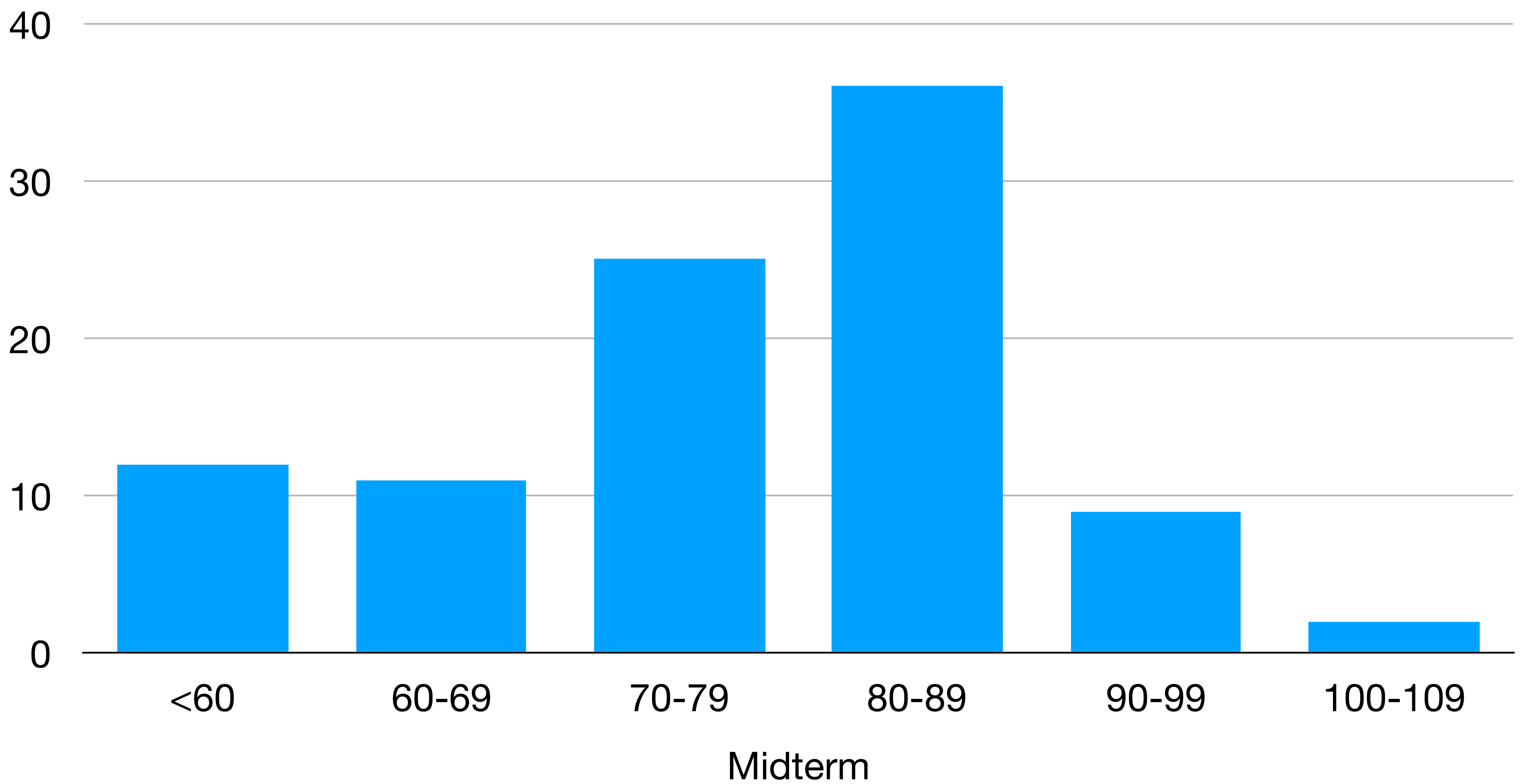
Introduction to Algorithms

Meng-Tsung Tsai

12/10/2019


40
30
20
10
0
<60
60-69
70-79
80-89
90-99
100-109
Midterm

Linear Programming

Happy Farm

Your farm can produce two popular products:

strawberry milk and strawberry yogurt.

The following table shows the amount of ingredients needed to produce each product.

milk		sugar		strawberry		product
2 units	+	2 units	+	5 units	=	1 liter of S-milk
1 unit	+	7 units	+	6 units	=	1 liter of S-yogurt

Happy Farm

Every day in your farm you can gather [1] 5 units of milk, [2] 14 units of sugar, and [3] 15 units of strawberry.

The price of strawberry milk is \$200/liter, and the price of strawberry yogurt is \$250/liter.

Devise a plan to produce strawberry milk and yogurt so that:

(1) the amount of used ingredients $\leq$ the capacity that your farm can produce.

(2) the profit is maximized.

Happy Farm

Every day in your farm you can gather [1] 5 units of milk, [2] 14 units of sugar, and [3] 15 units of strawberry.

The price of strawberry milk is \$200/liter, and the price of strawberry yogurt is \$250/liter.

Suppose that your plan is to produce X liters of S-milk and Y liters of S-yogurt every day.

Then, the problem can be formulated as maximizing
$200 X + 250 Y$ subject to

[1] $2X + Y \leq 5$
[2] $2X + 7Y \leq 14$
[3] $5X + 6Y \leq 15$
[4] $X \geq 0, Y \geq 0.$

milk		sugar		strawberry		product
2 units	+	2 units	+	5 units	=	1 liter of S-milk
1 unit	+	7 units	+	6 units	=	1 liter of S-yogurt

Happy Farm

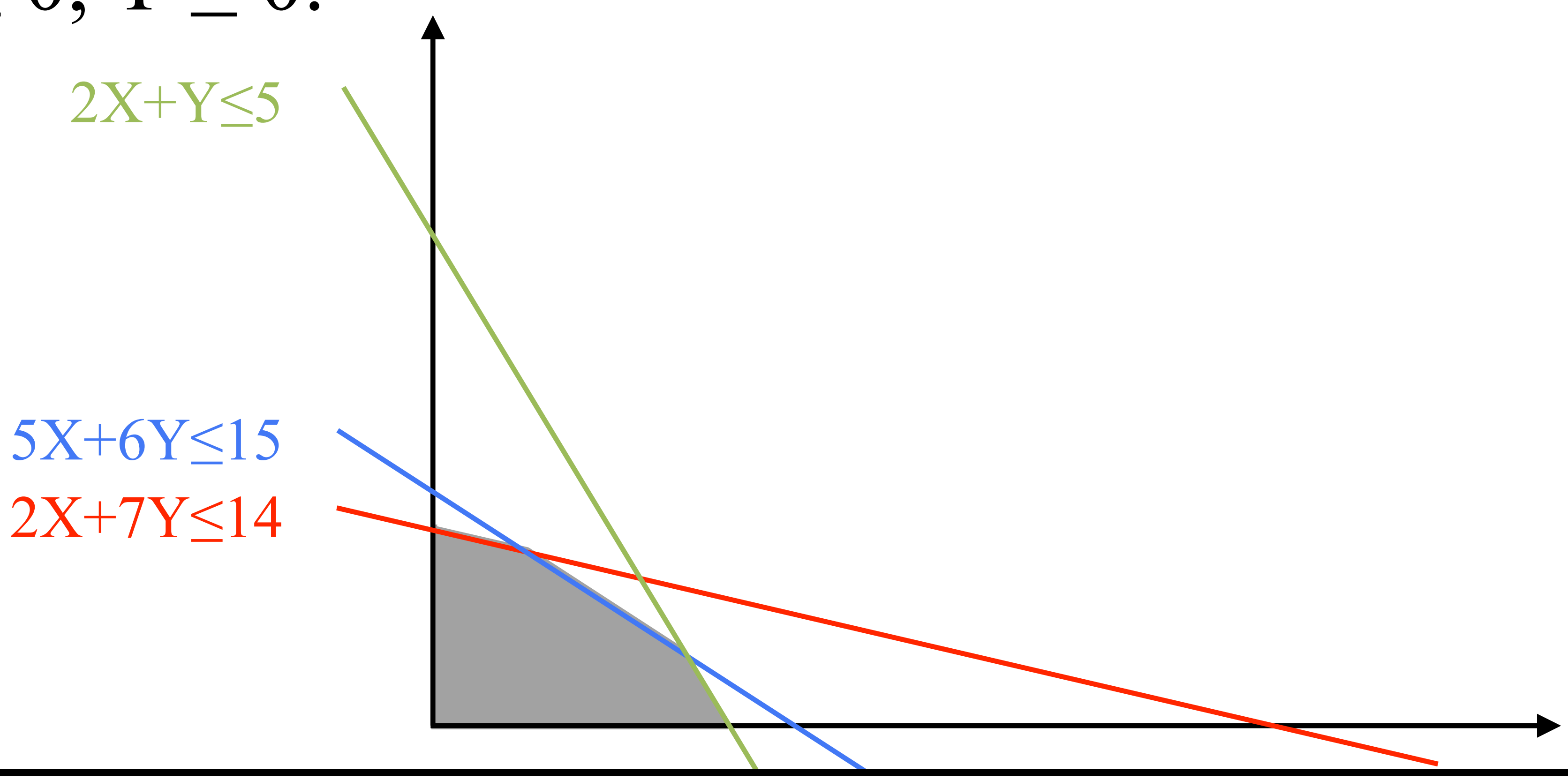
Then, the problem can be formulated as maximizing
$200\ X + 250\ Y$ subject to

[1] $2\ X + Y \leq 5$

[2] $2\ X + 7\ Y \leq 14$

[3] $5\ X + 6\ Y \leq 15$

[4] $X \geq 0,\ Y \geq 0.$

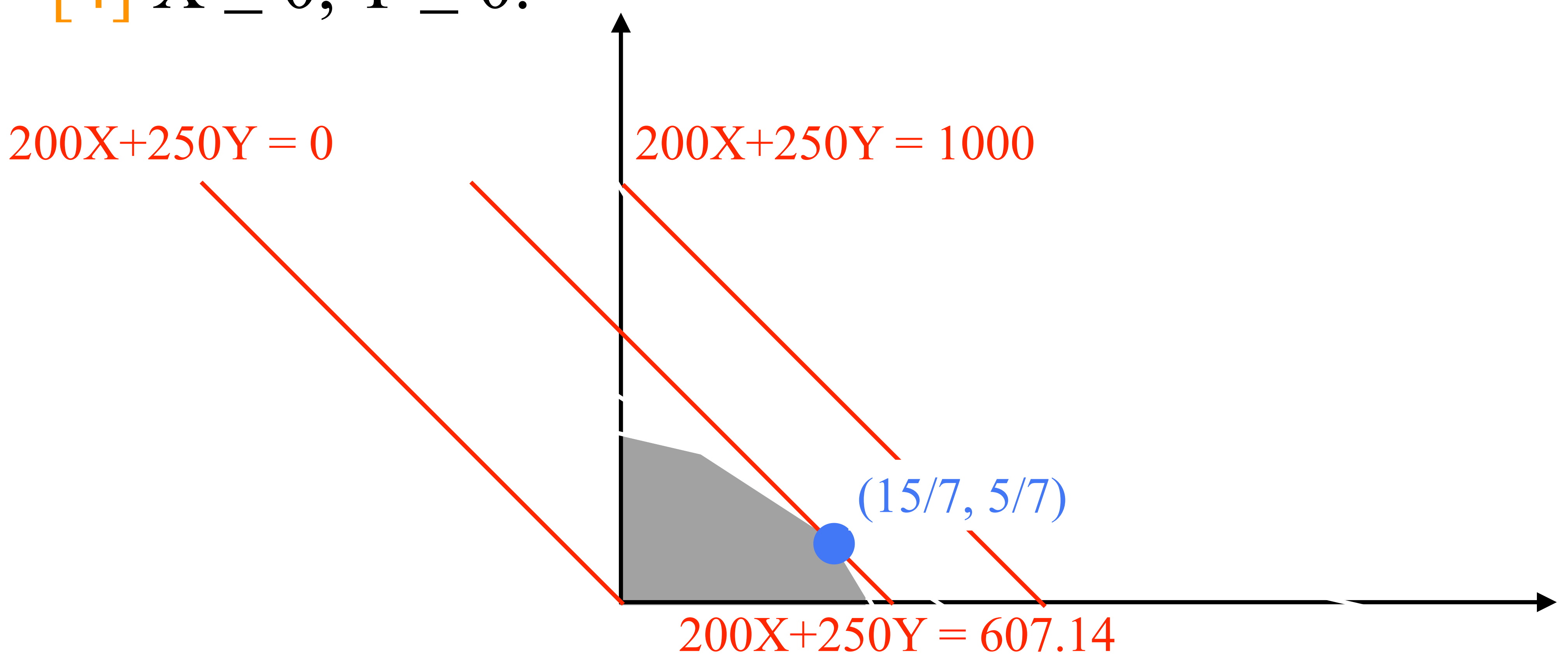


In the grayed region (feasible region), every point represents a feasible solution.

Happy Farm

Then, the problem can be formulated as maximizing
200 X + 250 Y subject to

[1] 2 X + Y ≤ 5
[2] 2 X + 7 Y ≤ 14
[3] 5 X + 6 Y ≤ 15
[4] X ≥ 0, Y ≥ 0.



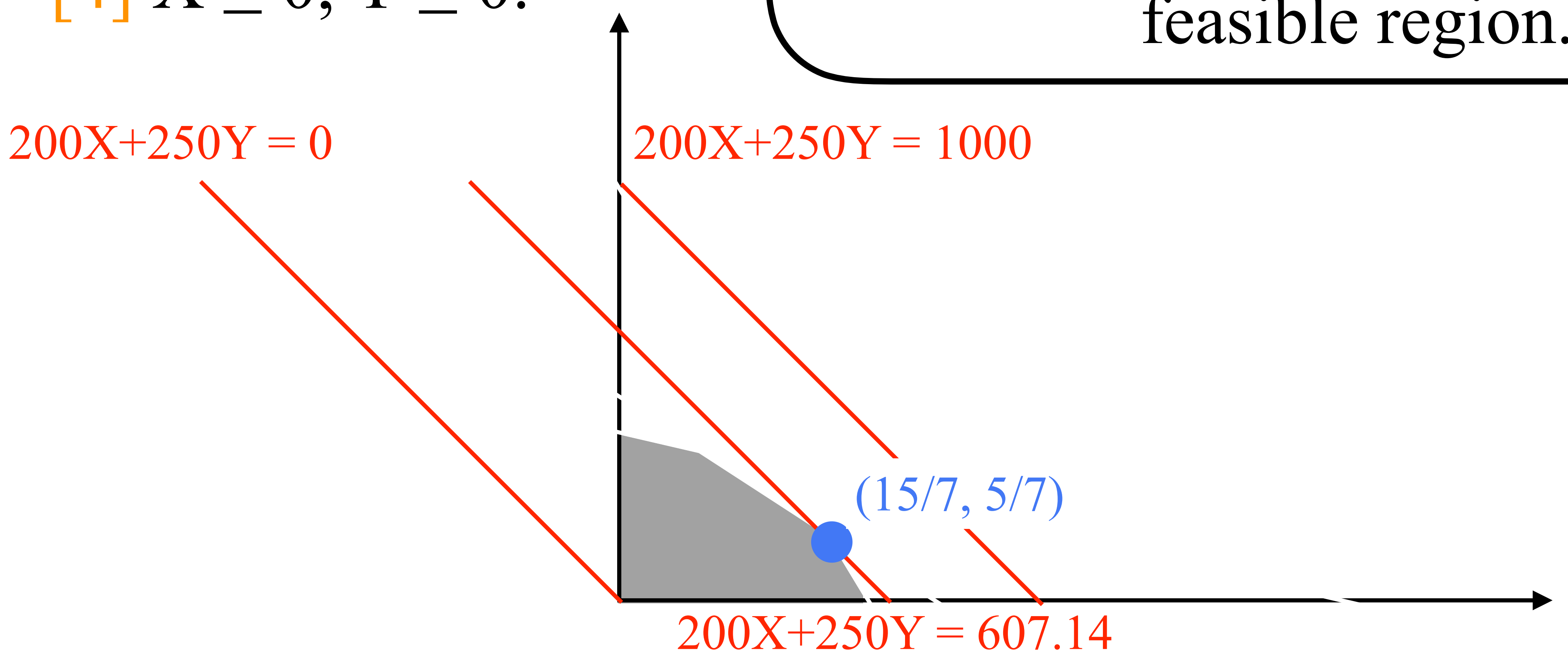
We'd like to find the feasible solution that maximize the profit.

