Prune-and-search Strategy

Outlines

- The general method
- The selection problem
- Linear programming with two variable
- The 1-center problem

學習目標

- Prune-and-Search策略設計的概念
- Prune-and-Search演算法時間複雜度分析

The general method P&S

- The prune-and-search strategy always consists of several iterations.
- At each iteration, it prunes away a fraction, say f $(0<f<1)$ of the input data, and then it invokes the same algorithm recursively to solve the problem for the remaining data.
- After p iterations, the size of input data will be q, which is so small that the problem can be solved directly in some constant time c'.

The time-complexity analysis

- **Assume that the time needed to execute the prune-and-search in each iteration is $O(n^k)$ for some constant k, and the worst case run time of the prune-and-search algorithm is $T(n)$.**

- ***Then*** $T(n)=T((1-f)n)+O(n^k)$

  $\Rightarrow T(n)=O(n^k)$

The general prune-and-search

- It consists of many iterations.
- At each iteration, it prunes away a fraction, say f, of the input data, and then it invokes the same algorithm recursively to solve the problem for the remaining data.
- After p iterations, the size of input data will be q which is so small that the problem can be solved directly in some constant time c.
- Assume that the time needed to execute the prune-and-search in each iteration is $O(n^k)$ for some constant k and the worst case run time of the prune-and-search algorithm is $T(n)$. Then

$$T(n) = T((1-f)\, n) + O(n^k)$$

Time complexity

- We have

$$T(n) \le T((1-f)\,n) + cn^k \qquad \text{for sufficiently large } n.$$
$$\le T((1-f)^2 n) + cn^k + c(1-f)^k n^k$$
$$\vdots$$
$$\le c' + cn^k + c(1-f)^k n^k + c(1-f)^{2k} n^k + \ldots + c(1-f)^{pk} n^k$$
$$= c' + cn^k(1 + (1-f)^k + (1-f)^{2k} + \ldots + (1-f)^{pk}).$$

Since $1 - f < 1$, as $n \to \infty$,

$\therefore\ T(n) = O(n^k)$

- Thus, the time-complexity of the whole prune-and-search process is of the same order as the time-complexity in each iteration.

A simple example: Binary search

- sorted sequence : (search 9)

	1	4	5	7	9	10	12	15
step 1			↑					
step 2					↑			
step 3				↑				

- After each comparison, a half of the data set are pruned away.
- Binary search can be viewed as a special divide-and-conquer method, since there exists no solution in another half and then no merging is done.

Question:

- **Assume that the time needed to execute the prune-and-search in each iteration is $O(n^2)$. What is the time complexity of the algorithm ?**

(1) $O(n^2)$

(2) $O(n)$

(3) $O(n^2 \log n)$

(4) $O(n \log n)$

Ans. 1

The selection problem

學習目標

- Selection Problem 問題定義
- 以prune-and-search 策略設計Selection Problem的演算法
- 以prune-and-search 策略設計Selection Problem的演算法時間複雜度分析

The selection problem

- Input: A set S of n elements
- Output: The ***kth*** smallest element of S
- The median problem: to find the $\left\lceil \frac{n}{2} \right\rceil$-th smallest element.
- The straightforward algorithm:
  - step 1: Sort the n elements
  - step 2: Locate the kth element in the sorted list.
  - Time complexity: **O(nlogn)**

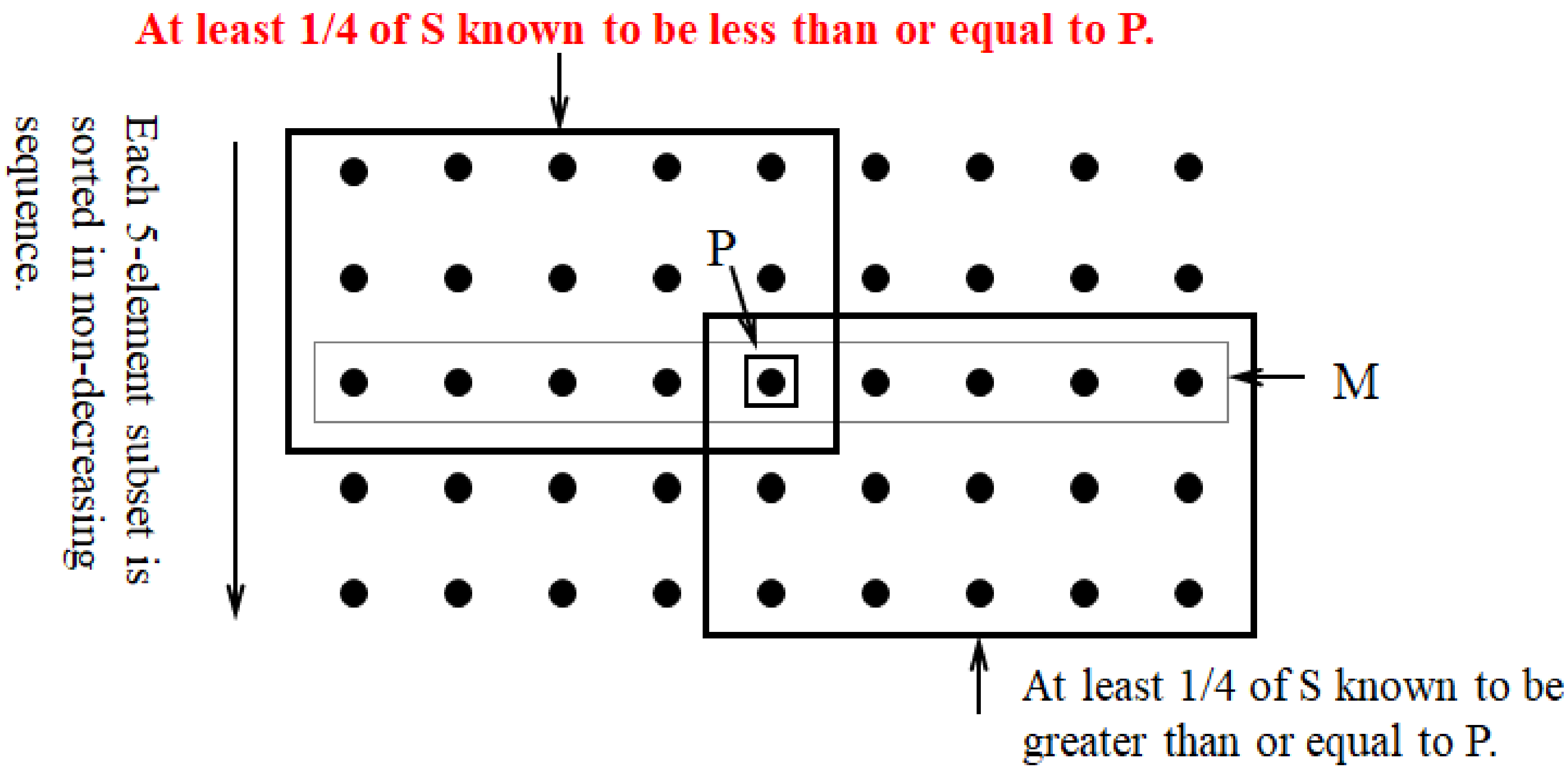
Prune-and-search concept for the selection problem

- $S=\{a_1, a_2, \ldots, a_n\}$
- Let $p \in S$, use p to <u>partition</u> S into 3 subsets S_1 , S_2 , S_3:
  - $S_1=\{ a_i \mid a_i < p , 1 \le i \le n\}$
  - $S_2=\{ a_i \mid a_i = p , 1 \le i \le n\}$
  - $S_3=\{ a_i \mid a_i > p , 1 \le i \le n\}$

- 3 cases:
  - **If $|S_1| > k$** , then the kth smallest element of S is in S_1, prune away S_2 and S_3.
  - Else, **if $|S_1| + |S_2| > k$**, then p is the kth smallest element of S.
  - Else, the kth smallest element of S is the **$(k - |S_1| - |S_2|)$-th** smallest element in S_3, prune away S_1 and S_2.

How to select P?

- The n elements are divided into $\left\lceil \frac{n}{5} \right\rceil$ subsets. (Each subset has 5 elements.)



Prune-and-search approach

- Input: A set S of n elements.
- Output: The kth smallest element of S.

Step 1: Divide S into $\lceil n/5 \rceil$ subsets. Each subset contains five elements. Add some dummy ∞ elements to the last subset if n is not a net multiple of S.

