2D Linear Programming

slides by Andy Mirzaian

(a subset of the original slides are used here)

The LP Problem

maximize $c_1 x_1 + c_2 x_2 + \cdots + c_d x_d$

subject to:

$$
\begin{aligned}
a_{11}x_1 + a_{12}x_2 + \cdots + a_{1d}x_d &\le b_1 \\
a_{21}x_1 + a_{22}x_2 + \cdots + a_{2d}x_d &\le b_2 \\
&\vdots \\
a_{n1}x_1 + a_{n2}x_2 + \cdots + a_{nd}x_d &\le b_n
\end{aligned}
$$

Applications

- The most widely used Mathematical Optimization Model.
- Management science (Operations Research).
- Engineering, technology, industry, commerce, economics.
- Efficient resource allocation:
  - Airline transportation,
  - Communication network – opt. transmission routing,
  - Factory inventory/production control,
  - Fund management, stock portfolio optimization.
- Approximation of hard optimization problems.
- . . .

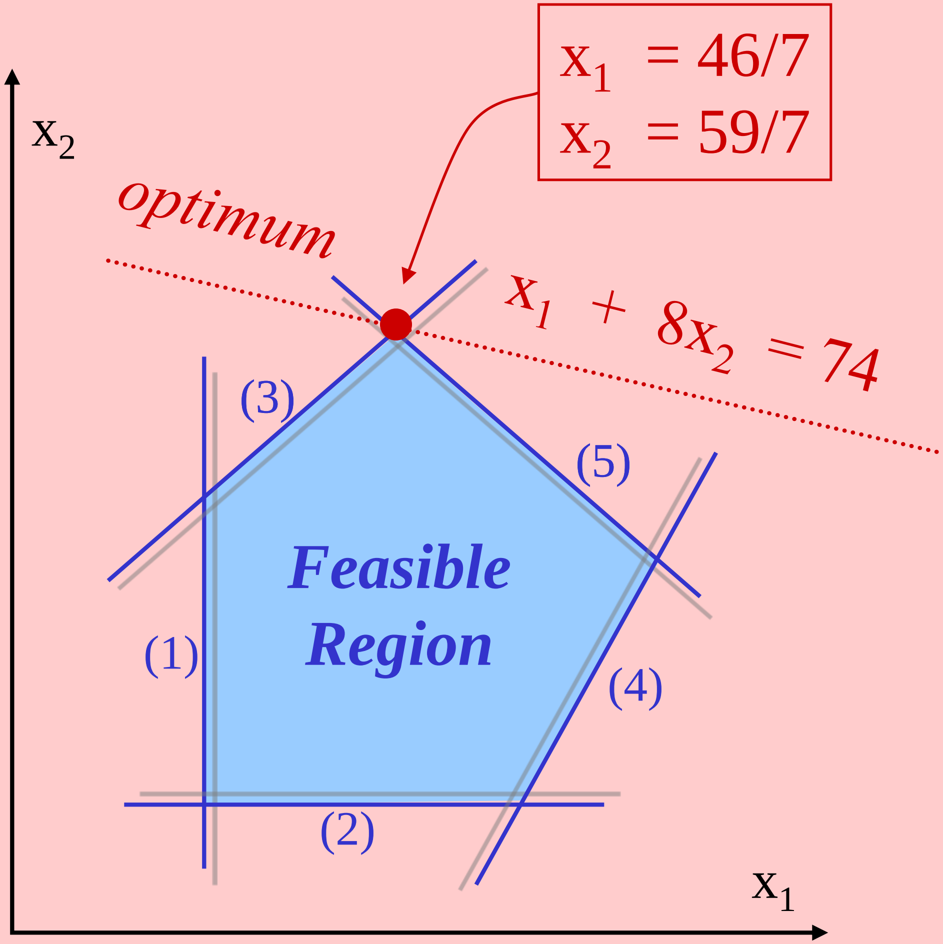
Example in 2D

max $\quad x_1 + 8x_2$

subject to:

(1)	x_1		$\geq$	3
(2)		x_2	$\geq$	2
(3)	$-3x_1$	$+ \; 4x_2$	$\leq$	14
(4)	$4x_1$	$- \; 3x_2$	$\leq$	25
(5)	x_1	$+ \; x_2$	$\leq$	15

optimum basic constraints



Example in 3D

maximize z

subject to:

