

INTERSECTING HALF-PLANES and related problems

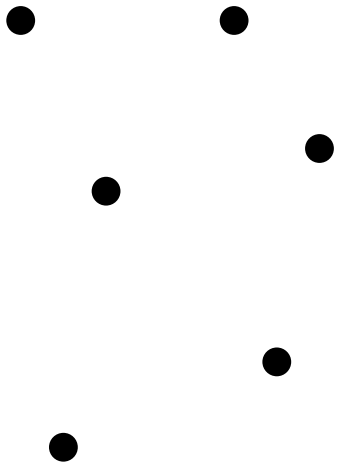
Vera Sacristán
Rodrigo Silveira

Research Group on
Discrete, Combinatorial and Computational Geometry
Universitat Politècnica de Catalunya

WHAT DO THESE TWO PROBLEMS HAVE IN COMMON?

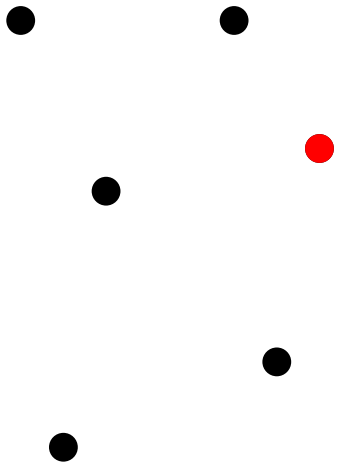
WHAT DO THESE TWO PROBLEMS HAVE IN COMMON?

Given n points in the plane,
find the rightmost one



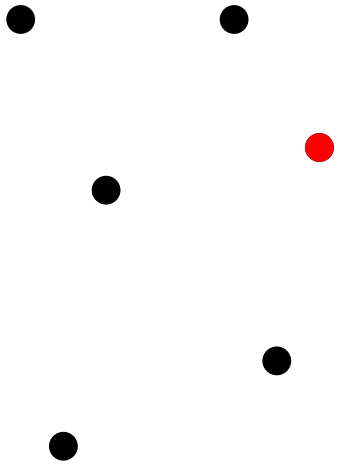
WHAT DO THESE TWO PROBLEMS HAVE IN COMMON?

Given n points in the plane,
find the rightmost one

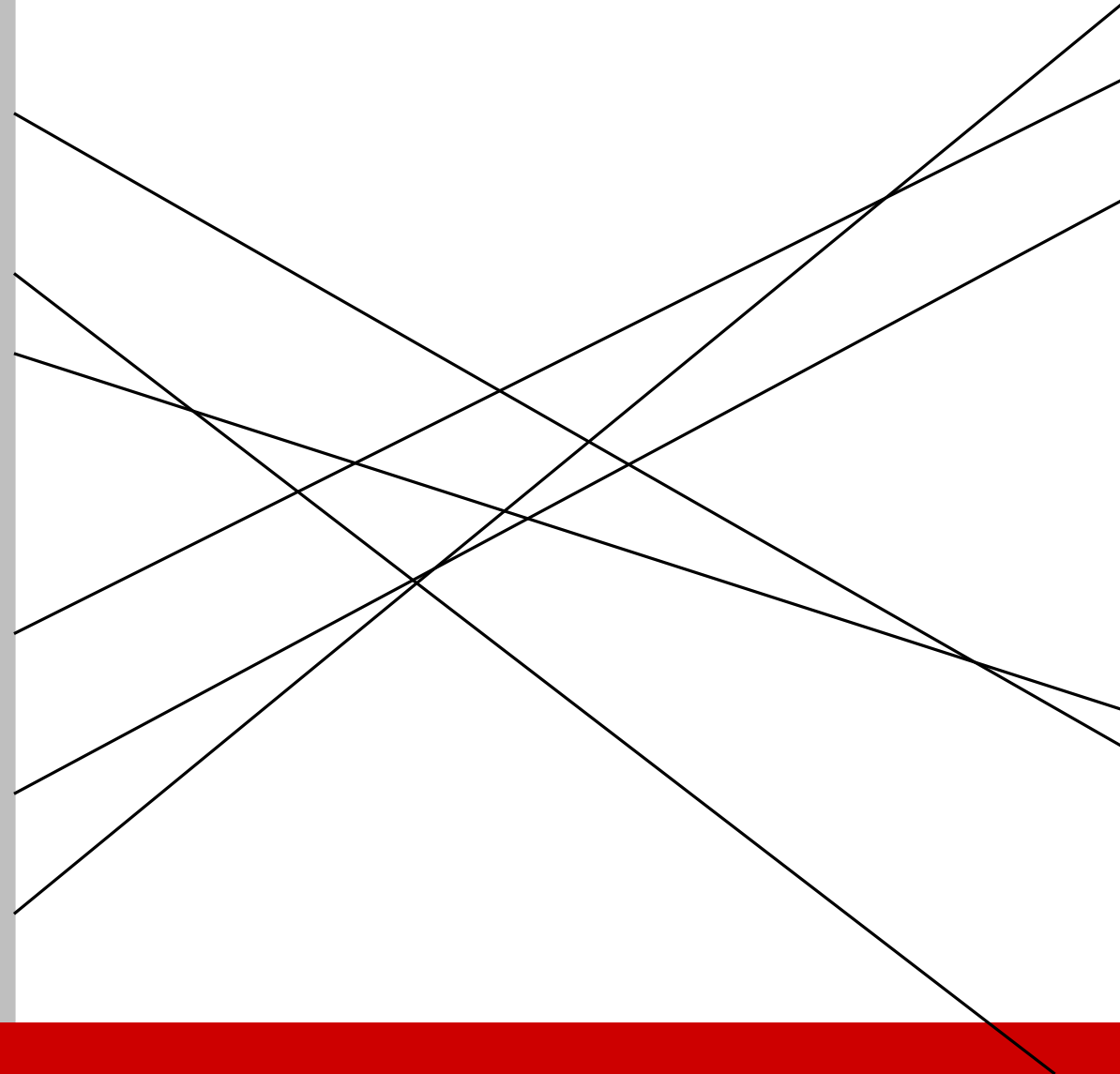


WHAT DO THESE TWO PROBLEMS HAVE IN COMMON?

Given n points in the plane,
find the rightmost one

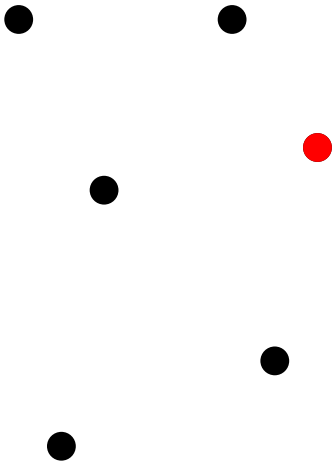


Given n lines in the plane,
find the one with largest slope

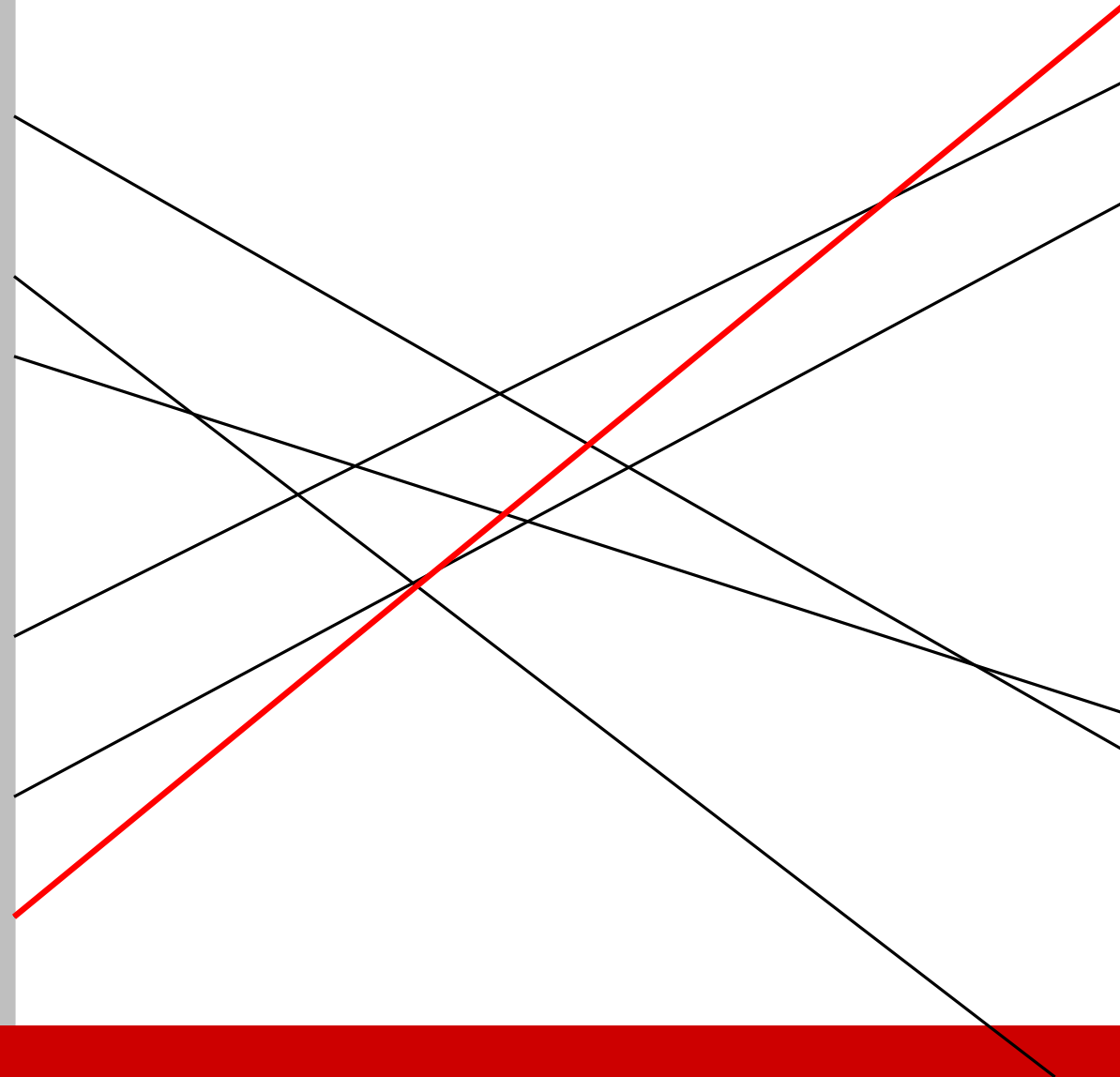


WHAT DO THESE TWO PROBLEMS HAVE IN COMMON?

Given n points in the plane,
find the rightmost one



Given n lines in the plane,
find the one with largest slope



DUALIZING POINTS AND LINES

Duality

DUALIZING POINTS AND LINES

Duality

In the plane, dualities transform points into lines and lines into points.

One of the most frequently used is:

$$\begin{array}{rcl} \mathbb{R}^2 & \longrightarrow & \mathbb{R}^2 \\ p = (a, b) & \mapsto & D(p) = p^* : y = ax - b \\ \ell : y = mx + n & \mapsto & D(\ell) = \ell^* = (m, -n) \end{array}$$

DUALIZING POINTS AND LINES

Duality

In the plane, dualities transform points into lines and lines into points.

One of the most frequently used is:

$$\begin{array}{rcl} \mathbb{R}^2 & \longrightarrow & \mathbb{R}^2 \\ p = (a, b) & \mapsto & D(p) = p^* : y = ax - b \\ \ell : y = mx + n & \mapsto & D(\ell) = \ell^* = (m, -n) \end{array}$$

Properties

1. $D^2 = Id$.
2. D is a bijection between points and non-vertical lines.
3. A point p lies above/on/below a line ℓ if and only if the point ℓ^* lies above/on/below the line p^* .

DUALIZING POINTS AND LINES

Duality

In the plane, dualities transform points into lines and lines into points.

One of the most frequently used is:

$$\begin{array}{ccc} \mathbb{R}^2 & \longrightarrow & \mathbb{R}^2 \\ p = (a, b) & \mapsto & D(p) = p^* : y = ax - b \\ \ell : y = mx + n & \mapsto & D(\ell) = \ell^* = (m, -n) \end{array}$$

Properties

1. $D^2 = Id$.
2. D is a bijection between points and non-vertical lines.
3. A point p lies above/on/below a line ℓ if and only if the point ℓ^* lies above/on/below the line p^* .

Proof: If $p = (a, b)$ and ℓ is the line of equation $y = mx + n$, the relative position of p wrt ℓ is given by the (in)equation $b \lessgtr ma + n$, which is equivalent to $-n \lessgtr am - b$, and this expresses the relative position of $\ell^* = (m, -n)$ wrt $p^* : y = ax - b$.

DUALIZING POINTS AND LINES

Duality

In the plane, dualities transform points into lines and lines into points.

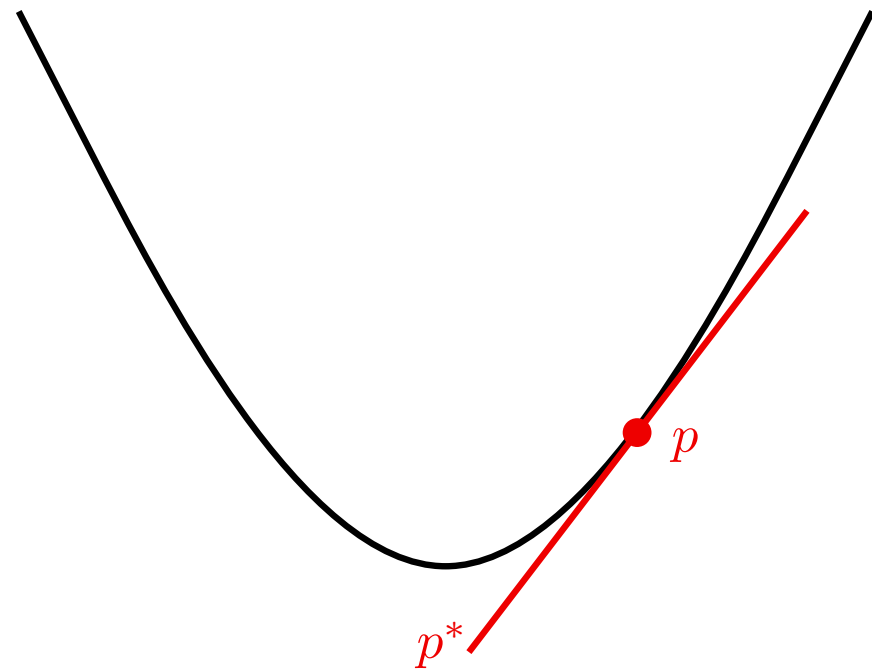
One of the most frequently used is:

$$\begin{array}{rcl} \mathbb{R}^2 & \longrightarrow & \mathbb{R}^2 \\ p = (a, b) & \mapsto & D(p) = p^* : y = ax - b \\ \ell : y = mx + n & \mapsto & D(\ell) = \ell^* = (m, -n) \end{array}$$

Duality and parabola

Consider the parabola $y = x^2/2$.

- If p is a point on the parabola, then p^* is the line tangent to the parabola at point p .



DUALIZING POINTS AND LINES

Duality

In the plane, dualities transform points into lines and lines into points.

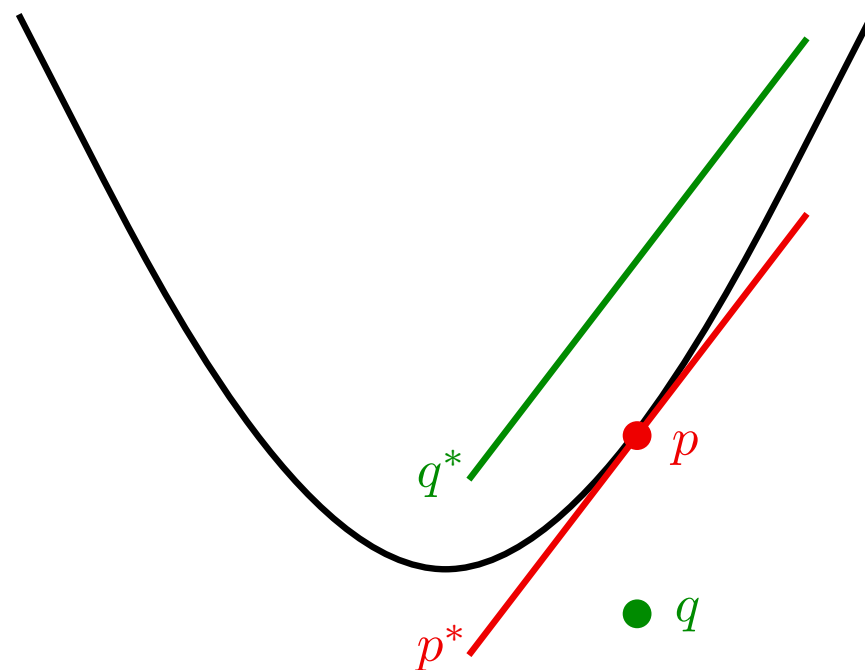
One of the most frequently used is:

$$\begin{array}{rcl} \mathbb{R}^2 & \longrightarrow & \mathbb{R}^2 \\ p = (a, b) & \mapsto & D(p) = p^* : y = ax - b \\ \ell : y = mx + n & \mapsto & D(\ell) = \ell^* = (m, -n) \end{array}$$

Duality and parabola

Consider the parabola $y = x^2/2$.

- If p is a point on the parabola, then p^* is the line tangent to the parabola at point p .
- Let q be any point in the plane and let p be its vertical projection onto the parabola. Then q^* is the line parallel to p^* whose vertical distance from p^* is the opposite to the vertical distance from q to p .



DUALIZING POINTS AND LINES

Duality

In the plane, dualities transform points into lines and lines into points.

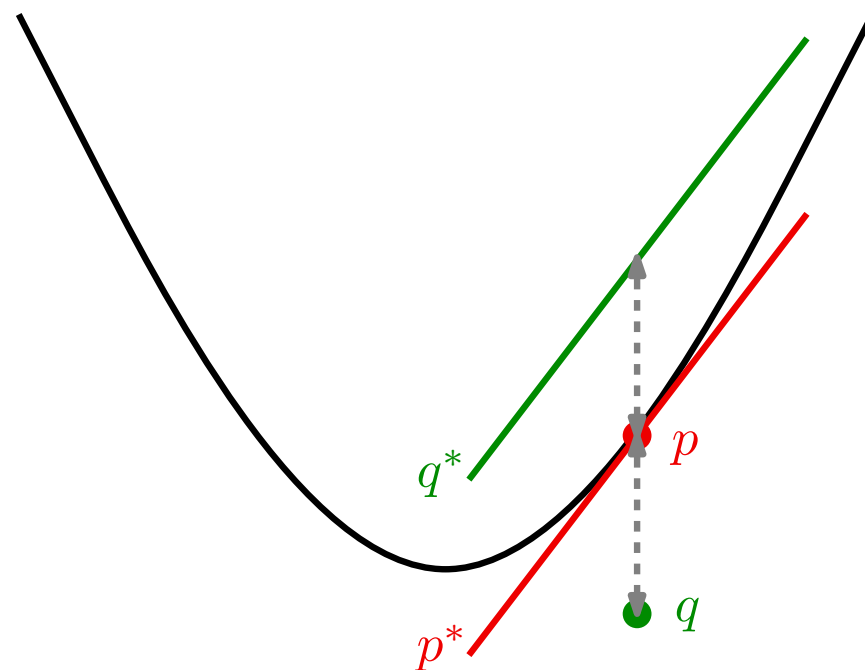
One of the most frequently used is:

$$\begin{array}{rcl} \mathbb{R}^2 & \longrightarrow & \mathbb{R}^2 \\ p = (a, b) & \mapsto & D(p) = p^* : y = ax - b \\ \ell : y = mx + n & \mapsto & D(\ell) = \ell^* = (m, -n) \end{array}$$

Duality and parabola

Consider the parabola $y = x^2/2$.

- If p is a point on the parabola, then p^* is the line tangent to the parabola at point p .
- Let q be any point in the plane and let p be its vertical projection onto the parabola. Then q^* is the line parallel to p^* whose vertical distance from p^* is the opposite to the vertical distance from q to p .



DUALIZING POINTS AND LINES

Duality

In the plane, dualities transform points into lines and lines into points.

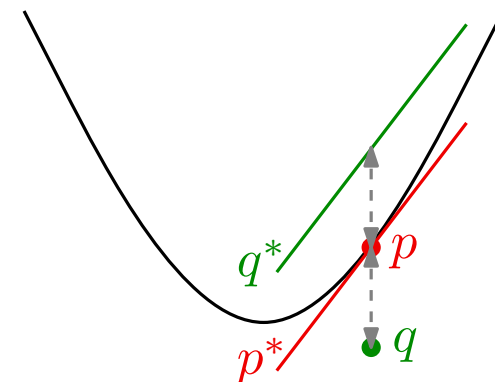
One of the most frequently used is:

$$\begin{array}{rcl} \mathbb{R}^2 & \longrightarrow & \mathbb{R}^2 \\ p = (a, b) & \mapsto & D(p) = p^* : y = ax - b \\ \ell : y = mx + n & \mapsto & D(\ell) = \ell^* = (m, -n) \end{array}$$

Duality and parabola

Consider the parabola $y = x^2/2$.

- If p is a point on the parabola, then p^* is the line tangent to the parabola at point p .
- Let q be any point in the plane and let p be its vertical projection onto the parabola. Then q^* is the line parallel to p^* whose vertical distance from p^* is the opposite to the vertical distance from q to p .



Proof:

Since $y'(x) = x$, the line tangent to the parabola at a point $p = (a, a^2/2)$ has equation $y - a^2/2 = a(x - a)$ i.e., $y = ax - a^2/2$.

For any other point $q = (a, b)$:

$$d_{vertical}(p, q) = b - a^2/2;$$

$$d_{vertical}(p^*, q^*) = -b + a^2/2.$$

DUALIZING POINTS AND LINES

Duality

In the plane, dualities transform points into lines and lines into points.

One of the most frequently used is:

$$\begin{array}{lcl} \mathbb{R}^2 & \longrightarrow & \mathbb{R}^2 \\ p = (a, b) & \mapsto & D(p) = p^* : y = ax - b \\ \ell : y = mx + n & \mapsto & D(\ell) = \ell^* = (m, -n) \end{array}$$

PLAY WITH OUR DEMO!

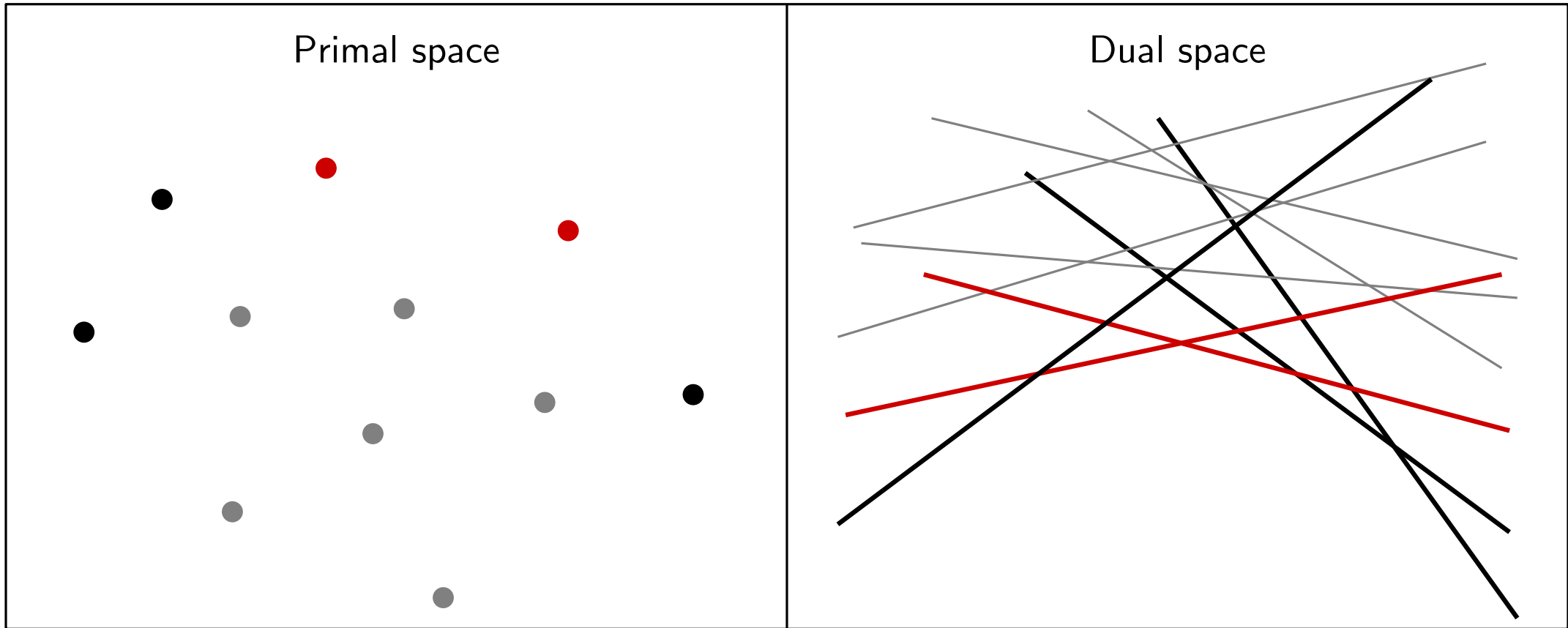
<https://dccg.upc.edu/teaching/applets/duality.html>

INTERSECTING HALF-PLANES

Intersection of half-planes and convex hulls

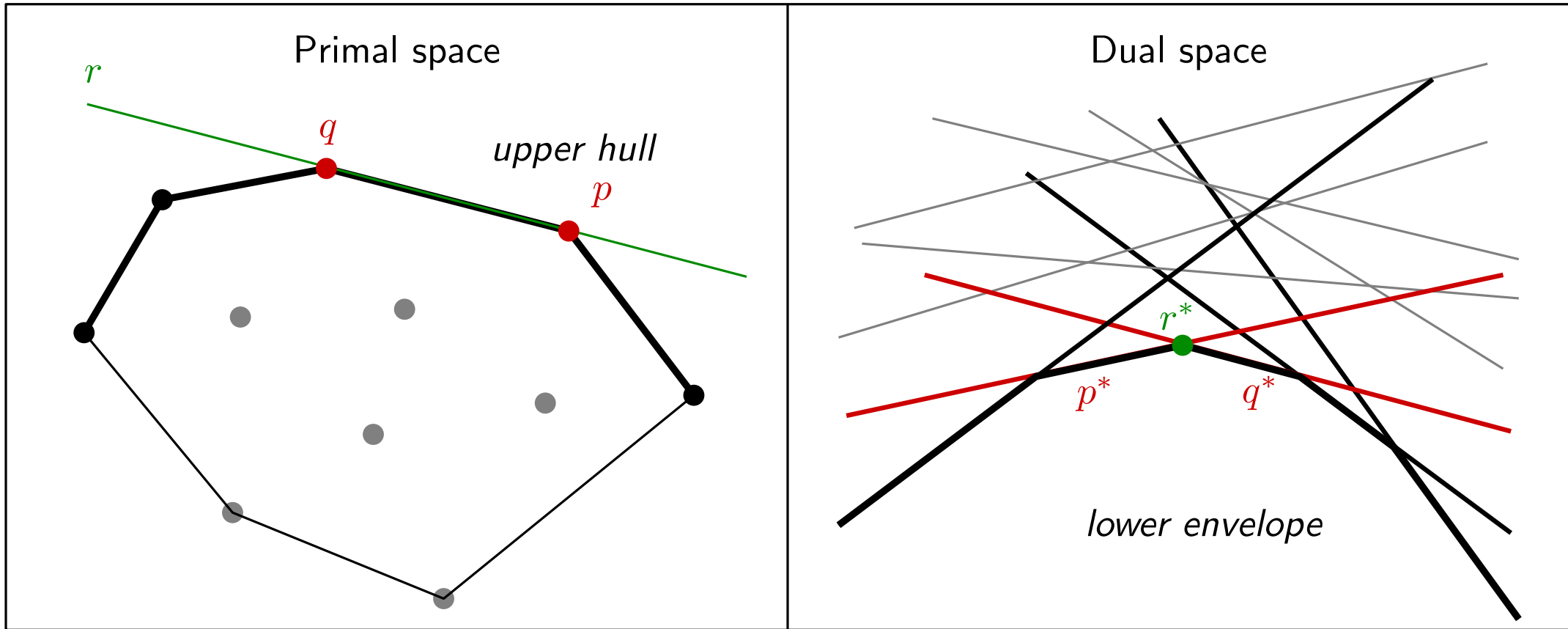
INTERSECTING HALF-PLANES

Intersection of half-planes and convex hulls



INTERSECTING HALF-PLANES

Intersection of half-planes and convex hulls



Lemma Let P be a set of points in the plane. The counterclockwise order of the points along the upper (resp. lower) convex hull of P is equal to the left-to-right order of the sequence of lines on the lower (resp. upper) envelope of the dual P^*

INTERSECTING HALF-PLANES

Computing the intersection of half-planes

Input: n half-planes

Output: Sorted list of vertices (and, possibly, half-lines) defining the boundary of their intersection (a convex polygon or an unbounded convex polygonal region).

INTERSECTING HALF-PLANES

Computing the intersection of half-planes

Input: n half-planes

Output: Sorted list of vertices (and, possibly, half-lines) defining the boundary of their intersection (a convex polygon or an unbounded convex polygonal region).

Algorithm 1: by duality

Solves the problem in $O(n \log n)$ time by dualizing and computing two convex hulls.

INTERSECTING HALF-PLANES

Computing the intersection of half-planes

Input: n half-planes

Output: Sorted list of vertices (and, possibly, half-lines) defining the boundary of their intersection (a convex polygon or an unbounded convex polygonal region).

Algorithm 1: by duality

Solves the problem in $O(n \log n)$ time by dualizing and computing two convex hulls.

Algorithm 2: incremental

Solves the problem in $O(n \log n)$ time, by incrementally intersecting a convex polygon (or unbounded convex polygonal region) with a halfplane.

INTERSECTING HALF-PLANES

Computing the intersection of half-planes

Input: n half-planes

Output: Sorted list of vertices (and, possibly, half-lines) defining the boundary of their intersection (a convex polygon or an unbounded convex polygonal region).

Algorithm 1: by duality

Solves the problem in $O(n \log n)$ time by dualizing and computing two convex hulls.

Algorithm 2: incremental

Solves the problem in $O(n \log n)$ time, by incrementally intersecting a convex polygon (or unbounded convex polygonal region) with a halfplane.

Algorithm 3: by divide and conquer

Solves the problem in $O(n \log n)$ time: the merging step consists in computing the intersection of two convex polygons (or unbounded convex polygonal regions) in $O(n)$ time.

INTERSECTING HALF-PLANES

Computing the intersection of half-planes

Input: n half-planes

Output: Sorted list of vertices (and, possibly, half-lines) defining the boundary of their intersection (a convex polygon or an unbounded convex polygonal region).

Algorithm 1: by duality

Solves the problem in $O(n \log n)$ time by dualizing and computing two convex hulls.

Algorithm 2: incremental

Solves the problem in $O(n \log n)$ time, by incrementally intersecting a convex polygon (or unbounded convex polygonal region) with a halfplane.

Algorithm 3: by divide and conquer

Solves the problem in $O(n \log n)$ time: the merging step consists in computing the intersection of two convex polygons (or unbounded convex polygonal regions) in $O(n)$ time.

Lower bound

Duality reduces in $O(n)$ time the problem of computing the convex hull of n points to the problem of computing the intersection of n half-planes. Therefore, computing the intersection of n half-planes has an $\Omega(n \log n)$ lower bound.

SOLVING LINEAR PROGRAMS

Linear program (2D)

Optimize a linear function $ax + by$

subject to n linear restrictions $a_i x + b_i y + c_i \leq 0$, where $i = 1, \dots, n$.

SOLVING LINEAR PROGRAMS

Example of a typical linear program

SOLVING LINEAR PROGRAMS

Example of a typical linear program

A manufacturer produces two products, X and Y , with two machines, A and B .

SOLVING LINEAR PROGRAMS

Example of a typical linear program

A manufacturer produces two products, X and Y , with two machines, A and B .

The cost of producing each unit of X is:

- for machine A : 50 minutes,
- for machine B : 30 minutes.

The cost of producing each unit of Y is:

- for machine A : 24 minutes,
- for machine B : 33 minutes.

SOLVING LINEAR PROGRAMS

Example of a typical linear program

A manufacturer produces two products, X and Y , with two machines, A and B .

The cost of producing each unit of X is:

- for machine A : 50 minutes,
- for machine B : 30 minutes.

The cost of producing each unit of Y is:

- for machine A : 24 minutes,
- for machine B : 33 minutes.

Working plans for a particular week are:

- 40 hours of work on machine A ,
- 35 hours of work on machine B .

SOLVING LINEAR PROGRAMS

Example of a typical linear program

A manufacturer produces two products, X and Y , with two machines, A and B .

The cost of producing each unit of X is:

- for machine A : 50 minutes,
- for machine B : 30 minutes.

Working plans for a particular week are:

- 40 hours of work on machine A ,
- 35 hours of work on machine B .

The cost of producing each unit of Y is:

- for machine A : 24 minutes,
- for machine B : 33 minutes.

The week starts with:

- A stock of 30 units of X and 90 of Y ,
- A demand of 75 units of X and 95 of Y .

SOLVING LINEAR PROGRAMS

Example of a typical linear program

A manufacturer produces two products, X and Y , with two machines, A and B .

The cost of producing each unit of X is:

- for machine A : 50 minutes,
- for machine B : 30 minutes.

The cost of producing each unit of Y is:

- for machine A : 24 minutes,
- for machine B : 33 minutes.

Working plans for a particular week are:

- 40 hours of work on machine A ,
- 35 hours of work on machine B .

The week starts with:

- A stock of 30 units of X and 90 of Y ,
- A demand of 75 units of X and 95 of Y .

How to plan the production, in order to end the week with the maximum stock?

SOLVING LINEAR PROGRAMS

Example of a typical linear program

A manufacturer produces two products, X and Y , with two machines, A and B .

The cost of producing each unit of X is:

- for machine A : 50 minutes,
- for machine B : 30 minutes.

The cost of producing each unit of Y is:

- for machine A : 24 minutes,
- for machine B : 33 minutes.

Working plans for a particular week are:

- 40 hours of work on machine A ,
- 35 hours of work on machine B .

The week starts with:

- A stock of 30 units of X and 90 of Y ,
- A demand of 75 units of X and 95 of Y .

How to plan the production, in order to end the week with the maximum stock?

Maximize $(x + 30 - 75) + (y + 90 - 95)$

x = units of X to be produced

y = units of Y to be produced

SOLVING LINEAR PROGRAMS

Example of a typical linear program

A manufacturer produces two products, X and Y , with two machines, A and B .

The cost of producing each unit of X is:

- for machine A : 50 minutes,
- for machine B : 30 minutes.

The cost of producing each unit of Y is:

- for machine A : 24 minutes,
- for machine B : 33 minutes.

Working plans for a particular week are:

- 40 hours of work on machine A ,
- 35 hours of work on machine B .

The week starts with:

- A stock of 30 units of X and 90 of Y ,
- A demand of 75 units of X and 95 of Y .

How to plan the production, in order to end the week with the maximum stock?

Maximize $(x + 30 - 75) + (y + 90 - 95)$

x = units of X to be produced

y = units of Y to be produced

restricted to:

$$50x + 24y \leq 40 \times 60$$

$$30x + 33y \leq 35 \times 60$$

$$x \geq 75 - 30$$

$$y \geq 95 - 90$$

SOLVING LINEAR PROGRAMS

Linear program (2D)

Optimize a linear function $ax + by$

subject to n linear restrictions $a_i x + b_i y + c_i \leq 0$, where $i = 1, \dots, n$.

SOLVING LINEAR PROGRAMS

Linear program (2D)

Optimize a linear function $ax + by$

subject to n linear restrictions $a_i x + b_i y + c_i \leq 0$, where $i = 1, \dots, n$.

or, equivalently (apply the linear transform $Y = \pm(ax + by)$, $X = x$):

Minimize Y

restricted to $a_i X + b_i Y + c_i \leq 0$, with $i = 1, \dots, n$.

SOLVING LINEAR PROGRAMS

Linear program (2D)

Optimize a linear function $ax + by$

subject to n linear restrictions $a_i x + b_i y + c_i \leq 0$, where $i = 1, \dots, n$.

or, equivalently (apply the linear transform $Y = \pm(ax + by)$, $X = x$):

Minimize Y

restricted to $a_i X + b_i Y + c_i \leq 0$, with $i = 1, \dots, n$.

Algorithm 1

Compute the feasible region R in $O(n \log n)$ time. Any linear function on R can then be optimized in $O(\log n)$ time by binary search.

SOLVING LINEAR PROGRAMS

Linear program (2D)

Optimize a linear function $ax + by$

subject to n linear restrictions $a_i x + b_i y + c_i \leq 0$, where $i = 1, \dots, n$.

or, equivalently (apply the linear transform $Y = \pm(ax + by)$, $X = x$):

Minimize Y

restricted to $a_i X + b_i Y + c_i \leq 0$, with $i = 1, \dots, n$.

Algorithm 1

Compute the feasible region R in $O(n \log n)$ time. Any linear function on R can then be optimized in $O(\log n)$ time by binary search.

Can we do it faster?

SOLVING LINEAR PROGRAMS

Linear program (2D)

Optimize a linear function $ax + by$

subject to n linear restrictions $a_i x + b_i y + c_i \leq 0$, where $i = 1, \dots, n$.

or, equivalently (apply the linear transform $Y = \pm(ax + by)$, $X = x$):

Minimize Y

restricted to $a_i X + b_i Y + c_i \leq 0$, with $i = 1, \dots, n$.

Algorithm 1

Compute the feasible region R in $O(n \log n)$ time. Any linear function on R can then be optimized in $O(\log n)$ time by binary search.

Yes, we can! Algorithm 2 (Megiddo, Dyer)

In $O(n)$ time, it is possible to find the vertex of R achieving the optimum, without computing the entire feasible region R .

This is done by a **prune and search strategy**, which at each step eliminates a constant fraction of the restrictions:

- either because they are redundant,
- or because they are not needed to define the solution vertex.

SOLVING LINEAR PROGRAMS

Minimize y

restricted to $a_i x + b_i y + c_i \leq 0$, with $i = 1, \dots, n$.

SOLVING LINEAR PROGRAMS

Minimize y

restricted to $a_i x + b_i y + c_i \leq 0$, with $i = 1, \dots, n$.



SOLVING LINEAR PROGRAMS

Minimize y

restricted to $a_i x + b_i y + c_i \leq 0$, with $i = 1, \dots, n$.

Partition $I = \{1, \dots, n\}$ into three sets:



SOLVING LINEAR PROGRAMS

Minimize y

restricted to $a_i x + b_i y + c_i \leq 0$, with $i = 1, \dots, n$.

Partition $I = \{1, \dots, n\}$ into three sets:

$I_0 = \{i \mid b_i = 0\}$, corresponds to vertical restrictions $a_i x + c_i \leq 0$.

Let's define $u_1 = \max\{-c_i/a_i \mid i \in I_0, a_i \leq 0\}$ and $u_2 = \min\{-c_i/a_i \mid i \in I_0, a_i \geq 0\}$.



SOLVING LINEAR PROGRAMS

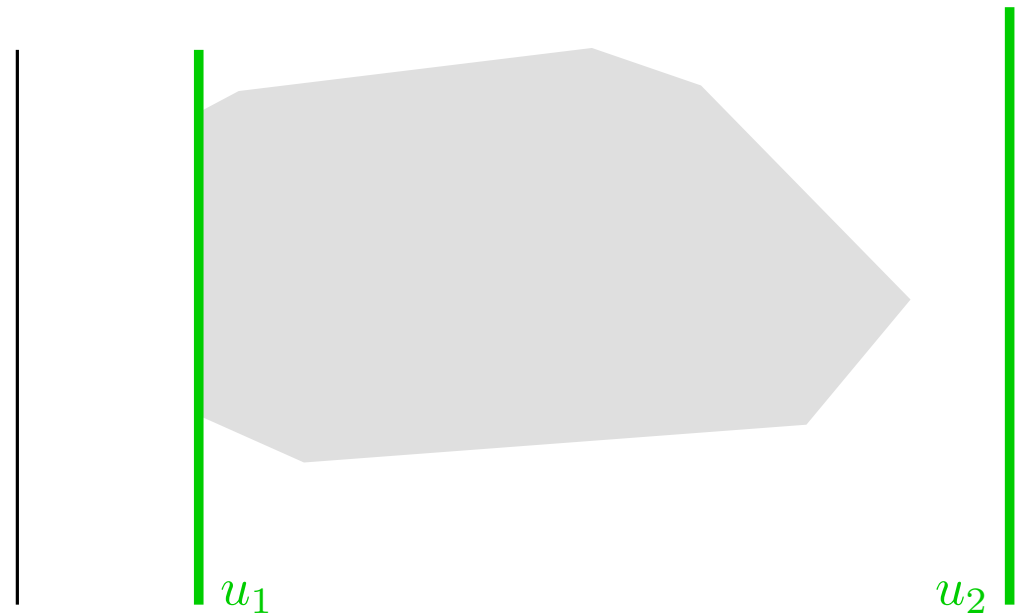
Minimize y

restricted to $a_i x + b_i y + c_i \leq 0$, with $i = 1, \dots, n$.

Partition $I = \{1, \dots, n\}$ into three sets:

$I_0 = \{i \mid b_i = 0\}$, corresponds to vertical restrictions $a_i x + c_i \leq 0$.

Let's define $u_1 = \max\{-c_i/a_i \mid i \in I_0, a_i \leq 0\}$ and $u_2 = \min\{-c_i/a_i \mid i \in I_0, a_i \geq 0\}$.



SOLVING LINEAR PROGRAMS

Minimize y

restricted to $a_i x + b_i y + c_i \leq 0$, with $i = 1, \dots, n$.

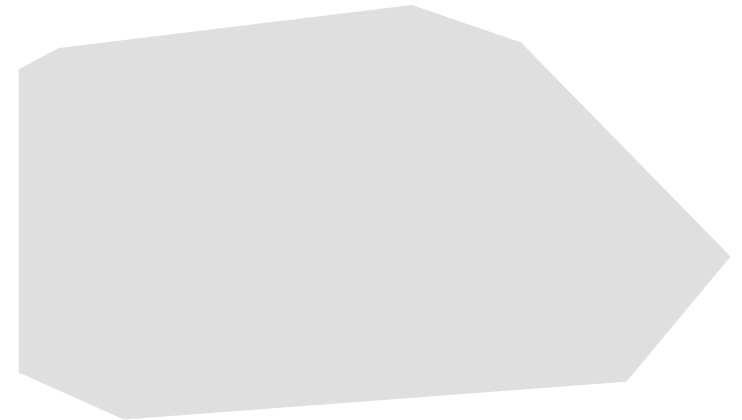
Partition $I = \{1, \dots, n\}$ into three sets:

$I_0 = \{i \mid b_i = 0\}$, corresponds to vertical restrictions $a_i x + c_i \leq 0$.

Let's define $u_1 = \max\{-c_i/a_i \mid i \in I_0, a_i \leq 0\}$ and $u_2 = \min\{-c_i/a_i \mid i \in I_0, a_i \geq 0\}$.

$I_+ = \{i \mid b_i > 0\}$, corresponds to restrictions of the form $y \leq d_i x + e_i$.

Let's define $F_+(x) = \min\{d_i x + e_i \mid i \in I_+\}$.



SOLVING LINEAR PROGRAMS

Minimize y

restricted to $a_i x + b_i y + c_i \leq 0$, with $i = 1, \dots, n$.

