Africa Business School

Optimization

EMINES - 2023

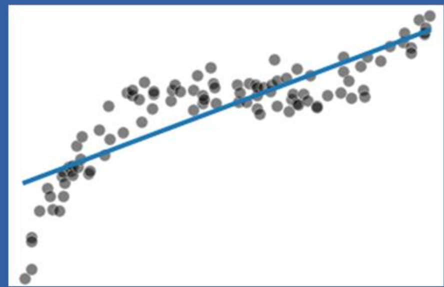
abs.um6p.ma

Combinatorial optimization

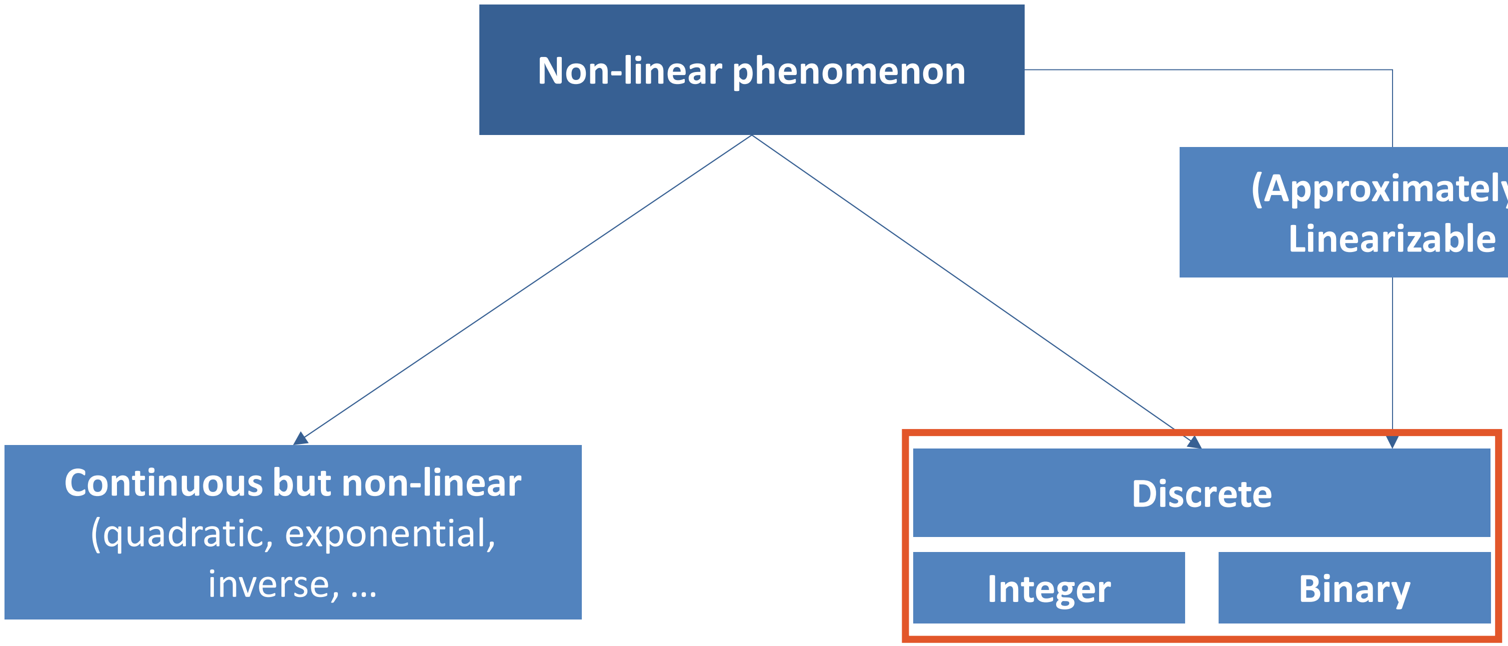
Linear optimization is a very powerful tool but..
« But we all know the world is nonlinear »

Actually…

"The world is NOT linear"



So what can we do ?



Introduction to Mixed Integer Linear Optimization

A linear program (LP)

Min	Sum[i=1,…,n] $c_i\ x_i$
Subject to	Sum[i=1,…,n] $a_{i,j}\ x_i \leq b_j$
	$x_i \geq 0$, all i=1,…, n
$\boldsymbol{x} \in \mathbb{R}^{\boldsymbol{n}}$	

A Mixed-Integer Linear Program (MILP)

Min	Sum[i=1,…,n] $c_i\ x_i$
Subject to	Sum[i=1,…,n] $a_{i,j}\ x_i \leq b_j$
	$x_i \geq 0$, all i=1,…, n
$x \in \mathbb{R}^{\boldsymbol{n}}$	
$\boxed{x_i \in \mathbb{N}\text{ , all i in J}}$	

- X in {0,1} is a particular case, widely-used
- Huge modeling power, but **much more difficult to solve**

Example : Set covering problem

Problem statement

Decision	Locating fire stations in 6 cities
Objective	Install as little fire stations as possible
Constraints	Being able to reach each city in at least 15 minutes from at least one stations

Data

Travel times table

	1	2	3	4	5	6
1		10	20	30	30	20
2	10		25	35	20	10
3	20	25		15	30	20
4	30	35	15		15	25
5	30	20	30	15		14
6	20	10	20	25	14	

Solution

Variables

x_i = 1 if we install a station in the city (for i = 1, …, 6), 0 otherwise

Objective

Min $\Sigma_i x_i$

Constraints

Let **J(i)** be the set of city's index such that the travel time between i and these cities be less than 15 minutes.

Ex :

- J(1) = {1, 2}
- J(2) = {1, 2, 6}
- ….

Then, for all i, at least one city of J(i) must host a fire station :

$$\Sigma_{j \text{ in } J(i)} x_j \geq 1$$

Example : Set covering problem. 2^{nd} formulation

Problem statement

Decision

Locating fire stations in 6 cities

Objective

Install as little fire stations as possible

Constraints

Being able to reach each city in at least 15 minutes from at least one stations

Data

Travel times table

	1	2	3	4	5	6
1		10	20	30	30	20
2	10		25	35	20	10
3	20	25		15	30	20
4	30	35	15		15	25
5	30	20	30	15		14
6	20	10	20	25	14	

Solution

Variables

x_i = 1 if we install a station in the city (for i = 1, …, 6), 0 otherwise
y_{ij} = 1 if the city i is served by a fire station located in city j