Happy Farm

Then, the problem can be formulated as maximizing
200 X + 250 Y subject to

[1] $2X + Y \leq 5$
[2] $2X + 7Y \leq 14$
[3] $5X + 6Y \leq 15$
[4] $X \geq 0,\ Y \geq 0.$

If we draw the lines: 200X+250Y = z for every z, some of them touches the feasible region.



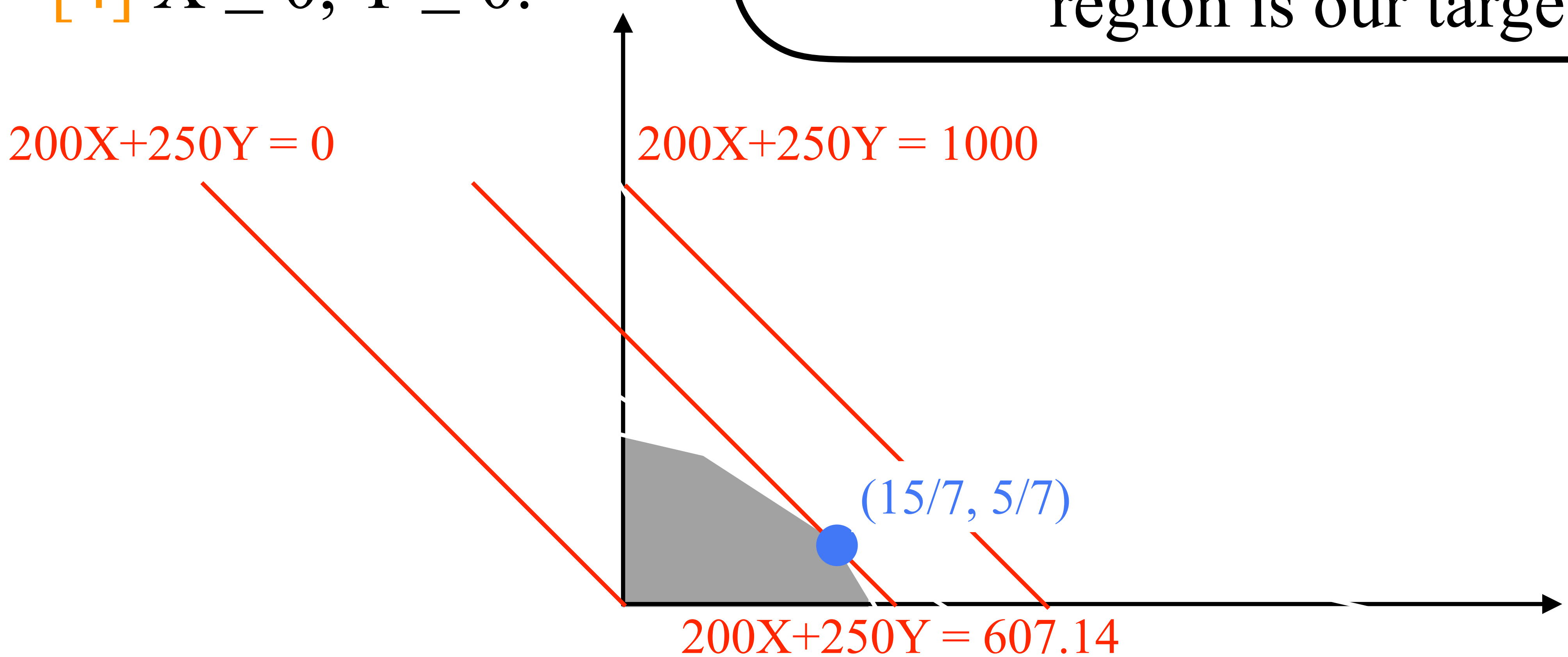
We'd like to find the feasible solution that maximize the profit.

Happy Farm

Then, the problem can be formulated as maximizing $200\,X + 250\,Y$ subject to

[1] $2\,X + Y \leq 5$
[2] $2\,X + 7\,Y \leq 14$
[3] $5\,X + 6\,Y \leq 15$
[4] $X \geq 0,\ Y \geq 0.$

The line ℓ: $200X+250Y=z$ so that z is maximized and ℓ touches the feasible region is our target.



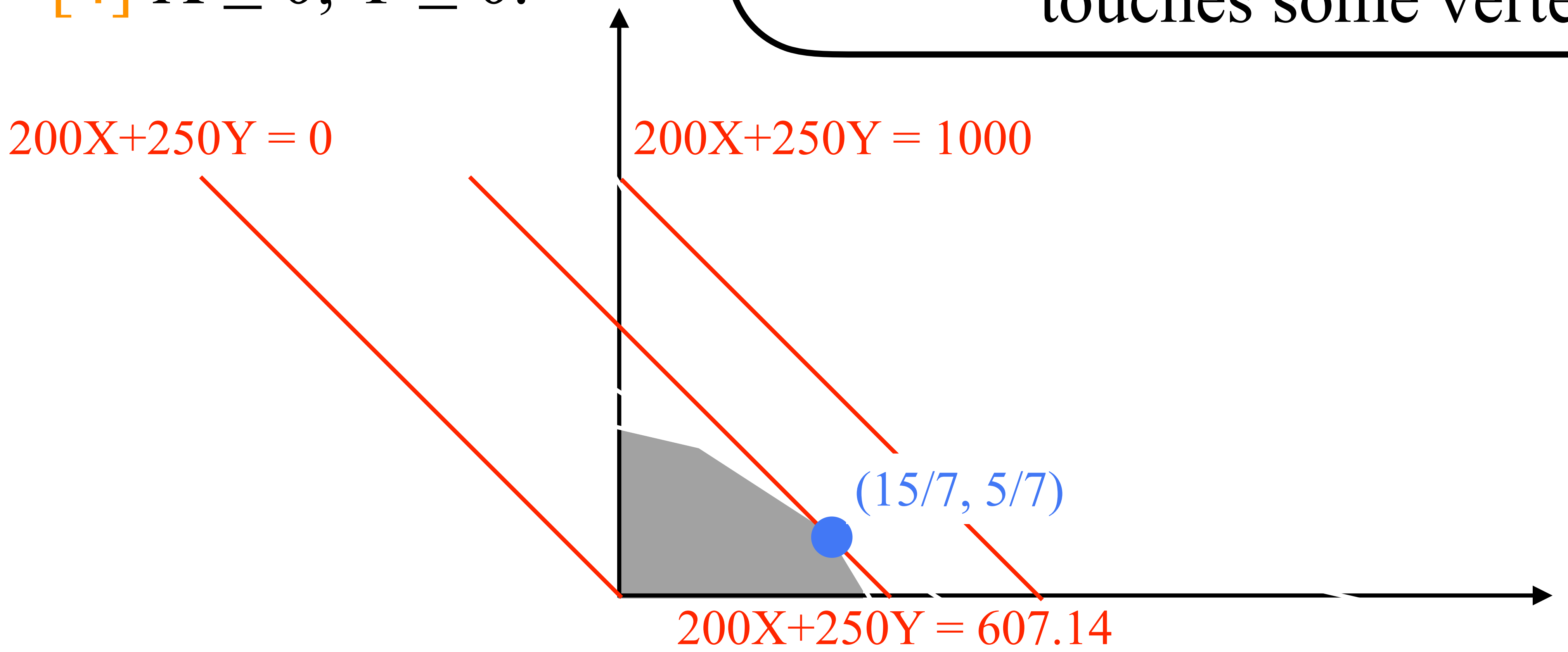
We'd like to find the feasible solution that maximize the profit.

Happy Farm

Then, the problem can be formulated as maximizing
200 X + 250 Y subject to

[1] $2X + Y \leq 5$
[2] $2X + 7Y \leq 14$
[3] $5X + 6Y \leq 15$
[4] $X \geq 0$, $Y \geq 0$.

Such an optimal line ℓ must touch a vertex or an edge. In the latter case, it still touches some vertex.



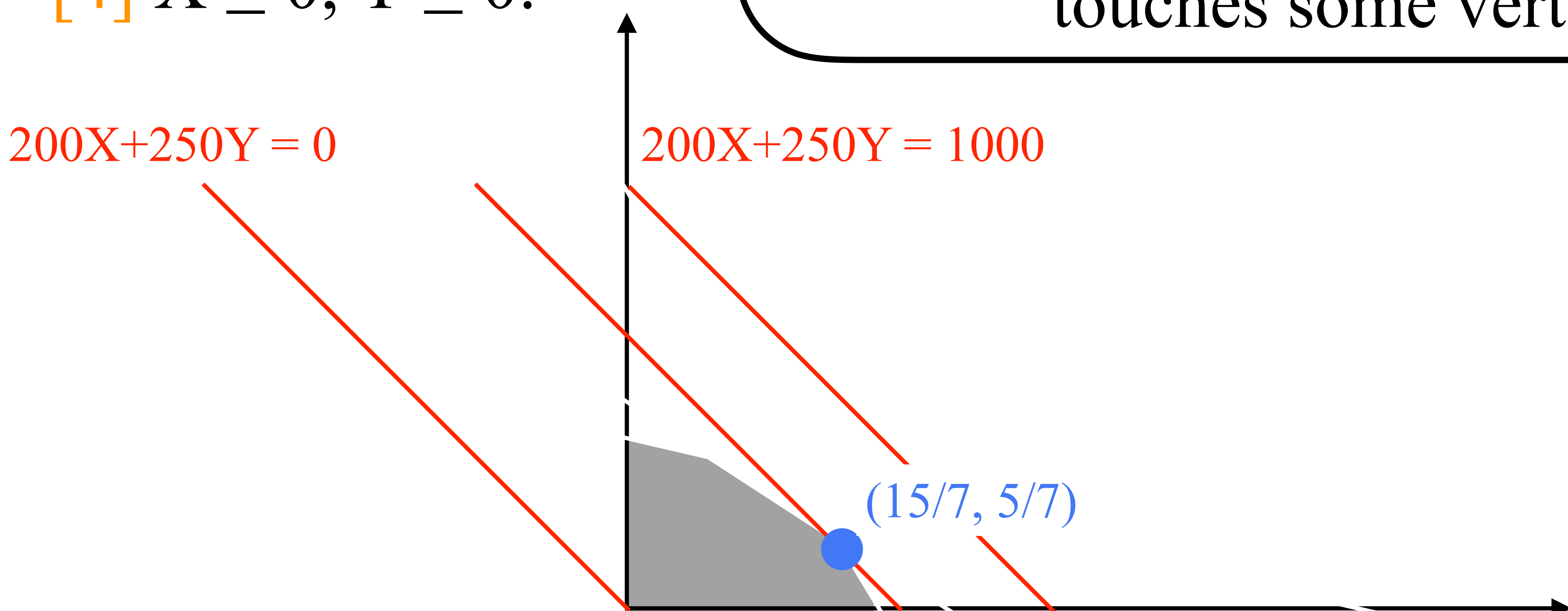
We'd like to find the feasible solution that maximize the profit.

Happy Farm

Then, the problem can be formulated as maximizing
200 X + 250 Y subject to

[1] 2 X + Y ≤ 5
[2] 2 X + 7 Y ≤ 14
[3] 5 X + 6 Y ≤ 15
[4] X ≥ 0, Y ≥ 0.

Such an optimal line ℓ must touch a vertex or an edge. In the latter case, it still touches some vertex.



Solution: Pick a vertex in the feasible region that maximizes the objective function.

General LP

It is the problem of optimizing a linear objective function subject to a set of linear constraints. Equivalently, we can use the following form to represent all the problem instances.

maximize $\sum_{1 \le j \le n} c_j x_j$

subject to $\sum_{1 \le j \le n} a_{ij} x_j \le b_i$ for $i = 1, 2, \ldots, m$

where c_j's, a_{ij}'s, and b_i's are constants.

General LP

It is the problem of optimizing a linear objective function subject to a set of linear constraints. Equivalently, we can use the following form to represent all the problem instances.

maximize $\sum_{1\leq j\leq n} c_j x_j$

subject to $\sum_{1\leq j\leq n} a_{ij} x_j \leq b_i$ for $i = 1, 2, ..., m$

where c_j's, a_{ij}'s, and b_i's are constants.

In contrast, to minimize an objective function $\sum_{1\leq j\leq n} d_j x_j$, one can rewrite it as maximizing $\sum_{1\leq j\leq n} -d_j x_j$.

General LP

It is the problem of optimizing a linear objective function subject to a set of linear constraints. Equivalently, we can use the following form to represent all the problem instances.

maximize $\sum_{1\leq j\leq n} c_j x_j$

subject to $\sum_{1\leq j\leq n} a_{ij} x_j \leq b_i$ for $i = 1, 2, ..., m$

where c_j's, a_{ij}'s, and b_i's are constants.

To add a constraint that $\sum_{1\leq j\leq n} d_j x_j \geq e_i$, one can add an alternative constraint $-\sum_{1\leq j\leq n} d_j x_j \leq -e_i$.

General LP

It is the problem of optimizing a linear objective function subject to a set of linear constraints. Equivalently, we can use the following form to represent all the problem instances.

maximize $\sum_{1\leq j\leq n} c_j x_j$

subject to $\sum_{1\leq j\leq n} a_{ij} x_j \leq b_i$ for $i = 1, 2, ..., m$

where c_j's, a_{ij}'s, and b_i's are constants.

To add a constraint that $\sum_{1\leq j\leq n} d_j x_j = e_i$, one can add two alternative constraints $\sum_{1\leq j\leq n} d_j x_j \leq e_i$ and $-\sum_{1\leq j\leq n} d_j x_j \leq -e_i$.

General LP

It is the problem of optimizing a linear objective function subject to a set of linear constraints. Equivalently, we can use the following form to represent all the problem instances.

maximize $\sum_{1 \le j \le n} c_j x_j$

subject to $\sum_{1 \le j \le n} a_{ij} x_j \le b_i$ for $i = 1, 2, ..., m$

where c_j's, a_{ij}'s, and b_i's are constants.

There are software packages can solve LP.

Algorithms for LP

	simplex	ellipsoid method	interior-point method
analytically	exponential-time in the worst case	polynomial-time	polynomial-time
in practice	fast in practice	slow in practice	fast for large n, m

GLPK (GNU Linear Programming Kit) contains the implementations of the above algorithms.

Linear Programming in Low Dimensions

LP in low dimensions

maximize $\sum_{1 \leq j \leq n} c_j x_j$

subject to $\sum_{1 \leq j \leq n} a_{ij} x_j \leq b_i$ for $i = 1, 2, \ldots, m$

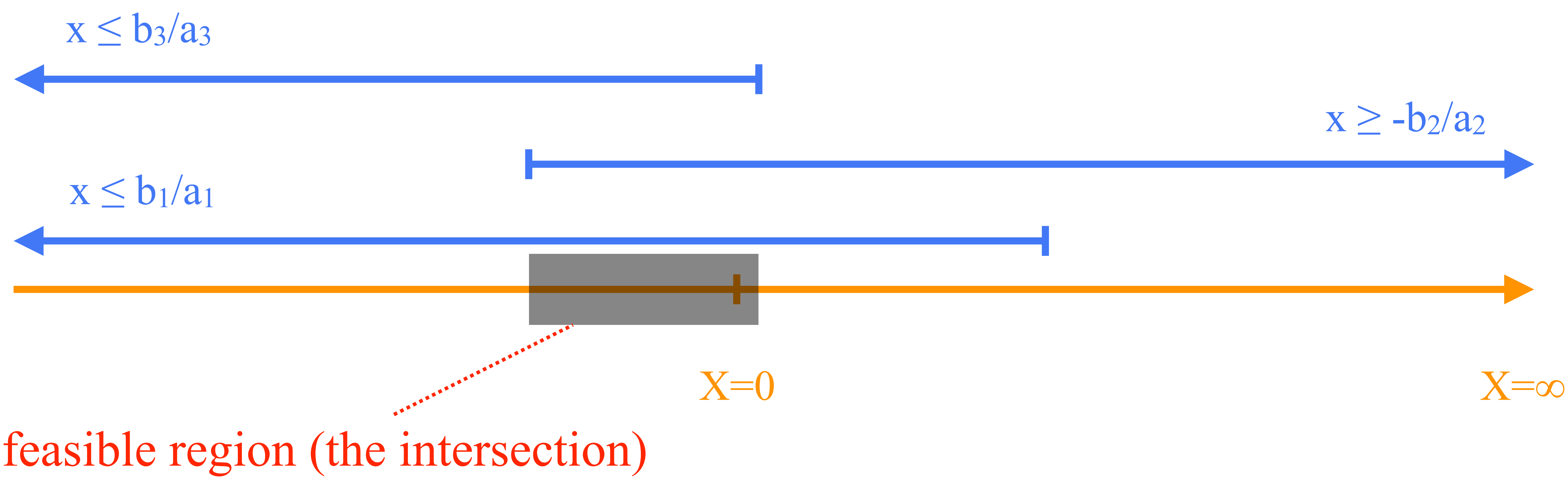
where c_j's, a_{ij}'s, and b_i's are constants, and $n = O(1)$.

