Memoria de la práctica 3

Jaime Lorenzo Sánchez

Francisco Cruceira Lloret

Alejandro Del Moral Rodríguez

Francisco Julián Reyes Urbano

11 de febrero de 2022

Índice general

Capítulo 1

Descripción del problema

El objetivo del algoritmo es extraer grafos con restricciones temporales de manera que se cumplan las siguientes condiciones:

1. Mejor solución: Representa el número de registros que satisface el grafo.
2. Cantidad de eventos incluidos en la solución: Representa el número de nodos del grafos. Debe ser mínimo 4 eventos.
3. Menor rango de restricciones temporales.
4. Tiempo de cómputo.

Por tanto, el grafo óptimo será aquel que cumpla las siguientes condiciones:

1. Satisface el mayor número de registros.
2. Posee un mayor número de eventos.
3. Posee el menor rango de restricciones temporales.

Capítulo 2

Acciones necesarias antes de comenzar a utilizar el algoritmo

2.1. Argumentos de entrada

Primero, se comprobará que se han introducido los siguientes argumentos:

1. Nombre del script.
2. Fichero de entrada.
3. Número de individuos de la población.
4. Número de generaciones.
5. Número máximo de nodos del grafo
6. Valor K de selección por torneo
7. Valor de la probabilidad de cruce
8. Valor de la probabilidad de cruce

En caso de no haber introducido alguno de dichos argumentos, se le notificará al usuario los argumentos a indicar en la ejecución del programa y se cerrará el programa.

Una vez introducidos todos los argumentos de entrada, se obtendrán los registros de la base de datos.

2.2. Obtención y formato de los registros

Para obtener los registros necesarios para la evaluación de nuestros grafos se ha de procesar un fichero de entrada en la que cada linea representa un registro y las cuales tienen el siguiente formato:

Tipo de evento : **Tiempo del evento**

En este caso los los tipos de evento van desde la **A** hasta la **E** pudiendo haber mas de un evento del mismo tipo en un registro determinado.

En un paso previo a la lectura de nuestro fichero para nuestro script de python principal, se utiliza un script auxiliar para leer linea a linea , es decir, registro a registro nuestro fichero para almacenarlo en una lista que nos permite poder aplicar un ordenamiento de los registros.Una vez ordenados los registros se procede a guardarlos en un fichero el cual será el apropiado para procesarlo en nuestro script principal.

En nuestro script principal de python abrimos el fichero obtenido tras el ordenamiento y por cada linea del fichero creamos un diccionario que contiene los tipos de eventos,los valores de tiempo de cada evento y el número de nodos o eventos con el siguiente formato:

Los tipos de eventos son representados con valores enteros los cuales componen la clave de cada diccionario y tienen la siguiente correspondencia :

- **A ->1**
- **B ->2**
- **C ->3**
- **D ->4**
- **E ->5**

Cada evento en concreto se representa mediante su valor de tiempo como valor de la clave de su tipo de evento correspondiente, por tanto cada tipo de evento o clave de cada diccionario contendrá como valores una lista con tantos elementos como eventos de ese tipo tenga el registro.Por ejemplo:

Registro X ->A:3 A:2 B:5 C:3 B:4 B:7 E:11
Diccionario ->{1:[3,2],2:[5,4,7],3:[3],5:[11],'Nnodos':7}

Por ultimo, cada uno de estos diccionarios son guardados en una lista la cual se utilizará a la hora de evaluar cada uno de los grafos generados por nuestro script de python.

2.3. Representación de los grafos

Para representar cada uno de los grafos que se generarán se ha procedido a la creación de una clase que posee las siguientes características:

Atributos de la clase

- **tipoEvento**: Se trata de una lista de 5 posiciones la cual contiene el número de eventos de cada tipo que posee nuestro grafo.
- **relaciones**: Se trata de una lista que almacena cada una de las relaciones de los nodos o eventos de nuestro grafo.La cual poseerá el siguiente formato: **[NodoX,NodoY][Tiempo Mínimo,Tiempo Máximo]** siendo NodoX y NodoY el valor del tipo de evento más el número de evento de ese tipo divido entre 100, es decir, si tenemos la siguiente relación:
  - **[1.01,1.02] [1,5]**

  El 1.01 indica que se trata de un nodo tipo 1 por la parte entera (un evento A) y el cual es identificado como el primero de ese tipo de nodo o evento mediante el valor centesimal de 0.01.
- **nRegistros**: Contiene el número de registros validos que cumple el grafo.
- **nNodo**: Contiene el número de nodos de nuestro grafo.

Métodos de la clase

- **addNodo**: Recibe por parámetro el tipo de nodo (1..5) que se desea añadir al grafo y se actualiza los atributos de la clase según sea este.

- **creaRelaciones**: Una vez añadidos todos los nodos o eventos necesarios a nuestro grafo, este método nos permite crear las relaciones entre cada uno de los nodos o eventos.

