Bachelor's degree in Bioinformatics

Optimization Methods for Computational Biology

Prof. Renato Bruni

bruni@dis.uniroma1.it

Department of Computer, Control, and Management Engineering (DIAG)

"Sapienza" University of Rome

Brief outline of the course

- What is Optimization
- Types of Optimization models
- Solver for Optimization models
- Introduction to Data Mining and Machine Learning
- Heuristic Algorithms for Combinatorial Optimization

Material of the course

- Slides of the course, available from the home page of the professor (http://www.diag.uniroma1.it//~bruni/)
- On the same page you can find the zoom links for the online lessons

Important Advices

- If you need to contact me, use bruni@diag.uniroma1.it and ALWAYS use the subject Course on Optimization Methods
- Always try to understand what we will see, do not learn by heart pretending that you understood
- The slides may be updated during the course, so be sure you are using the latest version

Outline part 1

- What is Operations Research and Optimization
- Using Optimization Models
- Types of Optimization Models
- Linear Programming
- Duality in Linear Programming
- Modeling Techniques

Operations Research

- **Operations Research**: the application of a wide range of mathematical techniques and methods to support decision-making or improve efficiency. AKA Management Science
- The name comes after an "**Operational Research Section**" of the **Royal Air Force** created in 1941 in UK to support military decisions with scientific methods
- They were **very successful** during the course of World War II, otherwise we would not even know their name
- Today it is applied in a **very wide range of fields**, for example:
  - Production planning
  - Investment selection
  - Portfolio optimization
  - Data mining
  - Sequence analysis
  - Personnel scheduling
  - Network design
  - …

Optimization

- **Optimization**: the selection of a best element (with regard to some mathematical criterion) from some set of feasible alternatives
- We search for **maximum** or **minimum** (= **optimal solutions**) of some **objective function**, that is a function expressing our criterion

- Usually we want to **minimize costs** or **maximize gains or performance**
- Minimize costs – example: build a bridge that can be used by **100 cars every hour** by selecting the geometry, the materials, etc. in order to have **minimum cost**
- Maximize performance – example: build a bridge with a **budget of 10 MEuro** by selecting the geometry, the materials, etc. in order to allow the **maximum number of cars**

Why we need OR ?

- Consider the following example, due to George Dantzig (1914-2005), one of the founders of Operations Research
- We have a company with **70 workers** and **70 works** waiting to be done
- For each assignment work-worker, we have a **measure** of how **good** is the assignment.
- Example: worker A → job 1 gives a value of 3,
- worker A → job 2 value 0.5,
- worker B → job 1 value 1, etc.
- We want to **assign** each worker a work, so that each has one, and we want to **maximize the total value** of the assignment

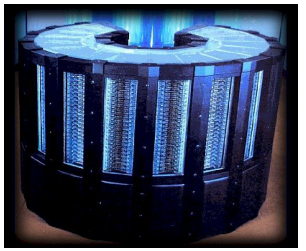
Complete Enumeration

- Problem: assign **70 workers** to **70 works**
- This may look easy; let's solve it by **complete enumeration**: we compute the value of each solution and chose the best
- **How many** solutions? For the first worker we chose among 70 works, for the second among 69, and so on.
- Total: 70 x 69 x … x 1 = **70!** (permutations of 70 elements, $\approx 10^{100}$)
- How long does it take to **compute all** the 70! solutions and **chose the best one**?

Computation

- **How long** does it take to compute all solutions and chose the best? Dantzig estimated this:

  - If we try using a **1MHz computer** (now obsolete, at that time futuristic)
  - We start it. After a few days it has not finished, so we stop it
  - Maybe we just need more time: let's give it all the time from **Big Bang** up to now. Has it finished now? **No**!



  - So we take a **1THz supercomputer** (yet to come) and give it all the time form Big Bang up to now. Is it enough? **No**!!

  - So we cover the **whole Earth** of these supercomputers working in parallel. Moreover, we give them all the time form Big Bang up to now. Is it enough now? **Again No**!!
  - So **what do we need**?? According to Dantzig, we need 10^{40} Earths covered in supercomputers and working since the Big Bang!

So how do we solve?

- So, the problem of 70 workers and 70 jobs is **too difficult** to be solved? We just abandon all hopes?
- **No**. Similar problems are solved **very easily** now, by many organizations
- How do they do? They do not use complete enumeration!
- Instead, they use the **right models and algorithms**
- Complete enumeration is computationally inapplicable, unless we have very small problems…
- We will learn **how to chose** the right models and algorithms
- We will understand what is computational complexity

Algorithms

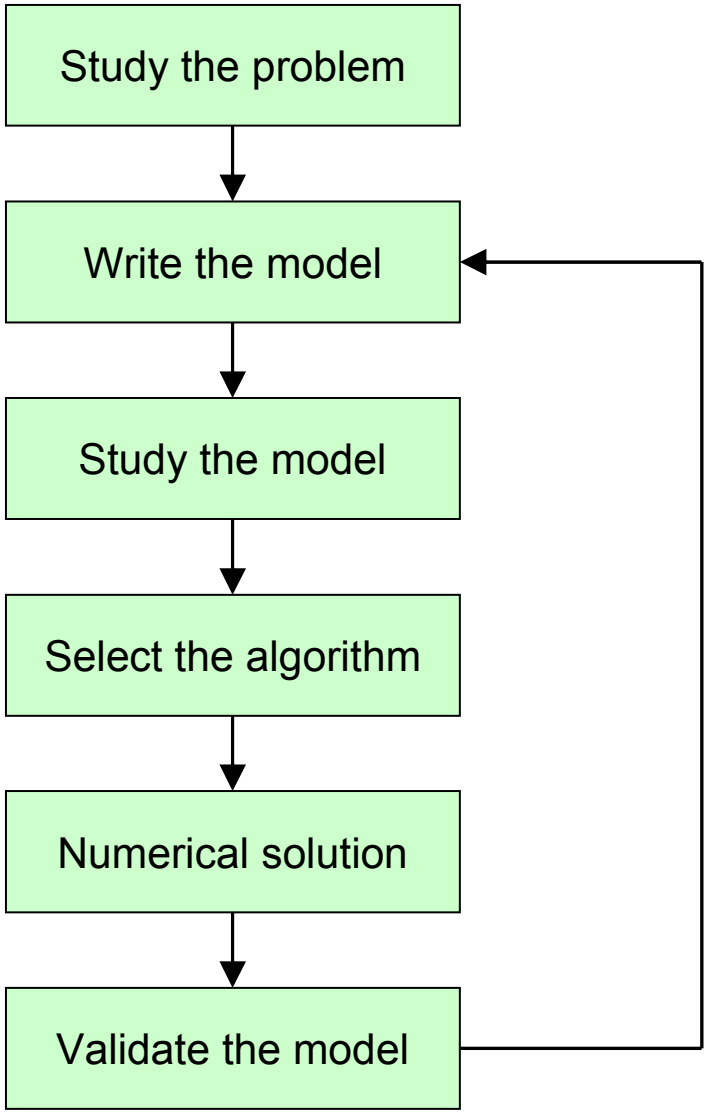
- What is an **algorithm**? Algorithms are what make it possible for computers to solve problems.
- One possible definition:
- "An algorithm is a procedure, (a finite set of well-defined instructions) for accomplishing some task which, given an initial state, will terminate in a defined end-state"
- An algorithm must obviously be **correct** (= not giving wrong answers)
- But it should also be **efficient** (= not wasting computational resources, which in computers are **time** and memory **space**)
- In many cases, a correct but inefficient algorithm **cannot solve** a problem in practice

Using an optimization model

- When we have a problem, usually we can use **2 approaches**
- 1) study the specific problem; invent an algorithm for that specific problem; solve it
- 2) study the specific problem; make a **model** representing the essential of that problem but forgetting about all inessential aspects; **study the model** obtained; **take a known algorithm** to solve that type of model, solve it.
- Which one is better? If we use 1) we need to **start from scratch** for every new problem, and the algorithm we invent may be correct but inefficient
- If we use 2) we can **take advantage** of the work of many researchers worldwide and use an efficient algorithm, maybe even already implemented in some programming language

How to make a model ?

- We may follow this approach:
- Real world relationships among quantities are converted into **mathematical relationships**
- The algorithm must be selected from the ones available for that type of model
- When we compute a numerical solution, we **evaluate if it is reasonable**. If it is not, we probably forgot some essential aspect of the problem in the definition of the model. We need to **go back to model definition** and solve again
- Example: we obtain a negative value for something that must be >=0? We forgot to specify non-negativity in the model



Advantages of the model

- We use the **power of mathematics** to find a solution
- We may mathematically **discover important properties** of the practical problem (for example, we discover that a quantity a is always 3.5 b, and this was previously unknown)
- We may use mathematical **simulations** (for example, we do need to build a bridge and see whether it falls down or not, we simulate its behavior)

- **Criticisms** to the use of mathematical models
- the quality of the answer depends on the **quality of the data** (garbage in, garbage out) but this is inevitable
- Not everything can be **quantified** (for example, subjective evaluations). However, we can do our best…

Optimization models

- The **general form** of an optimization model is $\begin{cases} \min f(x) \\ x \in S \end{cases}$

f is called **objective function**, it is a function $f: \mathrm{R}^n \rightarrow \mathrm{R}$ and represents what we want to maximize or minimize

x is the set of **decision variables** (typically a vector $x_1, \ldots, x_n$), and represents what we can decide of the problem

S is a set in the space of the x called **feasible set**, and represents all the set of decisions (= all the points x) which are admissible

- We just write the minimization form because every maximization can be expressed as a minimization: $\min f(x) = - \max(- f(x))$

Optimal Solutions

Given $\begin{cases} \min f(x) \\ x \in S \end{cases}$

Every x in the feasible set S is called **feasible solution**

If there exists at least one $x^* = x^*_1, \ldots, x^*_n \in S$ such that $f(x^*) <= f(x)$ for all $x \in S$, then:

$x^* = x^*_1, \ldots, x^*_n$ is a global minimum, and it's called **optimal solution**

and $f^* = f(x^*)$ is the **optimal value of the objective function**

and the problem is said to have an optimal solution. In this case, it may either have **only one** optimal solution, **or more than one** optimal solution, which will however give the same optimal value to the objective function (by definition)

Do we always have optima?

- There exist problems **without** optimal solutions

If there is no optimal solution, then the problem may be:

- **infeasible**: there is not even one feasible solution, in other words the set S is empty! (example: we want to build a bridge holding at least 100 cars, and we have a budget of 3 Euro. Are there solutions? Probably not)
- or **unbounded**: given any solution, there will always be a better one, so it is impossible to say that one solution is optimal (example: how much money do we want as a gift?)

Example 1

- A company makes 3 **products** called P1, P2, P3
- Each Kg of product needs some **material**, called M1 and M2, according to the following table

	M1	M2
P1	2	4
P2	1	5
P3	3	4

- Material M1 is only **available** in 50 kg per week, material M2 in 80 kg per week
- P1 is **sold** at 10 euro/Kg, P2 at 14 euro/Kg, P3 at 18 euro/Kg
- We want to plan our production in order to **maximize income**

Building the first model 1/3

- How can we solve? The first step is understanding what we can decide
- Can we simply increase prices and earn a lot? **No**, the prices are fixed **by the market**, if we increase them we just don't sell
- We can **only decide** how much of P1, P2, P3 we produce, so we use these 3 decision variables:
- $x_1 =$ Kg of P1 produced every week
- $x_2 =$ Kg of P2 produced every week
- $x_3 =$ Kg of P3 produced every week
- We use Kg per week since all numbers are expressed so, otherwise we convert to any reasonable common measure

Building the first model 2/3

- Now, we write our objective as **maximize our weekly income**, but expressed as a function of the unknown values of our production

$$\max f(x) = \max (10\, x_1 + 14\, x_2 + 18\, x_3)$$

- In other words, we **don't know** how much we will earn every week, because we still don't know how much of each product we will produce, but **we can express it** using our decision variables
- Now, are we done? To answer this question every time we are writing a model, let's try to solve the problem as it is now. What happens? The problem seems unbounded. However, in reality it is clearly not so: we cannot produce at will. We need to consider the **limits** on the materials

Building the first model 3/3

- We write that the amount of M1 used every week **cannot be more** than 50 kg. Again, we do not know how much M1 we will use, because we still don't know how much of each product we will produce, but we can **express it** using our decision variables

$$2\,x_1 + 1\,x_2 + 3\,x_3 \leq 50$$

- Similarly, we write that the amount of M2 used every week **cannot be more** than 80 kg

$$4\,x_1 + 5\,x_2 + 4\,x_3 \leq 80$$

- Moreover, we need to say that the decision variables are in this case **real non-negative values** (we can produce also fractions of Kg but no negative values)

$$x_i \geq 0 \quad \text{for} \quad i = 1, 2, 3 \qquad\qquad x_i \in \mathrm{R} \quad \text{for} \quad i = 1, 2, 3$$

Our first model

- By using all the pieces we obtain our **model** of the problem

$$\begin{cases} \max \; 10\,x_1 + 14\,x_2 + 18\,x_3 \\ 2\,x_1 + 1\,x_2 + 3\,x_3 \le 50 \\ 4\,x_1 + 5\,x_2 + 4\,x_3 \le 80 \\ x_i \ge 0 \quad \text{for} \quad i = 1, 2, 3 \\ x \in R^3 \end{cases}$$

  - All the expressions defining the feasible set are called **constraints**
  - Below the constraints, we write the **domain** of the variables (for example real values, integer numbers, etc.)
  - when the domain is simply real numbers, it is often omitted

- Note that this is not the solution; it is just the complete expression of the **essential elements** of the practical problem
- Now we could select an **algorithm** for a model of this type (it is called a linear programming model) and solve it to find $x^*_1 \; x^*_2 \; x^*_3$

Example 2

- We have a **budget** of 100.000 Euro and we need to invest it
- We find 4 possible investments A, B, C, D, each having a cost and a revenue

	A	B	C	D
cost	20.000	30.000	15.000	50.000
revenue	35.000	40.000	40.000	70.000

- Each investment can either be fully done or not done, it cannot be done partially and it cannot be done more than once
- We want to choose which investments we do in order to **maximize revenue**

Building the second model 1/2

- With only 4 investments we could simply **think** and **see** the solution
- However, if we had **400 investments**, we could not do that!
- So we need to **build** again a model
- What is the aspect we can **decide**? The revenue of each investment is already estimated, we cannot change it
- For each investment, we can **choose** whether we **do that** or **not**
- The **variables** of our model will be
  - $x_i = \begin{cases} 1 & \text{if we do investment } i \\ 0 & \text{otherwise} \end{cases}$
- These variables **cannot be real numbers**, they must be **binary**: either 1 or 0

Building the second model 2/2

- Now, we write our objective as **maximize the total revenue**, expressed as usual as a function of the variables

$$\max 35000\, x_{\mathrm{A}} + 40000\, x_{\mathrm{B}} + 40000\, x_{\mathrm{C}} + 70000\, x_{\mathrm{D}}$$

- Note that, when selecting the variables, we usually associate 1 to **action done**, 0 otherwise, and not for example the other way around, or +1 and -1, etc. This allows us to write the model as the above sum of products, **easily readable**
- Do we have **constraints**? Yes, we need to say that we must respect our budget: the total cost of investments done cannot exceed 100000

$$20000\, x_{\mathrm{A}} + 30000\, x_{\mathrm{B}} + 15000\, x_{\mathrm{C}} + 50000\, x_{\mathrm{D}} \leq 100000$$

Our second model

- In conclusion, here is our second model

$$\begin{cases} \max 35000\, x_A + 40000\, x_B + 40000\, x_C + 70000\, x_D \\ 20000\, x_A + 30000\, x_B + 15000\, x_C + 50000\, x_D \leq 100000 \\ x \in \{0,1\}^4 \end{cases}$$

Note the binary variables! This domain can **equivalently** be written

$$x \in \mathrm{B}^4 \quad \text{or} \quad x_i \in \{0,1\} \quad \text{for} \quad i = 1, 2, 3, 4$$

- This model is called **binary linear programming** and it is a subcase of **integer linear programming** (ILP)
- This type of models are generally much **more computationally difficult** than the corresponding linear programming (LP). They require specific algorithms

The Solutions

$$\begin{cases} \max\ 35000\,x_A + 40000\,x_B + 40000\,x_C + 70000\,x_D \\ 20000\,x_A + 30000\,x_B + 15000\,x_C + 50000\,x_D \ \le\ 100000 \\ x \in \{0,1\}^4 \end{cases}$$

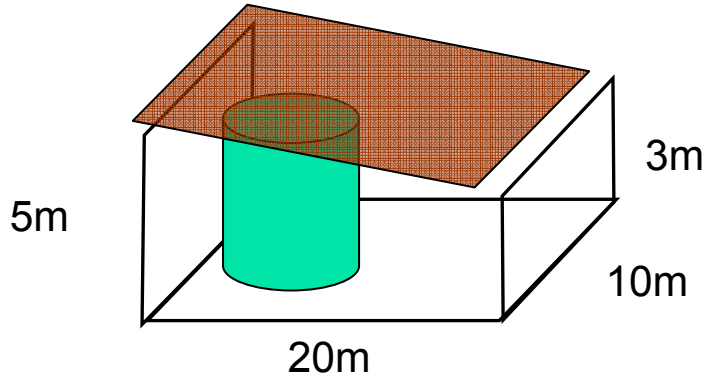
- Since this is a very small example, it is easy to see which are the **feasible** and the **optimal** solutions
- Of course, **this cannot be done** for real-world problems because they are usually **much larger**
- The **feasible** set is composed of 15 binary points (only 1,1,1,1 is not feasible)

