Algoritmos Genéticos en Python: Introducción a las Técnicas de Optimización Metaheurísticas

Daniel Gutiérrez Reina, dgutierrezreina@us.es

SWYP: Students, Women in Engineering and Young Professionals,

Contenido día I

ACE-TI

Grupo de investigación
Ingeniería Electrónica

- Sobre mi.
- Motivación.
- Optimización metaheurística versus métodos tradiciones basados en gradiente.
- Complejidad de los problemas: P Vs NP.
- Introducción a los algoritmos genéticos.
- El problema del viajero o TSP.
- Ejemplo en Python utilizando deap.

Sobre mi

- Doctor Ingeniero en Electrónica por la Universidad de Sevilla (2015).
- Profesor del **Departamento de Ingeniería Electrónica**, **Universidad de Sevilla**, contratado postdoctoral Plan Propio de Investigación de la Universidad de Sevilla.
- He impartido alrededor de 20 cursos de Python, Optimización y Machine Learning.

Twitter:

@Dany503;

https://twitter.com/Dany503

Github:

https://github.com/Dany503/WebinarIEEEAG2020

Google scholar:

https://scholar.google.es/citations?user=-BCyV50AAAAJ&hl=en

Motivación

ACE-TI
Grupo de investigación
Ingeniería Electrónica

¿Sabes qué es un algoritmo genético?

Un algoritmo es una serie de pasos que describen el proceso de búsqueda de una solución a un problema concreto. Y un algoritmo genético es cuando se usan mecanismos que simulan los de la evolución de las especies de la biología para formular esos pasos. Es una técnica de inteligencia artificial inspirada en la idea de que el que sobrevive es el que está mejor adaptado al medio, es decir la misma que subyace a la teoría de la evolución que formuló Charles Darwin y que combina esa idea de la evolución con la genética.

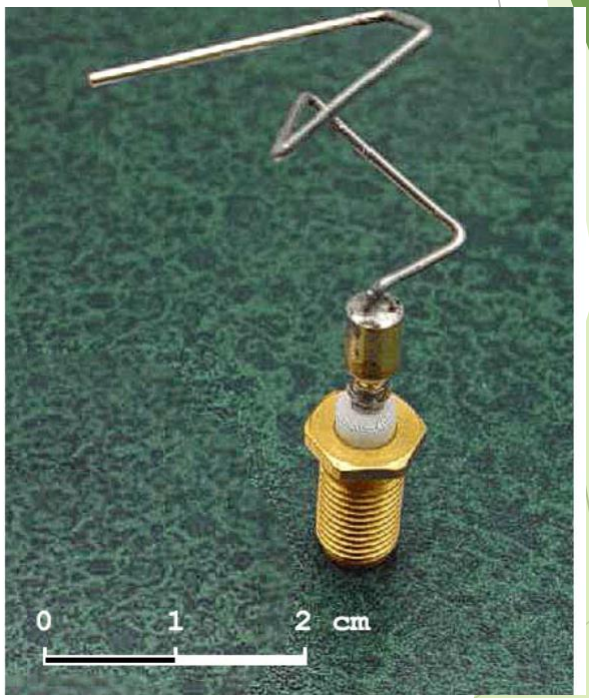
https://elpais.com/elpais/2019/01/31/ciencia/1548933080_909466.html

Términos: Algoritmo · Evolución de las especies · Inteligencia artificial · Genética

Motivación

¿Sabes qué es un algoritmo genético?

Los algoritmos genéticos persiguen encontrar la solución a algunos problemas imitando los mecanismos que condicionan la evolución biológica



https://www.xataka.com/investigacion/computacion-biologica-que-como-nos-esta-ayudando-a-resolver-algunos-grandes-retos-a-que-se-enfrenta-humanidad

Motivación

ACE-TI
Grupo de investigación
Ingeniería Electrónica

¿Sabes qué es un algoritmo genético?

Algoritmo genético

Un algoritmo es una serie de pasos organizados que describe el proceso que se debe seguir, para dar solución a un problema específico.

En los años 1970, de la mano de John Henry Holland, surgió una de las líneas más prometedoras de la inteligencia artificial, la de los **algoritmos genéticos**, (AG).[1] [2] Son llamados así porque se inspiran en la evolución biológica y su base genético-molecular.

Estos algoritmos hacen evolucionar una población de individuos sometiéndola a acciones aleatorias semejantes a las que actúan en la evolución biológica (mutaciones y recombinaciones genéticas), así como también a una selección de acuerdo con algún criterio, en función del cual se decide cuáles son los individuos más adaptados, que sobreviven, y cuáles los menos aptos, que son descartados.

Los algoritmos genéticos se enmarcan dentro de los algoritmos evolutivos, que incluyen también las estrategias evolutivas, la programación evolutiva y la programación genética.

https://es.wikipedia.org/wiki/Algoritmo_gen%C3%A9tico

Problema específico 1970 Acciones aleatorias Selección Algoritmos evolutivos

Motivación

ACE-TI
Grupo de investigación
Ingeniería Electrónica

¿Sabes qué es un algoritmo genético?

Un algoritmo de inteligencia artificial → Un herramienta

Un algoritmo de optimización → Maximizar / Minimizar.

Resolver problemas específicos → Complejos.

Basado en la teoría de la evolución → Teoría de Darwin.

Idea de los años 70 → No es nada nuevo.

Motivación

Inteligencia artificial

Machine Learning

Deep Learning

Aprendizaje por refuerzo

Algoritmos bio-inspirados o metaheurísticos

Algoritmos de inteligencia artificial

Motivación

ACE-TI
Grupo de investigación
Ingeniería Electrónica

Aplicaciones: planificación de rutas de vehículos autónomos

Lago Ypacarai (*Asunción, Paraguay*)

Catamarán autónomo

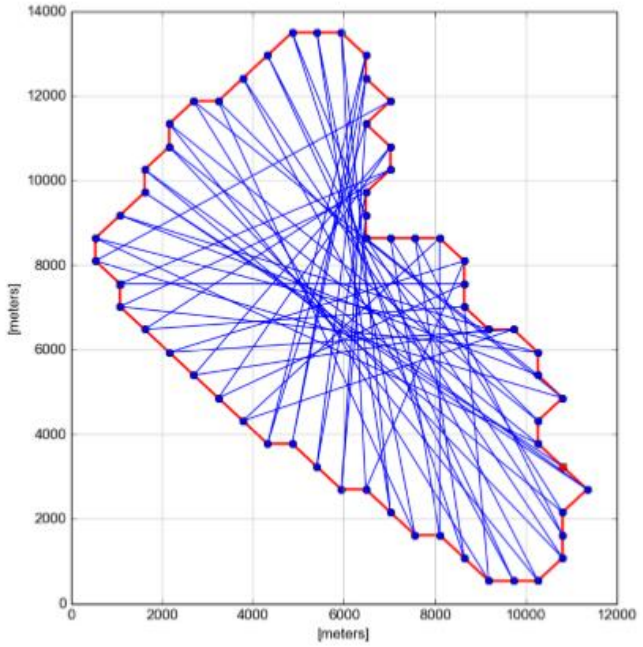
Motivación

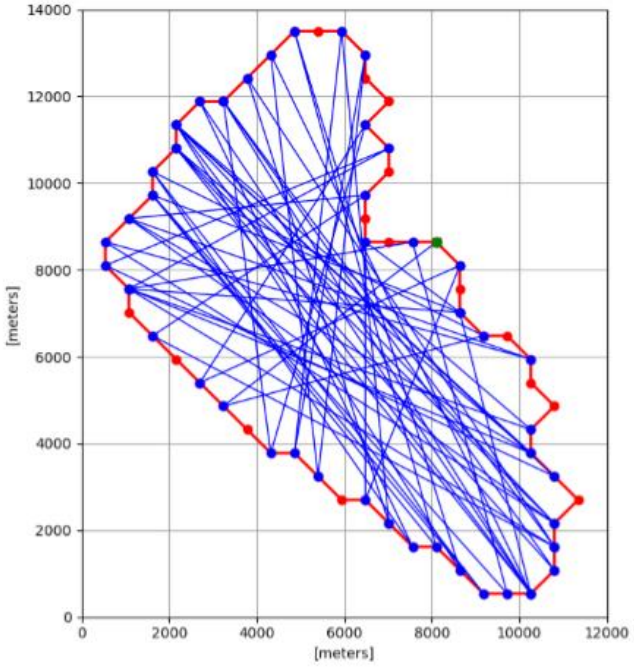
ACE-TI
Grupo de investigación
Ingeniería Electrónica

Aplicaciones: planificación de rutas de vehículos autónomos

Objetivo:

Queremos maximizar la cobertura





Motivación

ACE-TI
Grupo de investigación
Ingeniería Electrónica

Aplicaciones: despacho económico en redes eléctricas

Microred

Elementos.

1. Generado eólico.
2. Paneles fotovoltaicos
3. Microturbina
4. Generador Diesel
5. Sistema de almacenamiento de energía (baterías).
6. Viviendas (demanda).

