Linear Programming

(Read Chapter 4)

1

Motivation: Winning an Election

- In a fictional country a fictional candidate wants to figure out how to campaign to win the presidential election
- Campaign staff estimate votes obtained per $ spent advertising in support or against a particular issue

Issue \ Demographic	Urban	Suburban	Rural
Restrict immigration	1	3	−5
Gun control	8	2	−3
Farm subsidies	0	0	9
Public transportation	7	−1	2
Population	1,000,000	2,000,000	500,000

- *Goal*: win a majority *in each* demographic while spending a minimum amount of money

2

An Algebraic Representation

- One variable per issue: x_1, x_2, x_3, x_4

$$\begin{aligned}
&\min x_1 + x_2 + x_3 + x_4 \\
&\text{subject to} \\
&\quad x_1 + 8x_2 + 7x_4 \geq 500000 \\
&\quad 3x_1 + 2x_2 - x_4 \geq 1000000 \\
&\quad -5x_1 - 3x_2 + 9x_3 + 2x_4 \geq 250000 \\
&\quad x_1, x_2, x_3, x_4 \geq 0
\end{aligned}$$

- Optimum $x_1 = 3{,}500{,}000/11 \quad x_3 = 7{,}000{,}000/33$
  $x_2 = 250{,}000/11 \quad x_4 = 0$

  Total: $18{,}250{,}000/33$

3

Linear Programming

- There are two common *standard forms* depending on whether we express the problem as minimization or maximization

Minimization

$$\begin{aligned}
&\min c_1x_1 + c_2x_2 + \cdots + c_nx_n \\
&\text{subject to:} \\
&\begin{cases}
a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n \geq b_1 \\
a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n \geq b_2 \\
\vdots \\
a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n \geq b_m \\
\forall i, x_i \geq 0
\end{cases}
\end{aligned}$$

Maximization

$$\begin{aligned}
&\max c_1x_1 + c_2x_2 + \cdots + c_nx_n \\
&\text{subject to:} \\
&\begin{cases}
a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n \leq b_1 \\
a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n \leq b_2 \\
\vdots \\
a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n \leq b_m \\
\forall i, x_i \geq 0
\end{cases}
\end{aligned}$$

Or in matrix form, a maximization LP is:

Maximize $f_{\mathbf{c}}(\mathbf{x})$ subject to $A\mathbf{x} \leq \mathbf{b}$ where $f_{\mathbf{c}}(\mathbf{x}) = \mathbf{c} \cdot \mathbf{x}$

4

Applications

- Many problems from operations research, economics, production planning, transportation, etc., can be formulated as linear programming problems
  - *The Healthy Diet Problem*. Find the cheapest combination of foods that will satisfy your nutritional requirements
  - *Portfolio Optimization*. Minimize the risk in your investment portfolio subject to achieving a given return
  - *Airline Crew Scheduling*. Minimize costs of accommodations making sure that each flight is covered while meeting regulations
  - *Telecommunications*, including call routing and network design.
  - Solution of NP-hard problems, including TSP

5

Exercise

- A company makes two products (X and Y) using three machines (A, B, and C)
  - Each unit of X yields a profit of \$3 and takes 1 hour on machine A and 3 hours on machine C
  - Each unit of Y yields a profit of \$5 and takes 2 hours on machine B and 2 hours on machine C
- Machine A is available for 4 hours, B for 12 hours and C for 18 hours

<u>Goal</u>: maximize the total profit

- Formulate as a linear programming problem

6

Exercise…

- Notation

  x = number of units of X produced

  y = number of units of Y produced

- Maximize $3x + 5y$

  subject to

  $x \leq 4$ (time for machine A)

  $2y \leq 12$ (time for machine B)

  $3x + 2y \leq 18$ (time for machine C)

  $-x \leq 0$ (non-negative production for X, i.e., $x \geq 0$)

  $-y \leq 0$ (idem for unit Y)

7

Linear Programming

- There are two common *standard forms* depending on whether we express the problem as minimization or maximization

Minimization

$$\min c_1x_1 + c_2x_2 + \cdots + c_nx_n$$

subject to:

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n \geq b_1 \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n \geq b_2 \\ \vdots \\ a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n \geq b_m \\ \forall i, x_i \geq 0 \end{cases}$$

Maximization

$$\max c_1x_1 + c_2x_2 + \cdots + c_nx_n$$

subject to:

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n \leq b_1 \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n \leq b_2 \\ \vdots \\ a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n \leq b_m \\ \forall i, x_i \geq 0 \end{cases}$$

Or in matrix form, a maximization LP is:

Maximize $f_{\mathbf{c}}(\mathbf{x})$ subject to $A\mathbf{x} \leq \mathbf{b}$ where $f_{\mathbf{c}}(\mathbf{x}) = \mathbf{c} \cdot \mathbf{x}$

8

Notation

f_c is the objective function, defined on d variables

$H = \{h_1, \ldots, h_n\}$ is the set of constraints (halfplanes)

$C = \bigcap_{h \in H} h$ is the feasible region

A point $p \in C \subset R^d$ is a feasible solution

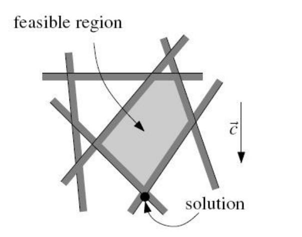
A point $p \in C$ that maximizes f_c is an optimal solution

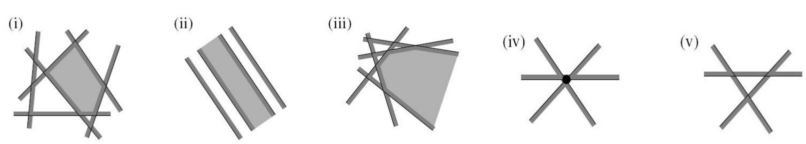
A *linear program* is denoted by the pair (H, $\boldsymbol{c}$)

9

Observations

- Feasible regions is a convex polyhedral region, possibly empty or unbounded



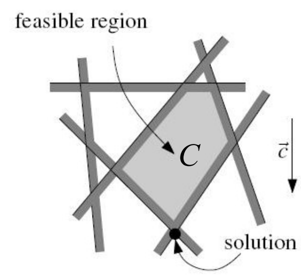


(i) and (iv) are bounded, (ii) and (iii) are unbounded, (v) is empty (hence, not feasible)