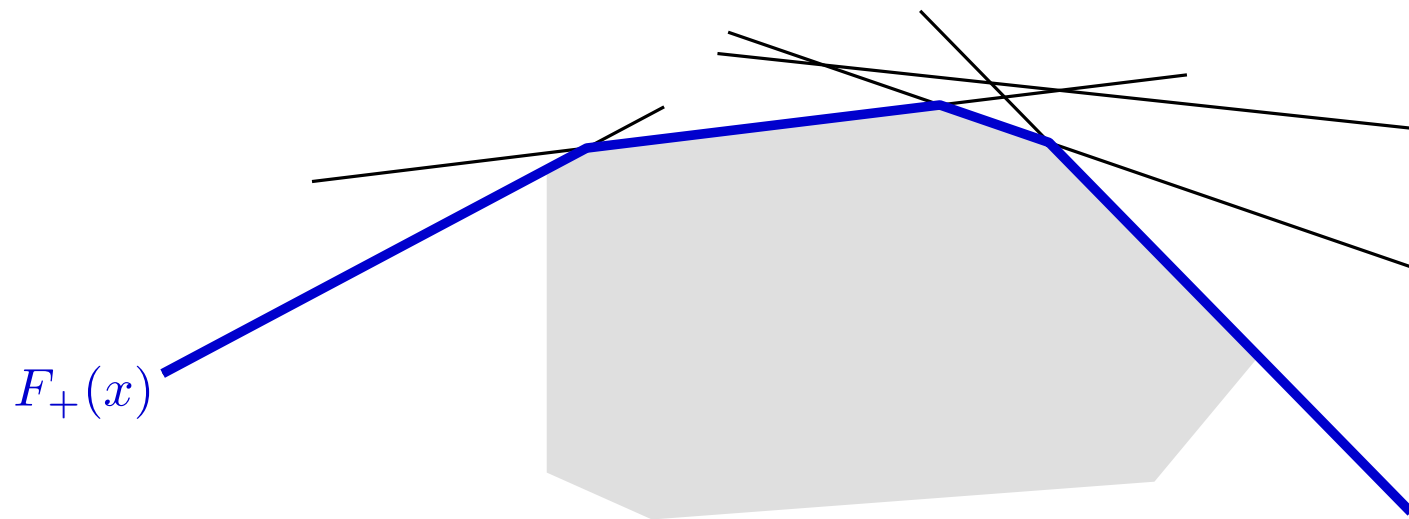
Partition $I = \{1, \dots, n\}$ into three sets:

$I_0 = \{i \mid b_i = 0\}$, corresponds to vertical restrictions $a_i x + c_i \leq 0$.

Let's define $u_1 = \max\{-c_i/a_i \mid i \in I_0, a_i \leq 0\}$ and $u_2 = \min\{-c_i/a_i \mid i \in I_0, a_i \geq 0\}$.

$I_+ = \{i \mid b_i > 0\}$, corresponds to restrictions of the form $y \leq d_i x + e_i$.

Let's define $F_+(x) = \min\{d_i x + e_i \mid i \in I_+\}$.



SOLVING LINEAR PROGRAMS

Minimize y

restricted to $a_i x + b_i y + c_i \leq 0$, with $i = 1, \dots, n$.

Partition $I = \{1, \dots, n\}$ into three sets:

$I_0 = \{i \mid b_i = 0\}$, corresponds to vertical restrictions $a_i x + c_i \leq 0$.

Let's define $u_1 = \max\{-c_i/a_i \mid i \in I_0, a_i \leq 0\}$ and $u_2 = \min\{-c_i/a_i \mid i \in I_0, a_i \geq 0\}$.

$I_+ = \{i \mid b_i > 0\}$, corresponds to restrictions of the form $y \leq d_i x + e_i$.

Let's define $F_+(x) = \min\{d_i x + e_i \mid i \in I_+\}$.

$I_- = \{i \mid b_i < 0\}$, corresponds to restrictions of the form $y \geq d_i x + e_i$.

Let's define $F_-(x) = \max\{d_i x + e_i \mid i \in I_-\}$.



SOLVING LINEAR PROGRAMS

Minimize y

restricted to $a_i x + b_i y + c_i \leq 0$, with $i = 1, \dots, n$.

Partition $I = \{1, \dots, n\}$ into three sets:

$I_0 = \{i \mid b_i = 0\}$, corresponds to vertical restrictions $a_i x + c_i \leq 0$.

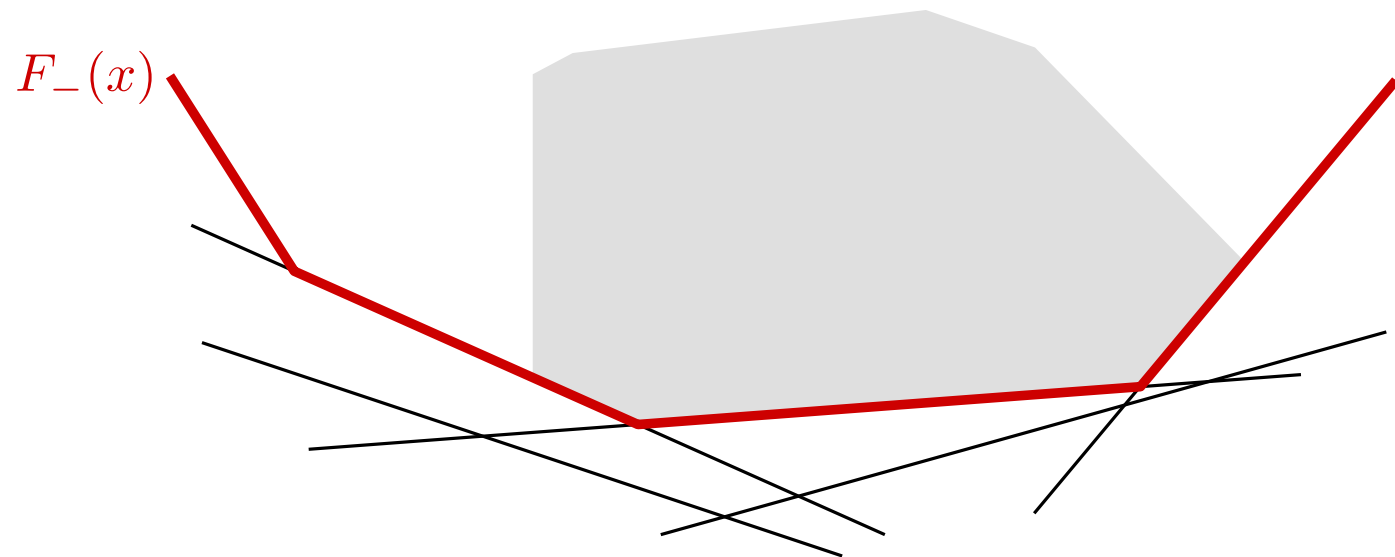
Let's define $u_1 = \max\{-c_i/a_i \mid i \in I_0, a_i \leq 0\}$ and $u_2 = \min\{-c_i/a_i \mid i \in I_0, a_i \geq 0\}$.

$I_+ = \{i \mid b_i > 0\}$, corresponds to restrictions of the form $y \leq d_i x + e_i$.

Let's define $F_+(x) = \min\{d_i x + e_i \mid i \in I_+\}$.

$I_- = \{i \mid b_i < 0\}$, corresponds to restrictions of the form $y \geq d_i x + e_i$.

Let's define $F_-(x) = \max\{d_i x + e_i \mid i \in I_-\}$.



SOLVING LINEAR PROGRAMS

Minimize y

restricted to $a_i x + b_i y + c_i \leq 0$, with $i = 1, \dots, n$.

Partition $I = \{1, \dots, n\}$ into three sets:

$I_0 = \{i \mid b_i = 0\}$, corresponds to vertical restrictions $a_i x + c_i \leq 0$.

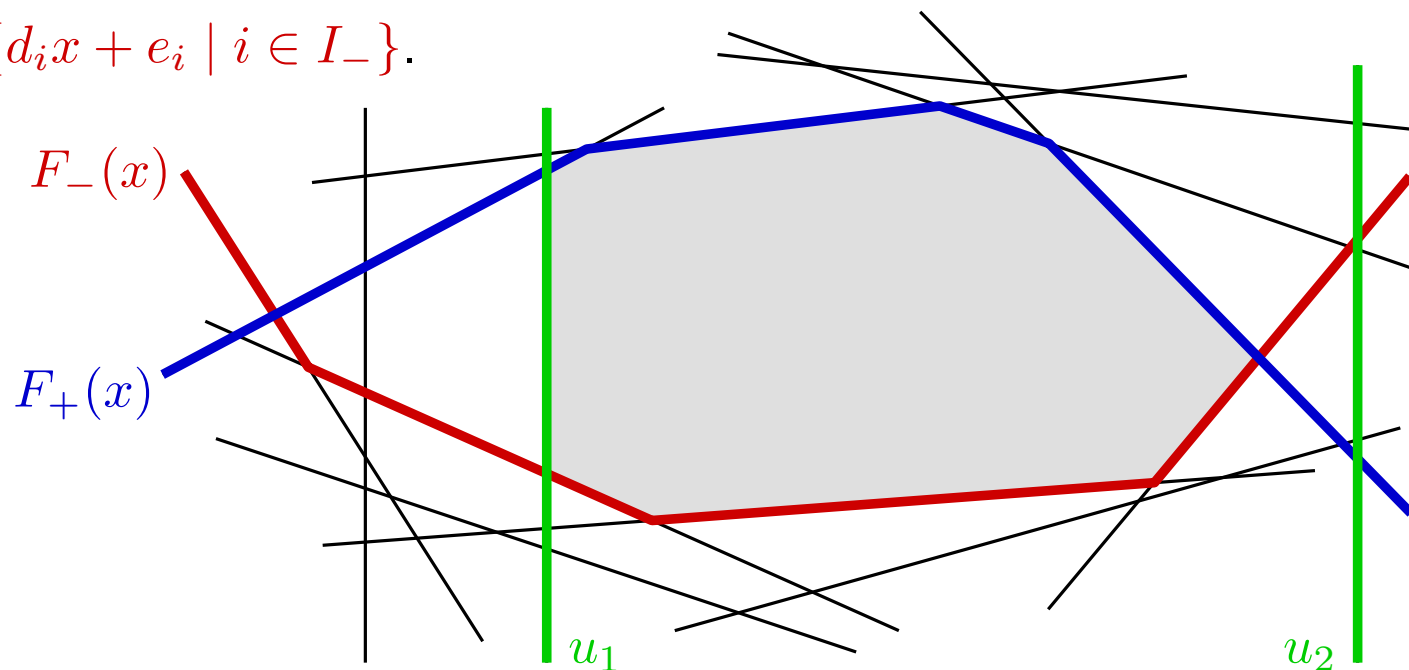
Let's define $u_1 = \max\{-c_i/a_i \mid i \in I_0, a_i \leq 0\}$ and $u_2 = \min\{-c_i/a_i \mid i \in I_0, a_i \geq 0\}$.

$I_+ = \{i \mid b_i > 0\}$, corresponds to restrictions of the form $y \leq d_i x + e_i$.

Let's define $F_+(x) = \min\{d_i x + e_i \mid i \in I_+\}$.

$I_- = \{i \mid b_i < 0\}$, corresponds to restrictions of the form $y \geq d_i x + e_i$.

Let's define $F_-(x) = \max\{d_i x + e_i \mid i \in I_-\}$.



SOLVING LINEAR PROGRAMS

Minimize y

restricted to $a_i x + b_i y + c_i \leq 0$, with $i = 1, \dots, n$.

Partition $I = \{1, \dots, n\}$ into three sets:

$I_0 = \{i \mid b_i = 0\}$, corresponds to vertical restrictions $a_i x + c_i \leq 0$.

Let's define $u_1 = \max\{-c_i/a_i \mid i \in I_0, a_i \leq 0\}$ and $u_2 = \min\{-c_i/a_i \mid i \in I_0, a_i \geq 0\}$.

$I_+ = \{i \mid b_i > 0\}$, corresponds to restrictions of the form $y \leq d_i x + e_i$.

Let's define $F_+(x) = \min\{d_i x + e_i \mid i \in I_+\}$.

$I_- = \{i \mid b_i < 0\}$, corresponds to restrictions of the form $y \geq d_i x + e_i$.

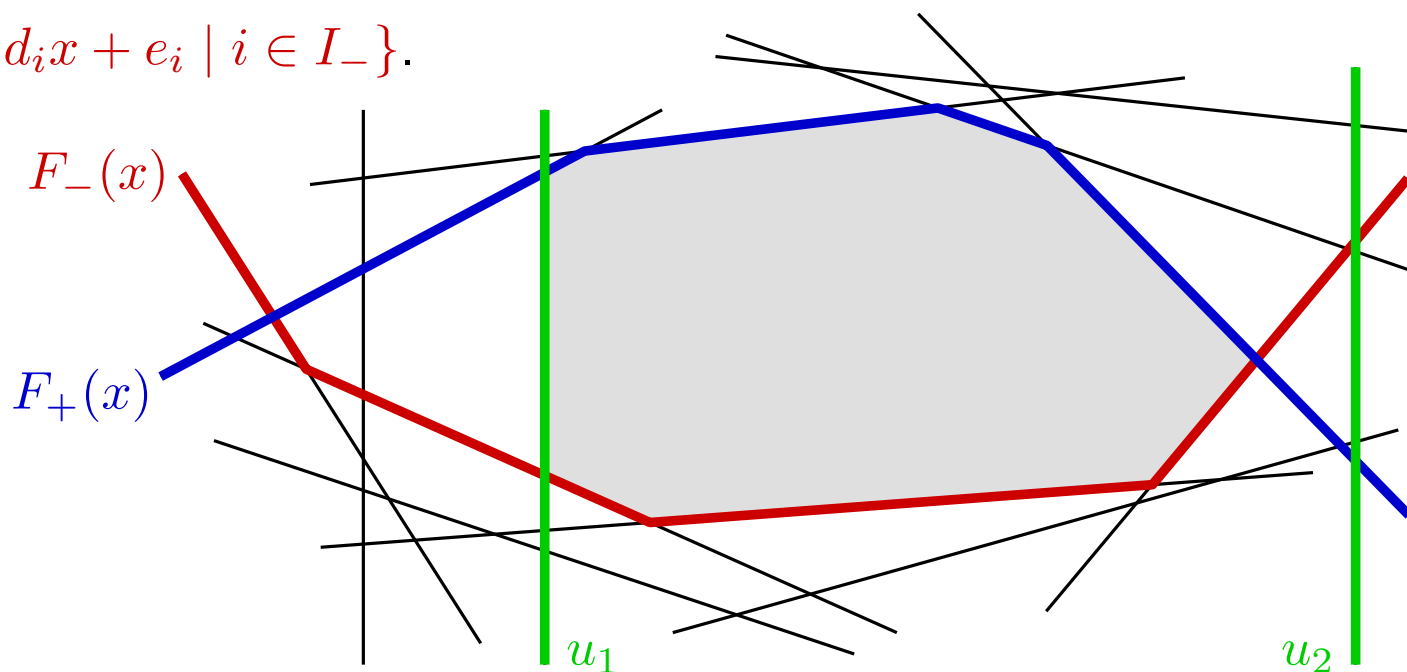
Let's define $F_-(x) = \max\{d_i x + e_i \mid i \in I_-\}$.

Minimize $F_-(x)$

restricted to

$$u_1 \leq x \leq u_2$$

$$F_-(x) \leq F_+(x)$$



SOLVING LINEAR PROGRAMS

Minimize y

restricted to $a_i x + b_i y + c_i \leq 0$, with $i = 1, \dots, n$.

Partition $I = \{1, \dots, n\}$ into three sets:

$I_0 = \{i \mid b_i = 0\}$, corresponds to vertical restrictions $a_i x + c_i \leq 0$.

Let's define $u_1 = \max\{-c_i/a_i \mid i \in I_0, a_i \leq 0\}$ and $u_2 = \min\{-c_i/a_i \mid i \in I_0, a_i \geq 0\}$.

$I_+ = \{i \mid b_i > 0\}$, corresponds to restrictions of the form $y \leq d_i x + e_i$.

Let's define $F_+(x) = \min\{d_i x + e_i \mid i \in I_+\}$.

$I_- = \{i \mid b_i < 0\}$, corresponds to restrictions of the form $y \geq d_i x + e_i$.

Let's define $F_-(x) = \max\{d_i x + e_i \mid i \in I_-\}$.

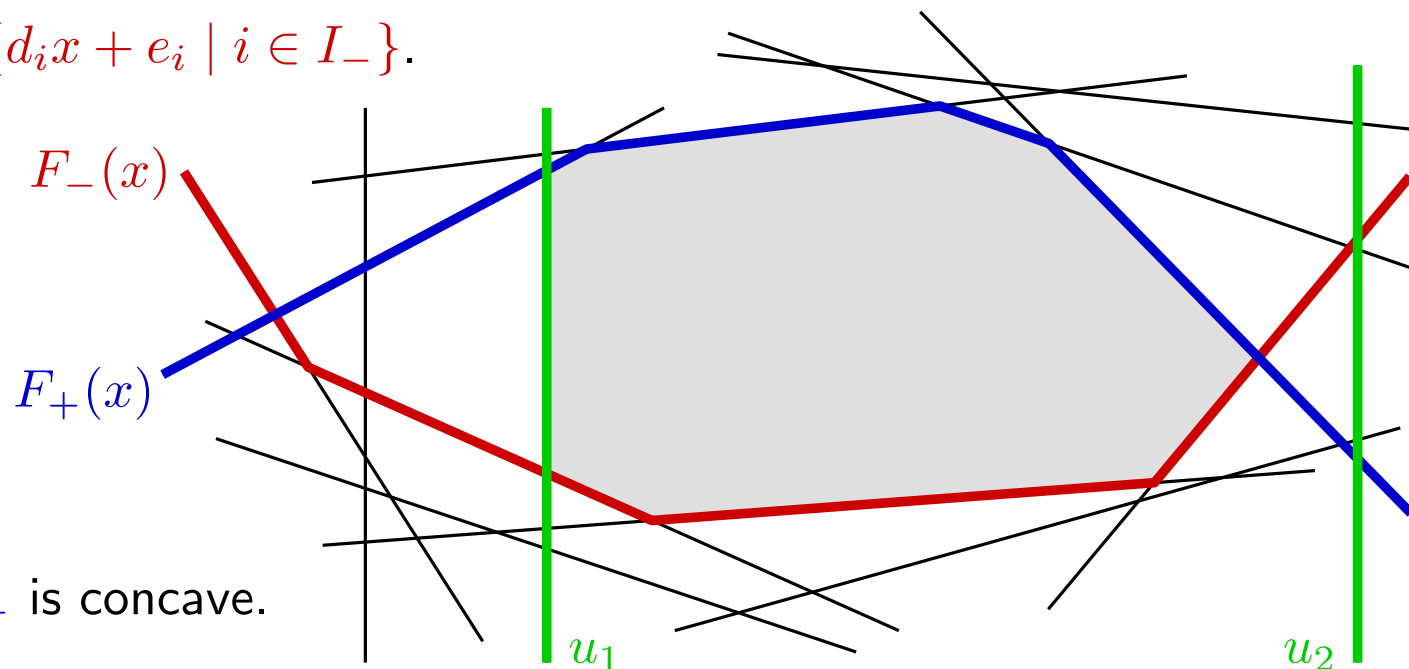
Minimize $F_-(x)$

restricted to

$$u_1 \leq x \leq u_2$$

$$F_-(x) \leq F_+(x)$$

Notice that F_- is convex and F_+ is concave.



SOLVING LINEAR PROGRAMS

The search (i.e., solving the decision problem)

SOLVING LINEAR PROGRAMS

The search (i.e., solving the decision problem)

Given $x' \in [u_1, u_2]$, in $O(n)$ time it is possible to arrive at one of the following conclusions:

1. x' is feasible, and minimizing $F_-(x)$.
2. x' is feasible, and the solution of the problem lies to its right (left).
3. x' is infeasible, and the solution of the problem does not lie to its left (right).
4. x' is infeasible, and the problem has no solution.

SOLVING LINEAR PROGRAMS

The search (i.e., solving the decision problem)

Given $x' \in [u_1, u_2]$, in $O(n)$ time it is possible to arrive at one of the following conclusions:

1. x' is feasible, and minimizing $F_-(x)$.
2. x' is feasible, and the solution of the problem lies to its right (left).
3. x' is infeasible, and the solution of the problem does not lie to its left (right).
4. x' is infeasible, and the problem has no solution.

This is done by analyzing the values of F_+ , F_- and their slopes f_+^l , f_+^r , f_-^l , f_-^r at x' , where the slopes are defined as follows:

- If $F_-(x')$ (resp. $F_+(x')$) is defined by a unique index $i \in I_-$ (resp. I_+), then $f_-^l(x') = f_-^r(x') = d_i$ (resp. $f_+^l(x') = f_+^r(x') = d_i$).
- If $F_-(x')$ (resp. $F_+(x')$) is defined by two indexes $i, j \in I_-$ (resp. I_+), then $f_-^l(x') = \min(d_i, d_j)$ and $f_-^r(x') = \max(d_i, d_j)$ (resp. $f_+^l(x') = \max(d_i, d_j)$ and $f_+^r(x') = \min(d_i, d_j)$).

SOLVING LINEAR PROGRAMS

The search (i.e., solving the decision problem)

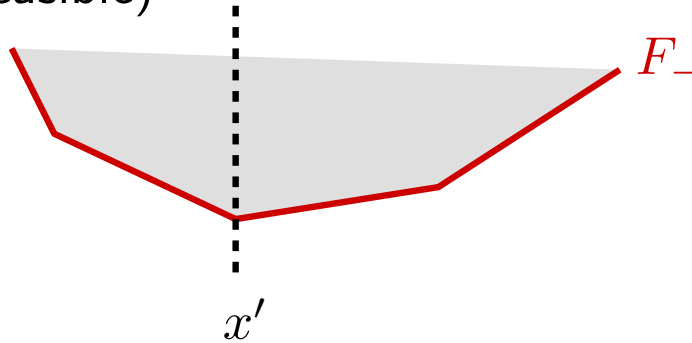
Case 1. $F_-(x') \leq F_+(x')$ (x' is feasible)

SOLVING LINEAR PROGRAMS

The search (i.e., solving the decision problem)

Case 1. $F_-(x') \leq F_+(x')$ (x' is feasible)

If $f_-^l(x') < 0 < f_-^r(x')$,
then x' is the solution.

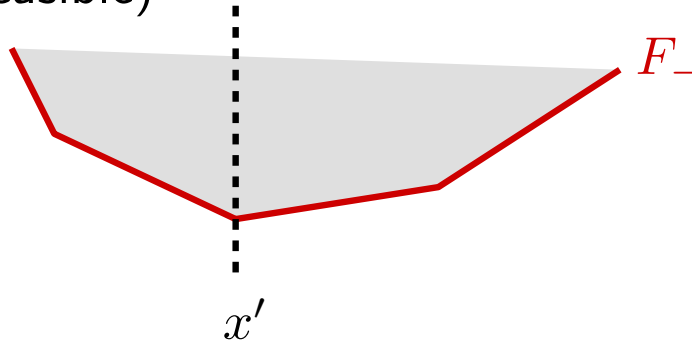


SOLVING LINEAR PROGRAMS

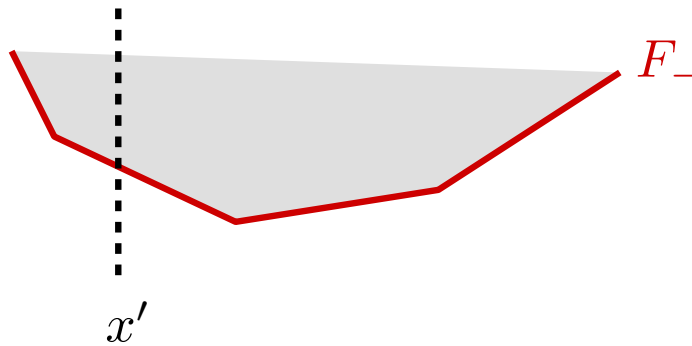
The search (i.e., solving the decision problem)

Case 1. $F_-(x') \leq F_+(x')$ (x' is feasible)

If $f_-^l(x') < 0 < f_-^r(x')$,
then x' is the solution.



If $f_-^r(x') < 0$,
then search to the right of x' .

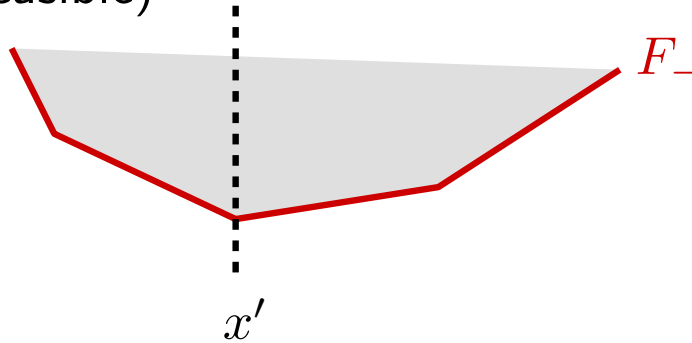


SOLVING LINEAR PROGRAMS

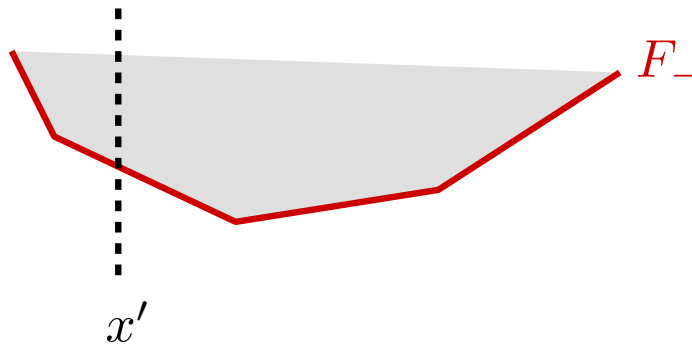
The search (i.e., solving the decision problem)

Case 1. $F_-(x') \leq F_+(x')$ (x' is feasible)

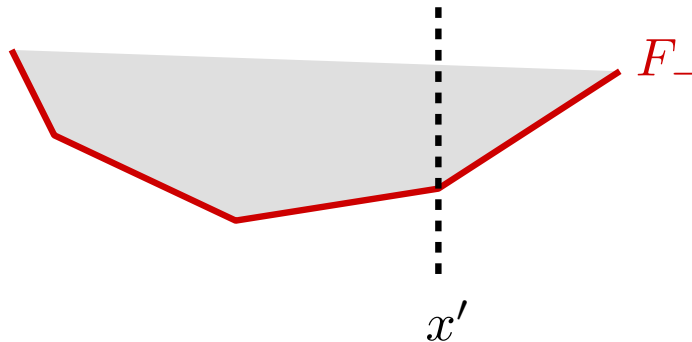
If $f_-^l(x') < 0 < f_-^r(x')$,
then x' is the solution.



If $f_-^r(x') < 0$,
then search to the right of x' .



If $f_-^l(x') > 0$,
then search to the left of x' .



SOLVING LINEAR PROGRAMS

The search (i.e., solving the decision problem)

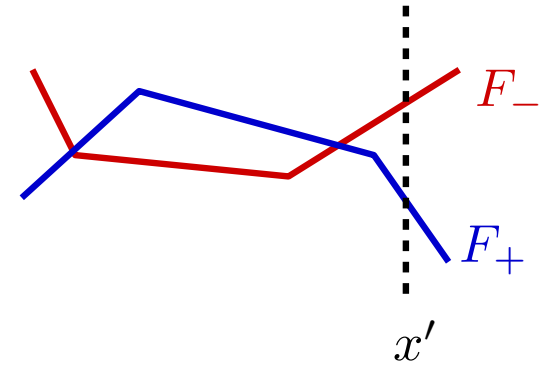
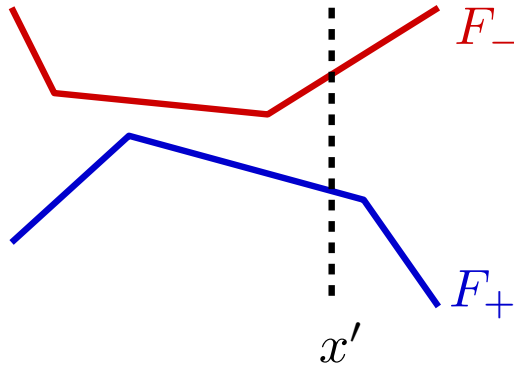
Case 2. $F_-(x') > F_+(x')$ (x' is infeasible)

SOLVING LINEAR PROGRAMS

The search (i.e., solving the decision problem)

Case 2. $F_-(x') > F_+(x')$ (x' is infeasible)

If $f_-^l(x') > f_+^l(x')$,
then search to the left of x' .

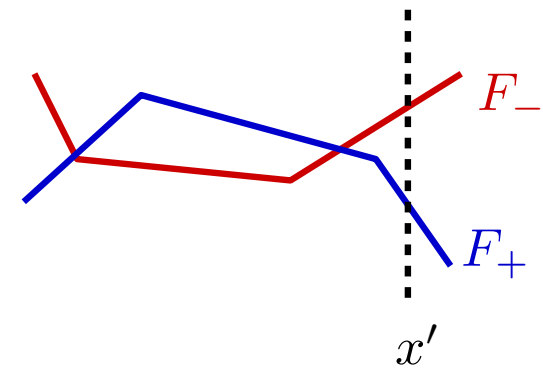
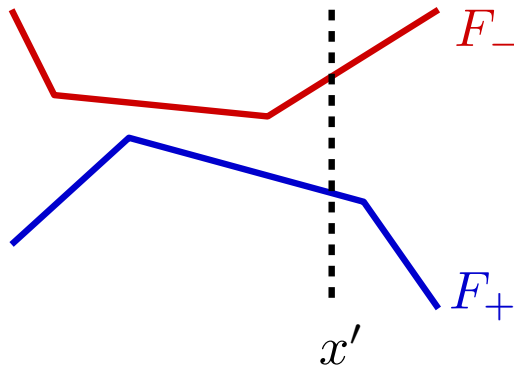


SOLVING LINEAR PROGRAMS

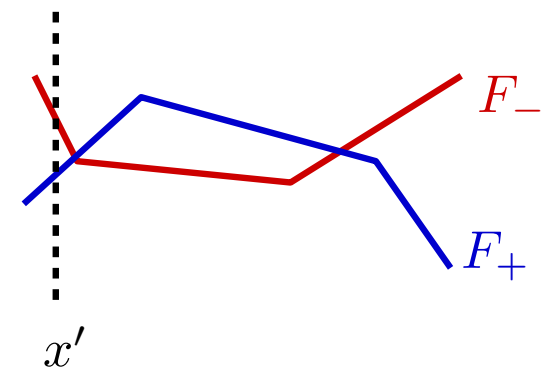
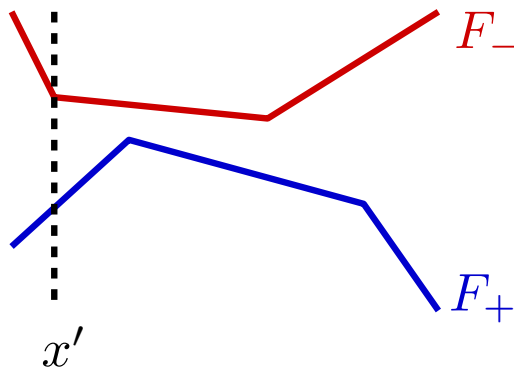
The search (i.e., solving the decision problem)

Case 2. $F_-(x') > F_+(x')$ (x' is infeasible)

If $f_-^l(x') > f_+^l(x')$,
then search to the left of x' .



If $f_-^r(x') < f_+^r(x')$,
then search to the right of x' .

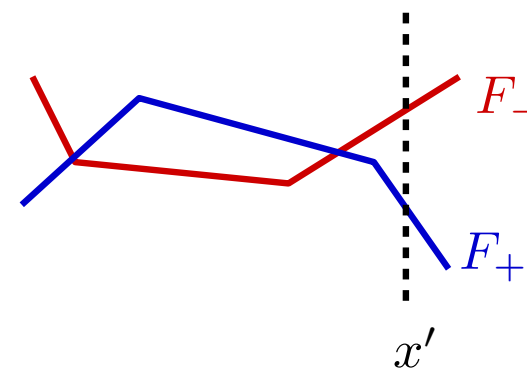
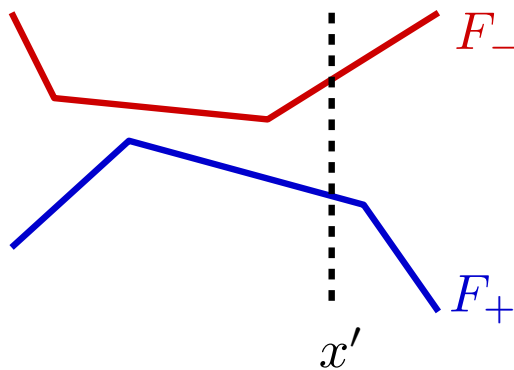


SOLVING LINEAR PROGRAMS

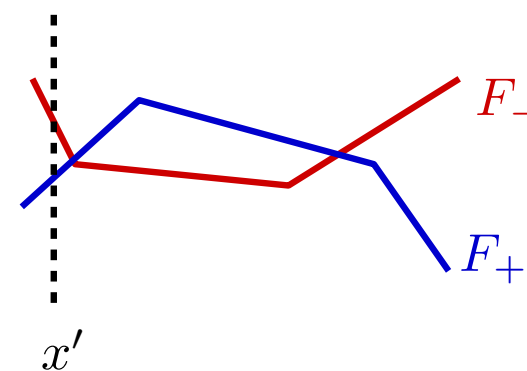
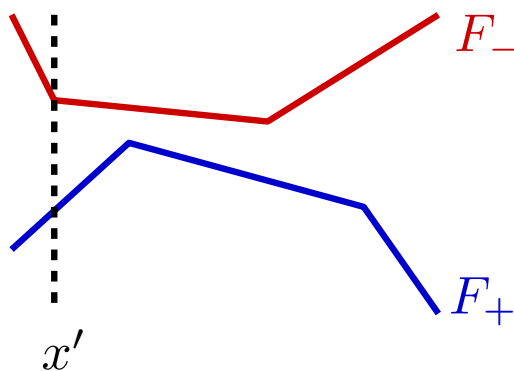
The search (i.e., solving the decision problem)

Case 2. $F_-(x') > F_+(x')$ (x' is infeasible)

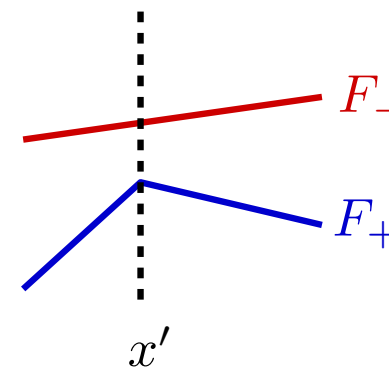
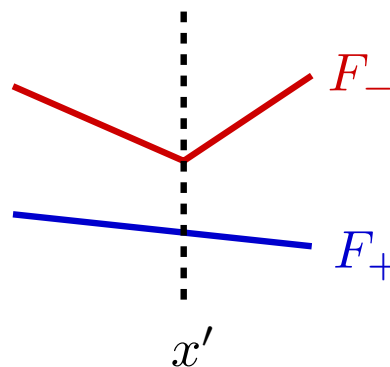
If $f_-^l(x') > f_+^l(x')$,
then search to the left of x' .



If $f_-^r(x') < f_+^r(x')$,
then search to the right of x' .



If $f_-^l(x') \leq f_+^l(x')$ and $f_-^r(x') \geq f_+^r(x')$
then there is no solution.



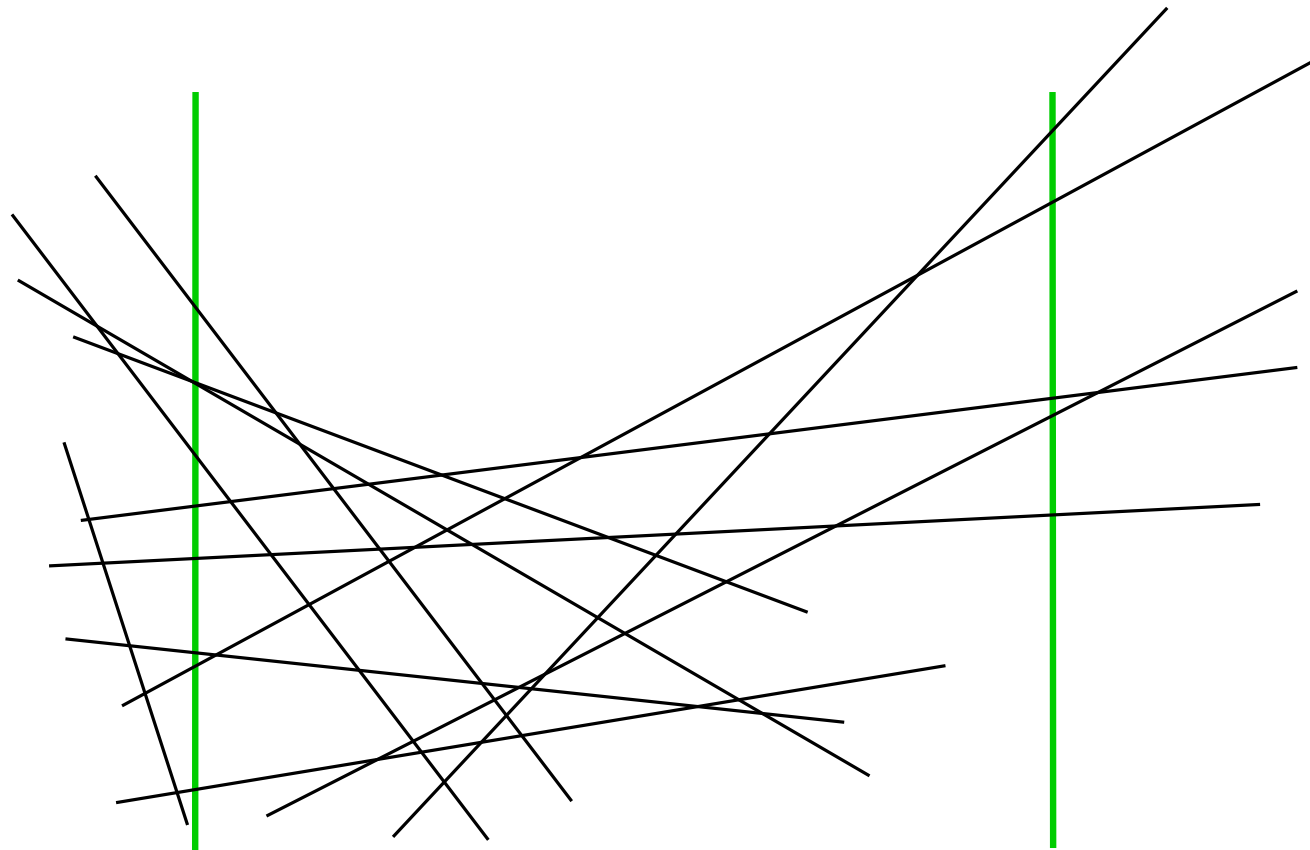
SOLVING LINEAR PROGRAMS

The algorithm

SOLVING LINEAR PROGRAMS

The algorithm

Initialization Eliminate all $i \in I_0$ except those corresponding to u_1 and u_2 , if they exist.



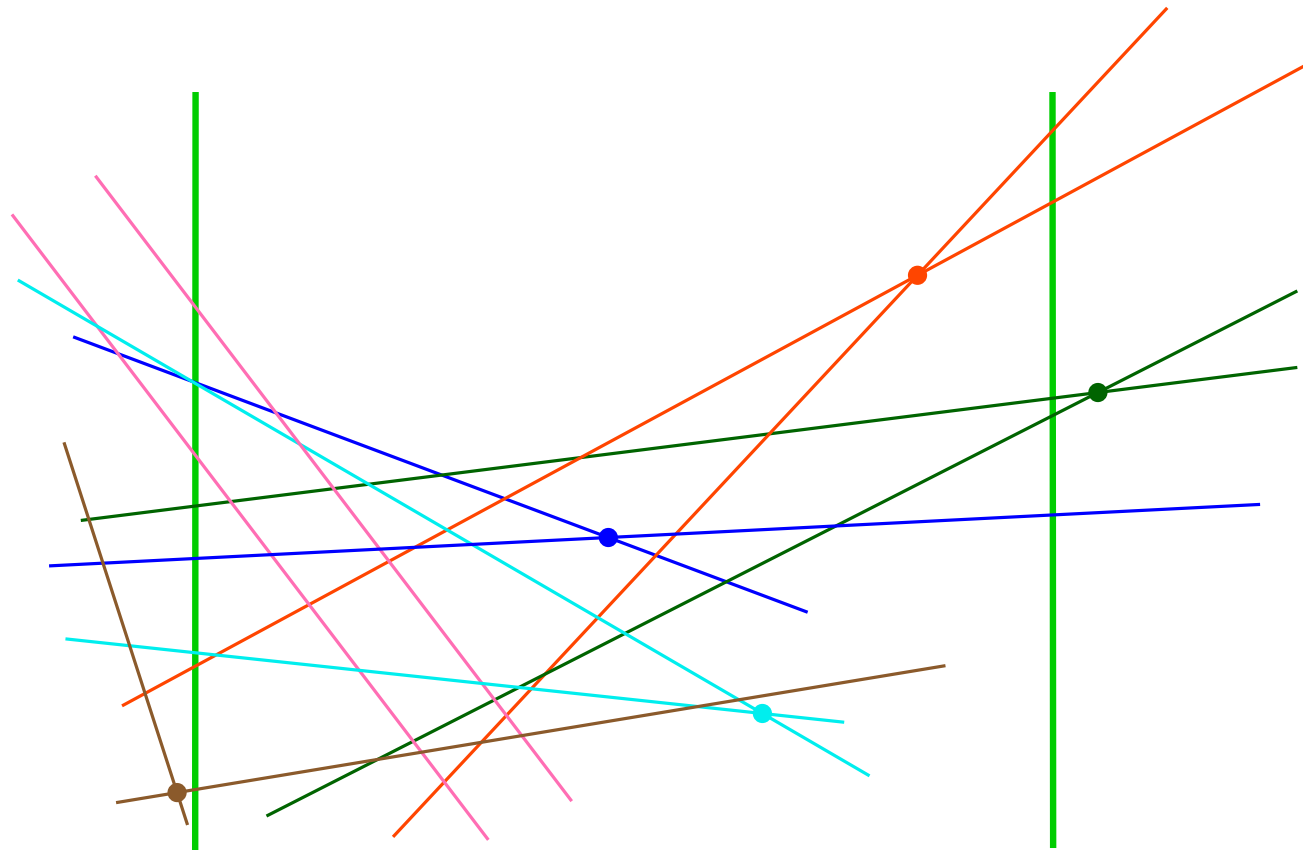
SOLVING LINEAR PROGRAMS

The algorithm

Initialization Eliminate all $i \in I_0$ except those corresponding to u_1 and u_2 , if they exist.

Advance. Repeat as many times as necessary:

1. Pair up all restrictions corresponding to I_- (possibly but one). Analogously for I_+ . Compute the abscissa of the intersection point of each pair of lines, if it exists.