For n = 1

maximize cx

subject to $a_i x \leq b_i$ for i = 1, 2, ..., m

where c, a_i's, and b_i's are constants.



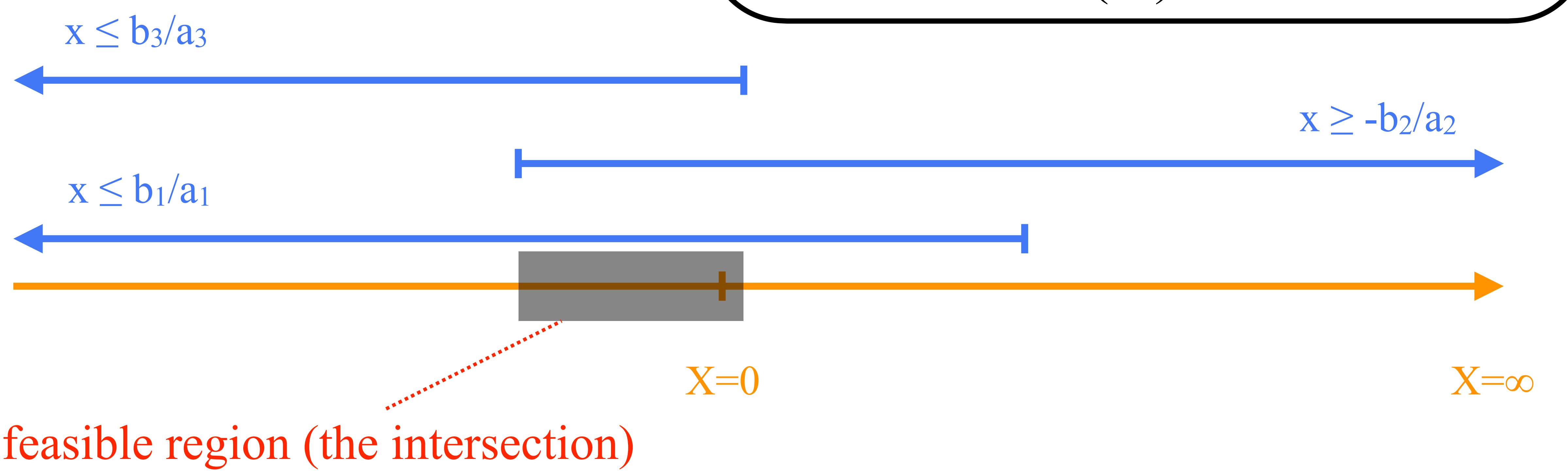
For n = 1

maximize cx

subject to $a_i x \leq b_i$ for i = 1, 2, ..., m

where c, a_i's, and b_i's are constants.

The intersection can be found in O(m) time.



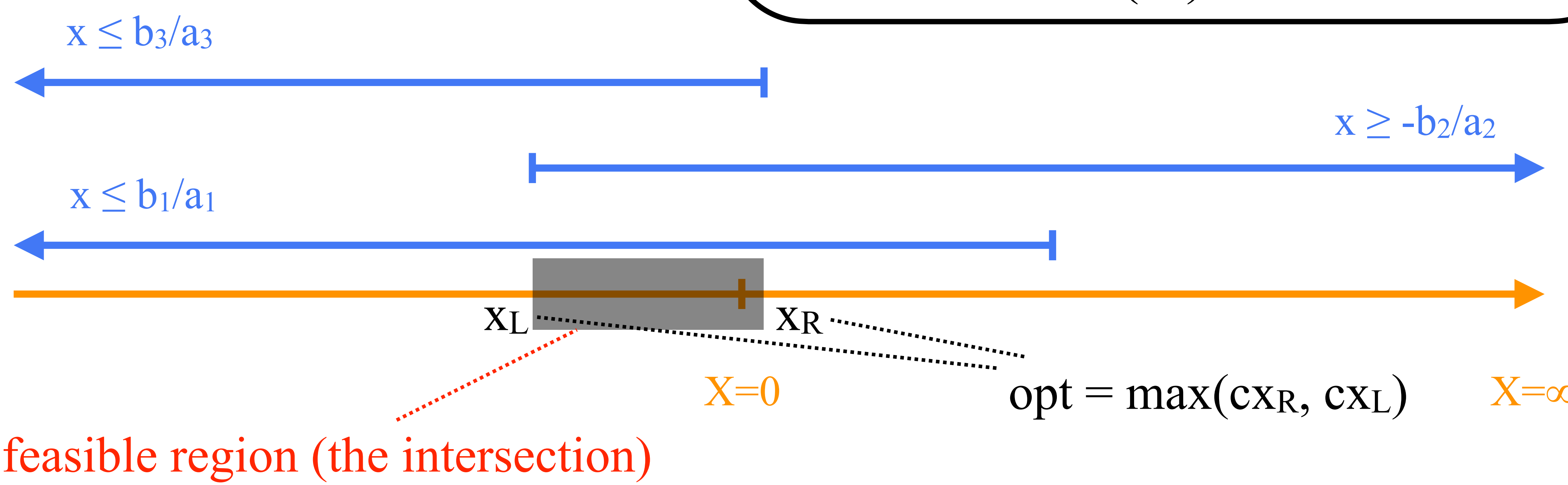
For n = 1

maximize cx

subject to $a_i x \leq b_i$ for $i = 1, 2, ..., m$

where c, a_i's, and b_i's are constants.

The intersection can be found in O(m) time.

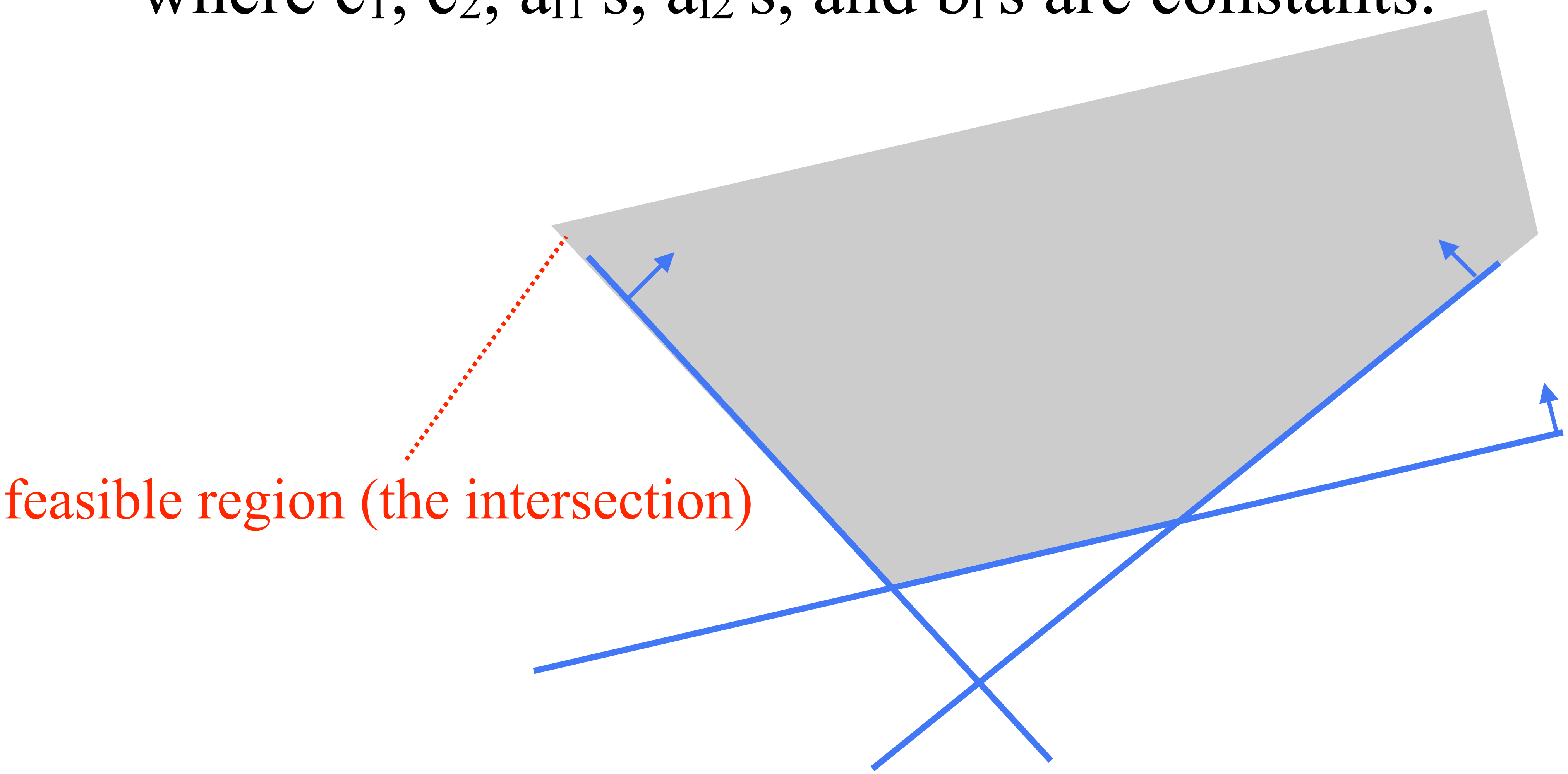


For n = 2

maximize $c_1x_1+c_2x_2$

subject to $a_{i1}x_1 + a_{i2}x_2 \leq b_i$ for $i = 1, 2, ..., m$

where c_1, c_2, a_{i1}'s, a_{i2}'s, and b_i's are constants.

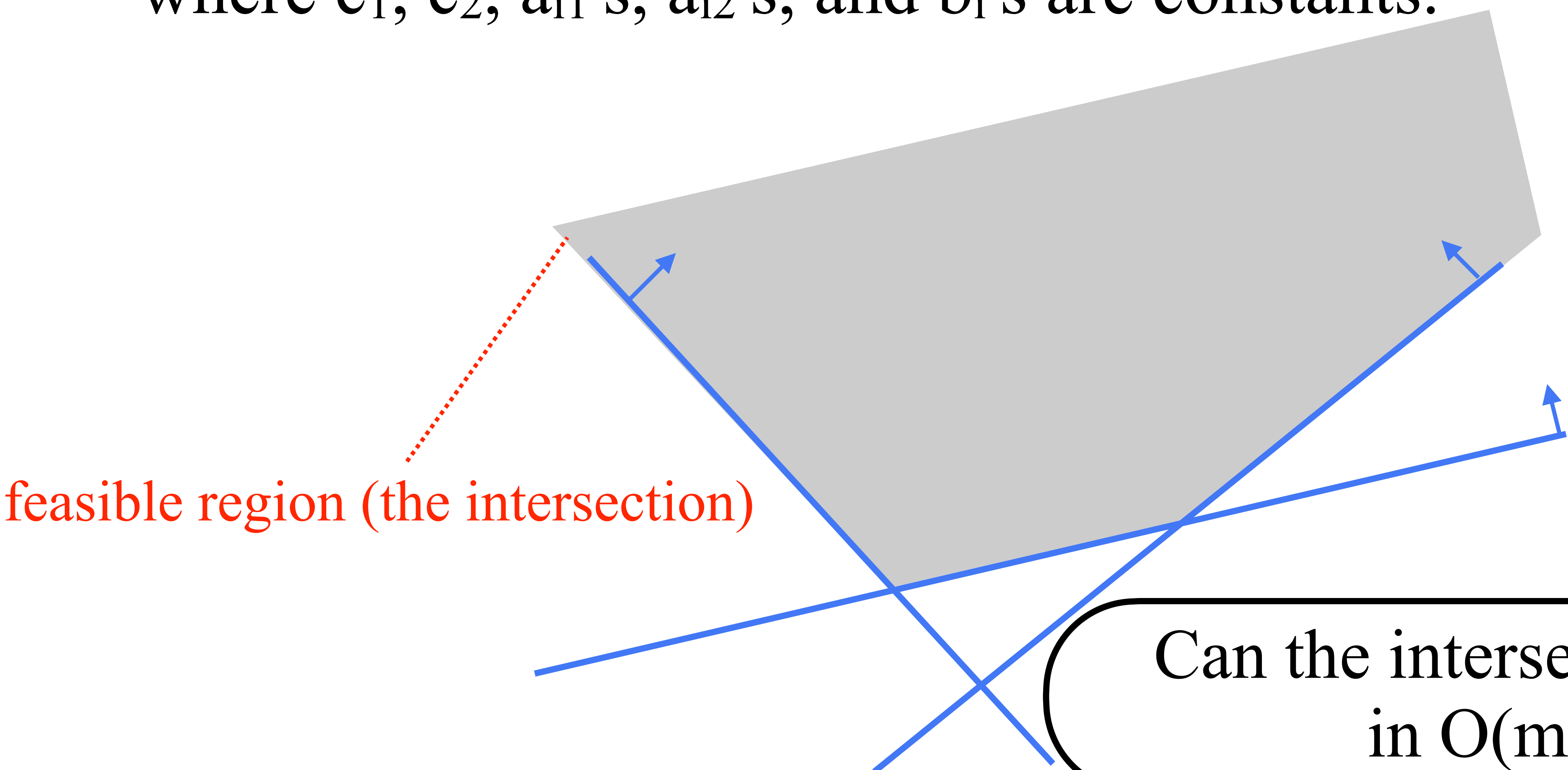


For n = 2

maximize $c_1x_1+c_2x_2$

subject to $a_{i1}x_1 + a_{i2}x_2 \leq b_i$ for $i = 1, 2, \ldots, m$

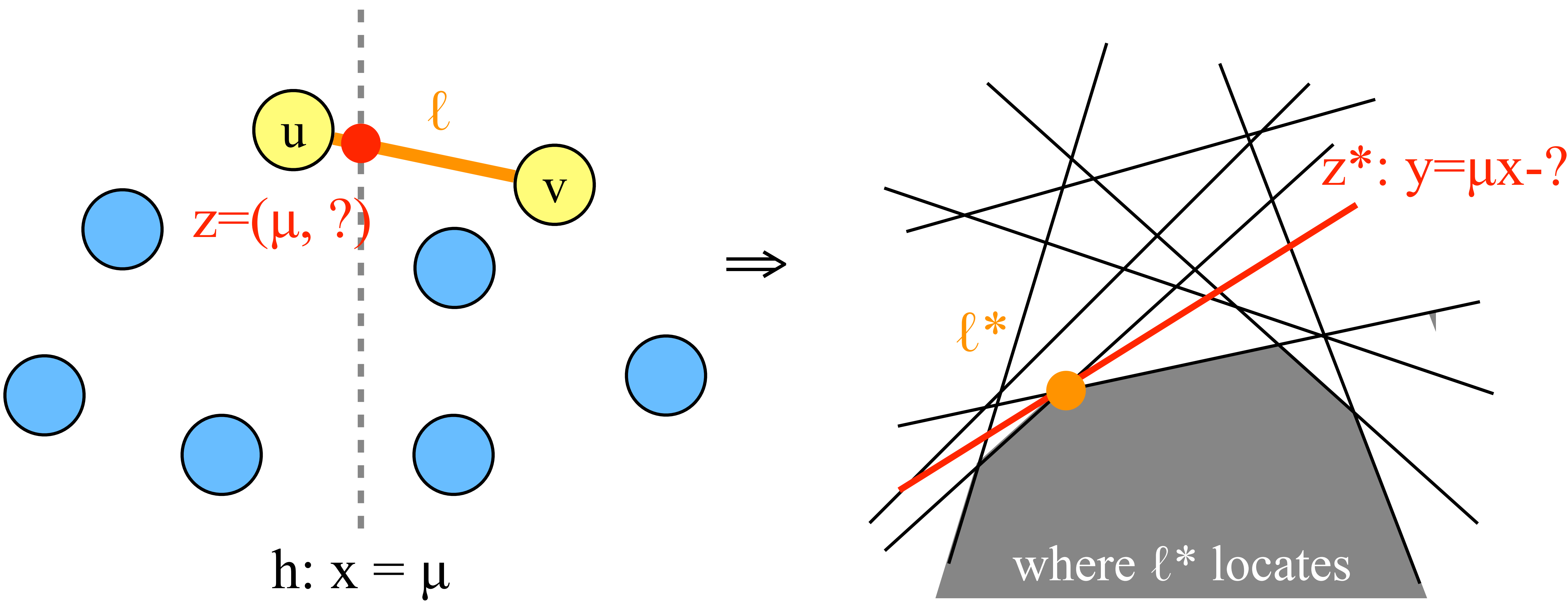
where c_1, c_2, a_{i1}'s, a_{i2}'s, and b_i's are constants.



