Linear Programming
4 march 2004

Lecture 8, CS 4235

Antoine Vigneron

antoine@comp.nus.edu.sg

National University of Singapore

News

- Midterm 2 on Thursday 11 march (next week)
- at lecture time slot, SR1
- lecture will be on Friday, at tutorial time slot
- covers lectures 1–8
- emphasis on lectures 5–8
- new:
  - open book: you can bring *any written material* you want

Outline

- linear programming, definition
- one dimensional case
- a randomized algorithm in two dimension
- generalization to any dimension
- reference
  - textbook chapter 4
  - D. Mount's lecture 8,9, and 10
- to know more about polytopes
  - Matousek's book *Lectures on discrete geometry*
  - Ziegler's book *Lectures on polytopes*
- deterministic algorithm
  - not covered this year, in last year's lecture 4

Example

Example

- you can build two kinds of house: X and Y
- a house of type X requires 10,000 bricks, 4 doors and 5 windows
- a house of type Y requires 8,000 bricks, 2 doors and 10 windows
- a house X can be sold \$200,000 and a house Y can be sold \$250,000
- you have 168,000 bricks, 60 doors and 150 windows
- how many houses of each type should you build so as to maximize their total price?

Formulation

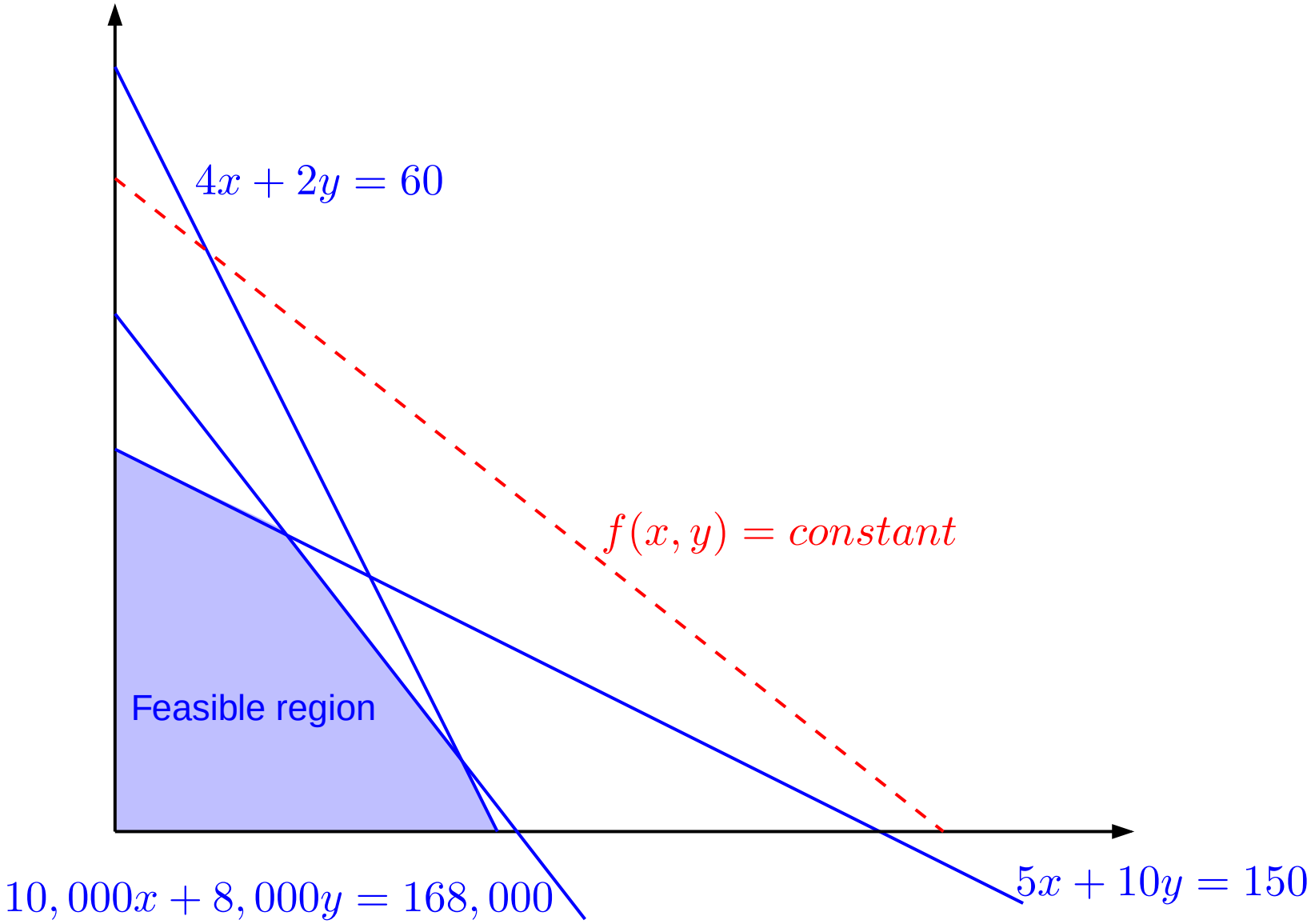
- x (resp. y) denotes the number of houses of type X (resp. Y).
- maximize the price

$$f(x, y) = 200,000x + 250,000y$$

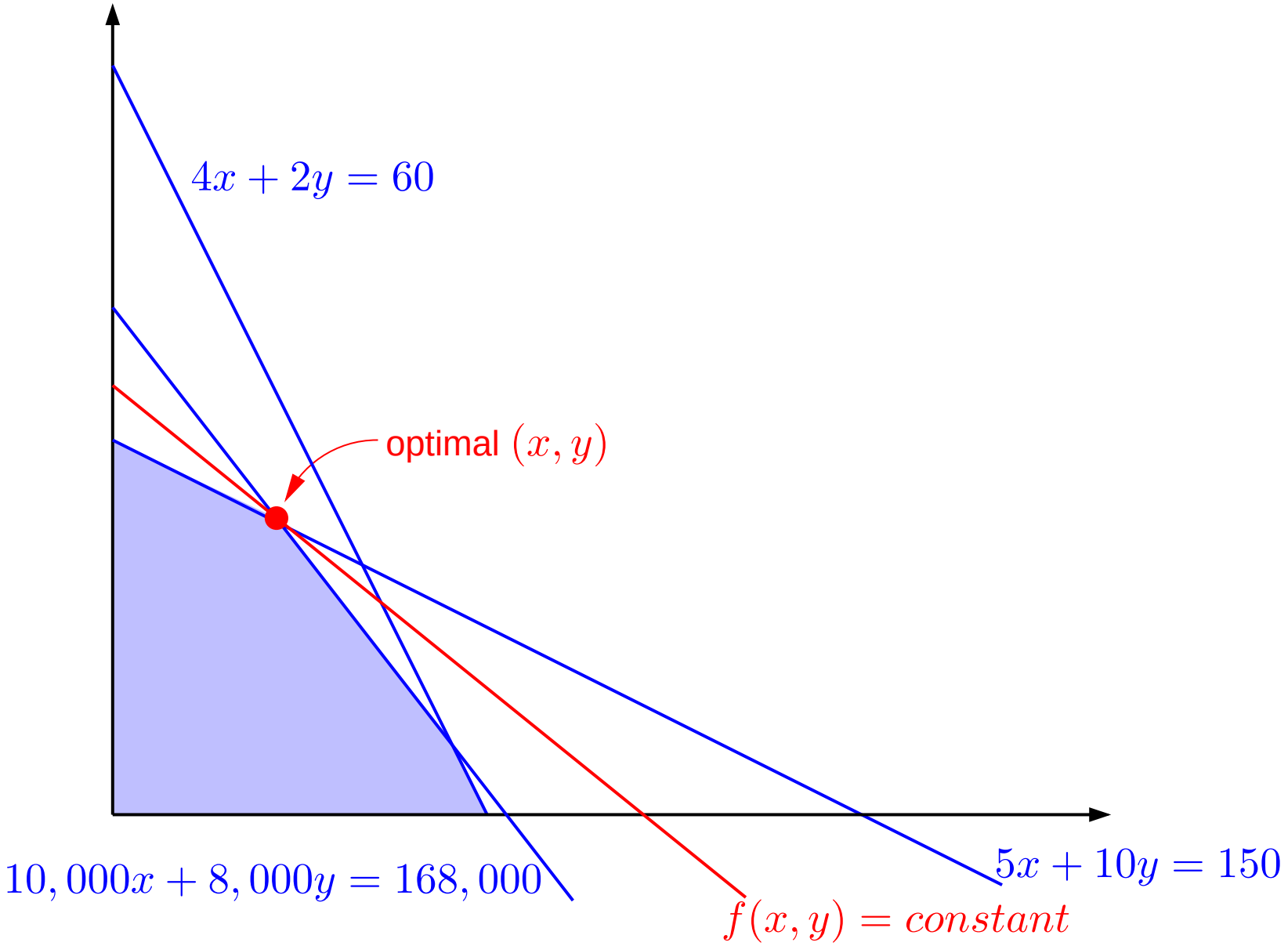
- under the *constraints*

$$\begin{array}{rcrcr} -x & & & \leq & 0 \\ & & -y & \leq & 0 \\ 10,000x & + & 8,000y & \leq & 168,000 \\ 4x & + & 2y & \leq & 60 \\ 5x & + & 10y & \leq & 150 \end{array}$$

