PROBLEMA DEL COLOREADO DE GRAFOS

Para cualquier grafo y número de colores

David Iglesias Dominguez
Estudiante de Ingeniería Informática de Software
Inteligencia Artificial - Tercer Curso
España, Sevilla
did2505@gmail.com

Uvus: **davigldom**
Dni: 44244847T

Joaquín García Macías
Estudiante de Ingeniería Informática de Software
Inteligencia Artificial - Tercer Curso
España, Sevilla
donjoaquingarcia@gmail.com

Uvus: **joagarmac**
Dni: 30262158T

I. INTRODUCCION

Este documento contiene los diferentes contenidos teóricos que el grupo ha contemplado durante el desarrollo del proyecto, para poder resolver el problema del coloreado de grafos mediante técnicas de mutación y enfriamiento simulado, dado un número de colores y un grafo.

A lo largo del desarrollo, se mostrarán las diferentes técnicas empleadas para resolver el problema, junto con trozos de código o pseudocódigo si fuese necesario, capturas u otros elementos que ayuden a esclarecer y demostrar los puntos a los que se ha llegado durante la realización de proyecto.

Se incluirán pruebas de los diferentes grafos empleados por el grupo y las técnicas y valores usados para resolver sus problemas, pudiendo así obtener diferentes tablas que permitan comparar y llegar a conclusiones sobre la efectividad de un método sobre el otro, y los valores óptimos para obtener las soluciones.

Del mismo modo, se tratará de mencionar toda la bibliografía consultada durante el transcurso del proyecto, haciendo alusiones en los apartados necesarios para que el lector pueda comprobar y revisar aquellos elementos de los que se hable en un determinado apartado.

Junto con este documento, se entregarán dos notebooks para poder usarlos y abrirlos en Jupyter, que contendrán todo el código del proyecto y la interfaz gráfica. También se incluirá un txt con información acerca del uso de la interfaz, y otro txt con los grafos utilizados para las pruebas.

II. ENTORNO DE DESARROLLO

A. Jupyter Notebook

Para poder comenzar, es necesario tener iniciado el entorno Jupyter e importar los dos notebooks asociados en formato ipynb. Estos contendrán todo el código, junto con anotaciones y explicaciones paso a paso del desarrollo del mismo, y ejemplos donde se pueda ver de forma directa el funcionamiento de los diferentes métodos desarrollados.

Para poder seguir el desarrollo y la línea de trabajo del grupo, basta con ejecutar paso a paso las celdas del notebook.

III. LIBRERIAS

Para poder utilizar correctamente el código desarrollado, es necesario importar previamente las librerías usadas durante el código aportado.

Las librerías que se usarán durante el desarrollo de la práctica son:

- Random
- Deap
- Numpy
- Networkx
- Matplotlib
- Simanneal
- Tkinter
- PIL

Todas estas librerías deben estar correctamente instaladas en sus versiones de Python3 para el correcto funcionamiento del código, en caso contrario, se pueden producir excepciones no asociadas al proyecto mismo.

Para su instalación, deberá comprobar el sistema operativo del entorno donde desee utilizar el proyecto y buscar las instrucciones correspondientes al mismo.

De forma genérica, se puede proceder a la instalación de las librerías en sistemas Linux/MacOS mediante:

Pip3 install <Inserte nombre librería>

Del mismo modo, en Windows puede usarse un comando parecido, desde la consola de Anaconda.

Estas librerías serán usadas para resolver el problema mediante algoritmos genéticos, con mutación y enfriamiento simulado, dibujar el grafo resultante y obtener la interfaz gráfica que permita interactuar con la aplicación.

IV. GRAFO

Para poder resolver el problema, el primer paso a realizar es declarar la variable del grafo que se vaya a utilizar.

En este problema, se utilizarán los diferentes grafos que ofrece la librería networkx, pudiendo, además, configurar en

la interfaz gráfica, que tipo de grafo queremos usar, y los elementos esenciales del mismo.

V. INICIALIZAR LA POBLACION

Para inicializar la población del problema, se utiliza un número dado por el usuario de individuos que contendrá dicha población.