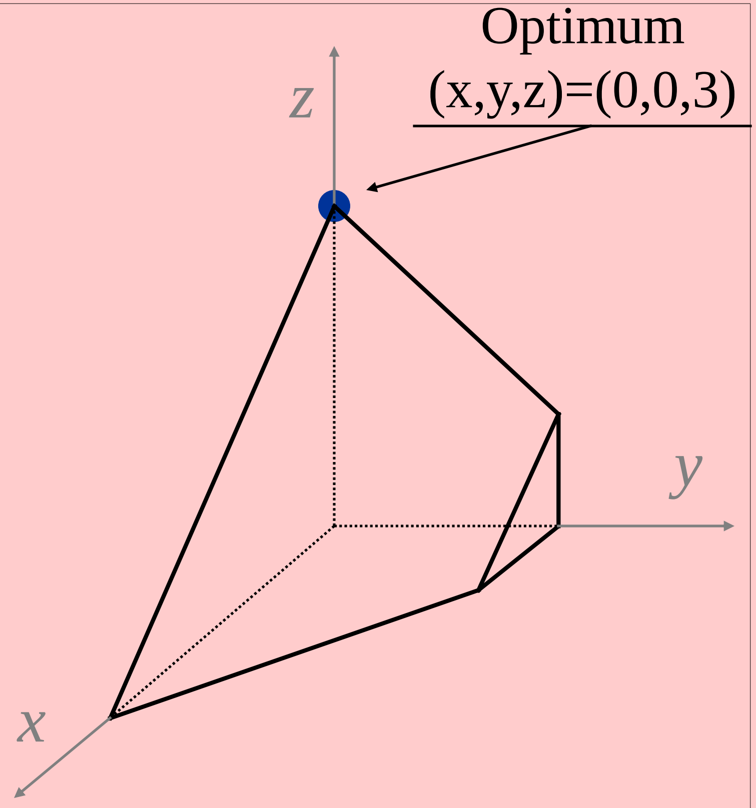
$$x + y + z \leq 3$$

$$y \leq 2$$

$$x \geq 0$$

$$y \geq 0$$

$$z \geq 0$$



History of LP

- 3000-200 BC: Egypt, Babylon, India, China, Greece: [geometry & algebra]
  Egypt: polyhedra & pyramids.
  India: Sulabha suutrah (Easy Solution Procedures) [2 equations, 2 unknowns]
  China: Jiuzhang suanshu (9 Chapters on the Mathematical Art)
  [Precursor of Gauss-Jordan elimination method on linear equations]
  Greece: Pythagoras, Euclid, Archimedes, …
- 825 AD: Persia: Muhammad ibn-Musa Alkhawrazmi (author of 2 influential books):
  "Al-Maqhaleh fi Hisab al-jabr w'almoqhabeleh" (An essay on Algebra and equations)
  "Kitab al-Jam'a wal-Tafreeq bil Hisab al-Hindi" (Book on Hindu Arithmetic).
  originated the words algebra & algorithm for solution procedures of algebraic systems.

- Fourier [1826], Motzkin [1933] [Fourier-Motzkin elimination method on linear inequalities]
- Minkowski [1896], Farkas [1902], De la Vallée Poussin [1910], von Neumann [1930's], Kantorovich [1939], Gale [1960] [LP duality theory & precursor of Simplex]

- **George Dantzig [1947]: Simplex algorithm.**
  Exponential time in the worst case, but effective in practice.
- **Leonid Khachiyan [1979]: Ellipsoid algorithm.**
  The first weakly polynomial-time LP algorithm: poly(n,d,L).
- **Narendra Karmarkar [1984]: Interior Point Method.**
  Also weakly polynomial-time. IPM variations are very well studied.
- **Megiddo-Dyer [1984]: Prune-&-Search method.**
  O(n) time if the dimension is a fixed constant. Super-exponential on dimension.

LP: Fundamental Facts

Fundamental Theorem of LP

For any instance of LP exactly one of the following three possibilities holds:

(a) Infeasible.

(b) Feasible but no bounded optimum.

(c) Bounded optimum.

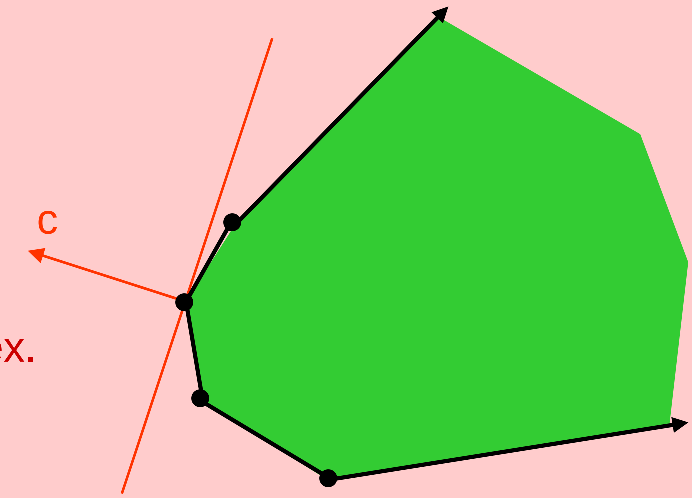
[Note: Feasible polyhedron could be unbounded even if optimum is bounded.
It depends on the direction of the objective vector.]

Moreover, if A has full rank (i.e., ∃ basis), then every nonempty face of the feasible polyhedron contains a BFS, and this implies:

(1) ∃ feasible solution $\Rightarrow$ ∃ BFS.

(2) ∃ optimum solution $\Rightarrow$ ∃ optimum solution that is a BFS.

