15-780: Grad AI
Lec. 8: Linear programs, Duality

Geoff Gordon (this lecture)
Tuomas Sandholm
TAs Erik Zawadzki, Abe Othman

Admin

- Test your handin directories
  - /afs/cs/user/aothman/dropbox/USERID/
  - where USERID is your Andrew ID
- Poster session:
  - Mon 5/2, 1:30–4:30PM, room TBA
- Readings for today & Tuesday on class site

Project idea

- Answer the question: *what is fairness?*

In case anyone thinks of slacking off

LPs, ILPs, and their ilk

Boyd & Vandenberghe. Convex Optimization. Sec 4.3 and 4.3.1.

((M)I)LPs

- Linear program:

  min $3x + 2y$ s.t.

  $x + 2y \leq 3$

  $x \leq 2$

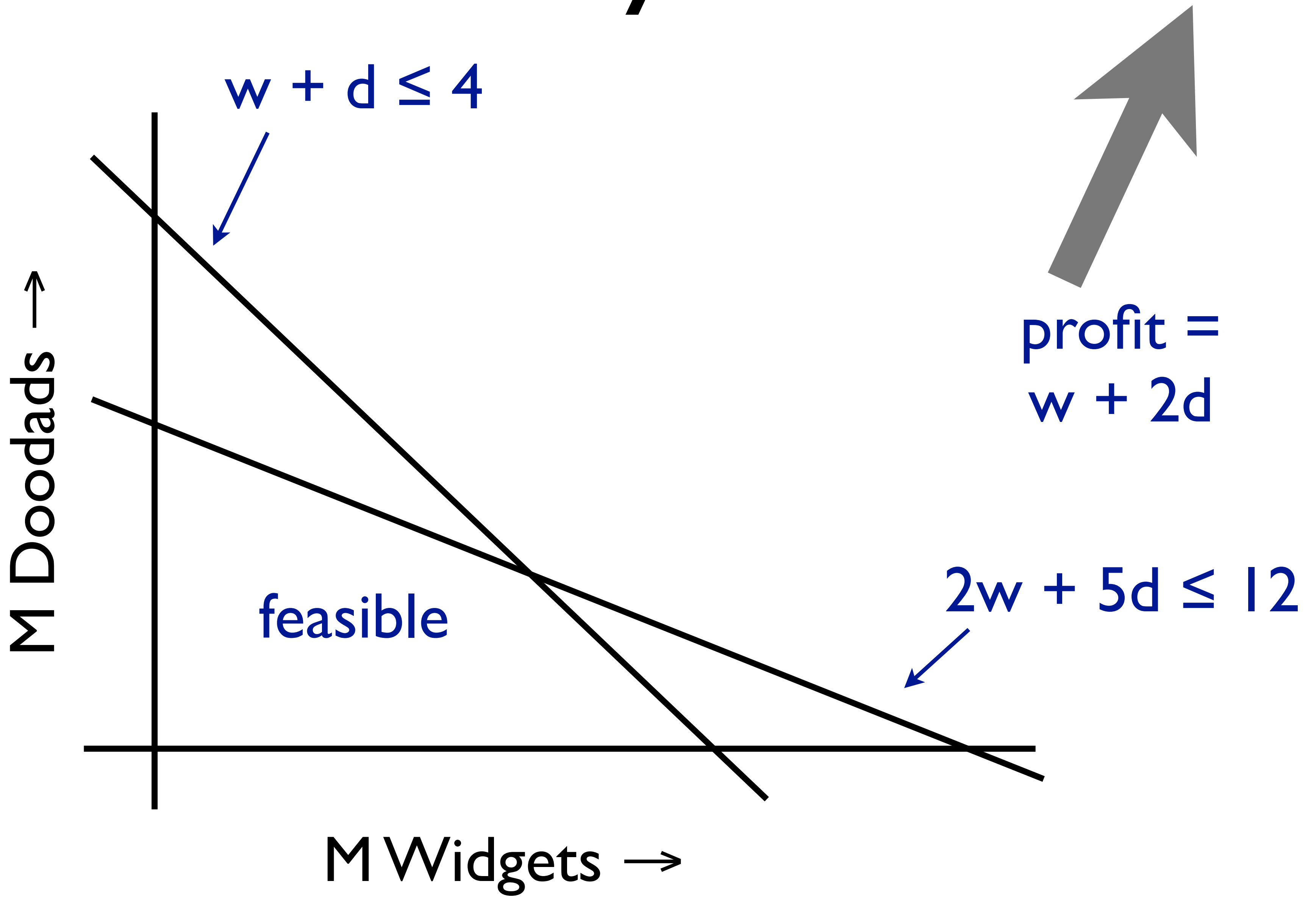
  $x, y \geq 0$

- Integer linear program: constrain $x, y \in \mathbb{Z}$
- Mixed ILP: $x \in \mathbb{Z}$, $y \in \mathbb{R}$

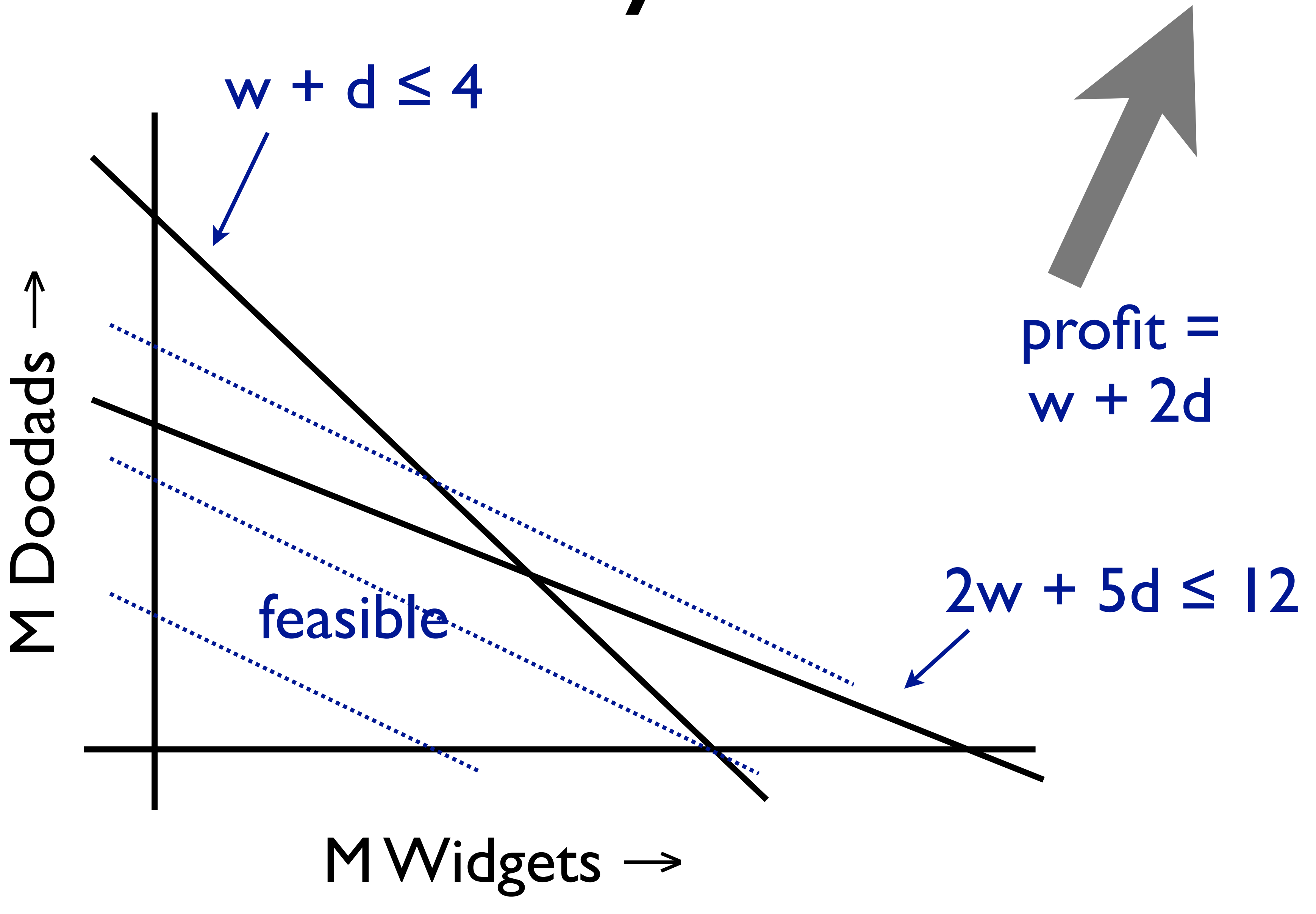
Example LP

- Factory makes widgets and doodads
- Each widget takes 1 unit of wood and 2 units of steel to make
- Each doodad uses 1 unit wood, 5 of steel
- Have 4M units wood and 12M units steel
- Maximize profit: each widget nets $1, each doodad nets $2

