Tarea 1: Trazar grafos en Gnuplot

María Gabriela Sandoval Esquivel

Febrero 2018

1 Introducción

El objetivo de esta tarea es aprender a trazar grafos con nodos y aristas haciendo uso de la herramienta Gnuplot. Esta herramienta permite visualizar datos y funciones matemáticas de manera gráfica e interactiva. Las instrucciones a Gnuplot se hacen a través líneas de comando y se puede elegir entre varias terminales para el tipo de gráfico que deseamos producir.
En este reporte se incluyen primero las instrucciones para realizar grafos sencillos a partir de datos de un archivo de texto generado en Python. Después, se describen algunas de las opciones que hay para dar formato a estos grafos para representar de mejor manera los datos. Finalmente, se incluye un ejemplo de aplicación donde se visualizan las instancias y soluciones de un problema de diseño territorial.

2 Trazar Grafos

Para generar un grafo sencillo se necesita información sobre el posicionamiento de los nodos y las aristas que lo conforman. La posición de los nodos se puede describir, por ejemplo, con coordenadas en un plano; mientras que las aristas se pueden describir mediante las parejas de nodos que estas conectan.

Para crear ejemplos aleatorios de grafos el primer paso es generar los datos sobre nodos y aristas. Esto se puede hacer en un script de Python (ejemplo en el archivo `generar_datos.py`) donde se define el número de nodos cuya posición se generará de manera aleatoria y la probabilidad de unir una pareja de nodos con una arista. Los datos generados se pueden escribir en dos archivos con extensión ".dat" para que puedan ser usados por Gnuplot para generar el grafo. De estos archivos uno contiene las coordenadas de los nodos del grafo y el otro las parejas de nodos que estarán conectados.

Estos archivos deben de tener un formato muy sencillo y cada renglón representa un objeto diferente. Por ejemplo: para el archivo de nodos cada renglón representa la información de un nodo y las coordenadas de un mismo nodo se separan por espacio simple. (ejemplo en `nodos.dat`) Una vez que se tienen los

datos, se pueden indicar las instrucciones a Gnuplot para trazar el grafo. En el ambiente de Gnuplot lo primero que se debe de hacer es asegurarnos que el directorio en el que estamos trabajando sea el mismo en el que están guardados los archivos de datos. Después se establece la terminal del output que deseamos para el grafo. Hay varias opciones de terminales, por ejemplo, se puede elegir que simplemente se abra una ventana con el grafo, o que se genere una imagen formato PNG. La lista de algunas de las opciones de terminales está en: `http://www.gnuplotting.org/output-terminals/` (el resto en la documentación de Gnuplot) y se indica escribiendo en la ventana de comandos de Gnuplot:

```
set term <nombre de la terminal>
```

Después se nombra el archivo de output escribiendo con la extensión correspondiente a la terminal:

```
set output \nombrearchivo.png"
```

Para graficar los datos de un archivo usamos la función `plot` seguido de la indicación del archivo de donde se obtienen los datos y el tipo objeto con el que se graficarán los datos. Para este caso usamos el comando:

```
plot 'nodos.dat' with points pt 7
```

El tipo de símbolo que se usa para los puntos se indica con `'pt 7'`. Números diferentes dan símbolos diferentes. Para incluir más de un archivo de datos en una misma gráfica solo se necesita separar por comas a los diferentes archivos como en el siguiente ejemplo:

```
plot 'nodos.dat' with points pt 7, 'nodos1.dat' with points pt 6
```

Si se desea usar solo ciertas columnas del archive de datos se puede indicar de la siguiente manera para usar solo las columnas 1 y 3

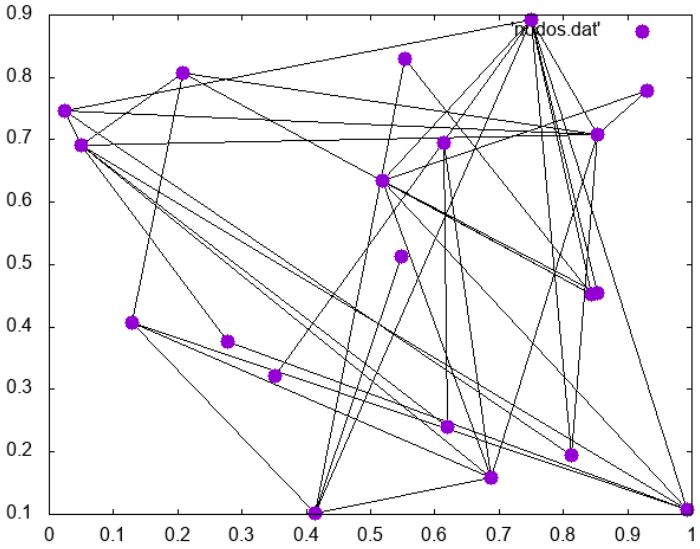
```
plot 'nodos.dat' using 1:3 with points pt 7
```