Geometric interpretation



Geometric interpretation



Solution

- from previous slide, at the optimum
  - $x = 8$
  - $y = 11$
- luckily these are integers
- so it is the solution to our problem
- if we add the constraint that all variables are integers, we are doing *integer programming*
  - we do not deal with it in CS4235
  - we consider only linear inequalities, no other constraint
- our example was a special case where the linear program has an integer solution, hence it is also a solution to the integer program

Introduction

Problem statement

- maximize the *objective function*

$$f(x_1, x_2 \ldots x_d) = c_1 x_1 + c_2 x_2 + \ldots + c_d x_d$$

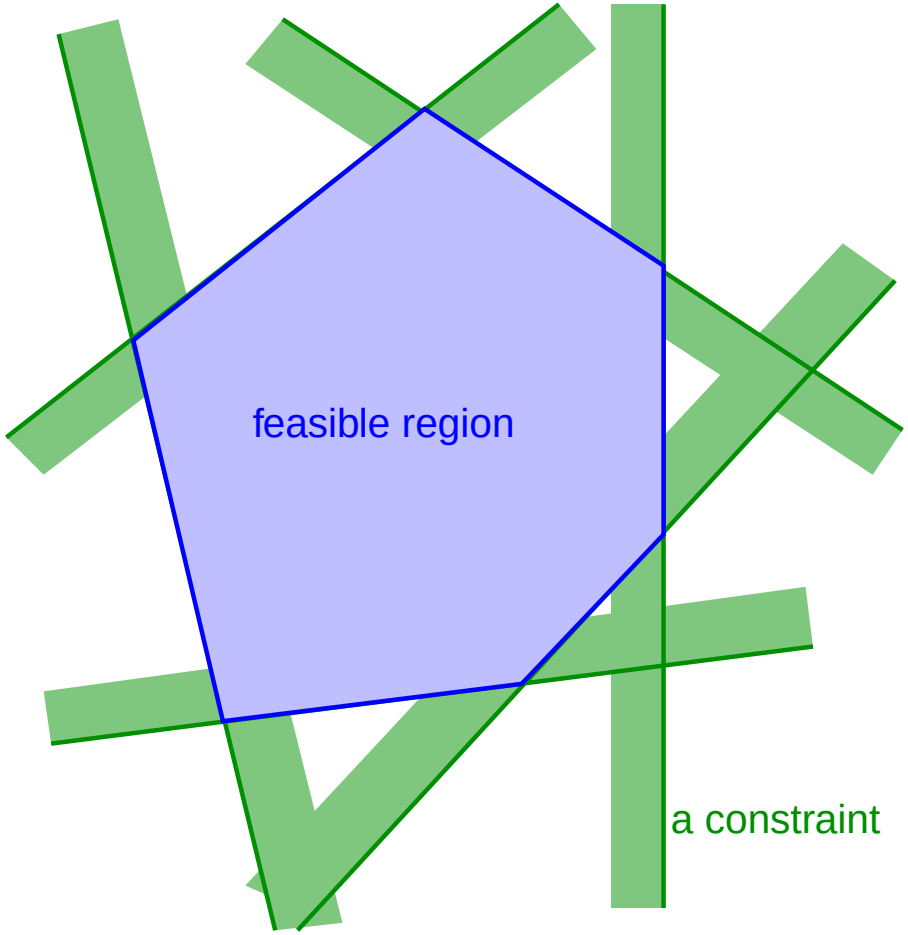
  under the *constraints*

$$\begin{array}{rcrcl} a_{1,1}x_1 & + \ldots + & a_{1,d}x_d & \leq & b_1 \\ a_{2,1}x_1 & + \ldots + & a_{2,d}x_d & \leq & b_2 \\ \vdots & & \vdots & & \vdots \\ a_{n,1}x_1 & + \ldots + & a_{n,d}x_d & \leq & b_n \end{array}$$

- this is linear programming in dimension d

Geometric interpretation

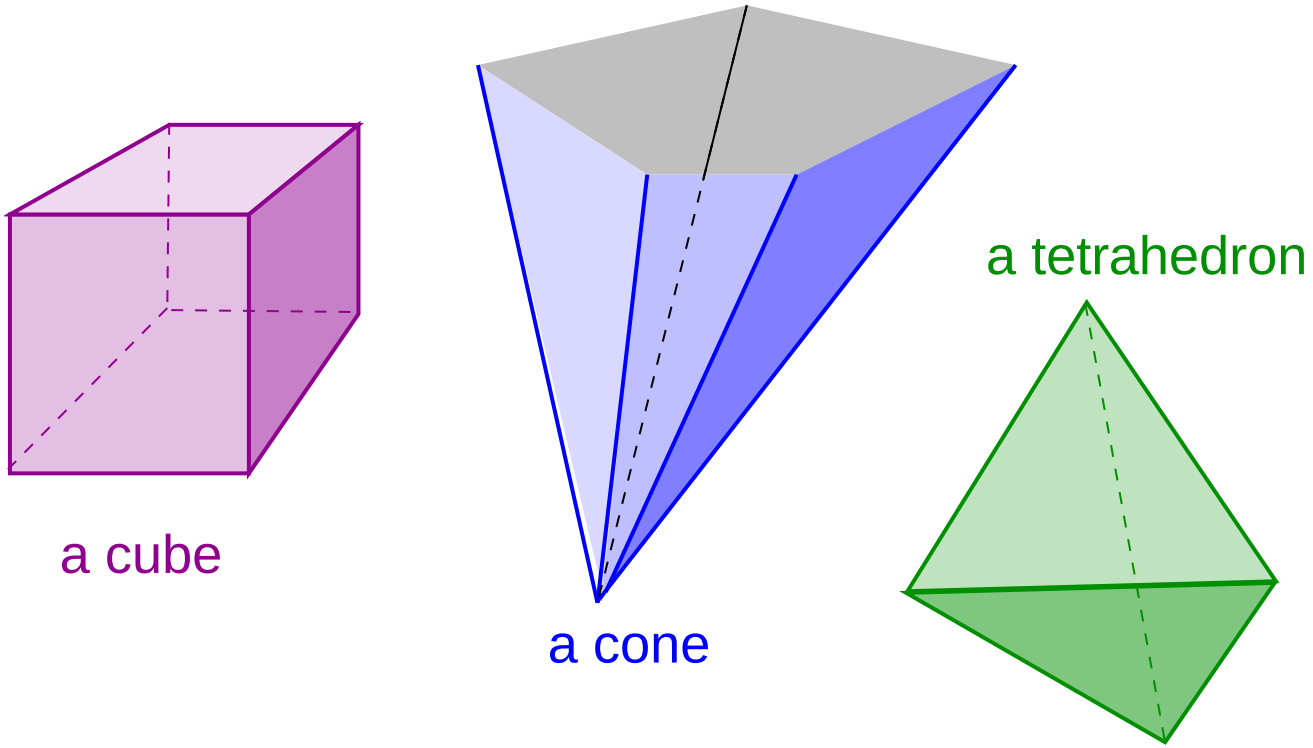
- each constraint represents a half space in $\mathbb{R}^d$
- the intersection of these half–spaces forms the *feasible region*
- the feasible region is a *convex polyhedron* in $\mathbb{R}^d$



Convex polyhedra

- definition: a convex polyhedron is an intersection of half spaces in $\mathbb{R}^d$
- it is not necessarily bounded
- a bounded convex polyhedron is a *polytope*
- special case: a polytope in $\mathbb{R}^2$ is a convex polygon

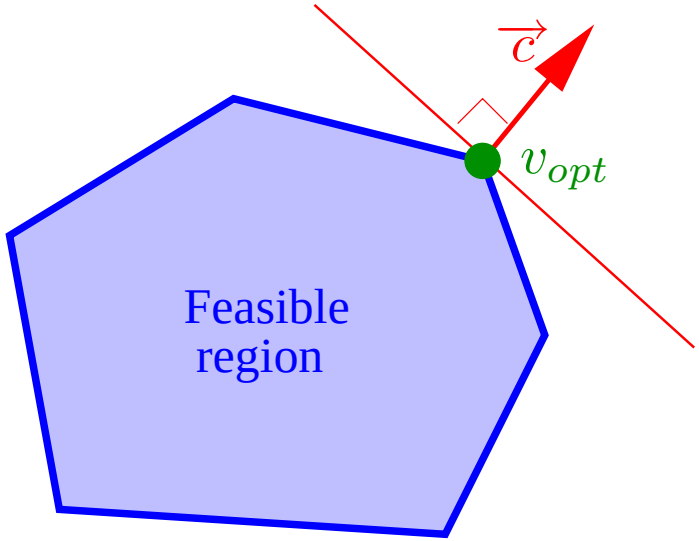
Convex polyhedra in $\mathbb{R}^3$



- faces of a convex polyhedron in $\mathbb{R}^3$
  - vertices, edges and facets
  - example: a cube has 8 vertices, 12 edges and 6 facets