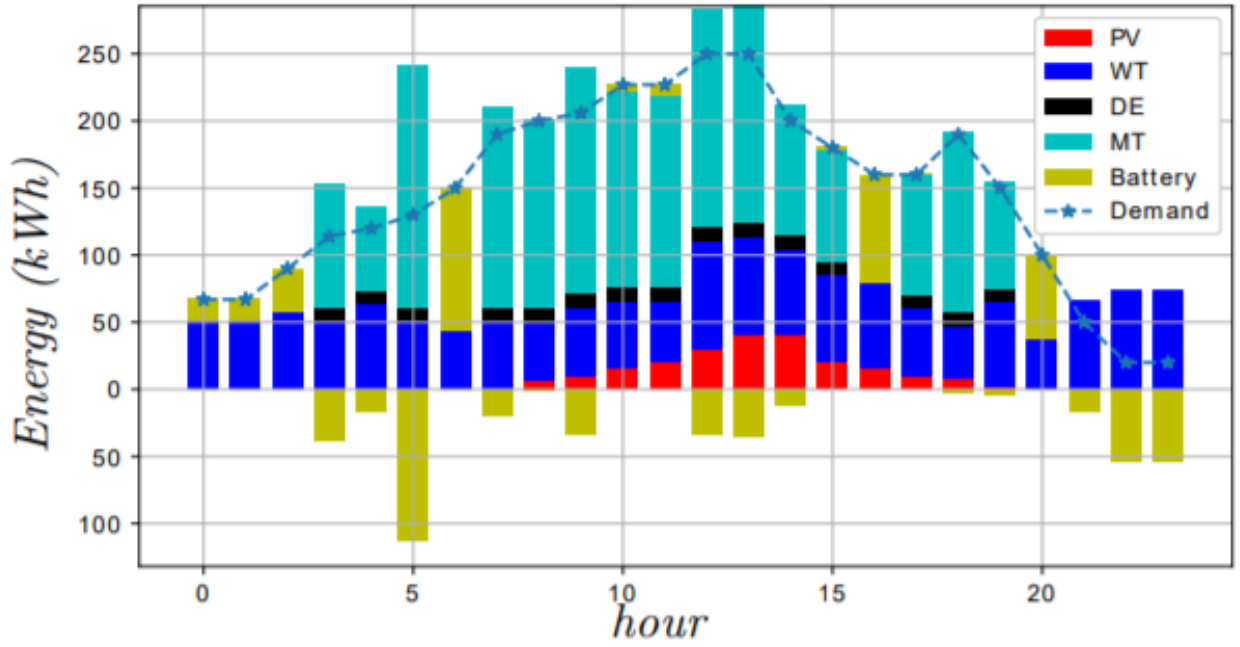
Motivación

ACE-TI
Grupo de investigación
Ingeniería Electrónica

Aplicaciones: despacho económico redes eléctricas

Objetivo:

Queremos minimizar el coste de operación



Motivación

ACE-TI
Grupo de investigación
Ingeniería Electrónica

¿Sabes qué es un algoritmo genético?

Un algoritmo de optimización → Maximizar / Minimizar.

Resolver problemas específicos → Complejos.

Basado en la teoría de la evolución → Teoría de Darwin.

Idea de los años 70 → No es nada nuevo.

¿Qué es optimizar?

Dado un conjunto de posibles soluciones a un problema, el objetivo de la optimización es encontrar la mejor solución (óptimo) de acuerdo a una determinada medida de bondad de la solución (fitness function).

Tipos:

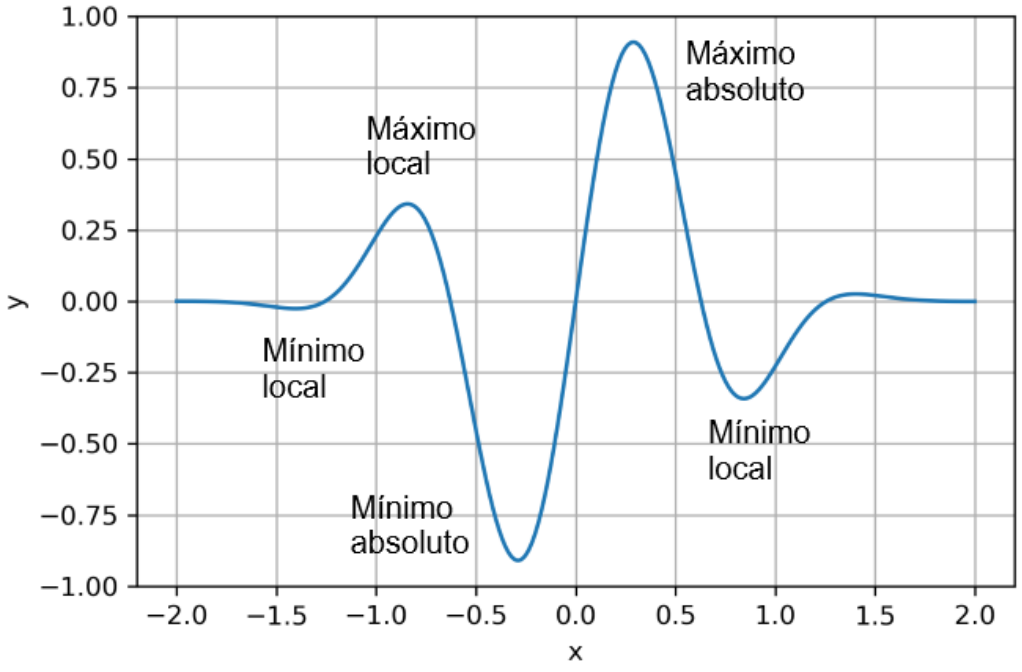
1. Maximización del rendimiento y/o minimización de una función de coste.
2. Soluciones analíticas vs soluciones numéricas.

Conceptos básicos de optimización

Espacio de búsqueda

Espacio que contiene el conjunto de soluciones posibles a un problema de optimización – Óptimo local/global

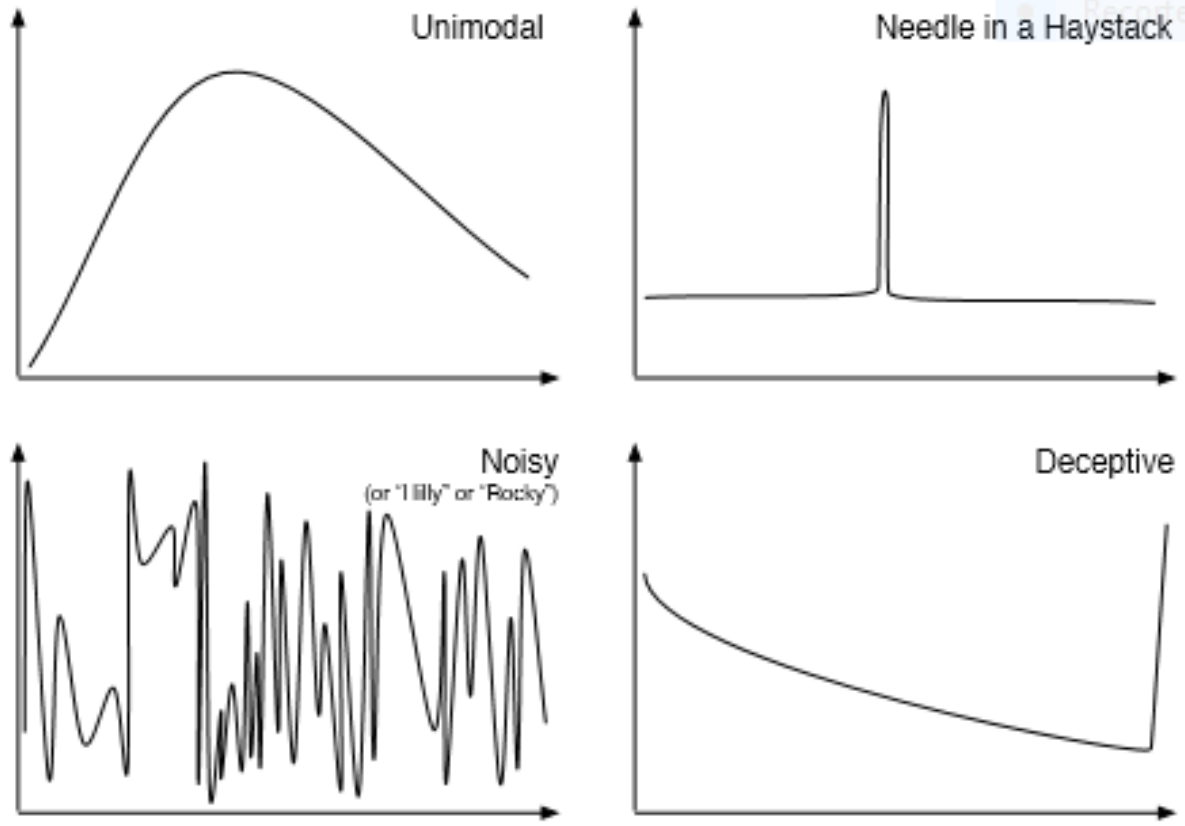
$$y = f(x)$$



Conceptos básicos de optimización

ACE-TI
Grupo de investigación
Ingeniería Electrónica

Tipos de espacios de búsqueda



Conceptos básicos de optimización

ACE-TI
Grupo de investigación
Ingeniería Electrónica

¿Cómo resolver un problema de optimización?

- ***Fuerza bruta:*** Evaluar todas las posibles soluciones. Poco eficiente computacionalmente o imposible. Muy ineficiente y poco práctico.
- ***Búsqueda aleatoria:*** Búsqueda aleatoria sin ninguna guía. No garantiza soluciones óptimas, ni quasi-óptimas. Simplemente reduzco el número de evaluaciones.
- ***Técnicas basadas en gradiente:*** Requiere la existencia de derivadas; búsqueda local- problema de óptimos locales.
- ***Algoritmos meta heurísticos:*** Tenemos que ser capaces de evaluar las soluciones. Medir la calidad. **No hay garantías de obtener el óptimo.**

Conceptos básicos de optimización

ACE-TI
Grupo de investigación
Ingeniería Electrónica

Métodos de optimización basados en gradientes

- Lo que nos han enseñado en las clases de cálculo (Aquellos maravillosos años en el instituto).
- 1) Cálculo de los puntos críticos → Primera derivada e igualamos a 0: *x1 tal que* ***f'(x1) = 0***.
- 2) Puntos de inflexión → Segunda derivada y evaluábamos los puntos críticos: ***f''(x1) > 0*** *mínimo ó* ***f''(x1) < 0*** *máximo relativos*.
- Método de Newton.

Conceptos básicos de optimización

ACE-TI
Grupo de investigación
Ingeniería Electrónica

Métodos de optimización basados en gradientes

- Funciones con múltiples variables $\nabla f(x,y) = [fx, fy]$
- Matrices Hessianas → Segundas derivadas parciales.
- Multiplicadores de Lagrange. Optimización con restricciones.

- SIEMPRE TENEMOS QUE CALCULAR LAS DERIVADAS O APROXIMARLAS!!!!
- ¿Es siempre posible?
  - No tenemos la función matemática.
  - Dimensiones del problema muy grandes.
  - ***¿Queremos el óptimo global o nos vale con una buena solución?***

Motivación

ACE-TI
Grupo de investigación
Ingeniería Electrónica

¿Sabes qué es un algoritmo genético?