Las aristas se deben de establecer como objetos para que aparezcan en el grafo. Se establecen de la siguiente manera:

```
set arrow <arrow number> from <x1>, <y1> to <x2>, <y2> nohead
```

Donde `<arrow number>` será un índice para cada arista (se recomienda definir de manera sucesiva), se deben de indicar también las coordenadas de los nodos que une cada arista. La indicación `'nohead'` hace que las aristas no sean dirigidas. Si no se incluye, estas tendrán forma de flecha. Una vez que se definan todas las aristas se vuelve a dar la indicación de plot para que en el grafo aparezcan las aristas.

Para agilizar el proceso de generar el grafo se puede programar en Python el proceso para crear un archivo que pueda leer Gnuplot para realizar el grafo a partir de los archivos de datos. En el archivo `grafo_general.py` encontramos este programa que genera un archivo con extención `.plt`. Este último archivo se carga en Gnuplot al escribir: `load 'grafo_general.plt'`



3 Formato

Hasta ahora se tiene un grafo sencillo al que se le pueden agregar especificaciones del formato tanto como para los nodos como para las aristas.

3.1 Nodos

Para cambiar el formato simple de los nodos se usan las siguientes indicaciones:

- Tamaño: `set pointsize 2`
- Color: `plot 'nodos.dat' with points pt 7 palette cb 3`

Las indicaciones anteriores hacen que todos los nodos tengan el mismo color y el mismo tamaño. Sin embargo, podemos hacer que el tamaño y el color de los nodos describa ciertas características de los datos. Una manera de hacer que el tamaño de los nodos dependa de una tercera columna de datos, por ejemplo, es usar el objeto circles en lugar de points para graficar los nodos. La instrucción es la siguiente y la figura resultante es la 3.1

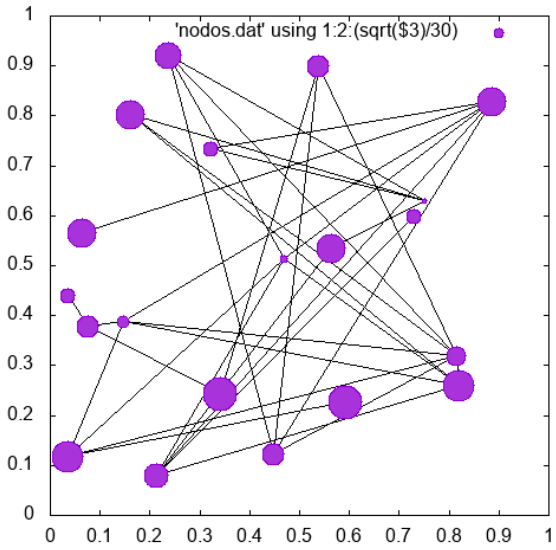
```
set style fill transparent solid 0.8
plot 'nsize.dat' using 1:2:(sqrt(\$3/50) ) with circles
```

Se pueden ajustar los valores de tamaño según convenga como lo indica la expresión `sqrt($3/50)` en la instrucción anterior.

Para cambiar el color de los nodos de acuerdo con una tercera columna de datos se define primero la paleta que se desea usar, por ejemplo:

```
set palette defined (0 'blue', 3 'green', 6 'yellow', 10 'red')
plot 'nodosr.dat' using 1:2:3 with points pt 7 palette
```

Figure 1: Ejemplo de grafo con nodos de diferente tamaño



Para combinar tamaños y colores que dependan de una tercera y cuarta columna respectivamente se puede hacer con la siguiente instrucción:

```
plot ‘nodosr.dat’ using 1:2:(sqrt(\$3/50) ):4 with circles palette
```

3.2 Aristas

El formato que se le puede dar a las aristas es definir indentación, con la siguiente instrucción:

```
set arrow 1 dashtype 2
```

También se puede cambiar el grosor de las aristas de la siguiente manera:

```
set style arrow 1 linewidth 3
```