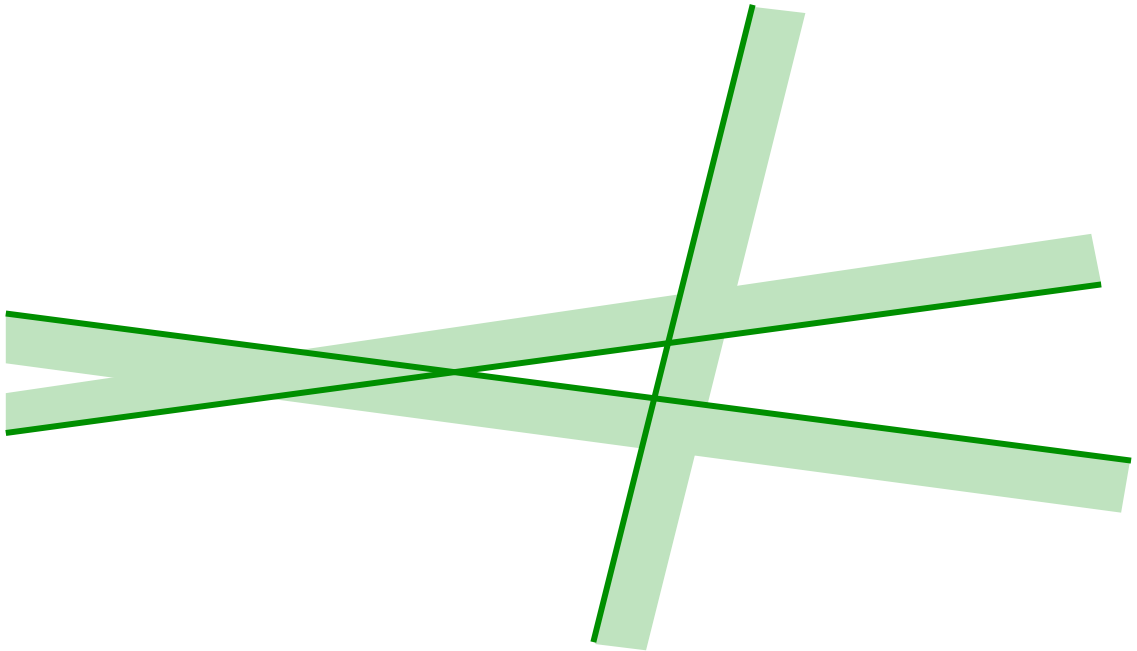
Geometric interpretation

- let $\overrightarrow{c} = (c_1, c_2, \ldots c_d)$
- we want to find a point v_{opt} of the feasible region such that $\overrightarrow{c}$ is the outer normal at v_{opt} if there is one



Infeasible linear programs

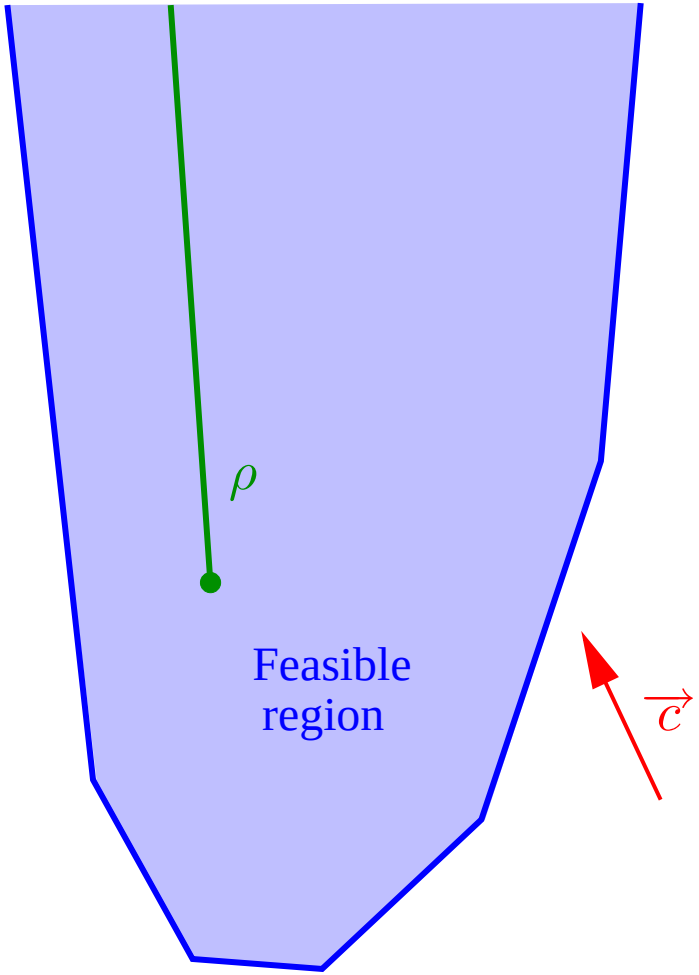
- the feasible region can be empty



- in this case there is no solution to the linear program
- the program is said to be *infeasible*
- we would like to know when it is the case

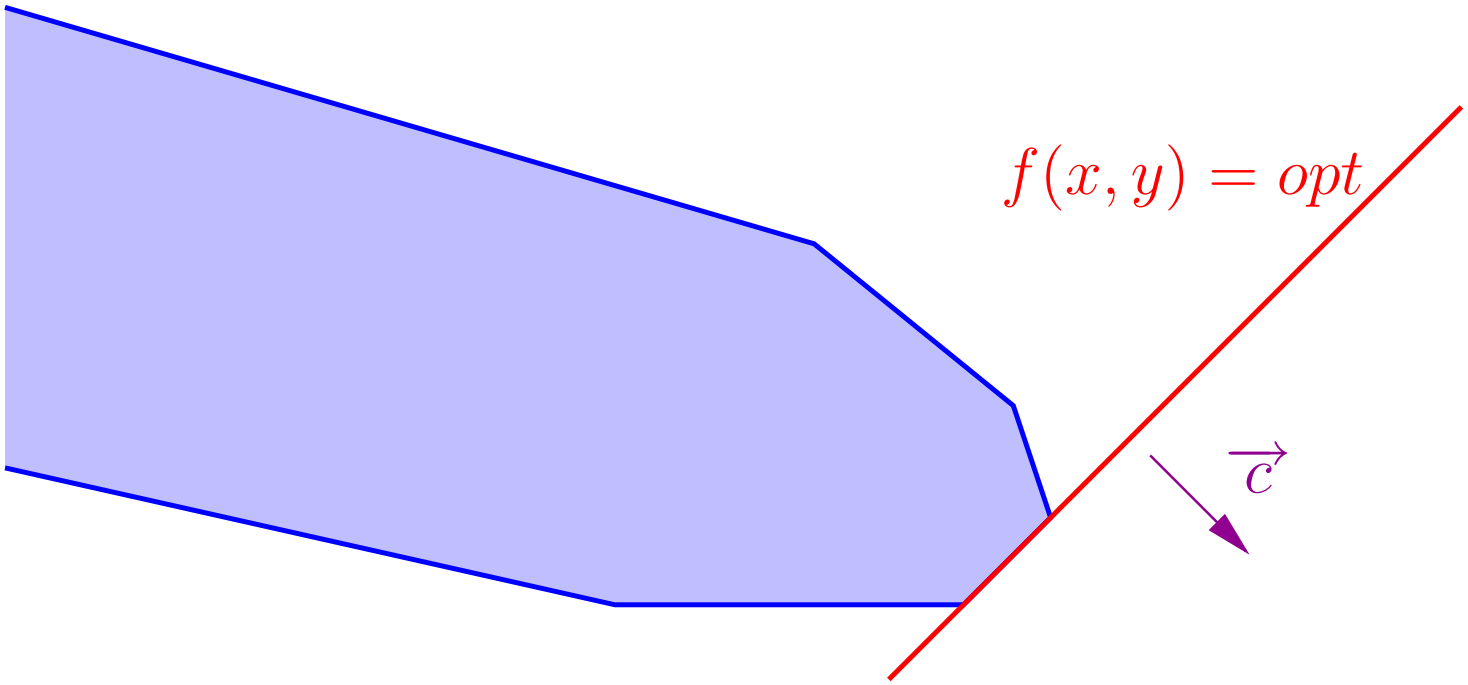
Unbounded linear programs

- the feasible region may be unbounded in the direction of $\overrightarrow{c}$
- the linear program is called *unbounded*
- in this case, we want to return a ray ρ in the feasible region along which f takes arbitrarily large values



Degenerate cases

- a linear program may have an infinite number of solution



- in this case, we report only one solution

Background

- linear programming is one of the most important problems in operations research
- many optimization problems in engineering and in economics are linear programs
- a practical algorithm: the simplex algorithm
  - people used it without computers
  - exponential time in the worst case
- there are polynomial time algorithms
  - ellipsoid method, interior point method
- integer programming is NP–hard

Background

- computational geometry techniques give good algorithms in low dimension
  - running time is $O(n)$ when d is constant
  - but exponential in d: running time $O(3^{d^2} n)$
- this lecture: Seidel's algorithm
  - simple, randomized
  - expected running time $O(d!n)$
  - this is $O(n)$ when $d = O(1)$
  - in practice, very good for low dimension

One dimensional case

Formulation

- maximize the *objective function*

$$f(x) = cx$$

under the *constraints*

$$\begin{aligned} a_1 x &\leq b_1 \\ a_2 x &\leq b_2 \\ \vdots &\quad \vdots \\ a_n x &\leq b_n \end{aligned}$$

