§Prune-and-Search

- **The selection problem**
  input: A set S of n elements
  output: The kth smallest element of S
- the <u>median</u> 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.

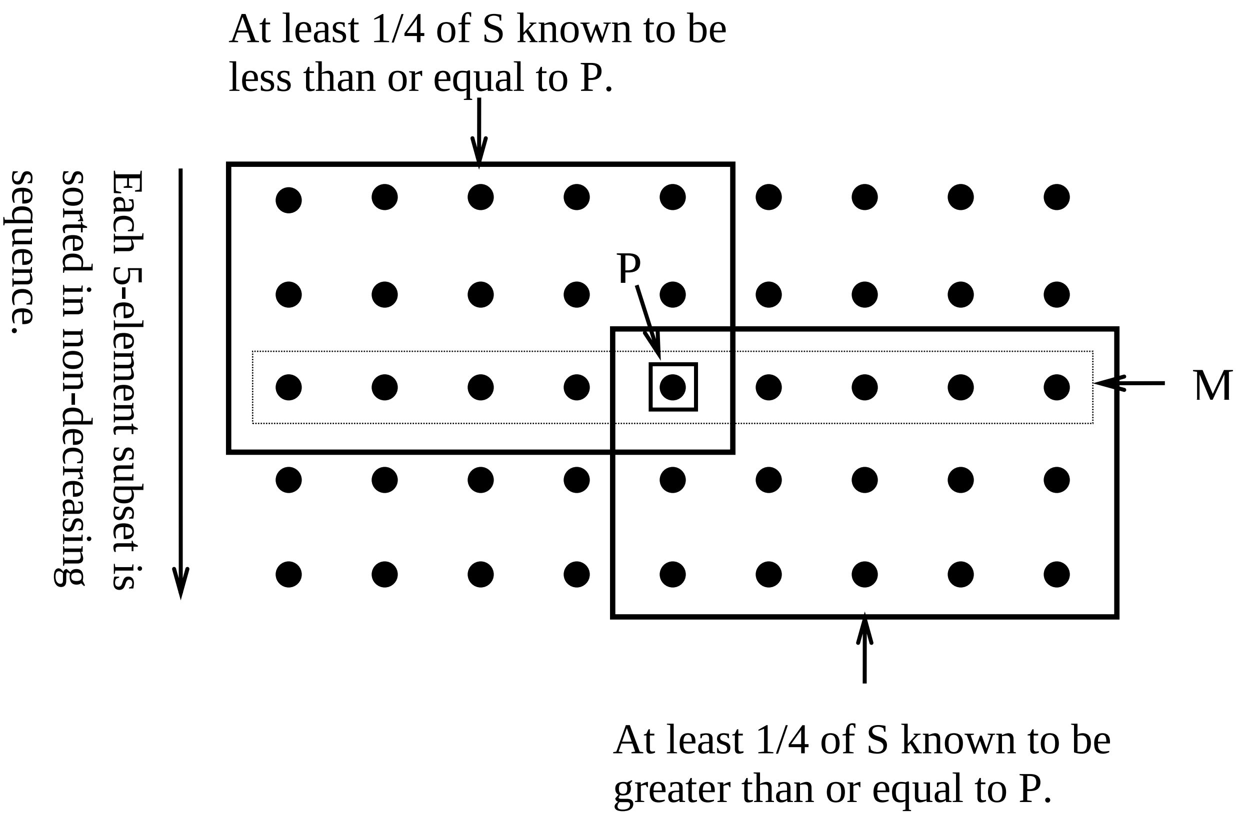
  time complexity: O(nlogn)
- prune-and-search
  $S=\{a_1, a_2, \ldots, a_n\}$
  Let $p \in S$, use p to partition S into 3 subsets S_1, S_2, S_3:
  $S_1=\{ a_i \mid a_i < p , 1 \leq i \leq n\}$
  $S_2=\{ a_i \mid a_i = p , 1 \leq i \leq n\}$
  $S_2=\{ a_i \mid a_i > p , 1 \leq i \leq n\}$

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.)



Algorithm 7.1 A Prune-and-Search Algorithm to Find the Kth Smallest Element

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.
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.

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

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

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

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

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

$$
\begin{aligned}
\Rightarrow T(n) &\leq cn + T(19n/20) \\
&\leq cn + (19/20)cn + T((19/20)^2 n) \\
&\quad \vdots \\
&\leq cn + (19/20)cn + (19/20)^2 cn + \dots \\
&\quad + (19/20)^p cn + T((19/20)^{p+1} n) \quad , \\
&\quad (19/20)^{p+1} n \leq 1 \leq (19/20)^p n \\
&= \frac{1-(\frac{19}{20})^{p+1}}{1-\frac{19}{20}} cn + b \\
&\leq 20\, cn + b \\
&= O(n)
\end{aligned}
$$

general form:

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

Let $1/(1-f) = a$, $a^p = n$, $p = \log_a n$

$$\begin{aligned}
T(n) &= T((1-f)n) + cn^k \\
&= T((1-f)^2 n) + c(1-f)^k n^k + cn^k \\
&\vdots \\
&= c + cn^k + c(1-f)^k n^k + c(1-f)^{2k} n^k + \dots + c(1-f)^{pk} n^k \\
&= c + cn^k (1 + (1-f)^k + (1-f)^{2k} + \dots + (1-f)^p) \\
&\leq c + cn^k/(1-(1-f)) \\
&= c + cn^k/f \\
&= O(n^k)
\end{aligned}$$

- **Linear programming with two variables**

$$\begin{cases} \text{minimize } ax + by \\ \text{subject to } a_i x + b_i y \geq c_i \quad , i = 1, 2, \ldots, n \end{cases}$$

- Simplified two-variable linear programming problem:

$$\begin{cases} \text{minimize } y \\ \text{subject to } y \geq a_i x + b_i \quad , i = 1, 2, \ldots, n \end{cases}$$

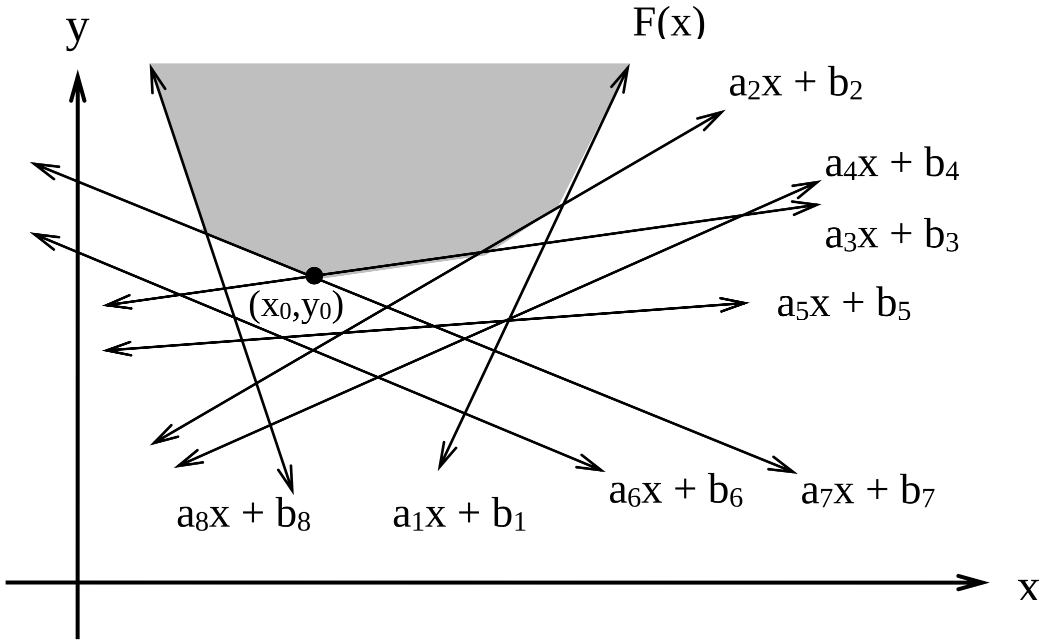


Fig. 7-2 An Example of the Special Two-Variable Linear Programming Problem

the boundary F(x):

$$F(x) = \max_{-\infty < x < \infty} \{a_i x + b_i\}$$

the optimum solution x_0:

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

Delete constraints:

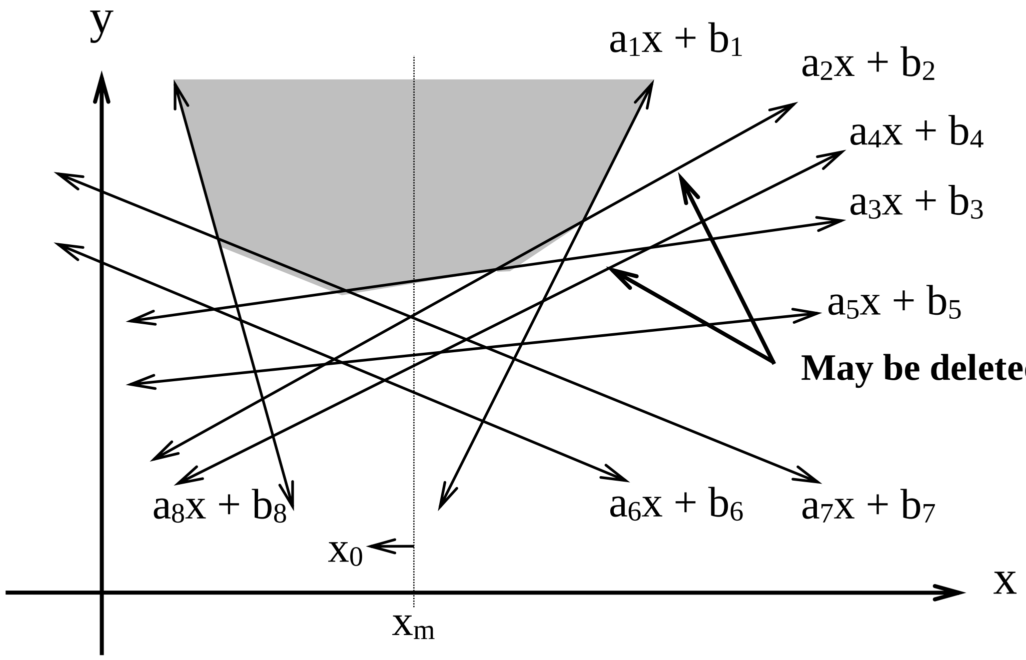


Fig. 7-3 Constraints which May be Eliminate in the Two-Variable Linear Programming Problem

If $x_0 < x_m$ and the intersection of $a_1x + b_1$ 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 $x_0 > x_m$.

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

Let $y_m = F(x_m) = \max\limits_{1<i<n}\{a_i x_m + b_i\}$

Case 1: y_m is on only one constraint.

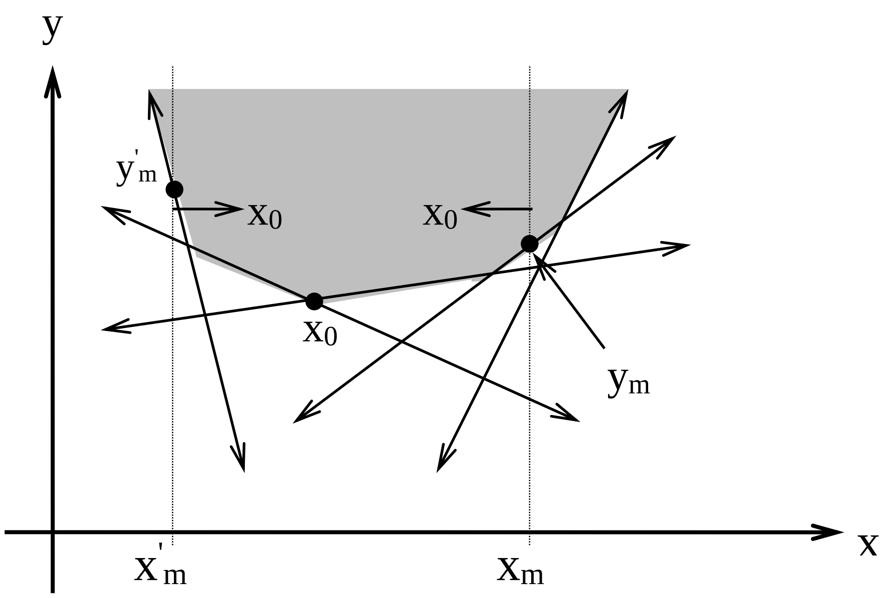


Fig.7-5 The Cases where x_m Is on Only One Constrain

Let g denote the slope of this constraint.

If g > 0, then $x_0 < x_m$.

If g < 0, then $x_0 > x_m$.

Case 2: y_m is the intersection of several constraints.

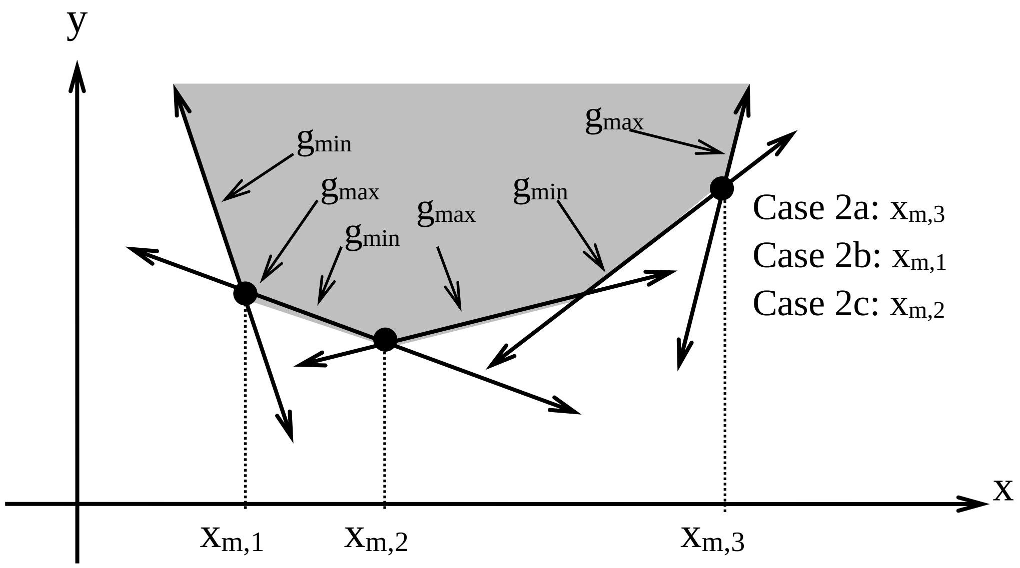


Fig. 7-6 Cases of x_m on the Intersection of Several Constraints

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

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

Case 2a: $g_{min} > 0$, $g_{max} > 0 \Rightarrow x_0 < x_m$

Case 2b: $g_{min} < 0$, $g_{max} < 0 \Rightarrow x_0 > x_m$

Case 2c: $g_{min} < 0$, $g_{max} > 0 \Rightarrow (x_m, y_m)$ is the optimum solution.

