

# Prune-and-Search

The prune-and-search method consists of several iterations, and in each iteration, a fraction of the solution space is pruned.

The pruning will continue until the remaining solution space is small enough, in which the desired solution can be found quickly.

In general, the time complexity can be expressed as

$$T(n) \leq T((1 - \alpha)n) + p(n),$$

where  $0 < \alpha < 1$  is the fraction to prune the solution space in each iteration, and  $p(n)$  is the time needed for each iteration.

**Ex. The binary search**

$$\alpha = 1/2, \quad p(n) = O(1)$$

**Ex. The selection problem**

**Given  $n$  distinct numbers  $a_1, a_2, \dots, a_n$ , determine the  $k$ th smallest one.**

**Approach 1. Choose a number  $p$  arbitrarily from these  $n$  numbers and partition them into three subsets:**

$$S_1 = \{x \mid x < p\}, \quad S_2 = \{p\}, \quad S_3 = \{x \mid x > p\}.$$

**The  $k$ th smallest number is located in one of these three subsets, and the partitioning is repeated for the designated subset until the  $k$ th smallest number is found.**

**This approach cannot guarantee that a fraction of the solution space is always discarded after each iteration.**

**In fact,  $O(n^2)$  time is needed in the worst case.**

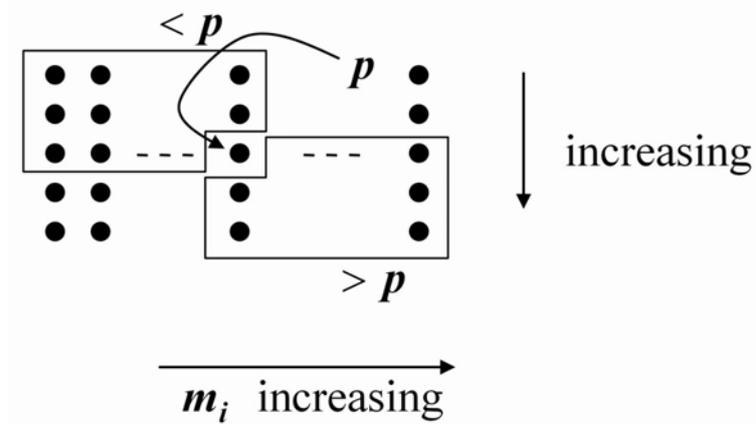
**Approach 2.** First, divide the  $n$  numbers into  $n/r$  groups each of  $r$  numbers, where  $r > 1$  is odd.

If  $n$  is not a multiple of  $r$ , sufficient numbers are added artificially.

Then, sort every group and let  $m_i$  be the median of group  $i$ , where  $1 \leq i \leq \lceil n/r \rceil$ .

Choose the median of  $m_i$ 's as  $p$  and perform the partitioning repeatedly until the solution space is small enough.

**Ex. Assume  $r = 5$ .**



**At least one fourth of the solution space is discarded after each iteration.**

**Thus,**

$$T(n) \leq T(n/5) + T(3n/4) + O(n),$$

**from which  $T(n) = O(n)$  can be derived (refer to *Computer Algorithms*, by Horowitz, Sahni, and Rajasekaran)**

**Ex. The 2-dimensional linear programming problem.**

$$\begin{aligned} &\text{minimize} && a'x' + b'y' \\ &\text{subject to} && a_i'x' + b_i'y' + c_i' \leq 0 \quad i = 1, 2, \dots, n. \end{aligned}$$

**To simplify the problem, we transform the original form into the following equivalent form by letting  $y = a'x' + b'y'$  and  $x = x'$  (座標旋轉).**

$$\begin{aligned} &\text{minimize} && y \\ &\text{subject to} && a_i x + b_i y + c_i \leq 0 \quad i = 1, 2, \dots, n. \end{aligned}$$

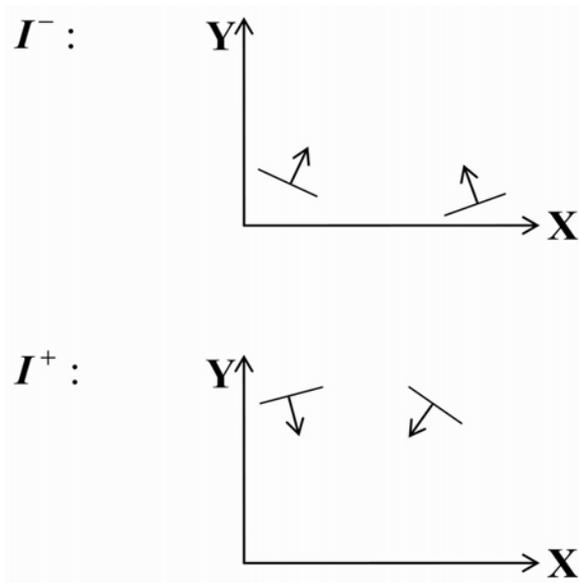
**The transformation takes  $O(n)$  time.**

**Thus, we only have to compute the smallest  $y$ -coordinate in the feasible region.**

The  $n$  constraints may be partitioned into three classes  $I^0, I^-, I^+$ , depending on whether  $b_i$  is zero, negative, or positive.