SOLVING LINEAR PROGRAMS

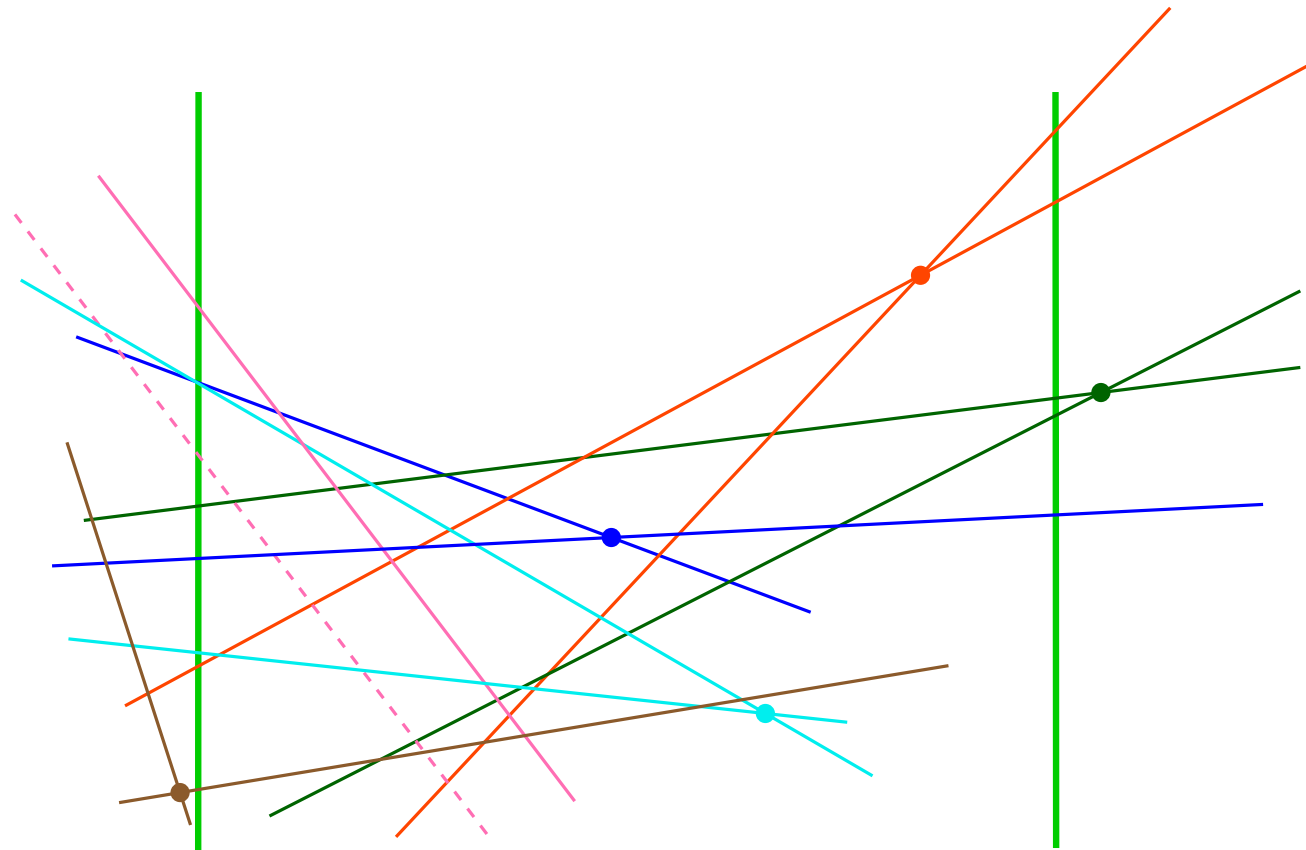
The algorithm

Initialization Eliminate all $i \in I_0$ except those corresponding to u_1 and u_2 , if they exist.

Advance. Repeat as many times as necessary: (recall inequalities look like $y \leq d_i x + e_i$)

2. For all pairs $i, j \in I_-$ (analogously for I_+), do:

- If $d_i = d_j$, eliminate the line corresponding to $\min(e_i, e_j)$ (resp. $\max(e_i, e_j)$).



SOLVING LINEAR PROGRAMS

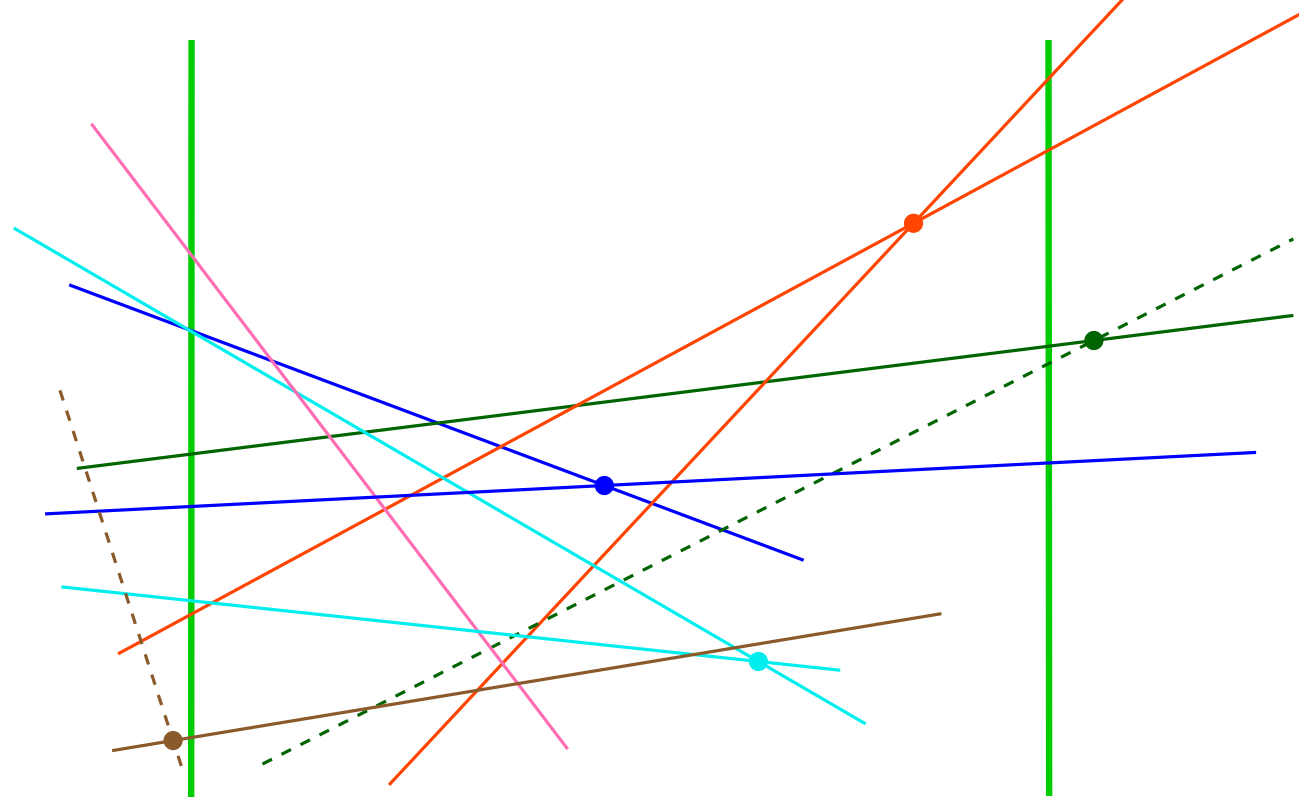
The algorithm

Initialization Eliminate all $i \in I_0$ except those corresponding to u_1 and u_2 , if they exist.

Advance. Repeat as many times as necessary:

2. For all pairs $i, j \in I_-$ (analogously for I_+), do:

- If $d_i = d_j$ (parallel lines), remove line corresp. to $\min(e_i, e_j)$ (resp. $\max(e_i, e_j)$).
- If $d_i \neq d_j$ and x_{ij} is the abscissa of the intersection point of the two lines, then do:
 - If $x_{ij} < u_1$, eliminate the line with $\min(d_i, d_j)$ (resp. $\max(d_i, d_j)$).
 - If $x_{ij} > u_2$, eliminate the line with $\max(d_i, d_j)$ (resp. $\min(d_i, d_j)$).



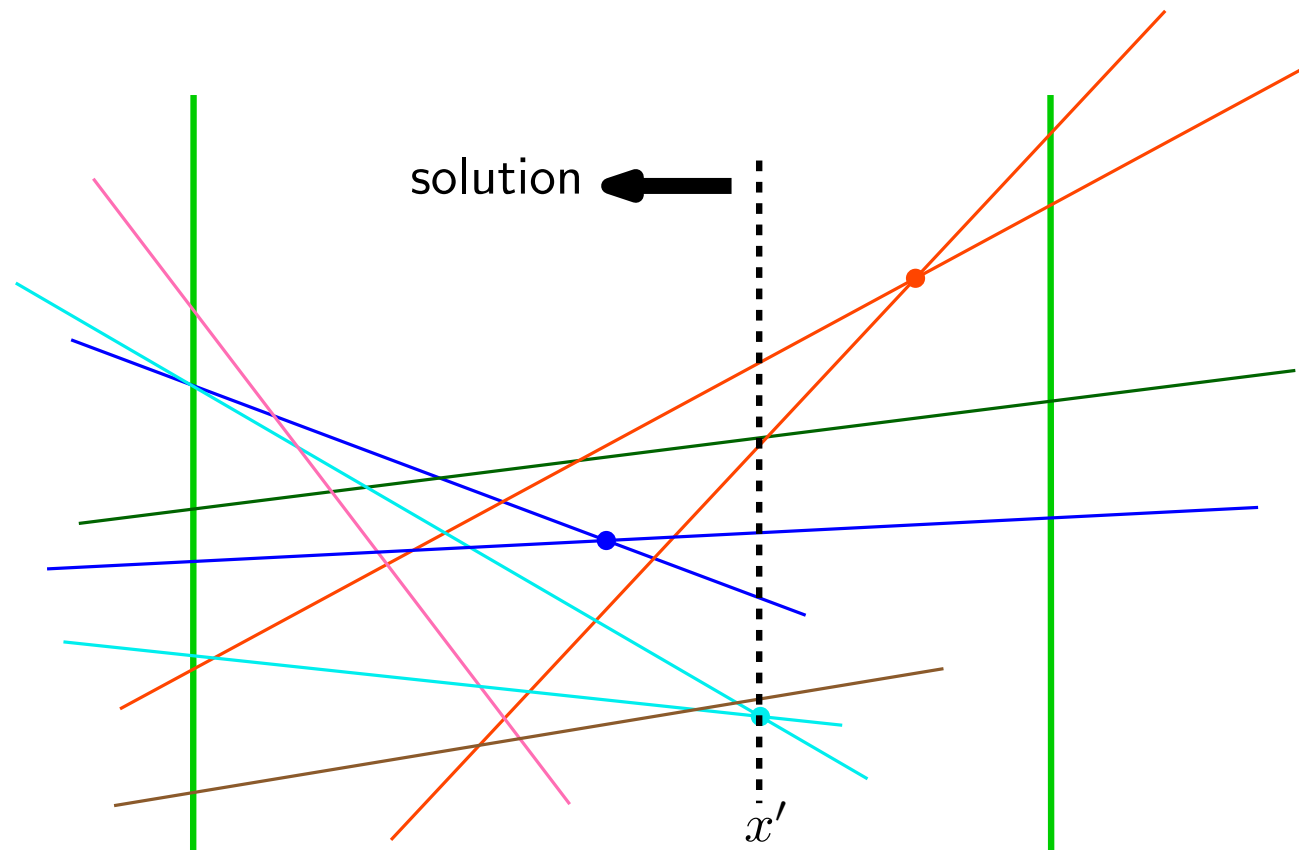
SOLVING LINEAR PROGRAMS

The algorithm

Initialization Eliminate all $i \in I_0$ except those corresponding to u_1 and u_2 , if they exist.

Advance. Repeat as many times as necessary:

3. Compute the median value x' of the surviving x_{ij} .
4. Search: Apply the search procedure to x' . If the answer is x' , return x' .



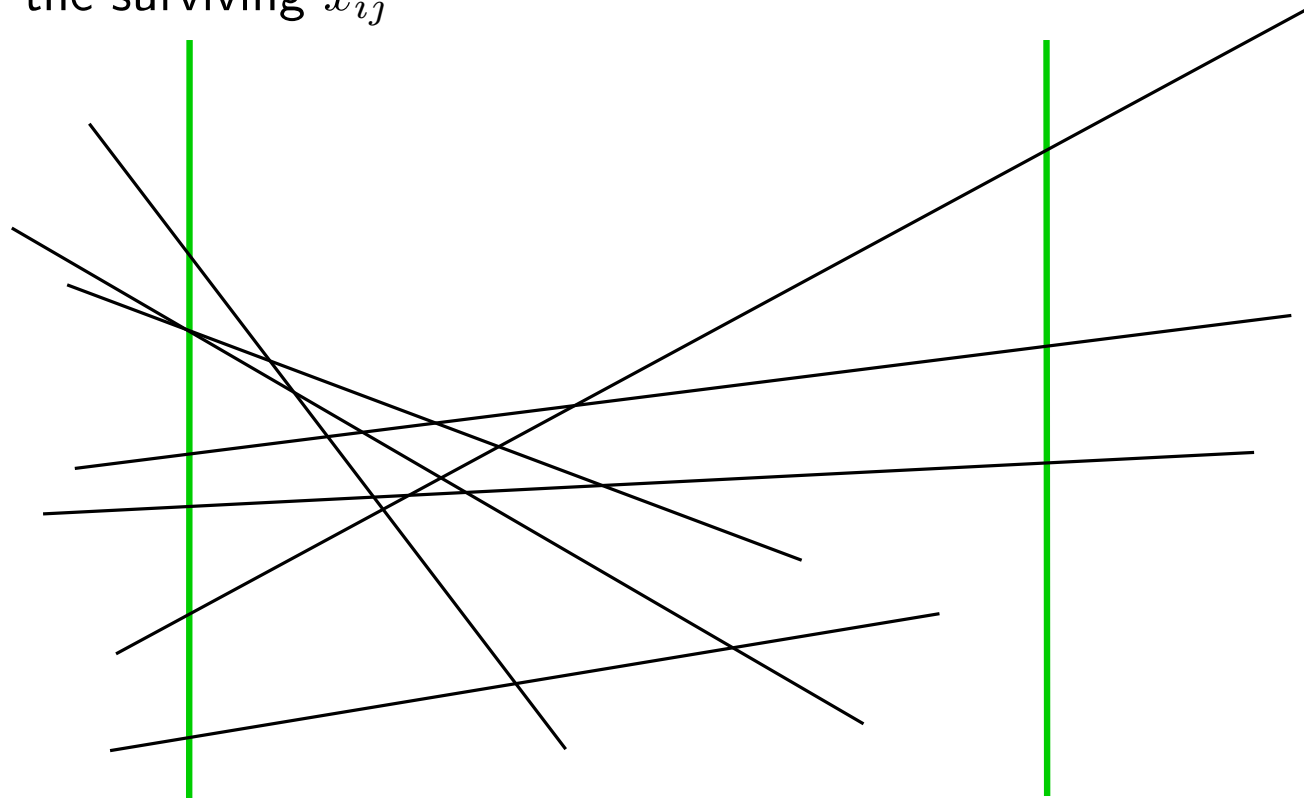
SOLVING LINEAR PROGRAMS

The algorithm

Initialization Eliminate all $i \in I_0$ except those corresponding to u_1 and u_2 , if they exist.

Advance. Repeat as many times as necessary:

1. Pair lines up
2. Prune parallel and external lines
3. Compute the median value x' of the surviving x_{ij}
4. Search
5. Prune



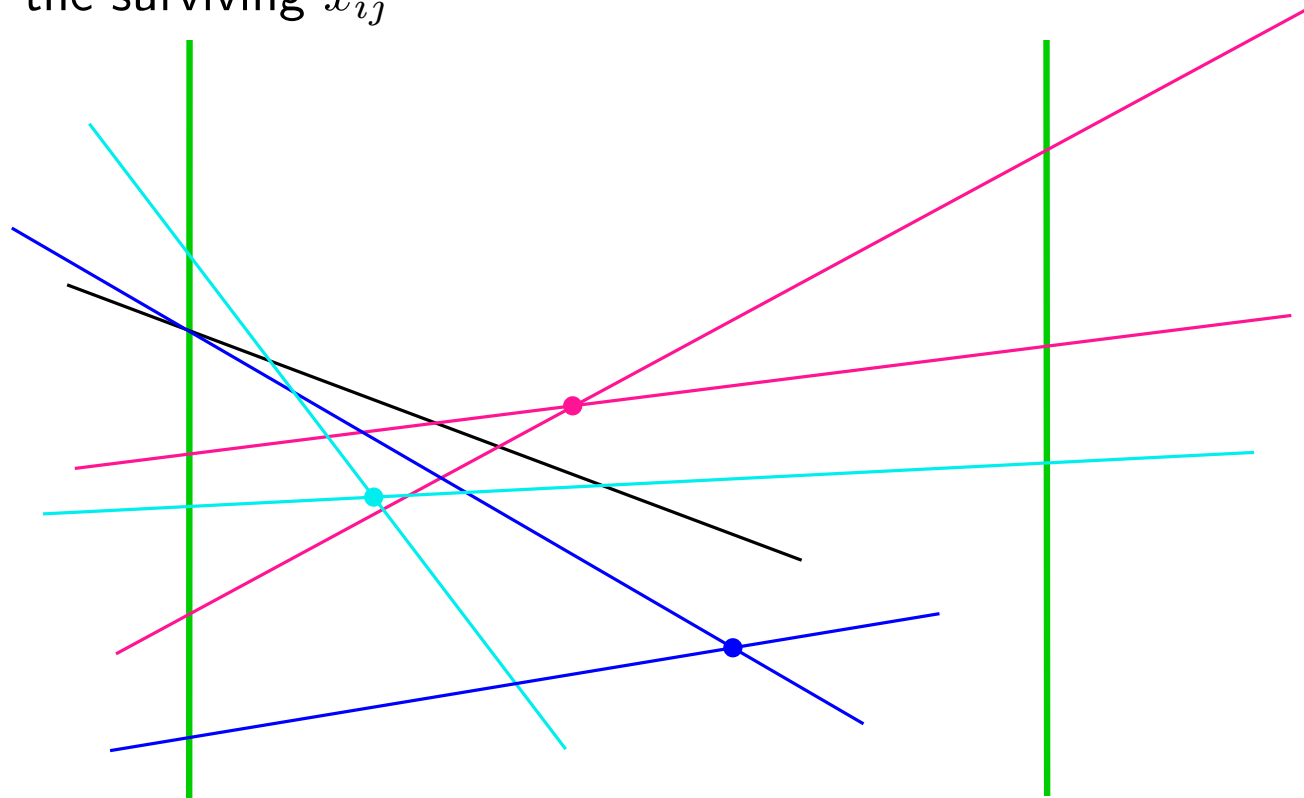
SOLVING LINEAR PROGRAMS

The algorithm

Initialization Eliminate all $i \in I_0$ except those corresponding to u_1 and u_2 , if they exist.

Advance. Repeat as many times as necessary:

1. Pair lines up
2. Prune parallel and external lines
3. Compute the median value x' of the surviving x_{ij}
4. Search
5. Prune



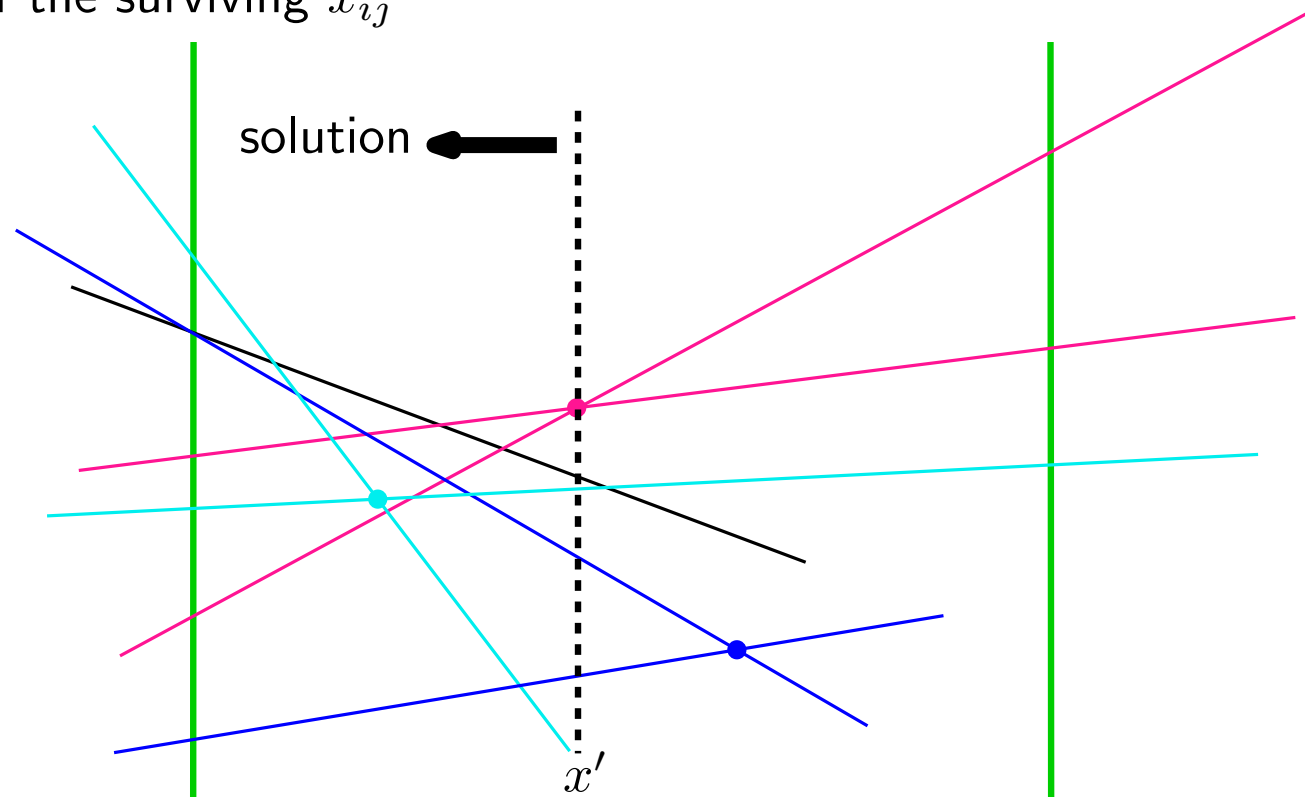
SOLVING LINEAR PROGRAMS

The algorithm

Initialization Eliminate all $i \in I_0$ except those corresponding to u_1 and u_2 , if they exist.

Advance. Repeat as many times as necessary:

1. Pair lines up
2. Prune parallel and external lines
3. Compute the median value x' of the surviving x_{ij}
4. Search
5. Prune



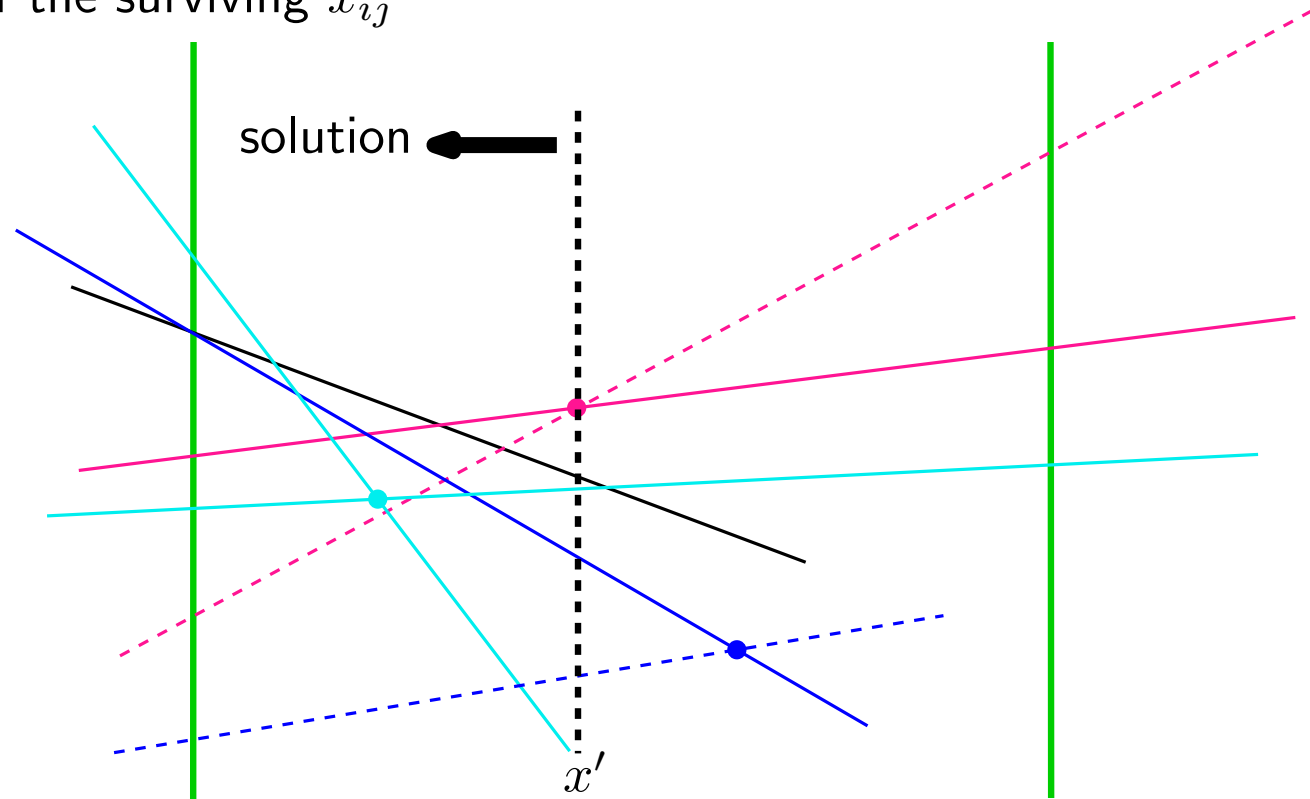
SOLVING LINEAR PROGRAMS

The algorithm

Initialization Eliminate all $i \in I_0$ except those corresponding to u_1 and u_2 , if they exist.

Advance. Repeat as many times as necessary:

1. Pair lines up
2. Prune parallel and external lines
3. Compute the median value x' of the surviving x_{ij}
4. Search
5. Prune



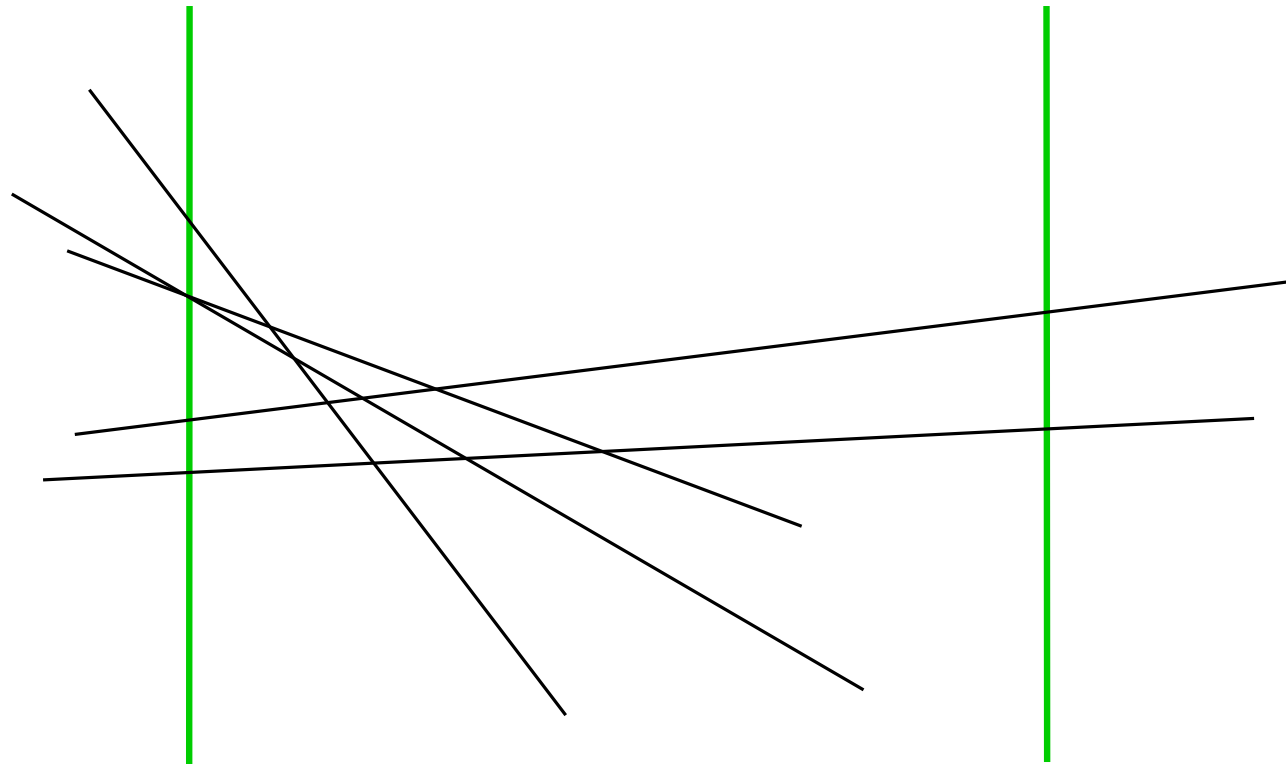
SOLVING LINEAR PROGRAMS

The algorithm

Initialization Eliminate all $i \in I_0$ except those corresponding to u_1 and u_2 , if they exist.

Advance. Repeat as many times as necessary:

1. Pair lines up
2. Prune parallel and external lines
3. Compute the median value x' of the surviving x_{ij}
4. Search
5. Prune



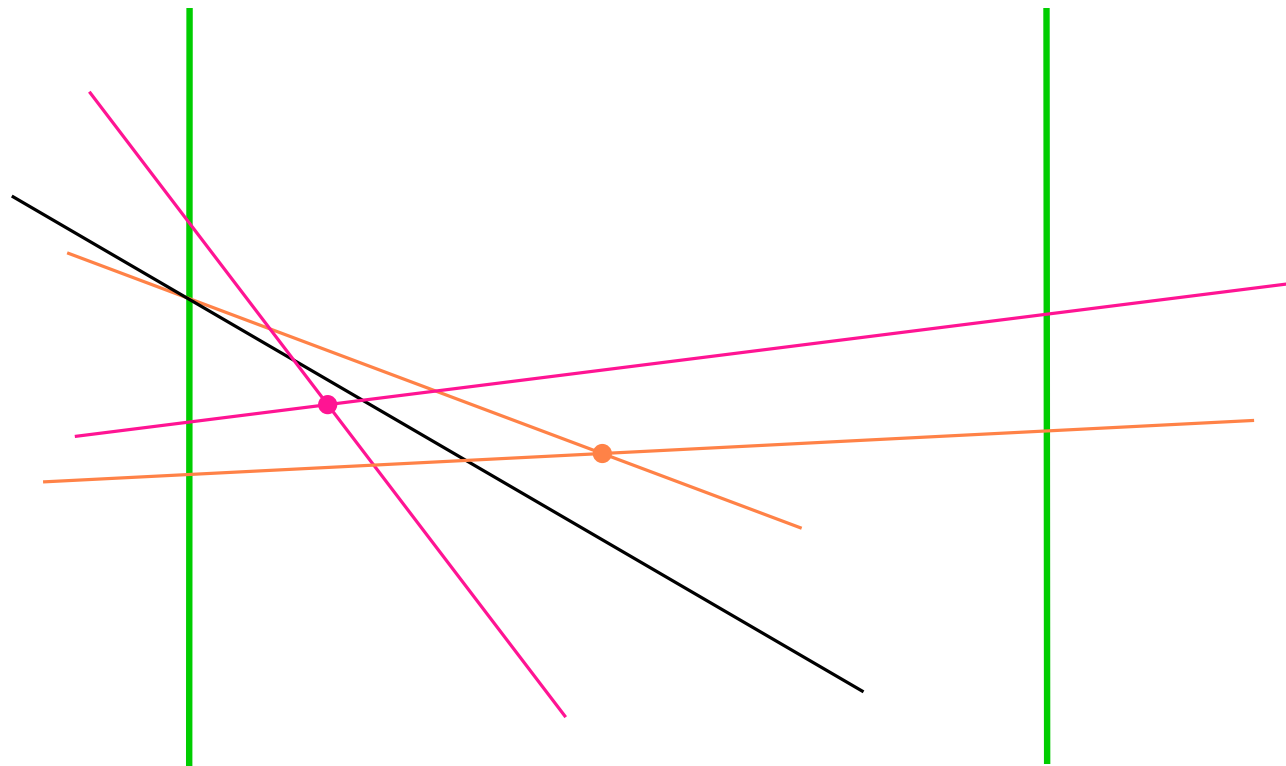
SOLVING LINEAR PROGRAMS

The algorithm

Initialization Eliminate all $i \in I_0$ except those corresponding to u_1 and u_2 , if they exist.

Advance. Repeat as many times as necessary:

1. Pair lines up
2. Prune parallel and external lines
3. Compute the median value x' of the surviving x_{ij}
4. Search
5. Prune



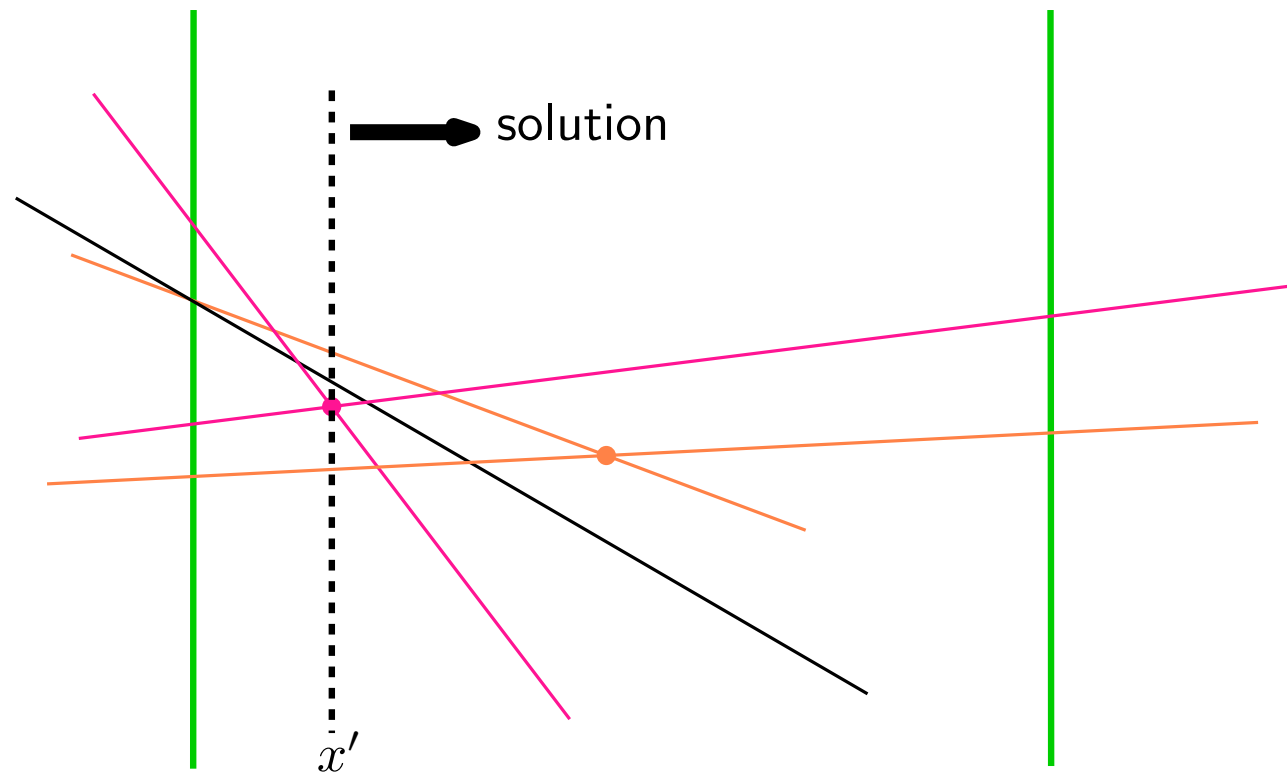
SOLVING LINEAR PROGRAMS

The algorithm

Initialization Eliminate all $i \in I_0$ except those corresponding to u_1 and u_2 , if they exist.

Advance. Repeat as many times as necessary:

1. Pair lines up
2. Prune parallel and external lines
3. Compute the median value x' of the surviving x_{ij}
4. Search
5. Prune



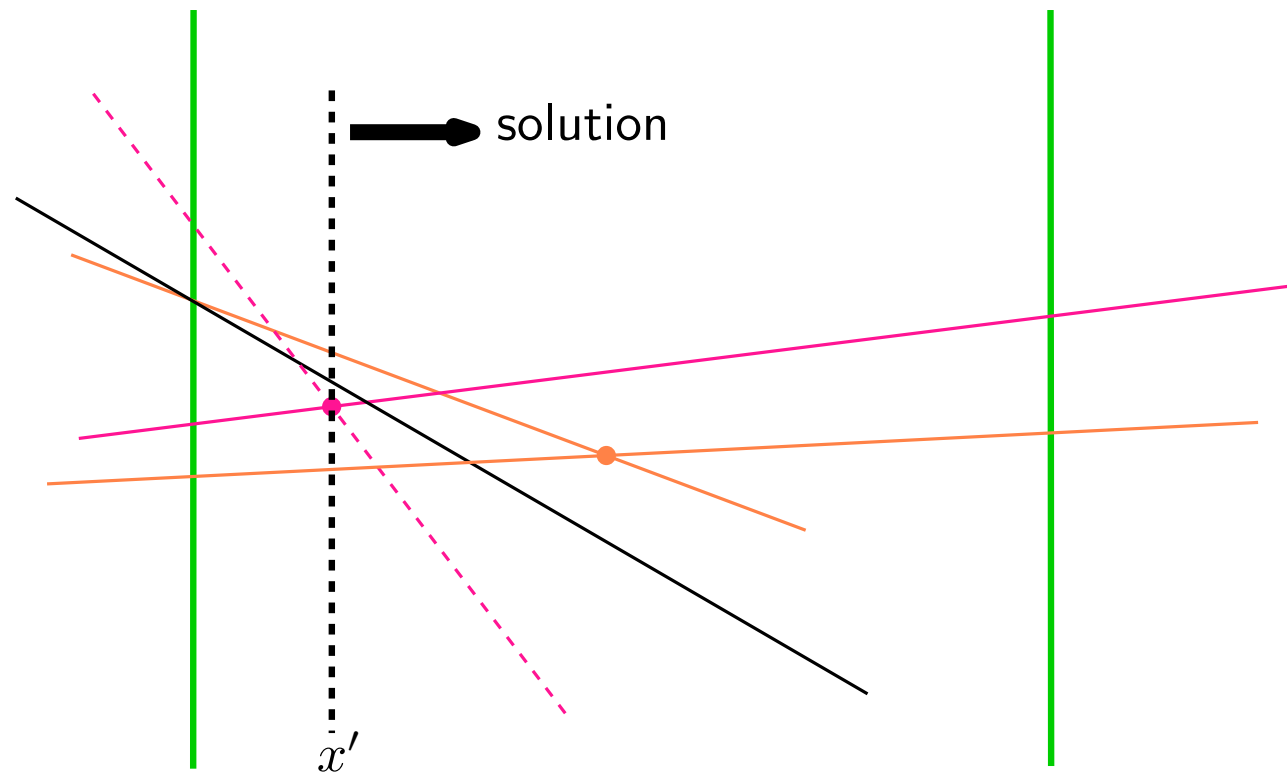
SOLVING LINEAR PROGRAMS

The algorithm

Initialization Eliminate all $i \in I_0$ except those corresponding to u_1 and u_2 , if they exist.

Advance. Repeat as many times as necessary:

1. Pair lines up
2. Prune parallel and external lines
3. Compute the median value x' of the surviving x_{ij}
4. Search
5. Prune



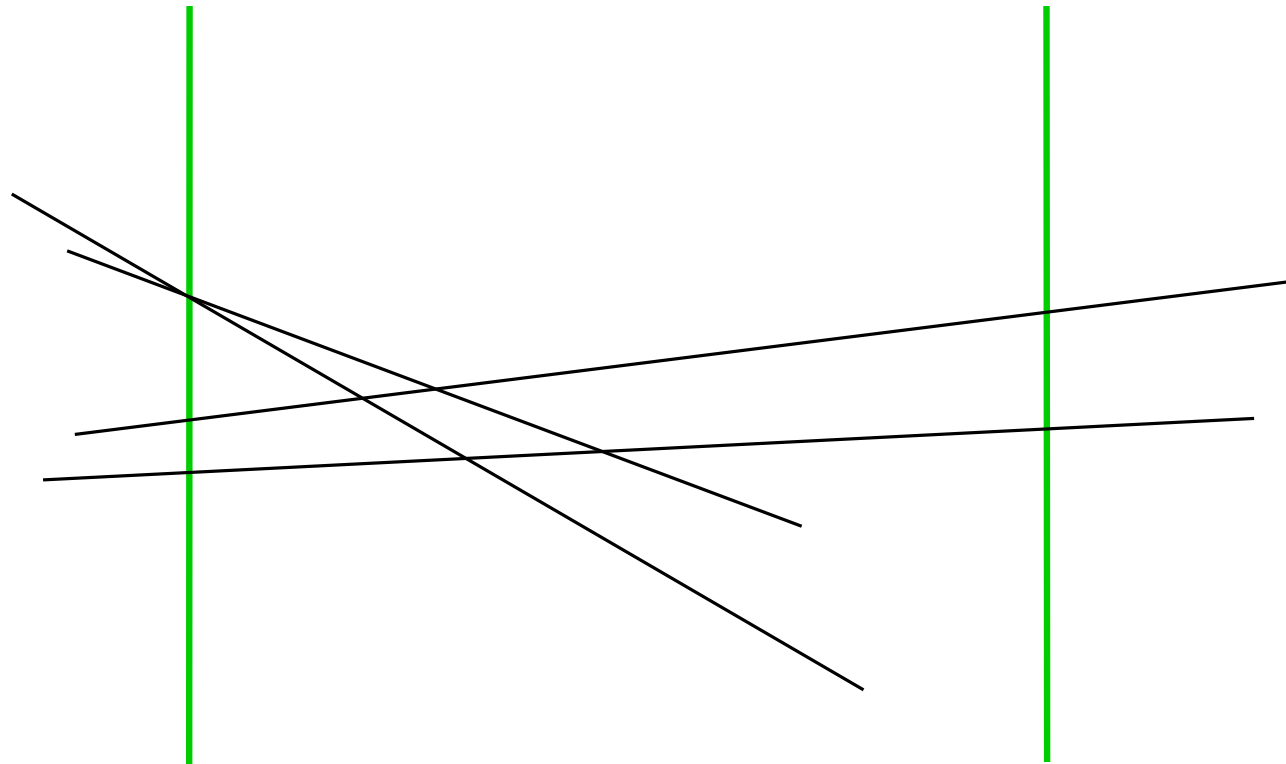
SOLVING LINEAR PROGRAMS

The algorithm

Initialization Eliminate all $i \in I_0$ except those corresponding to u_1 and u_2 , if they exist.