Objective

Min $\Sigma_i x_i$

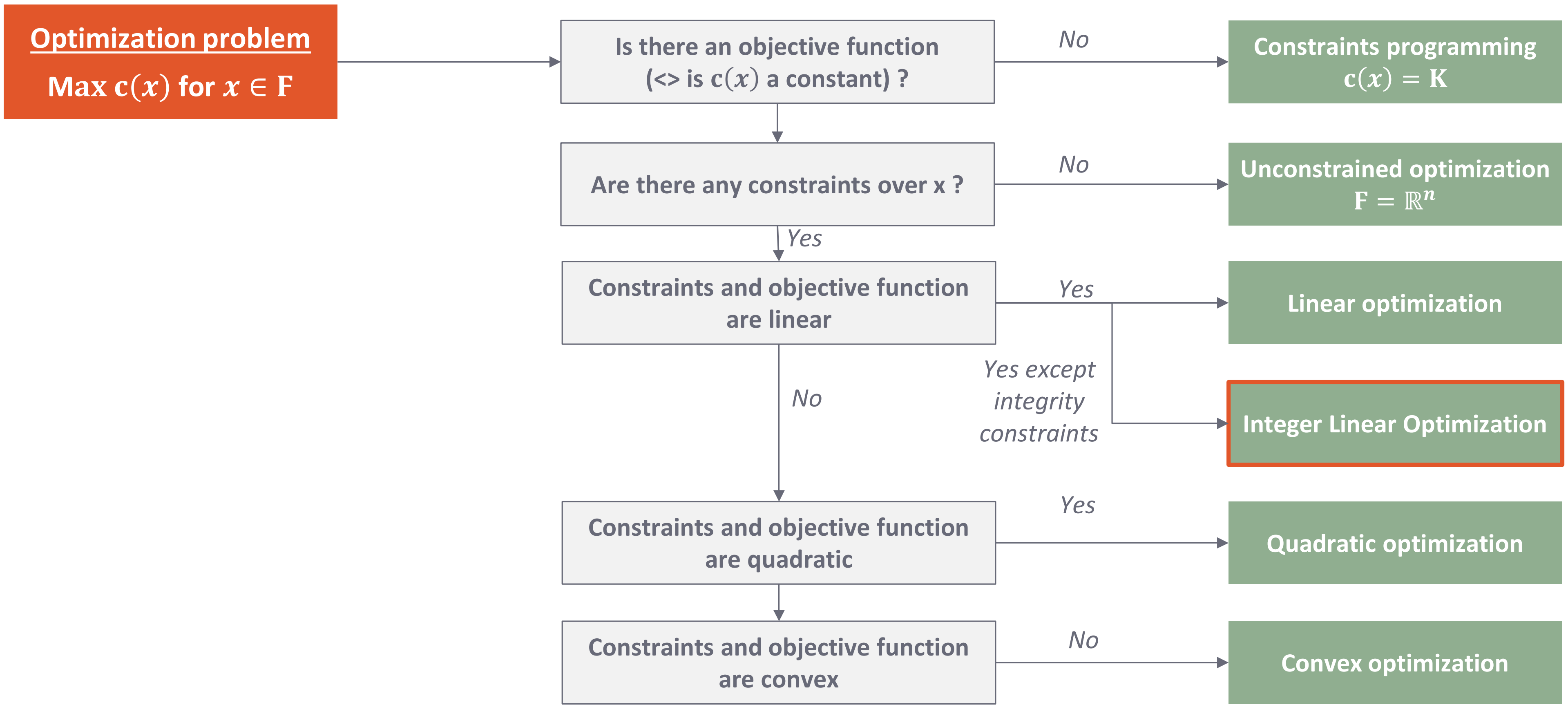
Constraints

$d_{ij} y_{ij} \leq 15$, for all i,j

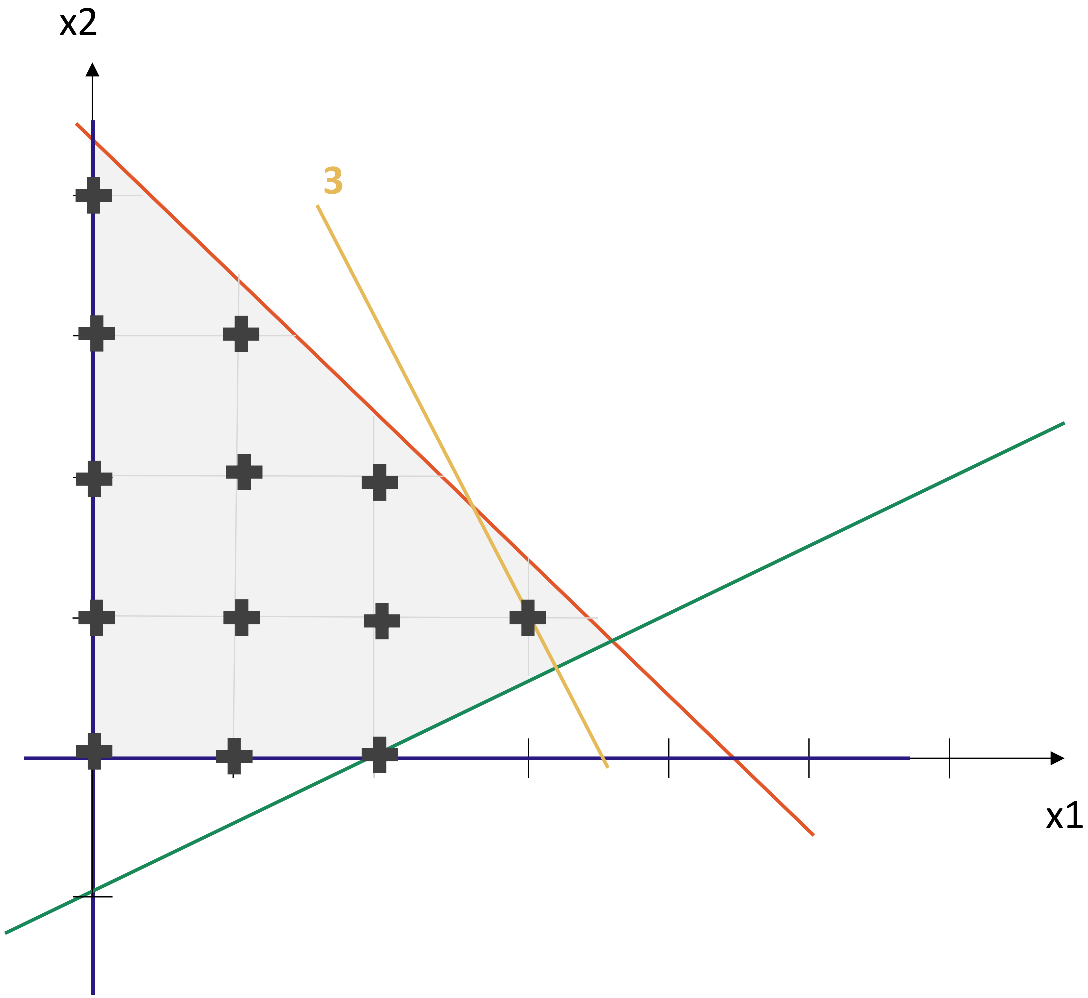
$\Sigma_j y_{ij} = 1$, for all i

$y_{ij} \leq x_i$, for all i,j

The choice of the resolution algorithm depends on the mathematical structure of the problem



Geometry of Linear Programming



The feasible set is a countable set with 13 solutions (✚)

Max 2x1 + x2

x1 + x2 ≤ 4.5

- x1 + 2 x2 ≥ -2

x1 ≥ 0, x2 ≥ 0

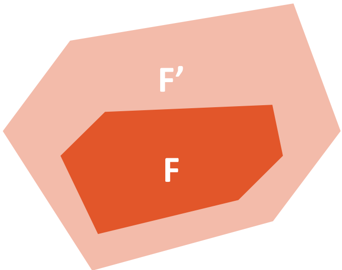
x1, x2 integer

Relaxation of an optimization problem

Optimization problem

(P) $p^* = \mathrm{Min}_{x \text{ in } F}\ f(x)$

(P') $p'^* = \mathrm{Min}_{x \text{ in } F'}\ f(x)$



The problem (P') is a relaxation of the problem (P)

Relationship between p* and p'* ?

Typical example : constraint removal

Notion of linear relaxation (or continuous relaxation)

A linear relaxation = continuous relaxation

Linear relaxation := **Relax (remove) the integer constraints**

The linear relaxation is a classical linear program.

What can we say on the obtained optimal value ?

The linear relaxation provides a lower bound (for a min problem) of the optimal solution

The **tighter** it is, the easier is the problem to solve

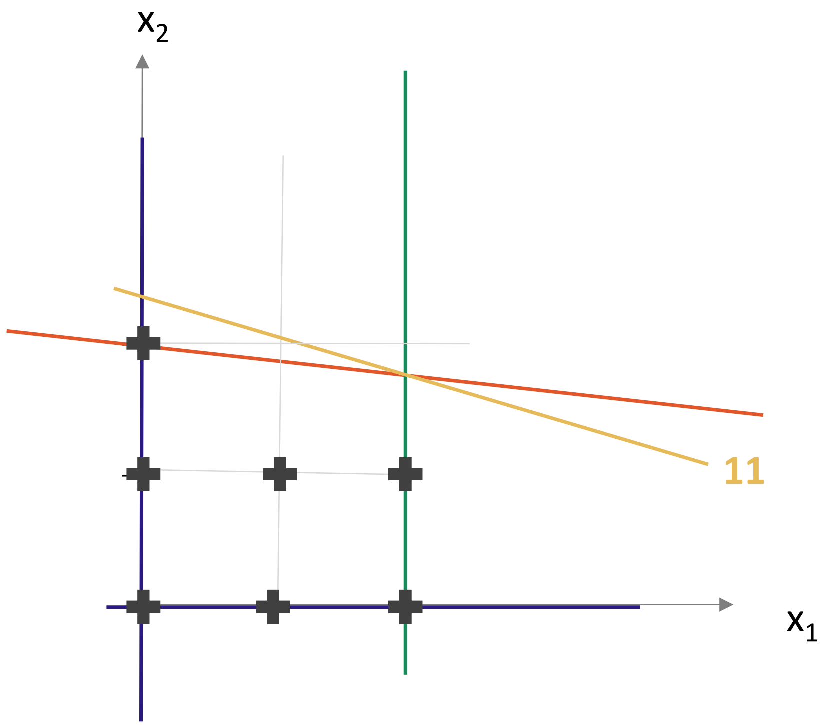
What about rounding the solution ?

Ex : $\Sigma_j x_j = 1$
What if we have : 0.2 + 0.3 + 0.1 + 0.4 = 1 ?

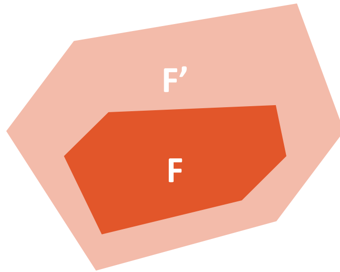
Why not solve the problem relaxed and round up the solution?

$$\begin{cases} \text{Max} & x_1 + 5x_2 \\ \text{S.t.} & x_1 + 10\,x_2 \leq 2 \\ & x_1 \leq 2 \\ & x_1, x_2 \geq 0 \\ & x_1, x_2 \in \mathbb{N} \end{cases}$$

- The optimal solution of the linear relaxation is (2, 1.8)
- Rounding :
  - (2,2) is not feasible
  - (2,1) → p = 7
- Optimal solution (0,2) → p = 10



How to (lower & upper) bound the optimal value ?



$\text{Min}_{x \text{ in } F}\ f(x)$

Feasible solution — $p = f(x)$, tt x in F

Optimal value — $p^* = f(x^*)$

Relaxation — $p'^* = \text{Min}_{x \text{ in } F'}\ f(x)$

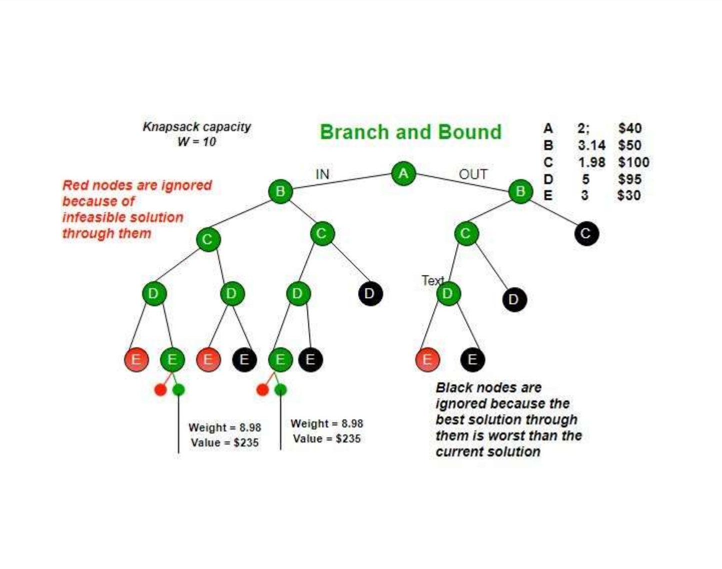
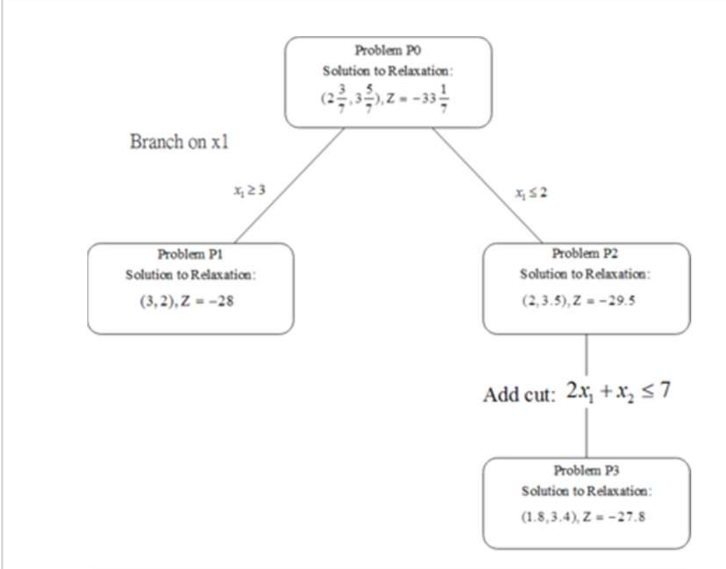
$\text{Max}_{x \text{ in } F}\ f(x)$

