Linear Programming in Low Dimensions

(most slides by Nati Srebro)

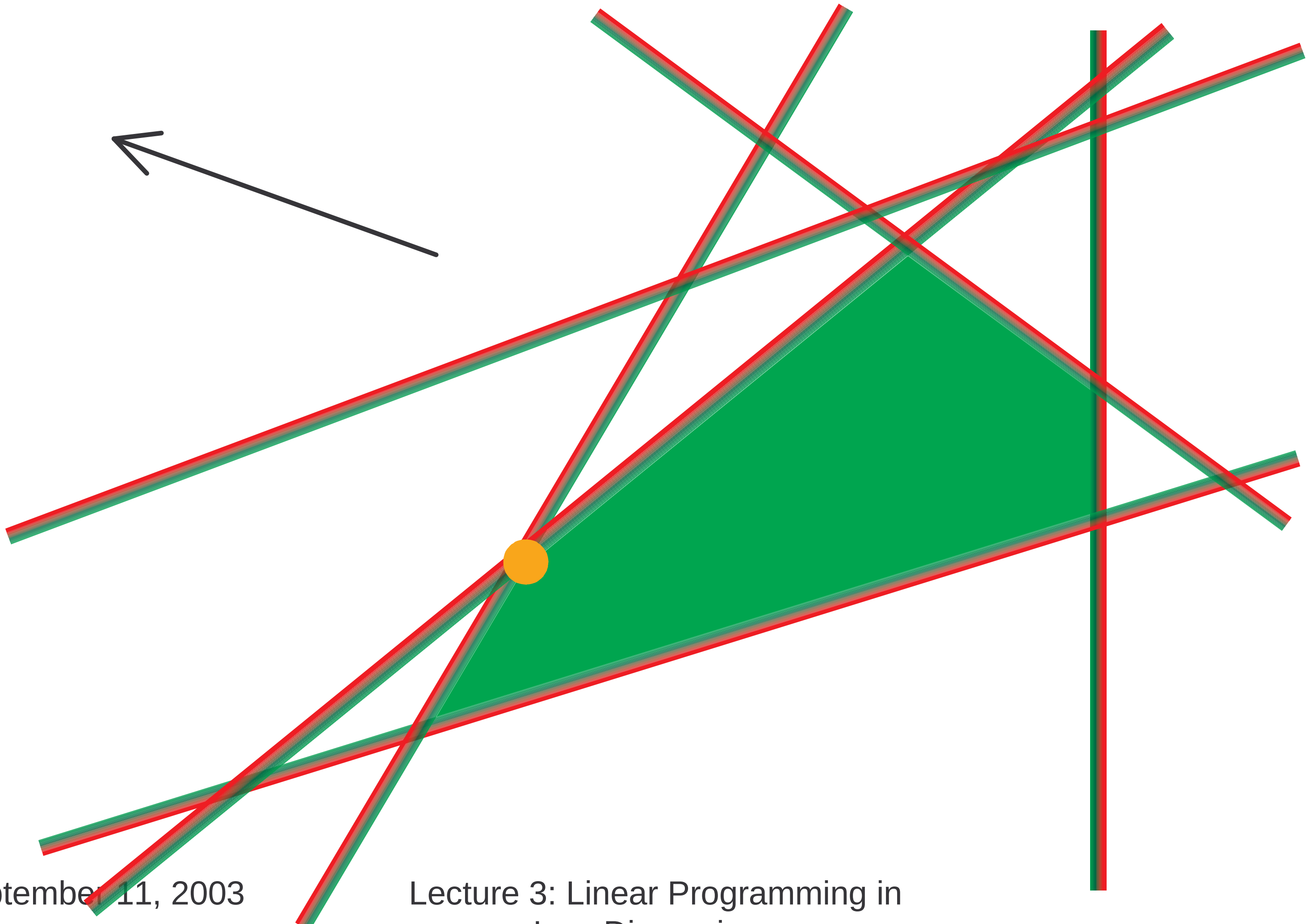
Linear Programming

$$
\begin{aligned}
\text{Maximize:} \quad & c_1 x_1 + c_2 x_2 + \cdots + c_d x_d \\
\text{Subject to:} \quad & a_{1,1} x_1 + a_{1,2} x_2 + \cdots + a_{1,d} x_d \le b_1 \\
& a_{2,1} x_1 + a_{2,2} x_2 + \cdots + a_{2,d} x_d \le b_2 \\
& \vdots \qquad\qquad\qquad \ddots \qquad\qquad\qquad \vdots \\
& a_{n,1} x_1 + a_{n,2} x_2 + \cdots + a_{n,d} x_d \le b_n
\end{aligned}
$$

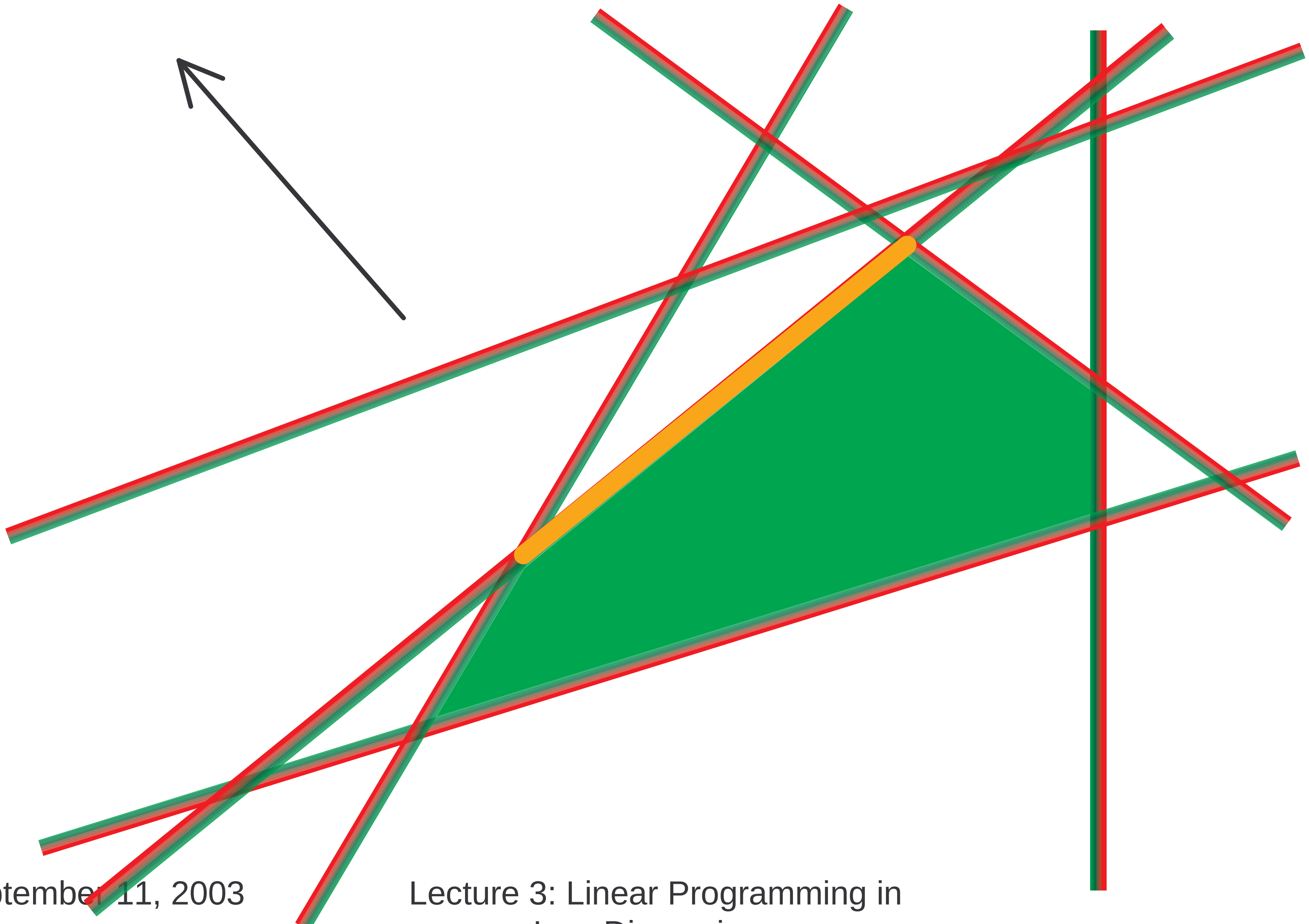
Linear Programming in 2D

$$
\begin{aligned}
\text{Maximize:} \quad & c_x x + c_y y \\
\text{Subject to:} \quad & a_{1,x} x + a_{1,y} y \le b_1 \\
& a_{2,x} x + a_{2,y} y \le b_2 \\
& \vdots \qquad\quad \ddots \qquad\quad \vdots \\
& a_{n,x} x + a_{n,y} y \le b_n
\end{aligned}
$$