10

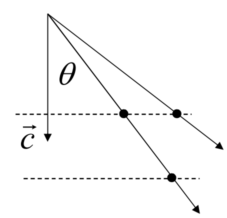
Observations…

- The objective function can be viewed as a direction in R^d
- Maximizing f_c means finding a point $(x_1,\ldots,x_d)\in C$ that is extreme in direction $\vec{c}$
- Every optimal solution corresponds to a point on the boundary of C



- Which of a set of points is extreme in direction $\vec{c}$?

$$f_c(\mathbf{x}) = \boldsymbol{c} \cdot \mathbf{x} = |\vec{c}|\ |\mathbf{x}| \cos\theta$$



A Deterministic Algorithm

1. Find feasible region C
2. Find optimal solution by evaluating f_c at each vertex of C

- Back to our example:

Max	$3x + 5y$	(profit)
s.t.	$x \leq 4$	(machine A)
	$2y \leq 12$	(machine B)
	$3x + 2y \leq 18$	(machine C)
	$x, y \geq 0$	(non-negativity)

Example

Max $3x + 5y$ such that

$$x \le 4$$
$$2y \le 12$$
$$3x + 2y \le 18$$
$$x, y \ge 0$$

Y

Every point in the first quadrant is associated with a specific production alternative

(point = decision)