Step 2: Sort each subset of elements.

Step 3: **Find the element p which is the median of the medians of the $\lceil n/5 \rceil$ subsets.**

Prune-and-search approach

Step 4: Partition S into S_1, S_2 and S_3, which contain the elements less than, equal to, and greater than p, respectively.

Step 5: If $|S_1| \geq k$, then discard S_2 and S_3 and solve the problem that selects the kth smallest element from S_1 during the next iteration;

else if $|S_1| + |S_2| \geq k$ then p is the kth smallest element of S;

otherwise, let $k' = k - |S_1| - |S_2|$, solve the problem that selects the k'th smallest element from S_3 during the next iteration.

Example

- S = **{1, 25, 2, 24, 3, 23, 4, 22, 5, 21, 6, 20, 7, 19, 8, 18, 9, 17, 10, 16, 11, 15, 12, 14, 13}**，找第k小的元素。

Ans

[Step 1]

1	23	6	18	11
25	4	20	9	15
2	22	7	17	12
24	5	19	10	14
3	21	8	16	13

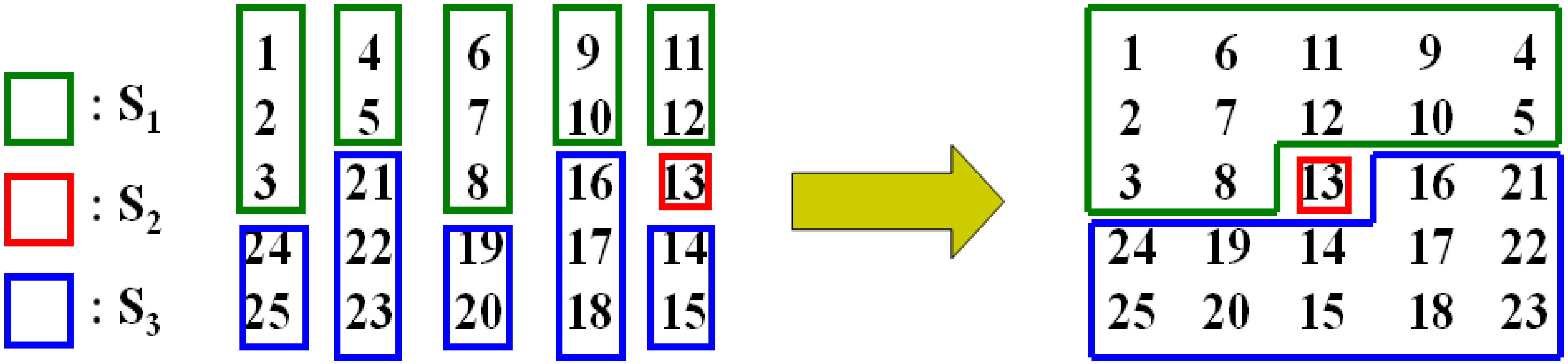
[Step 2]

1	4	6	9	11
2	5	7	10	12
3	21	8	16	13
24	22	19	17	14
25	23	20	18	15

[Step 3]

1	4	6	9	11
2	5	7	10	12
3	21	8	16	$\boxed{13}$
24	22	19	17	14
25	23	20	18	15

[Step 4]



[Step 5] 利用三個判斷條件以找出第k小的元素

- 若 k = 11 (搜尋範圍 $|S_1|$)
- 若 k = 13 (搜尋範圍 $|S_1|+|S_2|$)
- 若 k = 22 (搜尋範圍 $|S_3|$)

Time complexity

- At least n/4 elements are pruned away during each iteration.
- The problem remaining in step 5 contains at most **3n/4** elements.
- Time complexity: T(n) = O(n)
  - step 1: O(n)
  - step 2: O(n)
  - step 3: **T(n/5)**
  - step 4: O(n)
  - step 5: T(3n/4)
  - **T(n) = T(3n/4) + T(n/5) + O(n)**

Let $T(n) = a_0 + a_1 n + a_2 n^2 + \ldots \quad , a_1 \neq 0$

$T(3n/4) = a_0 + (3/4)a_1 n + (9/16)a_2 n^2 + \ldots$

$T(n/5) \quad = a_0 + (1/5)a_1 n + (1/25)a_2 n^2 + \ldots$

$T(3n/4 + n/5) = T(19n/20) = a_0 + (19/20)a_1 n + (361/400)a_2 n^2 + \ldots$

$T(3n/4) + T(n/5) \leq a_0 + T(19n/20)$

$\Rightarrow$ **$T(n) \leq cn + T(19n/20)$**

$\leq cn + (19/20)cn + T((19/20)^2 n)$

$\vdots$

$\leq cn + (19/20)cn + (19/20)^2 cn + \ldots + (19/20)^p cn + T((19/20)^{p+1} n) \quad , (19/20)^{p+1} n \leq 1 \leq (19/20)^p n$

$= \dfrac{1-(\frac{19}{20})^{p+1}}{1-\frac{19}{20}} cn + b$

$\leq 20\ cn + b$

$=$ **O(n)**

Applying the formula obtained in Section 6.1

Question:

- What is the time complexity of the selection problem?

(1) $O(n^2)$

(2) $O(n)$

(3) $O(n^2 \log n)$

(4) $O(n \log n)$

Ans. 2

Linear programming with two variables

學習目標

- Linear Programming Problem 問題定義
- 以prune-and-search 策略設計Linear Programming Problem 的演算法
- 以prune-and-search 策略設計Linear Programming Problem 演算法時間複雜度分析

Linear Programming

Maximize or minimize $c_1 x_1 + c_2 x_2 + \cdots + c_d x_d$

subject to:

