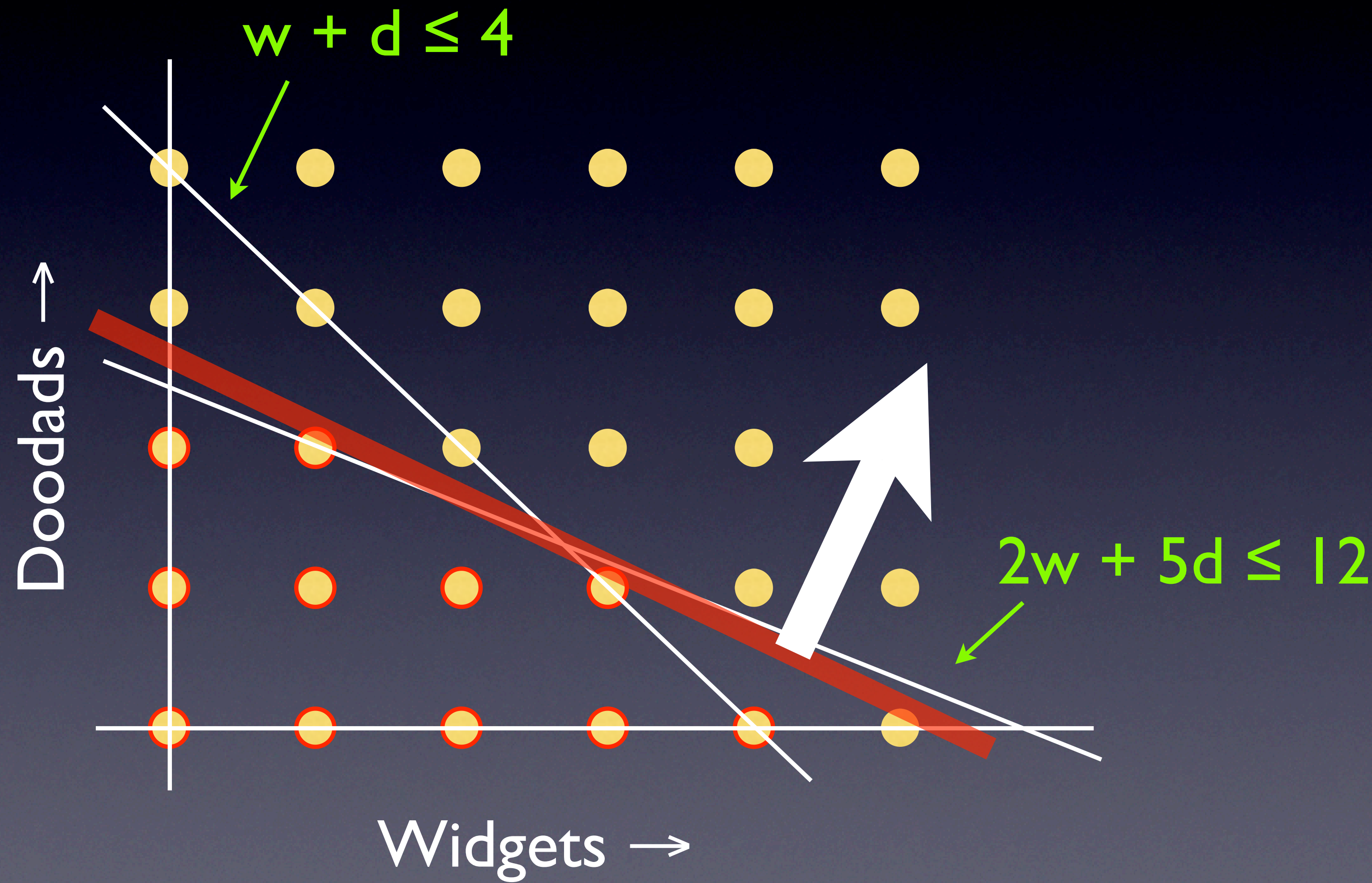
  $a, b \geq 0$

- Called the **dual**

# The dual LP



# Best bound, as constraint



# Bound from dual

- a = b = 1/3 yields bound of 16/3 = 5 1/3
- Same as bound from original relaxation!
- No accident: dual of an LP always* has same objective value
- And dual of dual is original LP (called **primal**)

# So why bother?

- Reason 1: any feasible solution to dual yields upper bound (compared with only optimal solution to primal)
- Reason 2: dual might be easier to work with

# Primal/dual bounds

- Each feasible point of dual is an upper bound on objective
- Each feasible point of primal is a lower bound on objective
  - for ILP, each integral feasible point
- So (answering earlier question) if we have a primal feasible point w/ value 4 and a dual feasible point w/ value 6, we know we're at least 66% of best objective

# More about the dual

# Recipe

- If we have an LP in matrix form,

  maximize c'x subject to

  Ax ≤ b

  x ≥ 0

- Its dual is a similar-looking LP:

  minimize b'y subject to

  A'y ≥ c

  y ≥ 0

Ax ≤ b means every component of Ax is ≤ corresponding component of b

# Recipe with equalities

- If we have an LP with equalities,

  maximize $c'x$ s.t.