Interpretation

- if $a_i > 0$ then constraint i corresponds to the interval

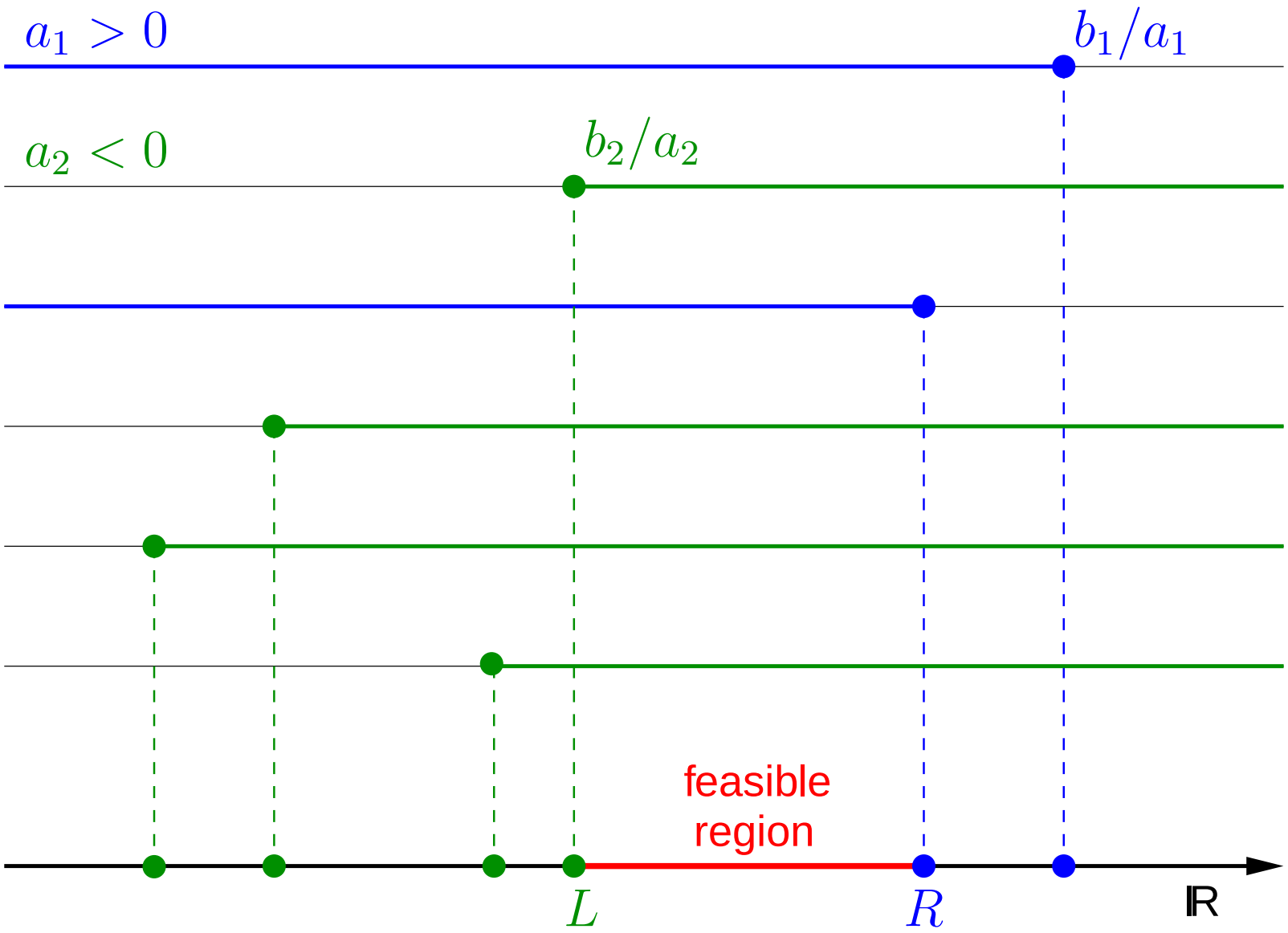
$$\left(-\infty, \frac{b_i}{a_i}\right]$$

- if $a_i < 0$ then constraint i corresponds to the interval

$$\left[\frac{b_i}{a_i}, \infty\right)$$

- the feasible region is an intersection of intervals
- so the feasible region is an interval

Interpretation



Algorithm

- assume there is (i_1, i_2) such that $a_{i_1} < 0 < a_{i_2}$
- compute

$$R = \min_{a_i > 0} \frac{b_i}{a_i}$$

- compute

$$L = \max_{a_i < 0} \frac{b_i}{a_i}$$

- it takes $O(n)$ time
- if $L > R$ then the program in infeasible
- otherwise
  - if $c > 0$ then the solution is $x = R$
  - if $c < 0$ then the solution is $x = L$

Algorithm

- assume $a_i > 0$ for all i
  - compute $R = \min \frac{b_i}{a_i}$
  - if $c > 0$ then the solution is $x = R$
  - if $c < 0$ then the program is unbounded and the ray $(-\infty, R]$ is a solution
- assume $a_i < 0$ for all i
  - compute $L = \max \frac{b_i}{a_i}$
  - if $c < 0$ then the solution is $x = L$
  - if $c > 0$ then the program is unbounded and the ray $[L, \infty)$ is a solution

Linear programming in $\mathbb{R}^2$

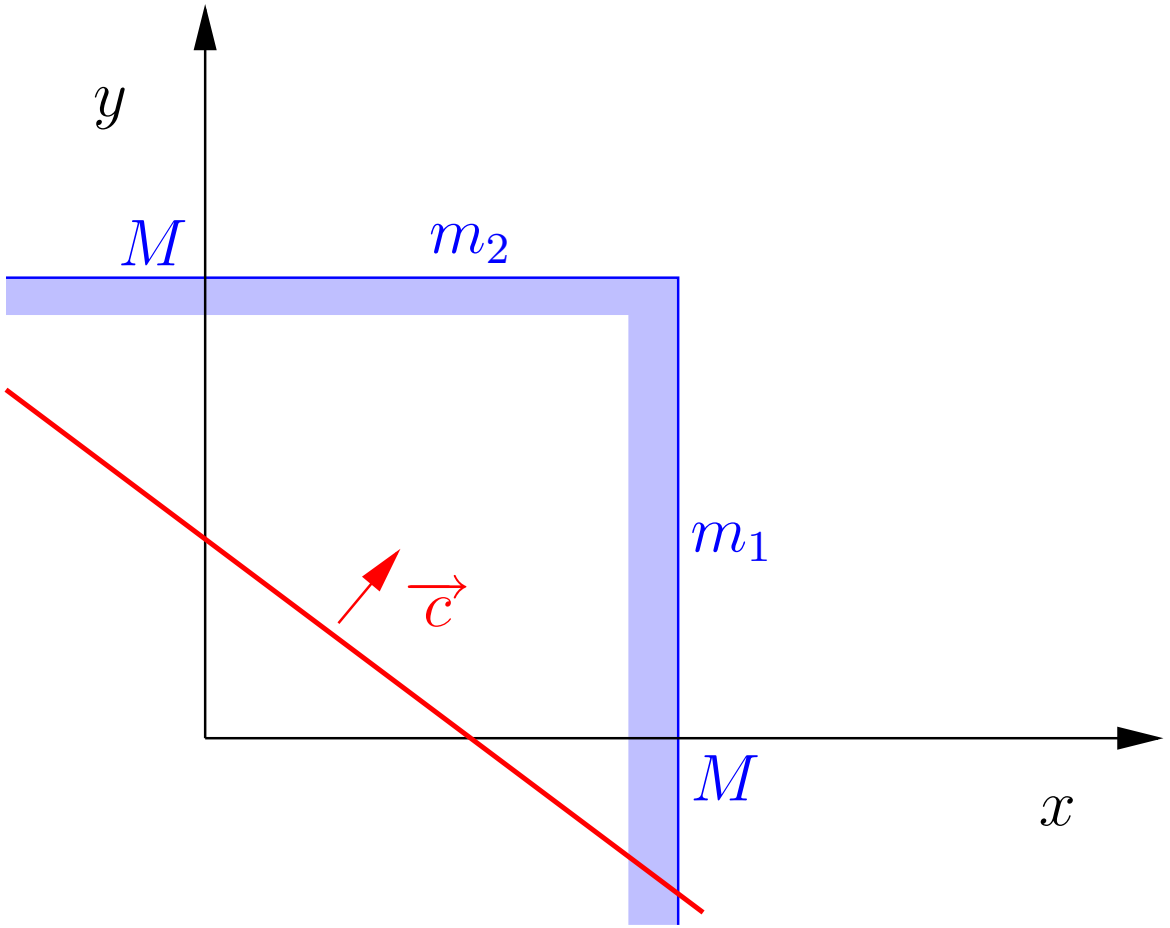
First approach

- compute the feasible region
  - in $O(n \log n)$ time by divide and conquer+plane sweep
  - other method: see later, lecture on duality
- the feasible region is a convex polyhedron
- find an optimal point
  - can be done in $O(\log n)$ time (see tutorial 1)
- overall, it is $O(n \log n)$ time
- this lecture: an expected $O(n)$ time algorithm

Preliminary

- we only consider bounded linear programs
- we make sure that our linear program is bounded by enforcing two additional constraints m_1 and m_2
  - objective function: $f(x, y) = c_1 x + c_2 y$
  - let M be a large number
  - if $c_1 \geq 0$ then m_1 is $x \leq M$
  - if $c_1 \leq 0$ then m_1 is $x \geq -M$
  - if $c_2 \geq 0$ then m_2 is $y \leq M$
  - if $c_2 \leq 0$ then m_2 is $y \geq -M$
- in practice, it often comes naturally
- for instance, in our first example, it is easy to see that $M = 30$ is sufficient