Cada individuo de la población, del mismo modo, será generado aleatoriamente en base al número de colores con el que se quiera conseguir el coloreado del grafo, de modo que los individuos serán listas de números enteros (1,2,3…) que representen el color del nodo, y el índice de cada uno, el nodo en cuestión al que hacen referencia en el grafo.

Evolución y problemas durante la implementación

Durante la implementación de esta fase, se ha partido de una población fija de 50 individuos con un numero de 3 colores para ellos, partiendo siempre desde la misma semilla, con el objetivo de conseguir aplicar correctamente el resto de los pasos del algoritmo genético.

Una vez conseguido para este caso de prueba cerrado, se ha ampliado para diferentes ejemplos de población y colores, con el objetivo de confirmar la correcta implementación del algoritmo.

VI. GENES

Para resolver un problema de algoritmos genéticos, se han de definir los genes de los individuos de la población. Estos, para el problema del coloreado de grafos, serán números enteros desde 1 hasta n, que representarán el color del nodo.

VII. FENOTIPO

El fenotipo del problema será una función que reciba un individuo de la población, la cual, como se ha comentado anteriormente, contiene una lista de números enteros que representan los diferentes colores, y el índice de cada elemento de la lista, representa el nodo del grafo.

Este método ha de recibir dicho individuo, y separar en diferentes listas, los nodos que tienen dicho color, de modo que, para un problema de coloreado de 4 colores, tendremos 4 listas, cada una de la cual contendrá los nodos en función de su color (El nodo con el número uno irá a la primera lista, el nodo con el número dos irá a la segunda lista, etc.)

Evolución y problemas durante la implementación

En el proceso que se ha seguido en la práctica, nos hemos encontrado con dificultades principalmente a nivel de código, pues el algoritmo de clasificación previamente descrito se diseño en una fase temprana del proyecto, pero requirió un tiempo para poder implementarse debido a la poca familiarización en el lenguaje de programación empleado.

VIII. EVALUACION DE LOS INDIVIDUOS

Una vez definida la función fenotipo, se ha de definir una función de evaluación fitness que se encarga de penalizar aquellos individuos que no cumplan los requisitos para poder aplicar correctamente el algoritmo.

Para ello, se ha implementado una función que, dado un individuo de la población, llame a la función fenotipo previamente definida, pasándole dicho individuo, y recibiendo de este modo una lista, de listas, donde se encuentren los nodos asociados a su color.

En posesión de dicha lista, se ha pasado a comprobar que cada nodo del grafo, dado su color, no repita el color de sus vecinos, en cuyo caso, se produciría la penalización, repitiéndose todas las veces que este caso se produzca.

Para comprobar si dos nodos son vecinos, se ha implementado una función que haciendo uso de la librería Networkx, calcule dado dos vértices, si estos son vecinos.

Evolución y problemas durante la implementación

En un primer momento, aunque se diseño el algoritmo en papel, no se supo como implementar por parte del grupo debido al desconocimiento del lenguaje de programación y librerias.

Tras unos días de investigación y pruebas con diferentes librerías (jgraph, igraph, etc.), se encontró la librería antes mencionada, que facilitaba el trabajo con grafos y ofrecía un amplio catálogo de métodos para poder trabajar con ellos.

IX. CAJA DE HERRAMIENTAS (TOOLBOX)

Se ha de destacar, que el principal elemento utilizado para la resolución del problema ha sido la librería DEAP, que proporciona un amplío marco de trabajo para la computación con algoritmos evolutivos.

Esta librería implementa de la forma correcta, ha permitido resolver el problema para el coloreado del grafo mediante algoritmos genéticos, para ello, se ha debido declarar como ya se ha visto previamente una población sobre la que trabajar, un fenotipo que transforme los individuos de dicha población y una función de evaluación fitness que permita diferenciar y otorgar un valor tangible a aquellos individuos que sean mejoras que los demás.

Posteriormente, DEAP requiere de la declaración de una caja de herramientas (Toolbox), en la que se deberán ir declarando aquellas características relevantes del problema para poder resolverlo, esto es, como será el individuo, que genes tendrá, y cual será el tamaño de la población.