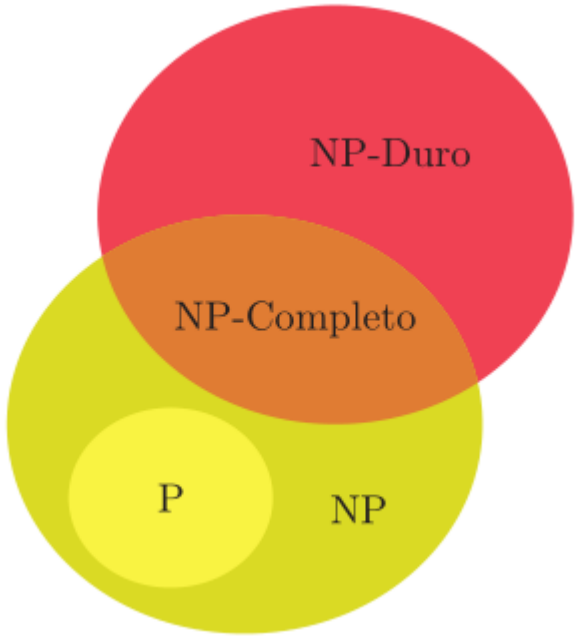
Un algoritmo de optimización → Maximizar / Minimizar.

Resolver problemas específicos → Complejos.

Basado en la teoría de la evolución → Teoría de Darwin.

Idea de los años 70 → No es nada nuevo.

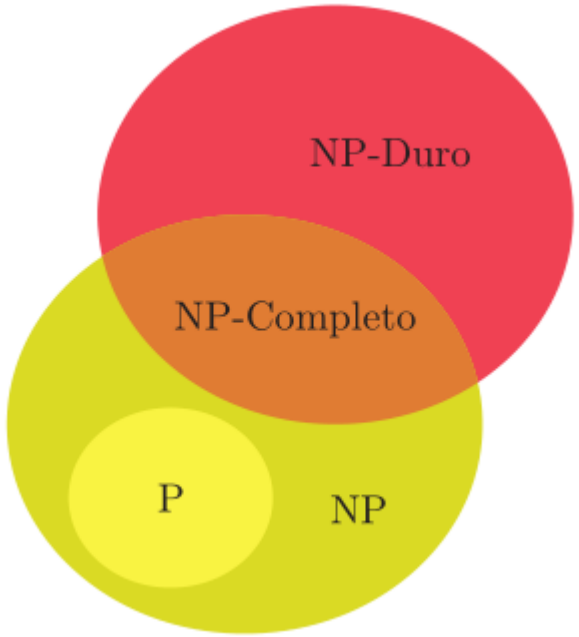
Complejidad de los problemas: P vs NP



Problemas NP (non deterministic polynomial time):

- No existe un algoritmo determinista que los resuelva en tiempo Polinomial para cualquier dimensión del problema.
- Sí podemos comprobar si una solución en correcta en tiempo polinomial.

Complejidad de los problemas: P vs NP



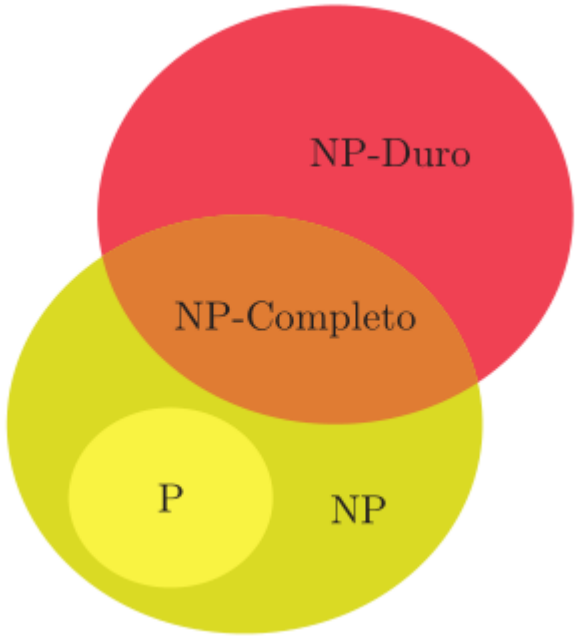
Problemas NP (non deterministic polynomial time):

¿Tiempo polinomial? Número de operaciones que se requieren para resolverlo.

Ejemplos: N^2, 2*N, log(N), siendo N la dimensión del problema (variables)

Tiempo no polinomial: 2^N, 3^N, exp(N)

Complejidad de los problemas: P vs NP



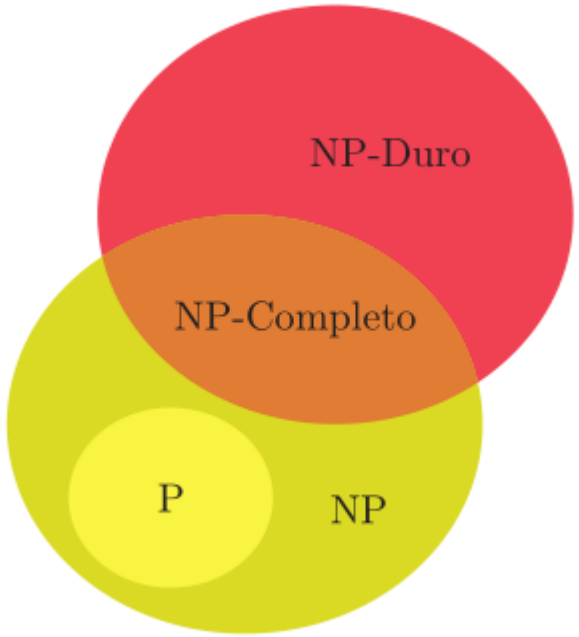
Problemas NP (non deterministic polynomial time):

No determinista: No tiene ninguna componente aleatoria

```
import random
l1 = [10, 11, 4, 2, 1, 10]

minimo = l1[0]
N = len(l1)
for i in range(N):
    indice = random.randint(0, N-1)
    if l1[indice] < minimo:
        minimo = l1[indice]
print(minimo)
```

Complejidad de los problemas: P vs NP

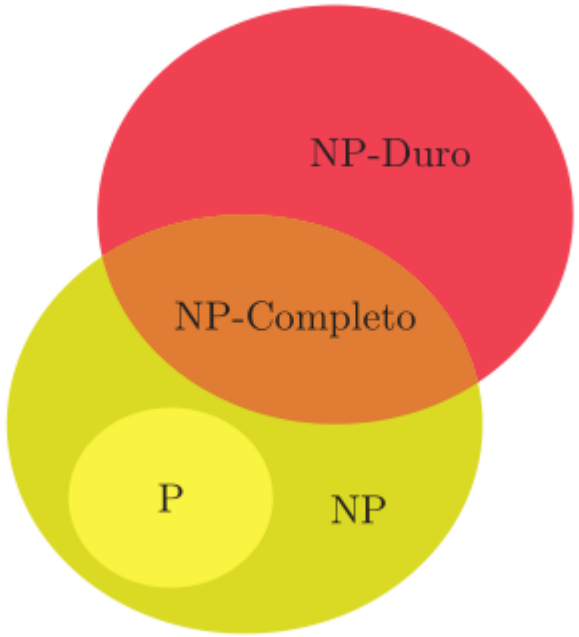


Problemas NP (non deterministic polynomial time):

Ejemplo: Sudoku

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

Complejidad de los problemas: P vs NP



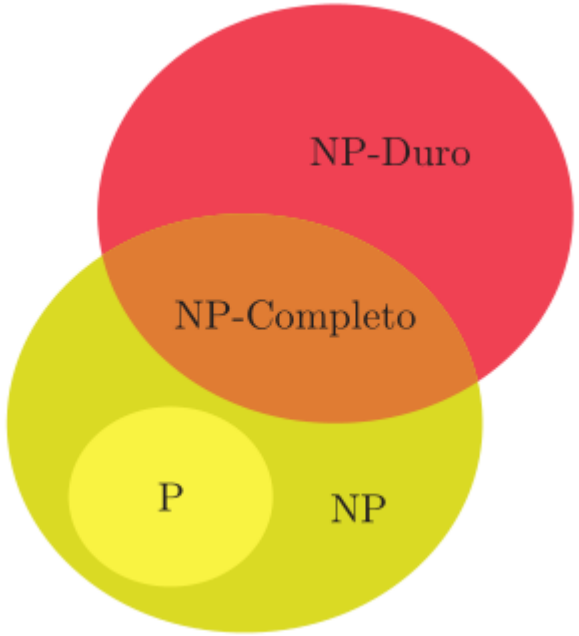
Problemas P (polynomial time):

- Existe un algoritmo determinista que los resuelve en tiempo Polinomial → Básicamente solo necesitamos operaciones sencillas, sumas, restas, comparaciones, etc.

- Podemos comprobar si la solución es correcta en tiempo polinomial-

Ejemplos: algoritmo de ordenación de la burbuja (N^2), búsqueda binaria (log(N)), multiplicación de matrices (N^3).

Complejidad de los problemas: P vs NP



¿Por qué P dentro de NP?

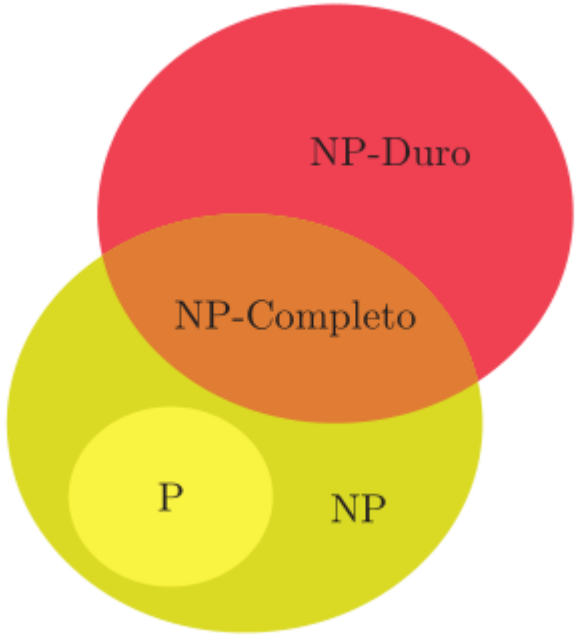
- Un problema que ahora es NP en el futuro puede pasar a P, o no.
- Problema del milenio: ¿P = NP?

http://www.claymath.org/millennium-problems

P vs NP Problem

If it is easy to check that a solution to a problem is correct, is it also easy to solve the problem? This is the essence of the P vs NP question. Typical of the NP problems is that of the Hamiltonian Path Problem: given N cities to visit, how can one do this without visiting a city twice? If you give me a solution, I can easily check that it is correct. But I cannot so easily find a solution.