0 *X*

Example…

Max $3x + 5y$ such that

$$x \le 4$$
$$2y \le 12$$
$$3x + 2y \le 18$$
$$x, y \ge 0$$

Y

(0,0) **(4,0)** *X*

Example…

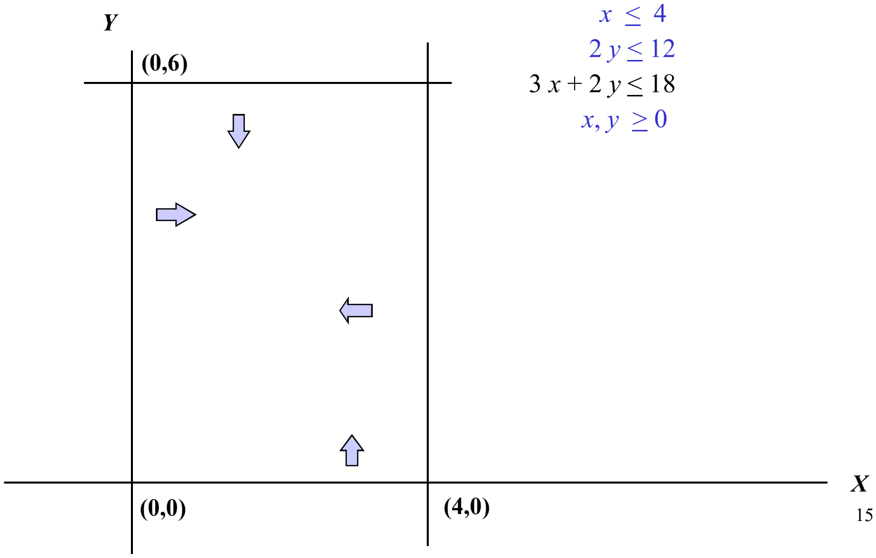
Max $3x + 5y$ such that

$x \le 4$

$2y \le 12$

$3x + 2y \le 18$

$x, y \ge 0$



15

Example…

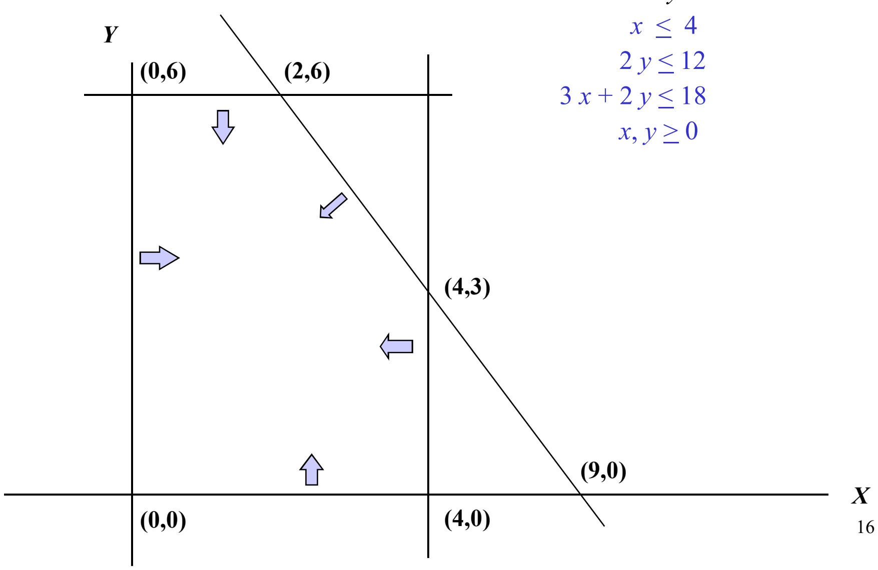
Max $3x + 5y$ such that

$x \le 4$

$2y \le 12$

$3x + 2y \le 18$

$x, y \ge 0$

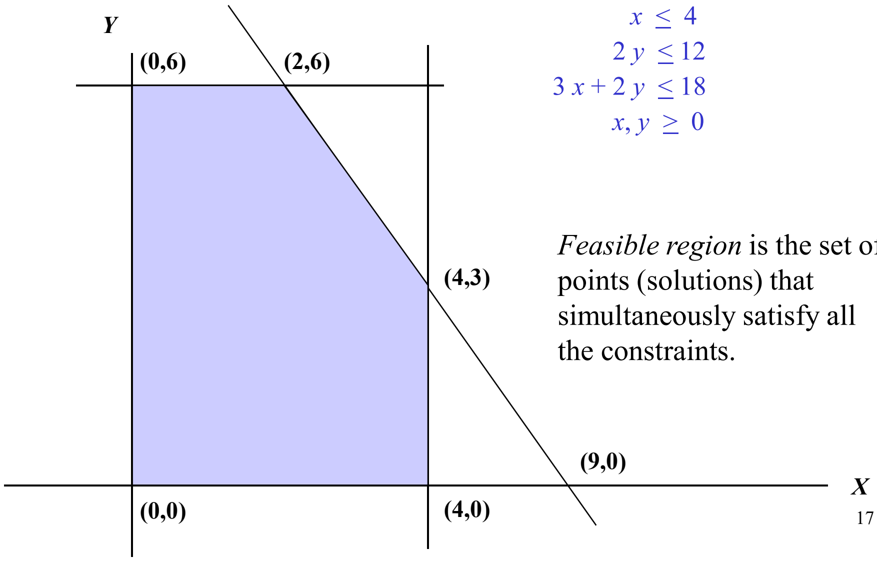


16

Example…

Max $3x + 5y$ such that

$$x \le 4$$
$$2y \le 12$$
$$3x + 2y \le 18$$
$$x, y \ge 0$$



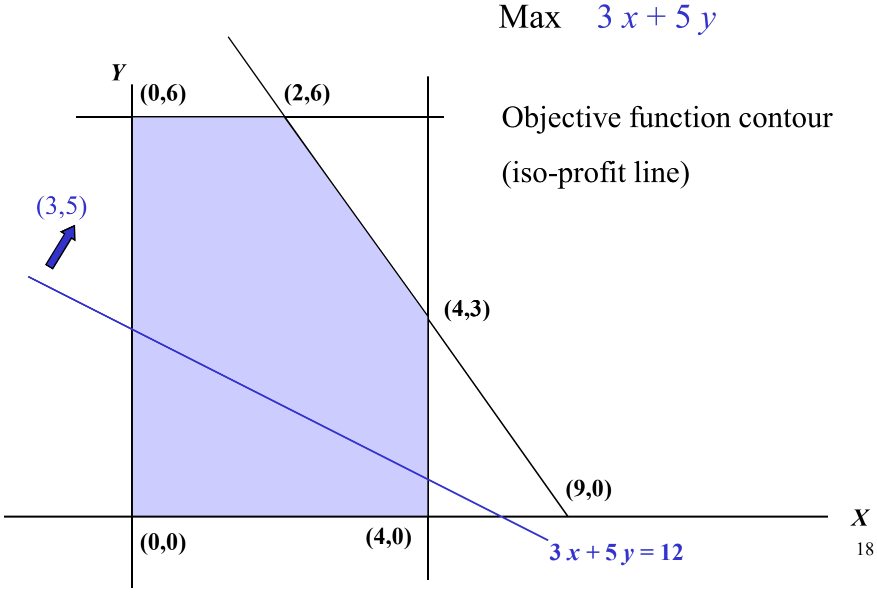
Feasible region is the set of points (solutions) that simultaneously satisfy all the constraints.

Example…

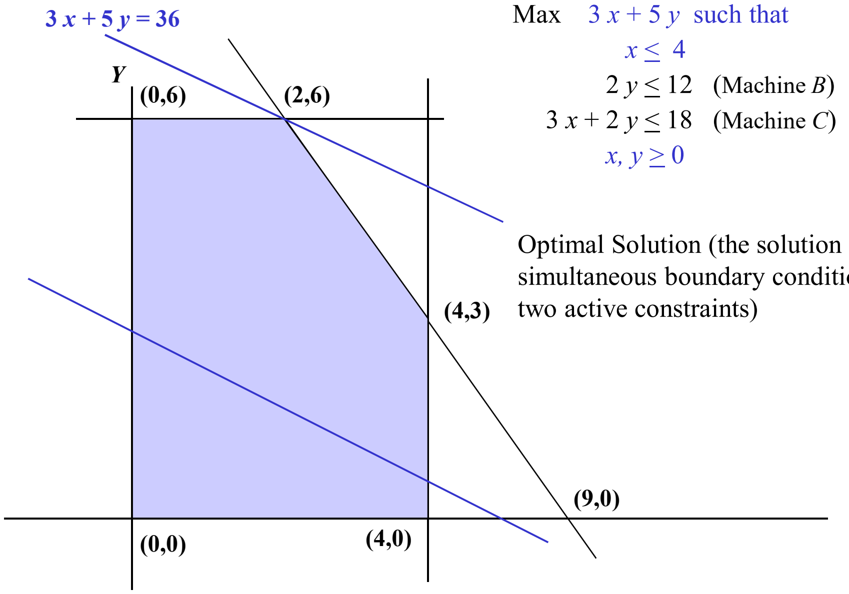
Max $3x + 5y$

Objective function contour

(iso-profit line)



Example

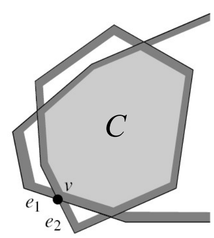


Max $3x + 5y$ such that
$x \leq 4$
$2y \leq 12$ (Machine *B*)
$3x + 2y \leq 18$ (Machine *C*)
$x, y \geq 0$

Optimal Solution (the solution for the simultaneous boundary conditions of two active constraints)

Halfplane Intersection

- Feasible region C can be found by divide-and-conquer using a sweep to merge the two answers



Algorithm INTERSECTHALFPLANES(H)
Input. A set H of n half-planes in the plane.
Output. The convex polygonal region $C := \bigcap_{h \in H} h$.

if card(H) $= 1$
  then $C \leftarrow$ the unique half-plane $h \in H$
  else Split H into sets H_1 and H_2 of size $\lceil n/2 \rceil$ and $\lfloor n/2 \rfloor$
    $C_1 \leftarrow$ INTERSECTHALFPLANES(H_1)
    $C_2 \leftarrow$ INTERSECTHALFPLANES(H_2)
    $C \leftarrow$ INTERSECTCONVEXREGIONS(C_1, C_2)

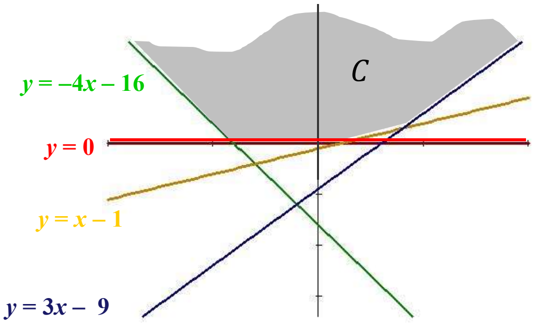
- Feasible region C can be found in time

$$T(n) = 2T(n/2) + n \in \Theta(n \log n)$$

Deterministic Algorithm…

- Halfplane intersection requires $\Omega(n \log n)$ time
  reduction from sorting: $a \to y \geq a\,x - a^2$
- Our deterministic algorithm is optimal!

Sort $\langle 1, -4, 3, 0 \rangle$



- Questions
  - Can you solve the linear program without computing C?
  - How do you deal with an unbounded feasible region C?