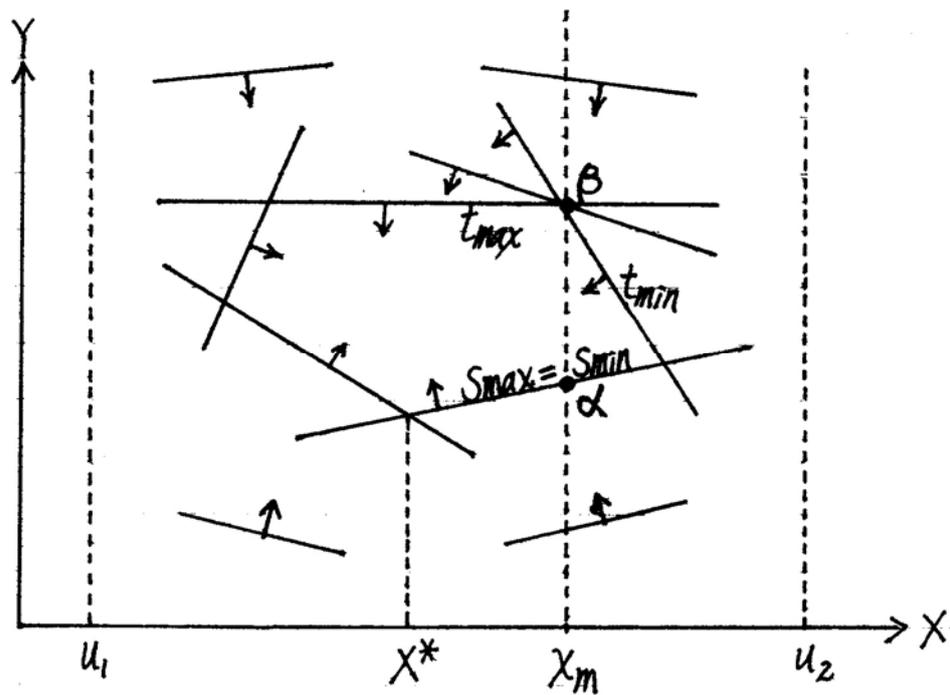
$I^0$ : may be combined into a single constraint

$$u_1 \leq x \leq u_2$$



**Observation (1) :** The feasible region, if not empty, is a convex polygon with possibly one open side.

**Observation (2) :**

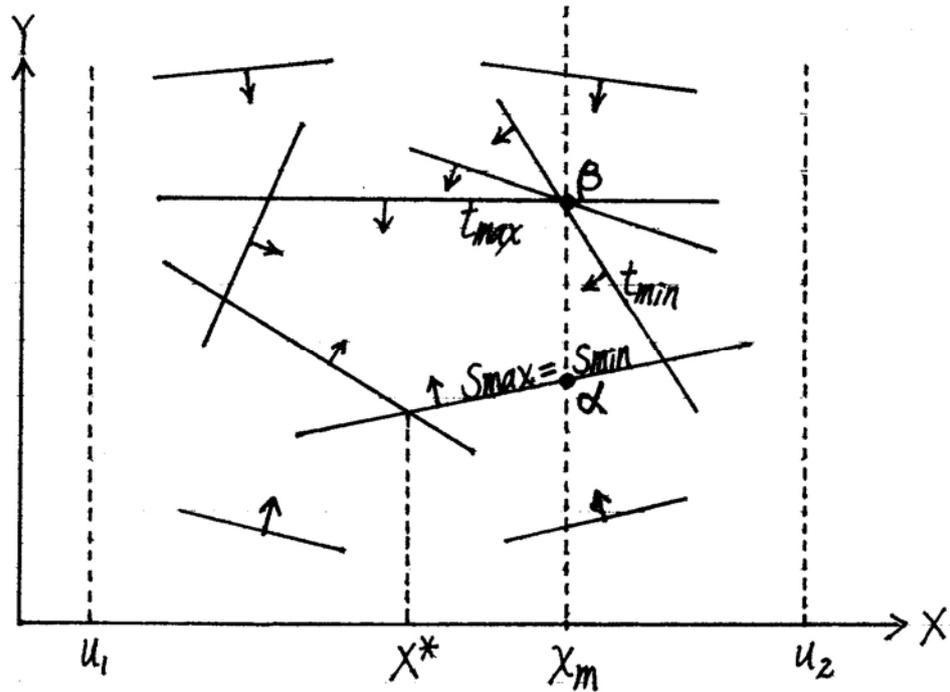


$x_m$  : any  $x$ -value in  $[u_1, u_2]$ ;

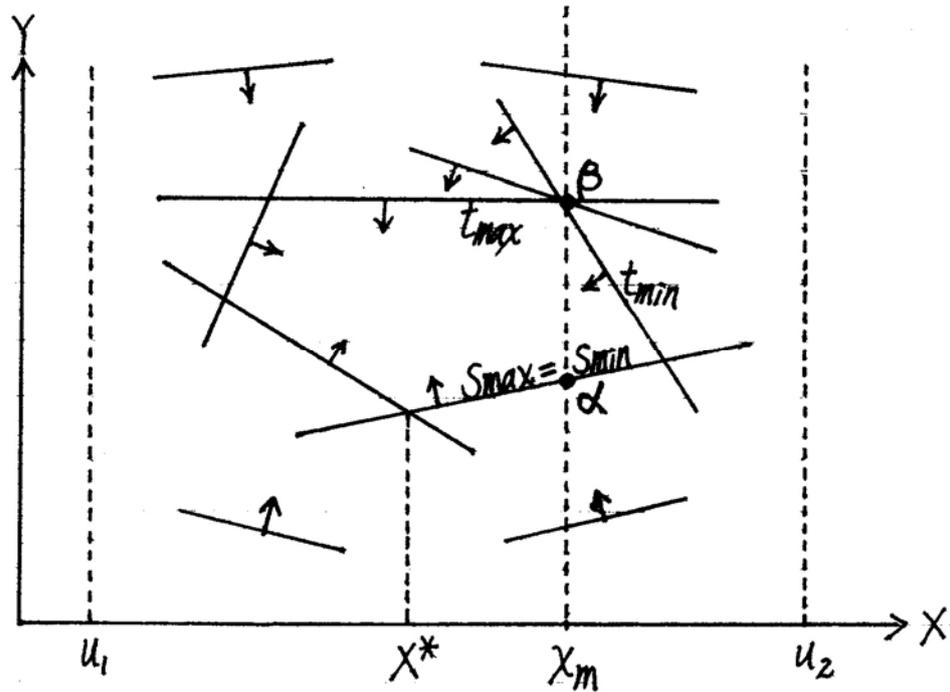
$x^*$  : the optimal  $x$ -value;