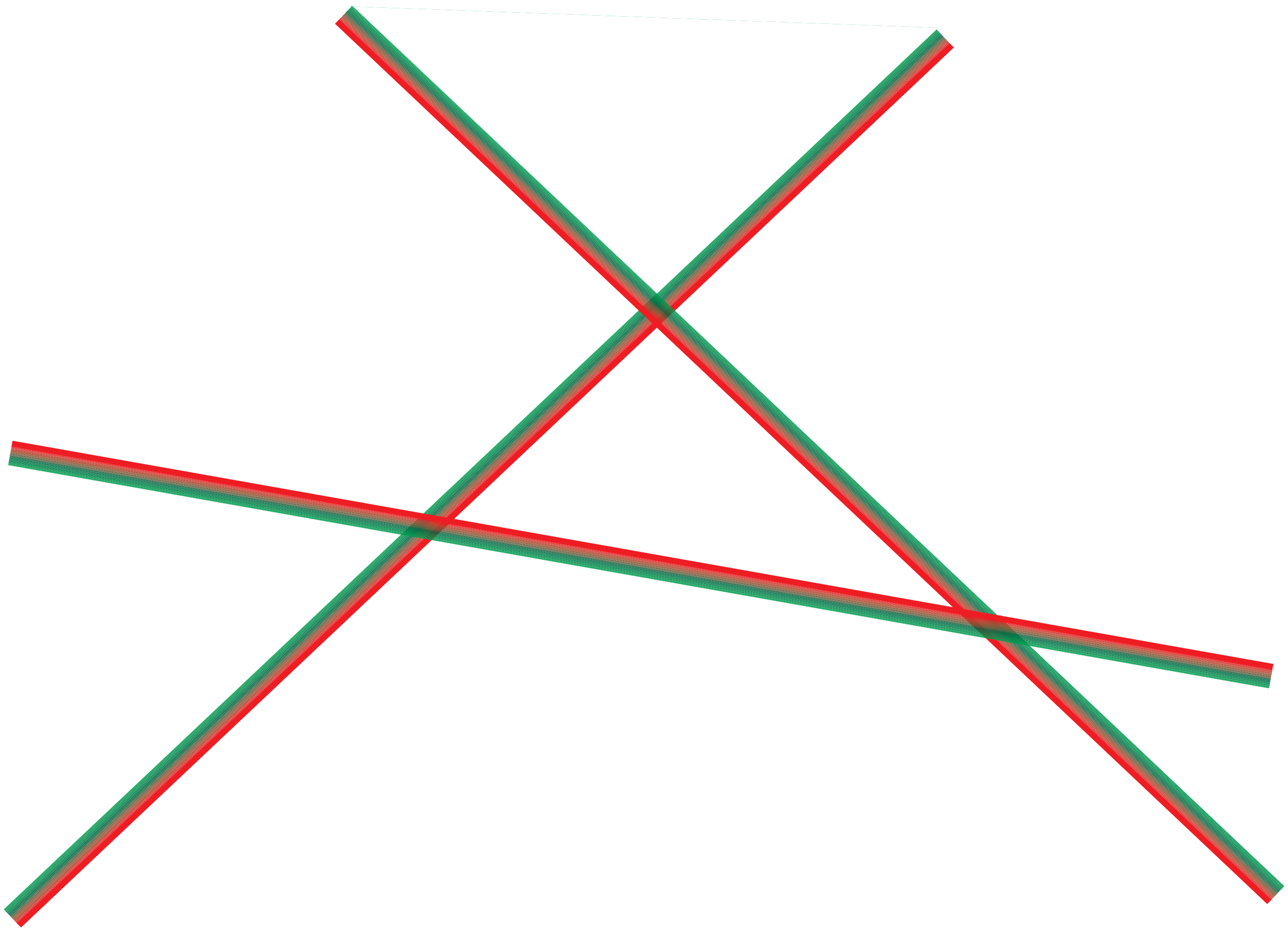
Linear Programming in 2D



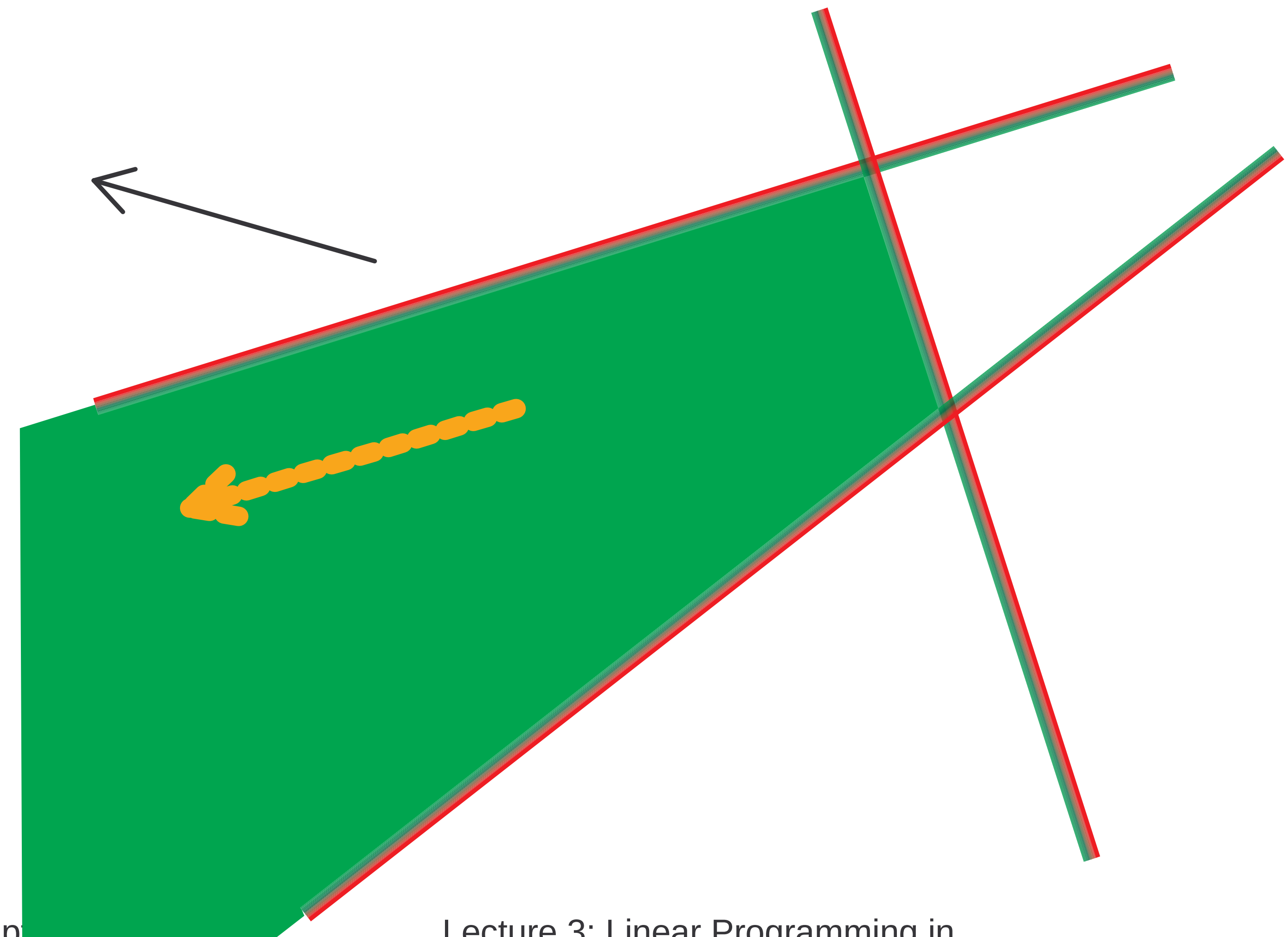
Non-unique solution



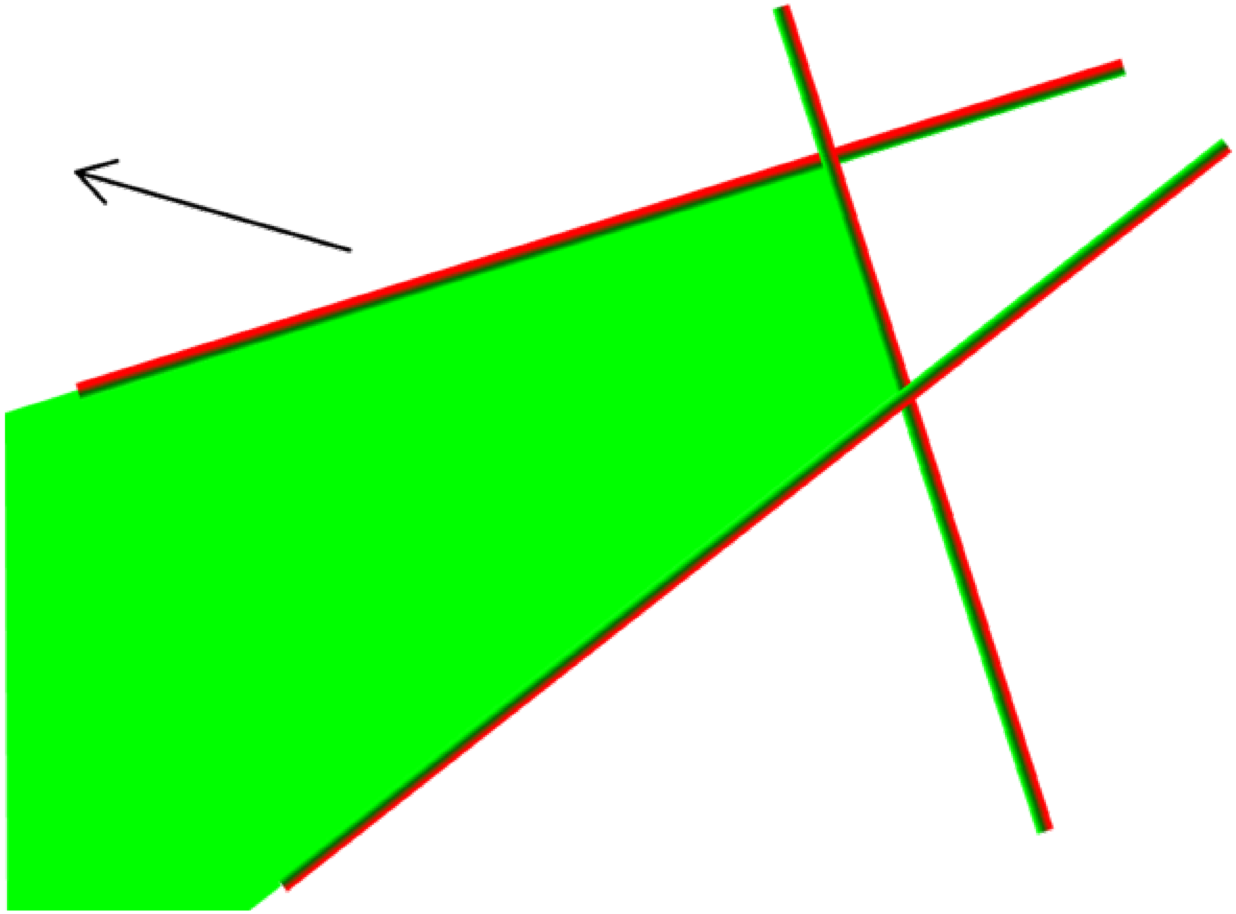
An Infeasible Linear Program



An Unbounded LP

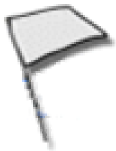


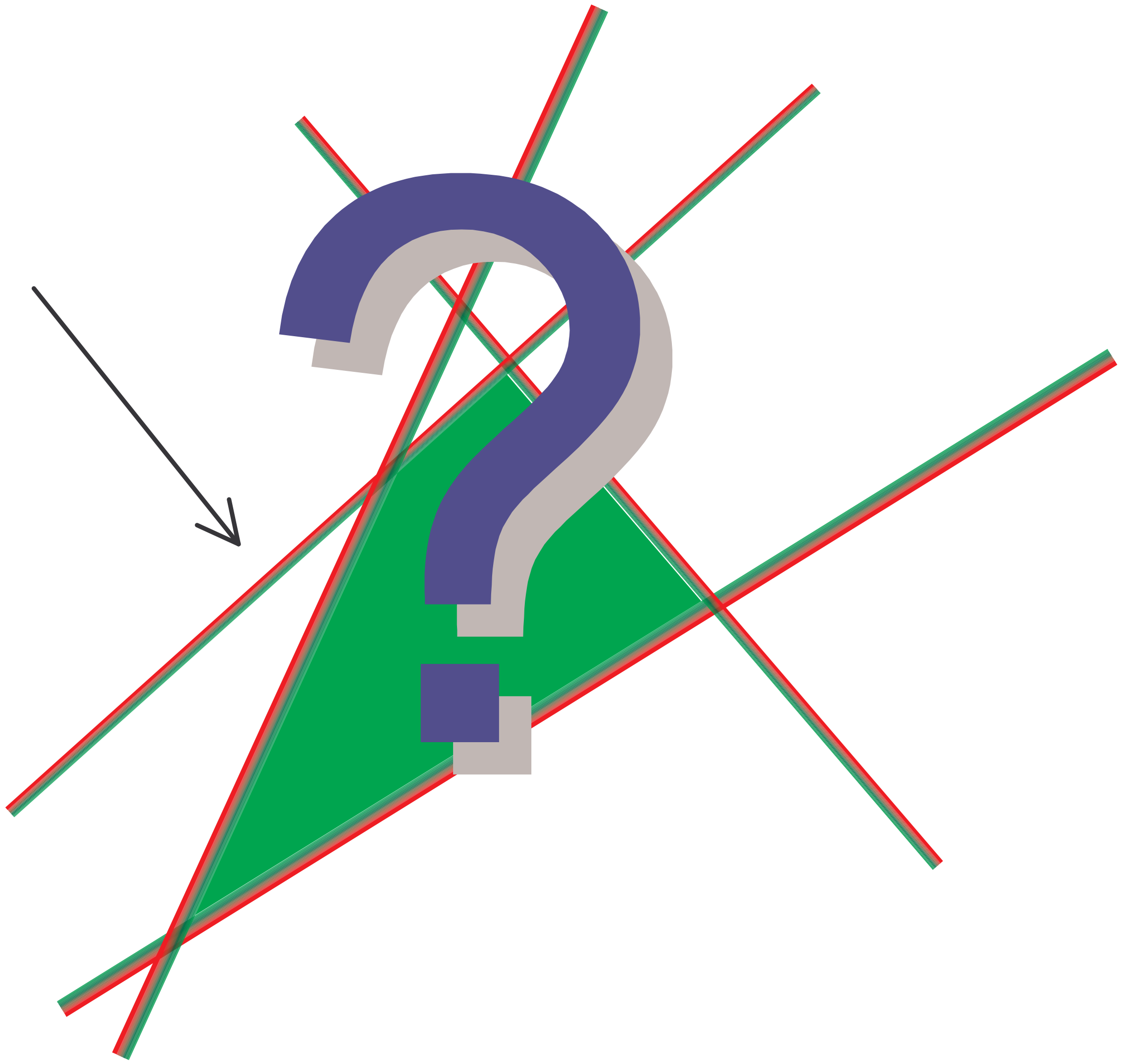
An ~~Un~~bounded LP

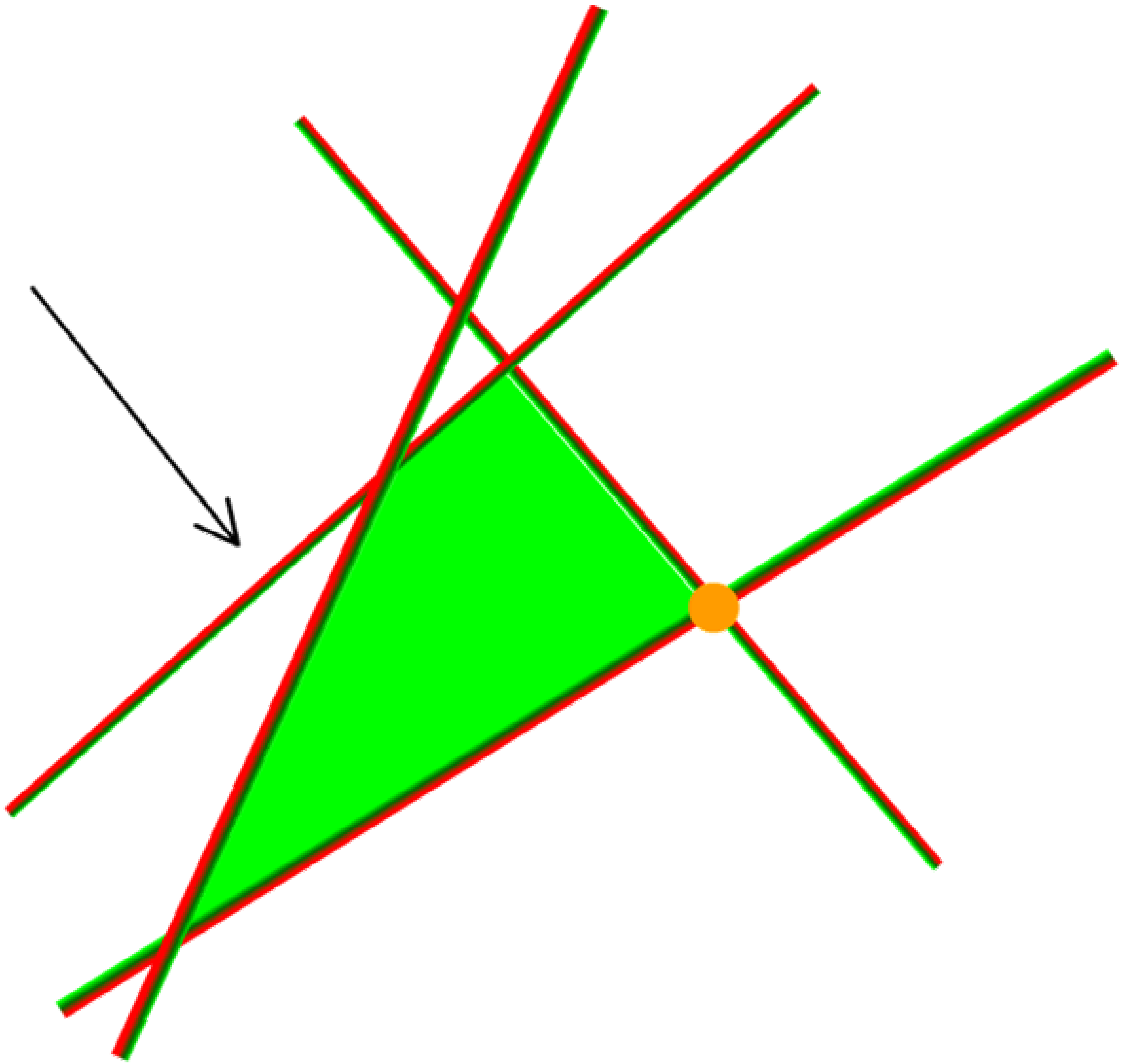