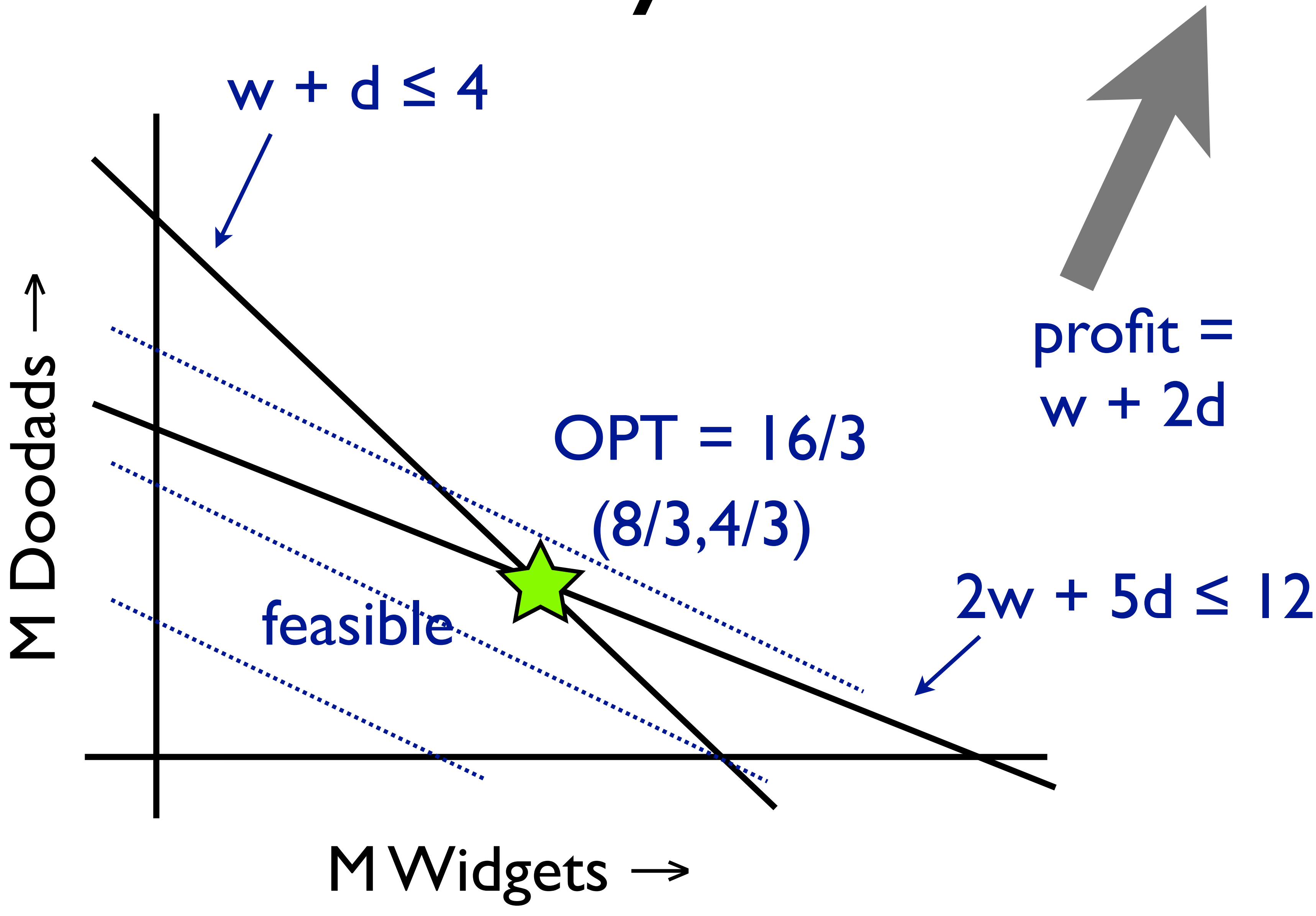
Factory LP



Factory LP



Factory LP

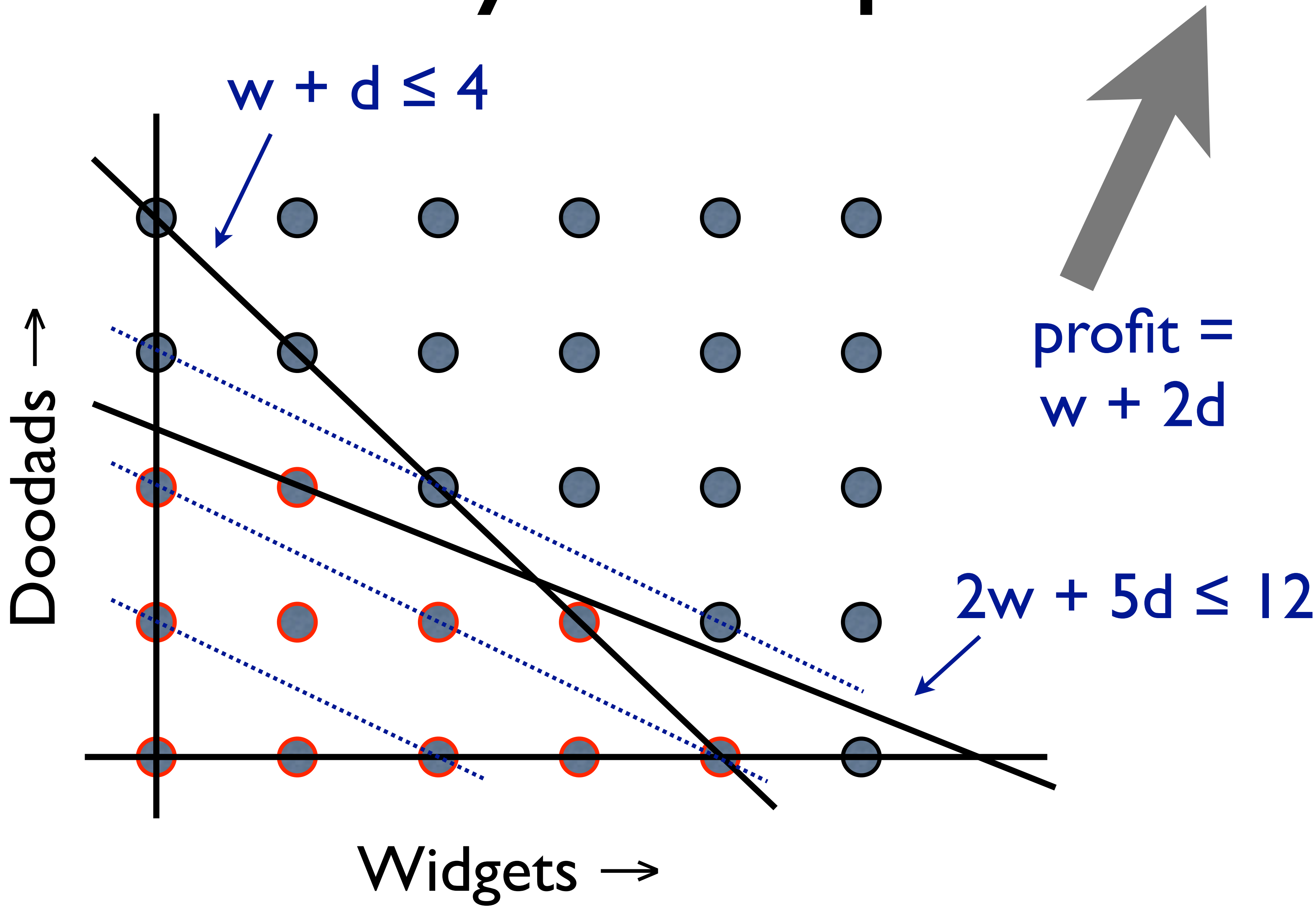


w + d ≤ 4
profit =
w + 2d
M Doodads →
OPT = 16/3
(8/3,4/3)
2w + 5d ≤ 12
feasible
M Widgets →

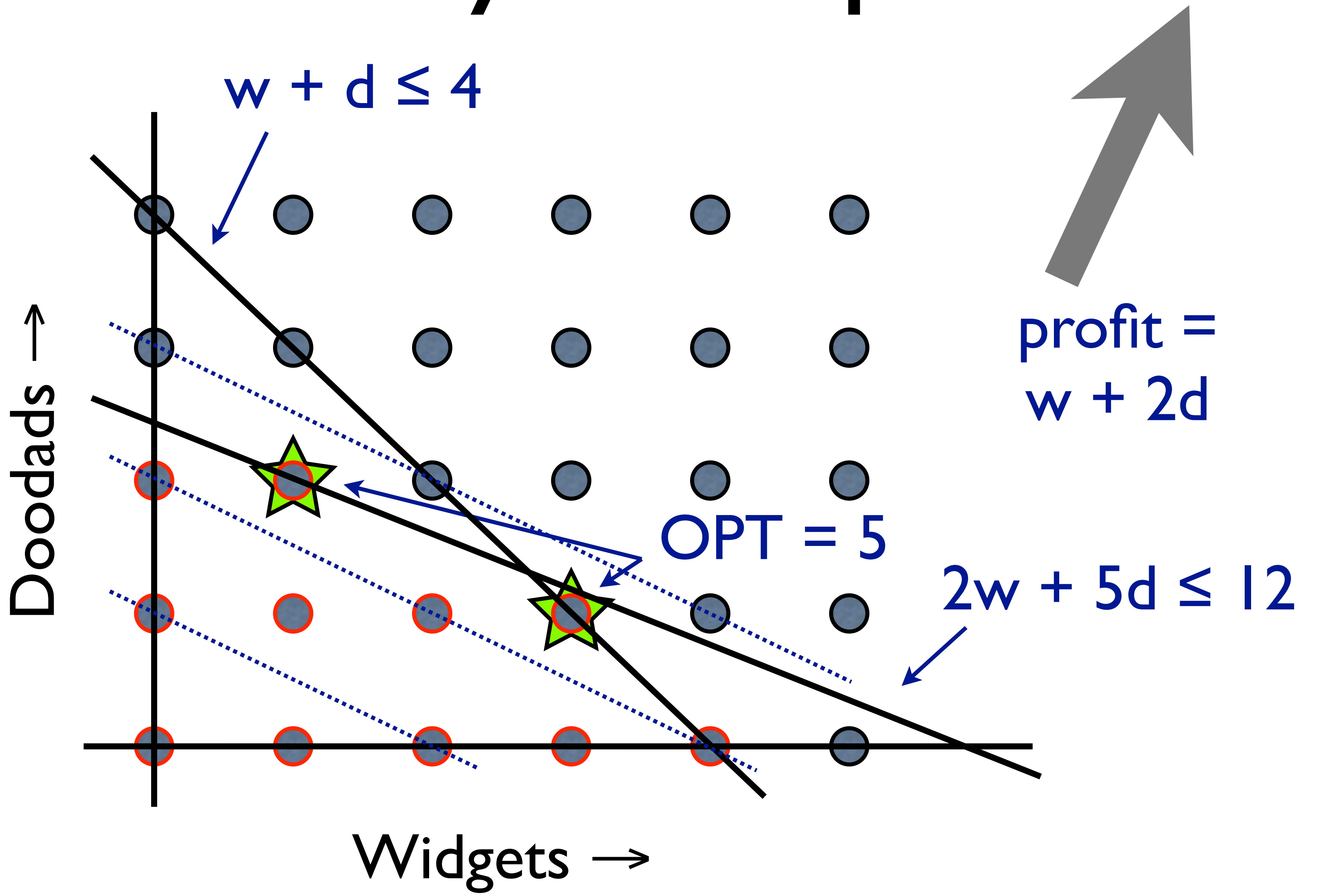
Example ILP

- Instead of 4M units of wood, 12M units of steel, have 4 units wood and 12 units steel