$\alpha$  : the boundary point of the feasible region that is represented by  $I^-$ , where  $x = x_m$  passes;

$\beta$  : the boundary point of the feasible region that is represented by  $I^+$ , where  $x = x_m$  passes;



- $S_{\max}$  ( $S_{\min}$ ) : the *maximum (minimum)* slope of the bounding lines (in  $I^-$ ) that pass through  $\alpha$ ;
- $t_{\max}$  ( $t_{\min}$ ) : the *maximum (minimum)* slope of the bounding lines (in  $I^+$ ) that pass through  $\beta$ .



$\alpha = (\alpha_x, \alpha_y)$  and  $\beta = (\beta_x, \beta_y)$  can be determined in  $O(n)$  time as follows:

$$\alpha_x \leftarrow x_m; \quad \beta_x \leftarrow x_m;$$

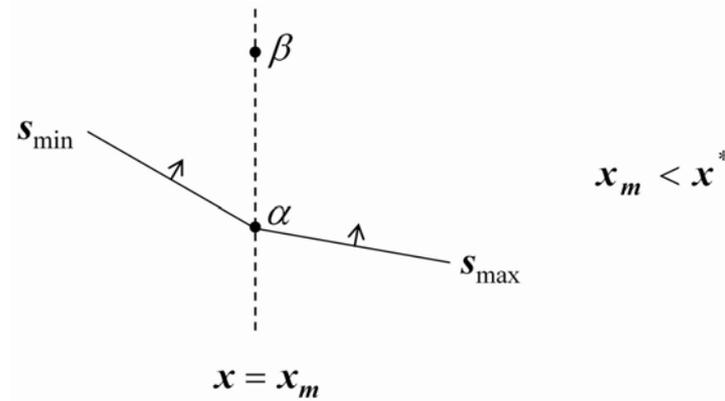
$$\alpha_y \leftarrow \max\{(-a_i x_m - c_i)/b_i \mid a_i x + b_i y + c_i \leq 0 \in I^-\};$$

$$\beta_y \leftarrow \min\{(-a_i x_m - c_i)/b_i \mid a_i x + b_i y + c_i \leq 0 \in I^+\}.$$

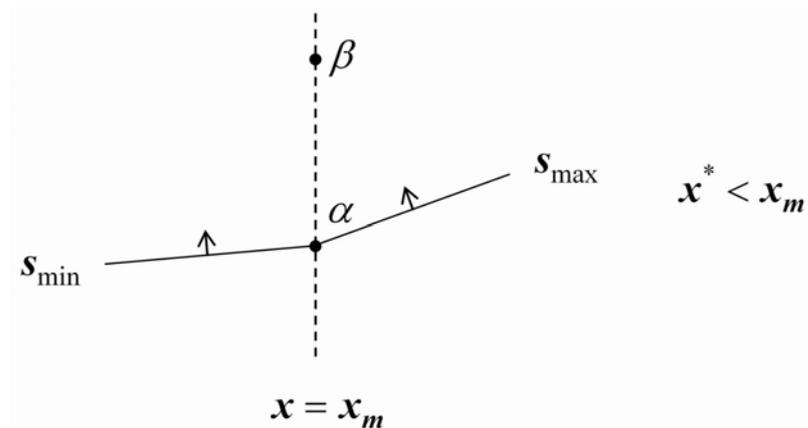
Also,  $s_{\max}$ ,  $s_{\min}$ ,  $t_{\max}$  and  $t_{\min}$  can be determined as well.

Depending on the values of  $(\alpha_x, \alpha_y)$ ,  $(\beta_x, \beta_y)$ ,  $s_{\max}$ ,  $s_{\min}$ ,  $t_{\max}$  and  $t_{\min}$ , the following six cases are discussed.

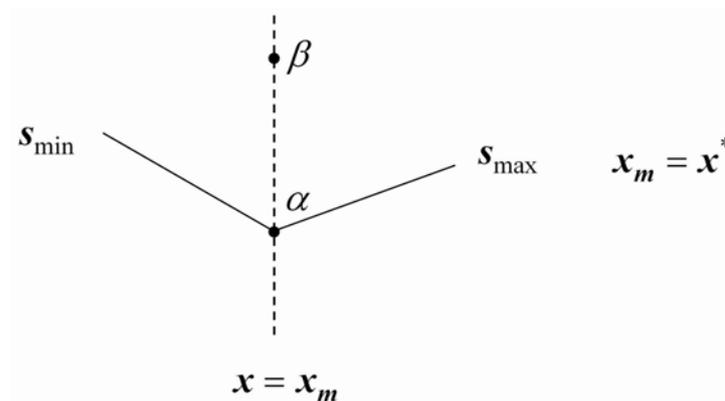
**Case 1.**  $\alpha_y \leq \beta_y$  and  $(s_{\min} \leq) s_{\max} < 0$ .