Advance. Repeat as many times as necessary:

1. Pair lines up
2. Prune parallel and external lines
3. Compute the median value x' of the surviving x_{ij}
4. Search
5. Prune



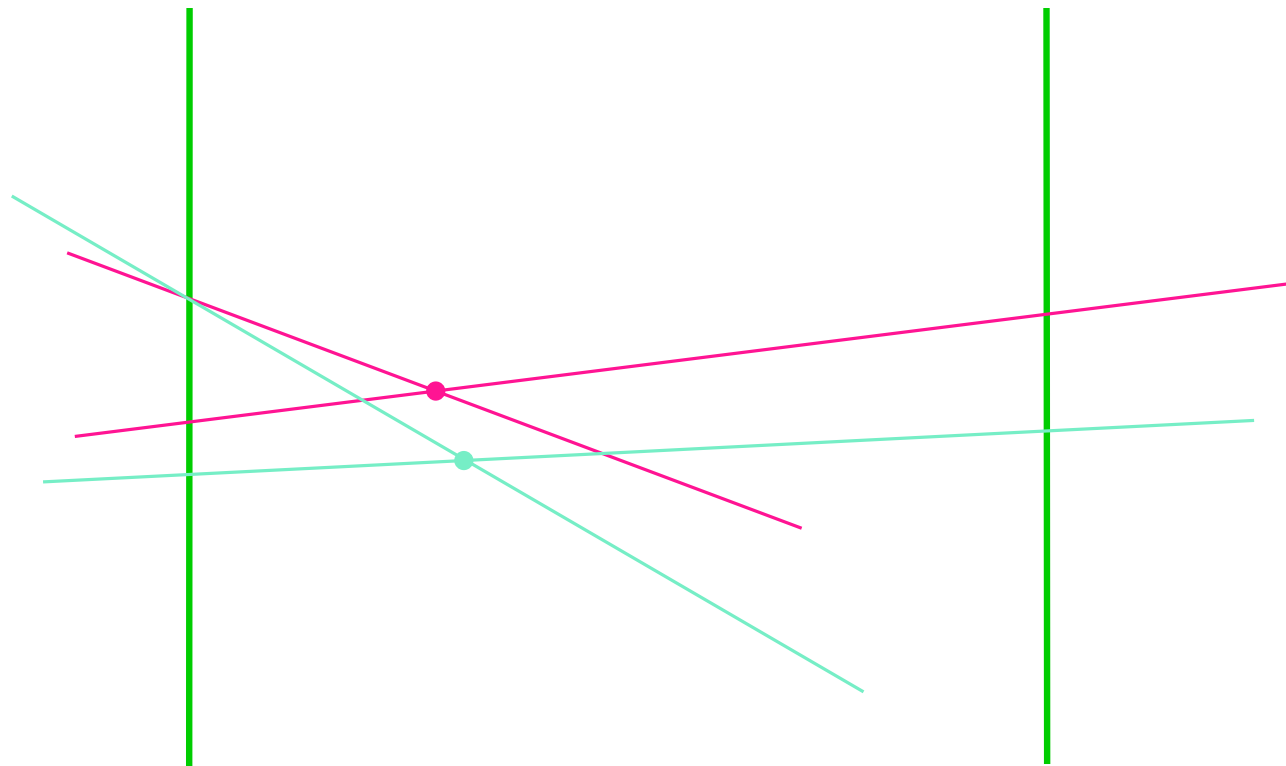
SOLVING LINEAR PROGRAMS

The algorithm

Initialization Eliminate all $i \in I_0$ except those corresponding to u_1 and u_2 , if they exist.

Advance. Repeat as many times as necessary:

1. Pair lines up
2. Prune parallel and external lines
3. Compute the median value x' of the surviving x_{ij}
4. Search
5. Prune



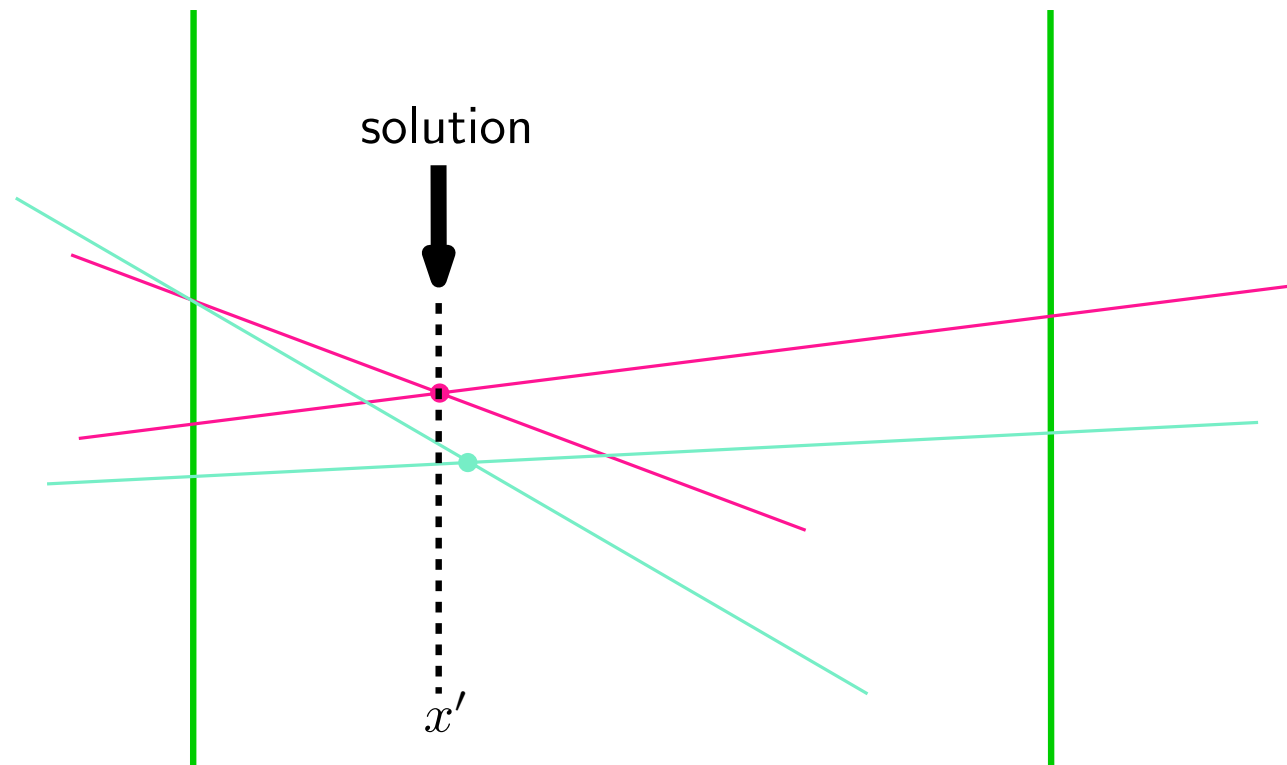
SOLVING LINEAR PROGRAMS

The algorithm

Initialization Eliminate all $i \in I_0$ except those corresponding to u_1 and u_2 , if they exist.

Advance. Repeat as many times as necessary:

1. Pair lines up
2. Prune parallel and external lines
3. Compute the median value x' of the surviving x_{ij}
4. Search
5. Prune



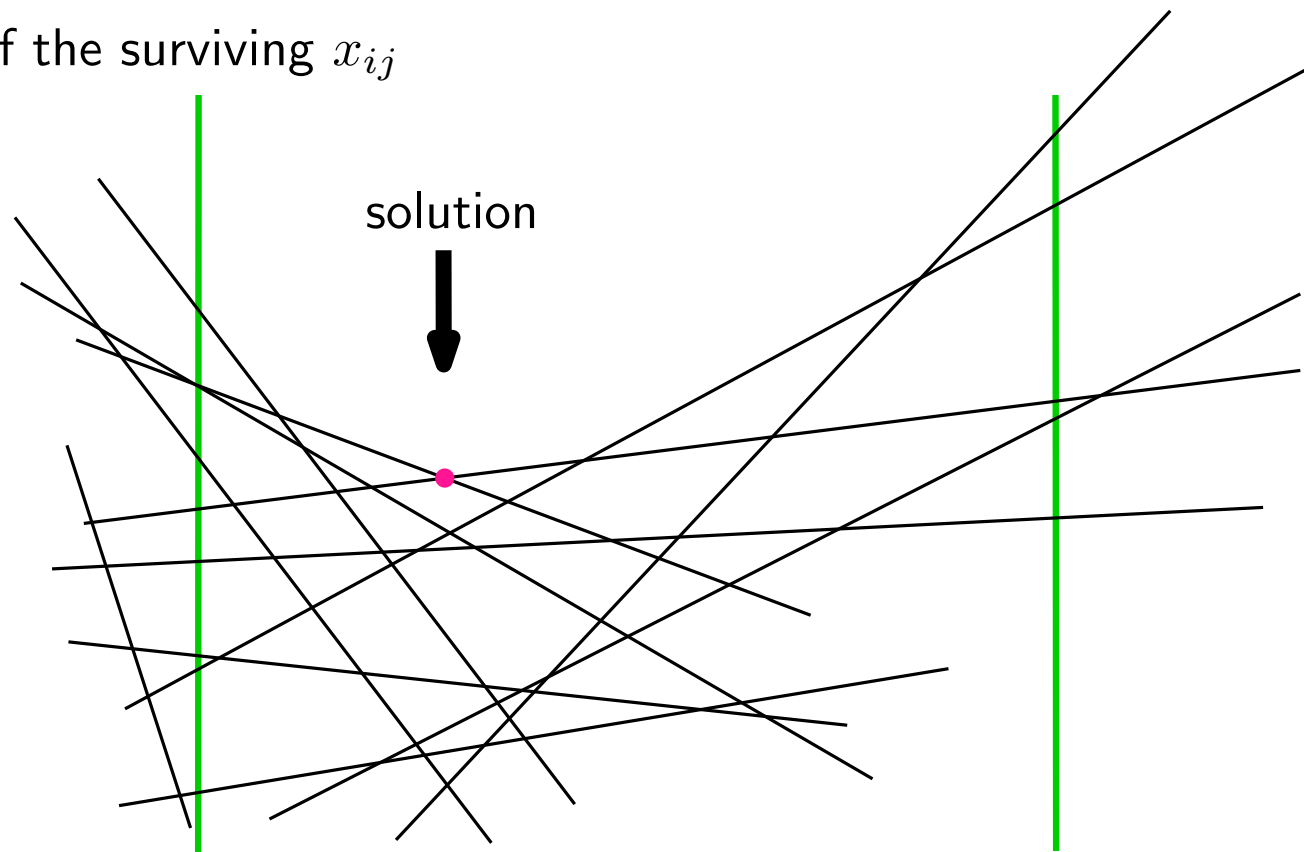
SOLVING LINEAR PROGRAMS

The algorithm

Initialization Eliminate all $i \in I_0$ except those corresponding to u_1 and u_2 , if they exist.

Advance. Repeat as many times as necessary:

1. Pair lines up
2. Prune parallel and external lines
3. Compute the median value x' of the surviving x_{ij}
4. Search
5. Prune



SOLVING LINEAR PROGRAMS

The algorithm

Initialization Eliminate all $i \in I_0$ except those corresponding to u_1 and u_2 , if they exist.

Advance. Repeat as many times as necessary:

1. Pair up restrictions and compute the abscissa of the intersection points. $O(n)$ time
2. For all pairs, do: $O(n)$ time
 - If $d_i = d_j$, eliminate one of the two.
 - If $d_i \neq d_j$ and x_{ij} is the abscissa of the intersection point of the two lines, then do:
 - If $x_{ij} < u_1$, eliminate one of the two.
 - If $x_{ij} > u_2$, eliminate one of the two.
3. Compute the median value x' of the surviving x_{ij} . $O(n)$ time (see the appropriate reference)
4. Search: $O(n)$ time
5. Prune: $O(n)$ time

SOLVING LINEAR PROGRAMS

The algorithm

Initialization Eliminate all $i \in I_0$ except those corresponding to u_1 and u_2 , if they exist. $O(n)$

Advance. Repeat as many times as necessary:

1. Pair up restrictions and compute the abscissa of the intersection points. $O(n)$ time
2. For all pairs, do: $O(n)$ time
 - If $d_i = d_j$, eliminate one of the two.
 - If $d_i \neq d_j$ and x_{ij} is the abscissa of the intersection point of the two lines, then do:
 - If $x_{ij} < u_1$, eliminate one of the two.
 - If $x_{ij} > u_2$, eliminate one of the two.
3. Compute the median value x' of the surviving x_{ij} . $O(n)$ time (see the appropriate reference)
4. Search: $O(n)$ time
5. Prune: $O(n)$ time

How many restrictions are pruned at each step?

SOLVING LINEAR PROGRAMS

The algorithm

Initialization Eliminate all $i \in I_0$ except those corresponding to u_1 and u_2 , if they exist. $O(n)$

Advance. Repeat as many times as necessary:

1. Pair up restrictions and compute the abscissa of the intersection points. $O(n)$ time
2. For all pairs, do: $O(n)$ time
 - If $d_i = d_j$, eliminate one of the two.
 - If $d_i \neq d_j$ and x_{ij} is the abscissa of the intersection point of the two lines, then do:
 - If $x_{ij} < u_1$, eliminate one of the two.
 - If $x_{ij} > u_2$, eliminate one of the two.
3. Compute the median value x' of the surviving x_{ij} . $O(n)$ time (see the appropriate reference)
4. Search: $O(n)$ time
5. Prune: $O(n)$ time

How many restrictions are pruned at each step? $O(n/4)$

SOLVING LINEAR PROGRAMS

The algorithm

Exact counting:

- The initial number of vertical restrictions is k , where $k \geq 0$.
- The vertical restrictions prune step eliminates all of them but at most 2.
- The initial number of pairs of non vertical lines (i.e., points x_{ij}) is $\frac{m}{2}$, $\frac{m-1}{2}$ or $\frac{m}{2} - 1$, depending on the parities of $|I_-|$ and $|I_+|$, where $m + k = n$.
- The prune step eliminates roughly half of them.

Alltogether, the eliminated restrictions are, at least:

$$k + \left\lfloor \frac{\frac{m}{2} - 1}{2} \right\rfloor = \left\lfloor \frac{m}{4} + k - \frac{1}{2} \right\rfloor = \left\lfloor \frac{n}{4} + \frac{3}{4}k - \frac{1}{2} \right\rfloor \geq \frac{n}{4}.$$

SOLVING LINEAR PROGRAMS

The algorithm

Exact counting:

- The initial number of vertical restrictions is k , where $k \geq 0$.
- The vertical restrictions prune step eliminates all of them but at most 2.
- The initial number of pairs of non vertical lines (i.e., points x_{ij}) is $\frac{m}{2}$, $\frac{m-1}{2}$ or $\frac{m}{2} - 1$, depending on the parities of $|I_-|$ and $|I_+|$, where $m + k = n$.
- The prune step eliminates roughly half of them.

Alltogether, the eliminated restrictions are, at least:

$$k + \left\lfloor \frac{\frac{m}{2} - 1}{2} \right\rfloor = \left\lfloor \frac{m}{4} + k - \frac{1}{2} \right\rfloor = \left\lfloor \frac{n}{4} + \frac{3}{4}k - \frac{1}{2} \right\rfloor \geq \frac{n}{4}.$$

Running time

Hence, the running time of the algorithm is

$$cn + c\frac{3}{4}n + c\left(\frac{3}{4}\right)^2 n + c\left(\frac{3}{4}\right)^3 n + \dots = cn \sum_{k=0}^{\lceil \log_{3/4} n \rceil} \left(\frac{3}{4}\right)^k < cn \frac{1}{1 - \frac{3}{4}} = 4cn \in O(n).$$

MINIMUM SPANNING (or ENCLOSING) CIRCLE

MIN-MAX FACILITY LOCATION

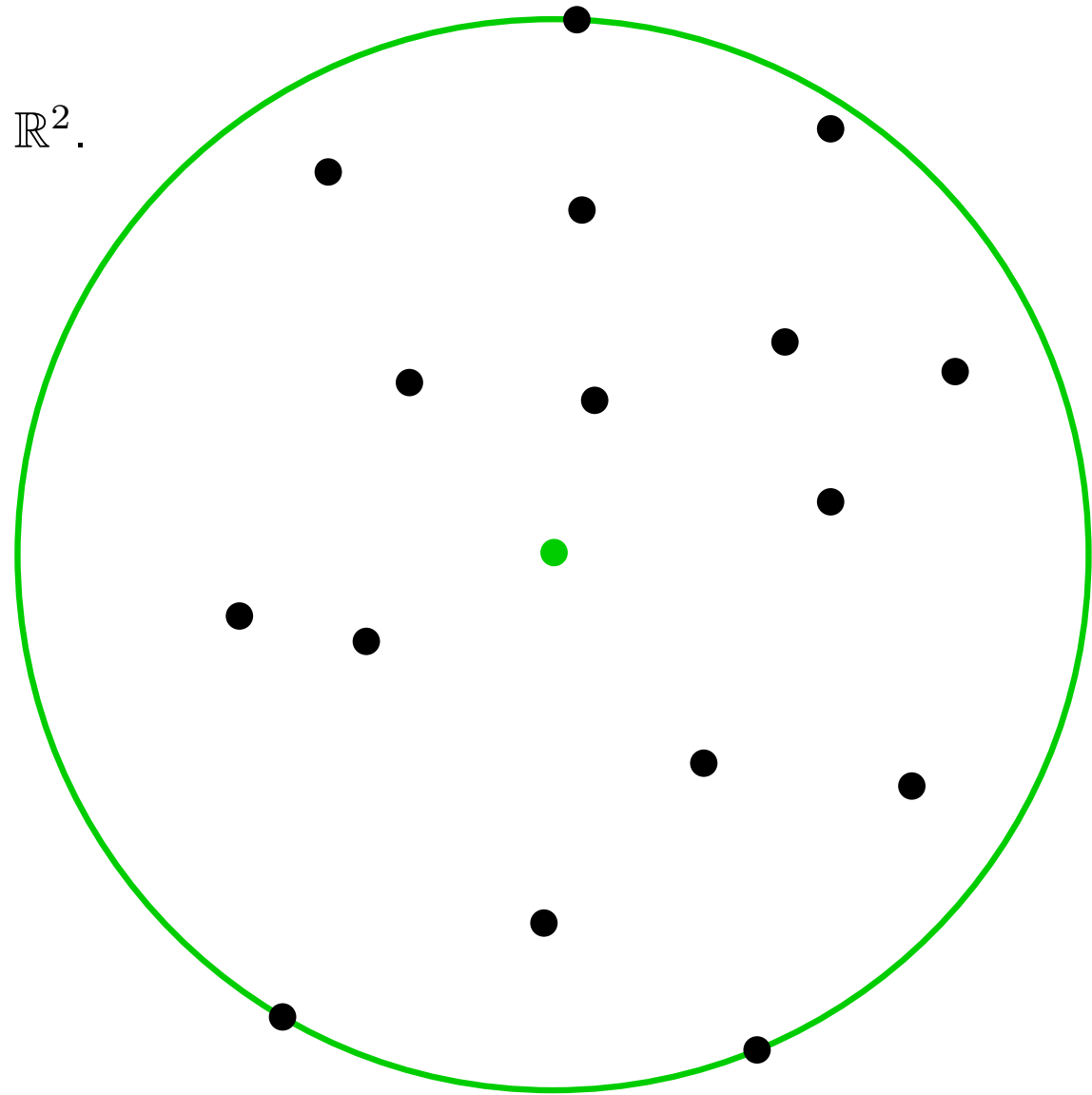
Let $P = \{p_1, \dots, p_n\}$ be a set of n points in \mathbb{R}^2 .

Find the point x on the plane achieving

$$\min_{x \in \mathbb{R}^2} \max_{p_i \in P} d(x, p_i).$$

Geometrically

Find the center of the circle of minimum radius enclosing P .



MINIMUM SPANNING (or ENCLOSING) CIRCLE

MIN-MAX FACILITY LOCATION

Let $P = \{p_1, \dots, p_n\}$ be a set of n points in \mathbb{R}^2 .

Find the point x on the plane achieving

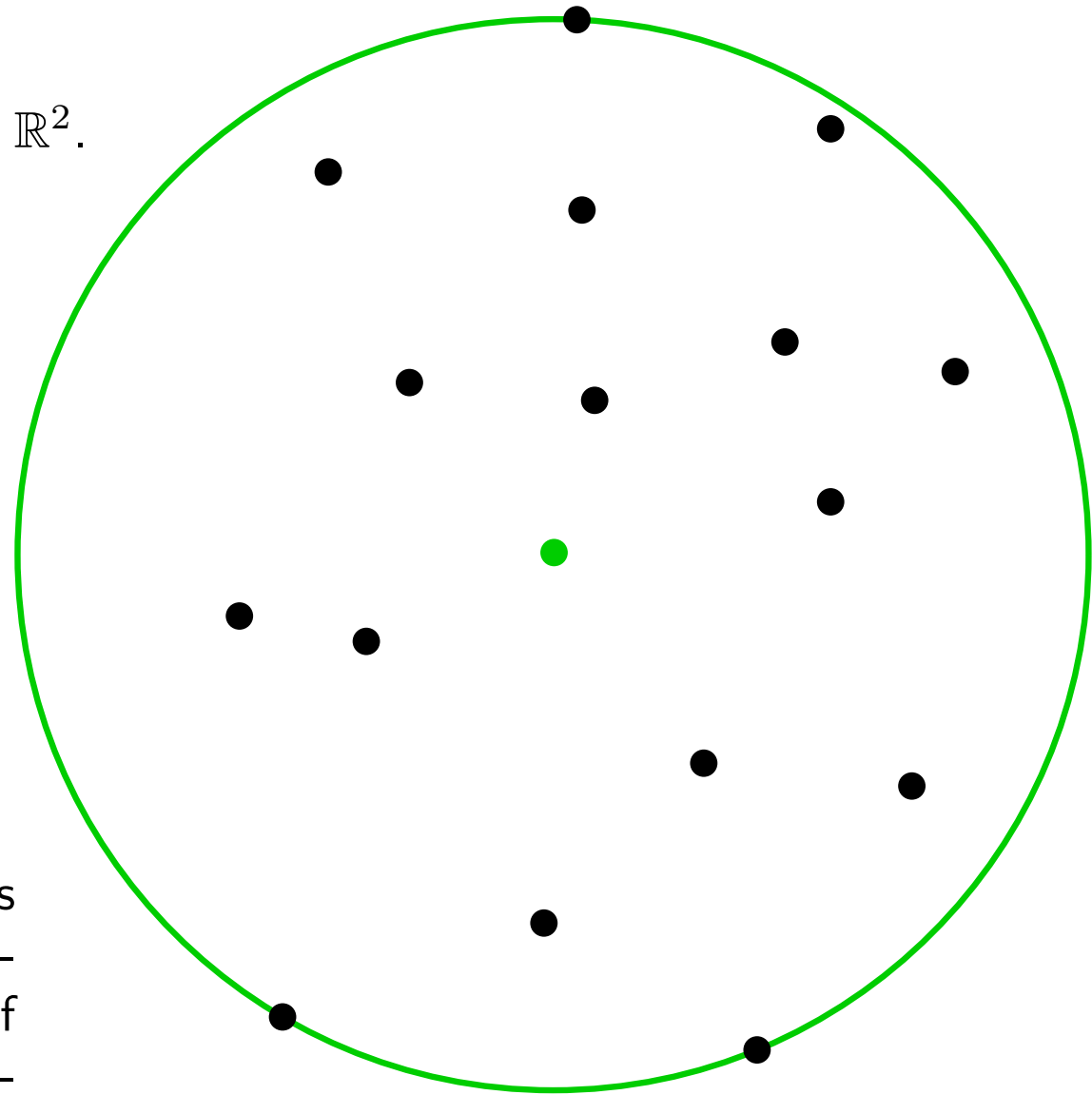
$$\min_{x \in \mathbb{R}^2} \max_{p_i \in P} d(x, p_i).$$

Geometrically

Find the center of the circle of minimum radius enclosing P .

Brute-force algorithm

Observe that we only need to consider circles defined by two or three points in P . There are $O(n^3)$ candidate circles. Try each of them until finding the smallest one containing all points. Total: $O(n^4)$ time.



COMPUTING THE MINIMUM SPANNING CIRCLE

COMPUTING THE MINIMUM SPANNING CIRCLE

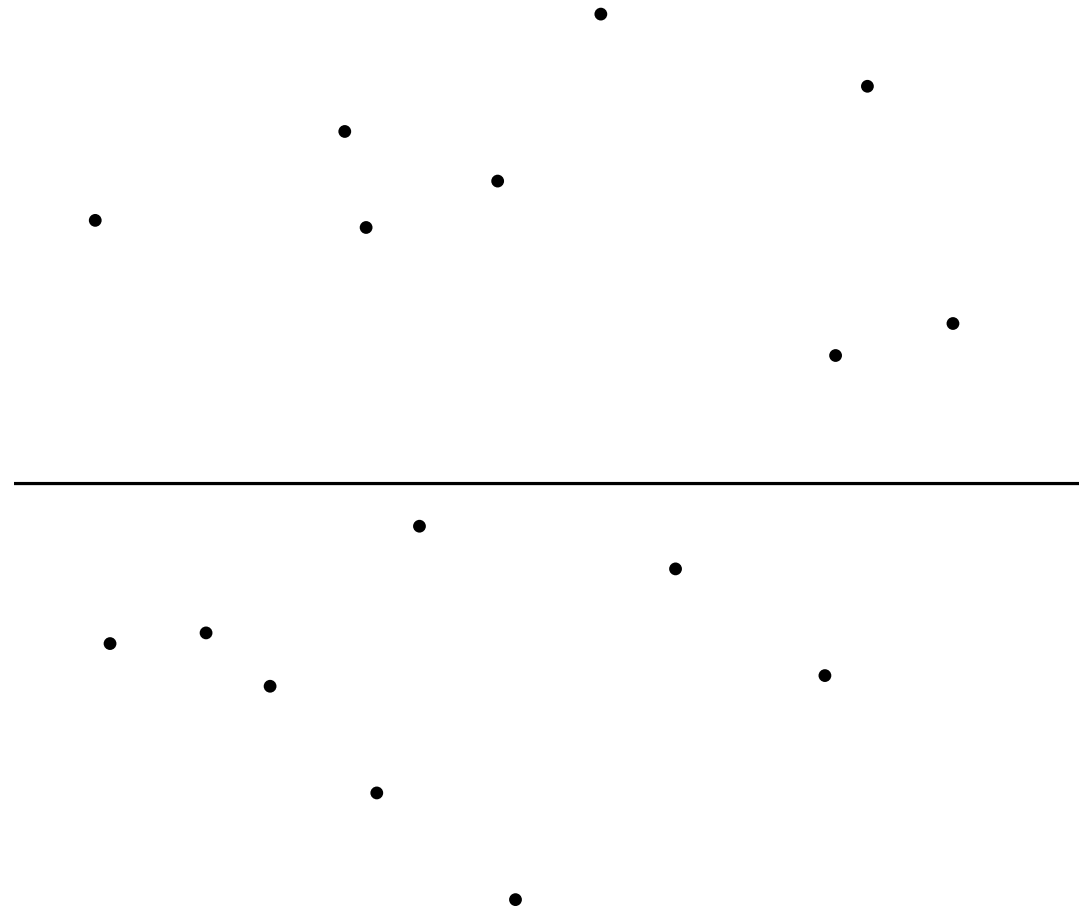
First we solve a subproblem: MSC restricted to line $y = 0$

COMPUTING THE MINIMUM SPANNING CIRCLE

First we solve a subproblem: MSC restricted to line $y = 0$

Input: a set of n points (a_i, b_i) , $i = 1, \dots, n$.

Output: $\min_{x \in \mathbb{R}} \max_{i=1 \dots n} (x - a_i)^2 + (y - b_i)^2 = \min_{x \in \mathbb{R}} \max_{i=1 \dots n} (x - a_i)^2 + b_i^2$

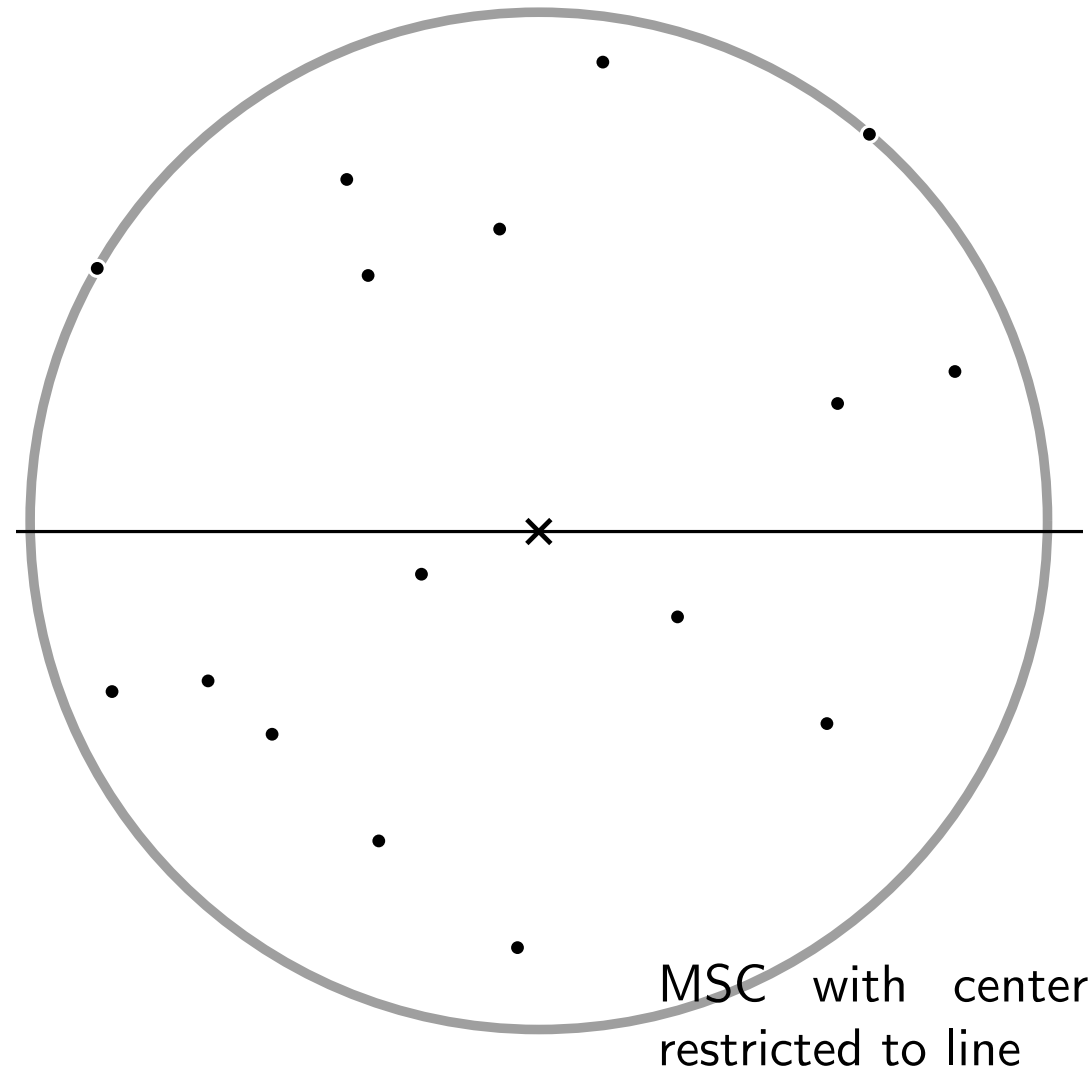


COMPUTING THE MINIMUM SPANNING CIRCLE

First we solve a subproblem: MSC restricted to line $y = 0$

Input: a set of n points (a_i, b_i) , $i = 1, \dots, n$.

Output: $\min_{x \in \mathbb{R}} \max_{i=1 \dots n} (x - a_i)^2 + (y - b_i)^2 = \min_{x \in \mathbb{R}} \max_{i=1 \dots n} (x - a_i)^2 + b_i^2$



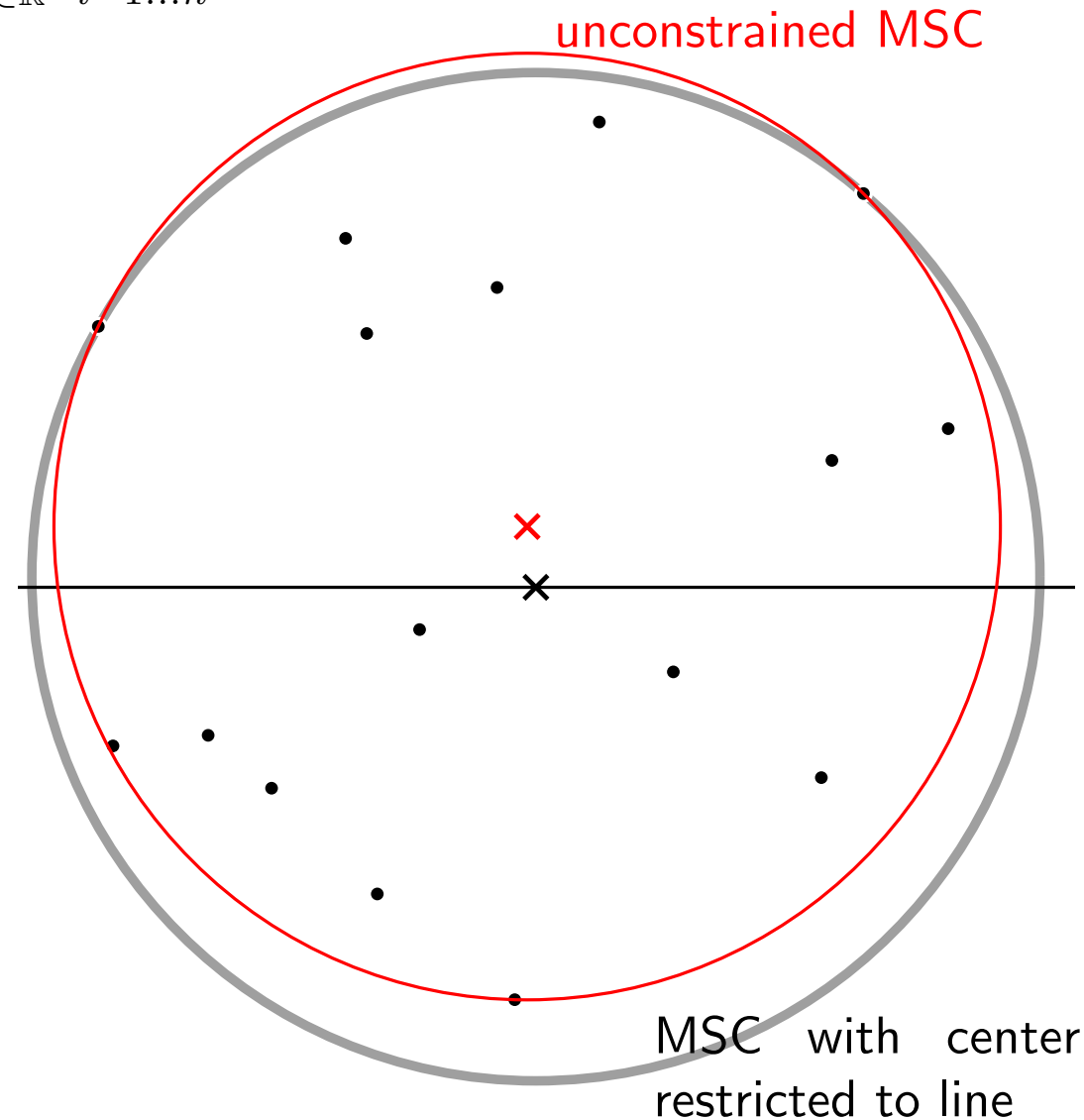
COMPUTING THE MINIMUM SPANNING CIRCLE

First we solve a subproblem: MSC restricted to line $y = 0$

Input: a set of n points (a_i, b_i) , $i = 1, \dots, n$.

Output: $\min_{x \in \mathbb{R}} \max_{i=1 \dots n} (x - a_i)^2 + (y - b_i)^2 = \min_{x \in \mathbb{R}} \max_{i=1 \dots n} (x - a_i)^2 + b_i^2$

Note that the solution to the MSC restricted to a line can be different from the solution to the MSC without restrictions!



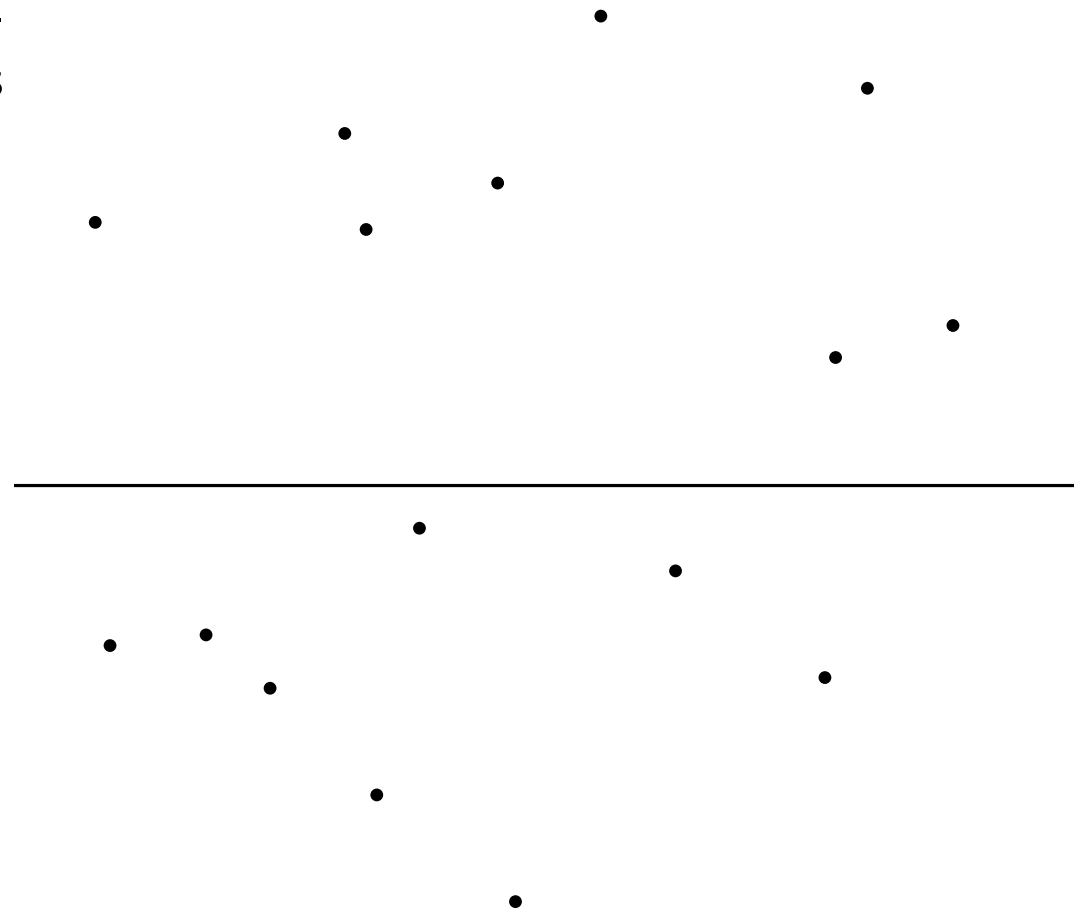
COMPUTING THE MINIMUM SPANNING CIRCLE

First we solve a subproblem: MSC restricted to line $y = 0$

Input: a set of n points (a_i, b_i) , $i = 1, \dots, n$.

Output: $\min_{x \in \mathbb{R}} \max_{i=1 \dots n} (x - a_i)^2 + (y - b_i)^2 = \min_{x \in \mathbb{R}} \max_{i=1 \dots n} (x - a_i)^2 + b_i^2$

1. Pair up the points. For each pair p_i, p_j , compute its perpendicular bisector b_{ij} and find its intersection x_{ij} with the line $y = 0$.



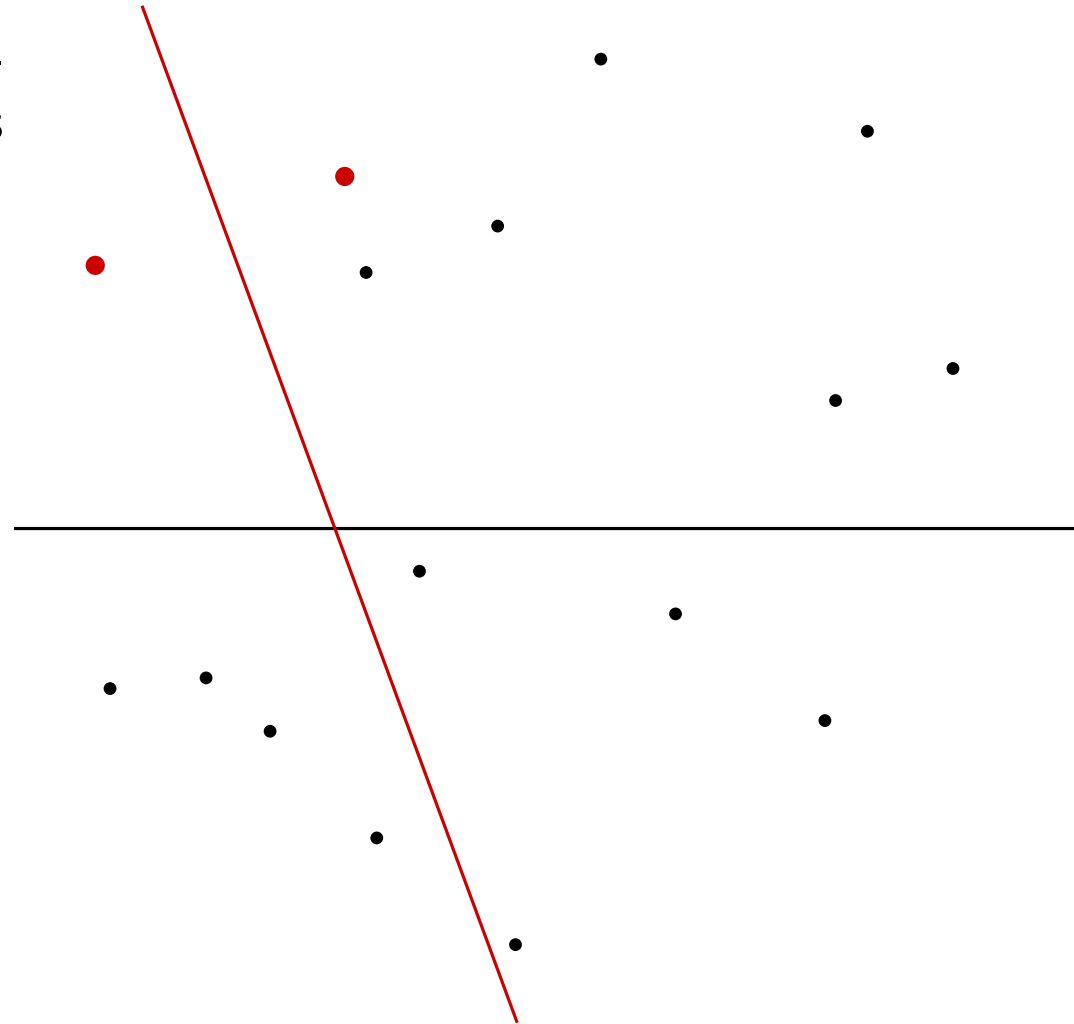
COMPUTING THE MINIMUM SPANNING CIRCLE

First we solve a subproblem: MSC restricted to line $y = 0$

Input: a set of n points (a_i, b_i) , $i = 1, \dots, n$.

Output: $\min_{x \in \mathbb{R}} \max_{i=1 \dots n} (x - a_i)^2 + (y - b_i)^2 = \min_{x \in \mathbb{R}} \max_{i=1 \dots n} (x - a_i)^2 + b_i^2$

1. Pair up the points. For each pair p_i, p_j , compute its perpendicular bisector b_{ij} and find its intersection x_{ij} with the line $y = 0$.



