

# Optimización de Flujo en Redes:

## Reporte 2

Alcantar Gómez Alan Arnoldo

Matricula: 1935040

4 de marzo de 2018

### 1. Introducción

El presente trabajo se desarrolló para introducir las herramientas de programación **Python**: *Clases y funciones*, con el objetivo de reestructurar el código empleado en el reporte 1 y tener un código más robusto y flexible.

### 2. Descripción del sistema termodinámico

Considérese un conjunto de  $N$  partículas ubicadas de forma aleatoria que están sometidas a un campo de temperatura uniforme a lo largo del eje  $x$ , y disminuye a lo largo del eje  $y$ , de este modo la temperatura de cada partícula estará dada por su posición. Por último, definir que todo el sistema se encuentra en un espacio cuadrado de 1 metro.

El radio de interacción de la partícula  $i$  se modela como la multiplicación entre su coordenada  $y(i)$  y un parámetro  $\alpha$ . Por otro lado la temperatura de la partícula  $i$ ,  $T(i)$ , se modela como la multiplicación entre su coordenada  $y(i)$  y la temperatura máxima del sistema  $T_{max}$ .

$$R(i) = y(i) * \alpha \quad (1)$$

$$T(i) = y(i) * T_{max} \quad (2)$$

De modo que la partícula  $i$  podrá interaccionar con cualquier partícula  $j$  ( $i \neq j$ ) si la distancia entre estas es igual o menor al radio de interacción de la partícula  $i$ .

$$D(i, j) \leq R(i) \quad (3)$$

### 3. Descripción del algoritmo para representar el sistema por medio de un grafo

Los parámetros necesarios para generar el grafo son  $n$  (número de partículas),  $t$  (temperatura máxima),  $a$  (parámetro  $\alpha$ ) y modo (simple, dirigido, ponderado, campechano).

Los componentes principales del código son: una *clase* llamada Grafo y dentro de ella existen tres *funciones*.

- *vertices(self,n,t)*: generar  $n$  nodos, cuyas coordenadas y temperaturas son guardadas
- *unir(self,n,a)*: genera la arista entre los nodos  $i, j$  si y solo si cumple con la condición 3
- *graficar(self,modo)*: genera el archivo de salida para **Gnuplot** de acuerdo al tipo de grafo que se desea

---

```
from math import sqrt
from random import random
class Grafo:

    def __init__(self):
        self.nodos = dict()
        self.aristas = dict()
        self.vecinos = dict()

    def vertices(self, i, t):
        for j in range(i):
            y = random()
            self.nodos[j] = (random(), y, y*t)

    def unir(self, i, alpha):
        for k in range(i-1):
            for l in range(k+1, i):
                d = sqrt((self.nodos[l][0] - self.nodos[k][0])**2 +
                        (self.nodos[l][1] - self.nodos[k][1])**2)
                if d <= self.nodos[k][1]*alpha:
                    xm = (self.nodos[l][0] + self.nodos[k][0])/2
                    ym = (self.nodos[l][1] + self.nodos[k][1])/2
                    w = (self.nodos[l][2] + self.nodos[k][2])/10
                    self.aristas[(k,l)] = (xm, ym, w)
                    if not k in self.vecinos:
                        self.vecinos[k] = set()
                    if not l in self.vecinos:
                        self.vecinos[l] = set()
                    self.vecinos[k].add(l)
                    self.vecinos[l].add(k)
```

---

## 4. Resultados

A continuación, se presentan los resultados.

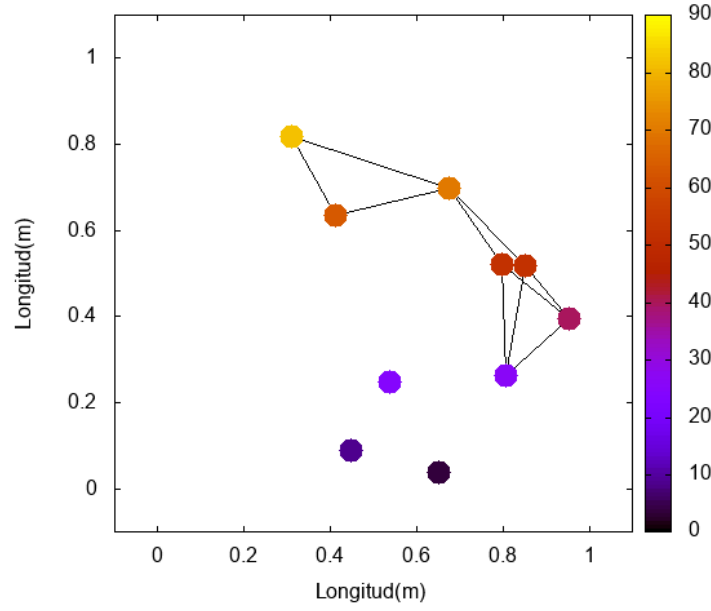


Figura 1: Gráfica de interacciones,  $N = 10$ ,  $T_{max} = 100$  y  $\alpha = 0,7$ , *modo : simple*.