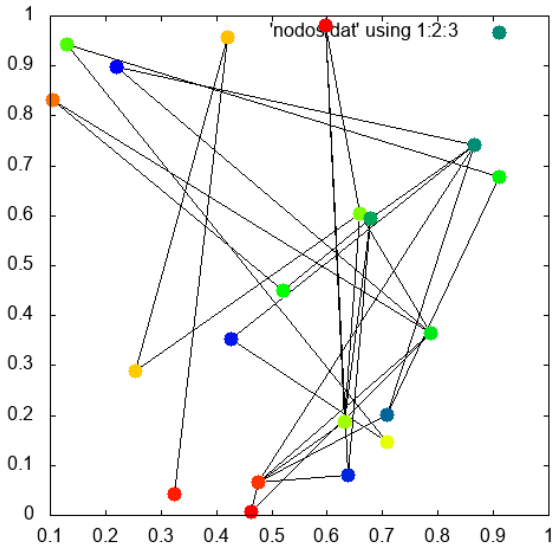
Para cambiar el color:

```
Set arrow 1 linecolor ‘blue’
```

3.3 Ventana

- Quitar título: `plot 'nodos.dat' with points pt 7 notitle`
- Quitar la caja de color: `unset color box`
- Definir el rango de los ejes:
  ```
  set xrange [0,200]
  set yrange [0,200]
  ```

Figure 2: Ejemplo de grafo coloreado



4 Aplicación

Al aprender sobre esta herramienta me di cuenta de que me puede ser muy útil para visualizar instancias y soluciones del problema en el que estoy trabajando para mi tesis. Se trata de un problema de diseño territorial en el que se busca dividir un área geográfica en territorios de acuerdo con ciertos criterios de planeación.
La formulación matemática para este problema considera al área geográfica como un conjunto de nodos que están conectados. Los nodos representan unidades básicas que al agruparse forman territorios, pueden ser por ejemplo colonias, municipios o manzanas. Las unidades básicas tienen un cierto tamaño de acuerdo con medidas de actividad como número de habitantes o capacidad económica. Los nodos se conectan por aristas si las unidades básicas son vecinos geográficos debido a que se busca que los territorios que se formen sean contiguos y compactos.
Hice un programa en Python que usa los datos que tengo para mis instancias para crear un archivo con las instrucciones necesarias para que Gnuplot grafique las unidades básicas en el plano con círculos que representan el tamaño del distrito y los nodos indican vecindad. (Los archivos correspondientes son: instancias.py e instancias.plt)
Las figuras 3 y 4 muestran ejemplos de las gráficas de las instancias que uso para hacer las pruebas de mi algoritmo.

Las soluciones que se obtienen indican a qué territorio pertenece cada unidad básica. Pude usar esta información para que el color de cada nodo represente el territorio al que pertenece.El programa de Python que usa información de