$x = (0,0,0,0)$	$x = (0,1,0,1)$	$x = (1,0,1,0)$
$x = (0,0,0,1)$	$x = (0,1,1,0)$	$x = (1,0,1,1)$
$x = (0,0,1,0)$	$x = (0,1,1,1)$	$x = (1,1,0,0)$
$x = (0,0,1,1)$	$x = (1,0,0,0)$	$x = (1,1,0,1)$
$x = (0,1,0,0)$	$x = (1,0,0,1)$	$x = (1,1,1,0)$

- There is 1 **optimal** solution: $x^* = (0,1,1,1)$ with value $f(x^*) = 150000$

Example 3

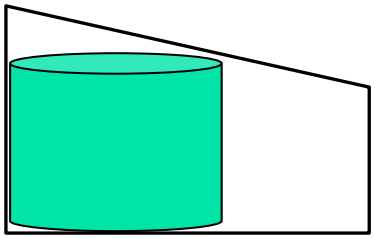
- We need to build a cylindrical **silo** inside a rectangular room. The room is 10m x 20m and has a pitched roof that slopes down from 5m to 3m



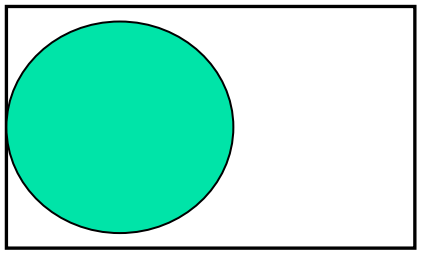
- We have 200 mq of flat material to build the silo
- We want to design the silo in order to maximize its capacity

Building the third model 1/4

- As usual, we need to understand which our **decisions** are
- We realize that the silo must be attached to the short wall of the room



side view



from above

- So we only need to find the **radius** x of the base and the **height** y of the silo (in meters)
- So we have only 2 variables, x and y. They can take real non-negative values

Building the third model 2/4

- Now, we write our objective as **maximize the total volume**, expressed as usual as a function of the variables. The volume is obviously base area times height, and the base is a circle with area πx^2

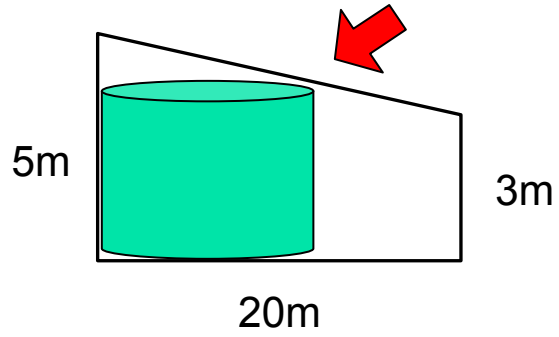
$$\max \pi x^2 y$$

- Do we have **constraints**? Yes, and several types. First, we need to say that the silo must be contained in the room, so the base diameter cannot exceed 10m

$$2x \leq 10$$

Building the third model 3/4

- Then we need to say that the silo must be under the roof. Actually, we just need that the **corner** of the silos **cannot be higher** than the roof over it



- How high is the **roof**? It goes from 5m to 3m in 20m, so it goes down 10cm per each horizontal meter of distance from the left wall. The formula of the height of the roof is $5 - 0.1\ (2\ x)$ so we have

$$y \leq 5 - 0.2\ x$$

Building the third model 4/4

- Finally, the total area cannot be more than 200 mq, because we only have 200 mq of material
- We compute the area as: **base area** x 2 plus **lateral area**
- Base area is πx^2
- Lateral area is the circumference multiplied by the height: $2\pi x y$
- The constraint is

$$2\pi x^2 + 2\pi x y \leq 200$$

Our third model

- By moving variables in the left and doing some simplifications, the complete **third model** is

$$\begin{cases} \max \pi x^2 y \\ x \leq 5 \\ 0.2x + y \leq 5 \\ 2\pi x^2 + 2\pi x y \leq 200 \\ x \geq 0 \quad y \geq 0 \end{cases}$$

- This is a **non-linear programming** model: the expressions of the variables are not linear (we have variables multiplying other variables, variables to the 2nd power, etc.)
- Non-linear models are more computationally **difficult**, they need specific algorithm
- When possible, we always try to use linear models if the effects of nonlinearities are small enough to be **neglected**

Many types of Models

- **Continuous Optimization**: the variable can take real values, sometimes they must be non-negative. Example: kg of goods to be produced, etc.
- **Discrete Optimization**: The variables can take only discrete values. Usually they can be either integer or binary. Example: number of persons, etc.
- **Mixed Optimization**: same variables are continuous and some are discrete

- **Linear** or **Non-linear**: it depends on the expressions of the variables
- **Constrained** or **Unconstrained**: if we have constrains or not
- **Single objective** or **Multi-objective**: one or more objective functions

Linear Programming (LP)

- Objective and constraints **are linear** if they can be put in the following form, where c_i are constants (which can also have complicated expressions but cannot contain variables)

$$c_1 x_1 + \ldots + c_n x_n$$

- In LP there is a **single linear Objective** and a **finite number of linear constraints** in the form of =, ≥, ≤
- The variables are **continuous**, and sometimes non-negative
- Example

$$\begin{cases} \max \ 10 x_1 + 14 x_2 + 18 x_3 \\ 2 x_1 + 1 x_2 + 3 x_3 \le 50 \\ 4 x_1 + 5 x_2 + 4 x_3 \le 80 \\ x_i \ge 0 \quad \text{for} \ \ i = 1, 2, 3 \\ x \in R^3 \end{cases}$$

Compact notation

- We call c the **vector** of coefficients in the objective function, we call A the **constraint matrix** and b the **right-hand side** of the constraints
- The number of **variables** is called n, the number of **constraints** m
- We can write the problem on the left in compact notation or matrix from, obtaining the problem on the right

$$\begin{cases} \max \; 10\,x_1 + 14\,x_2 + 18\,x_3 \\ 2\,x_1 + 1\,x_2 + 3\,x_3 \leq 50 \\ 4\,x_1 + 5\,x_2 + 4\,x_3 \leq 80 \\ x_i \geq 0 \quad \text{for} \quad i = 1, 2, 3 \\ x \in R^3 \end{cases} \qquad \begin{cases} \max \; c'x \\ Ax \leq b \\ x \geq 0^n \\ x \in R^n \end{cases}$$

More advanced example

- A plant produces a liquor using 5 possible ingredients: **Alcohol 90°, Fruit extract, Aroma1, Aroma2, Aroma3**. The prices and the alcoholic content of each ingredient is reported below

	Alcohol	Fruit extract	A1	A2	A3
Alc. degrees	90°	0°	15°	20°	11°
Cost per liter	2	10	190	250	220

- We want to produce 100 Kg of liquor per week. One liter weighs 0.91 kg. The alcoholic degree of the liquor must be between 30° and 33°. To keep the flavor, the fruit extract must be at least 15% of the alcohol, the aromas must be at least 3% of the total, and A1 cannot be more than 1/3 of the sum of A2 and A3. We want to minimize the total cost for the ingredients

Building the model 1/5

- What can we decide? The amount of each ingredient to be used, in liters per week

  x_i = liters of ingredient i used every week,
  with i = Alcohol 90°, Fruit extract, Aroma1, Aroma2, Aroma3

- Objective: minimize **total cost** of the ingredients

$$\min 2\,x_1 + 10\,x_2 + 190\,x_3 + 250\,x_4 + 220\,x_5$$

- Now we analyze every single requirements and write the corresponding constraint(s)

- We want to produce 100 kg per week ≈ 110 liters

$$x_1 + x_2 + x_3 + x_4 + x_5 \geq 110$$

Building the model 2/5

- The **alcoholic degree** of the liquor must be between 30° and 33°. To write this constraint, think what happens if we mix just 1liter of alcohol 90° and 1 liter of water. What is the degree? 45°? How it is obtained? 90+0 / 2. What is 2? Is the total amount of mix, 1+1, NOT the number of ingredients. For example, 1liter of alcohol 90° and 9 liters of water gives 90+0 / 10 = 9°
- So, if we mix our 5 ingredients, the alcoholic degree of the mix will be

$$\frac{90\,x_1 + 0\,x_2 + 15\,x_3 + 20\,x_4 + 11\,x_5}{x_1 + x_2 + x_3 + x_4 + x_5}$$

- This must be between 30 and 33

$$30 \leq \frac{90\,x_1 + 0\,x_2 + 15\,x_3 + 20\,x_4 + 11\,x_5}{x_1 + x_2 + x_3 + x_4 + x_5} \leq 33$$

Building the model 3/5

- Note that if we simply write

$$30 \leq \quad 90\,x_1 + 0\,x_2 + 15\,x_3 + 20\,x_4 + 11\,x_5 \;\leq\; 33$$

- that would be **terribly wrong**!! It would compare the total volume of alcohol to 30 and 33, which are percentages. We must always compare homogeneous quantities
- Another **bad mistake** is to take the table and simply write all its number multiplied by the x. You need to understand what information you are using and what you need to express

Building the model 4/5

- The fruit extract must be at least 15% of the alcohol

$$x_2 \geq 0.15\, x_1$$

- The aromas must be at least 3% of the total

$$x_3 + x_4 + x_5 \geq 0.03\,(x_1 + x_2 + x_3 + x_4 + x_5)$$

- A1 cannot be more than 1/3 of the sum of A2 and A3

$$x_3 \leq 0.33\,(x_4 + x_5)$$

- Finally, all variables here are **real** and **non-negative**. When this is required, we cannot simply omit it, otherwise we may obtain a solution with negative values which would be useless in practice

$$x_i \geq 0 \qquad\qquad x_i \in \mathrm{R}$$

Complete model

$$
\begin{cases}
\min\ 2x_1 + 10x_2 + 190x_3 + 250x_4 + 220x_5 \\
x_1 + x_2 + x_3 + x_4 + x_5 \geq 110 \\
90x_1 + 0x_2 + 15x_3 + 20x_4 + 11x_5 \leq 33(x_1 + x_2 + x_3 + x_4 + x_5) \\
90x_1 + 0x_2 + 15x_3 + 20x_4 + 11x_5 \geq 30(x_1 + x_2 + x_3 + x_4 + x_5) \\
x_2 \geq 0.15x_1 \\
x_3 + x_4 + x_5 \geq 0.03(x_1 + x_2 + x_3 + x_4 + x_5) \\
x_3 \leq 0.33(x_4 + x_5) \\
x_i \geq 0 \\
x_i \in \mathrm{R}
\end{cases}
$$

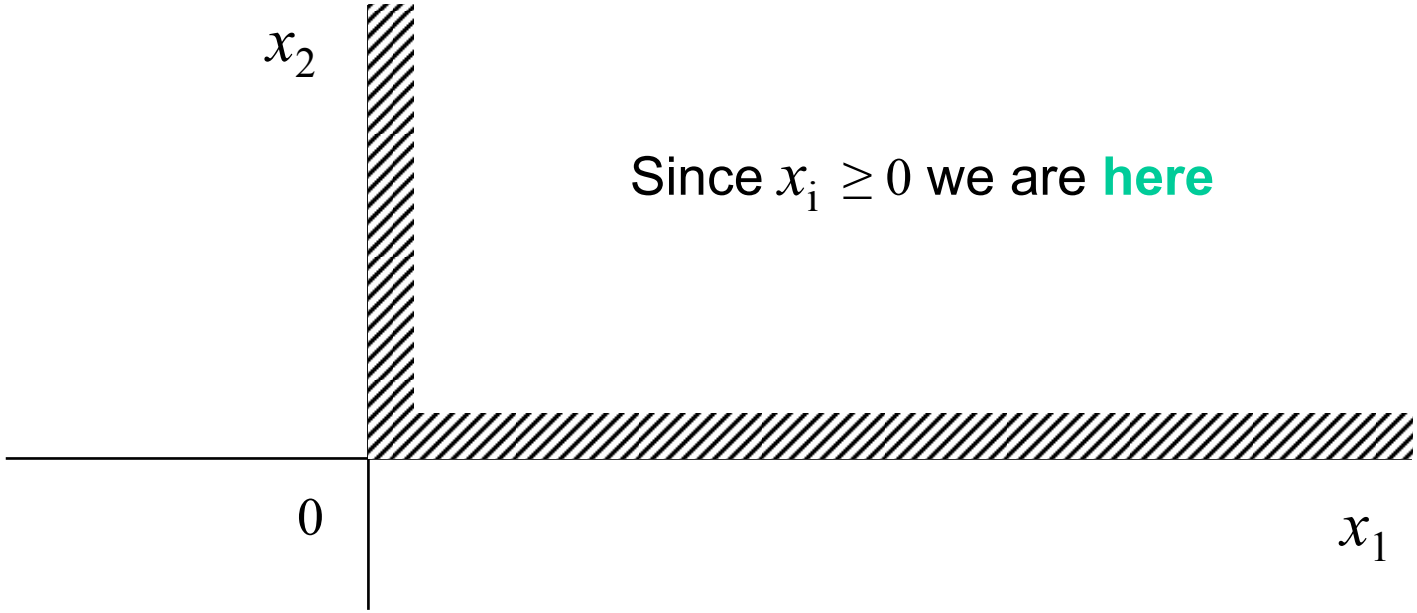
- It is getting complex.. In real world problems we easily have hundreds of variables and constraints. We need **efficient** solution algorithms

Geometry of linear programming

$$
\begin{cases}
\max\ 5\,x_1 + 10\,x_2 \\
10\,x_1 + 5\,x_2 \le 25 \\
4\,x_1 + 10\,x_2 \le 20 \\
x_1 + 1.5\,x_2 \le 4.5 \\
x_i \ge 0 \\
x_i \in \mathrm{R}
\end{cases}
$$

- Let's see what this means form a **geometrical** point of view

When dealing with LP the real domain is often implicit



Since $x_i \ge 0$ we are **here**

Geometry of linear programming

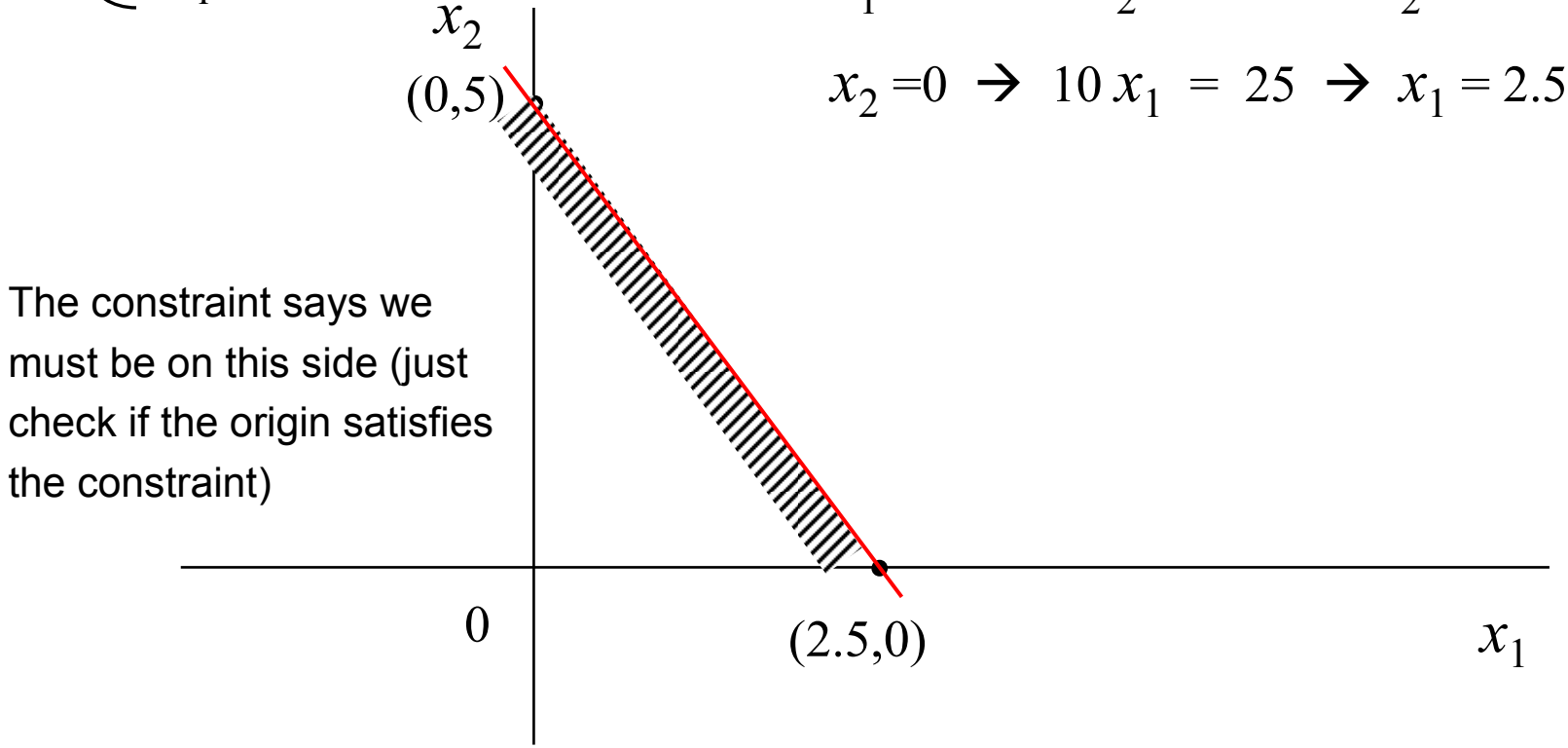
$$\begin{cases} \max\ 5x_1 + 10x_2 \\ 10x_1 + 5x_2 \le 25 \\ 4x_1 + 10x_2 \le 20 \\ x_1 + 1.5x_2 \le 4.5 \\ x_i \ge 0 \end{cases}$$