New constraints

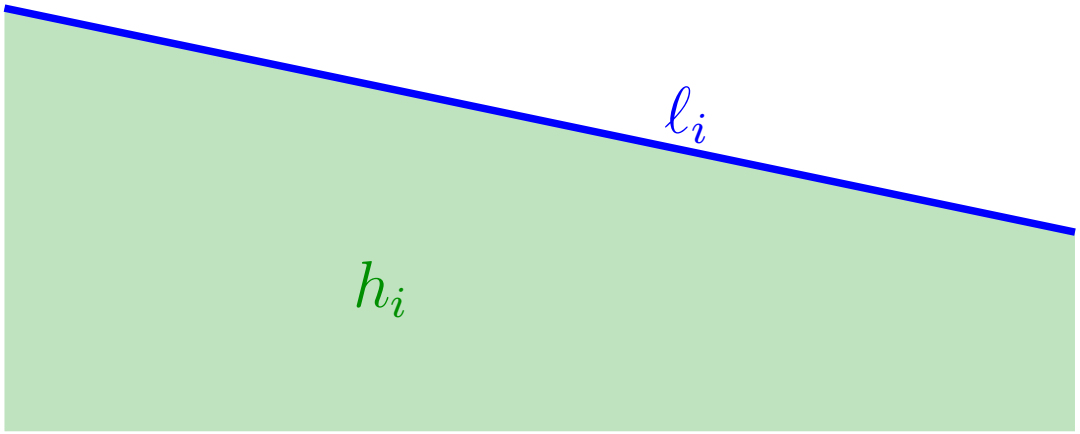


Notation

- the i–th constraint is

$$a_{i,1}x + a_{i,2}y \leq b_i$$

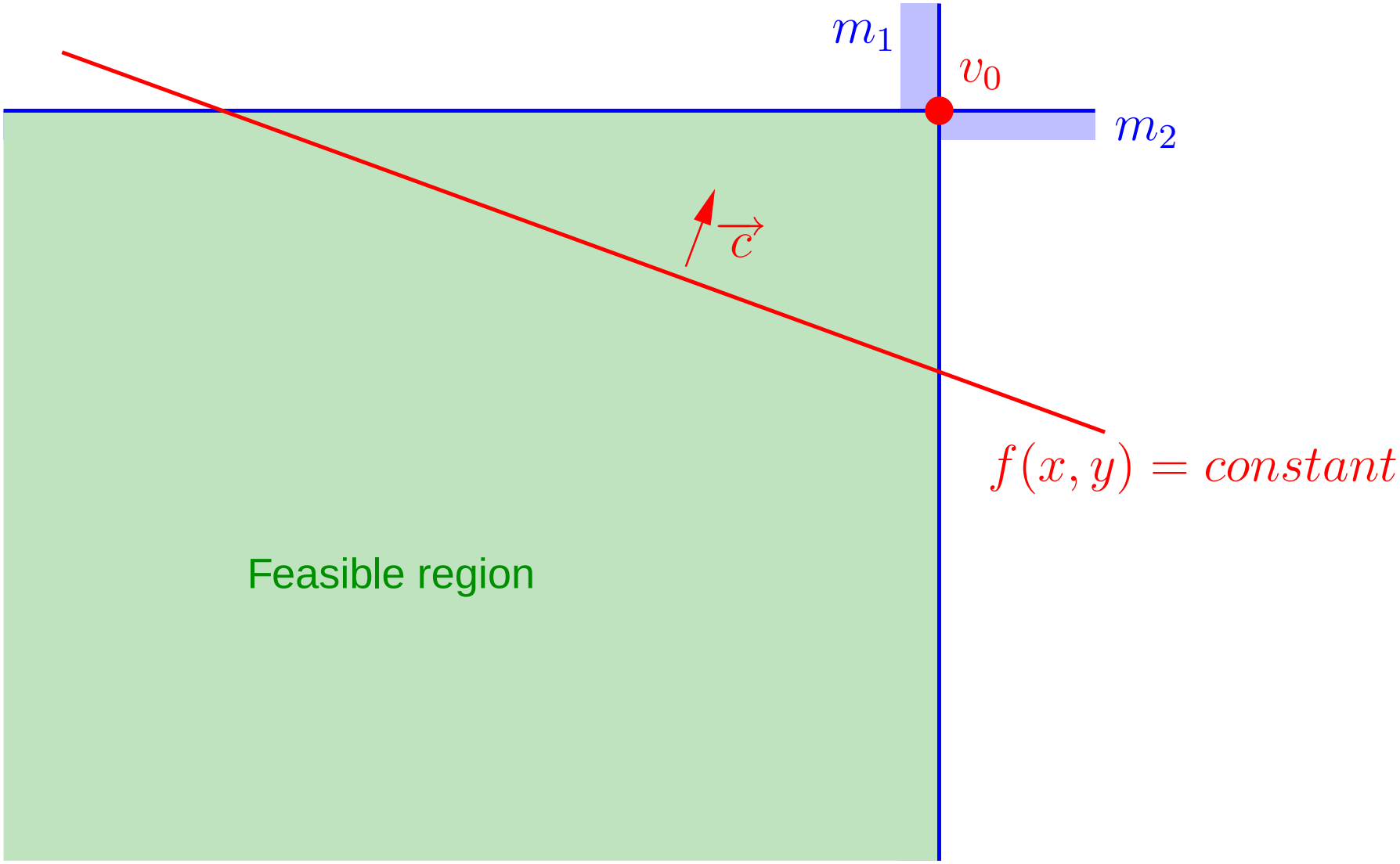
  it defines an half–plane h_i
- ℓ_i is the line delimiting h_i



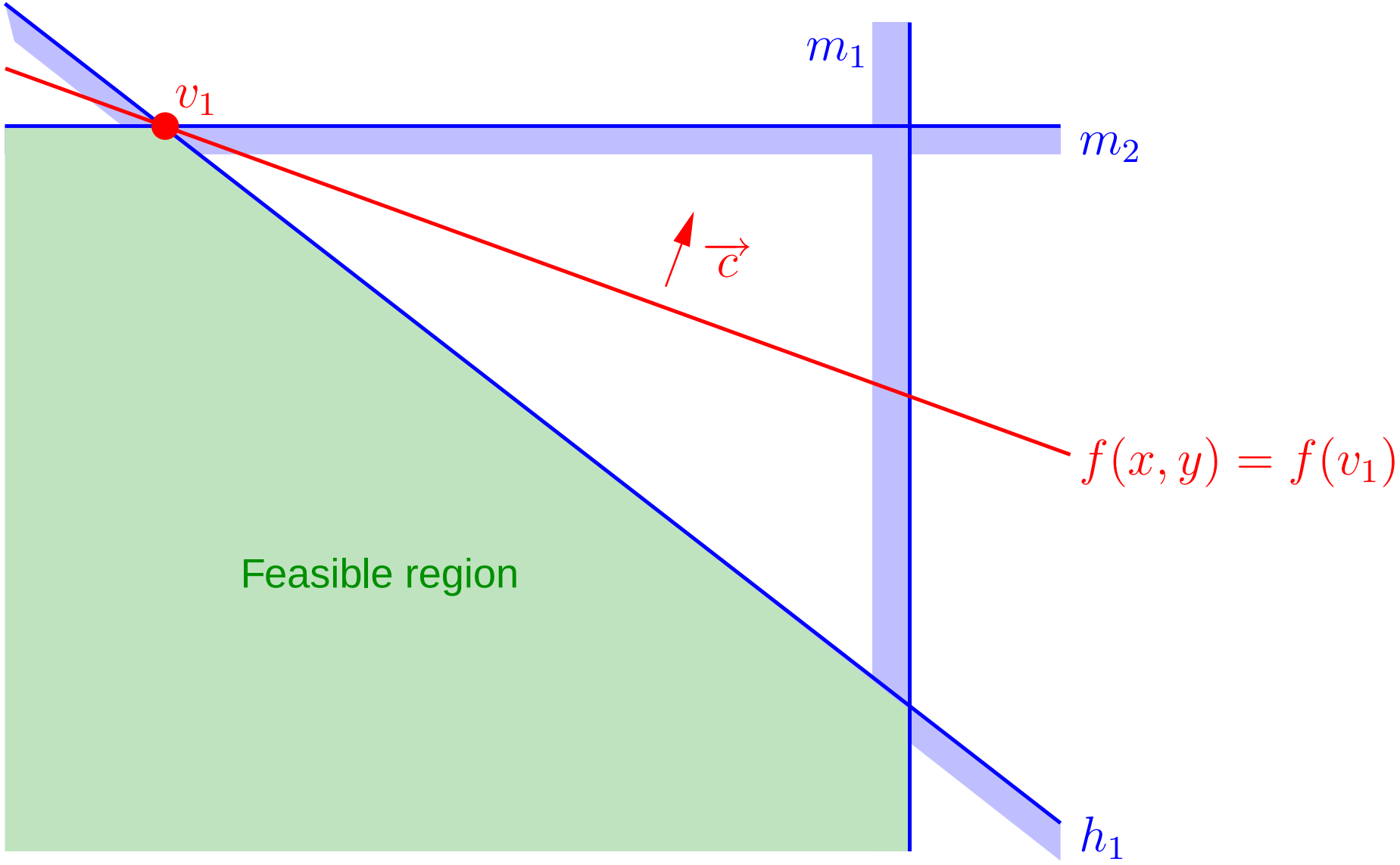
Algorithm

- a randomized incremental algorithm
- we first compute a random permutation of the constraints $(h_1, h_2 \dots h_n)$
- we denote $H_i = \{m_1, m_2, h_1, h_2 \dots h_i\}$
- we denote by v_i a vertex of $\bigcap H_i$ that maximizes the objective function
  - in other words, v_i is a solution to the linear program where we only consider the first i constraints
- v_0 is simply the vertex of the boundary of $m_1 \cap m_2$
- idea: knowing v_{i-1}, we insert h_i and find v_i

