Matemáticas discretas 2020
Informe proyecto 3: Teoría de grafos

Escuela de Ingeniería Civil Informática

Universidad de Valparaíso

Autores: Maximiliano Baranda, Aaron Cárdenas, Vicente Reyes, Marco Vivar, Pedro pablo López

1. Recursos utilizados:
   - ❖ Lenguaje(s) de programación utilizado(s):
     - Python 3.8.5 o superior
   - ❖ Sistema(s) operativo(s)usado(s)
     - Windows 10 Home Single Language
   - ❖ Librería(s):
     - **Tkinter**
     - **Random**
     - **Sys**
     - Numpy
     - Networkx
     - Matplotlib.pyplot

Las librerías en **negrito** vienen por defecto en el lenguaje de programación Python 3, debido a que son parte de las librerías estándar de este lenguaje.

2. Lógica del software:

<u>Descripción de librerías:</u>

Tkinter: El Tkinter está incluido en Python como un paquete estándar, y es considerado para la interfaz gráfica de usuario (GUI).

Random: Este módulo implementa generadores de números pseudoaleatorios para varias distribuciones, también proporciona la clase **SystemRandom** la cual usa la función del sistema **os.urandom()** para generar números aleatorios a partir de fuentes proporcionadas por el sistema operativo.

sys: El módulo sys es el encargado de proveer variables y funcionalidades, directamente relacionadas con el intérprete (consola o programa).

Numpy: La biblioteca numpy da soporte para crear vectores y matrices grandes multidimensionales, junto con una gran colección de funciones matemáticas de alto nivel para operar con ellas.

Networkx: La biblioteca Networkx para el estudio de grafos y análisis de redes.

Matplotlib: biblioteca para la generación de gráficos a partir de datos contenidos en listas o arrays.

3. Diseño de la solución:

Inicialmente nos encargamos de leer y entender a pequeña escala como iba a funcionar el flujo de datos y que operaciones teníamos que hacer con los mismo. Notamos que el dato de ruido no iba a ser necesario por lo cual lo descartamos totalmente. Notamos a su vez que para poder calcular las aristas teníamos que ver a quien le respondía cada nodo, por lo que considerando que íbamos a tener por cada interacción un emisor y un receptor, se nos ocurrió generar una matriz cuadrada, que en las filas almacenara cada nodo emisor y en las columnas cada nodo receptor, de modo que cada vez que hubiere una interacción, podíamos sumar un 1 a la intersección del emisor y el receptor en la matriz, de esa forma calculamos los vectores.

Para calcular el prompting, simplemente recorrimos todos los datos de nodos, y sumamos en un vector de largo igual a la cantidad de nodos un 1 cada vez que el determinado nodo intervenía. de modo que por ejemplo el nodo uno tenía su contador en la casilla cero de tal vector. Al final teníamos un vector que nos indicaba la cantidad total de intervenciones de cada nodo a lo largo del tiempo.

por último, para lograr todo lo anterior tuvimos que descomponer los datos que nos entregaron.

leímos cada línea del archivo y en cada línea la oración cada vez que había una coma, de modo que por línea terminamos con 4 strings, las cuales en orden serían el identificador, el ruido, el nodo y por último el tiempo. de ese modo pudimos operar cada dato por separado.

4. Funcionamiento del algoritmo:

En el archivo principal se importa la librería Tkinter y las funciones del archivo Grafos_betha.py, luego se definen los parámetros para la creación de la interfaz gráfica, una vez hecho eso se procede a llamar la función main del otro archivo, en el cual, antes de ejecutar verifica si las librerías requeridas por el software están

```
try:
    # importing numpy
    import numpy as np
    # importing networkx
    import networkx as nx
    # importing matplotlib.pyplot
    import matplotlib.pyplot as plt

except:#Si no se logran importar, se procede a instalarlas en el sistema del usuario
    print("Error when importing necessary libraries  to make the program work, it will proceed to start the intallation of these\n")
    import subprocess
    subprocess.call(['pip', 'install', "-r","requeriments.txt"])
    print("Libraries were installed to execute the program\n")
```

Librerias

instaladas en el computador del usuario, si no lo están procede a instalarlas en el sistema del usuario. Luego de esto hay que ejecutar el programa nuevamente para su uso.