- First constraint: we examine the associated straight line

$$10x_1 + 5x_2 = 25$$

$$x_1 = 0 \rightarrow 5x_2 = 25 \rightarrow x_2 = 5$$

$$x_2 = 0 \rightarrow 10x_1 = 25 \rightarrow x_1 = 2.5$$



The constraint says we must be on this side (just check if the origin satisfies the constraint)

Geometry of linear programming

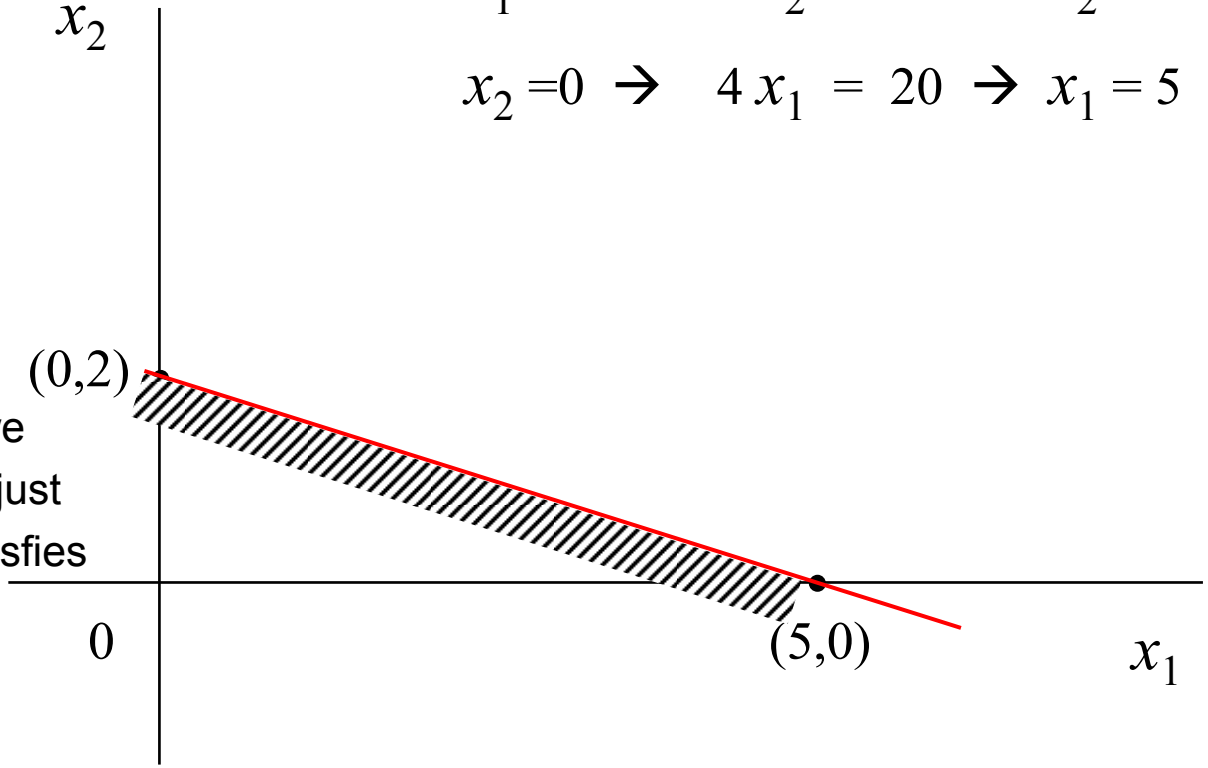
$$\begin{cases} \max\ 5x_1 + 10x_2 \\ 10x_1 + 5x_2 \le 25 \\ 4x_1 + 10x_2 \le 20 \\ x_1 + 1.5x_2 \le 4.5 \\ x_i \ge 0 \end{cases}$$

- Second constraint: we examine the associated straight line

$$4x_1 + 10x_2 = 20$$

$$x_1 = 0 \rightarrow 10x_2 = 20 \rightarrow x_2 = 2$$

$$x_2 = 0 \rightarrow 4x_1 = 20 \rightarrow x_1 = 5$$



The constraint says we must be on this side (just check if the origin satisfies the constraint)

Geometry of linear programming

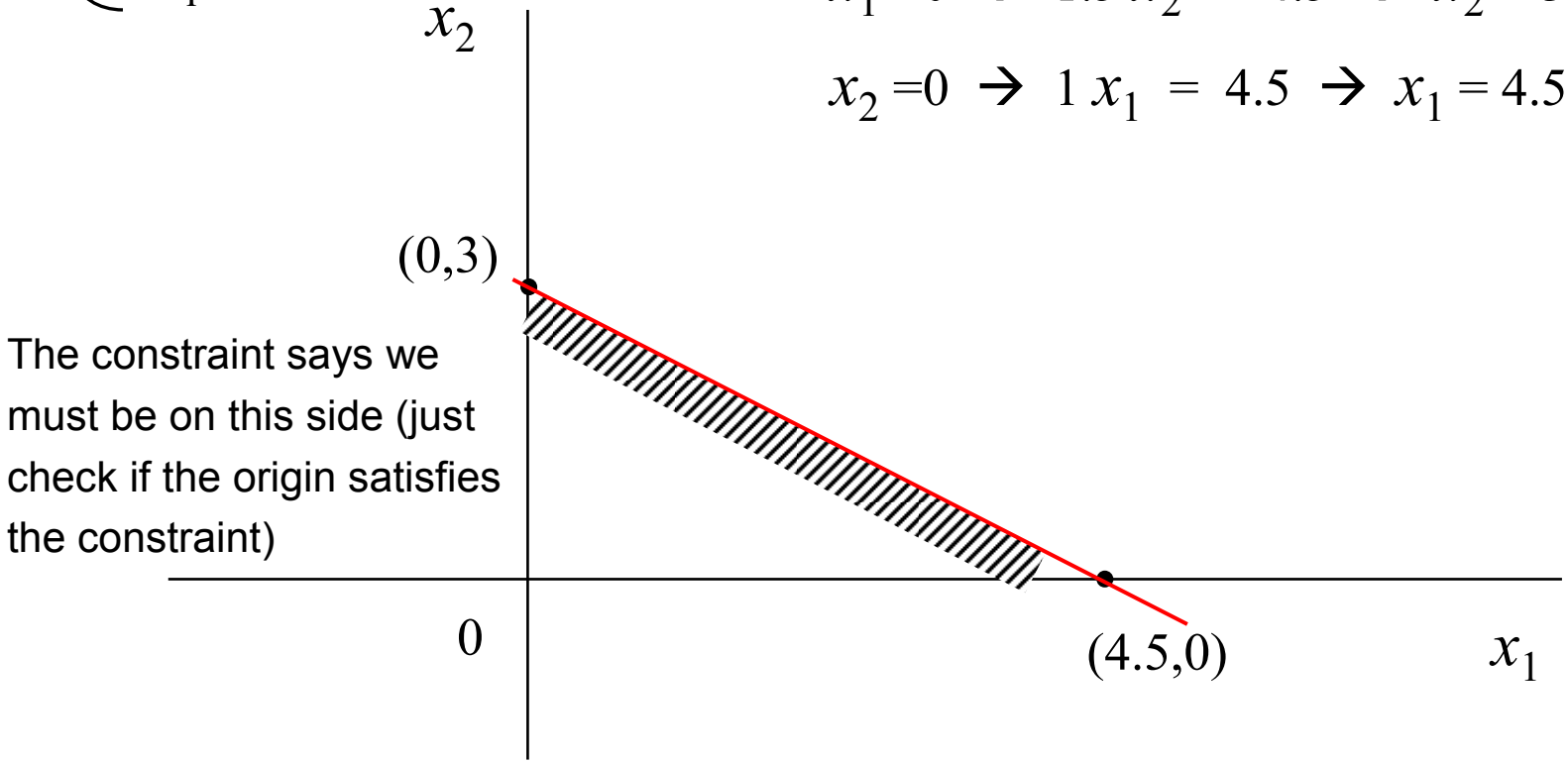
$$\begin{cases} \max\ 5\,x_1 + 10\,x_2 \\ 10\,x_1 + 5\,x_2 \le 25 \\ 4\,x_1 + 10\,x_2 \le 20 \\ x_1 + 1.5\,x_2 \le 4.5 \\ x_i \ge 0 \end{cases}$$

- Third constraint: we examine the associated straight line

$$1\,x_1 + 1.5\,x_2 = 4.5$$

$$x_1 = 0 \rightarrow 1.5\,x_2 = 4.5 \rightarrow x_2 = 3$$

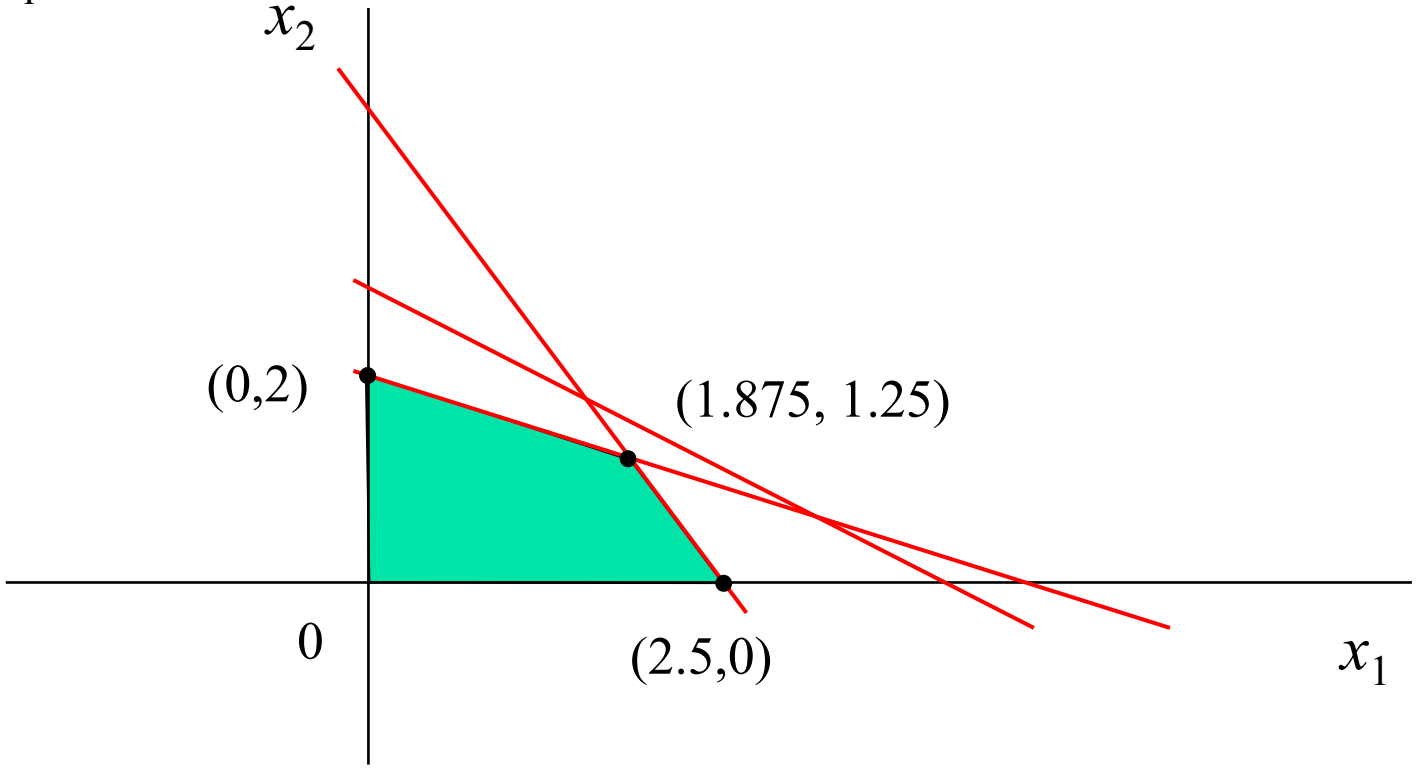
$$x_2 = 0 \rightarrow 1\,x_1 = 4.5 \rightarrow x_1 = 4.5$$



Geometry of linear programming

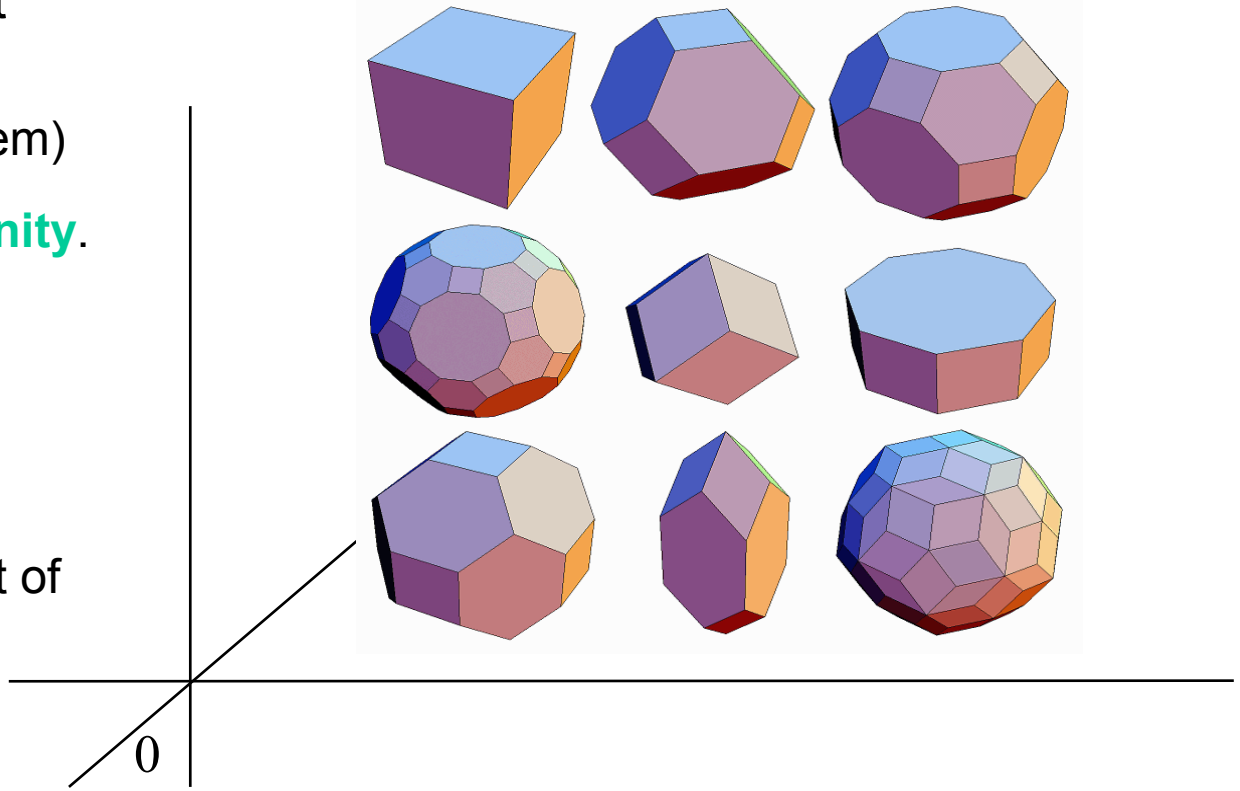
$$\begin{cases} \max\ 5\,x_1 + 10\,x_2 \\ 10\,x_1 + 5\,x_2 \le 25 \\ 4\,x_1 + 10\,x_2 \le 20 \\ x_1 + 1.5\,x_2 \le 4.5 \\ x_i \ge 0 \end{cases}$$

- All constraints must be respected at the same time
- The **feasible set** is:



Geometry of linear programming

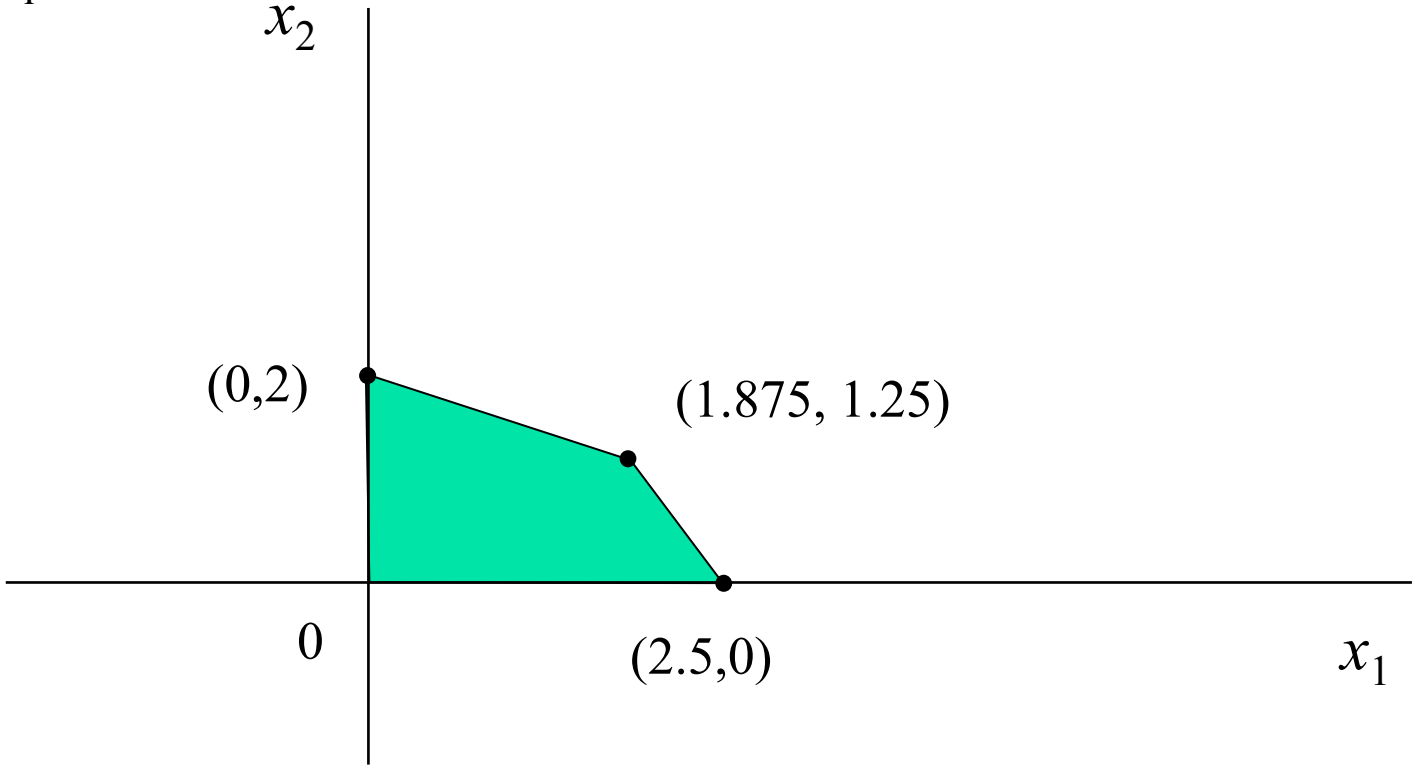
- In general, the feasible set is a **polyhedron**: the **intersection** of a **finite** number of **hyperplanes** (given by the equality constrains) and **closed half-spaces** (given by inequality constraints)
- It is a **convex set**: the linear combination of any two points of the set is still in the set
- It may be **empty** (infeasible problem)
- It may **go to infinity**. In this case, sometimes the problem is unbounded
- An exterme point of a polyherdon is called **vertex**



Geometry of linear programming

$$\begin{cases} \max\ 5x_1 + 10x_2 \\ 10x_1 + 5x_2 \le 25 \\ 4x_1 + 10x_2 \le 20 \\ x_1 + 1.5x_2 \le 4.5 \\ x_i \ge 0 \end{cases}$$

- Within this set we want to find the point(s) giving **maximum** value to the objective function
- Let's study how this function is



Geometry of linear programming

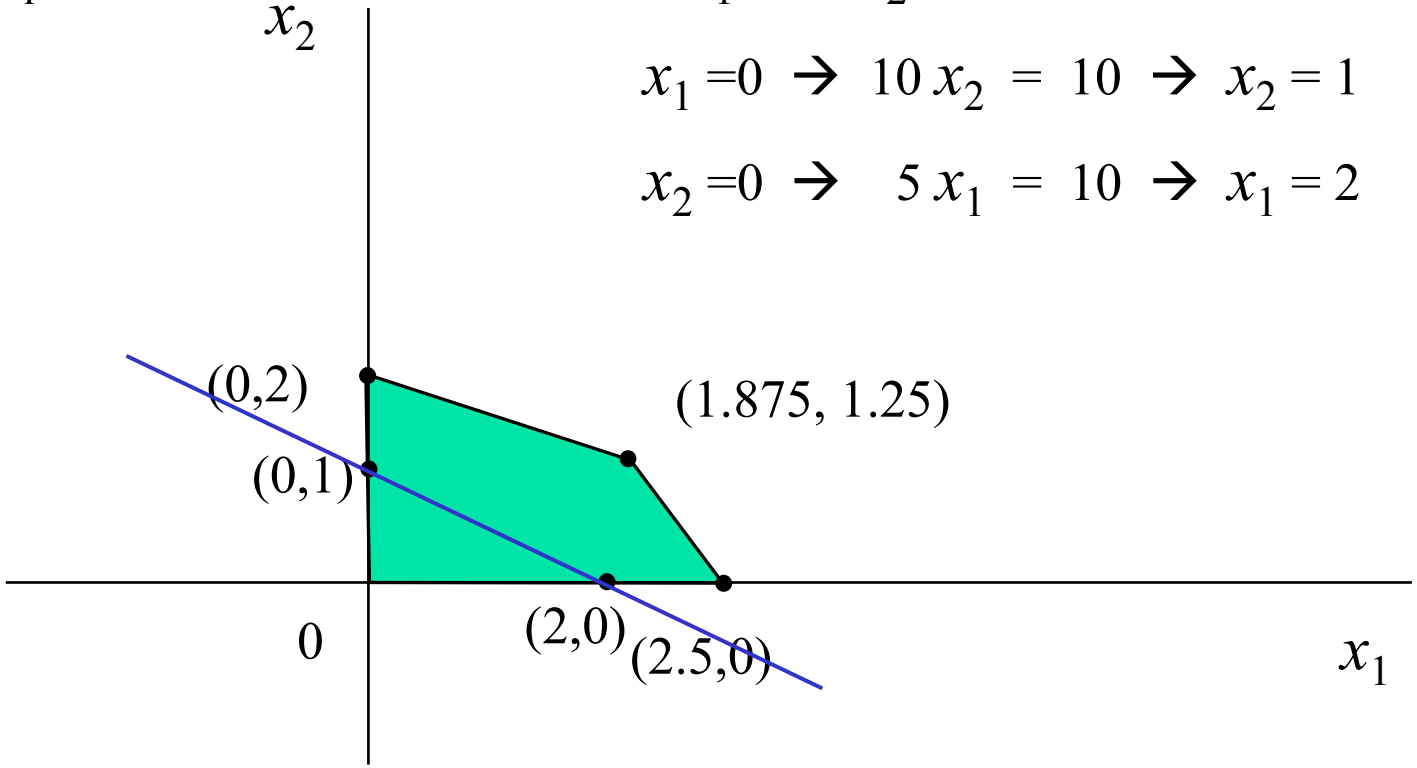
$$\begin{cases} \max\ 5x_1 + 10x_2 \\ 10x_1 + 5x_2 \le 25 \\ 4x_1 + 10x_2 \le 20 \\ x_1 + 1.5x_2 \le 4.5 \\ x_i \ge 0 \end{cases}$$

- A linear objective is **constant** over straight lines
- For instance let's find the line for

$$5x_1 + 10x_2 = 10$$

$$x_1 = 0 \rightarrow 10x_2 = 10 \rightarrow x_2 = 1$$

$$x_2 = 0 \rightarrow 5x_1 = 10 \rightarrow x_1 = 2$$



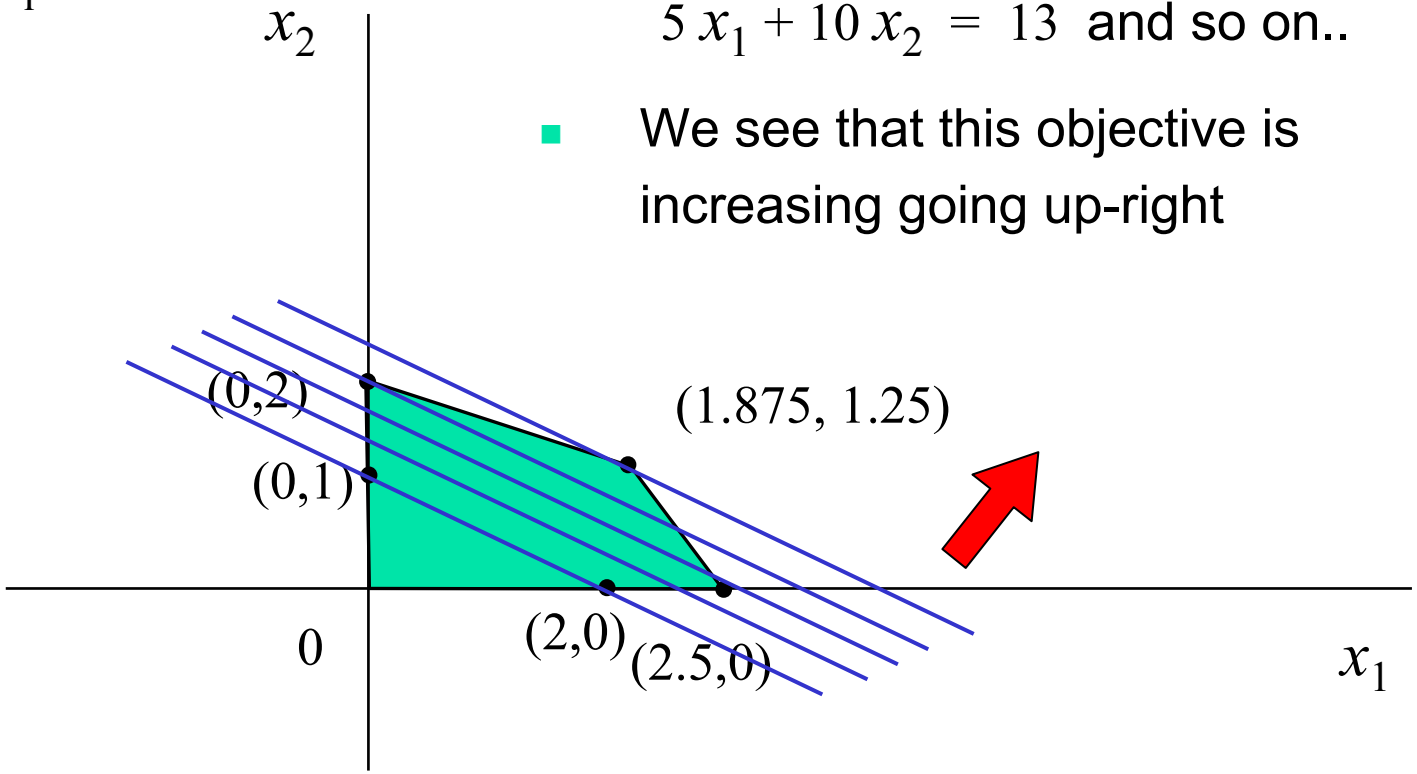
Geometry of linear programming

$$\begin{cases} \max\ 5x_1 + 10x_2 \\ 10x_1 + 5x_2 \le 25 \\ 4x_1 + 10x_2 \le 20 \\ x_1 + 1.5x_2 \le 4.5 \\ x_i \ge 0 \end{cases}$$

- In the same way we could see which is the line for

$$5x_1 + 10x_2 = 11$$
$$5x_1 + 10x_2 = 12$$
$$5x_1 + 10x_2 = 13$$ and so on..

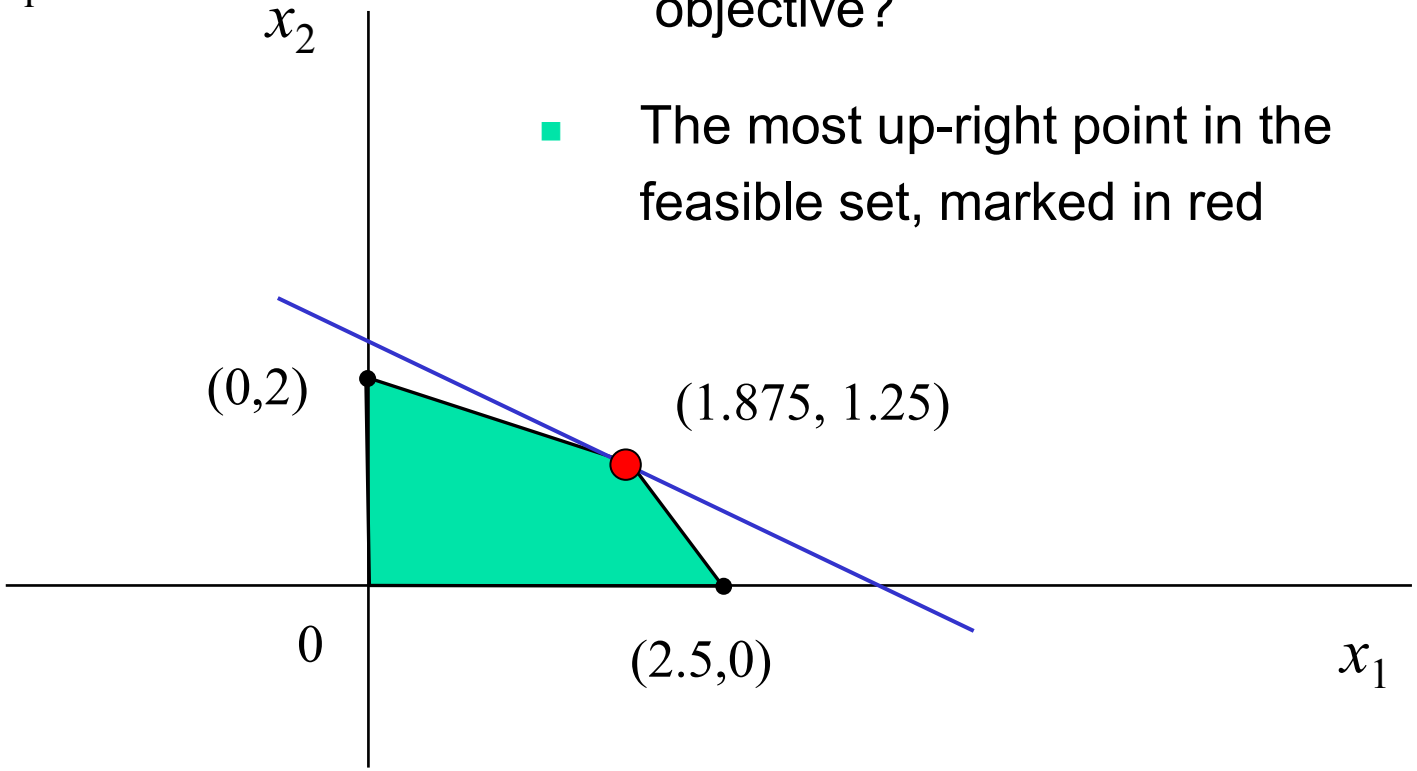
- We see that this objective is increasing going up-right



Geometry of linear programming

$$\begin{cases} \max\ 5x_1 + 10x_2 \\ 10x_1 + 5x_2 \le 25 \\ 4x_1 + 10x_2 \le 20 \\ x_1 + 1.5x_2 \le 4.5 \\ x_i \ge 0 \end{cases}$$