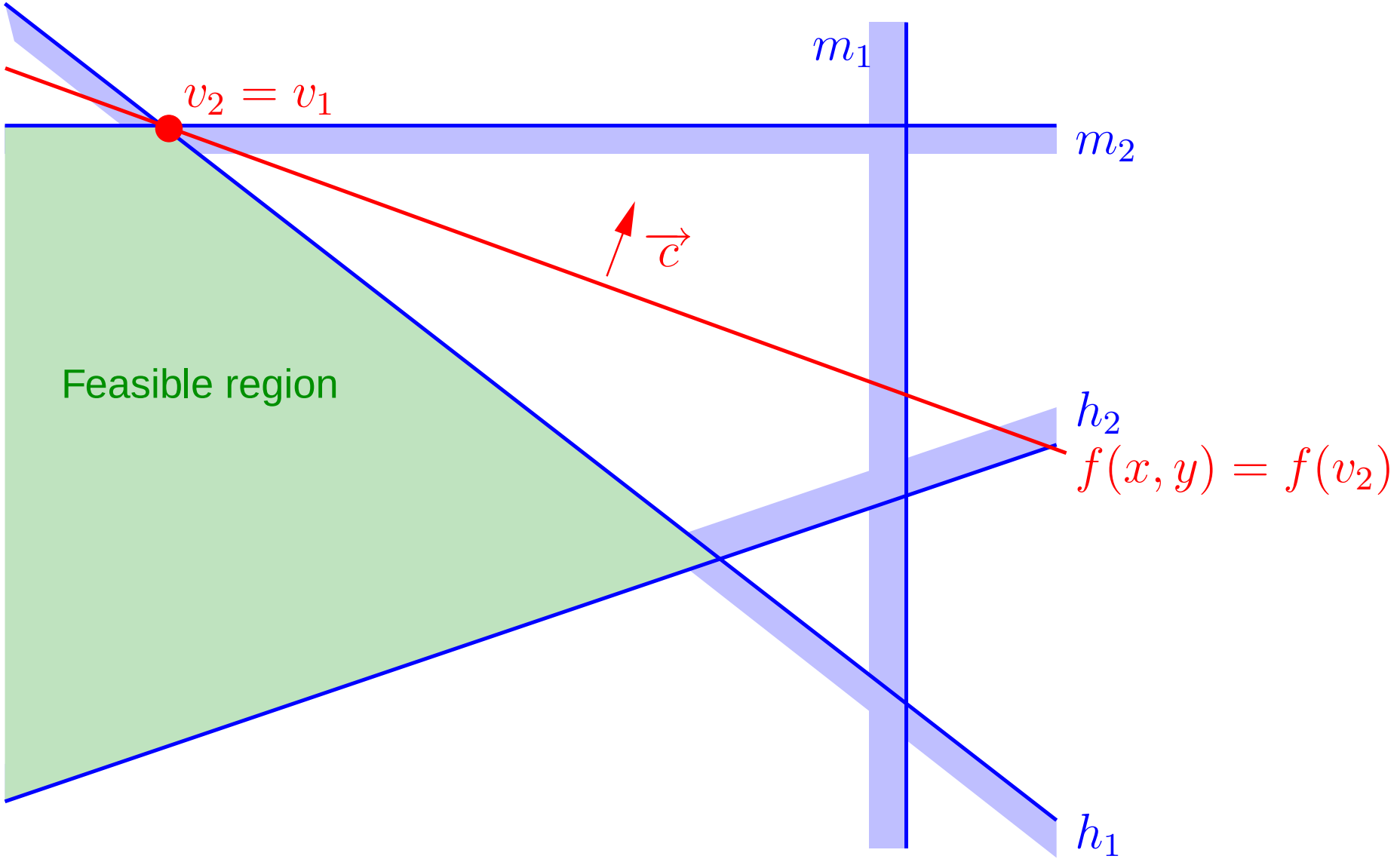
Example



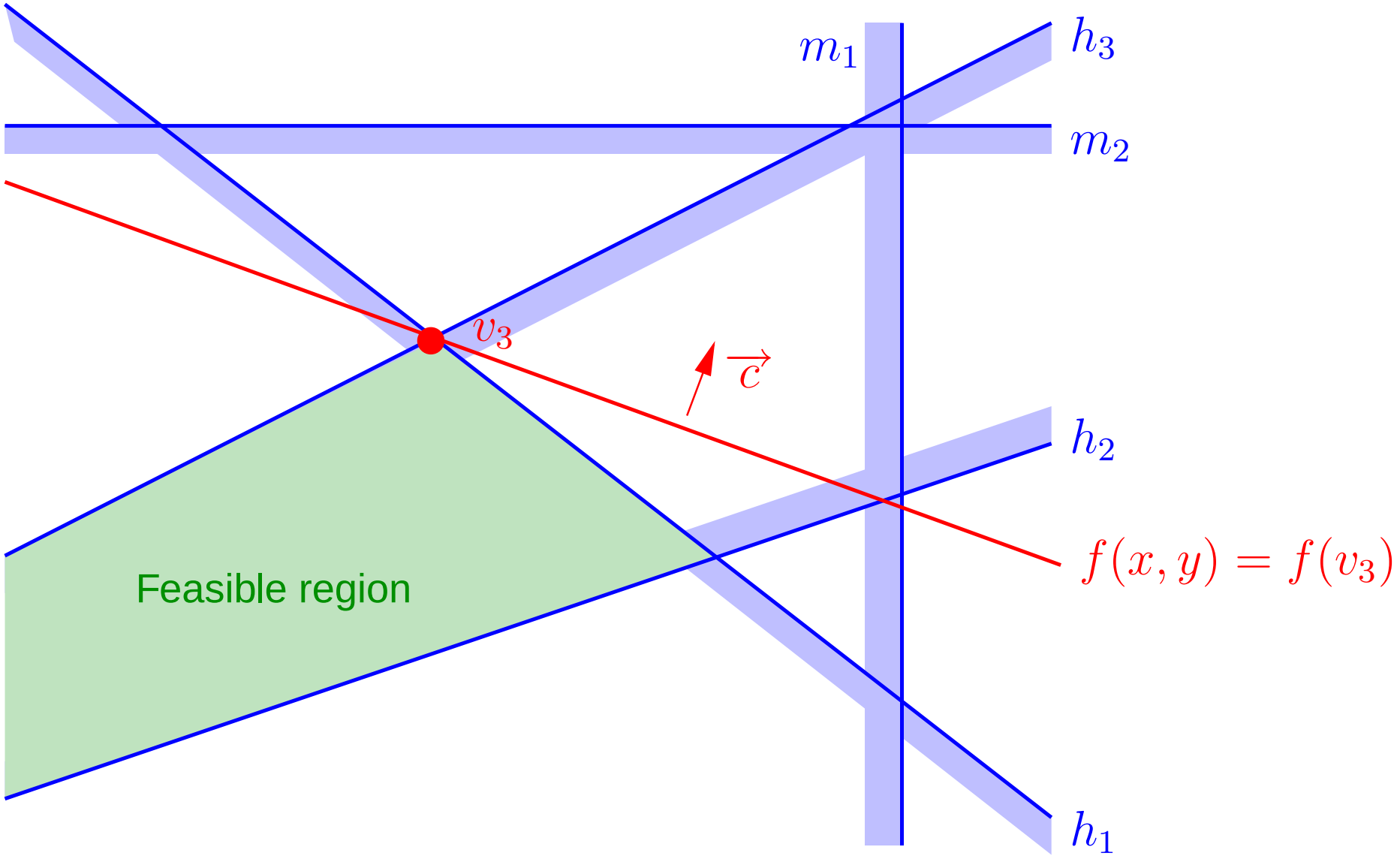
Example



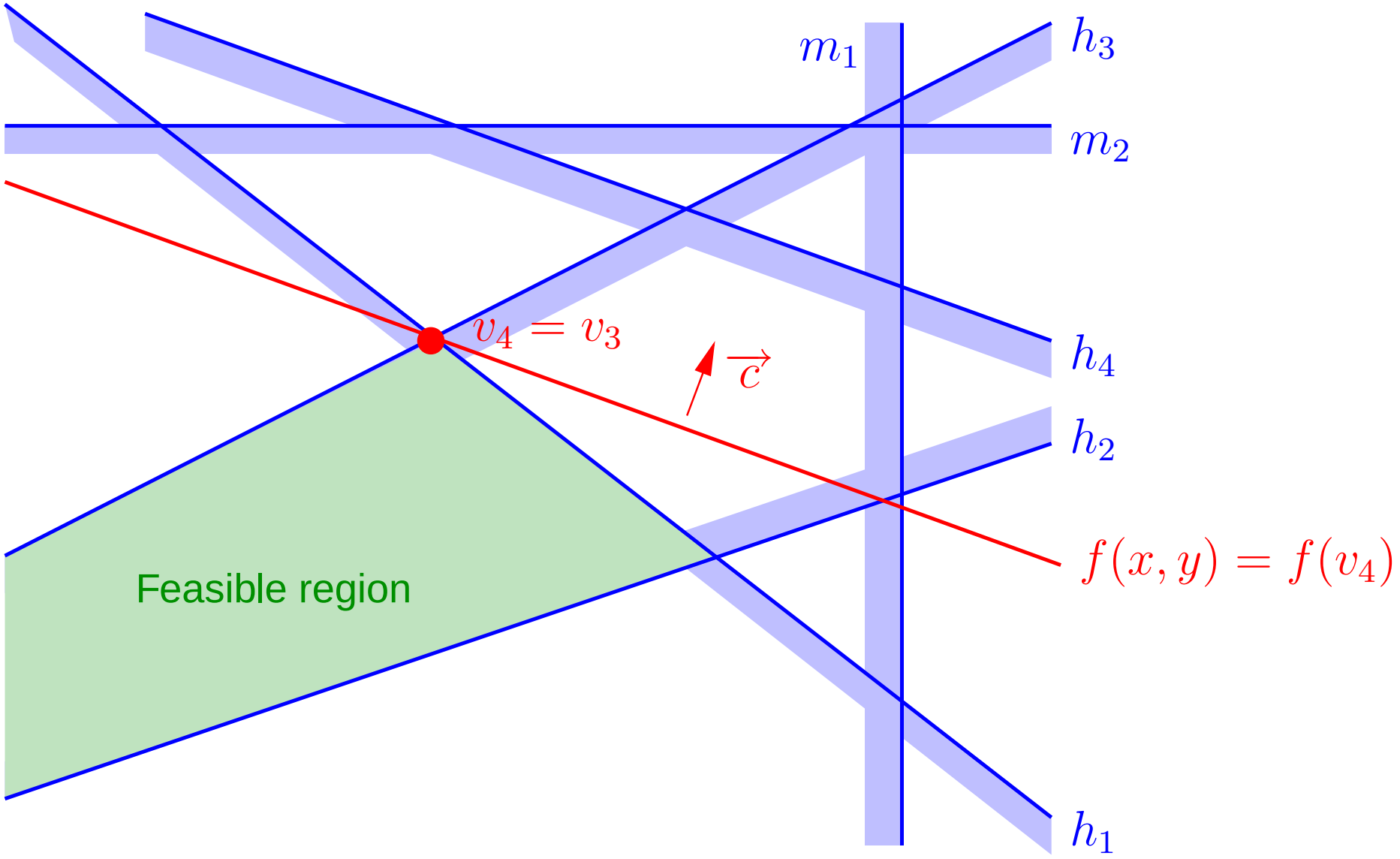
Example



Example



Example

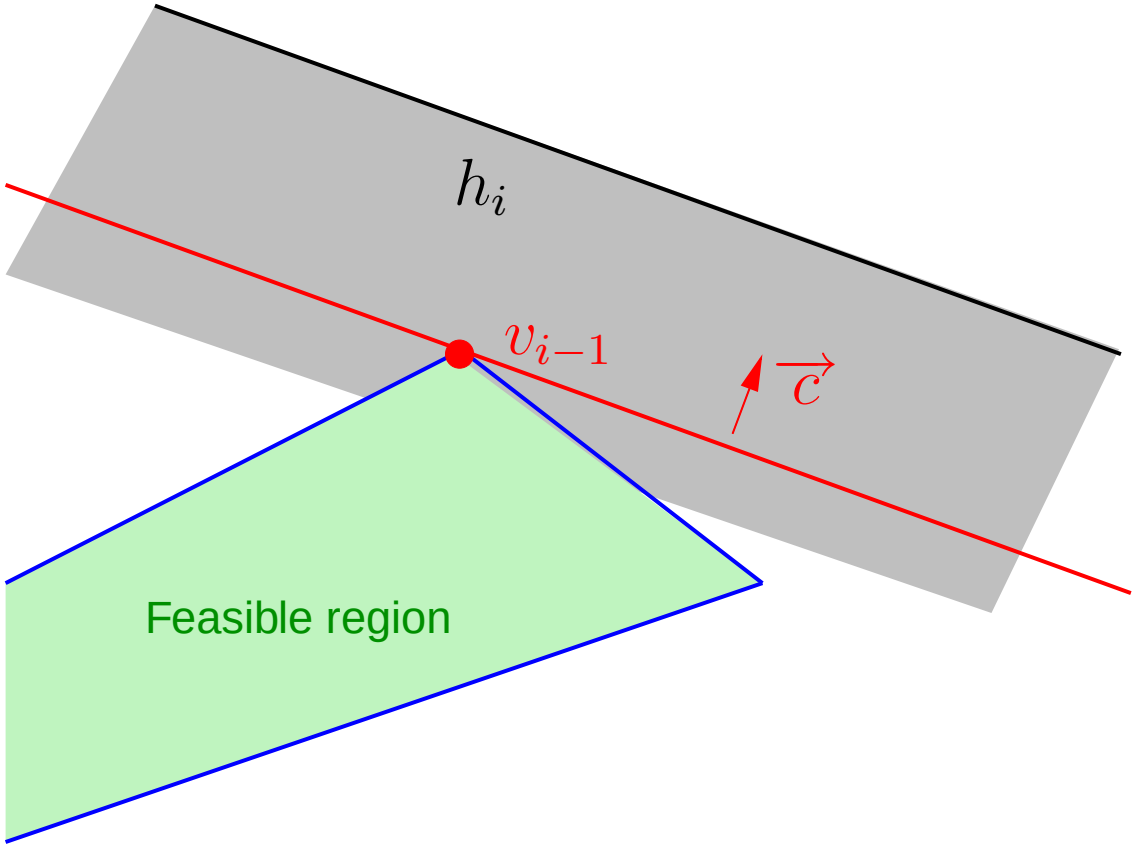


Algorithm

