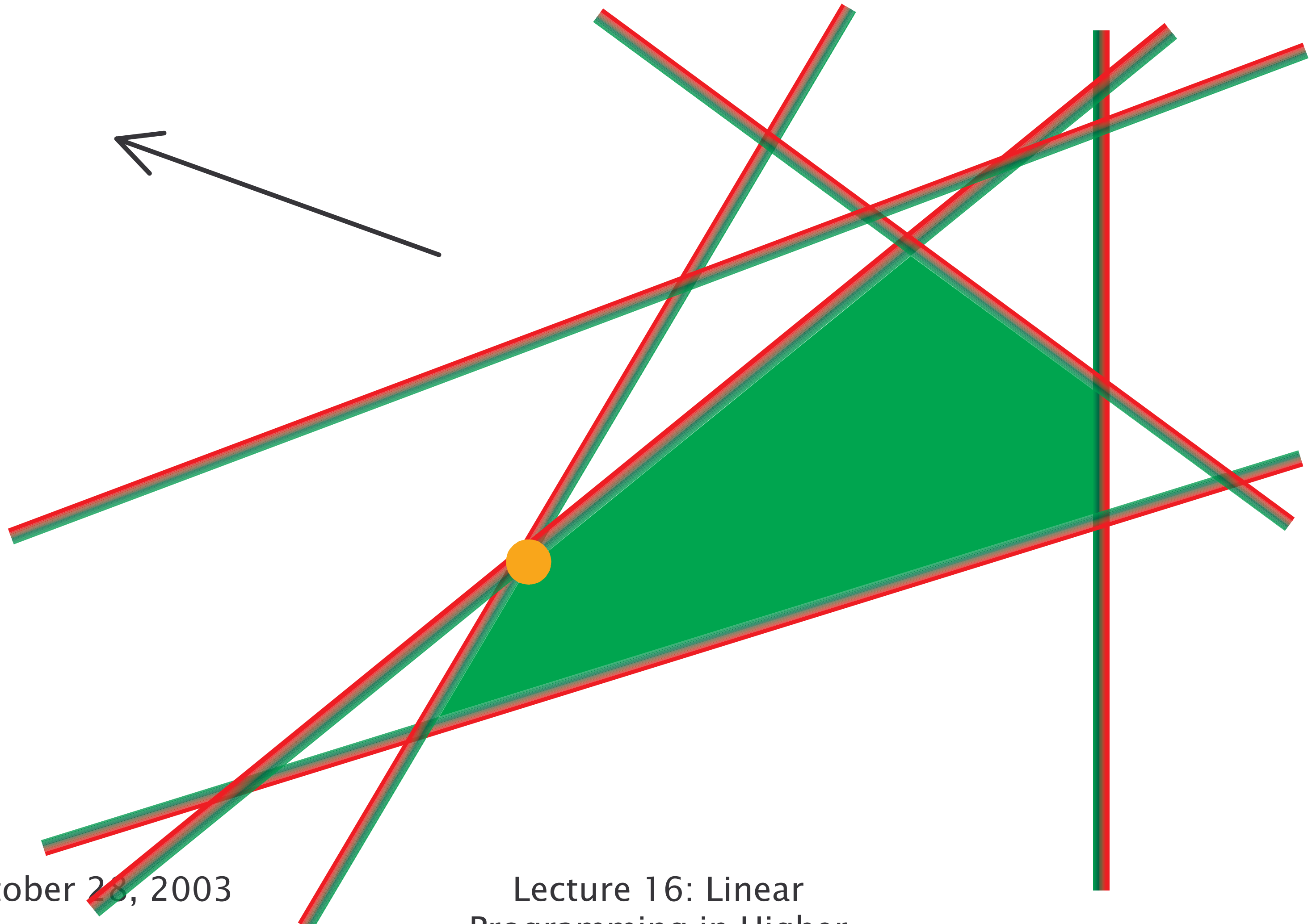
Linear Programming in Higher Dimensions

Piotr Indyk

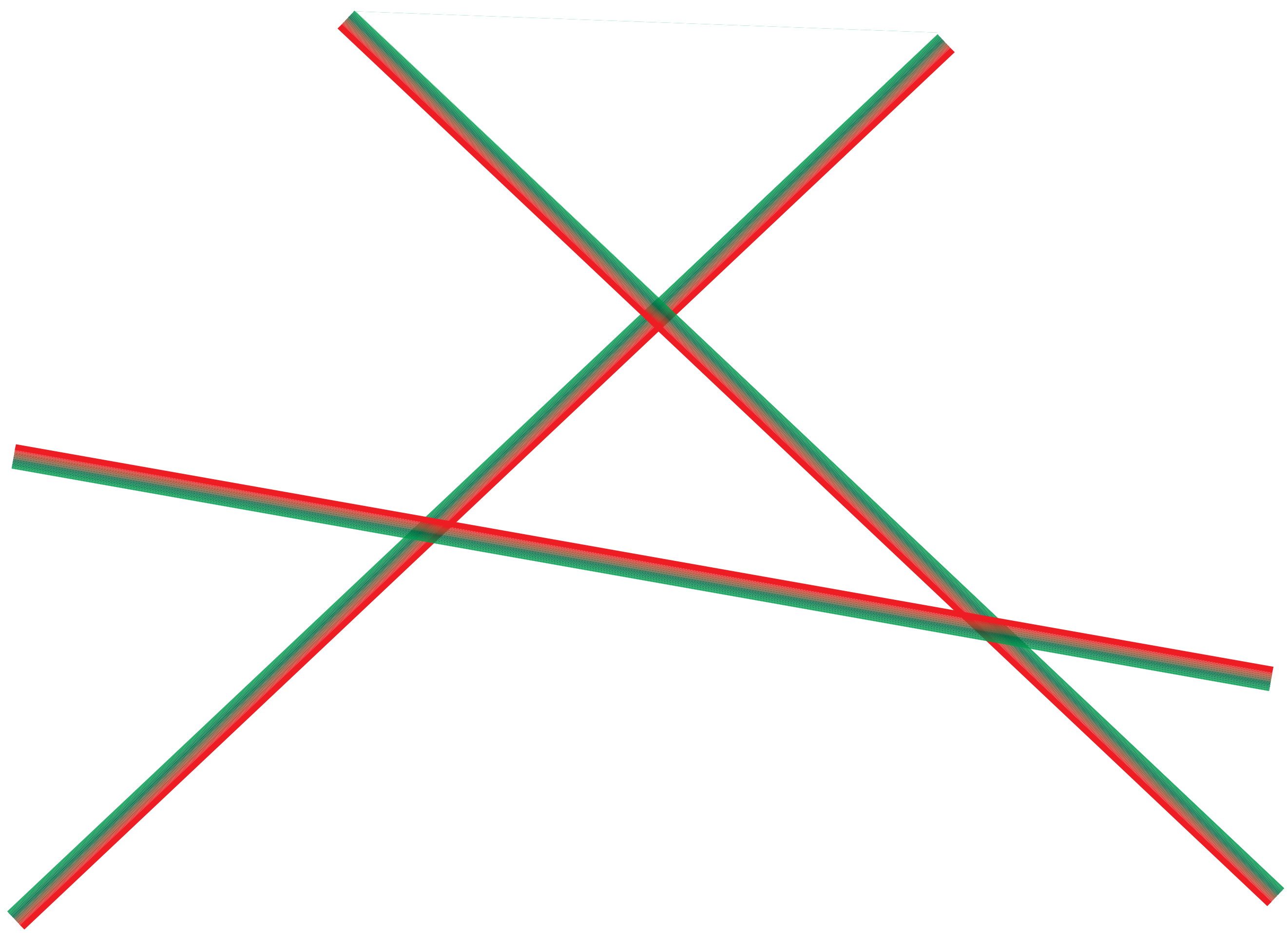
Linear Programming

$$\begin{aligned}
\text{Maximize:}\quad & c_1x_1 + c_2x_2 + \cdots + c_dx_d \\
\text{Subject to:}\quad & a_{1,1}x_1 + a_{1,2}x_2 + \cdots + a_{1,d}x_d \le b_1 \\
& a_{2,1}x_1 + a_{2,2}x_2 + \cdots + a_{2,d}x_d \le b_2 \\
& \vdots \qquad\qquad \ddots \qquad\qquad \vdots \\
& a_{n,1}x_1 + a_{n,2}x_2 + \cdots + a_{n,d}x_d \le b_n
\end{aligned}$$