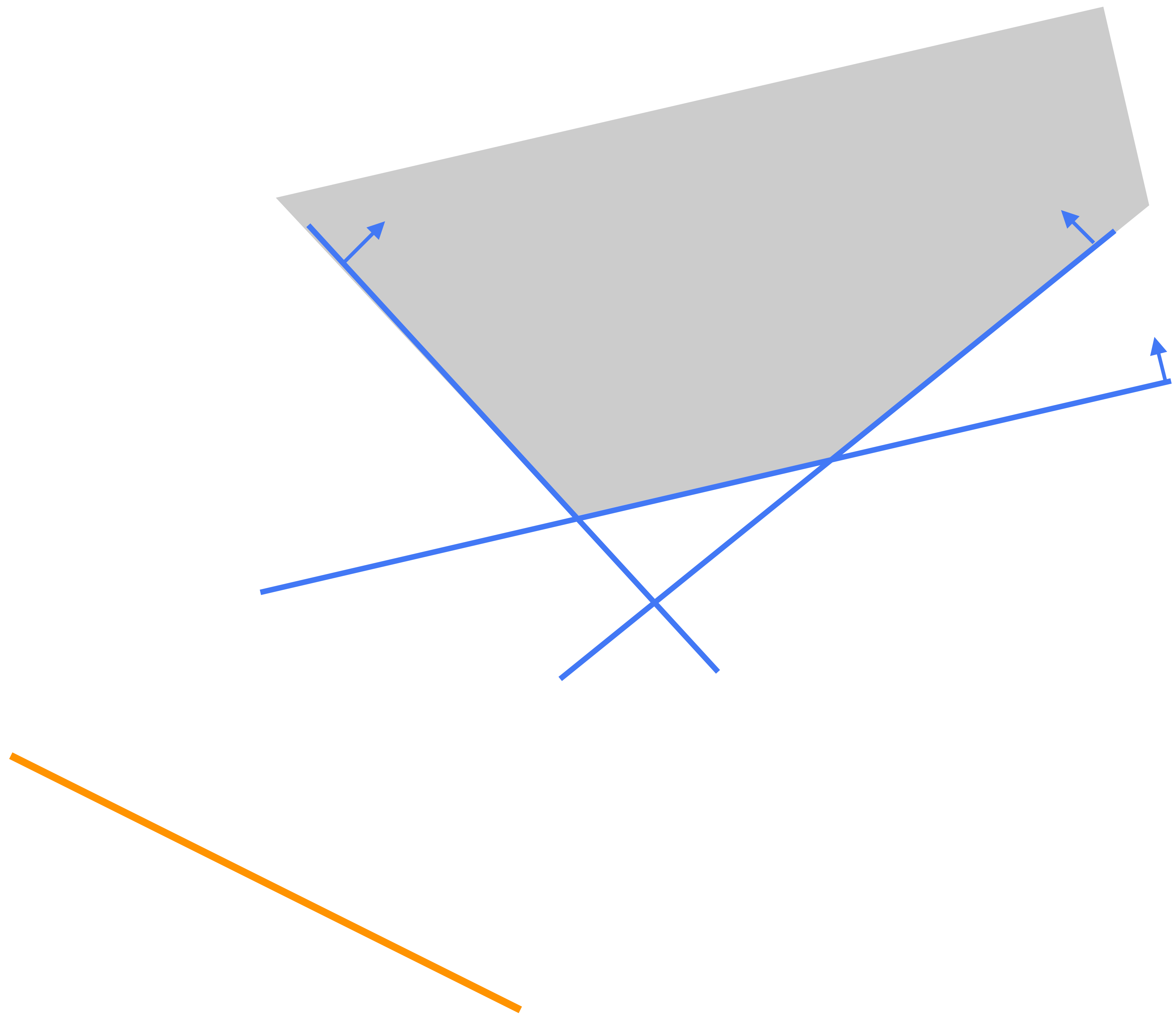
Can the intersection be found in O(m) time?

The intersection cannot be found in O(m) time in comparison-based model

Otherwise, the upper hull in the primal plane can be found in O(m) time, where m denotes # of points in the primal plane.