Una vez definido estos valores básicos, hemos de añadir a la caja de herramientas la función de evaluación que usará en la resolución del algoritmo.

Dado que los algoritmos genéticos implementados en el paquete DEAP esperan un operador de cruce y un operador de mutación, estos serán proporcionados también.

Para completar la caja de herramientas, se ha de añadir un método de selección por torneo, en el que, para seleccionar un individuo, se elige al azar una cierta cantidad de individuos y de entre ellos, se selecciona el mas apto. Este se proporciona mediante un valor fijo de 1.

Evolución y problemas durante la implementación

Este apartado ha requerido de una parte de estudio del grupo de la práctica 4 para entender correctamente el funcionamiento de la caja de herramientas y el porqué de su utilidad, estudiando de la misma forma, la librería DEAP para ver su funcionamiento interno.

X. EASIMPLE Y VARAND

Una vez definido correctamente todos los elementos previos, se procederá a resolver el problema mediante la llamada al método eaSimple, que implementa la librería DEAP.

Aunque este método se implementa de forma directa, posteriormente se verá que se modificará para poder aplicar la técnica de enfriamiento simulado, en un apartado eaSimpleModificado y VarAndModificado.

Evolución y problemas durante la implementación

Este apartado ha resultado ser uno de los más difíciles de implementar, pues el desconocimiento del código interno y algoritmos utilizados hacía difícil entender y rastrear que se estaba realizando en cada paso, lo cual resultaba un inconveniente dado que se tardó un tiempo prolongado en depurar todo el código para conseguir que la primera prueba funcionase.

Para conseguirlo, el grupo estudió la documentación oficial de DEAP, y buscó información adicional en portales de programación, como son stackoverflow y reddit.

XI. SALON DE LA FAMA (HALLOFFAME)

El salón de la fama es otro elemento característico de la librería DEAP, que permite recoger aquellos individuos de mejor fitness.

Es necesario declararlo antes de hacer la llamada final a eaSimple, pues es un parámetro de la función.

Para declararlo, se especificará el número de elementos que queremos almacenar en el salón de la fama, que para la resolución de este problema será uno, pues solo nos interesa obtener una solución al problema.

Finalmente, podremos imprimir por pantalla todos los elementos del salón de la fama, que, serán, las soluciones del problema.

XII. ENFRIAMIENTO SIMULADO

Para implementar el Algoritmo Genético Mixto con un paso de Enfriamiento Simulado se ha utilizado una clase llamada “Annealer” la cual viene dada por una librería de Python llamada “simanneal”. Para el uso de dicha librería se ha consultado las siguientes páginas web:

- https://github.com/perrygeo/simanneal/blob/master/simanneal/anneal.py
- https://github.com/perrygeo/simanneal/blob/master/examples/salesman.py

En la primera de dichas páginas, se ha extraído la información referente a los distintos métodos y a la documentación en general de la clase “Annealer”. Después, para facilitar el entendimiento de su uso, en la segunda página aparece un ejemplo del Enfriamiento Simulado aplicado al “Problema del viajante”, el cual ha sido de gran ayuda para la implementación del Enfriamiento Simulado en el proyecto.

A. Métodos para el Enfriamiento Simulado

Para la implementación del Enfriamiento Simulado mediante dicha librería, es necesario crear una clase (llamada en el proyecto “GraphColorProblem) la cual extienda de la clase “Anneal”, para así poder utilizar sus métodos ya definidos. Aparte, en esta clase creada se deben declarar tres nuevos métodos:

- Un método “_init_”, el cual inicialice la clase llamando a la clase padre, es decir, a “Anneal”, y pasándole el estado o individuo actual del problema.
- Un método “move” el cual declara como se actualizará el estado o individuo actual de forma aleatoria, para comprobar si dicho estado actualizado tiene mejor “fitness” que el estado anterior.
- Un método “energy”, el cual calcula la energía de un estado. En términos de Algoritmo Genético, equivale al “fitness” de un individuo. Por tanto, en el proyecto este método es igual al método “fitness” usado, cambiando simplemente la manera en la que obtiene el individuo actual.

