



UNIVERSIDAD POLITÉCNICA DE MADRID
FACULTAD DE INFORMÁTICA

TRABAJO FIN DE CARRERA

APLICACIÓN INTEGRAL DE GRAFOS: COLORACIÓN

AUTOR: Abraham Iniesto Díaz
TUTOR: Gregorio Hernández Peñalver

ÍNDICE

1.INTRODUCCIÓN Y OBJETIVOS	1
2.TEORÍA DE GRAFOS	2
2.1. Introducción	2
2.2. Historia.....	2
2.3. Nociones Básicas.....	3
2.3.1. Vértice	3
2.3.2. Arista	3
2.3.3. Grafo	4
2.3.4. Subgrafos	5
2.3.5. Algunos ejemplos de grafos	6
2.3.6. Grafos conexos y componentes conexas.....	7
2.3.7. Digrafos.....	8
2.4. Coloración de Grafos	9
2.4.1. El Teorema de los cuatro colores.....	9
2.4.2. Concepto de Independencia.....	10
2.4.3. Coloración de Vértices.....	11
2.4.4. Algoritmos de Coloración de Grafos	11
2.4.5. Algoritmo de Coloración Secuencial Básico	12
2.4.6. Algoritmo de Coloración de Welsh y Powell	14
2.4.7. Algoritmo de Coloración de Matula, Marble, Isaacson	15
2.4.8. Algoritmo de Brelaz.....	16
2.4.9. Algoritmo de Independencia MCI	17
3.DISEÑO DE LA APLICACIÓN	19
3.1. Lenguaje de programación	19
3.2. Diseño de Alto Nivel.....	20
3.2.1. Definición de los límites del sistema	20
3.2.2. Diagramas de casos de uso.....	21
3.2.2.1. Generar Grafo.....	24

3.2.2.2.	Establecer Configuración	27
3.2.2.3.	Ejecutar Algoritmo	28
3.2.3.	Descripción de los casos de uso en formato extendido.....	29
3.3.	Diseño de Bajo Nivel.....	49
3.3.1.	Diagrama de clases.....	49
3.3.1.1.	Visión general	50
3.3.1.2.	Grafo	52
3.3.1.3.	Algoritmo	54
4.	MANUAL DE USUARIO	63
4.1.	Partes de la aplicación.....	63
4.1.1.	Barra de Menús	63
4.1.1.1.	Archivo	63
4.1.1.2.	Editar.....	65
4.1.1.3.	Grafos	65
4.1.1.4.	Herramientas	66
4.1.1.5.	Ayuda	67
4.1.2.	Barra de Iconos (Botones).....	68
4.1.3.	Información.....	71
4.1.4.	Posición.....	73
4.1.5.	Lienzo.....	74
4.2.	Operaciones Genéricas.....	76
4.2.1.	Crear Nuevo Grafo Automático	76
4.2.2.	Crear Nuevo Grafo Personalizado	78
4.2.3.	Guardar Grafo	80
4.2.4.	Abrir Grafo.....	80
4.2.5.	Cerrar	81
4.2.6.	Limpiar algoritmos.....	82
4.2.7.	Exportar Grafo	82
4.2.8.	Salir	83
4.2.9.	Editar nodo.....	83
4.2.10.	Insertar Vértices	85
4.2.11.	Insertar Arista.....	85

4.2.12.	Eliminar Nodo	85
4.2.13.	Suprimir Arista.....	86
4.2.14.	Cambiar las opciones	86
4.2.15.	Imagen de fondo.....	89
4.2.16.	Estadísticas	89
4.2.17.	Matriz de adyacencia.....	90
4.2.18.	Matriz de ponderación.....	92
4.2.19.	Grafos tipo.....	93
4.2.20.	Ejecución.....	94
4.2.21.	Ejecución Automática	98
4.2.22.	Ejecución Interactiva.....	99
4.2.23.	Coloración: Algoritmo Secuencial Básico	99
4.2.24.	Coloración: Algoritmo Secuencial de Welsh-Powell	101
4.2.25.	Coloración: Algoritmo Secuencial de Matula-Marble-Isaacson.....	103
4.2.26.	Coloración: Algoritmo de Coloración de Brelaz	103
4.2.27.	Coloración: Algoritmo de Independencia MCI.....	105
4.2.28.	Búsqueda: Algoritmo DFS.....	108
4.2.29.	Búsqueda: Algoritmo BFS	110
4.2.30.	Búsqueda: Conectividad: algoritmo de bloques.....	112
4.2.31.	Búsqueda: Algoritmo de Dijkstra.....	113
5.	ESTUDIO ESTADISTICO	116
6.	CONCLUSIONES	118
7.	BIBLIOGRAFÍA	119

LISTADO DE ILUSTRACIONES

Figura 1. Problema de los puentes de Königsberg	3
Figura 2: Representación de un grafo.....	5
Figura 3: Un grafo G, un subgrafo H, un subgrafo generador H' y el subgrafo H* inducido por {a,b,c}	6
Figura 4: El resultado de eliminar un vértice o una arista	6
Figura 5: Representaciones de Grafos completos de 3, 4 y 5 vértices (K3, K4 y K5)	6
Figura 6: Grafo Bipartito Completo K5,4	7
Figura 7: Grafo Regular de grado 2.....	7
Figura 8: Grafo conexo (izquierda). Grafo no conexo con 3 componentes conexas (derecha).....	8
Figura 9: Digrafo simple	8
Figura 10: Mapa del mundo coloreado de verde, amarillo, azul y rojo	10
Figura 11: Grafo sobre el que se ejecutará el algoritmo de coloración básico	12
Figura 12: Visualización de los pasos de la ejecución del algoritmo secuencial básico.	13
Figura 13: Coloración del grafo usando el algoritmo de Welsh-Powell.....	14
Figura 14: Coloración del grafo usando el algoritmo de Matula, Marble, Isaacson.....	15
Figura 15: Situación inicial, antes de la ejecución del algoritmo de Brelaz.....	16
Figura 16: Situación final, después de la ejecución del algoritmo de Brelaz	16
Figura 17: Evolución paso a paso del algoritmo de Independencia MCI.....	18
Figura 18: Diagrama de nivel 0.....	23
Figura 19: Diagrama de nivel 1: GenerarGrafo.....	24
Figura 20: Diagrama de nivel 2: GenerarGrafoAleatorio.....	24
Figura 21: Diagrama de nivel 2: GenerarGrafoPersonalizado	25
Figura 22: Diagrama de nivel 2: GenerarGrafoTipo	26
Figura 23: Diagrama de nivel 1: EstablecerConfiguración	27
Figura 24: Diagrama de nivel 1: EjecutarAlgoritmo	28
Figura 25: Diagrama de Clases que muestra la Visión General del Proyecto	51
Figura 26: Diagrama de Clases que muestra la clase Grafo (y las clases con las que se relaciona).....	53
Figura 27: Diagrama de Clases que muestra los Algoritmos DFS y BFS	55
Figura 28: Diagrama de Clases que muestra los Algoritmos de Bloques y Dijkstra.....	57
Figura 29: Diagrama de Clases que muestra los Algoritmos de Coloración (Secuencial y Brelaz).....	59
Figura 30: Diagrama de Clases que muestra los Algoritmos de Coloración de Conjuntos Independientes	62
Figura 31: Diagrama de Clases que muestra la Configuración	62
Figura 32: Menú Archivo.....	64

Figura 33: Menú Editar.....	65
Figura 34: Menú Grafos.....	65
Figura 35: Menú Herramientas.....	66
Figura 36: Menú Ayuda.....	67
Figura 37. Barra de Iconos.....	68
Figura 38. Árbol de un grafo	71
Figura 39. Leyenda del algoritmo de Búsqueda en Profundidad	72
Figura 40. Información de la ejecución de un algoritmo	73
Figura 41. Cuadro de posición del cursor	73
Figura 42. Lienzo de la aplicación mostrando un grafo.....	74
Figura 43. Menú contextual de edición de grafo	74
Figura 44. Menú contextual de ejecución de algoritmo	75
Figura 45. Ventana de información de vértice sobre el lienzo.....	75
Figura 46. Ventana de creación de un Nuevo Grafo Automático	76
Figura 47. Detalle de la aplicación una vez creado un grafo	77
Figura 48. Ventana de confirmación para salvar el grafo	77
Figura 49. Ventana de creación de un Nuevo Grafo Personalizado.....	78
Figura 50. Ventana de edición de Nodos en la creación de un grafo (no dirigido) personalizado.....	78
Figura 51. Ventana de edición de Nodos en la creación de un grafo (dirigido) personalizado.....	79
Figura 52. Ventana de selección del archivo (Grafo) a cargar en la aplicación.....	81
Figura 53. Menú Archivo → Exportar a.....	82
Figura 54. Ventana de edición de un grafo no dirigido.....	83
Figura 55. Ventana de edición de un grafo dirigido	84
Figura 56. Ventana de opciones.....	87
Figura 57. Detalle de un elemento de la ventana de opciones	87
Figura 58. Ventana de selección de color	87
Figura 59. Ventana de selección de algoritmos para la generación de estadísticas	90
Figura 60. Ventana de la Matriz de Adyacencia.....	91
Figura 61. Ventana de la Matriz de Ponderación.....	92
Figura 62. Ventana de selección de Algoritmos de Coloración.....	94
Figura 63. Detalle de la tabla del algoritmo de Brelaz.....	95
Figura 64. Ventana de selección de Algoritmos de Búsqueda.....	96
Figura 65. Detalle de la pestaña del algoritmo de Dijkstra en la ventana de selección de Algoritmos.....	97
Figura 66. Menú contextual de selección de color (algoritmos de coloración)	99
Figura 67. Detalle de un vértice coloreado	100
Figura 68. Ventana de Alerta: Color Incorrecto	100
Figura 69. Ventana de alerta: Color no Adecuado.....	100

Figura 70. Ventana de alerta: Vértice ya coloreado	101
Figura 71. Detalle del cuadro de información del algoritmo de coloración secuencial básico.....	101
Figura 72. Ventana de alerta: Vértice incorrecto.....	102
Figura 73. Ventana de alerta: Vértice no adecuado.....	102
Figura 74. Ventana de alerta: Vértice incorrecto.....	103
Figura 75. Detalle de la tabla del algoritmo de Brelaz	104
Figura 76. Menú contextual en la ejecución del algoritmo MCI	105
Figura 77. Detalle del lienzo en el algoritmo MCI, mostrando vértices no disponibles.....	105
Figura 78. Detalle del lienzo en el algoritmo MCI, después de recalcular el grafo.....	106
Figura 79. Detalle del lienzo en el algoritmo MCI, con los vértices coloreados.....	107
Figura 80. Menú contextual en la ejecución del algoritmo DFS	108
Figura 81. Ventana de Alerta: Vértice Incorrecto	109
Figura 82. Ventana de alerta: Vértice no adecuado.....	109
Figura 83. Ventana de Algoritmo de Búsqueda BFS. Selección de Vértice de Inicio	110
Figura 84. Menú contextual en la ejecución del algoritmo BFS	110
Figura 85. Ventana de Alerta: Vértice Incorrecto	111
Figura 86. Ventana de alerta: Vértice no adecuado.....	111
Figura 87. Menú contextual en la ejecución del algoritmo de Bloques.....	112
Figura 88. Ventana de Alerta: vértice no determinado correctamente	112
Figura 89. Ventana de Algoritmo de Dijkstra. Selección de Vértice de Inicio	113
Figura 90. Menú contextual en la ejecución del algoritmo de Dijkstra. Seleccionar vértice.....	113
Figura 91. Detalle del lienzo en el algoritmo de Dijkstra.....	114
Figura 92. Ventana de Alerta: Vértice seleccionado incorrecto	114
Figura 93. Menú contextual en la ejecución del algoritmo de Dijkstra. Establecer distancia	114
Figura 94. Ventana de Alerta: Distancia incorrecta	115

AGRADECIMIENTOS

En primer lugar, querría dar las gracias al Tutor del Proyecto, **Gregorio Hernández Peñalver**, del departamento de Matemática Aplicada de la Facultad de Informática de Madrid, ya que fue quien nos asignó el proyecto, y quien nos ha ayudado con todas las dudas que nos han ido surgiendo durante todo este tiempo.

De igual manera quiero agradecerles a mis padres **Ángel** y **Carolina**, todo el esfuerzo que han hecho, para que haya podido cursar la carrera, sin su ayuda hubiese sido imposible llegar hasta aquí. No quiero olvidarme de mi hermano **Alberto**, gracias por tu apoyo.

Quiero agradecerle de manera muy especial a **Verónica**, por estar siempre ahí, en los momentos buenos y en los malos, y sobre todo, por recordarme, durante todo este tiempo, que este proyecto teníamos que acabarlo, que no se podía quedar sin finalizar.

Y no hay que olvidar a **Juan**, el otro componente de este proyecto, gracias por la ayuda, gracias por estar siempre que te he necesitado. Supongo que el hacer un proyecto junto a un amigo facilita mucho las cosas, pero de todos modos, gracias por la paciencia. Después de estar más de tres años realizando el proyecto, voy a echar de menos las tardes en las que quedábamos y nos dábamos cuenta que teníamos más errores que el día anterior. Tendremos que buscarnos un nuevo proyecto.

Antes de acabar quiero recordar al resto de compañeros de la universidad, del trabajo y de amigos, que me han ofrecido su apoyo y cariño durante todo este tiempo. Gracias a todos.

1. INTRODUCCIÓN Y OBJETIVOS

El presente documento pretende dar una información detallada de la aplicación desarrollada en Java “AIG” (Aplicación Integral de Grafos).

AIG es una herramienta de diseño de grafos, ya que le permitirá al usuario generar cualquier tipo de grafo que desee. Además permite la generación de una serie de grafos predefinidos.

La aplicación permite al usuario la ejecución de una serie de algoritmos, sobre los grafos generados previamente. Los algoritmos se pueden dividir en dos bloques:

- Algoritmos de Coloración.
- Algoritmos de Búsqueda.

Este documento se centrará principalmente en los algoritmos de coloración, aunque en determinadas partes del documento, existirán referencias al resto de algoritmos implementados (Búsqueda).

El desarrollo de la aplicación ha sido pensado para que pueda ser utilizada como material docente, ya que el usuario dispone de la posibilidad de ejecutar los algoritmos de forma automática o interactiva. Siendo el usuario en éste último caso quien tendrá que ir indicando los pasos a seguir para poder finalizar el algoritmo. De esta manera el usuario tiene una herramienta donde podrá poner en práctica los conocimientos teóricos que ha adquirido.

El usuario podrá configurar gran parte del aspecto visual de la aplicación.

La aplicación está pensada para que se puedan introducir nuevas funcionalidades. Para ello, durante la fase de análisis y desarrollo de la aplicación se ha tomado en consideración este detalle, con lo que en un futuro será bastante sencillo añadir nuevos módulos de ejecución a la aplicación. A la hora del diseño visual de la aplicación se ha intentado conseguir un aspecto lo más profesional posible, intentando que tenga un aspecto similar a las aplicaciones comerciales.

En este documento también podremos encontrar un estudio estadístico de los diferentes algoritmos de coloración implementados en la aplicación sobre un conjunto determinado de grafos.

2. TEORÍA DE GRAFOS

2.1. Introducción

La teoría de grafos estudia las propiedades de los grafos.

Un grafo es un conjunto, no vacío, de objetos llamados vértices (o nodos) y una selección de pares de vértices, llamados aristas que pueden ser orientados o no. Un grafo se representa mediante una serie de puntos (los vértices) conectados por líneas (las aristas).

2.2. Historia

El trabajo de **Leonhard Euler** en 1736, sobre el **problema de los puentes de Königsberg** es considerado el primer resultado de la teoría de grafos.

El problema, formulado originalmente de manera informal, consistía en responder a la siguiente pregunta:

“Dado el mapa de Königsberg, con el río Pregolya dividiendo el plano en cuatro regiones distintas, que están unidas a través de los siete puentes, ¿es posible dar un paseo comenzando desde cualquiera de estas regiones, pasando por todos los puentes, recorriendo sólo una vez cada uno, y regresando al mismo punto de partida?”

La respuesta en negativa, es decir no existe una ruta con dichas características. Euler en 1736, en su publicación “Solutio problematis ad geometriam situs pertinentis”, demuestra una solución generalizada del problema. Para dicha demostración, Euler recurre a una abstracción del mapa, enfocándose exclusivamente en las regiones terrestres y las conexiones entre ellas. Cada puente lo representó mediante una línea que unía a dos puntos, cada uno de los cuales representaba una región diferente. Así el problema se reduce a decidir si existe o no un camino que comience por uno de los puntos, transite por todas las líneas una única vez, y regrese al mismo punto de partida.



Figura 1. Problema de los puentes de Königsberg

Posteriormente en 1845 **Gustav Kirchhoff** publico sus leyes de los circuitos para calcular el voltaje y la corriente en los circuitos eléctricos.

En 1852 **Francis Guthrie** planteo el **problema de los cuatro colores** que plantea si es posible, utilizando solamente cuatro colores, colorear cualquier mapa de países de tal forma que dos países vecinos nunca tengan el mismo color. Este problema, que no fue resuelto hasta un siglo después por Kenneth Appel y Wolfgang Haken, puede ser considerado como el nacimiento de la teoría de grafos. Al tratar de resolverlo, los Matemáticos definieron términos y conceptos teóricos fundamentales de los grafos.

2.3. Nociones Básicas

A continuación se exponen una serie de conceptos relativos a la Teoría de Grafos necesarios para este proyecto.

2.3.1. Vértice

Los **vértices o nodos**, constituyen uno de los dos elementos que forman un grafo. Se define también como la unidad fundamental de la que están compuestos los grafos.

Los vértices son tratados, en la teoría de Grafos, como unidades indivisibles y sin propiedades, aunque pueden tener estructuras adicionales dependiendo del uso del grafo al que pertenecen.

2.3.2. Arista

Las aristas, junto con los vértices, forman los elementos principales de un grafo. Se definen como las uniones entre nodos o vértices. Usualmente las aristas denotan relaciones entre los vértices, como el orden, la vecindad o la herencia.

En algunos casos es necesario asignar un sentido a las aristas, por ejemplo, si se quiere representar la red de las calles de una ciudad con sus direcciones únicas. El conjunto de aristas será ahora un subconjunto de todos los posibles pares

ordenados de vértices, con $(a, b) \neq (b, a)$. Los grafos que contienen aristas dirigidas se denominan grafos orientados o dirigidos.

Las aristas no orientadas se consideran bidireccionales para efectos prácticos (equivale a decir que existen dos aristas orientadas entre los nodos, cada una en un sentido).

En algunos tipos de grafos las aristas llevan asociadas un número que indica una información asociada a ambos vértices. Pueden indicar un coste o ser una representación del trabajo necesario para recorrer el camino de un vértice al otro. Puede darse el caso en que el camino de A hacia B tenga un coste diferente que el de B hacia A. Finalmente, no existe obligación de que dados dos vértices exista una arista que los una. Es decir, la relación entre vértices y aristas no tiene porque producirse.

2.3.3. Grafo

Un grafo es un par $G = (V, A)$, donde V es un conjunto finito no vacío (a cuyos elementos llamaremos vértices o nodos) y A es una familia finita de pares no ordenados $\{u, v\}$ de vértices de V (a cuyos elementos llamaremos aristas).

El número de vértices, es decir, la cardinalidad del conjunto V se denomina orden del grafo y se denota por $|V|$. Por lo general se utiliza n para denotar el orden de G .

El número de aristas, es decir, la cardinalidad del conjunto A se denomina tamaño del grafo y se denota por $|A|$. Por lo general se utiliza m para denotar el tamaño de G .

Un grafo simple es un par $G = (V, A)$, donde V es un conjunto finito no vacío y A es un conjunto finito de pares no ordenados $\{u, v\}$ de vértices distintos de V .

Si $a = \{u, v\}$ es una arista de G , escribiremos sólo $a = uv$, y diremos que:

- los vértices u y v son **adyacentes**
- la arista uv es **incidente** con los vértices u y v (también diremos que uv une los vértices u y v).
- los vértices u y v son **incidentes** con la arista uv (también diremos que u y v son extremos de uv).
- el vértice u (o el vértice v) y la arista uv son incidentes

Además, diremos que:

- dos aristas son adyacentes si y sólo si tienen algún extremo común.
- un vértice es aislado si no tiene otros vértices adyacentes.

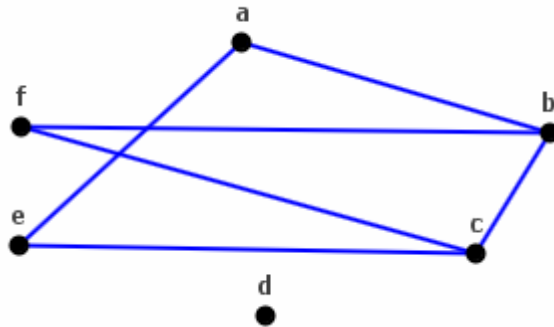


Figura 2: Representación de un grafo

De este grafo podemos decir, por ejemplo, que los vértices a y b son adyacentes y son incidentes con (o son extremos de) la arista ab, que la arista ab es incidente con (o une) los vértices a y b, que el vértice a y la arista ab son incidentes, que las aristas ab y bc son adyacentes (en este caso, el extremo común es el vértice b) y que el vértice e es aislado.

Se llama grado (o valencia de un vértice v al número de aristas incidentes en él, (cada bucle se cuenta, por tanto, dos veces). Se designa por $d(v)$.

En el grafo de la Figura 2 se tiene que $d(a) = 2$, $d(b) = 3$, $d(c) = 3$, $d(d) = 0$, $d(e) = 2$ y $d(f) = 2$.

2.3.4. Subgrafos

Un subgrafo de $G = (V, A)$ es otro grafo $H = (V', A')$ tal que $V' \subseteq V$ y $A' \subseteq A$. Si $V' = V$, se dice que H es un subgrafo generador de G.

Si en A' están todas las aristas del grafo G que tienen sus extremos en V' se dice que H es el subgrafo inducido por V' . Se indica por $H = G[V']$ o $H = \langle V' \rangle$.

Si B es un subconjunto de A se llama subgrafo inducido por B, y se designa con $\langle B \rangle$ al grafo de G cuyos vértices son los extremos de las aristas de B y cuyas aristas son las de B.

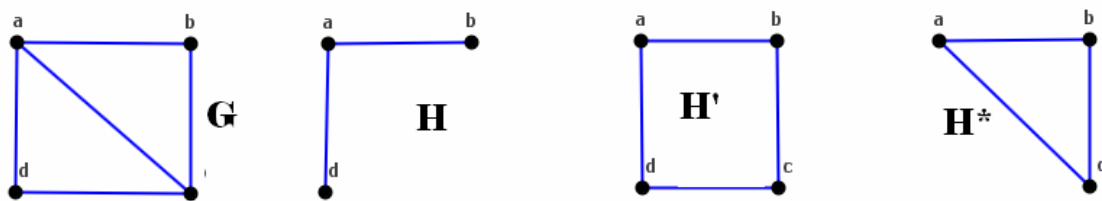


Figura 3: Un grafo G , un subgrafo H , un subgrafo generador H' y el subgrafo H^* inducido por $\{a, b, c\}$

Si x es un vértice del grafo $G = (V, A)$, se indica por $G - \{x\}$, o simplemente $G - x$, al subgrafo de G que tiene como conjunto de vértices $V - \{x\}$ y como aristas todas las de G menos las incidentes con x .

Si e es una arista de $G = (V, A)$, se indica por $G - \{e\}$, o simplemente $G - e$, al subgrafo de G que tiene como conjunto de vértices V y $A - \{e\}$ como conjunto de aristas.

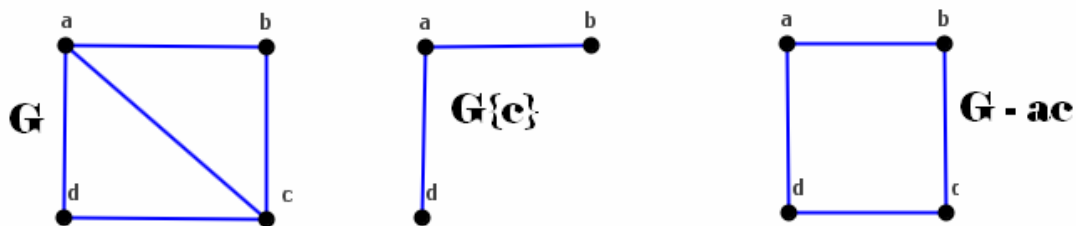


Figura 4: El resultado de eliminar un vértice o una arista

2.3.5. Algunos ejemplos de grafos

• Grafo Completo

Un grafo completo es un grafo simple en el que todo par de vértices está unido por una arista. Se representa con K_n al grafo completo de n vértices. Este grafo tiene $n(n-1)/2$ aristas.

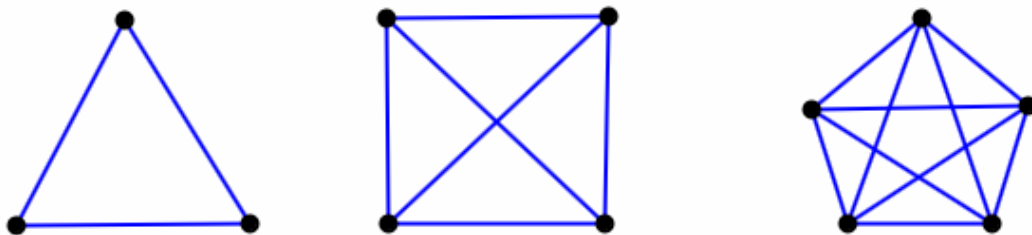


Figura 5: Representaciones de Grafos completos de 3, 4 y 5 vértices (K_3 , K_4 y K_5)

• Grafo Bipartido

Un grafo $G=(V,A)$ se llama bipartido (o bipartito) si existe una partición de V , $V=X \cup Y$, tal que cada arista de G une un vértice de X con otro de Y .

Se designa por $K_{r,s}$ al grafo bipartido completo en que $|X| = r$, $|Y| = s$, y hay una arista que conecta cada vértice de X con cada vértice de Y . Este grafo tiene rs aristas.

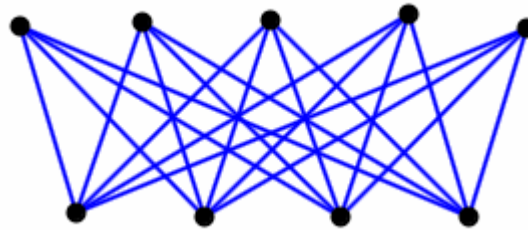


Figura 6: Grafo Bipartito Completo $K_{5,4}$

• Grafo Regular

Un grafo regular es un grafo simple cuyos vértices tienen el mismo grado. Recordemos que se llama grado de un vértice v al número de aristas incidentes en él, y que se denota por $d(v)$.

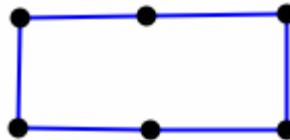


Figura 7: Grafo Regular de grado 2

2.3.6. Grafos conexos y componentes conexas.

Un grafo es conexo si para cada par de vértices u y v existe un camino de u a v . En caso contrario se dirá que el grafo es no conexo (o desconexo).

Si G es un grafo no conexo, cada uno de sus subgrafos conexos maximales se llama componente conexa de G (con subgrafo conexo maximal queremos decir que si añadimos al mismo cualquier otro vértice de G , el subgrafo deja de ser conexo).

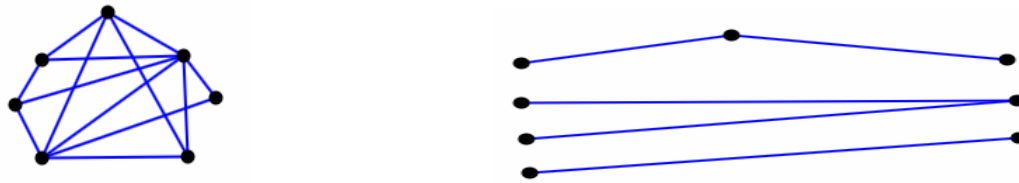


Figura 8: Grafo conexo (izquierda). Grafo no conexo con 3 componentes conexas (derecha)

Un vértice v se llama vértice-corte (o punto de articulación) de G si el grafo $G - \{v\}$ tiene más componentes conexas que G .

Una arista a de un grafo G se llama puente si $G - \{a\}$ tiene más componentes conexas que G .

Los bloques de un grafo G son los subgrafos de G sin vértices-corte y maximales con respecto a esta propiedad.

2.3.7. Digrafos.

Un digrafo o grafo dirigido es un par $D = (V, A)$ donde V es un conjunto no vacío (a cuyos elementos llamaremos vértices o nodos) y A es una familia finita de pares ordenados (u, v) de vértices de V (a cuyos elementos llamaremos aristas).

Un digrafo simple es un par $D = (V, A)$ donde V es un conjunto no vacío y A es un conjunto finito de pares ordenados (u, v) de vértices distintos de V .

Si $a = (u, v)$ es un arco escribiremos $a = uv$, y diremos que u es extremo inicial de a y que v es extremo final de a .

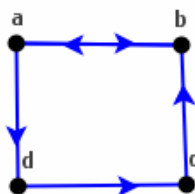


Figura 9: Digrafo simple

Se llama grado de entrada de un vértice v al número de arcos que lo tienen como extremo final y se llama grado de salida de v al número de arcos que lo tienen como extremo inicial. Se designan por de y ds , respectivamente. Podemos ver que el grado de entrada y de salida para los vértices b y c , del digrafo simple representado en la Figura 9 es el siguiente:

$$de(c) = 1 \quad ds(c) = 1 \quad de(b) = 2 \quad ds(b) = 1$$

2.4. Coloración de Grafos

Hay muchos problemas, como la asignación de tareas y los problemas de almacenamiento, donde es necesario partir el conjunto de vértices (aristas) de un grafo asociado, de tal forma que vértices (aristas) adyacentes pertenezcan a diferentes conjuntos de la partición. Tales particiones se llaman coloraciones.