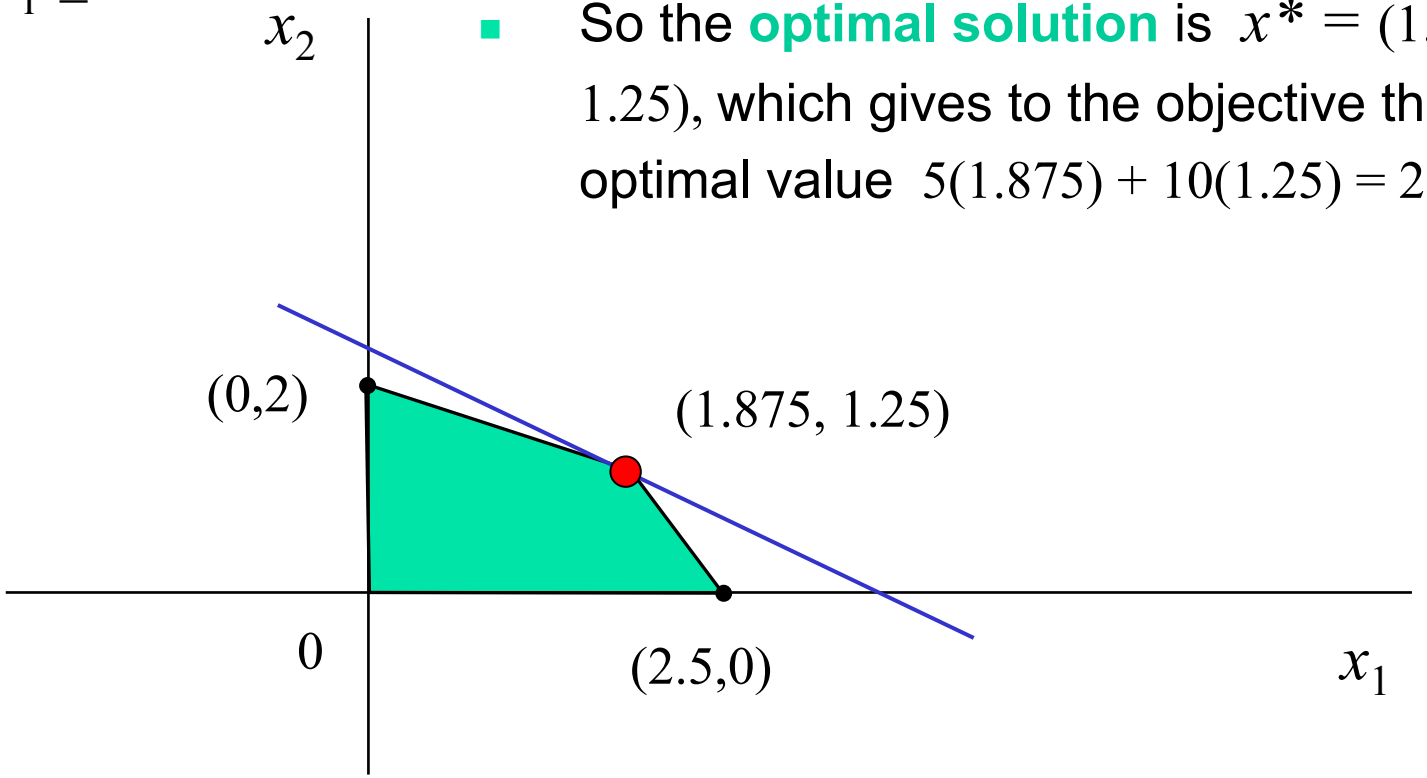
- So which is the point intersecting the line with the **highest value**? In other words, which is the point giving maximum value to the objective?
- The most up-right point in the feasible set, marked in red



Geometry of linear programming

$$\begin{cases} \max\ 5x_1 + 10x_2 \\ 10x_1 + 5x_2 \le 25 \\ 4x_1 + 10x_2 \le 20 \\ x_1 + 1.5x_2 \le 4.5 \\ x_i \ge 0 \end{cases}$$

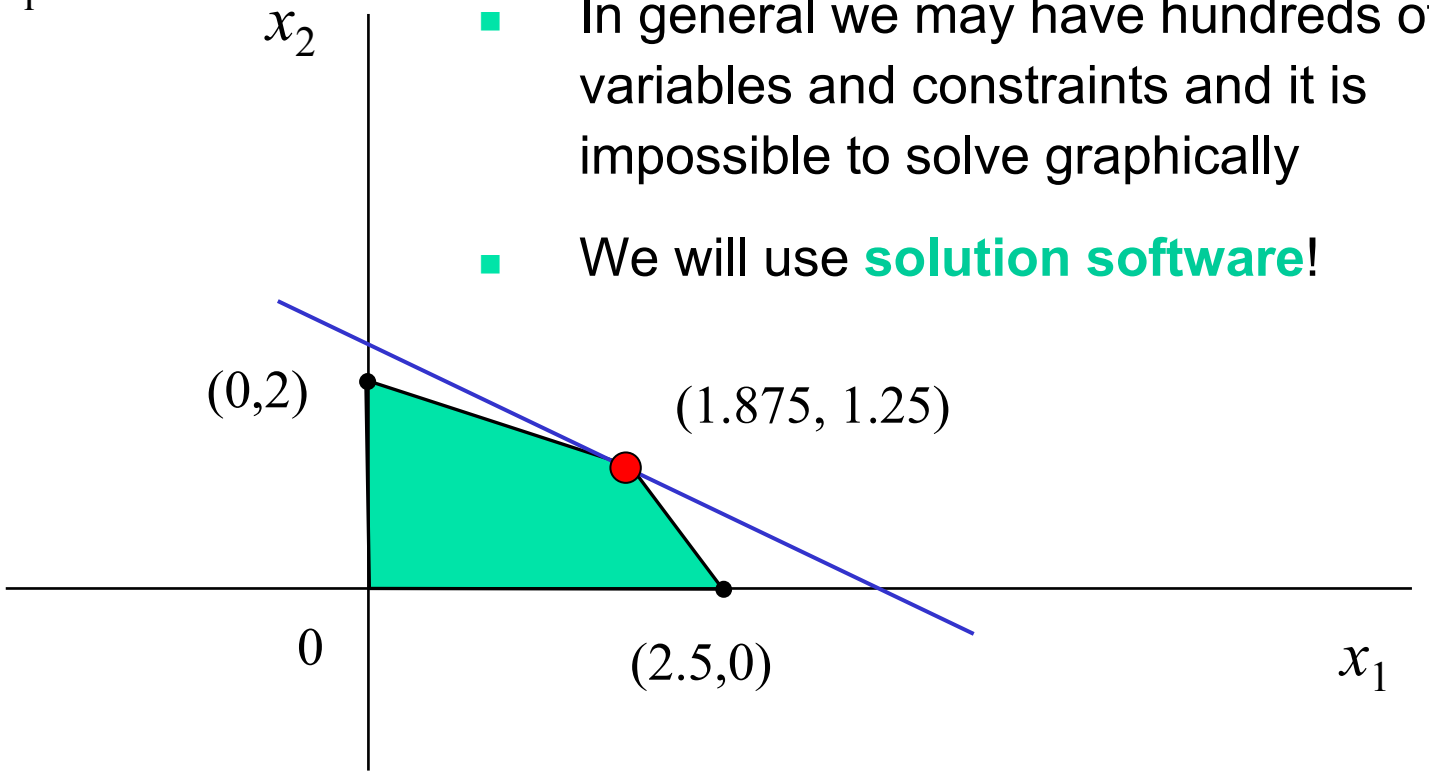
- The coordinates of the red point are obtained as the **intersection** between two lines, if we compute it we obtain $x_1 = 1.875, \quad x_2 = 1.25$
- So the **optimal solution** is $x^* = (1.875, 1.25)$, which gives to the objective the optimal value $5(1.875) + 10(1.25) = 23.75$



Geometry of linear programming

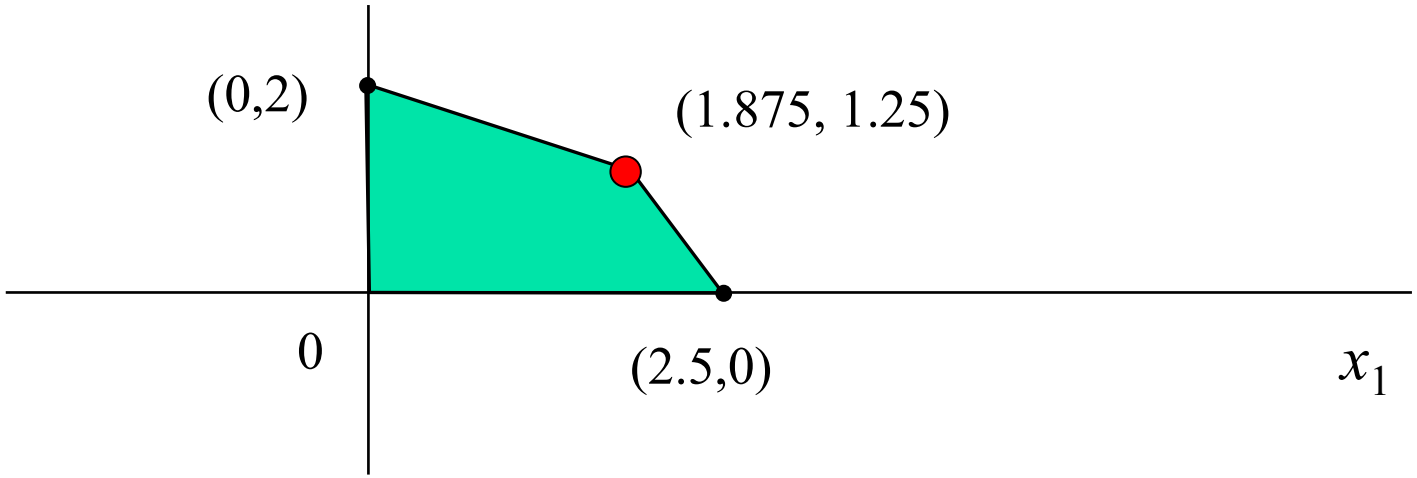
$$\begin{cases} \max 5x_1 + 10x_2 \\ 10x_1 + 5x_2 \le 25 \\ 4x_1 + 10x_2 \le 20 \\ x_1 + 1.5x_2 \le 4.5 \\ x_i \ge 0 \end{cases}$$

- This problem has been solved **graphically**
- This was possible because there are only 2 variables and it is very simple
- In general we may have hundreds of variables and constraints and it is impossible to solve graphically
- We will use **solution software**!



Geometry of linear programming

- When the problem is not infeasible nor unbounded, we have **one** (or more than one) **optimal solutions**
- Due to the linearity of the objective, optimal solutions must stay on the boundary of the polyhedron (or frontier), not in the inside
- If the feasible set contains **vertices** (the vast majority of the cases) and the problem has optimal solution, we have at least one **optimal vertex**
- Therefore, we generally search for **just one** optimal vertex (other optimal solutions, even if existing, cannot be better by definition)



Duality of linear programming

- Given any LP, we can build another LP called its **dual problem**, and the first one will be the **primal problem**
- The dual is very **useful** to obtain several information about the primal
- Let's introduce its importance with a very simple **example**: A company produces 2 products, called "standard" and "de luxe". They are obtained using 2 ingredients I1 and I2 and a machine. Each Kg of product needs the following amount of resources. We want to plan our production in order to maximize the income obtained from selling

	De luxe	Standard	Weekly availability
Kg of I1	3	2	1200
Kg of I2	4	1	1000
Machine hours	2	1	700
Selling price	24	14	

The model of our example

	De luxe	Standard	Weekly availability
Kg of I1	3	2	1200
Kg of I2	4	1	1000
Machine hours	2	1	700
Selling price	24	14	

- We choose as variables the **amount** in Kg of each product to be produced weekly. We have 2 variables: x_1 and x_2
- We know how to build the **model** and we obtain

$$\begin{cases} \max\ 24\,x_1 + 14\,x_2 \\ 3\,x_1 + 2\,x_2 \le 1200 \\ 4\,x_1 + 1\,x_2 \le 1000 \\ 2\,x_1 + 1\,x_2 \le 700 \\ x_1 \ge 0 \quad x_2 \ge 0 \end{cases}$$

Solving the model

$$\begin{cases} \max\ 24\,x_1 + 14\,x_2 \\ 3\,x_1 + 2\,x_2 \le 1200 \\ 4\,x_1 + 1\,x_2 \le 1000 \\ 2\,x_1 + 1\,x_2 \le 700 \\ x_1 \ge 0 \quad x_2 \ge 0 \end{cases}$$

- We take this model and we **solve** with some solution software
- We find the **optimal solution**: $x_1^* = 160,\ x_2^* = 360$ with value 8880
- This means that we maximize income when we produce 160 Kg per week of De Luxe and 360 Kg per week of Standard, and the weekly income is 8880 Euro
- We come back to the **owner of the company** and explain him this "best solution"
- Now, imagine that he says "**No, 8880 is not enough, I want more!**"
- What can we do??

More than the best?

- So the owner says "**No, 8880 is not enough, I want more!**"
- We explain that this is the **best solution, no other feasible solution can have a higher value** but the owner remains on his position
- He understands that, **given the limitations in the resources** (I1, I2, machine time), no more products can be produced weekly
- But says: "I can take **more ingredients** or **another machine**, I can do anything, but I want to earn more!"
- What do we have to do? Take **more I1**? Take **more I2**? Take **another machine**?
- And since we need to **pay** for these resources, how much can we pay? How much would we **earn** if we buy more of these resources?
- All these questions are answered by the **dual**

What is the dual problem?

- Given a **generic LP** in compact form, with $c \in \mathrm{R}^n$, $x \in \mathrm{R}^n$, $b \in \mathrm{R}^m$, $A \in \mathrm{R}^{m \times n}$

$$\begin{cases} \min c^{\mathrm{T}} x \\ A x \geq b \end{cases}$$

- Consider an optimal solution x^*. For all $u \in \mathrm{R}^m$, $u \geq 0$, we have

$$c^{\mathrm{T}} x^* \geq c^{\mathrm{T}} x^* + \underbrace{u^{\mathrm{T}} (b - A x^*)}_{\substack{\underbrace{\geq 0 \quad \leq 0}_{\leq 0}}} = c^{\mathrm{T}} x^* + b^{\mathrm{T}} u - u^{\mathrm{T}} A x^* = b^{\mathrm{T}} u + \underbrace{(c^{\mathrm{T}} - u^{\mathrm{T}} A)}_{\text{}} x^*$$

- If we use u such that $A^{\mathrm{T}} u = c$, this disappears and we have $c^{\mathrm{T}} x^* \geq b^{\mathrm{T}} u$
- In other words $b^{\mathrm{T}} u$ is a **lower bound** for the optimal value $c^{\mathrm{T}} x^*$. If we want to take the closest lover bound, we solve this:

$$\begin{cases} \max b^{\mathrm{T}} u \\ A^{\mathrm{T}} u = c \\ u \geq 0 \end{cases}$$

- This is the dual: an LP representing the search for the **best lower bound** of the primal

How to build the dual

- The dual problem can be obtained for **every** LP using the table below
- If the **primal** is min with constraints = or ≥, then the **dual** is max and is the best lower bound
- If the **primal** is max with constraints = or ≤, then the **dual** is min and is the best upper bound
- If the primal is not in one of two "natural" forms, we put it in that form

primal [dual]		dual [primal]	
objective	$\min c^{\mathrm{T}} x$	$\max b^{\mathrm{T}} u$	objective
constraints	$= b_i \quad i \in I$ $\geq b_i \quad i \in J$	u_i free $\quad i \in I$ $u_i \geq 0 \quad i \in J$	variables
variables	$x_j \geq 0 \quad j \in M$ x_j free $\quad i \in N$	$\leq c_j \quad j \in M$ $= c_i \quad i \in N$	constraints

Let's apply the table…

- …to the model seen **before**

$$\begin{cases} \max\ 24\,x_1 + 14\,x_2 \\ 3\,x_1 + 2\,x_2 \le 1200 \\ 4\,x_1 + 1\,x_2 \le 1000 \\ 2\,x_1 + 1\,x_2 \le 700 \\ x_1 \ge 0 \quad x_2 \ge 0 \end{cases}$$

objective	$\min c^T x$	$\max b^T u$	objective
constraints	$= b_i \quad i \in I$ $\ge b_i \quad i \in J$	u_i free $\quad i \in I$ $u_i \ge 0 \quad i \in J$	variables
variables	$x_j \ge 0 \quad j \in M$ x_j free $\quad i \in N$	$\le c_j \quad j \in M$ $= c_i \quad i \in N$	constraints

- We put the model in one of the 2 **forms in the table**. In this case, it is max with $\le$ constraints, so in the form on the right. Remember that the dual of the dual is the primal
- This primal has 2 variables ≥ 0 and 3 constraints $\le$, its dual will have 2 constraints $\ge$ and 3 variables ≥ 0
- We only have to fill the dots

$$\begin{cases} \min\ ..\,u_1 + ..\,u_2 + ..\,u_3 \\ ..\,u_1 + ..\,u_2 + ..\,u_3 \ge \dots \\ ..\,u_1 + ..\,u_2 + ..\,u_3 \ge \dots \\ u_1 \ge 0 \quad u_2 \ge 0 \quad u_3 \ge 0 \end{cases}$$

And complete this dual

- The primal always has **3 elements**: $\begin{cases} \text{obj } c^{\mathrm{T}} x \\ A x \geq b \end{cases}$
- constraint matrix, usually called A
- right-hand side of the constraints, usually called b
- objective coefficients, usually called c

$$A = \begin{pmatrix} 3 & 2 \\ 4 & 1 \\ 2 & 1 \end{pmatrix}$$

$$b = (\ 1200\ \ 1000\ \ 700\ )^{\mathrm{T}}$$

$$c = (\ 24\ \ \ 14\ )^{\mathrm{T}}$$

- We use:
- A^{T} for the constraint matrix of the dual
- c for the right-hand side of the dual constraints
- b for the dual objective coefficients

- So we obtain the dual

$$\begin{cases} \min 1200\, u_1 + 1000\, u_2 + 700\, u_3 \\ 3\, u_1 + 4\, u_2 + 2\, u_3 \geq 24 \\ 2\, u_1 + 1\, u_2 + 1\, u_3 \geq 14 \\ u_1 \geq 0 \quad u_2 \geq 0 \quad u_3 \geq 0 \end{cases}$$

Duality implications

- Given any **primal-dual** couple, for example the following

$$\begin{cases} \min c^{\mathrm{T}} x \\ Ax \geq c \\ x \geq 0 \end{cases} \qquad \begin{cases} \max b^{\mathrm{T}} u \\ A^{\mathrm{T}} u \leq c \\ u \geq 0 \end{cases}$$

- For construction, we have the **Weak Duality theorem**: for every feasible solution of primal and dual we have $c^{\mathrm{T}} x \geq b^{\mathrm{T}} u$

  (since the dual max was built to be the best lower bound to the primal)

- And the **Strong Duality theorem**: for the two optimal solutions of primal and dual we have $c^{\mathrm{T}} x^* = b^{\mathrm{T}} u^*$

- Therefore, to find the optimal value of the primal we can search for the optimal value of the dual

We solve the dual

$$\begin{cases} \min\ 1200\,u_1 + 1000\,u_2 + 700\,u_3 \\ 3\,u_1 + 4\,u_2 + 2\,u_3 \geq 24 \\ 2\,u_1 + 1\,u_2 + 1\,u_3 \geq 14 \\ u_1 \geq 0 \quad u_2 \geq 0 \quad u_3 \geq 0 \end{cases}$$

and we obtain

$u^* = (\ 6.4,\ 1.2,\ 0\ )^{\mathrm{T}}$ with value 8880

- Now, coming back to the **owner of the company**, what happens when we buy more **resources** in the real world?
- We modify the **right-hand side** values of the primal $b = (\ 1200\ 1000\ 700\ )^{\mathrm{T}}$
- This means we modify the **objective coefficients** in the dual
- For variations small enough such that the optimal point u^* remains the same, the **dual objective** changes of u_j^* for each unit of resource
- Hence, the **optimal solution of the primal** changes of u_j^*
- Hence, one **additional unit** of resource j increases the income by u_j^*
- So, that is the **maximum price** we can pay for it (aka marginal value)

Which resource is useful?