Factory example



Factory example

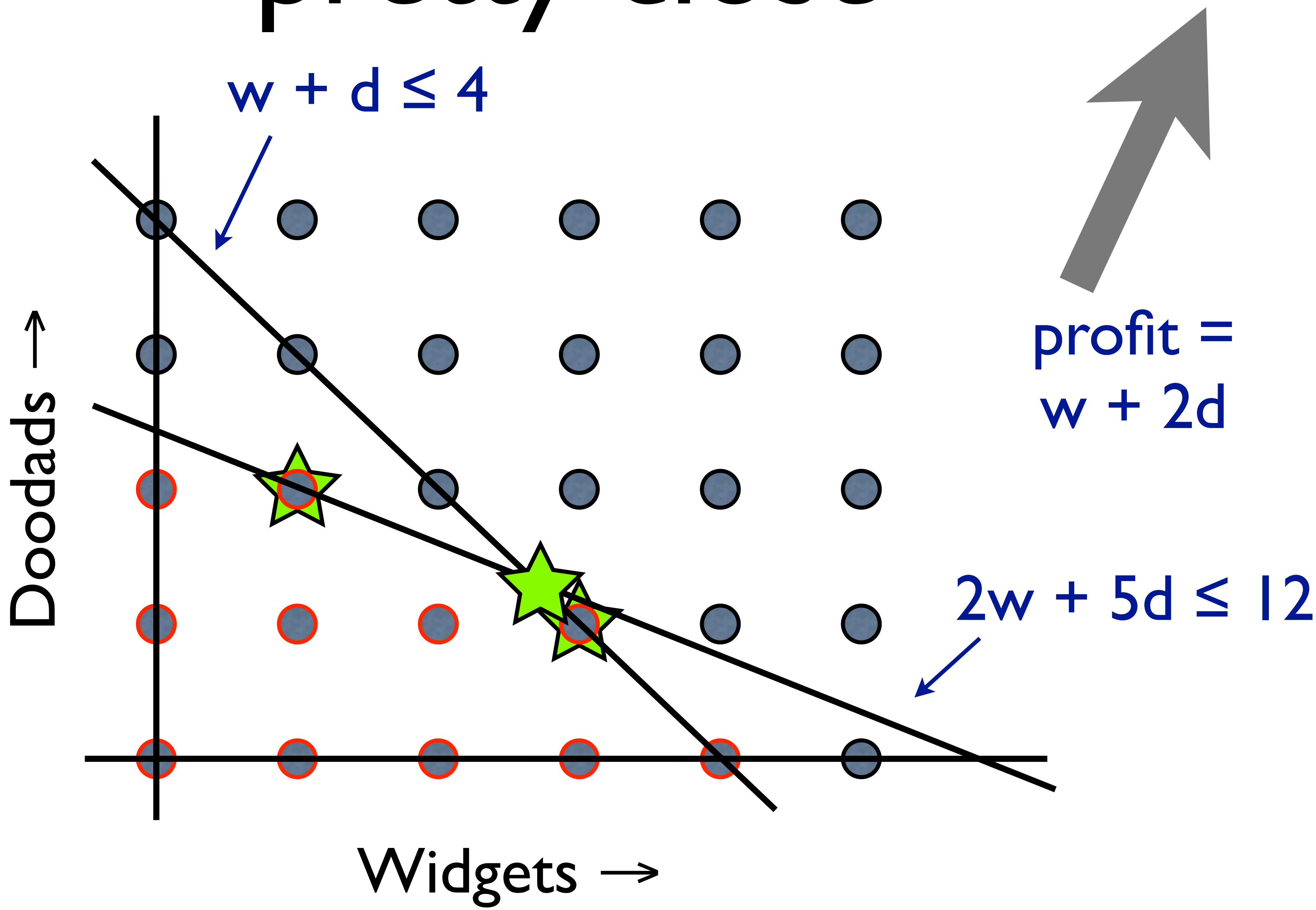


LP relaxations

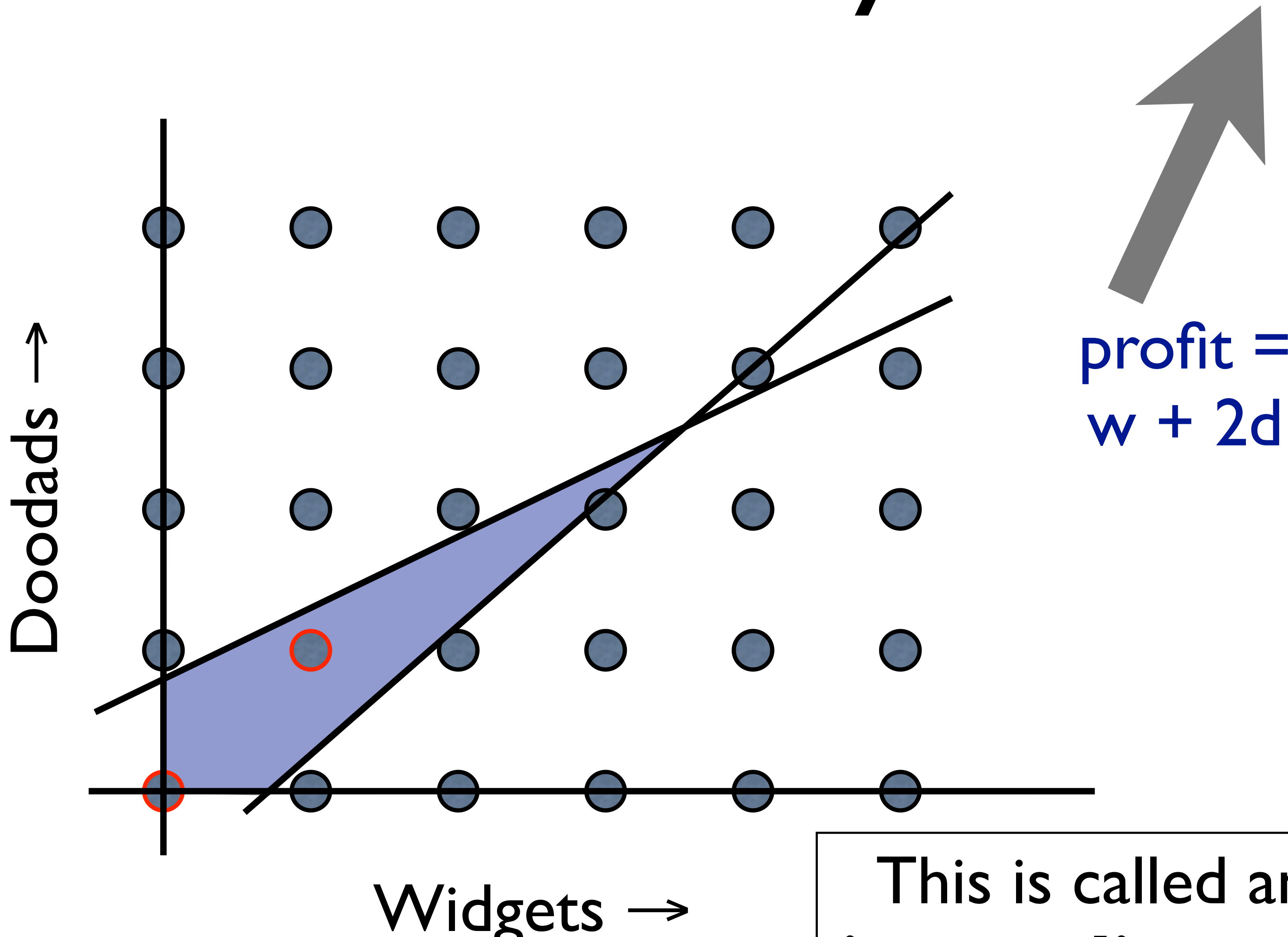
- Above LP and ILP are the same, except for constraint w, d $\in \mathbb{Z}$
- LP is a **relaxation** of ILP
- Adding any constraint makes optimal value **same or worse**
- So, OPT(relaxed) ≥ OPT(original)

  (in a maximization problem)