21

1D Linear Programming

- Maximize $f(x) = cx$ subject to $a_1 x \leq b_1, \ldots, a_n x \leq b_n$
- Each constraint defines a ray on the real line, bounded by a value $z_i = b_i / a_i$, on the left or right
- In linear time find

  x_{left} = the *largest* boundary point for rays bounded on the left

  x_{right} = the *smallest* boundary point for rays bounded on the right
- $[x_{\text{left}}, x_{\text{right}}]$ defines the feasible region C
  - If $x_{\text{left}} > x_{\text{right}}$ the linear program is infeasible
  - Otherwise, if C is bounded, then the optimal vertex is the better one of x_{left} and x_{right}

Theorem. A 1D linear program on n constraints can be solved in $O(n)$ time

22

2D Linear Programming

- Will concentrate on 2D first and then generalize

Maximize $f_{\mathbf{c}}(\mathbf{x}) = c_1 x_1 + c_2 x_2$

subject to

$$
\begin{aligned}
a_{1,1}x_1 + a_{1,2}x_2 &\le b_1 \qquad (h_1)\\
a_{2,1}x_1 + a_{2,2}x_2 &\le b_2 \qquad (h_2)\\
&\vdots\\
a_{n,1}x_1 + a_{n,2}x_2 &\le b_n \qquad (h_n)
\end{aligned}
$$

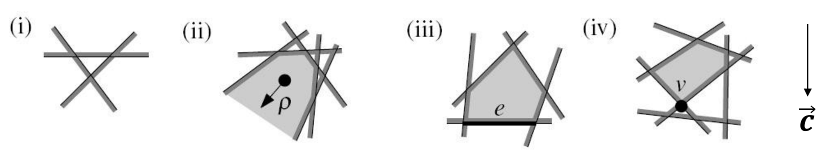
Or in matrix form:

Maximize $f_{\mathbf{c}}(\mathbf{x})$ subject to $A\mathbf{x} \le \mathbf{b}$

23

2D Linear Programming

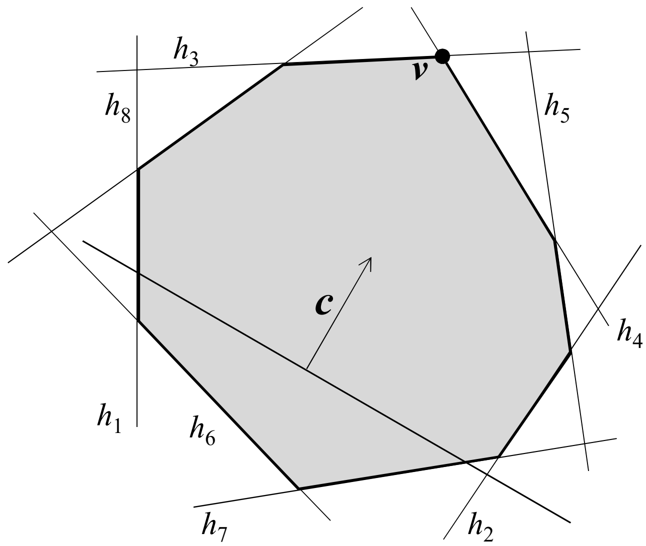
- C is a polygonal region, the intersection of n halfplanes



i. (H, c) is infeasible, as C is empty
ii. Feasible region C is unbounded in direction c and f_c takes arbitrarily large values along a ray ρ
iii. Solution is not unique as C has an extreme edge e with outward normal in direction c
iv. An extreme vertex of C if direction c is the unique solution of (H, c)

24

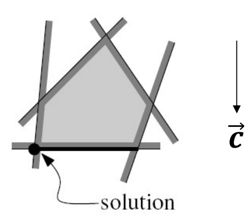
A 2D Bounded Linear Program (H, $\boldsymbol{c}$)



25

Bounded Linear Programming

- Will initially handle one type of case: *bounded* program with a *unique* solution
- To guarantee uniqueness we ask for the *lexicographically smallest* optimal solution
  - Does a unique solution always exist?



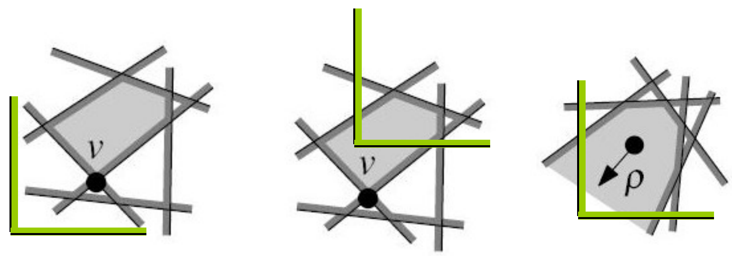
- In order to guarantee that the problem is bounded and has a unique solution we add two constraints

$$m_1 = \begin{cases} x \le M & \text{if } c_x > 0 \\ -x \le M & \text{otherwise} \end{cases} \qquad m_2 = \begin{cases} y \le M & \text{if } c_y > 0 \\ -y \le M & \text{otherwise} \end{cases}$$

26

Bounding Constraints

- M must be large enough so as not to interfere with real constraints, i.e., the choice of M should not alter the optimal solution of a bounded linear program



Correct Incorrect Correct

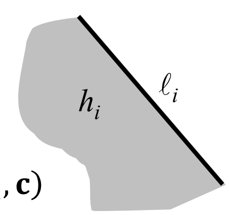
- In practice, M is handled symbolically

An Incremental Algorithm

- Starting from $C_0 = m_1 \cap m_2$, add the remaining halfplanes, one at a time, keeping track of the optimal solution
  (alternatively, we could use to bounding constraints from H, if they exist)

Notation:

$H_i = \{m_1, m_2, h_1, \ldots, h_i\}$
$C_i = m_1 \cap m_2 \cap h_1 \cap h_2 \cdots \cap h_i$
ℓ_i is the bounding line of h_i
v_i is the (unique) optimal solution of $(H_i, \mathbf{c})$



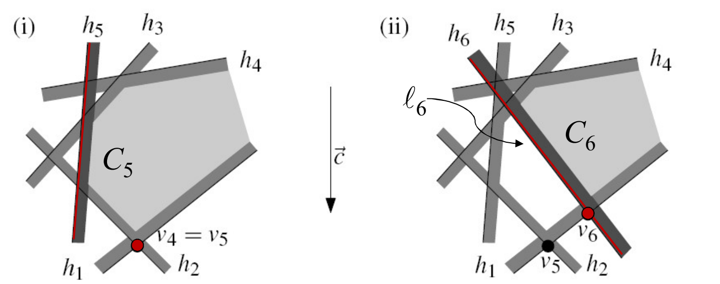
- Since $C_0 \supseteq C_1 \ldots \supseteq C_n = C$, if $C_i = \varnothing$ then $C_j = \varnothing$, for $j > i$ and we can report $(H,\mathbf{c})$ is infeasible

How does the solution change?