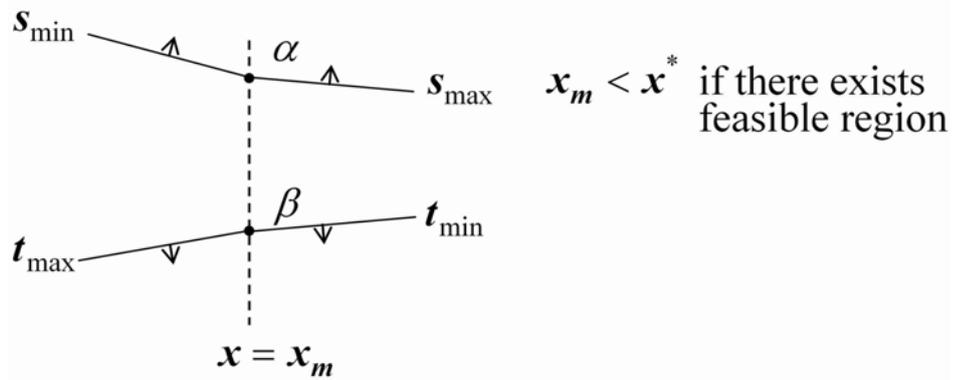
**Case 2.**  $\alpha_y \leq \beta_y$  and  $(s_{\max} \geq) s_{\min} > 0$ .



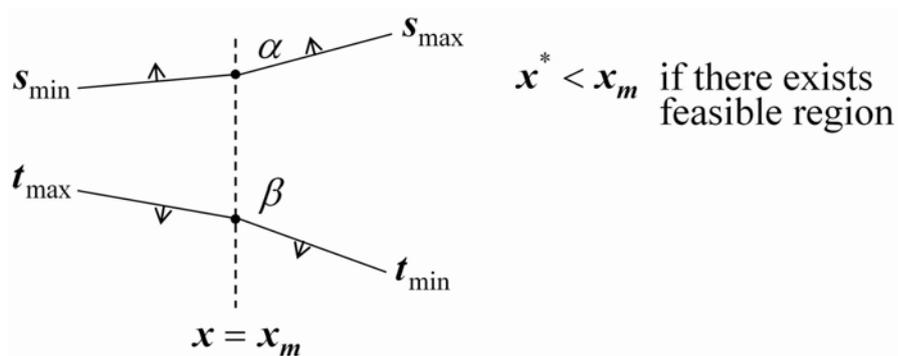
**Case 3.**  $\alpha_y \leq \beta_y$  and  $s_{\min} \leq 0 \leq s_{\max}$ .



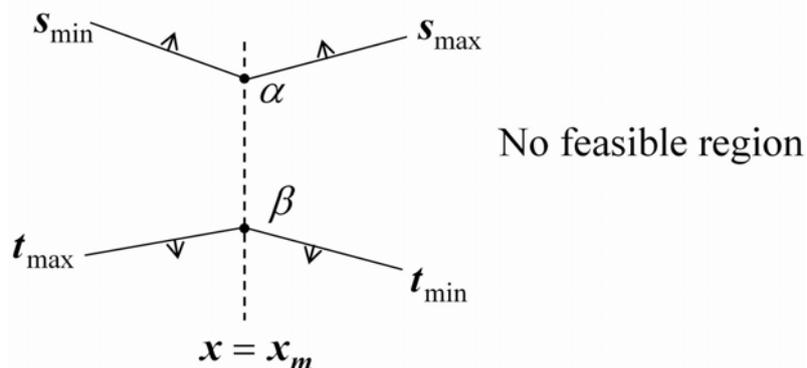
**Case 4.**  $\alpha_y > \beta_y$  and  $s_{\max} < t_{\min}$ .



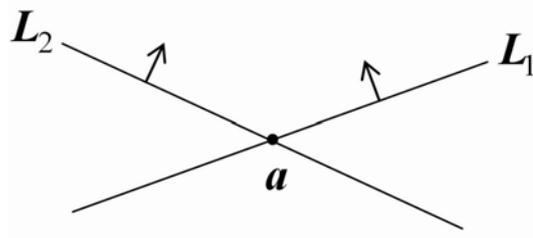
**Case 5.**  $\alpha_y > \beta_y$  and  $s_{\min} > t_{\max}$ .



**Case 6.**  $\alpha_y > \beta_y$  and  $(s_{\max} \geq t_{\min}$  and  $s_{\min} \leq t_{\max})$ .

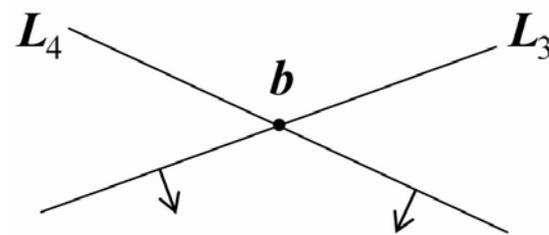


**Observation (3) :**



**If  $x^* < a_x$ , then  $L_1$  can be removed.**

**If  $a_x < x^*$ , then  $L_2$  can be removed.**



**If  $x^* < b_x$ , then  $L_4$  can be removed.**

**If  $b_x < x^*$ , then  $L_3$  can be removed.**

## **A prune-and-search algorithm.**

- 1.  $x_l \leftarrow u_1, x_r \leftarrow u_2$ .**
- 2. Partition  $I^-$  and  $I^+$  each into pairs of constraints (bounding lines) and determine the  $x$ -coordinates  $r_x$ 's of their intersections.**