- randomized incremental algorithm
- before inserting h_i, we only assume that we know v_{i-1}
- how to find v_i?

First case

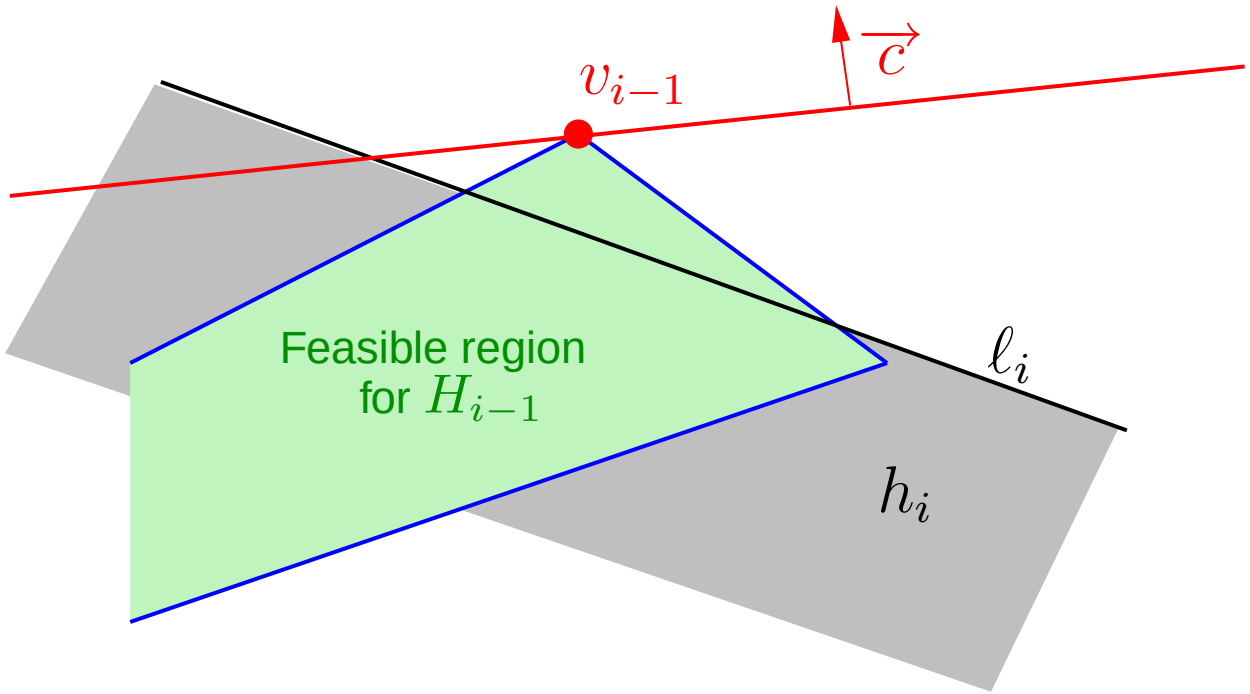
- first case: $v_{i-1} \in h_i$



- then $v_i = v_{i-1}$
- proof?