Relaxation — $p'^* = \text{Max}_{x \text{ in } F'}\ f(x)$

Optimal value — $p^* = f(x^*)$

Feasible solution — $p = f(x)$, tt x in F

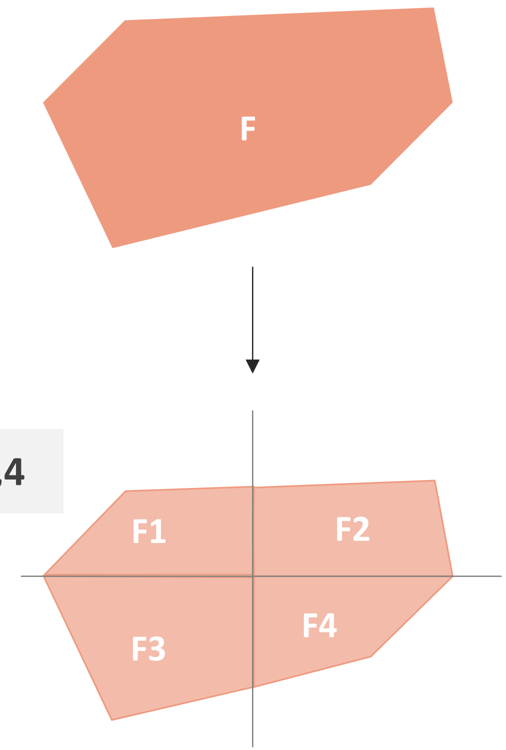
Several algorithms exist for solving MILPs

	Branch & Bound	Branch & Cut
Principle	Use the Linear relaxation to eliminate (= prune) some « infertile » branchs of the tree	Add valid constraints, in order to tighten the linear relaxation (and get closer to the convex hull)
Illustration		

The Branch & Bound algorithms combines two basic tenets

Divide & conquer (= branching)

$p^* = \min_{x \text{ in } F} f(x)$



$p_{i^*} = \min_{x \text{ in } Fi} f(x), i=1,\ldots,4$

$p^* = \min_{i=1,\ldots,4} \{p_i^*\}$

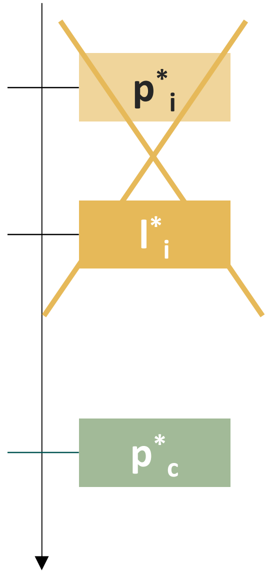
Detect and eliminate the none promising parts

Assume that F has been split into $F_1, \ldots, F_N$

Let i in {1, …, N} and assume that p_c^* is the optimal value on UF_j, for all $j < i$.

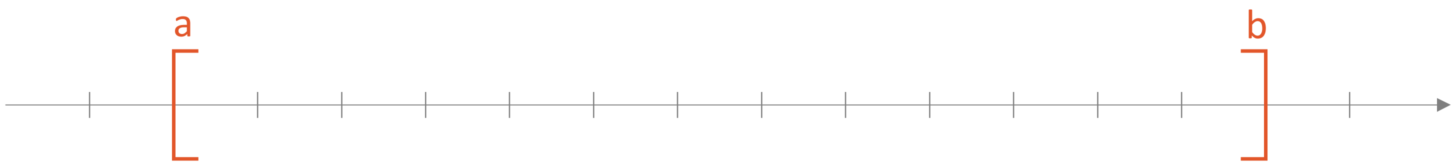
Suppose that we are able to compute (easily) l^*_i a lower bound of p_i^* : $l^*_i \leq p^*_i$

If l^*_i is such that $p_c^* < l^*_i$ then there is no hope that Fi contains the optimal solution. So no need to explore it further

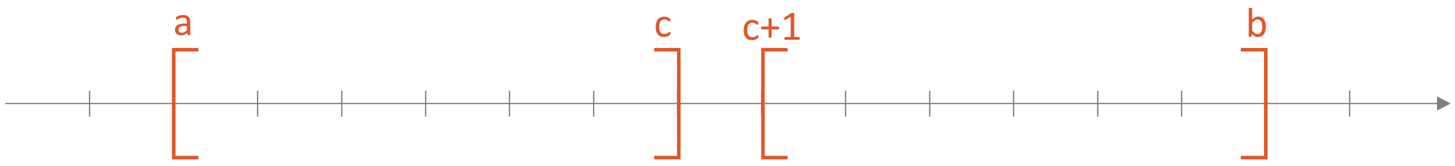


In the case of an integer optimization problem, there is a smart way to divide the feasible set into subsets

- Consider a variable x_1 and suppose that this variable has to be integer.
- Without loss of generality, we can consider that x_1 has to lie in the range [a, b]



- Let c an integer in [a, b]. Then we can split the feasible set in the following way :



At the same time we divide the feasible set in two and we eliminate non-integer solutions

Z* = min c(x) s.t. x in F

Branch & bound : what do we mean by branching ?

Suppose the variable x_1 has to be integer.

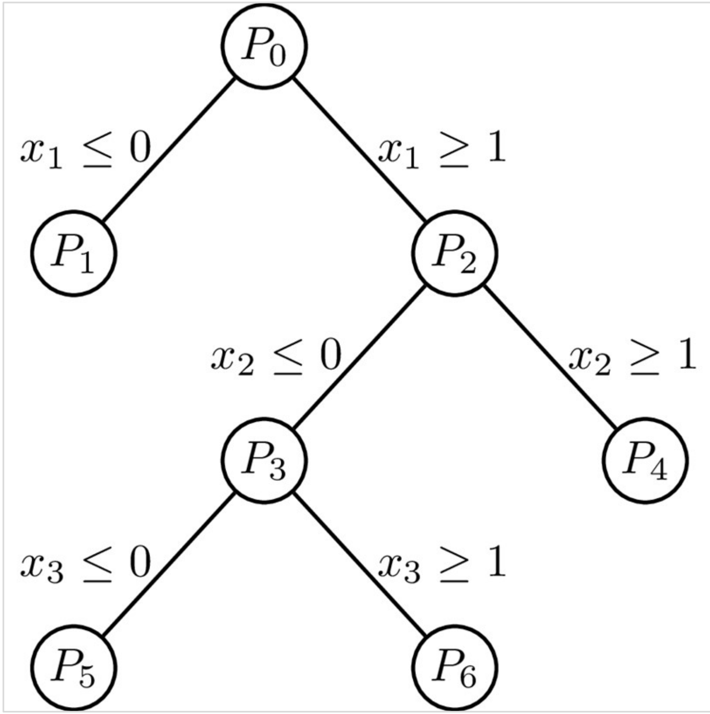
Solve the linear relaxation and get $x_1 = 0.57$

Then define 2 subproblems (branches)

P_1 : by adding $x_1 \leq 0$

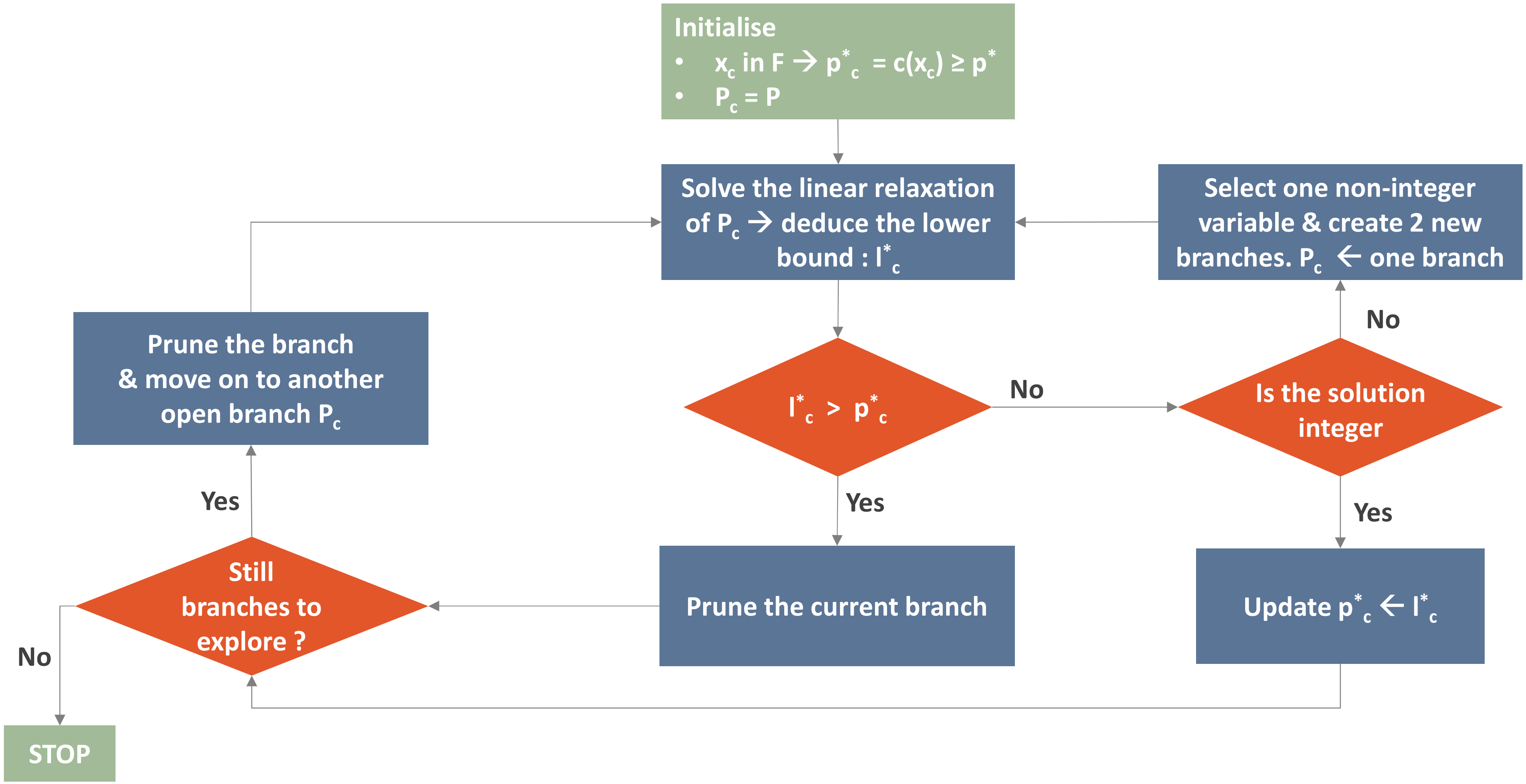
P_2 : by adding $x_1 \geq 1$

$Z^* = \min \{Z^*_1, Z^*_2\}$



Branch & bound

$Z^* = \min c(x)$ s.t. x in F



$Z^* = \min c(x)$ s.t. x in F

How to prune ?