Una vez implementada dicha clase, dentro del código del método “varAnd” del Algoritmo Genético, habrá que llamarla, indicarle los parámetros (sino se indican, usará los valores por defectos) e invocar al método “anneal” el cual viene definido en su clase padre.

Los distintos parámetros y sus valores por defecto serían:

- Temperatura máxima o de inicio (Tmax) = 25000.0
- Temperatura mínima o de parada (Tmin) = 2.5
- Pasos (steps) = 50000
- Actualizaciones (updates) = 100
- Estrategia de copia (copy_strategy) = ‘deepcopy’

- Mejor estado (best_state) = None
- Mejor energía (best_energy) = None

Respecto al método "anneal", el cual es el encargado de realizar el Enfriamiento Simulado, tendría un funcionamiento descrito a continuación (en pseudocódigo):

Inicializar el contador de pasos a 0

Guardar en una variable el factor de temperatura, es decir, el menos logaritmo neperiano del cociente entre la temperatura máxima y la mínima

Guardar en una variable la temperatura máxima

Guardar en una variable la energía del estado actual

Copiar el estado actual como estado previo

Copiar la energía actual como energía previa

Asignar al mejor estado el estado actual

Asignar a la mejor energía la energía actual

Mientras el contador de pasos sea menor a los pasos indicados en el parámetro "steps":

- Incrementar en uno el contador de pasos
- Guardar en una variable la temperatura, es decir, la temperatura máxima por "e" elevado al factor de la temperatura multiplicado por el contador de pasos y dividido por el número total de pasos
- Llamar al método "move" para actualizar el estado actual
- Calcular la energía del estado actualizado y guardarla en una variable
- Calcular la diferencia de energía, es decir, la energía del estado actualizado menos la energía previa
- Si la diferencia de energía es mayor que 0 y "e" elevado a menos el cociente de la diferencia de energía entre la temperatura actual es menor que un número aleatorio:
  - Restaurar el estado previo, es decir, asignar al estado actual el estado previo y a la energía actual, la energía previa
- Sino:
  - Asignar al estado previo el estado actual
  - Asignar a la energía previa la energía actual
- Si la energía actual es menor (mejor) que la mejor energía:
  - Asignar al mejor estado el estado actual
  - Asignar a la mejor energía la energía actual

Asignar al estado actual el mejor estado

Devolver el mejor estado y la mejor energía

B. Métodos para el Algoritmo Genético Mixto

Para implementar el Enfriamiento Simulado dentro de los métodos del Algoritmo Genético de la librería "deap", se renombró el método "eaSimple" a "eaSimpleEnfriamiento" en el cual el único cambio respecto al que se usa para el Algoritmo Genético con mutaciones en nuestro proyecto, es la llamada al método "varAnd", el cual se ha renombrado a "varAndEnfriamiento", siendo dicho método en el cual se realiza lo necesario para llamar a la clase "GraphColorProblem" y realizar así el Enfriamiento Simulado. Cabe destacar que, como se ha eliminado las mutaciones, los métodos "eaSimpleEnfriamiento" y "varAndEnfriamiento" ya no reciben como parámetro la probabilidad de mutación, como es lógico.

El inicio del método "varAndEnfriamiento" es igual al del método "varAnd" de nuestro proyecto, es decir, clonando individuo a individuo la población actual y asignándola a la variable "offspring" para actualizarla. Después, se aplica cruzamiento al "offspring" según si la probabilidad de cruzamiento es mayor que un número aleatorio, al igual que en el método "varAnd".