Los problemas sobre coloración de grafos fueron, en la segunda mitad del siglo XIX, uno de los hitos iniciales de la Teoría de Grafos. En aquel tiempo se planteó uno de los problemas clásicos, “**El Problema de los cuatro colores**”, que no se pudo resolver hasta 1976, con ayuda de ordenadores.

2.4.1. El Teorema de los cuatro colores.

El problema fue planteado en 1852 por el estudiante Francis Guthrie, cuando observo que podía colorear el mapa de los condados ingleses utilizando solo cuatro colores. En la coloración cada condado recibía un color y dos condados que compartían frontera (no solo un punto) recibían colores diferentes. El problema fue comunicado a Augustus De Morgan quien lo propuso a la sociedad matemática de la época. La conjetura se hizo famosa con la declaración de Arthur Cayley, en 1878, en el sentido de que la había abordado.

En 1976 la conjetura de los cuatro colores tomó el rango de teorema. Dos matemáticos, Appel y Haken, con la ayuda de Koch, un experto en computación, utilizaron el método de las cadenas de Kempe, junto con otras técnicas ideadas por Heesch en 1969, para implementar un programa de ordenador. Con más de 1.500 configuraciones posibles de mapas, y después de trabajar durante 1.200 horas, el programa probó que cuatro colores eran suficientes para colorear cualquier mapa. Fue la primera demostración de un teorema matemático que se llevaba a cabo en las tripas de un ordenador. No obstante, la demostración completa del teorema no se ha escrito nunca. No valdría mucho la pena, ya que todo el tiempo que dura la vida de una persona no bastaría para poder leerla.

El Teorema establece lo siguiente: “Dado cualquier mapa geográfico con regiones continuas, éste puede ser coloreado con cuatro colores diferentes, de forma que no queden regiones adyacentes (es decir, regiones que compartan no sólo un punto, sino todo un segmento de borde en común) con el mismo color.”

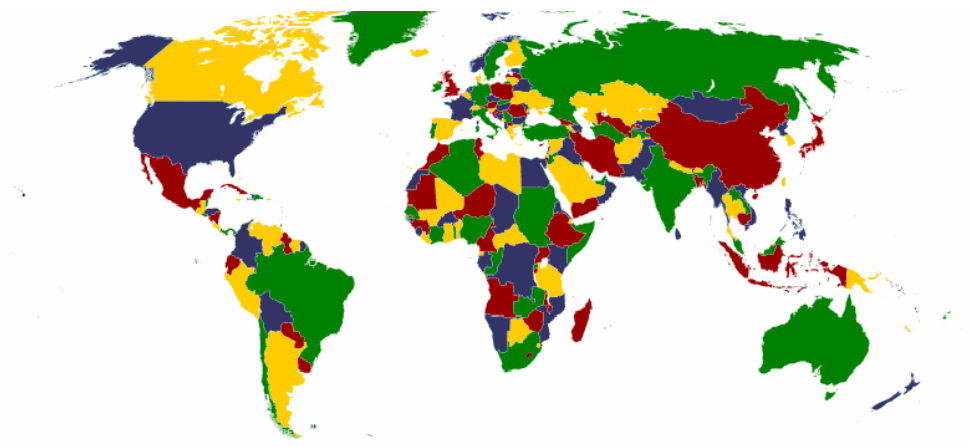


Figura 10: Mapa del mundo coloreado de verde, amarillo, azul y rojo

2.4.2. Concepto de Independencia.

Antes de comenzar con la coloración de vértices vamos a introducir el concepto de independencia.

Un conjunto S de vértices de un grafo G es independiente si no hay dos vértices de S que sean adyacentes en G .

Supongamos que se deben almacenar p productos químicos C_1, C_2, \dots, C_p . Algunos de estos productos no se pueden almacenar en el mismo depósito pues reaccionan entre sí. ¿Cuál es el mínimo número de depósitos que son necesarios para almacenar estos productos químicos de forma que sustancias que reaccionen se almacenen en diferentes depósitos?. Podemos modelar esta situación con un grafo G cuyos vértices sean los productos químicos. Dos vértices se unen por una arista si las correspondientes sustancias reaccionan. Así productos en el mismo depósito corresponden a un conjunto independiente de vértices G .

Un conjunto independiente S de vértices de un grafo G se llama un conjunto independiente maximal si S no es subconjunto propio de ningún conjunto independiente de vértices de G . El máximo cardinal de un conjunto independiente se llama número de independencia de G y se designa por $\beta(G)$. Cada conjunto independiente de cardinal $\beta(G)$ es maximal, pero el recíproco no es cierto. Por ejemplo, en $G=K_{3,4}$ los vértices del primer nivel forman un conjunto de cardinalidad 3 que es independiente maximal, pero $\beta(G)=4$.

Una claque en un grafo G es un subgrafo completo maximal, es decir, una claque es un subgrafo completo que no es subgrafo propio de otro subgrafo completo de G . El máximo orden de una claque de G es el número de claque de G y se designa por $\omega(G)$.

2.4.3. Coloración de Vértices

Una coloración de un grafo G es una asignación de colores a los vértices de G , a cada vértice un color, de forma que vértices adyacentes reciban colores distintos.

Si en la coloración se usan k colores diremos que es una k -coloración. Las coloraciones siempre existen, pues podemos asignar a cada vértice del grafo un color diferente si fuera necesario.

Cada coloración de G produce en $V(G)$ una partición en conjuntos independientes denominados clases de color.

Si existe una k -coloración de G se dice que el grafo G es k -coloreable. El mínimo k para el que un grafo G es k -coloreable se llama número cromático de G , y se designa por $\chi(G)$.

Podemos decir para simplificar, que la coloración de los vértices de un grafo, se basa en encontrar grupos de vértices que no sean adyacentes entre sí.

Afirmar que $\chi(G)$ es el número cromático de un grafo significa que podemos colorear el grafo con χ colores y que no existe una coloración con menor número de colores.

Si un grafo G tiene n vértices siempre podemos colorear los vértices de G con n colores, uno distinto para cada vértice, por lo que $\chi(G) \leq n$.

La cota anterior se alcanza en los grafos completos, que necesitan un color diferente en cada uno de sus vértices, pues dos cualesquiera de ellos son adyacentes.

2.4.4. Algoritmos de Coloración de Grafos

No es fácil determinar el número cromático de un grafo. De hecho, el siguiente problema de decisión, conocido por Chromatic Number Problem y abreviadamente por CN, es un problema NP-completo:

CN: Dado un grafo G y un entero k , ¿es cierto que $\chi(G) \leq k$?

Se conocen, además de la proporcionada por el teorema de Brooks, otras cotas para el número cromático de un grafo, pero en todos los casos existen grafos cuyo número cromático está lejos de la cota marcada.

Por ello y como CN es NP-completo, no sorprende que no se conozca ningún algoritmo eficiente para colorear los vértices de un grafo con $c\chi(G)$ colores, siendo c una constante de valor positivo.

Algoritmos eficientes existen, sin embargo, para colorear grafos de forma que el número de colores usados esté “próximo” a su número cromático. Las heurísticas que se utilizan en estos algoritmos son las siguientes:

- 1.- Colorear un vértice de grado alto es más difícil que otro de grado bajo
- 2.- Los vértices con los mismos vecinos deben colorearse al mismo tiempo
- 3.- Asignar a muchos vértices el mismo color es una buena idea.

A continuación vamos a presentar una serie de algoritmos de coloración de grafos. Comenzaremos explicando una serie de algoritmos de coloración secuenciales para finalizar con un algoritmo de coloración de conjuntos independientes.

2.4.5. Algoritmo de Coloración Secuencial Básico

El orden en que se colorean los vértices se decide antes de que se empiece a colorearlos.

Dada una ordenación de los vértices del grafo, los algoritmos secuenciales asignan el mínimo color posible al siguiente vértice. Es decir, si queremos colorear v , teniendo ordenados numéricamente los colores, asignamos a v el color más pequeño que no aparece entre los asignados a los vecinos de v ya coloreados.

Podemos resumir el funcionamiento del algoritmo Secuencial Básico de la siguiente manera:

Entrada: Una ordenación de los vértices de un grafo G

Salida: Una coloración de los vértices

Paso 1: Asignar el color 1 al vértice v_1

Paso 2: Si hemos coloreado v_1, v_2, \dots, v_k con j colores, asignamos a v_{k+1} el color t donde $t \leq j+1$ es el mismo color permitido para v_{k+1} , según los colores ya asignados a sus vecinos.

A continuación se va a mostrar la evolución del algoritmo de coloración secuencial básico, sobre un grafo.

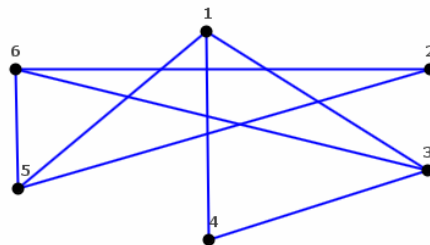


Figura 11: Grafo sobre el que se ejecutará el algoritmo de coloración básico

Se ha decidido que el orden de coloración de los vértices sea el siguiente:
2, 3, 6, 4, 1, 5

En la Figura 12, podemos observar la evolución que va sufriendo el grafo, debido a la ejecución del algoritmo de coloración secuencial básico.

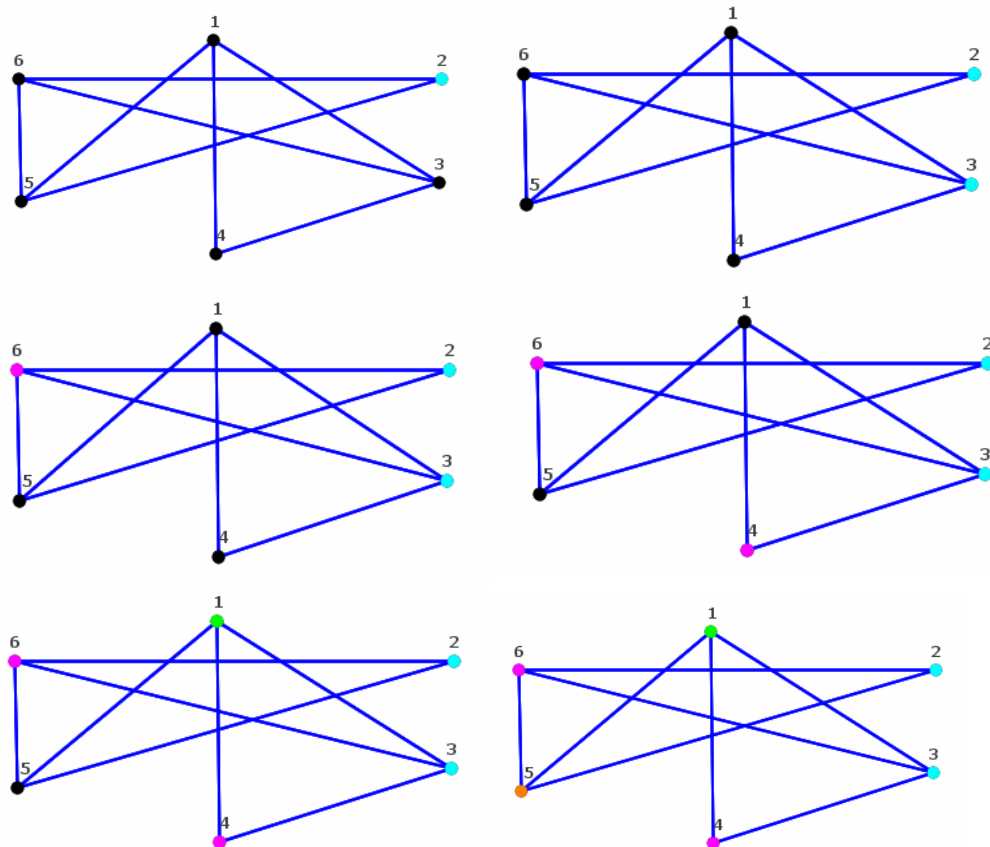


Figura 12: Visualización de los pasos de la ejecución del algoritmo secuencial básico.

Podemos comprobar que esta ordenación origina una 4-coloración.

Los grupos de colores quedan de la siguiente manera:

- 1º grupo: vértices 2 y 3
- 2º grupo: vértices 4 y 6
- 3º grupo: vértice 1
- 4º grupo: vértice 5

2.4.6. Algoritmo de Coloración de Welsh y Powell

Es una variante del algoritmo de coloración secuencial básico, también conocida como “Primero el de mayor grado”.

Es debida a Welsh y Powell, y en este algoritmo, los vértices se ordenan inicialmente de acuerdo a sus grados. Es decir, ordenamos de forma que $d(v_1) \geq d(v_2) \geq \dots \geq d(v_n)$.

Obtendremos el siguiente resultado:

$$X(G) \leq 1 + \max \{ \min(d(v_i), i-1) / i=1, \dots, n \}$$

Si tomamos como ejemplo el grafo de la Figura 11, el orden que se establecerá para colorear los vértices será el siguiente:

1, 3, 5, 6, 2, 4

En la Figura 13, podremos comprobar el estado final del grafo, una vez que se le ha aplicado el algoritmo de Welsh-Powell. Se puede apreciar que si se colorean primero los vértices de mayor grado, el algoritmo secuencial da una 3-coloración.

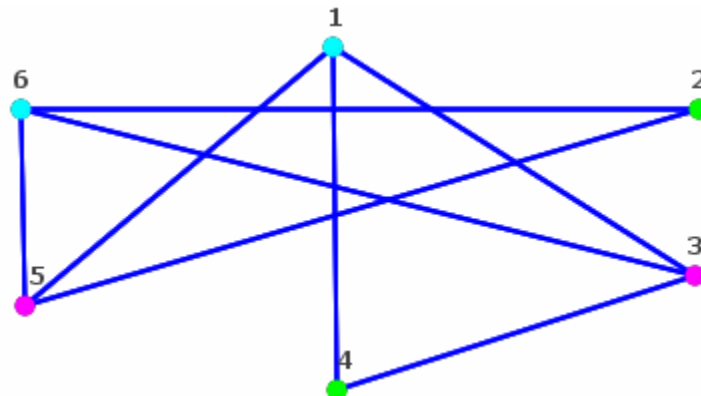


Figura 13: Coloración del grafo usando el algoritmo de Welsh-Powell

2.4.7. Algoritmo de Coloración de Matula, Marble, Isaacson

Es una variante del algoritmo de coloración secuencial básico, también conocida como “El de menor grado el último”.

Se debe a Marble, Matula e Isaacson, y en este algoritmo, los vértices se ordenan en orden inverso. Primero se elige v_n como el vértice de menor grado, luego se elige v_{n-1} como el vértice de menor grado en $G - \{v_n\}$, y así se continúa recursivamente, examinando los vértices de menor grado y eliminándolos del grafo.

Si tomamos como ejemplo el grafo de la Figura 11, el orden que se establecerá para colorear los vértices será el siguiente:

6, 5, 3, 1, 4, 2

En la Figura 14, podremos comprobar el estado final del grafo, una vez que se le ha aplicado el algoritmo de Marble, Matula e Isaacson. Se puede apreciar que este algoritmo da una 3-coloración.

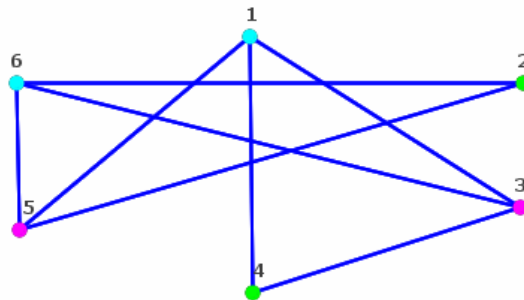


Figura 14: Coloración del grafo usando el algoritmo de Matula, Marble, Isaacson

2.4.8. Algoritmo de Brelaz

En los algoritmos anteriores, la única información que se tiene de cada vértice es su grado. En el algoritmo de Brelaz, tendremos en cuenta también la suma de los grados de los vecinos de cada vértice y los colores ya asignados a dichos vecinos.

Definimos el **grado de color** de un vértice v como el número de colores usados en los vecinos de v . El orden en que iremos coloreando vértices depende del grado y del grado de color.

Podemos resumir el funcionamiento del algoritmo de Brelaz de la siguiente manera:

Entrada: Un grafo G .

Salida: Una coloración de los vértices de G .

Paso 1: Ordenar los vértices en orden decreciente de grados.

Paso 2: Coloreamos un vértice de grado máximo con el color 1.

Paso 3: Seleccionamos un vértice, aún sin colorear, con grado de color máximo. Si hay varios, elegimos el de grado máximo.

Paso 4: Colorear el vértice seleccionado en el paso 3 con el menor color posible.

Paso 5: Si todos los vértices se han coloreado, FIN. En caso contrario, volver al paso 3.

A continuación vamos a ver la evolución del algoritmo de Brelaz sobre un grafo. Adicionalmente a la evolución del grafo, veremos una tabla, con información relativa al algoritmo. Esta información estará compuesta por el número del vértice, el grado del vértice, el grado de color del vértice, el orden en que ha sido coloreado y un identificador del color con el que se ha coloreado.

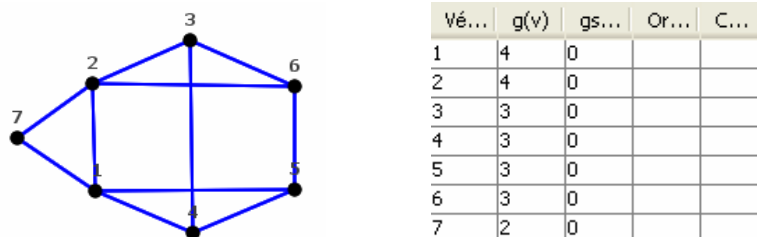


Figura 15: Situación inicial, antes de la ejecución del algoritmo de Brelaz

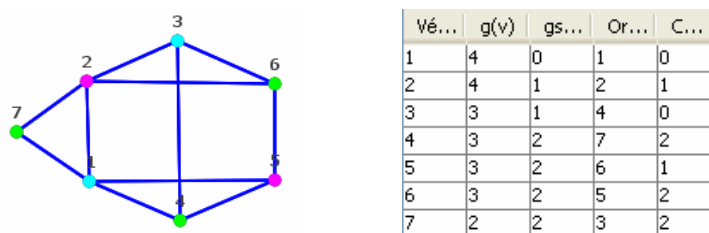


Figura 16: Situación final, después de la ejecución del algoritmo de Brelaz

2.4.9. Algoritmo de Independencia MCI

En los algoritmos secuenciales se colorean los vértices de uno en uno, con lo que el ataque al problema de la coloración se intenta vía el número de claque, atendiendo a la acotación $\chi(G) \geq \omega(G)$. Pero tenemos otra acotación para el número cromático. Recordemos que el número de independencia $\beta(G)$ nos indica que ninguna clase de color, vértices igualmente coloreados, puede tener más de $\beta(G)$ elementos. Por tanto, si designamos por n al número de vértices se tiene que $\chi(G) \geq n / \beta(G)$. Esta cota tiene a ser mejor que la anterior para grafos grandes.

Estas ideas sugieren algoritmos en los que, en un primer paso se localice un conjunto independiente de vértices de cardinal próximo a $\beta(G)$, se colorean todos con el color 1, después se borre este conjunto de vértices y se repita el proceso en el grafo que resulte, continuando así hasta colorear todos los vértices.

Una forma de obtener los conjuntos independientes de un grafo es la siguiente.

Paso 1: Obtener el vértice con menor grado

Paso 2: Añadir el vértice al conjunto independiente actual

Paso 3: Eliminar del grafo el vértice seleccionado y sus vecinos.

Paso 4: Si el grafo no tiene ningún vértice, ir al Paso 5, si el grafo tiene vértices, volver al Paso 1.

Paso 5: Eliminar del grafo los vértices pertenecientes al conjunto independiente actual.

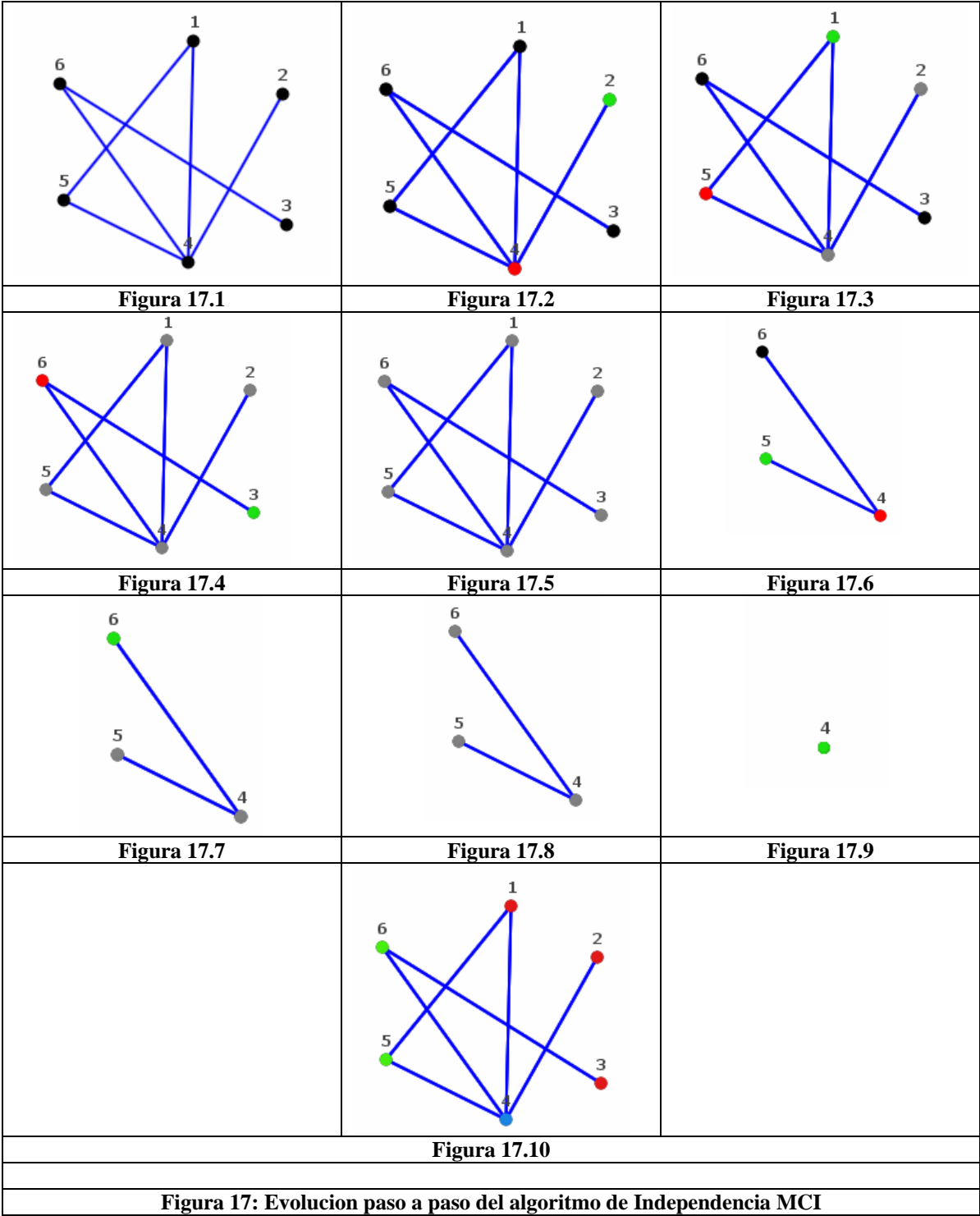
Paso 6: Si el grafo no tiene ningún vértice, se han obtenido todos los conjuntos independientes, ir al Paso 7. Si el grafo tiene vértices, inicializar el conjunto independiente actual y volver al Paso 1.

Paso 7: Fin del algoritmo.

A continuación vamos a ver la evolución, paso a paso, del algoritmo MCI (implementado en la aplicación).

Podemos observar en la Figura 17.1, como se selecciona el vértice 2, ya que es el que tiene menor grado, y como se marcan sus vértices vecinos, para no poder ser seleccionados como elementos pertenecientes al conjunto independiente actual. En la Figura 17.5, vemos que ya no tenemos más vértices con opciones de formar parte del conjunto independiente, con lo que en la Figura 17.6, se modifica el grafo, dejando únicamente los vértices que no forman parte de ningún conjunto independiente.

Este proceso se repite hasta llegar a la situación final (Figura 17.10) donde podemos observar el resultado final de la coloración de conjuntos independientes.



3. DISEÑO DE LA APLICACIÓN

3.1. Lenguaje de programación

Para el desarrollo de la aplicación se ha utilizado el lenguaje de programación Java. Se ha elegido este lenguaje porque, además de ser adecuado para implementar la aplicación que se pretendía desarrollar (condición que cumplen también otros lenguajes de programación), sus características nos han parecido muy interesantes. Algunas de las características más importantes de este lenguaje son las siguientes:

- Es orientado a objetos. Por lo tanto, al igual que otros lenguajes orientados a objetos, tiene la ventaja de que facilita la reutilización y la extensión del código.
- Es multiplataforma, es decir, un programa escrito en Java se pueda ejecutar en múltiples plataformas. El compilador de Java produce un código de bytes que puede ser ejecutado por un intérprete denominado *máquina virtual de Java*; hoy en día, casi todas las compañías de sistemas operativos y de navegadores han implementado máquinas virtuales según las especificaciones publicadas por Sun Microsystems, propietario de Java, para que sean compatibles con el lenguaje Java.
- Evita muchas preocupaciones al programador. La administración de la memoria es automática; el programador no tiene que ocuparse de la liberación de memoria reservada dinámicamente, ya que un *recolector de basura* libera la memoria asignada a un objeto cuando ya no exista ninguna referencia a ese objeto. Y en el proceso de compilación se realizan numerosas comprobaciones que permiten eliminar muchos errores posteriores.
- Entorno de desarrollo gratuito. Sun Microsystems proporciona un entorno de desarrollo Java, *Java Development Kit* (JDK), de forma gratuita.
- El programador no parte de cero, ya que dispone de una API (Interfaz de Programación de Aplicaciones) que ofrece numerosas clases e interfaces, agrupadas en distintos paquetes, listas para que el programador las utilice en sus propias aplicaciones y que abarcan desde los objetos básicos hasta la generación de XML y el acceso a bases de datos. Cabe destacar la utilidad, para la realización de este proyecto, de los paquetes para crear interfaces de usuario y pintar gráficos.

3.2. Diseño de Alto Nivel

El objetivo de este apartado es explicar el diseño empleado en la construcción de la aplicación. Como su nombre indica, no consiste en un diseño exhaustivo de cada una de las líneas de código del programa, sino una visión general del proceso realizado. Este proceso viene determinado por los siguientes puntos:

- Definición de los límites del sistema.
- Diagrama de casos de uso.
- Descripción de los casos de uso en formato extendido

Los objetivos de esta memoria no incluyen el realizar una documentación perfeccionista de Ingeniería del Software, por lo que describiremos el proceso de manera que se pueda entender fácilmente y sin estar sujeto a las rigurosas normas de nomenclatura que precisan dichos documentos.

3.2.1. Definición de los límites del sistema

En este apartado se delimitan las funcionalidades que realiza la aplicación. Todo lo que no esta definido aquí no lo realiza el sistema.

La herramienta permite crear, editar y borrar grafos, sobre los que se pueden ejecutar algoritmos, tanto de coloración (secuencial, Welsh, Matula, Brelaz e Independencia), como de búsqueda (DFS, BFS, conectividad y Dijkstra).

Además, la aplicación permite obtener datos estadísticos, tanto del grafo como de los algoritmos aplicados.

Como soporte a su utilización, la aplicación permite el guardado de un grafo, incluyendo la posibilidad de almacenar el estado de ejecución de un algoritmo, con el fin de continuar su ejecución en otro momento. El sistema permite generar grafos de forma tanto automática como manual; el usuario puede agregar o borrar aristas y vértices. Como combinación de ambos, una vez generado un grafo, el sistema también permite modificarlo de forma manual. Para mejorar la visibilidad y la maniobrabilidad, la herramienta permite mover los vértices del grafo sobre el panel de dibujo.

Siempre que el usuario quiera, puede visualizar tanto la matriz de adyacencia como la de ponderación del grafo generado.

El sistema ha incluido 10 colores diferentes, por defecto, para realizar las coloraciones. El usuario podrá modificarlos así como añadir hasta un máximo de otros 10 colores diferentes si lo considerase necesario.

La aplicación es multilenguaje, disponible en español e inglés.

3.2.2. Diagramas de casos de uso

Los casos de uso permiten describir el comportamiento de un sistema desde el punto de vista del usuario basándose en un conjunto de acciones y reacciones. Es por lo tanto una técnica que permite capturar los requisitos funcionales del sistema. De esta forma queda delimitado el alcance del sistema y cuál es su relación con el entorno.

En estos diagramas, el sistema queda reducido a una “caja negra”, ya que no interesa cómo lleva a cabo sus funciones, sino simplemente qué acciones visibles desde el exterior son las que realiza.

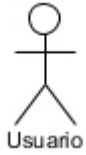
Los casos de uso están basados en lenguaje natural, lo que los hace accesibles a cualquier usuario. Además, aquellos casos de uso que resulten muy complejos pueden descomponerse en nuevos casos de uso de un nivel inferior, hasta llegar a un nivel tal que resulten fáciles de analizar.

Los diagramas de casos de uso están formados por tres elementos principales:

- **Actores.** Los actores son los participantes de los casos de uso. Se corresponden con los usuarios que interactúan con el sistema. Pueden ser humanos, dispositivos externos que interactúen con el sistema, o incluso temporizadores que envíen eventos al mismo. Un actor se caracteriza por la forma de interaccionar con el sistema, por lo que un mismo usuario puede ejercer de varios actores, y un actor puede representar a varios usuarios.
- **Casos de uso.** Son los escenarios de interacción de los actores. Representan el comportamiento del sistema en relación con los usuarios. De esta forma, un caso de uso define la secuencia de interacciones entre uno o más usuarios y el sistema.
- **Relaciones.** Representan el flujo de información intercambiada entre los actores y los casos de uso, o entre diferentes casos de uso. Normalmente, se emplean para que un caso de uso obtenga la información necesaria para llevar a cabo alguna acción, o para que el proceso proporcione algún resultado. Estas relaciones pueden ser unidireccionales o bidireccionales.