Let p^*_1 the optimal value of of P_1

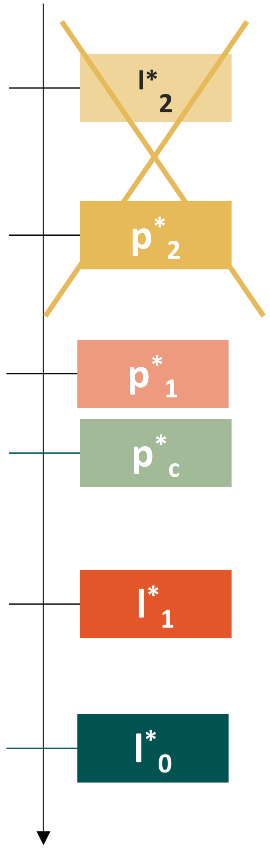
Let l^*_1 the optimal value of the linear relaxation of P_1

Let p*2 the optimal value of P2

Let l^*_2 the optimal value of the linear relaxation of P_2

p^*_c the best current solution

l^*_0 the linear relaxation of the ascendant branch of P_1 & P_2

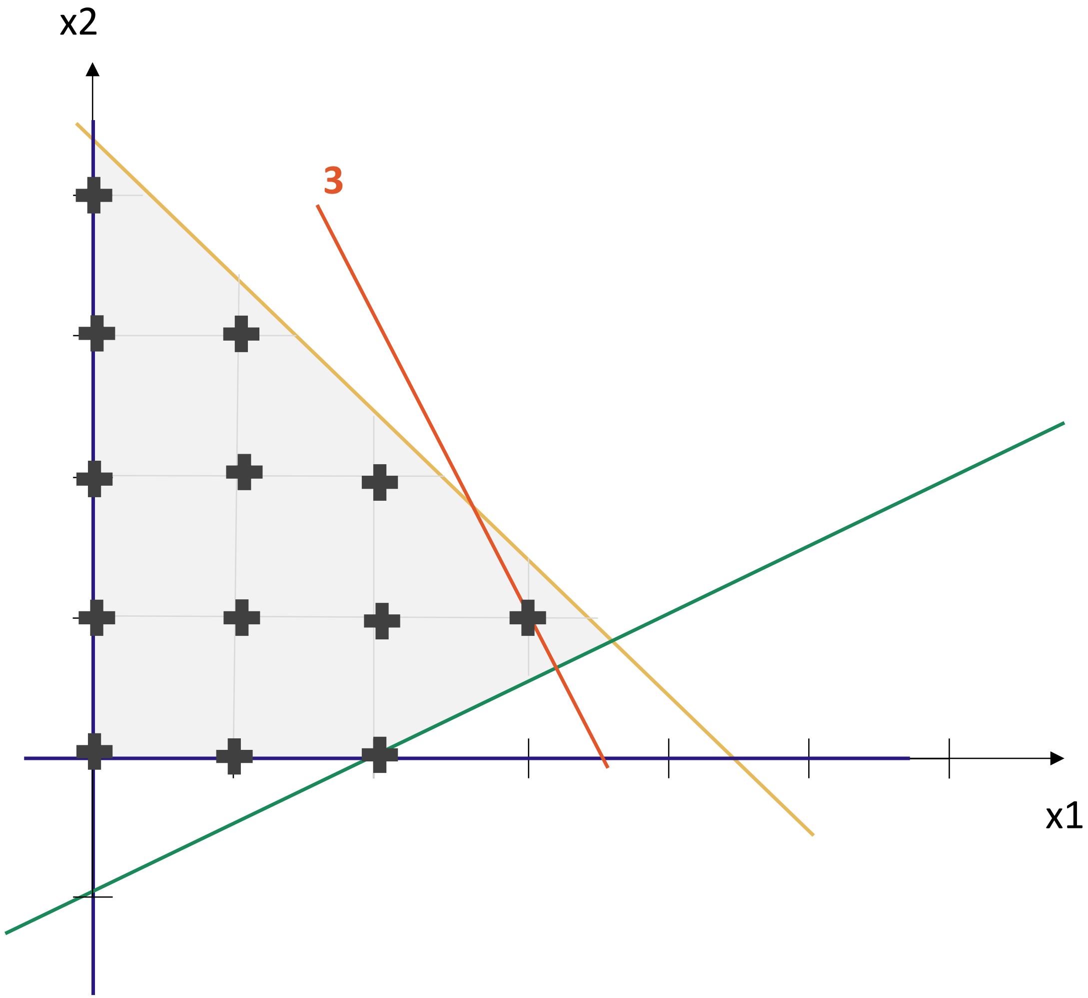


All descendants of a branch have an optimal value that is higher (= less good) than that of their ancestor, since the more you get in the tree, the more you add constraints

So if the ancestor is already worse than the current solution, we can delete the branch
→ ***no hope of improving the current solution on this branch***

→ So we can prune P2

Geometry of Integer Linear Programming



The feasible set is a countable set with 13 solutions (✚)

Max 2x1 + x2

x1 + x2 ≤ 4.5

- x1 + 2 x2 ≥ -2

x1 ≥ 0, x2 ≥ 0

x1, x2 integer

WARNING : different formulation may work for a same problem.
Which one is the best ?

Refresher : convexity

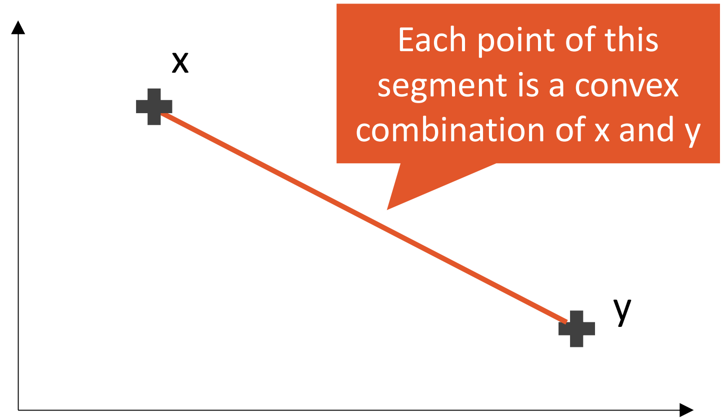
Convex combination

Definition

z is a **convex combination** of x and y if there exists λ in [0, 1] such that z = λ x + (1- λ) y

Illustration

Ex in $\mathbb{R}^2$



Convex set

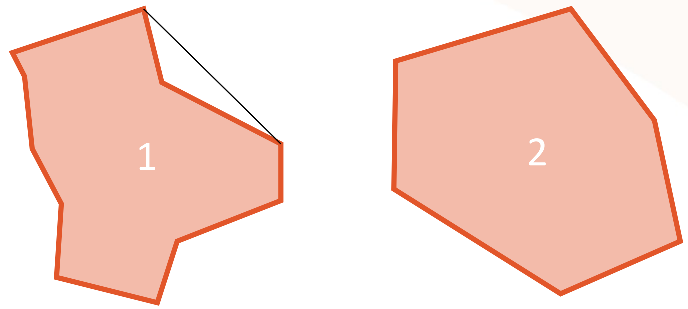
Definition

S is convex if and only if :

x,y in S ==> z in S, *for any z convex combination of x and y*

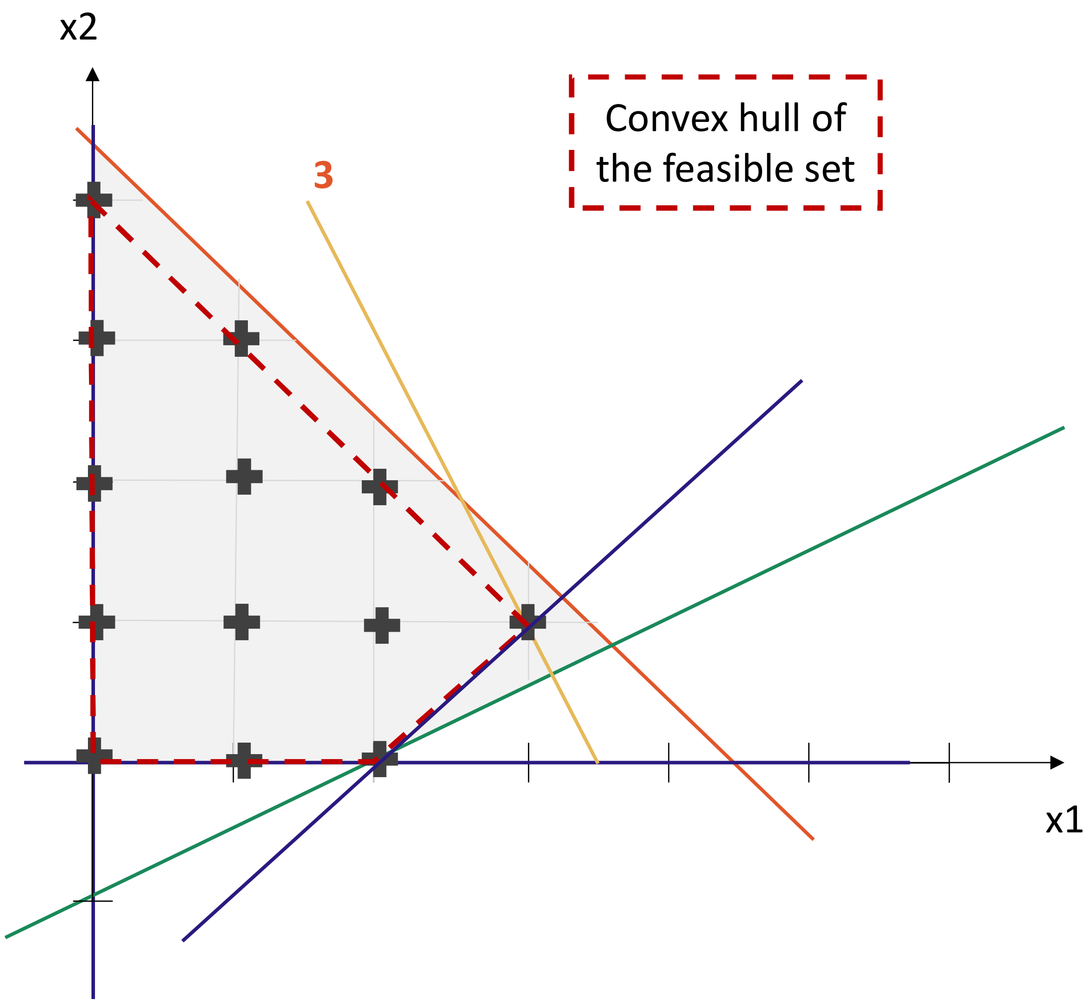
Illustration

Ex in $\mathbb{R}^2$



What about the feasible set of a MILP ?

Notion of convex hull



Definition : a convex hull is the smallest LP feasible region that contains all of the integer solutions

If you are able to describe **the convex hull** through a set of **linear constraints,** then **the integer solution is obtained by solving this new linear program**

Pb : determining the convex hull may be a very difficult problem

Definition : a constraint if valid if adding it does not reduce the feasible set

Branch & cut, or cutting planes method

Overview

To iteratively refine the feasible set of the relaxed problem (P) using linear inequalities, termed **cuts**

Basic idea

Let consider a linear constraint C added to the problem.

- Let F be the feasible set of a problem, and F_R the feasible set of its linear relaxation
- Let (P) be the initial problem and (P+C) the problem augmented with the constraint C.