- So, if we add 1 unit of **I1**, passing from 1200 to 1201, the objective becomes 8880 + 6.4 Euro
- Hence, we could buy other I1 until the price is **less than 6.4 per Kg**
- If we add 1 unit of **I2**, passing from 1000 to 1001, the objective becomes 8880 + 1.2 Euro
- Hence, we could buy other I2 until the price is **less than 1.2 per Kg**
- Finally, if we add 1 hour of **machine time**, passing from 700 to 701, the objective becomes 8880 + 0 Euro (remains the same)
- Hence, we there is **no advantage** in buying another machine! Actually, we are not even using all machine hours. The optimal solution x^* uses 160x2 + 360 = 680 hours
- So, we also discover which are the **limiting resources**, and which are those that we are not using to the **limit**!

Which resource is limiting us?

- If we **progressively increment** the limiting resources, at a certain point they will not be the limiting ones anymore
- Example: if we buy other 1000 kg of I1, that resource would cease to be our limit
- We will be producing more, and the limitation will maybe become the hours of machine time
- This means that the **variations were not small**, and the **dual solution** has **changed**!
- So, by studying the dual solution, we can also discover to which extent of variation a resource ceases to be the limit and another resource becomes the new limiting one

Modeling Techniques 1

- In many cases, we may have a situation called **mutual exclusivity**
- For example, imagine a situation where we can either produce P1 or P2, **but not both**! Mutual exclusivity is a common requirement for the variables in some practical problems: only one of them may be > 0
- How do we model this with linear constraints?
- If we have **binary variables** x_1 and x_2 , it is **easy**: $x_1 + x_2 \leq 1$
- But if the variables are **real** (or integer non-binary) it is more difficult
- We need **one additional binary variable** y for each couple of mutually exclusive variables x_1 and x_2
- And we add these constraints to the rest of the model, with M a large number, greater than any possible value of x :

$x_1 \leq \mathrm{M}\, y$
$x_2 \leq \mathrm{M}\,(1\text{-}y)$

- If x_1 is > 0, then y is 1 and x_2 is forced to 0
- If x_2 is > 0, then y is 0 and x_1 is forced to 0

Modeling Techniques 2

- In many cases, we may have another situation called **fixed cost**
- For example, imagine a situation where we produce something and the cost is a fixed setup cost f, paid only if we produce, plus a cost c for each unit of product:

$$\begin{cases} 0 & \text{if } x_i = 0 \\ c\,x_i + f & \text{if } x_i > 0 \end{cases}$$

- How do we model this with linear constraints?
- For each variable x_i having a fixed cost, we need **one additional binary variable** y
  - Then, we write the **total cost** as $c\,x_i + f\,y$
  - and we add this **constraint** to the rest of the model, with M a large number, greater than any possible value of x :

$$x_i \leq M\,y$$

  - If x_i is > 0, then y is 1 and we **pay** f in the cost
  - If x_i is $= 0$, then y can be 0 and we **don't have** to pay f in the cost, and we **will not** pay it, if the objective implies cost minimization

Example

- A farmer may raise 4 types animals: horses, cows, pigs, poultry. Each type of animal requires different structures and materials.
- Set-up costs are 200,000 E for horses, 160,000 E for cows, 120,000 E for pigs, 80,000 E for poultry.
- A horse is raised at a cost of 200 E (in average), and can be sold at 300 E (in average). For cows this becomes 200 and 250; for pigs 50 and 150, for poultry 2 and 3.
- The maximum population that can live in the farm is 500 in case of horses. Each cow counts as 0.8 horses, each pig as 0.4 horses, each unit of poultry as 0.01 horses. The land is not enough to build facilities for both horses and cows.
- We want to determine a “target composition” in order to maximize earnings, so that the farmer can plan the raising.

Example

x_i = number of heads in the “target composition”,
with i = horses, cows, pigs, poultry

- These variables are conceptually integer, but since we are pointing to a target composition and the numbers are not small, we may also use real non-negative variables and the model would be easier to solve

y_i = presence of animal i in the “target composition”,
with i = horses, cows, pigs, poultry

- These variables are binary, they make the model more difficult to solve, but we need them to express fixed costs and mutual exclusivity

Example

- Objective

$$\max (300\text{-}200)\, x_1 + (250\text{-}200)\, x_2 + (150\text{-}50)\, x_3 + (3\text{-}2)\, x_4 - 200{,}000\, y_1 - 160{,}000\, y_2 - 120{,}000\, y_3 - 80{,}000\, y_4$$

- Maximum population

$$x_1 + 0.8\, x_2 + 0.4\, x_3 + 0.01\, x_4 \leq 500$$

- Link between x and y (otherwise we simply have all y at 0 even if the corresponding x may be positive)

$$x_1 \leq 10000\, y_1 \qquad x_2 \leq 10000\, y_2$$

$$x_3 \leq 10000\, y_3 \qquad x_4 \leq 100000\, y_4$$

Example

- Mutual exclusivity: in general, we would need to add another binary variable z and 2 constraints of this type

$$x_1 \leq M z \qquad x_2 \leq M (1\text{-}z)$$

- However, here we can use the binary variables y and we obtain a simpler model

$$y_1 + y_2 \leq 1$$

Solution Algorithms

- For Linear programming we have: **simplex** algorithm (most used), barrier algorithm, interior point algorithm, …

- For integer (and binary) linear programming we have: **branch-and-bound** and its variants (branch-and-cut, …), cutting planes, …

The Simplex Algorithm

- We know that for LP the feasible set is a **polyhedron**, and that, if there are **vertices** and the problem has **optimal solution**, then there is an **optimal vertex**
- Therefore, we can **limit** our search to the vertices of the polyhedron
- We initially **find a vertex**, and we test whether it is optimal or not
- If that is not the optimal vertex, we do some operations to reach an **adjacent vertex** where the objective is better
- We continue until we find the optimal vertex, where we **stop**, or we prove the problem **unbounded**
- Since the number of vertices is finite, the algorithm stops
- This algorithm is in practice very efficient, and can solve problems with thousands of variables in seconds

Integer (or Binary) Lin Prog

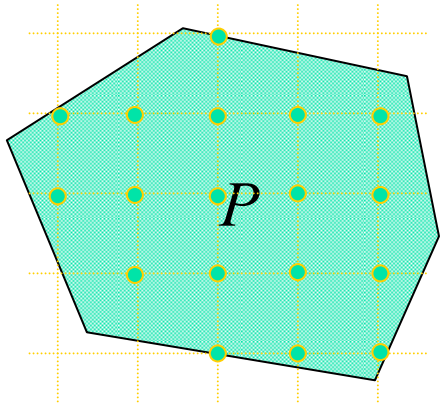
Given an **Integer** (or **Binary**) linear programming problem,

we want to find in the feasible set S, the **optimal point** x^*

$$\begin{cases} \min \ c^T x \\ x \in S \end{cases}$$

S is often expressed as the set of integer points (or binary points) within a polyhedron P, called formulation ($S = P \cap Z^n$)

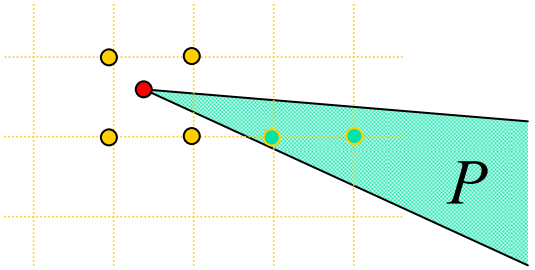
$$\begin{cases} \min \ c^T x \\ x \in P \quad (\text{e.g. } Ax \geq b) \\ x \in Z^n \quad (\text{or } x \in \{0,1\}^n) \end{cases}$$



How to solve?

- Since the number of feasible points is finite, one may think of some **enumeration**
- However, for real-world problems the number of feasible points is generally **huge**: complete enumeration cannot be used
- We will use a kind of implicit enumeration…

- Notice: **rounding** at the nearest integer the solution of the linear problem does not guarantee optimality and not even feasibility (especially for binary problems)



Basic ideas for Branch & Bound

Basic ideas:

- **Branching:** partition the feasible set P in smaller subproblems P_i . They are easier to solve!

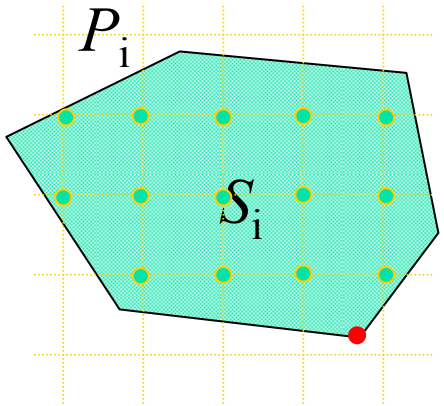
$$P_i \cap P_j = \varnothing \quad i \neq j$$

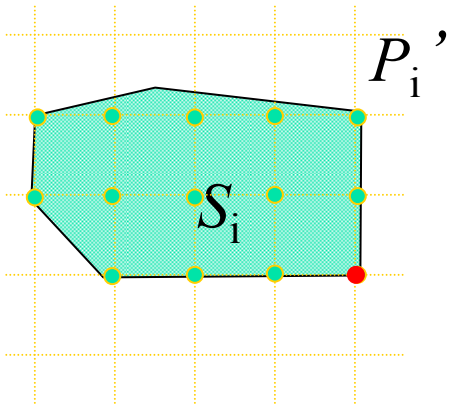
$$\bigcup_i P_i = P$$

- **Bounding:** find a feasible solution (**current solution**, for example solving one subproblem or by using approximation) and use it to **close** the subproblems for which the **best obtainable** solution is **not better** than the current solution that we already have. This saves time!

Bounding

- **Linear relaxation** (most used): remove the intgrality constraints from the generic subproblem P_i obtaining linear programming, easily solvable
- The minimum of $c^T x$ over all points (integers or not) **is** $\leq$ of the minimum of the same function choosing only over integer points
- Moreover, in some lucky cases, that minimum may already be integer
- There exist other bounding techniques, but are less used and we do not see them





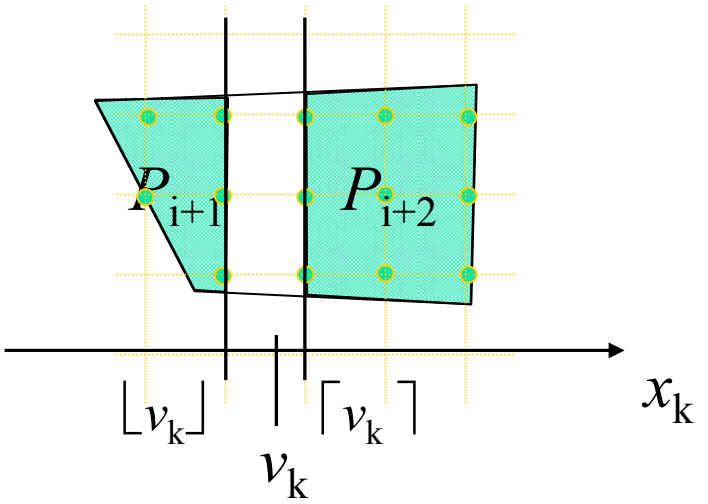
Branching

- **Binary** to the closest integers (most used): when we solve the linear relaxation and we find a solution x with some non-integer components (= **fractionary**), for example x_k with value v_k

P_i was a subset not easy enough → we **partition** it in P_{i+1} and P_{i+2}

$$P_{i+1} = P_i \cap \{x:\ x_k \leq \lfloor v_k \rfloor\}$$

$$P_{i+2} = P_i \cap \{x:\ x_k \geq \lceil v_k \rceil\}$$



- So we also remove a stripe of P_i **not containing** integer solutions: with these cuts we will sooner or later obtain integer sol. to the linear relaxations
- For binary problems we simply fix the variable to 0 or to 1

Putting all together

- Now we can write the Branch & Bound **scheme** for problems in this shape

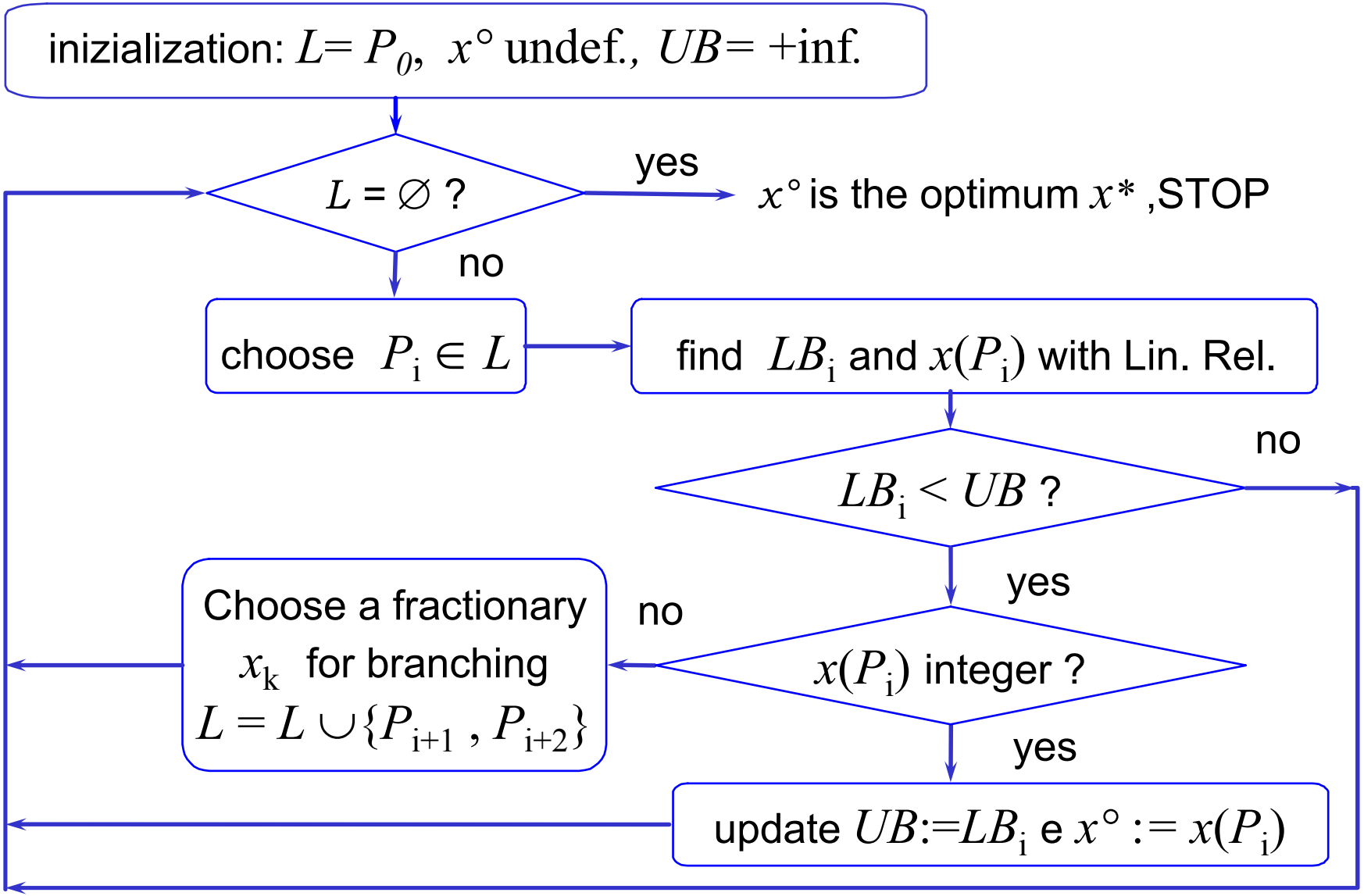
$$\begin{cases} \min \ c^{T}x \\ x \in P \\ x \in Z^{n} \quad (\text{o } x \in \{0,1\}^{n}) \end{cases}$$