Types of LPs

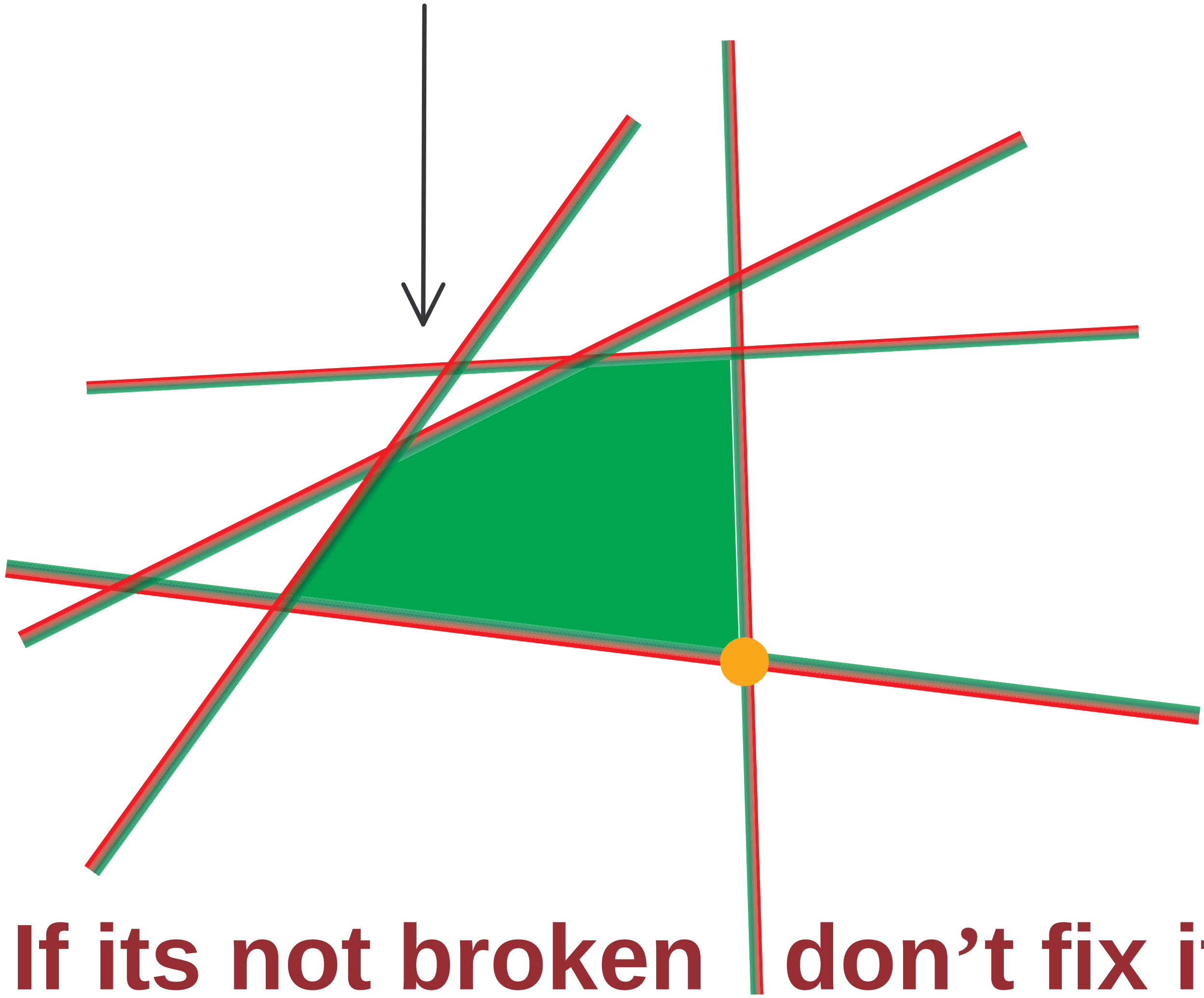
- Unique optimum
  - ØFind the optimum
- Optimal edge
  - ØFind an optimum
- Unbounded
  - ØFind unbounded ray
- Infeasible
  - ØDeclare as unfeasible





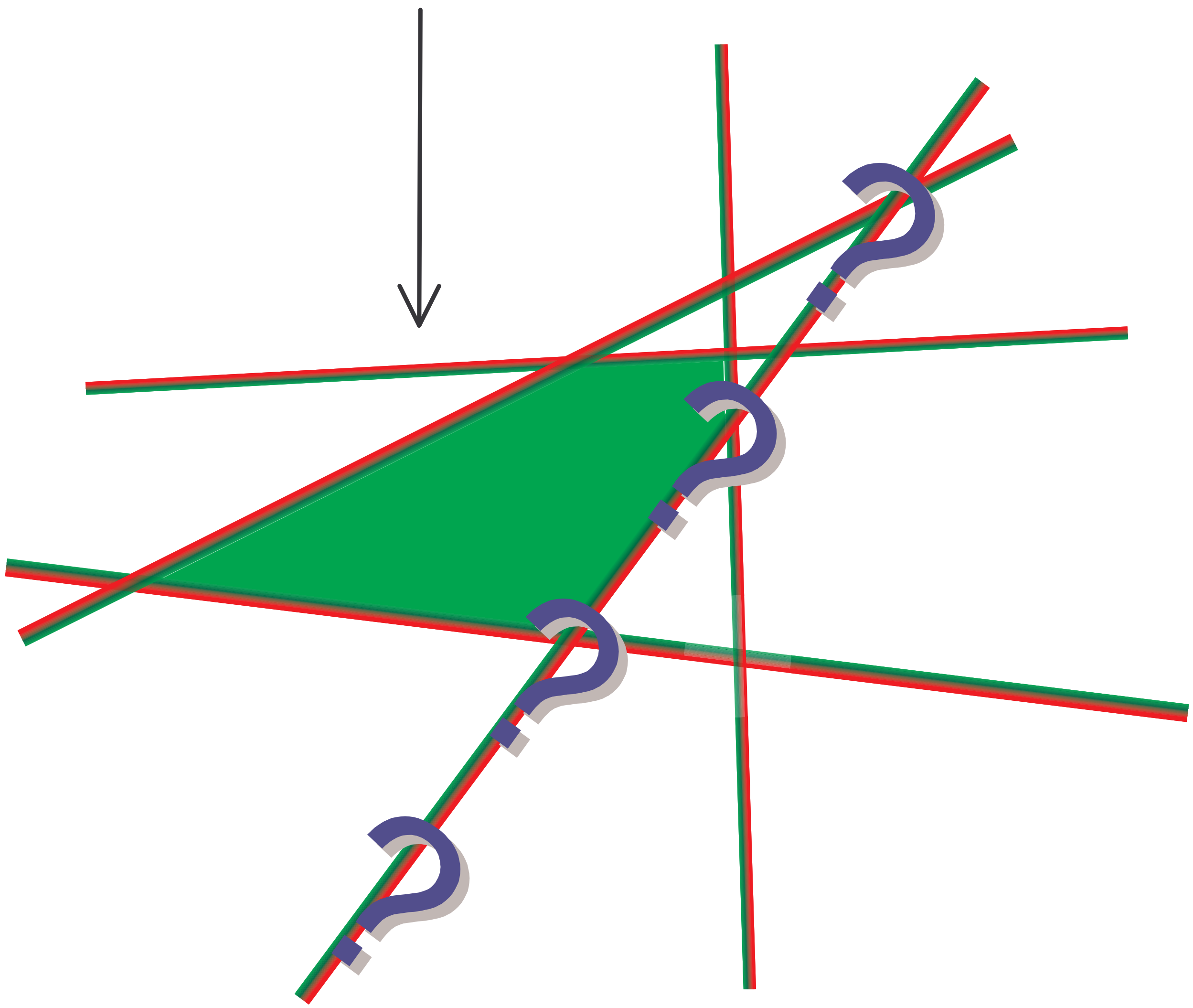


Adding a New Constraint

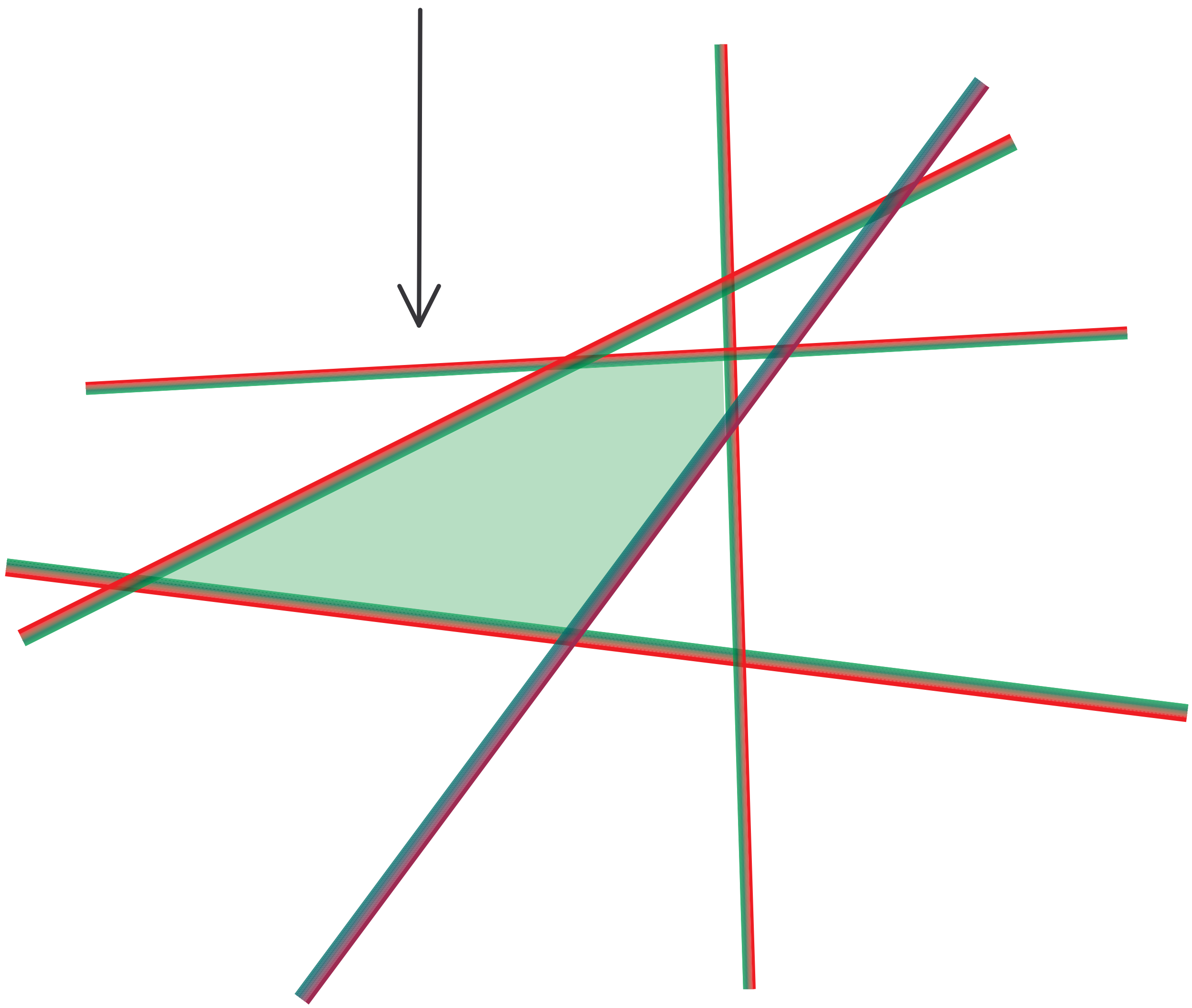


If its not broken don’t fix it !