**If the pair of bounding lines is in parallel or  $r_x \notin [x_l, x_r]$ , then one can be removed.**

- 3. For the set of  $r_x$ 's  $\in [x_l, x_r]$ , find their median.**
- 4. Let  $x_m$  be the median. If  $x_m = x^*$  or no feasible region, then terminate the algorithm. If  $x^* < x_m$ , then  $x_r \leftarrow x_m$ . If  $x_m < x^*$ , then  $x_l \leftarrow x_m$ .**

**5. Prune redundant constraints.**

**Since half of  $r_x$ 's are known to be smaller ( $x_m < x^*$ ) or greater ( $x^* < x_m$ ) than  $x^*$ , about one fourth of the constraints can be removed.**

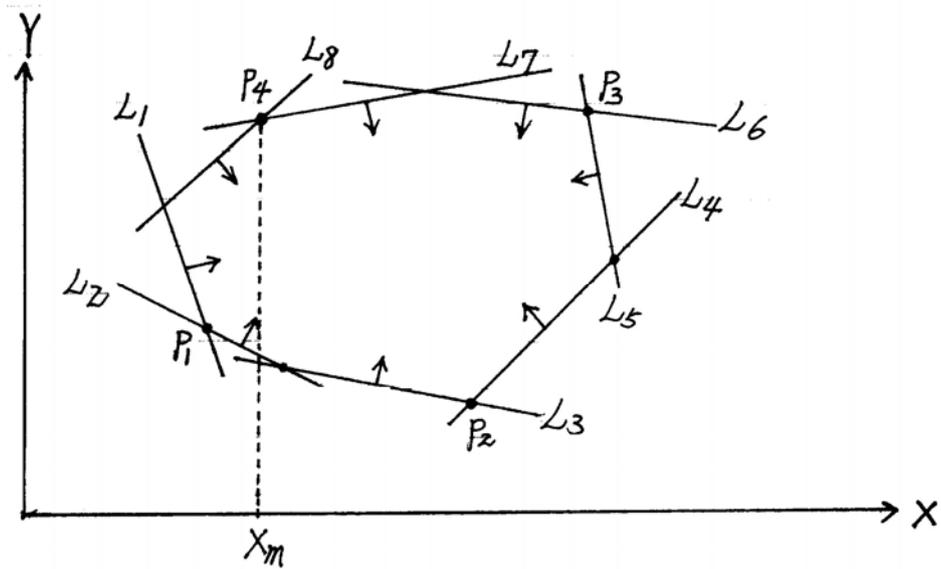
**6. Repeat 2 ~ 5 until the number of remaining constraints is small enough.**

**7. Solve the reduced problem directly.**

**Time complexity :**

$$\begin{aligned} T(n) &\leq T(3n/4) + O(n) \\ &\leq O(n). \end{aligned}$$

**Ex.**



1.  $x_l \leftarrow -\infty, x_r \leftarrow \infty$ .
2. Form pairs :  $(L_1, L_2), (L_3, L_4), (L_5, L_6)$  and  $(L_7, L_8)$ .
3.  $x_m \leftarrow x$ -coordinate of the intersection of  $L_7$  and  $L_8$ .

$$x_m < x^*, x_l \leftarrow x_m$$

4.  $L_8$  and  $L_1$  are pruned.

**Ex. The smallest circle enclosing  $n$  points.**

**(The Euclidean 1-center problem in the plane)**

**Given  $n$  points  $p_1, p_2, \dots, p_n$  in the Euclidean plane, find the smallest circle enclosing these  $n$  points.**

**Let  $(a_i, b_i)$  be the coordinate of  $p_i$ ,  $1 \leq i \leq n$ .**

**The problem is to determine a point so as to minimize**

$$f(x, y) = \max\{((x - a_i)^2 + (y - b_i)^2)^{1/2} \mid 1 \leq i \leq n\}.$$

**To begin with, we consider a restricted case (1-d 1-center problem) where the center of the circle lies on the line  $y = y_m$ .**

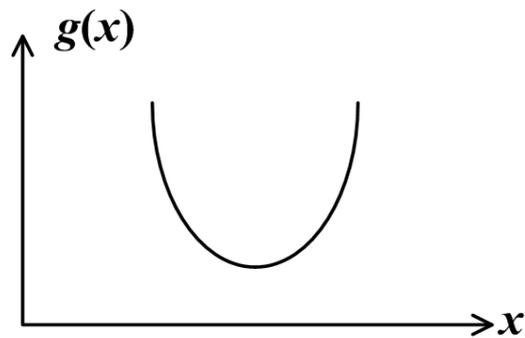
**The restricted problem is to minimize**

$$g(x) = \max \{((x - a_i)^2 + (y_m - b_i)^2)^{1/2} \mid 1 \leq i \leq n\}.$$

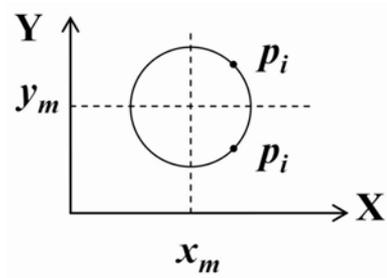
**Observation (1) :**

**Let  $x^*$  be the  $x$ -value that minimizes  $g(x)$ ,  $x_m$  be any  $x$ -value, and define  $I$  to be the set of points that are farthest from  $(x_m, y_m)$ .**

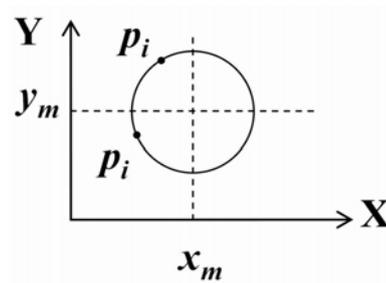
**We can recognize whether  $x_m < x^*$ ,  $x_m = x^*$ , or  $x_m > x^*$  as follows ( $g(x)$  is concave).**