Los diagramas de casos de uso se clasifican en diferentes niveles, en función del grado de detalle con el que se represente el funcionamiento del sistema. De esta forma, los diagramas de nivel 0 o contexto representan el sistema completo con un nivel de detalle muy bajo, mientras que al aumentar el nivel, el grado de detalle va incrementándose.

A continuación se muestra la notación empleada para la representación de los distintos elementos que forman los diagramas de casos de uso.



Los actores son representados mediante figuras de *hombre de palo*, con su correspondiente nombre debajo de la figura.



Los casos de uso, o procesos, se representan mediante una elipse, con su nombre correspondiente debajo de la misma.



Las relaciones se representan mediante flechas que unen los casos de uso, o el caso de uso y el actor, entre los que existe un flujo de información.

A continuación presentamos el diagrama de casos de uso de nivel 0:



Figura 18: Diagrama de nivel 0

Debido a su mayor complejidad, se ha decidido descomponer los siguientes casos de uso en subniveles: GenerarGrafo, EstablecerConfiguración y EjecutarAlgoritmo.

3.2.2.1. Generar Grafo

El caso de uso GenerarGrafo se puede descomponer a su vez en tres casos de uso, que se corresponden con la manera en que el usuario puede generar un grafo desde la aplicación:

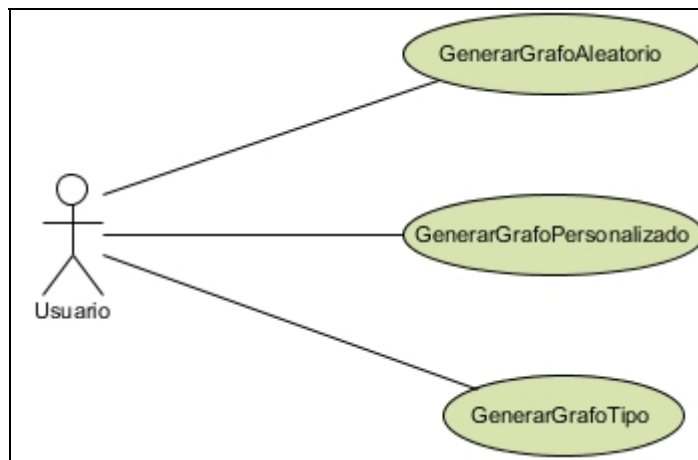


Figura 19: Diagrama de nivel 1: GenerarGrafo

A continuación descomponemos cada uno de estos tres casos de uso, obteniendo los correspondientes diagramas de nivel 2 para cada uno:

- **GenerarGrafoAleatorio**: El usuario introducirá los datos necesarios para generar el grafo (nombre, número de vértices, probabilidad de arista y si es o no dirigido), y el sistema obtendrá un grafo a partir de éstos y lo mostrará de forma gráfica.

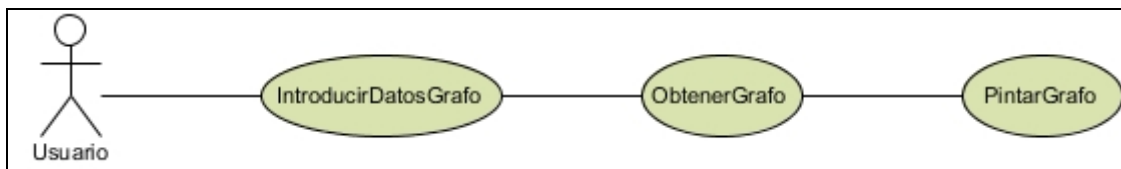


Figura 20: Diagrama de nivel 2: GenerarGrafoAleatorio

- **GenerarGrafoPersonalizado**: El usuario introducirá los datos necesarios para generar el grafo (nombre, número de vértices y si es o no dirigido). A continuación, deberá indicar, para cada vértice, cuáles serán sus vecinos (existirá una arista entre ellos). Dependiendo de si el grafo es o no dirigido, la arista resultante entre dos vértices tendrá o no sentido. El sistema obtendrá un grafo a partir de los datos anteriormente introducidos y lo mostrará de forma gráfica.



Figura 21: Diagrama de nivel 2: GenerarGrafoPersonalizado

- **GenerarGrafoTipo**: Dependiendo del grafo tipo que el usuario desee crear, deberá introducir un determinado tipo de datos u otro (nombre, número de vértices, niveles, nivel de enlace, grado, altura, etc.) para su generación. A continuación el sistema obtendrá un grafo a partir de éstos y lo mostrará de forma gráfica.

Los grafos tipo que el usuario puede generar son:

- Grafo nulo
- Grafo completo: completo K_n , bipartito y tripartito
- Grafo de rejilla: rectangular y triangular
- Grafo platónico: tetraedro, hexaedro, octaedro, icosaedro y dodecaedro
- Grafo enlazado: L_n y $L_{n,r}$
- Grafo rueda
- Grafo de estrella
- Grafo cubo
- Grafo de Harary
- Grafo de Petersen
- Grafo de Grötzsch
- Grafo de Tutte
- Grafo de Heawood
- Grafo de Herschel

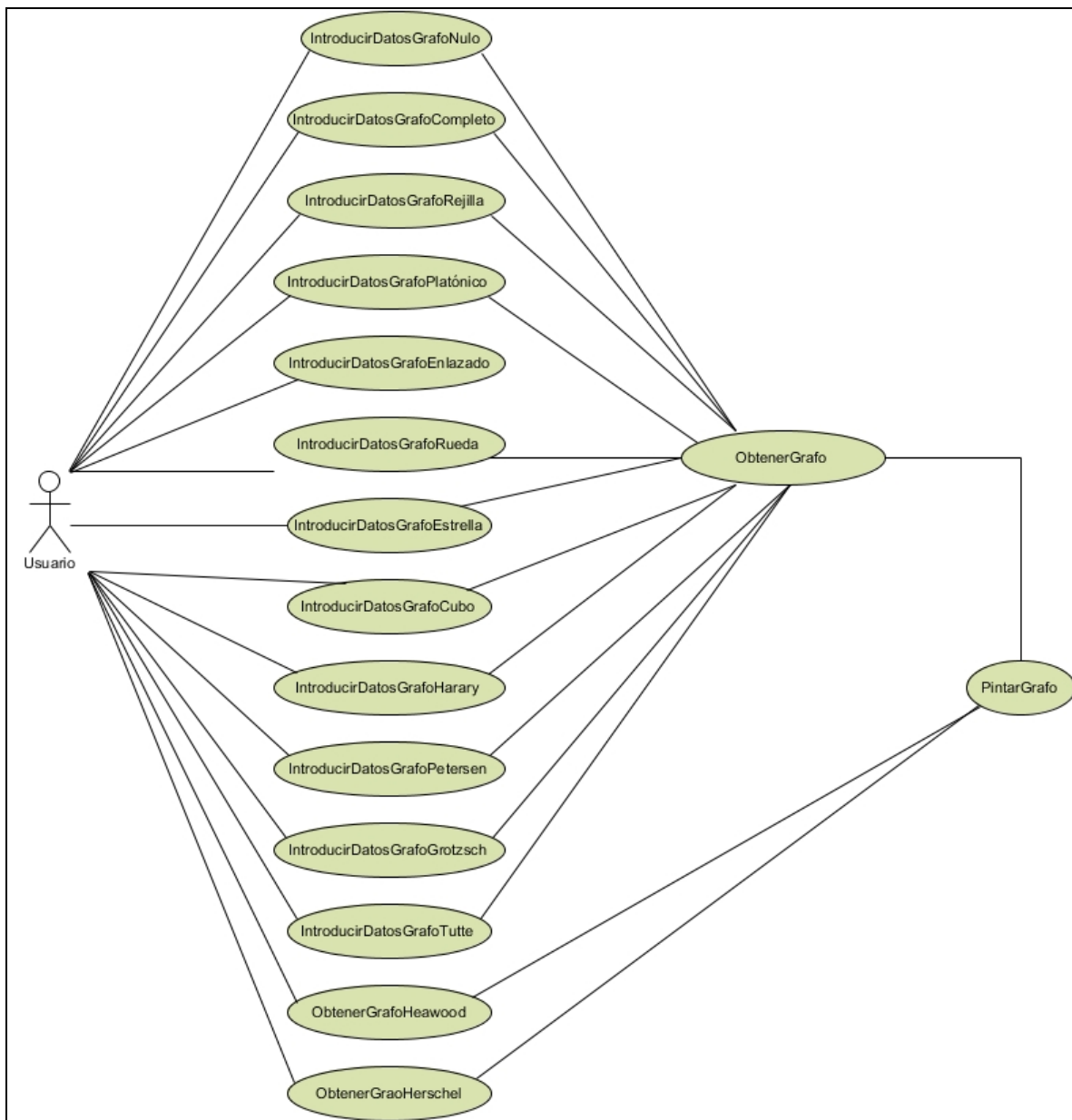


Figura 22: Diagrama de nivel 2: GenerarGrafoTipo

3.2.2.2. EstablecerConfiguración

El caso de uso EstablecerConfiguración lo vamos a descomponer a un nivel inferior para comprender mejor su funcionamiento:

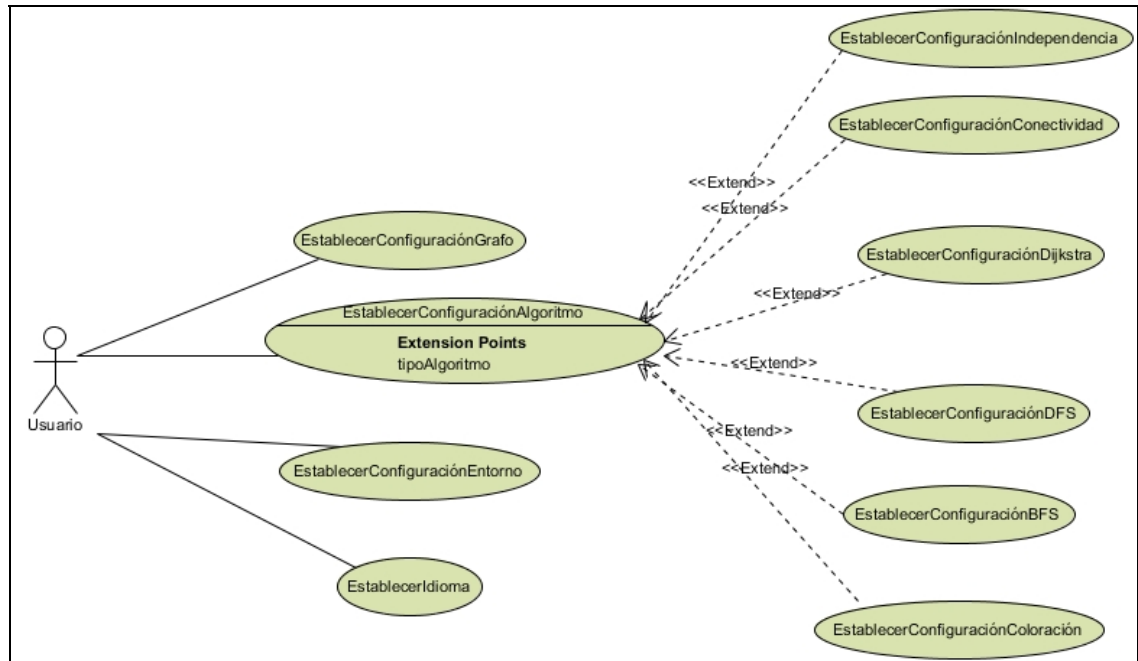


Figura 23: Diagrama de nivel 1: EstablecerConfiguración

Podemos ver que este caso de uso se descompone a su vez en cuatro casos de uso principales:

- **EstablecerConfiguraciónGrafo**, donde el usuario podrá establecer las opciones características del grafo (color de vértice, color de arista, posición de la flecha para el caso de dígrafos, etc.)
- **EstablecerConfiguraciónEntorno**, donde el usuario podrá establecer el número de pasos hacia atrás que el sistema puede dar durante la ejecución de un algoritmo.
- **EstablecerIdioma**, donde el usuario puede modificar el idioma de la aplicación.
- **EstablecerConfiguraciónAlgoritmo**, compuesto a su vez de otros seis casos de uso (que lo extienden), uno por cada algoritmo ejecutable, donde el usuario podrá establecer las opciones características de cada algoritmo.

3.2.2.3. EjecutarAlgoritmo

El caso de uso EstablecerConfiguración lo vamos a descomponer a un nivel inferior para comprender mejor su funcionamiento:

El caso de uso EjecutarAlgoritmo se puede descomponerse en un nivel inferior para una mayor comprensión:

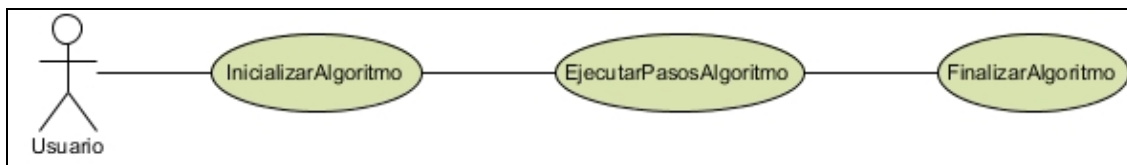


Figura 24: Diagrama de nivel 1: EjecutarAlgoritmo

Como puede apreciarse, el usuario inicializará el algoritmo seleccionado, introduciendo los datos iniciales (si fuesen requeridos). El sistema ejecutará los pasos necesarios hasta que el algoritmo finalice.

3.2.3. Descripción de los casos de uso en formato extendido

En esta sección expondremos los casos de uso con un nivel de detalle mayor, especificando cuáles son las interacciones habituales entre los actores de la aplicación.

El formato que utilizaremos, emplea los siguientes campos:

- Caso de Uso: Nombre del caso de uso.
- Objetivo: Explicación del caso del objetivo del caso de uso
- Actor principal: En este caso el Usuario
- Precondiciones: Condiciones que deben cumplirse antes de comenzar el escenario principal de éxito.
- Escenario principal de éxito: Pasos detallados de las interacciones entre actores.
- Garantías de éxito: Qué debe cumplirse cuando el caso de uso se acaba con éxito.
- Garantías de fracaso: Qué debe cumplirse cuando el caso de uso se abandona.

Caso de uso	AbrirGrafo
Objetivo	Crear un nuevo grafo cargado desde fichero
Actor principal	Usuario
Precondiciones	El fichero XML seleccionado debe seguir el formato correcto de descripción de un grafo (puede haber sido generado a partir de la funcionalidad de salvar grafo)
Escenario de éxito	<ol style="list-style-type: none"> 1. El usuario selecciona la opción Abrir grafo. 2. El sistema pide la ruta de acceso al fichero que contiene los datos del grafo. 3. El usuario indica la ruta 4. El sistema verifica la coherencia del fichero. 5. El sistema genera un nuevo grafo a partir del fichero suministrado. 6. El sistema muestra el grafo pintado sobre el lienzo.
Garantías de éxito	Se habrá creado un nuevo grafo a partir del fichero suministrado y se mostrará pintado sobre el lienzo.
Garantías de fracaso	El estado anterior al caso de uso en el que estaba el sistema.

Caso de uso	SalvarGrafo
Objetivo	Guardar el grafo, representado sobre el lienzo, en un fichero.
Actor principal	Usuario
Precondiciones	Debe existir un grafo
Escenario de éxito	<ol style="list-style-type: none"> 1. El usuario selecciona la opción Salvar grafo. 2. El sistema pide la ruta donde se almacenará el fichero que contendrá los datos del grafo. 3. El usuario indica la ruta 4. El sistema serializa el grafo en una cadena XML. 5. El sistema genera un fichero XML con la cadena generada y lo guarda en la ruta especificada.
Garantías de éxito	Se habrá creado un fichero XML que contendrá el grafo serializado.
Garantías de fracaso	El estado anterior al caso de uso en el que estaba el sistema.

Caso de uso	CerrarGrafo
Objetivo	Eliminar el grafo representado sobre el lienzo.
Actor principal	Usuario
Precondiciones	Debe existir un grafo
Escenario de éxito	<ol style="list-style-type: none"> 1. El usuario selecciona la opción Cerrar grafo. 2. El sistema pregunta si se desea salvar el grafo. 3.a. El usuario decide guardar el grafo (ver caso de uso relacionado; continuará en el punto 4 tras la ejecución del caso de uso) 3.b. El usuario no decide guardar el grafo 4. El sistema elimina el grafo de memoria y deja el lienzo en blanco.
Garantías de éxito	Se habrá eliminado el grafo de memoria y el lienzo se mostrará en blanco.
Garantías de fracaso	El estado anterior al caso de uso en el que estaba el sistema.

Caso de uso	EditarNodo
Objetivo	Editar las características del vértice (aristas de entrada y salida)
Actor principal	Usuario
Precondiciones	Debe existir un grafo
Escenario de éxito	<ol style="list-style-type: none"> 1. El usuario selecciona el vértice a editar. 2. El sistema muestra la pantalla de edición de nodos. 3. El usuario añade o elimina vértices vecinos. 4. El sistema añade o elimina aristas del grafo en función de las decisiones tomadas por el usuario, y muestra el grafo resultante sobre el lienzo.
Garantías de éxito	Se habrá modificado el grafo y el lienzo reflejará los cambios.
Garantías de fracaso	El estado anterior al caso de uso en el que estaba el sistema.

Caso de uso	ExportarGrafo
Objetivo	Exportar la información del grafo a un fichero Excel.
Actor principal	Usuario
Precondiciones	Debe existir un grafo
Escenario de éxito	<ol style="list-style-type: none"> 1. El usuario selecciona la opción de Exportar grafo. 2. El sistema pide la ruta donde se almacenará el fichero que contendrá los datos del grafo. 3. El usuario indica la ruta 4. El sistema obtiene los datos del grafo. 5. El sistema genera un fichero Excel con los datos y lo guarda en la ruta especificada.
Garantías de éxito	Se habrá creado un fichero Excel que contendrá los datos exportados del grafo.
Garantías de fracaso	El estado anterior al caso de uso en el que estaba el sistema.

Caso de uso	LimpiarAlgoritmo
Objetivo	Eliminar toda la información relativa al algoritmo que se está ejecutando sobre el grafo, tanto en memoria como sobre el lienzo.
Actor principal	Usuario
Precondiciones	Debe existir un grafo y se debe estar ejecutando un algoritmo.
Escenario de éxito	<ol style="list-style-type: none"> 1. El usuario selecciona la opción de Limpiar algoritmos. 2. El sistema elimina los datos relativos al algoritmo. 3. El sistema muestra sobre el lienzo el grafo original.
Garantías de éxito	Se habrá eliminado toda información relativa al algoritmo que se estaba ejecutando y se mostrará el grafo original sobre el lienzo.
Garantías de fracaso	El estado anterior al caso de uso en el que estaba el sistema.

Caso de uso	CambiarTipoProyecto
Objetivo	Cambiar el tipo de proyecto (búsquedas o coloración) para que el usuario pueda ejecutar otro conjunto de algoritmos.
Actor principal	Usuario
Precondiciones	No debe existir un algoritmo en ejecución.
Escenario de éxito	<ol style="list-style-type: none"> 1. El usuario selecciona otro tipo de algoritmo al actual. 2. El sistema registra la información del nuevo tipo de proyecto a ejecutar.
Garantías de éxito	Se habrá cambiado el tipo de proyecto a ejecutar.
Garantías de fracaso	El estado anterior al caso de uso en el que estaba el sistema.

Caso de uso	ObtenerMatrizAdyacencia
Objetivo	Mostrar la matriz de adyacencia del grafo.
Actor principal	Usuario
Precondiciones	Debe existir un grafo.
Escenario de éxito	<ol style="list-style-type: none"> 1. El usuario selecciona de Mostrar matriz de adyacencia. 2. El sistema genera la matriz de adyacencia. 3. El sistema muestra en una nueva ventana la matriz de adyacencia del grafo.
Garantías de éxito	Se muestra en una nueva ventana la matriz de adyacencia del grafo.
Garantías de fracaso	El estado anterior al caso de uso en el que estaba el sistema.

Caso de uso	ObtenerMatrizPonderación
Objetivo	Mostrar la matriz de ponderación del grafo.
Actor principal	Usuario
Precondiciones	Debe existir un grafo.
Escenario de éxito	<ol style="list-style-type: none"> 1. El usuario selecciona de Mostrar matriz de ponderación. 2. El sistema genera la matriz de ponderación. 3. El sistema muestra en una nueva ventana la matriz de ponderación del grafo.
Garantías de éxito	Se muestra en una nueva ventana la matriz de ponderación del grafo.
Garantías de fracaso	El estado anterior al caso de uso en el que estaba el sistema.

Caso de uso	AñadirVértice
Objetivo	Añadir un nuevo vértice.
Actor principal	Usuario
Precondiciones	Ninguna.
Escenario de éxito	<ol style="list-style-type: none"> 1. El usuario selecciona la opción de Añadir vértice. 2. El usuario selecciona sobre el lienzo la posición del nuevo vértice. 3. El sistema añade un nuevo vértice al grafo en la posición indicada y muestra el resultado sobre el lienzo.
Garantías de éxito	Se añade el nuevo vértice en la posición indicada.
Garantías de fracaso	El estado anterior al caso de uso en el que estaba el sistema.

Caso de uso	BorrarVértice
Objetivo	Eliminar el vértice indicado.
Actor principal	Usuario
Precondiciones	Debe existir un grafo.
Escenario de éxito	<ol style="list-style-type: none"> 1. El usuario selecciona la opción de Borrar vértice. 2. El usuario selecciona sobre el lienzo el vértice a borrar. 3. El sistema elimina todas las aristas, entrantes y salientes, del vértice a borrar. 4. El sistema elimina el vértice a borrar. 5. El sistema muestra sobre el lienzo el grafo resultante.
Garantías de éxito	Se modifica el grafo sin el vértice eliminado y sus aristas entrantes y salientes.
Garantías de fracaso	El estado anterior al caso de uso en el que estaba el sistema.

Caso de uso	MoverVértice
Objetivo	Mover el vértice indicado a la posición elegida.
Actor principal	Usuario
Precondiciones	Debe existir un grafo.
Escenario de éxito	<ol style="list-style-type: none"> 1. El usuario selecciona la opción de Mover vértice. 2. El usuario selecciona sobre el lienzo el vértice a mover. 3. El usuario arrastra el vértice seleccionado a la posición deseada. 4. El sistema cambia la posición del vértice seleccionado por la nueva. 5. El sistema muestra sobre el lienzo el grafo resultante.
Garantías de éxito	Se modifica el grafo con el vértice seleccionado en la nueva posición.
Garantías de fracaso	El estado anterior al caso de uso en el que estaba el sistema.

Caso de uso	AñadirArista
Objetivo	Añadir una nueva arista entre el vértice origen y el de destino.
Actor principal	Usuario
Precondiciones	Debe existir un grafo con al menos dos vértices.
Escenario de éxito	<ol style="list-style-type: none"> 1. El usuario selecciona la opción de Añadir arista. 2. El usuario selecciona sobre el lienzo el vértice de origen. 3. El usuario arrastra el puntero del ratón hasta el vértice destino. 4. El sistema añade una nueva arista entre el vértice origen y el de destino del grafo y muestra el resultado sobre el lienzo.
Garantías de éxito	Se añade una nueva arista entre los vértices indicados.
Garantías de fracaso	El estado anterior al caso de uso en el que estaba el sistema.

Caso de uso	BorrarArista
Objetivo	Eliminar la arista indicada.
Actor principal	Usuario
Precondiciones	Debe existir un grafo que contenga al menos una arista.
Escenario de éxito	<ol style="list-style-type: none"> 1. El usuario selecciona la opción de Borrar arista. 2. El usuario selecciona sobre el lienzo la arista a borrar. 3. El sistema elimina la arista del grafo. 4. El sistema muestra sobre el lienzo el grafo resultante.
Garantías de éxito	Se modifica el grafo sin la arista eliminada.
Garantías de fracaso	El estado anterior al caso de uso en el que estaba el sistema.

Generar grafo

Como hemos visto anteriormente este caso de uso ha sido descompuesto en niveles para mejor comprensión. Se ha tomado la decisión de explicar los casos de uso que conforman el nivel 1:

- Generar grafo aleatorio
- Generar grafo personalizado
- Generar grafo tipo

Caso de uso	GenerarGrafoAleatorio
Objetivo	Crear un grafo.
Actor principal	Usuario
Precondiciones	Ninguna.
Escenario de éxito	<ol style="list-style-type: none"> 1. El usuario selecciona la opción de Generar un grafo de forma aleatoria. 2. El sistema muestra una ventana de introducción de datos. 3. El usuario completa la información (número de vértices, probabilidad de arista y si es dirigido). 4. El sistema cierra la ventana y calcula un grafo en base a los datos introducidos por el usuario. 5. El sistema muestra el grafo generado sobre el lienzo.
Garantías de éxito	Se genera un grafo.
Garantías de fracaso	El estado anterior al caso de uso en el que estaba el sistema.

Caso de uso	GenerarGrafoPersonalizado
Objetivo	Crear un grafo.
Actor principal	Usuario
Precondiciones	Ninguna.
Escenario de éxito	<ol style="list-style-type: none"> 1. El usuario selecciona la opción de Generar un grafo de forma personalizada. 2. El sistema muestra una ventana de introducción de datos. 3. El usuario completa la información (número de vértices y si es dirigido). 4. El sistema cierra la ventana y genera internamente un grafo nulo con el número de vértices introducido. 5. El sistema muestra por pantalla una ventana similar a la de edición de nodos (ver caso de uso referido). 6. El usuario añade vértices vecinos para cada nodo del grafo y pulsa el botón de Aceptar. 7. El sistema cierra la ventana, añadiendo al grafo las aristas conforme a las opciones seleccionadas por el usuario. 8. El sistema muestra el grafo generado sobre el lienzo.
Garantías de éxito	Se genera un grafo.
Garantías de fracaso	El estado anterior al caso de uso en el que estaba el sistema.

Caso de uso	GenerarGrafoTipo
Objetivo	Crear un grafo (de los predefinidos en el sistema).
Actor principal	Usuario
Precondiciones	Ninguna.
Escenario de éxito	<ol style="list-style-type: none"> 1. El usuario selecciona el grafo tipo a generar. 2. El sistema muestra una ventana de introducción de datos. 3. El usuario completa la información (dependiendo del grafo tipo se mostrará una información u otra). 4. El sistema cierra la ventana y calcula el grafo tipo en base a los datos introducidos por el usuario. 5. El sistema muestra el grafo generado sobre el lienzo.
Garantías de éxito	Se genera un grafo.
Garantías de fracaso	El estado anterior al caso de uso en el que estaba el sistema.

Caso de uso	EstablecerConfiguración
Objetivo	Modificar la configuración del sistema
Actor principal	Usuario
Precondiciones	Ninguna
Escenario de éxito	<ol style="list-style-type: none"> 1. El usuario selecciona la opción Opciones. 2. El sistema muestra la ventana de opciones. 3. El usuario modifica tantas opciones del sistema como desee 4. El sistema almacena la nueva configuración. 5. El sistema aplica la nueva configuración en el entorno de trabajo.
Garantías de éxito	Se habrá modificado la configuración del sistema, según lo especificado por el usuario. Dichos cambios serán visibles en el entorno de trabajo.
Garantías de fracaso	El estado anterior al caso de uso en el que estaba el sistema.

Caso de uso	EjecutarAlgoritmo
Objetivo	Ejecutar un algoritmo sobre un grafo
Actor principal	Usuario
Precondiciones	Debe existir un grafo
Escenario de éxito	<ol style="list-style-type: none"> 1. El usuario selecciona la opción Ejecutar. 2. El sistema muestra la ventana de ejecución de algoritmos en función del tipo de proyecto seleccionado actualmente. 3. El usuario selecciona el algoritmo a ejecutar e introduce los datos necesarios para inicializar el algoritmo. 4. El sistema carga el estado inicial del algoritmo seleccionado. 5. El usuario inicia la ejecución del algoritmo. 6. El sistema ejecuta los pasos indicados por el usuario (dependiendo del tipo de ejecución), modificando el grafo por pantalla, hasta que se cumple la condición de fin del algoritmo.
Garantías de éxito	Se habrá ejecutado el algoritmo seleccionado sobre el grafo existente.
Garantías de fracaso	El estado anterior al caso de uso en el que estaba el sistema.

3.3. Diseño de Bajo Nivel

Para realizar el diseño de bajo nivel utilizaremos diagramas de clases, que nos permitirán tener una visión global de los diferentes elementos que conforman la aplicación

3.3.1. Diagrama de clases

El diseño del proyecto se ha realizado mediante una metodología orientada a objetos. Algunas de las ventajas que tiene el uso de esta metodología son la reutilización y la posibilidad de realizar una mayor abstracción durante la etapa de diseño. El uso de esta metodología permite abordar problemas más complejos facilitando el mantenimiento y evolutivo de los sistemas. Además es una metodología apropiada para desarrollos iterativos como el aplicado en este proyecto.

El desarrollo del proyecto se ha realizado utilizando el lenguaje de programación Java. La elección de este lenguaje permite desarrollar siguiendo la metodología OO con independencia de la plataforma, y además permite la utilización de sus librerías gráficas (Swing). Se ha optado por seguir el modelo vista-controlador como patrón de diseño.

En los siguientes diagramas de clases se ha incluido la información más relevante, excluyendo las librerías propias de Java, ya que no aportan información adicional.