Factory relaxation is pretty close



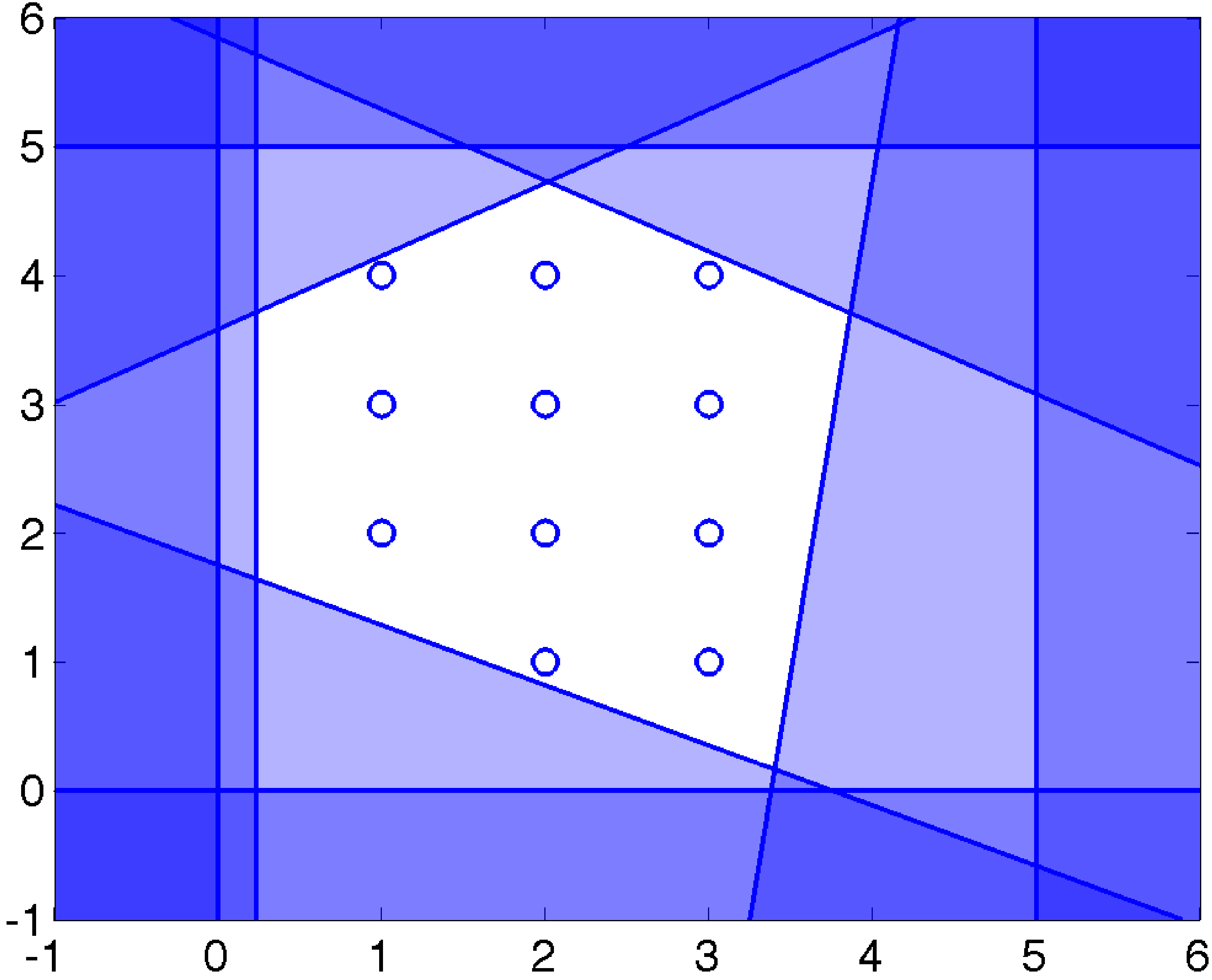
Unfortunately…

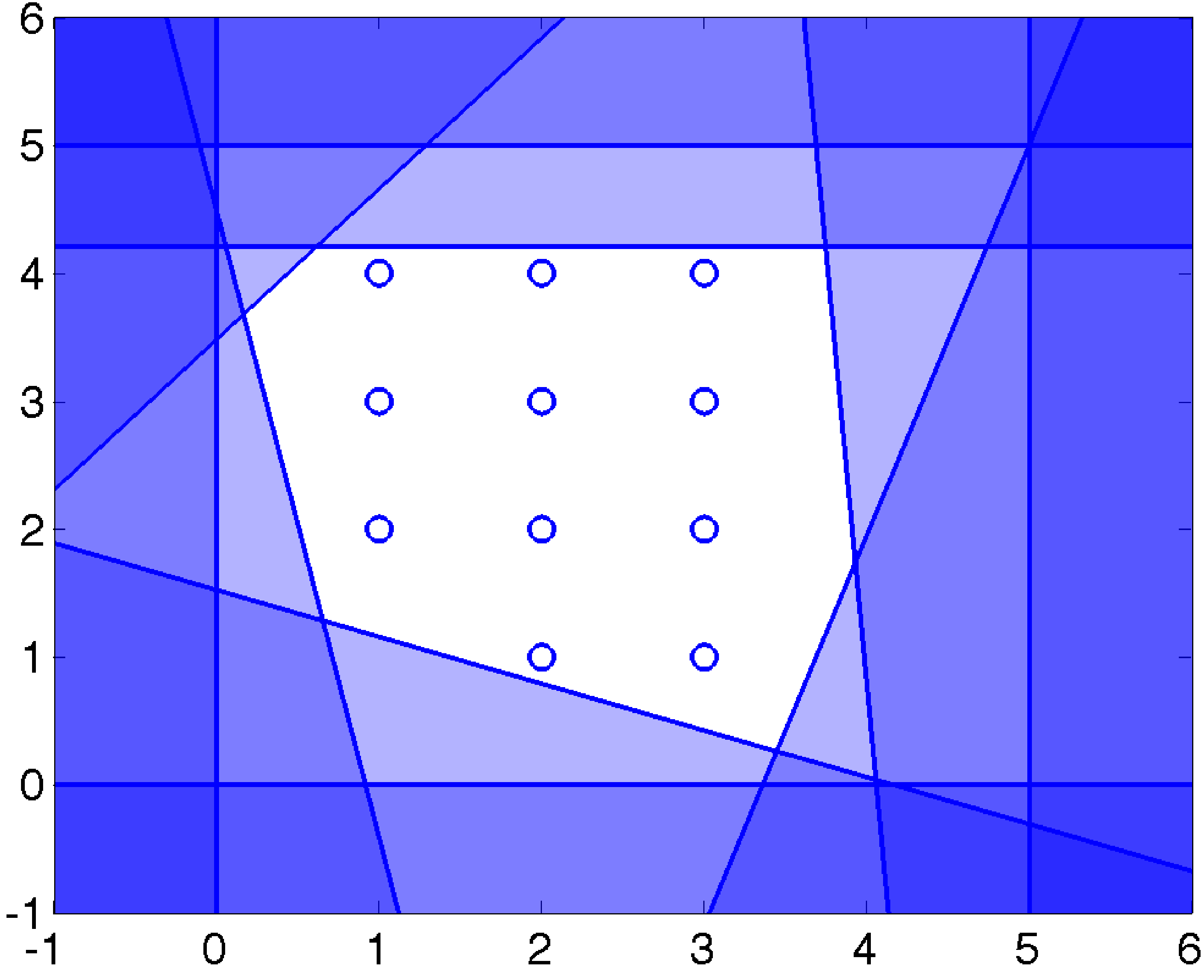


This is called an **integrality gap**

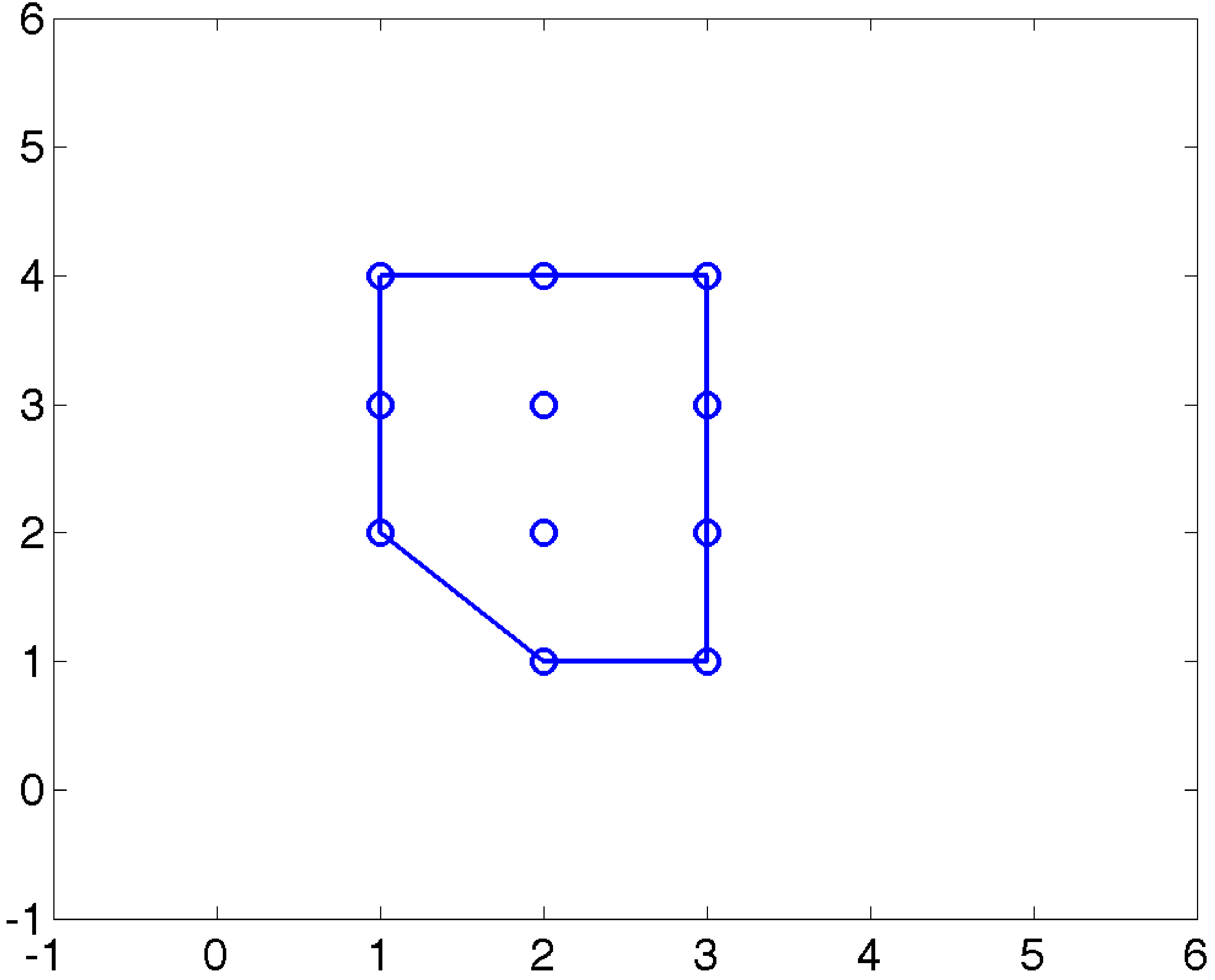
Falling into the gap

- In this example, gap is 3 vs 8.5, or about a ratio of 0.35
- Ratio can be arbitrarily bad
  - but, can sometimes bound it for classes of ILPs
- Gap can be different for different LP relaxations of "same" ILP


6
5
4
3
2
1
0
-1
-1
0
1
2
3
4
5
6



6
5
4
3
2
1
0
-1
-1
0
1
2
3
4
5
6



6
5
4
3
2
1
0
-1
-1 0 1 2 3 4 5 6

From ILP to SAT

- 0-1 ILP: all variables in {0, 1}
- SAT: 0-1 ILP, objective = constant, all constraints of form

  $x + (1-y) + (1-z) \geq 1$

- MAXSAT: 0-1 ILP, constraints of form

  $x + (1-y) + (1-z) \geq s_j$

  maximize $s_1 + s_2 + \ldots$

Pseudo-boolean inequalities

- Any inequality with integer coefficients on 0-1 variables is a PBI
- Collection of such inequalities (w/o objective): pseudo-boolean SAT
- Many SAT techniques work well on PB-SAT as well

Complexity

- Decision versions of ILPs and MILPs are NP-complete (e.g., ILP feasibility contains SAT)
  - so, no poly-time algos unless P=NP
  - in fact, no poly-time algo can approximate OPT to within a constant factor unless P=NP
- Typically solved by search + smart techniques for ordering & pruning nodes
- E.g., branch & cut (in a few lectures)—like DPLL (DFS) but with more tricks for pruning

Complexity

- There are poly-time algorithms for LPs
  - e.g., ellipsoid, log-barrier methods
  - rough estimate: n vars, m constraints ⇒ ~50–200 × cost of (n × m) regression
- No **strongly polynomial** LP algorithms known—interesting open question
  - simplex is "almost always" polynomial [Spielman & Teng]

Terminology

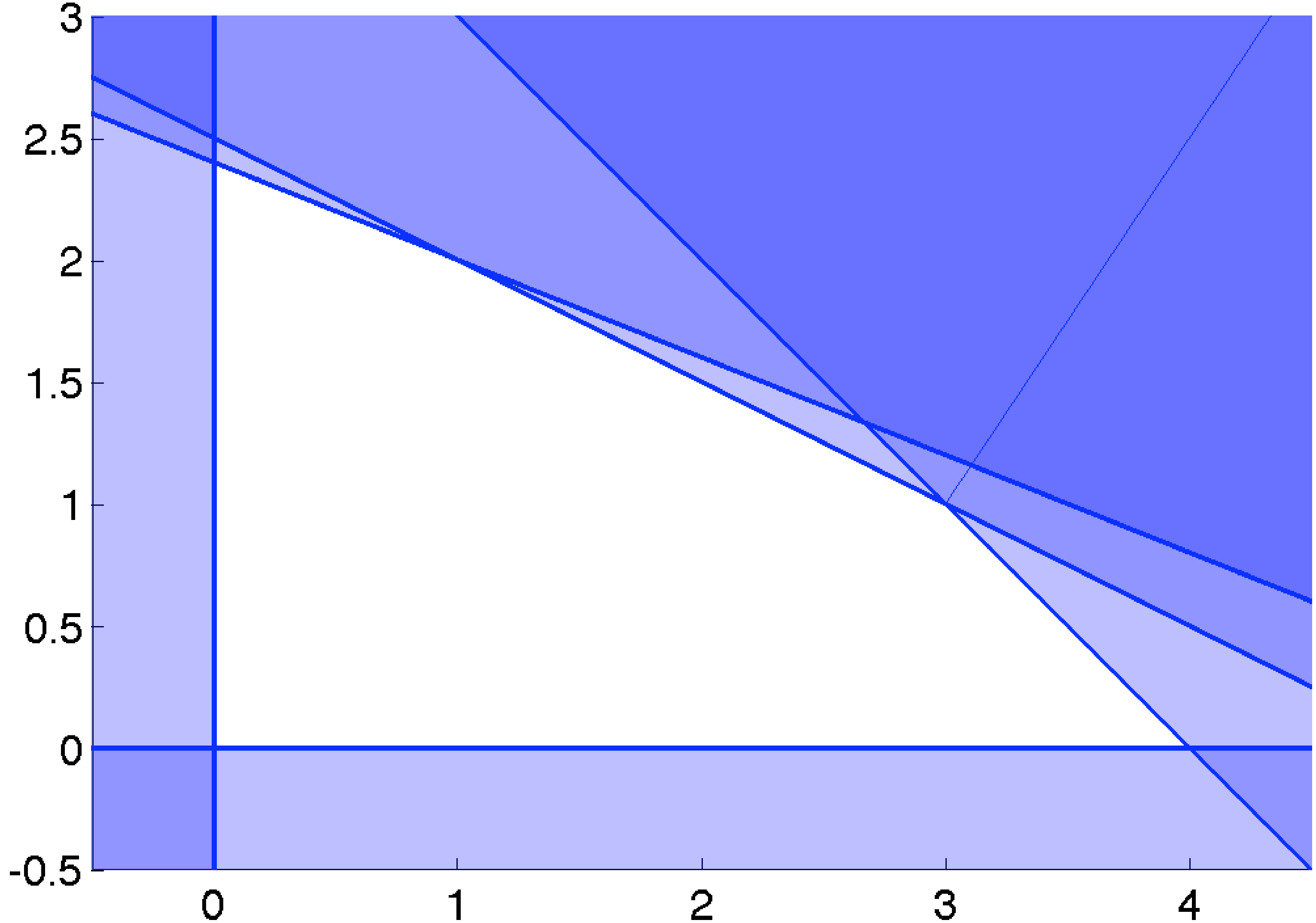
max 2x+3y s.t.

$x + y \leq 4$

$2x + 5y \leq 12$

$x + 2y \leq 5$

$x, y \geq 0$



Finding the optimum

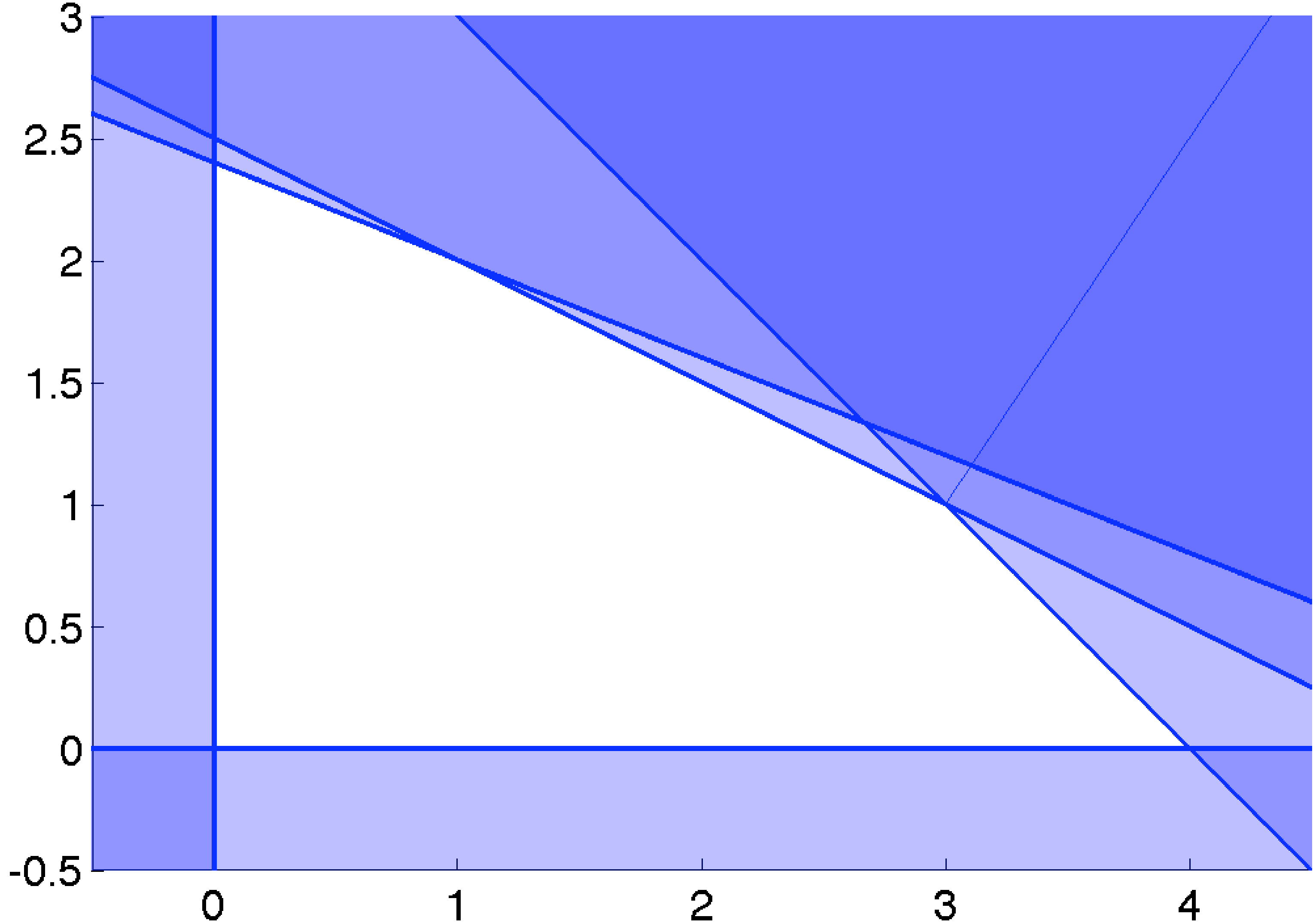
max 2x+3y s.t.