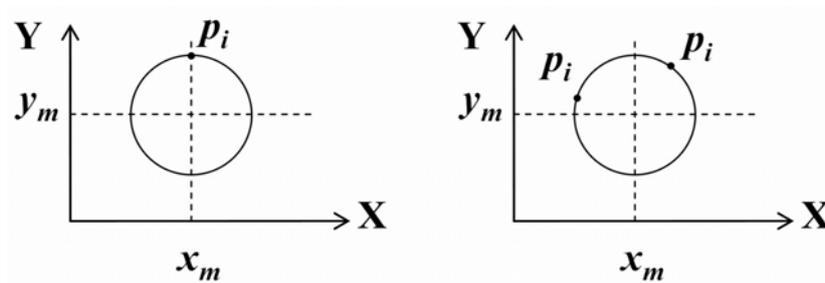
**Case 1.**  $x_m < x^*$ , if  $x_m < a_i$  for every  $p_i \in I$ .



**Case 2.**  $x^* < x_m$ , if  $a_i < x_m$  for every  $p_i \in I$ .



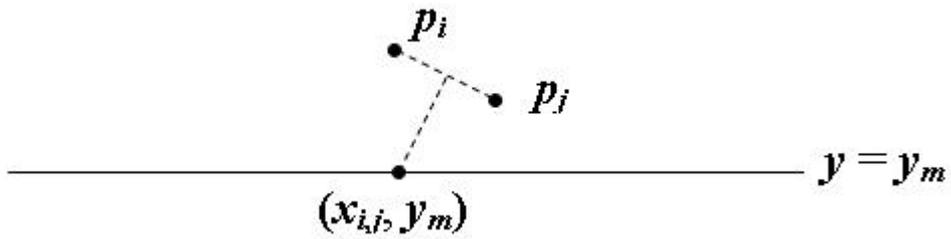
**Case 3.**  $x_m = x^*$ , otherwise.



**Observation (2) :**

**Let  $(x_{ij}, y_m)$  have equal distance from  $p_i$  and  $p_j$ .**

**Then,  $p_i$  ( $p_j$ ) can be discarded if  $x^* < x_{ij}$  ( $x_{ij} < x^*$ ).**



## **A prune-and-search algorithm.**

- 1. Partition points into pairs  $(p_1, p_2), (p_3, p_4), \dots$**
- 2. For each pair  $(p_i, p_{i+1})$ , if  $a_i = a_{i+1}$ , then discard the point that is closer to the line  $y = y_m$ .  
  
Otherwise, find the point  $(x_{i,i+1}, y_m)$  that has equal distance from  $p_i$  and  $p_j$ .**
- 3. Let  $x_m$  be the median of those  $x_{i,i+1}$ 's.**
- 4. Compute the set  $I$  of the points that are farthest from  $(x_m, y_m)$ , and then determine whether  $x_m < x^*$ ,  $x_m = x^*$ , or  $x_m > x^*$ .**

**5. Prune redundant points.**

**Since at least half of  $x_{i,j+1}$ 's are smaller or greater (depending on  $x_m < x^*$  or  $x^* < x_m$ ) than  $x^*$ , at least one quarter of the points can be removed.**

**6. Repeat 1 ~ 5 until the number of the remaining points is small enough.**

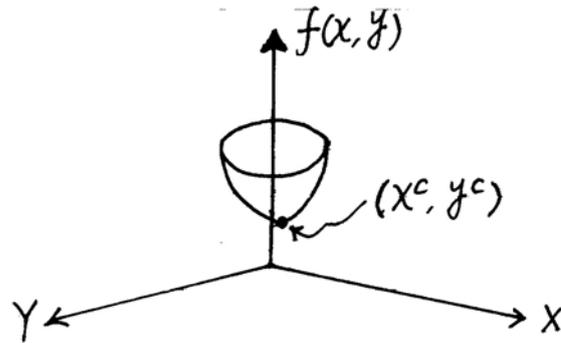
**7. Solve the reduced problem directly.**

**Time complexity :**

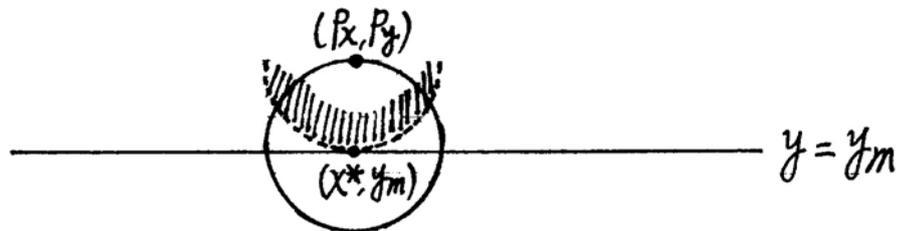
$$\begin{aligned} T(n) &\leq T(3n/4) + O(n) \\ &\leq O(n). \end{aligned}$$

**By slight modifications, the algorithm can find the optimal center that lies on any line  $y = ax + b$ .**

Let  $(x^c, y^c)$  be the optimal solution to the original problem. Since  $f(x, y)$  is concave, we can determine where to find  $(x^c, y^c)$  by checking the set  $I$ .



