Práctica 1

Metaheurísticas

Amin Kasrou Aouam

UNIVERSIDAD
DE GRANADA

2021-04-19

Índice

Práctica 1

Introducción

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

- Algoritmo *Greedy*
- Algoritmo de búsqueda local

Algoritmos

Greedy

El algoritmo *greedy* añade de forma iterativa un punto, hasta conseguir una solución de tamaño m.

En primer lugar, seleccionamos el elemento más lejano de los demás (centroide), y lo añadimos en nuestro conjunto de elementos seleccionados. A éste, añadiremos en cada paso el elemento correspondiente según la medida del *MaxMin*. Ilustramos el algoritmo a continuación:

Input: A list $[a_i]$, $i = 1, 2, \cdots, m$, that contains the chosen point and the distance
Output: Processed list

```
1  Sel = [ ]
2  centroid ← getFurthestElement()
3  for i ← 0 to m do
4      for element in Sel do
5          closestElements = [ ]
6          closestPoint ← getClosestPoint(element)
7          closestElements.append(closestPoint)
8      end for
9      maximum ← max(closestElements)
10     Sel.append(maximum)
11 end for
12 return Sel
```

Búsqueda local

El algoritmo de búsqueda local selecciona una solución aleatoria, de tamaño *m*, y explora durante un número máximo de iteraciones soluciones vecinas.

Para mejorar la eficiencia del algoritmo, usamos la heurística del primer mejor (selección de la primera solución vecina que mejora la actual). Ilustramos el algoritmo a continuación:

Input: A list $[a_i]$, $i = 1, 2, \cdots, m$, the solution
Output: Processed list

$Solutions = [\,]$
$firstSolution \leftarrow getRandomSolution()$
$Solutions.append(firstSolution)$
$lastSolution \leftarrow getLastElement(neighbour)$
$maxIterations \leftarrow 1000$
for $i \leftarrow 0$ **to** $maxIterations$ **do**
  while $neighbour \leq lastSolution$ **do**
    $neighbour \leftarrow getNeighbouringSolution(lastSolution)$
    $Solutions.append(neighbour)$
    $lastSolution \leftarrow getLastElement(neighbour)$
  end while
  $finalSolution \leftarrow getLastElement(Solutions)$
end for
return $finalSolution$

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 herramiento Nix, únicamente habría que ejecutar el siguiente comando en la raíz del proyecto:

```
nix-shell
```

Ejecución

La ejecución de ambos algoritmos se realiza mediante el siguiente comando:

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

Los parámetros posibles son:

dataset	algoritmo
Cualquier archivo de la carpeta data	greedy
	local