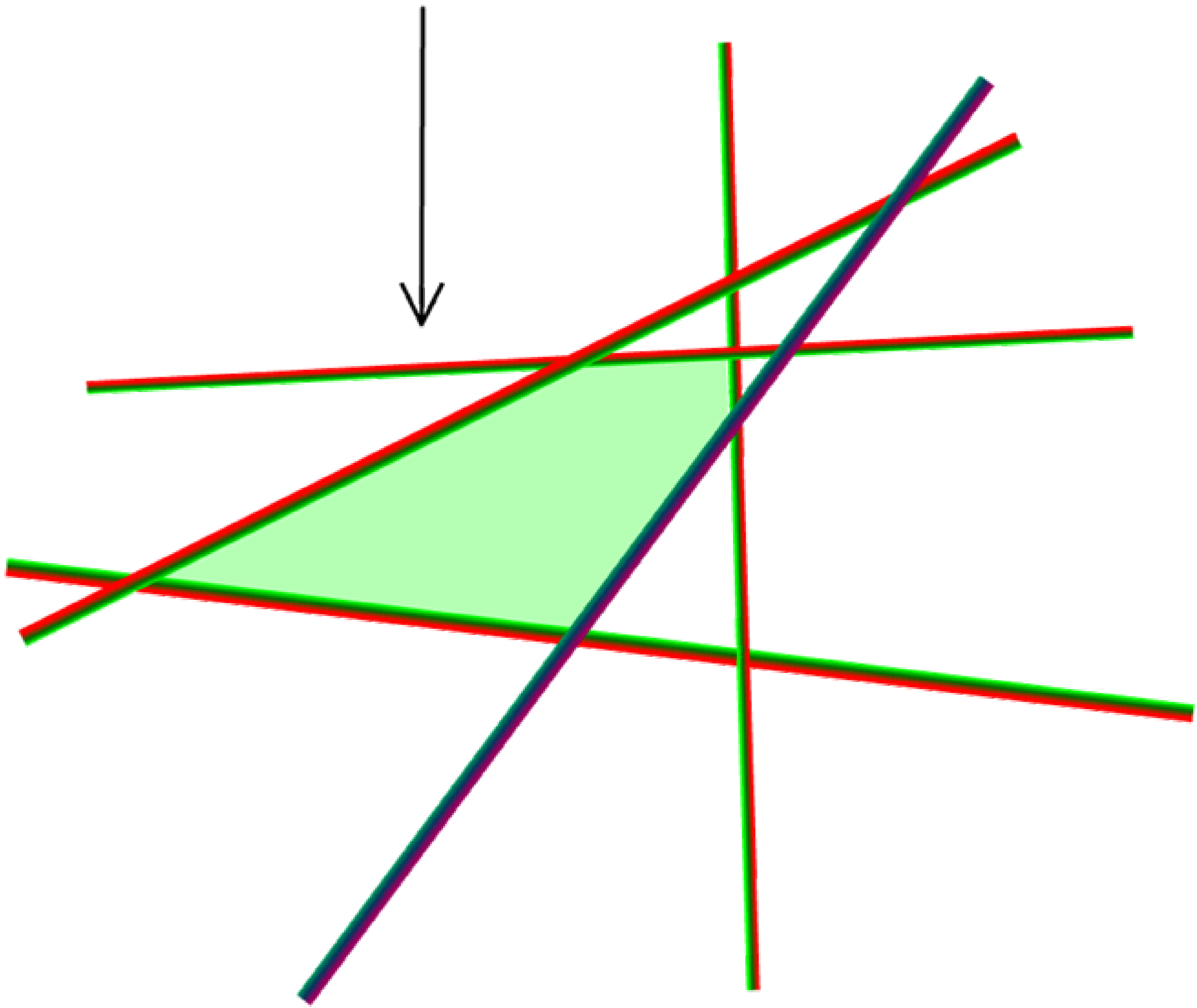
Adding a New Constraint



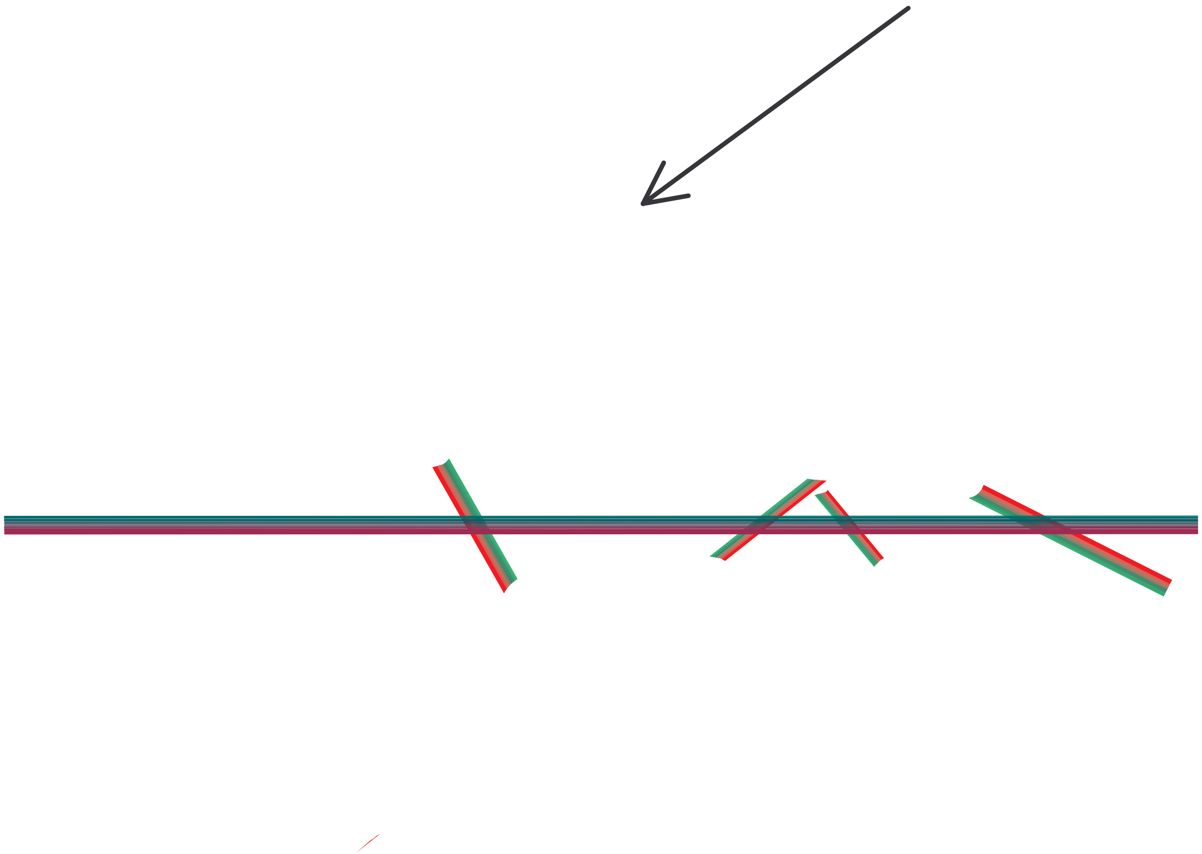
Adding a New Constraint



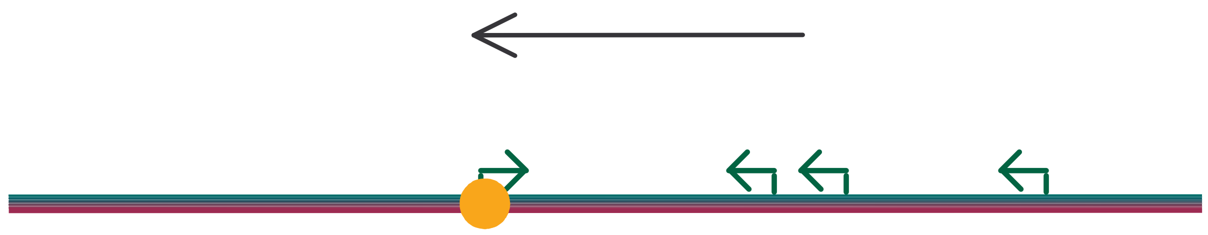
Adding a New Constraint



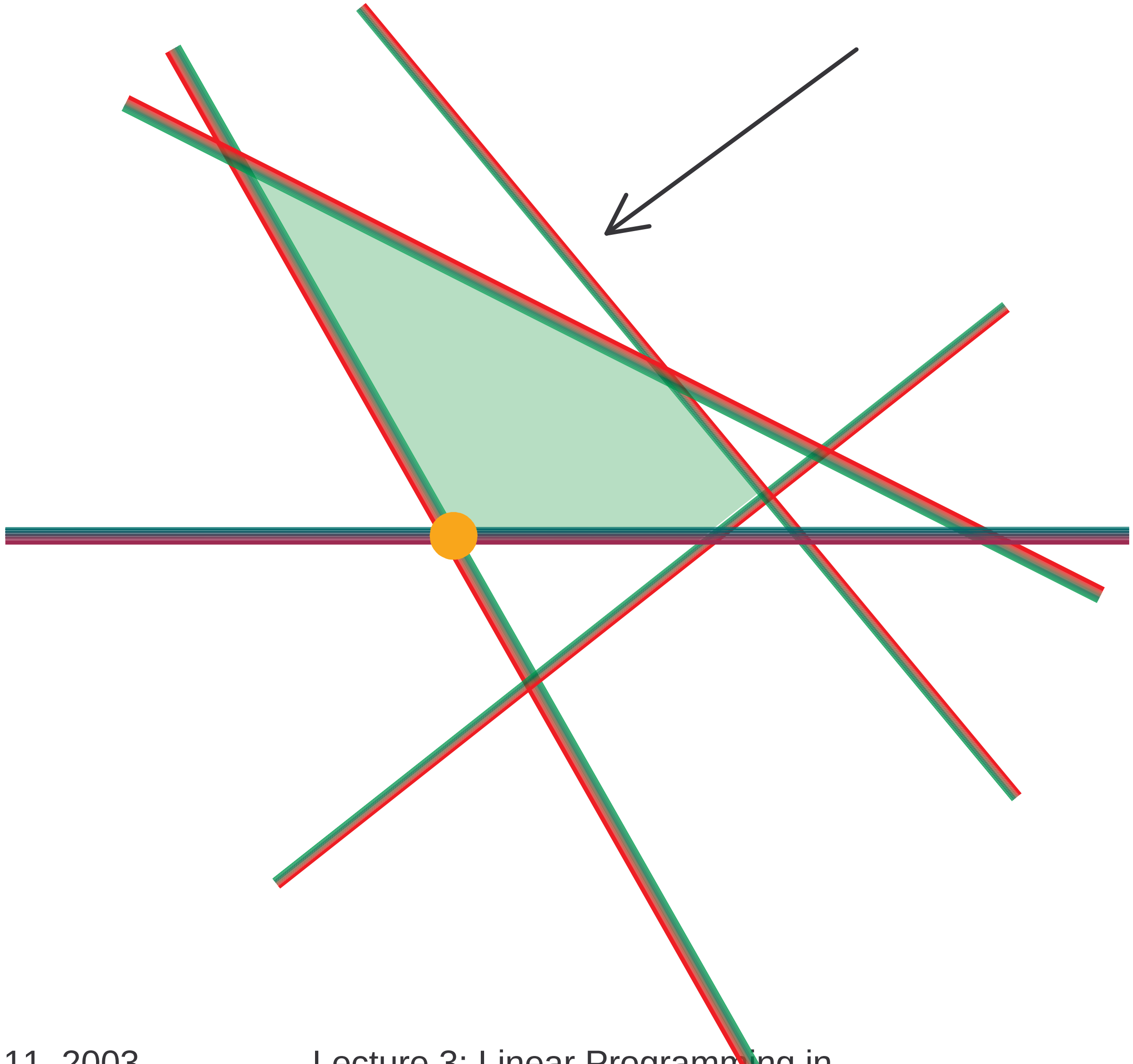
Adding a New Constraint