- **evaluarGrafo**: Una vez se hayan creado las relaciones del grafo, este método nos permite nos permite calcular el número de registros que son validos para nuestro grafo.Para ello es necesario pasarle por parámetro los registros que han sido procesados previamente de nuestro fichero de entrada.

- **calcularSubconjuntos**: Recibe por parámetro un registro determinado y nos permite calcular todos los posibles subconjuntos de ese registro para el grafo.´

- **getRelaciones**: Nos devuelve la lista que contiene las relaciones entre los nodos del grafo.

- **setRelaciones**: Recibe por parámetros una lista que nos permite modificar las relaciones que poseen los nodos del grafo entre si.

- **getFitness**: Nos devuelve el valor de fitness de nuestro grafo , el cual es calculado como un 90 % el ratio (el número de registros validos con repescto al total de registros) de registros validos, un 5 % el número de nodos del grafo (como máximo tendra 16 nodos) y un 5 % el valor normalizado del rango medio de tiempos de las relaciones del grafo.En caso de tratarse de un grafo cuyo número de nodos es inferior a 4 o superior a 16 , su valor de fitness es 0.

2.4. Creación de la población inicial

Para crear la población inicial necesitaremos para cada individuo de la población (es decir, para cada grafo a crear), crear los nodos del grafo y las relaciones entre sus nodos.

2.4.1. Nodos del grafo

Se crearán los nodos del grafo de manera aleatoria, siendo 4 el número mínimo de nodos del grafo y 16 el número máximo de nodos del grafo.

Una vez obtenido el número de nodos del grafo, se procederá a asignar los tipos de los nodos (A,B,C,D,E)

2.4.2. Tipos de nodos

La creación de los tipos de nodos se realizará de manera aleatoria entre [1,5], siendo los tipos de nodos A,B,C,D y E, respectivamente.

Una vez creados los nodos, se procederá a crear las relaciones existentes entre dichos nodos.

2.4.3. Relaciones entre los nodos

Antes de crear las relacione se comprobara que el número de nodos del grafo no sea inferior a 4 ni superior a 16.En caso contrario se descartará el proceso de creación de las relaciones.

Para crear las relaciones entre los nodos, utilizaremos los siguientes pasos:

1. Realizamos una copia de los nodos disponibles de nuestro grafo.
2. Seleccionamos un nodo de dicha copia de manera aleatoria.
3. Seleccionamos el numero de relaciones de dicho nodo de manera aleatoria.
4. Eliminamos el nodo seleccionado de nuestra copia, para evitar que dicho nodo se relacione consigo mismo.
5. Seleccionamos de manera aleatoria los nodos relacionados con el nodo anterior.

6. Generamos el manera aleatoria los tiempos máximo y mínimo de las relaciones del nodo entre [1,15] y [0,-10] respectivamente.

7. Añadimos las relaciones del nodo al vector de relaciones, comprobando que 2 nodos sólo estén relacionados una vez.

8. Almacenamos las relaciones en un vector copia para evitar problemas de referencia al almacenar las relaciones en el vector de relaciones.

9. Se eliminan las relaciones del vector de relaciones para evitar la generación de relaciones repetidas.

10. Se repite el proceso anterior hasta que todos los nodos estén relacionados.

11. Una vez que todos los nodos del grafo están relacionados, se almacenan las relaciones del vector copia en el vector de relaciones.

Una vez creadas las relaciones entre los nodos, evaluamos el grafo creado.

2.4.4. Evaluación del grafo

Para evaluar el grafo primeramente se ha de comprobar que el número de nodos del grafo no sea inferior a 4 o superior a 16.En caso contrario se descarta el proceso de evaluación para ese grafo.

Acto seguido debemos comprobar para cada registro del fichero de la base de datos lo siguiente:

1. Registro actual es igual al registro anterior.

2. Comprobamos si el registro posee los nodos del grafo a evaluar.

3. Comprobamos si el registro cumple las relaciones del grafo a evaluar.

Si el registro actual es igual al registro anterior, comprobamos si dicho registro posee un subconjunto válido, es decir, si podemos crear el grafo a evaluar a partir de la información aportada por el registro.

Si se cumple que el registro actual posee un subconjunto válido, aumentamos el número

de registros que cumple el grafo a evaluar.

Si el registro actual no es igual al registro anterior, procedemos a evaluar si podemos crear el grafo utilizando el registro actual.

Para ello, debemos comprobar si el registro puede crear los nodos y sus relaciones de nuestro grafo.

Para comprobar si la evaluación de las relaciones es correcta, debemos comprobar lo siguiente:

1. El número de nodos del registro debe ser mayor o igual al número de nodos del grafo.

2. Cada nodo de nuestro grafo debe estar localizado al menos una vez en el registro.

Si la evaluación de las relaciones es correcta, procedemos guardar en una lista los posibles subconjuntos del registro.

Una vez obtenidos los subconjuntos del registro, comprobamos existe algún subconjunto válido.