  $Ax \leq b$

  $Ex = f$

  $x \geq 0$

- Its dual has some unrestricted variables:

  minimize $b'y + f'z$ s.t.

  $A'y + E'z \geq c$

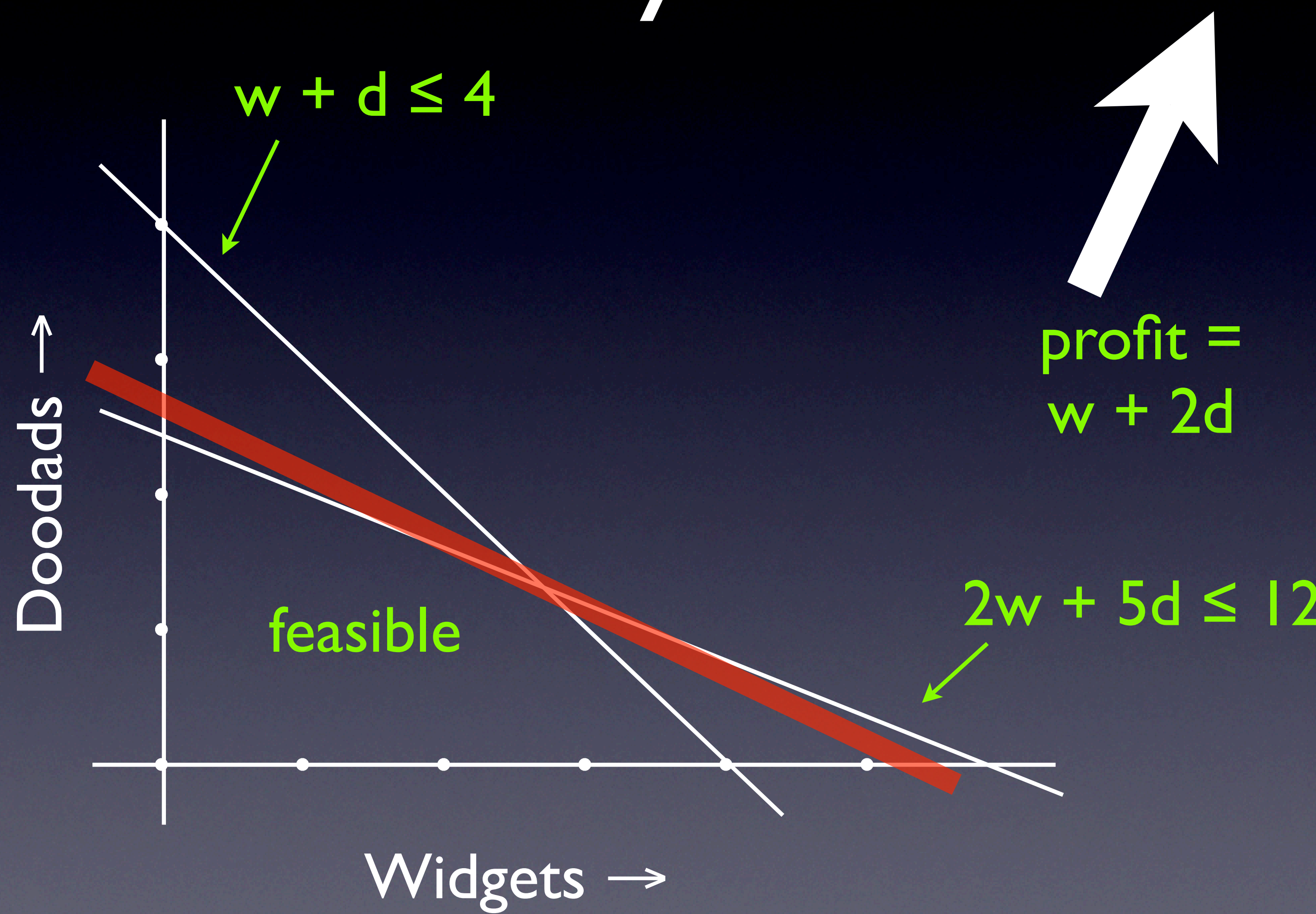
  $y \geq 0$

  $z$ unrestricted

# Interpreting the dual variables

- The primal variable variables in the factory LP were how many widgets and doodads to produce
- We interpreted dual variables as multipliers for primal constraints

# Factory LP



# Dual variables as prices

- "Multiplier" interpretation doesn't give much intuition
- It is often possible to interpret dual variables as **prices** for primal constraints
- Suppose we bought a quantity $\varepsilon$ of wood, loosening constraint to $(w + d \leq 4 + \varepsilon)$
- How much should we be willing to pay for this wood?