Theorem. Let v_i be the optimal solution of (H_i, $\boldsymbol{c}$). Then

i. If $v_{i-1} \in h_i$ then $v_i = v_{i-1}$

ii. If $v_{i-1} \notin h_i$ then either C_i is empty (infeasible) or $v_i \in \ell_i$

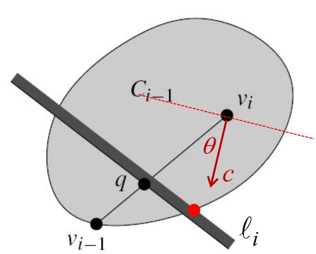


29

Proof

i. Since $C_{i-1} \supseteq C_i$, then $v_i \in C_{i-1}$, so v_i cannot be better than v_{i-1}. Therefore, $v_i = v_{i-1}$

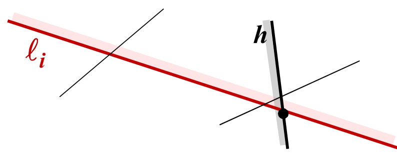
ii. (*By contradiction*) Assume C_i is not empty and $v_i \notin \ell_i$. Since $v_{i-1} \notin h_i$ and $v_i \in h_i$, then the segment $v_{i-1}v_i$ must intersect ℓ_i, at a point q. By convexity, this segment is contained in C_{i-1}, and $q \in C_i$. Since f_c is linear, it increases monotonically from v_i to v_{i-1}. Therefore, $f_c(q) > f_c(v_i)$, a contradiction



30

Case (ii)

- If $v_{i-1} \notin h_i$ we need to solve the following:
  Find the point p on ℓ_i that maximizes $f_c(p)$, subject to $p \in h$, for all $h \in H_{i-1}$
- After re-parameterizing ℓ_i as a function of a single parameter (e.g., t, x- or y-coordinate) we are left with a 1D linear program!
  Let $\sigma(h, \ell_i)$ be the x-coordinate of the intersection of h and ℓ_i
- Can this 1D linear program be unbounded?

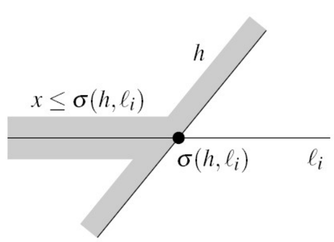


$\Longrightarrow$ $x \leq \sigma(h, \ell_i)$

31

Case (ii)…

- Find the point p on ℓ_i that maximizes $f_c(p)$, subject to $p \in h$, for all $h \in H_{i-1}$
- Parameterize ℓ_i, say on x-coordinate, (or y- if ℓ_i is vertical) and let $\sigma(h_j, \ell_i)$ denote the parameterized intersection of ℓ_i and ℓ_j
- Each constraint h of H_i, when restricted to ℓ_i has the form $x \leq \sigma(h, \ell_i)$ or $x \geq \sigma(h, \ell_i)$
- This is a 1D linear program and can be solved in $O(i)$ time



32

2d Bounded Linear Programming

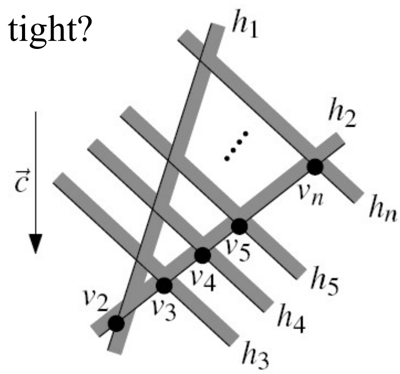
Algorithm 2DBoundedLP(H, c, m_1, m_2)

Let v_0 be the corner of C_0 // where $C_0 = m_1 \cap m_2$
Let $h_1,\ldots,h_n$ be the planes of H
for $i \leftarrow 1$ **to** n **do**
  if $v_{i-1} \in h_i$
    then $v_i \leftarrow v_{i-1}$
    else $v_i \leftarrow$ 1DLP(H_{i-1}, ℓ_i, c)
      if v_i does not exist then (H,c) is infeasible
return v_n

33

Performance of 2DBoundedLP

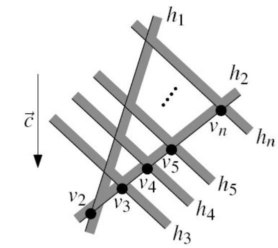
- In the worst case, 2DBoundedLP will have to solve a 1D linear program with every iteration
- Running time is $\sum i = O(n^2)$
- Is this bound tight?



34

What went wrong?

- Is the problem inherent to the input planes?
- What happens if the planes are added in the order
  $h_1, h_2, h_n, h_{n-1}, h_{n-2}, \ldots, h_3$
- Is it the case that for *any* (H,c) there is a "good order"? If so, how do we find it?



35

2d Randomized Bounded LP

Algorithm 2DRandomizedBoundedLP(H, c, m_1, m_2)

1. Let v_0 be the corner of C_0 // where $C_0 = m_1 \cap m_2$
2. Compute a random permutation $h_1,\ldots,h_n$ of the planes of H
3. **for** $i \leftarrow 1$ **to** n **do**
4. if $v_{i-1} \in h_i$
5. **then** $v_i \leftarrow v_{i-1}$
6. **else** $v_i \leftarrow$ 1DLP(H_{i-1}, ℓ_i, c)
7. if v_i does not exist then (H,c) is infeasible
8. return v_n

36

Randomized performance

Theorem. The randomized incremental 2D linear programming algorithm runs in $O(n)$ expected time

Proof sketch

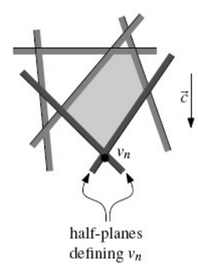
- There are n iterations
- An iteration takes constant time if the optimal vertex didn't change; otherwise we need to run a linear time algorithm
- Let X = total time spent solving 1D linear programs, and $X_i = I(v_{i-1} \notin h_i)$

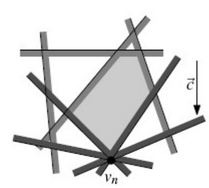
$$E(X) = E\left(\sum_{i=1}^{n} ciX_i\right) = c\left(\sum_{i=1}^{n} iE(X_i)\right) = c\left(\sum_{i=1}^{n} i\,\Pr(v_{i-1} \notin h_i)\right)$$

37

What is $\Pr(v_{i-1} \notin h_i)$?