3.3.1.1. Visión general

En este diagrama se muestra una visión general de la aplicación. A continuación se explican las clases que lo conforman:

- **AIGView:** Extiende de la clase `FrameView` para la construcción de la ventana principal de la aplicación. Aquí es donde se tratarán todos los eventos del sistema.
- **Canvas:** Extiende de la clase `JPanel` para construir el lienzo de la aplicación, donde se dibujarán los grafos y los algoritmos que sobre ellos se apliquen.
- **Controlador:** Es la clase principal de la aplicación, encargada de recibir información tanto de los eventos lanzados desde la vista como del canvas, sirviendo de puente entre ellos. Además, sirve para gestionar el modelo de datos. Conectados al controlador se encuentran otras clases, que serán explicadas en sus diagramas correspondientes para una mejor comprensión. Estos subdiagramas son:
 - **Grafo**
 - **Configuración**
 - **Algoritmo**

Podemos comprobar que estas tres clases utilizan tipos enumerados:

- **eBotonPulsado:** utilizado para enumerar los botones de la ejecución de los algoritmos (paso previo, ejecución, pausa y paso siguiente).
- **eAccion:** utilizado para enumerar los botones que son de aplicación directa sobre el lienzo (agregar un vértice o arista, eliminar un vértice o arista).
- **eResultadoPaso:** utilizado para enumerar los posibles valores tomados tras la ejecución interactiva del paso de un algoritmo.

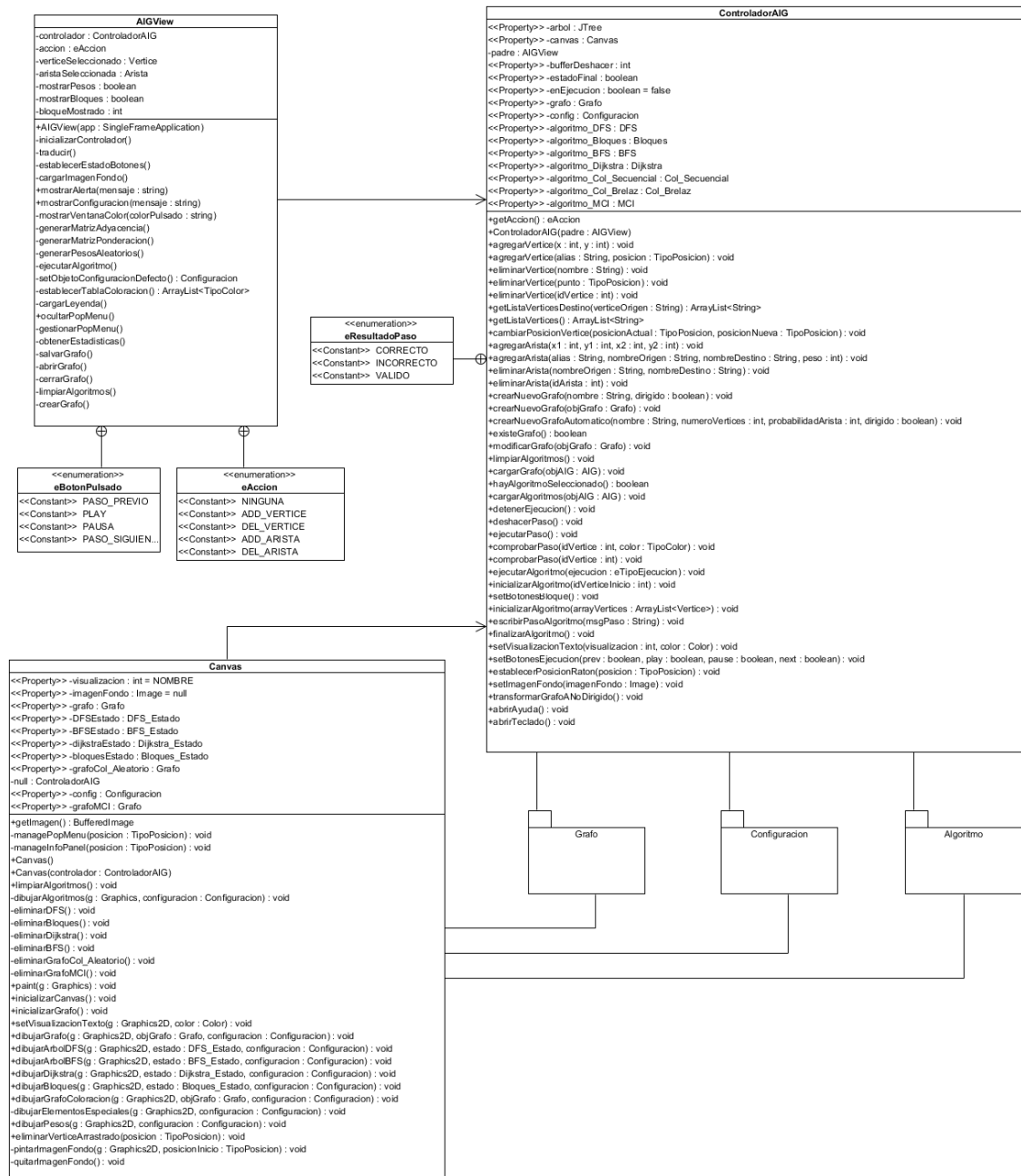


Figura 25: Diagrama de Clases que muestra la Visión General del Proyecto

3.3.1.2. Grafo

En este diagrama se explica cómo está formada la clase Grafo y las clases en las que se apoya (Vertice y Arista, principalmente):

- **Grafo:** Clase que sirve para instanciar el objeto que representa a un grafo. Se ha decidido modelizar un grafo mediante una lista con sus vértices y otra con sus aristas. Utiliza un tipo enumerado para representar la forma en que se genera (manual o automática).
- **Vertice:** Clase que sirve para instanciar el objeto que representa un vértice del grafo. En cada objeto tenemos una lista de vecinos (aquellos vértices del grafo a los que se puede ir, unidos a este objeto mediante una arista. Si el grafo no es dirigido, tenemos una lista de aristas. En cambio, si es dirigido, tenemos una lista de aristas de entrada y otra de aristas de salida. Esta clase se relaciona con las clases TipoPosicion y TipoColor.
- **Arista:** Clase que sirve para instanciar el objeto que representa una arista del grafo. Contiene información de los vértices de origen y destino, así como el peso. Se relaciona con la clase TipoColor.
- **TipoPosicion:** Representa una posición sobre el eje cartesiano (x,y) del lienzo.
- **TipoColor:** Representa un color mediante sus componentes RGB.

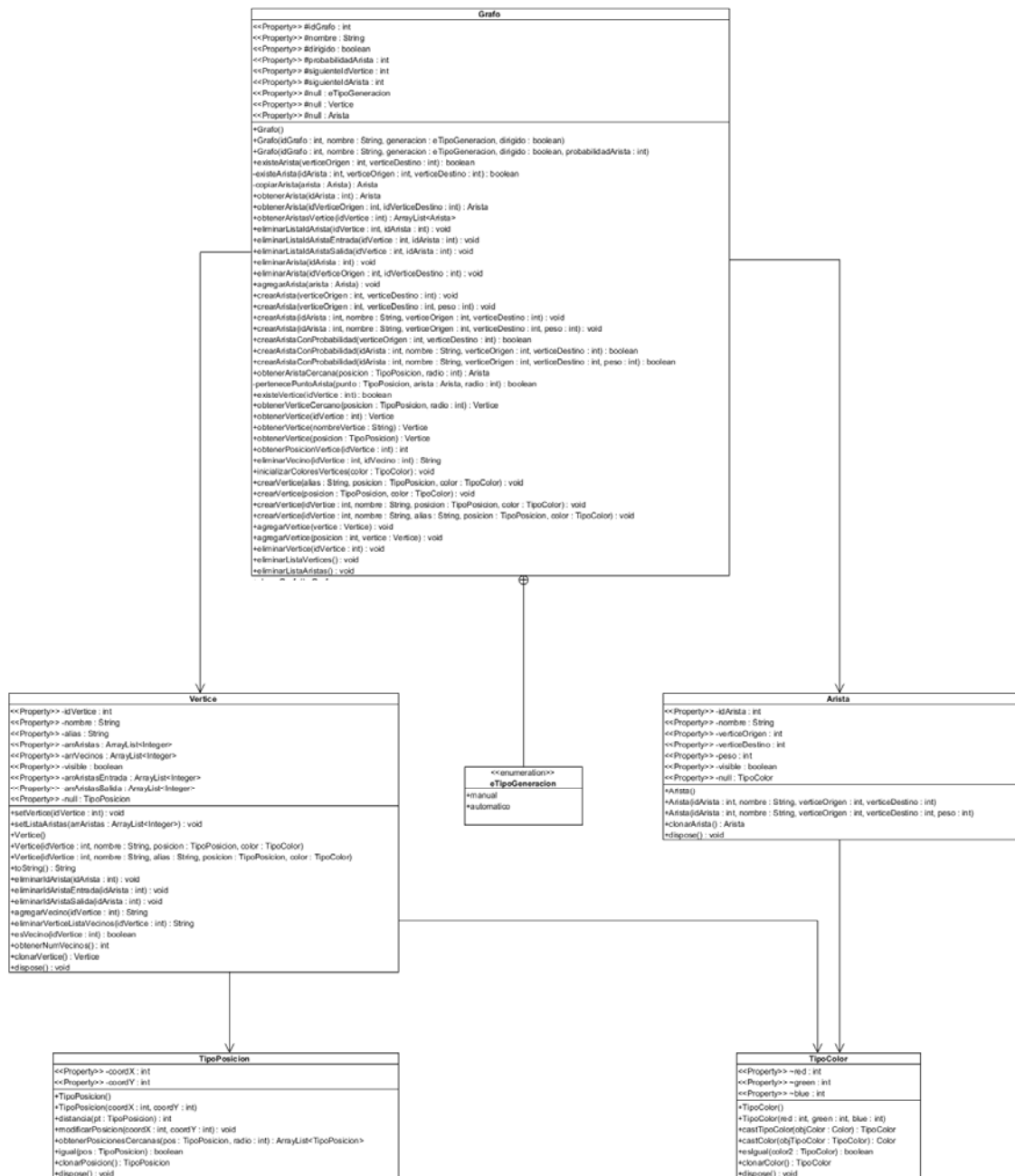


Figura 26: Diagrama de Clases que muestra la clase Grafo (y las clases con las que se relaciona)

3.3.1.3. Algoritmo

Debido a que la aplicación implementa distintos algoritmos, se ha tomado la decisión de separar los algoritmos en bloques para una mejor visión de las clases.

Algoritmos DFS y BFS

- **DFS**: Clase que instancia un objeto que representa al algoritmo DFS. Contiene información relativa tal como vértice de inicio, la estadística de ejecución, el estado actual (representado por una instancia de la clase DFS_Estado) y una lista de estados anteriores, entre otros.
- **DFS_Estado**: Clase que instancia un objeto que representa información de la ejecución de un paso del algoritmo. Entre esta información se encuentra una lista de vértices del árbol que se va formando, los vértices candidatos, las aristas utilizadas, etc. Se apoya en tres clases de tipos enumerados:
 - eTipoVértice
 - eTipoEstado
 - eTipoArista
- **BFS**: Clase que instancia un objeto que representa al algoritmo BFS. Contiene información relativa tal como vértice de inicio, vértice final, la estadística de ejecución, el estado actual (representado por una instancia de la clase BFS_Estado) y una lista de estados anteriores, entre otros.
- **BFS_Estado**: Clase que instancia un objeto que representa información de la ejecución de un paso del algoritmo. Entre esta información se encuentra una lista de vértices candidatos y visitados, las aristas utilizadas y las no del árbol, la distancia de cada vértice al origen, etc. Se apoya en tres clases de tipos enumerados:
 - eEstadoVértice
 - eEstadoArista
 - eTipoArista
- **Estadística**: Clase que instancia un objeto que representa la información estadística de la ejecución de cada algoritmo.

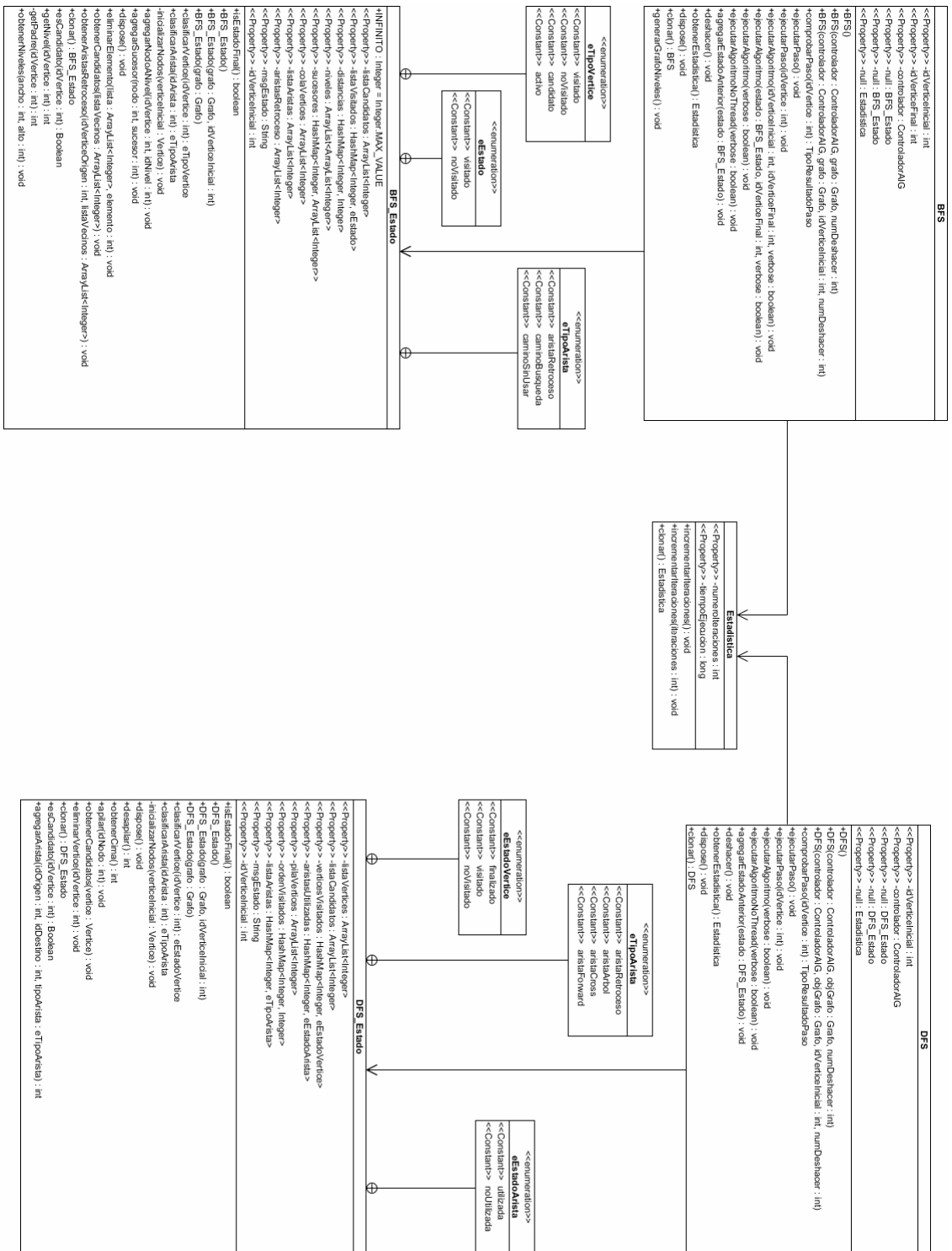


Figura 27: Diagrama de Clases que muestra los Algoritmos DFS y BFS

Algoritmos de Bloques y de Dijkstra

- **Bloques**: Clase que instancia un objeto que representa al algoritmo de Bloques. Contiene información relativa tal como vértice de inicio, la estadística de ejecución, el estado actual (representado por una instancia de la clase Bloques_Estado) y una lista de estados anteriores, entre otros.
- **Bloques _Estado**: Clase que instancia un objeto que representa información de la ejecución de un paso del algoritmo. Entre esta información se encuentra una lista de vértices de corte y una serie de listas de aristas (visitadas, de retroceso y puentes), los bloques generados, etc. Se apoya en dos clases de tipos enumerados:
 - eTipoVértice
 - eEstado
- **Dijkstra**: Clase que instancia un objeto que representa al algoritmo Dijkstra. Las propiedades más importantes son el vértice de inicio, la estadística de ejecución, el estado actual (representado por una instancia de la clase Dijkstra _Estado) y la lista de estados anteriores.
- **Dijkstra _Estado**: Clase que instancia un objeto que representa información de la ejecución de un paso del algoritmo. Entre esta información se encuentra una lista de vértices candidatos, visitados y predecesores, la distancia de cada vértice al origen y el camino mínimo. Se apoya en cuatro clases de tipos enumerados:
 - eEstado
 - eTipoVertice
 - eAccion
 - eTipoArista

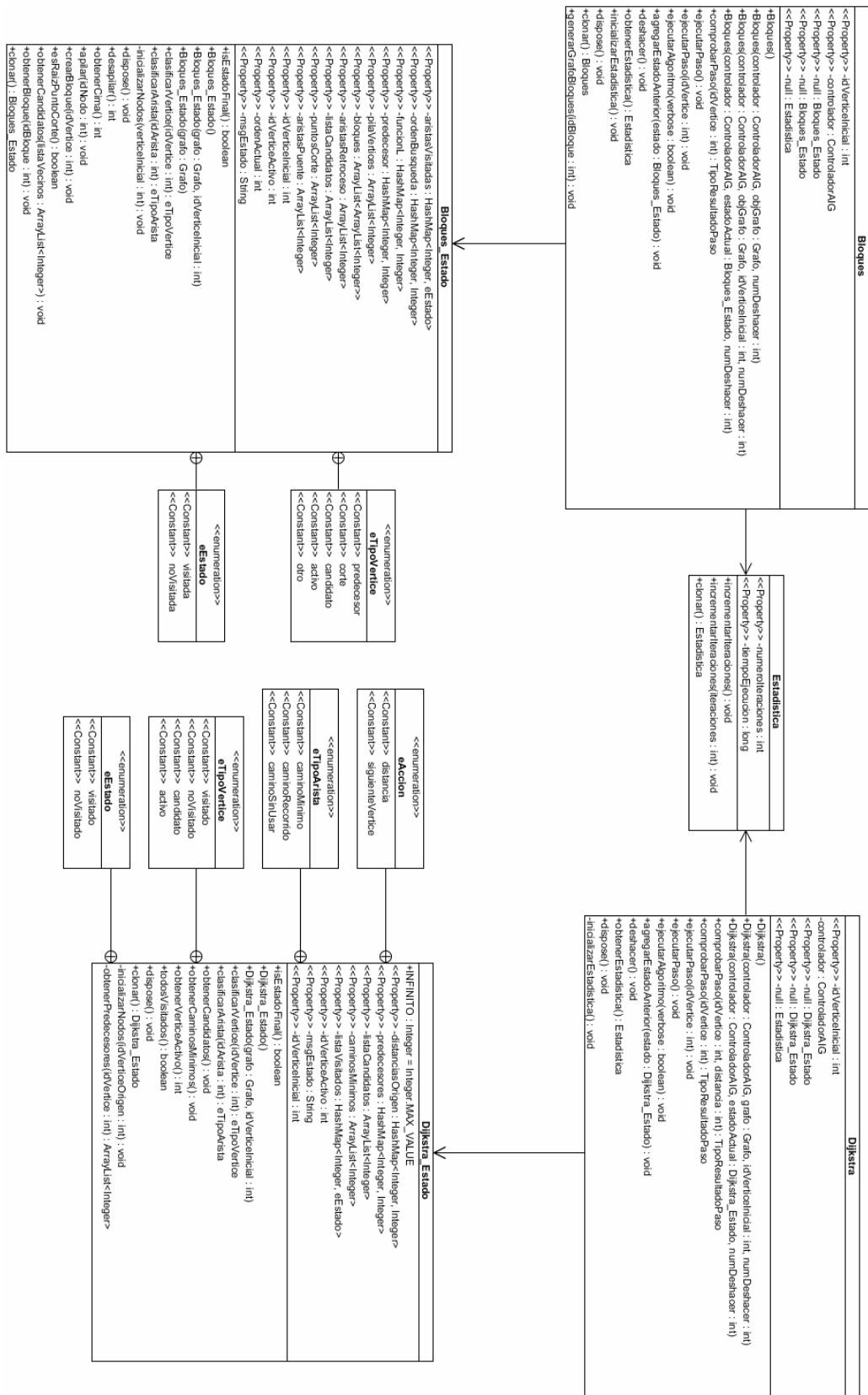


Figura 28: Diagrama de Clases que muestra los Algoritmos de Bloques y Dijkstra

Algoritmos de Coloración (Brelaz y Secuencial)

- **Col_Secuencial:** Clase que instancia un objeto que representa a los algoritmos de coloración secuencial. Se utiliza esta clase para implementar los algoritmos de coloración aleatoria, Welsh y Matula. Contiene información del estado actual y la estadística de ejecución.
- **Col_Secuencial_Estado:** Clase que instancia un objeto que representa información de la ejecución de un paso del algoritmo. Entre esta información se encuentra una lista de nodos (representa la lista en la que se irán coloreando los vértices del grafo, dependiendo del algoritmo que se esté ejecutando) y una matriz de grupos de colores.
- **Nodo_Secuencial:** Clase que instancia un objeto que representa un nodo del grafo en la ejecución del algoritmo.
- **Col_Brelaz:** Clase que instancia un objeto que representa al algoritmo de Brelaz. La información más relevante de esta clase viene representada por el estado actual y la estadística de ejecución.
- **Col_Brelaz_Estado:** Clase que instancia un objeto que representa información de la ejecución de un paso del algoritmo. Entre esta información se encuentra la lista de los nodos del grafo y una matriz de grupos de colores.
- **Nodo:** Clase que instancia un objeto que representa un nodo del grafo en la ejecución del algoritmo. Para cada nodo del grafo tendremos representado el grado del Vértice, el grado de Saturación y el orden en que ha sido coloreado.

Algoritmos de coloración de conjuntos independientes

- **MCI**: Clase que instancia un objeto que representa al algoritmo de independencia. Contiene información del estado actual, la lista de estados anteriores y la estadística de ejecución.
- **MCI_Estado**: Clase que instancia un objeto que representa información de la ejecución de un paso del algoritmo. Entre esta información se encuentra el grafo de la iteración, la lista de vértices que conforman el conjunto independiente actual, una lista de conjuntos independientes, una lista de vértices no disponibles y los vecinos del nodo actual.

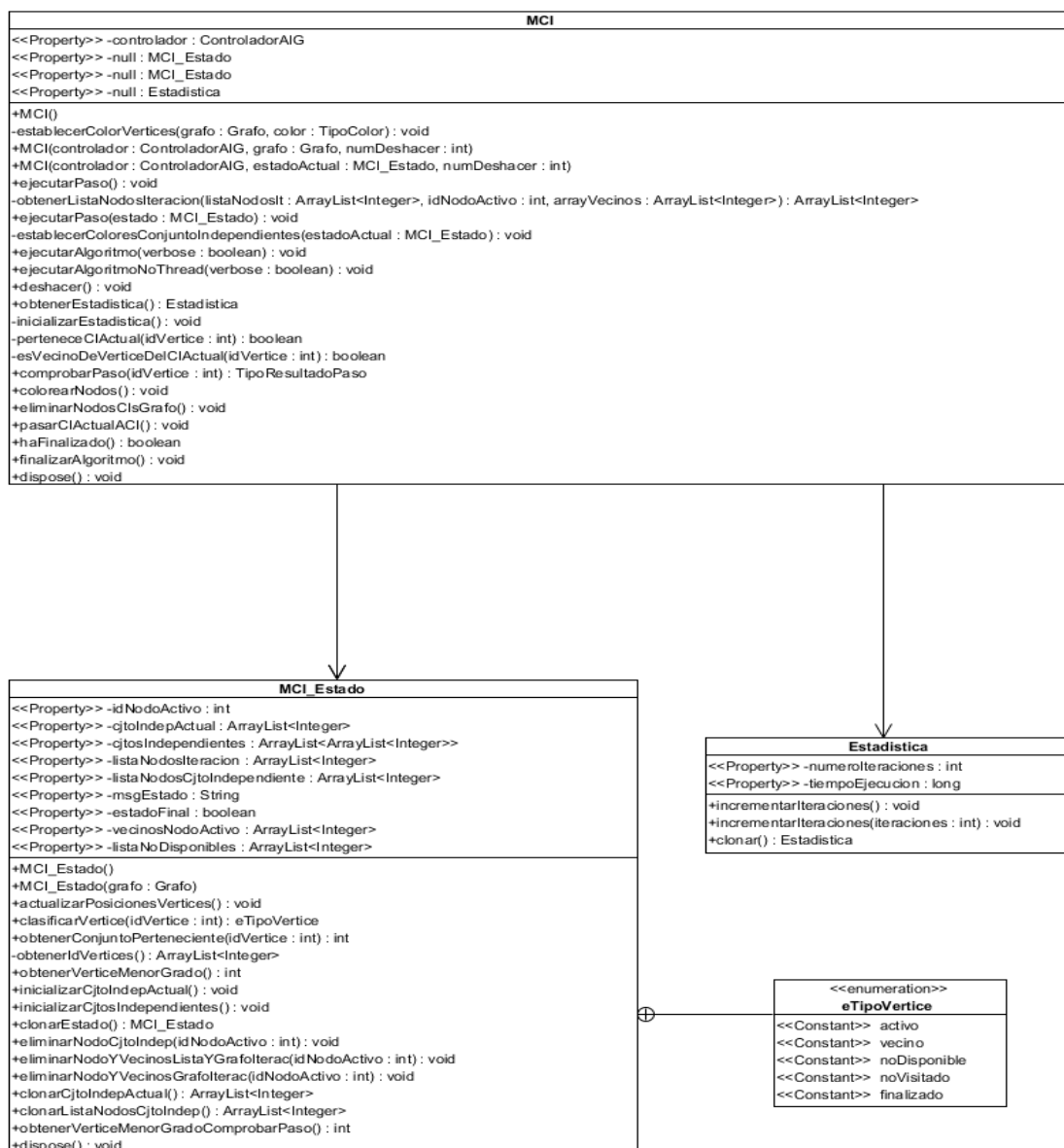


Figura 30: Diagrama de Clases que muestra el Algoritmos de Coloración de Conjuntos Independientes

Algoritmos de coloración de conjuntos independientes

En este diagrama se explica las clases que implementan los datos configurables de la aplicación:

- **Configuracion**: Clase que sirve para representar la configuración del sistema. Las propiedades de esta clase son instancias de las clases explicadas a continuación.
- **Configuracion_Grafo**: Contiene la información relativa a la configuración del grafo, entre la que se encuentra el color y grosor de los vértices y aristas.
- **Configuracion_Idioma**: Los textos mostrados por la aplicación pueden estar en español o en inglés. Esta clase sirve para determinar el idioma de la aplicación en cada momento.
- **ConfiguracionEntorno**: Se apoya en cuatro tipos enumerados para guardar información relativa al tipo de proyecto (coloración o búsqueda), al algoritmo en ejecución, a la visualización en el lienzo del nombre o alias de los vértices del grafo y al tipo de ejecución del algoritmo (automática o interactiva).

Existen además cinco clases para configurar las características de los elementos mostrados en el lienzo, asociadas a cada uno de los algoritmos de la aplicación. Son: Configuracion_DFS, Configuracion_BFS, Conguracion_Bloques, Configuracion_Dijkstra y Configuracion_MCI.

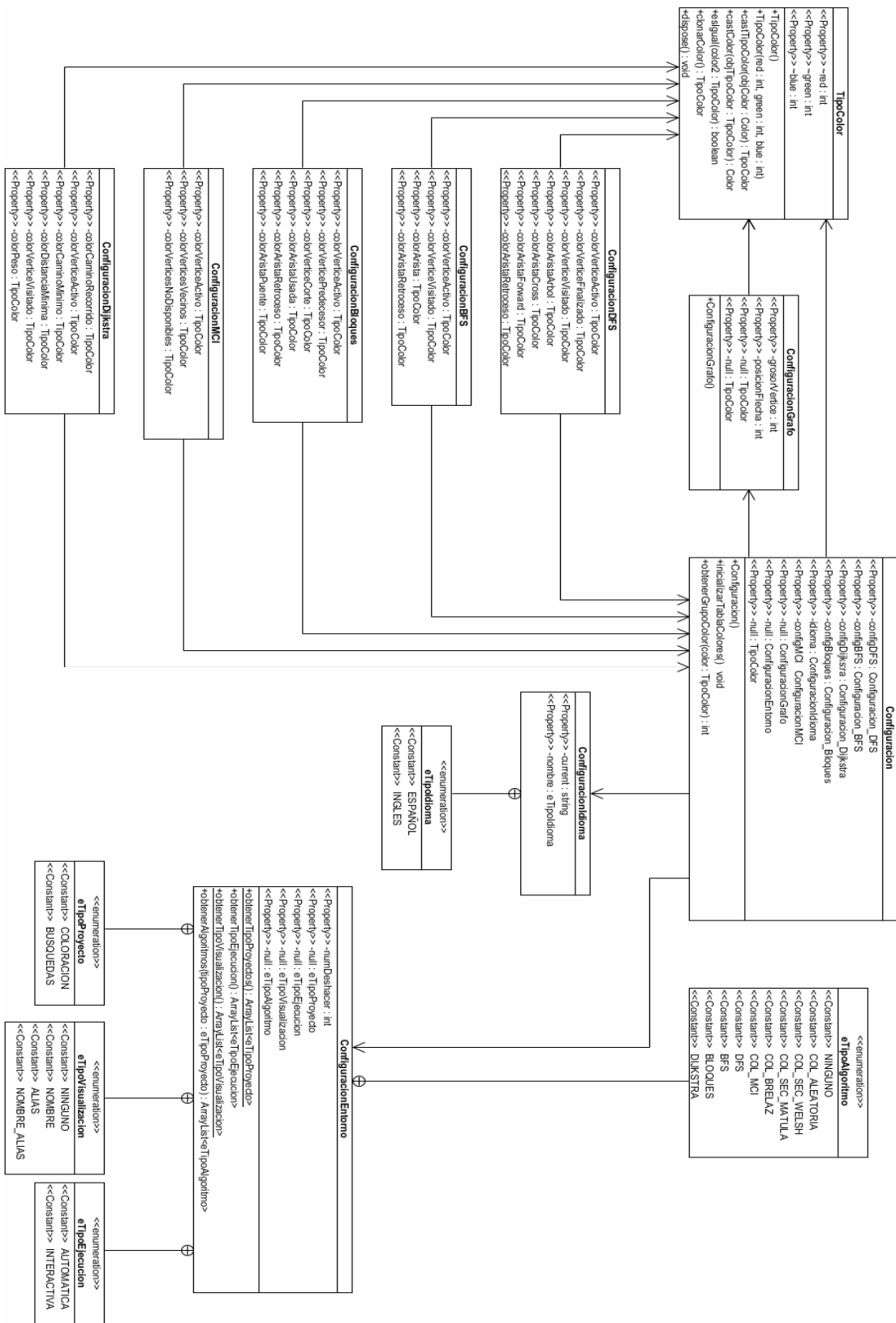


Figura 31: Diagrama de Clases que muestra la Configuración

4. MANUAL DE USUARIO

AIG (Aplicación Integral de Grafos) es una aplicación JAVA distribuida en un paquete JAR y por tanto ejecutable en un sistema con instalación de JVM. La distribución de los elementos en la aplicación es similar a la empleada en cualquier otra herramienta gráfica.