Para comprobar si un subconjunto es válido, se debe cumplir que la relación entre 2 nodos del subconjunto cumple la relación entre dichos nodos en el grafo a evaluar.

Una vez evaluado el grafo, guardamos en un vector el grafo evaluado y su valor de fitness y añadimos dicho vector a la población.

Una vez obtenida la población, la ordenamos y guardamos como población élite los 5 primeros individuos de la población obtenidos anteriormente.

Para cada generación de la población, aplicamos el algoritmo.

Capítulo 3

Algoritmo

3.1. Descripción del algoritmo utilizado

Antes de decidir el algoritmo a utilizar, primero debemos estudiar el problema. Tenemos que generar el mejor grafo posible utilizando un número de registros elevado.

Como tenemos que tener en cuenta que el tiempo de cómputo del algoritmo debe ser el menor posible, la creación del grafo debe ser de manera aleatoria (este método tiene un tiempo de cómputo menor a pedir e introducir los datos del grafo por teclado).

Por tanto, vamos a emplear un algoritmo genético con modelo generacional y elitismo, en el cuál sobreviven los mejores individuos (grafos). Además, emplearemos un método de reemplazo de la población, en el cuál cada generación reemplazará a la generación anterior.

Los operadores genéticos a utilizar son los siguientes:

1. Selección de la población por torneo de tamaño
2. Probabilidad de cruce
3. Probabilidad de mutación
4. Numero de individuos
5. Numero de generaciones

3.2. Funcionamiento del algoritmo

Una vez creada la población inicial, aplicaremos los operadores genéticos a cada generación de la población.

Para ello, realizaremos los siguientes pasos:

1. Selección de los padres de la población por torneo.
2. Cruce de los padres.
3. Mutación de las relaciones de los eventos de los padres cruzados.
4. Reemplazo de la población.

3.2.1. Selección por torneo

Para seleccionar los padres candidatos por torneo, realizaremos una selección aleatoria de k padres de la población.

Una vez seleccionados, ordenaremos los padres candidatos de manera ascendente en función de su valor de fitness.

Una vez ordenados los padres, seleccionaremos el mejor padre candidato.

3.2.2. Cruce de los padres

Una vez seleccionados los padres a través del torneo, se procede a comprobar si se dará el cruce de cada uno de los padres dos a dos:

En caso de darse el proceso de cruce, se realizarán los siguientes pasos:

1. Se obtienen la listas o vectores de tipos de evento de cada grafo, es decir, la información que contiene cuantos eventos de cada tipo posee cada grafo.
2. Seguidamente se procede a generar un punto aleatorio de cruce entre ambas listas de los tipos de evento.
3. Se realiza el cruce de ambos tipos de eventos de los dos grafos en ese punto, generando así dos nuevas listas que contienen otra diversidad de los nodos de los grafos.

4. Se actualizan las listas actuales de tipos de eventos de ambos grafos por las obtenidas tras el cruce.

5. Se actualizan los valores de los números de nodos de ambos grafos.

6. Se generan nuevas relaciones para los grafos obtenidos en el cruce,

7. Finalmente se evalúan los nuevos grafos generados a partir del cruce.

Este tipo de cruce nos permite aumentar la diversidad de los individuos de la población, evitando así que se produzca una excesiva intensificación.

3.2.3. Mutación de los padres

En este caso el proceso de mutación consiste en la reducción a la mitad de los rangos de tiempos máximo y mínimos de cada relación del grafo.Es decir, si inicialmente tenemos la siguiente relación:

Relaciones Inicialmente:
[1.01,1.02][-5,6] [1.02,2.01][0,6][2.01,5.01][-1,8][3.01,5.01][-2,7]

Relaciones tras la mutación:
[1.01,1.02][-2,3] [1.02,2.01][0,3][2.01,5.01][0,4][3.01,5.01][-1,3]

Esto hace que los grafos se vuelvan más restrictivos en cuanto a los valores de tiempo de sus relaciones y por tanto aumenta su valor de fitness (en el caso de mantener el mismo número de registros validos) al poseer un valor medio de tiempos menor.

Capítulo 4

Mejores resultados obtenidos

La mejor solución obtenida tiene las siguientes características poblacionales:

1. Selección de la población por torneo de tamaño: K = 3
2. Probabilidad de cruce: cProb = 0.7
3. Probabilidad de mutación: mProb = 0.3
4. Número de individuos: 40
5. Número de generaciones: 30

A continuación, se muestran los datos obtenidos tras un número elevado de iteraciones:

Numero Nodos	Fitness obtenido	Número de registros	Tiempo de cómputo
4	0.6469	656	65.26 segundos

Cuadro 4.1: Tabla1

Diversidad eventos	Relaciones
A:1, B:1, C:2, D:0, E:0	[A->C2],[0,3];[C2->B],[-2,0];[C1->B],[-2,2]

Cuadro 4.2: Tabla2