- **How do we 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.

Algorithm 7.2 A Prune-and-Search Algorithm to Solve a Special Linear Programming Problem.

Input: Constrains S: $a_j x + b_j$, i=1, 2, …, n.

Output: The value x_0 such that y is minimized at x_0 subject to $y \geq a_j x + b_j$, 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 $a_i x + b_i$ and $a_j x + b_j$, find the intersection p_{ij} of them and denote its x-value as x_{ij}.

Step 3. Among the x_{ij}'s (at most n/2) of them , find the median x_m.

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

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

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

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 $x_0 > x_m$, for each pair of constraints whose x-coordinate intersection is less than x_m, prune away the constraint which is always smaller than the other for $x \geq x_m$.

Let S denote the remaining of contains. 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)

- **The general two-variable linear programming problem:**

$$\begin{cases} \text{minimize } ax + by \\ \text{subject to } a_i x + b_i y \geq c_i \quad , i = 1, 2, \ldots, n \end{cases}$$

Let $x' = x$

$y' = ax + by$

$\Downarrow$

$$\begin{cases} \text{minimize } y' \\ \text{subject to } a_i' x' + b_i' y' \geq c_i' \quad , i = 1, 2, \ldots, n \\ \text{where } a_i' = a_i - b_i a/b \\ \qquad\quad b_i' = b_i/b \\ \qquad\quad c_i' = c_i \end{cases}$$

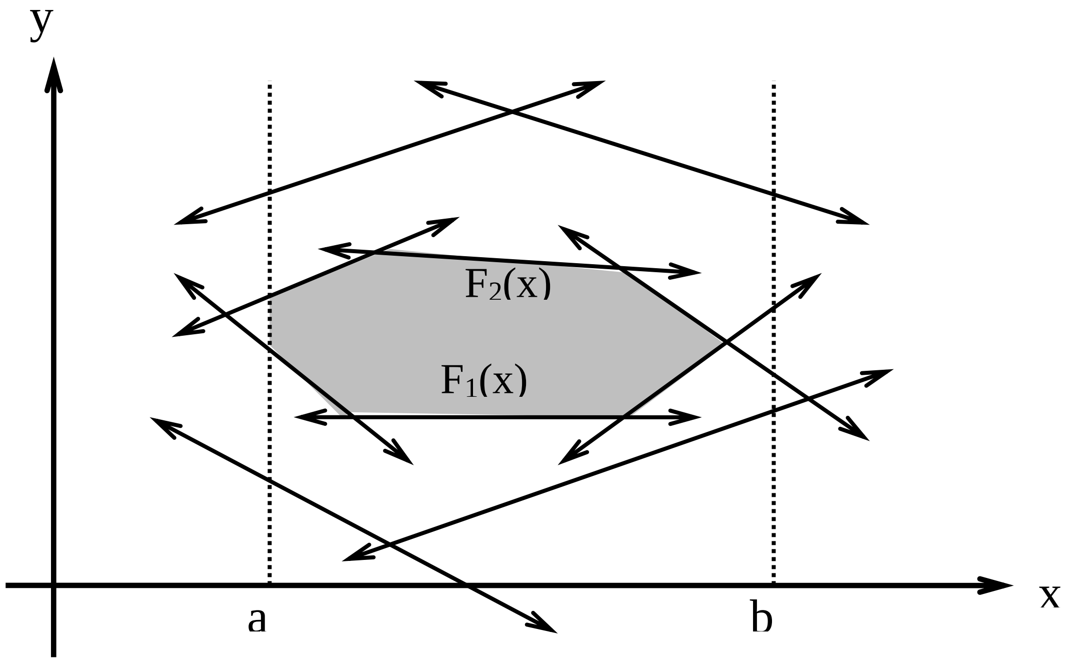
Change the symbols and rewrite as:

$$\begin{cases} \text{minimize } y \\ \text{subject to } y \geq a_i x + b_i y \ (i \in I_1) \\ \qquad\qquad\quad y \leq a_i x + b_i y \ (i \in I_2) \\ \qquad\qquad\quad a \leq x \leq b \end{cases}$$

define:

$F_1(x) = \max \{a_i x + b_i \ , \ i \in I_1\}$

$F_2(x) = \min \{a_i x + b_i \ , \ i \in I_2\}$



$\Downarrow$

minimize $F_1(x)$

subject to $F_1(x) \le F_2(x)$

$a \le x \le b$

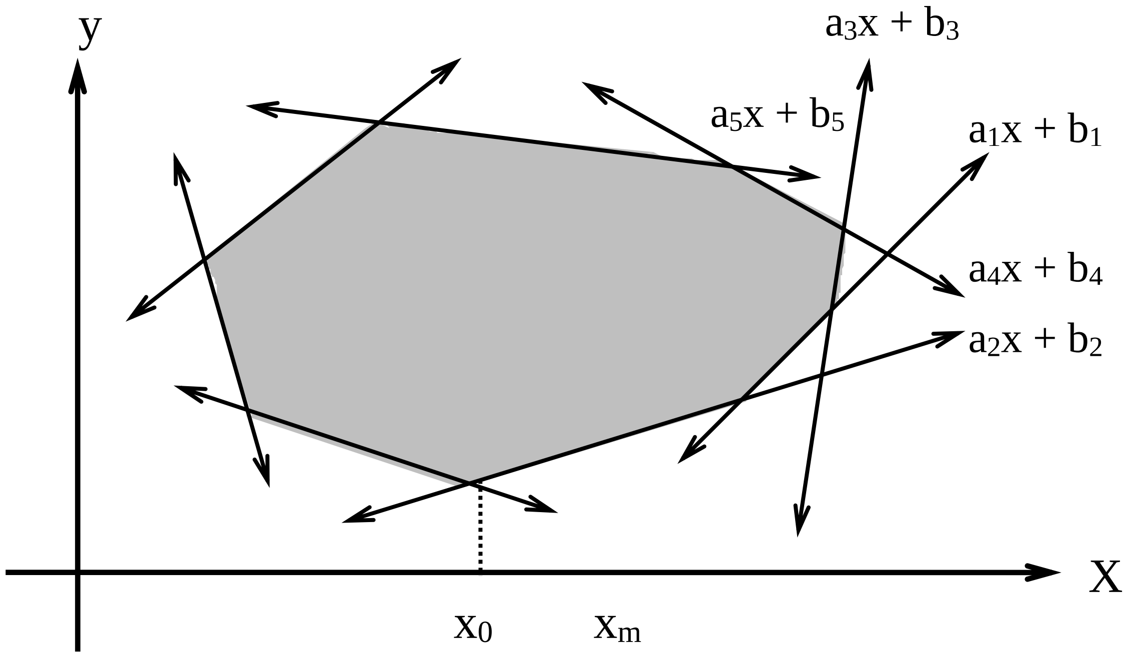


Fig. 7-9 The Pruning of Constraints for the General Two-Variable Linear Programming Problem

In Fig. 7-9, if we know $x_0 < x_m$, then $a_1x + b_1$ can be delete because $a_1x + b_1 < a_2x + b_2$ for $x < x_m$.