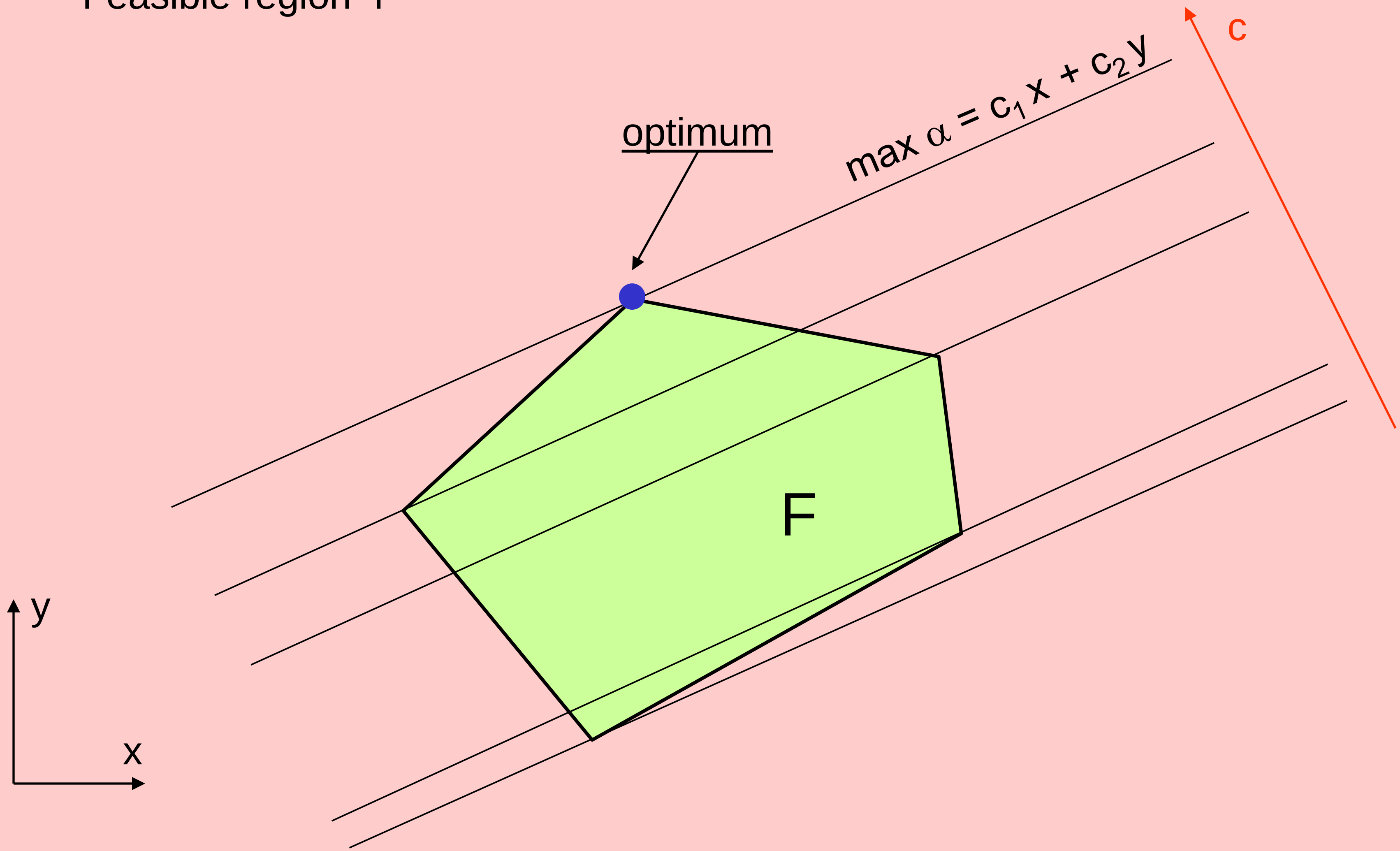
Proof: If there is a basis, the basic cone contains the feasible region but does not contain any line. So the feasible region does not contain any line, hence it is pointed. So every non-empty face of it (including the optimal face, if non-empty) is pointed, and thus contains a vertex. (For details see exercise 4.)