Complejidad de los problemas: P vs NP

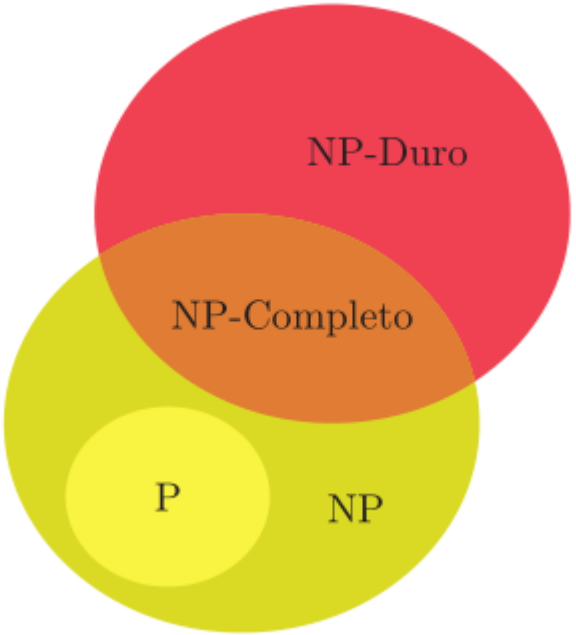


Problemas NP duros (Intratables)

- Son los problemas más complicados.
- No existe un algoritmo determinista que los resuelva en tiempo polinomial.
- No podemos comprobar si la solución es correcta en tiempo polinomial.

Complejidad de los problemas: P vs NP

ACE-TI
Grupo de investigación
Ingeniería Electrónica



Problemas NP duros

Ejemplos:

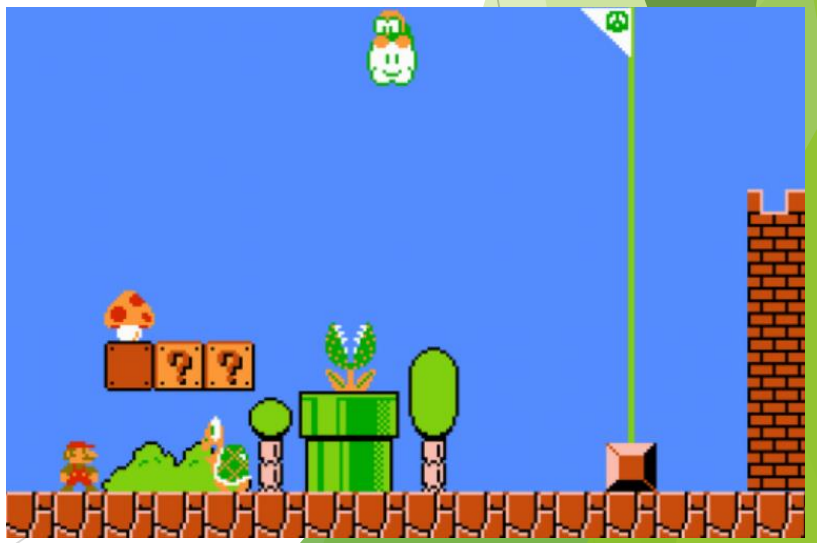
Candy crash

(a) Candy Crush Saga

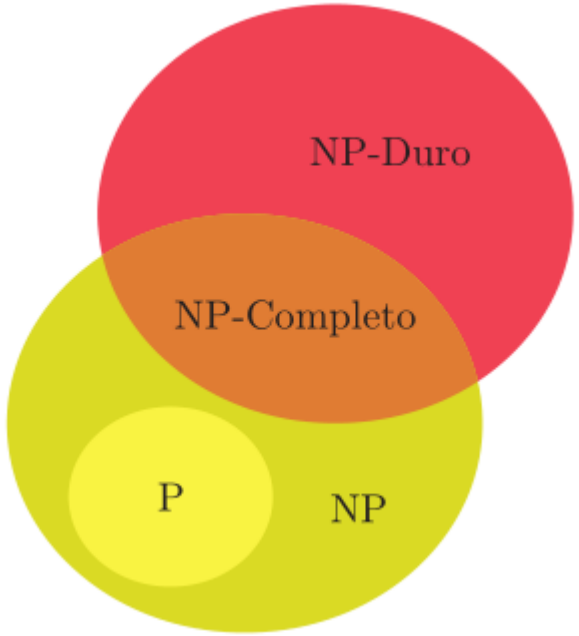
(b) Bejeweled

(c) Shariki

Mario Bros



Complejidad de los problemas: P vs NP

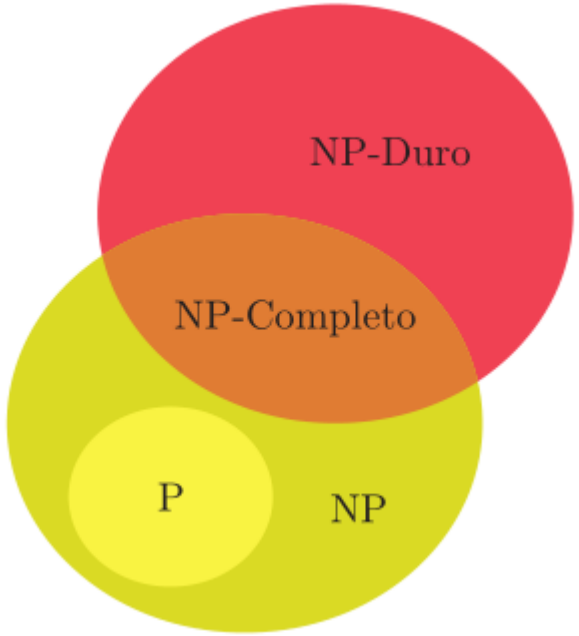


Problemas NP Completos

- Son problemas que son NP-duros y NP a la vez.
- Básicamente son NP porque requieren tiempo exponencial y son NP porque sí puedo comprobar la solución en tiempo polinomial.

Complejidad de los problemas: P vs NP

ACE-TI
Grupo de investigación
Ingeniería Electrónica



Problemas NP Completos

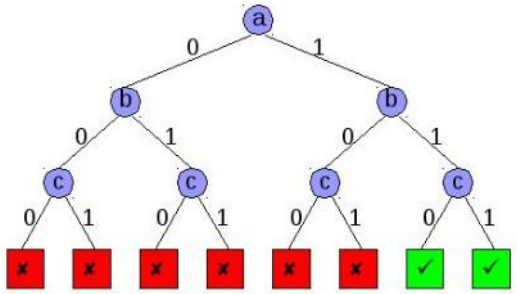
Ejemplos:

Sudoku

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

Factibilidad Booleana

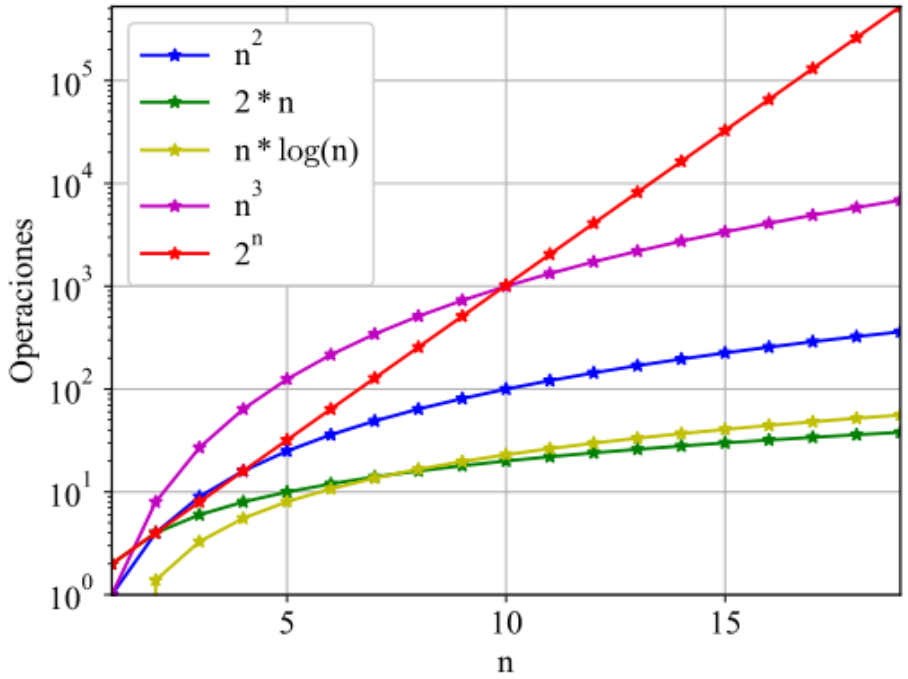
$(a \vee b) \wedge (a \vee \neg c) \wedge (\neg a \vee b) \wedge (a \vee c)$



¿Qué ocurre con el tiempo exponencial?

ACE-TI
Grupo de investigación
Ingeniería Electrónica

Necesitamos hacer muchas operaciones



https://www.youtube.com/watch?v=YX40hbAHx3s

Motivación

ACE-TI
Grupo de investigación
Ingeniería Electrónica

¿Sabes qué es un algoritmo genético?

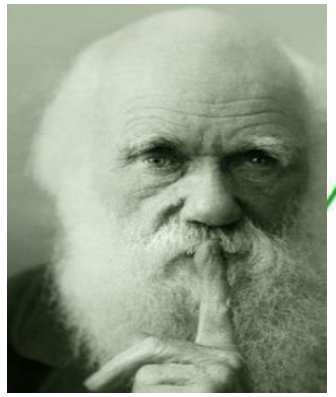
Un algoritmo de optimización → Maximizar / Minimizar.

Resolver problemas específicos → Complejos.

Basado en la teoría de la evolución → Teoría de Darwin.

Idea de los años 70 → No es nada nuevo.