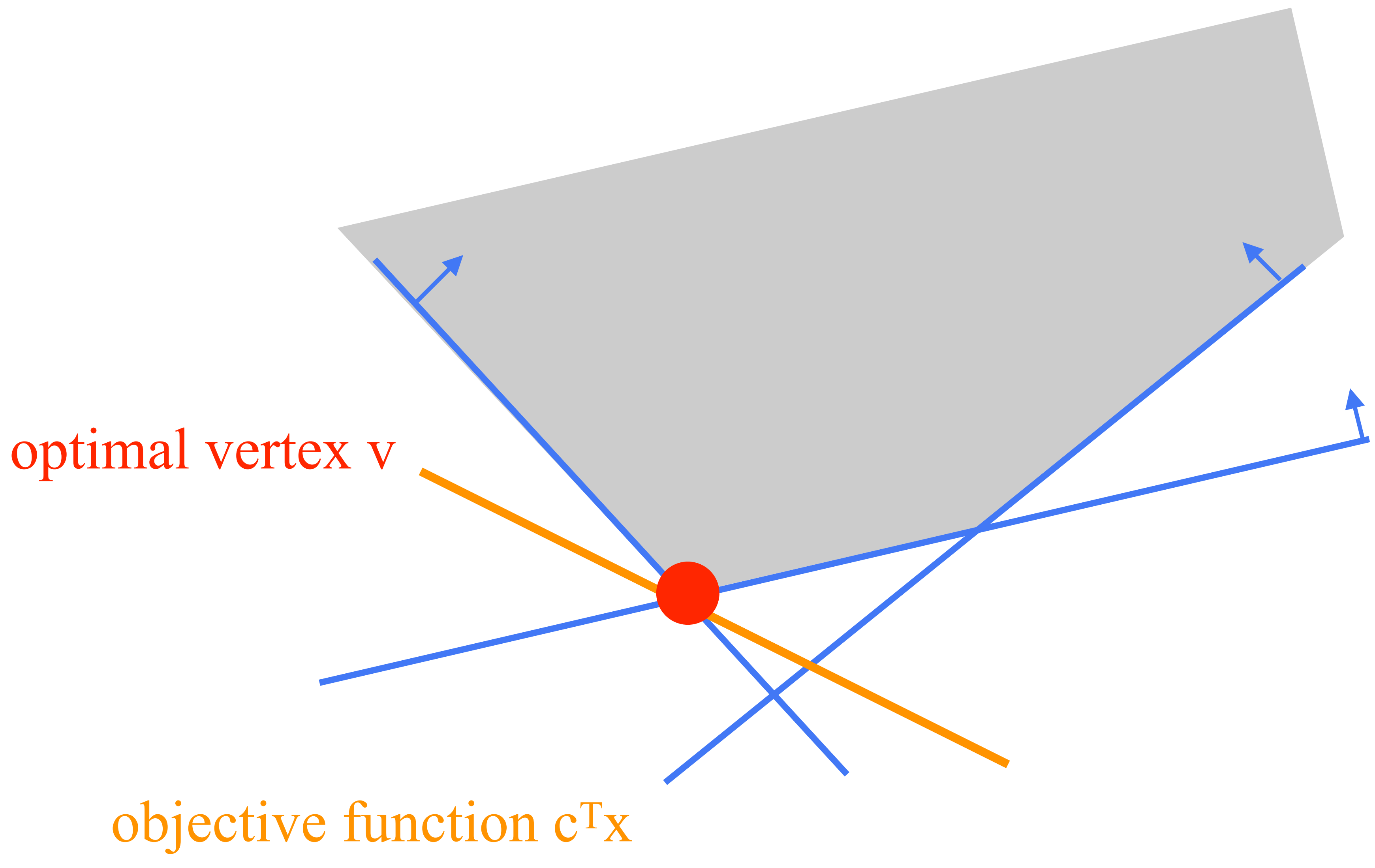


Observations

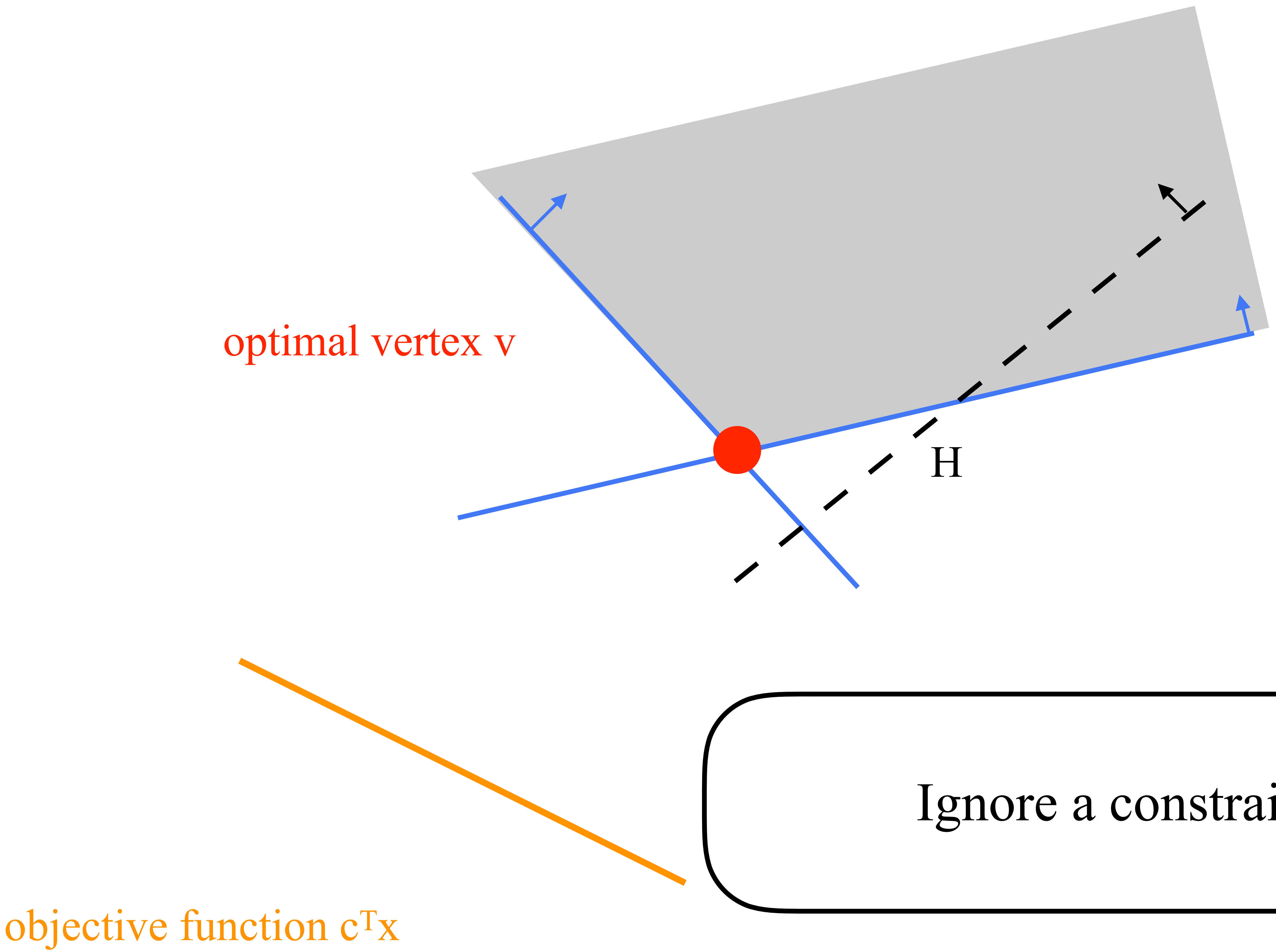


objective function $c^T x$

Observations

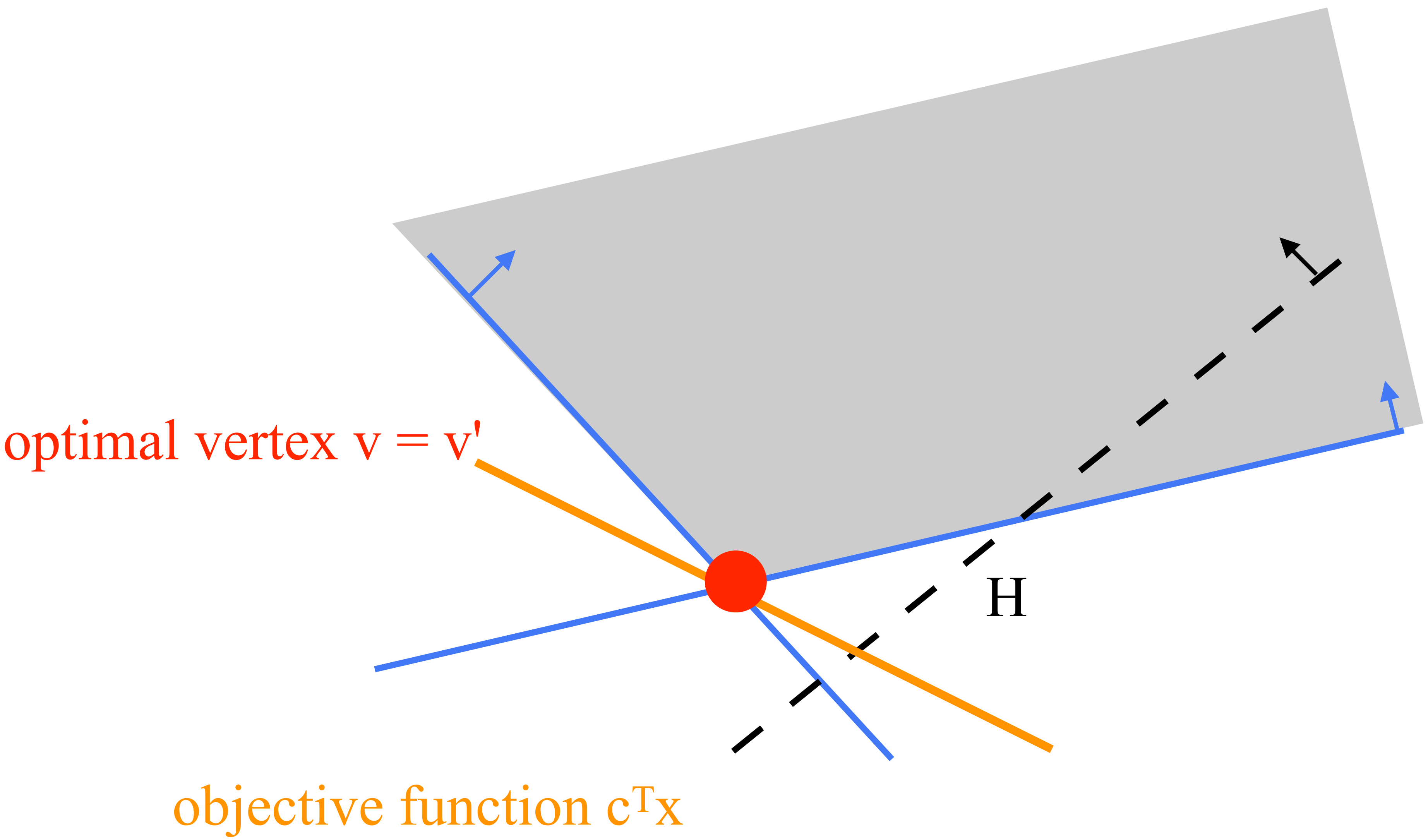


Observations



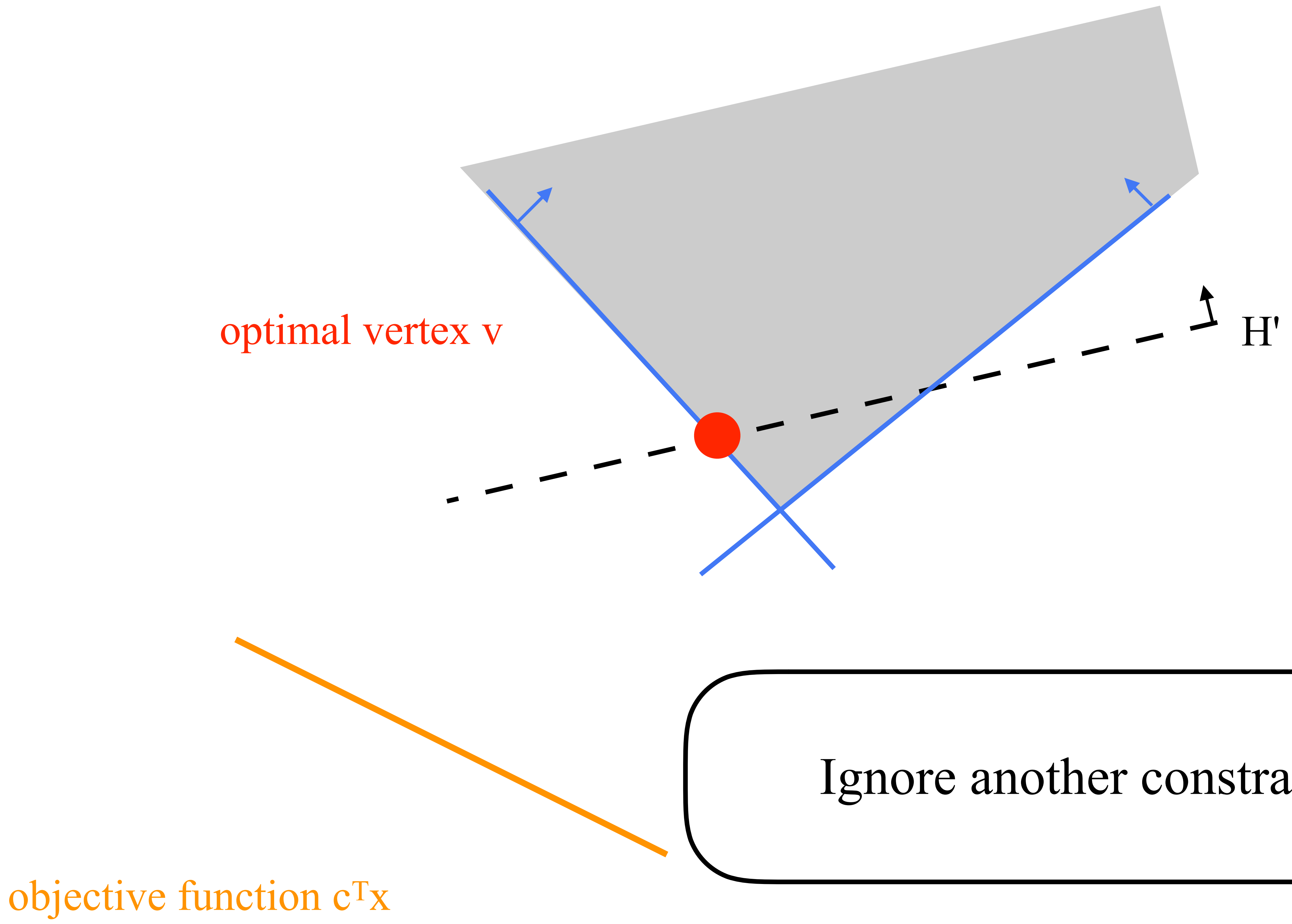
Ignore a constraint H.

Observations



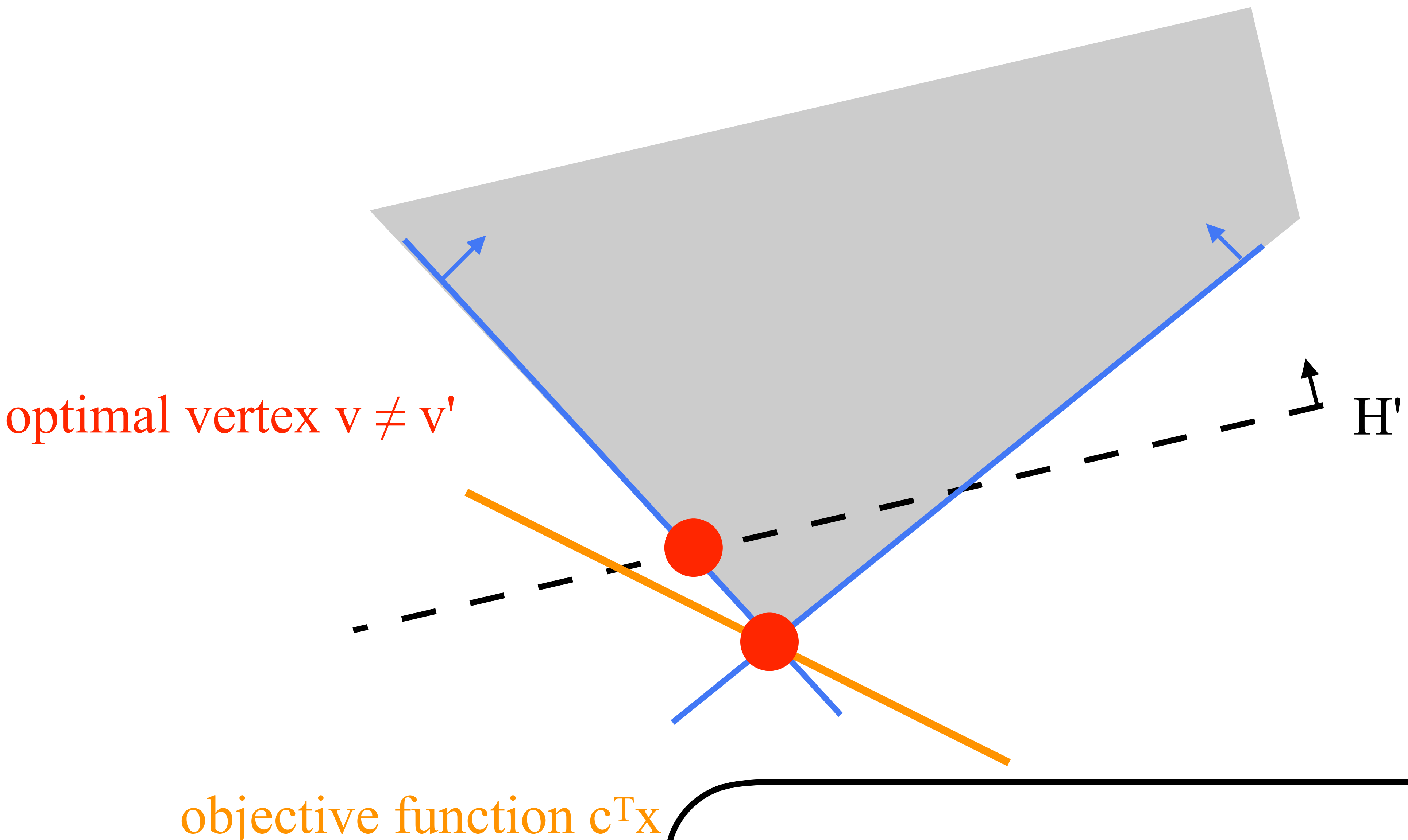
If we ignore a constraint H whose halfspace contains v', then v' = v.

Observations



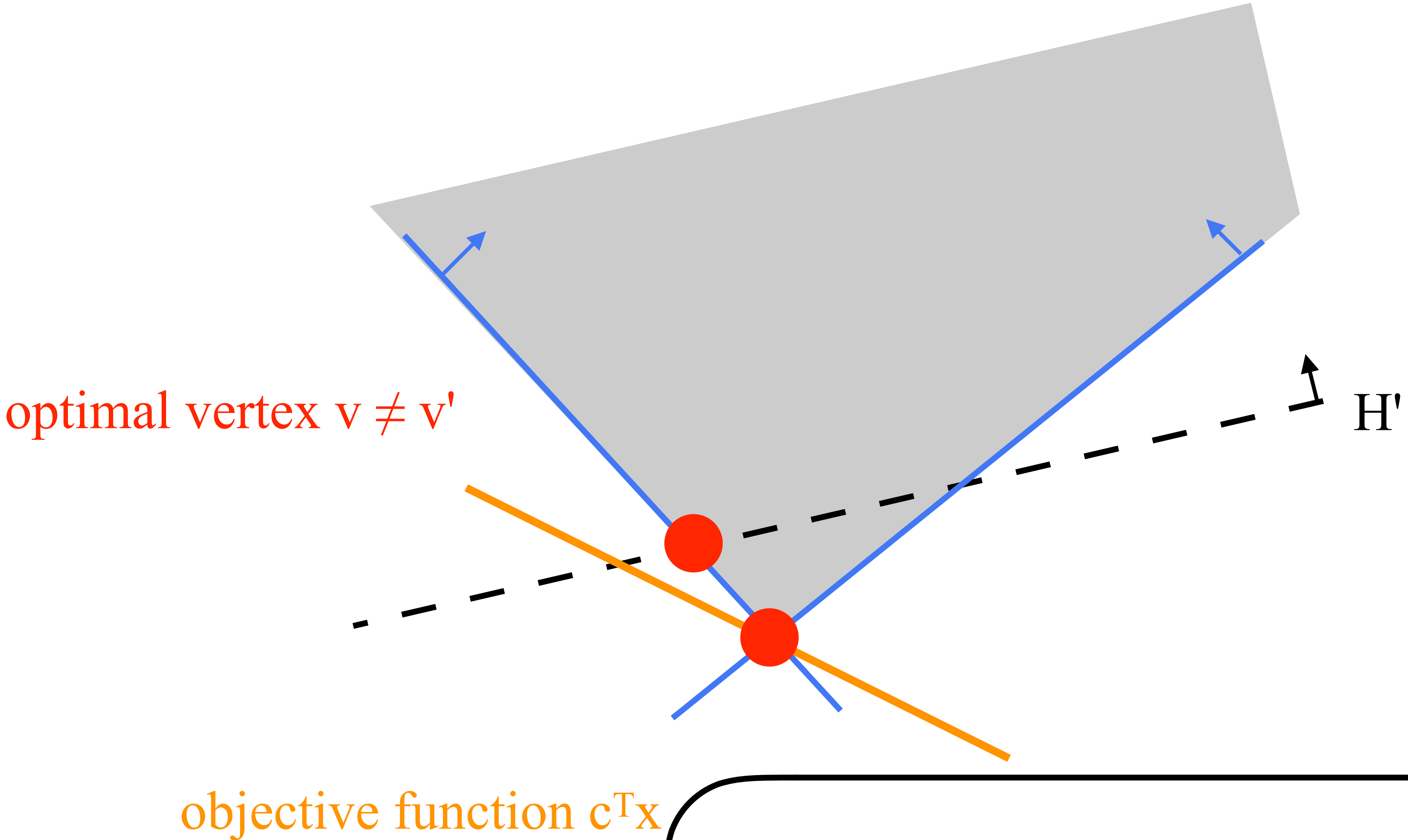
Ignore another constraint H'.

Observations



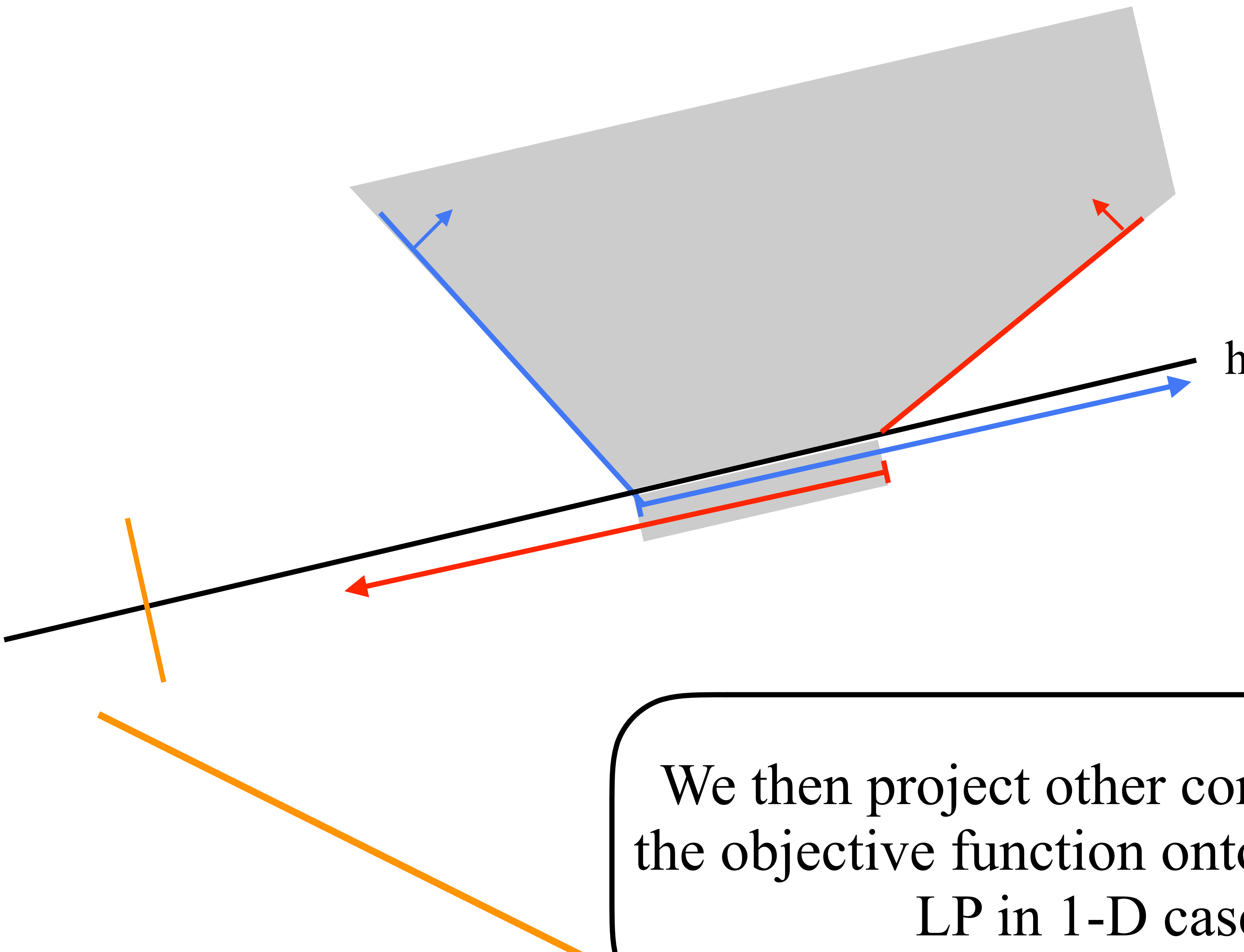
If we ignore a constraint H' whose hyperplane doesn't contains v', then v' ≠ v.

Observations



However, in this case, we know that the optimal vertex v is on the hyperplane described by H'.