4.1. Partes de la aplicación

La ventana principal de la aplicación AIG se compone de los siguientes elementos:

- Barra de Menús
- Barra de Iconos (Botones)
- Información
- Posición
- Lienzo

4.1.1. Barra de Menús

A continuación describiremos los menús de la aplicación indicando la utilidad de cada una de las opciones. En esta sección no entraremos a explicar en profundidad cada una de las opciones.

Como puede apreciarse, al iniciarse la aplicación existen opciones del menú que están deshabilitadas. Estas se habilitan al abrir o crear un grafo, y en función del grafo activo en cada momento, se activarán unas opciones u otras.

4.1.1.1. Archivo

Como en la mayoría de aplicaciones de ventana, el primer menú que nos encontramos es “Archivo”. A continuación explicaremos brevemente cada una de las funcionalidades que nos encontraremos en este menú.

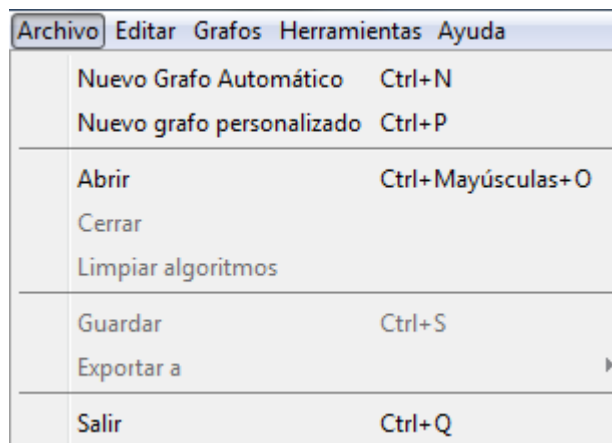


Figura 32: Menú Archivo

- **Nuevo Grafo Automático:** Nos permite crear un grafo de manera automática, tan solo tendremos que especificar el número de nodos y la probabilidad de que exista arista entre los nodos.
- **Nuevo Grafo Personalizado:** A diferencia con la opción “Nuevo Grafo Automático”, esta funcionalidad nos permite especificar manualmente las aristas existentes.
- **Abrir:** Nos permite cargar en la aplicación un grafo previamente salvado. Si en el momento en que el grafo fue guardado, estábamos inmersos en la ejecución de un algoritmo, se podrá continuar con la ejecución del algoritmo desde el punto en que fue almacenado.
- **Cerrar:** Elimina el grafo, dejando la aplicación en la misma situación que cuando se arranca.
- **Limpiar algoritmos:** Esta funcionalidad, elimina de pantalla cualquier referencia a la ejecución de un algoritmo, dejando solamente el estado inicial del grafo.
- **Guardar:** Nos permite la opción de almacenar el grafo con el que estamos trabajando. Dicho grafo se almacenará en formato xml. Si en el momento en que se decide almacenar el grafo, estamos ejecutando un algoritmo, también se almacenará el estado de la ejecución del algoritmo, para permitir posteriormente con la ejecución.
- **Exportar:** La aplicación nos permite exportar toda la información relativa al grafo en formato Excel.
- **Salir:** Nos permite cerrar la aplicación.

4.1.1.2. Editar

En el menú editar podremos editar el grafo en ejecución.

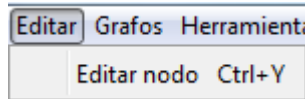


Figura 33: Menú Editar

Este menú contiene la siguiente opción:

- **Editar Nodo:** Esta funcionalidad permite cambiar información relativa a cada nodo. Cambio de alias y modificación (inserción/eliminación) de las aristas de cada vértice.

4.1.1.3. Grafos

En este menú encontraremos una serie de grafos predefinidos que pueden cargarse para trabajar con ellos. Algunos de ellos, como Tutte, son grafos fijos, mientras que otros, como Estrella, permiten que el usuario introduzca el número de vértices que se desea que tenga. Las características de cada uno se explicarán en la sección “Grafos Tipo”.

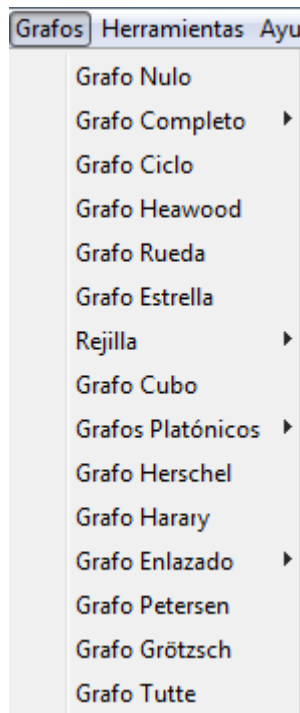


Figura 34: Menú Grafos

4.1.1.4. Herramientas

El siguiente menú ofrece una serie de herramientas que el usuario podrá seleccionar durante su trabajo con la aplicación.

Herramientas	Ayuda
Ejecución	Ctrl+Mayúsculas+R
Opciones	Ctrl+Mayúsculas+T
Imagen de Fondo	Ctrl+Mayúsculas+I
Limpiar Fondo	Ctrl+Mayúsculas+I
Estadísticas	Ctrl+Mayúsculas+S
Matriz de Adyacencia	Ctrl+Mayúsculas+A
Matriz de Ponderación	Ctrl+Mayúsculas+W

Figura 35: Menú Herramientas

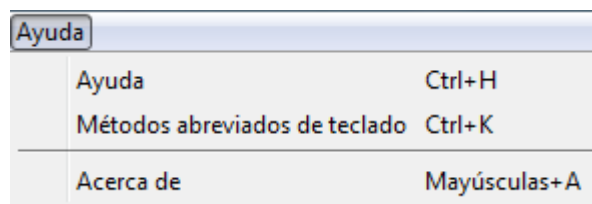
Este menú contiene las siguientes opciones:

- **Ejecución:** Permite al usuario seleccionar un algoritmo para iniciar su ejecución. El usuario podrá seleccionar si desea ejecutar el algoritmo de manera automática o interactiva. Esta opción permanecerá deshabilitada mientras no exista un grafo activo en la aplicación.
- **Opciones:** El usuario podrá configurar una serie de parámetros predefinidos en la aplicación, tales como, el color de los nodos, aristas, tamaño de la arista, etc. La aplicación permite que se pueda modificar el idioma.
- **Imagen de fondo:** Permite que el usuario pueda elegir que el lienzo tenga una imagen de fondo.
- **Limpiar fondo:** Elimina la imagen de fondo cargada, restableciendo el fondo blanco.
- **Estadísticas:** Esta funcionalidad permite que el usuario compare la ejecución de diversos algoritmos respecto al grafo activo en la aplicación. El resultado se generará en formato excel.
- **Matriz de Adyacencia:** Mostrará la matriz de adyacencia asociada al grafo activo en la aplicación.

- **Matriz de Ponderación:** Mostrará la matriz de ponderación asociada al grafo activo en la aplicación. Muestra el peso de las aristas existentes entre vértices.

4.1.1.5. Ayuda

El siguiente menú ofrece ver la documentación del menú de usuario, los métodos abreviados del teclado e información acerca de los desarrolladores del proyecto.



Ayuda	
Ayuda	Ctrl+H
Métodos abreviados de teclado	Ctrl+K
Acerca de	Mayúsculas+A

Figura 36: Menú Ayuda







4.1.2. Barra de Iconos (Botones)

La barra de iconos (botones) permite al usuario seleccionar las operaciones más frecuentes, además de poder seleccionar una serie de opciones tanto para modificar gráficamente el grafo, como configurar el proyecto sobre el que se está trabajando.







Figura 37. Barra de Iconos

A continuación explicaremos brevemente la funcionalidad de cada elemento de la barra de iconos:



-  **Nuevo Grafo Automático:** Permite crear un Nuevo Grafo Automático. Realiza la misma funcionalidad que si seleccionáramos **Archivo → Nuevo Grafo Automático**.
-  **Guardar:** Permite salvar un grafo activo a un archivo en formato xml. Realiza la misma funcionalidad que si seleccionáramos **Archivo → Guardar**.
-  **Abrir:** Permite cargar un grafo previamente guardado en la aplicación. Realiza la misma funcionalidad que si seleccionáramos **Archivo → Abrir**.
-  **Imagen de fondo:** Inserta una imagen de fondo en el lienzo. Realiza la misma funcionalidad que si seleccionáramos **Herramientas → Imagen de fondo**.
-  **Ejecutar algoritmo:** Permite la ejecución de algoritmos sobre el grafo activo en la aplicación. Realiza la misma funcionalidad que si seleccionáramos **Herramientas → Ejecución**. Este botón permanecerá deshabilitado mientras no exista un grafo activo en la aplicación.
-  **Opciones:** Permite al usuario configurar una serie de parámetros en la aplicación. Realiza la misma funcionalidad que si seleccionáramos **Herramientas → Opciones**.






A continuación tenemos una serie de cuatro botones, que permiten el manejo manual del grafo sobre el lienzo. Estos botones están relacionados entre sí, y no puede haber dos botones seleccionados a la vez, con lo que si tenemos uno seleccionado y pulsamos sobre otro, el primero se deselectionará. Inicialmente ninguno de los botones está seleccionado, con lo que solamente se podrán mover los nodos del grafo, para ello el usuario colocará el puntero del ratón sobre el vértice deseado, pulsará con el botón izquierdo del ratón sobre él, y sin soltar el botón izquierdo del ratón, podrá mover libremente el vértice seleccionado hasta la posición deseada. Pasaremos a explicar la funcionalidad de cada uno de estos cuatro botones.

-  **Insertar nodo:** Permite al usuario insertar un nodo, únicamente realizando un clic con el botón izquierdo del ratón sobre la posición deseada. Si el lugar deseado coincide con la posición donde está un nodo ya creado, no se insertará un nuevo vértice en dicha posición. Realiza una funcionalidad similar que si seleccionáramos **Editar → Insertar → Nodo**.
-  **Eliminar nodo:** Permite al usuario eliminar un nodo, únicamente realizando un clic con el botón izquierdo del ratón sobre el vértice que se desea borrar. Hecho esto, se eliminarán tanto el nodo seleccionado, como todas las aristas que lleguen o salgan de dicho vértice. Realiza una funcionalidad similar que si seleccionáramos **Editar → Suprimir → Nodo**.
-  **Insertar arista:** Permite al usuario insertar una arista entre dos nodos, para ello pulsará con el botón izquierdo del ratón sobre el vértice origen de la arista y sin soltar dicho botón, moverá el cursor hasta el vértice destino de la arista, donde soltará el ratón. El usuario podrá comprobar que una vez que ha pulsado sobre el vértice origen y está moviendo el cursor, aparece en pantalla la arista que esta insertando. Realiza una funcionalidad similar que si seleccionáramos **Editar → Insertar → Arista**.
-  **Eliminar arista:** Permite al usuario eliminar una arista existen entre dos nodos. Para ello, el usuario únicamente tendrá que realizar un clic con el botón izquierdo del ratón, sobre la arista que se desea eliminar. Realiza una funcionalidad similar que si seleccionáramos **Editar → Suprimir → Arista**.

Después de este conjunto de botones podemos ver que tenemos dos cuadros de selección, el primero de ellos, nos permite seleccionar el tipo de proyecto con el que vamos a trabajar: Coloración o Búsquedas, la selección de uno u otro tipo de proyecto afecta a la hora de los algoritmos que puede ejecutar el usuario, ya que si esta seleccionado el proyecto Coloración, aparecerán unos algoritmos de ejecución, distintos a si tenemos seleccionado el proyecto Búsquedas. De manera similar ocurrirá cuando seleccionemos la opción Estadísticas. El segundo cuadro de selección permite al usuario la opción de que en el lienzo se muestren al lado de los vértices, su nombre, su alias, su nombre y alias o que no aparezca ninguna de estas opciones.

Para finalizar tenemos los siguientes botones:

-  **Mostrar ponderación (Distancias):** Permite al usuario poder visualizar el peso de cada arista sobre el lienzo. Si pulsamos dicho botón aparecerán sobre las aristas el peso de cada una de ellas.
-  **Mostrar Tooltip:** Permite al usuario visualizar sobre el lienzo, información de los vértices y aristas, según se vaya situando el cursor el ratón sobre ellos.

-  **Matriz de adyacencia:** Mostrará la matriz de adyacencia asociada al grafo activo en la aplicación. Realiza una función similar que si seleccionáramos **Herramientas → Matriz de Adyacencia**.
-  **Matriz de ponderación:** Mostrará la matriz de ponderación asociada al grafo activo en la aplicación. Muestra el peso de las aristas existentes entre vértices. Realiza una función similar que si seleccionáramos **Herramientas → Matriz de Ponderación**.
-  **Mostrar niveles:** Este icono sólo estará activo al finalizar la ejecución del algoritmo BFS. Mostrará el árbol de niveles generado por el algoritmo BFS sobre el grafo original.
-   **Mostrar bloques:** Estos iconos sólo estarán activos al finalizar la ejecución del algoritmo de bloques (conectividad). Una vez que se pulse el botón de la izquierda (mostrar bloques) se nos mostrará por pantalla el primer bloque existente. Para ver el resto de bloques se deberán pulsar las flechas de dirección situadas a la derecha de este botón.

4.1.3. Información

La zona de información esta situada a en la parte izquierda de la pantalla y se compone de tres partes diferenciadas.

- **Información del grafo (Árbol).** Es la parte superior de la zona de información, mostrará información de los nodos y aristas del grafo activo.

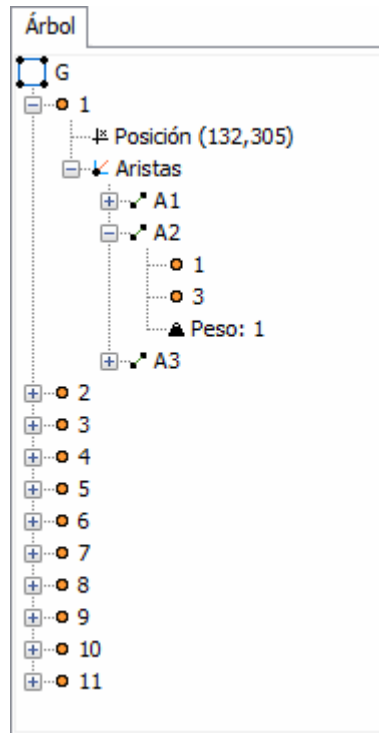


Figura 38. Árbol de un grafo

Podemos ver que nos aparecen listados de arriba hacia abajo los vértices, y si desplegamos un vértice nos aparecerá la posición de dicho vértice y las aristas que llegan y salen de él. Pulsando en una de las aristas podremos ver los dos nodos que unen y el peso de dicha arista.

Durante la ejecución de los algoritmos se mostrará en esta zona una pestaña donde podremos obtener información del significado de los colores usados en vértices y aristas durante el transcurso de la ejecución del algoritmo. Llamaremos a esta pestaña “Leyenda” y tendrá un aspecto similar a este:

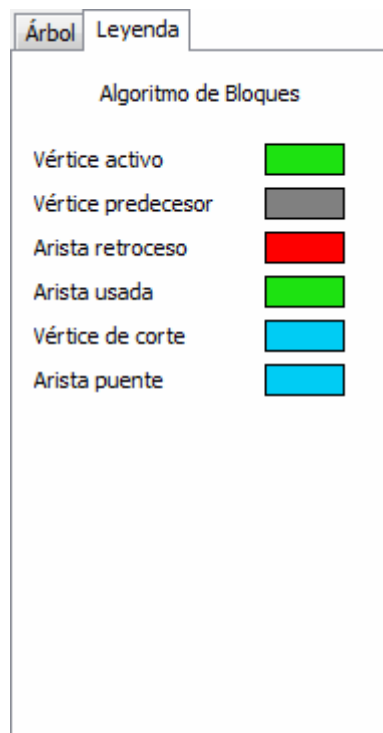


Figura 39. Leyenda del algoritmo de Búsqueda en Profundidad

Durante la ejecución del algoritmo de Coloración de Brelaz, aparecerá una pestaña junto a la pestaña del Árbol, donde se nos mostrará una tabla con información relativa al algoritmo.

- **Botones de Ejecución.** Aparecen en la parte central de la zona de información y son cuatro botones.

La ejecución automática de un algoritmo puede realizarse de dos maneras diferentes: paso a paso o de manera completa, pudiendo mezclar los dos tipos en la ejecución de un algoritmo

La funcionalidad de los botones es la siguiente:

- ⏮ **Paso hacia atrás.** Si durante la ejecución de un algoritmo paso a paso, queremos volver al paso anterior, se pulsará este botón.
- ▶ **Iniciar (Play).** Si el usuario desea ejecutar un algoritmo automático hasta que este finalice pulsará el botón de Play.
- ⏸ **Parar (Pause).** Una vez que el usuario ha pulsado el botón Play, puede parar la ejecución del algoritmo pulsando el botón de Pause.

Paso hacia delante. La manera de avanzar en la ejecución de un algoritmo paso a paso es pulsando este botón.

Estos botones inicialmente están deshabilitados y solamente se activarán cuando se seleccione la ejecución automática de un algoritmo.

- **Información del algoritmo en ejecución.** Es la parte inferior de la zona de información, aquí se nos mostrará información sobre la ejecución del algoritmo seleccionado.

Una vez que el usuario haya empezado la ejecución de un algoritmo irá apareciendo como se está realizando el algoritmo, la información que aparezca dependerá del algoritmo seleccionado.

La figura 40 nos presenta un ejemplo de la información que se obtiene al ejecutar un algoritmo.

```

Conectividad (algoritmo de bloques)

Pila = {1} --> Cima = 1
Candidatos = {2, 3, 4}

Pila = {2, 1} --> Cima = 2
Candidatos = {3, 5}

Pila = {3, 2, 1} --> Cima = 3
Candidatos = {1, 4, 5}

Retroceso = {(1,3)}
Candidatos = {4, 5}

Pila = {4, 3, 2, 1} --> Cima = 4
Candidatos = {1, 5, 6}

Retroceso = {(1,3),(1,4)}
Candidatos = {5, 6}

Pila = {5, 4, 3, 2, 1} --> Cima = 5
Candidatos = {3, 2, 6, 7}

Retroceso = {(1,3),(1,4),(3,5)}
Candidatos = {2, 6, 7}

Retroceso = {(1,3),(1,4),(3,5),(5,2)}
Candidatos = {6, 7}

Pila = {6, 5, 4, 3, 2, 1} --> Cima = 6
Candidatos = {4, 7, 10}
  
```

Figura 40. Información de la ejecución de un algoritmo

4.1.4. Posición

En la parte inferior derecha de la aplicación podremos ver la posición del cursor del ratón sobre el lienzo.

Posición 389 - 432

Figura 41. Cuadro de posición del cursor

4.1.5. Lienzo

Es la parte principal de la aplicación, situada en la zona central derecha de la ventana principal.

Sobre el lienzo se mostrará gráficamente el grafo con el que se desea trabajar y sobre se mostrarán las ejecuciones de los algoritmos.

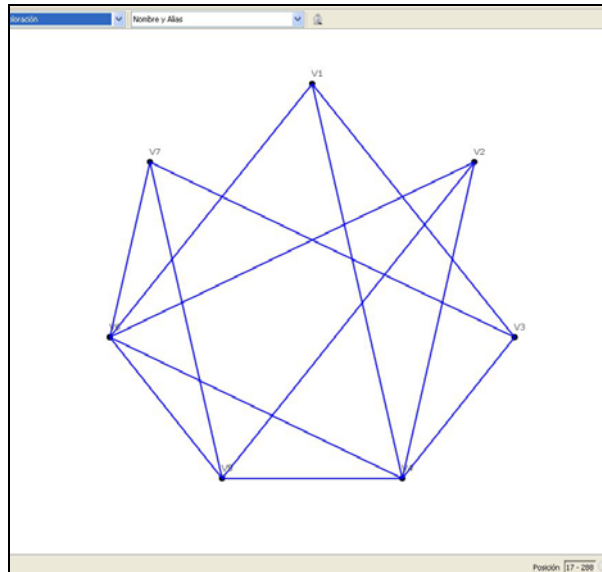


Figura 42. Lienzo de la aplicación mostrando un grafo

Todas y cada una de las acciones que se realicen tanto a través de los menús como de los botones tendrán un efecto sobre el lienzo o sobre el grafo del lienzo.

Adicionalmente se han habilitado un menú contextual cuando se pulsa con el botón derecho sobre el lienzo. Este menú variará dependiendo de si estamos en la ejecución de un algoritmo o editando un grafo.

Si estamos editando un grafo, disponemos de cuatro funcionalidades:

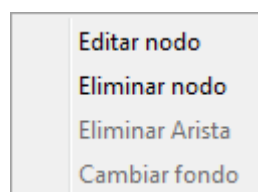


Figura 43. Menú contextual de edición de grafo

- **Editar Nodo:** Esta opción estará únicamente disponible cuando se pulse sobre un vértice. Permite al usuario editar un nodo. Realiza una funcionalidad similar que si seleccionamos **Editar** → **Editar Nodo**, con la

diferencia a que ahora el nodo seleccionado aparece cargado en la ventana de edición del nodo.

- **Eliminar Nodo:** Esta opción estará únicamente disponible cuando se pulse sobre un vértice. Permite al usuario eliminar un nodo, Se eliminarán tanto el nodo seleccionado, como todas las aristas que lleguen o salgan de dicho vértice.
- **Eliminar Arista:** Esta opción estará únicamente disponible cuando se pulse sobre una arista. Permite al usuario eliminar una arista existen entre dos nodos.
- **Cambiar Fondo:** Inserta una imagen de fondo en el lienzo. Realiza la misma funcionalidad que si seleccionáramos **Herramientas → Imagen de fondo**.

Si estamos ejecutando un algoritmo, se nos mostrará un menú similar al de la Figura 44:

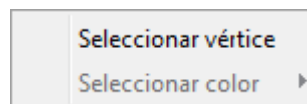



Figura 44. Menú contextual de ejecución de algoritmo

Dependiendo del algoritmo en ejecución variarán las opciones disponibles en este menú. Se explicarán ampliamente en la sección “Ejecución” en la parte correspondiente de cada algoritmo.

El usuario dispone de la posibilidad de visualizar información relativa a los vértices y a las aristas sobre el lienzo. Para ello deberá tener seleccionado el botón “Mostrar Tooltip” . Una vez realizado esto, cuando el usuario pase el cursor del ratón por encima de un vértice o de una arista, se mostrará una ventana similar a la que nos muestra la Figura 45.

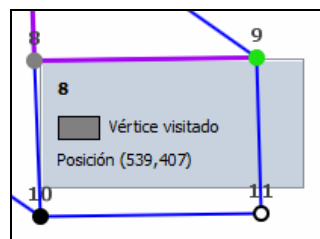


Figura 45. Ventana de información de vértice sobre el lienzo

La información mostrada variará dependiendo de si estamos o no en la ejecución de un algoritmo, y si se está en ejecución, del algoritmo que se esté ejecutando en ese determinado momento.

4.2. Operaciones Genéricas

4.2.1. Crear Nuevo Grafo Automático

En el caso de que se seleccione “Nuevo Grafo Automático” en el menú “Archivo” o que se pulse el icono “Nuevo Grafo Automático”. La Figura 46 nos presenta la ventana para crear un nuevo grafo de manera automática.

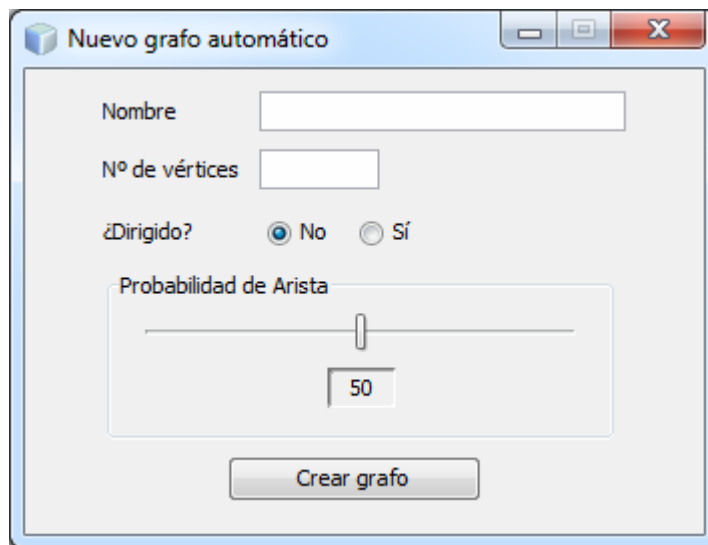


Figura 46. Ventana de creación de un Nuevo Grafo Automático

En ella deberemos introducir el nombre del grafo (de manera opcional), el número de vértices que compondrán el grafo, si es dirigido o no y la probabilidad de que exista una arista entre dos nodos.

El campo Nº de vértices solamente acepta caracteres numéricos.

La probabilidad de Arista varía entre 0 y 100. Si el usuario pone como probabilidad 0, solamente aparecerán los nodos (sin ninguna arista) y si el usuario decide que la probabilidad sea 100, resultará un grafo completo, existiendo una arista desde un nodo hacia cada uno de los nodos existentes.

Una vez que el usuario pulsa el botón “Crear Grafo”, la ventana de la aplicación tendrá un aspecto similar al que nos muestra la Figura 47:

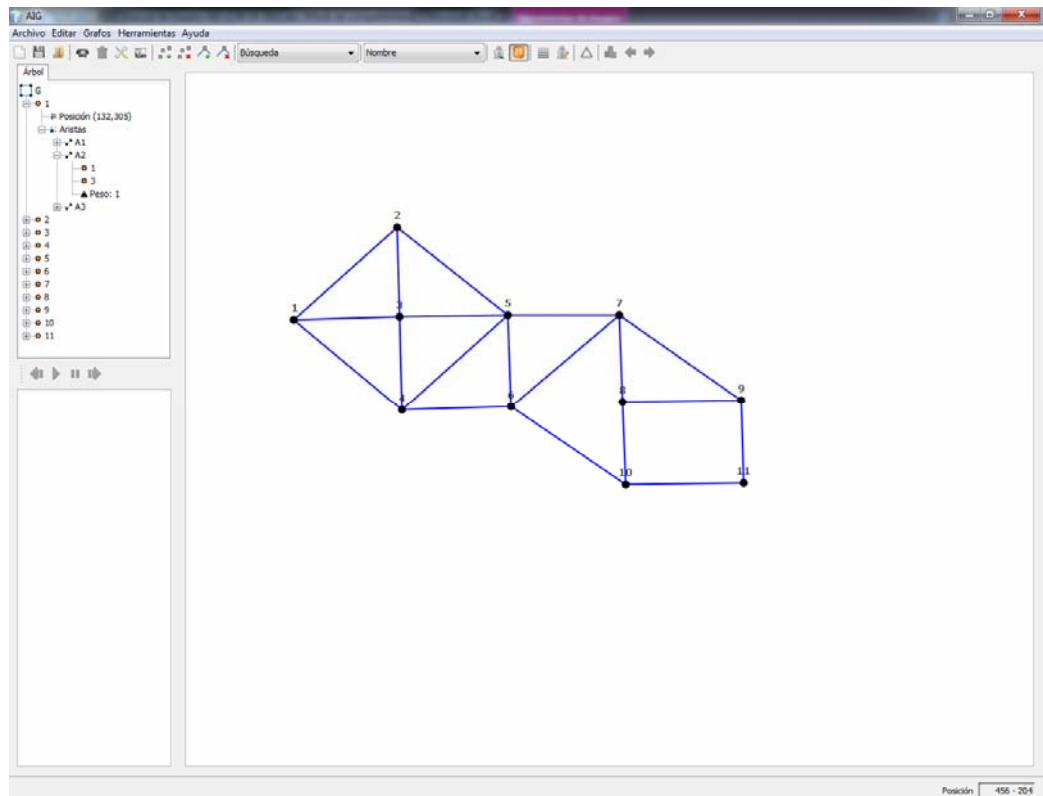


Figura 47. Detalle de la aplicación una vez creado un grafo

Podemos comprobar que en el lienzo aparece la representación gráfica del grafo y que en la pestaña “Árbol” de la zona de información aparecerá el grafo en forma de árbol, donde podremos ver sus nodos, posición de estos, y aristas de que se compone.

Si en el momento en el que usuario decide crear un nuevo grafo ya existe un grafo en la aplicación se nos mostrará una ventana preguntándonos si deseamos salvar el grafo:

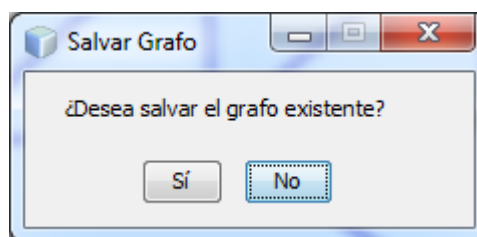


Figura 48. Ventana de confirmación para salvar el grafo

Si el usuario pulsa en el botón “Sí”, se nos mostrará la ventana para salvar el grafo (ver el punto 4.2.3 Guardar Grafo), si en su defecto el usuario pulsa el botón “No” se nos mostrará la ventana para la creación del nuevo grafo automático.