El cambio respecto al método del Algoritmo Genético con mutaciones se refleja después del cruzamiento, ya que se ha suprimido el código referente a aplicar la mutación y se han añadido las siguientes líneas de código para el Enfriamiento Simulado (en pseudocódigo):

Para cada individuo de la población ("offspring"):

- Guardar en una variable el individuo
- Inicializar la clase "GraphColorProblem"
- Cambiar el número de pasos a 1
- Cambiar el número de actualizaciones a 0
- Cambiar la estrategia de copia a "slice"
- Invocar al método "anneal" para que devuelva el mejor estado y la mejor energía
- Actualizar el individuo asignándole el mejor estado devuelto por el Enfriamiento Simulado
- Borrar el valor fitness guardado de ese individuo
- Devolver la población actualizada ("offspring")

La decisión sobre los distintos valores asignados a los parámetros de la clase "GraphColorProblem" se ha debido a las siguientes circunstancias:

- El número de pasos es igual 1, ya que solo se debía introducir un paso de Enfriamiento Simulado. Si el número de pasos fuera mayor, el Enfriamiento Simulado llegaría él solo a la solución óptima, sin necesidad de requerir el resto del Algoritmo Genético. Esto haría que, en vez de ser un Algoritmo Genético Mixto, fuera simplemente un Enfriamiento Simulado.
- El número de actualizaciones es igual a 0, ya que dicha variable se utiliza para actualizar por consola el estado actual del Enfriamiento Simulado, lo cual para una población grande hacía que se llenara la consola de datos innecesarios, ya que el método "eaSimpleEnfriamiento" devuelve las estadísticas de cada generación. Además, al no tener los datos un estilo muy bien definido, se sobreponían entre ellos en la consola, quedando de una forma muy poco estética.

- La estrategia de copia es “slice” debido a que los distintos estados (individuos) son listas, y “slice” es la forma más eficiente de copia para dichos tipos de datos.

  (Fuente: https://github.com/perrygeo/simanneal)

C. Evolución y problemas de la implementación

Al inicio del proyecto, después de recopilar información acerca de qué era el Enfriamiento Simulado y cómo funcionaba, se intentó diseñar un algoritmo de Enfriamiento Simulado tomando como referencias ciertos ejemplos de Internet (fuentes:

- https://am207.github.io/2017/wiki/lab4.html
- http://www.theprojectspot.com/tutorial-post/simulated-annealing-algorithm-for-beginners/6

Después de comprobar que de dicha forma no estaba siendo la implementación muy eficiente ni muy comprensible, se optó por la utilización de librerías externas que ofrecieran la facilidad de adaptar el Enfriamiento Simulado a un problema dado, además de que la eficiencia del algoritmo estuviera garantizada. Debido a esto, se decidió trabajar con la librería antes nombrada, “simanneal”.

Al inicio de la implementación, se estaba trabajando con valores para el número de pasos de entre 100000 y 1000000. El problema de ello era que el alto tiempo que tardaba el Algoritmo Genético tardaba en ejecutarse, aparte de que, como se ha dicho anteriormente, el Enfriamiento Simple encontraba por sí mismo la solución en la primera generación, dejando inútil al resto del Algoritmo Genético. Por tanto, se decidió bajar el número a 1 que así no solo dependa del Enfriamiento Simple para llegar a la solución, sino que también dependa del resto del Algoritmo Genético.

XIII. INTERFAZ

Para realizar la interfaz, se ha recurrido a la librería Tkinter, que permite implementar los diferentes apartados necesarios para poder establecer, por parte del usuario, las características del grafo a colorear, así como los valores de mutación, colores, y otras variables que permiten a la aplicación ser polivalente en el problema a desarrollar.

La librería se ha implementado comenzando por el tamaño de la ventana principal, junto con los diferentes campos a rellenar por el usuario.

Estos valores se recogerán en variables, que se pasarán al código desarrollado y sustituirán aquellos valores que previamente eran fijos.

Una vez determinado los valores de la interfaz, se han declarado dos botones para poder resolver el problema, por mutación o enfriamiento simulado. Cada botón, recogerá internamente la secuencia de código asociado para su resolución.

Finalmente, para representar la solución obtenida, se han utilizado las librerías networkx, matplotlib y PIL.

A través de la librería networkx, conseguimos dibujar el grafo en la consola de Jupyter, dado que nos permite colorear el grafo y mostrarlo, recibiendo el individuo ganador por el procedimiento realizado. Una vez hecho, matplotlib permite guardar el grafo coloreado en una imagen en formato png, que se almacenará en el directorio del entorno donde se esté trabajando. Esta imagen se leerá a través de la librería PIL, cargará, y renderizada en la interfaz gráfica, para así mostrárselo al usuario ejecutor, junto con el individuo seleccionado.