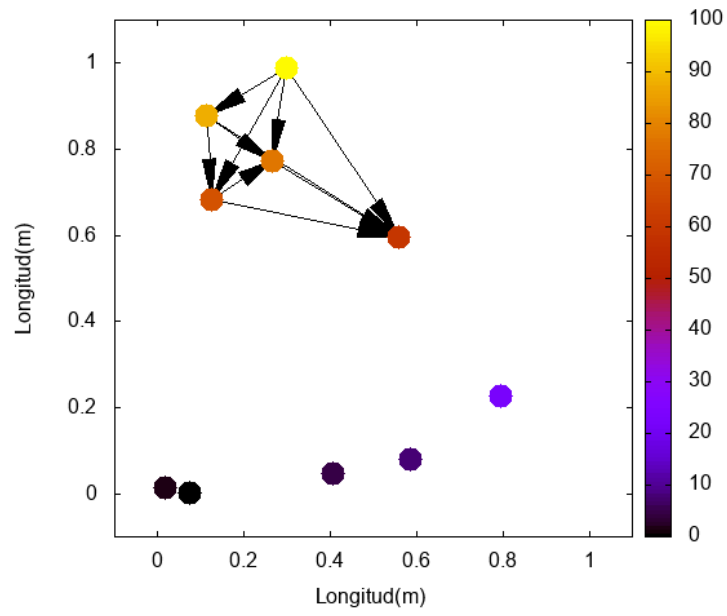


Figura 2: Gráfica de interacciones,  $N = 10$ ,  $T_{max} = 100$  y  $\alpha = 0,7$ , *modo : dirigido*.

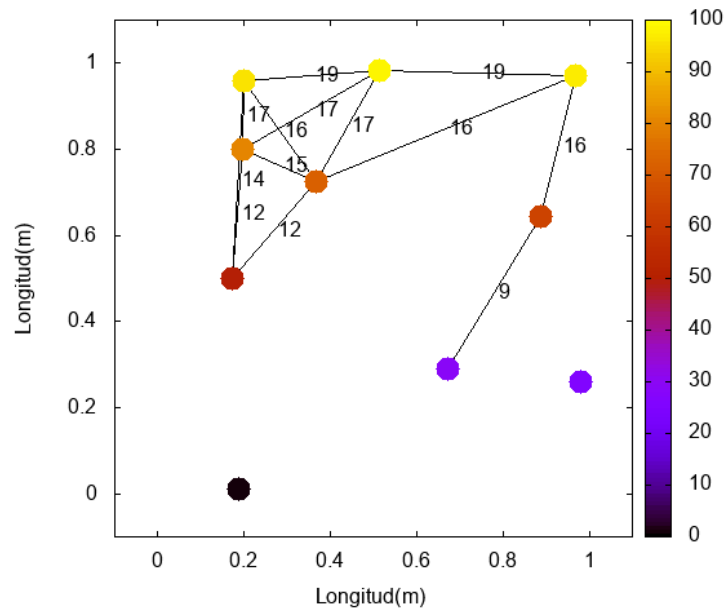


Figura 3: Gráfica de interacciones,  $N = 10$ ,  $T_{max} = 100$  y  $\alpha = 0,7$ , *modo : ponderado*.

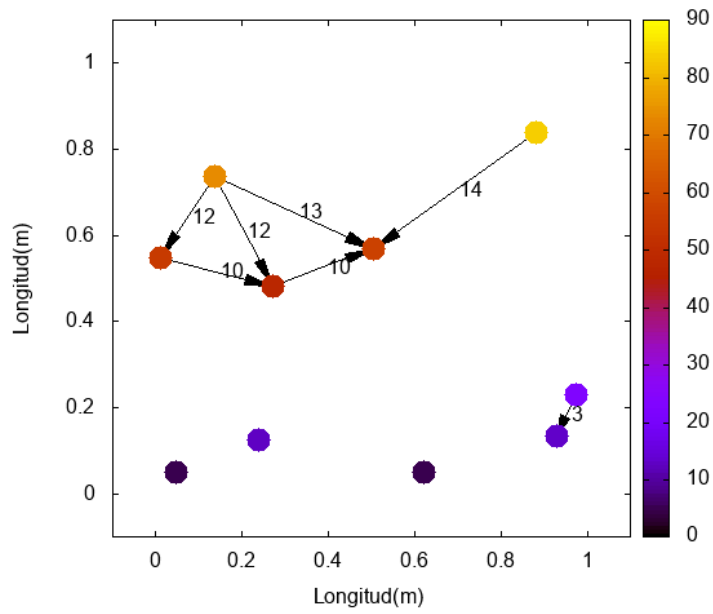


Figura 4: Gráfica de interacciones,  $N = 10$ ,  $T_{max} = 100$  y  $\alpha = 0,7$ , *modo : campechano*.

## 5. Conclusión

Como se observó en la parte de experimentación el utilizar *clases* y *funciones* permite que podamos generar varios tipos de grafos a través de los parámetros con los que trabajen cada una de las *funciones*. Esto nos permite utilizar el mismo código para resolver distintos problemas donde las direcciones o capacidades entre los nodos tengas una característica importante.