Denote by L the **list of open subproblems** P_i

by x^o the **current optimum**, by UB (**upper bound**) the value $c^{T}x^o \geq c^{T}x^*$

by LB_i and $x(P_i)$ the **lower bound** obtained for **subproblem** P_i and the corresponding **solution**

Branch & Bound (written for min)



Observations

- It is an **exact** algorithm, that is, it guarantees, given enough time and up to the numerical precision used, to find the optimum if it exists
- For max problems all is inverted: from linear relaxations we have UB_i ; current optimum is an LB, initialized to $-\inf$; the check is $UB_i > LB$
- The evolution of the algorithm can be seen as a **tree** (a special kind of graph)
- It need to solve a **large number** of linear relaxations: integer linear programming is more difficult than linear programming
- However, if the solution of the first linear relaxation is already integer, we obtain the optimal solution without any branching (hence rapidly). This happens when we have a very good formulation of the problem…

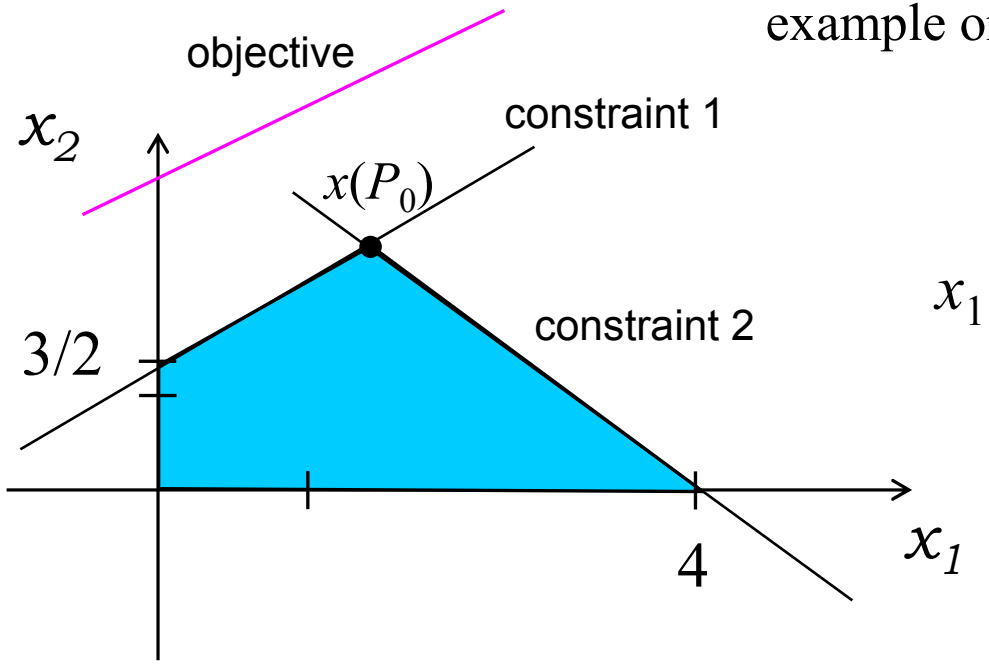
Example 1

$$\begin{cases} \max -x_1 + 2x_2 \\ -4x_1 + 6x_2 \le 9 \\ x_1 + x_2 \le 4 \\ x_1 , x_2 \ge 0 \\ x_1 , x_2 \in Z \end{cases} \quad P_0$$

$x(P_0)$ $x_1 = 3/2$, $x_2 = 5/2$

$UB_0 = 7/2$

Fractionary solution, no current optimum, we need banching, for example on x_1



x_1

$x_1 \le 1$

P_1

$x_1 \ge 2$

P_2

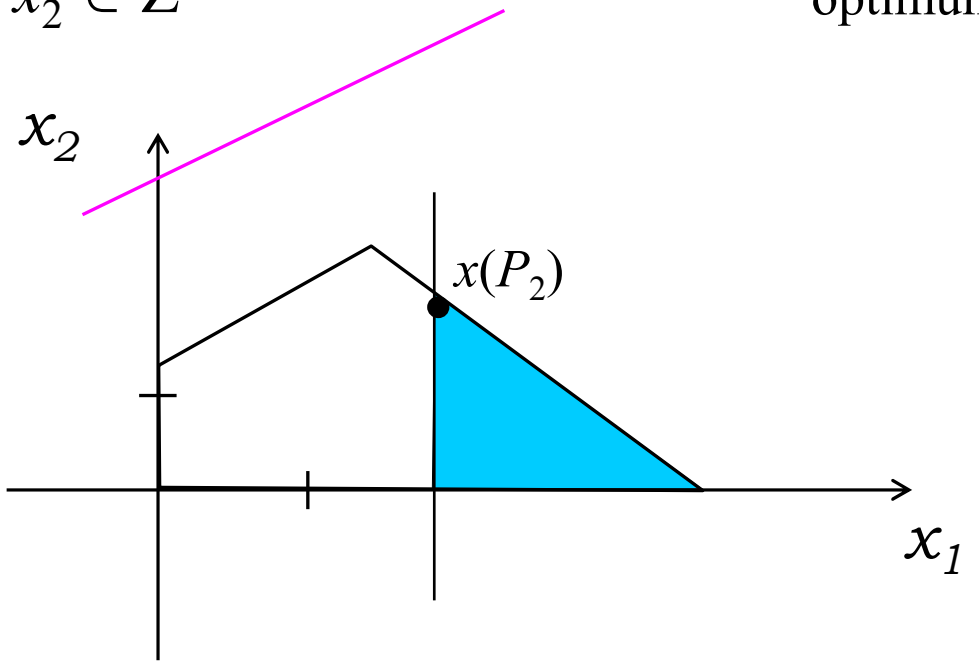
Example 1

$$P_2 \begin{cases} \max -x_1 + 2x_2 \\ -4x_1 + 6x_2 \le 9 \\ x_1 + x_2 \le 4 \\ x_1 \ge 2 \\ x_1 , x_2 \ge 0 \\ x_1 , x_2 \in Z \end{cases}$$

$$x(P_2) \quad \begin{matrix} x_1 = 2 \\ x_2 = 2 \end{matrix}$$

$$UB_2 = 2$$

Integer solution, update current optimum, no branching



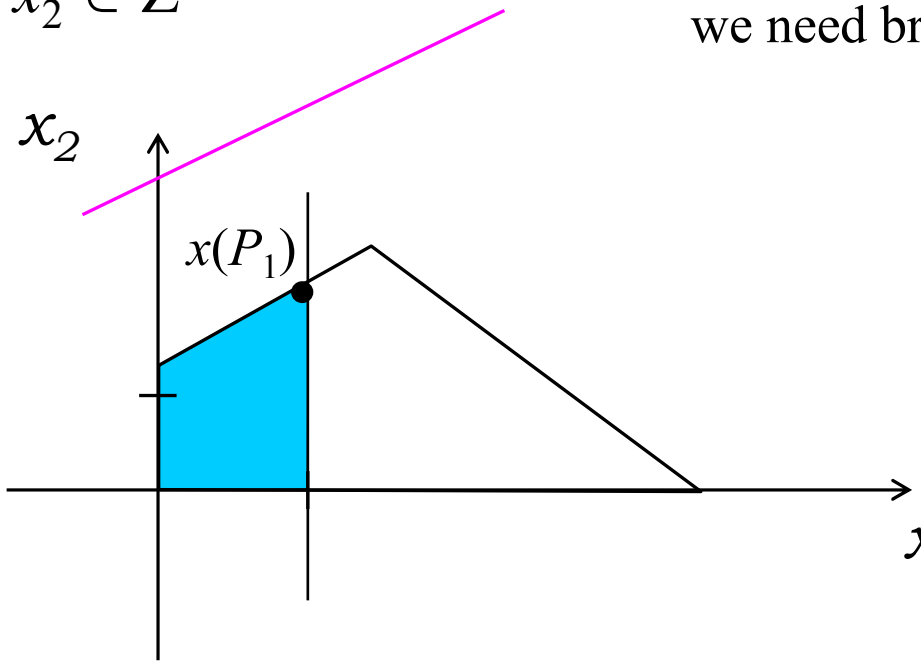
Example 1

$$P_1 \begin{cases} \max -x_1 + 2x_2 \\ -4x_1 + 6x_2 \le 9 \\ x_1 + x_2 \le 4 \\ x_1 \le 1 \\ x_1 , x_2 \ge 0 \\ x_1 , x_2 \in Z \end{cases}$$

$x(P_1)$ $x_1 = 1$, $x_2 = 13/6$

$UB_1 = 10/3$

UB_1 > current optimum, we cannot close, fractionary solution, we need branching



x_2

$x_2 \le 2$

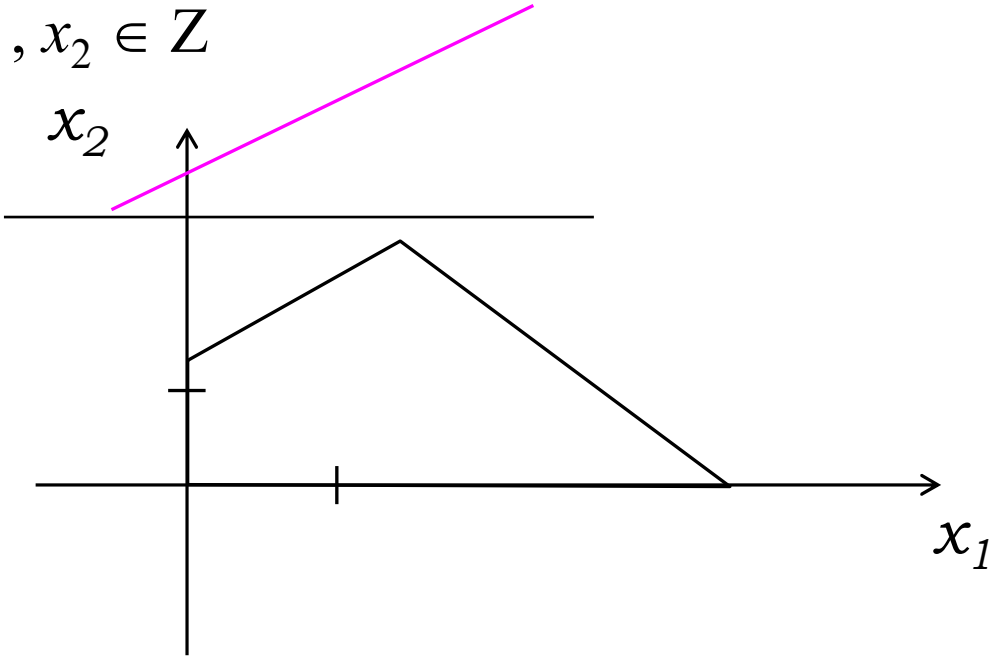
P_3

$x_2 \ge 3$

P_4

Example 1

$$\begin{cases} \max -x_1 + 2x_2 \\ -4x_1 + 6x_2 \le 9 \\ x_1 + x_2 \le 4 \\ x_1 \le 1 \\ x_2 \ge 3 \\ x_1 , x_2 \ge 0 \\ x_1 , x_2 \in Z \end{cases} \qquad P_4$$



Infeasible problem, we close it

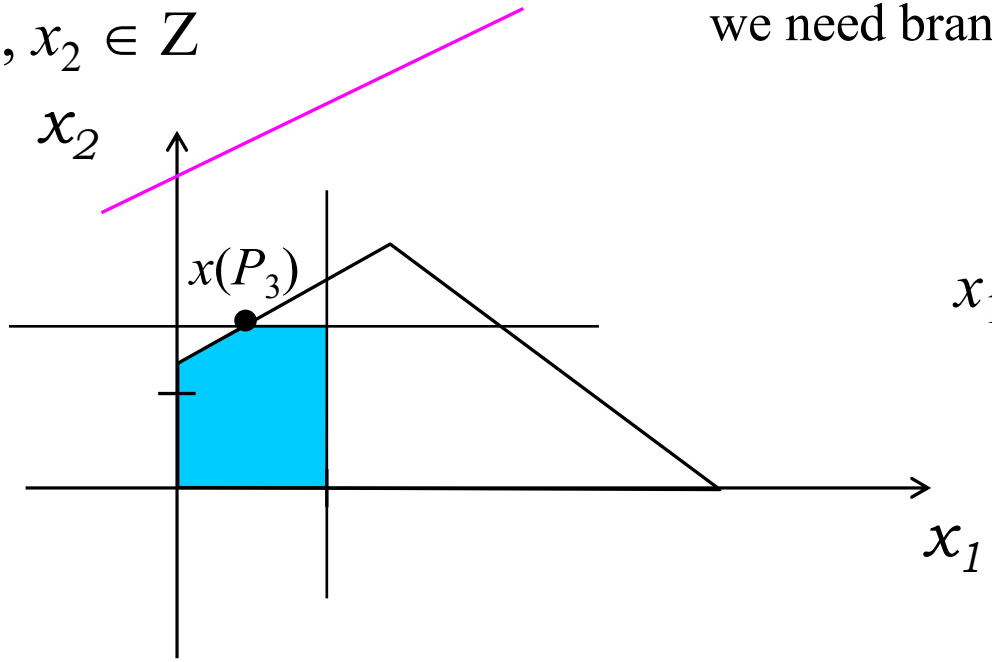
Example 1

$$\begin{cases} \max -x_1 + 2x_2 \\ -4x_1 + 6x_2 \le 9 \\ x_1 + x_2 \le 4 \\ x_1 \le 1 \\ x_2 \le 2 \\ x_1 , x_2 \ge 0 \\ x_1 , x_2 \in Z \end{cases} \qquad P_3$$

$x(P_3)$ $x_1 = 3/4$, $x_2 = 2$

$UB_3 = 13/4$

$UB_3 >$ current optimum, we cannot close, fractionary solution, we need branching



x_1

$x_1 \le 0$

P_5

$x_1 \ge 1$

P_6

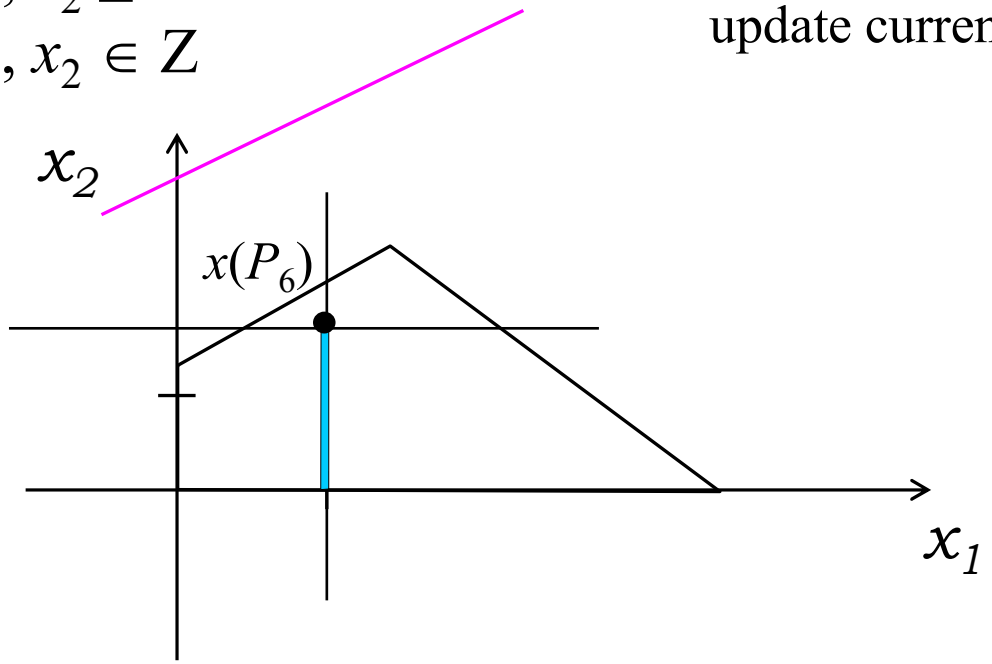
Example 1

$$
\begin{cases}
\max -x_1 + 2x_2 \\
-4x_1 + 6x_2 \le 9 \\
x_1 + x_2 \le 4 \\
x_1 \le 1 \\
x_2 \le 2 \\
x_1 \ge 1 \\
x_1 , x_2 \ge 0 \\
x_1 , x_2 \in Z
\end{cases}
$$

