

---

# Práctica 2

Metaheurísticas

Amin Kasrou Aouam



**UNIVERSIDAD  
DE GRANADA**

2021-06-22

## Índice

<b>Práctica 2</b>	<b>3</b>
Introducción . . . . .	3
Algoritmos . . . . .	3
Genético . . . . .	3
Memético . . . . .	4
Implementación . . . . .	4
Instalación . . . . .	5
Ejecución . . . . .	5
<b>Análisis de los resultados</b>	<b>5</b>

## Práctica 2

### Introducción

En esta práctica, usaremos distintos algoritmos de búsqueda, basados en poblaciones, para resolver el problema de la máxima diversidad (MDP). Implementaremos:

- Algoritmo genético
- Algoritmo memético

### Algoritmos

#### Genético

Los algoritmos genéticos se inspiran en la evolución natural y la genética. Generan un conjunto de soluciones inicial (i.e. población), seleccionan un subconjunto de individuos sobre los cuales se opera, hacen operaciones de recombinación y mutación, y finalmente reemplazan la población anterior por una nueva.

El procedimiento general del algoritmo queda ilustrado a continuación:

---

**Input:** A list  $[a_i], i = 1, 2, \dots, n$ , that contains the population of individuals  
**Output:** Processed list

```

1  $P(t) \leftarrow initializePopulation()$   $P(t) \leftarrow evaluatePopulation()$ 
2 while  $\neg stopcondition$  do
3    $t = t + 1$ 
4    $parents \leftarrow selectParents(P(t - 1))$ 
5    $offspring \leftarrow recombine(parents)$ 
6    $offspring \leftarrow mutate(offspring)$ 
7    $P(t) \leftarrow replacePopulation(P(t - 1), offspring)$ 
8    $P(t) \leftarrow evaluatePopulation()$ 
9 end while
10 return  $P(t)$ 

```

---

Procedemos a la implementación de 4 variantes distintas, según 2 criterios:

#### 1. Criterio de reemplazamiento

- **Generacional:** la nueva población reemplaza totalmente a la población anterior
- **Estacionario:** los dos mejores hijos reemplazan los dos peores individuos en la población anterior

## 2. Operador de cruce

- **Uniforme:** mantiene las posiciones comunes de ambos padres, las demás se eligen de forma aleatoria de cada padre (requiere reparador)
- **Posición:** mantiene las posiciones comunes de ambos padres, elige el resto de elementos de cada padre y los baraja. Genera 2 hijos.

**Memético**

Los algoritmos meméticos surgen de la hibridación de un algoritmo genético, con un algoritmo de búsqueda local. El resultado es un algoritmo que posee un buen equilibrio entre exploración y explotación.

El procedimiento general del algoritmo queda ilustrado a continuación:

---

```

Input: A list  $[a_i], i = 1, 2, \dots, n$ , that contains the population of individuals
Output: Processed list
1  $P(t) \leftarrow initializePopulation()$   $P(t) \leftarrow evaluatePopulation()$ 
2 while  $\neg stopcondition$  do
3   if certainiteration then
4      $P(t) \leftarrow localSearch(P(t - 1))$ 
5   end if
6    $t = t + 1$ 
7    $parents \leftarrow selectParents(P(t - 1))$ 
8    $offspring \leftarrow recombine(parents)$ 
9    $offspring \leftarrow mutate(offspring)$ 
10   $P(t) \leftarrow replacePopulation(P(t - 1), offspring)$ 
11   $P(t) \leftarrow evaluatePopulation()$ 
12 end while
13 return  $P(t)$ 

```

---

Procedemos a la implementación de 3 variantes distintas:

- Búsqueda local sobre todos los cromosomas
- Búsqueda local sobre un subconjunto aleatorio de cromosomas
- Búsqueda local sobre un el subconjunto de los mejores cromosomas

**Implementación**

La práctica ha sido implementada en *Python*, usando las siguientes bibliotecas:

- NumPy

- Pandas

## Instalación

Para ejecutar el programa es preciso instalar Python, junto con las bibliotecas **Pandas** y **NumPy**.

Se proporciona el archivo `shell.nix` para facilitar la instalación de las dependencias, con el gestor de paquetes [Nix](#). Tras instalar la herramienta Nix, únicamente habría que ejecutar el siguiente comando en la raíz del proyecto:

```
1 nix-shell
```

## Ejecución

La ejecución del programa se realiza mediante el siguiente comando:

```
1 python src/main.py <dataset> <algoritmo> <parámetros>
```

Los parámetros posibles son:

dataset	algoritmo	parámetros
Cualquier archivo de la carpeta data	genetic	uniform/position generation/stationary
	memetic	all/random/best

También se proporciona un script que ejecuta 1 iteración de cada algoritmo, sobre cada uno de los *datasets*, y guarda los resultados en una hoja de cálculo. Se puede ejecutar mediante el siguiente comando:

```
1 python src/execution.py
```

**Nota:** se precisa instalar la biblioteca [XlsxWriter](#) para la exportación de los resultados a un archivo Excel.

## Análisis de los resultados

Desafortunadamente, debido a un tiempo de ejecución excesivamente alto (incluso tras ajustar los metaparámetros) no podemos proporcionar resultados de la ejecución de los algoritmos.