2D Linear Programming

Objective function $\alpha = c_1 x + c_2 y$, $c^T = (c_1, c_2)$

Feasible region F

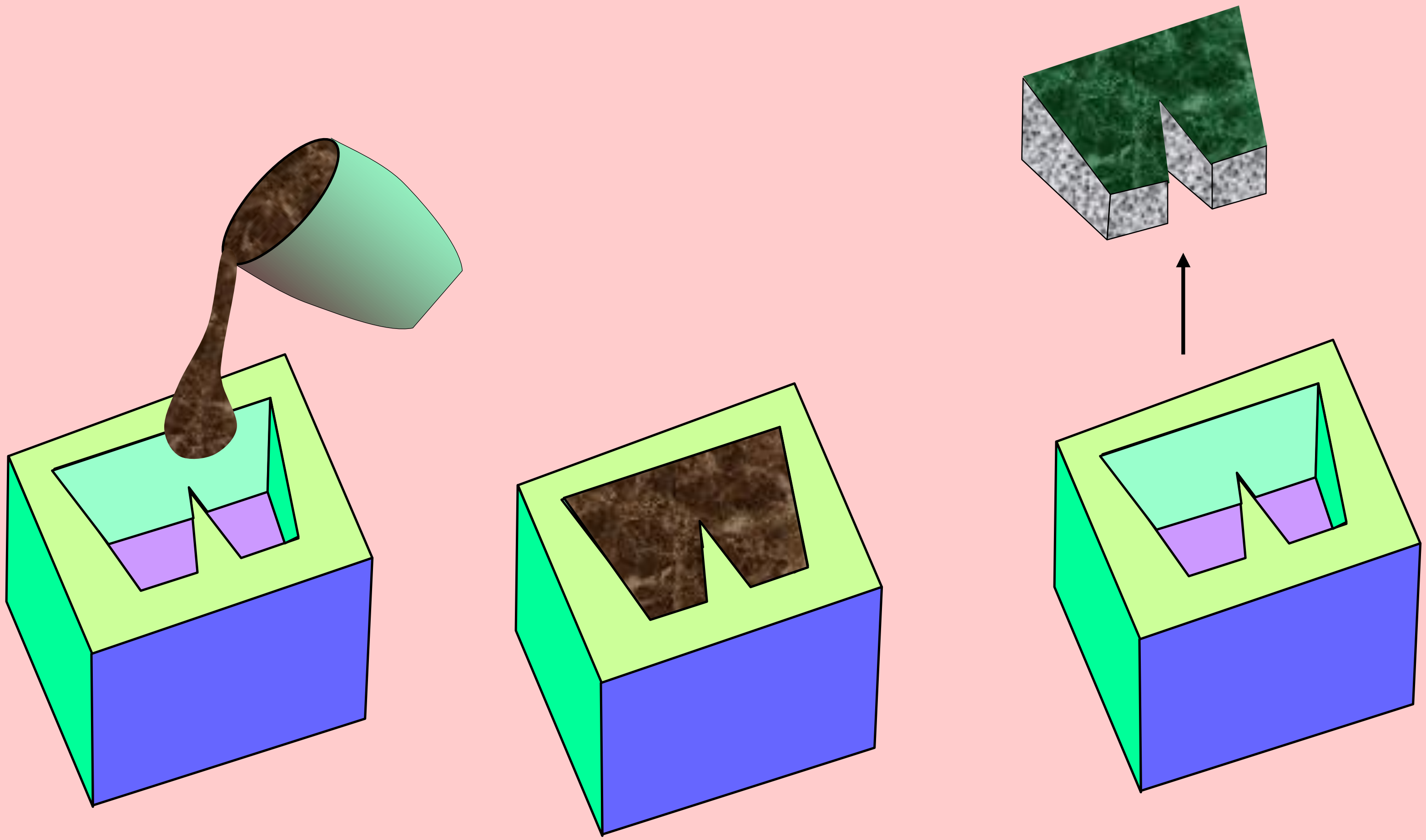


2D Linear Programming

- Feasible region F is the intersection of n half-planes.
- F is (empty, bounded or unbounded) convex polygon with $\leq$ n vertices.
- F can be computed in O(n log n) time by divide-&-conquer
  (See Lecture-Slide 3).
- If F is empty, then LP is infeasible.
- Otherwise, we can check its vertices, and its possibly up to
  2 unbounded edges, to determine the optimum.
- The latter step can be done by binary search in O(log n) time.
- If objective changes but constraints do not, we can update the optimum
  in only O(log n) time. (We don't need to start from scratch).

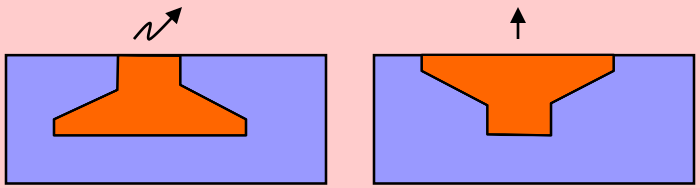
- <u>Improvement Next</u>:
  Feasible region need not be computed to find the optimum vertex.
  Optimum can be found in O(n) time both randomized & deterministic.

2D LP Example: Manufacturing with Molds



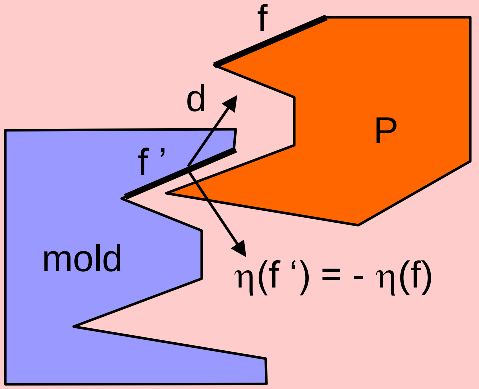
2D LP Example: Manufacturing with Molds

<u>The Geometry of Casting</u>: Is there a mold for an n-faceted 3D polytope P such that P can be removed from the mold by translation?



<u>Lemma</u>: P can be removed from its mold with a single translation in direction d

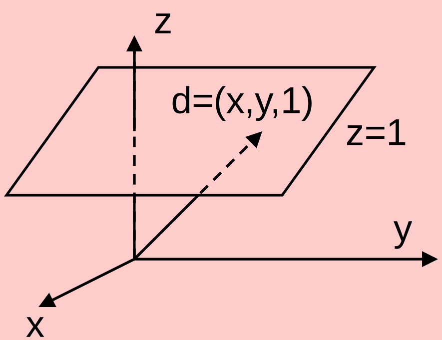
$\Leftrightarrow$ d makes an angle $\geq 90°$ with the outward normal of all non-top facets of P.

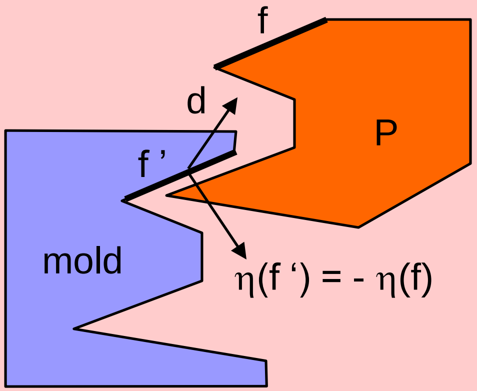


<u>Corollary</u>: Many small translations possible $\Leftrightarrow$ Single translation possible.

2D LP Example: Manufacturing with Molds

<u>The Geometry of Casting</u>: Is there a mold for an n-faceted 3D polytope P such that P can be removed from the mold by translation?





$\eta(f) = (\ \eta_x(f)\ ,\ \eta_y(f)\ ,\ \eta_z(f)\ )$ outward normal to facet f of P.

$d^T.\eta(f) \le 0 \quad \forall$ non-top facet f of P $\Leftrightarrow$

$$\boxed{\eta_x(f)\ .\ x + \eta_y(f)\ .\ y + \eta_z(f) \le 0 \quad \forall f}$$

n-1 constraints

THEOREM: The mold casting problem can be solved in O(n log n) time.
(This will be improved to O(n) time on the next slides.)