4.2.2. Crear Nuevo Grafo Personalizado

Existe la opción de que el usuario personalice el grafo en mayor profundidad, para ello deberá pulsar en la pestaña “Archivo” y a continuación seleccionar la opción “Nuevo Grafo Personalizado”. Una vez realizado esto, nos aparecerá la siguiente ventana que se nos muestra en la Figura 49:

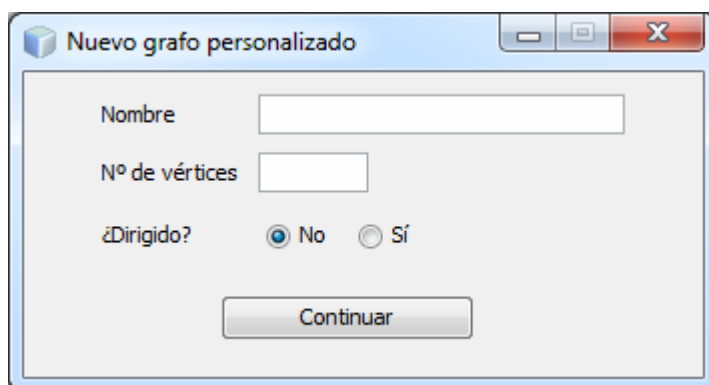


Figura 49. Ventana de creación de un Nuevo Grafo Personalizado

Podemos comprobar que la ventana es similar a la que aparece al crear un grafo de manera automática, con la salvedad que ha desaparecido la probabilidad de que exista una arista entre los nodos.

Si hemos seleccionado que el grafo no será dirigido nos aparecerá la siguiente ventana:

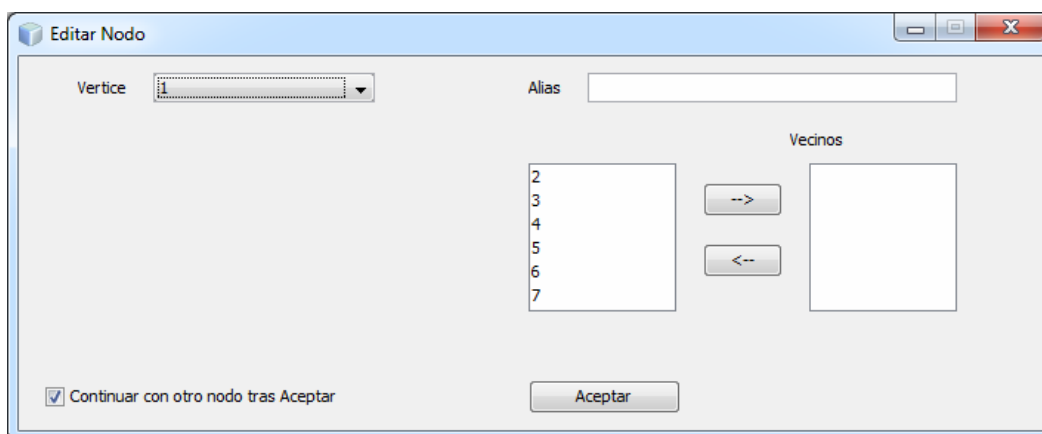


Figura 50. Ventana de edición de Nodos en la creación de un grafo (no dirigido) personalizado

En ella podremos ir seleccionando los vértices uno a uno, y podremos ponerle un alias al nodo e indicar cuales son los nodos vecinos, para ello seleccionaremos del cuadro de la izquierda los nodos vecinos para el vértice en edición y pulsaremos el botón con la flecha hacia la derecha (\rightarrow). Si por alguna razón el usuario decide

quitar algún nodo vecino, tendrá que seleccionar del cuadro de vecinos dicho nodo y pulsar el botón con la flecha hacia la izquierda (\leftarrow). Se puede ver que cuando se pulsa a uno de estos dos botones, los nodos seleccionados pasan de un cuadro a otro.

Cuando el usuario pulse el botón “Aceptar” podrá comprobar que en el lienzo se han creado las aristas entre los nodos seleccionados como vecinos y el nodo editado.

El usuario podrá ir cambiando de nodo e ir editando cada uno de ellos.

En la parte inferior izquierdo hay un check-box, donde se nos indica que tras Aceptar se continuará con la edición de otro nodo. Si el usuario deselecciona esta opción, cuando se pulse el botón Aceptar, se cerrará la ventana de edición del nodo.

Si por el contrario el usuario al crear el nodo ha decidido que sea dirigido aparecerá la siguiente ventana:

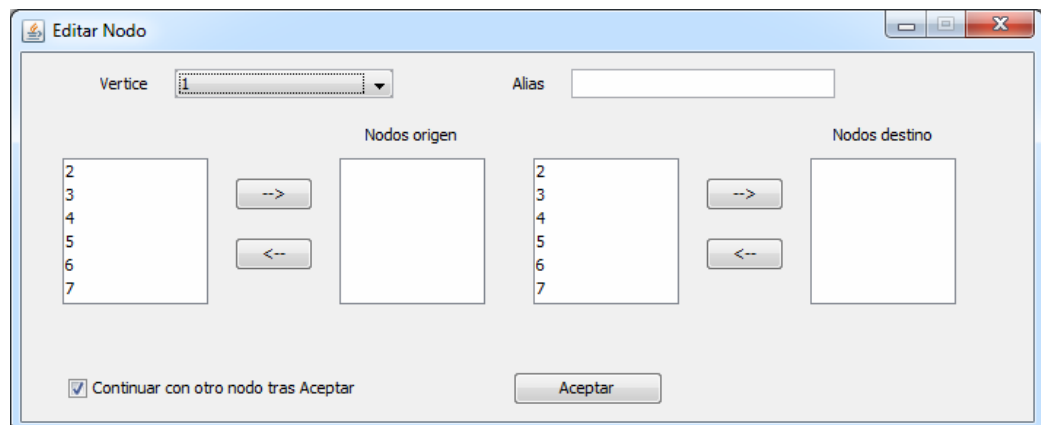


Figura 51. Ventana de edición de Nodos en la creación de un grafo (dirigido) personalizado

Esta pantalla es similar a la que nos aparece cuando el grafo no es dirigido, con la salvedad, que como ahora el grafo es dirigido, podemos:

- Elegir los vértices que son origen de un arista que llega hasta el vértice seleccionado, que se corresponde con los dos cuadros más a la izquierda de la pantalla (los nodos seleccionados que pasen al cuadro con el texto “Nodo origen”, tendrán una arista cuyo origen será dichos nodos y cuyo destino será el vértice que se está editando).
- Elegir los vértices que son destino de un arista cuyo origen será el vértice seleccionado, que se corresponde con los dos cuadros más a la derecha de la pantalla (los nodos seleccionados que pasen al cuadro con el texto “Nodo destino”, tendrán una arista cuyo destino será dichos nodos y cuyo origen será el vértice que se está editando).

Según vayamos dando al botón Aceptar, podremos ir observando los cambios en el lienzo.

Si en el momento en el que usuario decide crear un nuevo grafo ya existe un grafo en la aplicación se nos mostrará una ventana preguntándonos si deseamos salvar el grafo.

Si el usuario pulsa en el botón “Sí”, se nos mostrará la ventana para salvar el grafo (ver el punto 4.2.3 Guardar Grafo”), si en su defecto el usuario pulsa el botón “No” se nos mostrará la ventana para la creación del nuevo grafo personalizado.

4.2.3. Guardar Grafo

Una vez que tenemos un grafo activo en la aplicación, se ofrece al usuario la posibilidad de guardar dicho grafo para usarlo en el futuro, solamente tendrá que pulsar en el menú Archivo y seleccionar la opción “Guardar” o pulsar sobre el botón de guardar en la barra de Iconos. El usuario tendrá que introducir el nombre deseado para el grafo. El grafo se salvará en formato xml.

4.2.4. Abrir Grafo

En un determinado momento, el usuario podrá decidir cargar en la aplicación un grafo previamente salvado, para ello deberá pulsar en el menú Archivo y seleccionar la opción “Abrir” o pulsar sobre el botón de abrir en la barra de Iconos.

Una vez realizado una de estas opciones nos aparecerá la siguiente ventana:

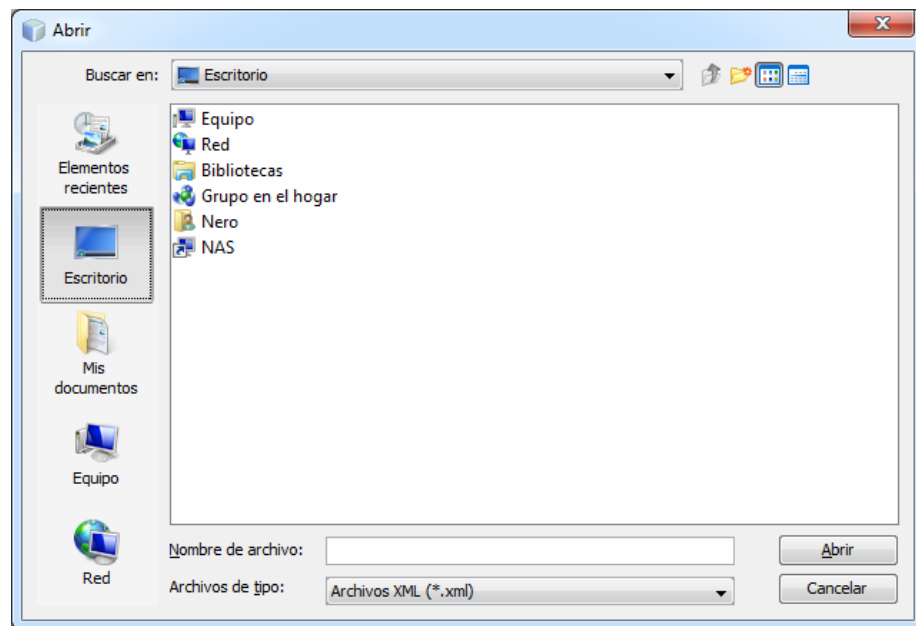


Figura 52. Ventana de selección del archivo (Grafo) a cargar en la aplicación

Fíjese que es una ventana similar a la que nos aparece en la opción Guardar. El usuario deberá ir navegando por su árbol de directorios hasta llegar a la carpeta donde se aloja el grafo que se desea cargar en la aplicación, le seleccionará y pulsará sobre el botón Abrir.

Una vez realizado esto el grafo se cargará en la aplicación y se podrá ver su representación gráfica sobre el lienzo, y en el árbol de la zona de información, podremos ver sus características.

Si seleccionamos un archivo que no contiene guardado un grafo, se producirá un error durante el proceso de carga del archivo y se mostrará por pantalla un mensaje.

4.2.5. Cerrar

El usuario dispone de la posibilidad de cerrar el grafo con el que está trabajando dejando la aplicación de la misma manera que cuando se inicializa.

Para ello deberá pulsar en el menú Archivo y seleccionar la opción “Cerrar”, una vez hecho esto la aplicación nos indicará si deseamos salvar el grafo en ejecución. Después de que el usuario decida si salva o no el grafo activo, la aplicación volverá a su situación inicial, sin ningún grafo activo en la aplicación, mostrando tanto el lienzo como el árbol de la zona de información vacíos.

4.2.6. Limpiar algoritmos

La funcionalidad principal de la aplicación es la de ejecutar una serie de algoritmos sobre un grafo que esté cargado en la aplicación. Durante la ejecución de estos algoritmos, sobre el grafo mostrado en el lienzo se irán haciendo diversas modificaciones, se colorearán tanto nodos como aristas dependiendo del algoritmo seleccionado.

Una vez terminada la ejecución de un algoritmo, o durante la ejecución de uno, el usuario puede decidir que quiere ejecutar otro algoritmo, o simplemente volver al estado inicial del grafo (sin las modificaciones comentadas anteriormente).

Para ello el usuario pulsará en el menú Archivo y seleccionará la opción “Limpiar”, una vez hecho esto, el grafo volverá a la situación inicial antes de la ejecución del algoritmo. Además se limpiará la ventana de información del algoritmo de todo texto mostrado durante la ejecución del algoritmo.

4.2.7. Exportar Grafo

La aplicación ofrece la posibilidad de generar documentación en formato xls del grafo activo. Para ello el usuario pulsará en Archivo y dentro de la opción “Exportar a” elegirá el formato Excel.

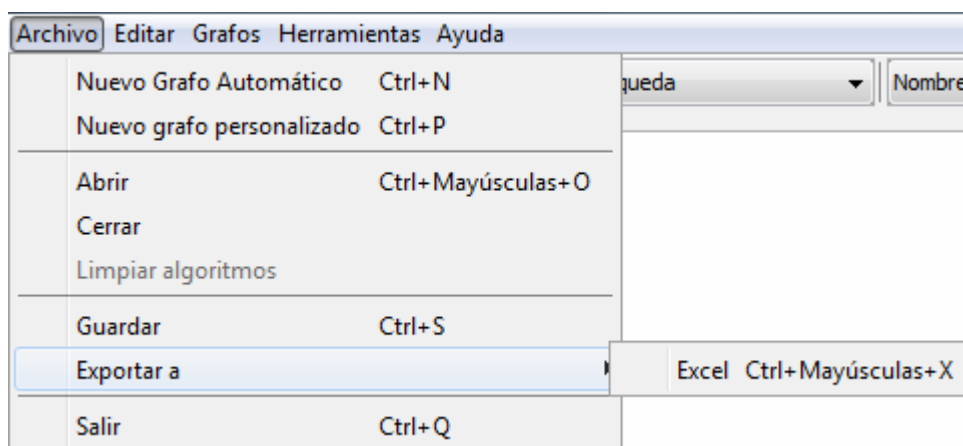


Figura 53. Menú Archivo → Exportar a

Una vez seleccionada una de estas dos opciones se nos abrirá una ventana similar a la de la opción Guardar, donde tendremos que elegir la ubicación y el nombre del archivo que se va a generar (en formato xls).

El archivo xls generado se compone de cuatro pestañas:

- Información: Contiene información general del grafo: nombre, si es dirigido, número de vértices y de aristas. Además incluye una representación gráfica del mismo.
- Vértices: Nos muestra la lista de vértices del grafo. Para cada vértice podremos ver su nombre, posición y las aristas que llegan/salen del nodo y el vértice de origen/destino de dicha arista.
- Aristas: Nos muestra la lista de aristas del grafo. Pudiendo ver para cada arista, su nombre, peso y los vértices de origen y destino.
- Matriz de adyacencia: Representa la matriz de adyacencia del grafo.

4.2.8. Salir

Para salir de la aplicación el usuario deberá pulsar en el menú Archivo y seleccionar la opción Salir.

4.2.9. Editar nodo

Se ofrece al usuario la posibilidad de editar un nodo. Para ello deberá pulsar en el menú Editar y seleccionar Editar Nodo, o pulsar con el botón derecho del ratón sobre el nodo que se desea editar y en el menú que se abrirá seleccionar Editar Nodo.

Una vez que el usuario ha realizado esta operación se nos mostrará la ventana de edición de nodo. Esta ventana ya se ha visto en la sección “Crear Nuevo Grafo Personalizado”. La ventana de edición de nodo será diferente si el grafo es o no dirigido.

- Grafo No Dirigido: Se nos mostrará la siguiente pantalla:

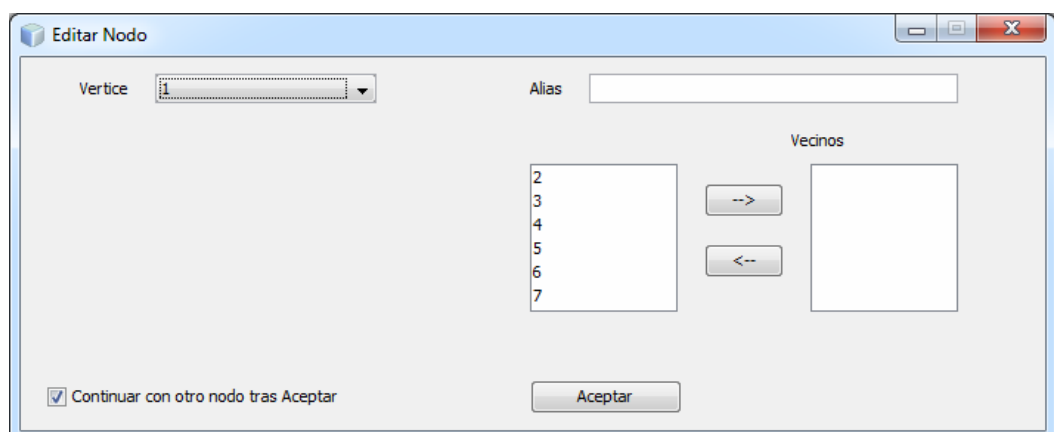


Figura 54. Ventana de edición de un grafo no dirigido

En el cuadro de selección Vértice podremos seleccionar el vértice que deseamos editar. Una vez seleccionado se cargarán los datos referidos a dicho vértice: su alias (si tuviese). Los vértices vecinos al nodo en edición (vértices unidos a través de una arista) aparecerán en el cuadro de la derecha bajo la etiqueta Vecinos. El resto de nodos del grafo, que no son vecinos del nodo en edición aparecerán en el cuadro de la izquierda. Se dispone de la posibilidad de añadir un vecino seleccionando un vértice del cuadro de la izquierda y pulsando el botón \rightarrow . El añadir un vecino implica la creación de una arista entre estos dos vértices.

Asimismo disponemos de la posibilidad de eliminar a un vecino. Se seleccionará el vecino del cuadro de la derecha y se pulsará al botón \leftarrow . El eliminar un vecino implica la desaparición de la arista que unía a estos dos vértices. Para confirmar los cambios se pulsará al botón Aceptar.

- Grafo Dirigido: Se nos mostrará la siguiente pantalla:

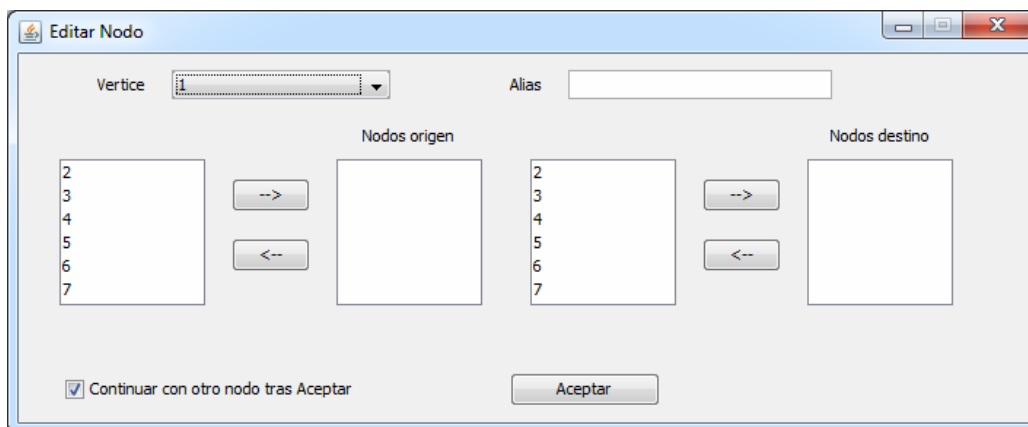


Figura 55. Ventana de edición de un grafo dirigido

En el cuadro de selección Vértice podremos seleccionar el vértice que deseamos editar. Una vez seleccionado se cargarán los datos referidos a dicho vértice: su alias (si tuviese).

Además nos mostrará los vértices que son origen de una arista cuyo destino es el vértice en edición (serán los vértices que aparecen en el cuadro “Nodos origen”) y los vértices que son destino de una arista cuyo origen es el vértice en edición (serán los vértices que aparecen en el cuadro “Nodos destino”).

Al igual que con los grafos no dirigidos se dispone de la posibilidad de añadir/suprimir aristas seleccionando un vértice y añadiéndolo o quitándolo del cuadro correspondiente.

Para confirmar los cambios se pulsará al botón Aceptar.

4.2.10. Insertar Vértices

Para añadir un vértice, el usuario tendrá que pulsar sobre el botón “Insertar Nodo” de la barra de Iconos (Botones).

Una vez pulsado este botón, se podrá comprobar que el botón queda seleccionado y el usuario únicamente tendrá que pulsar con el botón izquierdo del ratón en la zona del lienzo donde desea situar el vértice.

4.2.11. Insertar Arista

Para añadir una arista entre dos vértices del grafo, el usuario dispone de dos posibilidades distintas.

- Edición del Nodo: Como ya se ha explicado en la sección Editar Nodo.
- Manualmente sobre el lienzo: Para ello, el usuario tendrá que pulsar sobre el botón “Insertar Arista” de la barra de Iconos (Botones).

Para crear la arista el usuario únicamente tendrá que pulsar con el botón izquierdo del ratón sobre el vértice que será origen de la arista, y sin soltar dicho botón, mover el ratón hasta colocar el puntero sobre el vértice que será destino de la arista, una vez encima del vértice, bastará con soltar el botón del ratón para generar la arista. Durante el moviendo del ratón se hace visible la arista que se moverá por el lienzo siguiendo al puntero del ratón, hasta que llegue al vértice deseado.

4.2.12. Eliminar Nodo

Para eliminar un vértice del grafo, el usuario dispone de dos maneras distintas para realizarlo. Hay que recordar que cuando se elimine un vértice del grafo, a su vez se eliminarán todas las aristas que inciden en él.

- Sobre el lienzo, con el botón derecho del ratón: El usuario pulsará con el botón derecho del ratón sobre el nodo que se desea eliminar, y pulsará en la opción Eliminar Vértice del menú que aparecerá junto al puntero del ratón. Una vez seleccionada esta opción el vértice y todas las aristas que lleguen hasta él desaparecerán.
- Sobre el lienzo, con el botón izquierdo del ratón: Para ello, el usuario tendrá que pulsar sobre el botón “Eliminar Vértices” de la barra de Iconos (Botones).

Una vez pulsado este botón, se podrá comprobar que el botón queda seleccionado. Para eliminar el vértice, el usuario únicamente tendrá que pulsar con el botón izquierdo del ratón sobre el vértice que desea borrar, una vez realizado esto, se podrá comprobar que el vértice y las aristas que incidían sobre él han sido borrados.

4.2.13. Suprimir Arista

Para eliminar una arista del grafo, el usuario dispone de dos maneras distintas para realizarlo.

- Sobre el lienzo, con el botón derecho del ratón: El usuario pulsará con el botón derecho del ratón sobre arista que se desea eliminar, y pulsará en la opción Eliminar Arista del menú que aparecerá junto al puntero del ratón.
- Sobre el lienzo, con el botón izquierdo del ratón: Para ello, el usuario tendrá que pulsar sobre el botón “Eliminar Arista” de la barra de Iconos (Botones).

Una vez pulsado este botón, se podrá comprobar que el botón queda seleccionado. Para eliminar la arista, el usuario únicamente tendrá que pulsar con el botón izquierdo del ratón sobre la arista que desea borrar, una vez realizado esto, se podrá comprobar que la arista ha sido borrada.

4.2.14. Cambiar las opciones

Esta ventana nos permite modificar en cualquier momento la configuración de la aplicación, modificándose tanto para el grafo abierto, como para los que se abran a continuación.

Para acceder a la ventana de opciones el usuario deberá pulsar en el menú Herramientas y seleccionar Opciones. O pulsar sobre el botón Opciones de la barra de Iconos (botones).

Una vez realizado esto se nos abrirá la ventana de opciones:

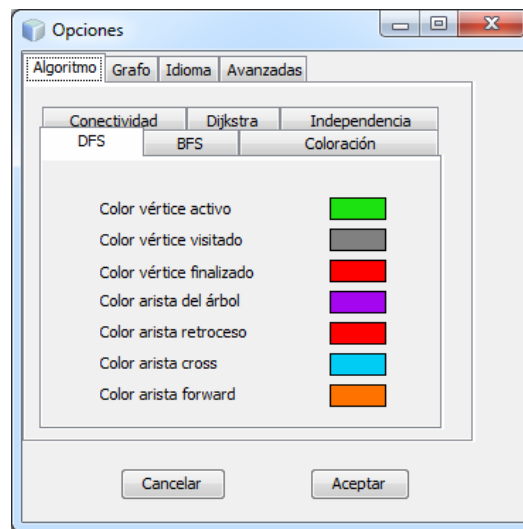


Figura 56. Ventana de opciones

Podremos ver en la mayoría de las pestañas de las opciones algo similar a esto:

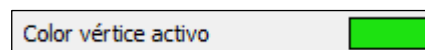


Figura 57. Detalle de un elemento de la ventana de opciones

En este ejemplo podemos comprobar que el Color para los vértices de corte es el rojo. Si el usuario decide cambiar dicho color solamente tendrá que hacer un clic con el botón izquierdo del ratón sobre el cuadro con color. Una vez hecho esto nos aparecerá una pantalla similar a la siguiente:

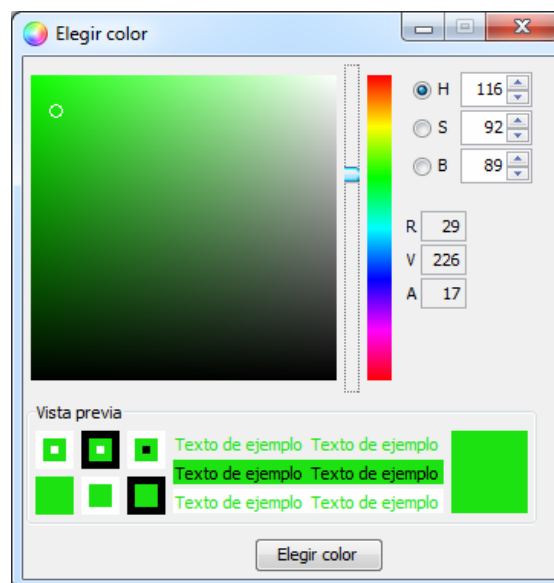


Figura 58. Ventana de selección de color

En ella el usuario podrá desplazar la barra vertical hasta colocarla en una tonalidad de color deseada y a continuación pulsar sobre el color exacto en el cuadro de color de la izquierda, una vez realizado esto, el usuario pulsará sobre el botón elegir color y el color seleccionado aparecerá seleccionado en las opciones.

El usuario podrá comprobar que según se va moviendo la barra de selección de color, los valores de la izquierda se van modificando. En estos cuadros de valor, el usuario podrá introducir manualmente el valor del color deseado (si lo conociese).

Pulsando sobre los botones de selección H, S, B se modificará el formato del cuadro de color de la izquierda, visualizándose de manera distinta.

A continuación iremos explicando cada una de estas partes que componen la ventana de opciones:

- Algoritmo: Configura cada una de las opciones requeridas en cada algoritmo, podemos ver que hay seis subpestañas:
 - DFS: Nos permite configurar el color del vértice activo, visitado y candidato así como el color de la arista, durante la ejecución del algoritmo de búsqueda DFS
 - BFS: Igual que el algoritmo de búsqueda DFS.
 - Coloración: Nos permite seleccionar hasta 20 colores diferentes para elegir durante los algoritmos de coloración.
 - Conectividad: Nos permite seleccionar los colores de vértice de corte, de arista puente y el color de los vértices y aristas visitados durante la ejecución de los algoritmos de conectividad (Búsqueda de vértices de corte y Búsqueda de aristas puente).
 - Dijkstra: Nos permite configurar el color de los vértices visitados, candidatos y activo, así como el color de las aristas, de la distancia y del peso.
 - Bloques: Nos permite configurar el color del vértice activo, del vértice predecesor, de la arista usada y de la arista de retroceso.
 - Independencia: Podremos configurar el color del vértice activo, de los vértices vecinos y de los vértices que no estarán disponibles durante la ejecución del algoritmo de Independencia.
- Grafo: Configura la visualización del grafo antes de la ejecución de algún algoritmo. Podremos seleccionar el color del vértice, de la arista y del vértice seleccionado, así como el grosor tanto del vértice como de la arista.

- Idioma: La aplicación AIG esta internacionalizada. Esto quiere decir que seleccionando cualquiera de los idiomas disponibles esta ventana, todos los textos que aparezcan en la aplicación aparecerán en dicho idioma. Actualmente los idiomas disponibles son español e inglés.
- Avanzadas: Nos permite seleccionar el número de pasos hacia atrás que se pueden dar durante la ejecución de cualquier algoritmo.

4.2.15.Imagen de fondo

A la hora de trabajar con el grafo, el usuario puede insertar una imagen de fondo que le permita colocar los vértices de un nodo sobre unos puntos específicos, por ejemplo, el usuario podría cargar como imagen de fondo un mapa, colocar los vértices sobre ciudades y añadir aristas entre las ciudades.

Para poder cargar una imagen de fondo, el usuario solamente tendrá que pulsar sobre el menú Herramientas y seleccionar Imagen de Fondo o pulsar sobre el botón Imagen de Fondo de la barra de iconos (botones).

En ese momento aparecerá una ventana similar a la de cargar grafo, donde el usuario tendrá que seleccionar el archivo deseado. Una vez seleccionado y pulsado el botón de Abrir, la imagen seleccionada aparecerá como Fondo del Lienzo. Si el archivo seleccionado no es un archivo de imagen, no se mostrará ninguna imagen de fondo.

4.2.16. Estadísticas

El usuario en un determinado momento puede desear obtener una información de cómo será la ejecución de varios algoritmos sobre un mismo grafo. Para poder obtener esta información, deberá pulsar en el menú Herramientas y a continuación seleccionar Estadísticas.

Se nos mostrará una ventana con los algoritmos existentes para realizar las estadísticas. Esta ventana variará dependiendo si tenemos seleccionado Coloración o Búsqueda en el cuadro de selección de Proyecto de la ventana principal de la aplicación.