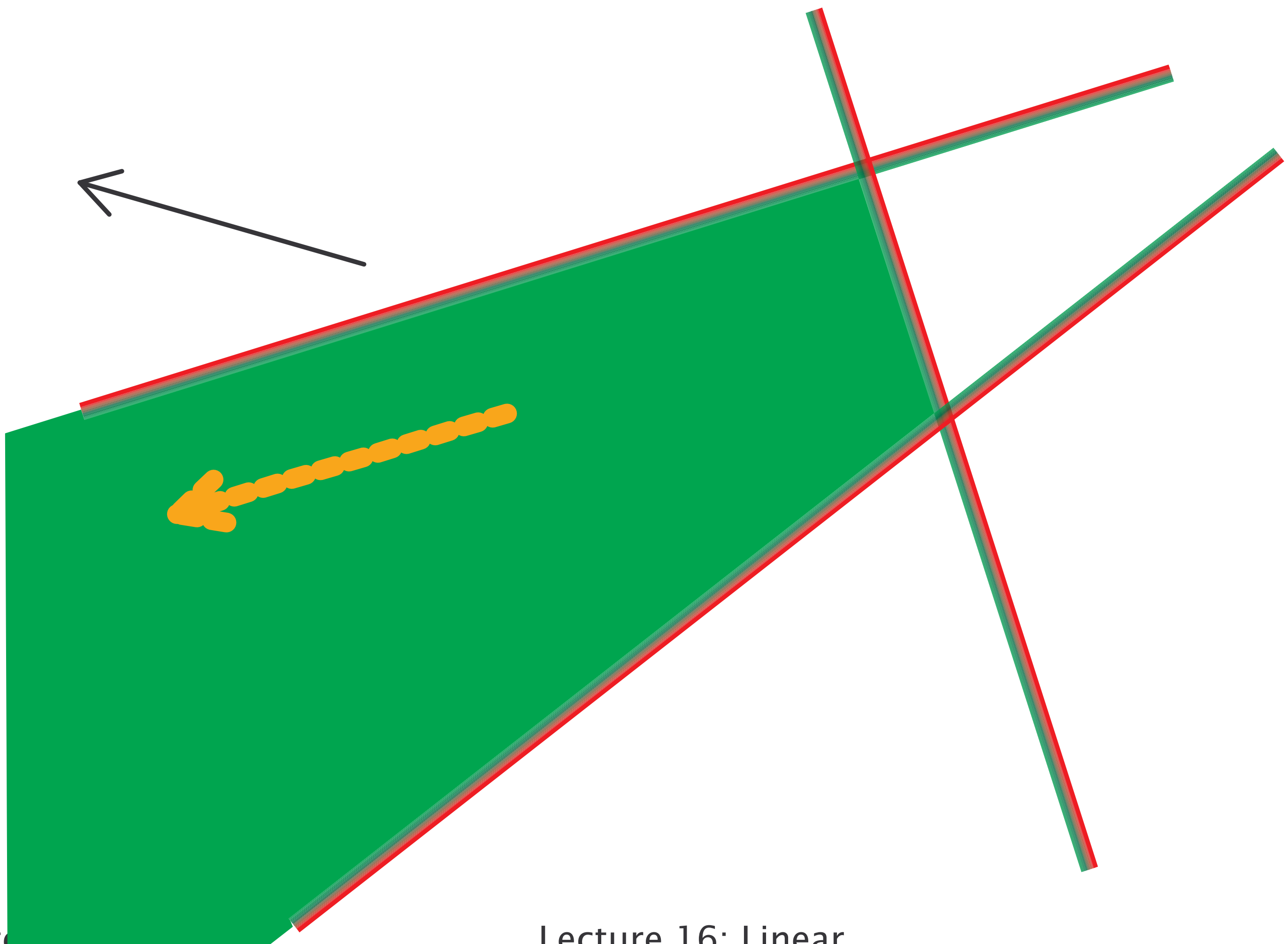
Linear Programming in 2D



An Infeasible Linear Program



An Unbounded LP

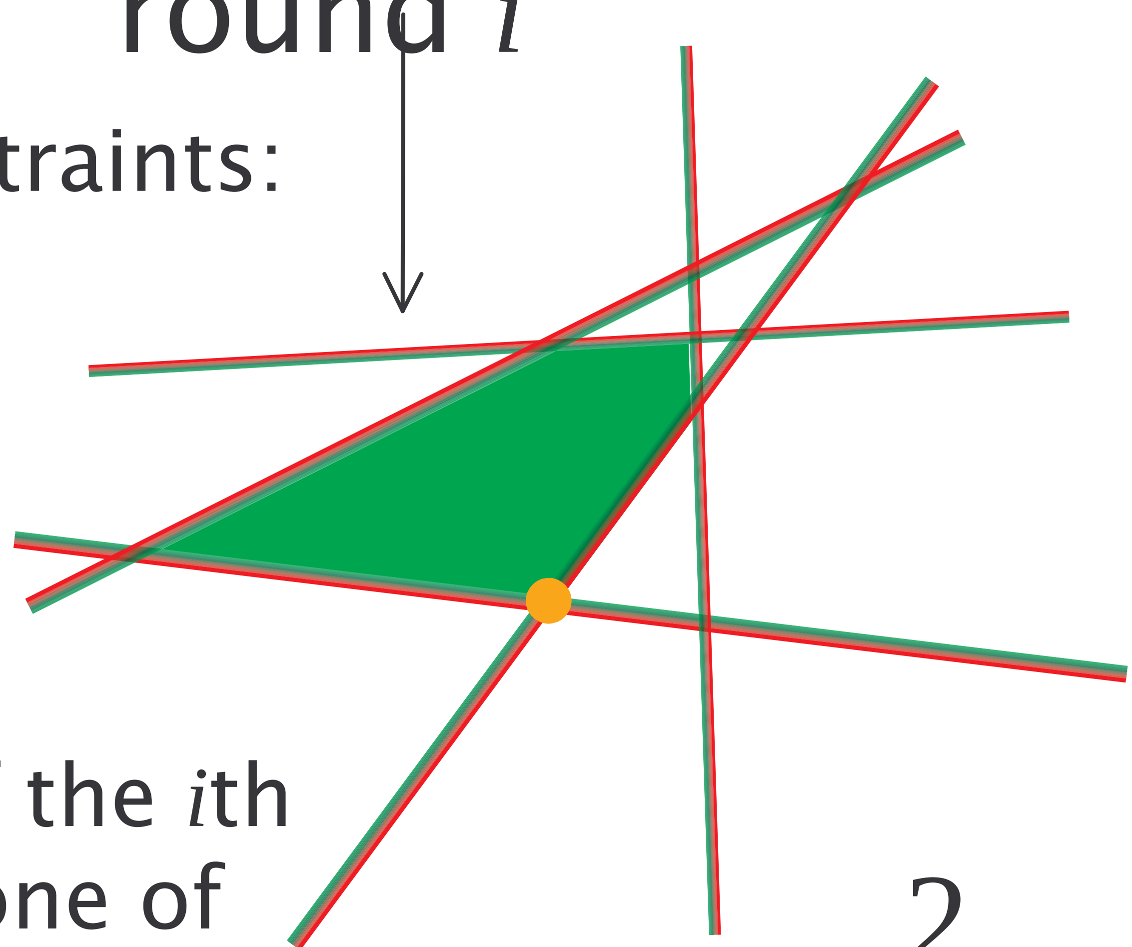


Incremental Algorithm

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

Probability of update at round i

Fix first i constraints:



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

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

Expected Run-Time Analysis

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

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

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

This Lecture

- $O(d!n)$ is not optimal:
  - $O(d^2 n + d^{O(1)} d!)$ [Clarkson]
  - A reduction from (n,d) –LP to a small number of $(O(d^2), d)$ –LP's
- Extensions:
  - $n^{O(\sqrt{d})}$ [Kalai, Matousek–Sharir–Welzl]
  - $O(d^2 n + d^{O(\sqrt{d})})$ [combined]