Evolución y problemas durante la implementación

Durante el desarrollo de la interfaz, se ha recurrido constantemente a información en foros y tutoriales de Tkinter, ya que ninguno de los integrantes del proyecto había programado previamente una interfaz, de modo que no se sabía muy bien cual era el funcionamiento o estructura de estas.

Se comenzó creando una ventana vacía que se desplegase al ejecutar el código, para, una vez logrado, pasar a mostrar texto dentro de esta.

Una vez conseguido, se investigó como se podían pedir datos y almacenar dichos datos, de modo que pudiésemos utilizarlos en variables para poder implementarlo en nuestro código.

Una vez conseguimos esto, pasamos a la implementación de los botones, lo cual, fue la parte más densa y pesada, ya que fue el momento de utilizar, estructurar, y pegar, todo nuestro código previamente desarrollado, y darle la forma y orden adecuado de modo que cada elemento se ejecutase en orden y sin conflictos.

Una vez conseguimos mostrar las soluciones del salón de la fama por pantalla, pasamos a averiguar como podíamos encontrar la forma de representar el grafo coloreado por la solución en la misma interfaz, de modo que recurrimos a la investigación de librerías y foros donde se resolvieran estos mismos problemas.

XIV. EXPERIMENTOS

Una vez finalizado todos los apartados anteriores, se ha procedido a realizar diversos experimentos con ambos algoritmos, con el objetivo de sacar conclusiones sobre la efectividad de los métodos y algoritmos implementados, y la importancia de los valores de mutación, así como la población seleccionada.

A continuación, se mostrará una tabla que recopilará todas las pruebas realizadas por el grupo:

Prueba 1 (Grafo completo)	DATOS			RESULTADOS			
	Colores	Nodos	Población	Tiempo (AG)	Tiempo (AGM)	Fitness (AG)	Fitness (AGM)
Iter. 1	8	200	10	4.8	24.6	2403000	2420000
Iter. 2	8	200	10	5.1	24.9	2435000	2455000
Iter. 3	8	200	10	5.2	24.8	2480000	2473000

Prueba 2 (Grafo aleatorio)	DATOS				RESULTADOS			
	Colores	Nodos	Pobl.	Densidad ejes	Tiempo (AG)	Tiempo (AGM)	Fitness (AG)	Fitness (AGM)
Iter. 1	6	80	20	0.15	2.1	5.3	59000	52000
Iter. 2	6	80	20	0.15	2	4.9	62000	56000
Iter. 3	6	80	20	0.15	1.8	5.1	67000	52000

Prueba 3 (Grafo aleatorio)	DATOS				RESULTADOS			
	Colores	Nodos	Pobl.	Densidad ejes	Tiempo (AG)	Tiempo (AGM)	Fitness (AG)	Fitness (AGM)
Iter. 1	3	20	5	0.15	0.8	0.9	6000	5000
Iter. 2	3	20	5	0.15	0.7	0.9	5000	1000
Iter. 3	3	20	5	0.15	0.5	0.8	9000	1000

Prueba 4 (Grafo aleatorio)	DATOS				RESULTADOS			
	Colores	Nodos	Pobl.	Densidad ejes	Tiempo (AG)	Tiempo (AGM)	Fitness (AG)	Fitness (AGM)
Iter. 1	3	20	50	0.15	1.1	1	3000	2000
Iter. 2	3	20	50	0.15	0.9	1.1	4000	1000
Iter. 3	3	20	50	0.15	1.2	1.3	4000	0

Prueba 5 (Grafo Petersen)	DATOS			RESULTADOS			
	Colores	Nodos	Población	Tiempo (AG)	Tiempo (AGM)	Fitness (AG)	Fitness (AGM)
Iter. 1	3	10	20	0.8	0.7	0	0
Iter. 2	3	10	20	0.9	1.1	2000	0
Iter. 3	3	10	20	0.8	0.8	1000	0