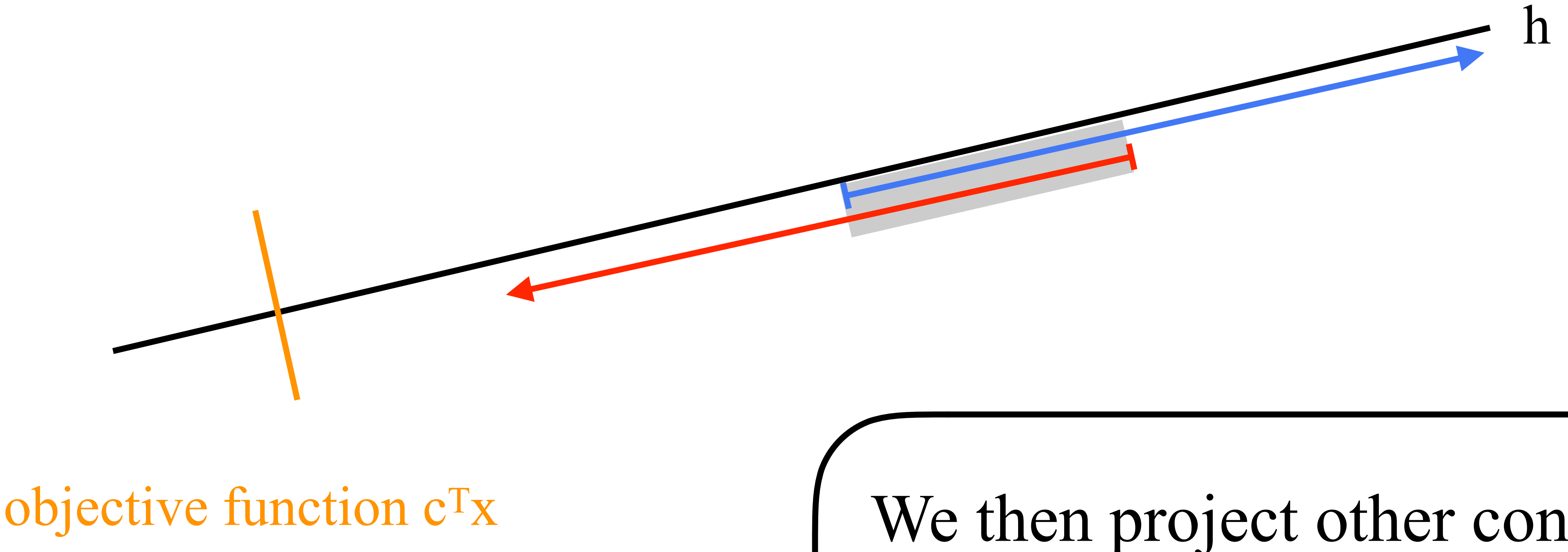
If v is on some hyperplane h



We then project other constraints and the objective function onto h, and solve LP in 1-D case.

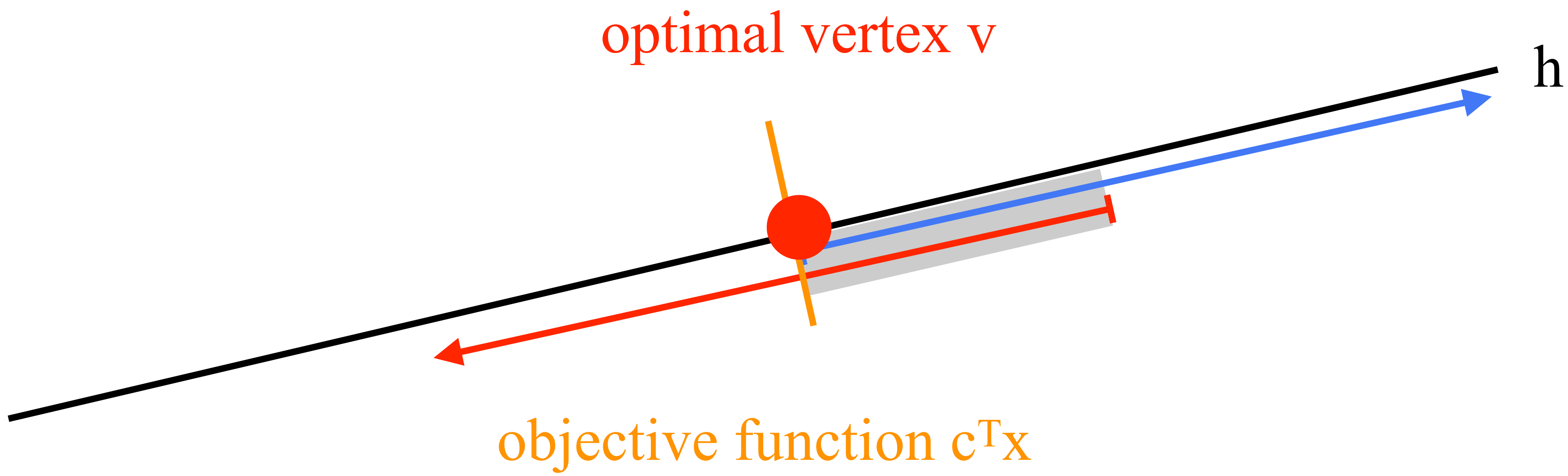
objective function c^Tx

If v is on some hyperplane h



We then project other constraints and the objective function onto h, and solve LP in 1-D case.

If v is on some hyperplane h



We then project other constraints and the objective function onto h, and solve LP in 1-D case.

Pseudocode

```
LP(m, d, c){ // Given an LP of m constraints in d dimension,
return a point v so that cTv is maximized.
    H ← a randomly picked constraint;
    if((v' ← LP(m-1, d, c)) contained in the halfspace of H){
        return v';
    }else{
        h ← the hyperplane described by H;
        return LP(m-1, d-1, c); // the m-1 constraints and the
        objective function are projected on h
    }
}
```

The optimal vertex v is the intersection point of d hyperplanes.
Hence, the else-case happens with probability d/m.

Running time

Let T(d, m) denotes the expected running time of LP(m, d, c).

$$T(d,m) \leq \begin{cases} O(m) & \text{if } d = 1 \\ O(d) & \text{if } m = 1 \\ T(d,m-1) + O(d) + \frac{d}{m}O(dm) + \frac{d}{m}T(d-1,m-1) & \text{otherwise} \end{cases}$$

By the substitution method, one can show that
T(d, m) = O(d!m).

There are some degenerate cases not covered in the slides.
If interested, please refer to "Small-Dimensional Linear Programming and Convex Hulls Made Easy" by R. Seidel.

Algorithms for LP

	simplex	ellipsoid method	interior-point method	Seidel's algorithm
analytically	exponential-time in the worst case	polynomial-time	polynomial-time	expected linear time for n = O(1)
in practice	fast in practice	slow in practice	fast for large n, m	fast for small dimension

Convex Sets

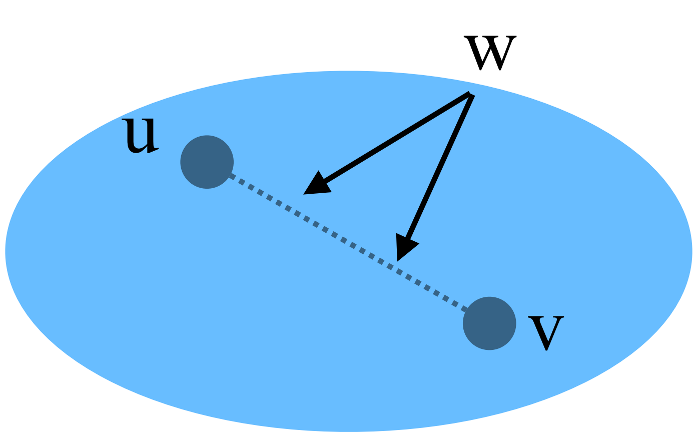
Convex Sets

We say a point w is a convex combination of points u and v if:

$w = \alpha u + (1-\alpha)v$ for some real number α in $[0, 1]$.

Let P be a point set in $\mathbf{R}^d$. We say P is a convex set if:

for any two points u, v in P, if w is a convex combination of u and v, then w is in P.



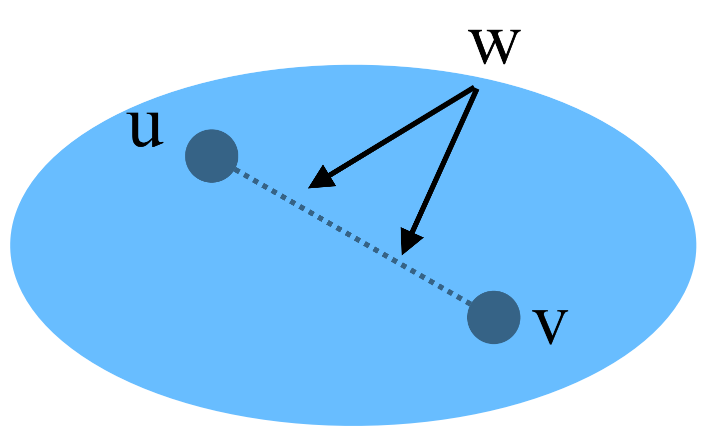
Convex Sets

We say a point w is a convex combination of points u and v if:

$w = \alpha u + (1-\alpha)v$ for some real number α in $[0, 1]$.

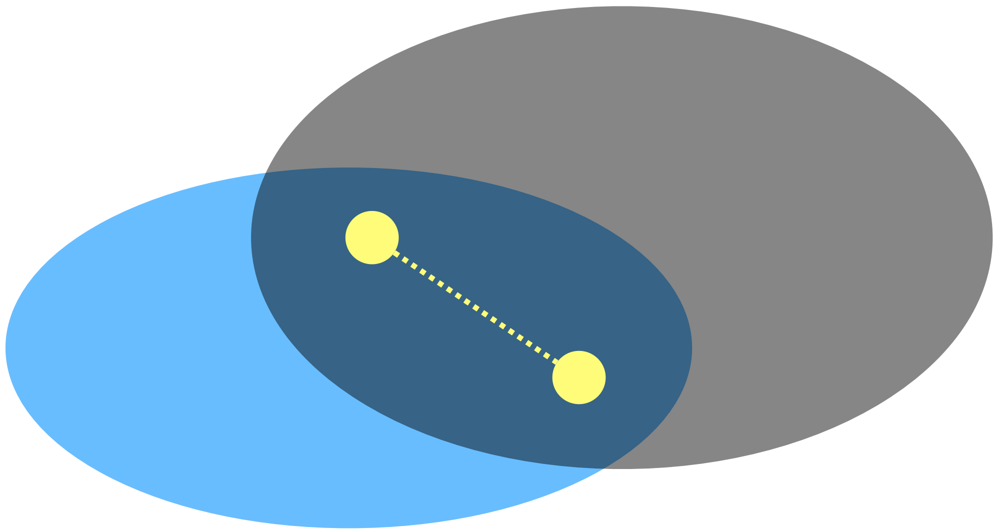
Let P be a point set in $\mathbf{R}^d$. We say P is a convex set if:

for any two points u, v in P, if w is a convex combination of u and v, then w is in P.

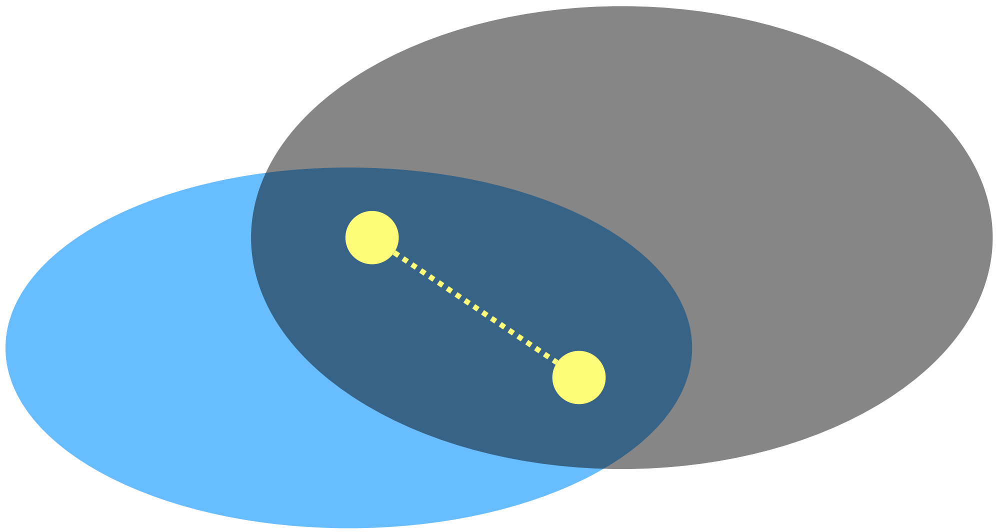


What is a linear combination?

Intersection of Convex Sets

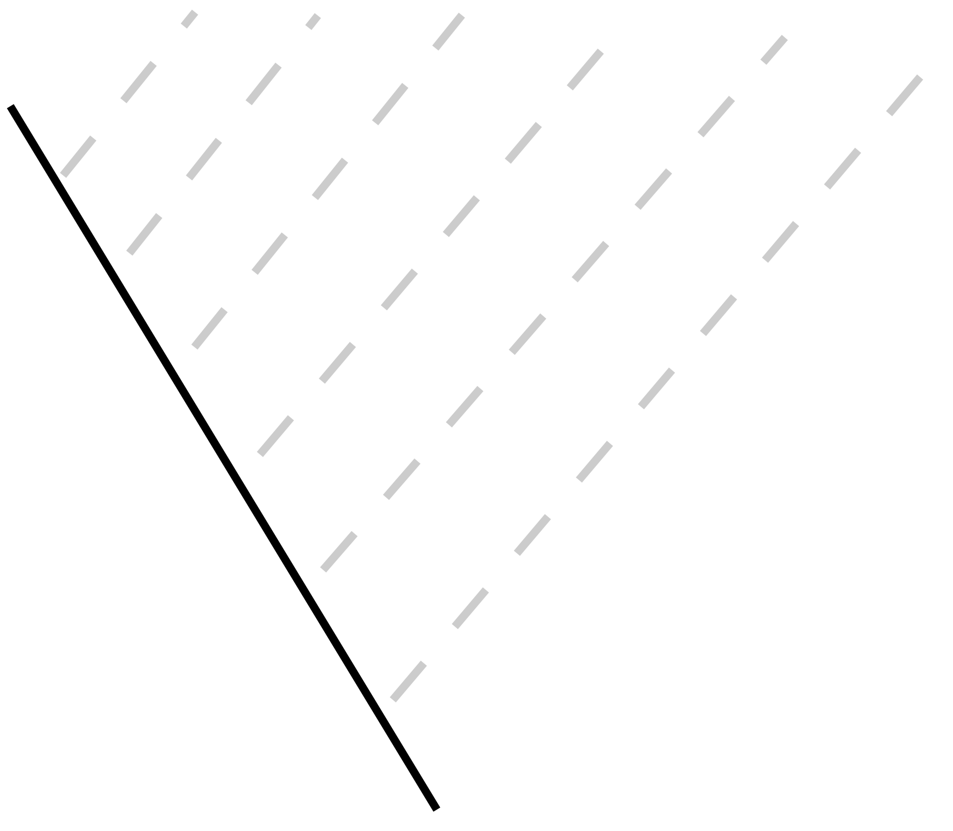


Intersection of Convex Sets

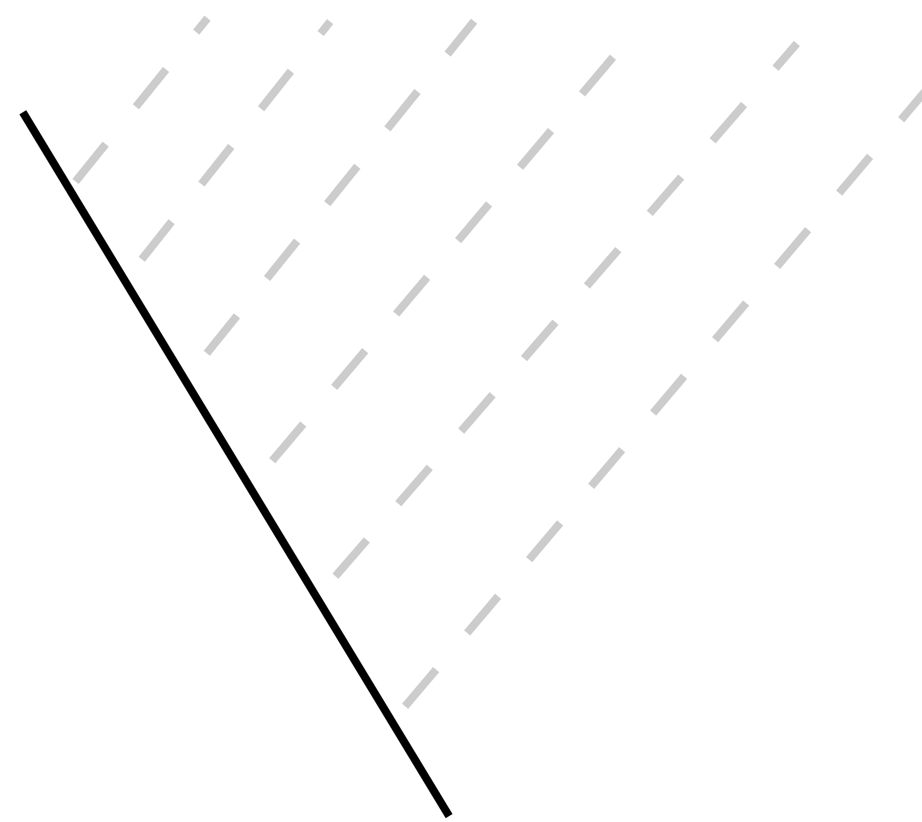


The intersection of convex sets is a convex set.