# Dual variables as prices

- RHS in primal is objective in dual, so previous solution a = b = 1/3 is still dual feasible
  - still optimal if ε is small enough
- Our bound changes to (4 + ε) a + 12 b, difference of ε * 1/3
- So we should pay up to $1/3 per unit of wood (in small quantities)

# Dual as a game

- Compare the following LP and game
- maximize c'x subject to

  $Ax \le b$

  $x \ge 0$
- $\max_{x\ge0} \min_{y\ge0} c'x + y'(b - Ax)$
- In game, each player picks a nonneg vector, and Y pays X the amount $[c'x + y'(b - Ax)]$

# Dual as a game

- $\max_{x \geq 0} \min_{y \geq 0} c'x + y'(b - Ax)$
- Suppose $(b - Ax)$ has -ve component (say $i^{th}$)
- Then Y will increase $y_i$ arbitrarily, making total payoff very -ve
- X doesn't like this
- So X will obey constraint $(b - Ax \geq 0)$

# Dual as a game

- $\max_{x \geq 0} \min_{y \geq 0} c'x + y'(b - Ax)$
- If X obeys constraint ($b - Ax \geq 0$), what should Y do?
- If $i^{th}$ component +ve, $y_i$ should be 0
- If $i^{th}$ component is 0, $y_i$ is indifferent
- Complementarity: y is 0 where $b - Ax$ is +ve
- Last term cancels, and X will maximize $c'x$

# Dual as a game

- $\max_{x \geq 0} \min_{y \geq 0} (c' - y'A)x + y'b$
- Suppose (c - A'y) has +ve $i^{th}$ component
- Then X will increase $x_i$ arbitrarily, making total payoff very +ve
- Y doesn't like this
- So Y will obey constraint ($c - A'y \leq 0$)

# Dual as a game

- $\max_{x \geq 0} \min_{y \geq 0} (c' - y'A)x + y'b$
- If Y obeys constraint ($c - A'y \leq 0$), what should X do?
- Complementarity again: $x_i$ should be 0 if $i^{th}$ component of ($c - A'y$) is nonzero
- First term cancels, and Y will minimize $y'b$

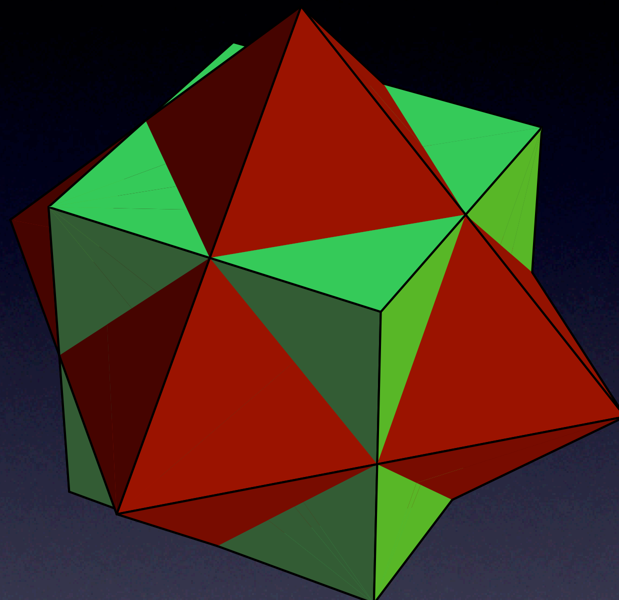
# Dual as a game

- $\max_{x \geq 0} \min_{y \geq 0} c'x + y'(b - Ax)$
- If X obeys constraint ($b - Ax \geq 0$), what should Y do?
- If $i^{th}$ component +ve, $y_i$ should be 0
- If $i^{th}$ component is 0, $y_i$ is indifferent
- Complementarity: y is 0 where b - Ax is +ve
- Last term cancels, and X will maximize c'x

# Yet another way to look at the dual

- Geometric duality:
  - points are dual to lines or halfspaces
  - sets of points are dual to sets of halfspaces = convex polygons
  - a set of points and its convex hull have same dual
- http://www.cs.cmu.edu/~ggordon/SVMs/svm-applet.html

# Geometric duality



- In 3d, point's dual is plane; set of points dualizes to polyhedron
- Escher example: cube's dual is octahedron