Prueba 6 (Grafo Petersen)	DATOS				RESULTADOS			
	Colores	Nodos	Pobl.	Prob. mut.	Tiempo (AG)	Tiempo (AGM)	Fitness (AG)	Fitness (AGM)
Iter. 1	3	10	20	0.6	1.2	1.4	0	0
Iter. 2	3	10	20	0.6	1	1.3	3000	0
Iter. 3	3	10	20	0.6	1.1	1.3	5000	0
Iter. 4	3	10	20	0.6	1.3	1.4	4000	0
Iter. 5	3	10	20	0.6	1	1.2	4000	0

Prueba 7 (Grafo Petersen)	DATOS				RESULTADOS			
	Colores	Nodos	Pobl.	Prob. mut.	Tiempo (AG)	Tiempo (AGM)	Fitness (AG)	Fitness (AGM)
Iter. 1	3	10	20	0.05	1	1.4	3000	0
Iter. 2	3	10	20	0.05	1.3	1.3	5000	0
Iter. 3	3	10	20	0.05	0.9	1.3	4000	0
Iter. 4	3	10	20	0.05	1.2	1.4	5000	0
Iter. 5	3	10	20	0.05	0.9	1.2	3000	0

Se ha comenzado las pruebas utilizando un grafo completo de 200 nodos y usando 8 colores. Dicho grafo, como es lógico, no tiene coloreado óptimo, por lo que ninguno de los dos algoritmos devolvería una solución del problema; solo se estaba comprobando cuál era más eficiente. Para el segundo grafo se ha usado un grafo aleatorio generado mediante el algoritmo de Erdos Renyi de 80 nodos y una densidad de ejes de 0.15, usando 6 colores. Dicho grafo tampoco tiene coloreado óptimo con dichos colores, y su utilidad era la misma que el anterior. Para las pruebas tres y cuatro, se ha vuelto a utilizar un grafo aleatorio, pero con un número cuatro veces menor de vértices, intentando que algún algoritmo alcanzase la solución. Por último, para las pruebas cinco, seis y siete, se ha usado un grafo de Peterson (el cual tiene diez nodos) y se ha jugado con la probabilidad de mutación para comparar resultados.

A. *Conclusiones sobre los experimentos*

En un primer instante, con la realización de los primeros experimentos, no pareciera que el mismo problema resuelto mediante algoritmo genético con mutación mixta mediante enfriamiento simulado, tuviese unos beneficios sobre la aplicación del algoritmo genético por mutación y cruzamiento, a pesar, de que la realización de enfriamiento simulado requiere una cantidad sustancial de tiempo con respecto a su rival en este proyecto.

Sin embargo, tras realizar numerosas pruebas, hemos podido comprobar que en todos aquellos grafos cuyo número de vértices no exceda los 80 vértices, la realización por enfriamiento simulado se acerca con mucha más veracidad a la solución. Esto, se atenúa incluso más, cuando los grafos son aún mas pequeños, donde el enfriamiento simulado es capaz de encontrar prácticamente siempre una solución, al cabo de una o dos repeticiones, mientras que, resolviendo el mismo problema mediante algoritmo genético por mutación y cruzamiento, no siempre hemos llegado a la solución, o solo se ha producido en una de todas las iteraciones realizadas.

Sin embargo, cuando los grafos poseen un número mayor de vértices, el enfriamiento simulado no resulta efectivo, ya que no solo tarda significativamente más que la mutación y cruzamiento, sino que el fitness obtenido para la solución que más se acerca, no difiere representativamente de su competidor.

B. *Valores de Mutacion y poblacion*

Una vez llegada a la conclusión anterior, se ha procedido a alterar los valores de la población inicial y la probabilidad de cruzamiento de los individuos.

Para la población inicial, como se puede observar en la prueba 3 y 4, al aumentar la población inicial por un factor de 10, conseguimos llegar a la solución utilizando la técnica de enfriamiento simulado, sin embargo, no se consigue llegar utilizando mutación y cruzamiento.