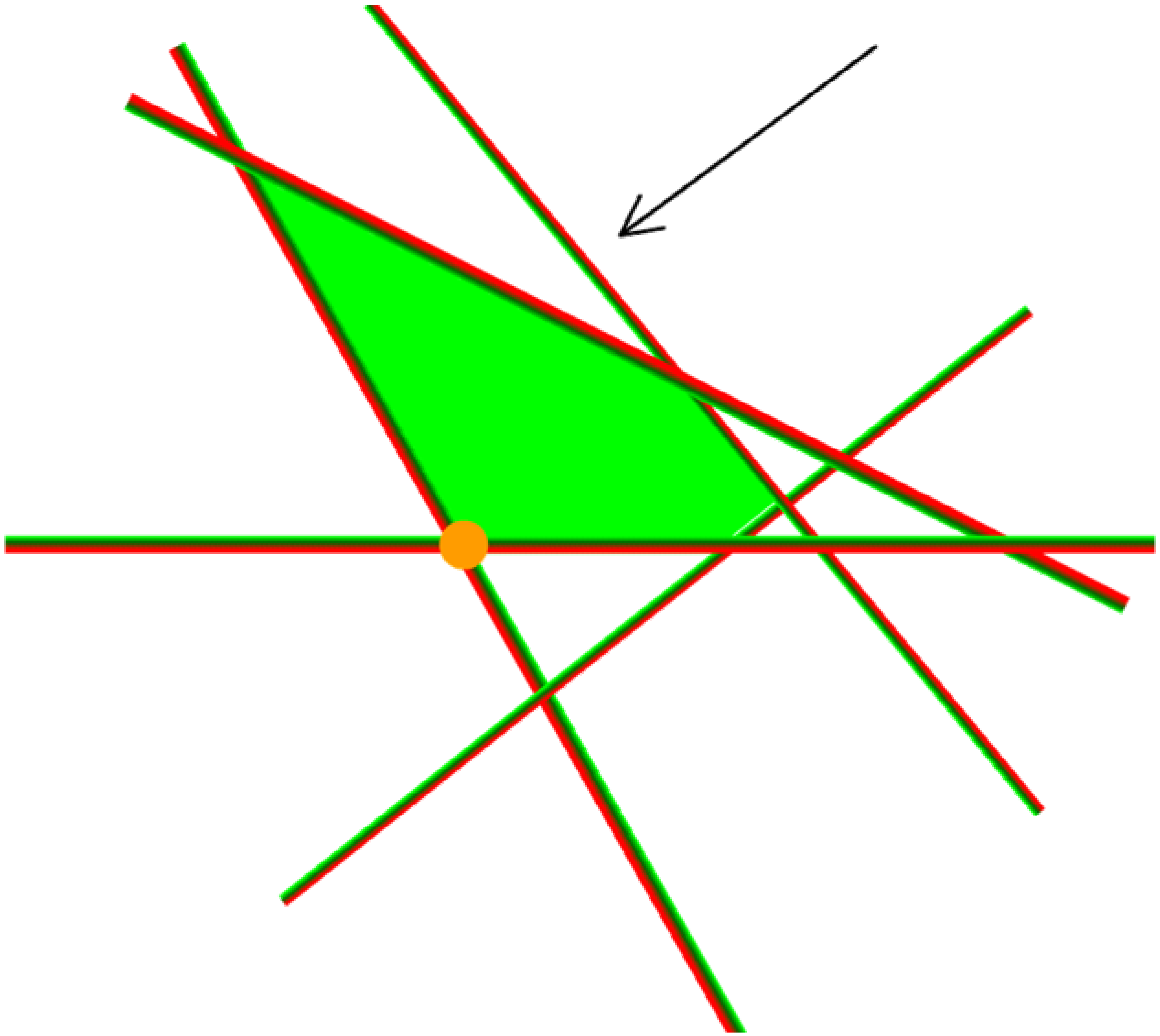


Adding a New Constraint

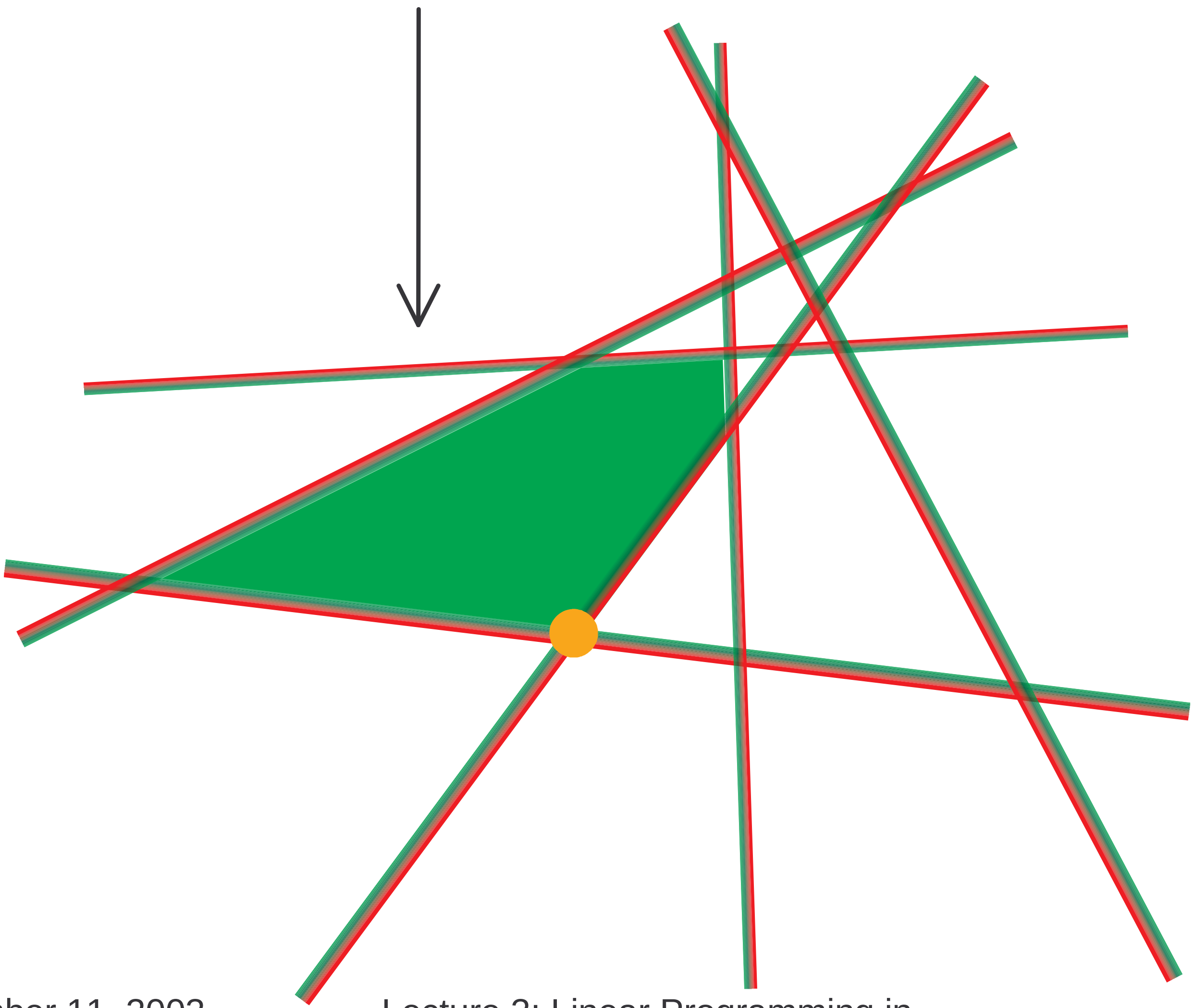


Adding a New Constraint

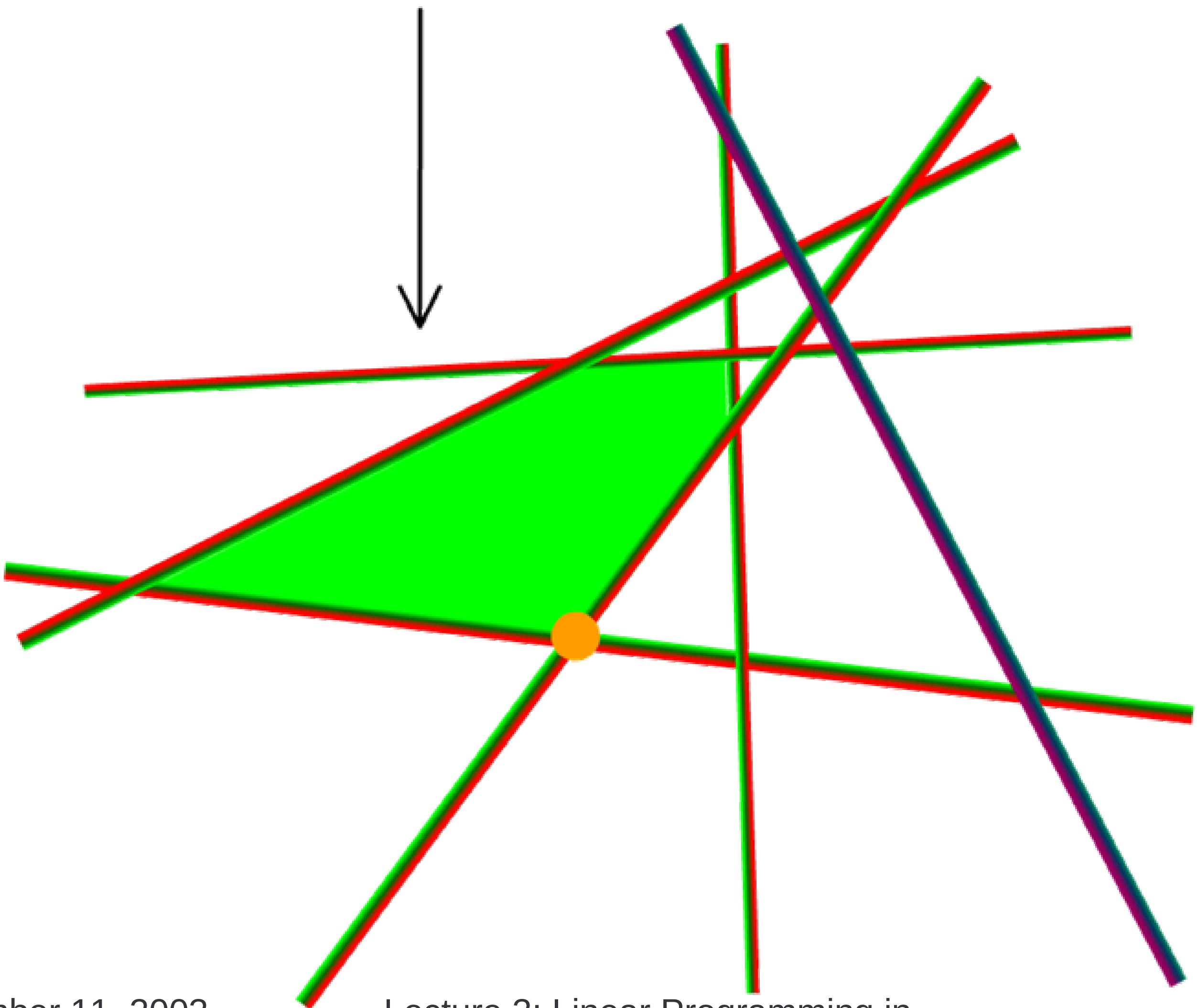




Adding a New Constraint



Adding a New Constraint



Adding a New Constraint

Adding a New Constraint

 Is a bad sign….