Teoría de la evolución de Darwin



ACE-TI
Grupo de investigación
Ingeniería Electrónica

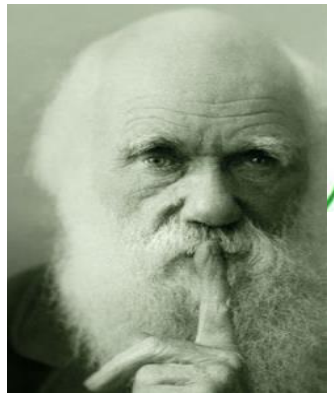
De manera simple …

Los individuos que **mejor se adaptan al medio** son aquellos que tienen **más probabilidades de dejar descendencia**, y que por lo tanto, sus genes pasarán a las siguientes generaciones.

La teoría de Darwin también describe que aquellas **modificaciones genéticas** que hacen que los **individuos se adapten mejor al medio, tienen mayor probabilidad de perdurar en el tiempo**.

¿Cómo lo relacionamos con problemas de optimización?

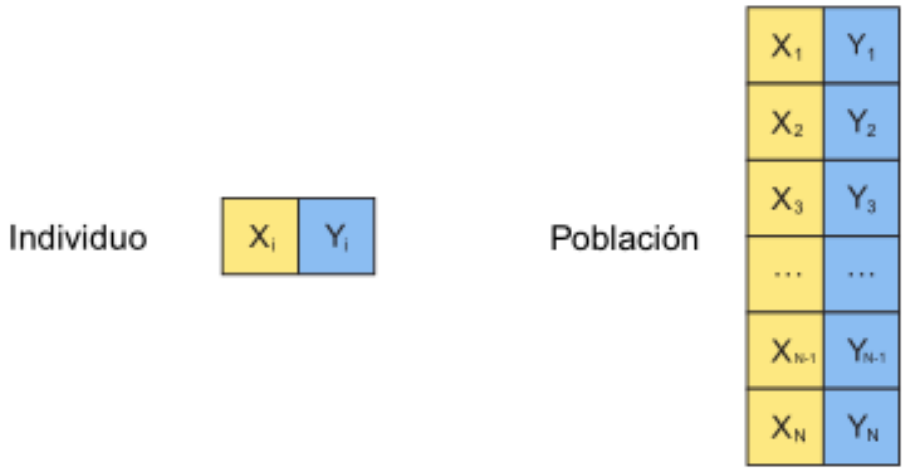
Teoría de la evolución de Darwin



ACE-TI
Grupo de investigación
Ingeniería Electrónica

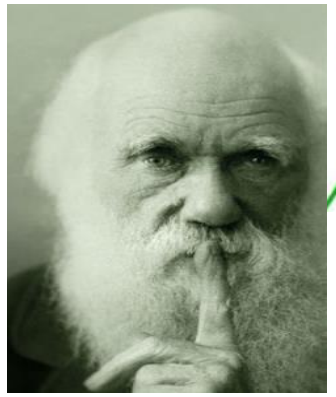
Darwin y optimización

¿Individuo? → Posible solución al problema



Codificación de las variables del problema como una cadena genes o cromosoma

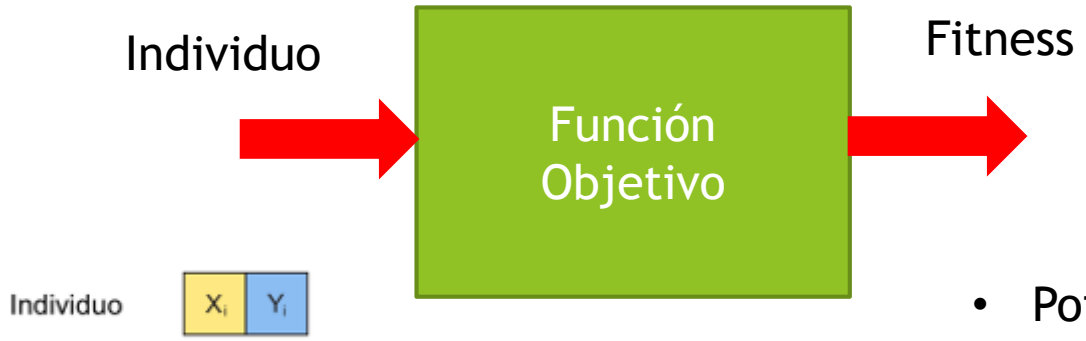
Teoría de la evolución de Darwin



ACE-TI
Grupo de investigación
Ingeniería Electrónica

Darwin y optimización

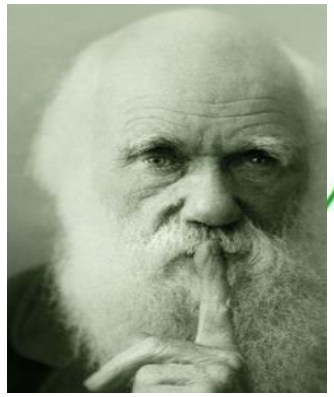
¿Adaptación al medio? → Desempeño o fitness en la función objetivo



Si el desempeño es alto, se adapta bien al medio (maximizar).

- Potencia
- Distancia recorrida
- Coste

Teoría de la evolución de Darwin

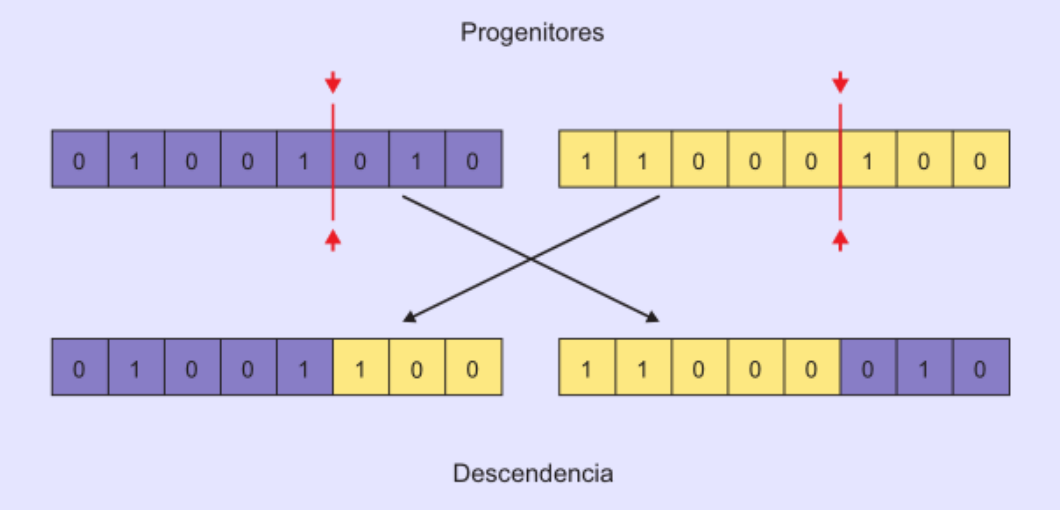


ACE-TI
Grupo de investigación
Ingeniería Electrónica

Darwin y optimización

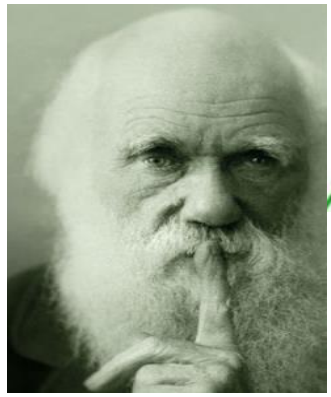
¿Modificaciones genéticas? → Creamos nuevos individuos (descendientes), dos tipos: **cruce y mutación**

Cruce: de padres se crean dos hijos



Cruce de un punto

Teoría de la evolución de Darwin

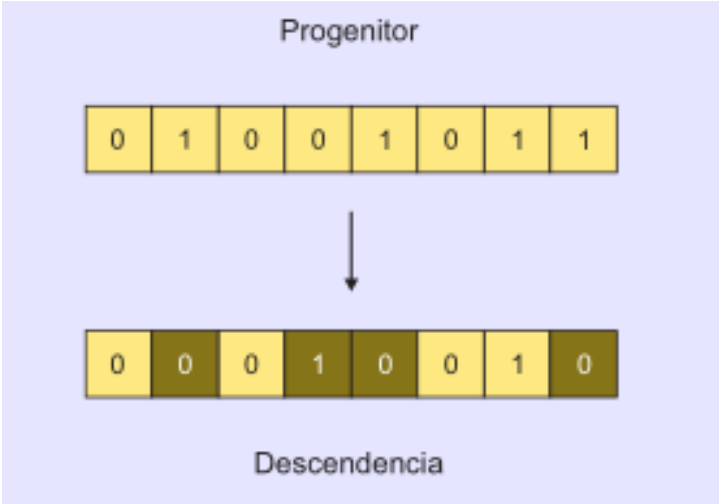


ACE-TI
Grupo de investigación
Ingeniería Electrónica

Darwin y optimización

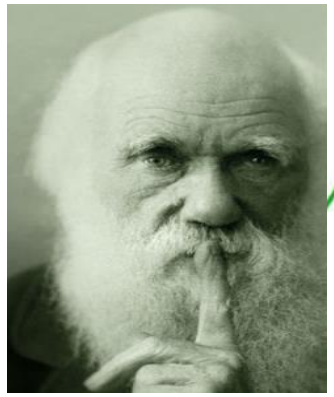
¿Modificaciones genéticas? → Creamos nuevos individuos (descendientes), dos tipos: cruce y mutación

Mutación: mutación de un individuo



BitFlip

Teoría de la evolución de Darwin

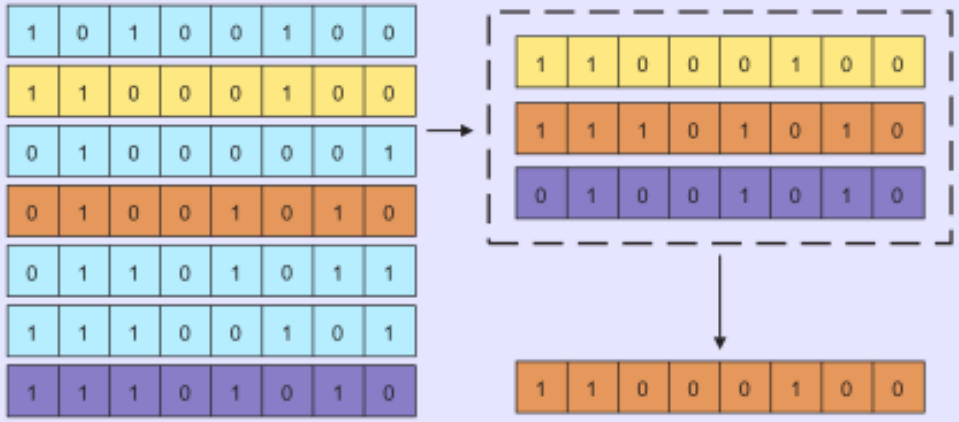


ACE-TI
Grupo de investigación
Ingeniería Electrónica

Darwin y optimización

¿Qué genes perduran en el tiempo? → Selección elitista para elegir los individuos que participarán en la operaciones genéticas

Selección mediante torneo



Los mejores individuos tienen mayor probabilidad de ser seleccionados.

