

Optimización

BIG DATA CON PYTHON

Biuse Casaponsa

Investigadora Posdoctoral IFCA

INTRODUCTION TO
PYTHON FOR BIG DATA



Outline

1. Introducción a la optimización

2. Conceptos básicos

- Variables, Función objetivo, Ligaduras

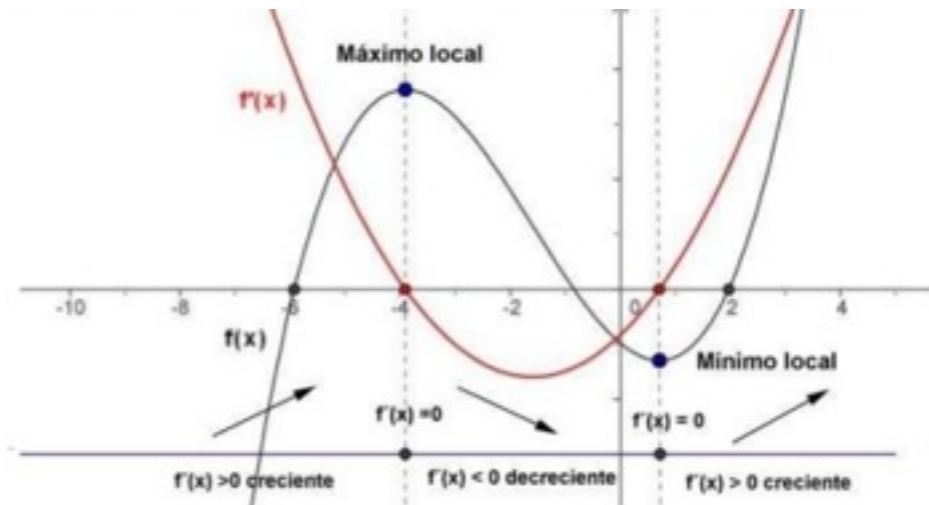
3. Resolver problemas

- Solución analítica, grid, algoritmos

4. Tipos de problemas

- Optimización lineal
- Optimización no lineal

--- Ejemplos python



5. Inferencia estadística

- interpolación y extrapolación
- ajuste mínimos cuadrados

--- Ejemplos python

6. Lab

Introducción: Problemas Optimización

- Las técnicas de machine learning son de gran utilidad para el análisis de muchos datos pero:
 - Hay otras herramientas. Dependiendo del problema ML no tiene porque ser lo más adecuado.
 - Para entender lo que hace una red neuronal hay que saber de optimización y hacer ajustes.
 - Cada vez recogemos y almacenamos más datos pero “*solo unos pocos % se analizan correctamente*” (*Cunef*)
 - Ej. “las empresas *data driven* tienen 23 veces más posibilidades de conseguir nuevos clientes y 6 veces menos de perderlos” (*Mckinsey Business Technology Office*)



Introducción: Problemas Optimización

Optimización: Escoger el mejor elemento (dadas unas condiciones) de entre varias alternativas disponibles.

Introducción: Problemas Optimización

Optimización: Escoger el mejor elemento (dadas unas condiciones) de entre varias alternativas disponibles.

- Mantener stock óptimo, de tal manera que pueda satisfacer la demanda y que me quepa en el almacén



Introducción: Problemas Optimización

Optimización: Escoger el mejor elemento (dadas unas condiciones) de entre varias alternativas disponibles.

- Mantener stock óptimo, de tal manera que pueda satisfacer la demanda y que me quepa en el almacén



- Escoger ruta óptima de reparto o de ventas. (Travelling Salesman Problem)



Introducción: Problemas Optimización

Optimización: Escoger el mejor elemento (dadas unas condiciones) de entre varias alternativas disponibles.

- **Medio Ambiente.** Minimizar impacto ecológico de alguna medida dadas una gran cantidad de variables. Maximizar eficiencia energética con menor contaminación,...

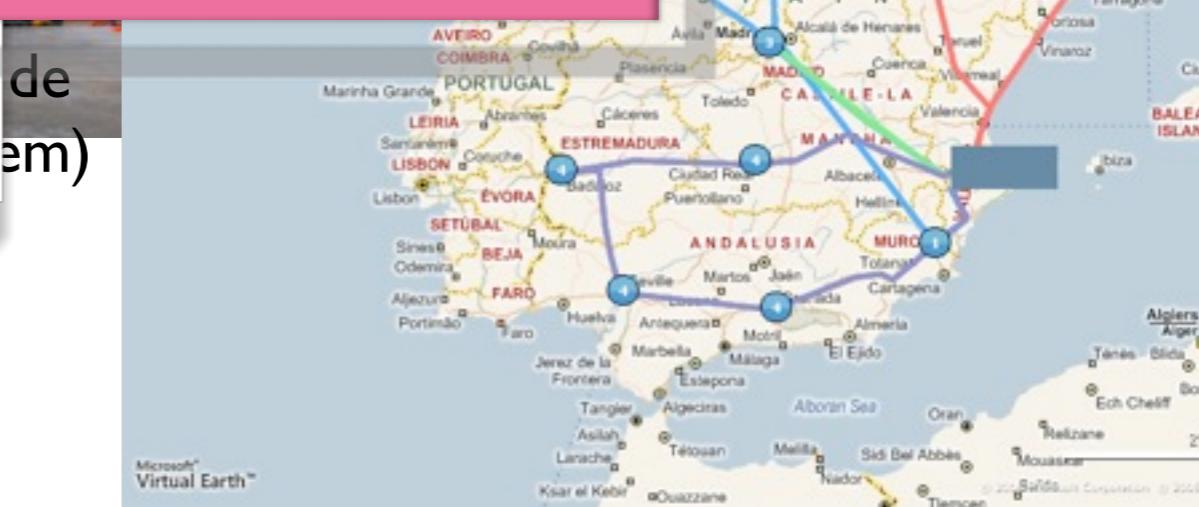


Introducción: Problemas Optimización

Optimización: Escoger el mejor elemento (dadas unas condiciones) de entre varias alternativas disponibles.

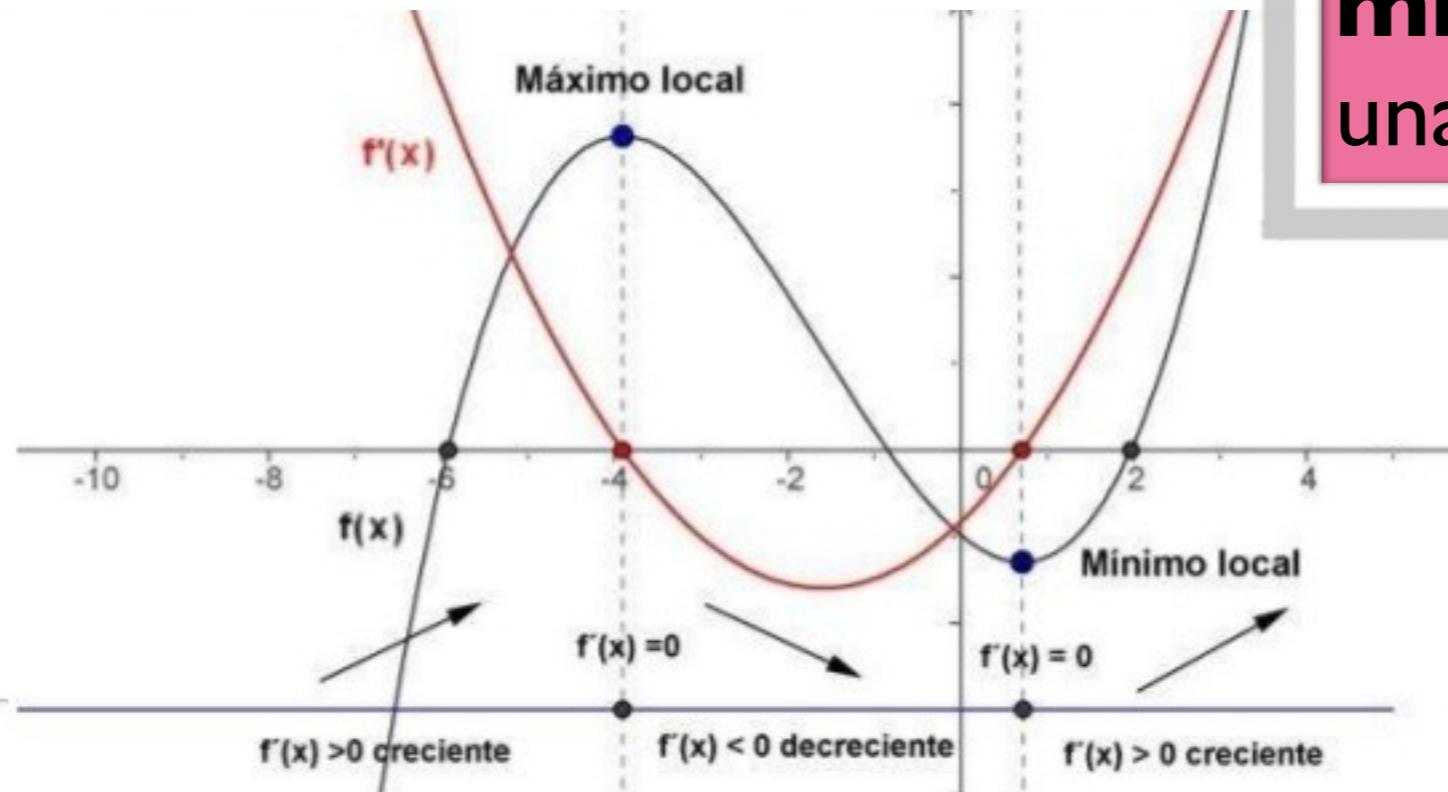
- **Medio Ambiente.** Minimizar impacto ecológico grandeza eficiencia consumo

Matemáticamente estos problemas requieren **minimizar o maximizar** una función.