Halfspace is a convex set



Halfspace is a convex set



The intersection of halfspaces is a convex set. The feasible region of LP is the intersection of halfspaces, and thus the feasible region is a convex set, a polyhedron.

Algebraic view

maximize $\sum_{1 \leq j \leq n} c_j x_j$

subject to $\sum_{1 \leq j \leq n} a_{ij} x_j \leq b_i$ for $i = 1, 2, ..., m$

where c_j's, a_{ij}'s, and b_i's are constants.

Algebraic view

maximize $\sum_{1 \le j \le n} c_j x_j$

subject to $\sum_{1 \le j \le n} a_{ij} x_j \le b_i$ for $i = 1, 2, \ldots, m$

where c_j's, a_{ij}'s, and b_i's are constants.

If x and y both satisfy the m constraints, we have:

(1) $\alpha \sum_{1 \le j \le n} a_{ij} x_j \le \alpha\, b_i$ for $i = 1, 2, \ldots, m$
(2) $(1-\alpha) \sum_{1 \le j \le n} a_{ij} y_j \le (1-\alpha)\, b_i$ for $i = 1, 2, \ldots, m$

$\Rightarrow \sum_{1 \le j \le n} a_{ij}(\alpha x_j + (1-\alpha) y_j) \le b_i$
(any convex combination of x and y is a feasible solution)

Algebraic view

Does the same argument applies to any linear combination? Why?

maximize $\sum_{1\leq j\leq n} c_j x_j$

subject to $\sum_{1\leq j\leq n} a_{ij} x_j \leq b_i$ for $i = 1, 2, \ldots, m$

where c_j's, a_{ij}'s, and b_i's are constants.

If x and y both satisfy the m constraints, we have:

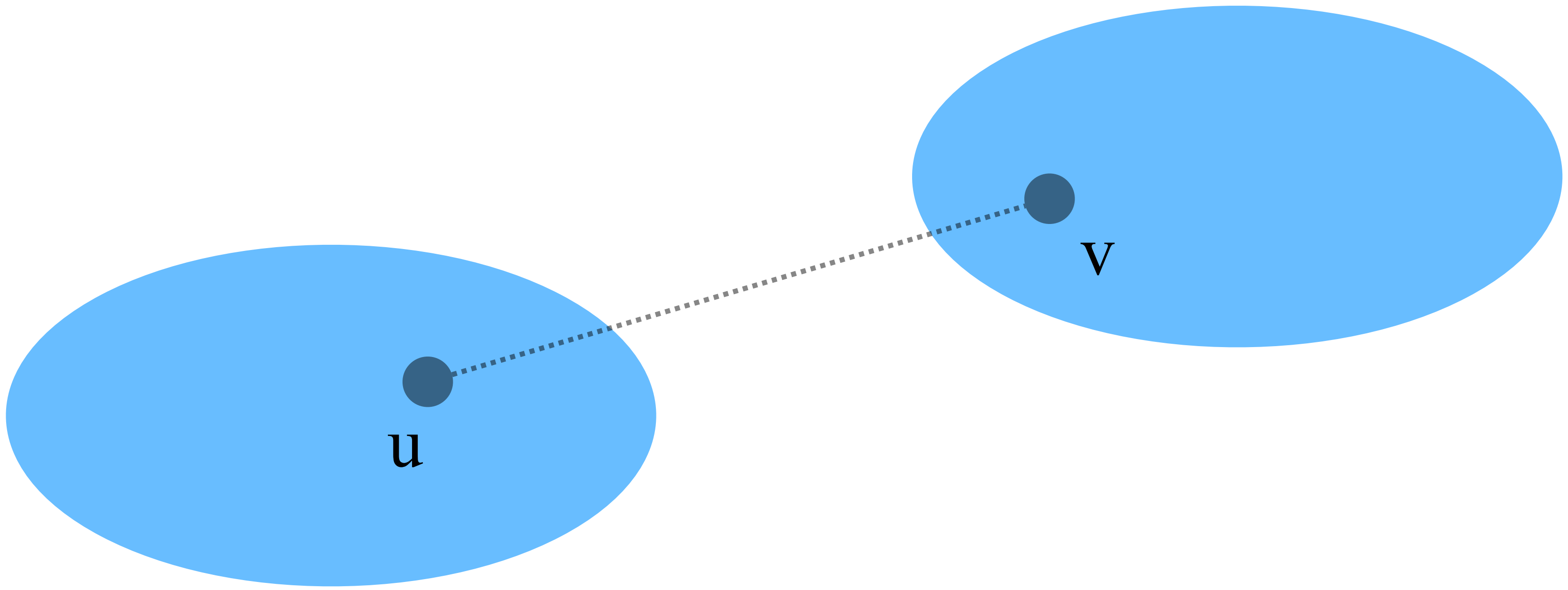
(1) $\alpha \sum_{1\leq j\leq n} a_{ij} x_j \leq \alpha\, b_i$ for $i = 1, 2, \ldots, m$

(2) $(1-\alpha) \sum_{1\leq j\leq n} a_{ij} y_j \leq (1-\alpha)\, b_i$ for $i = 1, 2, \ldots, m$

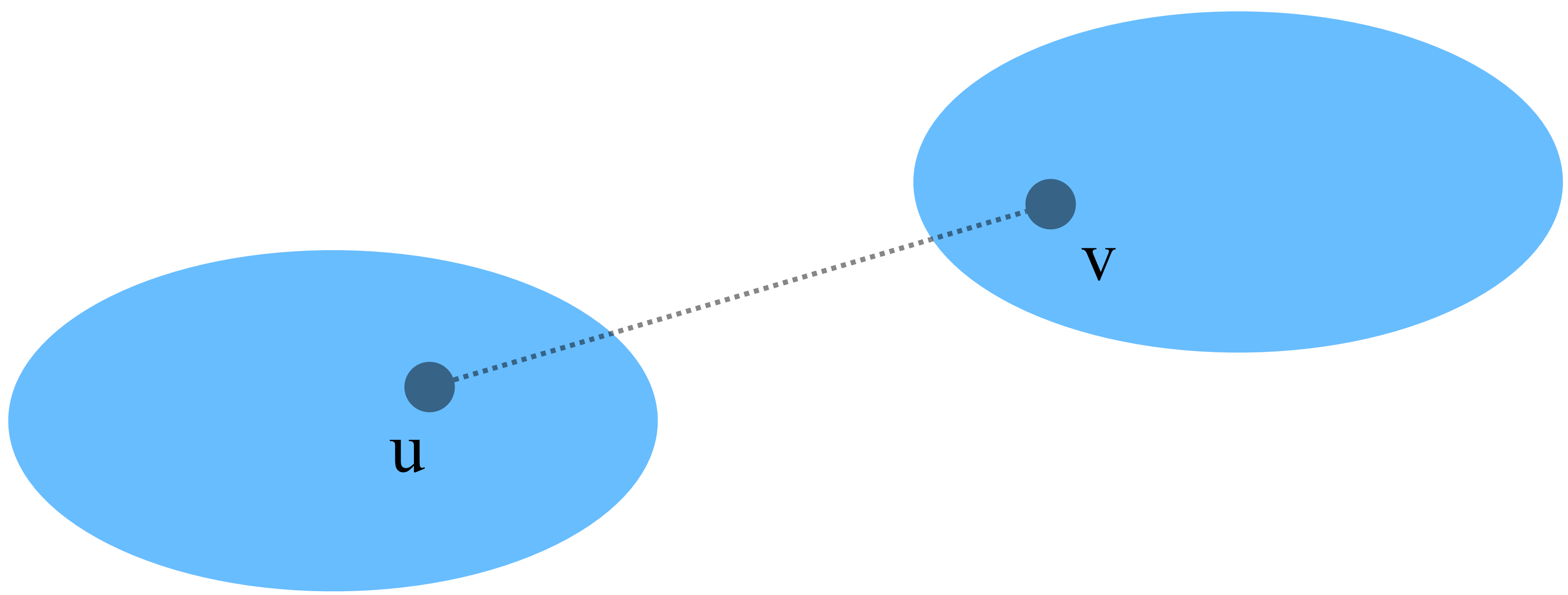
$\Rightarrow \sum_{1\leq j\leq n} a_{ij}(\alpha x_j + (1-\alpha) y_j) \leq b_i$

(any convex combination of x and y is a feasible solution)

Any convex set is connected



Any convex set is connected

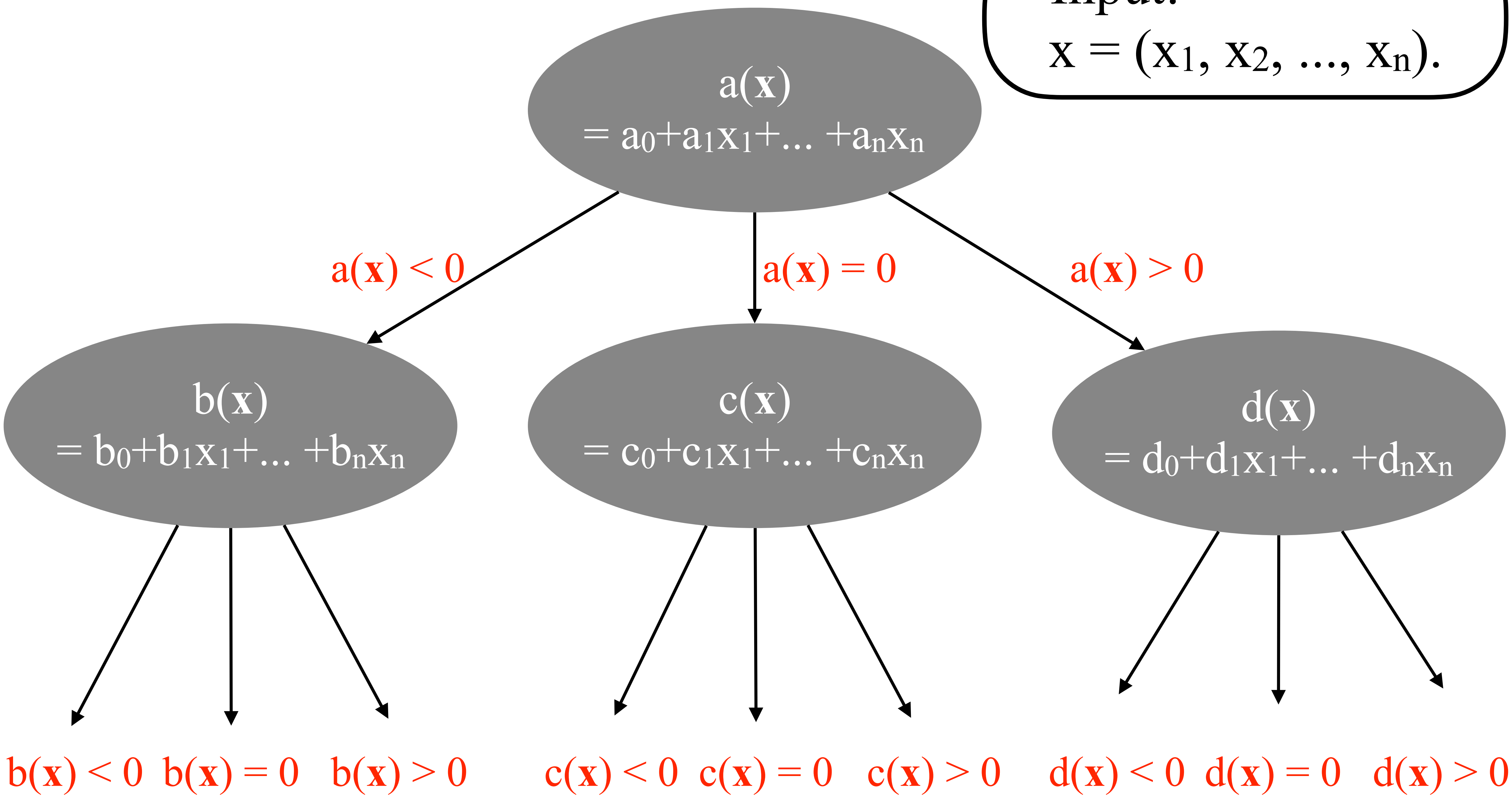


If it is not connected, find a point u in a connected subset and find a point v in another connected subset, and inspect the points that are convex combinatons of u and v, which forms a path from u to v →←.

Linear Decision Trees

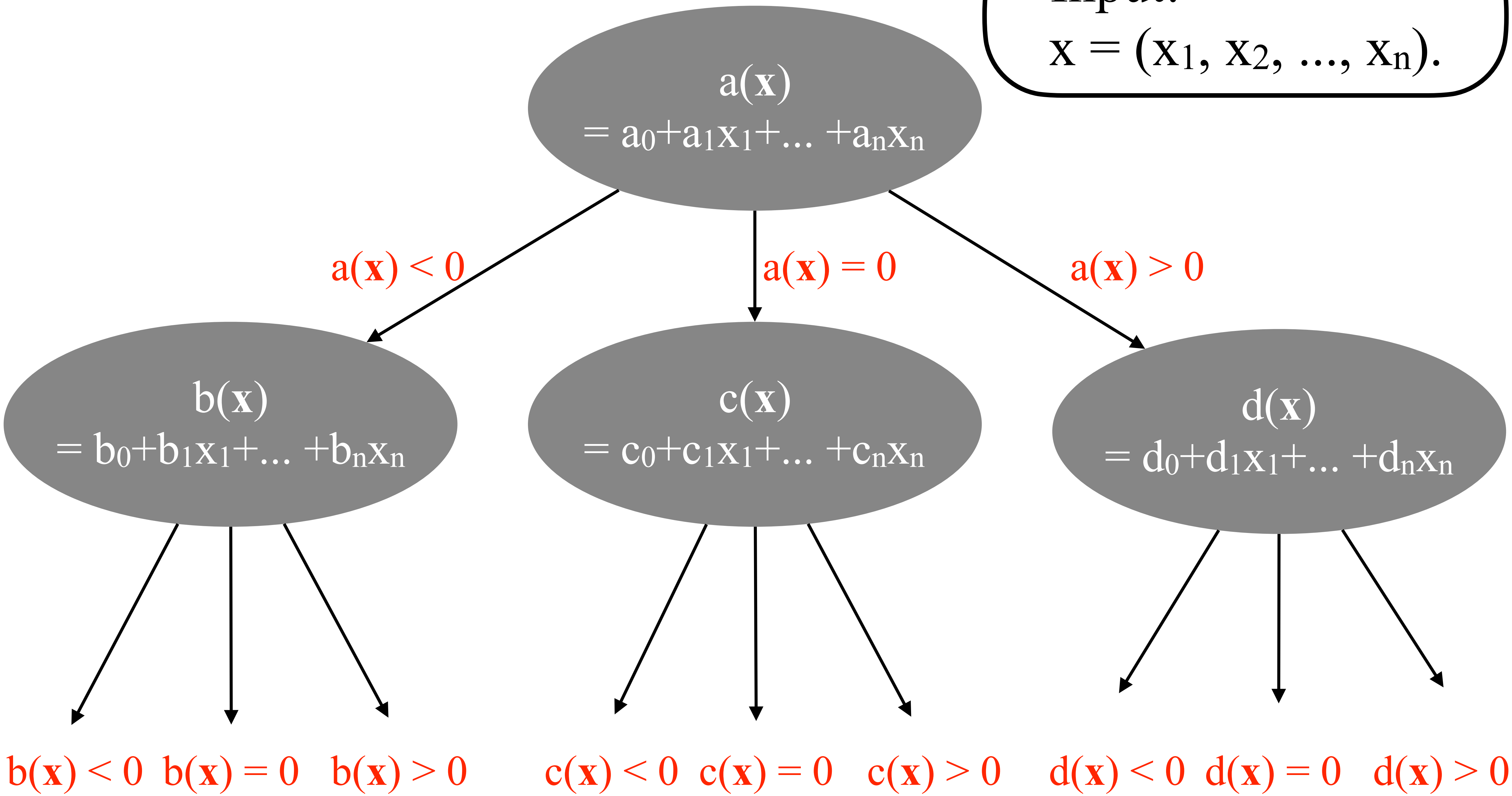
Linear decision trees

Input:
$x = (x_1, x_2, \ldots, x_n)$.