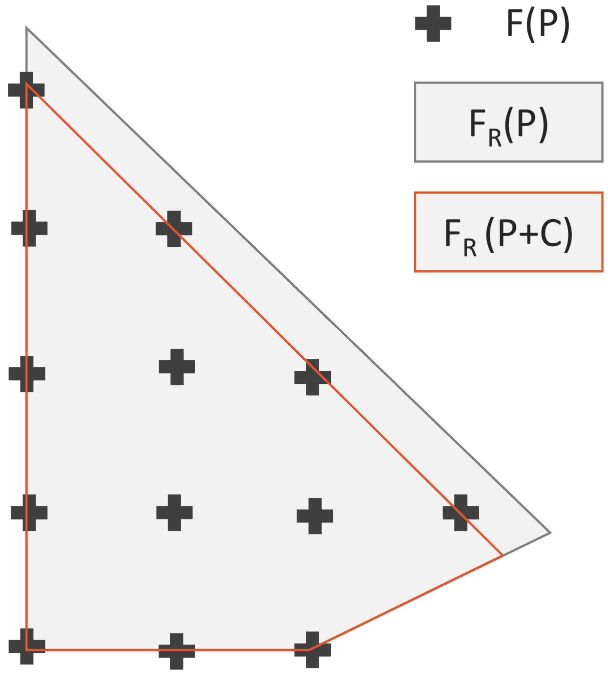
Then, **the constraint C is a cut if** :

– $F(P+C) = F(P)$: It does not eliminate feasible solutions of the original IP problem.

– $F_R(P+C) \subset F_R(P)$: The cut removes the optimal solution of the relaxed LP problem.

Ultimate goal

The optimal integer solution becomes an extreme point and therefore can be found by solving the LP problem.



Ex : the vehicle allocation problem

Example : the vehicle allocation problem

Consider a transportation company that has to drive people from a set of stations to their workplace.

Some itineraries have already been defined and for each of them the distance and number of people to drive is known.

You have different types of vehicles (large, medium, small) and you want to **allocate a set of vehicles to each itinerary i, in order to minimize the cost.**

Data :

- The distance per itinerary D_i
- The number of person to drive in each itinerary P_i
- The number of available vehicles : N_k (k : large, medium, small)
- The capacity K_k of each type of vehicle
- The cost per unit of distance per vehicle : C_k

Data

Itinerary	Demand	Distance
I1	36	12,9
I2	63	5,6
I3	58	16,1
I4	40	14,6
I5	37	4,7
I6	50	13,5
I7	25	14,5
I8	39	7

Vehicle	Capacity	Number	Cost
Large	40	5	2,30
Medium	17	8	3,10
Small	6	15	5,20

Example : the vehicle allocation problem

Data

Itinerary	Demand	Distance
I1	36	12,9
I2	63	5,6
I3	58	16,1
I4	40	14,6
I5	37	4,7
I6	50	13,5
I7	25	14,5
I8	39	7

Vehicle	Capacity	Number	Cost
Large	40	5	2,30
Medium	17	8	3,10
Small	6	15	5,20

Solution

Objective	372,55

Xij	Large	Medium	Small	Capacity
I1	0,67	0,53	0,00	36
I2	0,00	3,71	0,00	63
I3	1,45	0,00	0,00	58
I4	1,00	0,00	0,00	40
I5	0,00	1,47	2,00	37
I6	1,25	0,00	0,00	50
I7	0,63	0,00	0,00	25
I8	0,00	2,29	0,00	39
Total	5	8	2	

Example : the vehicle allocation problem

Data

Itinerary	Demand	Distance
I1	36	12,9
I2	63	5,6
I3	58	16,1
I4	40	14,6
I5	37	4,7
I6	50	13,5
I7	25	14,5
I8	39	7

Vehicle	Capacity	Number	Cost
Large	40	5	2,30
Medium	17	8	3,10
Small	6	15	5,20

Linear solution

Objective 372,55

Xij	Large	Medium	Small	Capacity
I1	0,67	0,53	0,00	36
I2	0,00	3,71	0,00	63
I3	1,45	0,00	0,00	58
I4	1,00	0,00	0,00	40
I5	0,00	1,47	2,00	37
I6	1,25	0,00	0,00	50
I7	0,63	0,00	0,00	25
I8	0,00	2,29	0,00	39
Total	5	8	2	

Integer solution

Objective 631,56

Xij	Large	Medium	Small	Capacity
I1	1,00	0,00	0,00	40
I2	0,00	3,00	2,00	63
I3	1,00	1,00	1,00	63
I4	1,00	0,00	0,00	40
I5	0,00	1,00	4,00	41
I6	0,00	3,00	0,00	51
I7	1,00	0,00	0,00	40
I8	1,00	0,00	0,00	40
Total	5	8	7	

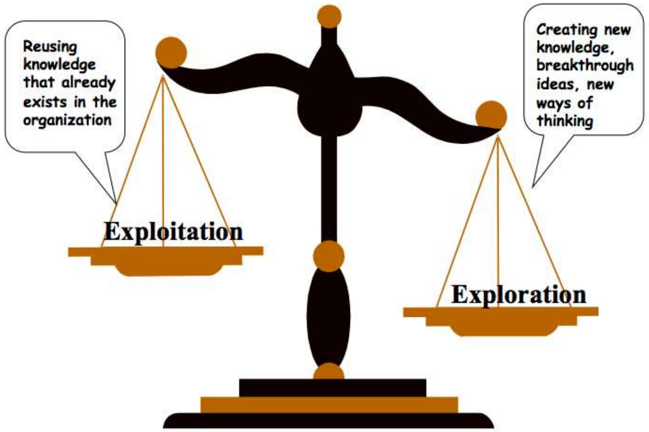
Metaheuristics

What are heuristics ?

Definition : heuristics	A heuristic is a technique that seeks a good (i.e. almost optimal) solution in a reasonable computation time without being able to guarantee neither optimality nor admissibility. Most heuristics are based on the notion of neighborhood.
Definition : neighborhood	A neighborhood is the set of solutions obtained from a given solution by performing a small number of simple transformations. The precise definition of neighborhood depends on the problem.
Example : neighborhood	For an optimization variables with binary variables, the neighborhood of a solution $(x_1, \dots, x_n)$ is the set of $(y_1, \dots, y_n)$ such that : - $y_k = 1 - x_k$ for a given k - $y_k = x_k$ otherwise

What are metaheuristics ?

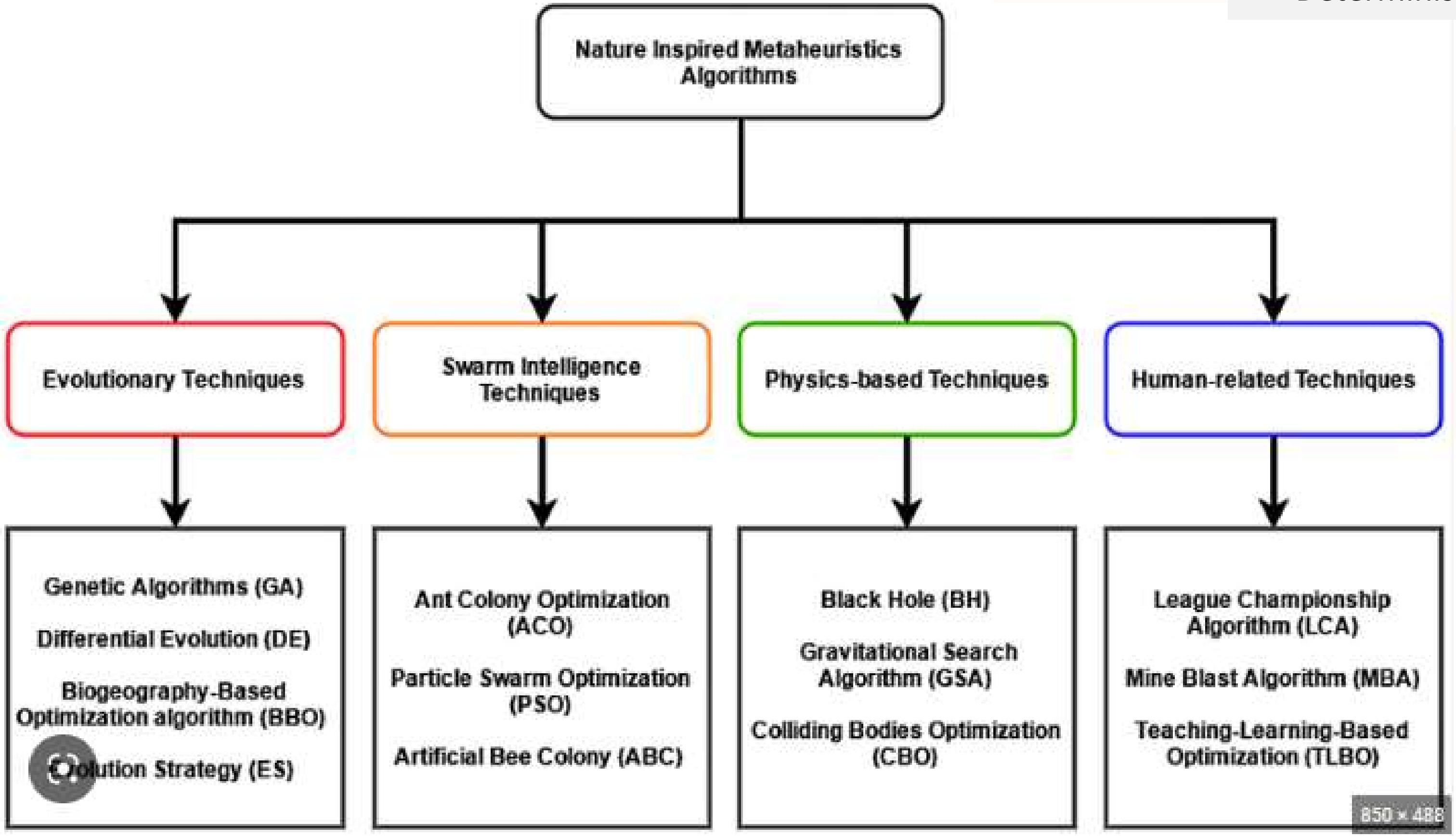
- Metaheuristics are strategies that “**guide**” the search process.
- The goal is to efficiently explore the search space in order to find **(near-)optimal solutions**.
- Techniques which constitute metaheuristic algorithms range **from simple local search procedures to complex learning processes.**
- Metaheuristics are **not problem-specific**.
- They balance between **exploitation/intensification and exploration/diversification**
- They may incorporate mechanisms to **avoid getting trapped in confined areas** (local optimas) of the search space.
- The majority of metaheuristics are population-based



Africa Business School

Metaheuristics are of various inspirations

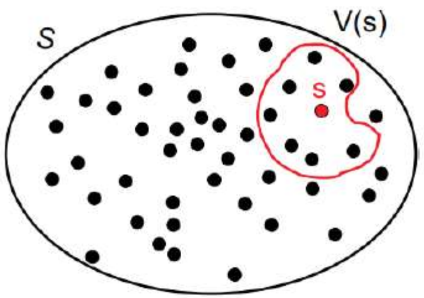
- Iterative versus greedy
- Nature versus nonnature inspired
- Deterministic versus stochastic



Local search

Neighborhood structure

A neighborhood structure N : $S \rightarrow 2^S$ is a function associating a set of neighbors, called N(x) ⊆ S, to every solution x ∈ S



Move

Neighborhood structures are often defined by a specific move.