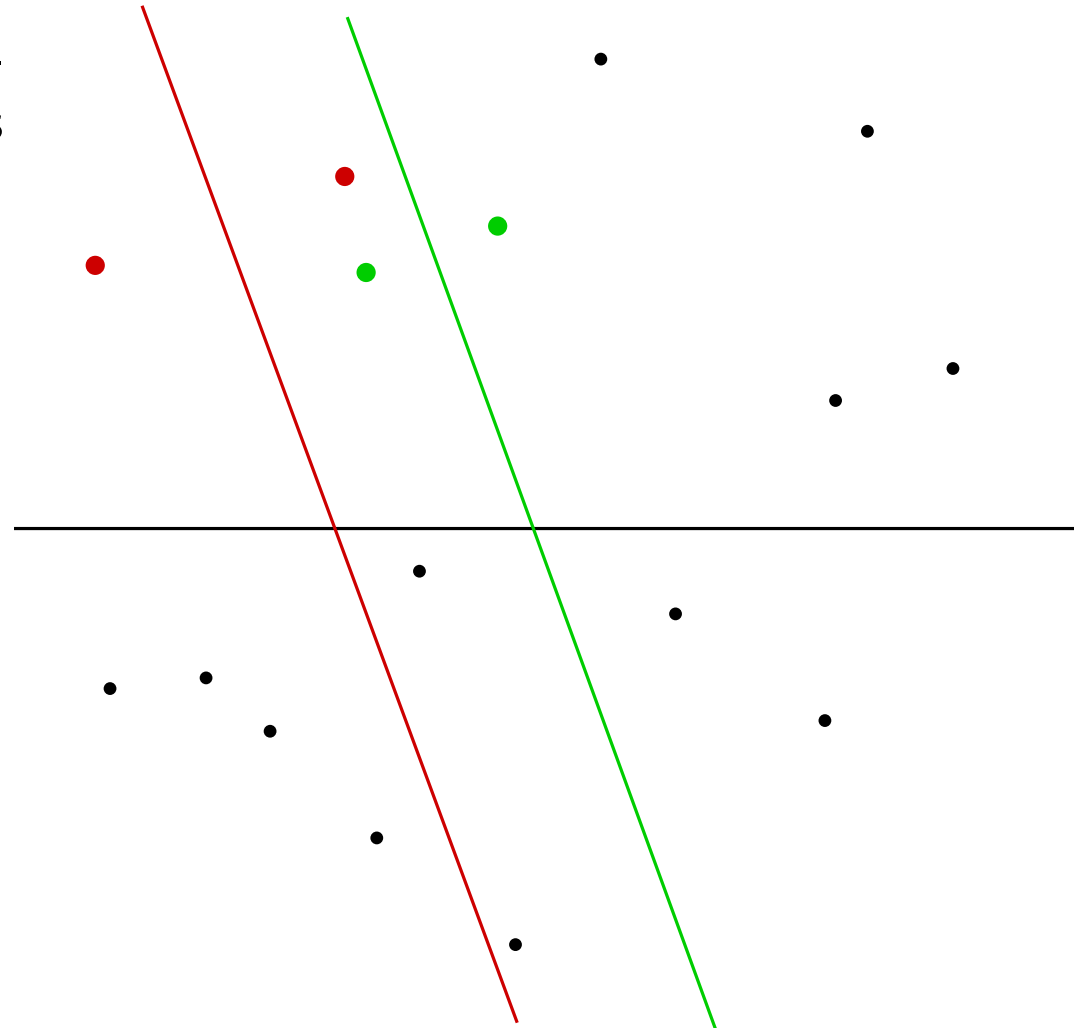
COMPUTING THE MINIMUM SPANNING CIRCLE

First we solve a subproblem: MSC restricted to line $y = 0$

Input: a set of n points (a_i, b_i) , $i = 1, \dots, n$.

Output: $\min_{x \in \mathbb{R}} \max_{i=1 \dots n} (x - a_i)^2 + (y - b_i)^2 = \min_{x \in \mathbb{R}} \max_{i=1 \dots n} (x - a_i)^2 + b_i^2$

1. Pair up the points. For each pair p_i, p_j , compute its perpendicular bisector b_{ij} and find its intersection x_{ij} with the line $y = 0$.



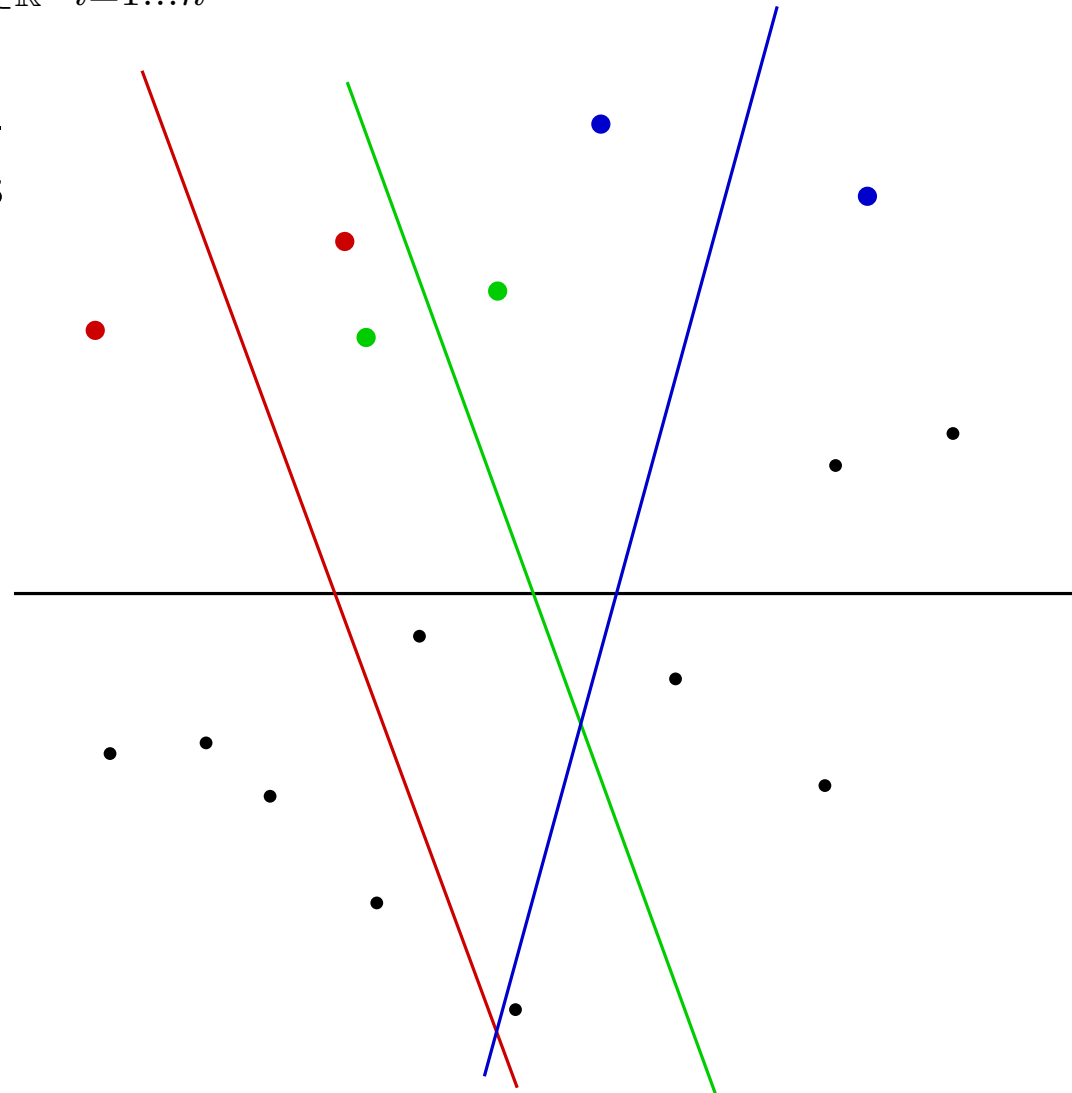
COMPUTING THE MINIMUM SPANNING CIRCLE

First we solve a subproblem: MSC restricted to line $y = 0$

Input: a set of n points (a_i, b_i) , $i = 1, \dots, n$.

Output: $\min_{x \in \mathbb{R}} \max_{i=1 \dots n} (x - a_i)^2 + (y - b_i)^2 = \min_{x \in \mathbb{R}} \max_{i=1 \dots n} (x - a_i)^2 + b_i^2$

1. Pair up the points. For each pair p_i, p_j , compute its perpendicular bisector b_{ij} and find its intersection x_{ij} with the line $y = 0$.



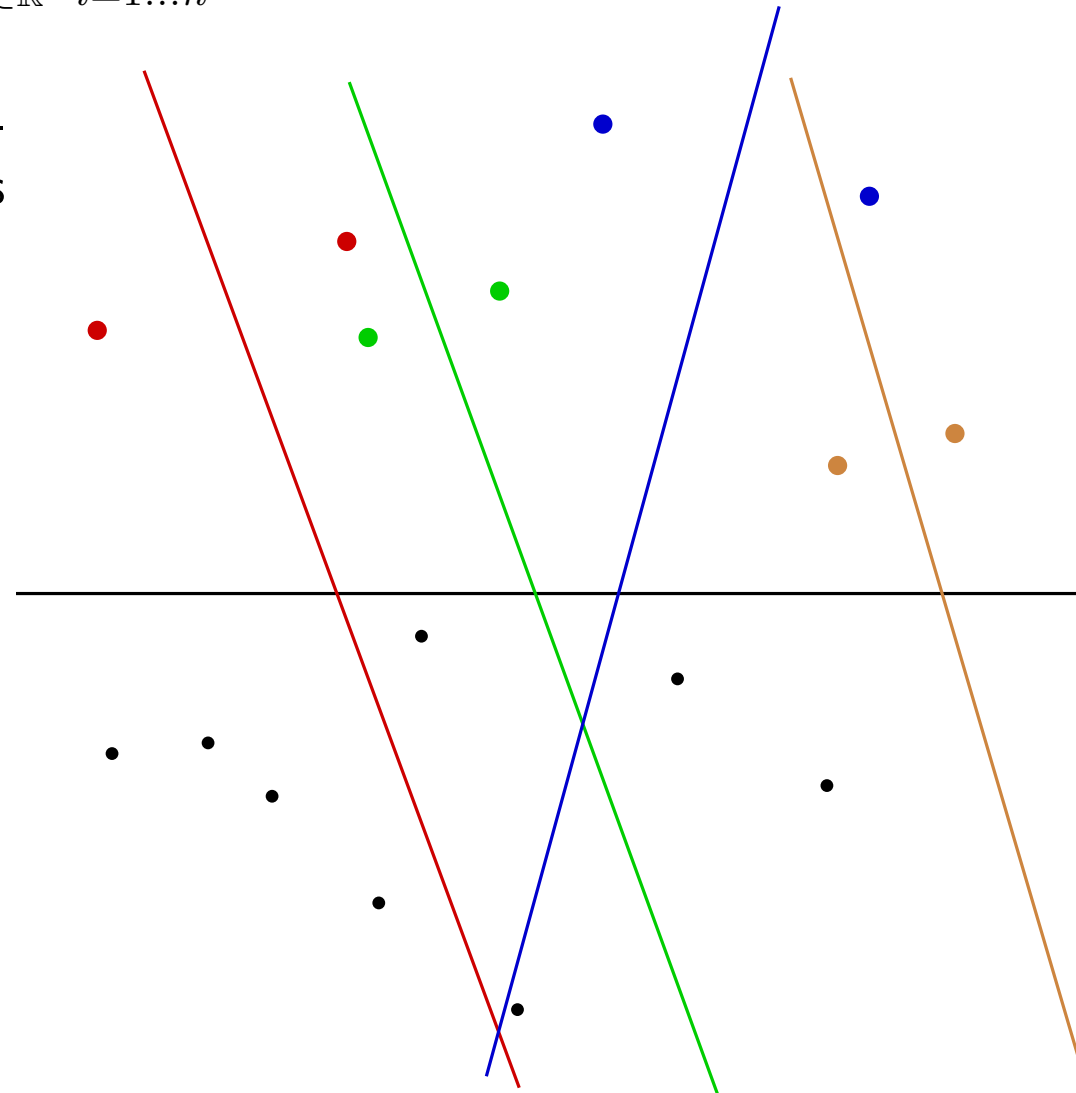
COMPUTING THE MINIMUM SPANNING CIRCLE

First we solve a subproblem: MSC restricted to line $y = 0$

Input: a set of n points (a_i, b_i) , $i = 1, \dots, n$.

Output: $\min_{x \in \mathbb{R}} \max_{i=1 \dots n} (x - a_i)^2 + (y - b_i)^2 = \min_{x \in \mathbb{R}} \max_{i=1 \dots n} (x - a_i)^2 + b_i^2$

1. Pair up the points. For each pair p_i, p_j , compute its perpendicular bisector b_{ij} and find its intersection x_{ij} with the line $y = 0$.



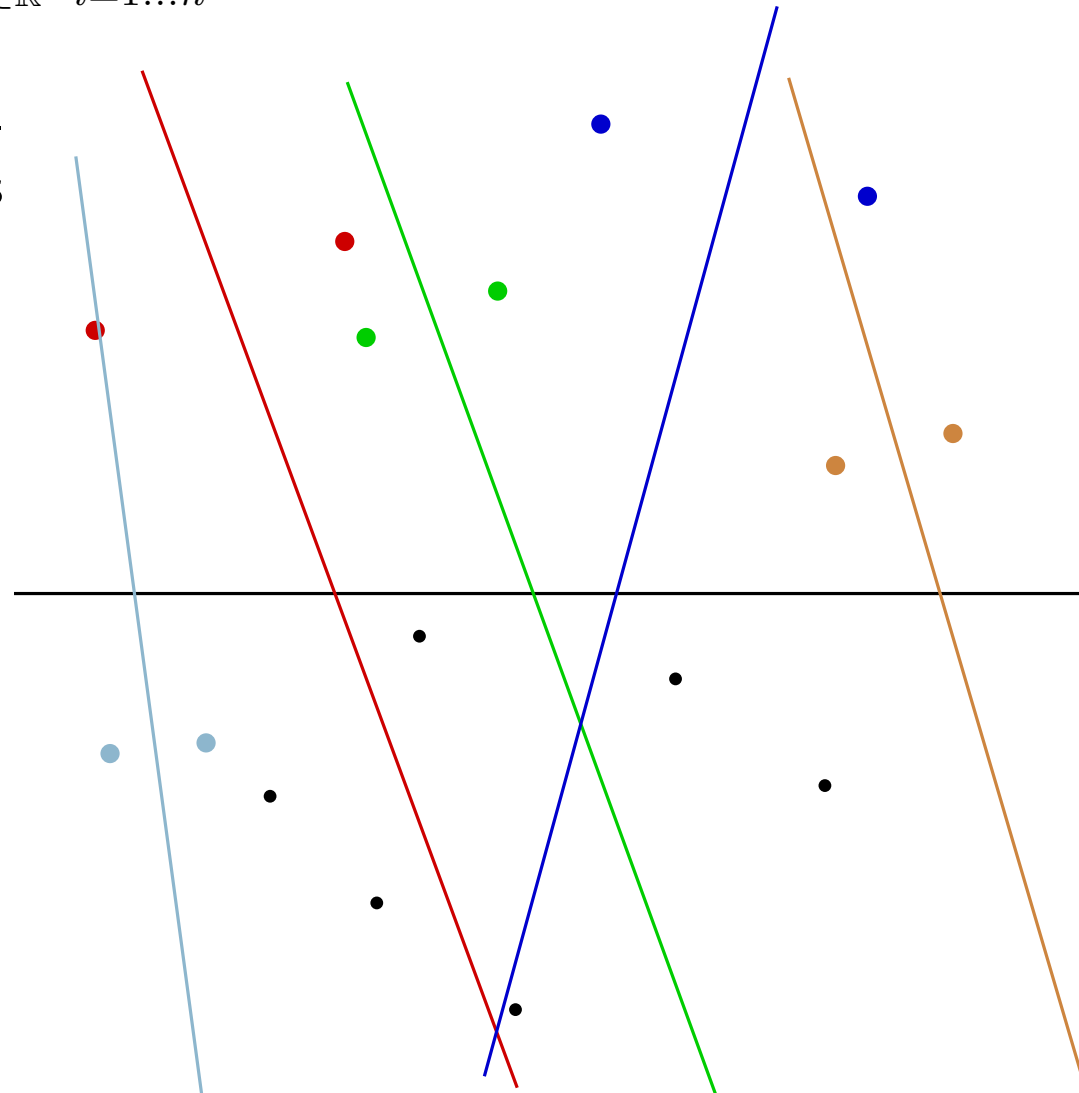
COMPUTING THE MINIMUM SPANNING CIRCLE

First we solve a subproblem: MSC restricted to line $y = 0$

Input: a set of n points (a_i, b_i) , $i = 1, \dots, n$.

Output: $\min_{x \in \mathbb{R}} \max_{i=1 \dots n} (x - a_i)^2 + (y - b_i)^2 = \min_{x \in \mathbb{R}} \max_{i=1 \dots n} (x - a_i)^2 + b_i^2$

1. Pair up the points. For each pair p_i, p_j , compute its perpendicular bisector b_{ij} and find its intersection x_{ij} with the line $y = 0$.



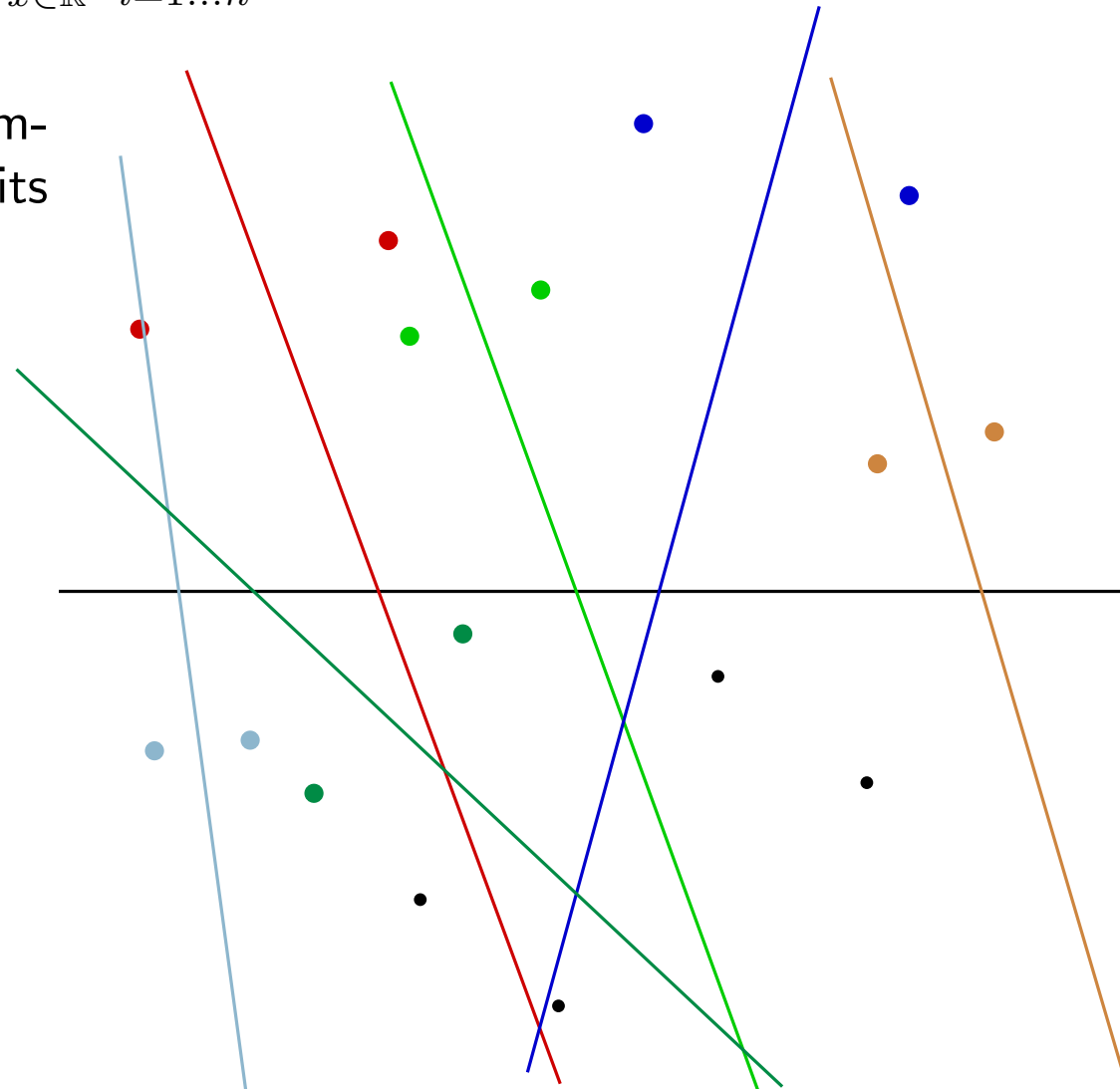
COMPUTING THE MINIMUM SPANNING CIRCLE

First we solve a subproblem: MSC restricted to line $y = 0$

Input: a set of n points (a_i, b_i) , $i = 1, \dots, n$.

Output: $\min_{x \in \mathbb{R}} \max_{i=1 \dots n} (x - a_i)^2 + (y - b_i)^2 = \min_{x \in \mathbb{R}} \max_{i=1 \dots n} (x - a_i)^2 + b_i^2$

1. Pair up the points. For each pair p_i, p_j , compute its perpendicular bisector b_{ij} and find its intersection x_{ij} with the line $y = 0$.



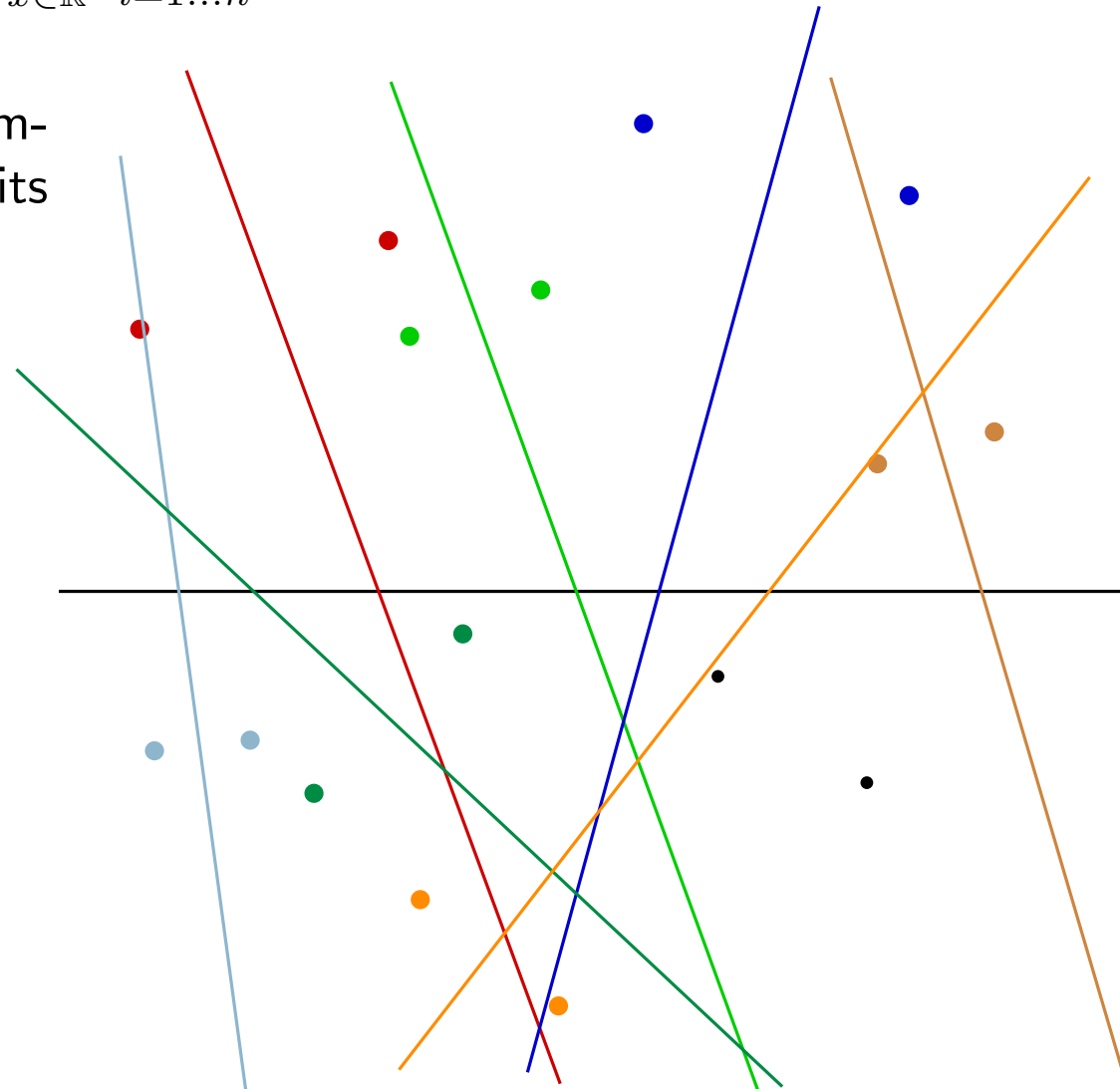
COMPUTING THE MINIMUM SPANNING CIRCLE

First we solve a subproblem: MSC restricted to line $y = 0$

Input: a set of n points (a_i, b_i) , $i = 1, \dots, n$.

Output: $\min_{x \in \mathbb{R}} \max_{i=1 \dots n} (x - a_i)^2 + (y - b_i)^2 = \min_{x \in \mathbb{R}} \max_{i=1 \dots n} (x - a_i)^2 + b_i^2$

1. Pair up the points. For each pair p_i, p_j , compute its perpendicular bisector b_{ij} and find its intersection x_{ij} with the line $y = 0$.



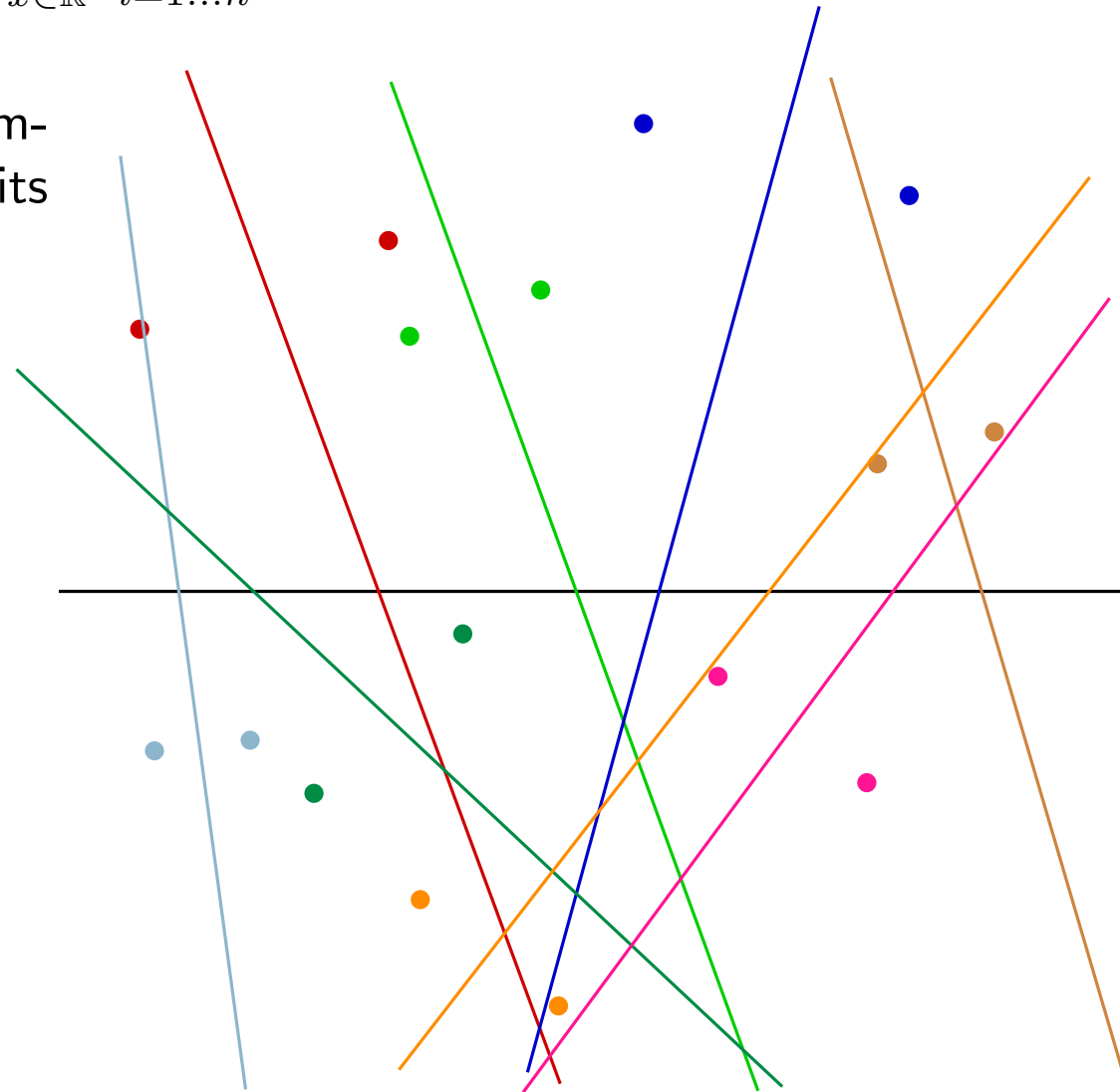
COMPUTING THE MINIMUM SPANNING CIRCLE

First we solve a subproblem: MSC restricted to line $y = 0$

Input: a set of n points (a_i, b_i) , $i = 1, \dots, n$.

Output: $\min_{x \in \mathbb{R}} \max_{i=1 \dots n} (x - a_i)^2 + (y - b_i)^2 = \min_{x \in \mathbb{R}} \max_{i=1 \dots n} (x - a_i)^2 + b_i^2$

1. Pair up the points. For each pair p_i, p_j , compute its perpendicular bisector b_{ij} and find its intersection x_{ij} with the line $y = 0$.



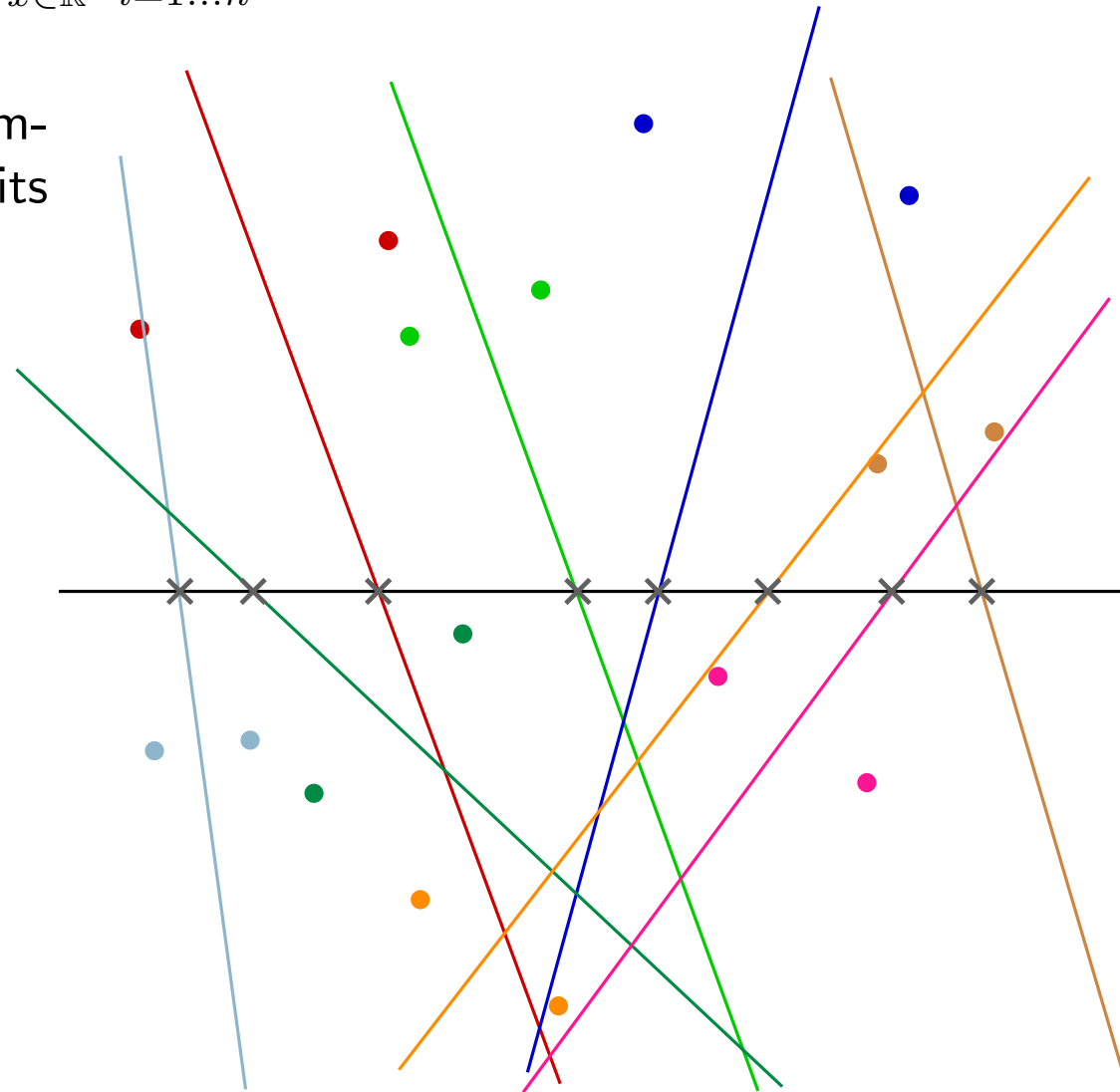
COMPUTING THE MINIMUM SPANNING CIRCLE

First we solve a subproblem: MSC restricted to line $y = 0$

Input: a set of n points (a_i, b_i) , $i = 1, \dots, n$.

Output: $\min_{x \in \mathbb{R}} \max_{i=1 \dots n} (x - a_i)^2 + (y - b_i)^2 = \min_{x \in \mathbb{R}} \max_{i=1 \dots n} (x - a_i)^2 + b_i^2$

1. Pair up the points. For each pair p_i, p_j , compute its perpendicular bisector b_{ij} and find its intersection x_{ij} with the line $y = 0$.



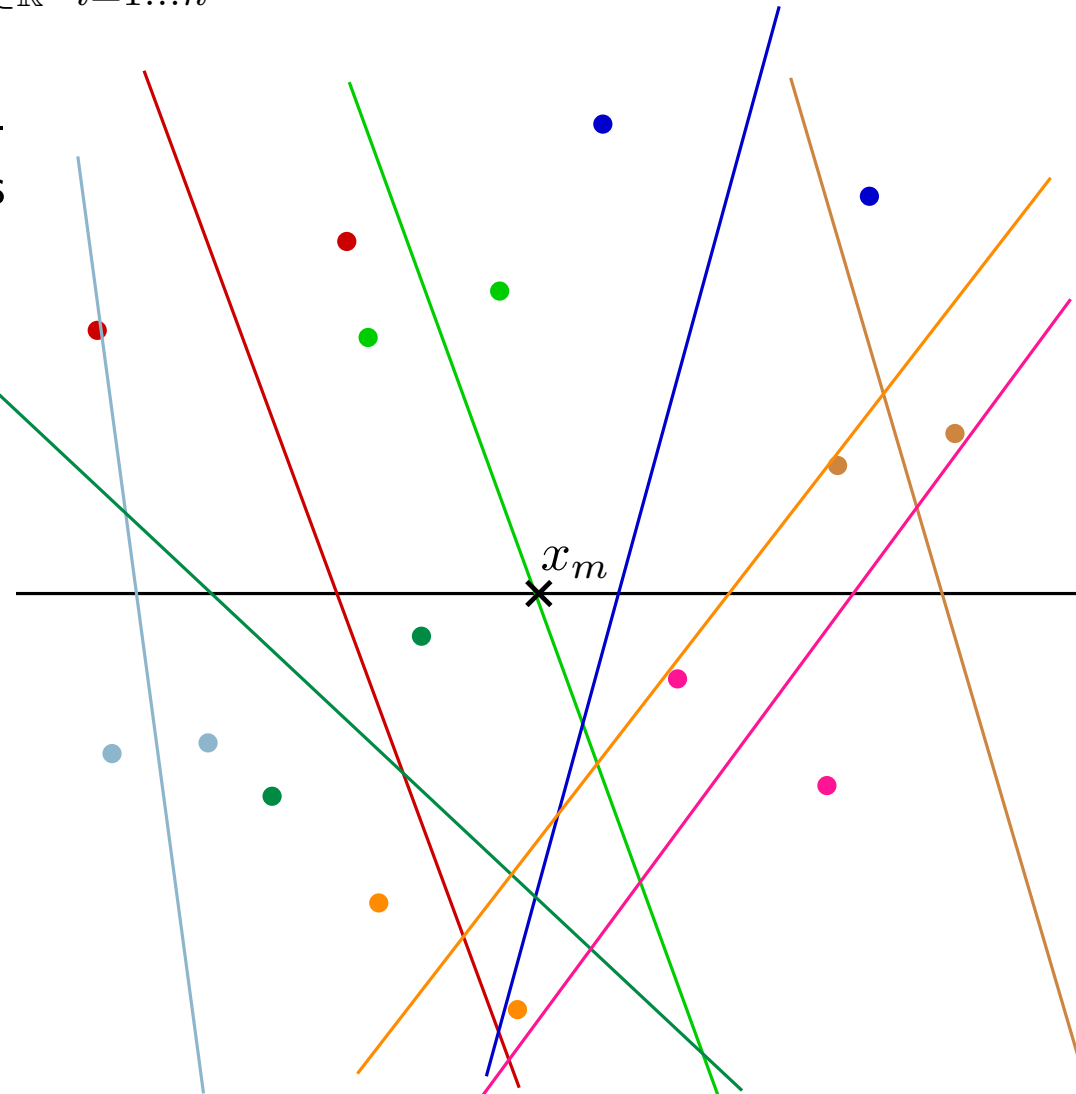
COMPUTING THE MINIMUM SPANNING CIRCLE

First we solve a subproblem: MSC restricted to line $y = 0$

Input: a set of n points (a_i, b_i) , $i = 1, \dots, n$.

Output: $\min_{x \in \mathbb{R}} \max_{i=1 \dots n} (x - a_i)^2 + (y - b_i)^2 = \min_{x \in \mathbb{R}} \max_{i=1 \dots n} (x - a_i)^2 + b_i^2$

1. Pair up the points. For each pair p_i, p_j , compute its perpendicular bisector b_{ij} and find its intersection x_{ij} with the line $y = 0$.
2. Compute x_m , the median value of the x_{ij} .



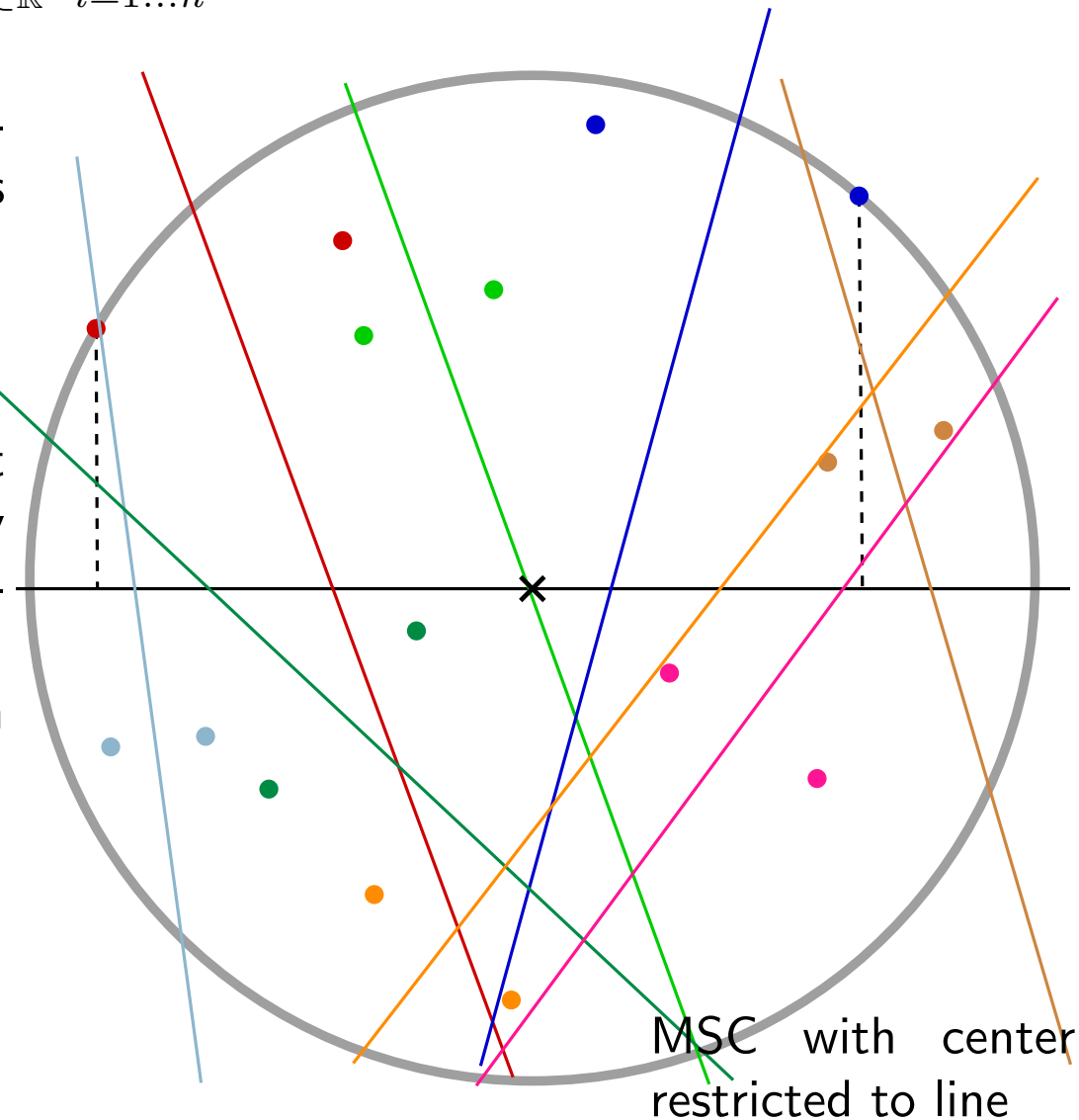
COMPUTING THE MINIMUM SPANNING CIRCLE

First we solve a subproblem: MSC restricted to line $y = 0$

Input: a set of n points (a_i, b_i) , $i = 1, \dots, n$.

Output: $\min_{x \in \mathbb{R}} \max_{i=1 \dots n} (x - a_i)^2 + (y - b_i)^2 = \min_{x \in \mathbb{R}} \max_{i=1 \dots n} (x - a_i)^2 + b_i^2$

1. Pair up the points. For each pair p_i, p_j , compute its perpendicular bisector b_{ij} and find its intersection x_{ij} with the line $y = 0$.
2. Compute x_m , the median value of the x_{ij} .
3. Search: Compute C_m , the MSC centered at x_m . If the points p_i lying in the boundary of C_m project orthogonally onto $y = 0$ to different sides of x_m , then x_m is the solution. If they all project onto the same side then search on that side.