Introducción: Problemas Optimización

Matemáticamente estos problemas requieren **minimizar o maximizar** una función.



Ejercicios colegio: lo resolvíamos de forma analítica:

$$\frac{dF}{dx} = 0$$

$$\frac{d^2F}{dx^2} > 0$$

$$\frac{d^2F}{dx^2} = 0$$

$$\frac{d^2F}{dx^2} < 0$$

PASOS BÁSICOS OPTIMIZACIÓN

- Pensar problema y definirlo matemáticamente
- Definir **variables**.
- Definir **función objetivo**: qué vamos a minimizar o maximizar con respecto a esas variables.
- Definir **ligaduras** entre las variables (si las hay)
- Escoger **algoritmo** para encontrar el valor de los parámetros que hacen mínima (o maxima) la función objetivo.

PASOS BÁSICOS OPTIMIZACIÓN

- Pensar problema y definirlo matemáticamente
 - Definir **variables**.
 - Definir **función objetivo**: qué vamos a minimizar o maximizar con respecto a esas variables.
 - Definir **ligaduras** entre las variables (si las hay)
-
- Escoger **algoritmo** para encontrar el valor de los parámetros que hacen mínima (o maxima) la función objetivo.

PROBLEMA OPTIMIZACIÓN. Ejemplo



Diseñar menú
lo más barato
y saludable posible

PROBLEMA OPTIMIZACIÓN. Ejemplo



Diseñar menú
lo más barato
y saludable posible

A	B	C	D	E	F	G	H	I	J
Foods	Price/Serving	Serving Size	Calories	Cholesterol (mg)	Total_Fat (g)	Sodium (mg)	Carbohydrates (g)	Dietary_Fiber (g)	Protein (g)
Frozen Broccoli	\$0.48	10 Oz Pkg	73.8	0	0.8	68.2	13.6	8.5	8
Frozen Corn	\$0.54	1/2 Cup	72.2	0	0.6	2.5	17.1	2	2.5
Raw Lettuce Iceberg	\$0.06	1 Leaf	2.6	0	0	1.8	0.4	0.3	0.2
Baked Potatoes	\$0.18	1/2 Cup	171.5	0	0.2	15.2	39.9	3.2	3.7
Tofu	\$0.93	1/4 block	88.2	0	5.5	8.1	2.2	1.4	9.4
Roasted Chicken	\$2.52	1 lb chicken	277.4	129.9	10.8	125.6	0	0	42.2
Spaghetti W/ Sauce	\$2.34	1 1/2 Cup	358.2	0	12.3	1237.1	58.3	11.6	8.2
Raw Apple	\$0.72	1 Fruit,3/Lb,Wo/Rf	81.4	0	0.5	0	21	3.7	0.3
Banana	\$0.45	1 Fruit,Wo/Skn&Seeds	104.9	0	0.5	1.1	26.7	2.7	1.2
Wheat Bread	\$0.15	1 Sl	65	0	1	134.5	12.4	1.3	2.2
White Bread	\$0.18	1 Sl	65	0	1	132.5	11.8	1.1	2.3
Oatmeal Cookies	\$0.27	1 Cookie	81	0	3.3	68.9	12.4	0.6	1.1
Apple Pie	\$0.48	1 Oz	67.2	0	3.1	75.4	9.6	0.5	0.5
Scrambled Eggs	\$0.33	1 Egg	99.6	211.2	7.3	168	1.3	0	6.7
Turkey Bologna	\$0.45	1 Oz	56.4	28.1	4.3	248.9	0.3	0	3.9
Beef Frankfurter	\$0.81	1 Frankfurter	141.8	27.4	12.8	461.7	0.8	0	5.4
Chocolate Chip Cookies	\$0.09	1 Cookie	78.1	5.1	4.5	57.8	9.3	0	0.9

PASOS BÁSICOS OPTIMIZACIÓN.

Variables

- Es lo que queremos encontrar, lo que tiene cierta variación y fijaremos al optimizar el problema.
- Es continua o discreta, entera, positiva...
- CASO DIETA:

PASOS BÁSICOS OPTIMIZACIÓN.

Variables

- Es lo que queremos encontrar, lo que tiene cierta variación y fijaremos al optimizar el problema.
- Es continua o discreta, entera, positiva...
- CASO DIETA:

x_1, x_2, x_3, \dots = Cantidad de cada elemento
(ej. gramos)

PASOS BÁSICOS OPTIMIZACIÓN.

Función objetivo

- Función que queremos minimizar o maximizar. Escribir matemáticamente
 - puede ser continua o discreta
 - lineal, cuadrática, más compleja,..
 - CASO DIETA --> queríamos la dieta más barata posible

PASOS BÁSICOS OPTIMIZACIÓN.

Función objetivo

- Función que queremos minimizar o maximizar. Escribir matemáticamente
 - puede ser continua o discreta
 - lineal, cuadrática, más compleja,..
 - CASO DIETA --> queríamos la dieta más barata posible

Minimzamos precio total:

$$P = x_1 p_1 + x_2 p_2 + x_3 p_3 + \dots$$

PASOS BÁSICOS OPTIMIZACIÓN.

Ligaduras (Constraints)

- Algo característico de nuestro problema, límites que no se pueden (o no se quieren) superar
 - Puede haber más de una
 - lineales o no lineales
 - Pueden ser más rígidas ($a+b=c$) o menos ($a+b \leq c$)
 - CASO DIETA: queríamos dieta saludable: definimos lo que es saludable

PASOS BÁSICOS OPTIMIZACIÓN.

Ligaduras (Constraints)

- Algo característico de nuestro problema, límites que no se pueden (o no se quieren) superar
 - Puede haber más de una
 - lineales o no lineales
 - Pueden ser más rígidas ($a+b=c$) o menos ($a+b \leq c$)
 - CASO DIETA: queríamos dieta saludable: definimos lo que es saludable

Condiciones que tiene el problema:

$$900 \leq x_1 cal_1 + x_2 cal_2 + x_3 cal_3 + \dots \leq 2000$$

$$50 \leq x_1 prot_1 + x_2 prot_2 + x_3 prot_3$$

...

PASOS BÁSICOS OPTIMIZACIÓN.

Problema matemático

$$\text{minimize} \quad \sum_i x_i p_i$$

$$\text{subject to} \quad 900 \leq \sum_i x_i cal_i \leq 2000$$

$$50 \leq \sum_i x_i prot_i$$

$$x_i \geq 0$$

¡Primera parte hecha!

PASOS BÁSICOS OPTIMIZACIÓN

- Pensar problema y definirlo matemáticamente
- Definir **variables**.
- Definir **función objetivo**: qué vamos a minimizar o maximizar con respecto a esas variables.
- Definir **ligaduras** entre las variables (si las hay)
- Escoger **algoritmo** para encontrar el valor de los parámetros que hacen mínima (o maxima) la función objetivo.

PASOS BÁSICOS OPTIMIZACIÓN.

Resolver el problema.

- El último paso es buscar la **herramienta matemática** que me permita resolver el problema. Encontrar valores **óptimos** de las variables.
- Hay distintas maneras de abordarlo dependiendo de las características del problema.
 - Analíticamente
 - Búsqueda directa
 - Algoritmo numérico

PASOS BÁSICOS OPTIMIZACIÓN.

Resolver el problema.

- El último paso es buscar la **herramienta matemática** que me permita resolver el problema. Encontrar valores **óptimos** de las variables.
- Hay distintas maneras de abordarlo dependiendo de las características del problema.
 - Analíticamente
 - Búsqueda directa
 - Algoritmo numérico

o buscamos paquete de python que nos lo resuelva :)!