Summary

- Check is optimum is feasible ***O(1)***
- Optimum is feasible:

  We're fine, don't do anything
- Optimum isn't feasible:

  Find optimum on new constraint (line) ***O(n)***

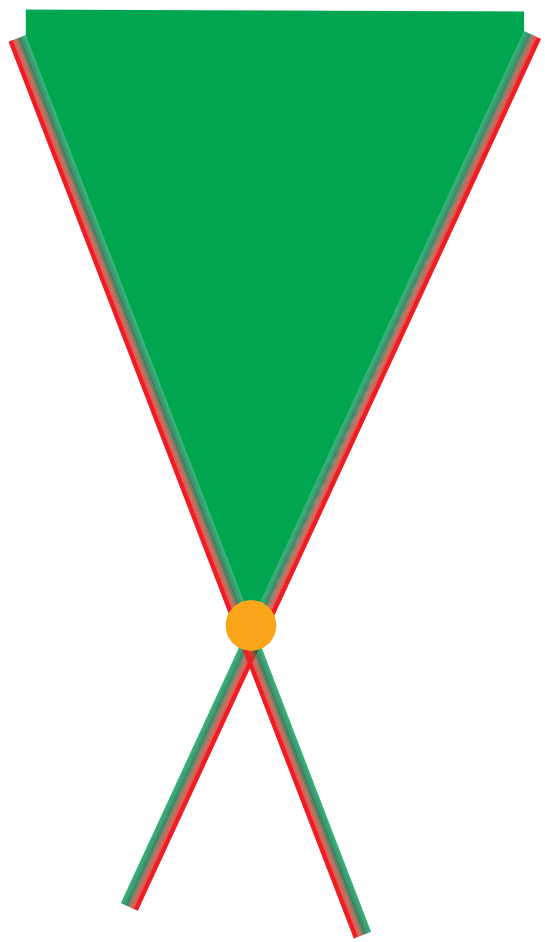
  No feasible points on new constraint:

  LP isn't feasible

Incremental Algorithm

- Choose two constraints and initialize the solution
- Add new constraints one by one, keeping track of current optimum

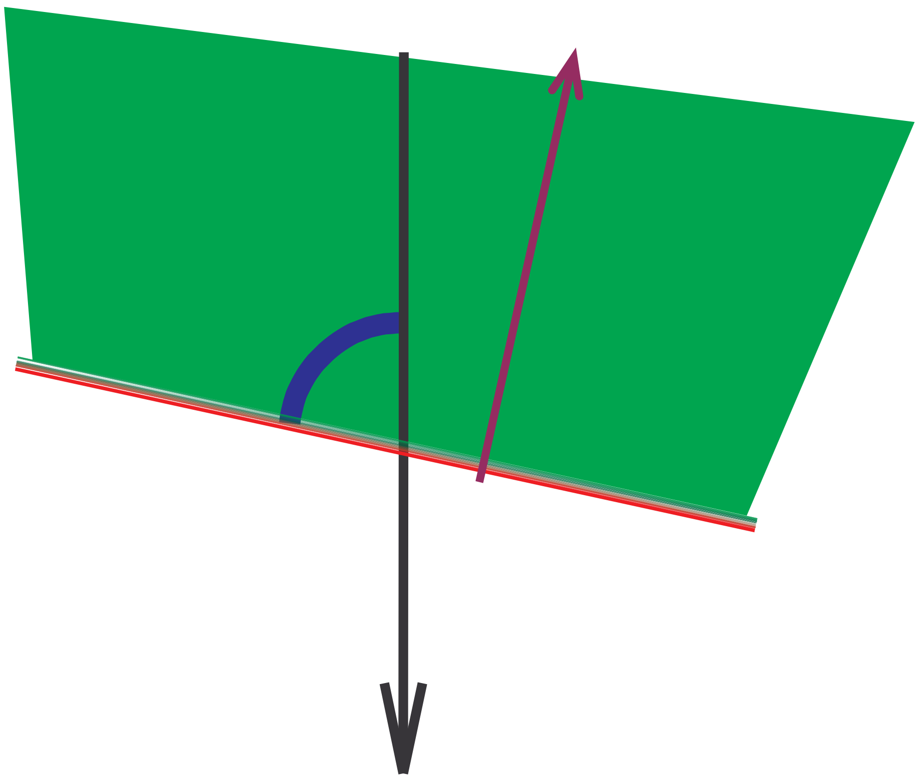
Initialization



Find two constraints that together bound the LP

Initialization

- Choose the constraint *h* defined by a vector that is the closest to "up"



Initialization

- Choose the constraint *h* defined by a vector that is the closest to "up"
- For every other constraint,
  check if it bounds the LP with *h*
- If no constraint is good---
  LP is unbounded

or unfeasible because of parallel constraints

Incremental Algorithm

- Find two constraints that bound LP
  - *If none exist, LP is unbounded* $O(n)$
- Add all other constraints one by one, keeping track of current optimum $O(n^2)$

Adding a New Constraint

- Check is optimum is feasible ***O(1)***
- Optimum is feasible:

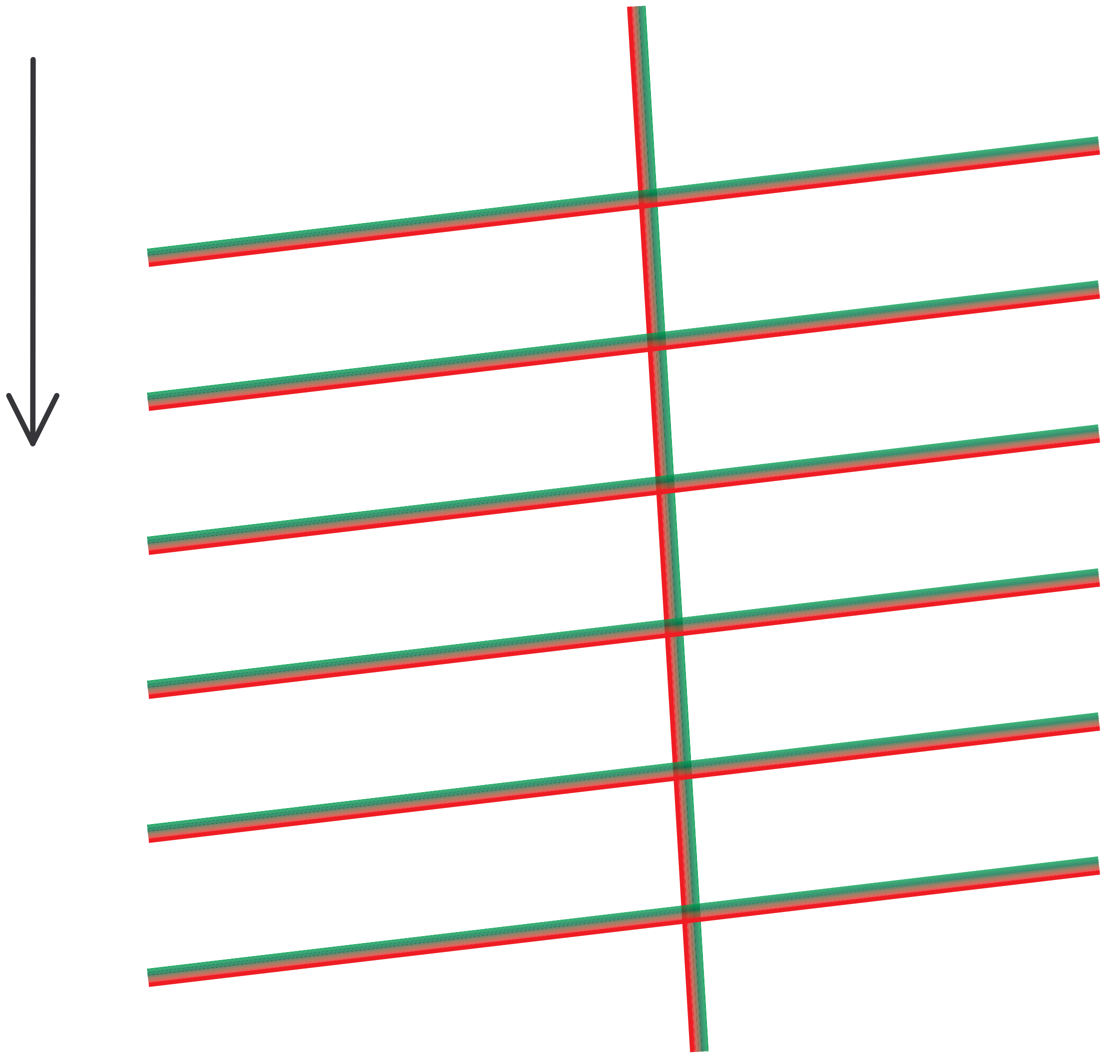
  We're fine, don't do anything
- Optimum isn't feasible:

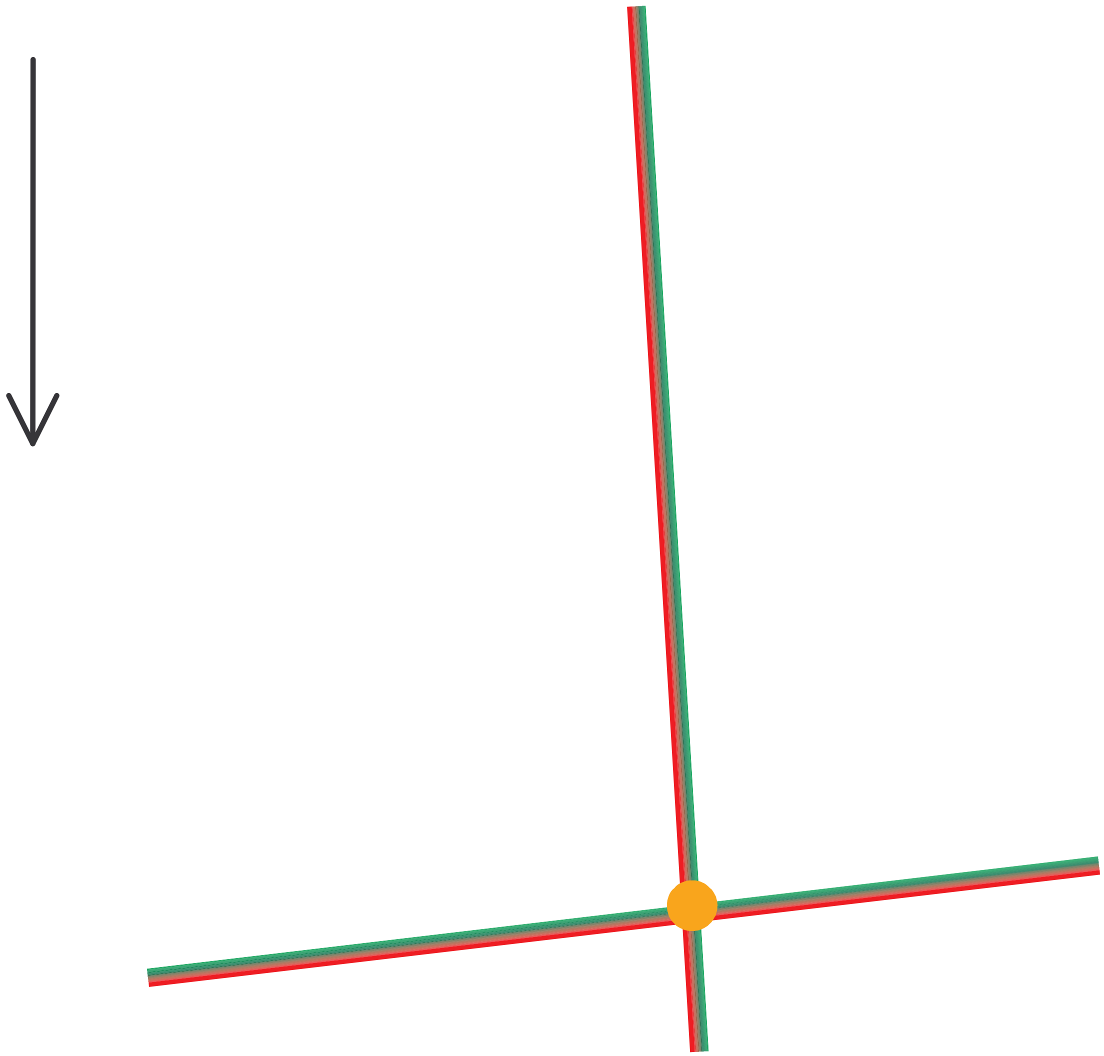
  Find optimum on new constraint (line) ***O(n)***