Randomized Incremental Algorithm

Randomization

Random(k): Returns an integer $i \in 1..k$, each with equal probability $1/k$.
[Use a random number generator.]

Algorithm RandomPermute (A) O(n) time

Input: Array A[1..n]

Output: A random permutation of A[1..n] with each of n! possible permutations equally likely.

for $k \leftarrow n$ downto 2 **do** Swap A[k] with A[Random(k)]

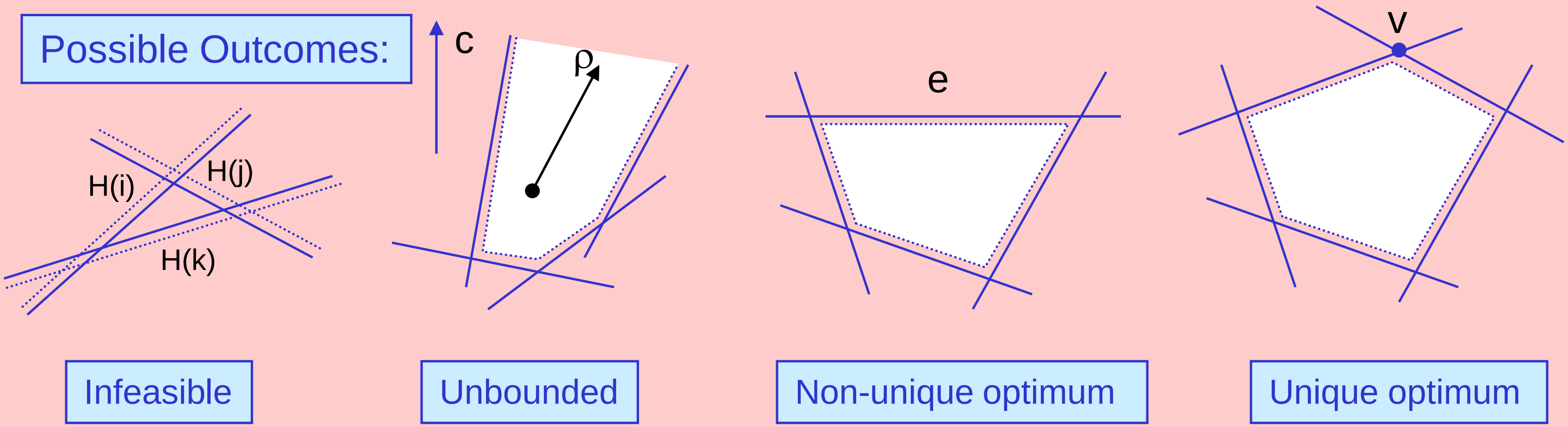
end.

This is a basic "initial" part of many randomized incremental algorithms.

2D LP: Incremental Algorithm

Method: Add constraints one-by-one, while maintaining the current optimum vertex.

Possible Outcomes:



Input: (H, c), H = { H(1), H(2), … , H(n)} n half-planes, c = objective vector

Output: Infeasible: (i,j,k), or

Unbounded: ρ, or

Optimum: $v = \text{argmax}_x \{ c^T x \mid x \in H(1) \cap H(2) \cap \ldots \cap H(n) \}$.

Define: $C(i) = H(1) \cap H(2) \cap \ldots \cap H(i)$, for i = 1..n

v(i) = optimum vertex of C(i) , for i=2..n. $c^T v(i) = \max \{ c^T x \mid x \in C(i) \}$

Note: $C(1) \supseteq C(2) \supseteq \ldots \supseteq C(n)$.