XV. CONCLUSIONES

Como finalización de este proyecto, se ha querido sacar algunas conclusiones de lo aprendido mediante la realización del mismo. La intención del proyecto consiste en aprender los fundamentos, y a su vez, profundizar en los Algoritmos Genéticos dados en la parte teórica de la asignatura de Inteligencia Artificial, investigando nuevas técnicas como es la incorporación del Enfriamiento Simulado para implementar un Algoritmo Genético Mixto.

Además, hemos podido profundizar en un lenguaje nuevo y tan necesario en el mundo de la informática actual como es Python, para el que no solo hemos desarrollado código, sino que también hemos conseguido desarrollar una interfaz gráfica funcional, siendo la primera vez en el Grado de Ingeniería del Software en el que vemos este tipo de desarrollo.

También es necesario destacar, que es la primera vez en una asignatura que se hace un uso activo de librería externas que no han sido facilitadas o enseñadas por los profesores; las cuales no solo complementan el código, sino que facilitan el desarrollo y la creación de aplicaciones más potentes y eficientes que han permitido expresarse de la forma que se pretendía en una primera instancia.

Como complemento a lo anterior, se ha realizado este documento científico siguiendo las pautas internacionales y rellenándolo con contenido obtenido de la investigación y documentación de los diferentes portales informáticos en diferentes lenguajes y/o idiomas.

Para finalizar este apartado, se considera que la práctica ha servido para enriquecer de forma significativa el conocimiento sobre, no solo la materia impartida a lo largo del presente curso académico, sino también sobre los diferentes campos anteriormente citados.

Sin embargo, debido a la brevedad de tiempo de la que se ha dispuesto para desarrollar el proyecto, teniendo en cuenta las fechas en las que ha coincidido su producción dentro de los últimos meses del año lectivo, no se ha podido profundizar en elementos tan interesantes y llamativos como la interfaz gráfica, para la que hubiera sido interesante trabajar en una aplicación con mayores opciones y personalización. Además, también hubiera sido satisfactorio implementar una mayor variedad de librerías, que hubiesen permitido la visualización en 3D de los grafos ya coloreados.

En vista de los Trabajos de Fin de Grado del próximo curso, esta experiencia puede servir como preparación para la realización de un completo TFG, ya que se ha podido mezclar tanto elementos de investigación y divulgación sobre temas ajenos a los conocimientos de los autores, como la posterior realización de un documento que recabe todo este contenido y lo exponga de una forma formal, elegante y, sobre todo, transparente.

BIBLIOGRAFÍA

[1] https://stackoverflow.com/questions/39179948/how-do-i-install-pil-pillow-for-python-3-6

[2] http://devdocs.io/python/library/stdtypes#sequence-types-list-tuple-range

[3] http://deap.readthedocs.io/en/master/api/algo.html

[4] https://wiki.python.org/moin/TkInter

[5] https://python-para-impacientes.blogspot.com/2015/12/tkinter-interfaces-graficas-en-python-i.html

[6] http://www.tutorialspoint.com/python/python_gui_programming.htm

[7] https://likegeeks.com/python-gui-examples-tkinter-tutorial/

[8] https://www.youtube.com/watch?v=RJB1Ek2Ko_Y

[9] https://github.com/perrygeo/simanneal/blob/master/examples/salesman.py

[10] https://github.com/perrygeo/simanneal/blob/master/simanneal/anneal.py

[11] https://github.com/DEAP

[12] https://networkx.github.io/documentation/stable/index.html

[13] https://networkx.github.io/documentation/networkx-1.7/index.html

[14] http://gmendezm.blogspot.com/2012/12/tutorial-tkinter-python-gui.html

[15] https://docs.python.org/3/library/tkinter.html

[16] http://effbot.org/tkinterbook/grid.htm

[17] https://pythonprogramming.net/tkinter-adding-text-images/

[18] https://matplotlib.org/examples/color/colormaps_reference.html

[19] https://www.reddit.com/r/Python/comments/4a4waf/how_to_returnprint_entry_text_in_tkinter/

[20] http://igraph.org/python/doc/tutorial/tutorial.html

[21] https://www.cs.rhul.ac.uk/home/tamas/development/igraph/tutorial/tutorial.html