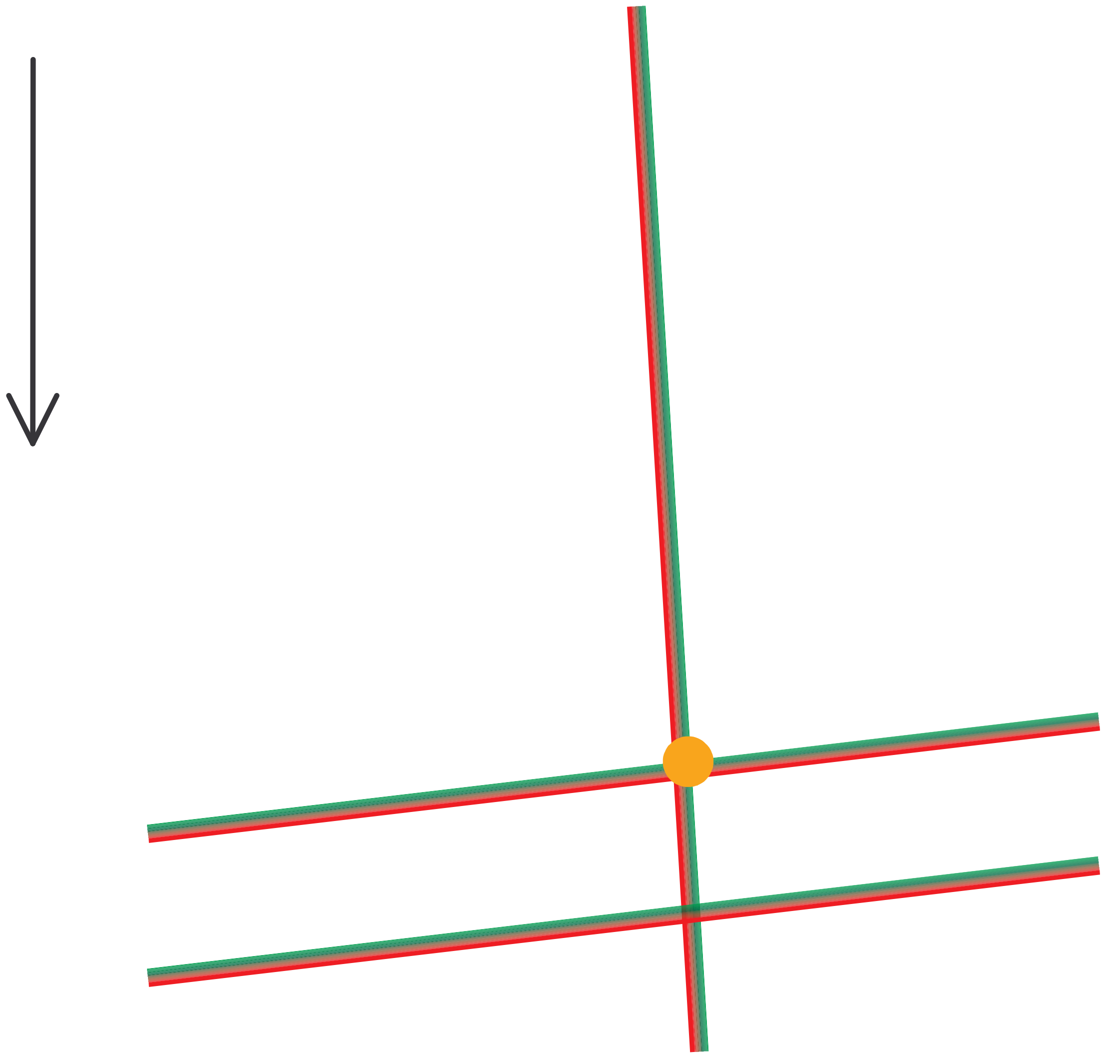
  No feasible points on new constraint:

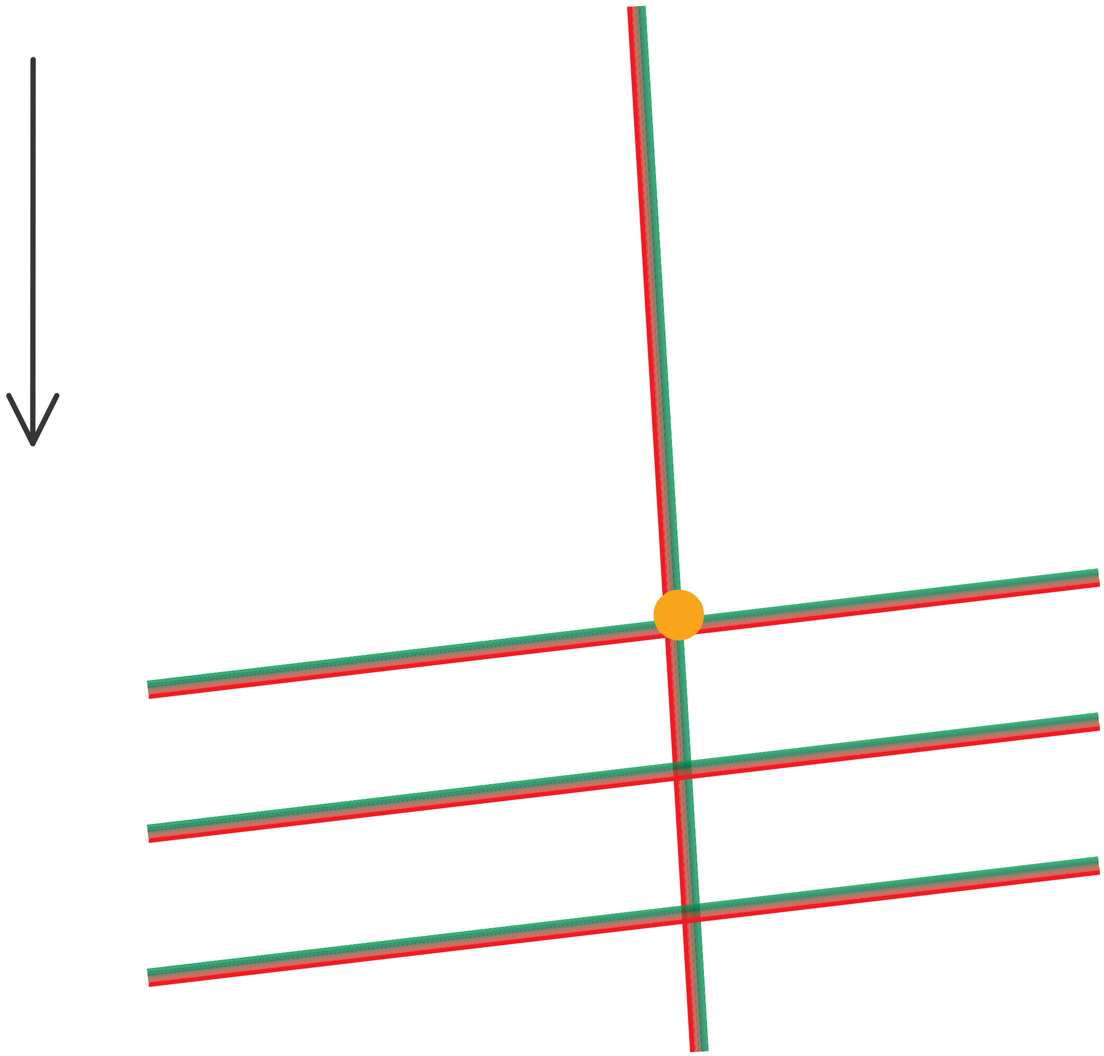
  LP isn't feasible

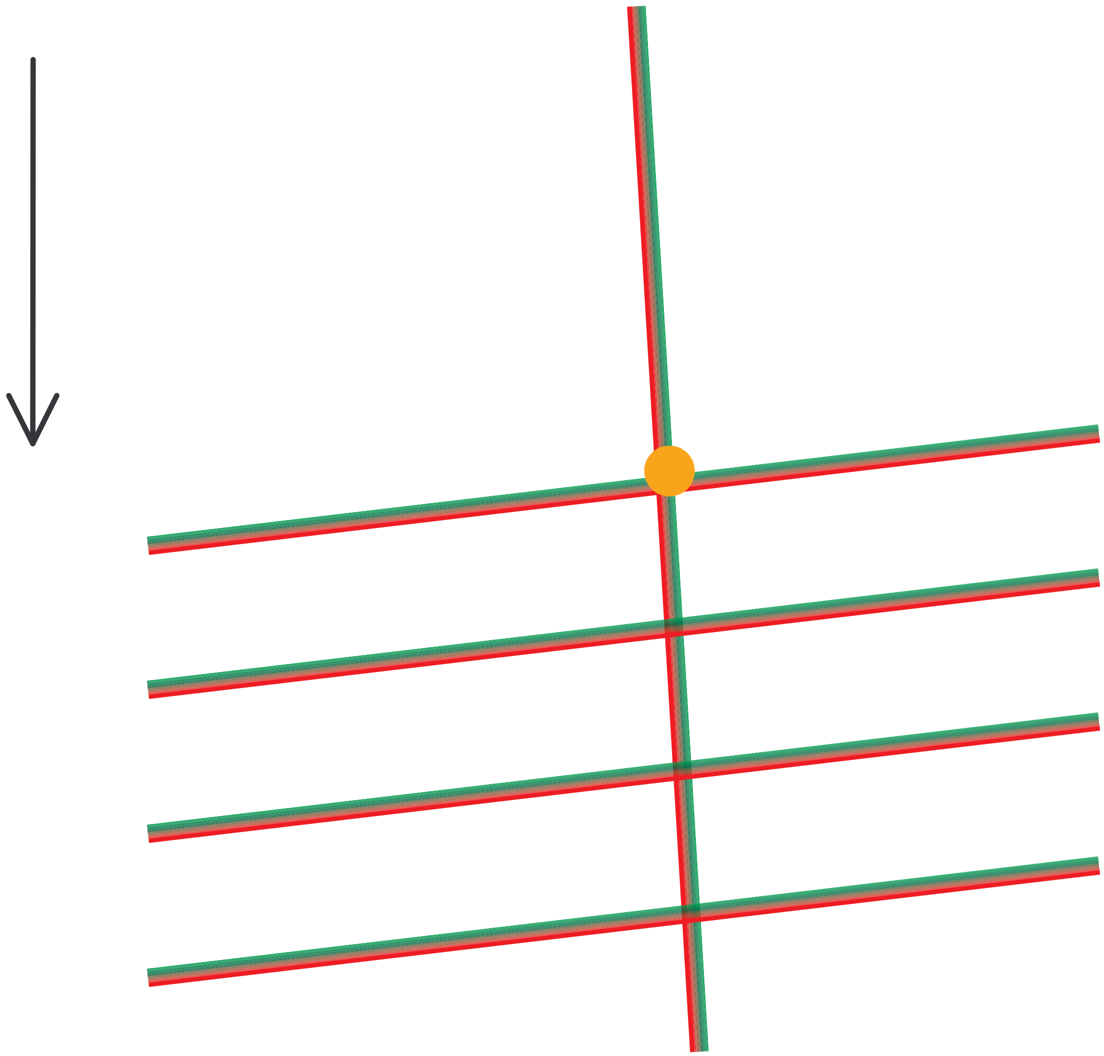
Maybe we only rarely have to update optimum ?