**Case 1.  $I$  contains one point.**



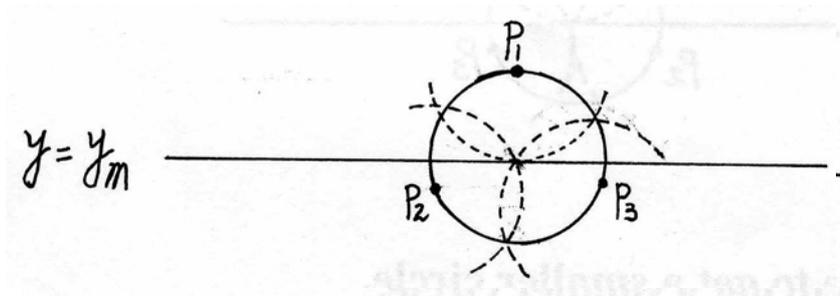
**$p_x = x^*$  (otherwise, moving the circle right or left can result in a smaller circle)**

**Moving the center to the shaded area can result in a smaller circle, i.e.,  $y^c > y_m$ .**

**Case 2.  $I$  contains more than one point.**

**$O(n)$  time is sufficient to determine if there exists an arc  $< 180^\circ$  covering all of the points in  $I$ .**

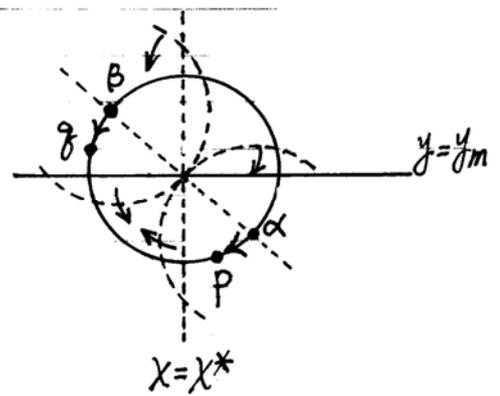
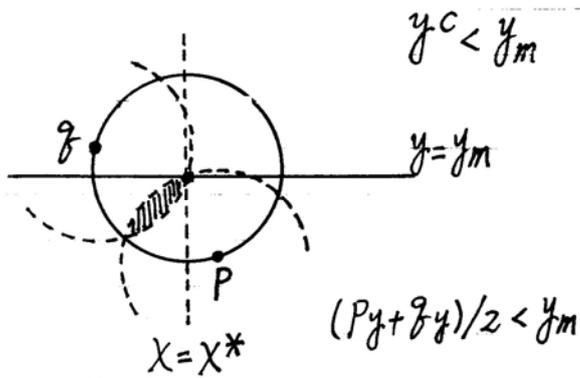
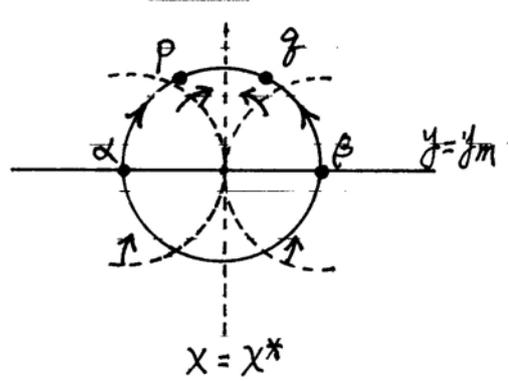
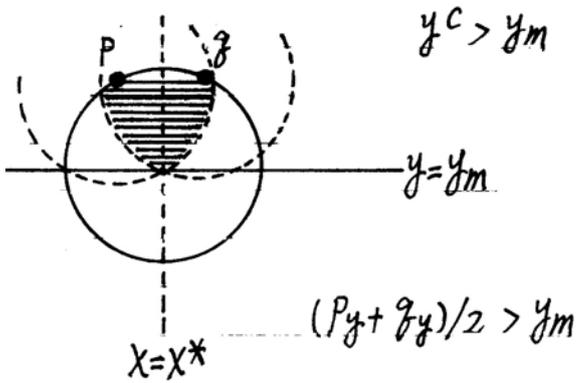
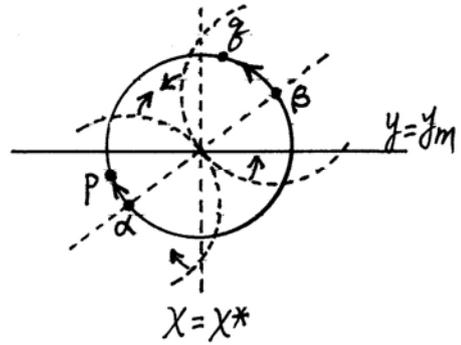
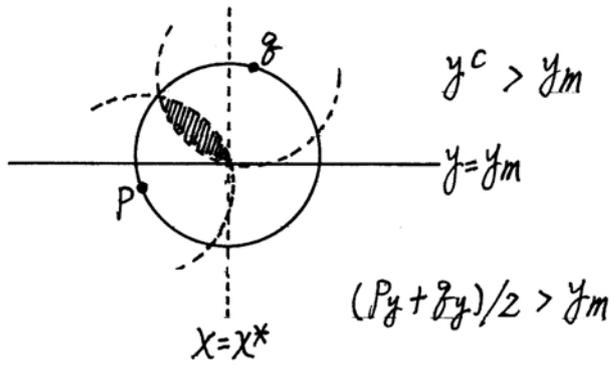
**(1) No such arc can be found.**



**It is impossible to get a smaller circle,  
i.e.,  $(x^c, y^c) = (x^*, y_m)$ .**

**(2) Such an arc can be found.**

**Suppose that  $p = (p_x, p_y)$  and  $q = (q_x, q_y)$  are the two end points of the arc, where  $p$  and  $q$  are in the opposite sides of  $x = x^*$  (otherwise, moving the circle left or right can result in a smaller circle).**