Algoritmos genéticos

Definición

Los GAs son métodos de **búsqueda probabilísticos,** inspirados en los mecanismos de **selección natural y la genética**, para la búsqueda de los óptimos globales de problemas complejos.

- Las soluciones que mejor se adaptan al entorno (problema) sobreviven
- Idea original de John Holland en los 70's.

["Adaptation in natural systems", by J. Holland, 1975]

Algoritmos genéticos

ACE-TI
Grupo de investigación
Ingeniería Electrónica

Nomenclatura:

- *Individuo:* solución o candidato al problema de optimización.
- *Población:* conjunto de candidatos al problema.
- *Fitness:* calidad del individuo.
- *Función objetivo o de fitness:* problema que queremos resolver.
- *Cromosoma:* estructura genética que representa al individuo. Variables de nuestro problema de optimización.
- *Gen:* posición en particular en la estructura cromosómica. Variable.
- *Operaciones genéticas:* operaciones selección, cruce y mutación, para generar nuevos individuos.

Algoritmos genéticos

Idea global de los GAs

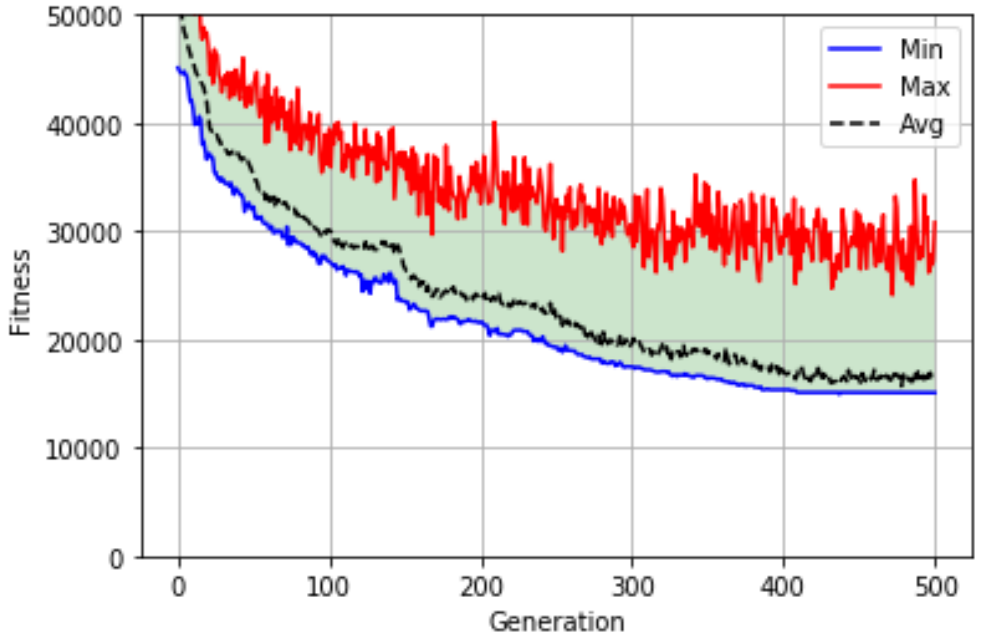
- Población de individuos o posible soluciones que evolucionan a lo largo de un número de generaciones, creándose mejores individuos mediante operaciones genéticas: selección, cruce y mutación).

Población inicial es aleatoria

Algoritmos genéticos

¿Cuándo paro?

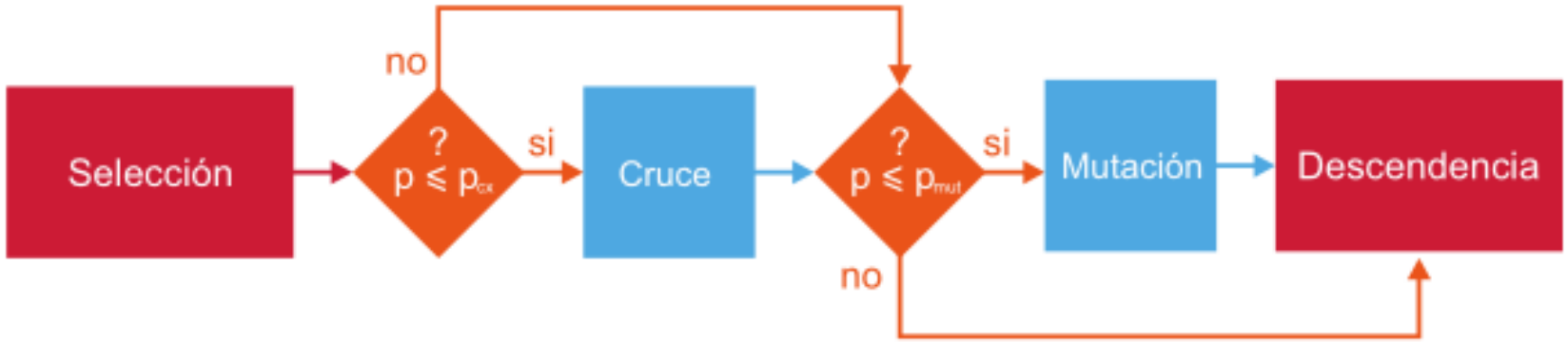
- Si llego al límite de generaciones → Lo puedo fijar por defecto.
- Si considero que el algoritmo ha convergido → Los individuos que se crean no mejoran durante un número de generaciones.



Algoritmos genéticos

Características principales

- GAs utilizan métodos heurísticos, basados en probabilidad (no son métodos exactos) → *Pero sí podemos obtener buenas soluciones!! En muchos casos ni sabes el óptimo.*
  - *Los operadores genéticos son probabilísticos*



Algoritmos genéticos

Características principales

- GAs son computacionalmente intensivos → necesito un buen PC.
  - **¿Cuántas veces voy a evaluar la función objetivo?**
  - M individuos x N generaciones.
  - Dando algunos números: M=100 y N=100, tendríamos 10000 evaluaciones, si tardamos 1 segundo en cada una, 2 horas y 42 minutos.

Algoritmos genéticos

ACE-TI
Grupo de investigación
Ingeniería Electrónica

Exploración versus explotación del espacio de búsqueda

Un buen algoritmo meta heurístico debe hacer un balance adecuado entre dos características contrapuestas del proceso de búsqueda:

- **Exploración**: cantidad de esfuerzo empleado en la búsqueda en regiones distantes del espacio (diversificación).

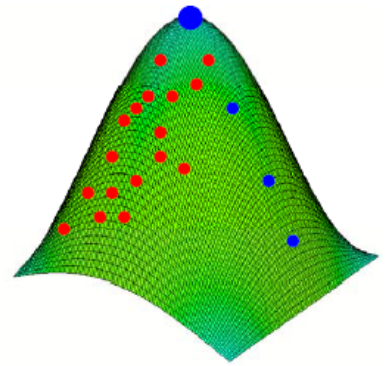
- **Explotación**: cantidad de esfuerzo empleado en la búsqueda en la región actual (intensificación).

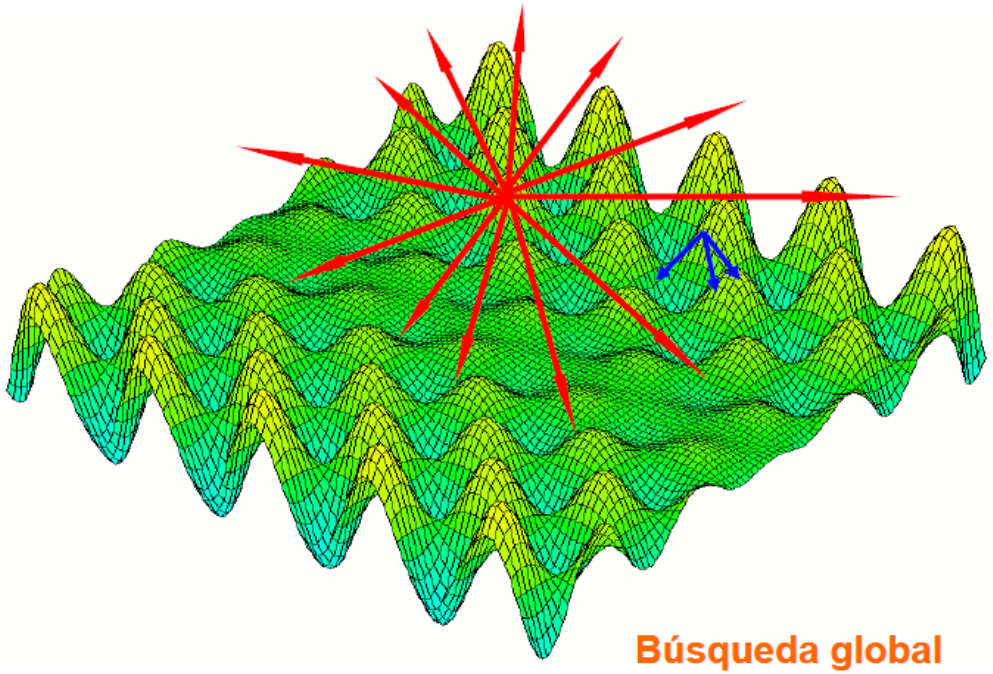
Conceptos básicos de optimización

ACE-TI
Grupo de investigación
Ingeniería Electrónica

Exploración versus explotación del espacio de búsqueda

Búsqueda local
Explotación
Intensificación





Búsqueda global
Exploración
Diversificación

Algoritmos genéticos

ACE-TI
Grupo de investigación
Ingeniería Electrónica

Exploración versus explotación del espacio de búsqueda

¿Cómo se consigue explorar/intensificar con los algoritmos genéticos?