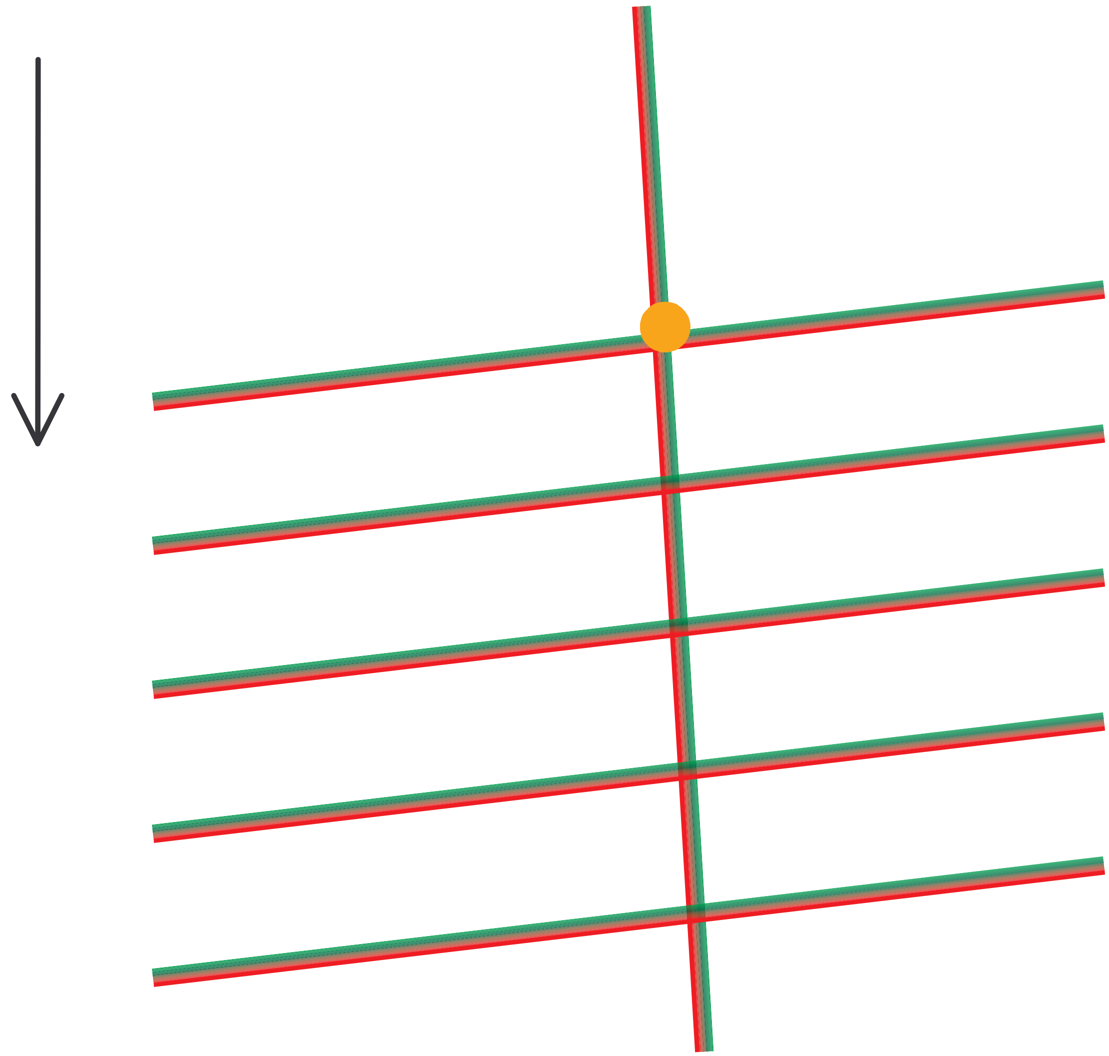


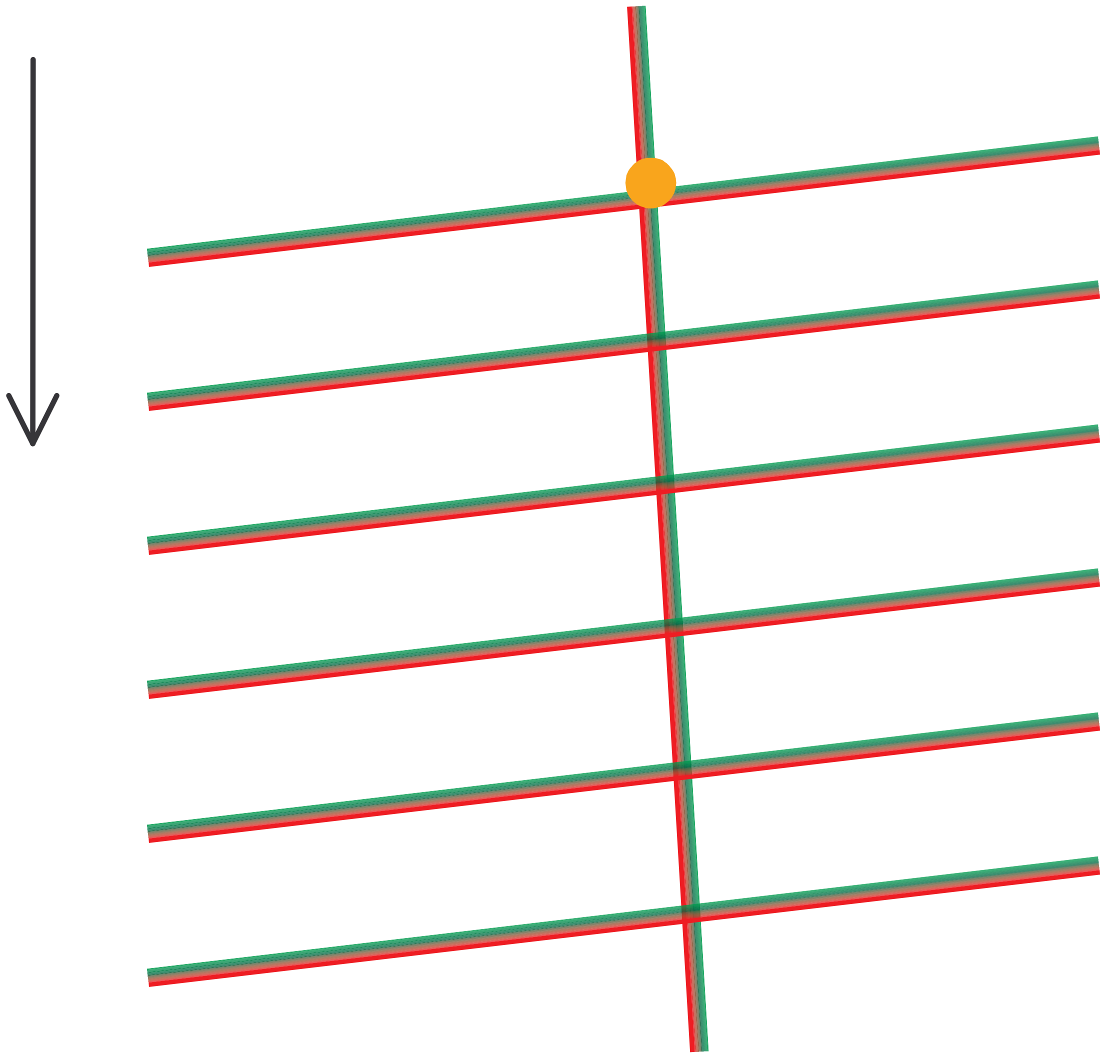






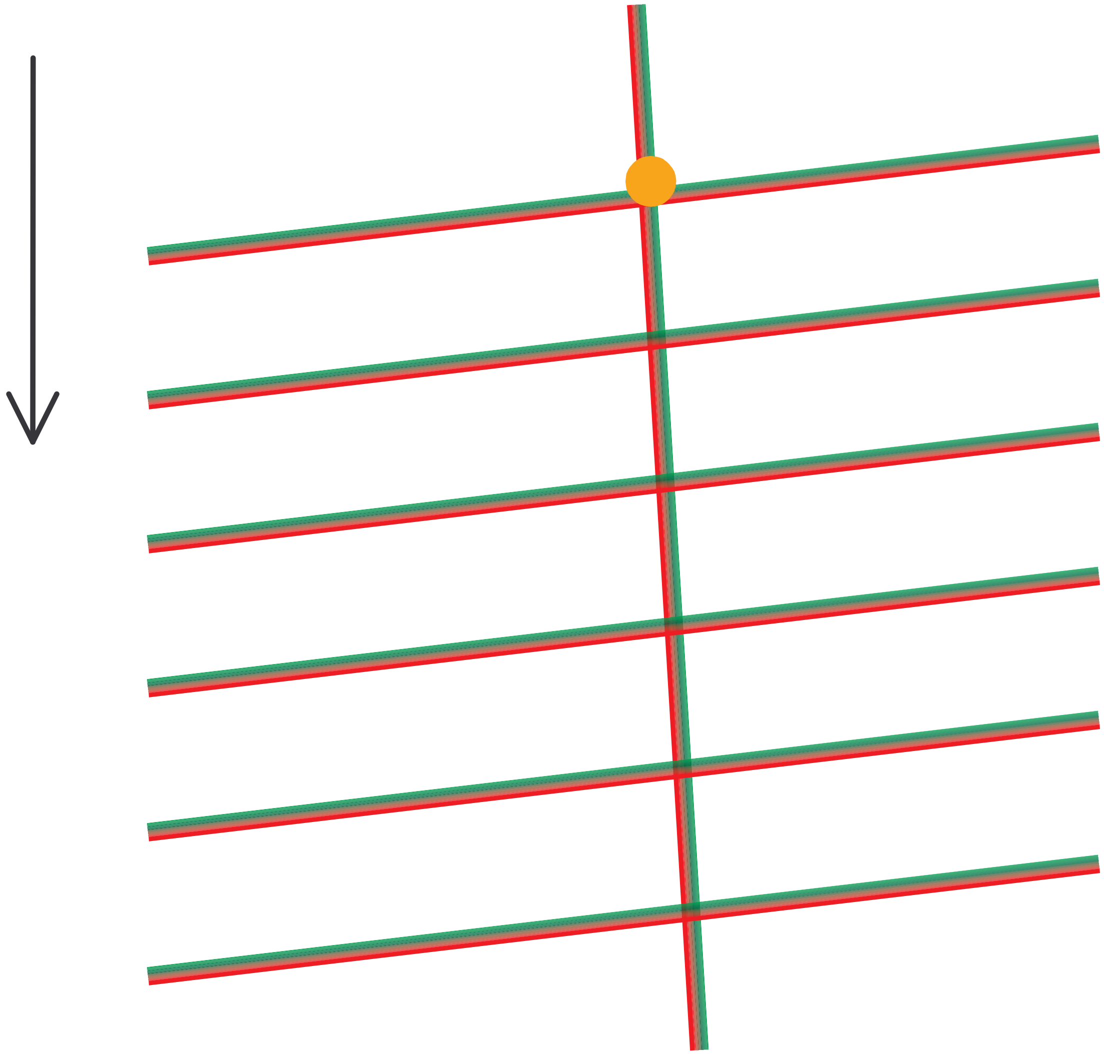






O(*n*) updates…

But…



Use a random permutation !

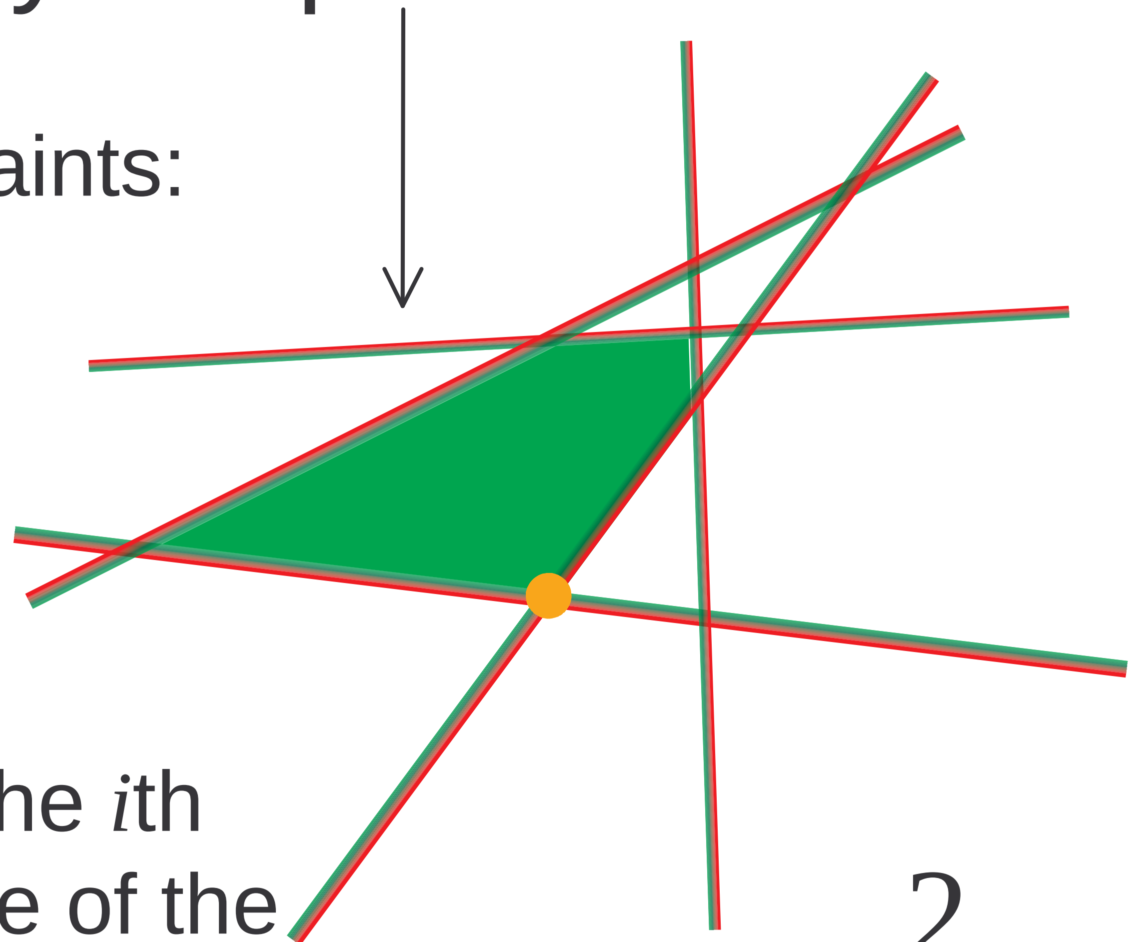
Expected time spent updating.

Let T_i be the time spent at time i.

$$E\left[\sum_{i=3}^{n} T_i\right] = \sum_{i=3}^{n} E[T_i] = \sum_{i=3}^{n} P(\text{Update at round } i)\,O(i)$$

Probability of update at round i

Fix first i constraints:



Update only if the ith constraint is one of the two *defining constraints*

$$P \leq \frac{2}{i-2}$$

Expected Run-Time Analysis

Expectation is over algorithm randomness, not over input

Expected time spent updating:

$$E\left[\sum_{i=3}^{n} T_i\right] = \sum_{i=3}^{n} E[T_i] = \sum_{i=3}^{n} P(\text{Update at round } i)\,O(i)$$

$$\leq \sum_{i=3}^{n} \frac{2}{i-2} O(i) = O(n)$$

What about d>2 ?

- Incrementally add new constraints
- Probability of update: *d/(i-d)*
- On update: solve *d-1* dimensional LP

$$T(d,n) \leq O(dn) + \sum_{i=d+1}^{n} \frac{d}{i-d} T(d-1, i-1)$$

$$T(d,n) = O(d!n)$$

Further results

- $O(d!n)$ is not optimal:
  - $O(d^2 n + d^2\, d!)$ [Clarkson]
  - $n^{O(\sqrt{d})}$ [Kalai, Matousek-Sharir-Welzl]
  - $O(d^2 n + d^{O(\sqrt{d})})$ [combined]
- Same time for finding minimum enclosing ball of n points
- First algorithms of this type were due to Meggido
- Weakly polynomial-time algorithms known [Khachiyan,Karmarkar]