Linear decision trees

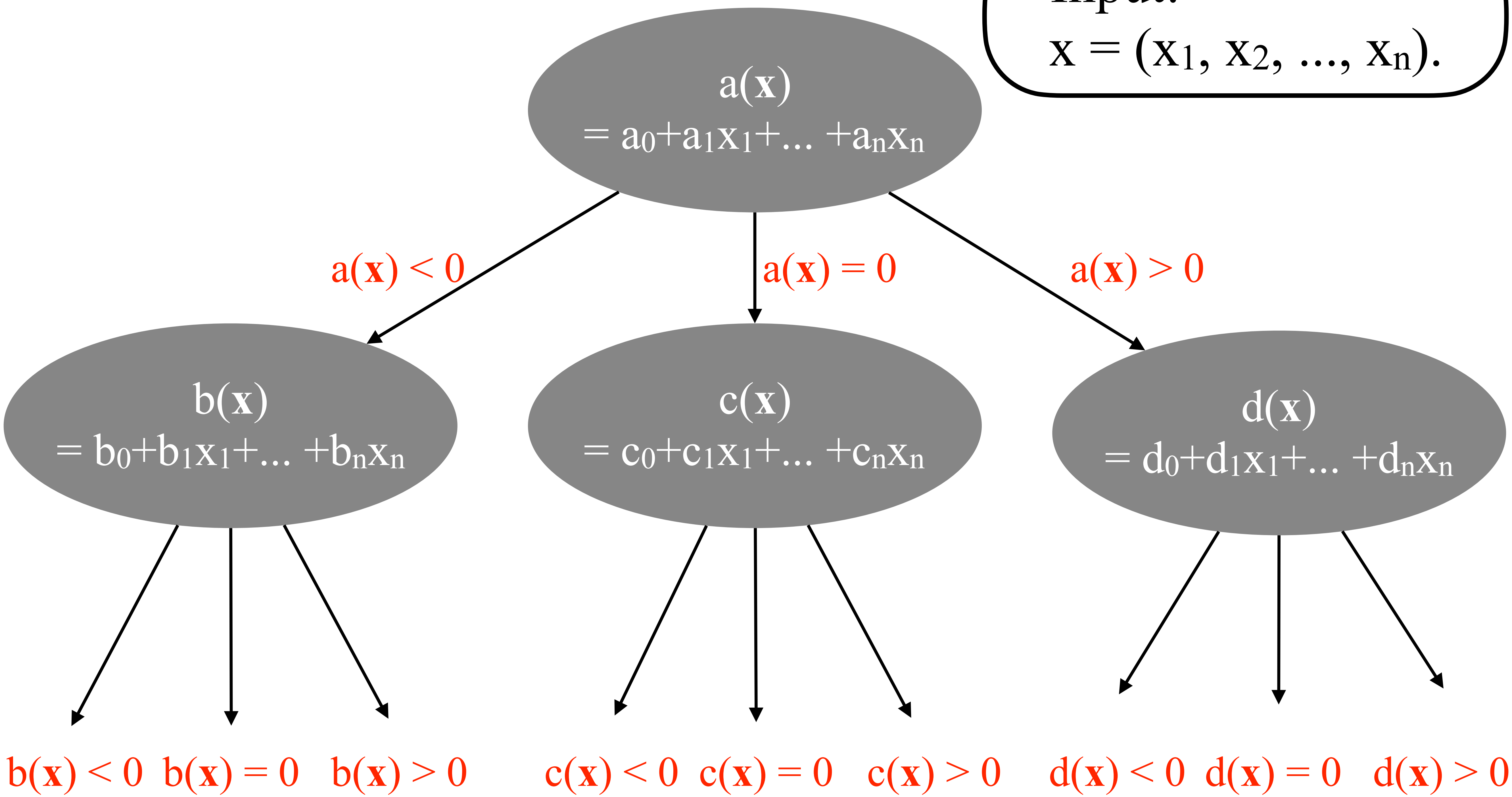
Input:
$\mathbf{x} = (x_1, x_2, \ldots, x_n)$.



By setting $(a_i, a_j) = (1, -1)$ and other a_k's as 0 for each node, it becomes a comparison-based decision tree.

Linear decision trees

Input:
$x = (x_1, x_2, \ldots, x_n)$.



For any node, all the input that can reach it form a convex set.

Element Uniqueness Problem

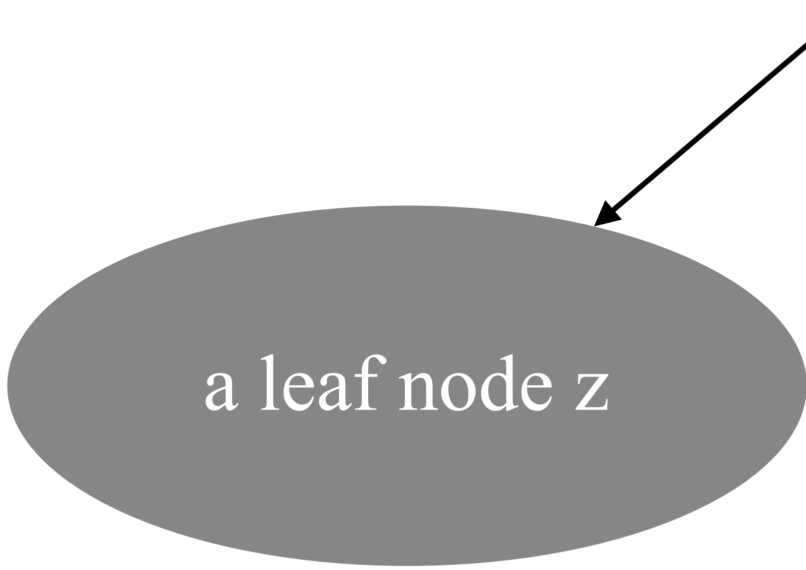
Input: n real numbers $s_1, s_2, \ldots, s_n$

Output: 'No', if $s_i = s_j$ for some $i \neq j$, or 'Yes' otherwise.

This problem has a lower bound $\Omega(n \log n)$
in the linear decision trees model, and also in the
comparison-based model.

Proof

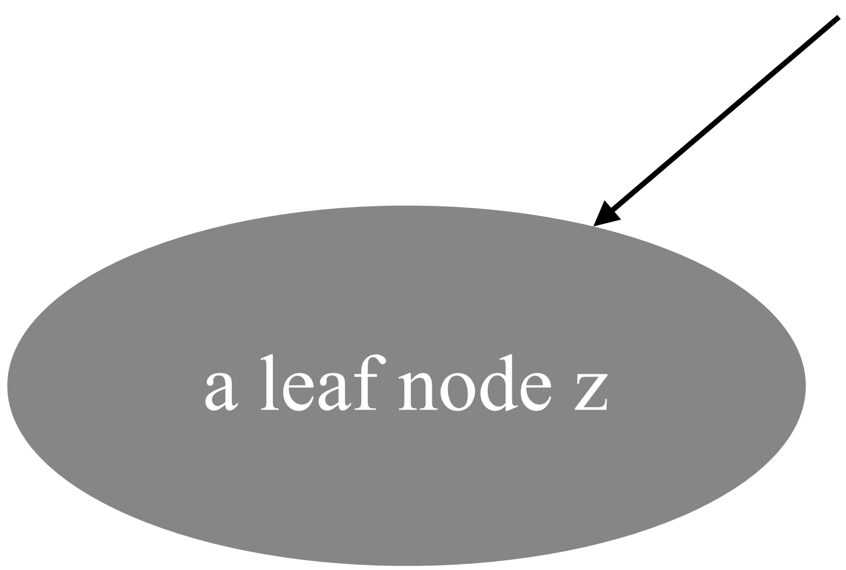
Let T be any linear decision tree that solves the element uniqueness problem, and represent the input $(s_1, s_2, ..., s_n)$ by a point in $\mathbf{R}^n$.



Recall that all the points (inputs) that can reach the leaf node z form a convex set.

Proof

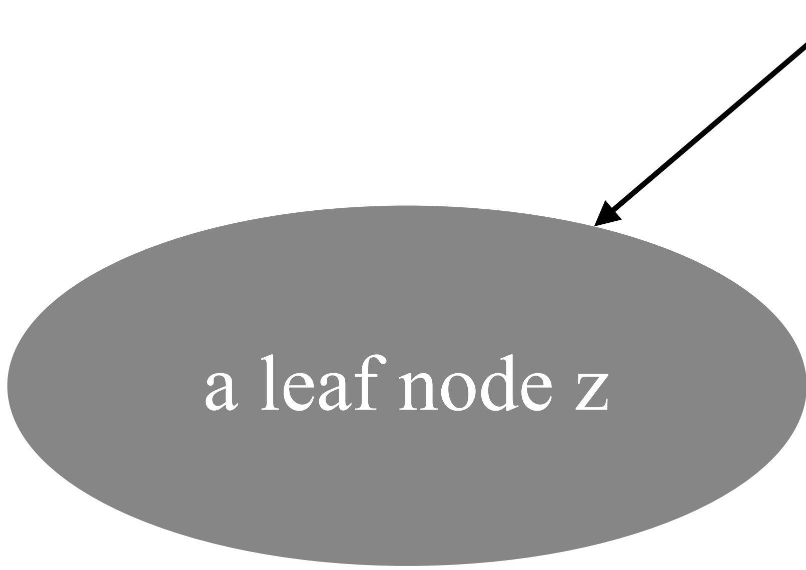
Let T be any linear decision tree that solves the element uniqueness problem, and represent the input $(s_1, s_2, ..., s_n)$ by a point in $\mathbf{R}^n$.



All the points (inputs) that can reach the leaf node z shall return a unified answer, either "Yes" or "No".

Proof

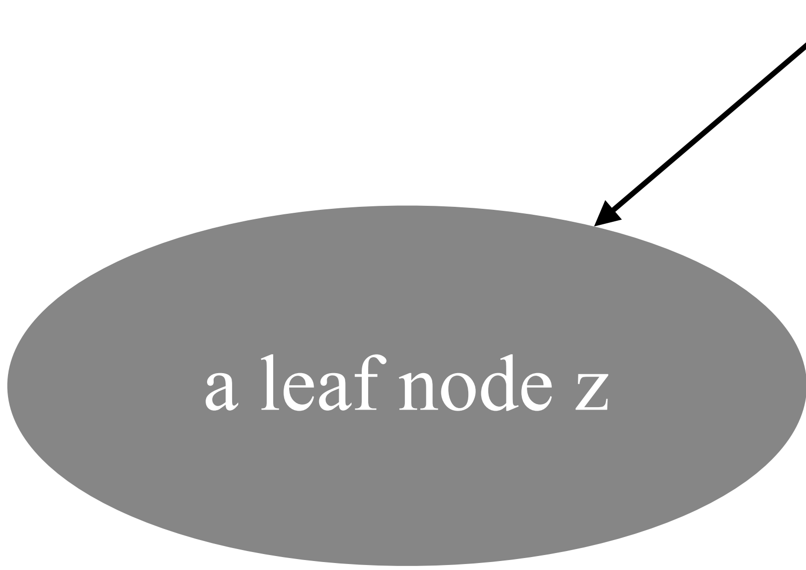
Let T be any linear decision tree that solves the element uniqueness problem, and represent the input $(s_1, s_2, ..., s_n)$ by a point in $\mathbf{R}^n$.



Let x and y be two different inputs (points) so that the coordinates of x is a permutation of {1, 2, ..., n} and those of y is another permutation of {1, 2, ..., n}.

Proof

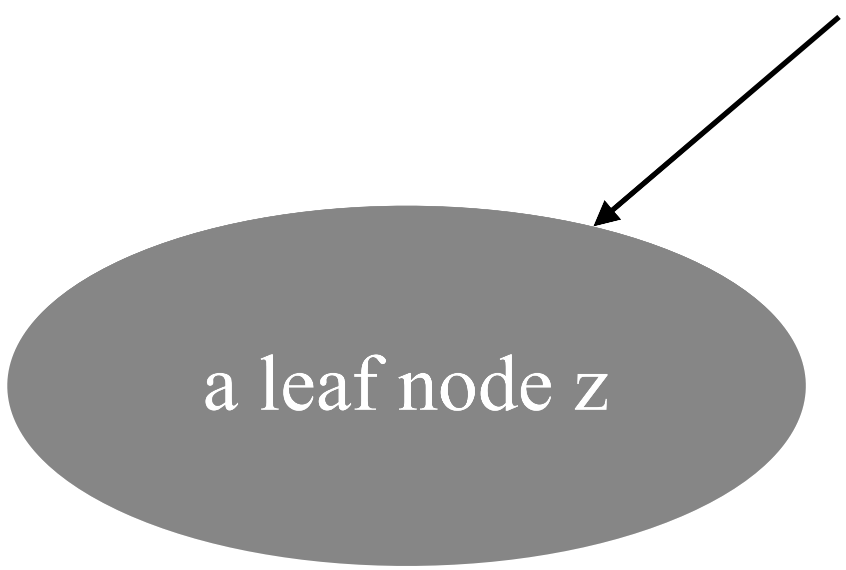
Let T be any linear decision tree that solves the element uniqueness problem, and represent the input $(s_1, s_2, \ldots, s_n)$ by a point in $\mathbf{R}^n$.



Let $x = (x_1, x_2, \ldots, x_n)$ and $y = (y_1, y_2, \ldots, y_n)$. Then, there exist i, j so that $x_i < x_j$ and $y_i > y_j$. Thus, any path that connects x and y pass a point z $(\ldots, z_i, \ldots, z_j, \ldots)$ with $z_i = z_j$.

Proof

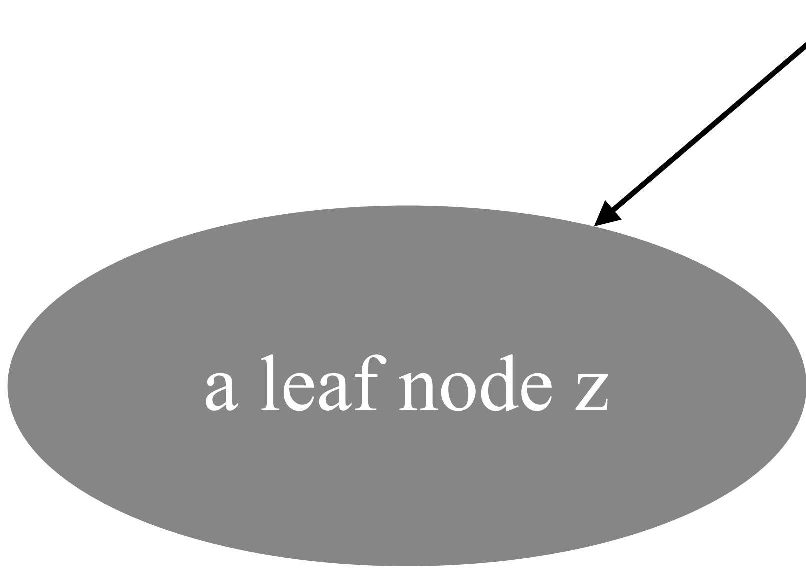
Let T be any linear decision tree that solves the element uniqueness problem, and represent the input $(s_1, s_2, ..., s_n)$ by a point in $\mathbf{R}^n$.



For input x and y, ALGO_T returns "Yes", but for input z, ALGO_T returns "No". ⇒ x and y are disconncted. ⇒ x and y are contained in different leaf nodes.

Proof

Let T be any linear decision tree that solves the element uniqueness problem, and represent the input $(s_1, s_2, \ldots, s_n)$ by a point in $\mathbf{R}^n$.



For any T, T has $\Omega(n!)$ leaves $\Rightarrow$ T has depth $\Omega(n \log_3 n)$
$\Rightarrow$ for some input, ALGO_T requires $\Omega(n \log n)$ time.
This lower bound is tight because sorting runs in $O(n \log n)$ time.