$x + y \leq 4$

$2x + 5y \leq 12$

$x + 2y \leq 5$

$x, y \geq 0$



Finding the optimum

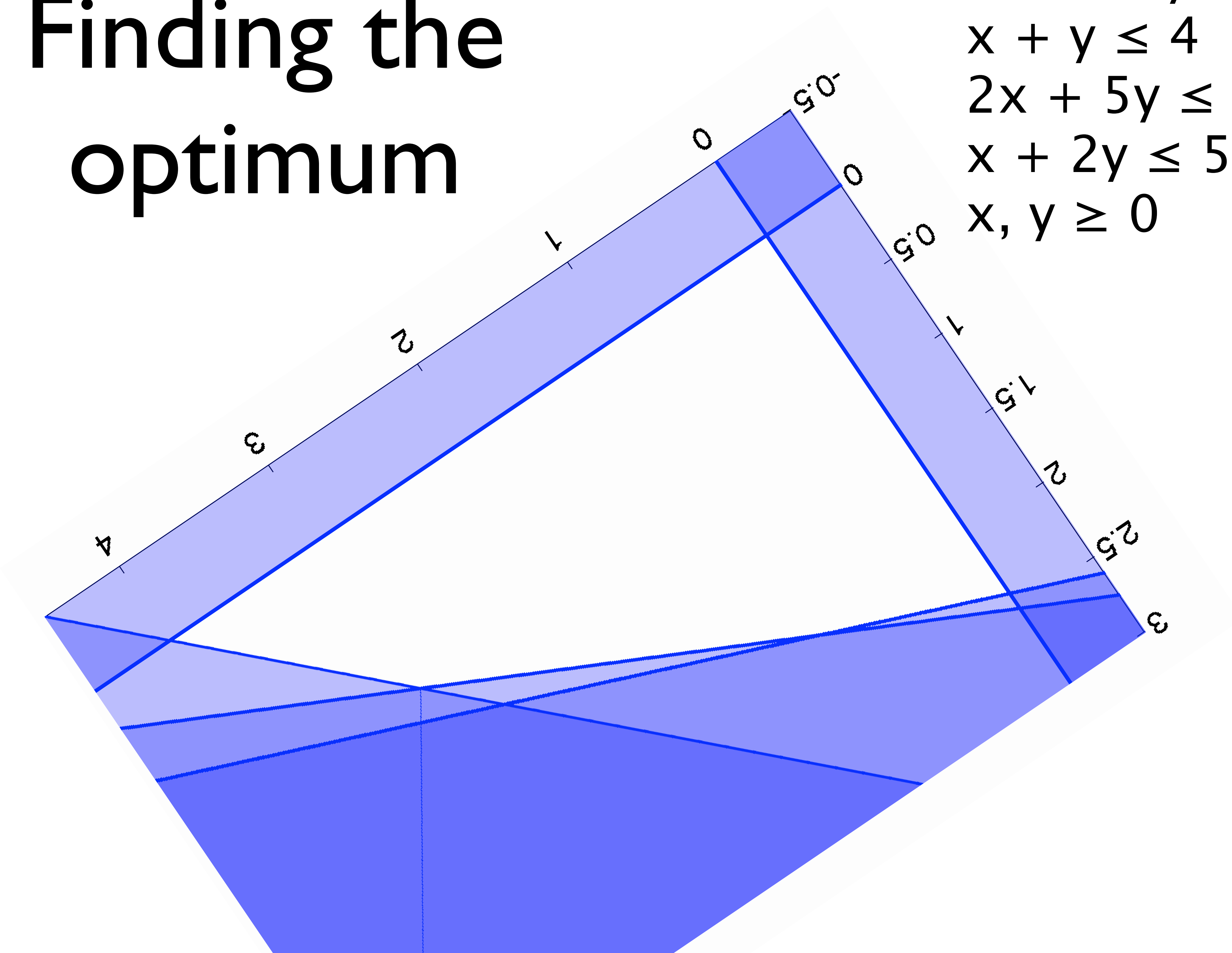
max $2x+3y$ s.t.

$x + y \leq 4$

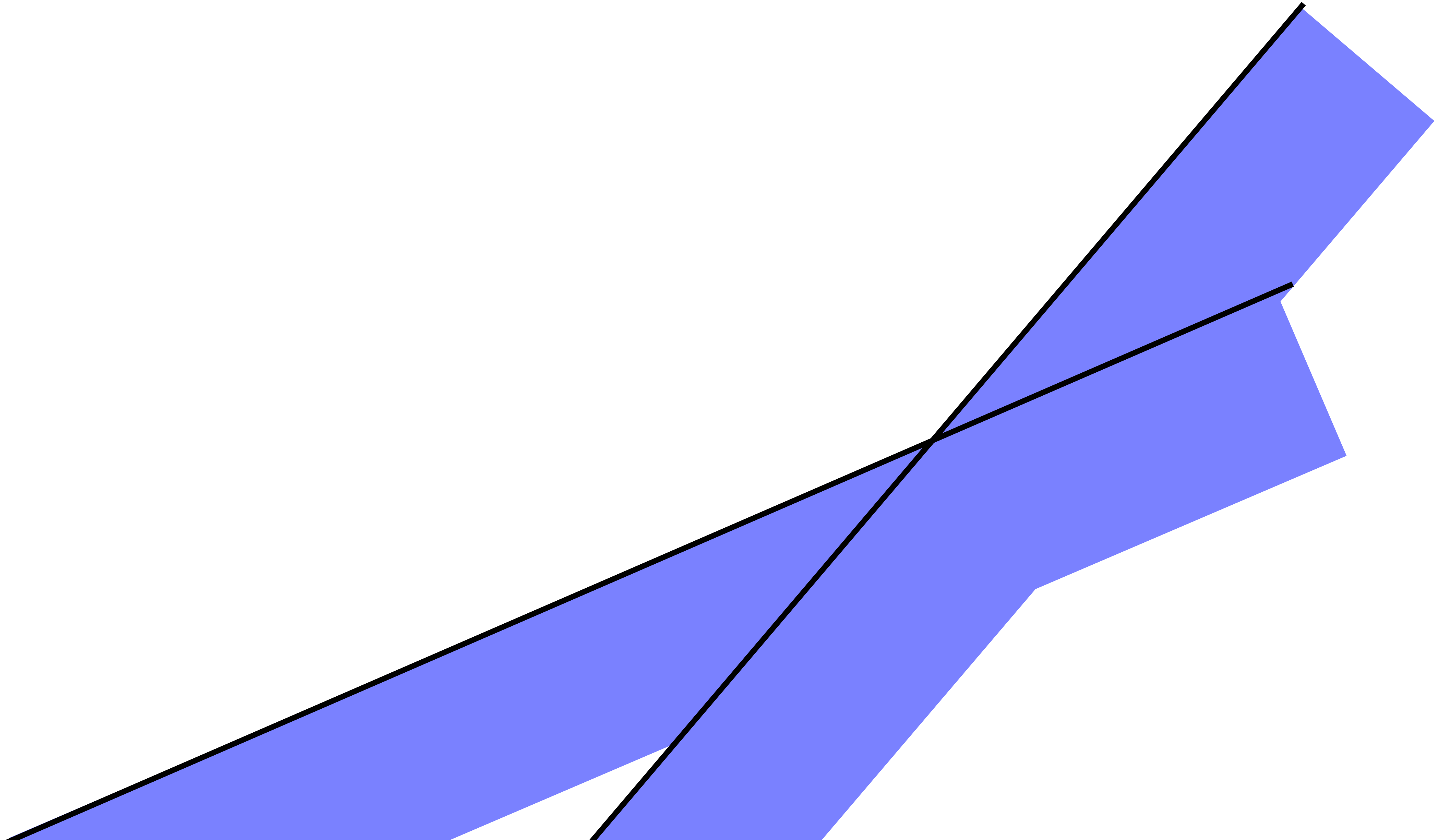
$2x + 5y \leq 12$

$x + 2y \leq 5$

$x, y \geq 0$



Where's my ball?



Unhappy ball

- min $2x + 3y$ subject to
- $x \geq 5$
- $x \leq 1$

Transforming LPs

- Getting rid of inequalities (except variable bounds)

- Getting rid of unbounded variables

Standard form LP

$\max\ 2x+3y$ s.t.
$x + y \leq 4$
$2x + 5y \leq 12$
$x + 2y \leq 5$
$x, y \geq 0$

- all variables are nonnegative
- all constraints are equalities
- E.g.: $q = (x\ y\ u\ v\ w)^T$

$\max\ c^T q$ s.t.
$Aq = b,\ q \geq 0$
(componentwise)

Why is standard form useful?

- Easy to find corners
- Easy to manipulate via row operations
- Basis of simplex algorithm

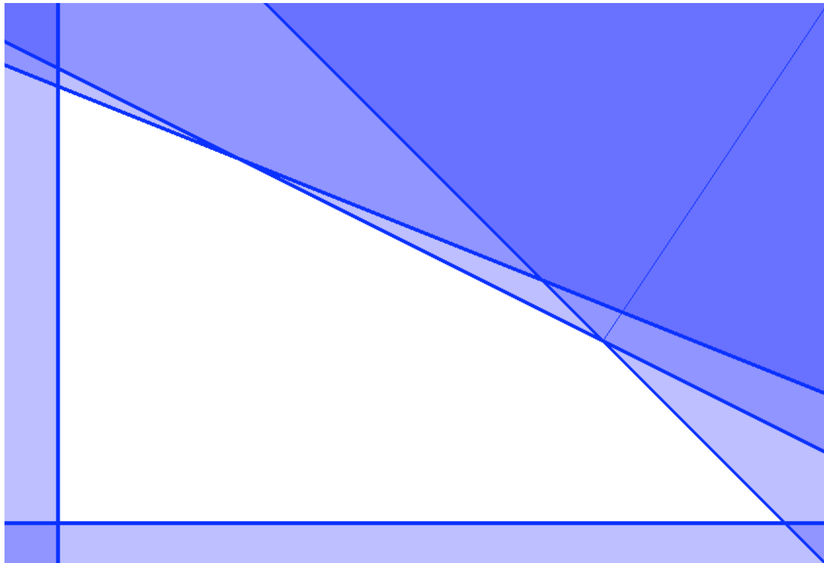
Bertsimas and Tsitsiklis. Introduction to Linear Optimization. Ch. 2–3.