Let $F(x) = F_1(x) - F_2(x)$

x_m is feasible $\Leftrightarrow F(x_m) \le 0$

define:

$g_{min} = \min \{a_i \mid i \in I_1, a_ix_m + b_i = F_1(x_m)\}$, min. slope

$g_{max} = \max\{a_i \mid i \in I_1, a_ix_m + b_i = F_1(x_m)\}$, max. slope

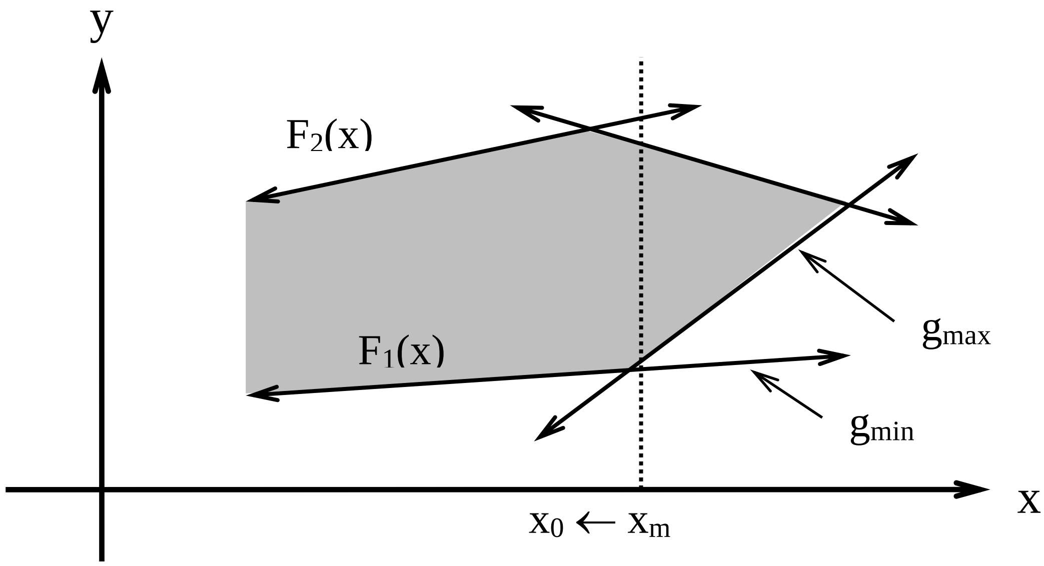
$h_{min} = \min \{a_i \mid i \in I_2, a_ix_m + b_i = F_2(x_m)\}$, min. slope

$h_{min} = \max\{a_i \mid i \in I_2, a_ix_m + b_i = F_2(x_m)\}$, max. slope

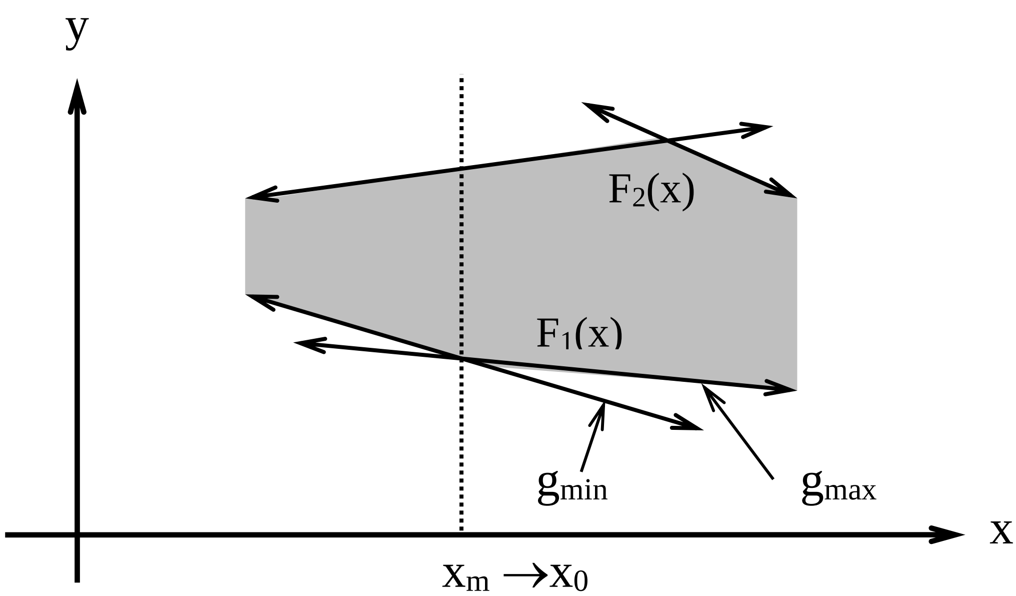
Case 1: $F(x_m) \leq 0$

$\Rightarrow x_m$ is feasible

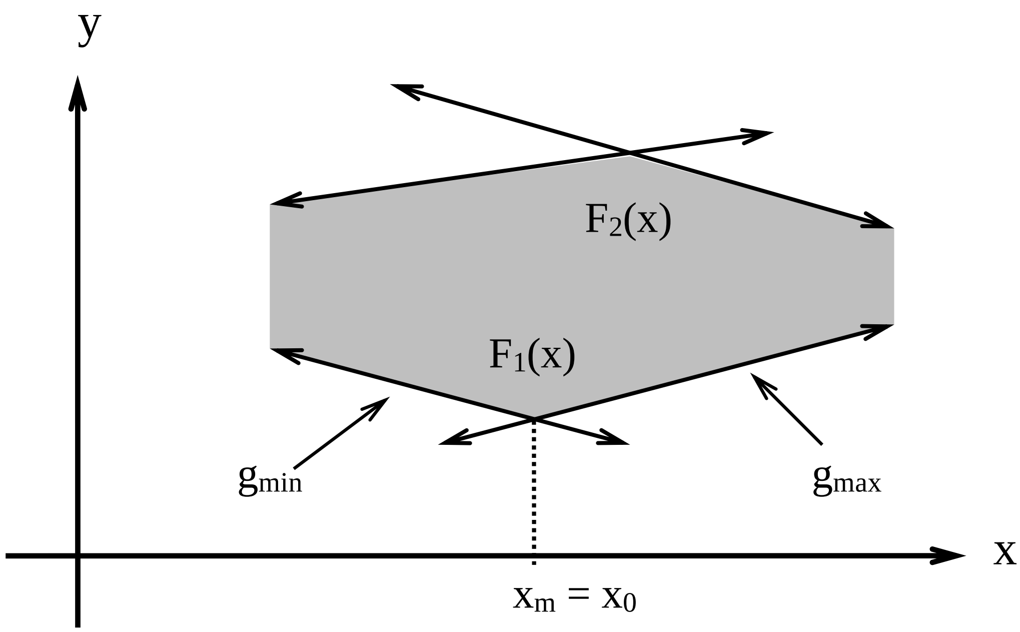
(a) If $g_{min} > 0$, $g_{max} > 0$, then $x_0 < x_m$.