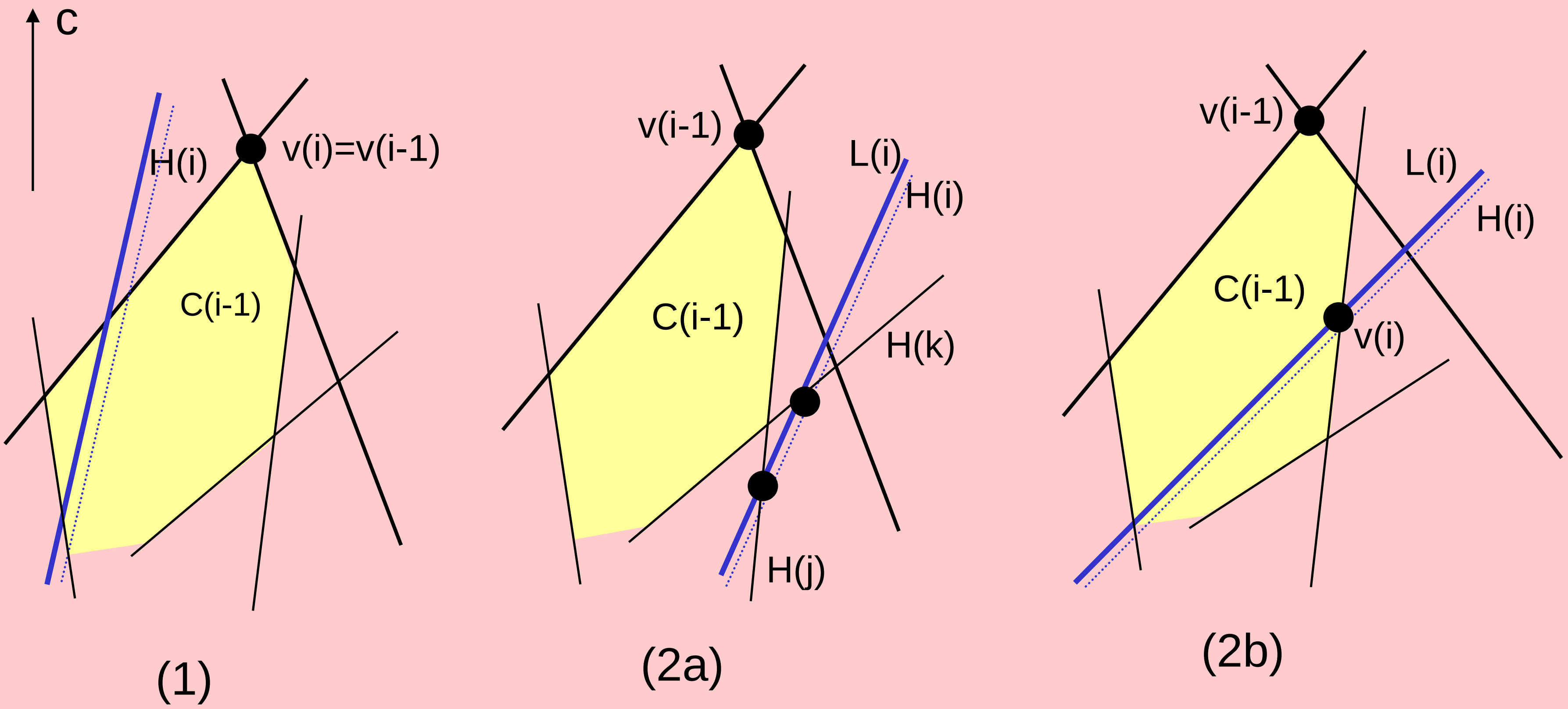
2D LP: Incremental Algorithm

LEMMA: (1) v(i-1) $\in$ H(i) $\Rightarrow$ v(i-1) $\in$ C(i) $\Rightarrow$ v(i) $\leftarrow$ v(i-1).

(2) v(i-1) $\notin$ H(i) $\Rightarrow$

(2a) C(i) = $\varnothing$, or

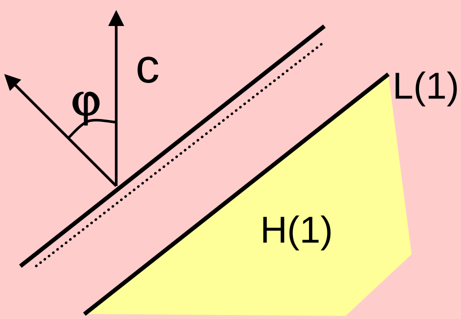
(2b) v(i) $\in$ L(i) $\cap$ C(i-1), L(i) =bounding-line of H(i).



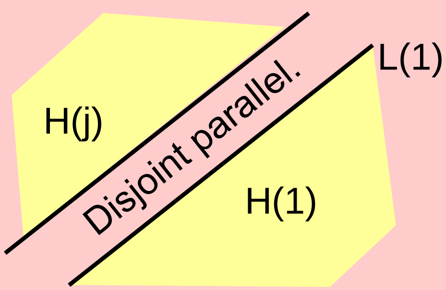
2D LP: Incremental Algorithm

Algorithm PreProcess (H,c) O(n) time

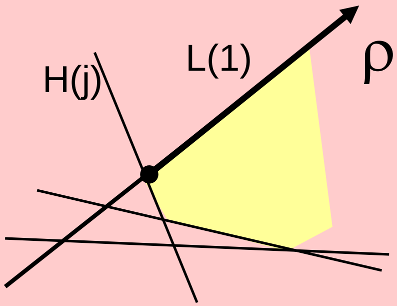
1. $\varphi \leftarrow$ min { angle between c and outward normal of H(i) | i=1..n }
   H(i) = most restrictive constraint with angle φ
   Swap H(i) with H(1) (L(1) is bounding-line of H(1)).
2. If $\exists$(parallel) H(j) with angle $\pi-\varphi$ and H(1)$\cap$H(j)=$\varnothing$ then return "infeasible".
3. If L(1) $\cap$ H(j) is unbounded for all H(j) $\in$H, then
   $\rho \leftarrow$ most restrictive L(1) $\cap$ H(j) over all H(j) $\in$H
   return ("unbounded", ρ)
4. If L(1) $\cap$ H(j) is bounded for some H(j)$\in$H, then
   Swap H(j) with H(2) and return "bounded".



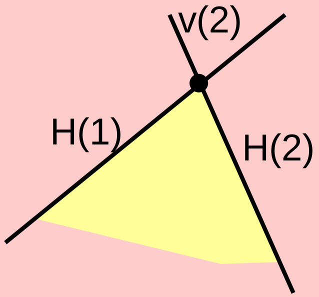
(1)



(2)



(3)



(4)

Randomized Incremental 2D LP Algorithm

Input: (H, c), H = { H(1), H(2), … , H(n)} n half-planes, c = objective vector
Output: Solution to max { $c^T x$ | $x \in H(1) \cap H(2) \cap \ldots \cap H(n)$ }

1. **if** **PreProcess(H,c)** returns ("unbounded", ρ) or "infeasible"
   then return the same answer
   (* else bounded or infeasible *)
2. $v(2) \leftarrow$ vertex of $H(1) \cap H(2)$
3. **RandomPermute (H[3..n])**
4. **for** $i \leftarrow 3..n$ **do**
5. **if** $v(i-1) \in H(i)$ **then** $v(i) \leftarrow v(i-1)$
6. **else** $v(i) \leftarrow$ optimum vertex p of $L(i) \cap (H(1) \cap \ldots \cap H(i-1))$ (* 1D LP *)
7. **if** p does not exist **then return** "infeasible"
8. **end-for**
9. **return** ("optimum", v(n))

end.