A continuación, se definen variables globales a ocupar y se ejecuta la función main. Se pasa por parámetros el nombre de nuestro a archivo a leer y verifica si el archivo pasado por parámetros existe, si no existe en el directorio del usuario se procederá a dar un aviso y se finalizará el programa.

```
fileName = 'Grupo1-393371.csv'
try:
    archivo = open(fileName, 'r')
    contenedor = archivo.readlines()
    archivo.close()
except:
    sys.exit(f'Error file {fileName} not found.')
```

Archivo

Luego se obtiene las cantidades de nodos totales que se intercomunican con la función numberofNodes(), por consiguiente, se llama a la función RandomColor() en el cual hay una lista con cinco colores, de modo que si la cantidad de los nodos son cinco se le asignara un color de la lista a cada nodo, sin embargo cuando la cantidad de los nodos es diferente a cinco se recorre un for de 0 hasta la cantidad de nodos en el cual asigna un color al azar desde la lista que contiene los cinco colores, sin embargo, en este caso dos o mas nodos puede tener el mismo color como identificador.

```python
def randomColor(nodeNumbers):
    arrayColors = ["green", "blue", "red", "magenta", "yellow"]
    if(nodeNumbers == len(arrayColors)):
        return arrayColors

    nodeColors = []
    for i in range(0, nodeNumbers):
        nodeColors.append(random.choice(arrayColors))
    return nodeColors
```

Función randomColor

Obtenidos los nodos, luego se procede a obtener el peso total de cada nodo, en el cual, primero se rellena una matriz cuadrada (dependerá de la cantidad de lo nodos, el tamaño de la matriz) con ceros. Obtenido la matriz se llama la función matrizAdyacencia_Prompting() en que se hace un for del largo del archivo, así obteniendo la cantidad total que habla cada nodo(peso de los nodos) y la cantidad de veces que un nodo le responde al otro (peso de las aristas). Si el tiempo transcurrido supera o es igual a los 60 seg se llama a la función calcularPromting() la cual devuelve el prompting de cada uno de los nodos en ese instante especifico, y lo guarda en una lista de matrices, para luego ser utilizada en el gráfico. Esta función hace un for de la listaPesos y dependiendo del caso, si es “1” (“Agrega a la lista el peso del índice divido la suma de todos los pesos) o “2” (Rellena un diccionario con cantidad de porcentaje de participación de cada nodo) se ejecutara el procedimiento siempre será caso 1 cuando sea llamado desde la función matrizAdyacencia_Pormpting().

```python
def matrizAdyacencia_Prompting(matriz, contenedor):
    anterior = -1
    tiempoVentanas = 60
    tiempoReal = 0
    for a in range(1, len(contenedor)):
        aux = (contenedor[a].replace('"', "")).split(",")
        listaContador[int(aux[2])] += 1
        if(anterior == -1):
            anterior = int(aux[2])
        else:
            matriz[int(aux[2])][int(anterior)] += 1
            anterior = int(aux[2])

        if( round(float(aux[3])) >= tiempoVentanas):
            tiempoVentanas += 60
            calcularPrompting(listaContador, 1, listaPromptingFinal)
```

Función MatrizAdyacencia_Promting

```
def calcularPrompting(listaPesos, option, listaPrompting):

    for i in range(len(listaPesos)):
        if(option == 1 and listaPrompting != None):
            listaPrompting[i].append(listaPesos[i]/sum(listaPesos))
        elif(option == 2):
            labelsPercents[i] = "Usuario " + str(i) + " (" + str(round((listaPesos[i]/sum(listaPesos))*100)) +"%)"
```

Función calcularPromting

Finalmente se llama nuevamente la función calcularPrompting() pero siendo esta vez caso 2.Después de finalizar la función main de grafos_betha.py se procede a ejecutar en el archivo principal la función window.mainloop(), la cual crea una interfaz de usuario grafica.