(b) If $g_{min} < 0$, $g_{max} < 0$, then $x_0 > x_m$.



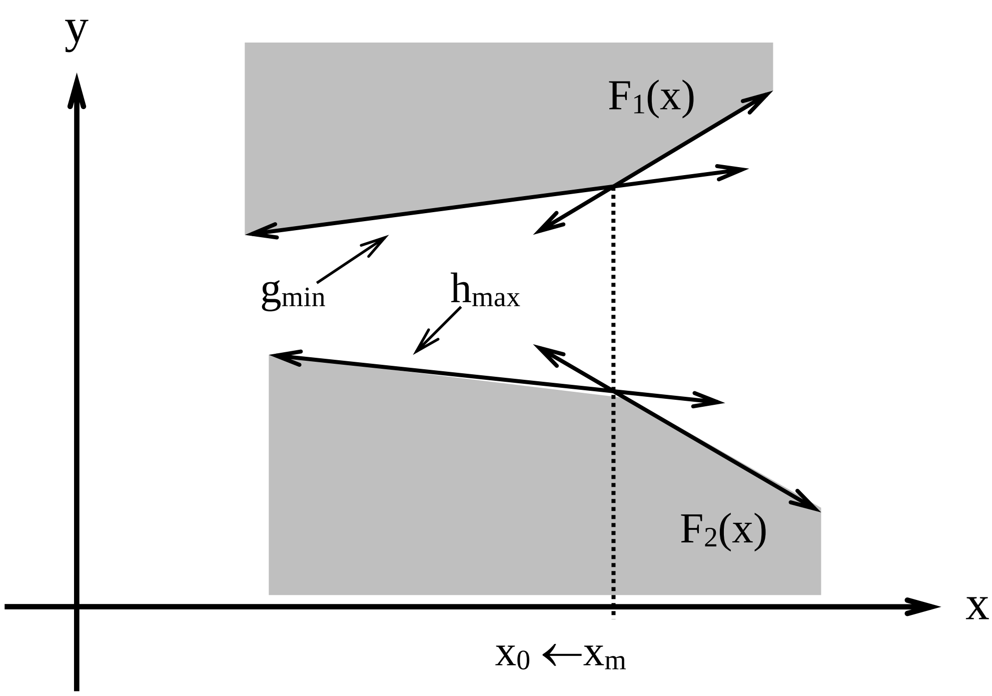
(c) If $g_{min} < 0$, $g_{max} > 0$, then x_m is the optimum solution.



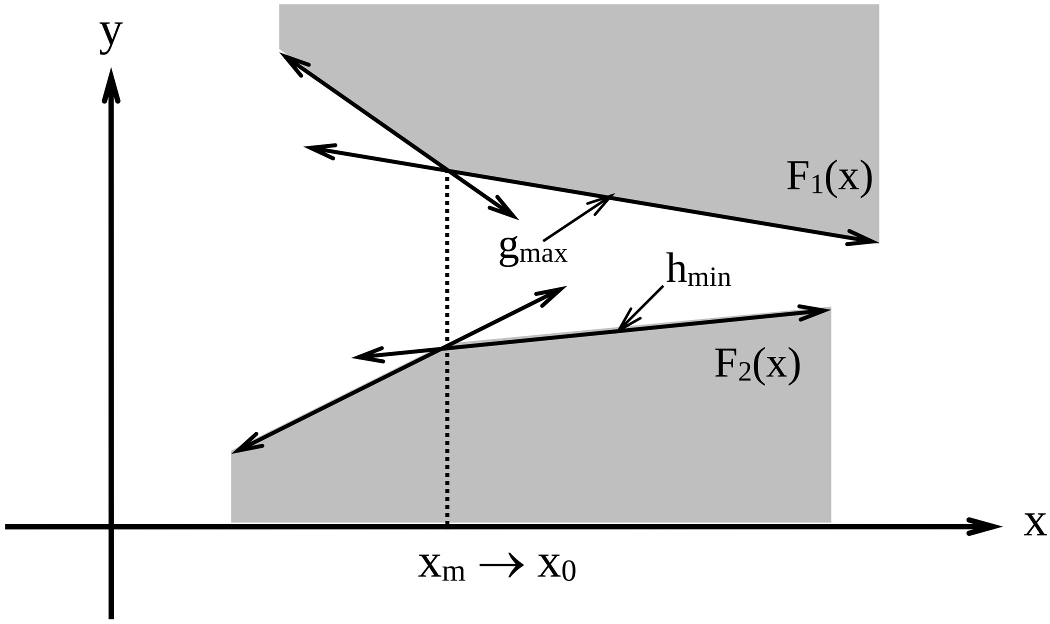
Case 2: $F(x_m) > 0$

$\Rightarrow x_m$ is infeasible

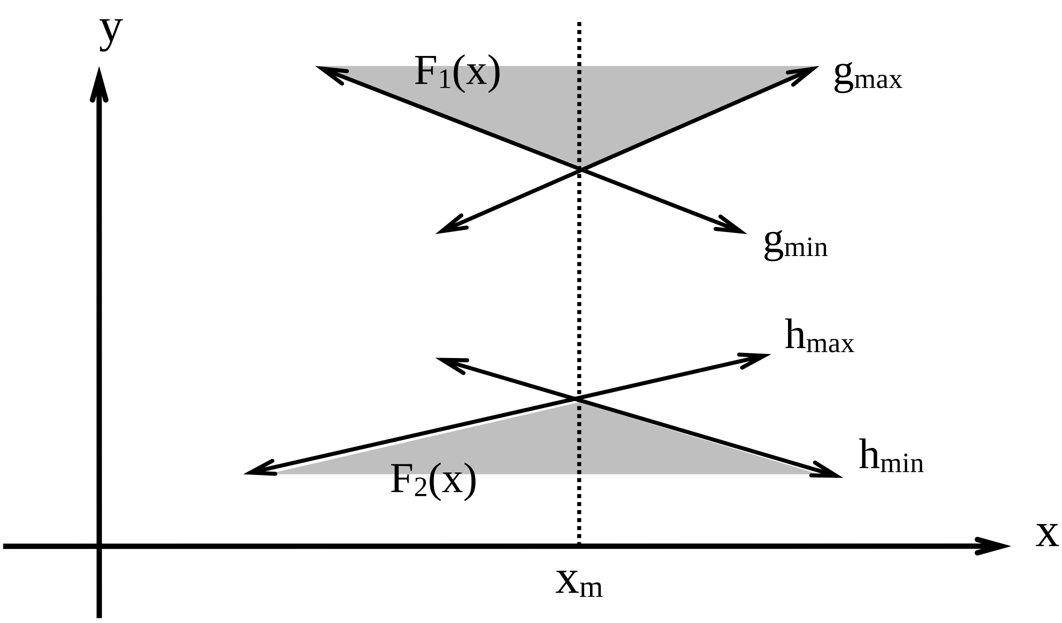
(a) If $g_{min} > h_{max}$, then $x_0 < x_m$.



(b) If $g_{min} < h_{max}$, then $x_0 > x_m$.