Randomized Incremental 2D LP Algorithm

THEOREM: 2D LP Randomized Incremental algorithm has the following complexity:

Space complexity = O(n)

Time Complexity: (a) $O(n^2)$ worst-case

(b) O(n) expected-case.

Proof of (a):

Line 6 is a 1D LP with i-1constraints and takes O(i) time.

Total time over for-loop of lines 4-8: $\sum_{i=3}^{n} O(i) = O(n^2)$.

Randomized Incremental 2D LP Algorithm

Proof of (b): Define 0/1 random variables $X(i) = \begin{cases} 1 & \text{if } v(i-1) \notin H(i) \\ 0 & \text{otherwise} \end{cases}$ for $i = 3..n$.

Lines 5-7 take O(i*X(i) +1) time. Total time is $T = O(n) + \sum_{i=3}^{n} O(i) * X(i)$

Expected time:

$$E[T] = O(n) + E[\sum_{i=3}^{n} O(i) * X(i)]$$
$$= O(n) + \sum_{i=3}^{n} O(i) * E[X(i)]$$

linearity of expectation

$$E[X(i)] = \Pr[v(i\text{-}1) \notin H(i)] \le 2/(i\text{-}2)$$

Backwards Analysis

"Fix" C(i) = { H(1), H(2)} $\cup$ $\underbrace{\{H(3), \ldots, H(i)\}}_{\text{random}}$

Random : C(i-1) = C(i) – {H(i)}

v(i) is defined by 2 H(j)'s. The probability that one of them is H(i) is $\le$ 2/(i-2).
This does not depend on C(i). Hence, remove the "Fix" assumption.

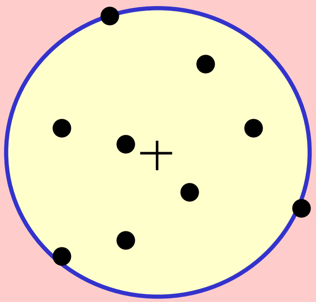
Therefore : $E[T] \le O(n) + \sum_{i=3}^{n} O(i) * \frac{2}{i-2} = O(n).$

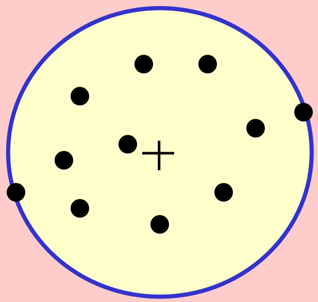
Randomized Incremental Algorithm for Smallest Enclosing Disk

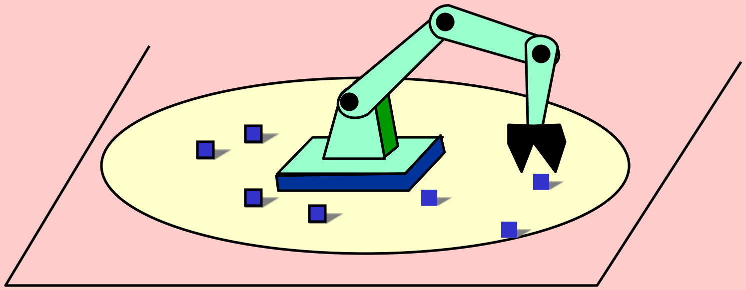
Smallest Enclosing Disk

Input: A set P={p_1, p_2, … , p_n } of n points in the plane.

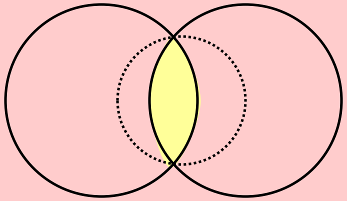
Output: Smallest enclosing disk D of P.







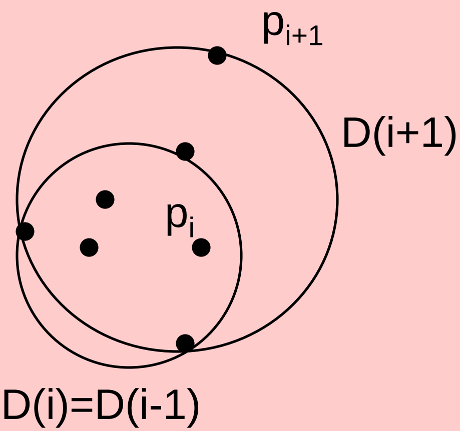
Lemma: Output is unique.



Incremental Construction:

P[1..i] = {p_1, p_2, … , p_i }
D(i)= smallest enclosing disk of P[1..i] .

Lemma: (1) $p_i \in$ D(i-1) $\Rightarrow$ D(i) = D(i-1)
(2) $p_i \notin$ D(i-1) $\Rightarrow$ p_i lies on the boundary of D(i).

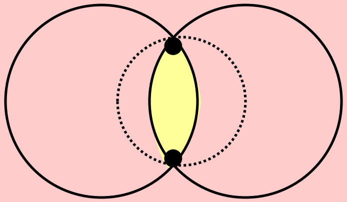


Smallest Enclosing Disk

LEMMA: Let P and R be disjoint point sets in the plane. $p \in P$, R possibly empty.
Define MD(P, R) = minimum disk D such that $P \subseteq D$ & $R \subseteq \partial D$ (∂D = boundary of D).

(1) If MD(P, R) exists, then it's unique,
(2) $p \in MD(P-\{p\}, R) \Rightarrow MD(P,R) = MD(P-\{p\}, R)$,
(3) $p \notin MD(P-\{p\}, R) \Rightarrow MD(P, R) = MD(P-\{p\}, R \cup \{p\})$.