Figure 3: Instance 2DU60-05-2

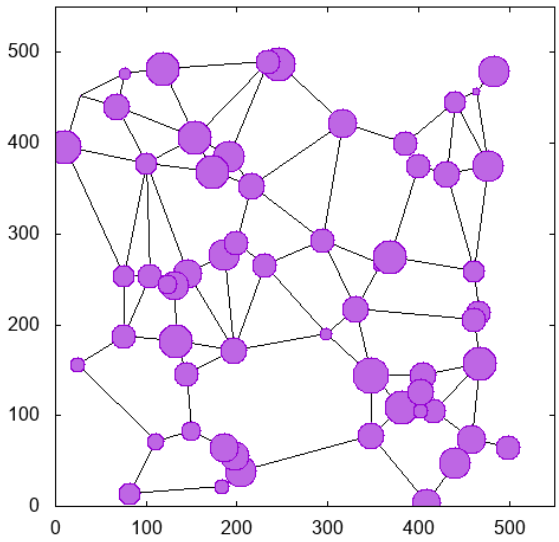


Figure 4: Instance 2DU60-05-3

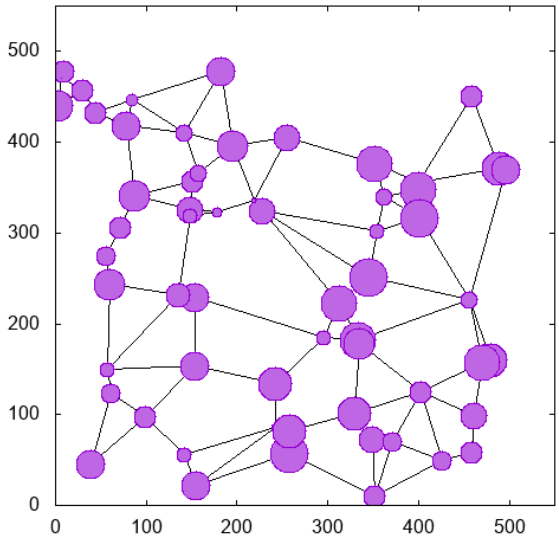
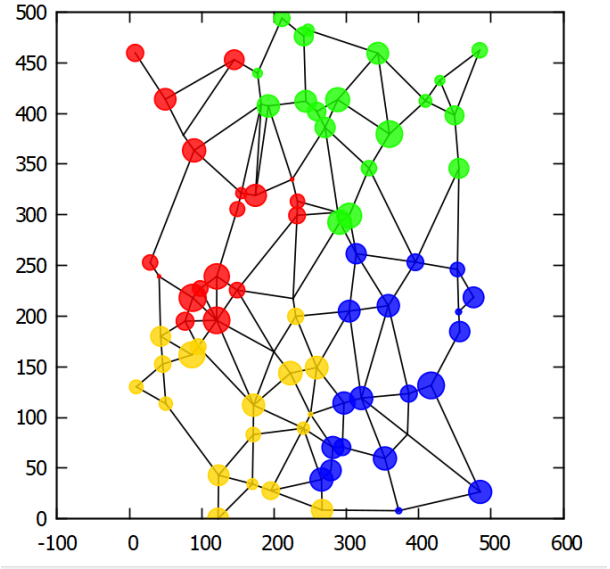


Figure 5: Solution 2DU80-05-1



las instancias y las soluciones para generar un grafo con nodos de tamaños diferentes (según la actividad) y colores diferentes (según al territorio al cual son asignados) se muestra en el apéndice de este reporte. En la figura 5 se muestra un ejemplo de la gráfica de una solución.

5 Conclusión

Con esta actividad pude aprender a usar varias funciones de Gnuplot para trazar grafos con características especiales. También descubrí que es útil generar los archivos con instrucciones para Gnuplot mediante un programa en Python. Esto me resulta especialmente útil para analizar los reultados de los algoritmos que estoy desarrollando para mi tema de tesis. En especial será útil para visualizar las soluciones parciales que resultan en cada fase de los algoritmos iterativos. De esta manera será fácil analizar cómo evolucionan las soluciones y el tipo de cortes que se están realizando. En un principio me resultó dificil adaptarme al nuevo ambiente de trabajo con estas herramientas, pero su utilidad hacen que valga la pena.

6 Apéndice: Código para trazar soluciones

```
n = 80
p = 4
center = []

basicUnit = []
neighbors = []
territory = []
terr = [None]*n
BU_file = "psol80-05-2.dat"

with open("2DU80-05-2.dat",'r') as archivo:
        n = int(archivo.readline())
        for i in range(n):
            stringLine = archivo.readline()
            splitLine = stringLine.split(" ")
            valueList = [float(e) for e in splitLine]
            index, x, y, a, b, c = valueList
            basicUnit.append((x,y,a))
        neigh = int(archivo.readline())
        for i in range(neigh):
            stringLine = archivo.readline()
            splitLine = stringLine.split(" ")
            valueList = [int(e) for e in splitLine]
            neigh1, neigh2 = valueList
            neighbors.append((neigh1,neigh2))

archivo.close()

with open("sol80-05-2.txt",'r') as solucion:
    for i in range(p):
        c = int(solucion.readline())
        center.append(c)
    for i in range(p):
        string_terr = solucion.readline()
        splitLine = string_terr.split(" ")
        splitLine = splitLine[0:(len(splitLine)-1)]
        valueList = [int(e) for e in splitLine]
        territory.append(valueList)
        for j in range(len(splitLine)):
            idx = valueList[j]
            #print(idx)
            terr[idx-1]= i+1
solucion.close()
```

```
with open("psol80-05-2.dat",'w') as psolucion:
    for i in range(n):
        x,y,size= basicUnit[i]
        print(x,y,size,terr[i], file = psolucion)
psolucion.close()

with open("solution.plt",'a') as aristas:
        print("set term png", file = aristas)
        print("set output 'psol.png' ", file = aristas)
        print("set pointsize 1", file = aristas)
        print("set size square", file = aristas)
        print("set style fill transparent solid 0.8", file = aristas)
        print("set palette defined (0 'blue', 3 'green', 6 'yellow', 10 'red') "
        print("unset colorbox", file = aristas)
        num = 1
        for i in range(neigh):
            neigh1, neigh2 = neighbors[i]
            x1, y1, s1 = basicUnit[neigh1]
            x2, y2, s2 = basicUnit[neigh2]
            print("set arrow", num, "from", x1, "," ,y1,"to", x2, ",", y2, "no
            num +=1

        print("plot 'psol80-05-2.dat' using 1:2:(2*sqrt(\$3-600)):4 with circles
        print("unset arrow", file = aristas)
        print("unset output", file = aristas)
aristas.close()
```