COMPUTING THE MINIMUM SPANNING CIRCLE

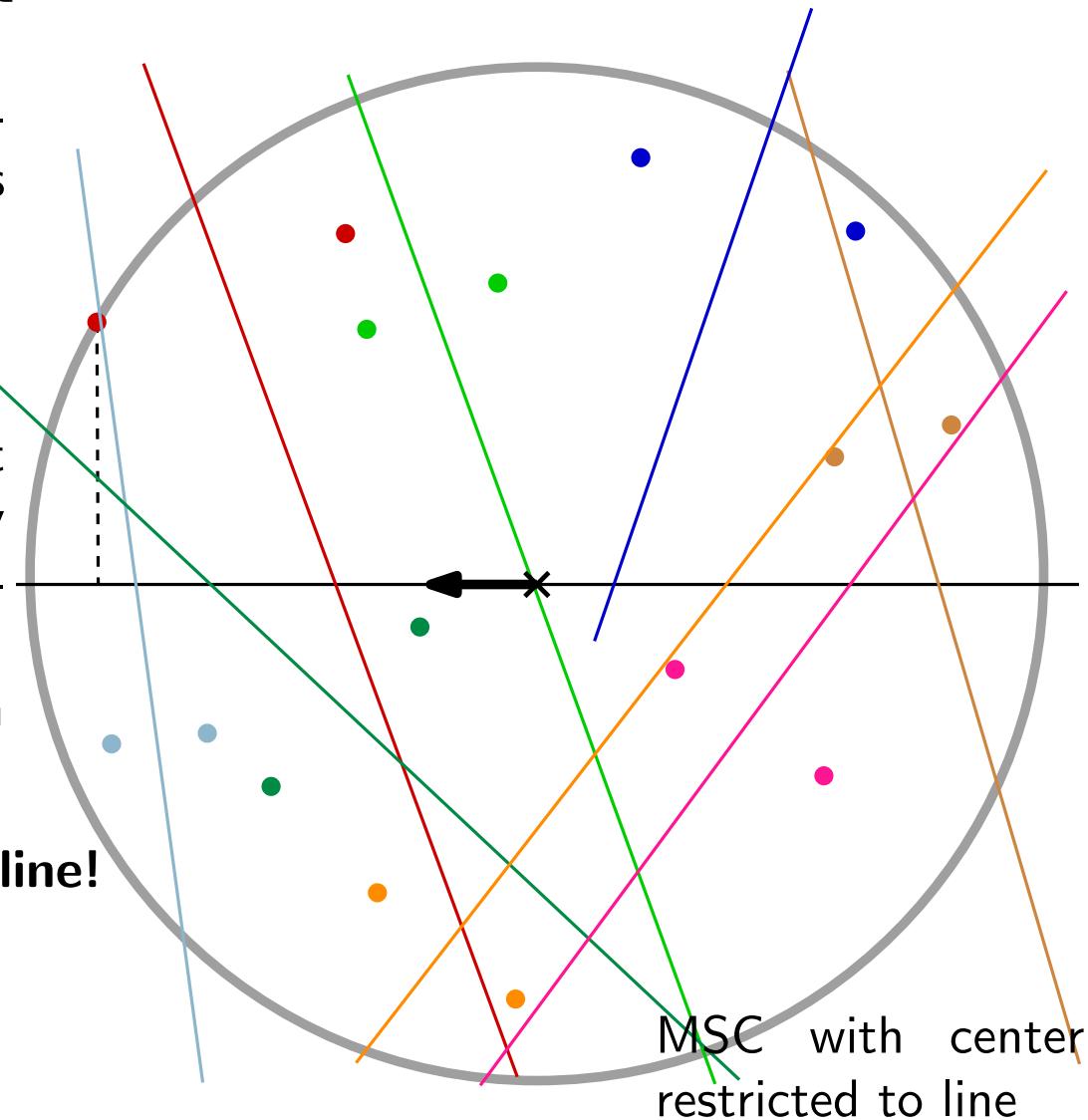
First we solve a subproblem: MSC restricted to line $y = 0$

Input: a set of n points (a_i, b_i) , $i = 1, \dots, n$.

Output: $\min_{x \in \mathbb{R}} \max_{i=1 \dots n} (x - a_i)^2 + (y - b_i)^2 = \min_{x \in \mathbb{R}} \max_{i=1 \dots n} (x - a_i)^2 + b_i^2$

1. Pair up the points. For each pair p_i, p_j , compute its perpendicular bisector b_{ij} and find its intersection x_{ij} with the line $y = 0$.
2. Compute x_m , the median value of the x_{ij} .
3. Search: Compute C_m , the MSC centered at x_m . If the points p_i lying in the boundary of C_m project orthogonally onto $y = 0$ to different sides of x_m , then x_m is the solution. If they all project onto the same side then search on that side.

Notice that the radius is unimodal along the line!



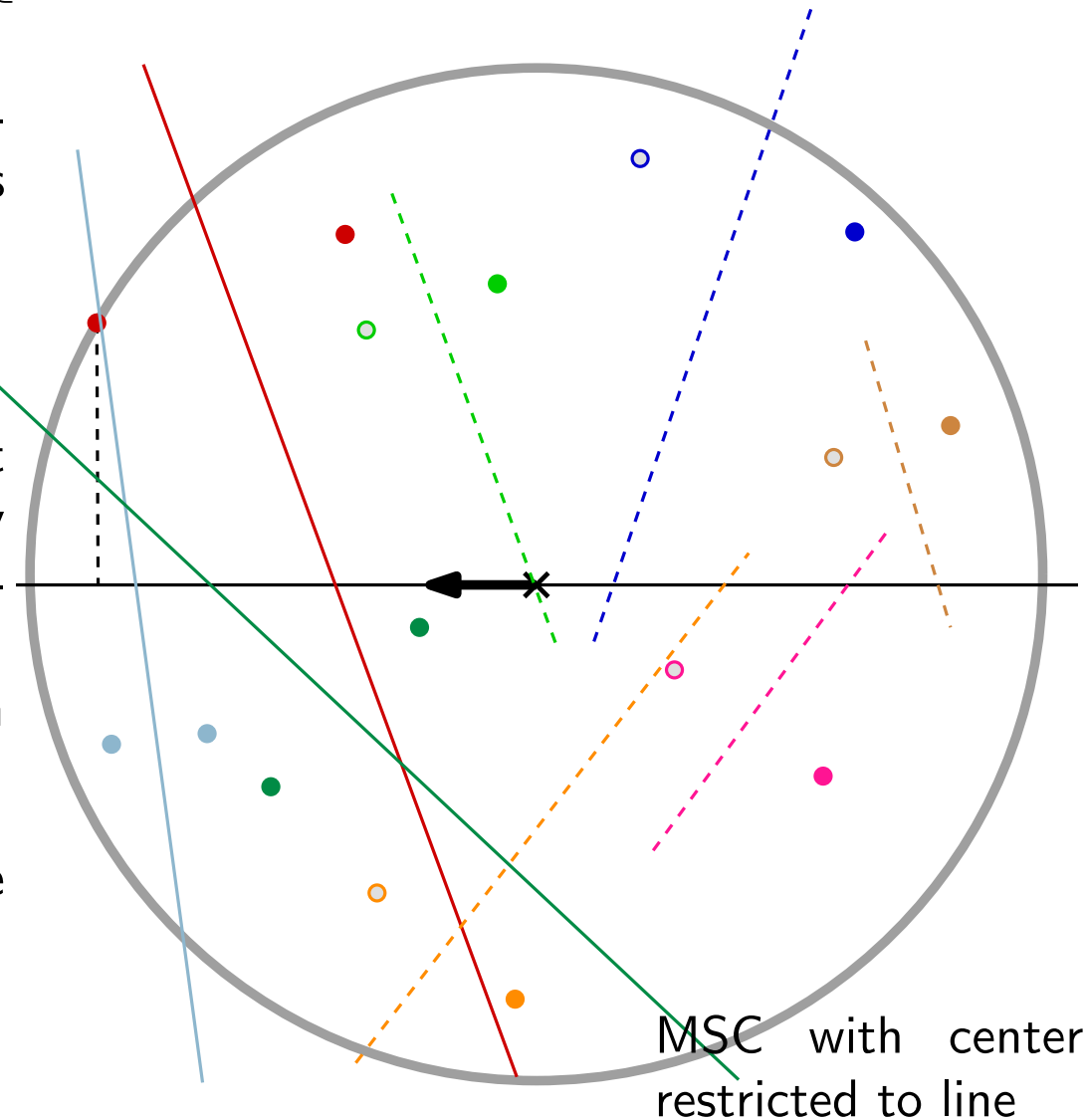
COMPUTING THE MINIMUM SPANNING CIRCLE

First we solve a subproblem: MSC restricted to line $y = 0$

Input: a set of n points (a_i, b_i) , $i = 1, \dots, n$.

Output: $\min_{x \in \mathbb{R}} \max_{i=1 \dots n} (x - a_i)^2 + (y - b_i)^2 = \min_{x \in \mathbb{R}} \max_{i=1 \dots n} (x - a_i)^2 + b_i^2$

1. Pair up the points. For each pair p_i, p_j , compute its perpendicular bisector b_{ij} and find its intersection x_{ij} with the line $y = 0$.
2. Compute x_m , the median value of the x_{ij} .
3. Search: Compute C_m , the MSC centered at x_m . If the points p_i lying in the boundary of C_m project orthogonally onto $y = 0$ to different sides of x_m , then x_m is the solution. If they all project onto the same side then search on that side.
4. Prune: For all x_{ij} located in the side opposite to the solution, eliminate the point p_i or p_j which is closest to x_m .



COMPUTING THE MINIMUM SPANNING CIRCLE

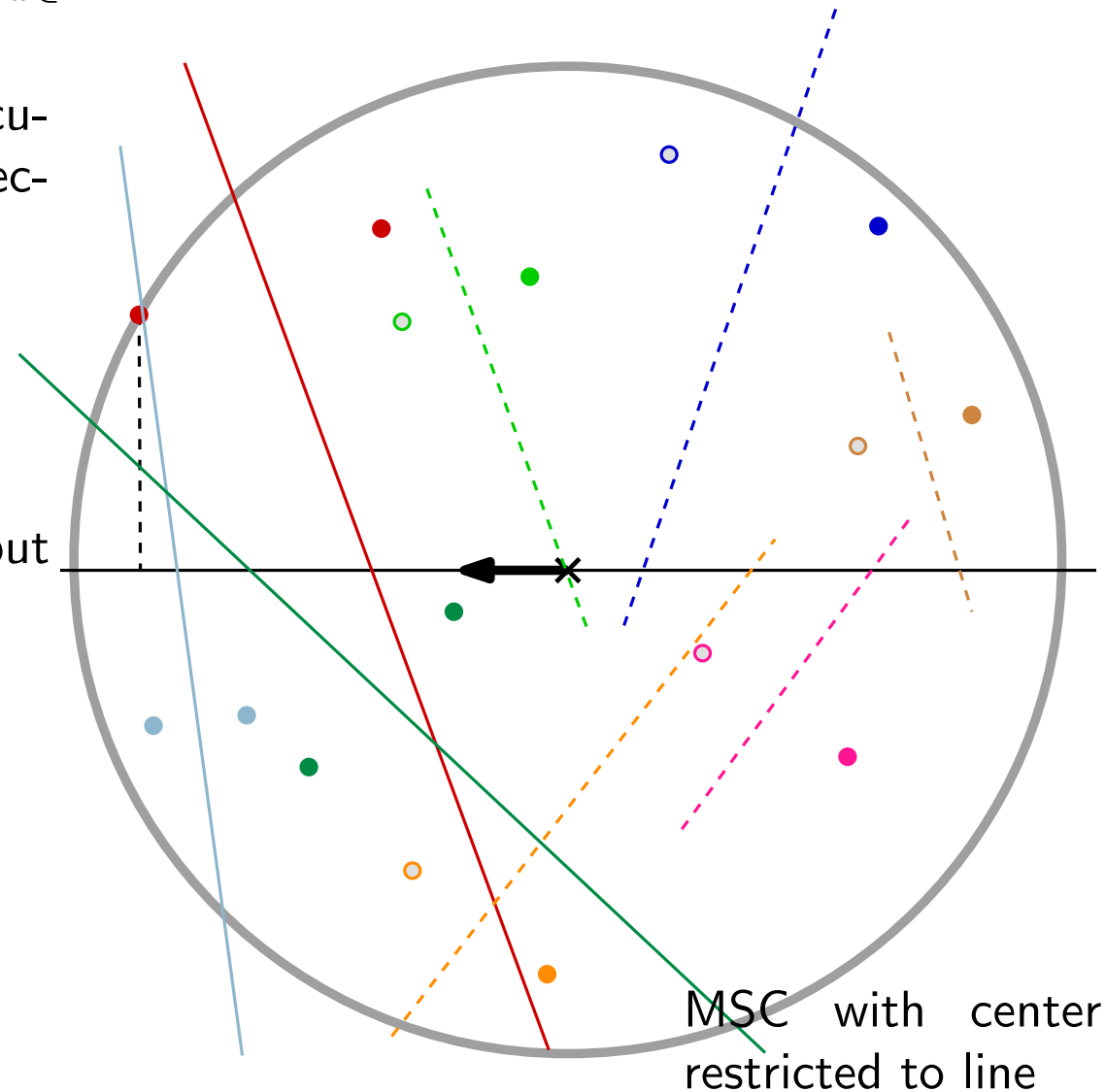
First we solve a subproblem: MSC restricted to line $y = 0$

Input: a set of n points (a_i, b_i) , $i = 1, \dots, n$.

Output: $\min_{x \in \mathbb{R}} \max_{i=1 \dots n} (x - a_i)^2 + (y - b_i)^2 = \min_{x \in \mathbb{R}} \max_{i=1 \dots n} (x - a_i)^2 + b_i^2$

1. Pair up the points, compute the perpendicular bisector of each pair, and find its intersection with the line $y = 0$. $O(n)$ time
2. Compute a median value. $O(n)$ time
3. Search: $O(n)$ time
4. Prune: In $O(n)$ time, at least $1/4$ of the input is pruned.

The problem is solved in $O(n)$ time.

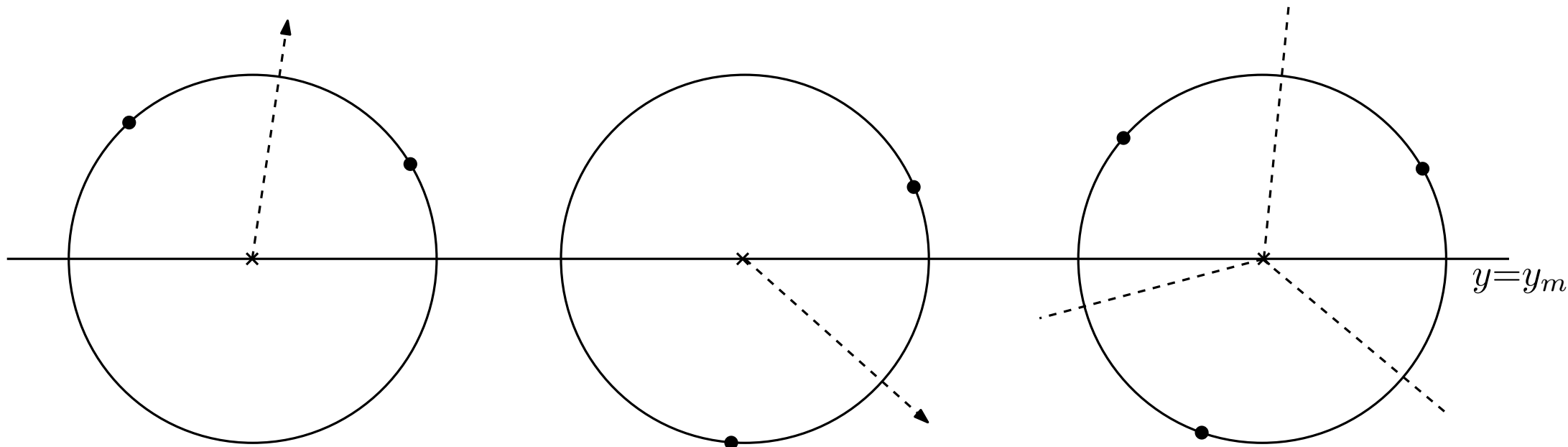


COMPUTING THE MINIMUM SPANNING CIRCLE

Why is the restricted version useful?

We can use it to find out where the unrestricted solution is!

We will use the solution to the restricted subproblem to decide whether the unrestricted solution has been found, or if it lies above/below the horizontal line $y = y_m$



The solution
lies above

The solution
lies below

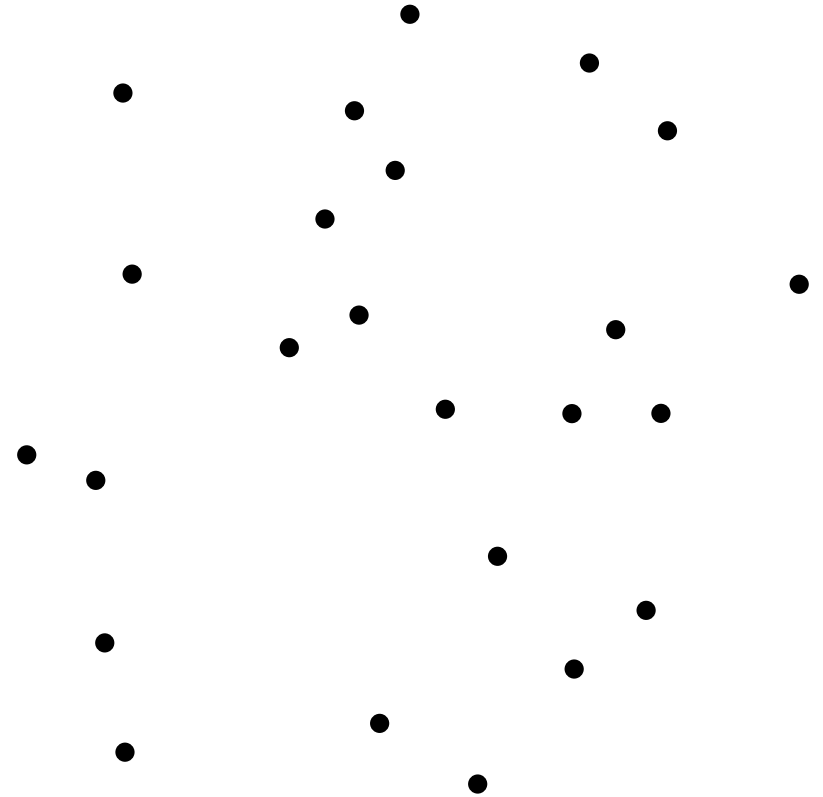
The solution
has been found

COMPUTING THE MINIMUM SPANNING CIRCLE

MSC in the plane

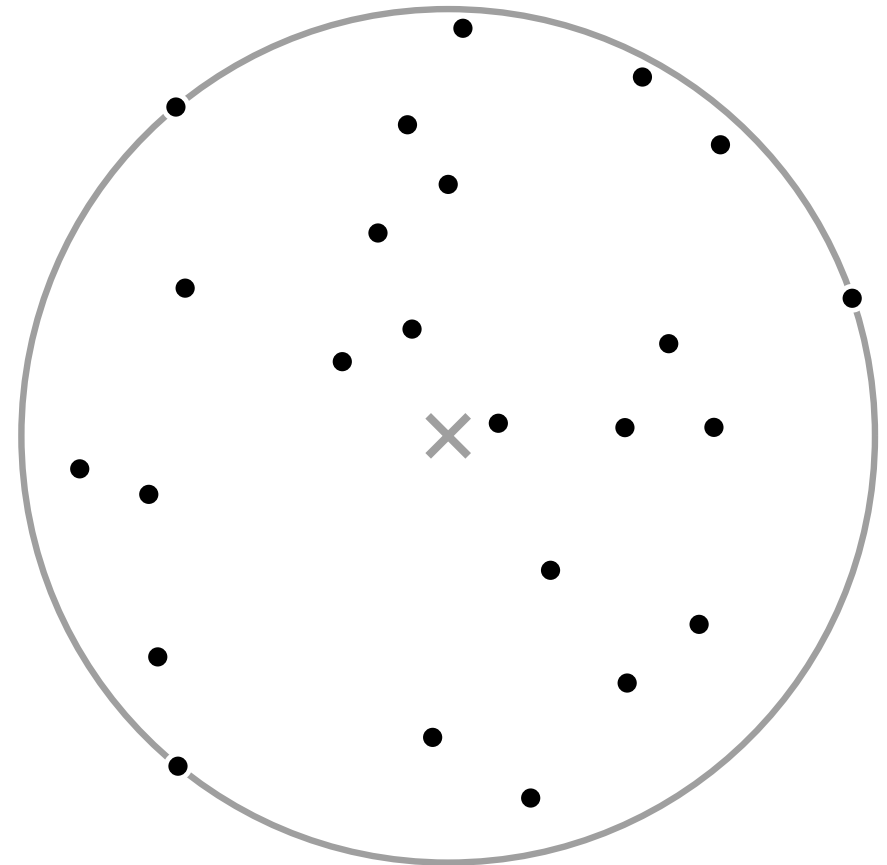
COMPUTING THE MINIMUM SPANNING CIRCLE

MSC in the plane



COMPUTING THE MINIMUM SPANNING CIRCLE

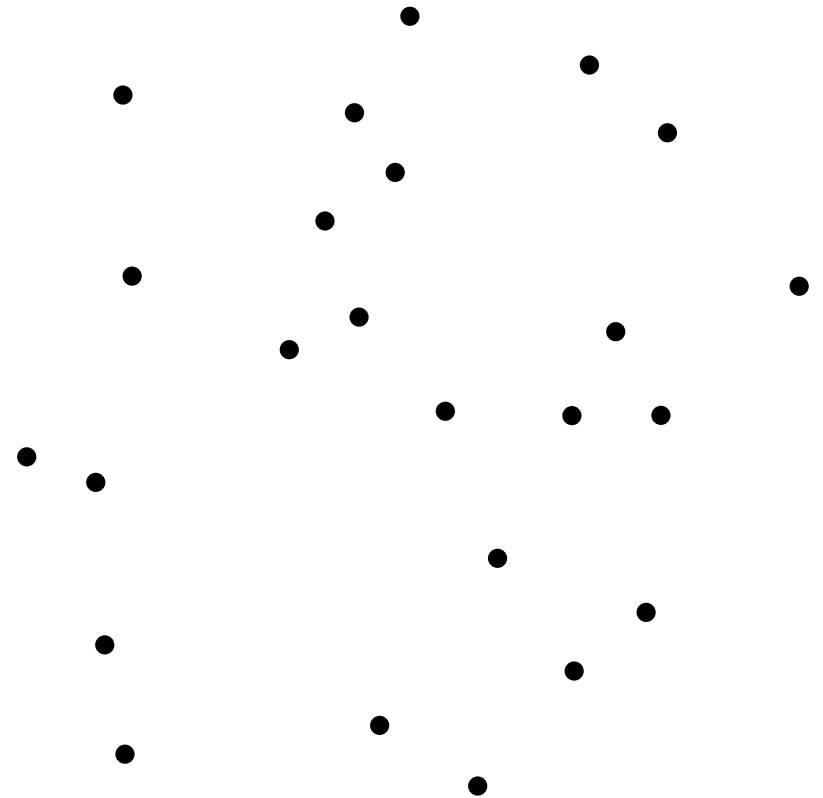
MSC in the plane



COMPUTING THE MINIMUM SPANNING CIRCLE

MSC in the plane

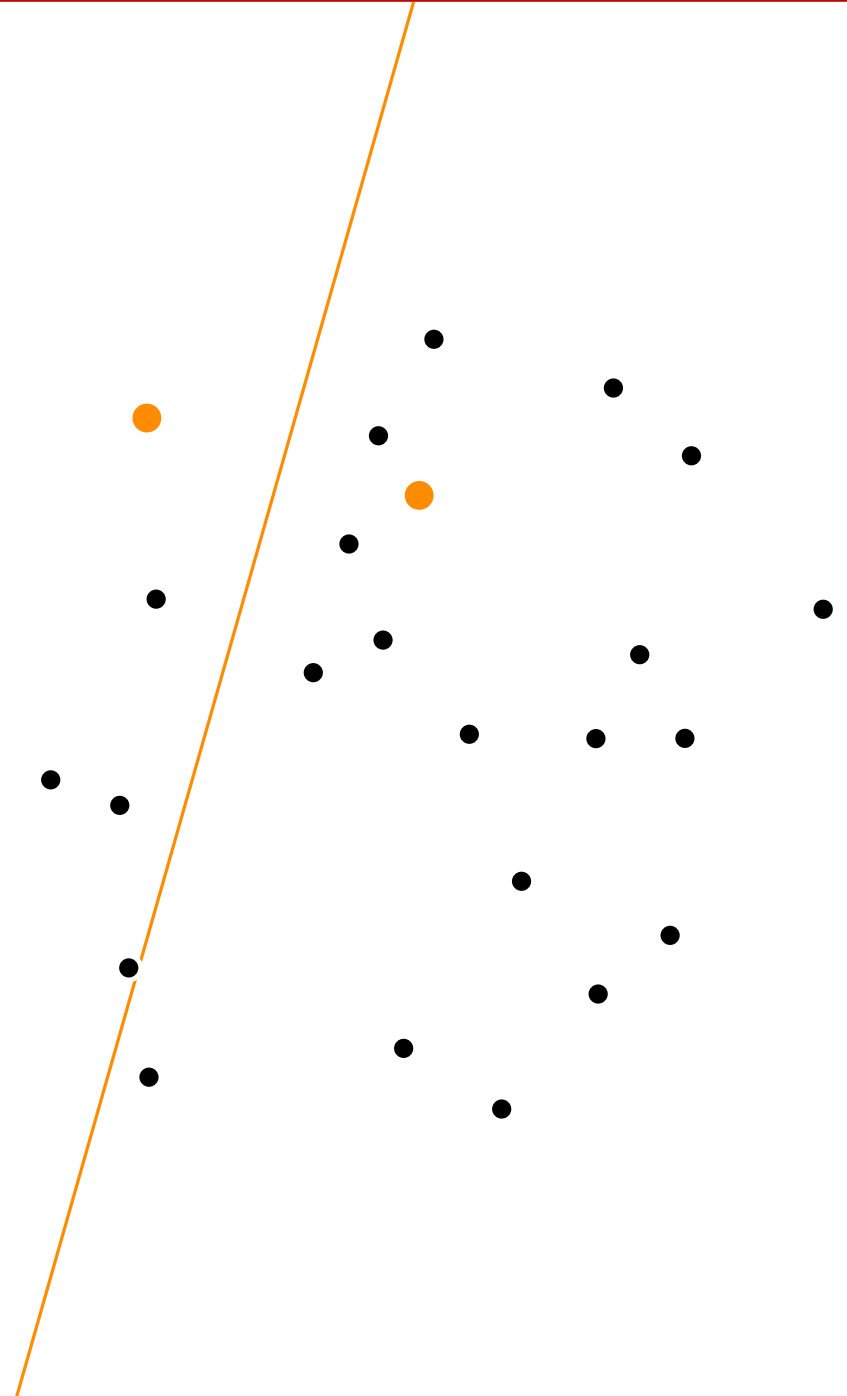
1. Pair up the points, and compute the perpendicular bisector of each pair.



COMPUTING THE MINIMUM SPANNING CIRCLE

MSC in the plane

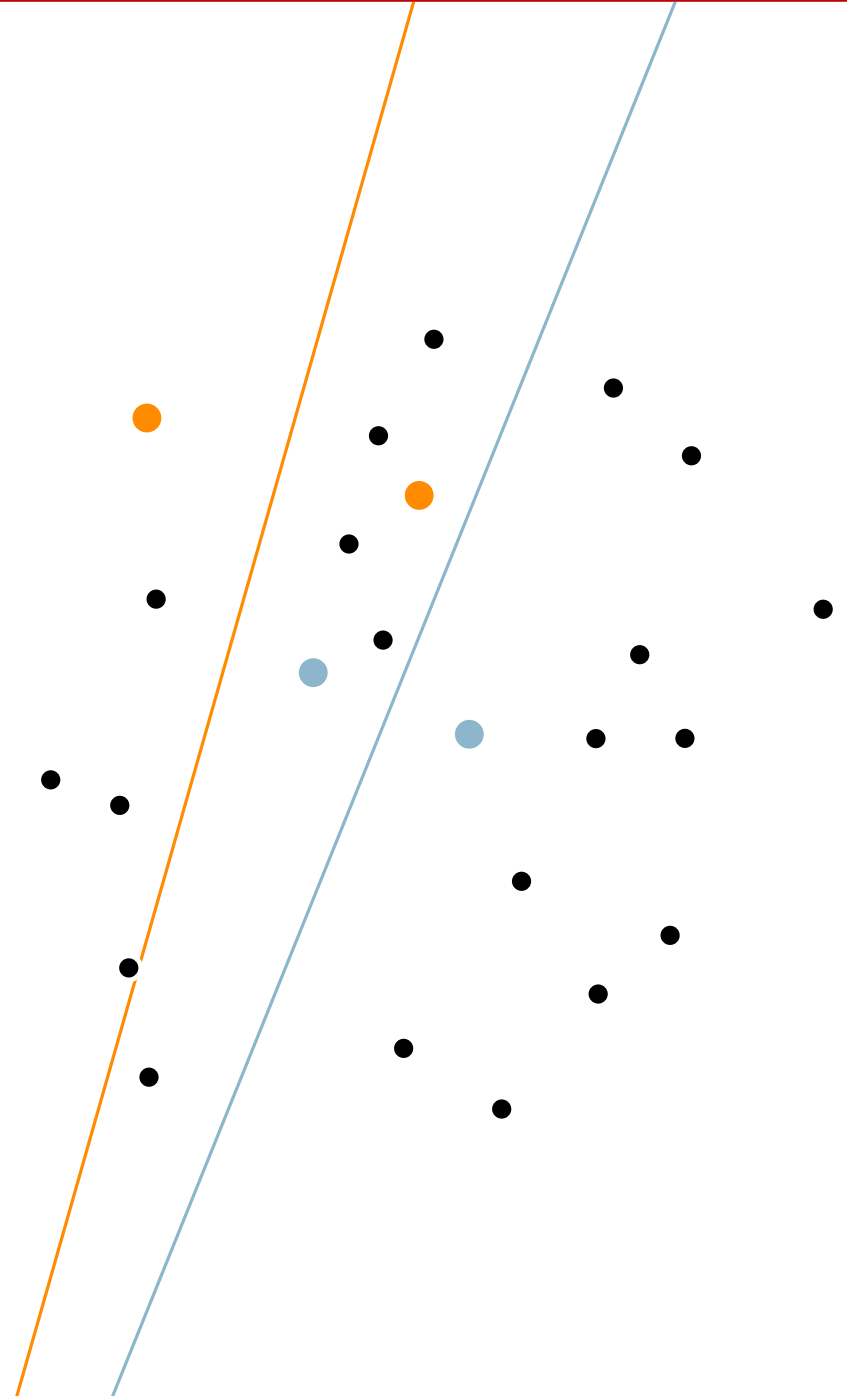
1. Pair up the points, and compute the perpendicular bisector of each pair.



COMPUTING THE MINIMUM SPANNING CIRCLE

MSC in the plane

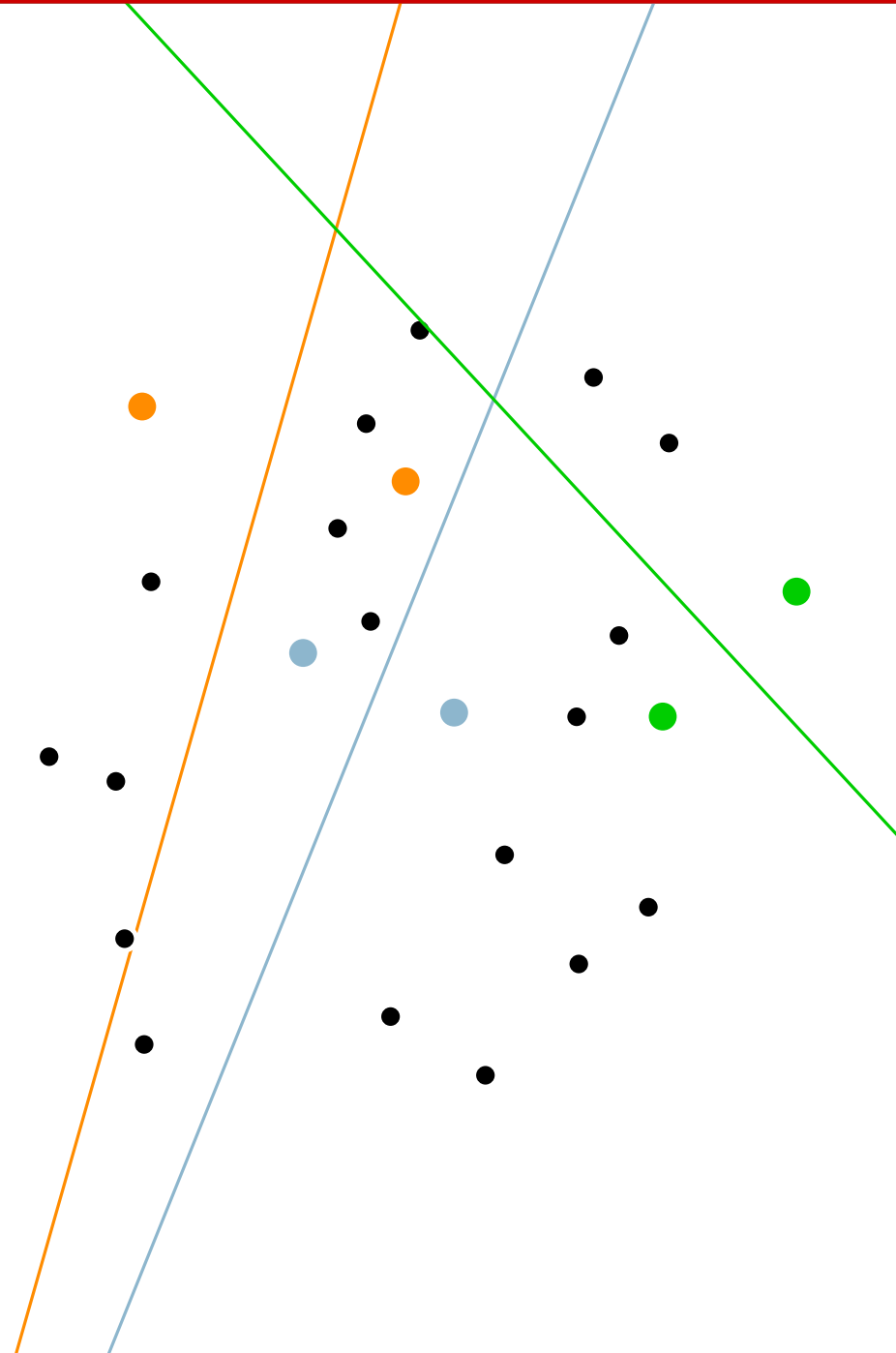
1. Pair up the points, and compute the perpendicular bisector of each pair.



COMPUTING THE MINIMUM SPANNING CIRCLE

MSC in the plane

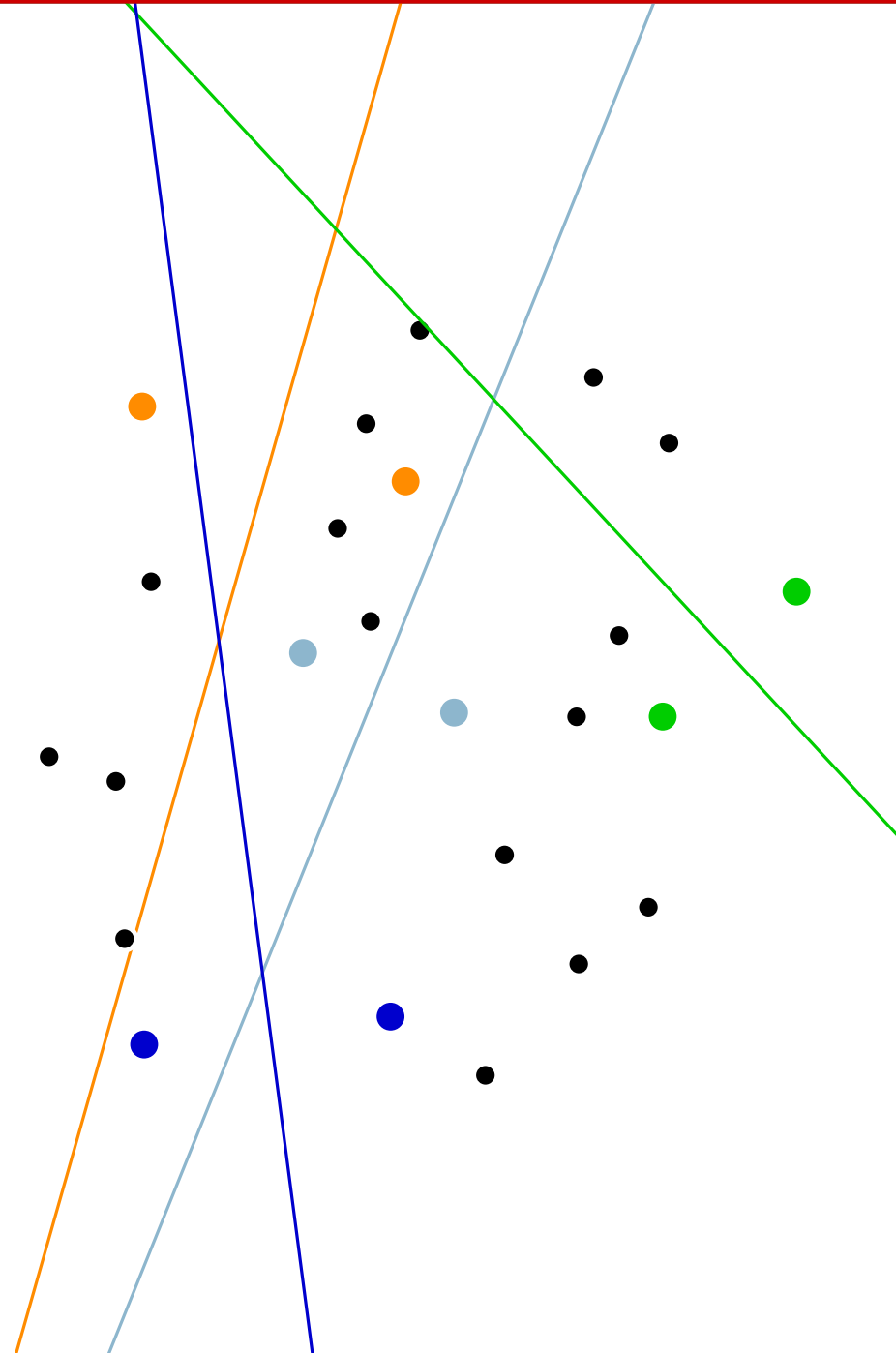
1. Pair up the points, and compute the perpendicular bisector of each pair.



COMPUTING THE MINIMUM SPANNING CIRCLE

MSC in the plane

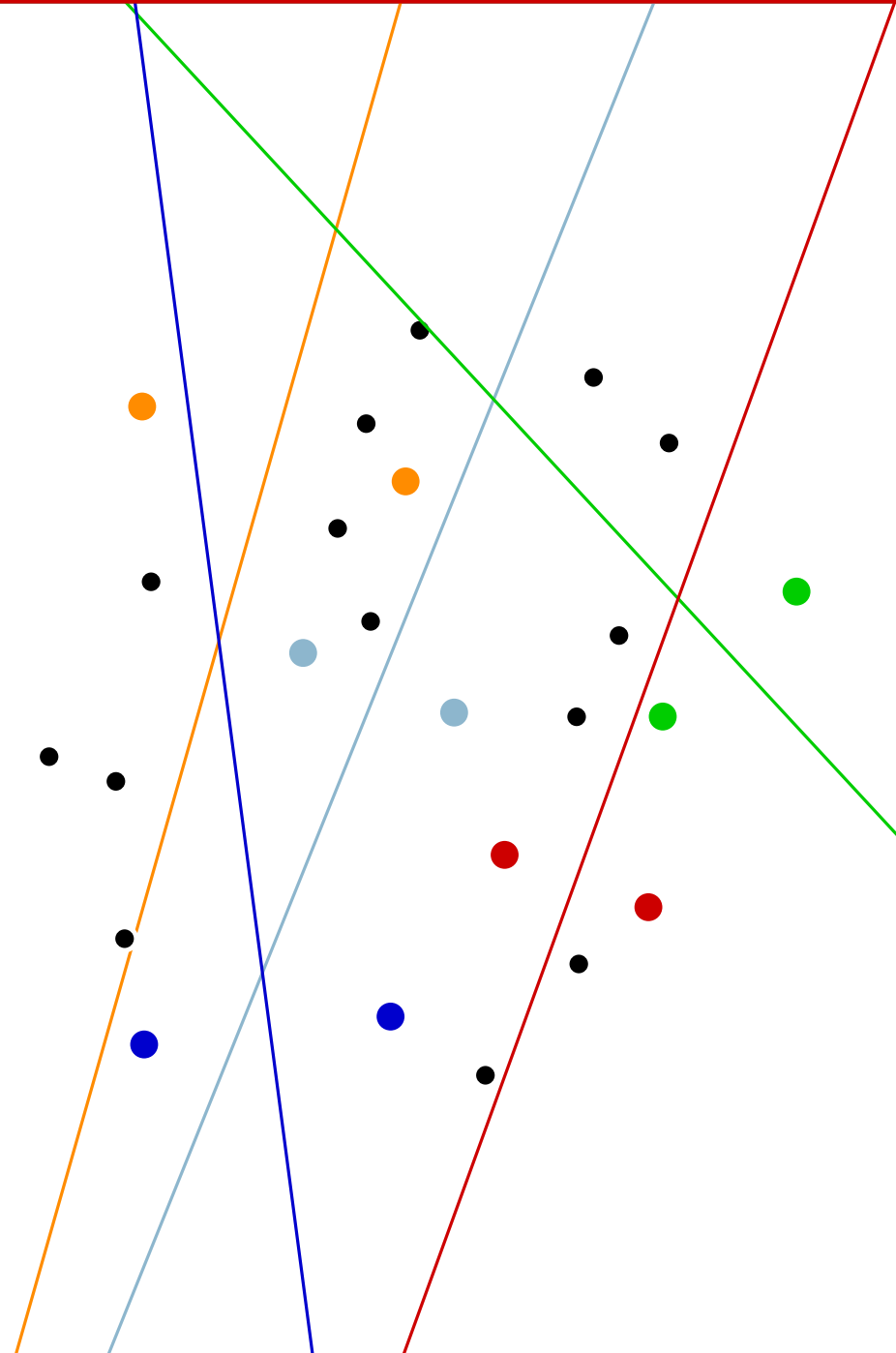
1. Pair up the points, and compute the perpendicular bisector of each pair.