<u>Proof</u>: (1) If non-unique $\Rightarrow \exists$ smaller such disk:



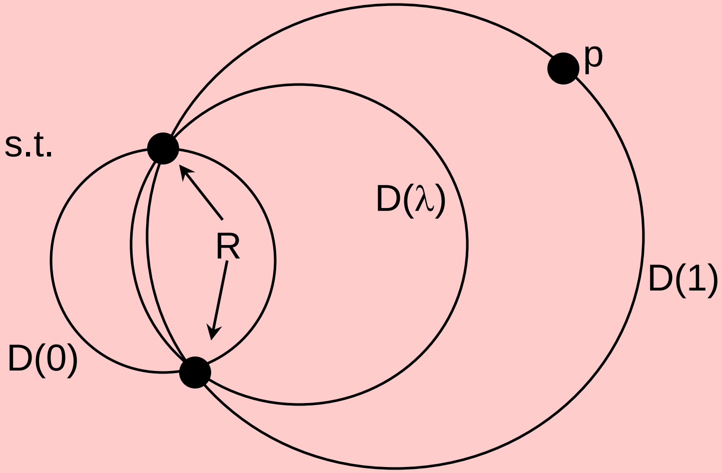
(2) is obvious.

(3) $D(0) \leftarrow MD(P-\{p\}, R)$
$D(1) \leftarrow MD(P, R)$
$D(\lambda) \leftarrow (1-\lambda)\, D(0) + \lambda\, D(1) \quad 0 \le \lambda \le 1$

As λ goes from 0 to 1, $D(\lambda)$ continuously deforms from D(0) to D(1) s.t.

$$\partial D(0) \cap \partial D(1) \subseteq \partial D(\lambda).$$

$p \in D(1) - D(0) \Rightarrow$ by continuity $\exists$ smallest λ^*, $0 < \lambda^* \le 1$ s.t.
$p \in D(\lambda^*) \Rightarrow p \in \partial D(\lambda^*)$.
$\Rightarrow P \subseteq D(\lambda^*)$ & $R \subseteq \partial D(\lambda^*) \Rightarrow \lambda^*=1$ by uniqueness.
Therefore, p is on the boundary of D(1).



Smallest Enclosing Disk

Algorithm MinDisk (P[1..n])
1. RandomPermute(P[1..n])
2. D(2) $\leftarrow$ smallest enclosing disk of P[1..2]
3. **for** i $\leftarrow$ 3..n **do**
4. **if** $p_i \in$ D(i-1) **then** D(i) $\leftarrow$ D(i-1)
5. **else** D(i) $\leftarrow$ MinDiskWithPoint (P[1..i-1] , p_i)
6. **return** D(n)

Procedure MinDiskWithPoint (P[1..j],q)
1. RandomPermute(P[1..j])
2. D(1) $\leftarrow$ smallest enclosing disk of p_1 and q
3. **for** i $\leftarrow$ 2..j **do**
4. **if** $p_i \in$ D(i-1) **then** D(i) $\leftarrow$ D(i-1)
5. **else** D(i) $\leftarrow$ MinDiskWith2Points (P[1..i-1] , q, p_i)
6. **return** D(j)

Procedure MinDiskWith2Points (P[1..j],q_1,q_2)
1. D(0) $\leftarrow$ smallest enclosing disk of q_1 and q_2
2. **for** i $\leftarrow$ 1..j **do**
3. **if** $p_i \in$ D(i-1) **then** D(i) $\leftarrow$ D(i-1)
4. **else** D(i) $\leftarrow$ Disk (q_1, q_2, p_i)
5. **return** D(j)

Show random permutation only once is enough

Smallest Enclosing Disk

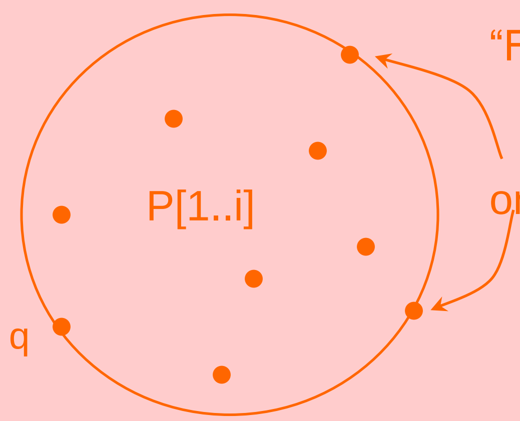
THEOREM: The smallest enclosing disk of n points in the plane can be computed in randomized O(n) expected time and O(n) space.

<u>Proof</u>: Space O(n) is obvious.
MinDiskWith2Points (P,q_1,q_2) takes O(n) time.
MinDiskWithPoint (P,q) takes time:

$$T = O(n) + \sum_{i=2}^{n} O(i) * X(i) \qquad \text{where} \qquad X(i) = \begin{cases} 1 & \text{if } p_i \in \partial D(i) - D(i-1) \\ 0 & \text{otherwise} \end{cases}$$

$$E[X(i)] \le 2/i \quad \text{(by backwards analysis)}$$

$$E[T] = O(n) + \sum_{i=2}^{n} O(i) * \frac{2}{i} = O(n).$$



“Fix” P[1..i] = $\{p_1, \ldots, p_i\}$
P[1..i-1] = $\{p_1, \ldots, p_i\} - \{p_i\}$

backwards

one of these is p_i with prob. $\le 2/i$

Apply this idea once more: expected running time of MinDisk is also O(n).