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

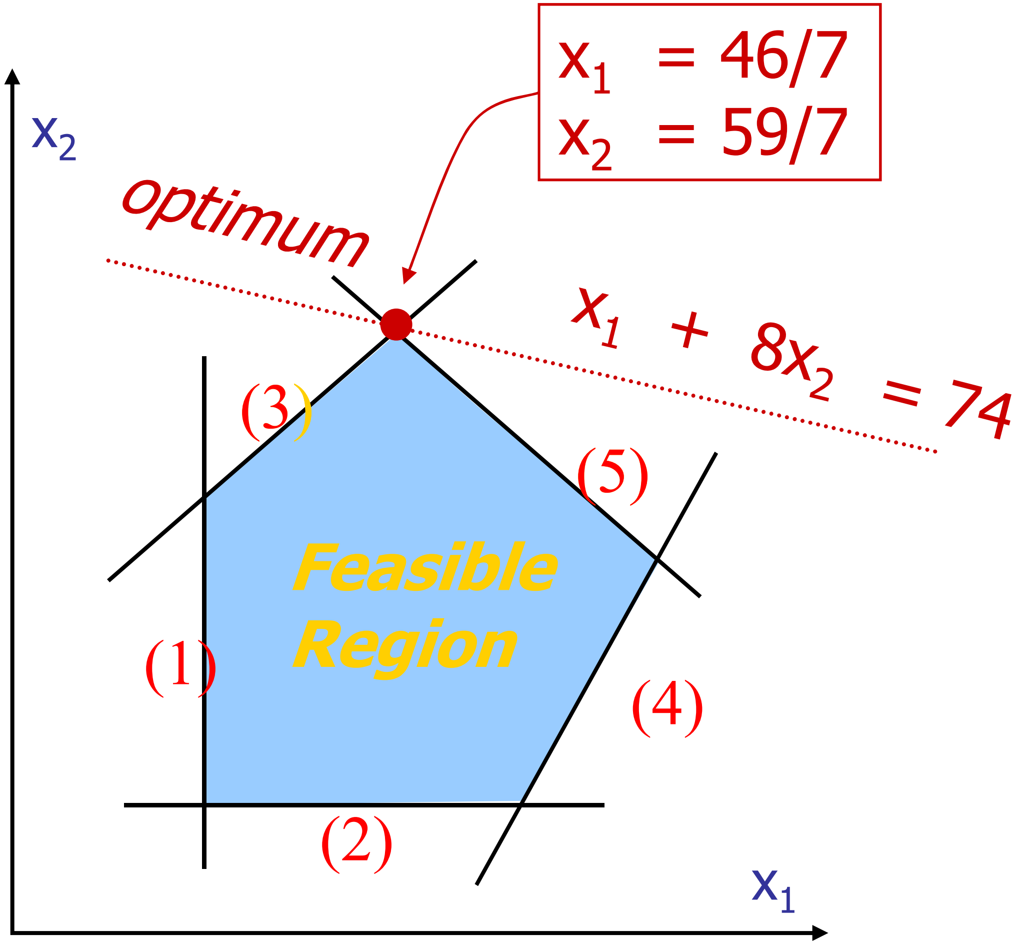
Example in 2D

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

constraints



Linear programming with two variables

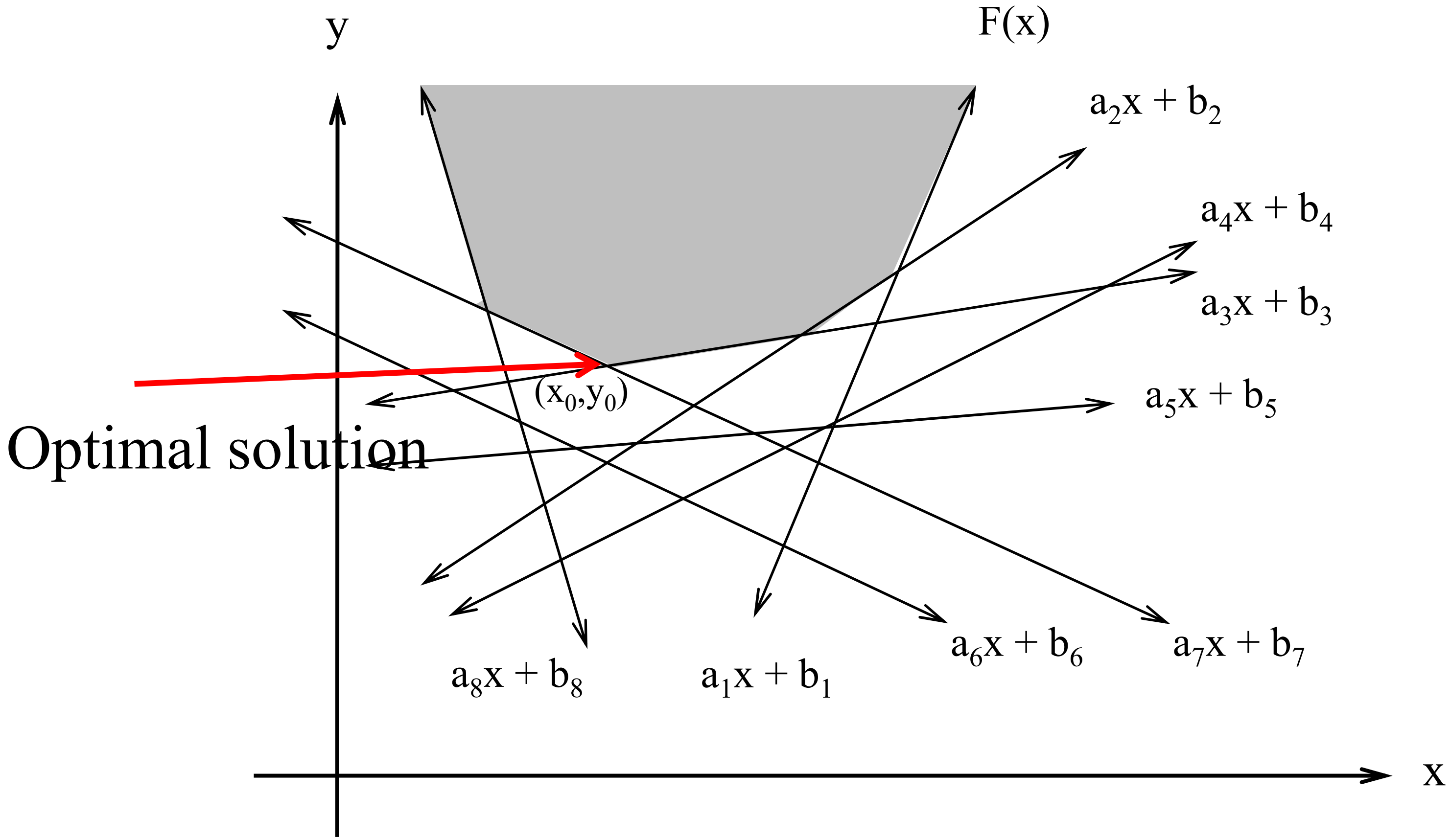
- **Minimize $ax + by$**

  subject to $a_i x + b_i y \geq c_i$, $i = 1, 2, \ldots, n$

- **Simplified two-variable linear programming problem:**

  Minimize y

  subject to $y \geq a_i x + b_i$, $i = 1, 2, \ldots, n$



- The boundary F(x):

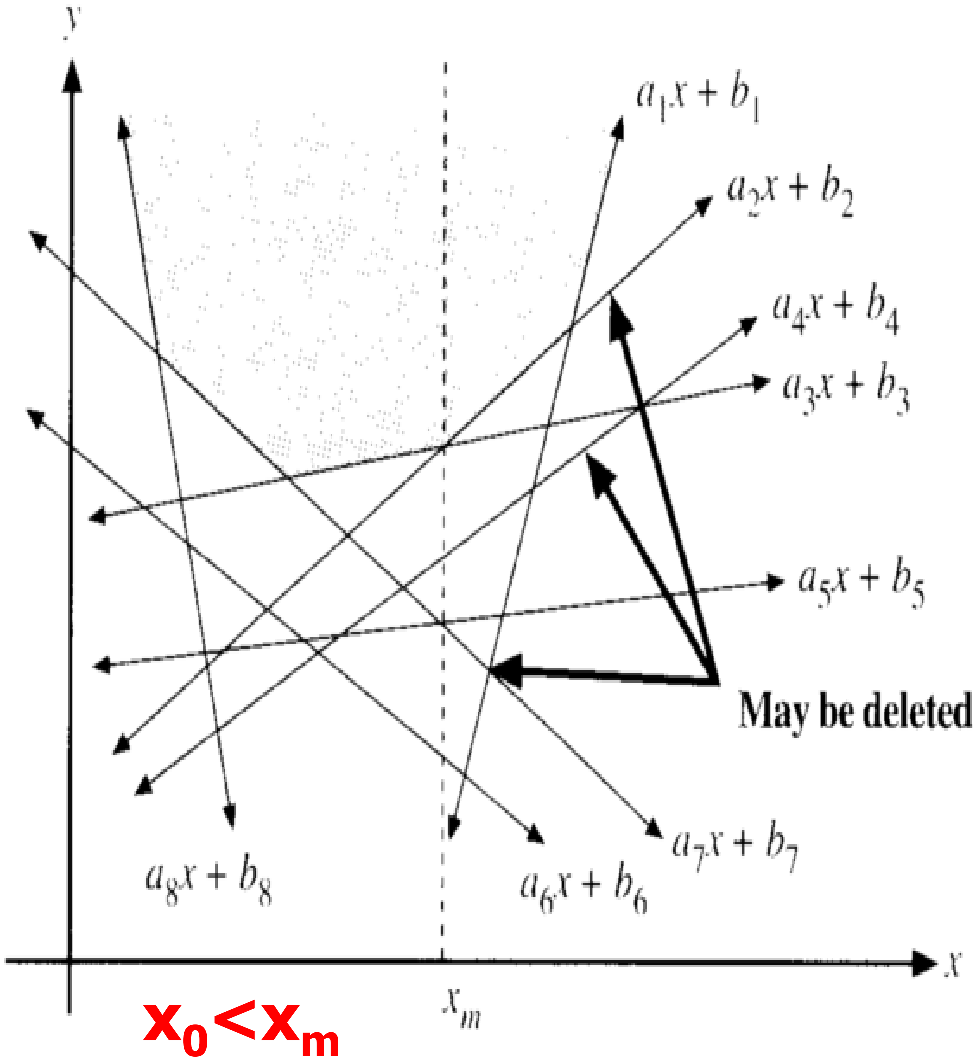
$$F(x) = \max_{1 \le x \le n} \{a_i x + b_i\}$$