RESOLVER PROBLEMAS de OPTIMIZACIÓN

Resolver problemas de Optimización

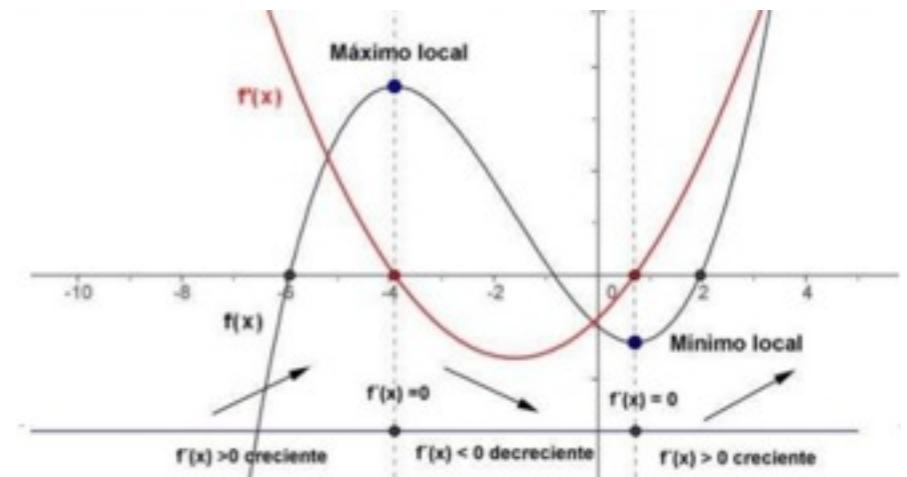
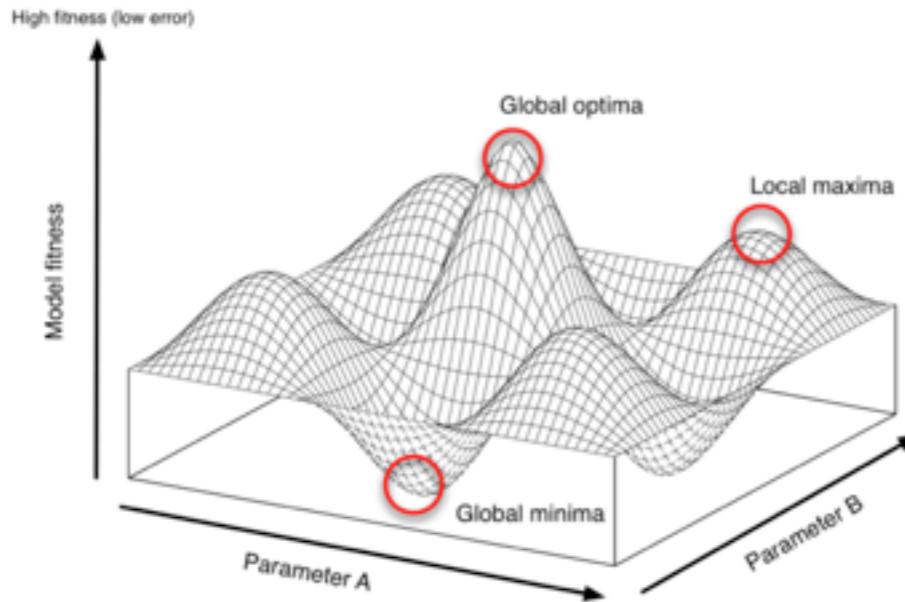
I. Solución analítica.

En una o dos dimensiones podemos verlo gráficamente.

Si es continuo y suave, podemos encontrar el valor de las variables que optimizan el problema

Encontrar derivada de la función objetivo e igualar a 0.

$$\frac{dF(x)}{dx} = 0$$



Resolver problemas de Optimización

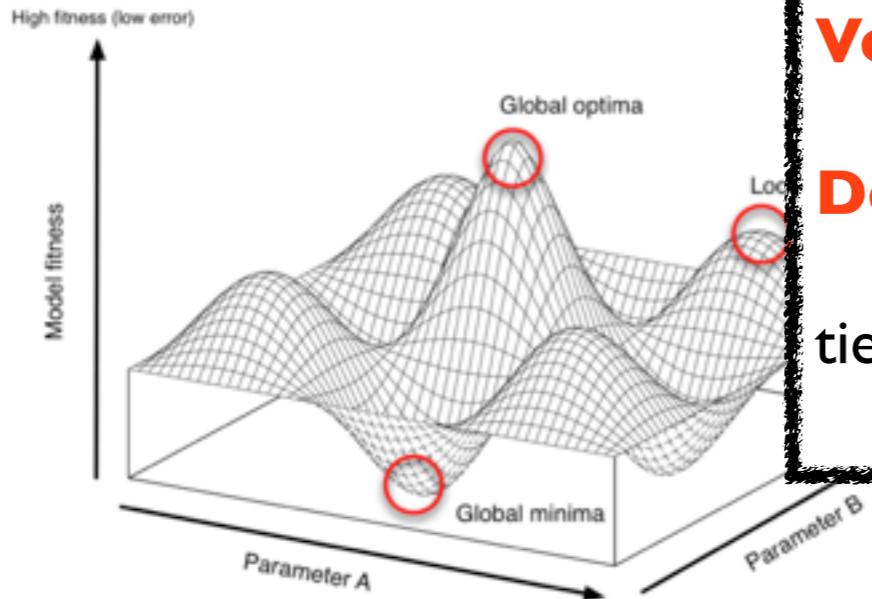
I. Solución analítica.

En una o dos dimensiones podemos verlo gráficamente.

Si es continuo y suave, podemos encontrar el valor de las variables que optimizan el problema

Encontrar derivada de la función objetivo e igualar a 0.

$$\frac{dF(x)}{dx} = 0$$



Ventajas: Es muy rápido, solución exacta

Desventajas: Hay que derivar a mano
No siempre se puede hacer (la función tiene que ser derivable)
Cuando hay muchas variables no es trivial

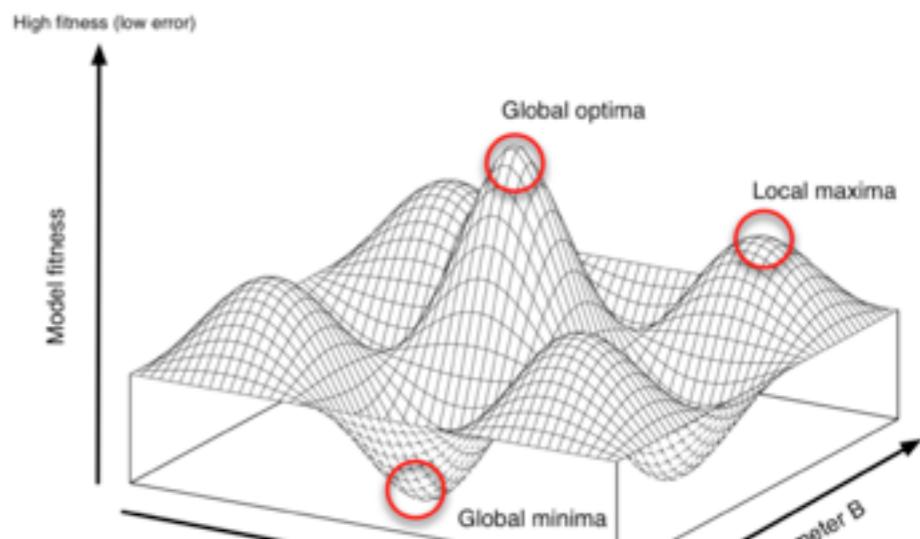
Resolver problemas de Optimización

2. GRID (malla)

2. **Grid.** Podemos calcular todos los posibles resultados, y coger el que de resultado mínimo.

x	y	$F(x,y)$
-3	5	34
-2	4	20
-1	3	10
0	2	4
1	1	2
2	0	4
3	-1	10

Explorando todos los valores de las variables (en un cierto rango y un cierto paso) calculamos la función objetivo



Resolver problemas de Optimización

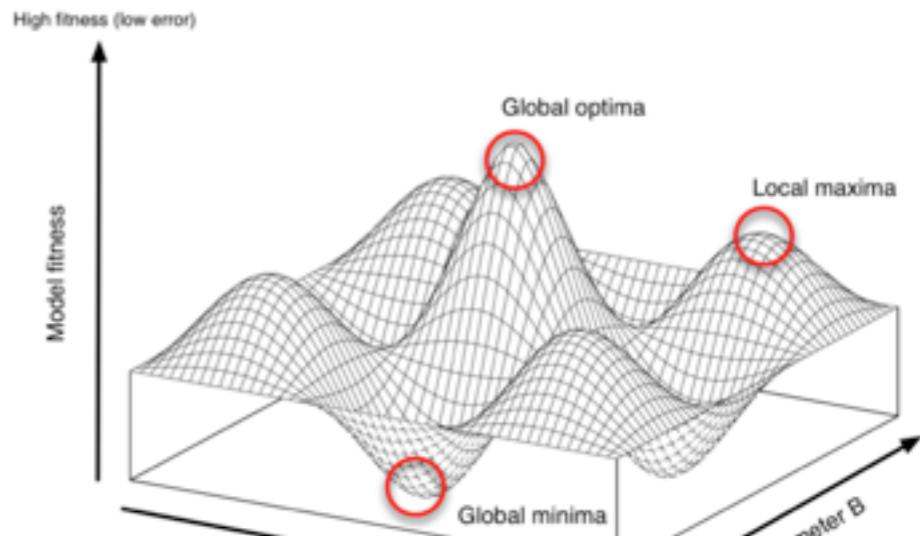
2. GRID (malla)

2. Grid. Podemos calcular todos los posibles resultados, y coger el que de resultado mínimo.

x	y	F(x,y)
-3	5	34
-2	4	20
-1	3	10
0	2	4
1	1	2
2	0	4
3	-1	10

Explorando todos los valores de las variables (en un cierto rango y un cierto paso) calculamos la función objetivo

Buscamos donde la función es mínima, y tenemos el valor de las variables (x e y) en ese punto.