- A move is an operation applied to a solution x yielding another solution x'.
- The same problem may have multiple different neighborhoods defined on it.
- Different neighborhood structures originate different local search algorithms

Pseudo-code

1. Generate an initial solution → s_0
2. Current solution $s_i = s_0$
3. Pick $s_j \in N(s_i)$.
4. If $f(s_j) < f(s_i)$, then $s_i = s_j$
5. Else, $N(s_i) = N(s_i) - s_j$
6. If $N(s_i) \neq \emptyset$, then GO TO 3
7. Else END

Africa Business School

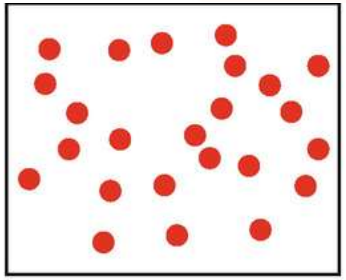
Simulated annealing

Introduction

- Simulated annealing is a metaheuristic to approximate global optimization
- The name and inspiration comes from annealing in metallurgy, a technique involving heating and controlled cooling of a material to increase the size of its crystals and reduce their defects.
- The idea is to run an algorithm that doesn't get stuck in a local optima by renewing the process as we do for cooling.
- It is quite similar to gradient descent

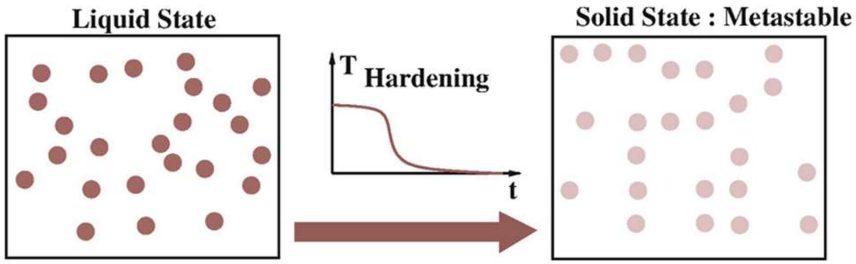
Real annealing

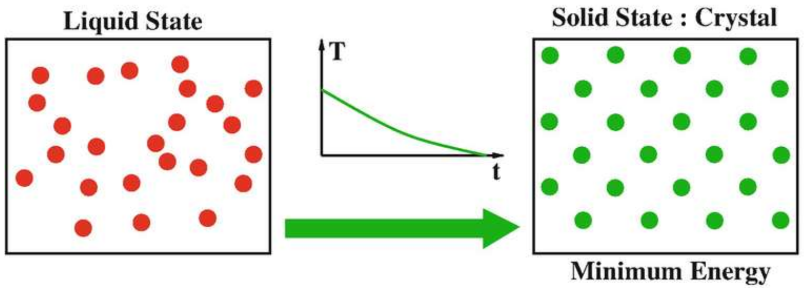
- In the liquid phase all particles arrange themselves randomly, whereas in the ground state of the solid, the particles are arranged in a highly structured lattice, for which the corresponding energy is minimal.
- The ground state of the solid is obtained only if:
  - the maximum value of the temperature is sufficiently high
  - the cooling is done sufficiently slow
- Strong solid are grown from careful and slow cooling.



Simulated annealing

If the cooling process is done quickly	The particles don't have enough time to form a structured crystal network. We obtain a metastable solid which translate to a fragile metal.
If the cooling process is slowly enough	The particles have enough time to form a structured crystal network. We obtain a stable state ie a minimum energy which translates into a strong solid





Simulated annealing

Principles

- **Fundamental idea** : allow moves resulting in solutions of worse quality than the current solution (**uphill moves**) in order to escape from local minima.
- Uphill moves are accepted with a probability given generally by the Boltzmann distribution:

$$\boldsymbol{P}(\boldsymbol{T},\boldsymbol{s},\boldsymbol{s}^*) = \boldsymbol{exp}\left(-\frac{\boldsymbol{f}(\boldsymbol{s}^*)-\boldsymbol{f}(\boldsymbol{s})}{\boldsymbol{T}}\right)$$

- It uses a control parameter, called temperature, to determine the probability of accepting non-improving solutions.
- This temperature will decrease as long as the research goes, according to the so-called **Cooling schedule :**

$$T_{k+1} = Q(T_k, k)$$

Algorithm

S* ← GenerateInitialSolution()

$T \leftarrow T_0$

While termination conditions not met do

    s' ← PickAtRandom(N (s))

    If (f (s) < f (s*)) **then**

        s* ← s

    Else s* ← s with probability p(T, s , s*)

    Endif

    Update(T) according to cooling schedule

Endwhile

Genetic Algorithm

Genetic algorithm

- **Step 1** Encoding of a solution in a binary string
- **Step 2** Random generation of a population
- **Step 3** Calculate fitness (:= quality) of each solution
- **Step 4** Select a set of parents among the population based on fitness
- **Step 5** Generate new individuals with **crossover** and **mutation** from selected parents

Genetic algorithm : 3 questions arise

1 How to represent the solution ?

Combine :
- **Completeness**
- **Connexity**
- **Efficiency**

Examples of possible representation of solutions

Binary encoding 1 0 1 0 0 0 1 1 0 0 *Ex : Knapsack problem*	**Vector of discrete values** 5 7 6 4 0 1 9 7 3 2 *Ex : assignment problem*
Vector of real values 5.23 8.45 6.32 4.10 *Ex : Continuous optimization*	**Permutation** 1 4 3 9 8 5 7 0 2 6 *Ex : Traveling salesman problem*

2 Which fitness function to guide the search

→ How describing the fitness of the solution

→ In a way to guide the selection for next iterations

3 How to deal with the constraints ?

→ A priori : by « creating » only feasible individuals

→ A posteriori : by penalizing (in the fitness function) the non-satisfaction of the constraints.

Genetic algorithm : cross-over and mutation

Cross-over

0	1	1	1	0	0	1	0	1	1	Parent 1
1	1	1	0	1	0	1	1	0	1	Parent 2
0	1	1	1	0	0	1	1	0	1	Offspring

Mutations

0	1	1	0	1	$\boxed{1}$	1	0	0	1
0	1	1	0	1	0	1	0	0	1

Metaheuristics : let's practice on a genetic algorithm

Knapsack problem :

- N items
- Each item has a weight w_i
- Each item has a value v_i
- A knapsack of capacity K

→ Find the list of item to put in your knapsack that maximize the value while satisfying the capacity

Let's practice !

Solve this problem with a genetic algorithm :

- Implement Step 1
- Implement Step 3

Genetic algorithm

Step 1	Encoding of a solution in a binary string
Step 2	Random generation of a population
Step 3	Calculate fitness of each solution
Step 4	Select a pair of parents based on fitness
Step 5	Generate new string with crossover and mutation until a new population has been produced

Repeat step 2 to 5 until satisfying solution is obtained

Pros and cons of metaheuristics

– **Cons**

1. Efficiency
2. Not necessary to have an analytic form of the problem
3. Applicability : no requirement on the problem structure and size
4. Can exploit specificity of real-life instances

Pros +

1. No guarantee of optimality
2. Requires parameter tuning
3. Interpretability of the solution (vs. heuristic)
4. Lack of stability

My advice : always start with exact methods

When using metaheuristics?

Balancing criteria :

1. **Problem complexity**
2. **Size and structures of the instances : all large instances are not necessarily hard**
3. **Required search time, frequency of the resolution, target machines**
4. **Development cost**

Metaheuristics are recommanded in case of :

- Easy problem with huge instances
- Easy problem with hard real-time resolution constraints
- NP-hard problems with moderate size and/or difficult instances
- Nonanalytic (« Black-box ») problem, that requires simulation or human evaluation of the solutions

Example of GPS software :

GPS software are actually using heuristics to solve their shortest path between 2 locations problem.

Even if the problem is polynomial, the size of the instances makes the use of polynomial algorithm (such as Dijkstra algorithm) too time consuming.

Modeling tips

Binary variables are SOOOOOOO powerful

X in {0, 1}

- **Yes/No decisions:** Do I create a new route or not ?
- **What is the best option** among a finite set of possibilities ?
- **State of a system :**
  - Is my plant **ON or OFF**
  - **Is my plant currently producing product i,j,…**
- **Logical constraints :**
  - *If my plant if ON, then I can …*
  - *If my plant if ON, then I shall …*
  - *If my plant is ON **AND** another conditions, then ..*
  - *If condition A **OR** condition B, then …*
- **Priorisation :** I can use the resource A only if I used all of my resource B

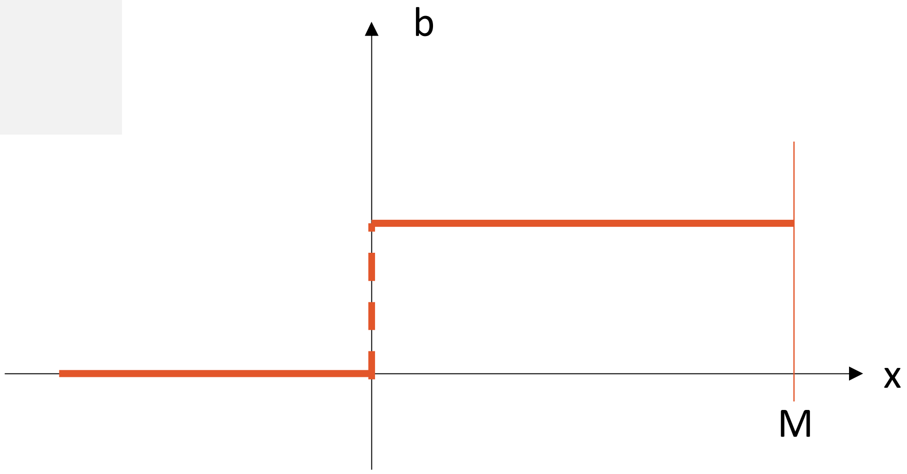
Generally (~95% of cases) : integer variables are binary variables

Tips 1 : BigM constraints : or how to link continuous and binary variables

Let x be continuous, and b binary :

We want that $x > 0$ implies $b = 1$,

or conversely : $b = 0$ implies $x \leq 0$



The constraint for that is :

$$x \leq M\,b$$

- **If $b = 1$** : the constraint becomes

  $x \leq M$

  (with M sufficiently big for never being binding)