- The optimum solution x_0:

$$F(x_0) = \min_{-\infty < x < \infty} F(x)$$

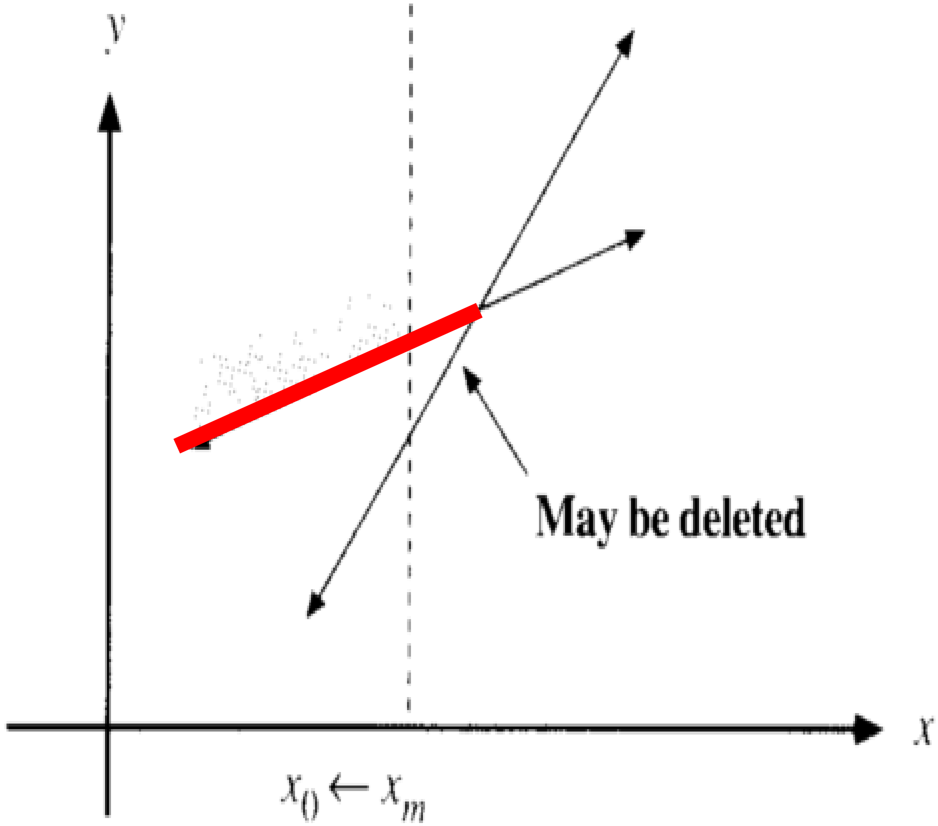
Minimize y
subject to $y \geq a_i x + bi$, i = 1, 2, …, n

Constraints deletion



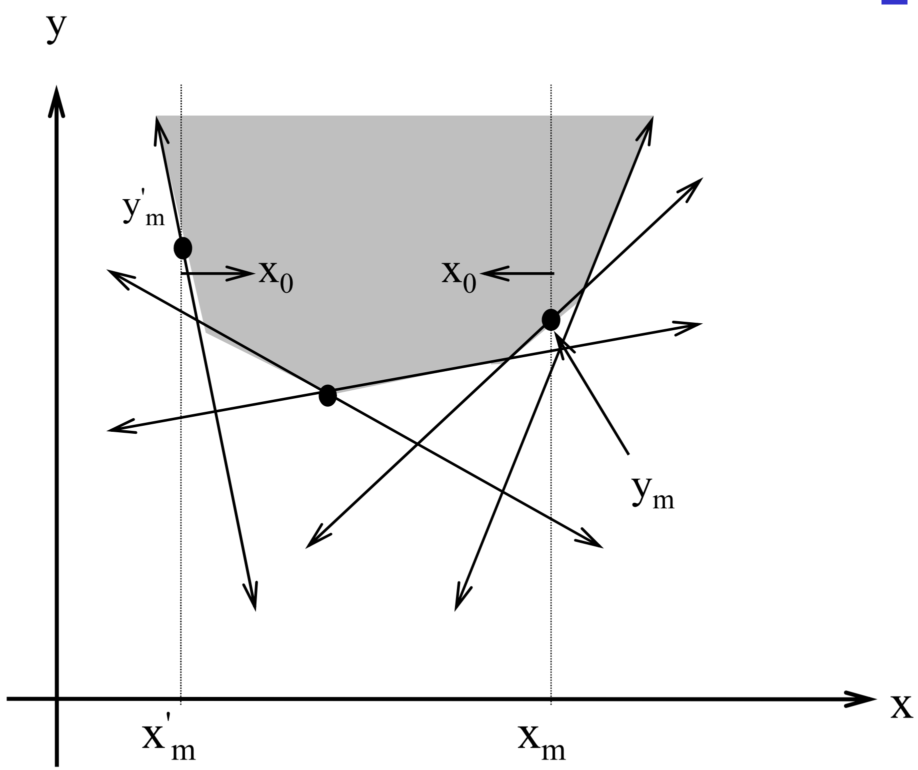
- Assume x_m is known
- If $x_0 < x_m$ and the intersection of $a_3x + b_3$ and $a_2x + b_2$ is greater than x_m, then one of these two constraints is always smaller than the other for $x < x_m$. Thus, this constraint can be deleted.
- It is similar for $x_0 > x_m$.

FIGURE 6–4 An illustration of why a constraint may be eliminated.



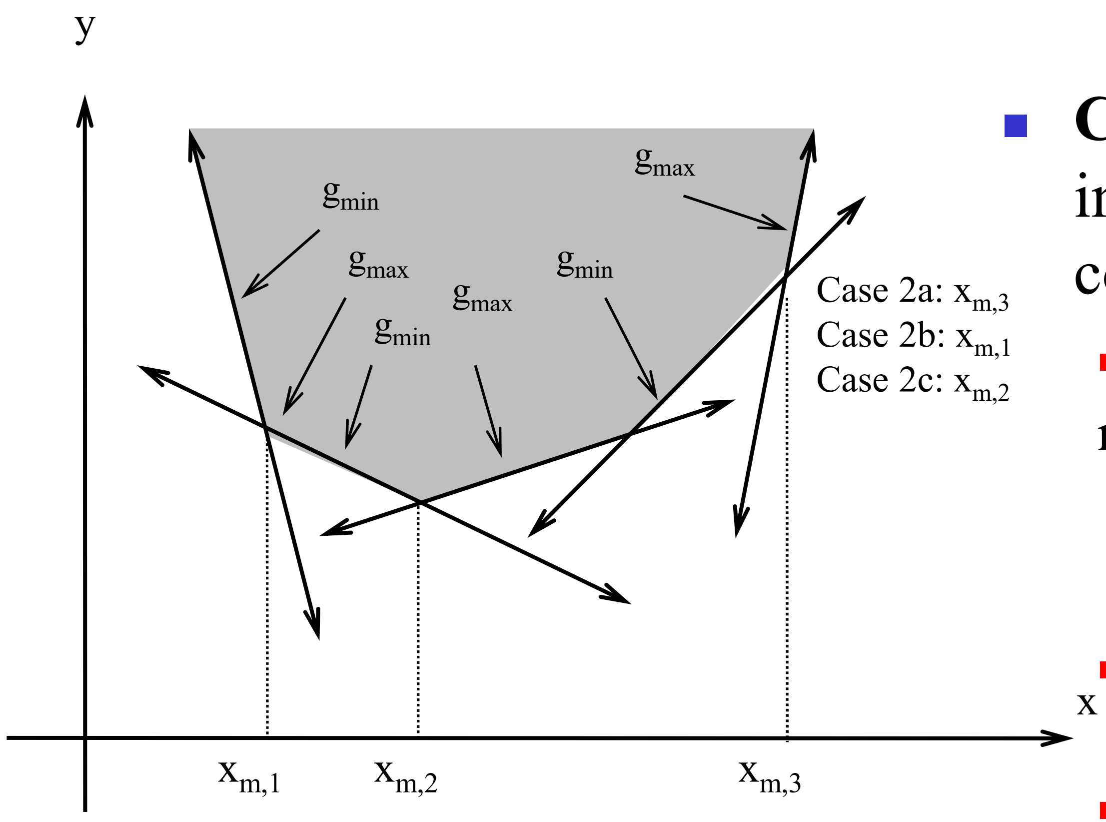
Determining the direction of the optimum solution

Suppose an x_m is known.
How do we know whether
$x_0 < x_m$ or $x_0 > x_m$?