5. Interpretación de los datos

Gráfico 1:

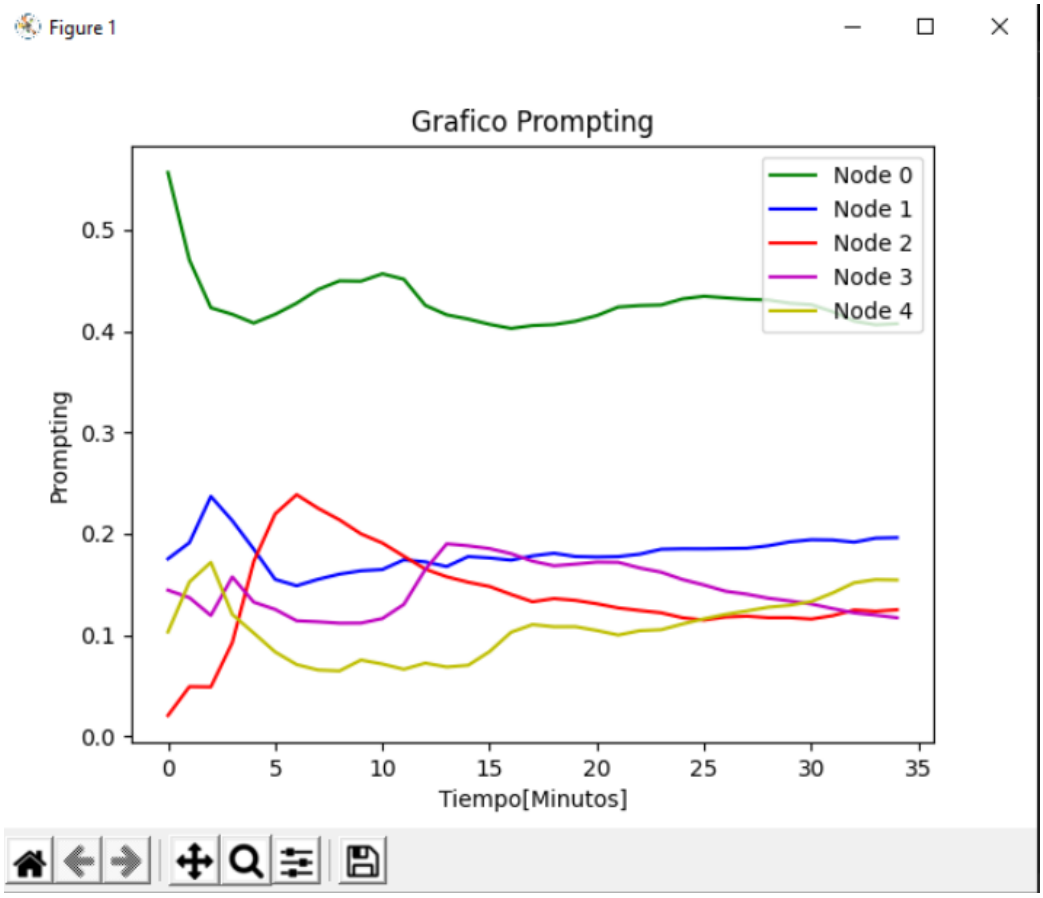

Gráfico Prompting

En este grafico se muestra la participación de cada nodo a lo largo del tiempo.