(c) If $g_{min} \leq h_{max}$, and $g_{max} \geq h_{min}$, then no feasible solution exists.



Algorithm 7.3 A Prune-and-Search Algorithm to Solve the Two-Variable Linear Programming Problem.

Input: I_1: $y \ge a_i x + b_i$, $i = 1, 2, \ldots, n_1$
I_2: $y \le a_i x + b_i$, $i = n_1+1, n_1+2, \ldots, n.$
$a \le x \le b$

Output: The value x_0 such that
y is minimized at x_0
subject to $y \ge a_i x + b_i$, $i = 1, 2, \ldots, n_1$
$y \le a_i x + b_i$, $i = n_1+1, n_1+2, \ldots, n.$
$a \le x \le b$

Step 1. Arrange the constraints in I_1 and I_2 into arbitrary disjoint pairs respectively. For each pair, if $a_i x + b_i$ is parallel to $a_j x + b_j$, delete $a_i x + b_i$ if $b_i < b_j$ for $i, j \in I_1$ or $b_i > b_j$ for $i, j \in I_2$. Otherwise, find the intersection p_{ij} of $y = a_i x + b_i$ and $y = a_j x + b_j$. Let the x-coordinate of p_{ij} be x_{ij}.

Step 2. Find the median x_m of x_{ij}'s (at most $\left\lfloor \frac{n}{2} \right\rfloor$ of them).

Step 3. (a) If x_m is optimal, report this and exit.
(b) If no feasible solution exists, report this and exit.
(c) Otherwise, determine whether the optimum solution lies to the left, or right, of x_m.

Step 4. Discard at least 1/4 of the constraints.
Go to Step 1.

time complexity: O(n)

- **The 1-center problem**

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

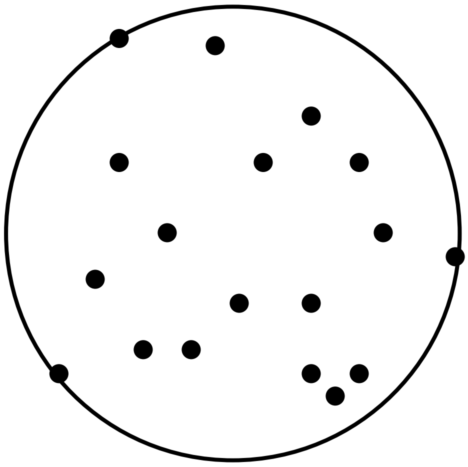
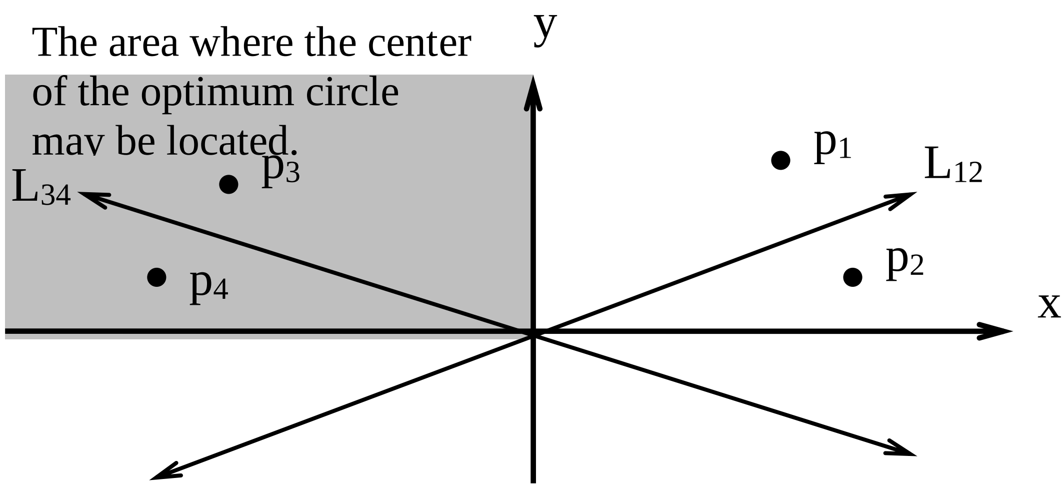


Fig. 7-16 The 1-Center Problem



$L_{1\,2}$: bisector of $\overline{P_1P_2}$, $L_{3\,4}$: bisector of $\overline{P_3P_4}$

P_1 can be eliminated without affecting our solution.

- **The constrained 1-center problem:**

  The center is restricted to lying on a straight line.

Algorithm 7.4 An Algorithm to Solve the Constrained 1-Center Problem.

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 the 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 $x^* < x_m$ (as illustrated in Fig. 7-18),
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 $x^* > x_m$,
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}.

Step 7. Go to Step 1.

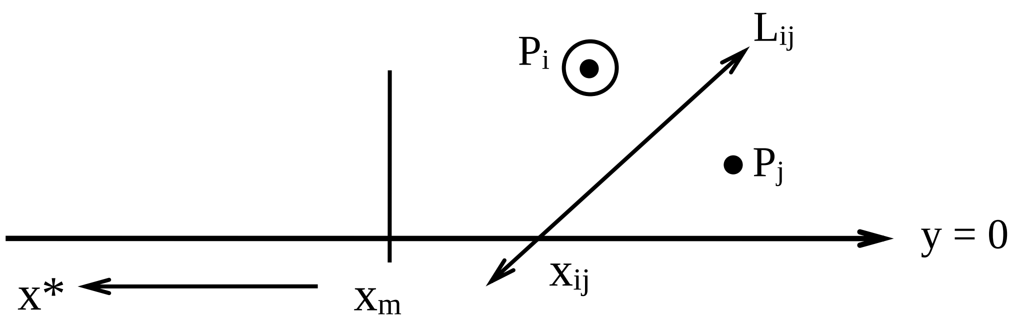


Fig. 7-18 The Pruning of Points in the Constrained 1-Center Problem

time complexity O(n)

- **The general 1-center problem**

  For the constrained 1-center problem, let $(x^*, 0)$ be the center on the line y = 0.

  Let (x_s, y_s) be the center of the optimum circle.

  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.

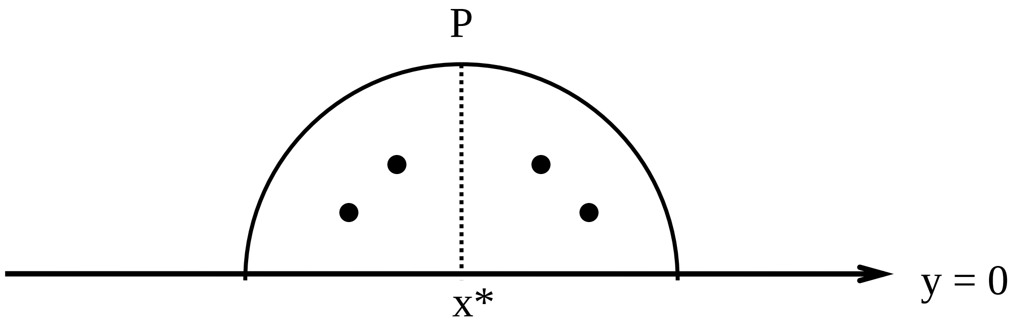
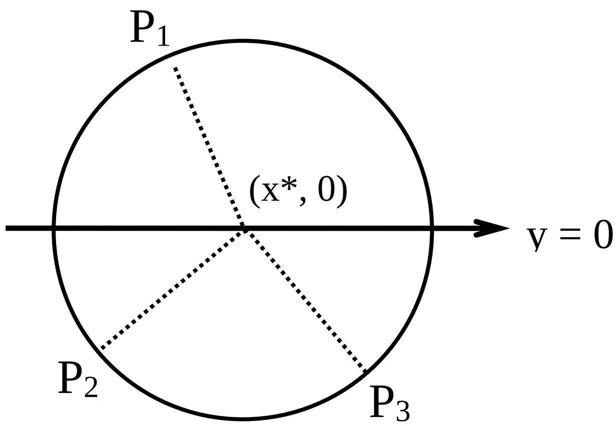