COMPUTING THE MINIMUM SPANNING CIRCLE

MSC in the plane

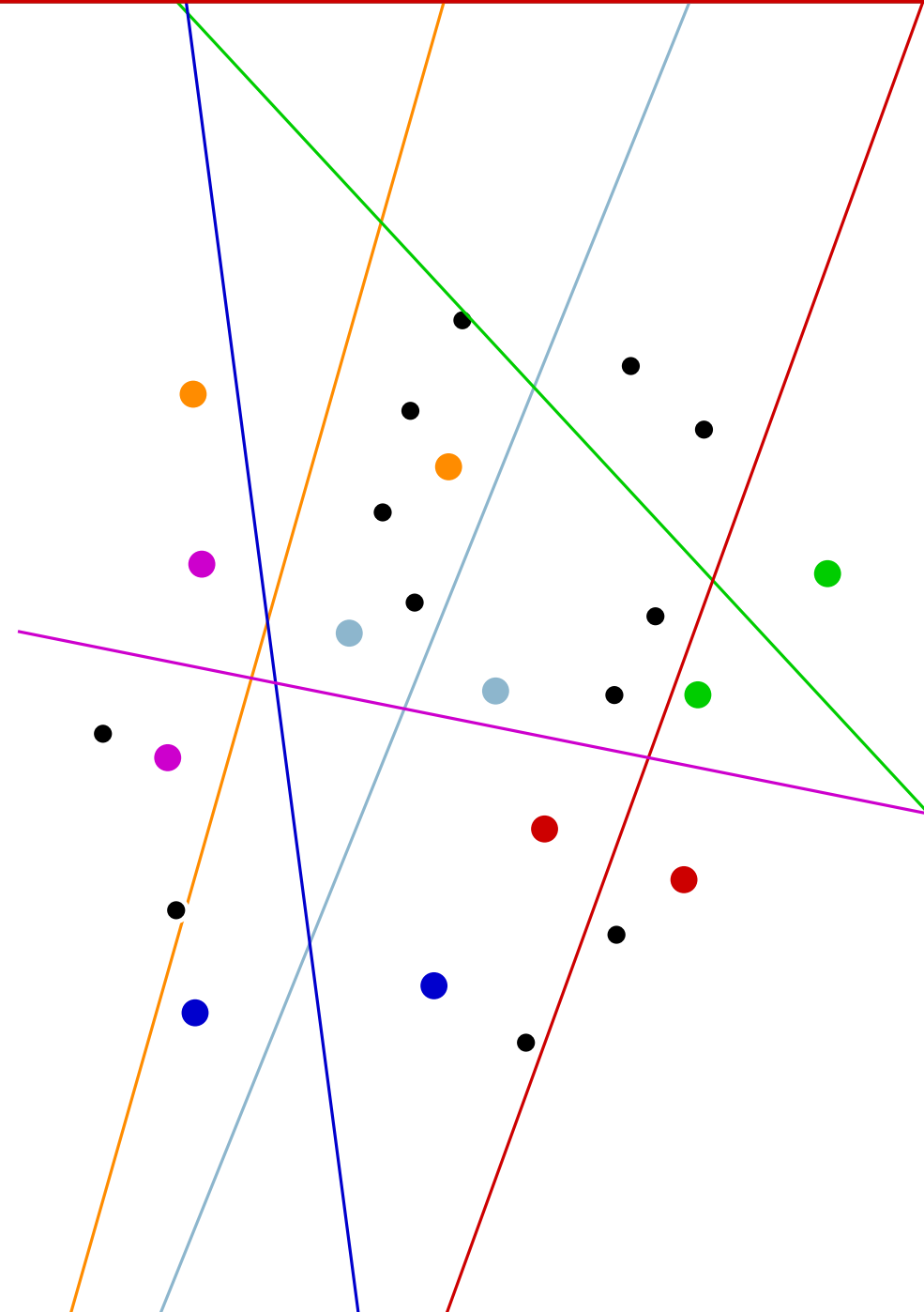
1. Pair up the points, and compute the perpendicular bisector of each pair.



COMPUTING THE MINIMUM SPANNING CIRCLE

MSC in the plane

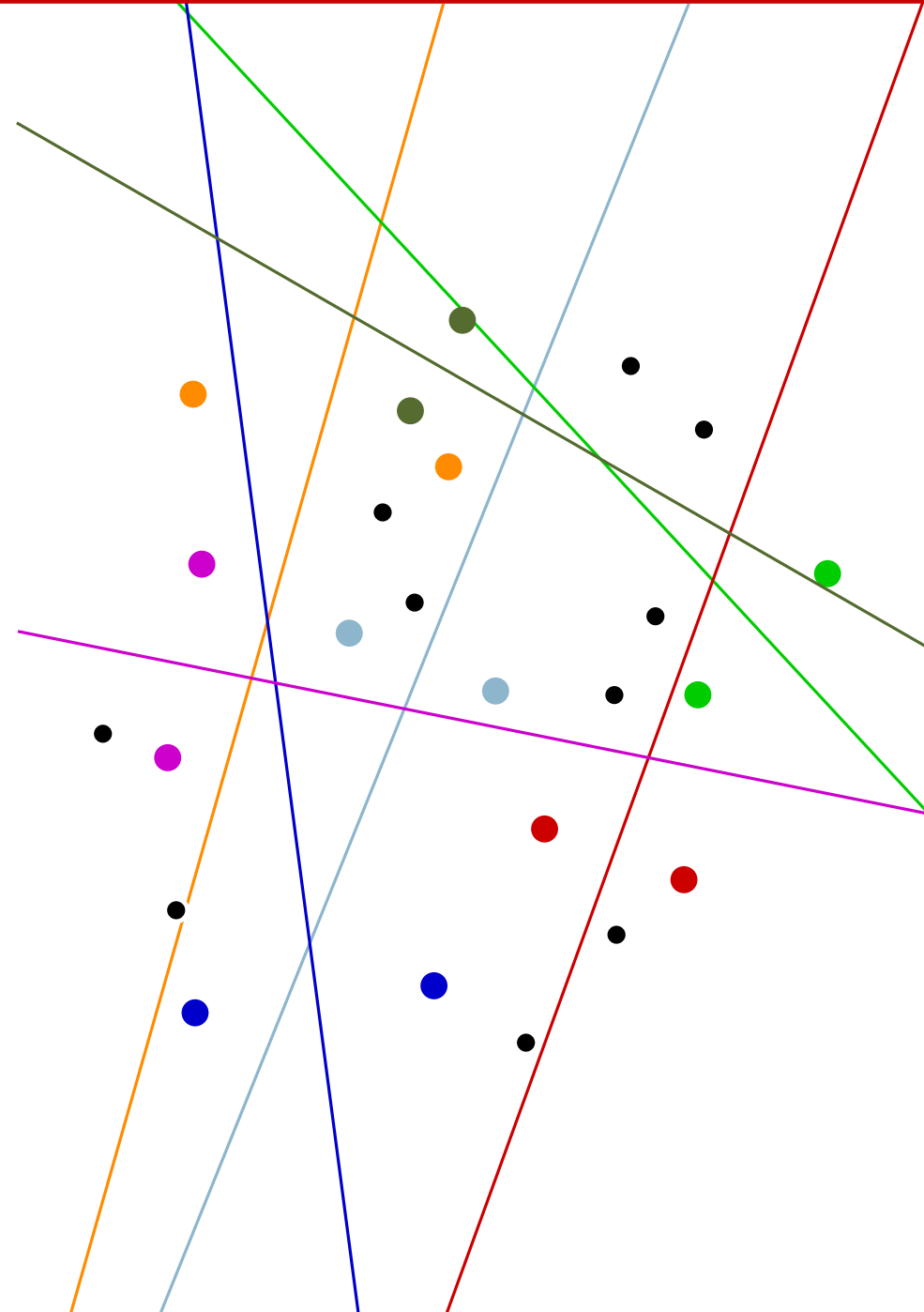
1. Pair up the points, and compute the perpendicular bisector of each pair.



COMPUTING THE MINIMUM SPANNING CIRCLE

MSC in the plane

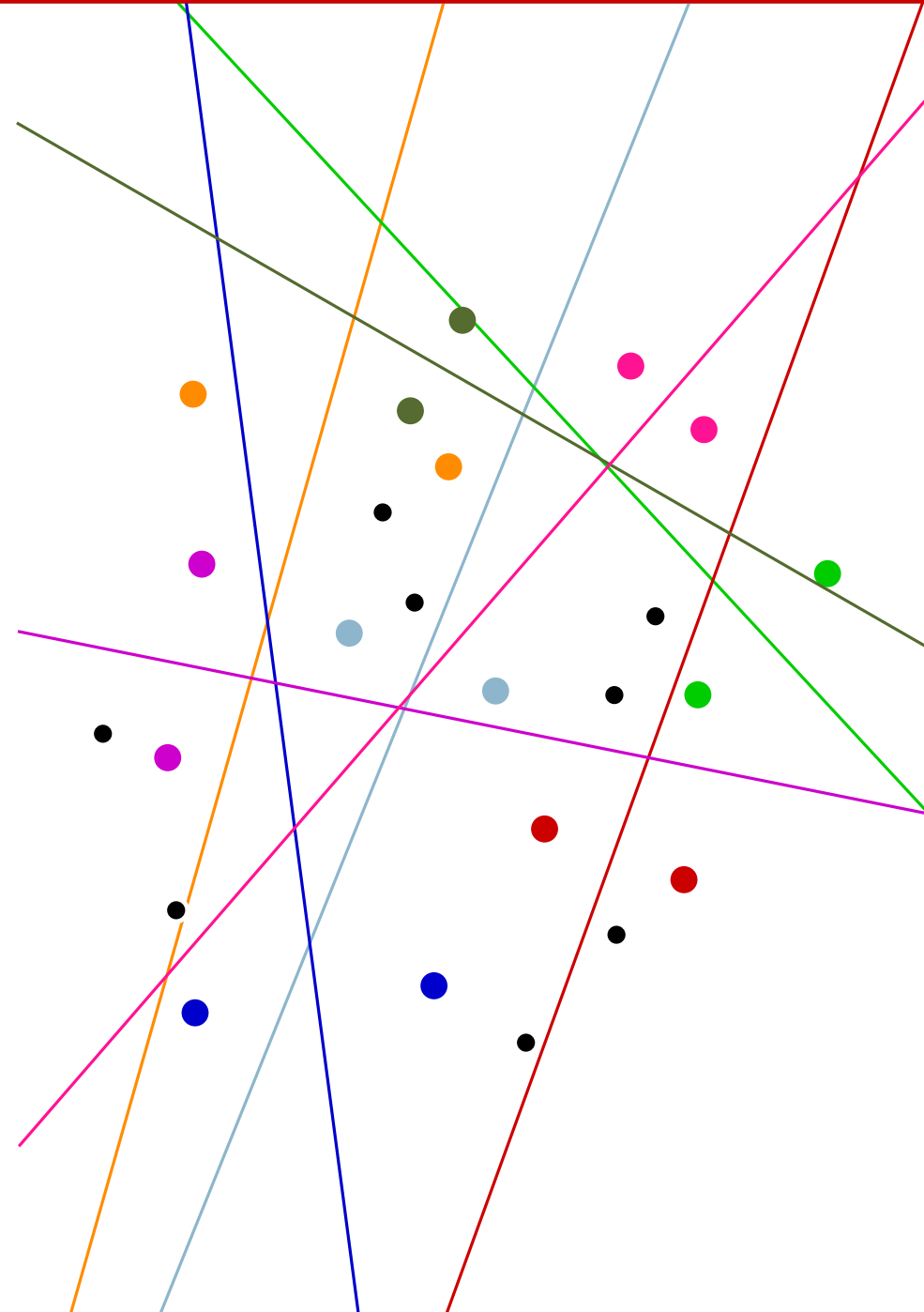
1. Pair up the points, and compute the perpendicular bisector of each pair.



COMPUTING THE MINIMUM SPANNING CIRCLE

MSC in the plane

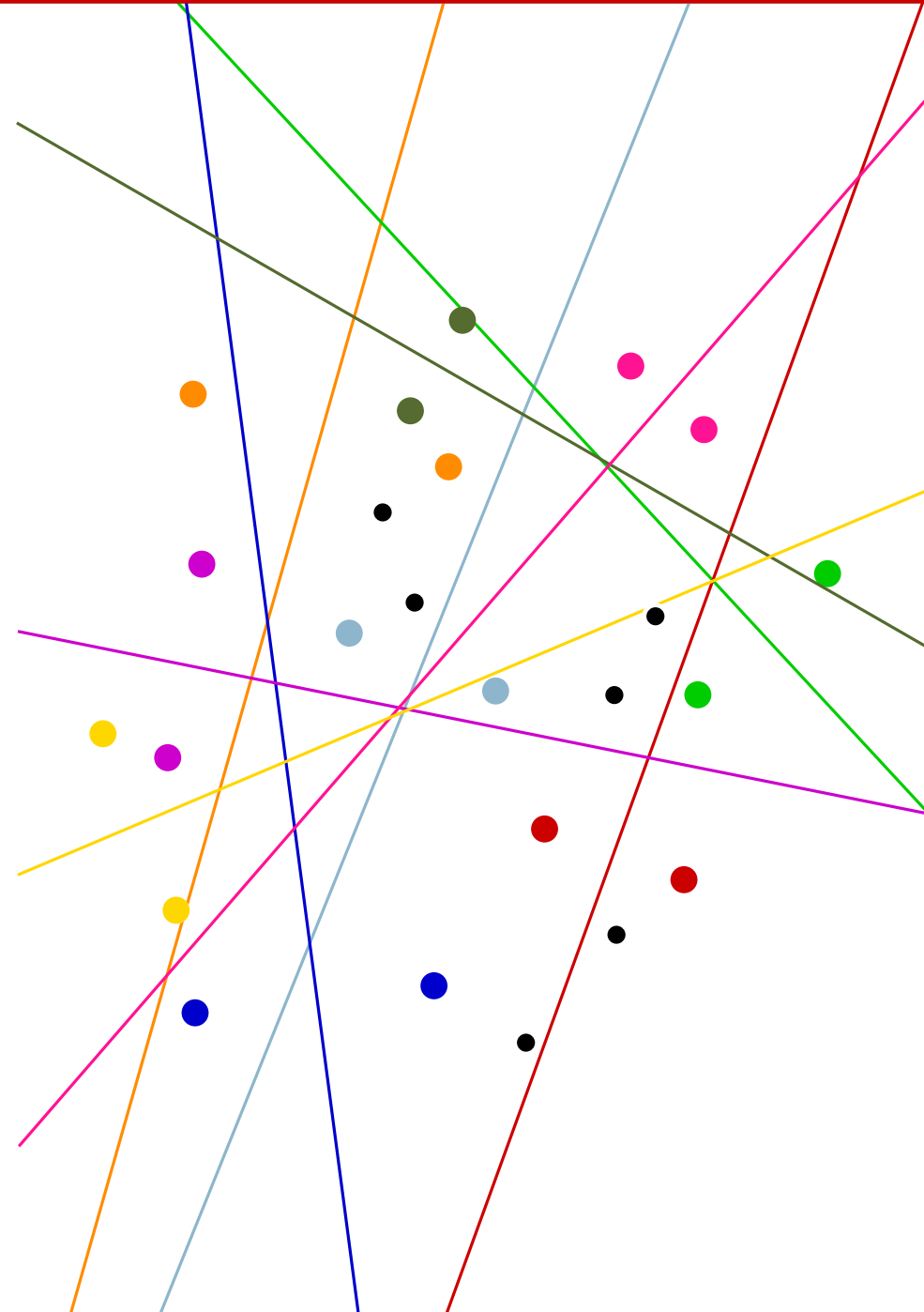
1. Pair up the points, and compute the perpendicular bisector of each pair.



COMPUTING THE MINIMUM SPANNING CIRCLE

MSC in the plane

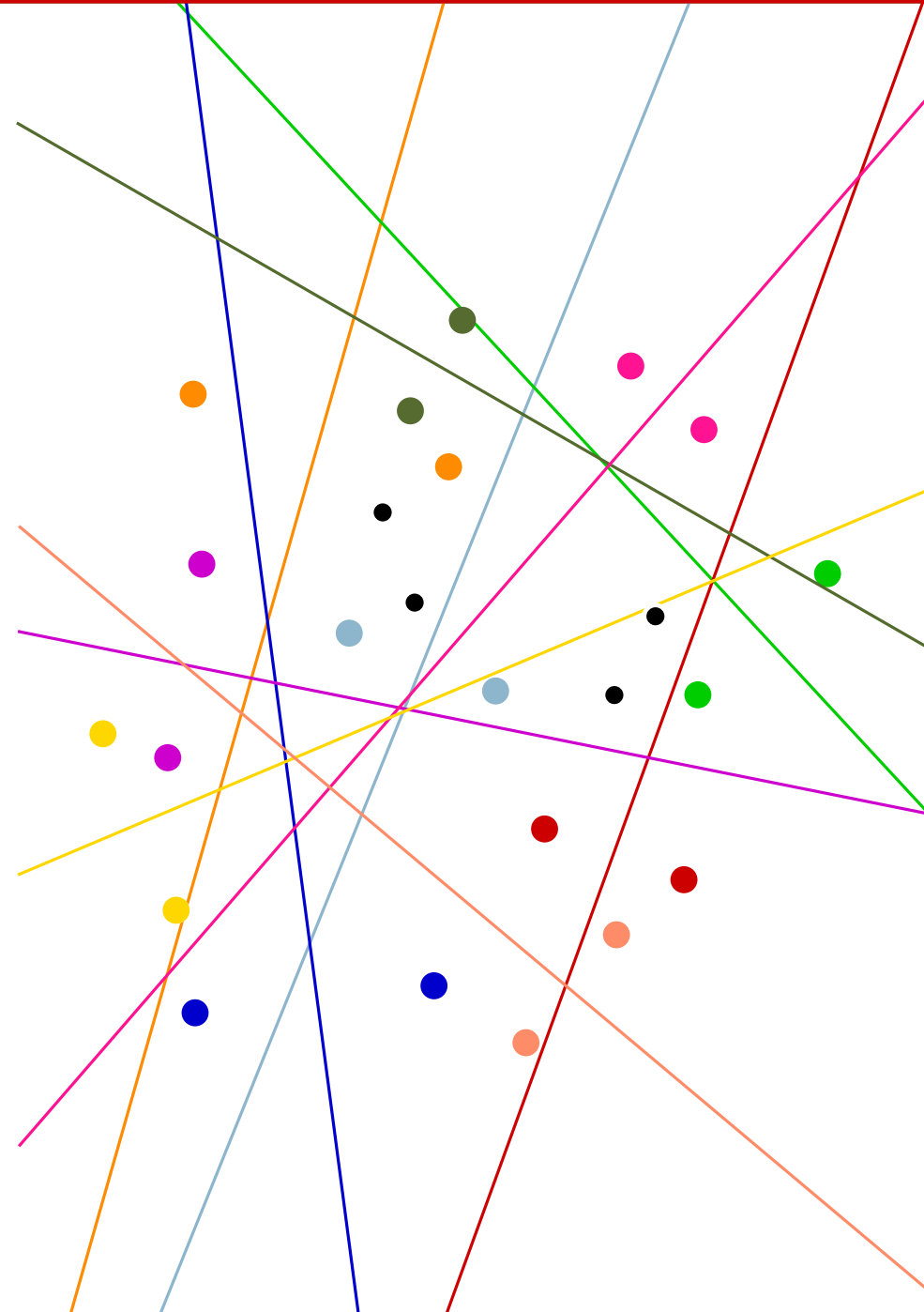
1. Pair up the points, and compute the perpendicular bisector of each pair.



COMPUTING THE MINIMUM SPANNING CIRCLE

MSC in the plane

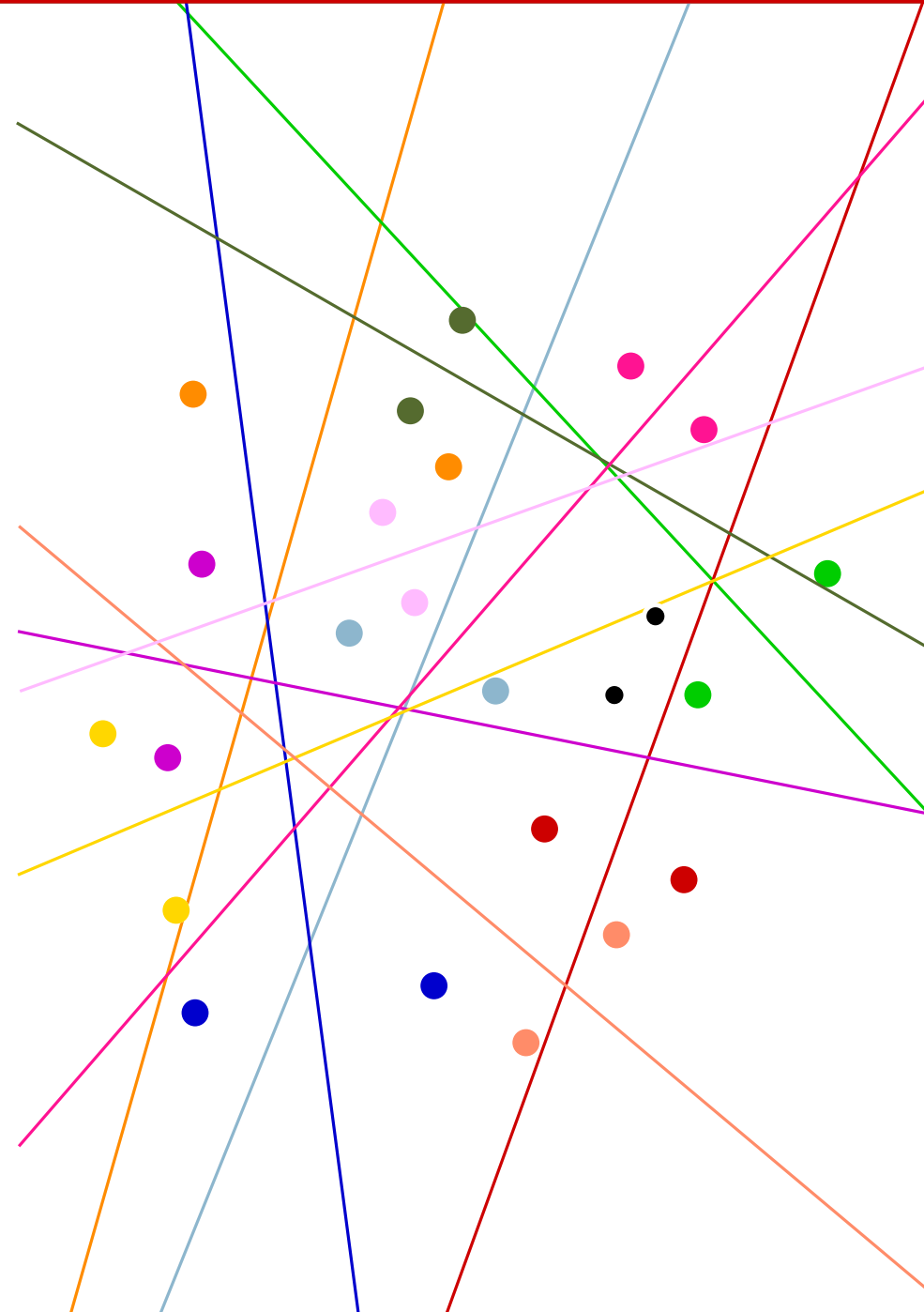
1. Pair up the points, and compute the perpendicular bisector of each pair.



COMPUTING THE MINIMUM SPANNING CIRCLE

MSC in the plane

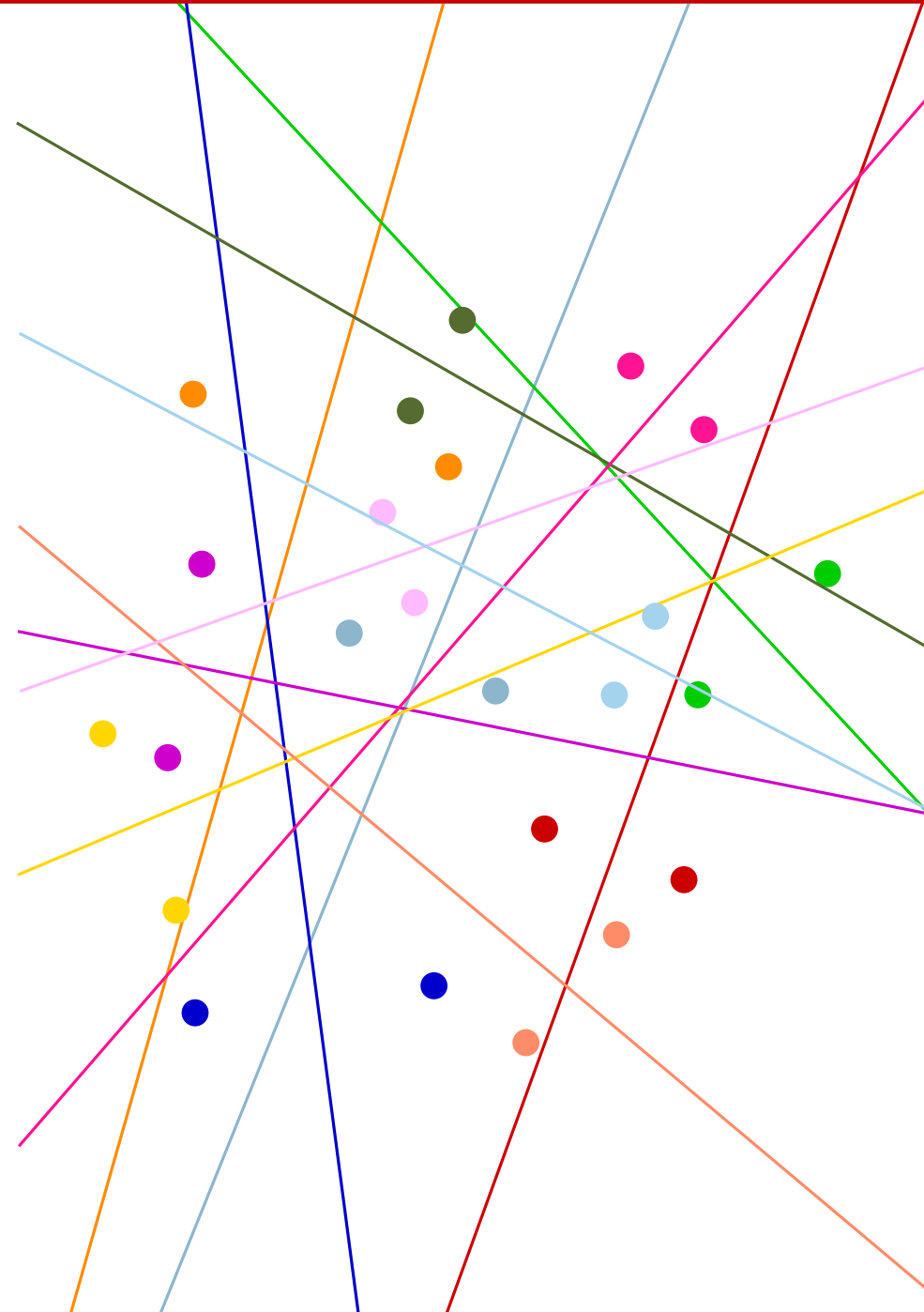
1. Pair up the points, and compute the perpendicular bisector of each pair.



COMPUTING THE MINIMUM SPANNING CIRCLE

MSC in the plane

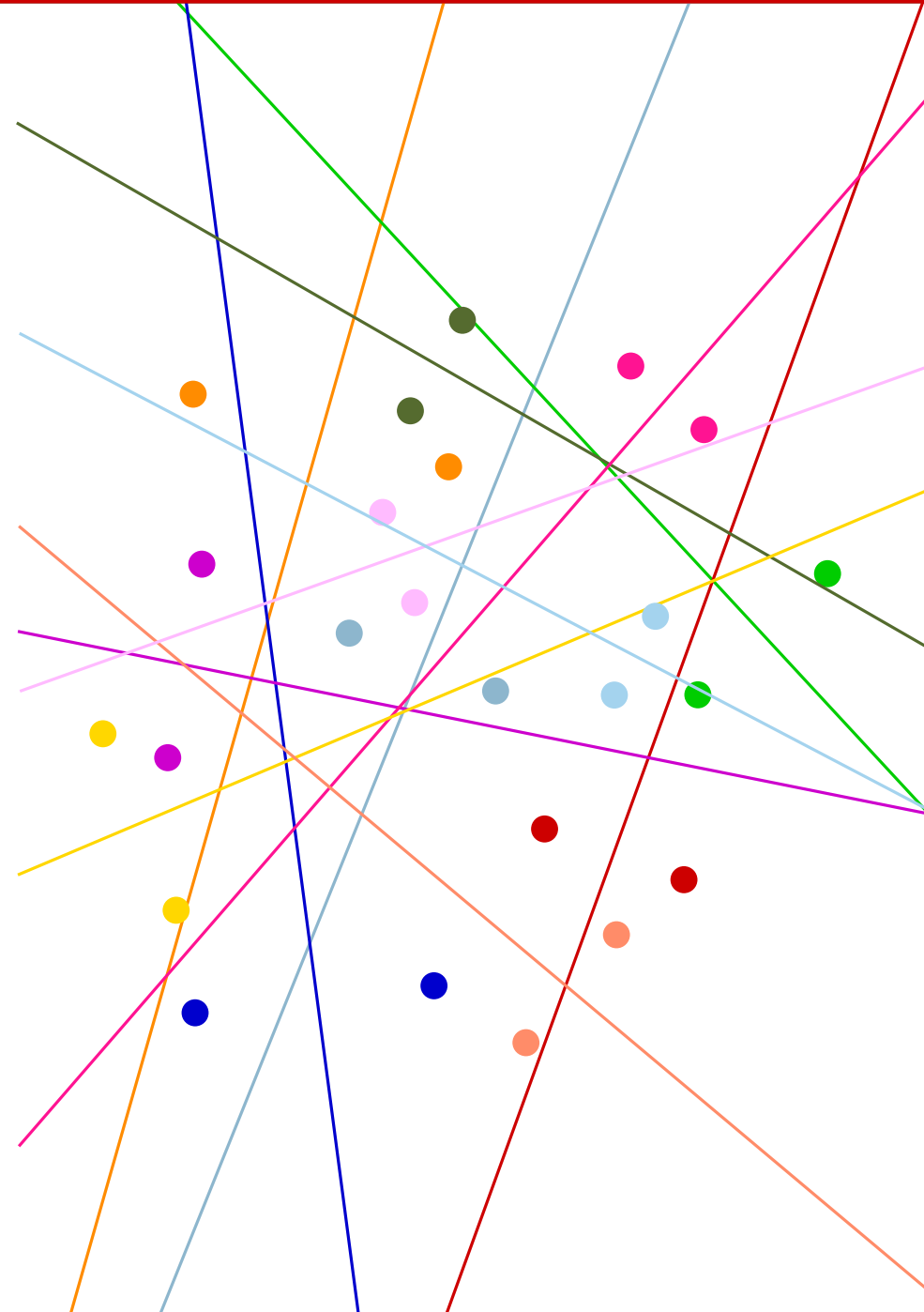
1. Pair up the points, and compute the perpendicular bisector of each pair.



COMPUTING THE MINIMUM SPANNING CIRCLE

MSC in the plane

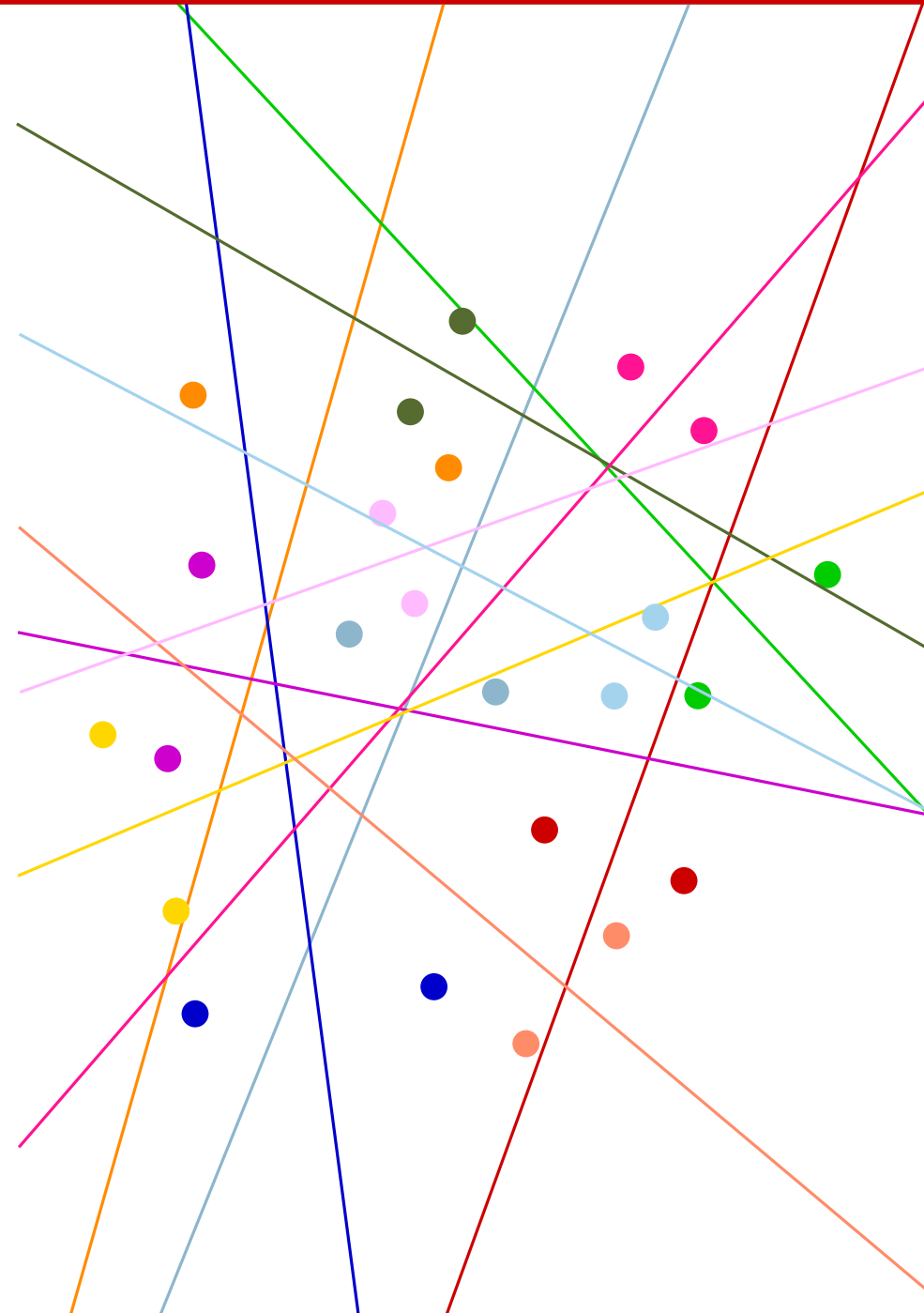
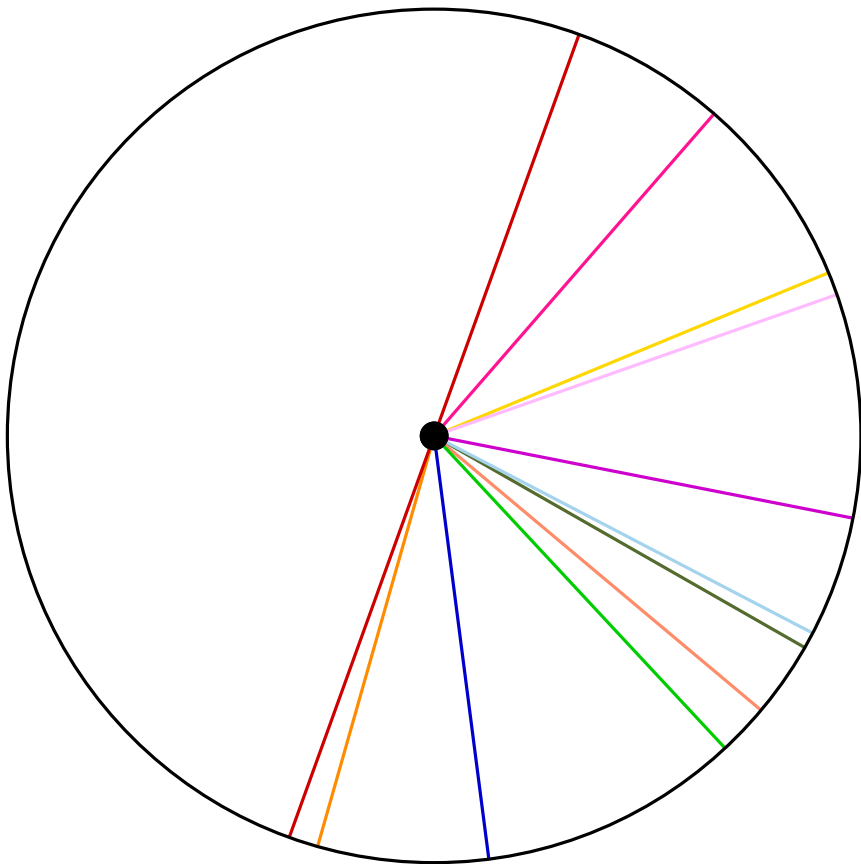
1. Pair up the points, and compute the perpendicular bisector of each pair.
2. Compute the median slope of the non-vertical bisectors, and take it as horizontal.



COMPUTING THE MINIMUM SPANNING CIRCLE

MSC in the plane

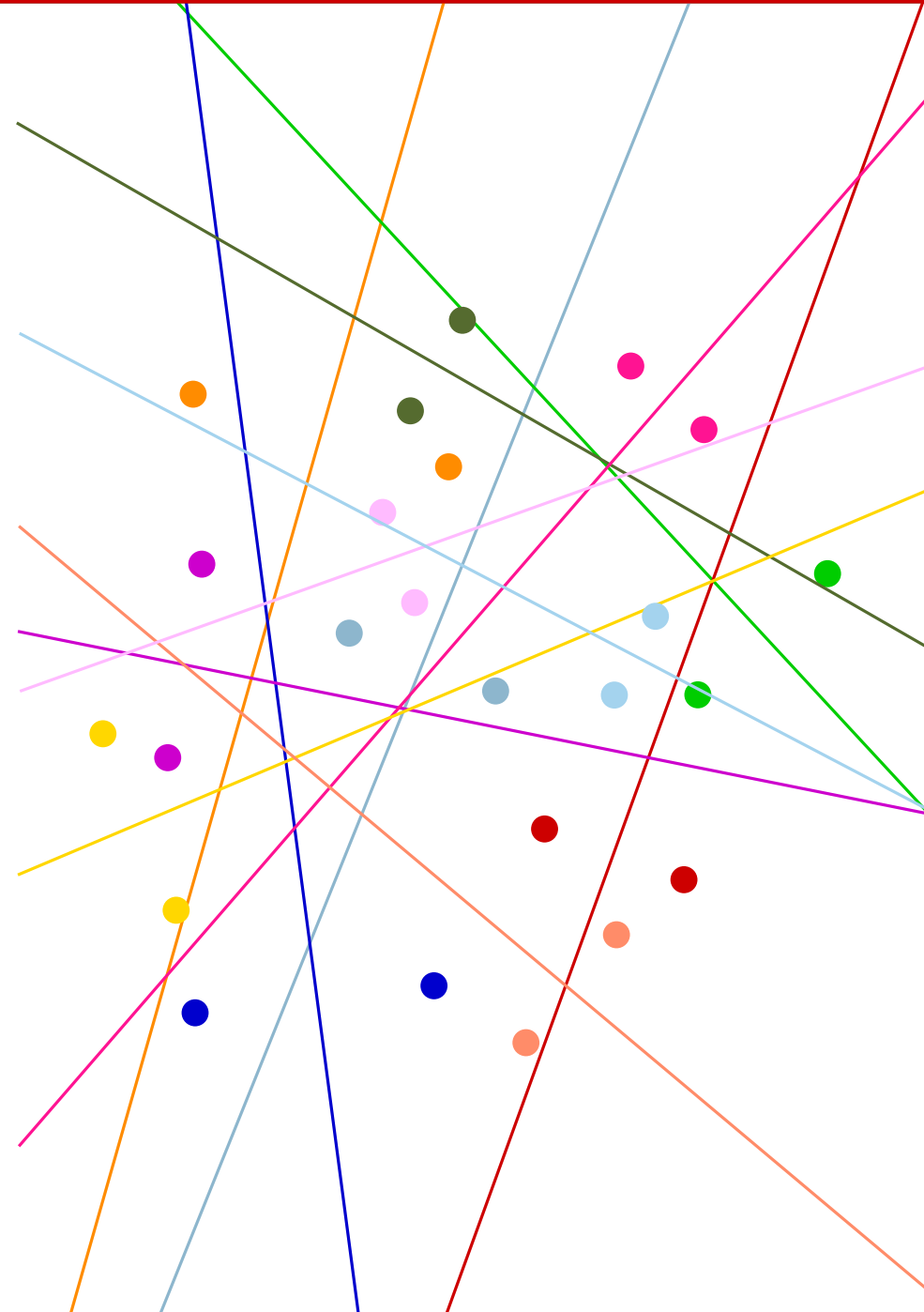
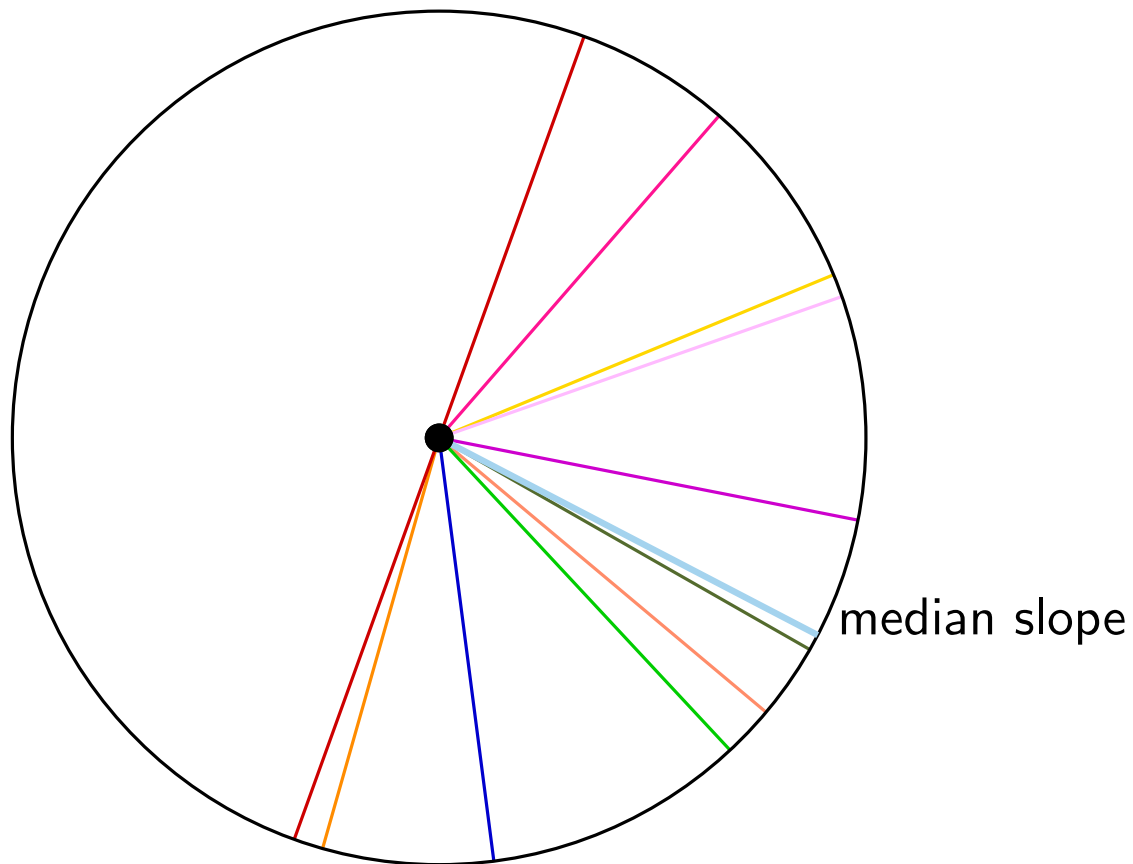
1. Pair up the points, and compute the perpendicular bisector of each pair.
2. Compute the median slope of the non-vertical bisectors, and take it as horizontal.