**Moving the center to the shaded area will result in a smaller circle.**

**The shaded area is the intersection of the two circles centered at  $p$  and  $q$  with the same radius as the enclosing circle.**

**(1)  $(p_y + q_y) / 2 > y_m$ .**

**The shaded area is above  $y = y_m$ , i.e.,  $y^c > y_m$ .**

**(2)  $(p_y + q_y) / 2 < y_m$ .**

**The shaded area is below  $y = y_m$ , i.e.,  $y^c < y_m$ .**

**In brief, we can determine on which side of a given line  $(x^c, y^c)$  is located in  $O(n)$  time.**

**A prune-and-search algorithm for the smallest circle problem.**

- 1. Partition the points into pairs  $(p_1, p_2), (p_3, p_4), \dots$**
- 2. For each pair  $(p_i, p_{i+1})$ , find the perpendicular bisector  $L_i$  of the line segment  $p_i p_{i+1}$ .**
- 3. Compute the angles  $\alpha_i$  ( $-\pi/2 \leq \alpha_i < \pi/2$ ) which  $L_i$  forms with the positive direction of the  $x$ -axis.**
- 4. Compute the median  $\alpha_m$  of  $\alpha_i$ 's.**

5. Partition the set of  $L_i$ 's into two equal-size (about  $n/4$ ) subsets  $L^+$  and  $L^-$  so that each  $L_i$  in  $L^+$  has  $\alpha_i \geq \alpha_m$  and each  $L_j$  in  $L^-$  has  $\alpha_j < \alpha_m$ .
6. Construct disjoint pairs of  $(l_i^+, l_i^-)$ ,  $i = 1, 2, \dots, n/4$ , where  $l_i^+ \in L^+$  and  $l_i^- \in L^-$ .
7. For each pair  $(l_i^+, l_i^-)$ , compute their intersection  $(l_{x_i}, l_{y_i})$ .
8. Find a line (assume  $y = ax + b$ ) parallel to  $L_m$  (with angle  $\alpha_m$ ), which partitions the set of  $(l_{x_i}, l_{y_i})$ 's into two equal-size (about  $n/8$ ) subsets.

**9. Determine on which side of  $y = ax + b$   $(x^c, y^c)$  is located.**

**Without losing generality, assume that  $(x^c, y^c)$  is below  $y = ax + b$ .**

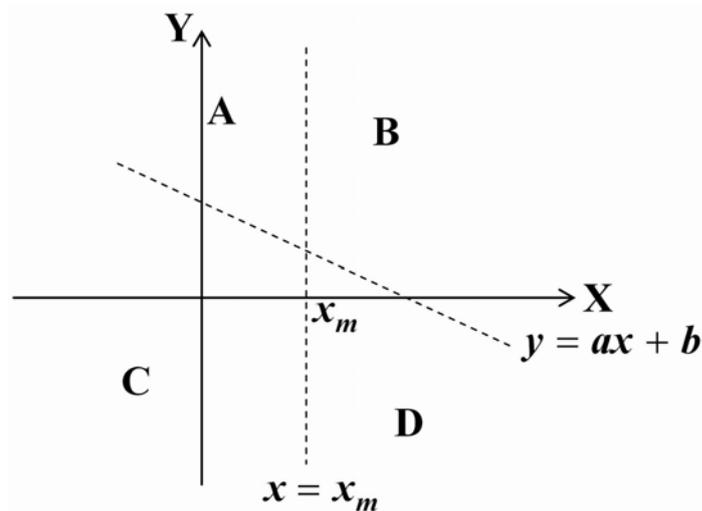
**10. Find a vertical line (assume  $x = x_m$ ), which partitions the set of those  $(l_{x_i}, l_{y_i})$ 's that are above  $y = ax + b$  into two equal-size (about  $n/16$ ) subsets.**

**11. Determine on which side of  $x = x_m$   $(x^c, y^c)$  is located.**

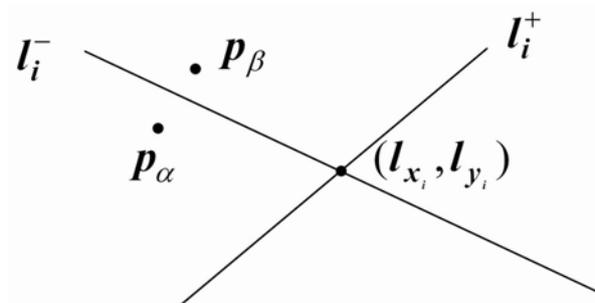
**Without losing generality, assume  $x^c < x_m$ .**

## 12. Prune redundant points.

About  $n/16$  points can be discarded.



- (1)  $(x^c, y^c) \in C$ .
- (2) There are about  $n/16$   $(l_{x_i}, l_{y_i})$ 's in  $B$ .
- (3) For each  $(l_{x_i}, l_{y_i})$  in  $B$ , a point  $(p_\alpha)$  can be discarded because it is closer to  $(x^c, y^c)$  than the other point  $(p_\beta)$ .



$l_i^-$  does not intersect with  $C$

- 13. Repeat 1 ~ 12 until the number of remaining point is small enough, for which the optimal center  $(x^c, y^c)$  can be determined in constant time.**

**Time complexity :**

$$\begin{aligned} T(n) &\leq T(15n/16) + O(n) \\ &\leq O(n) \end{aligned}$$

## **Program Assignment 4 :**

**Write an executable program to solve the  
2-dimensional linear programming problem.**