- Suppose x_m is chosen.
- Let $y_m = F(x_m) = \max_{1<i<n} \{a_i x_m + b_i\}$
- **Case 1:** y_m is on only one constraint.
  - Let g denote the slope of this constraint.
  - If $g > 0$, then $x_0 < x_m$.
  - If $g < 0$, then $x_0 > x_m$.

The cases where x_m is on only one constrain.



Cases of x_m on the intersection of several constraints.

- **Case 2:** y_m is the intersection of several constraints.
  - $g_{max} = \max_{1<i<n} \{a_i \mid a_i x_m + b_i = F(x_m)\}$
    max. slope
    $g_{min} = \min_{1<i<n} \{a_i \mid a_i x_m + b_i = F(x_m)\}$
    min. slop
  - **If** $g_{min} > 0$, $g_{max} > 0$, then $x_0 < x_m$
  - **If** $g_{min} < 0$, $g_{max} < 0$, then $x_0 > x_m$
  - **If** $g_{min} < 0$, $g_{max} > 0$, then (x_m, y_m) is the optimum solution.

How to choose x_m?

- We arbitrarily group the n constraints into n/2 pairs. For each pair, find their intersection. Among these n/2 intersections, choose the median of their x-coordinates as x_m.

Prune-and-Search approach

- Input: Constrains S: $a_jx + b_j$, i=1, 2, …, n.
- Output: The value x_0 such that y is minimized at x_0 subject to $y \geq a_jx + b_j$, i=1, 2, …, n.

Step 1: If S contains no more than two constraints, solve this problem by a brute force method.

Step 2: Divide S into n/2 pairs of constraints. For each pair of constraints $a_ix + b_i$ and $a_jx + b_j$, find the intersection p_{ij} of them and denote its x-value as x_{ij}.

Step 3: Among the x_{ij}'s, find the median x_m.

Step 4: Determine $y_m = F(x_m) = \max_{1<i<n} \{a_i x_m + b_i\}$

$$g_{min} = \min_{1<i<n} \{a_i \mid a_i x_m + b_i = F(x_m)\}$$

$$g_{max} = \max_{1<i<n} \{a_i \mid a_i x_m + b_i = F(x_m)\}$$

Step 5:

Case 5a: If g_{min} and g_{max} are not of the same sign, y_m is the solution and exit.

Case 5b: otherwise, $x_0 < x_m$, if $g_{min} > 0$, and $x_0 > x_m$, if $g_{min} < 0$.

Step 6:

Case 6a: If $x_0 < x_m$, for each pair of constraints whose x-coordinate intersection is larger than x_m, prune away the constraint which is always smaller than the other for $x \leq x_m$.

Case 6b: If $x_0 > x_m$, do similarly.

Let S denote the set of remaining constraints. Go to Step 2.

- There are totally $\lfloor n/2 \rfloor$ intersections. Thus, $\lfloor n/4 \rfloor$ constraints are pruned away for each iteration.
- Time complexity: O(n)

Question:

- What is the time complexity of the linear programming problem with two variables?

(1) $O(n^2)$

(2) $O(n)$

(3) $O(n^2 \log n)$

(4) $O(n \log n)$

Ans. 2

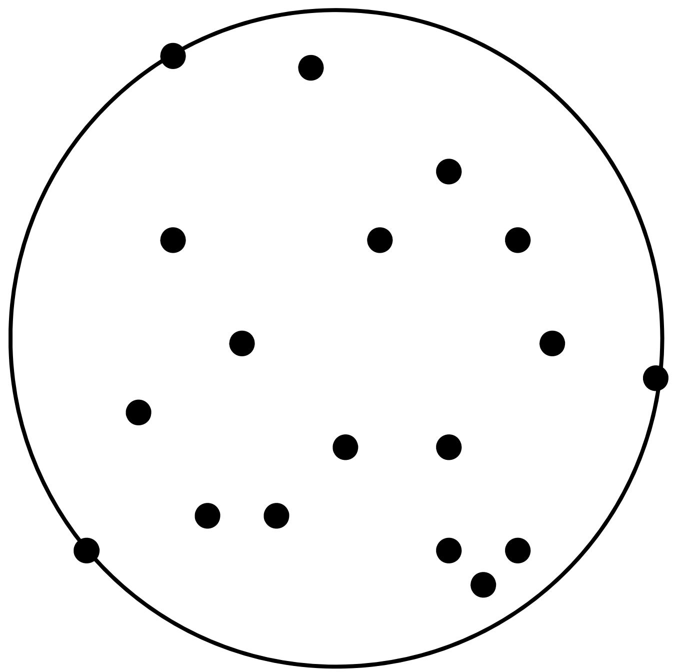
The 1-center problem

學習目標

- 1-center problem 問題定義
- 以prune-and-search 策略設計constrained 1-center problem的演算法
- 以prune-and-search 策略設計constrained 1-center problem的演算法時間複雜度分析

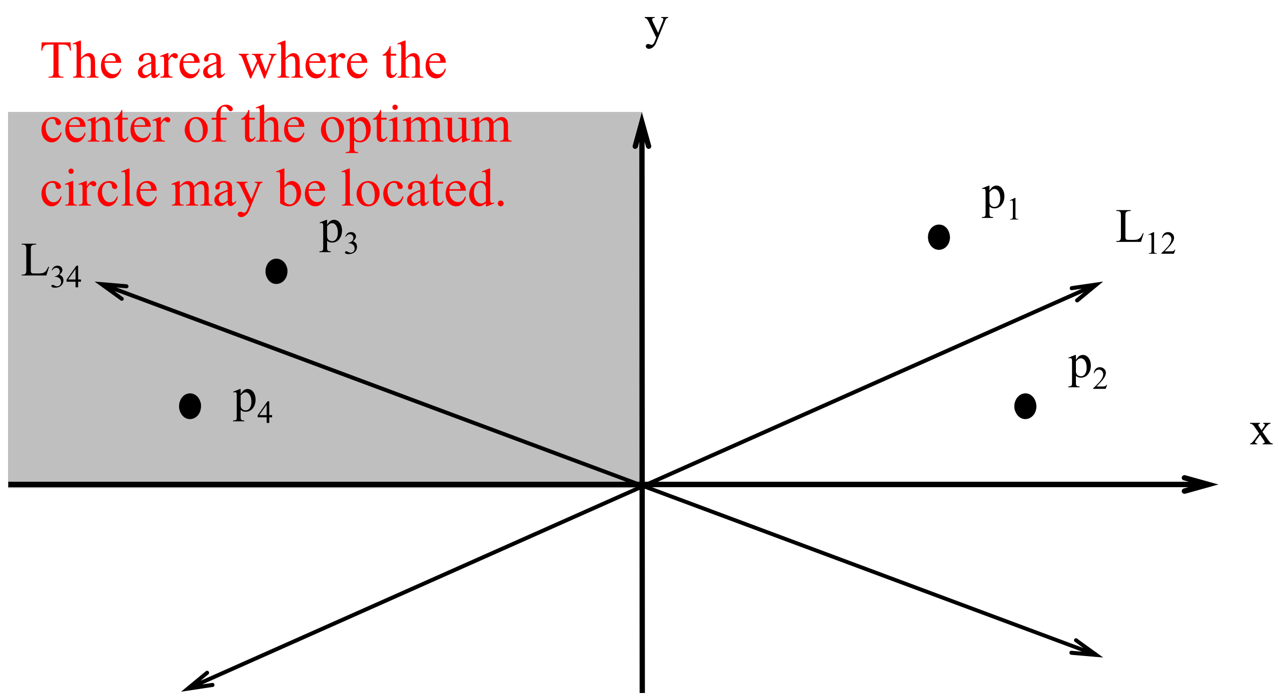
The 1-center problem

- Given n planar points, find a smallest circle to cover these n points.