Finding corners

x	y	u	v	w	RHS	
1	1	1	0	0	4	set x, y = 0
2	5	0	1	0	12	
1	2	0	0	1	5	

1	1	1	0	0	4	set v, w = 0
2	5	0	1	0	12	
1	2	0	0	1	5	

1	1	1	0	0	4	set x, u = 0
2	5	0	1	0	12	
1	2	0	0	1	5	



Row operations

- Can replace any row with linear combination of existing rows
  - as long as we don't lose independence
- Elim. x from 2nd and 3rd rows of A
- And from c:

x	y	u	v	w	RHS
1	1	1	0	0	4
2	5	0	1	0	12
1	2	0	0	1	5
2	3	0	0	0	↑

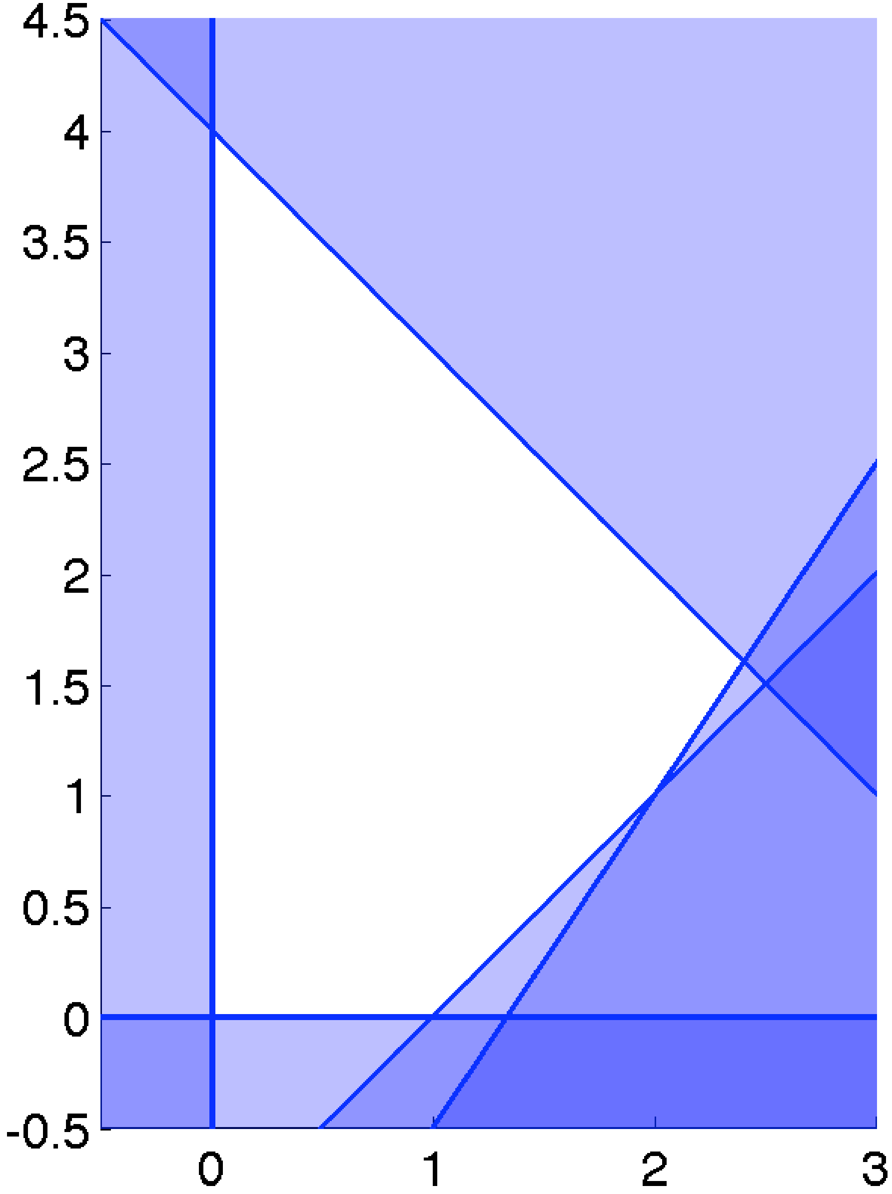
Presto change-o

- Which are the slacks now?
  -
  - vars that appear in

x	y	u	v	w	RHS
1	1	1	0	0	4
0	3	−2	1	0	4
0	1	−1	0	1	1
0	1	−2	0	0	↑

- Terminology: “slack-like” variables are called ***basic***

The "new" LP



max $y - 2u$
$y + u \leq 4$
$3y - 2u \leq 4$
$y - u \leq 1$
$y, u \geq 0$

x	y	u	v	w	RHS
1	1	1	0	0	4
0	3	−2	1	0	4
0	1	−1	0	1	1
0	1	−2	0	0	↑

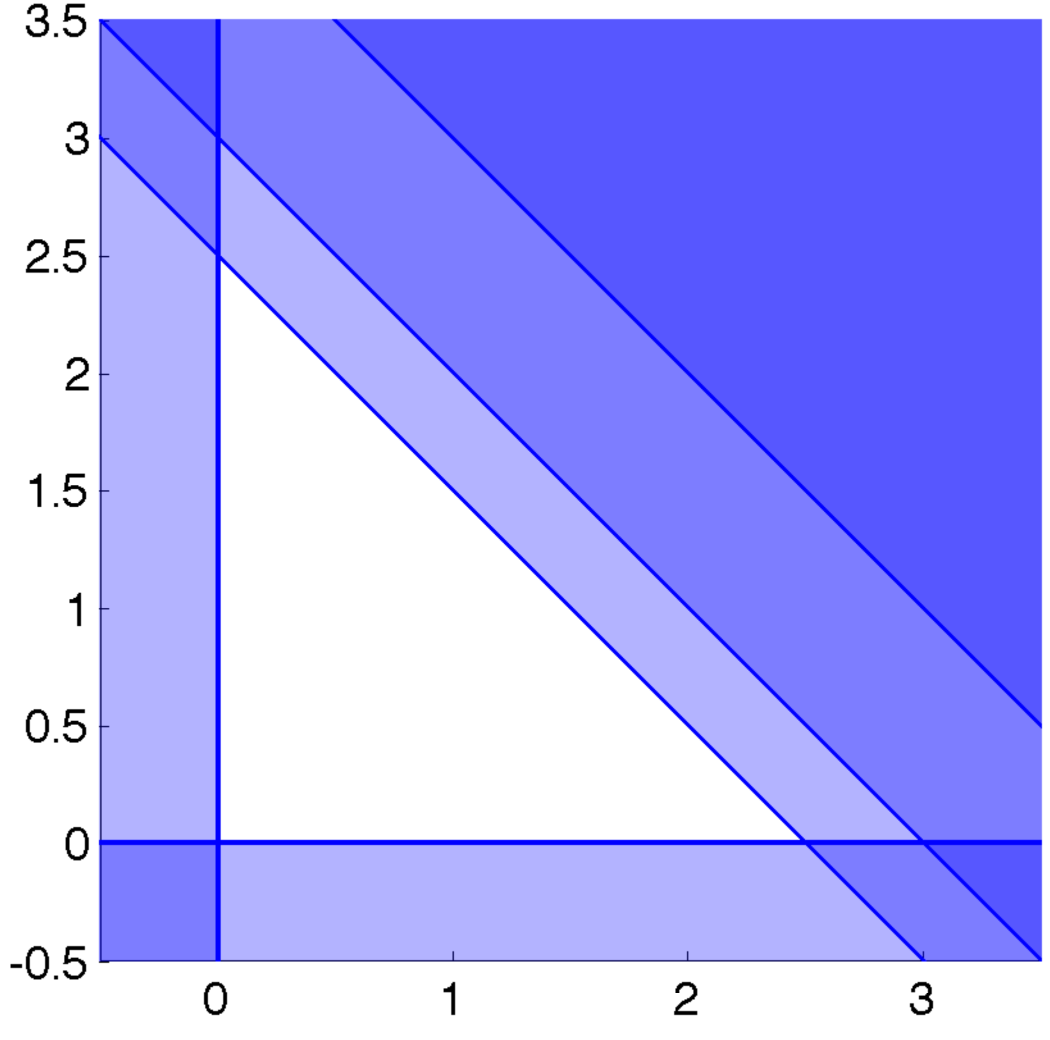
Many different-looking but equivalent LPs, depending on which variables we choose to make into slacks

Or, many corners of same LP

Basis

- Which variables can we choose to make basic?

x	y	u	v	w	RHS
1	1	1	0	0	4
2	2	0	1	0	5
3	3	0	0	1	9
2	1	0	0	0	↑



Nonsingular

- We can assume
  - $n \geq m$ (at least as many vars as constrs)
  - A has full row rank
- Else, drop rows (w/o reducing rank) until true: dropped rows are either redundant or impossible to satisfy
  - easy to distinguish: pick a corner of reduced LP, check dropped = constraints
- Called ***nonsingular*** standard form LP
  - means basis is an invertible $m \times m$ submatrix

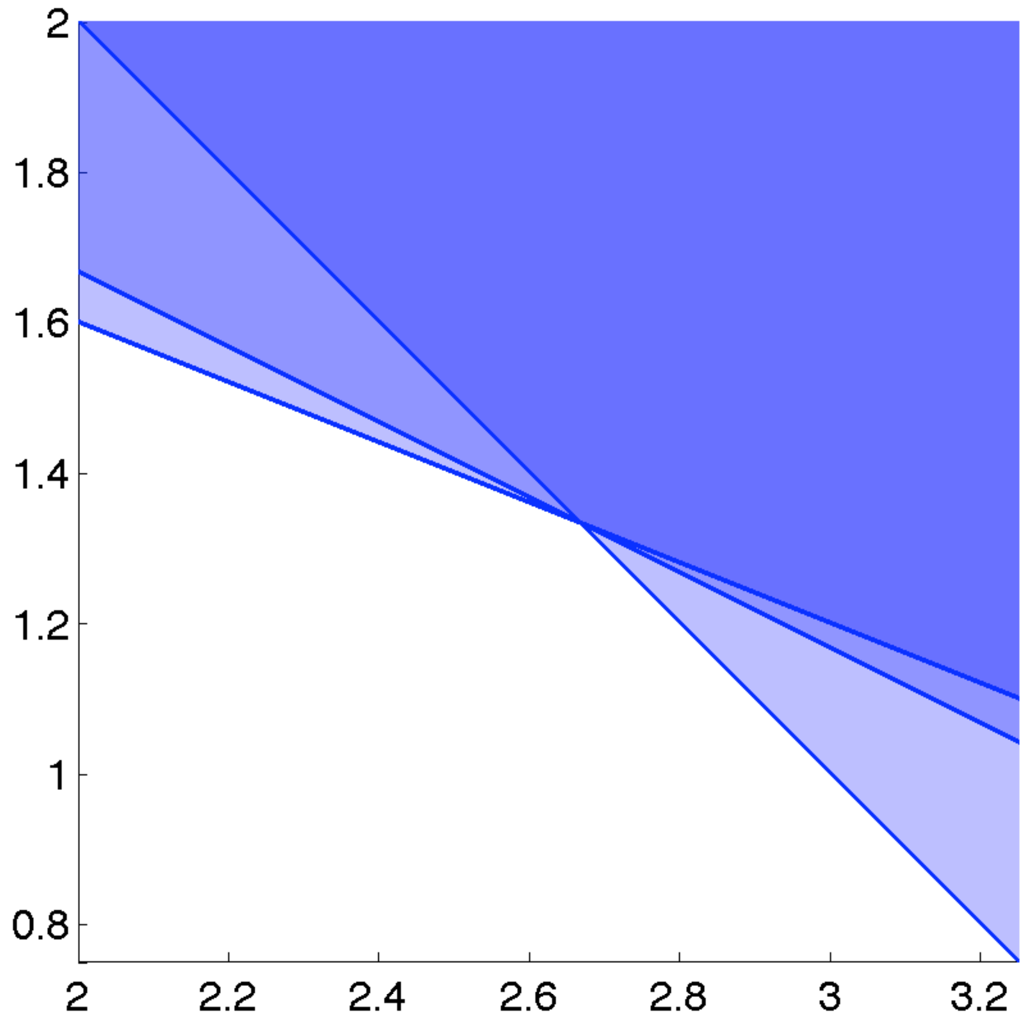
Naïve (slooow) algorithm

- Iterate through all subsets B of m vars
  - if m constraints, n vars, how many subsets?
- Check each B for
  - full rank ("basis-ness")
  - feasibility (A(:,B) \ RHS ≥ 0)
- If pass both tests, compute objective
- Maintain running winner, return at end

Degeneracy

- Not every set of m variables yields a corner
  - some have rank < m (not a basis)
  - some are infeasible
- Can the reverse be true? Can two bases yield the same corner? (Assume nonsingular standard-form LP.)