Mediante los operadores genéticos: cruce y mutación.

- **Exploración**: ¿Cruce y/o mutación?
- **Explotación**: ¿Cruce y/o mutación?

Motivación

ACE-TI
Grupo de investigación
Ingeniería Electrónica

¿Sabes qué es un algoritmo genético?

Un algoritmo de optimización → Maximizar / Minimizar.

Resolver problemas específicos → Complejos.

Basado en la teoría de la evolución → Teoría de Darwin.

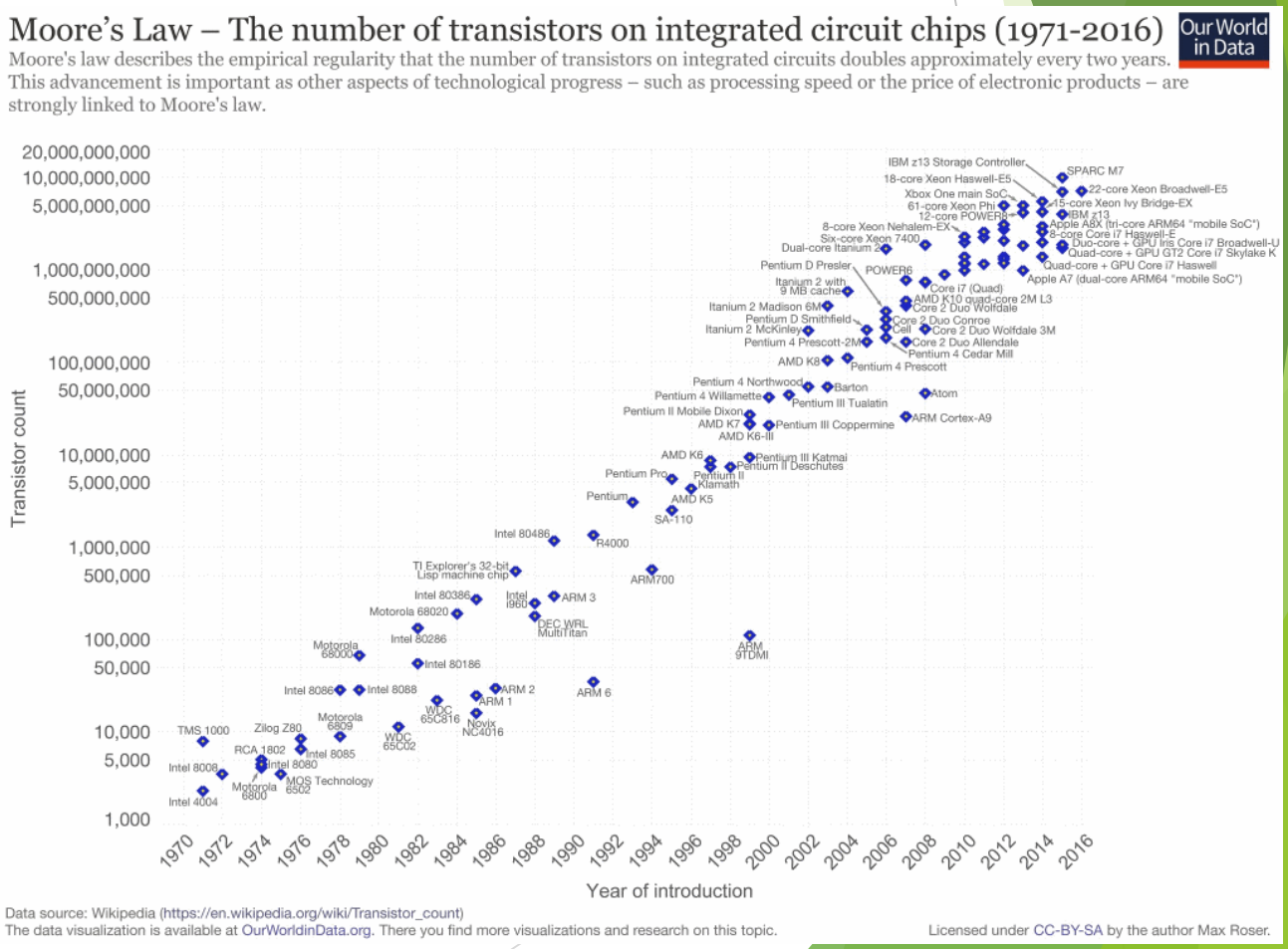
Idea de los años 70 → No es nada nuevo.

¿Por qué ahora?

ACE-TI
Grupo de investigación
Ingeniería Electrónica

Capacidad de computación y lenguajes de programación de alto nivel

- Ley de Moore
- Python



Contenido día I

ACE-TI
Grupo de investigación
Ingeniería Electrónica

- Motivación.
- Optimización metaheurística versus métodos tradiciones basados en gradiente.
- Complejidad de los problemas: P Vs NP.
- Introducción a los algoritmos genéticos.
- El problema del viajero o TSP.
- Ejemplo en Python utilizando deap.

Problema del viajero (TSP)

Objetivo

- Tenemos un agente vendedor que desear recorrer un conjunto de ciudades recorriendo las distancia mínima.
- Restricciones:
  - Empieza y acaba en la misma ciudad.
  - No puede visitar una ciudad más de una vez.
- Problema de tipo NP-duro.

DEAP: Distributed Evolutionary Algorithm in Python

ACE-TI
Grupo de investigación
Ingeniería Electrónica

Paquete de Python con el que podemos implementar algoritmos evolutivos. Ha sido desarrollado por Computer Vision and Systems Laboratory (CVSL) at Université Laval, Quebec, Canada → Proyecto active → Versión 1.3.0
https://deap.readthedocs.io/en/master/

Algoritmos de optimización que podemos usar con DEAP:

- Algoritmos genéticos (single objective).
- Algoritmos genéticos multi-objetivos.
- Programación genética.
- Estrategias evolutivas
- Particle Swarm Optimization.

DISTRIBUTED EVOLUTIONARY ALGORITHMS IN PYTHON

Problema del viajero (TSP)

¿Fuerza bruta?

- Considerando N ciudades (tamaño de la entrada = n), la dimensión del espacio de búsqueda (permutaciones) es: (n-1)!/2

- Utilizando la fuerza bruta y explorando 1.000 permutaciones por segundo

n	*Permutaciones*	*Tiempo estimado*
10	181.440	3 minutos.
12	19.958.400	5 horas y media.
20	60.822.550.204.416.000	1.928.670 años.

Problema del viajero (TSP)

ACE-TI
Grupo de investigación
Ingeniería Electrónica

Representación de los individuos: (posibles soluciones)

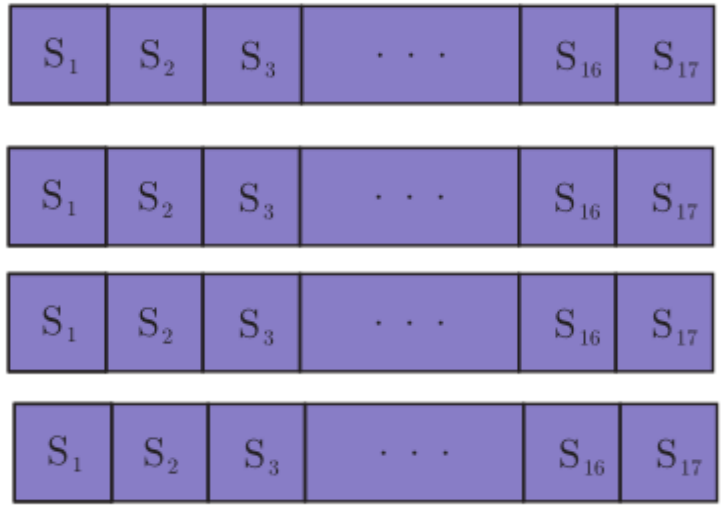
- **Lista de ciudades que visita el agente**.
- La posición de la ciudad en la lista determina el orden del visita.
- **Importante:** No puede haber ciudades repetidas.
- Necesitamos una función para generar individuos aleatorios (población)

Cada “Si” es una ciudad.

Problema del viajero (TSP)

Población: Conjunto de individuos sobre los que se aplican los operadores genéticos.

- Población inicial:
  - Aleatoria o dada.
  - Tamaño y diversidad.



Problema del viajero (TSP)

Función de fitness: medida de la bondad de cada individuo de una población. **Distancia que recorre el agente.**

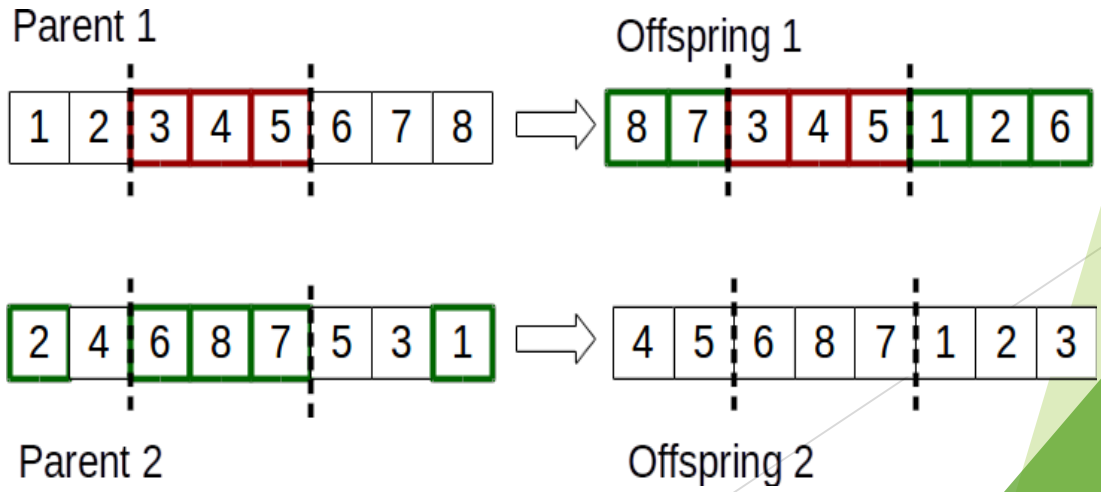
```
def evalTSP(individual):
    """ Función objetivo, calcula la distancia que recorre el viajante"""
    # distancia entre el último elemento y el primero
    distance = distance_map[individual[-1]][individual[0]]
    # distancia entre el resto de ciudades
    for gene1, gene2 in zip(individual[0:-1], individual[1:]):
        distance += distance_map[gene1][gene2]
    return distance,
```