P_6 ⇒ $x(P_6)$ $x_1 = 1$, $x_2 = 2$

$UB_6 = 3$

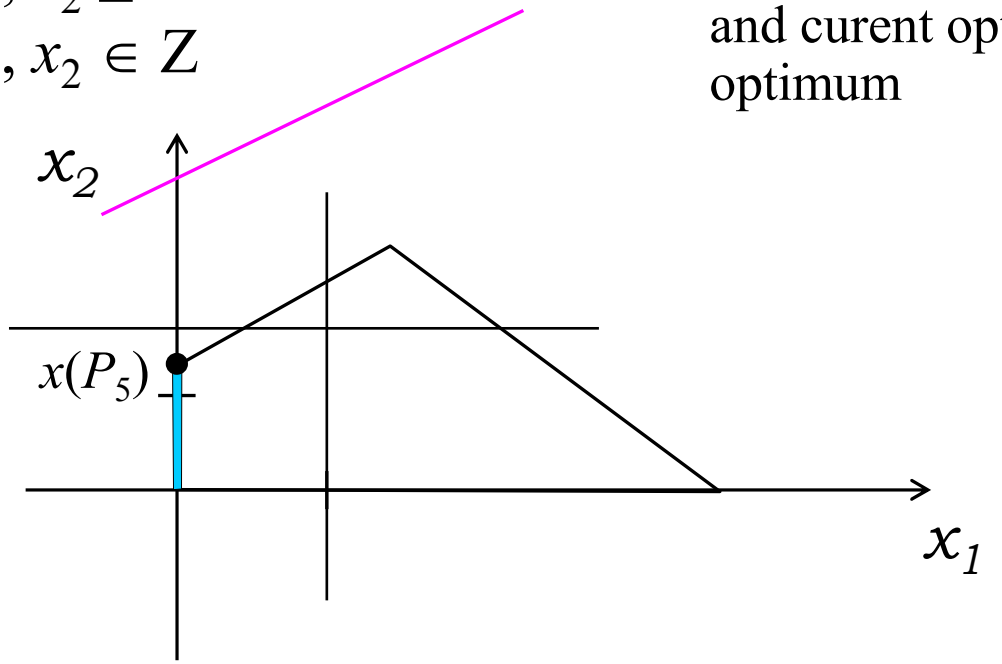
UB_6 > current optimum, we cannot close, integer solution, update current optimum



Example 1

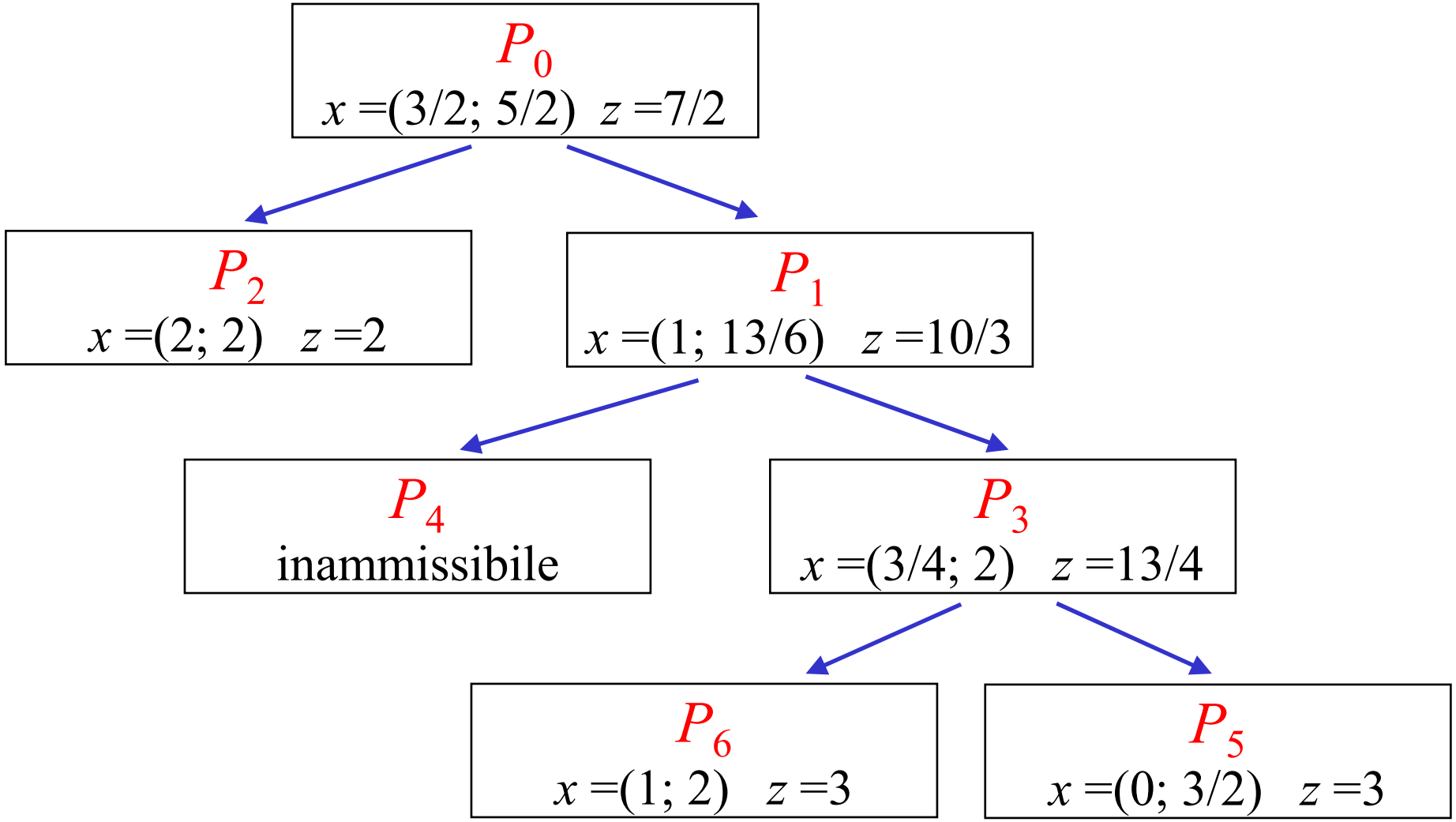
$$
\begin{cases}
\max -x_1 + 2x_2 \\
-4x_1 + 6x_2 \le 9 \\
x_1 + x_2 \le 4 \\
x_1 \le 1 \\
x_2 \le 2 \\
x_1 \le 0 \\
x_1 , x_2 \ge 0 \\
x_1 , x_2 \in Z
\end{cases}
\qquad P_5 \qquad \Rightarrow \qquad
x(P_5) \quad \begin{matrix} x_1 = 0 \\ x_2 = 3/2 \end{matrix}
\qquad UB_5 = 3
$$

$UB_5 \le$ current optimum, we can close it. There are no more open problems and curent optimum $x(P_6)$ is the final optimum



Branching tree

The evolution of the algorithm can be seen as this tree



Example 2

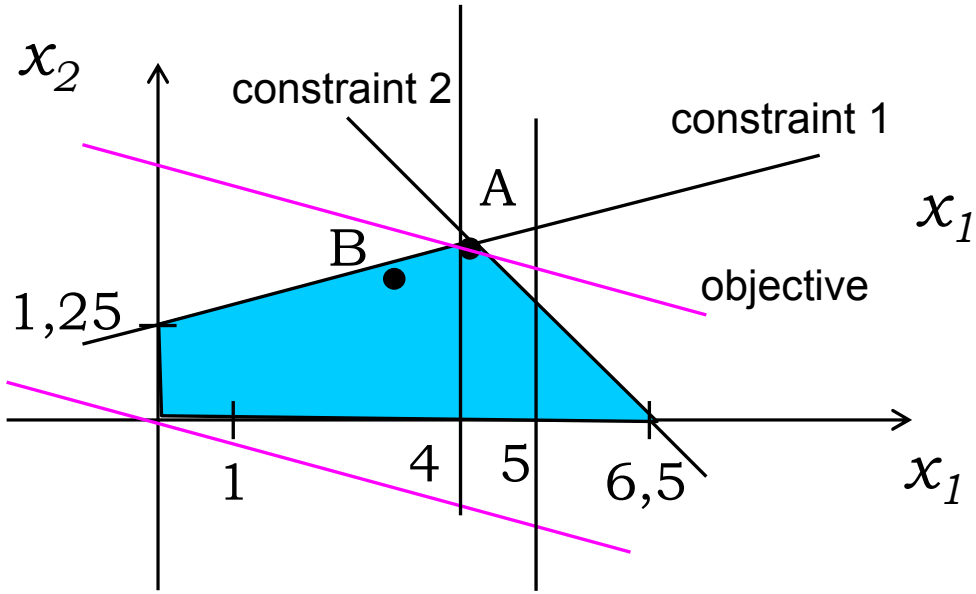
$$\max\ x_1 + 4x_2$$

$$P_0 \begin{cases} -x_1 + 4x_2 \le 5 \\ 2x_1 + 2x_2 \le 13 \\ x_1 \ge 0 \quad x_2 \ge 0 \\ x \in Z^2 \end{cases}$$

In this case we also have a feasible solution B

$$\Rightarrow \quad B \begin{cases} \hat{x}_1 = 3 \\ \hat{x}_2 = 2 \end{cases} \qquad LB = 11$$

$$\Rightarrow \quad A \begin{cases} \hat{x}_1 = 4{,}2 \\ \hat{x}_2 = 2{,}3 \end{cases} \qquad UB_0 = 13{,}4$$



$x_1 \le 4$

P_1

x_1

$x_1 \ge 5$

P_2

Example 2

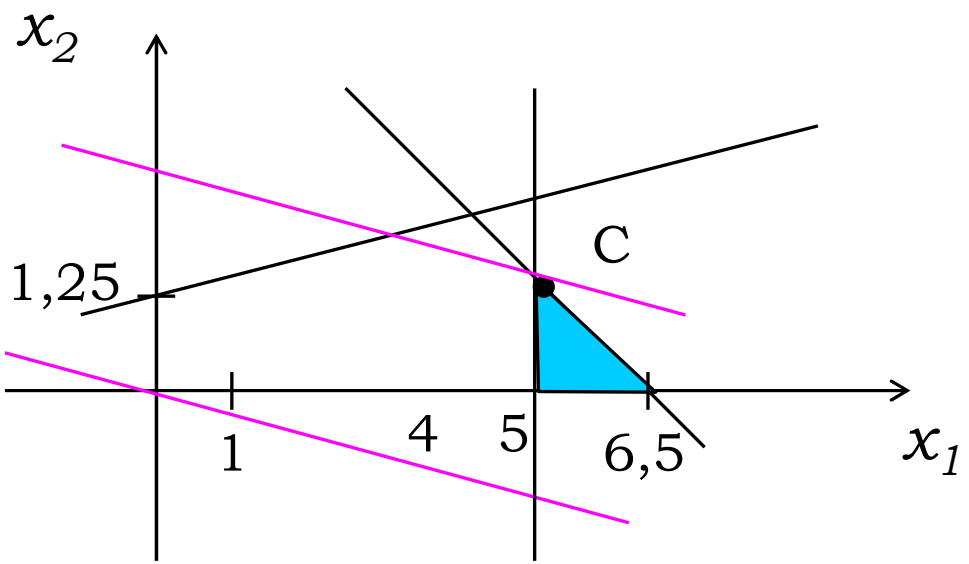
$$\max\ x_1 + 4x_2$$

$$P_2 \begin{cases} -x_1 + 4x_2 \le 5 \\ 2x_1 + 2x_2 \le 13 \\ x_1 \ge 0 \quad x_2 \ge 0 \\ x_1 \ge 5 \\ x \in Z^2 \end{cases}$$

$LB = 11$

$\Rightarrow \quad C \begin{cases} \hat{x}_1 = 5 \\ \hat{x}_2 = 1{,}5 \end{cases}$ $\qquad UB_2 = 11$

We close it because it is not better than LB



Example 2

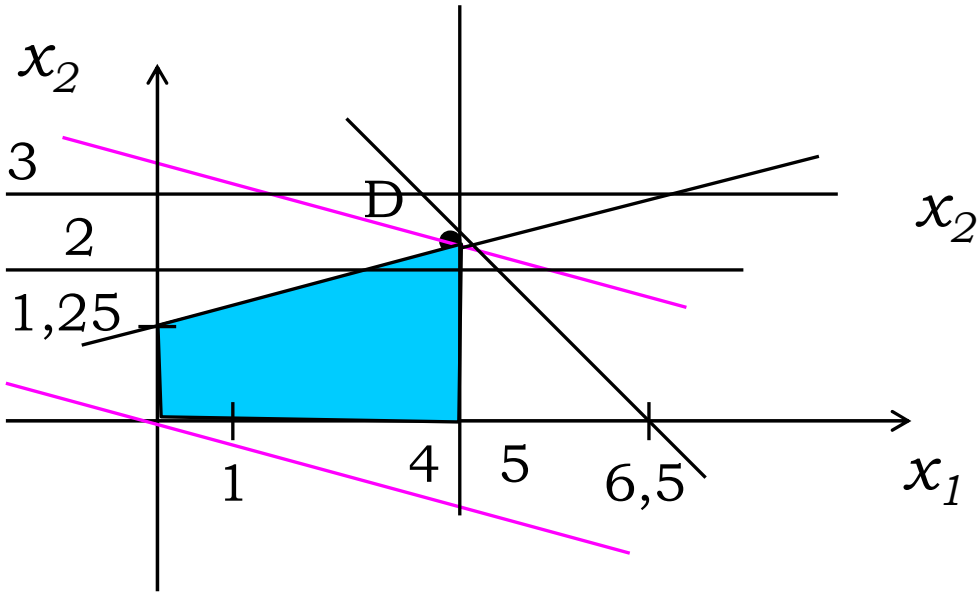
$$\max\ x_1 + 4x_2$$

$$P_1 \begin{cases} -x_1 + 4x_2 \le 5 \\ 2x_1 + 2x_2 \le 13 \\ x_1 \ge 0 \quad x_2 \ge 0 \\ x_1 \le 4 \\ x \in Z^2 \end{cases}$$

$LB = 11$

$$D \begin{cases} \hat{x}_1 = 4 \\ \hat{x}_2 = 2{,}25 \end{cases}$$

$UB_1 = 13$



x_2

$x_2 \le 2$

P_3

$x_2 \ge 3$

P_4 empty

Example 2

$$\max\ x_1 + 4x_2$$

$$P_3 \begin{cases} -x_1 + 4x_2 \le 5 \\ 2x_1 + 2x_2 \le 13 \\ x_1 \ge 0 \quad x_2 \ge 0 \\ x_1 \le 4 \\ x_2 \le 2 \\ x \in Z^2 \end{cases}$$

$LB = 11$

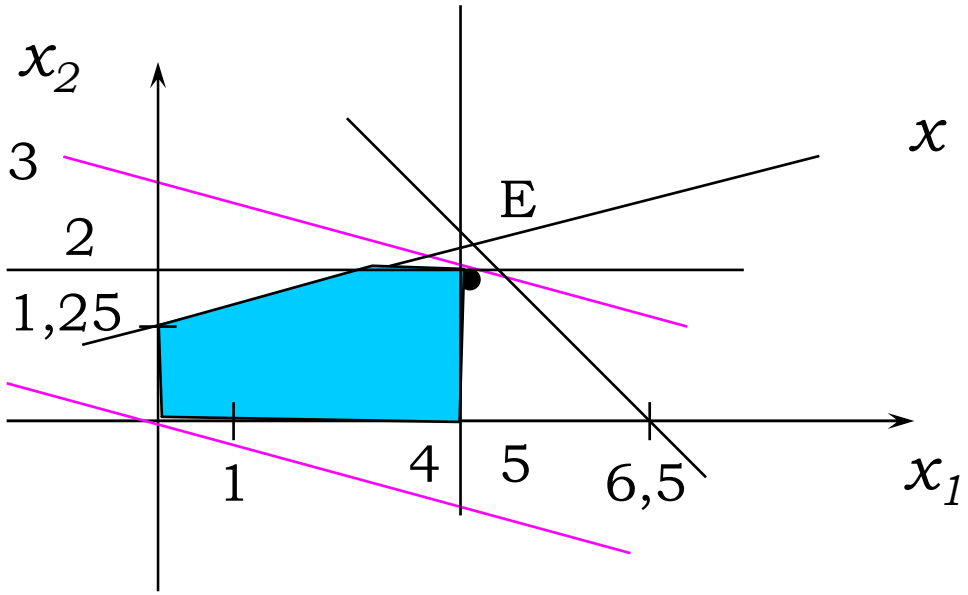
$$E \begin{cases} \hat{x}_1 = 4 \\ \hat{x}_2 = 2 \end{cases}$$

$UB_3 = 12$

x integer

update ***LB***

x **optimum**



Branching tree

The evolution of the algorithm can be seen as this tree