Second case

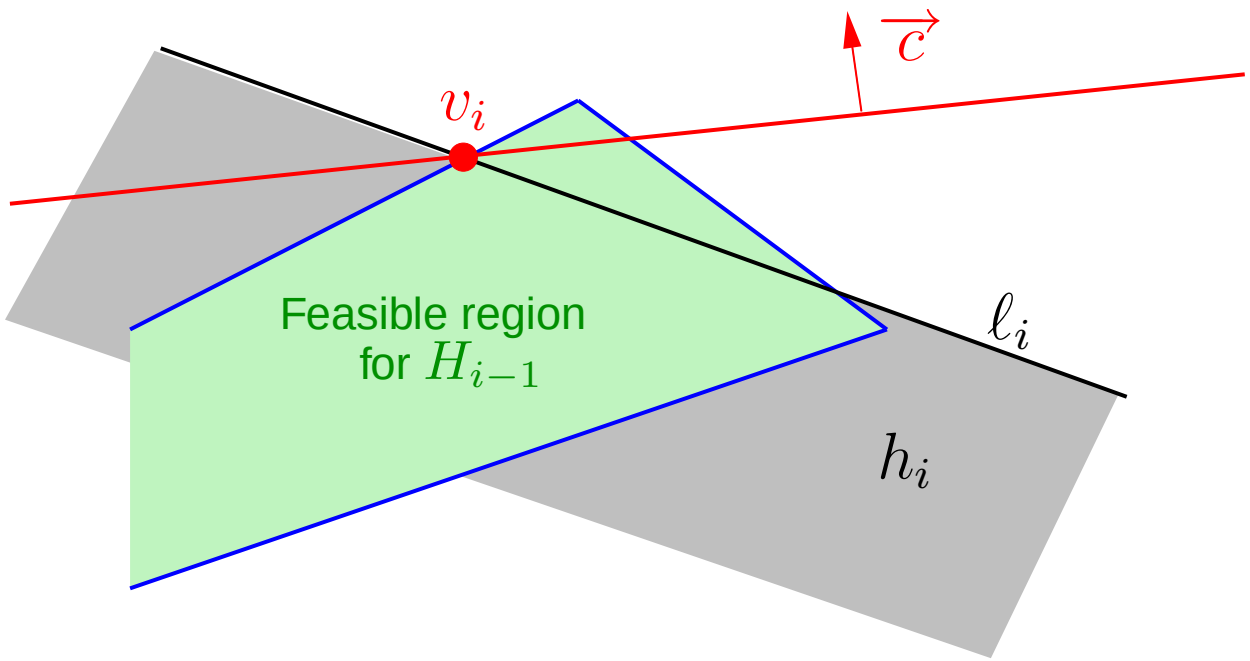
- second case: $v_{i-1} \notin h_i$



- then v_{i-1} is not in the feasible region of H_i
- so $v_i \neq v_{i-1}$

Second case

- what do we know about v_i?



- $v_i \in \ell_i$
- proof?
- how to find v_i?

Second case

- assume $a_{i,2} \neq 0$, then the equation of ℓ_i is

$$y = b_i - \frac{x}{a_{i,2}}$$

- we replace y by $b_i - \frac{x}{a_{i,2}}$ in all the constraints of H_i and in the objective function
- we obtain a one dimensional linear program
- if it is feasible, its solution gives us the x–coordinate of v_i
- we obtain the y–coordinate using the equation of ℓ_i
- if this linear program is infeasible, then the original 2D linear program is infeasible too and we are done

Analysis

- first case is done in $O(1)$ time: just check whether $v_{i-1} \in h_i$
- second case in $O(i)$ time: one dimensional linear program with $i+2$ constraints
- so the algorithm runs in $O(n^2)$ time
  - is there a worst case example where it runs in $\Omega(n^2)$ time?
- what is the expected running time?
- we need to know how often the second case happens
- we define the random variable X_i
  - $X_i = 0$ in first case ($v_i = v_{i-1}$)
  - $X_i = 1$ in second case ($v_i \neq v_{i-1}$)

Analysis

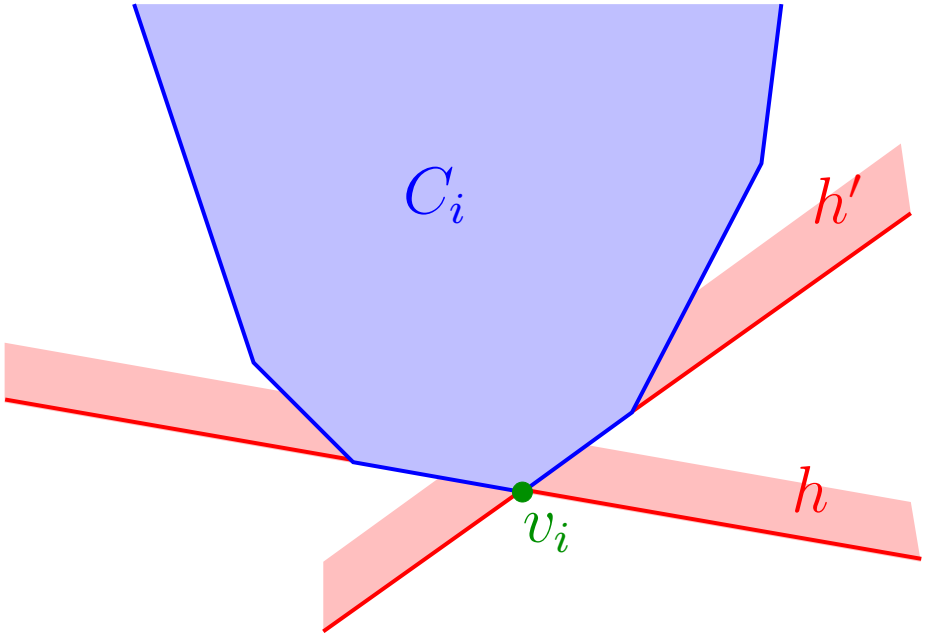
- when $X_i = 0$ we spend $O(1)$ time at i–th step
- when $X_i = 1$ we spend $O(i)$ time
- so the expected running time $ET(n)$ is

$$ET(n) = O\left(\sum_{i=1}^{n} 1 + i.E[X_i]\right)$$

- note: $E[X_i]$ is simply the probability that $X_i = 1$ in our case

Analysis

- we denote $C_i = \bigcap H_i$
- in other words, C_i is the feasible region at step i
- v_i is adjacent to two edges of C_i, these edges correspond to two constraints h and h'



- if $v_i \neq v_{i-1}$ then $h_i = h$ or $h_i = h'$

Analysis

- what is the probability that $h_i = h$ or $h_i = h'$?
- we use backwards analysis
- we assume that H_i is fixed, no other assumption
- so h_i is chosen uniformly at random in $H_i \setminus \{m_1, m_2\}$
- so the probability that $h_i = h$ or $h_i = h'$ is $2/i$
- so $E[X_i] \leq 2/i$
- so

$$ET(n) = O\left(\sum_{i=1}^{n} 1 + i.\frac{2}{i}\right) = O(n)$$

Generalization to higher dimension

First attempt

- each constraint is a half–space
- can we compute their intersection and get the feasible region?
- in $\mathbb{R}^3$ it can be done in $O(n \log n)$ time (not covered in CS4235)
- in higher dimension, the feasible region has $\Omega(n^{\lfloor \frac{d}{2} \rfloor})$ vertices in the worst case
- so computing the feasible region requires $\Omega(n^{\lfloor \frac{d}{2} \rfloor})$ time
- here, we will give a $O(n)$ expected time algorithm for $d = O(1)$

Preliminary

- a hyperplane in $\mathbb{R}^d$ is a set with equation

$$\alpha_1 x_1 + \alpha_2 x_2 + \ldots + \alpha_d x_d = \beta_d$$

  where

$$(\alpha_1, \alpha_2, \ldots, \alpha_d) \in \mathbb{R}^d \setminus \{0\}^d$$

- in general position, d hyperplanes intersect at one point
- each constraint h_i is an half–space, bounded by an hyperplane ∂h_i
  - we assume general position in that any d such hyperplanes intersect at exactly one point
  - no point belongs to $d+1$ such hyperplanes
  - no such hyperplane is orthogonal to $\overrightarrow{c}$

Algorithm

- we generalize the 2D algorithm
- we first find d constraints $m_1, m_2, \ldots m_d$ that make the linear program bounded:
  - if $c_i \geq 0$ then m_i is $x_i \leq M$
  - if $c_i < 0$ then m_i is $x_i \geq -M$
- we pick a random permutation $(h_1, h_2, \ldots h_n)$ of H
- then H_i is $\{m_1, m_2, \ldots m_d, h_1, h_2, \ldots h_i\}$
- we maintain v_i, the solution to the linear program with constraints H_i and objective function f
- v_0 is the vertex of

$$\bigcap_{i=1}^{d} m_i$$

Algorithm

- we assume $d = O(1)$
- inserting h_i is done in the same way as in $\mathbb{R}^2$:
- if $v_{i-1} \in h_i$ then $v_{i-1} = v_i$
- otherwise $v_i \in \partial h_i$
  - so we find v_i by solving a linear program with $i + d$ constraints in $\mathbb{R}^{d-1}$
    - if this linear program is infeasible, then the original linear program is infeasible too, so we are done
  - it can be done in expected $O(i)$ time (by induction)

Analysis

- what is the probability that $v_i \neq v_{i-1}$?
  - by our general position assumption, v_i belongs to exactly d hyperplanes that bound constraints in H_i
  - the probability that $v_i \neq v_{i-1}$ is the probability that one of these d constraints was inserted last
  - by backwards analysis, it is d/i
- so the expected running time of our algorithm is

$$ET(n) = O\left(\sum_{i=1}^{n} 1 + i.\frac{d}{i}\right) = O(dn) = O(n)$$

Conclusion

- this algorithm can be made to handle unbounded linear programs and degenerate cases
- a careful implementation of this algorithm runs in $O(d!n)$ time (see Seidel's paper)
- so it is only useful in low dimension
- it can be generalized to other types of problems
  - see textbook: smallest enclosing disk
- sometimes we can linearize a problem and use a linear programming algorithm
  - see tomorrow's tutorial: finding the enclosing annulus with minimum area