Resolver problemas de Optimización

2. GRID (malla)

2. **Grid.** Podemos calcular todos los posibles resultados, y coger el que de resultado mínimo.

x	y	F(x,y)
-3	5	34
-2	4	20

Explorando todos los valores de las variables (en un cierto rango y un cierto paso) calculamos la función objetivo

-1
0
1
2
3

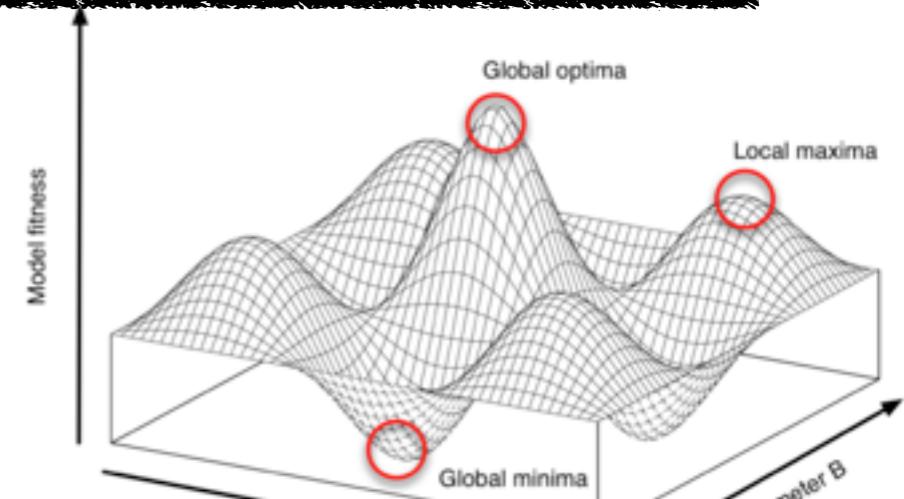
Ventajas:

Siempre se puede hacer
Podemos lidiar con variables discretas

mínima, y
es (x e y)

Desventajas:

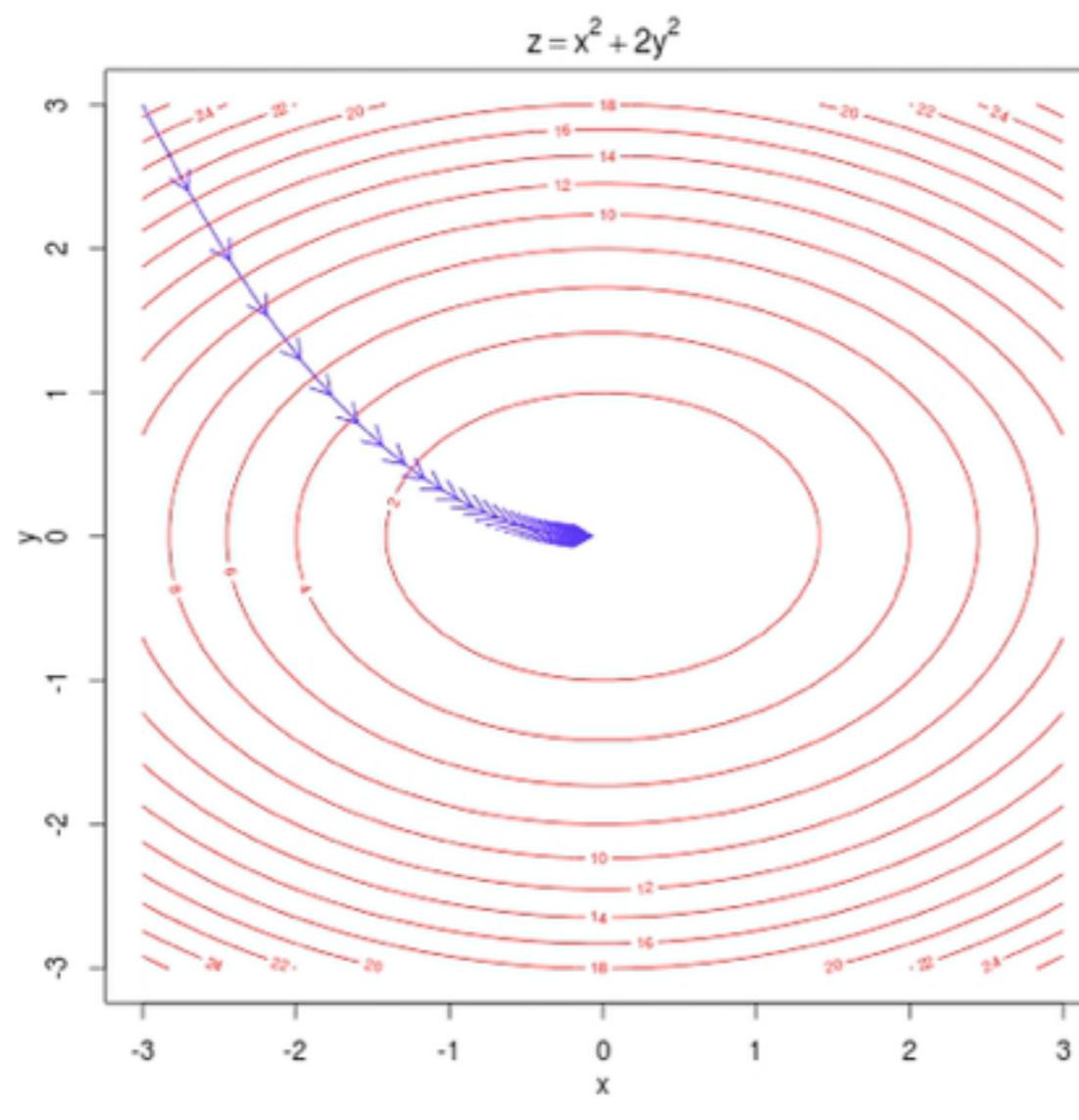
Puede ser muy costoso o imposible
(infinitas combinaciones)



Resolver problemas de Optimización

3. Algoritmo numérico

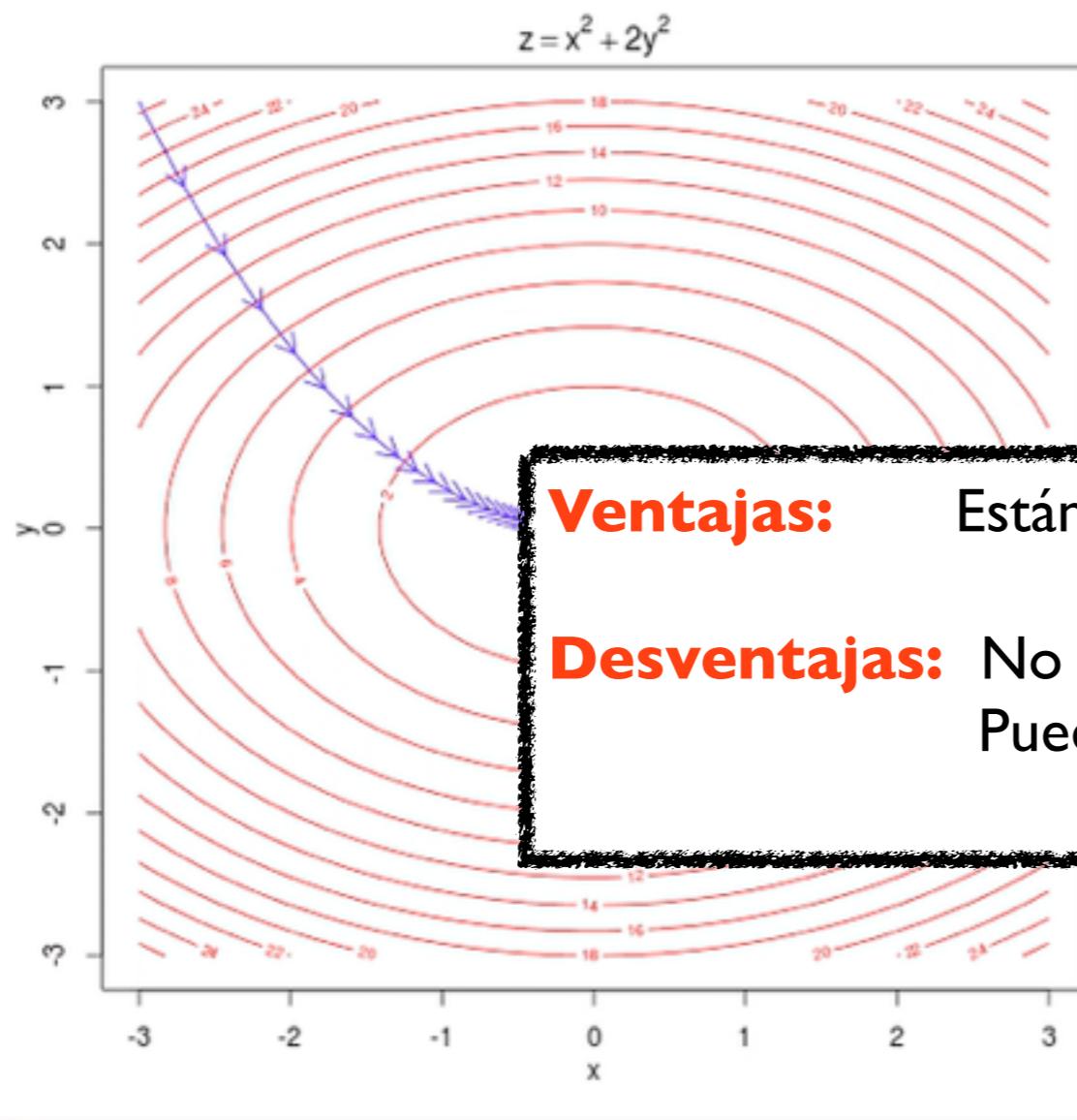
Buscamos el mínimo de la función con un algoritmo iterativo que me lleve al mínimo



Resolver problemas de Optimización

3. Algoritmo numérico

Buscamos el mínimo de la función con un algoritmo iterativo que me lleve al mínimo