Problema del viajero (TSP)

ACE-TI
Grupo de investigación
Ingeniería Electrónica

Operadores genéticos: construcción de la siguiente generación

- ***Cruce:*** dos individuos padres combinan su información para generar dos individuos hijos.
- En nuestro caso tenemos que tener cuidado → No nos vale cualquier operación.
- **Cruce de dos puntos:**

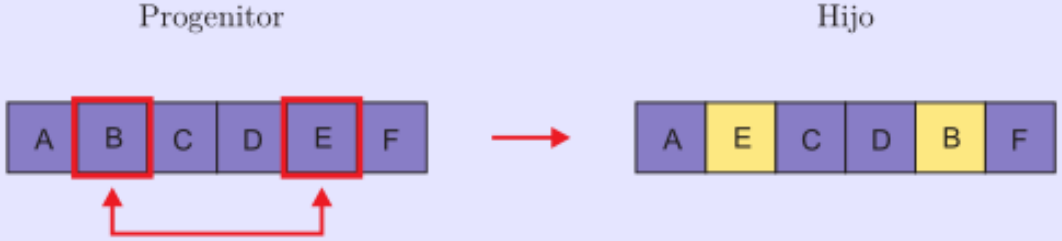


Problema del viajero (TSP)

Operadores genéticos: construcción de la siguiente generación.

- ***Mutación***: alteración de la información genética de un individuo al pasar a la siguiente generación.

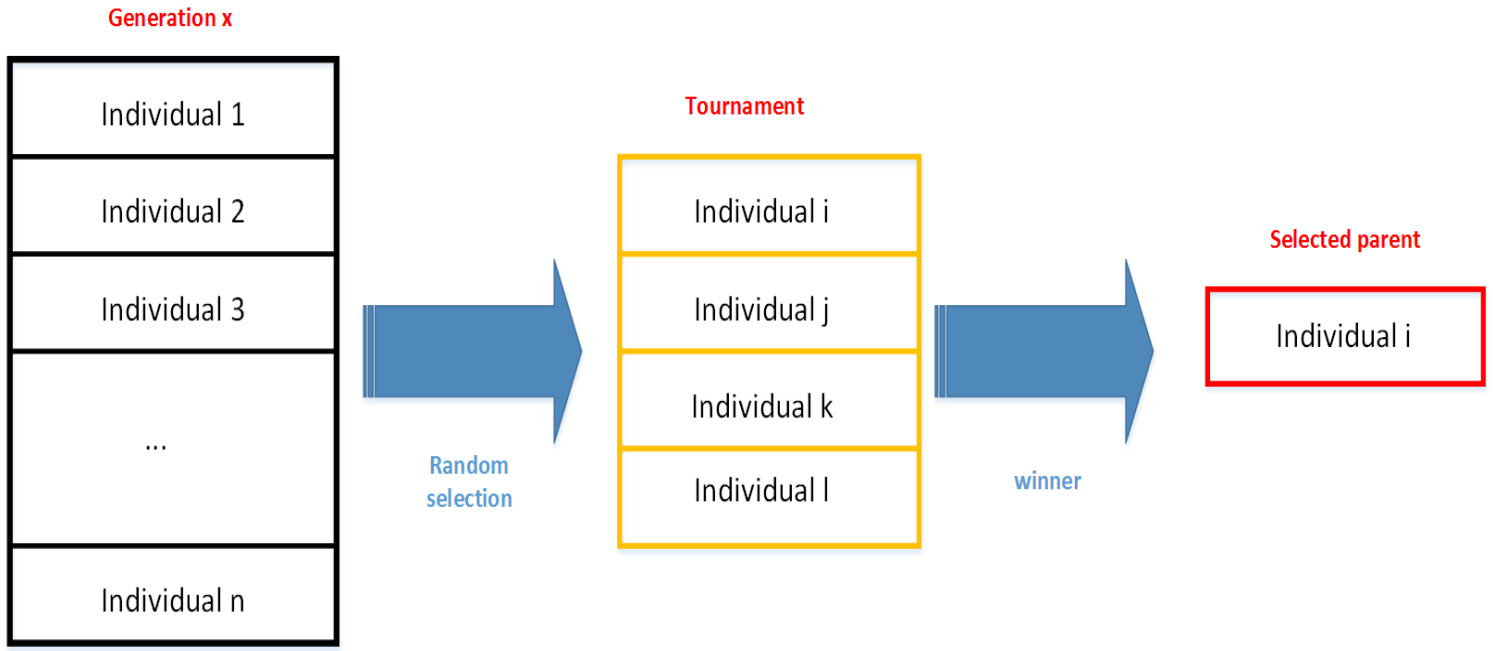
Operador — Mutación de mezcla de índices (*MutShuffleIndexes*[a]). Este operador de mutación consiste en seleccionar aleatoriamente dos genes e intercambiar mutuamente su información genética.



Problema del viajero (TSP)

Selección: ¿Cómo seleccionamos a los padres?

- Selección mediante torneo → Torneo de tamaño 3 suele funcionar bien.



Contenido día I

- Motivación.
- Optimización metaheurística versus métodos tradiciones basados en gradiente.
- Complejidad de los problemas: P Vs NP.
- Introducción a los algoritmos genéticos.
- El problema del viajero o TSP.
- Ejemplo en Python utilizando deap.

Bibliografía

- The Essential of Metaheuristics

Sean Luke
Department of Computer Science
George Mason University

Second Edition
Online Version 2.2
October, 2015

Figure 0 The Mona Lisa, estimated with the (5 + 1) Evolution Strategy. The objective is to find a set of fifty polygons which most closely approximates the original image. After Roger Alsing.

Curso que impartimos en la ESI

ACE-TI
Grupo de investigación
Ingeniería Electrónica

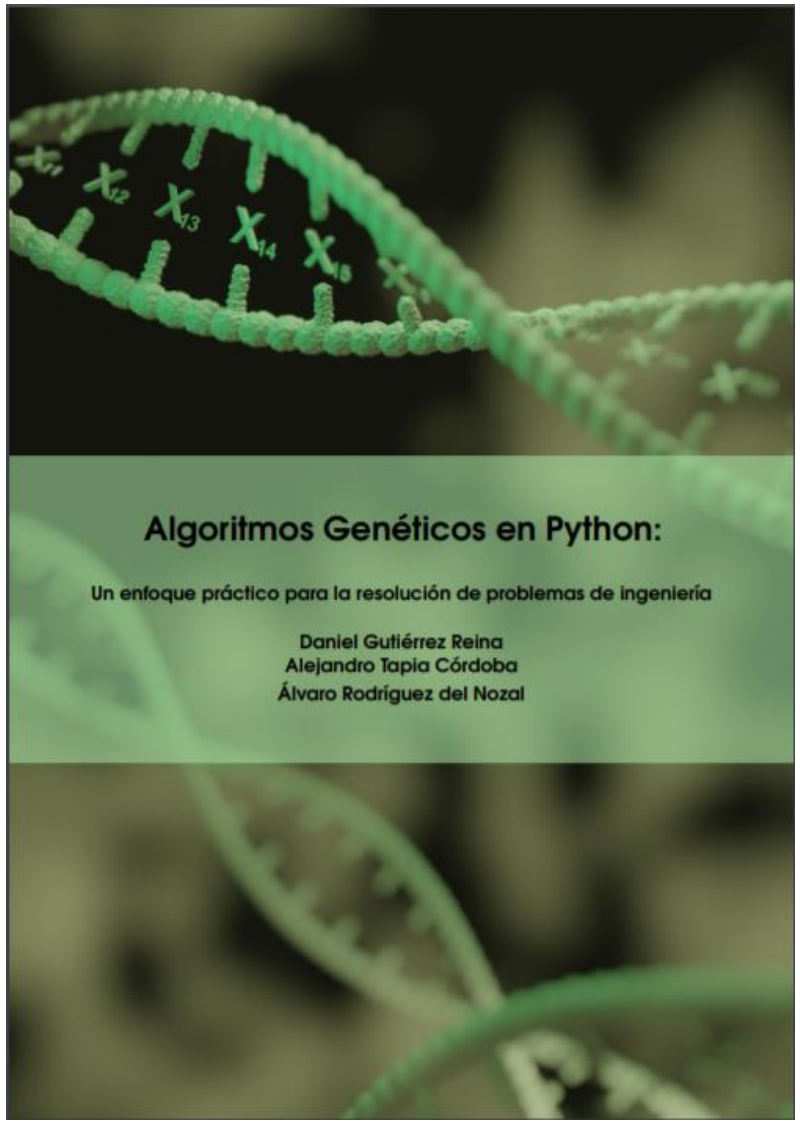
Python: Machine Learning, Optimización y Aplicaciones (IV edición)

- **Módulo I:** Conocimientos Básicos de Python y sus Módulos Principales. (20 horas).
- **Módulo 2:** Machine Learning en Python: Regresión, Clasificadores y Clustering. (16 horas).
- **Módulo 3:** Técnicas de Optimización en Python. (16 horas).
- **Módulo 4:** Deep Learning con TensorFlow y Keras. (16 horas).
- **Módulo 5:** Aplicaciones. (16 horas).

Universidad de Sevilla
CFP Centro de Formación Permanente
Vicerrectorado de Ordenación Académica

https://cfp.us.es/cursos/fc/python-machine-learning-optimizacion-y-aplicaciones/4012/

Próximamente disponible



Avisaré por twitter

ACE-TI
Grupo de investigación
Ingeniería Electrónica

MUCHAS GRACIAS

Daniel Gutiérrez Reina, dgutierrezreina@us.es