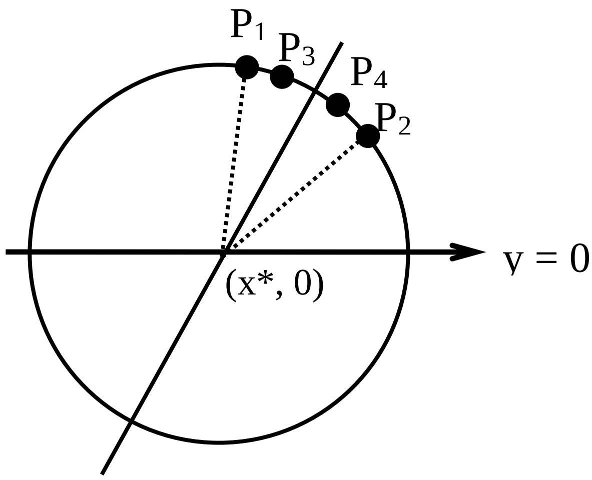
Fig. 7-20 The Case where I Contains Only One Point

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 $\frac{y_1 + y_2}{2}$.



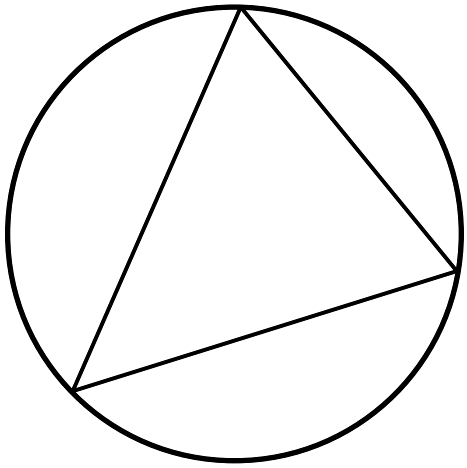
(a)



(b)

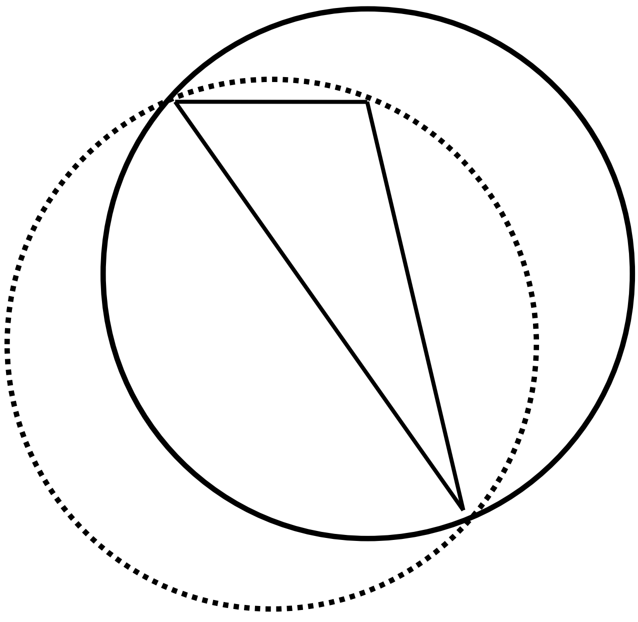
Why?

an acute triangle:



The circle is optimal.

an obtuse triangle:



The circle is not optimal.

Consider the case where the smallest spanning arc < 180^o. $(x_1 - x^*)$ and $(x_2 - x^*)$ must be of opposite signs. Otherwise, we can move x^* toward the direction where P_1 and P_2 are located.

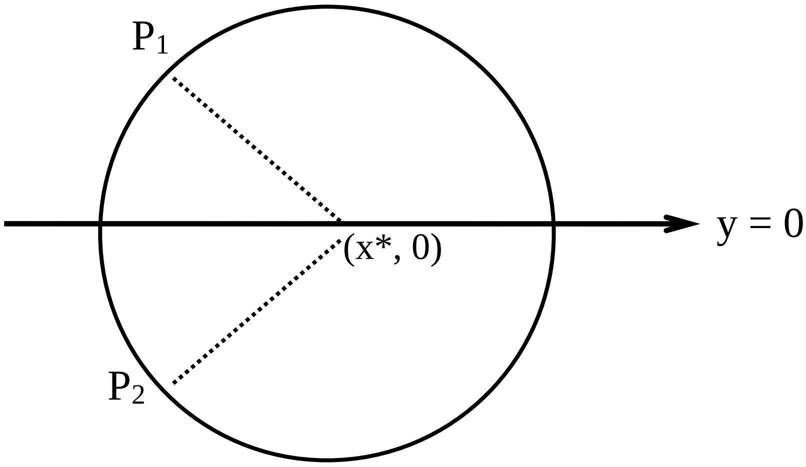


Fig.7-23 The Direction of x* where the Degree Is Less than 180^0

Let P_1 = (a, b) and P_2 = (c, d). without losing generality, we may assume that

$a > x^*$, $b > 0$

and $c < x^*$, $d > 0$.

Let the radius of the current circle be r.