Degeneracy

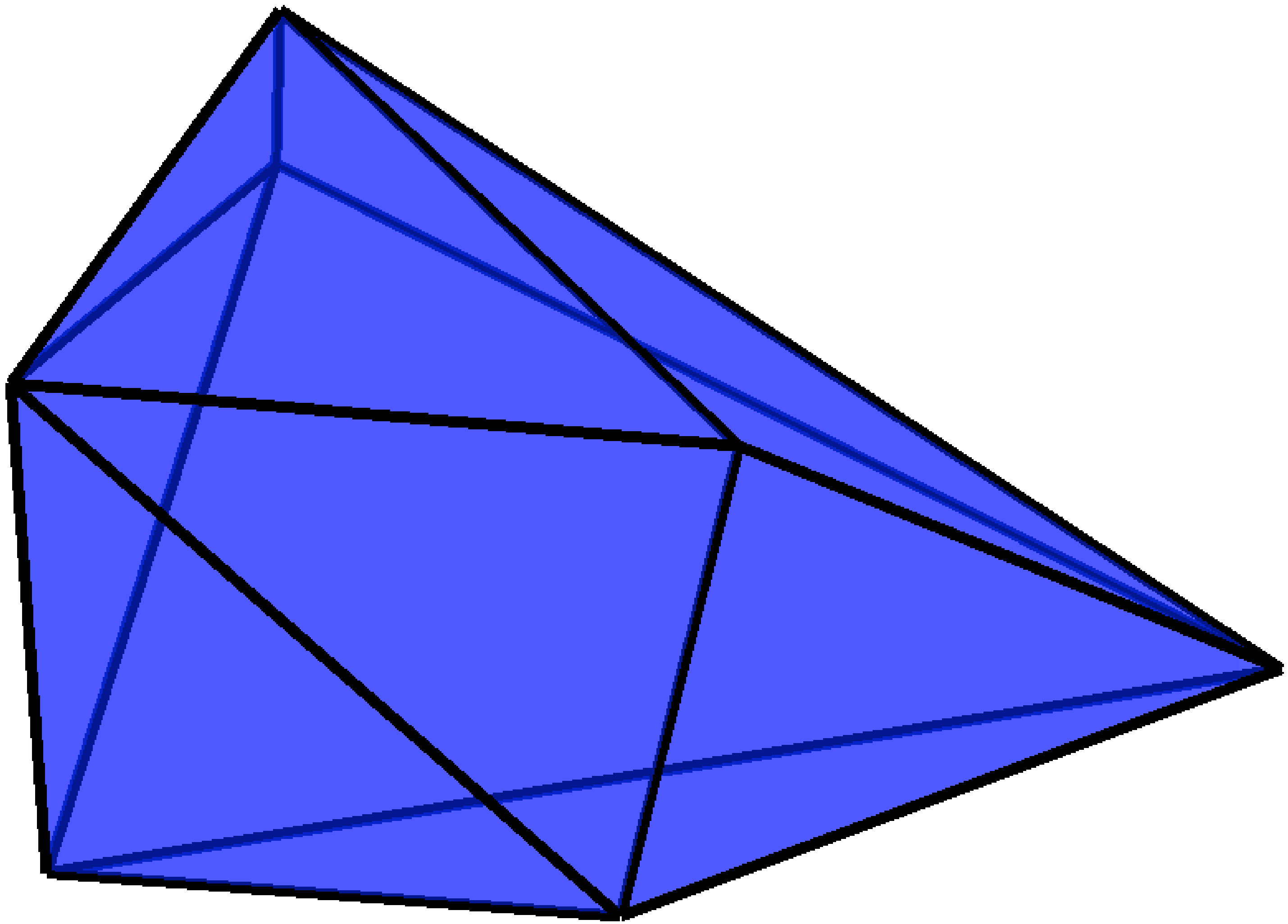


x	y	u	v	w	RHS
1	1	1	0	0	4
2	5	0	1	0	12
1	2	0	0	1	16/3

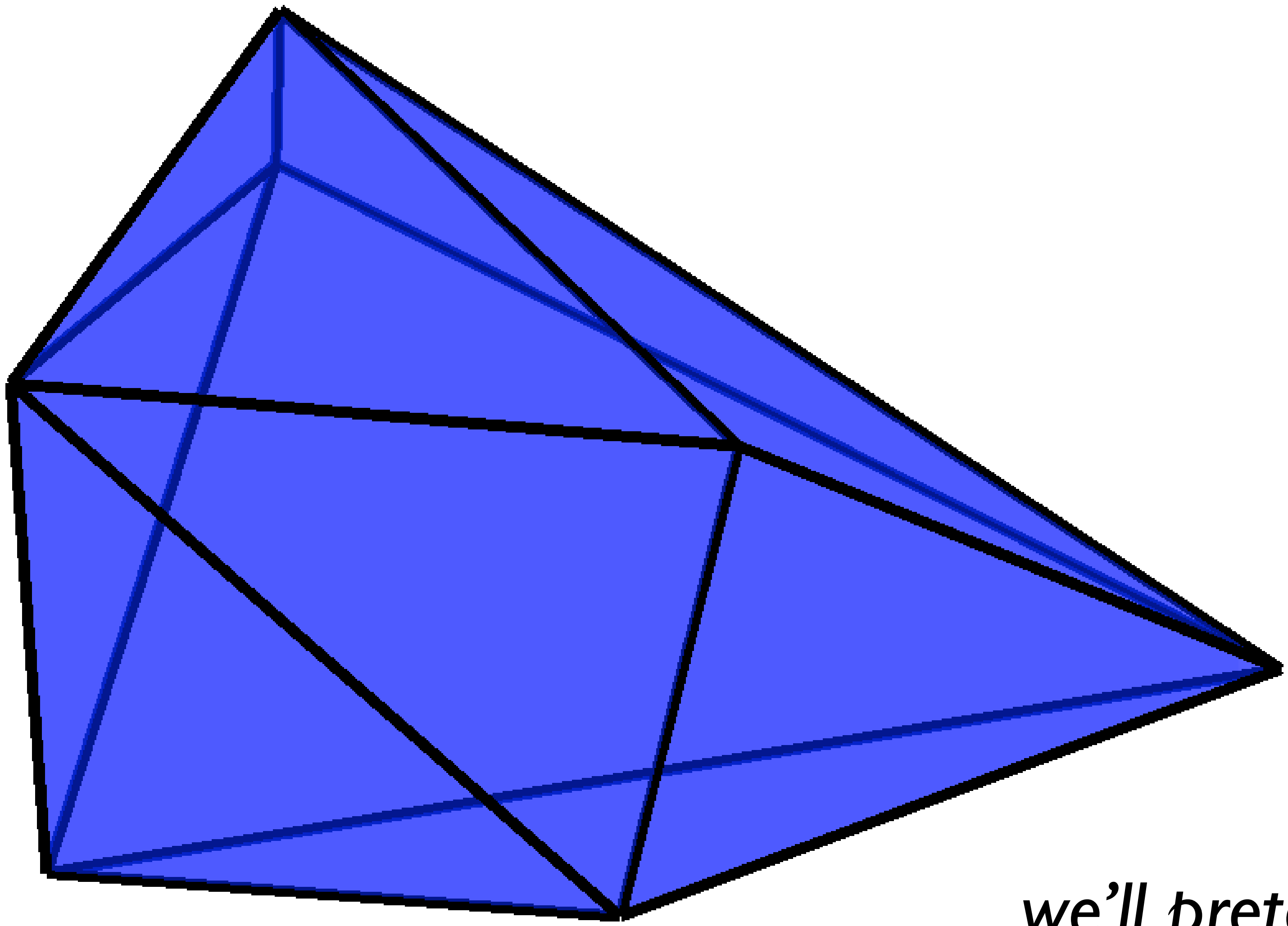
1	0	0	-2	5	8/3
0	1	0	1	-2	4/3
0	0	1	1	-3	0

1	0	2	0	-1	8/3
0	1	-1	0	1	4/3
0	0	1	1	-3	0

Degeneracy in 3D



Degeneracy in 3D

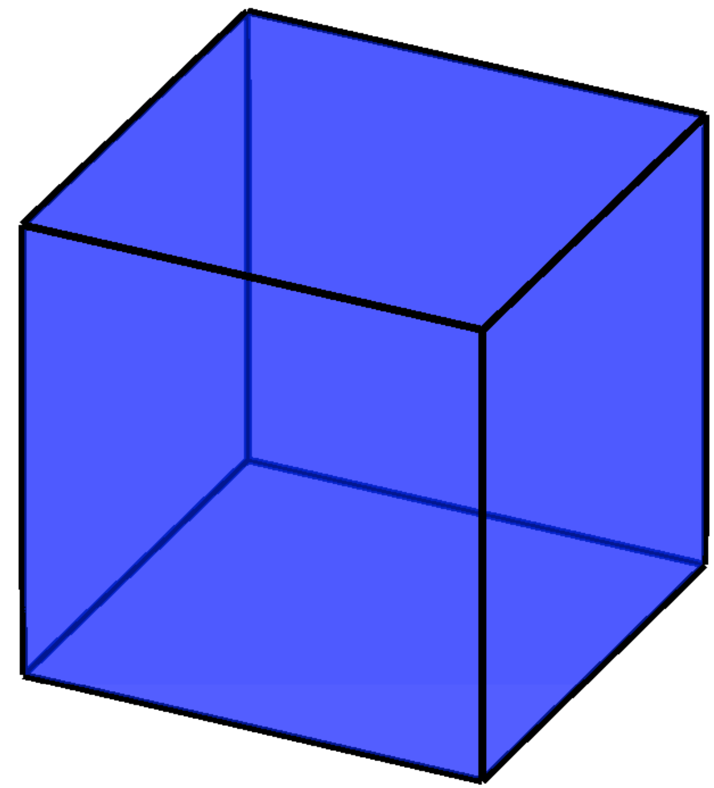


we'll pretend this never happens

Neighboring bases

- Two bases are ***neighbors*** if they share (m–1) variables
- Neighboring feasible bases correspond to vertices connected by an edge (note: degeneracy)

x	y	z	u	v	w	RHS
1	0	0	1	0	0	1
0	1	0	0	1	0	1
0	0	1	0	0	1	1

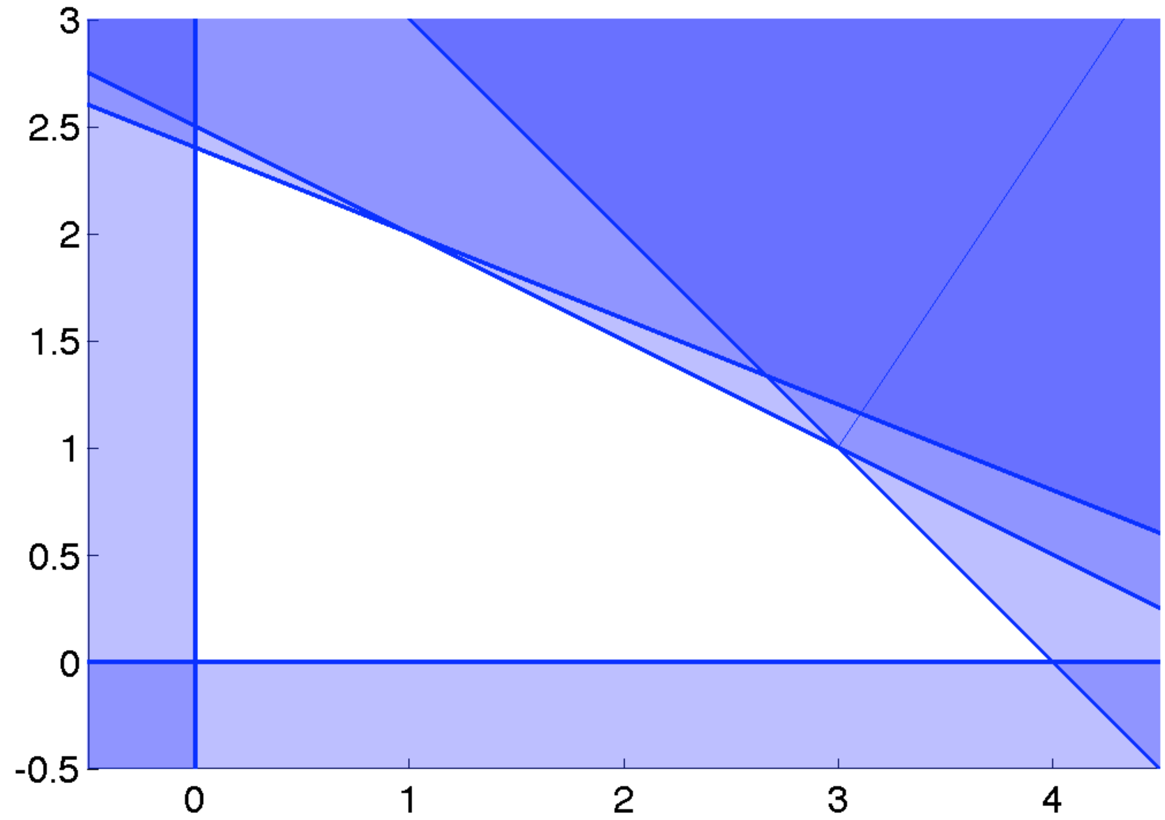


Improving our search

- Naïve: enumerate all possible bases
- Smarter: maybe neighbors of good bases are also good?
- ***Simplex*** algorithm: repeatedly move to a neighboring basis to improve objective
    - important advantage: rank-1 update is *fast*