COMPUTING THE MINIMUM SPANNING CIRCLE

MSC in the plane

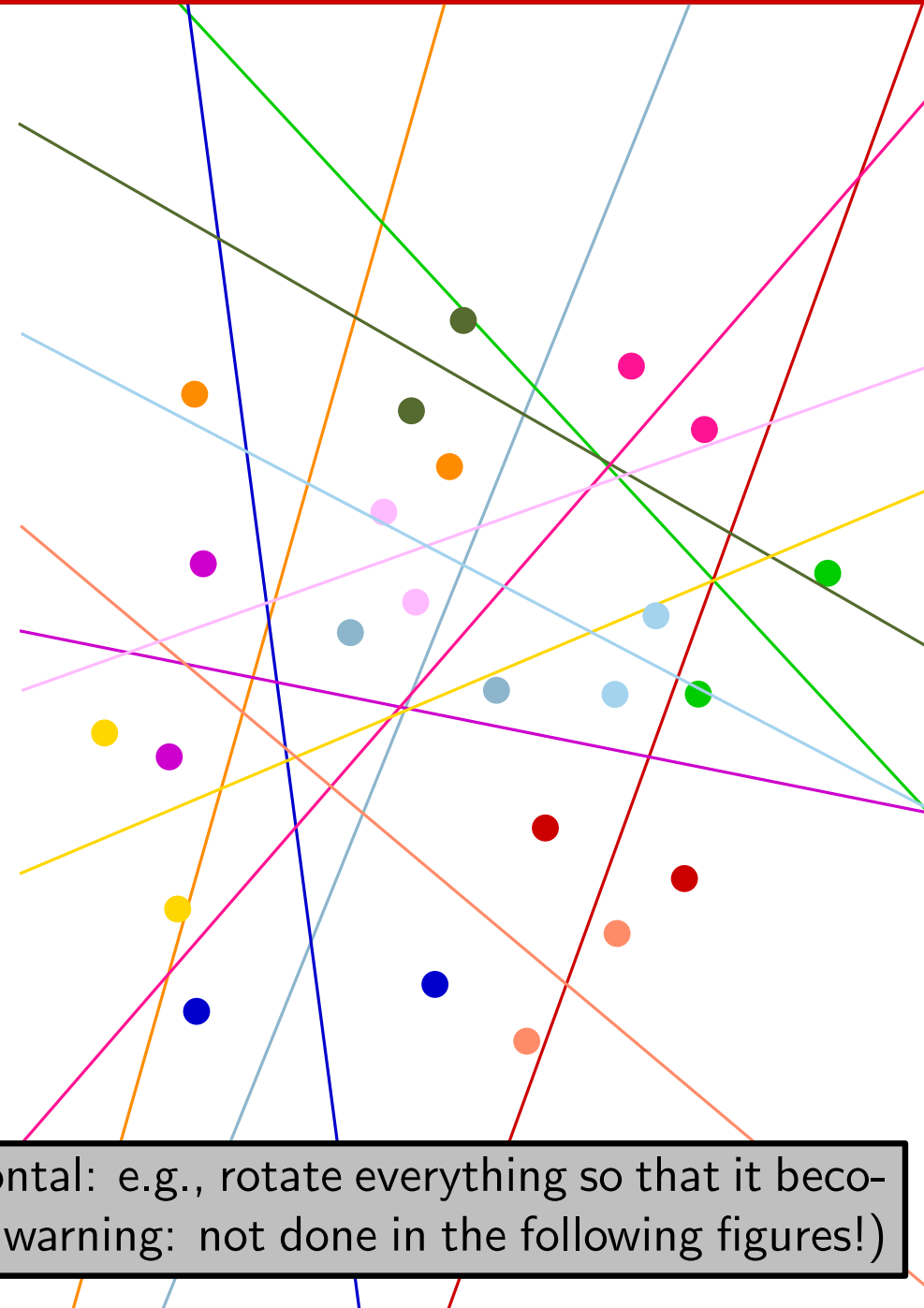
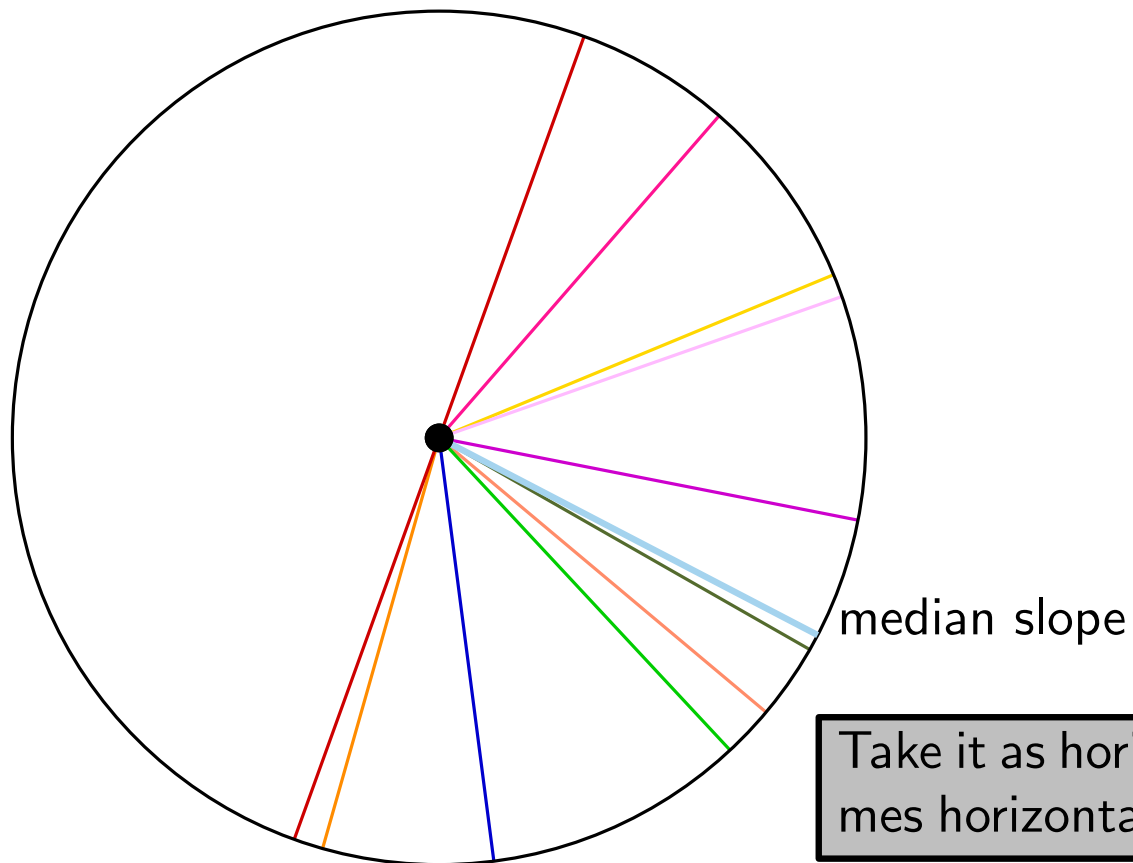
1. Pair up the points, and compute the perpendicular bisector of each pair.
2. Compute the median slope of the non-vertical bisectors, and take it as horizontal.



COMPUTING THE MINIMUM SPANNING CIRCLE

MSC in the plane

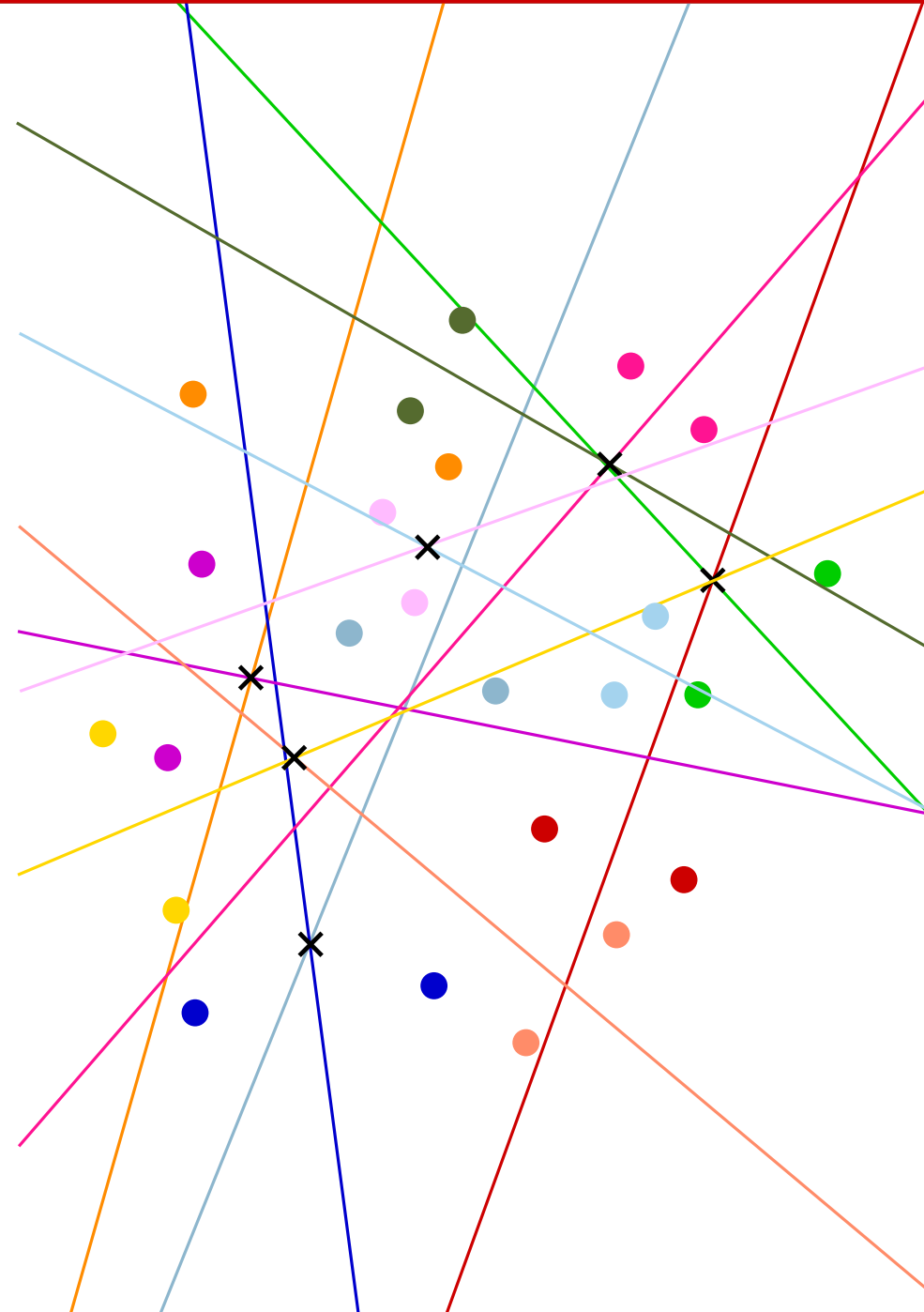
1. Pair up the points, and compute the perpendicular bisector of each pair.
2. Compute the median slope of the non-vertical bisectors, and take it as horizontal.



COMPUTING THE MINIMUM SPANNING CIRCLE

MSC in the plane

1. Pair up the points, and compute the perpendicular bisector of each pair.
2. Compute the median slope of the non-vertical bisectors, and take it as horizontal.
3. Pair up the non-vertical non-horizontal bisectors, always pairing up one positive slope bisector with one negative slope bisector. Compute the intersection point of each pair.

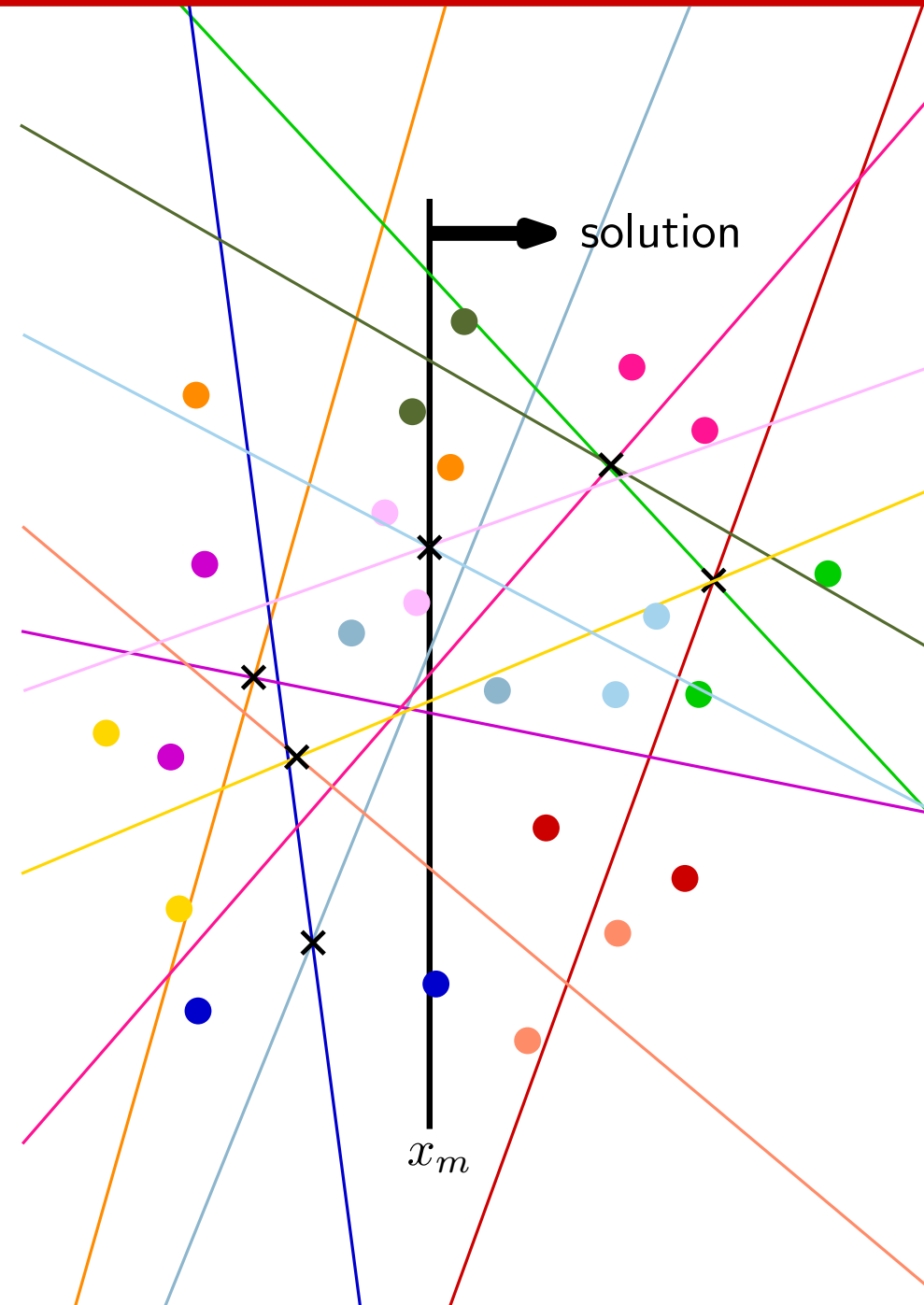


COMPUTING THE MINIMUM SPANNING CIRCLE

MSC in the plane

4. First search

- Compute the median value x_m of the x -coordinate of the intersection points (and vertical bisectors).
- Solve the MSC problem restricted to the line $x = x_m$.
- Use the solution to decide whether the unrestricted solution has been found or it lies to the left/right of the vertical line $x = x_m$.



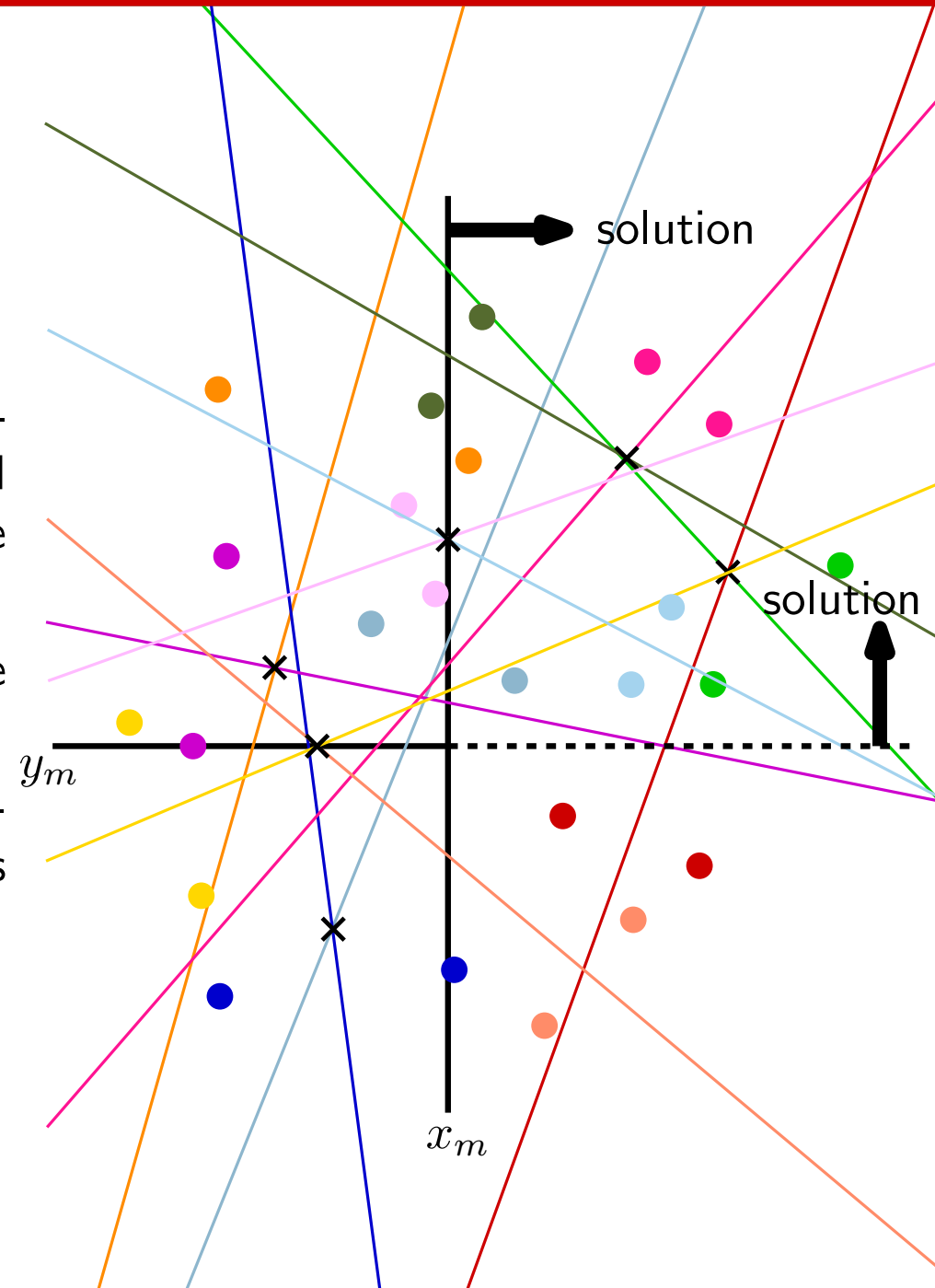
COMPUTING THE MINIMUM SPANNING CIRCLE

MSC in the plane

4. First search
5. Second search

If the solution has not been found,

- Compute the median value y_m of the y -coordinate of the intersection points (and horizontal bisectors) lying in the opposite halfplane.
- Solve the MSC problem restricted to the line $y = y_m$.
- Use the solution to decide whether the unrestricted solution has been found or it lies above/below the horizontal line $y = y_m$.



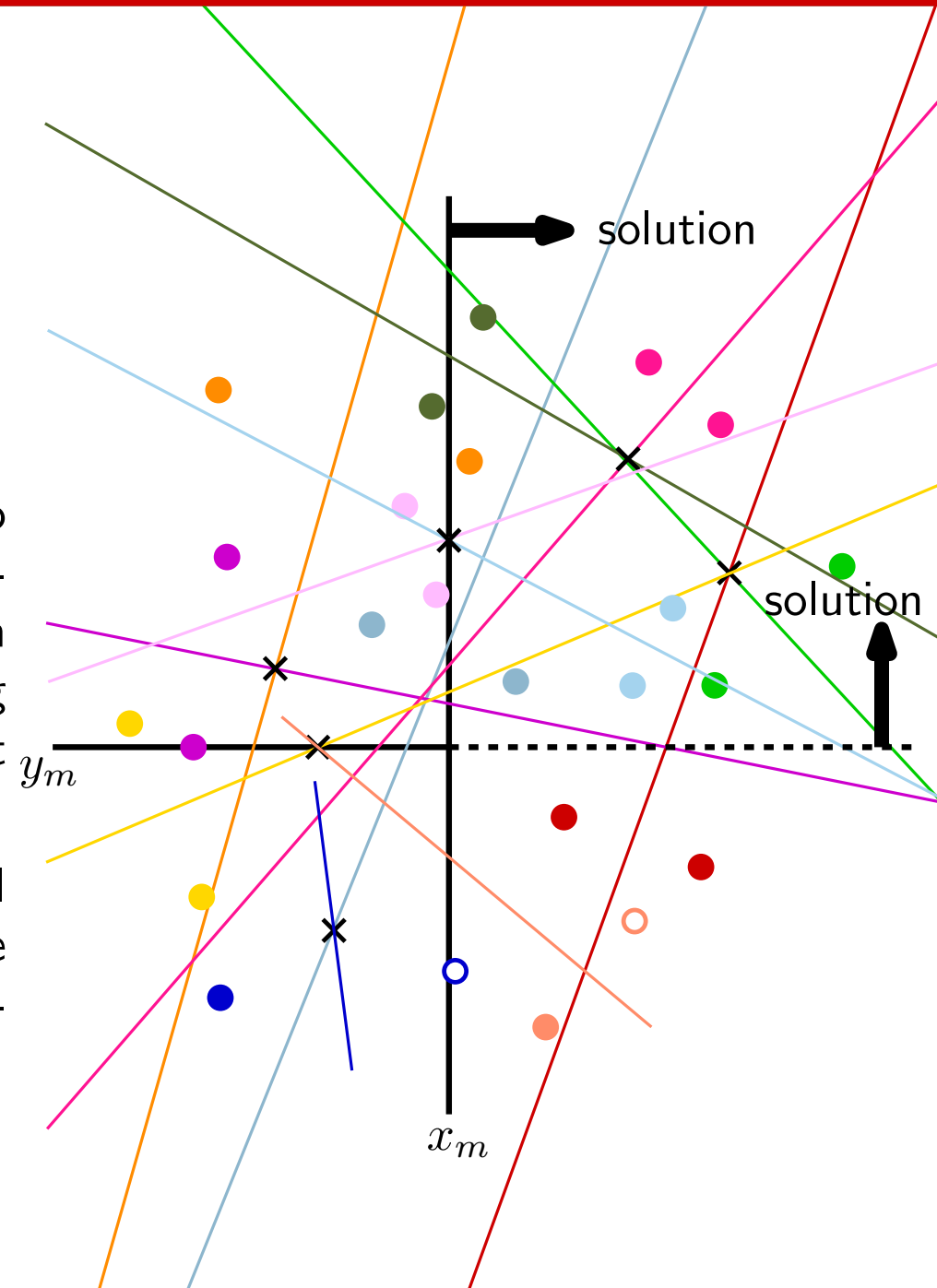
COMPUTING THE MINIMUM SPANNING CIRCLE

MSC in the plane

4. First search
5. Second search
6. Prune

If the solution has not been found, then

- Each intersection point lying opposite to the solution quadrant is defined by a bisector which does not intersect the solution quadrant. Among the two points defining it, the one closest to the solution quadrant can be eliminated.
- Analogously, for each horizontal/vertical bisector in the half-plane opposite to the solution quadrant one point can be eliminated.



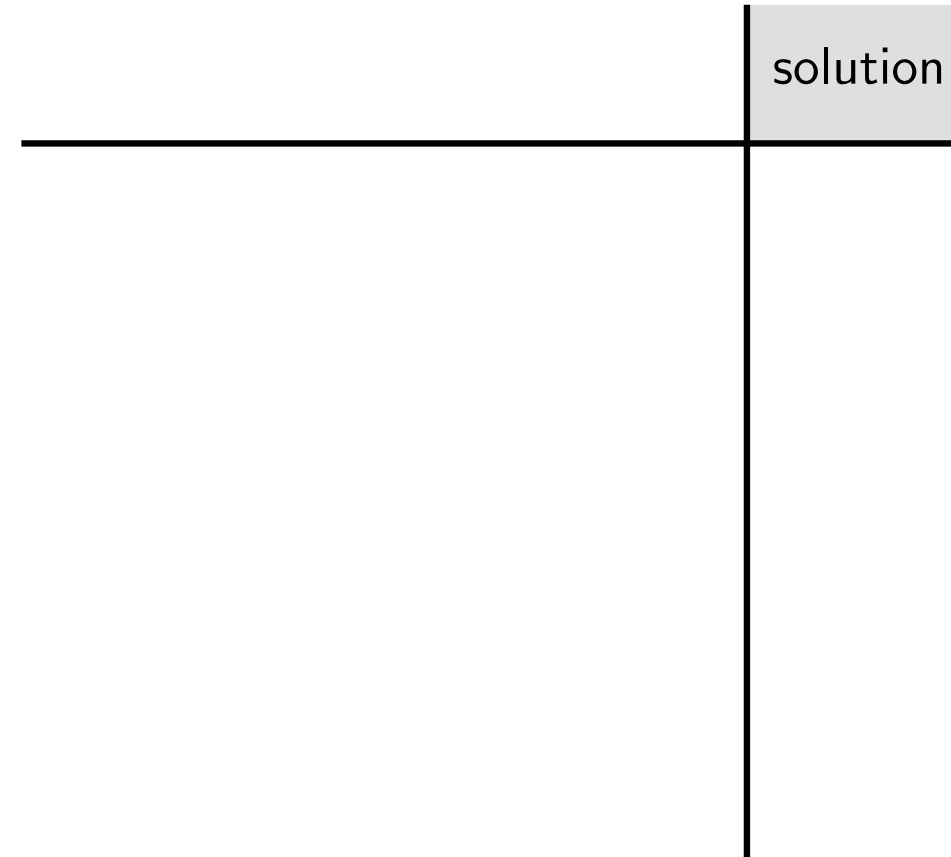
COMPUTING THE MINIMUM SPANNING CIRCLE

MSC in the plane

4. First search
5. Second search
6. Prune

If the solution has not been found, then

- Each intersection point lying opposite to the solution quadrant is defined by a bisector which does not intersect the solution quadrant. Among the two points defining it, the one closest to the solution quadrant can be eliminated.
- Analogously, for each horizontal/vertical bisector in the half-plane opposite to the solution quadrant one point can be eliminated.



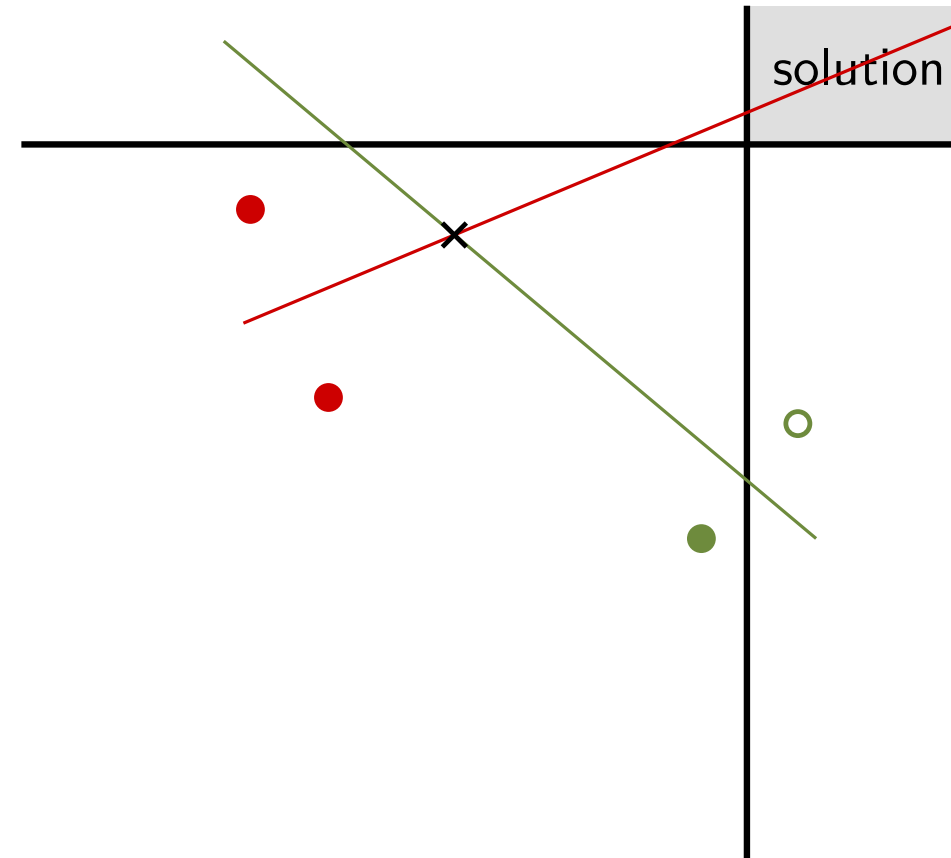
COMPUTING THE MINIMUM SPANNING CIRCLE

MSC in the plane

4. First search
5. Second search
6. Prune

If the solution has not been found, then

- Each intersection point lying opposite to the solution quadrant is defined by a bisector which does not intersect the solution quadrant. Among the two points defining it, the one closest to the solution quadrant can be eliminated.
- Analogously, for each horizontal/vertical bisector in the half-plane opposite to the solution quadrant one point can be eliminated.



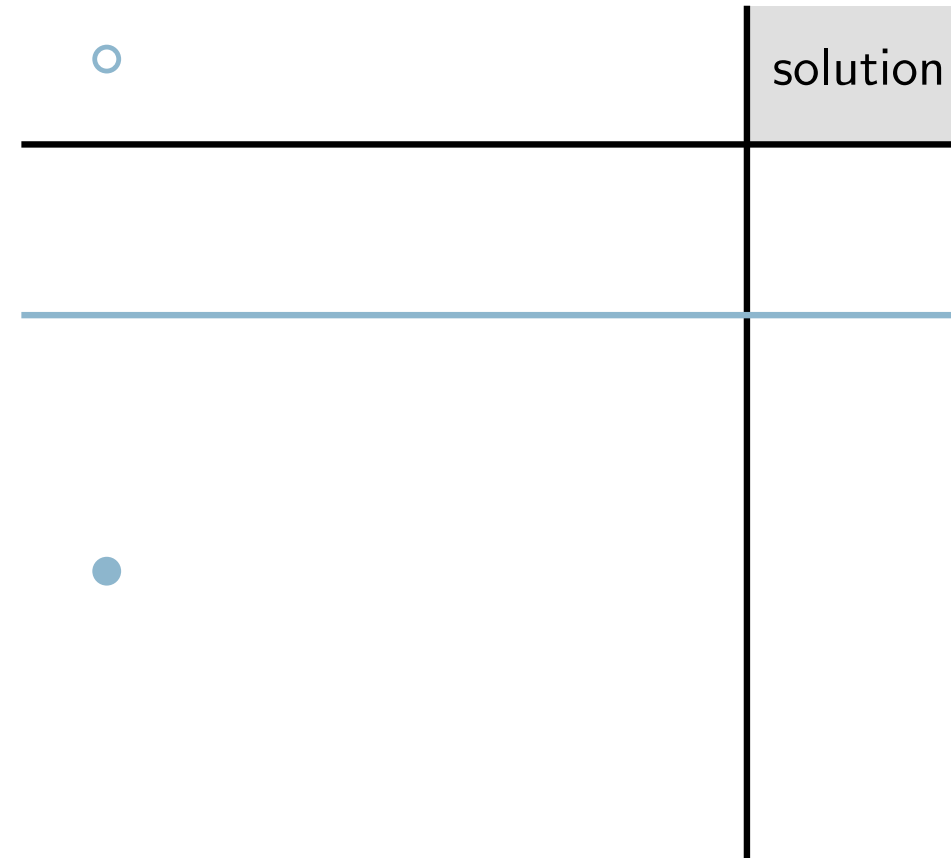
COMPUTING THE MINIMUM SPANNING CIRCLE

MSC in the plane

4. First search
5. Second search
6. Prune

If the solution has not been found, then

- Each intersection point lying opposite to the solution quadrant is defined by a bisector which does not intersect the solution quadrant. Among the two points defining it, the one closest to the solution quadrant can be eliminated.
- Analogously, for each horizontal/vertical bisector in the half-plane opposite to the solution quadrant one point can be eliminated.



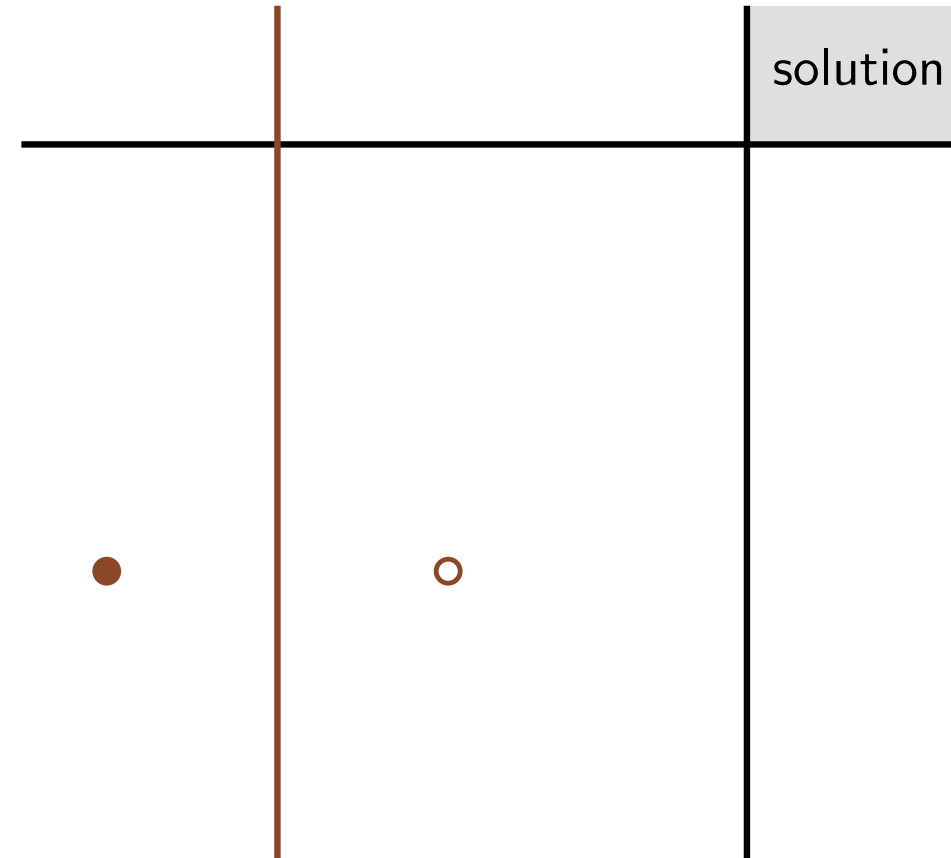
COMPUTING THE MINIMUM SPANNING CIRCLE

MSC in the plane

4. First search
5. Second search
6. Prune

If the solution has not been found, then

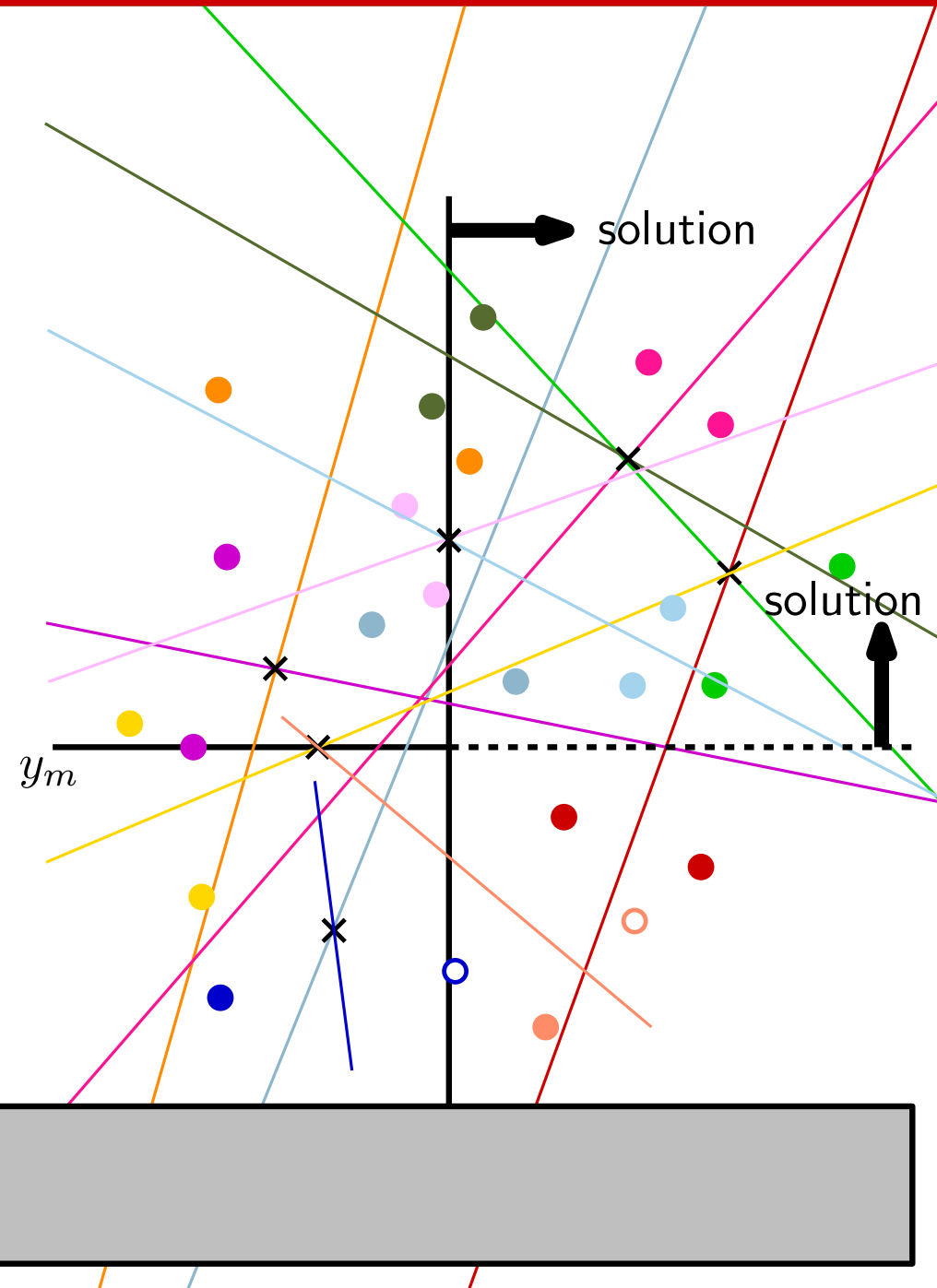
- Each intersection point lying opposite to the solution quadrant is defined by a bisector which does not intersect the solution quadrant. Among the two points defining it, the one closest to the solution quadrant can be eliminated.
- Analogously, for each horizontal/vertical bisector in the half-plane opposite to the solution quadrant one point can be eliminated.



COMPUTING THE MINIMUM SPANNING CIRCLE

MSC in the plane

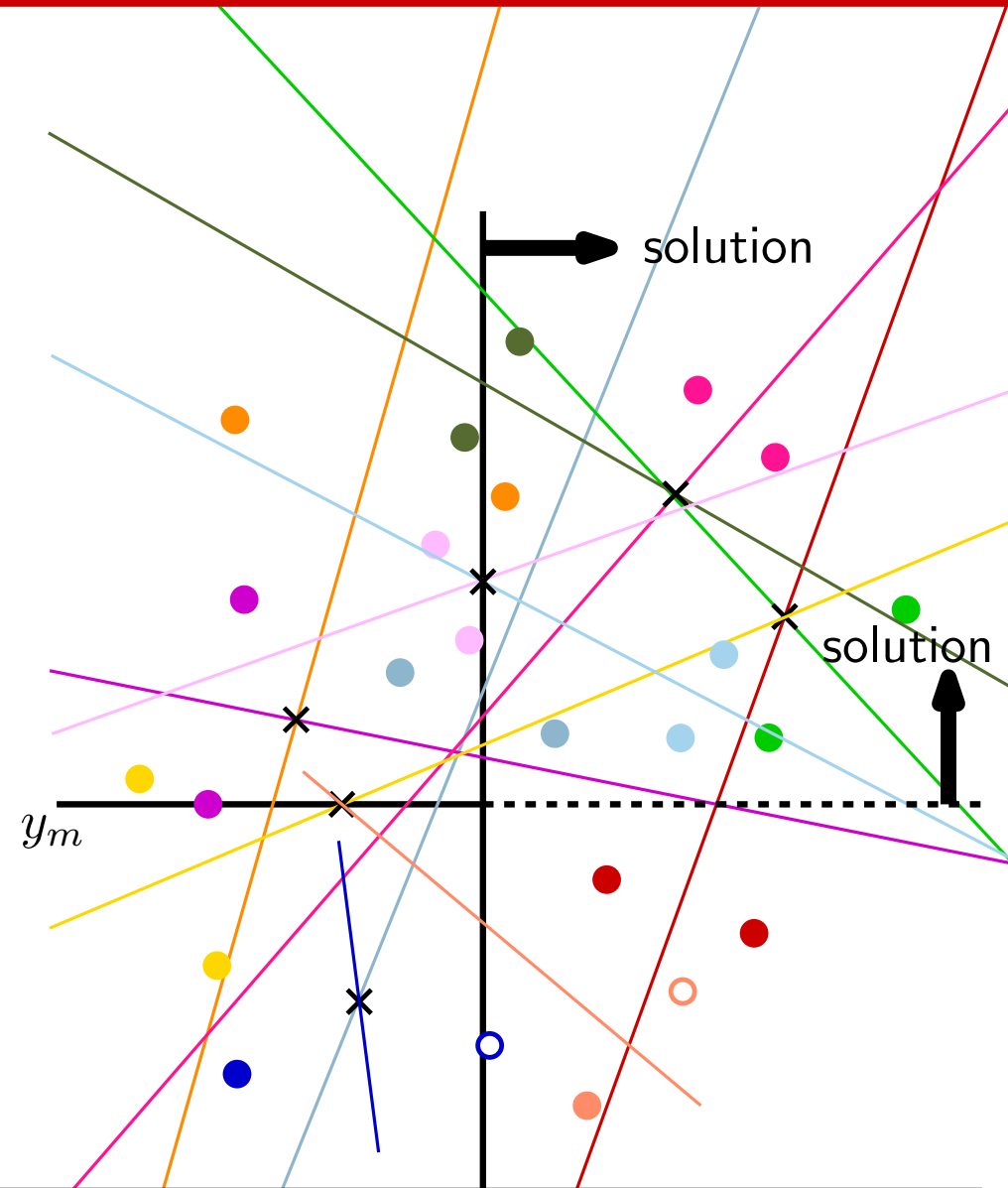
1. Pair up the points, and compute the perpendicular bisector of each pair. $O(n)$ time
2. Compute the median slope of the non-vertical bisectors, and take it as horizontal. $O(n)$ time
3. Pair up the non-vertical non-horizontal bisectors, and compute the intersection point of each pair. $O(n)$ time
4. First search: $O(n)$ time
5. Second search: $O(n)$ time
6. Prune: In $O(n)$ time, at least $1/16$ of the input is pruned.



COMPUTING THE MINIMUM SPANNING CIRCLE

MSC in the plane

1. Pair up the points, and compute the perpendicular bisector of each pair. $O(n)$ time
2. Compute the median slope of the non-vertical bisectors, and take it as horizontal. $O(n)$ time
3. Pair up the non-vertical non-horizontal bisectors, and compute the intersection point of each pair. $O(n)$ time
4. First search: $O(n)$ time
5. Second search: $O(n)$ time
6. Prune: In $O(n)$ time, at least $1/16$ of the input is pruned.



Conclusion

Given n points in the plane, the minimum spanning circle can be computed in linear time.

INTERSECTING HALF-PLANES AND RELATED PROBLEMS

TO LEARN MORE

- F. P. Preparata and M. I. Shamos, **Computational Geometry: an Introduction** (revised ed.). Springer-Verlag, 1993.
- N. Megiddo, Linear-time algorithms for linear programming in \mathbb{R}^3 and related problems, *SIAM Journal on Computing*, Vol. 12, N. 4, pp. 759-776, 1983.