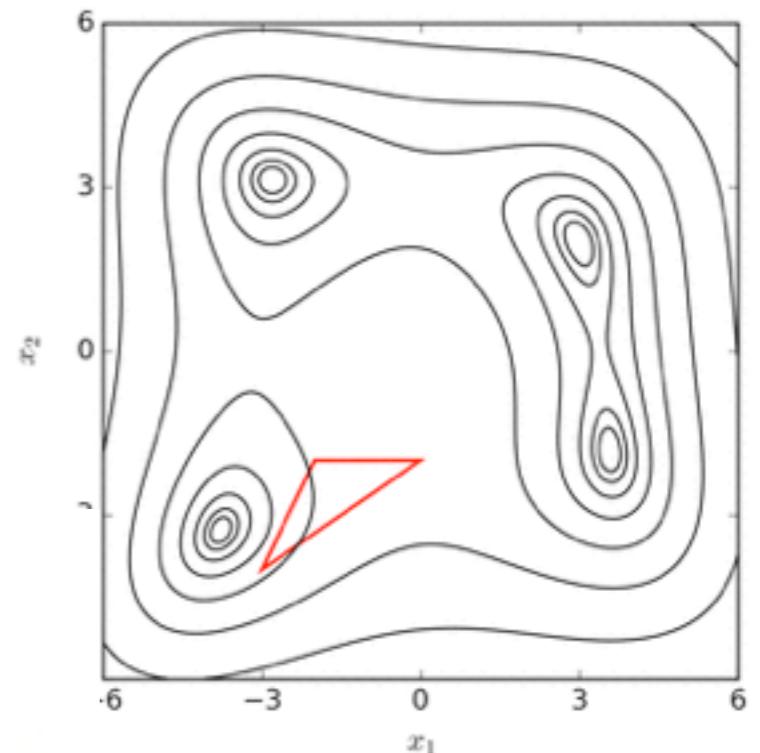
Ventajas: Están implementados en python
Desventajas: No todos son igual de eficientes
Pueden ser una “Caja negra”

Resolver problemas de Optimización

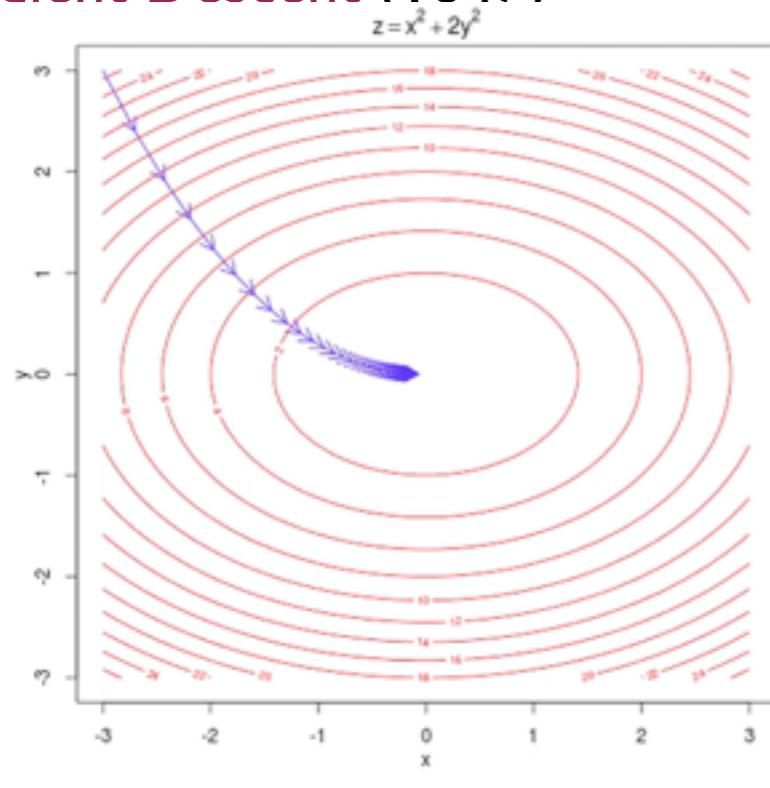
3. Algoritmo numérico

- Necesitan un **punto inicial** donde empezar la búsqueda
- Y buscan el mínimo de la función

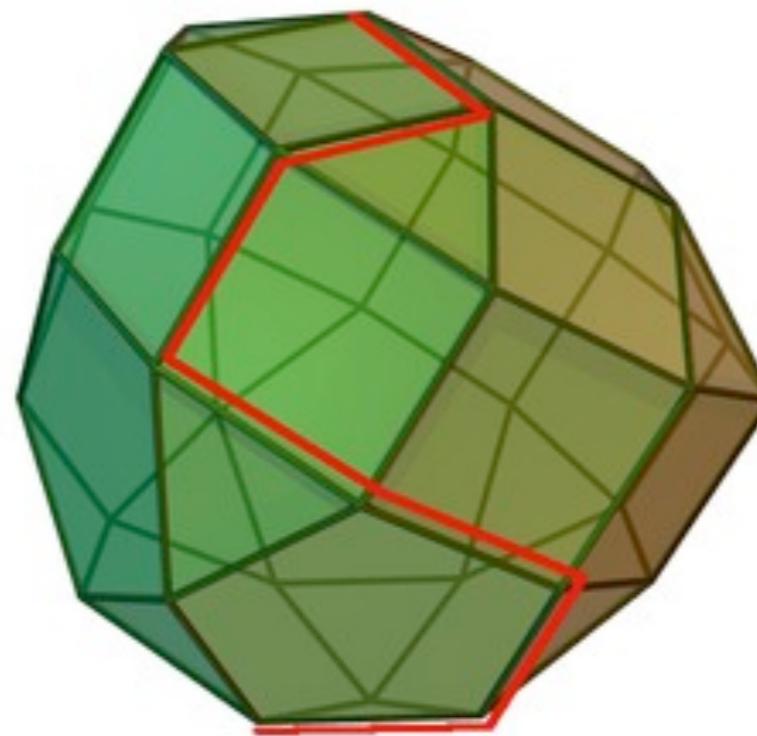
Nelder-Mead o downhill method (1965)



Gradient Descent (1847)



Simplex (1947)



Resolver problemas de Optimización

3. Algoritmo numérico. Tipos de algoritmos

USAN GRADIENTE (y/o segundas derivadas)	NO USAN GRADIENTE	FANCY
<ul style="list-style-type: none">- Métodos que necesitan tener derivadas definidas- Son los más usados para funciones continuas derivables para problemas sin ligaduras-Gradient Descent-BFGS, Levenberg-Marquardt, Gauss-Newton, gradiente-conjugado-Solución analítica	<ul style="list-style-type: none">- No necesitan derivada definida- Simplex-Nelder-Mead-Random Walk-Grid-Métodos de Monte-Carlo	<ul style="list-style-type: none">-Métodos más sofisticados o creativos-Ant colony-Bat colony-Brain storming optimization-Útiles para abordar problemas de difícil solución (NP-Hard)

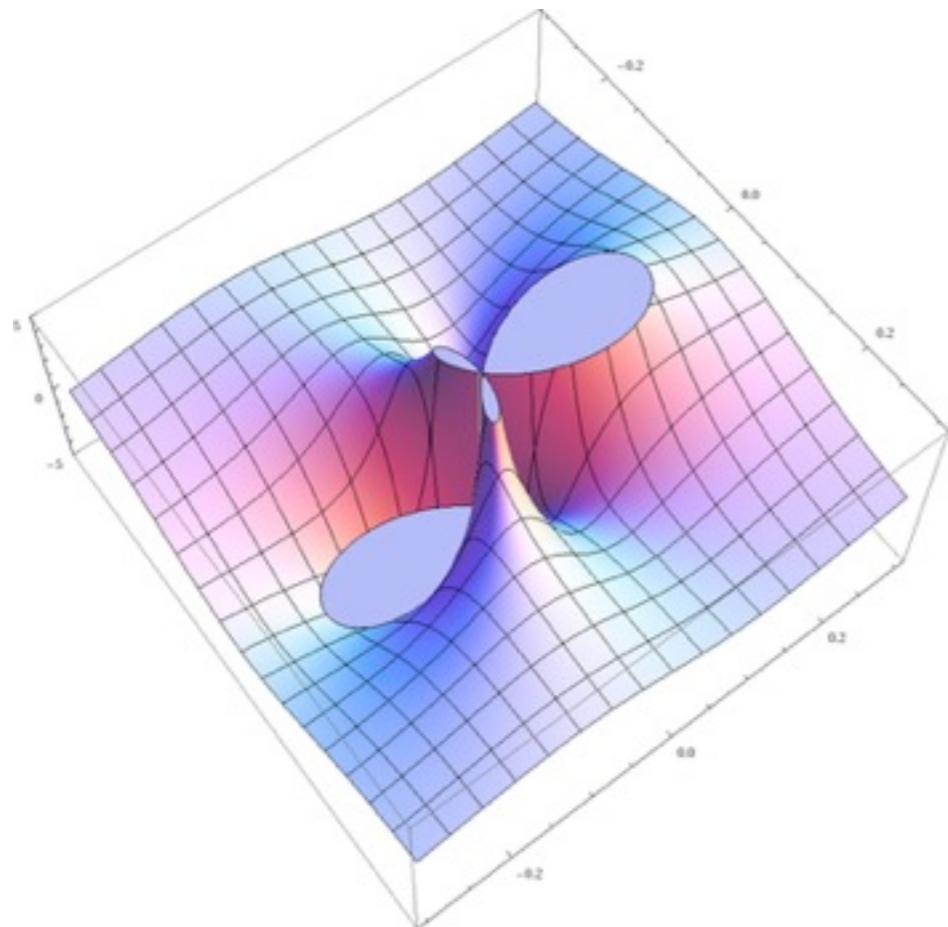
TIPOS de PROBLEMAS

Optimización con ligaduras o restricciones.