Las ventanas con los algoritmos que se mostrarán son las siguientes:

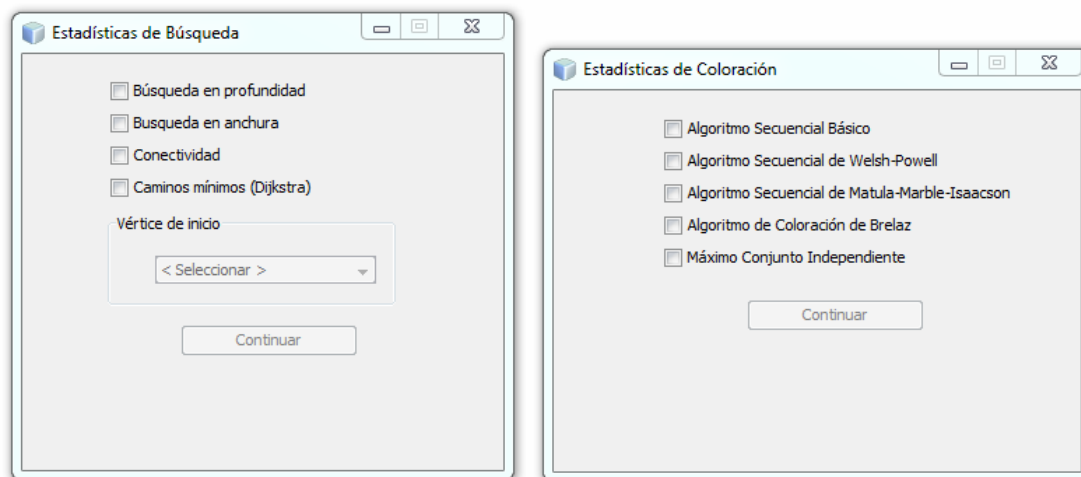


Figura 59. Ventana de selección de algoritmos para la generación de estadísticas

En estas ventanas podemos observar que podemos seleccionar cualquiera de los algoritmos de cada proyecto para obtener sus estadísticas respecto a un grafo.

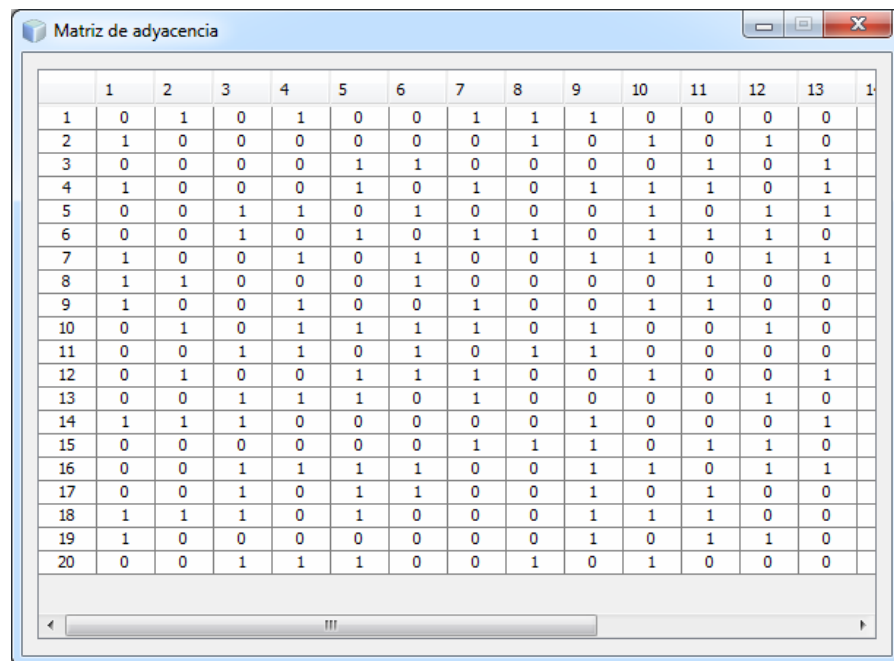
El archivo de estadísticas se generará en formato Excel. Una vez seleccionados los algoritmos y el formato de salida el usuario pulsara sobre el botón “Continuar”, abriéndose una ventana donde el usuario deberá indicar el nombre y la ubicación donde se alojará el archivo.

El archivo generado será el mismo que se genera cuando se exporta un grafo, al que se añadirán partes correspondientes a los algoritmos seleccionados.

Para cada uno de los algoritmos se mostrará información relativa a dicho algoritmo, número de pasos, tiempo de ejecución, estado final del grafo, etc.

4.2.17. Matriz de adyacencia

El usuario dispone de la posibilidad de obtener la matriz de adyacencia del grafo. Para ello deberá pulsar en el menú Herramientas y a continuación en Matriz de Adyacencias o bien pulsar sobre el botón “Matriz de Adyacencia” de la barra de iconos (botones). Una vez hecho esto se mostrará una ventana con la matriz de adyacencias del grafo activo.



	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	0	1	0	1	0	0	1	1	1	0	0	0	0	
2	1	0	0	0	0	0	0	1	0	1	0	1	0	
3	0	0	0	0	1	1	0	0	0	0	1	0	1	
4	1	0	0	0	1	0	1	0	1	1	1	0	1	
5	0	0	1	1	0	1	0	0	0	1	0	1	1	
6	0	0	1	0	1	0	1	1	0	1	1	1	0	
7	1	0	0	1	0	1	0	0	1	1	0	1	1	
8	1	1	0	0	0	1	0	0	0	0	1	0	0	
9	1	0	0	1	0	0	1	0	0	1	1	0	0	
10	0	1	0	1	1	1	1	0	1	0	0	1	0	
11	0	0	1	1	0	1	0	1	1	0	0	0	0	
12	0	1	0	0	1	1	1	0	0	1	0	0	1	
13	0	0	1	1	1	0	1	0	0	0	0	1	0	
14	1	1	1	0	0	0	0	0	1	0	0	0	1	
15	0	0	0	0	0	0	1	1	1	0	1	1	0	
16	0	0	1	1	1	1	0	0	1	1	0	1	1	
17	0	0	1	0	1	1	0	0	1	0	1	0	0	
18	1	1	1	0	1	0	0	0	1	1	1	0	0	
19	1	0	0	0	0	0	0	0	1	0	1	1	0	
20	0	0	1	1	1	0	0	1	0	1	0	0	0	

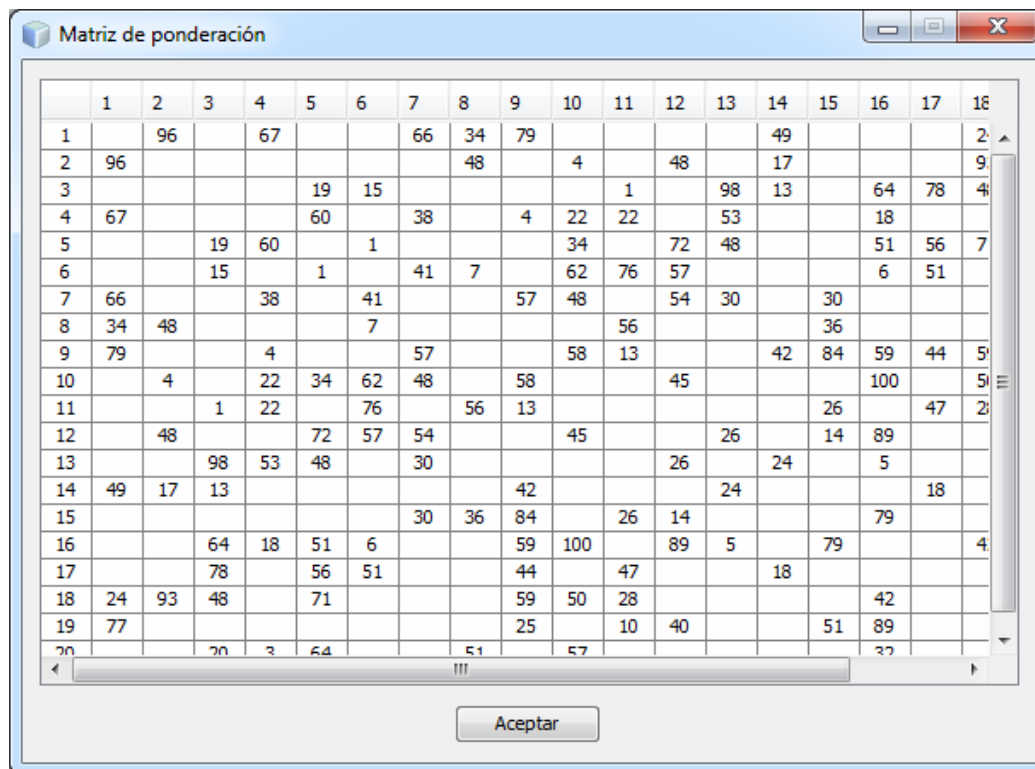
Figura 60. Ventana de la Matriz de Adyacencia

La matriz de adyacencia muestra la conectividad entre los vértices del grafo, apareciendo un 1 si existe una arista entre los vértices y un 0 si no existe dicha arista.

4.2.18. Matriz de ponderación

El usuario dispone de la posibilidad de obtener la matriz de ponderación del grafo. Para ello deberá pulsar en el menú Herramientas y a continuación en Matriz de Ponderación o bien pulsar sobre el botón “Matriz de Ponderación” de la barra de iconos (botones). Una vez hecho esto se mostrará una ventana con la matriz de ponderación del grafo activo.

La matriz de ponderación muestra el peso de las aristas existentes entre dos vértices del grafo. Los valores existentes se pueden modificar, para ello, el usuario deberá ir a la celda correspondiente, realizar un doble clic sobre ella y a continuación escribir el valor deseado. Una vez realizados todos los cambios, el usuario pulsará el botón Aceptar para salvar dichas modificaciones.



	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1		96		67			66	34	79					49						
2	96							48		4		48		17						
3					19	15					1		98	13		64	78			
4	67				60		38		4	22	22		53			18				
5			19	60		1				34		72	48			51	56			
6			15		1		41	7		62	76	57				6	51			
7	66			38		41			57	48		54	30		30					
8	34	48				7					56				36					
9	79			4			57			58	13			42	84	59	44			
10		4		22	34	62	48		58			45				100				
11			1	22		76		56	13						26		47			
12		48			72	57	54			45			26		14	89				
13			98	53	48		30					26		24		5				
14	49	17	13						42					24				18		
15							30	36	84		26	14				79				
16			64	18	51	6			59	100		89	5		79					4
17			78		56	51			44		47			18						
18	24	93	48		71				59	50	28						42			
19	77								25		10	40			51	89				
20								51		57										

Figura 61. Ventana de la Matriz de Ponderación

4.2.19. Grafos tipo

Hay determinados grafos que son utilizados frecuentemente en Teoría de Grafos por tener propiedades características o por ser base de otros grafos. Debido a su alto uso se le ofrece al usuario la posibilidad de utilizarlos sin tener que crearlos manualmente. Para ello el usuario deberá pulsar en el menú Grafos y seleccionar en grafo deseado. A continuación listaremos los grafos que el usuario podrá elegir:

- Nulo
- Completo (K_n)
- Bipartito completo ($K_{r,s}$)
- Tripartito completo ($K_{r,s,t}$)
- Ciclo
- Grafo de Heawood
- Estrella
- Rueda
- Grafos de Rejilla: Rectangular y Triangular
- Cubo
- Grafos Platónicos: Tetraedro, Hexaedro, Octaedro, Icosaedro y Dodecaedro
- Grafo de Herschel
- Grafos Harary
- Enlazados: Enlazado $L_{n,r}$ y Enlazado $L_{n,r,s}$
- Grafo de Petersen
- Grafo de Grötzsch
- Grafo Tutte

4.2.20. Ejecución

Como ya hemos comentado antes, la funcionalidad principal de la aplicación es la de ejecutar ciertos algoritmos de Búsqueda o de Coloración sobre un grafo activo en la aplicación.

Para poder realizar esto, una vez que ya se tiene cargado un grafo sobre la aplicación, el usuario pulsará sobre el icono de Ejecución en la barra de iconos o en el Menú Herramientas y en ejecución.

Una vez hecho esto aparecerá una ventana donde el usuario podrá seleccionar el algoritmo a ejecutar. Esta pantalla dependerá del tipo de Proyecto que se haya tenga seleccionado en este momento. A continuación mostraremos las pantallas de selección de algoritmos:

- Selección de algoritmos de Coloración:

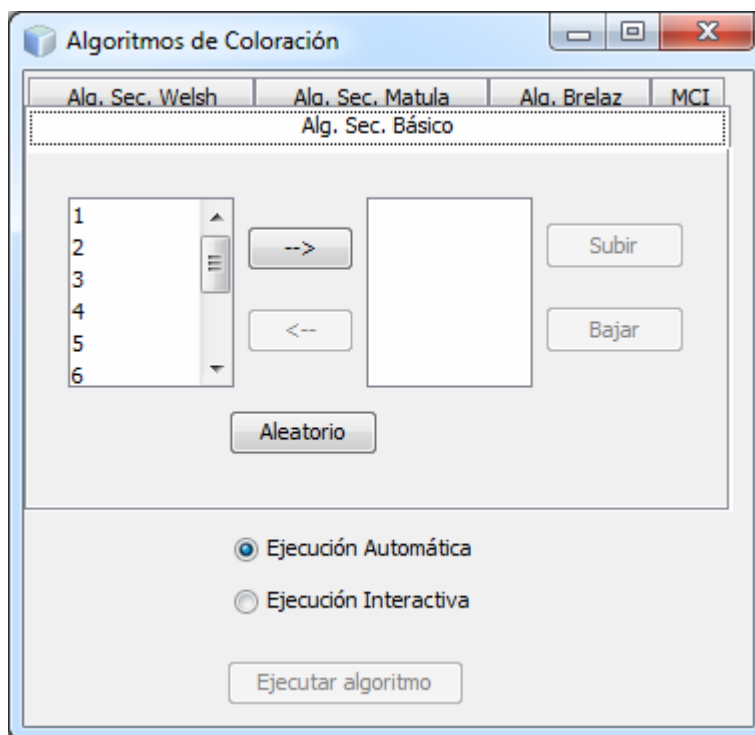


Figura 62. Ventana de selección de Algoritmos de Coloración

El usuario podrá elegir ejecutar uno de los cinco algoritmos posibles, estos son: Algoritmo Secuencial Básico (Aleatorio), Algoritmo Secuencial de Welsh, Algoritmo Secuencial de Matula, Algoritmo de Brelaz y el Algoritmo de Independencia (MCI).

Para poder elegir uno de estos algoritmos el usuario deberá pulsar sobre la pestaña correspondiente y a continuación, seleccionar si se desea que la ejecución sea Automática o Interactiva.

En el caso del algoritmo secuencial Básico, se ofrece al usuario la posibilidad de elegir el orden en el que se van a tratar los nodos. Si el usuario no desea ningún orden en especial, pulsará el botón Aleatorio para que los nodos se coloquen en un orden determinado por la aplicación. Si el usuario selecciona que la ejecución sea automática hasta que todos los vértices del grafo no estén colocados en un orden no se podrá proceder a la ejecución de este algoritmo. Sin embargo si la ejecución es interactiva no hace falta seleccionar ningún orden determinado, ya que este orden lo ira determinando el usuario durante la ejecución.

Si el usuario ha seleccionado el algoritmo de Brelaz, una vez que haya pulsado el botón de “Ejecutar Algoritmo” en la ventana de información, podremos ver que aparece una nueva pestaña con información relativa al algoritmo, que resultará útil para poder comprender la ejecución de este algoritmo:

<div> <div>Árbol</div> <div>Brelaz</div> </div>				
Vértice	$g(v)$	$gs(v)$	Orden	Color
V1	2	0		
V2	3	0		
V3	2	0		
V4	3	0		
V5	3	0		
V6	2	0		
V7	3	0		

Figura 63. Detalle de la tabla del algoritmo de Brelaz

Esta tabla nos mostrará información relativa a cada vértice, durante la ejecución de cada paso del algoritmo de Brelaz, en ella se nos muestra además del vértice, el grado del vértice ($g(v)$), el grado de saturación del vértice ($gs(v)$), el orden en que ha sido coloreado y el color que se le ha asignado.

- Selección de algoritmos de Búsqueda:

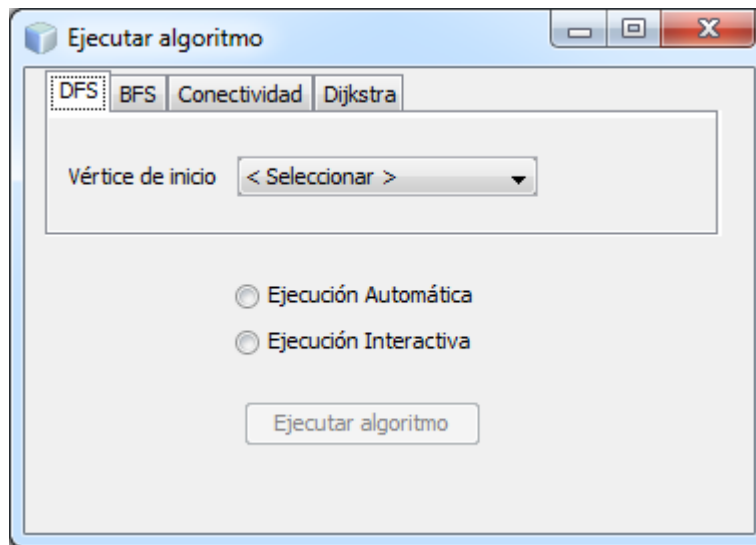


Figura 64. Ventana de selección de Algoritmos de Búsqueda

El usuario podrá elegir ejecutar uno de los cuatro algoritmos posibles, estos son: Algoritmo DFS, Algoritmo BFS, Conectividad (algoritmo de Bloques) y Algoritmo de Dijkstra.

Para poder elegir uno de estos algoritmos el usuario deberá pulsar sobre la pestaña correspondiente y a continuación, seleccionar si se desea que la ejecución sea Automática o Interactiva.

Dependiendo del algoritmo seleccionado el usuario deberá indicar una información suplementaria, por ejemplo, para los algoritmos DFS y BFS, el usuario deberá indicar cual es el vértice e inicio.

En el algoritmo de Dijkstra a parte de tener que seleccionar el vértice de inicio del algoritmo, nos encontramos con dos botones adicionales:

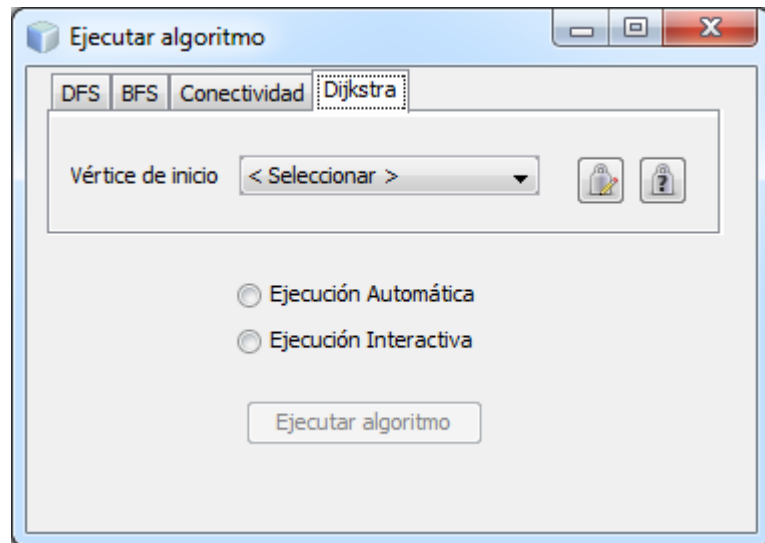




Figura 65. Detalle de la pestaña del algoritmo de Dijkstra en la ventana de selección de Algoritmos

Estos dos botones son para “Editar la ponderación” y “Generar una Ponderación Aleatoria”.

 El usuario podrá introducir manualmente el peso de las aristas existentes para este algoritmo.


 El usuario obtendrá una ponderación aleatoria del peso de las aristas existentes. Dicho peso podrá ser modificado por el usuario.


Una vez que el usuario ha seleccionado el algoritmo y la información necesaria y ha decidido la forma de ejecución, pulsará sobre el botón “Ejecutar Algoritmo” para comenzar la ejecución del mismo.


4.2.21. Ejecución Automática


Una vez que el usuario ha seleccionado el algoritmo deseado, ha suministrado la información previa necesaria, y ha seleccionado ejecución automática, la pantalla de Ejecución desaparecerá y el usuario podrá comprobar que se han activado los botones de ejecución del algoritmo situados en la parte central de la zona de información de la aplicación.

La funcionalidad de los botones es la siguiente:

 **Paso hacia atrás.** Si durante la ejecución de un algoritmo paso a paso, queremos volver al paso anterior, se pulsará este botón.

 **Iniciar (Play).** Si el usuario desea ejecutar un algoritmo automático hasta que este finalice pulsará el botón de Play.

 **Parar (Pause).** Una vez que el usuario ha pulsado el botón Play, puede parar la ejecución del algoritmo pulsando el botón de Pause.

 **Paso hacia delante.** La manera de avanzar en la ejecución de un algoritmo paso a paso es pulsando este botón.

Durante la ejecución del algoritmo el usuario podrá ir alternando la manera de visualizar la ejecución del algoritmo automático.

El usuario puede pulsar el botón “Play” y esperar a que finalice el algoritmo. También puede ir paso a paso, pulsando el botón “Paso hacia delante” para ver como va evolucionando el grafo.

El usuario dispone de la posibilidad de parar el algoritmo si ha pulsado previamente el botón “Play” y salvar la configuración del grafo para continuar posteriormente.

El usuario podrá comprobar que la ejecución del algoritmo ha finalizado observando que tanto el botón “Play” como el botón de “Paso hacia delante” están deshabilitados, además en el cuadro de información del algoritmo mostrará un texto indicando que dicho algoritmo ha llegado a su fin.

Existe la posibilidad de volver hacia atrás en cualquier momento, para que el usuario compruebe los pasos que ha ido realizando el algoritmo, por si tiene algún tipo de duda o para revisar los pasos realizados.

4.2.22.Ejecución Interactiva

Una vez que el usuario ha seleccionado el algoritmo deseado y ha seleccionado ejecución interactiva, la pantalla de selección de algoritmo desaparecerá.

A partir de este momento, la ejecución interactiva comenzará. Para ello el usuario deberá pulsar con el botón derecho sobre el vértice o arista deseada (dependiendo del algoritmo) y una vez hecho esto aparecerá un menú, donde se mostrarán activas las diversas posibilidades que puede seleccionar el usuario, dependiendo del algoritmo seleccionado.

A continuación veremos como realizar la ejecución interactiva para los algoritmos implementados.

4.2.23.Coloración: Algoritmo Secuencial Básico

En este algoritmo, el usuario tendrá que ir estableciendo los colores de los vértices, sin seguir ningún orden, únicamente tendrá que cumplir que los vértices vecinos no comparten color.

Para ello el usuario pulsará con el botón derecho sobre el vértice al que desea establecer el color, mostrándose el siguiente menú:



Figura 66. Menú contextual de selección de color (algoritmos de coloración)

Podemos comprobar que la única opción que puede elegir el usuario es “Seleccionar color”, una vez que el usuario coloca el puntero sobre dicha opción se nos mostrarán los diversos colores que puede elegir. Una vez que el usuario ha decidido el color deseado, pulsará sobre él y podrá comprobar que el color del nodo, se ha modificado por el color seleccionado.

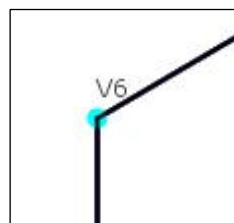


Figura 67. Detalle de un vértice coloreado

El usuario realizará esta operación, tantas veces como vértices existan, hasta que se haya coloreado todo el grafo.

Si en un determinado momento, el usuario selecciona un color que es el mismo que ya tiene un vecino del vértice seleccionado se nos mostrará la siguiente ventana de error:

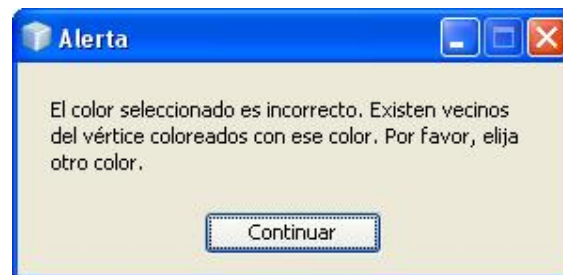


Figura 68. Ventana de Alerta: Color Incorrecto

También puede darse el caso, que el usuario no seleccione el color optimo para el nodo (siguiendo estrictamente el algoritmo ese color no sería el correcto), haciendo que no se utilicen los mínimos colores posibles para colorear el grafo. Cuando se de esta situación, se nos mostrará la siguiente pantalla:

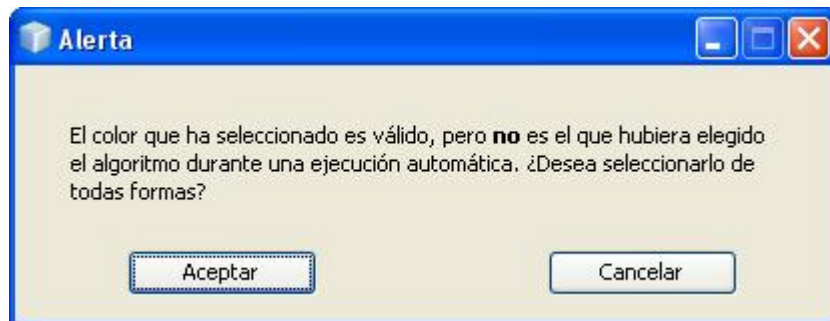


Figura 69. Ventana de alerta: Color no Adecuado

En este momento el usuario tiene la posibilidad de continuar la ejecución del algoritmo, utilizando el color seleccionado (aunque no realizará una coloración optima), pulsando sobre el botón "Aceptar", o seleccionar otro color, para encontrar el color optimo, con lo cual, deberá pulsar sobre el botón "Cancelar".

Existe la restricción de no poder modificar el color elegido a un vértice, con lo que si seleccionamos un color a un vértice al que ya le hayamos asignado uno, se nos mostrará la siguiente ventana:

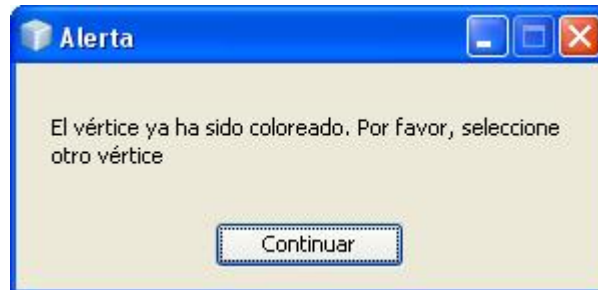


Figura 70. Ventana de alerta: Vértice ya coloreado

Según vayamos asignando colores a los vértices, en el cuadro de información del algoritmo, se irá mostrando información relativa a la ejecución. Mostraremos un pequeño ejemplo de ello:

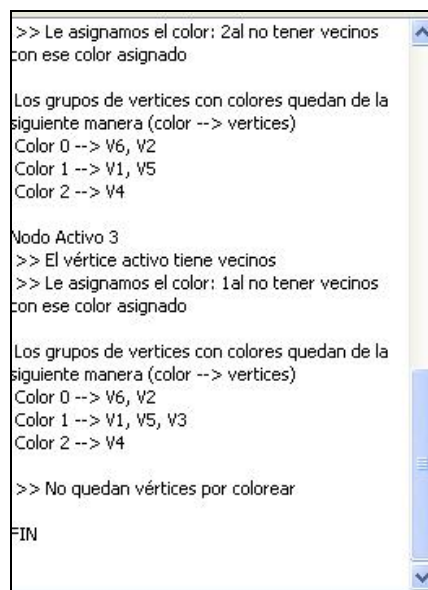


Figura 71. Detalle del cuadro de información del algoritmo de coloración secuencial básico

4.2.24. Coloración: Algoritmo Secuencial de Welsh-Powell

En este algoritmo, es similar al Algoritmo Secuencial Básico, con la única salvedad que el orden de coloración de los vértices. En este caso, el usuario tendrá que ir estableciendo los colores de los vértices, siguiendo un orden, de mayor a menor grado, es decir, en función del número de vértices adyacentes, además tendrá que cumplir que los vértices vecinos no comparten color.

La dinámica de ejecución es exactamente igual que el Algoritmo Secuencial Básico. Se mostrarán las mismas pantallas de alerta, en las mismas situaciones (cuando seleccionemos un color que ya ha sido asignado a un vértice vecino, cuando el nodo ya haya sido coloreado y cuando se seleccione un color que no es el óptimo). Además de estas pantallas de alerta tendremos dos pantallas adicionales:

Si el usuario selecciona un color para un vértice cuyo grado es menor que el grado de un vértice todavía no coloreado nos mostrará la siguiente ventana:

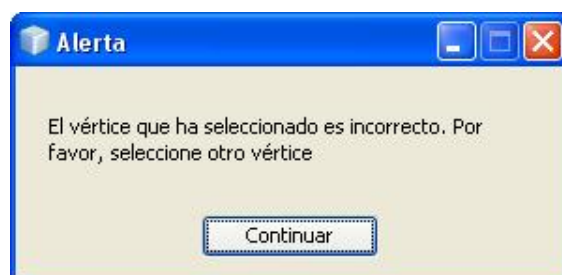


Figura 72. Ventana de alerta: Vértice incorrecto

Como hemos dicho antes, el algoritmo de Welsh-Powell, colorea los vértices en función del grado de los mismos, ordenando los vértices de mayor a menor grado. Cuando dos vértices tienen el mismo grado, el algoritmo los ordena en función del orden de aparición del vértice en el grafo. Pero se ofrece al usuario la posibilidad de no seguir estrictamente esta norma, con lo que se mostrará el siguiente mensaje:

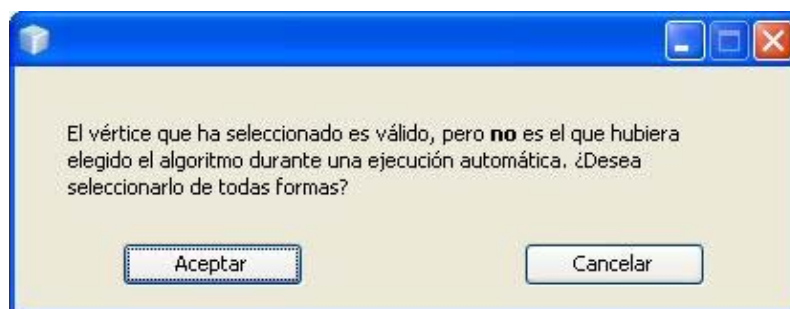


Figura 73. Ventana de alerta: Vértice no adecuado

Se indica que el vértice seleccionado no sigue el orden que hubiera seguido el algoritmo, pero como tiene el mismo grado que el vértice que debería haberse elegido, se le ofrece la posibilidad de poder continuar con la ejecución, coloreando dicho nodo, pero se le indica que no es la solución que hubiese elegido el algoritmo si se hubiese realizado una ejecución automática.