- Run the algorithm backwards!
  - Assume algorithm has just finished and computed v_n
  - Perform one step back. C_{n-1} is obtained by removing halfplane h_n from C_n.
  - When does the optimal solution change?
    - Only if h_n is one of two halfplanes defining v_n !
  - Probability of change is *at most* $2/n$





38

Performance of Randomized LP

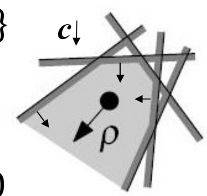
- In general, the probability that we need to compute a new optimal solution when adding h_i is the same as the probability that the optimal vertex changes when we remove h_i
- $E(X_i) \le 2/i$
- Since $X = c\Sigma\, i\, X_i$

$$E(X) = c\left(\sum_{i=1}^{n} iE(X_i)\right) = c\left(\sum_{i=1}^{n} i\Pr(v_{i-1} \notin h_i)\right) \le c\left(\sum_{i=1}^{n} i\frac{2}{i}\right) = O(n)$$

39

Unbounded Programs

- How do we recognize an unbounded LP (H,c)?
- Unbounded $\Rightarrow$ there is a ray $\rho = \{p + \lambda d\}$ contained in C along which f_c increases



Unbounded $\Rightarrow$ (a) $n_h \cdot d \ge 0$, $\forall h \in H$, where n_h = inward normal of h, and (b) $c \cdot d > 0$

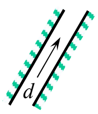
Theorem. A linear program (H,c) is unbounded iff there is a vector d that satisfies (1) $c \cdot d > 0$, (2) $n_h \cdot d \ge 0$, for all $h \in H$, (3) the program (H',c) is feasible, where $H' = \{h \in H\colon n_h \cdot d = 0\}$

Proof sketch. ($\Rightarrow$) obvious. ($\Leftarrow$) $p_0 \in H'$ exists, as H' is non-empty and f takes arbitrarily large values along $p_0 + \lambda d$. For planes $h \in H \setminus H'$ there is λ_h, such that $p_0 + \lambda d \in h$, for $\lambda \ge \lambda_h$. Take $\lambda' = \max \lambda_h$ and $p = p_0 + \lambda' d$. Then $\rho = p + \lambda d \in h$, for all $h \in H$. Hence (H, c) is unbounded.

40

Detecting an Unbounded Program

- Similar to half-plane intersection, but 1D only!
  - Wlog assume $c = (0,1)$
  - A direction d satisfying $d \cdot c > 0$ must be “pointing up” $\Rightarrow d$ can be normalized to $d = (d_x, 1)$ (a position vector on the line y=1)
  - The constraint $d \cdot n_h \geq 0$ becomes $d_x n_x \geq -n_y$ resulting in a 1D *half-space intersection problem* $\overline{H}$.
  - If $\overline{H}$ is feasible (w/solution d_x^*), we find $H' \subseteq H$, the halfplanes for which the solution is “tight” (bounded by lines parallel to d), and verify that H' is feasible. If so, (H, c) is unbounded; otherwise H' is infeasible and so is (H, c)



- If $\overline{H}$ is infeasible, then (H,c) is bounded and two mutually incompatible halfspaces from $\overline{H}$ constitute a certificate for the boundedness of (H,c) and can be used in lieu of m_1 and m_2

41

Algorithm 2DRANDOMIZEDLP$(H, \vec{c})$
Input. A linear program $(H, \vec{c})$, where H is a set of n half-planes and $\vec{c} \in \mathbb{R}^2$.
Output. If $(H, \vec{c})$ is unbounded, a ray is reported. If it is infeasible, then two or three certificate half-planes are reported. Otherwise, the lexicographically smallest point p that maximizes $f_{\vec{c}}(p)$ is reported.

Determine whether there is a direction vector $\vec{d}$ such that $\vec{d} \cdot \vec{c} > 0$ and $\vec{d} \cdot \vec{\eta}(h) \geqslant 0$ for all $h \in H$.
if $\vec{d}$ exists
  then compute H' and determine whether H' is feasible.
    if H' is feasible
      then Report a ray proving that $(H, \vec{c})$ is unbounded and quit.
      else Report that $(H, \vec{c})$ is infeasible and quit.
Let $h_1, h_2 \in H$ be certificates proving that $(H, \vec{c})$ is bounded and has a unique lexicographically smallest solution.
Let v_2 be the intersection of ℓ_1 and ℓ_2.
Let $h_3, h_4, \dots, h_n$ be a random permutation of the remaining half-planes in H.
for $i \leftarrow 3$ **to** n
  do if $v_{i-1} \in h_i$
    then $v_i \leftarrow v_{i-1}$
    else $v_i \leftarrow$the point p on ℓ_i that maximizes $f_{\vec{c}}(p)$, subject to the constraints in H_{i-1}.
      if p does not exist
        then Let h_j, h_k (with $j, k < i$) be the certificates (possibly $h_j = h_k$) with $h_j \cap h_k \cap \ell_i = \emptyset$.
          Report that the linear program is infeasible, with h_i, h_j, h_k as certificates, and quit.
return v_n

42

Higher Dimensions

- Approach generalizes easily to $d \geq 3$ dimensions
- Keep track of a unique solution
  - First determine if program is unbounded. If not, get d certificates $h_1,\ldots,h_d$ of boundedness, to be used as base case
  - Now insert $h_{d+1},\ldots,h_n$ in random order keeping track of the unique (and lexicographically smallest) solution
  - As before, if $v_{i-1} \notin h_i$, then $v_i \in g_i$, the hyperplane bounding h_i. In this case we need to find the optimal vertex of $g_i \cap C_{i-1}$
  - This is LP in dimension $d-1$, as f_c induces a linear function in g_i

Theorem. For each fixed dimension d, a d-dimensional linear program with n constraints can be solved in $O(n)$ expected time

43

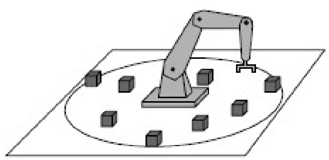
Algorithm RANDOMIZEDLP$(H,\vec{c})$
Input. A linear program $(H,\vec{c})$, where H is a set of n half-spaces in $\mathbb{R}^d$ and $\vec{c} \in \mathbb{R}^d$.
Output. If $(H,\vec{c})$ is unbounded, a ray is reported. If it is infeasible, then at most $d+1$ certificate half-planes are reported. Otherwise, the lexicographically smallest point p that maximizes $f_{\vec{c}}(p)$ is reported.