PROGRAMACIÓN LINEAL

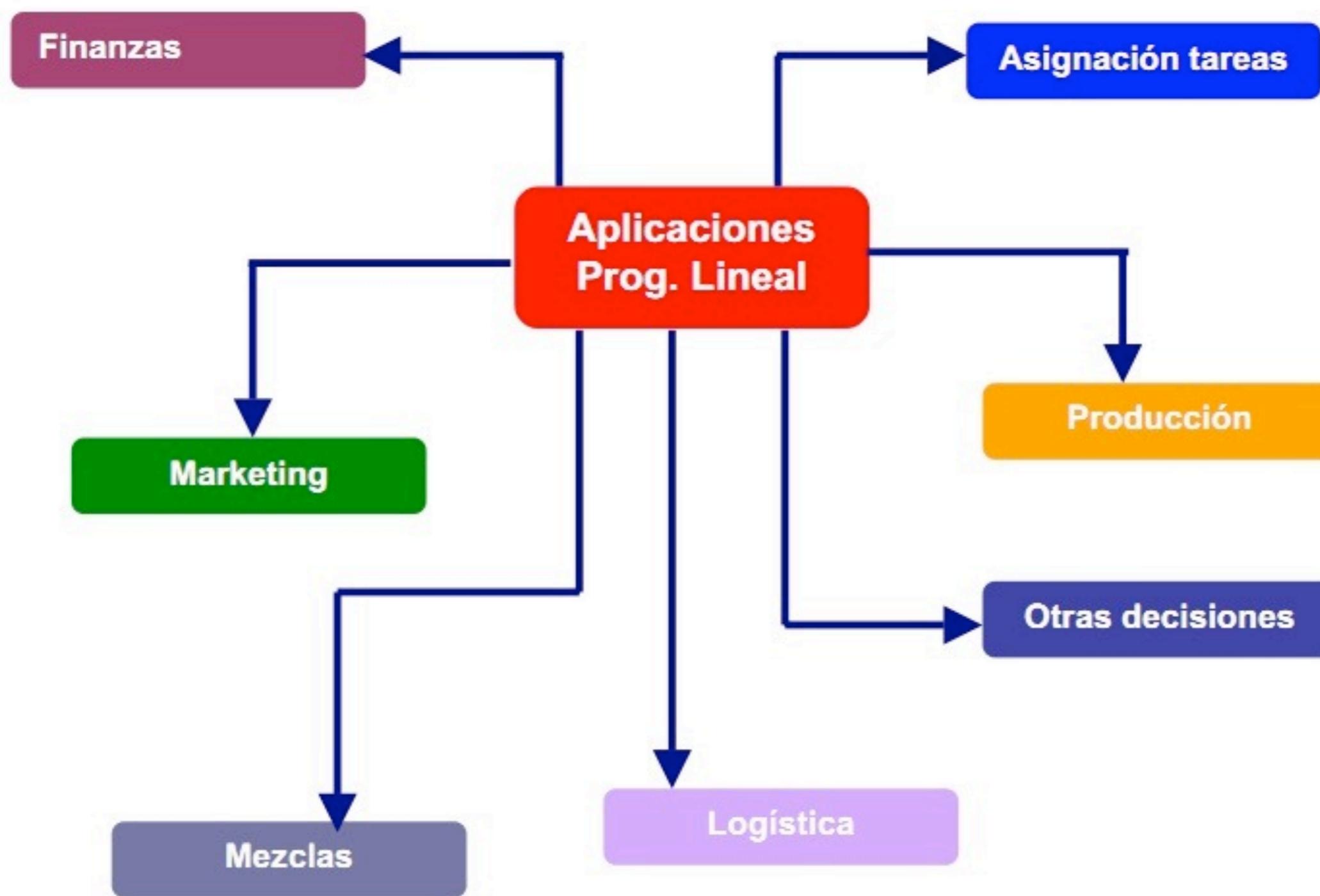
- Función objetivo lineal.
- Ligaduras lineales

$$\begin{aligned} & \text{minimiza} && z = -x_1 - 2x_2 \\ & \text{sujeto a} && -2x_1 + x_2 \leq 2 \\ & && -x_1 + x_2 \leq 3 \\ & && x_1 \leq 3 \\ & && x_1, x_2 \geq 0 \end{aligned}$$



Optimización con ligaduras o restricciones.

PROGRAMACIÓN LINEAL



PROGRAMACIÓN LINEAL

Otro ejemplo

Líneas de producción:

Tenemos 2 líneas de producción y 3 productos. La empresa se ha comprometido a proveer 600 refrescos, 400 zumos y 1000 botellines de agua cada día a una distribuidora.

Línea	Coste hora	Refresco	Zumo	Aqua
1	20	60	30	40
2	6	10	10	60

Objetivo

- Cumplir con los compromisos con el menor coste



PROGRAMACIÓN LINEAL

Matemáticas

Variables:

horas que tengo activa cada línea,

$$h_1, h_2$$

Función objetivo:

$$f = 20h_1 + 6h_2$$

Ligaduras: $60h_1 + 10h_2 > 600$

$$30h_1 + 10h_2 > 400$$

$$40h_1 + 60h_2 > 1000$$

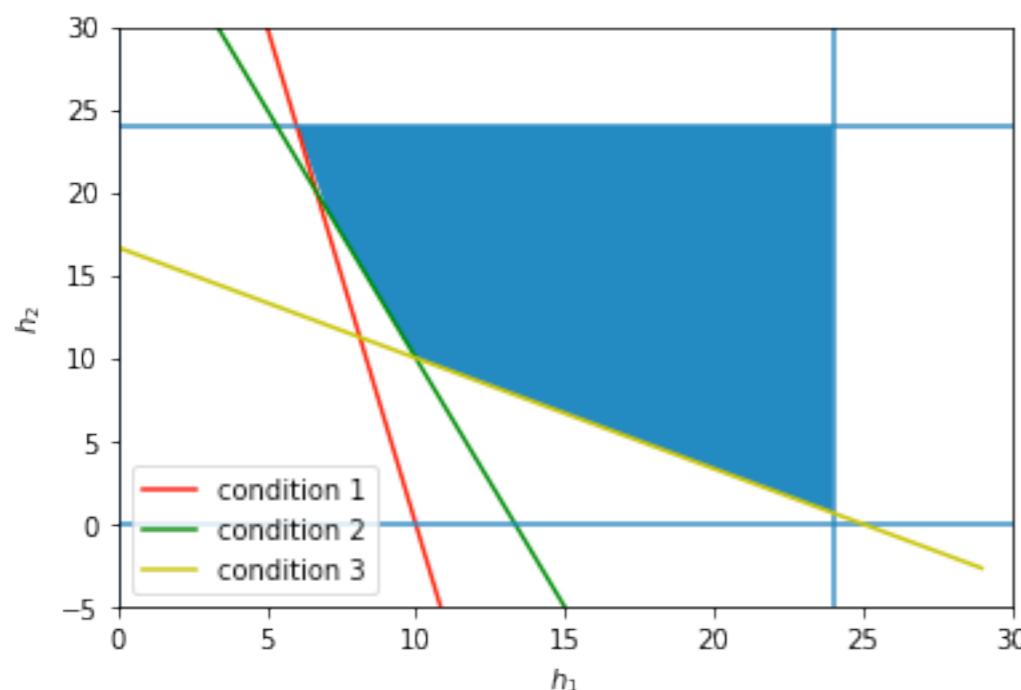
$$h_1, h_2 \leq 24$$

$$h_1, h_2 \geq 0$$



PROGRAMACIÓN LINEAL

Análisis geométrico



- Se hace un gráfico con las ecuaciones
- Se define la zona donde se cumplen todas las condiciones (espacio de soluciones factibles)
- *Fundamental theorem of Linear programming:* La solución óptima se encuentra en uno de los vértices o bordes.

PROGRAMACIÓN LINEAL

con Python

o resolver con python:

```
from scipy.optimize import linprog  
help(linprog) . Ejercicio
```

scipy.optimize.linprog

scipy.optimize.linprog(*c*, *A_ub=None*, *b_ub=None*, *A_eq=None*, *b_eq=None*, *bounds=None*, *method='simplex'*, *callback=None*, *options=None*)

Minimize a linear objective function subject to linear equality and inequality constraints.