The pruning rule

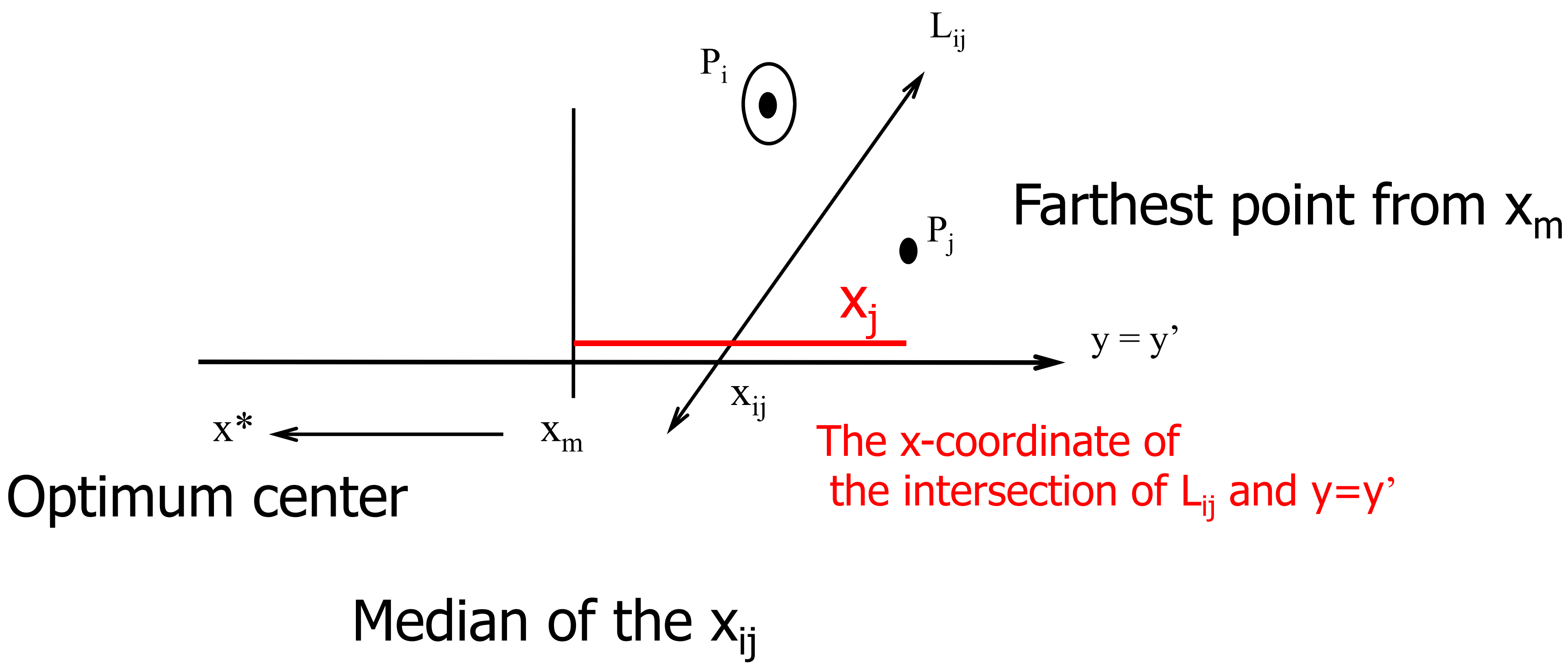
- $L_{1\,2}$: bisector of segment connecting p_1 and p_2 ,
- $L_{3\,4}$: bisector of segments connecting p_3 and p_4
- Assume that the center of optimum circle must be in the shaded area, then P_1 can be eliminated without affecting our solution.



We need to know the region where the optimum center is located?

The constrained 1-center problem

- The center is restricted to lying on a straight line (y=y').



Prune-and-search approach

- Input : n points and a straight line $y = y'$.
- Output: The constrained center on the straight line $y = y'$.

Step 1: If n is no more than 2, solve this problem by a brute-force method.

Step 2: Form disjoint pairs of points (p_1, p_2), (p_3, p_4), …,(p_{n-1}, p_n). If there are odd number of points, just let the final pair be (p_n, p_1).

Step 3: For each pair of points, (p_i, p_{i+1}), find the point $x_{i,i+1}$ on the line $y = y'$ such that $d(p_i, x_{i,i+1}) = d(p_{i+1}, x_{i,i+1})$.

Step 4: Find the **median** of the $\left\lfloor \frac{n}{2} \right\rfloor$ $x_{i,i+1}$'s. Denote it as x_m.

Step 5: Calculate the distance between p_i and x_m for all i. Let p_j be the point which is farthest from x_m. Let x_j denote the projection of p_j onto y = y'. **If x_j is to the left (right) of x_m, then the optimal solution, x^*, must be to the left (right) of x_m.**

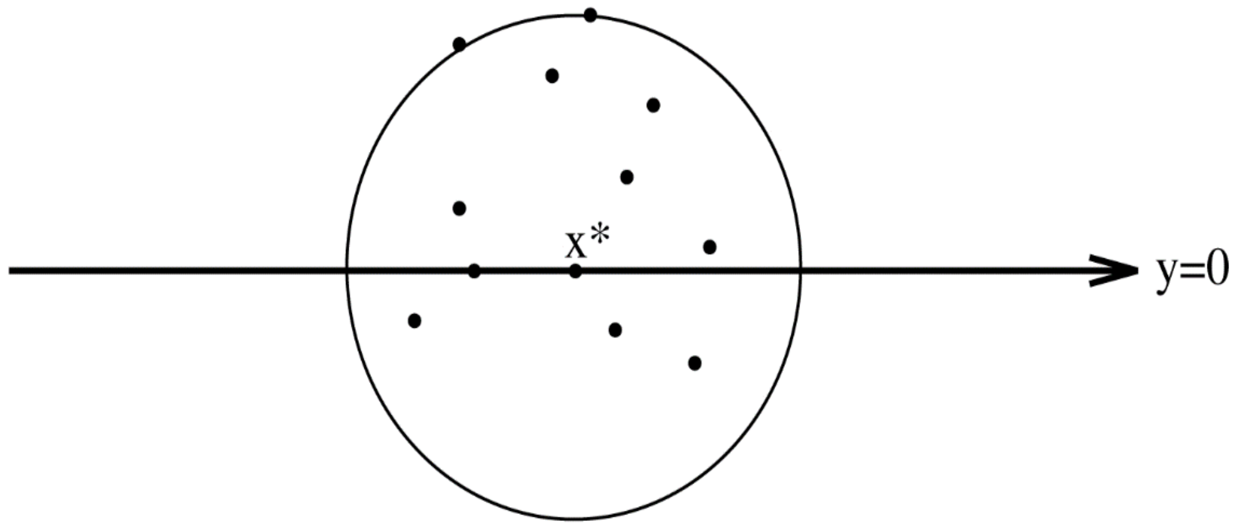
Step 6: If $x^* < x_m$, for each $x_{i,i+1} > x_m$, prune the point p_i if p_i is closer to x_m than p_{i+1}, otherwise prune the point p_{i+1};
If $x^* > x_m$, do similarly.

Step 7: Go to Step 1.

- Time complexity O(n)

The general 1-center problem

- By the constrained 1-center algorithm, we can determine the center $(x^*,0)$ on the line y=0.
- We can do more
  - Let (x_s, y_s) be the center of the optimum circle.
  - We can determine whether $y_s > 0$, $y_s < 0$ or $y_s = 0$.
  - Similarly, we can also determine whether $x_s > 0$, $x_s < 0$ or $x_s = 0$



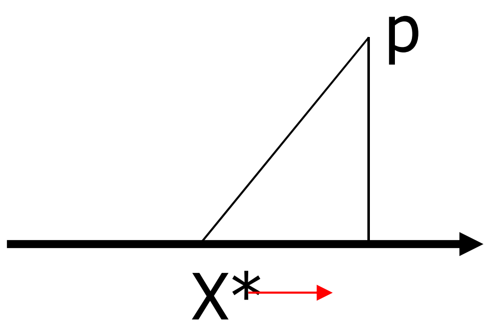
The sign of optimal y

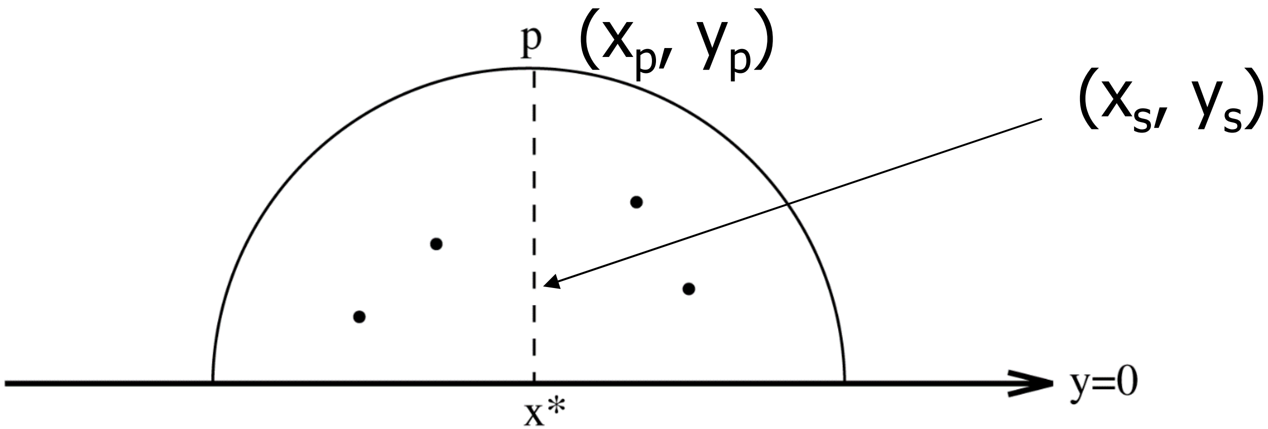
- Let I be the set of points which are farthest from $(x^*, 0)$.
- Case 1: I contains one point $P = (x_p, y_p)$.