Example

max 2x + 3y s.t.
x + y ≤ 4
2x + 5y ≤ 12
x + 2y ≤ 5



x	y	s	t	u	RHS
1	1	1	0	0	4
2	5	0	1	0	12
1	2	0	0	1	5
2	3	0	0	0	↑

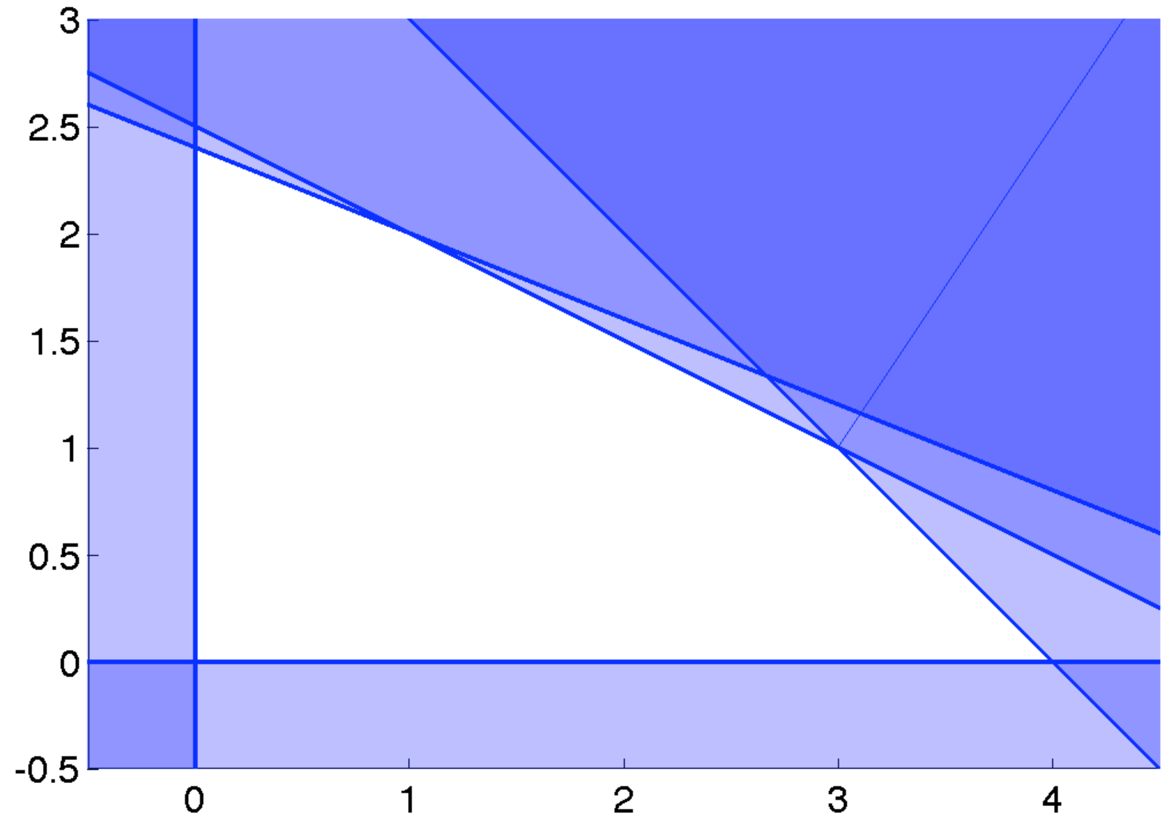
Example

max $2x + 3y$ s.t.

$x + y \leq 4$

$2x + 5y \leq 12$

$x + 2y \leq 5$



x	y	s	t	u	RHS
0.4	1	0	0.2	0	2.4
0.6	0	1	−0.2	0	1.6
0.2	0	0	−0.4	1	0.2
0.8	0	0	−0.6	0	↑

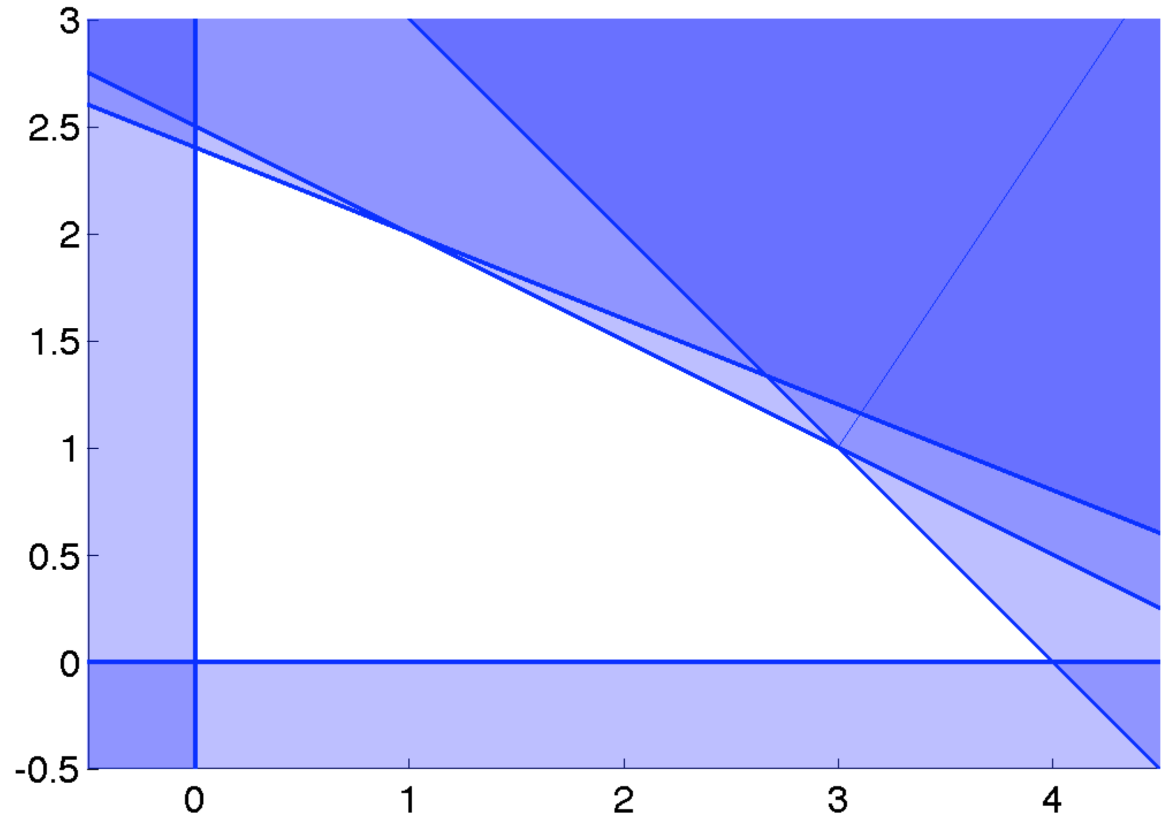
Example

max 2x + 3y s.t.

$x + y \leq 4$

$2x + 5y \leq 12$

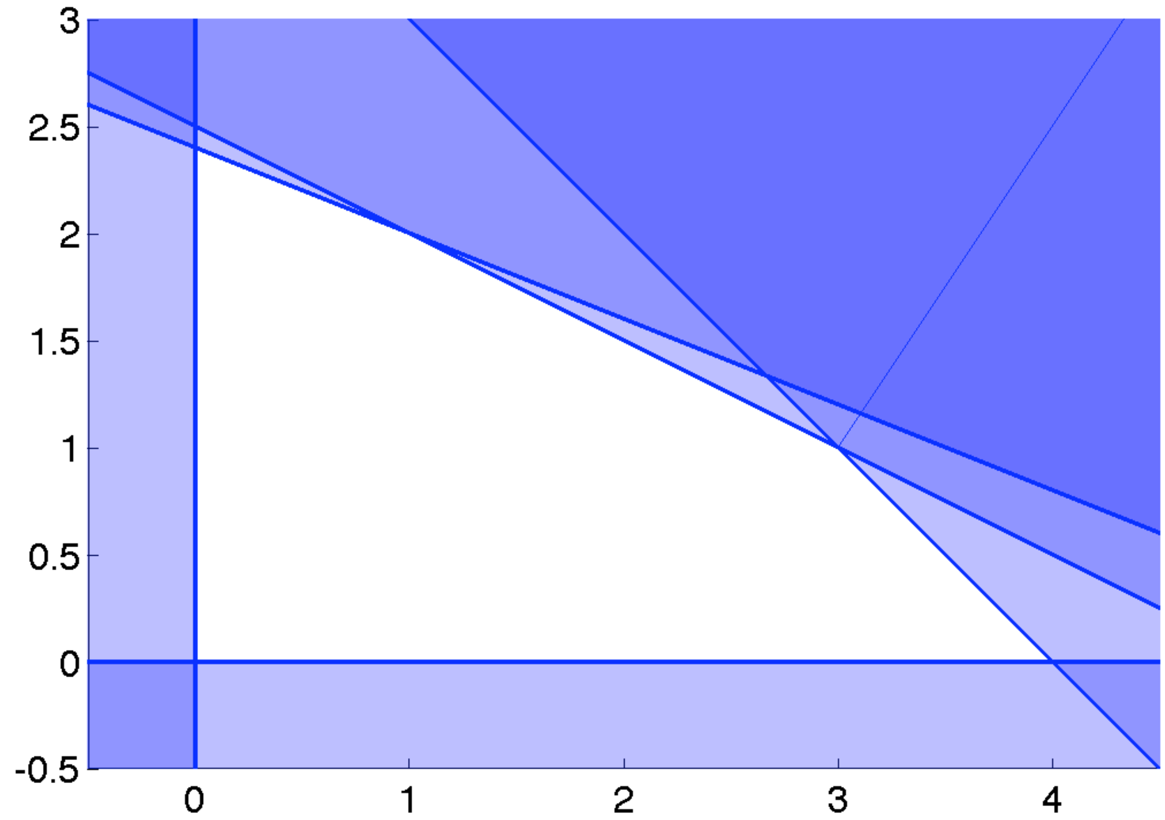
$x + 2y \leq 5$



x	y	s	t	u	RHS
1	0	0	−2	5	1
0	1	0	1	−2	2
0	0	1	1	−3	1
0	0	0	1	−4	↑

Example

max 2x + 3y s.t.
x + y ≤ 4
2x + 5y ≤ 12
x + 2y ≤ 5



x	y	s	t	u	RHS
1	0	2	0	−1	3
0	1	−1	0	1	1
0	0	1	1	−3	1
0	0	−1	0	−1	↑