Linear Programming is intended to solve the following problem form:

Minimize: $c^T * x$

Subject to: $A_{ub} * x \leq b_{ub}$

$A_{eq} * x == b_{eq}$

PROGRAMACIÓN LINEAL

Otras Aplicaciones



VectorStock®

VectorStock.com/18000951

PROGRAMACIÓN LINEAL

Otras Aplicaciones

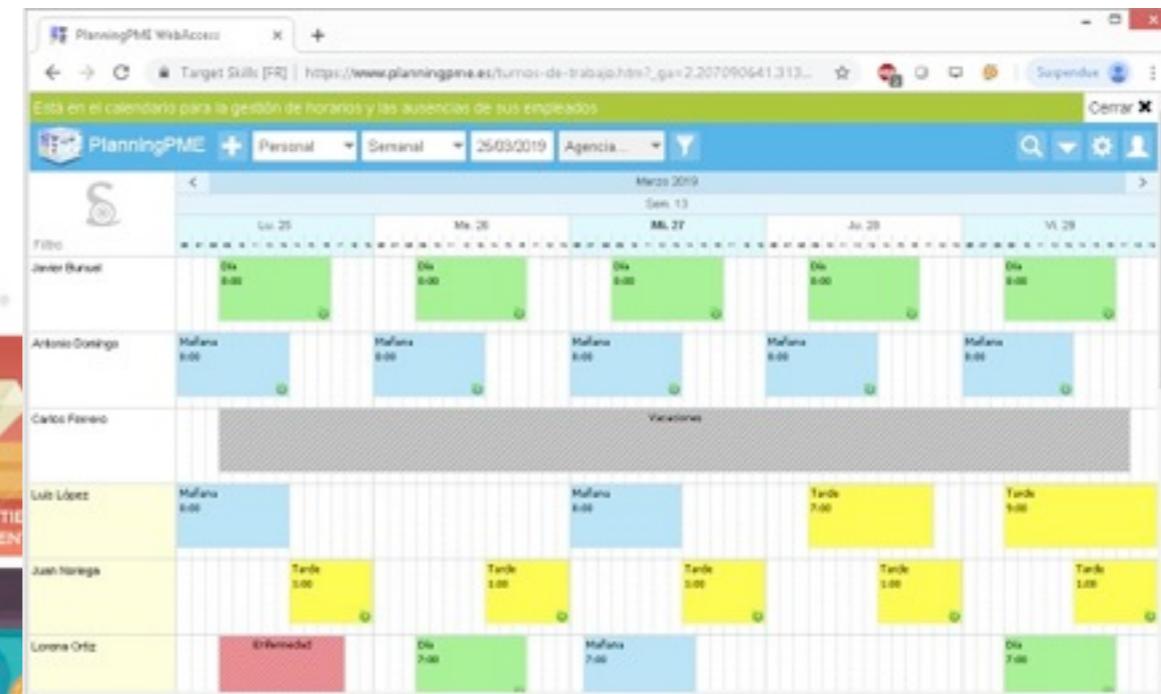
The collage illustrates various applications of linear programming:

- BONDS**: Represented by a document icon.
- MORTGAGE**: Represented by a house icon with a percentage sign.
- EXCHANGE**: Represented by a dollar sign and euro sign with a circular arrow.
- DEPOSIT**: Represented by a safe icon with a checkmark.
- COMMODITIES INVESTMENT**: Represented by a diamond icon.
- STOCK EXCHANGE ACCESS**: Represented by a document icon with a dollar sign.
- CASH BANKING**: Represented by a dollar sign icon.
- CREDIT PRODUCTS**: Represented by a percentage sign and dollar signs.
- BANK**: Represented by a bank building icon.
- SAFE DEPOSIT BOX**: Represented by a stack of boxes with a shield icon.
- PREMIUM CARDS**: Represented by a stack of cards with a diamond icon.
- SECURITY**: Represented by a shield icon with a checkmark and gears.

VectorStock® VectorStock.com/18000951

PROGRAMACIÓN LINEAL

Otras Aplicaciones



5	3		7			
6			1	9	5	
	9	8				6
8			6			3
4		8	3			1
7			2			6
	6			2	8	
		4	1	9		5
			8		7	9

OPTIMIZACIÓN NO-LINEAL

Se presentan problemas parecidos en programación lineal, pero en este caso la función objetivo o las ligaduras no son lineales con respecto a los parámetros que buscamos. Ej.

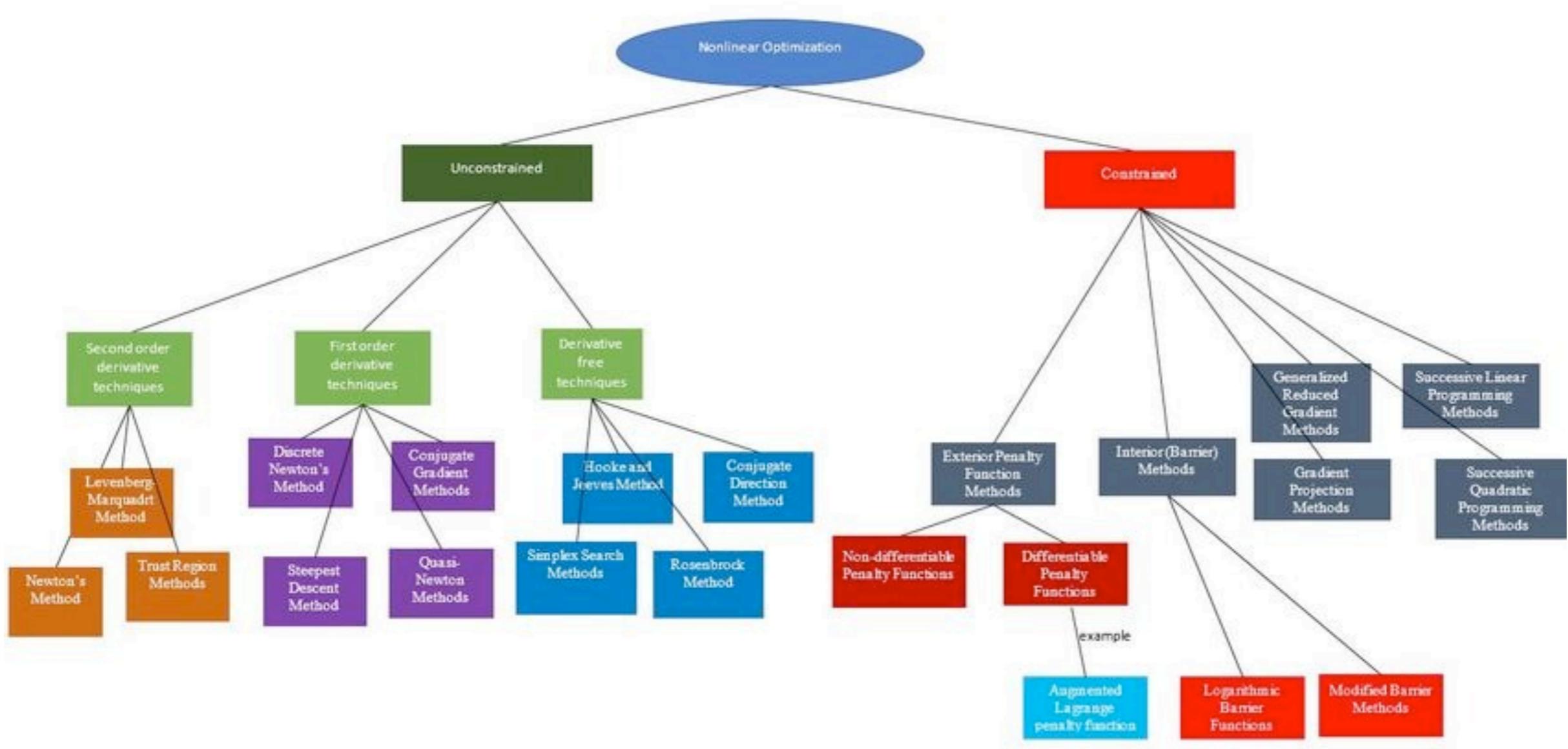
$$f(x) = Ax^\beta + Cx^2$$

En realidad, optimización no-lineal es el caso general, siendo la optimización lineal un caso específico de no-lineal.

Los algoritmos que son útiles para no-lineal pueden resolver un problema lineal, mientras que al revés no siempre es posible.

Los problemas típicos en Física, Ingeniería, ...

OPTIMIZACIÓN NO-LINEAL



OPTIMIZACIÓN NO-LINEAL

Optimización cuadrática.

Un caso específico de optimización no lineal es lo que se llama **programación cuadrática** que en este caso la función a minimizar es cuadrática y las desigualdades son lineales. Se pueden usar los mismos algoritmos o variantes del mismo que en el caso de programación lineal.

$$\begin{aligned} \underset{x_1, x_2}{\text{Min}} \quad & \frac{1}{2}x_1^2 + x_2^2 - x_1x_2 - 2x_1 - 6x_2 \\ \text{subject to:} \quad & x_1 + x_2 \leq 2 \\ & -x_1 + 2x_2 \leq 2 \\ & 2x_1 + x_2 \leq 3 \\ & x_1 \geq 0 \\ & x_2 \geq 0 \end{aligned}$$

Útil cuando queremos minimizar la distancia entre 2 cantidades

$$d = (obj - res)^2$$

OPTIMIZACIÓN NO-LINEAL

Optimización cuadrática.

Un caso específico de optimización no lineal es lo que se llama **programación cuadrática** que en este caso la función a minimizar es cuadrática y las desigualdades son lineales. Se pueden usar los mismos algoritmos o variantes del mismo que en el caso de programación lineal.

$$\begin{aligned} \text{Min}_{x_1, x_2} \quad & \frac{1}{2}x_1^2 + x_2^2 - x_1x_2 - 2x_1 - 6x_2 \\ \text{subject to:} \quad & x_1 + x_2 \leq 2 \\ & -x_1 + 2x_2 \leq 2 \\ & 2x_1 + x_2 \leq 3 \\ & x_1 \geq 0 \\ & x_2 \geq 0 \end{aligned}$$

Útil cuando queremos minimizar la distancia entre 2 cantidades

En python:

```
from scipy.optimize import minimize  
help(minimize)
```

OPTIMIZACIÓN NO-LINEAL

Optimización cuadrática.