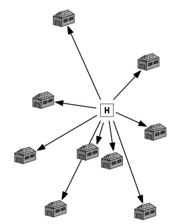
1. Determine whether a direction vector $\vec{d}$ exists such that $\vec{d} \cdot \vec{c} > 0$ and $\vec{d} \cdot \vec{\eta}(h) \geqslant 0$ for all $h \in H$.
2. **if** $\vec{d}$ exists
3.   **then** compute H' and determine whether H' is feasible.
4.   **if** H' is feasible
5.   **then** Report a ray proving that $(H,\vec{c})$ is unbounded and quit.
6.   **else** Report that $(H,\vec{c})$ is infeasible, provide certificates, and quit.
7. Let $h_1,h_2,\ldots,h_d$ be certificates proving that $(H,\vec{c})$ is bounded.
8. Let v_d be the intersection of $g_1,g_2,\ldots,g_d$.
9. Compute a random permutation $h_{d+1},\ldots,h_n$ of the remaining half-spaces in H.
10. **for** $i \leftarrow d+1$ **to** n
11.   **do if** $v_{i-1} \in h_i$
12.   **then** $v_i \leftarrow v_{i-1}$
13.   **else** $v_i \leftarrow$the point p on g_i that maximizes $f_{\vec{c}}(p)$, subject to the constraints $\{h_1,\ldots,h_{i-1}\}$
14.   **if** p does not exist
15.   **then** Let H^* be the at most d certificates for the infeasibility of the $(d-1)$-dimensional program.
16.   Report that the linear program is infeasible, with $H^* \cup h_i$ as certificates, and quit.
17. **return** v_n

44

Smallest Enclosing Disc

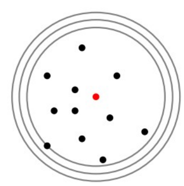
- Find the smallest disc enclosing a set $P = \{p_1, \dots, p_n\}$ of points in the plane
- Applications
  - Where should we place an antenna serving n locations so that the locations have maximum reception?
  - Where do we anchor a robot arm so that all objects in a set of n objects can be reached while minimizing the length of the arm?
  - Given a set of n houses in an isolated area, is there a location that would allow a helicopter ambulance to reach every house in 15 minutes or less?



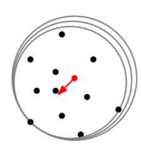


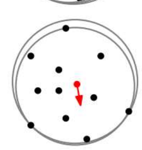
45

A Brute Force Algorithm



- How would a brute force algorithm work?
- *Key observation*. Smallest disc must contain some input points on the boundary
- *Idea*. If we limit the number of points on the boundary, we end up with a finite number of possibilities. But how many points are needed? 1, 2, 3, ….?
  - Starting with any disc that encloses the points, reduce its radius until it touches a point p. Now, move the center towards p until the boundary touches another point q
  - The new center lies on the bisector of p and q. Move it towards the midpoint of pq until the boundary contains a third point or the midpoint is reached
- *Claim*. The boundary of the smallest disc either has three points or two antipodal points. These points subdivide the circle into arcs of length at most π





46

A Brute Force Algorithm…

1. In $\Theta(n^3)$ time enumerate all triples of input points. For each triple, compute the resulting circle and, in additional $\Theta(n)$ time, check if it encloses all points. Keep track of the smallest enclosing circle
2. Then, for each pair of points, compute the circle with the pair as diameter, and check if it encloses all points. Keep track of the smallest enclosing circle
3. Report the best of (1) and (2)

- Running time? $O(n^4)$

Exercise. How do you find the circle through 3 points?

47

Linear Programming?

- Optimum circle has center $C = (c_x, c_y)$ and radius r
- We wish to find C, r with minimum r subject to

$$\left(p_{i,x} - c_x\right)^2 + \left(p_{i,y} - c_y\right)^2 \leq r^2, \text{ for } i = 1, \ldots, n$$

- Can we *linearize* the inequalities?

$$p_{i,x}^2 - 2p_{i,x}c_x + c_x^2 + p_{i,y}^2 - 2p_{i,y}c_y + c_y^2 \leq r^2$$

$$2p_{i,x}c_x + 2p_{i,y}c_y + \left(r^2 - c_x^2 - c_y^2\right) \geq p_{i,x}^2 + p_{i,y}^2$$

$$(2p_{i,x}c_x) + (2p_{i,y})c_y + R \geq p_{i,x}^2 + p_{i,y}^2 \quad (i = 1, \ldots, n)$$

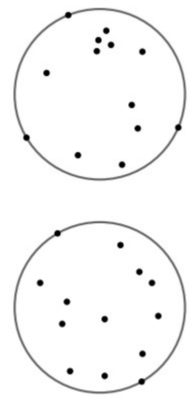
- What is the new objective function?

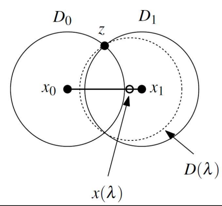
Minimize $r^2 = R + c_x^2 + c_y^2$, but this is not linear!

48

Some Useful Properties

- Let D be a minimum enclosing disc of a set of points P. Then
  1) D is unique
  2) D contains either 3 points or 2 antipodal points on its boundary that divide the circle into arcs of length at most π
  3) If D' is the minimum enclosing disc of P' and $P' \subset P$, then $radius(D') \leq radius(D)$



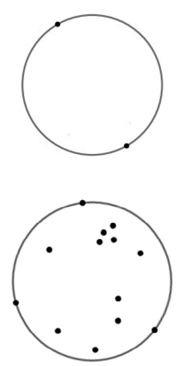


49

Smallest Enclosing Disc…

- Does an incremental randomized algorithm work?
  Let $P_i = \{p_1,\ldots,p_i\}$ and D_i = the smallest enclosing disc of P_i

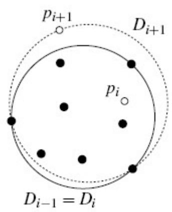
Algorithm MiniDisc(P)
1. Compute a random permutation of P
2. Let D_2 be the circle with diameter $p_1 p_2$
3. **for** $i \leftarrow 3$ **to** n **do**
   compute D_i from D_{i-1}