Notation

- H : set of n constraints
- $v(H)$: optimum subject to H
- A basis B for H: minimal set of constraints such that $v(B)=v(H)$
- We have $|B|=d$

Random Sampling I

- SolveLP1(H):
  - $G=\emptyset$
  - Repeat:
    - R=random subset of H, $|R|=r$
    - v=SolveLP($G+R$)
    - V=set of constraints in H violated by v
    - If $|V| \le t$, then $G=G+V$
  - Until $V=\emptyset$
- Correctness ?
- Running time analysis ?

Analysis

- Each time we augment G, we add to G a new constraint from the basis B of H
  - If v did not violate any constraint in B, it would be optimal
  - So V must contain an element from B, which was not in G earlier
- We can augment G at most d times
- The number of constraints in the recursive call is $|R|+|G| \leq r + dt$
- What is the probability of augmentation ?

Sampling Lemma

Lemma: The expected number of constraints V that violate $v(G+R)$ is at most nd/r.

Proof:

- Define a 0/1 random variable $d(R,h)$, which is $=1$ iff h violates $v(G+R)$
- Need to bound

$$E_R [\Sigma_h d(R,h)] = \Sigma_{|R|=r} \Sigma_h d(R,h) \;/\; \#R$$

$$= \Sigma_{|Q|=r+1} \Sigma_{h\in Q} d(Q-\{h\},h) \;/\; \#R$$

$$= [\#Q * (r+1) \;/\; \#R\,] * Pr_{Q,h\in Q} [d(Q-\{h\},h)]$$

$$\leq n * d/(r+1)$$

Analysis

- $t=2nd/r$ → expected # iterations per augmentation is constant
- Number of constraints in the recursive call is $r+O(d^2n/r) = O(r)$ for $r=d\ n^{1/2}$
- Total expected time

$$T_{LP1}(n) \leq 2d\ T_{LP}(d\ n^{1/2})+O(d^2\ n)$$

Analysis ctd.

- Can use Seidel's algorithm for LP
- This gives us $O(d^2n+d*d\ n^{1/2}\ d!)$
- We get better time if LP=LP2
- Idea: reduce the sample size

Random Sampling II

- SolveLP2(H):
  - $G=\emptyset$
  - Repeat:
    - R=random subset of H, $|R|=r$
    - v=SolveLP(R)
    - V=multiset of constraints in H violated by v
    - If $|V| \leq t$, then $H=H+V$
  - Until $V=\emptyset$
- As before, set $t=2*|H|d/r$

  → augmentation performed with prob. $>1/2$

Need to bound #augmentations

- Fix a basis B for H
- On the one hand:
  - In one iteration, the multiplicity of at least one constraint in B is doubled
  - In kd iterations, $|B| \geq 2^k$
- On the other hand:
  - In one iteration, $|H|$ increases by $\leq 2 |H| d/r$
  - After kd iterations:

    $|B| \leq |H| \leq n (1+2d/r)^{kd} \leq n \exp(2kd^2/r) = n \exp(2d^2/r)^k$
- Therefore, the total number of iterations is $O(dk)$, if k such that $2^k > n \exp(2d^2/r)^k$

Analysis, ctd.

$$2^k > n \exp(2d^2/r)^k$$

- Set $r=4d^2 \rightarrow 2^k > n\,(e^{1/2})^k$
- We get $k = O(\log n)$
- The total #iterations is $O(d \log n)$

Total Time

- The expected time
  - $T_{LP2}(n) = d \log n \, [\, T_{LP}(4d^2) + dn \,]$
- Plug in Seidel into LP2
  - $T_{LP2}(n) = O(\, d \log n \;\; (d^2\, d! + dn)\,)$
- Plug in LP2 into LP1
  - $T_{LP1}(n)=O(2d\, [d \log n\, (d^2\, d! + d^2 n^{1/2})] + d^2\, n)$
- After some cleaning
  - $T_{LP1}(n)=O(d^4\, d! \log n + d^2\, n)=O(d^5\, d! + d^2 n)$