- **If $b = 0$** : the constraint becomes

  $x \leq 0$

Africa Business School

Tips 2 : You can use binary variable to activate or desactivate constraint

Activate/desactivate a constraint $x \leq V$

We want that :

- **$b = 1$ implies $x \leq V$: Activation**
- **$b = 0$ then possibly $x > V$: Desactivation**

Activate/desactivate a constraint $x \leq V$

The constraint for that is :

$$x \leq V + M\,(1 - b)$$

If $b = 1$: the constraint becomes :

$$x \leq V$$

→ it is activated

If $b = 0$: the constraint becomes:

$$x \leq V+M$$

→ it is desactivated (*for M sufficiently big)*

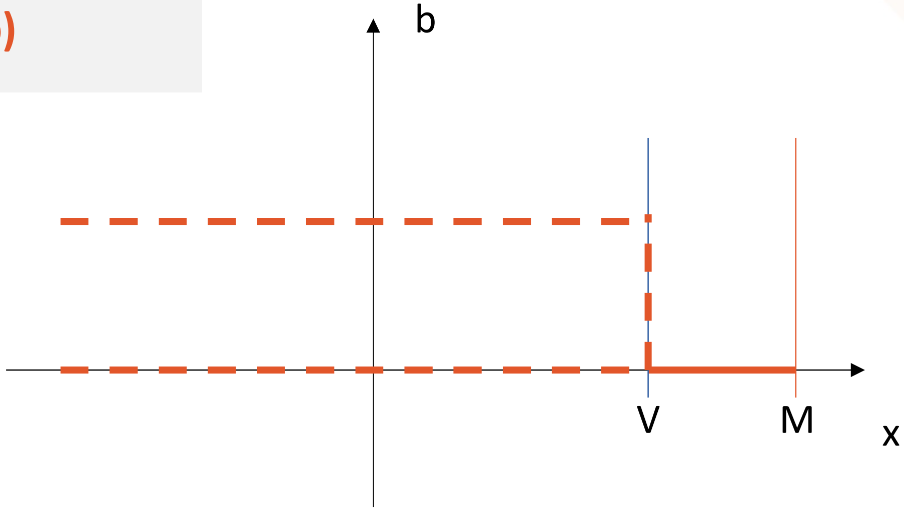
Africa Business School

Tips 2 : Another interpretation

Activate/desactivate a constraint $x \leq V$

The constraint for that is :

$$x \leq V + M\,(1 - b)$$



$x > V$ implies $b = 0$

Africa Business School

Tips 3 : the logical OR

Principle

We want $z = 1$ iff $x = 1$ OR $y = 1$ ***(with x,y,z binary)***

x	y	z
1	1	1
1	0	1
0	1	1
0	0	0

$z = x + y$

Add three constraints :

- $z \geq x$
- $z \geq y$
- $z \leq x + y$

Illustration

- *If the capacity A is binding or the capacity B is binding, then I can use capacity C*

- *isAllowedC ≥ isBindingA*
- *isAllowedC ≥ isBindingB*
- *isAllowedC ≤ isBindingA + isBindingB*
- *productionC ≤ M. IsAllowedC*

Tips 4 : the logical AND

Principle

We want $z = 1$ iff ($x = 1$ AND $y = 1$) ***(with x,y,z binary)***

x	y	z
1	1	1
1	0	0
0	1	0
0	0	0

$z = x * y$

Add three constraints :

- $x \geq z$
- $y \geq z$
- $z \geq x + y - 1$

Illustration

- *If the capacityA is binding or the capacity B is binding, then I can use capacity C*

- *isAllowedC ≤ isBindingA*
- *isAllowedC ≤ isBindingB*
- *isAllowedC ≥ isBindingA + isBindingB - 1*
- *productionC ≤ M. IsAllowedC*

Tip4 : Another interpretation : how to linearize a product a binary variables

Linearization of a product of binary variables

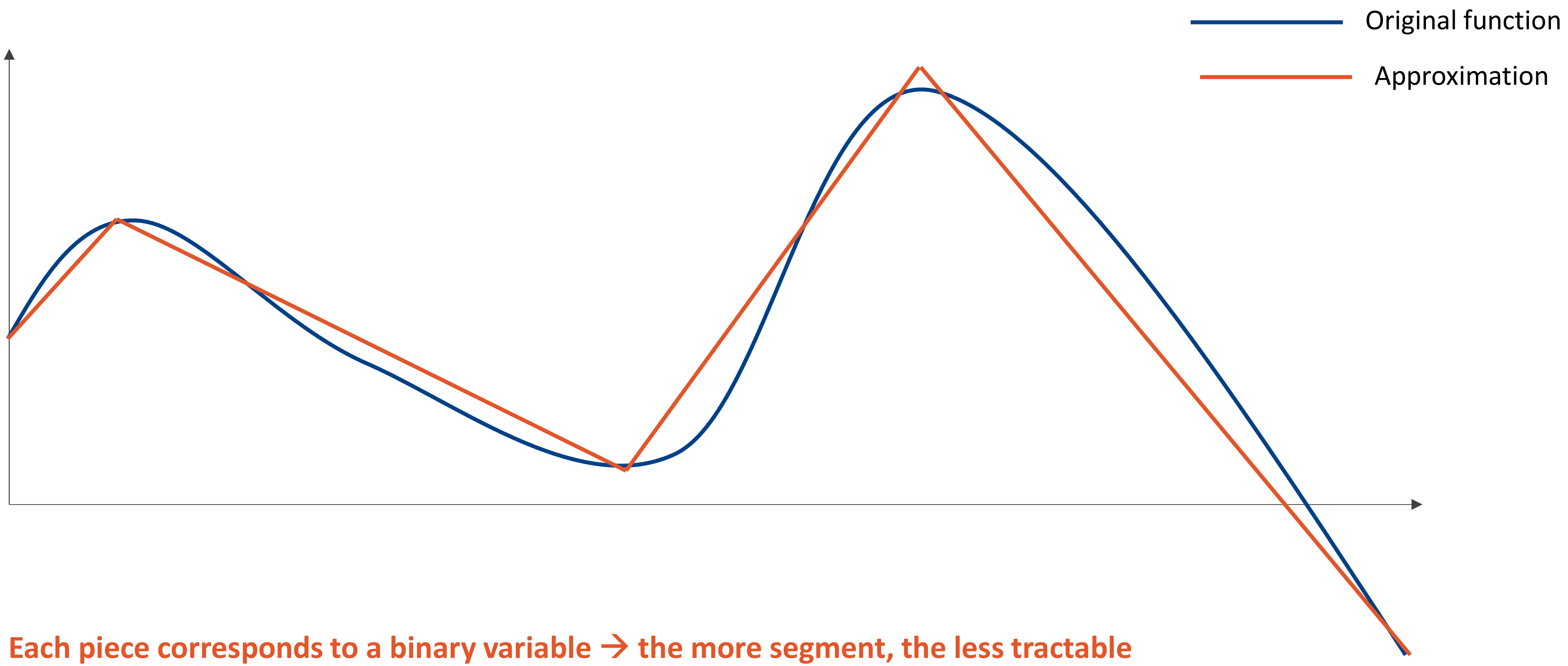
Let x,y,z be 3 binary variables.

We want z such that $z = xy$

It suffices to implement 3 constraints :

- $x \geq z$
- $y \geq z$
- $z \geq x + y - 1$

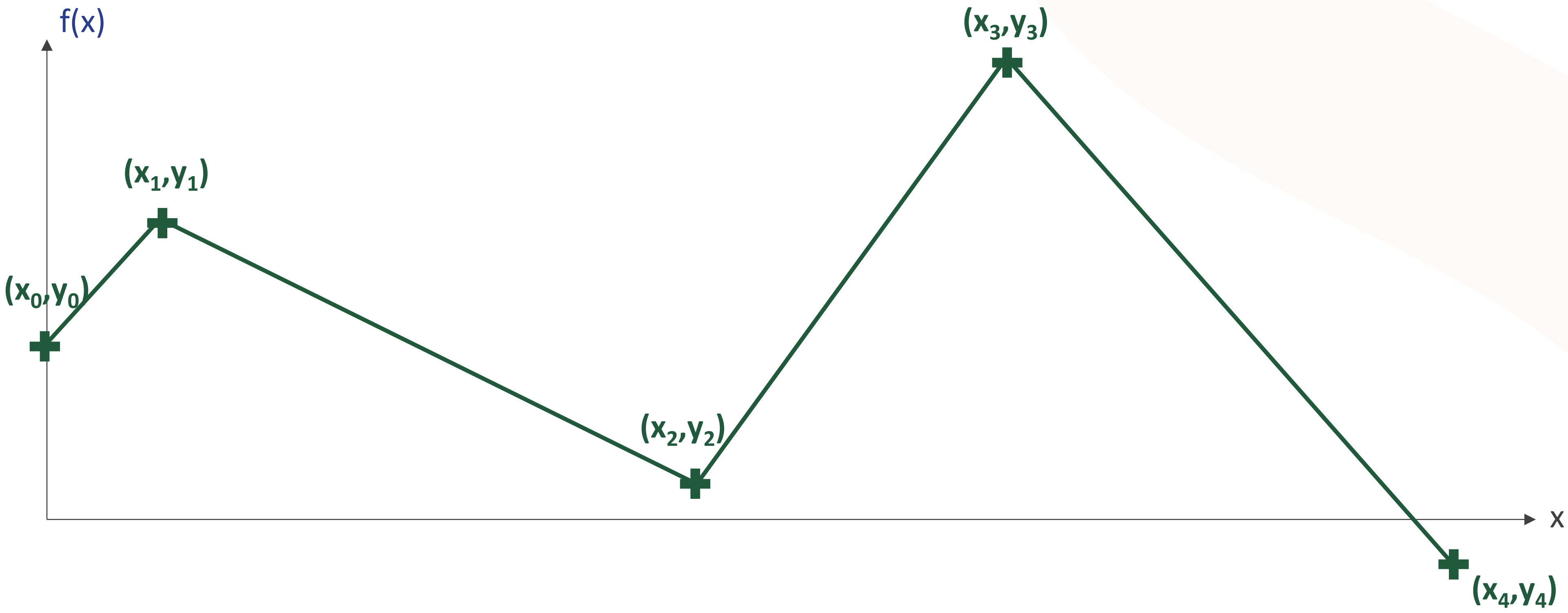
Approximating a nonlinear function by a piecewise linear function



Each piece corresponds to a binary variable → the more segment, the less tractable

Tips 5 : Use binary variables to formulate arbitrary piecewise linear functions

The function f is specified by ordered pairs (x_i, y_i)



Let's practice. How to compute f(x) using binary variables ?