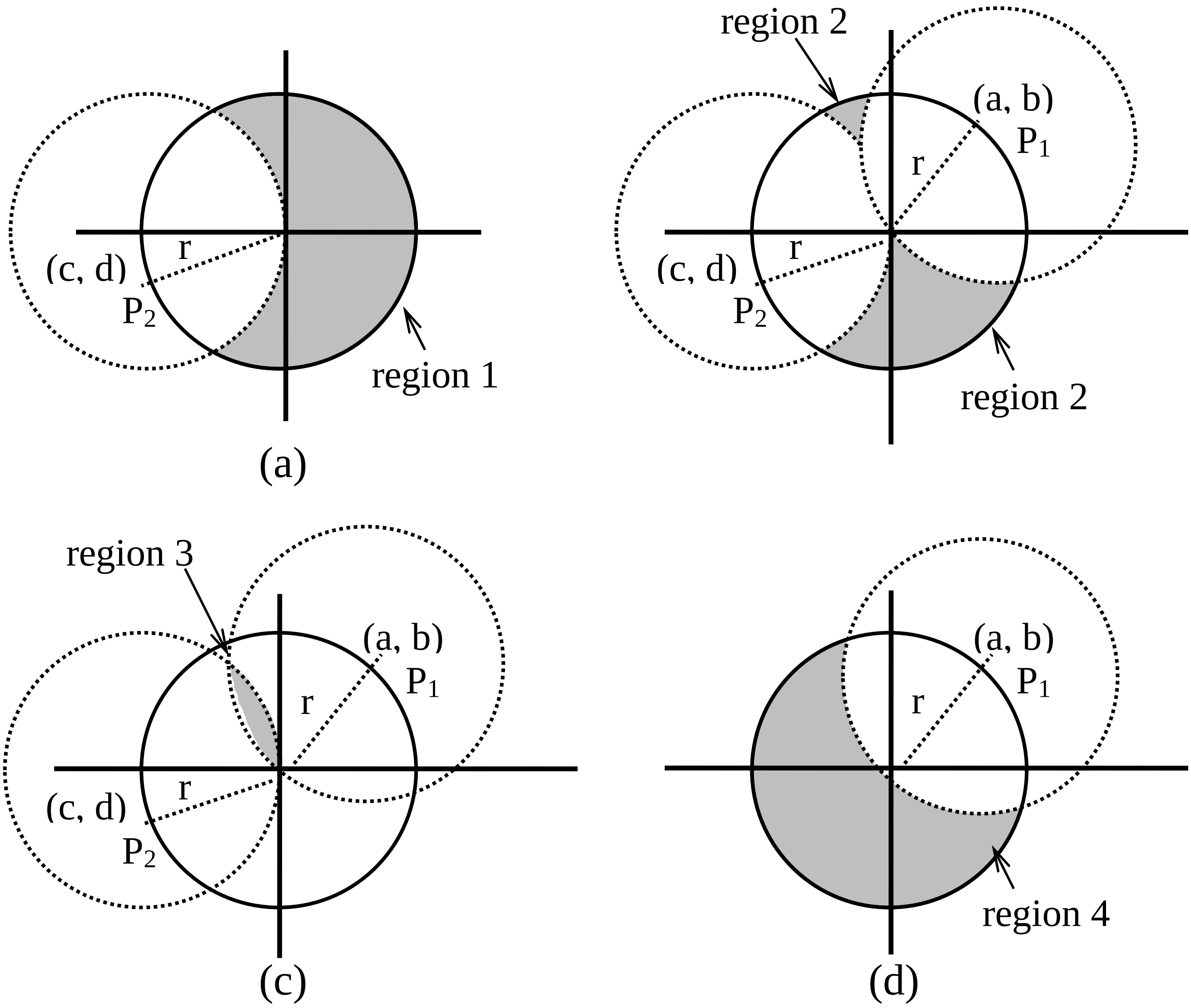


Fig. 7-24 The Direction of x* where the Degree is Larger than 180°

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 $\frac{a+c}{2} = \frac{x_1+x_2}{2}$.

<u>Algorithm 7.5 A Prune-and-Search Algorithm to</u>

Solve the 1-Center Problem

Input: A set S = {p_1, p_2, …, 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. From 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, …, n, and compute their slopes. Let the slope of L_k be denoted as s_k, for k = 1, 2, 3, …, 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_mx. 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, …, 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, …, n/4.
Step 6. Find the median of b_i's. Denote it as y*. apply the constrained 1-center subroutine of this constrained 1-center problem be (x', y*).
Step 7. Apply procedure 7.2, using S and (x', y*) as the parameters. If y_s = y*, report, "The circle

found in step 6 with (x', y^*) as the center is the optimal solution" and 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 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. Apply 7.2, using S and (x^*, y') as the parameters. If $x_s = x^*$, report "the circle found in Step 8 with (x^*, y') as the center is the optimal solution" and 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-}. (See Step 5.). 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 $y_s > y^*$.
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 $p_k(p_j)$.

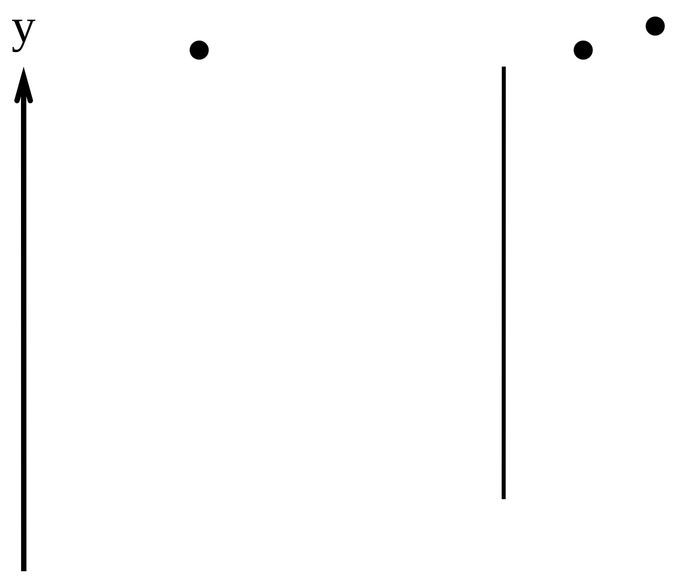
Case 3: $x_s < x^*$ and $y_s < y^*$.

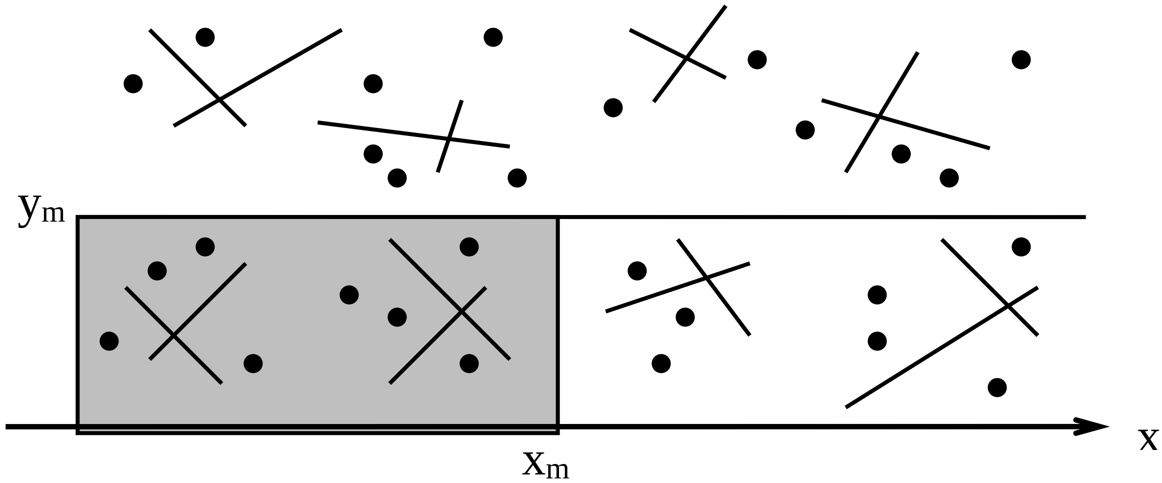
Find all (a_i, b_i)'s such that $a_i > x^*$ and $b_i > y^*$. Let (a_i, b_i) be the intersection of L_{i+} and L_{i-}. Let L_{i-} be the bisector of p_j and p_k. Prune away $p_j(p_k)$ if $p_j(p_k)$ is closer to (x^*, y^*) than $p_k(p_j)$.

Case 4: $x_s > x^*$ 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 $p_k(p_j)$.

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





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.

time complexity:

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