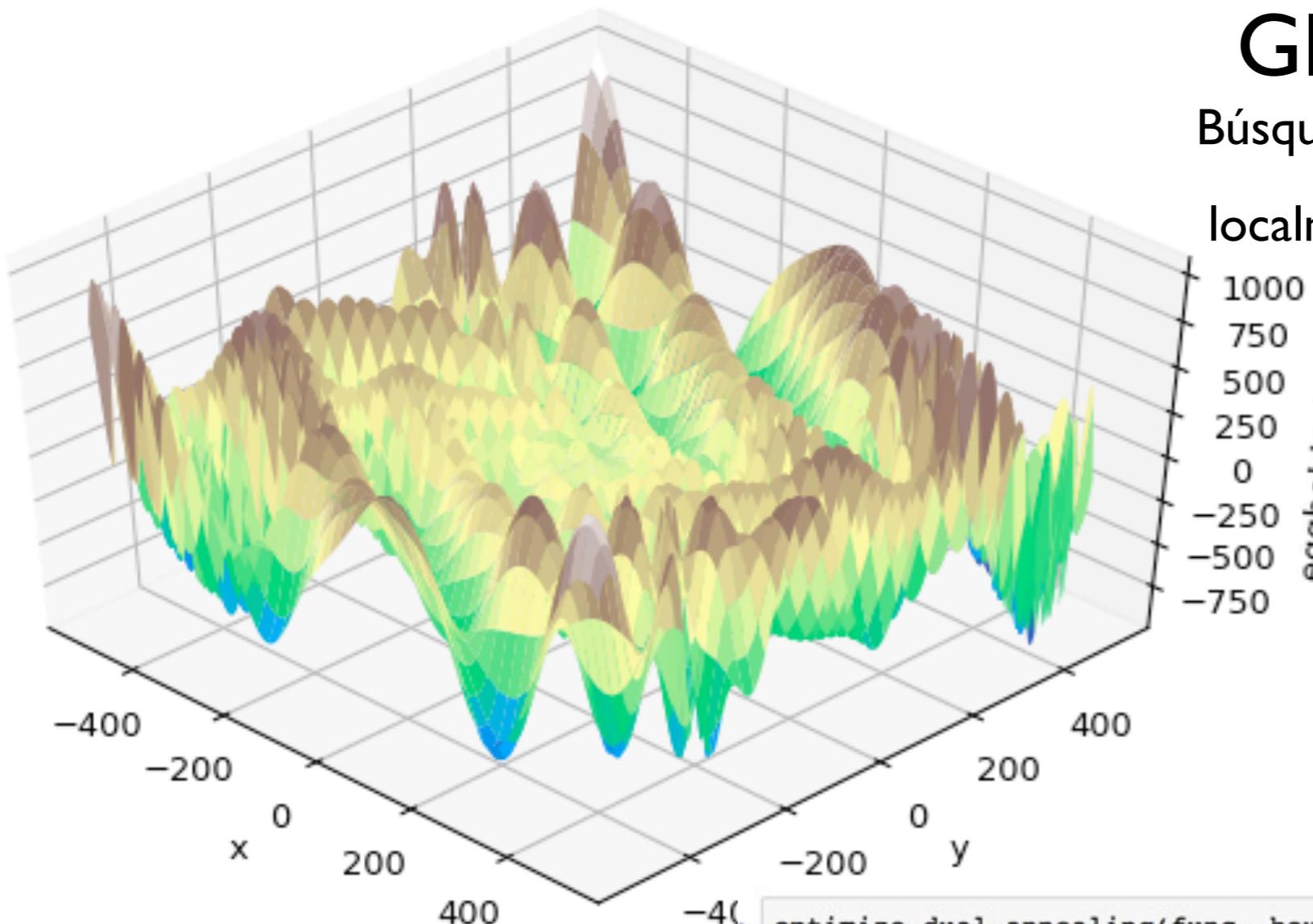
En el caso de la dieta:

Ej. Si quiero gastar 30 euros, buscamos el menú más saludable que te un precio más parecido a 30

$$D = (30 - P_{total})^2$$

$$D = (30 - x_1 p_1 - x_2 p_2 - x_3 p_3 - \dots)^2$$

¿Qué ALGORITMO (method,optimizer,...) utilizar? **Funciones extremas.**

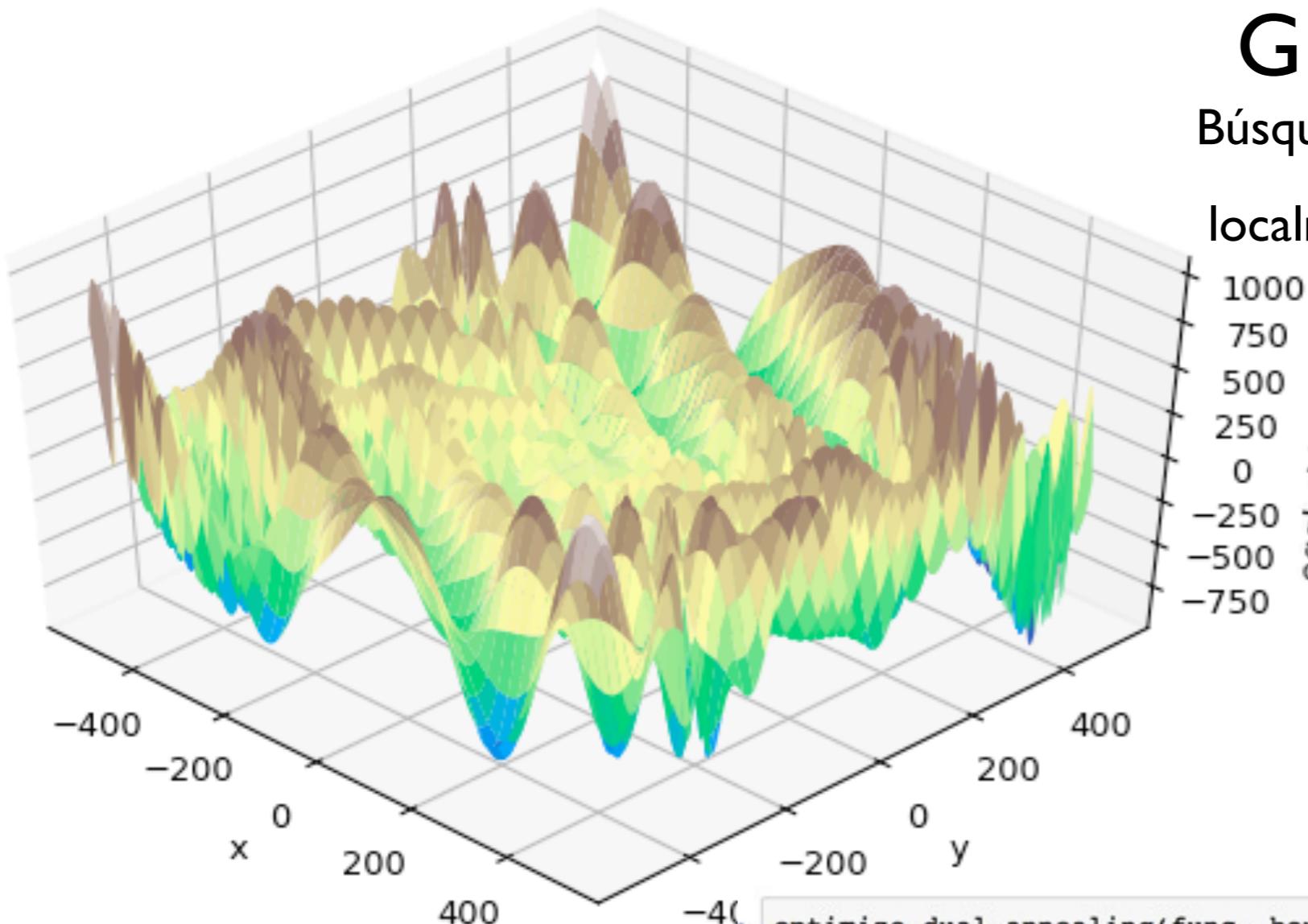


Global minimization.

Búsqueda directa, o exploración usando
localmente minimizadores anteriores...

```
optimize.dual_annealing(func, bounds)
optimize.basinhopping(func, bounds)
optimize.shgo(func, bounds)
optimize.differential_evolution(eggholder, bounds)
```

¿Qué ALGORITMO (method,optimizer,...) utilizar? **Funciones extremas.**



Global minimization.

Búsqueda directa, o exploración usando
localmente minimizadores anteriores...

```
optimize.dual_annealing(func, bounds)
optimize.basinhopping(func, bounds)
optimize.shgo(func, bounds)
optimize.differential_evolution(eggholder, bounds)
```

<https://docs.scipy.org/doc/scipy/reference/tutorial/optimize.html>

OPTIMIZACIÓN. Variables enteras.



- O si simplemente queremos resolverlo con números enteros

- ¿Cuál es el **número mínimo** de items que puedo comprar si quiero comer un **total de 2400 calorías**?

scipy.optimize no puede resolver problemas con variables enteras

OPTIMIZACIÓN. Variables enteras.



- O si simplemente queremos resolverlo con números enteros

- ¿Cuál es el **número mínimo** de items que puedo comprar si quiero comer un **total de 2400 calorías**?

scipy.optimize no puede resolver problemas con variables enteras

PULP

pip install pulp
conda install pulp

Optimizar con Python

(vemos ejemplo con notebook)

`scipy.optimize`

`fmin`

minimizar función

`linprog`

minimizar función lineal con constraints lineales

`minimize`

minimizar general (función no lineal con o sin constraints)

`dual_annealing, ...` global optimization (funciones con muchos mínimos)

`PULP` variables enteras (y para todo lo anterior)
(otros paquetes PyOpt, cvxpy, cvxopt,...)



PuLP 1.6.0

Slides, notebooks y set de datos:

<https://github.com/biuse/Optimizacion2019>