Data : n pieces
$(\boldsymbol{x_i}, \boldsymbol{y_i})\quad \forall \boldsymbol{i} = \boldsymbol{0}, \dots, \boldsymbol{n}$

Constraints
• $\boldsymbol{f}(\boldsymbol{x}) =$

Solution. How to model this using binary variables ?

Data : n pieces

$(x_i, y_i) \quad \forall i = 0, \dots, n$

Variables

$z_i \in \{0, 1\} \quad \forall i = 1, \dots, n$

→ *indicate whether* $x_{i-1} \leq x \leq x_i$

$\lambda_i \in [0, 1] \quad \forall i = 0, \dots, n$

→ *indicate the position over* $[x_{i-1}, x_i]$ *or* $[x_i \ x_{i+1}]$

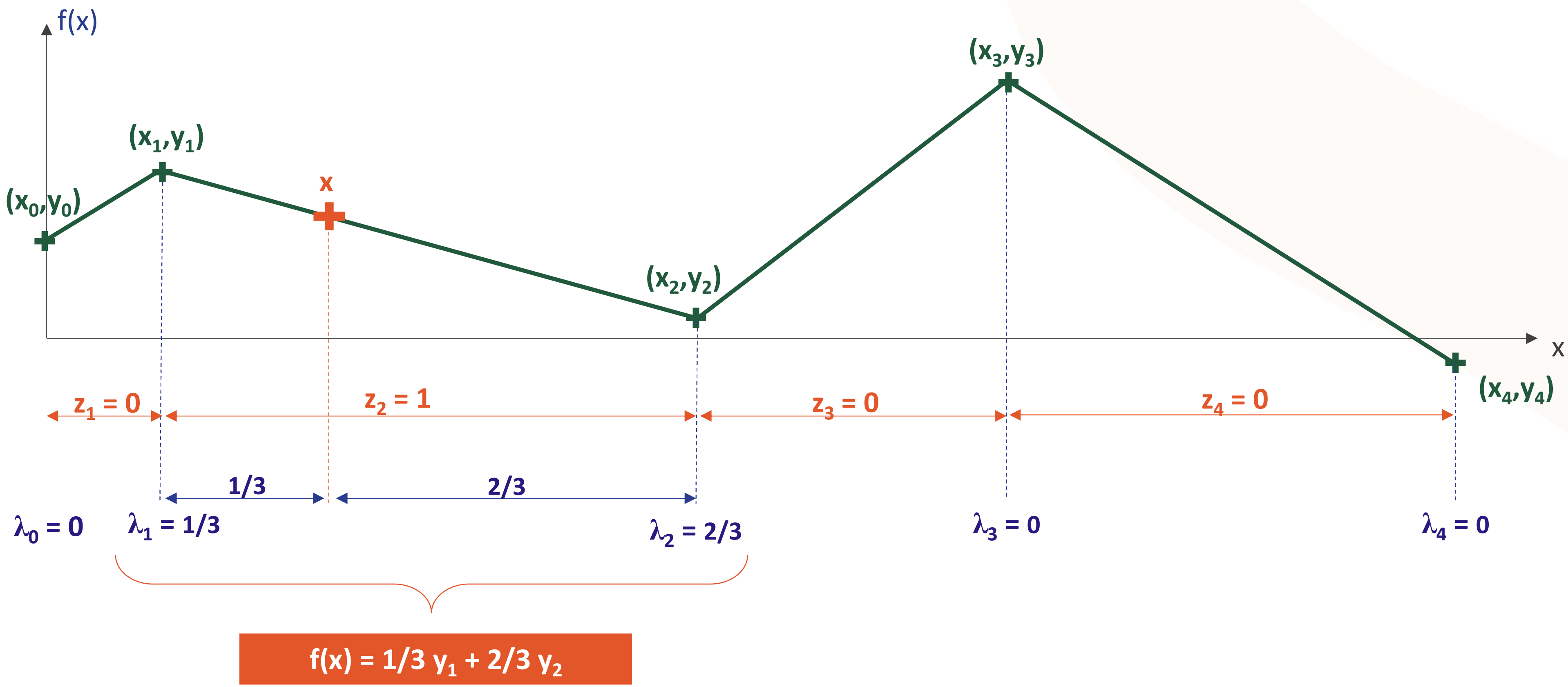
→ *equals 0 if x does not lie in* $[x_{i-1}, x_i]$ *or* $[x_i \ x_{i+1}]$

Constraints

- $f(x) = \sum_{i=0}^{n} \lambda_i y_i$
- $x = \sum_{i=0}^{n} \lambda_i x_i$
- $\sum_{i=0}^{n} \lambda_i = 1$
- $\lambda_i \leq z_i + z_{i+1}$, $i = 1, \dots, n-1$
- $\lambda_0 \leq z_1$
- $\lambda_n \leq z_n$
- $\sum_{i=1}^{n} z_i = 1$
- $\lambda_i \in [0, 1]$
- $z_i \in \{0, 1\}$

Africa Business School

Illustration



Blending problem

The blending problem : new constraint

You have to limit the number of raw materials used to 5 because you have only 5 units to store them

The blending problem : max number of raw materials

You have to limit the number of raw materials used to 5 because you have only 5 units to store them

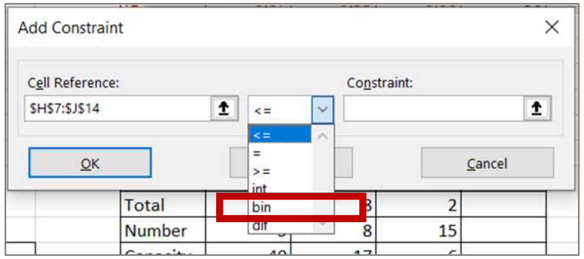
Recipe	Senior cat food	Junior cat food	High protein cat	RM_Volume	Capacity	IsUsed	IsUsed * Capacity
Chicken	75	0	8	83	1000	0	0
Beef	0	769	0	769	1000	0	0
Mutton	0	0	600	600	600	0	0
Rice	0	0	0	0	500	0	0
Wheat	0	500	0	500	500	0	0
Gel	800	0	0	800	800	0	0
Barley	125	231	392	748	1000	0	0
FG_Volume	1000	1500	1000				
Demande	1000	1500	1000		Total used	0	
Protein	1,68	1,98	1,06				
MinProtein	1,44	1,98	0,00				
MaxProtein	1,77	2,40	2,00				
Vitamin	3,42	2,46	2,34				
MinVitamin	3,42	2,34	2,34				
MaxVitamin	4,32	3,12	3,12				

The blending problem : max number of raw materials

Recipe	Senior cat food	Junior cat food	High protein cat	RM_Volume	Capacity	IsUsed	Capacity * IsUsed
Chicken	75	0	506	582	1000	1	1000
Beef	0	769	0	769	1000	1	1000
Mutton	0	0	0	0	600	0	0
Rice	0	0	0	0	500	0	0
Wheat	0	500	0	500	500	1	500
Gel	800	0	0	800	800	1	800
Barley	125	231	494	849	1000	1	1000
FG_Volume	1000	1500	1000				
Demande	1000	1500	1000		Total used	5	
Protein	1,68	1,98	0,84				
MinProtein	1,44	1,98	0,00				
MaxProtein	1,77	2,40	2,00				
Vitamin	3,42	2,46	2,34				
MinVitamin	3,42	2,34	2,34				
MaxVitamin	4,32	3,12	3,12				

Limit the number of raw materials used to 5

1. Define a binary variable stating whether or not you use this raw material : IsUsed = 1 if RM_Volume > 0
    - Define IsUsed as an binay variable
    - Add the BigM constraint : RM_Volume $\leq$ BigM * IsUsed, with BigM = Capacity
2. Add a constraint to ensure that $\sum$ IsUsed $\leq$ 5



Application to the blending problem : min volume per supplier

Suppliers refuse to serve you a volume lesser than 100

Recipe	Senior cat food	Junior cat food	High protein cat	RM_Volume	Capacity	IsUsed	Capacity * IsUsed	Vmin * IsUsed
Chicken	75	0	8	83	1000	0	0	0
Beef	0	769	0	769	1000	0	0	0
Mutton	0	0	600	600	600	0	0	0
Rice	0	0	0	0	500	0	0	0
Wheat	0	500	0	500	500	0	0	0
Gel	800	0	0	800	800	0	0	0
Barley	125	231	392	748	1000	0	0	0
FG_Volume	1000	1500	1000					
Demande	1000	1500	1000					
Protein	1,68	1,98	1,06					
MinProtein	1,44	1,98	0,00					
MaxProtein	1,77	2,40	2,00					
Vitamin	3,42	2,46	2,34					
MinVitamin	3,42	2,34	2,34					
MaxVitamin	4,32	3,12	3,12					

Application to the blending problem : min volume per supplier + discount

Suppliers refuse to serve you a volume lesser than 100
But 3 of your suppliers apply a discount if you take orders for their two products

Supplier	Raw materials	Cost (/kg)	Capacity (kg)	Protein (/kg)	Vitamin (/kg)
Butcher B	Chicken	2,86	1000	1,65	4,62
	Beef	2,73	1000	2,36	3,60
Butcher C	Mutton	2,28	600	1,74	3,84
Cereals'Best	Rice	2,11	500	1,26	1,74
	Wheat	2,40	500	2,31	1,84
GreatFood	Gel	2,31	800	1,95	3,84
	Barley	2,51	1000	2,15	3,27

Supplier	Discount
Butcher B	200
Cereals'Best	400
GreatFood	300

In order to consider the discount in the objective function, you need a binary variables that states whether you take the two products, i.e., if you take product 1 AND product 2

Application to the blending problem : min volume per supplier + discount

Supplier	Raw materials
Butcher B	Chicken
	Beef
Butcher C	Mutton
Cereals'Best	Rice
	Wheat
GreatFood	Gel
	Barley

Supplier	Discount
Butcher B	200
Cereals'Best	400
GreatFood	300

Recipe	Senior cat food	Junior cat food	High protein cat	RM_Volume	Capacity	IsUsed	Capacity * IsUsed	Vmin * IsUsed
Chicken	75	0	8	83	1000	1	1000	100
Beef	0	769	0	769	1000	1	1000	100
Mutton	0	0	600	600	600	1	600	100
Rice	0	0	0	0	500	0	0	0
Wheat	0	500	0	500	500	1	500	100
Gel	800	0	0	800	800	1	800	100
Barley	377	231	392	1000	1000	1	1000	100
FG_Volume	1252	1500	1000					
Demande	1000	1500	1000					
Protein	1,68	1,98	1,06					
MinProtein	1,44	1,98	0,00					
MaxProtein	1,77	2,40	2,00					
Vitamin	3,42	2,46	2,34					
MinVitamin	3,42	2,34	2,34					
MaxVitamin	4,32	3,12	3,12					

	AreBothUsed
Butcher B	0
Cereals'Best	0
GreatFood	0

Quiz: what happens if we have the discount but we remove the volume min constraint ?