Gráfico 2:

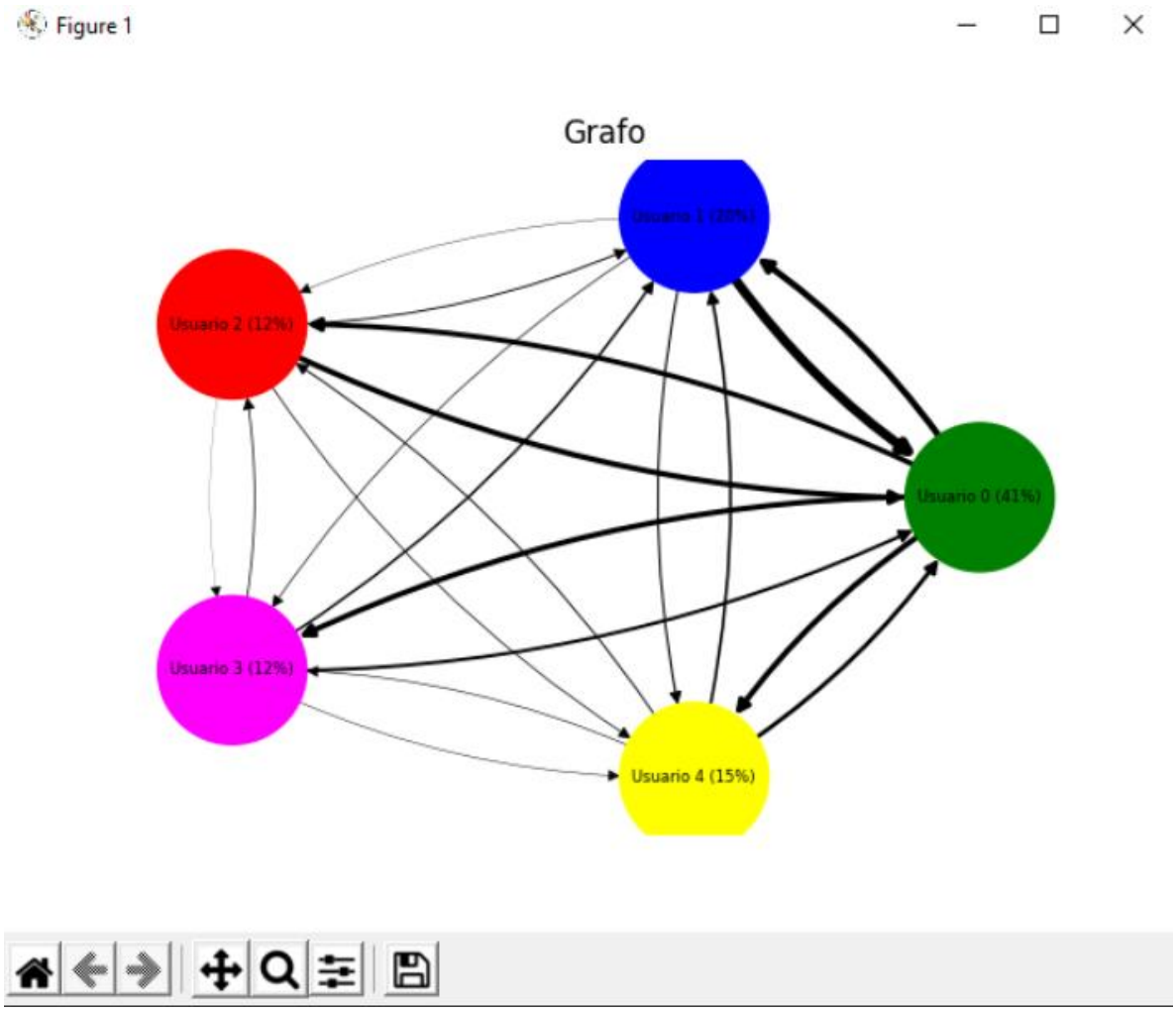

Grafico de grafo

En este grafico muestra como cada nodo esta interconectado con el otro nodo y su porcentaje de participación.