y_s has the same sign as that of y_p.

the x-value of p must be equal to x*

(proof by contradiction)





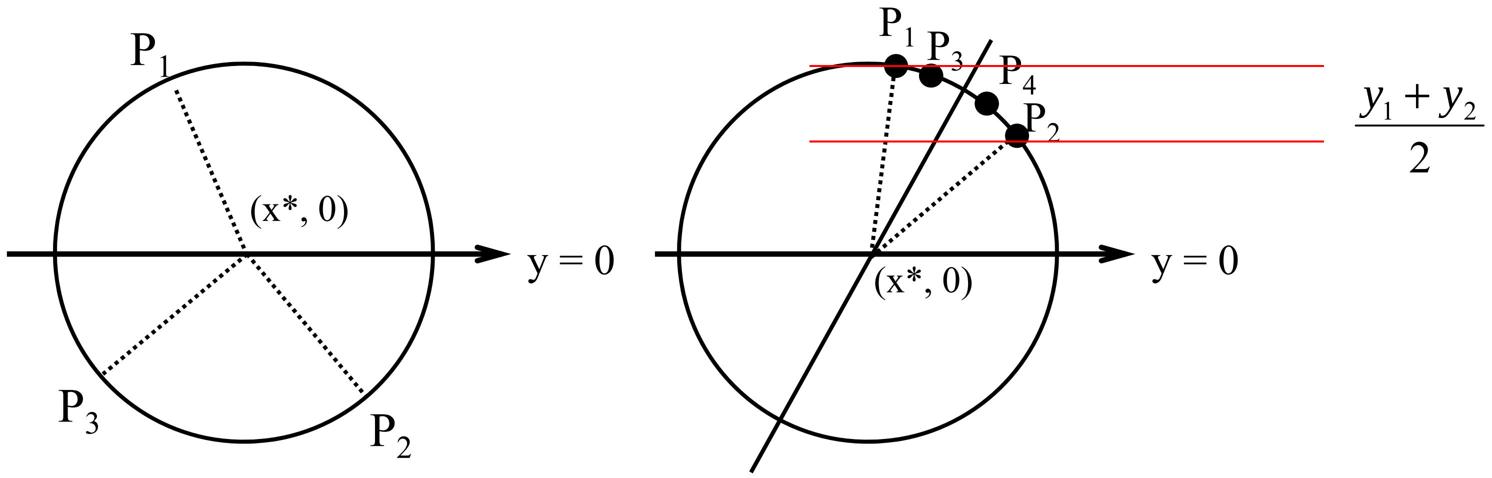
- Case 2 : I contains more than one point.

  Find the smallest arc spanning all points in I.

  Let $P_1 = (x_1, y_1)$ and $P_2 = (x_2, y_2)$ be the two end points of the smallest spanning arc.

  If this arc $\geq 180^o$, then $y_s = 0$.

  else y_s has the same sign as that of .

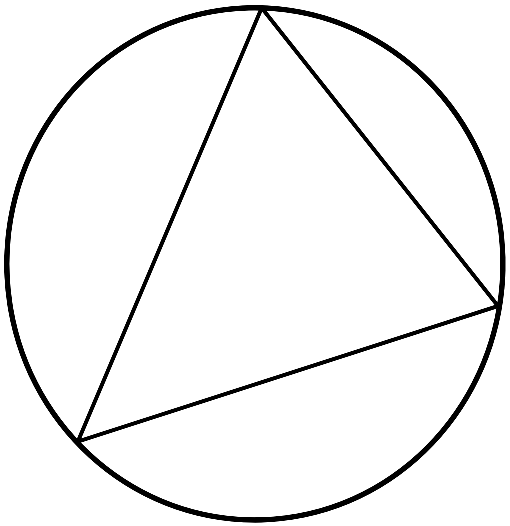


(a) (b)

(See the figure on the next page.)

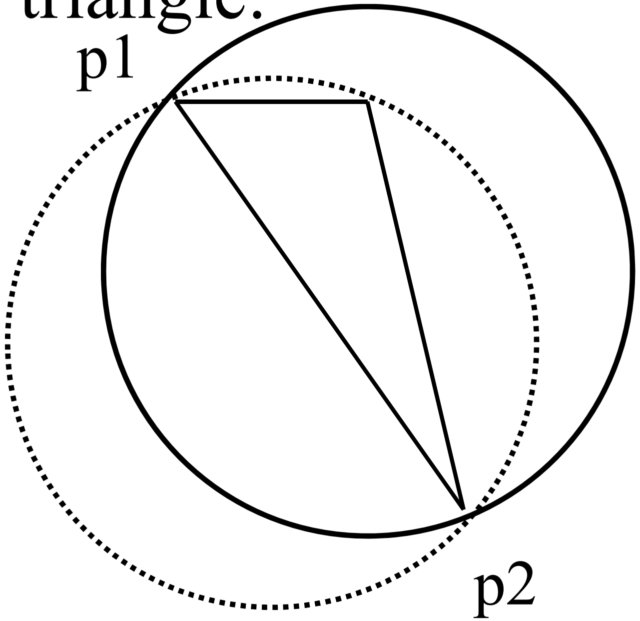
Optimal or not optimal

- an acute triangle:



The circle is optimal.

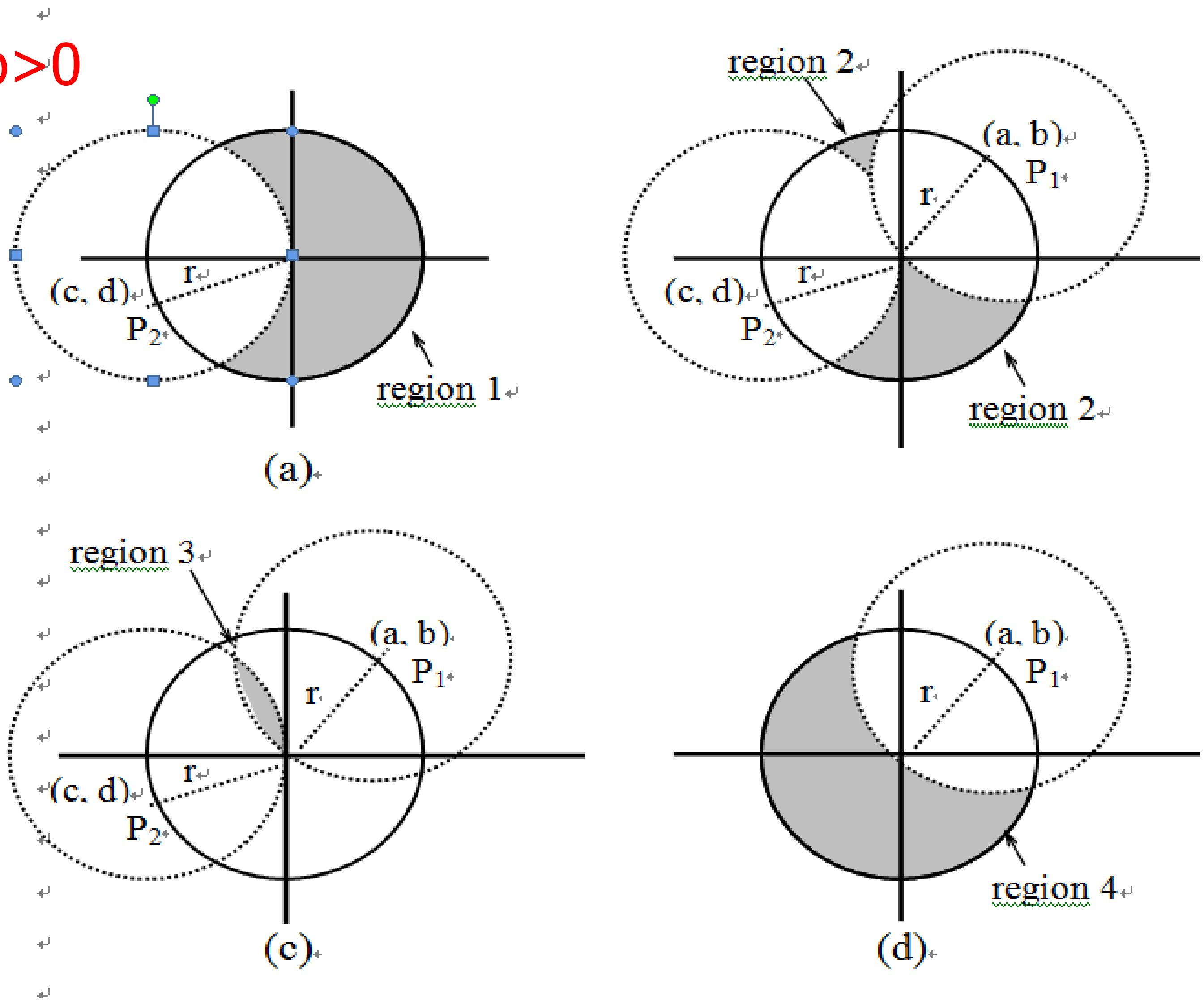
- an obtuse(鈍角) triangle:



The circle is not optimal.

The x-value of end points p1 and p2 must be of opposite signs

Assume a> x*, b>0
And c<x*, d<0

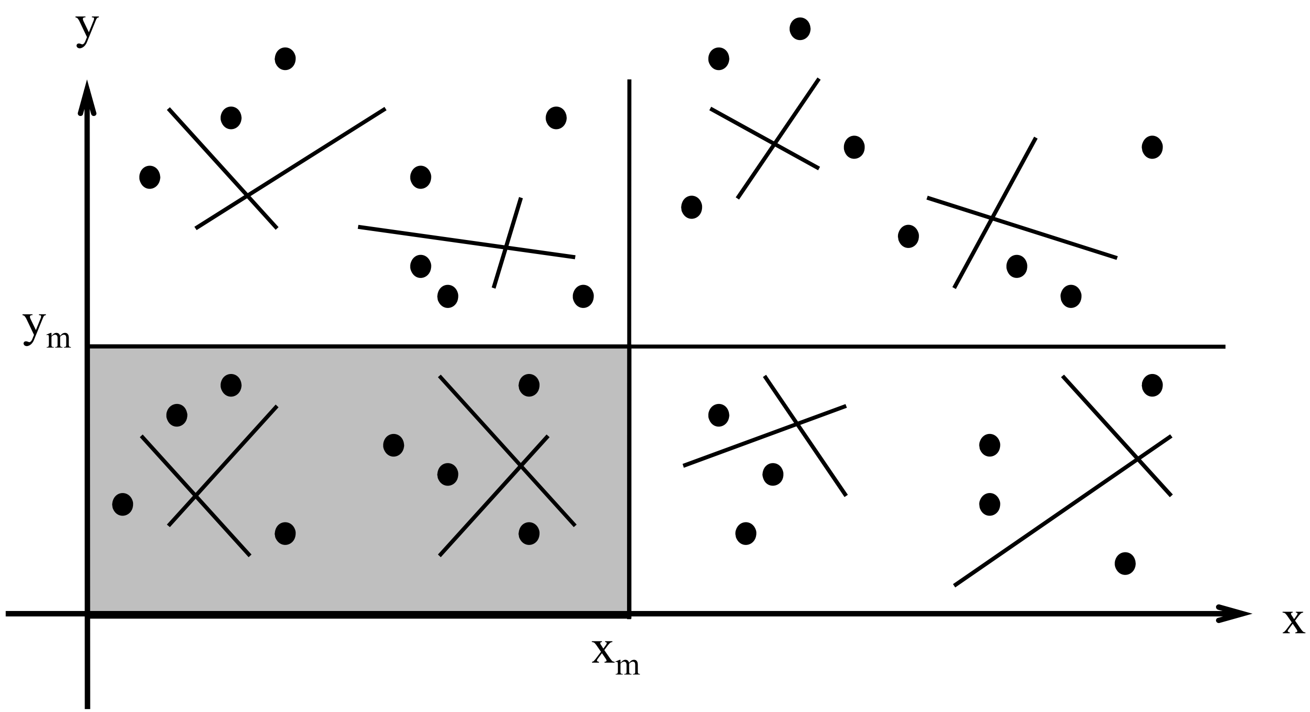


The optimum center must be located in region 3.

Thus, the sign of y_3 must be the sign of $\frac{b+d}{2} = \frac{y_1+y_2}{2}$.

Similarly, x_s has the same sign as that of $\frac{a+c}{2} = \frac{x_1+x_2}{2}$.

An example of 1-center problem



- One point for each of n/4 intersections of L_{i+} and L_{i-} is pruned away.
- Thus, n/16 points are pruned away in each iteration.

Prune-and-search approach

- Input: A set $S = \{p_1, p_2, \ldots, p_n\}$ of n points.
- Output: The smallest enclosing circle for S.

Step 1: If S contains no more than 16 points, solve the problem by a brute-force method.

Step 2: Form disjoint pairs of points, (p_1, p_2), (p_3, p_4), …,(p_{n-1}, p_n). For each pair of points, (p_i, p_{i+1}), find the perpendicular bisector of line segment $\overline{p_i p_{i+1}}$. Denote them as $L_{i/2}$, for $i = 2, 4, \ldots, n$, and compute their slopes. Let the slope of L_k be denoted as s_k, for $k = 1, 2, 3, \ldots, n/2$.

Step 3: Compute the median of s_k's, and denote it by s_m.

Step 4: Rotate the coordinate system so that the x-axis coincide with $y = s_m x$. Let the set of L_k's with positive (negative) slopes be I^+ (I^-). (Both of them are of size n/4.)

Step 5: Construct disjoint pairs of lines, (L_{i+}, L_{i-}) for $i = 1, 2, \ldots, n/4$, where $L_{i+} \in I^+$ and $L_{i-} \in I^-$. Find the intersection of each pair and denote it by (a_i, b_i), for $i = 1, 2, \ldots, n/4$.

Step 6: Find the median of b_i's. Denote it as y*. Apply the constrained 1-center subroutine to S, requiring that the center of circle be located on y=y*. Let the solution of this constrained 1-center problem be (x', y*).

Step 7: Determine whether (x', y*) is the optimal solution. If it is, exit; otherwise, record $y_s > y^*$ or $y_s < y^*$.

- Step 8: If $y_s > y^*$, find the median of a_i's for those (a_i, b_i)'s where $b_i < y^*$. If $y_s < y^*$, find the median of a_i's of those t hose (a_i, b_i)'s where $b_i > y^*$. Denote the median as x^*. Apply the constrained 1-center algorithm to S, requiring that the center of circle be located on $x = x^*$. Let the solution of this contained 1-center problem be (x^*, y').
- Step 9: Determine whether (x^*, y') is the optimal solution. If it is, exit; otherwise, record $x_s > x^*$ and $x_s < x^*$.

Step 10:

- Case 1: $x_s < x^*$ and $y_s < y^*$.

  Find all (a_i, b_i)'s such that $a_i > x^*$ and $b_i > y^*$. Let (a_i, b_i) be the intersection of L_{i+} and L_{i-}. Let L_{i-} be the bisector of p_j and p_k. Prune away $p_j(p_k)$ if $p_j(p_k)$ is closer to (x^*, y^*) than $p_k(p_j)$.
- Case 2: $x_s > x^*$ and $y_s > y^*$. Do similarly.
- Case 3: $x_s < x^*$ and $y_s > y^*$. Do similarly.
- Case 4: $x_s > x^*$ and $y_s < y^*$. Do similarly.

Step 11: Let S be the set of the remaining points. Go to Step 1.

- Time complexity :

$$T(n) = T(15n/16)+O(n)$$
$$= O(n)$$