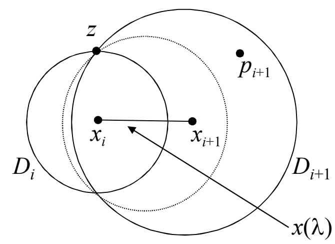


50

Lemma. Let $2 \leq i < n$. Then

1. If $p_{i+1} \in D_i$, then $D_{i+1} = D_i$
2. If $p_{i+1} \notin D_i$, then p_{i+1} lies on the boundary of D_{i+1}



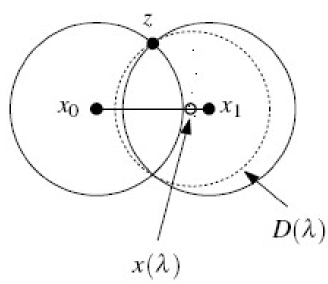


Proof of Case 2 (by contradiction). Suppose $p_{i+1} \notin \partial D_i$. We must have $\partial D_i \cap \partial D_{i+1} \neq \emptyset$. Let $z \in \partial D_i \cap \partial D_{i+1}$

Consider the discs $D(\lambda)$ with centers $x(\lambda) = (1-\lambda)x_i + \lambda x_{i+1}$ and $z \in \partial D(\lambda)$. Their size (hence their radius) increases continuously as $\lambda \to 1$. At some point $\partial D(\lambda^*)$ must contain p_{i+1} 51

Lemma. Let P be a set of points in the plane and R a possibly empty set of points disjoint from P. Then:

1. The smallest enclosing disc $md(P, R)$ that encloses P and has all points of R on its boundary is unique
2. If $p \in md(P \setminus \{p\}, R)$ then $md(P, R) = md(P \setminus \{p\}, R)$
3. If $p \notin md(P \setminus \{p\}, R)$ then $md(P, R) = md(P \setminus \{p\}, R \cup \{p\})$

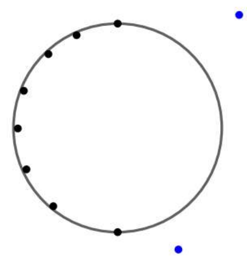


Proof. Similar to previous lemma

52

Question

- Suppose you have computed D_i and know the 2 or 3 points on its boundary.
- Suppose further that p_{i+1} is outside D_i.We know that p_{i+1} must lie on the boundary of D_{i+1}
- Is it the case that D_{i+1} is defined by p_{i+1} and some of the points on the boundary of D_i?



An Incremental Randomized Algorithm

Algorithm MINIDISC(P)
Input. A set P of n points in the plane.
Output. The smallest enclosing disc for P.

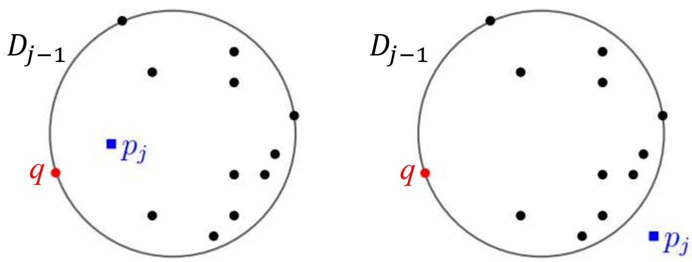
Compute a random permutation $p_1, \ldots, p_n$ of P.
Let D_2 be the smallest enclosing disc for $\{p_1, p_2\}$.
for $i \leftarrow 3$ **to** n
    do if $p_i \in D_{i-1}$
        then $D_i \leftarrow D_{i-1}$
        else $D_i \leftarrow$ MINIDISCWITHPOINT($\{p_1, \ldots, p_{i-1}\}, p_i$)
return D_n

- How do you solve MiniDiscWithPoint?

Smallest Disc With One Point Known

MINIDISCWITHPOINT(P, q)
Input. A set P of n points in the plane, and a point q such that there exists an enclosing disc for P with q on its boundary.
Output. The smallest enclosing disc for P with q on its boundary.

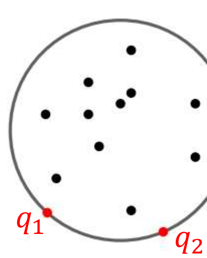
Compute a random permutation $p_1, \ldots, p_n$ of P.
Let D_1 be the smallest disc with q and p_1 on its boundary.
for $j \leftarrow 2$ **to** n
  do if $p_j \in D_{j-1}$
    then $D_j \leftarrow D_{j-1}$
    else $D_j \leftarrow$ MINIDISCWITH2POINTS($\{p_1, \ldots, p_{j-1}\}, p_j, q$)
return D_n



Smallest Disc With Two Points Known

MINIDISCWITH2POINTS(P, q_1, q_2)
Input. A set P of n points in the plane, and two points q_1 and q_2 such that there exists an enclosing disc for P with q_1 and q_2 on its boundary.
Output. The smallest enclosing disc for P with q_1 and q_2 on its boundary.

Let D_0 be the smallest disc with q_1 and q_2 on its boundary.
for $k \leftarrow 1$ **to** n
  do if $p_k \in D_{k-1}$
    then $D_k \leftarrow D_{k-1}$
    else $D_k \leftarrow$ the disc with q_1, q_2, and p_k on its boundary
return D_n



Theorem. The smallest enclosing disc for a set of n points in the plane can be computed in $O(n)$ expected time and $O(n)$ space.

Proof.

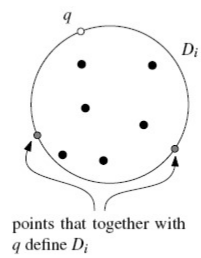
The running time for MiniDiscWith2Points is $O(n)$.

By backwards analysis, MiniDiscWithPoint runs in $O(n)$ expected time

For fixed $\{p_1,\ldots,p_i\}$, the probability that $D_{i-1} \neq D_i$ is $\leq 2/i$

The expected running time is

$$O(n) + \sum_{i=2}^{n} \frac{2}{i} O(i) = O(n)$$



Same argument implies that the expected running time of MiniDisc is $O(n)$

57

When does this approach work?

- The test whether the next input changes the solution so far must be possible and fast
- Finding the new solution must be easier than the original problem
- Only $O(1)$ new objects are created in the new solution

58