4.2.25. Coloración: Algoritmo Secuencial de Matula-Marble-Isaacson

En este algoritmo, es similar al Algoritmo Secuencial Básico, con la única salvedad que el orden de coloración de los vértices. En este caso, el orden de coloración de los nodos es inverso del orden de selección. El orden de selección se realizará primero estableciendo el vértice de menor grado. El siguiente vértice será el vértice de menor grado del grafo resultante de quitar al vértice seleccionado anteriormente al grafo original. Este procedimiento se ira haciendo hasta que se hayan seleccionado todos los vértices.

Una vez obtenido el orden de selección de los nodos, comenzaremos a colorear el último vértice seleccionado, a continuación el siguiente, y así hasta que lleguemos a colorear al primer vértice que se selecciono.

La dinámica de ejecución es exactamente igual que el Algoritmo Secuencial Básico. Se mostrarán las mismas pantallas de alerta, en las mismas situaciones (cuando seleccionemos un color que ya ha sido asignado a un vértice vecino, cuando el nodo ya haya sido coloreado y cuando se seleccione un color que no es el óptimo).

Si el usuario selecciona un color para un vértice, que no cumple con los requisitos del algoritmo, se nos muestra la siguiente ventana

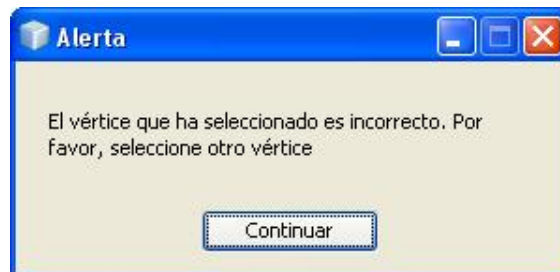


Figura 74. Ventana de alerta: Vértice incorrecto

4.2.26. Coloración: Algoritmo de Coloración de Brelaz

En este algoritmo, el orden de coloración de los vértices depende del grado de los vértices ($g(V)$), y del grado de saturación o color de los vértices ($gs(V)$), y es determinado en tiempo de ejecución. Por lo que es bastante importante recurrir a la pestaña de Brelaz que aparecerá en la zona de información de la aplicación.

Árbol		Brelaz		
Vértice	$g(v)$	$gs(v)$	Orden	Color
V1	2	0		
V2	3	0		
V3	2	0		
V4	3	0		
V5	3	0		
V6	2	0		
V7	3	0		

Figura 75. Detalle de la tabla del algoritmo de Brelaz

El grado de un vértice es el número de adyacentes del mismo, y el grado de saturación es el número de colores no repetidos usados en los adyacentes o vecinos.

La selección de nodos a colorear vendrá determinada primeramente por el grado de saturación, seleccionándose el vértice con mayor grado de saturación, y en caso de que existan varios con el mismo valor, se elegirá de entre estos, el vértice que tenga mayor grado. Si aun así existiese más de un vértice a seleccionar, se elegiría dependiendo del nombre, con lo que se $v1$ y $v4$ tienen el mismo grado de saturación y el mismo grado, se coloreará primeramente a $v1$ antes que a $v4$.

Una vez que se ha coloreado un vértice, el grado de saturación de cada vértice no coloreado volverá a recalcularse. A parte de esto, se debe seguir con el criterio de no colorear un vértice con un color con el que ya se haya coloreado a un vértice vecino (adyacente).

Se mostrarán ventanas de alerta cuando:

- Se seleccione un color ya usado por un vértice vecino
- Cuando se seleccione un color para un vértice ya coloreado
- Cuando se seleccione un vértice que no es el correcto a colorear.
- Cuando se seleccione un color que no coincide con el color que se deberá seleccionar.

Debemos fijarnos que para la ejecución de este algoritmo no se permite seleccionar un color no óptimo, ni seleccionar un vértice que no sea el correcto, aunque coincidan en grado de saturación y grado del vértice.

4.2.27. Coloración: Algoritmo de Independencia MCI

En este algoritmo lo que se busca son conjuntos de independencia, y una vez que ya hayan sido establecidos todos los conjuntos independientes, la aplicación asignará un color a los vértices, coloreando los vértices de cada conjunto independiente con el mismo color.

El usuario deberá ir seleccionando los vértices con menor grado, para ir añadiéndolos al conjunto independiente:



Figura 76. Menú contextual en la ejecución del algoritmo MCI

Una vez que el usuario ha seleccionado el vértice correcto, podemos comprobar que el color tanto del vértice seleccionado, como sus vecinos ha cambiado.

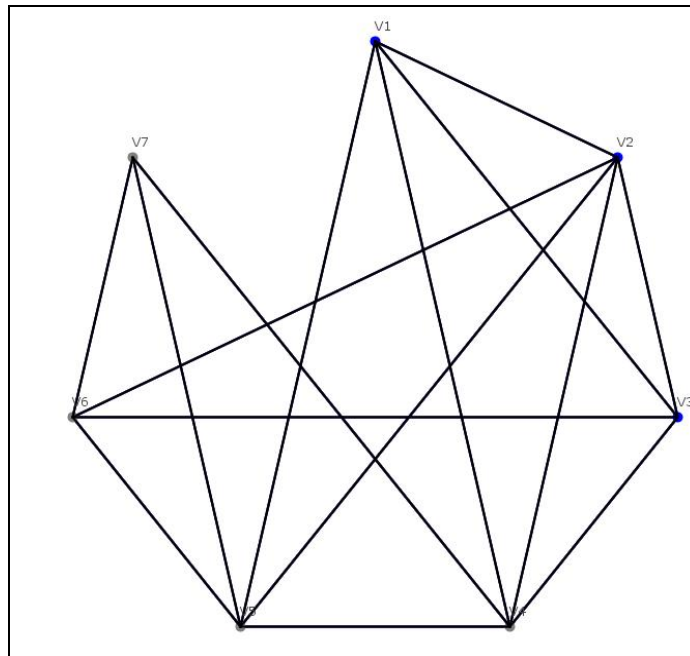


Figura 77. Detalle del lienzo en el algoritmo MCI, mostrando vértices no disponibles

Este cambio de color, nos indica que dichos vértices ya no pueden usarse para obtener el conjunto independiente, el vértice seleccionado, por que ya pertenece a él, y sus vecinos por que no pueden pertenecer al conjunto independiente ya que existe un vértice adyacente a ellos que ya pertenece a dicho conjunto independiente.

Con los vértices disponibles se sigue el mismo criterio de selección, dependiendo del grado de cada vértice, con la salvedad que para calcular el grado, no hay que tener en cuenta a los vértices no disponibles, con lo que para ver que vértice seleccionar, al grado real del vértice habrá que restarle el número de vértices vecinos que no pueden ser seleccionados para el conjunto independiente, por que ya hayan sido seleccionados o por que algún vecino pertenece al conjunto independiente.

Una vez que ya no quedan más vértices disponibles para formar parte del conjunto independiente se recalcula el grafo, mostrando los vértices que no forman parte de ningún conjunto independiente calculado previamente, y las aristas existentes entre estos vértices:

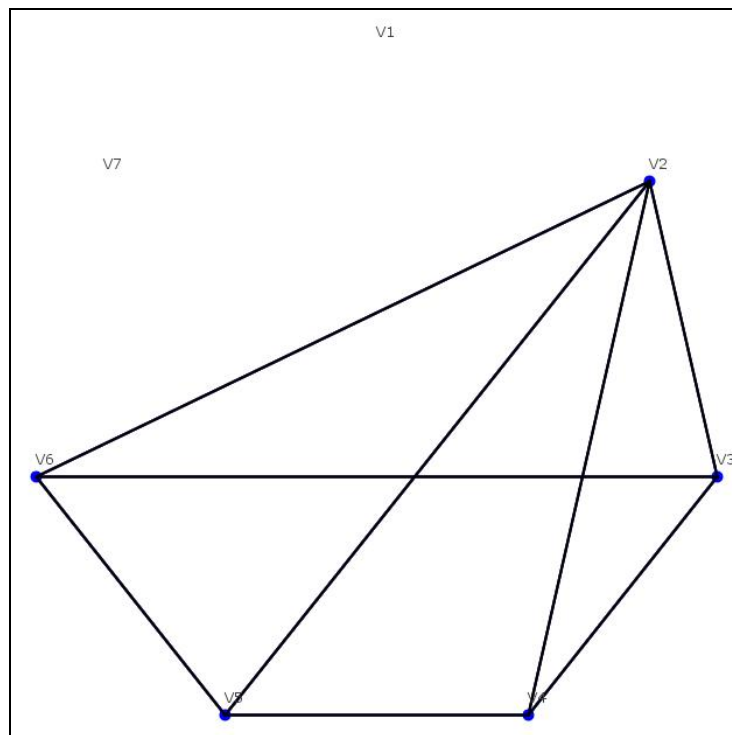


Figura 78. Detalle del lienzo en el algoritmo MCI, después de recalcular el grafo

Sobre el grafo resultante se volverá a calcular los vértices que forman parte del nuevo conjunto independiente.

Esto procedimiento se realizará hasta que no quede ningún vértice sin pertenecer a un conjunto independiente. Una vez que esto ocurra la

aplicación, volverá a mostrar el grafo inicial, coloreando los vértices según su conjunto independiente:

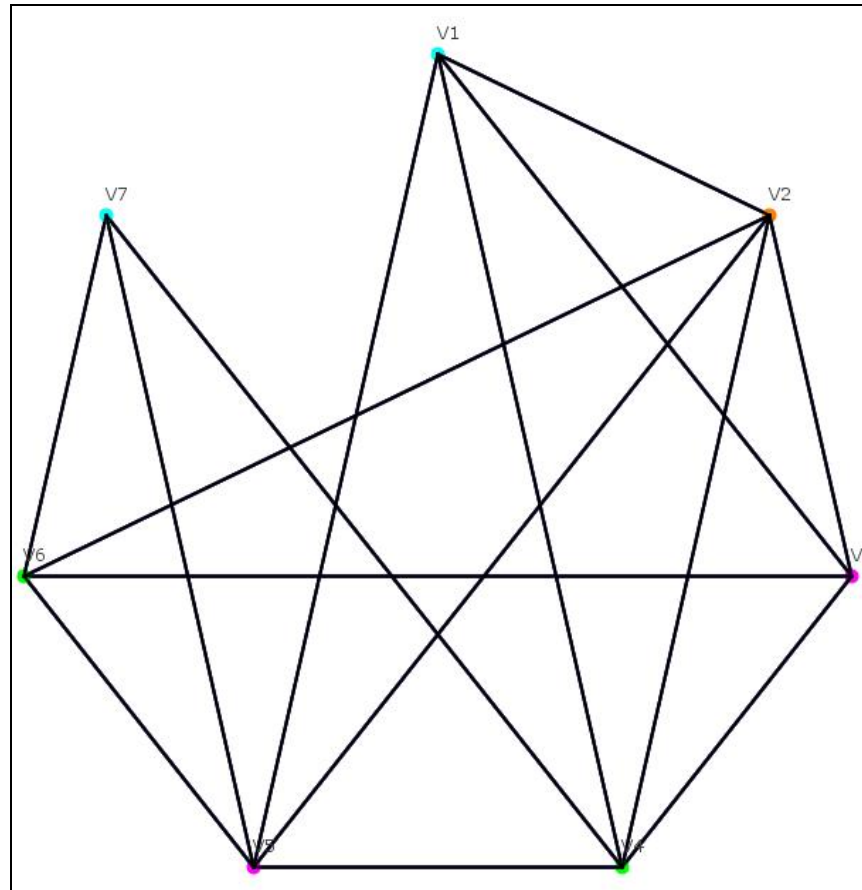


Figura 79. Detalle del lienzo en el algoritmo MCI, con los vértices coloreados

Debemos fijarnos que existen cuatro colores distintos, lo que quiere indicar que existen cuatro conjuntos independientes diferentes.

4.2.28.Búsqueda: Algoritmo DFS

En este algoritmo, el usuario tendrá que ir estableciendo los vértices que se van recorriendo, partiendo del vértice inicial seleccionado:

Para ello el usuario pulsará con el botón derecho sobre el vértice que desea seleccionar a continuación, mostrándose el siguiente menú:

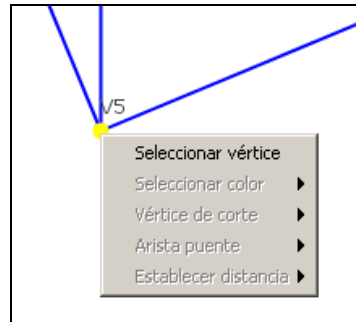


Figura 80. Menú contextual en la ejecución del algoritmo DFS

Podemos comprobar que la única opción que puede elegir el usuario es “Seleccionar vértice”. El usuario realizará esta operación para recorrer las aristas de cada vértice, tal y como indica el algoritmo, definiendo la búsqueda en profundidad sobre el grafo. Cuando un vértice es correctamente elegido, pasa a ser vértice actual (el vértice actual anterior pasa a ser visitado) y se reconfiguran los nuevos vértices candidato (identificados por un círculo blanco en el centro).

A medida que se vayan recorriendo las aristas, éstas definirán su naturaleza:

- arista del árbol: si el vecino al que alcanzan no ha sido aún visitado.
- arista de retroceso: si el vecino al que alcanzan ha sido visitado.

En el caso de que el grafo sea dirigido, aparecen dos nuevos tipos de arista:

- arista forward: si el vecino ha sido finalizado, en un orden de búsqueda posterior al del vértice activo.
- arista cross: si el vecino ha sido finalizado, en un orden de búsqueda anterior al del vértice activo.

El usuario deberá seleccionar uno de los vértices candidato. Si selecciona un vértice diferente, se mostrará el correspondiente mensaje de error:

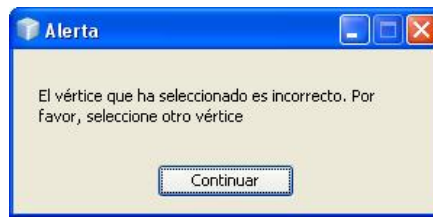


Figura 81. Ventana de Alerta: Vértice Incorrecto

Dentro de los vértices candidato, el algoritmo siempre elegirá aquél que esté situado en la cima de la pila de ejecución. Si el usuario selecciona otro diferente de entre los candidatos, el sistema mostrará un mensaje advirtiéndolo de que no es el que hubiera elegido el algoritmo, a pesar de ser una elección igualmente válida:

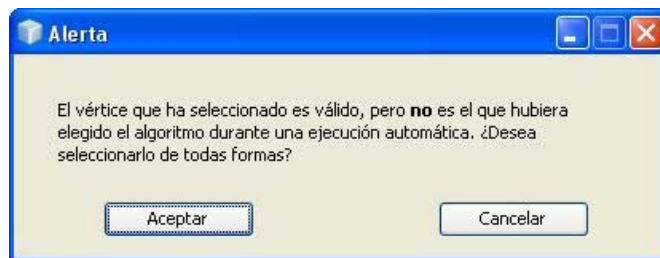


Figura 82. Ventana de alerta: Vértice no adecuado

En este momento el usuario tiene la posibilidad de continuar la ejecución del algoritmo utilizando el vértice seleccionado, pulsando sobre el botón “Aceptar”, o seleccionar otro vértice, pulsando sobre el botón “Cancelar”.

Si en un momento dado no hay candidatos (porque todos los vecinos del vértice actual han sido visitados), pero aún quedan vértices por visitar, significará que el grafo no es conexo, por lo que el usuario deberá seleccionar uno de los restantes vértices (el algoritmo elegiría el de menos índice, aunque cualquiera de éstos sería válido). El algoritmo finalizará cuando todos los vértices hayan sido visitados y finalizados.

Según vayamos seleccionando vértices, en el cuadro de información del algoritmo, se irá mostrando información relativa a la ejecución.

4.2.29. Búsqueda: Algoritmo BFS

En este algoritmo, el usuario tendrá que ir estableciendo los vértices que se van recorriendo, partiendo del vértice inicial seleccionado:

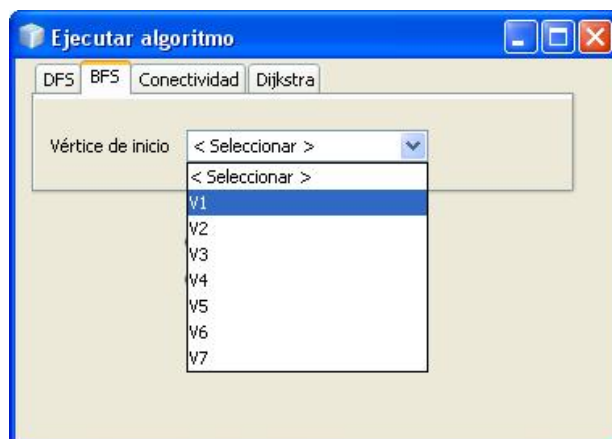


Figura 83. Ventana de Algoritmo de Búsqueda BFS. Selección de Vértice de Inicio

Para ello el usuario pulsará con el botón derecho sobre el vértice que desea seleccionar a continuación, mostrándose el siguiente menú:

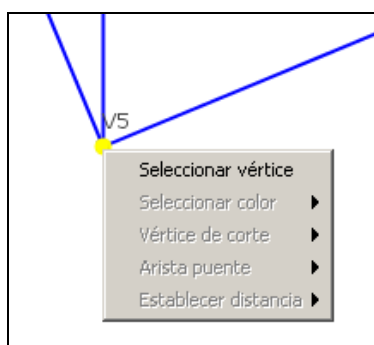


Figura 84. Menú contextual en la ejecución del algoritmo BFS

Podemos comprobar que la única opción que puede elegir el usuario es “Seleccionar vértice”. El usuario realizará esta operación, tantas veces como vértices existan, definiendo la búsqueda en anchura sobre el grafo. Cuando un vértice es correctamente elegido, pasa a ser vértice visitado (ya no será candidato).

A la hora de elegir vértice a seleccionar, en el grafo se mostrarán de un color diferente, aquéllos que son candidatos a ser seleccionados: no han sido seleccionados previamente, y son vecinos del vértice actual. El usuario deberá seleccionar uno de los vértices candidato. Si selecciona un vértice diferente, se mostrará el correspondiente mensaje de error:

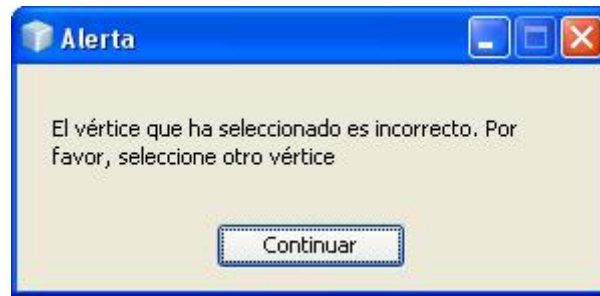


Figura 85. Ventana de Alerta: Vértice Incorrecto

Dentro de los vértices candidato, el algoritmo siempre elegirá aquél de menor índice. Si el usuario selecciona otro diferente de entre los candidatos, el sistema mostrará un mensaje advirtiéndolo de que no es el que hubiera elegido el algoritmo, a pesar de ser una elección igualmente válida:

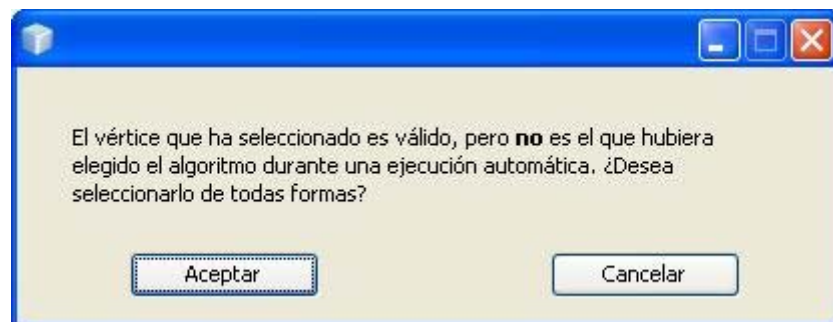


Figura 86. Ventana de alerta: Vértice no adecuado

En este momento el usuario tiene la posibilidad de continuar la ejecución del algoritmo utilizando el vértice seleccionado, pulsando sobre el botón “Aceptar”, o seleccionar otro vértice, pulsando sobre el botón “Cancelar”.

Si todos los vecinos del vértice actual han sido visitados, el usuario deberá seleccionar el vértice actual tal y como lo haría el algoritmo: ir hacia atrás sacando elementos de la cola hasta encontrar un vértice con vecinos por visitar. Si la cola se queda vacía y no hay candidatos, el algoritmo llega a su fin (aunque queden vértices por visitar).

Según vayamos seleccionando vértices, en el cuadro de información del algoritmo, se irá mostrando información relativa a la ejecución.

Una vez haya finalizado la ejecución del algoritmo, el usuario tiene la opción de visualizar el árbol de búsqueda resultante, utilizando el icono a tal efecto.

4.2.30. Búsqueda: Conectividad: algoritmo de bloques

La aplicación nos permite realizar una ejecución interactiva para determinar los bloques en los que se puede descomponer un grafo.

Recordemos que un bloque es un grafo no trivial, conexo y sin vértices de corte, por lo que el algoritmo generará bloques tras la detección de los vértices de corte.

El algoritmo de bloques utiliza el doble etiquetado (orden de búsqueda, función L). El usuario deberá ir pulsando con el botón derecho sobre los vértices del grafo candidatos siguiendo el algoritmo, y éste irá recalculando los valores del doble etiquetado.

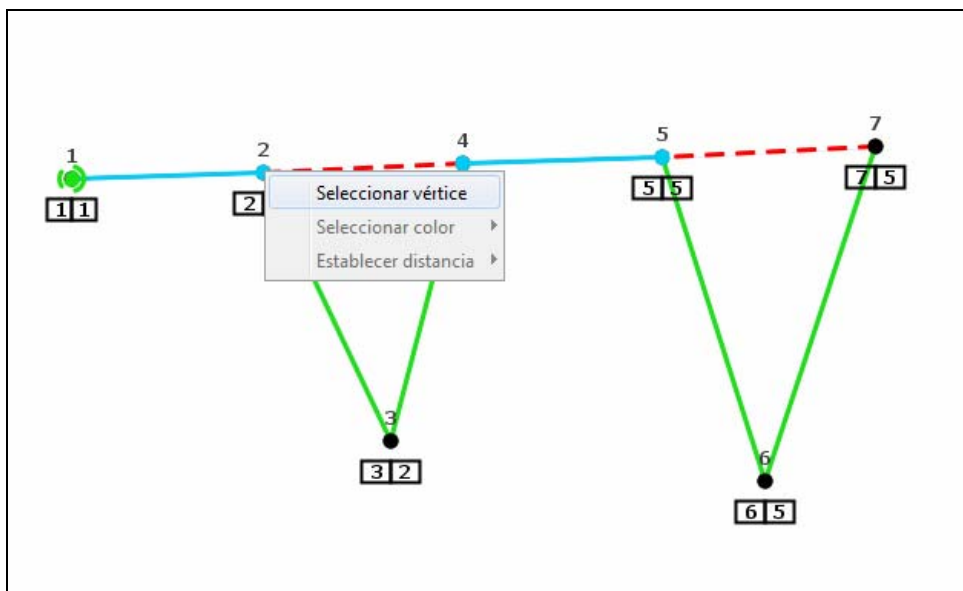


Figura 87. Menú contextual en la ejecución del algoritmo de Bloques

Si el usuario se ha equivocado al seleccionar un vértice (por no ser candidato), se nos mostrará la siguiente ventana:

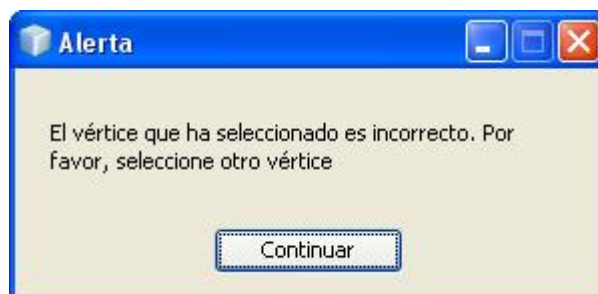


Figura 88. Ventana de Alerta: vértice no determinado correctamente

4.2.31. Búsqueda: Algoritmo de Dijkstra

En este algoritmo, el usuario tendrá que ir estableciendo los vértices que se van recorriendo, partiendo del vértice inicial seleccionado:

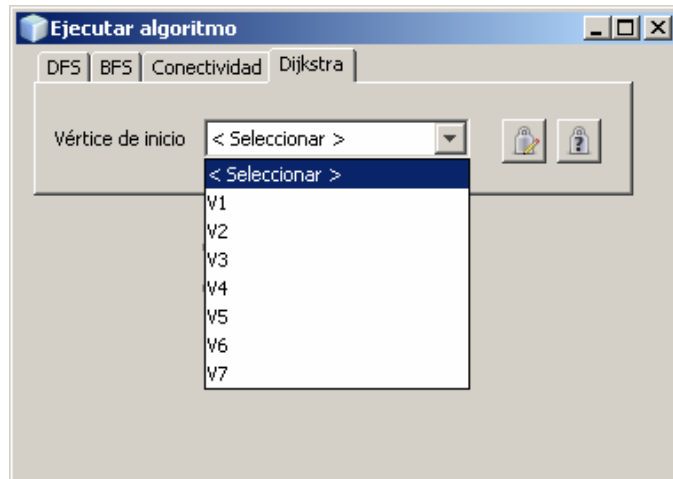




Figura 89. Ventana de Algoritmo de Dijkstra. Selección de Vértice de Inicio

Además de seleccionar el vértice de inicio, el usuario tiene la posibilidad de visualizar la matriz de ponderación (pesos o distancias de las aristas del grafo) con el fin de editarla mediante el icono , e incluso asignar valores aleatorios a las aristas (inicialmente establecidos en 1), mediante el icono .

A continuación el usuario pulsará con el botón derecho sobre el vértice que desea seleccionar pudiéndosele mostrar dos tipos de menú:

- Seleccionar el siguiente vértice activo:

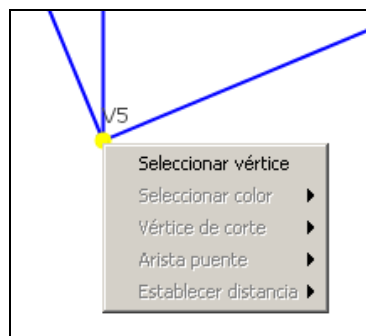


Figura 90. Menú contextual en la ejecución del algoritmo de Dijkstra. Seleccionar vértice

Si el vértice activo no tiene más candidatos, deberá seleccionar el siguiente vértice activo. El actual vértice activo parará a ser un vértice visitado (con su

color correspondiente en el lienzo). Si, por el contrario, todavía existen más candidatos, la aplicación mostrará un error.

A la hora de elegir el siguiente vértice activo, se deberá seleccionar aquél cuya distancia al origen sea mínima (las distancias mínimas se visualizan en color rojo en la imagen siguiente):

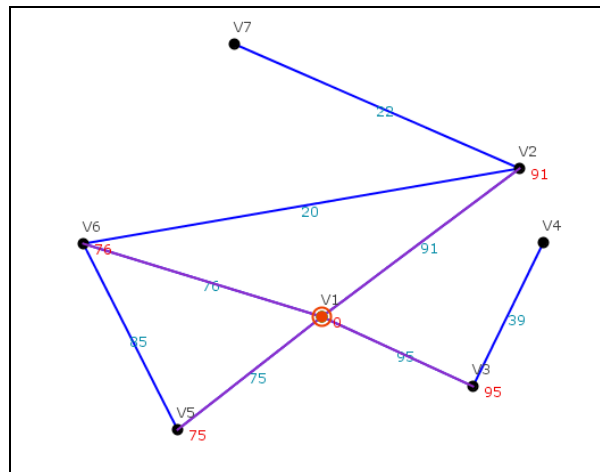


Figura 91. Detalle del lienzo en el algoritmo de Dijkstra

Si no fuera así, el sistema mostrará igualmente un error:

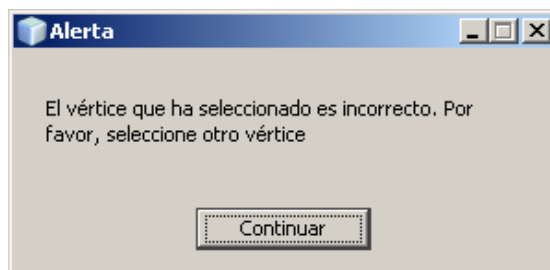


Figura 92. Ventana de Alerta: Vértice seleccionado incorrecto

- Seleccionar la distancia mínima al vértice seleccionado:

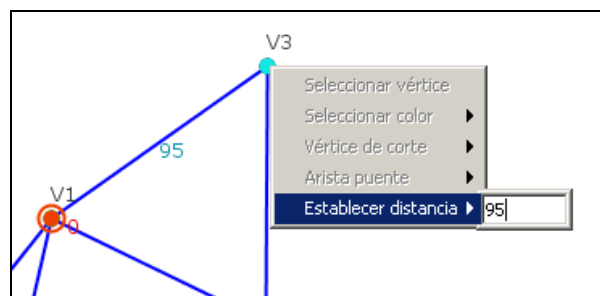


Figura 93. Menú contextual en la ejecución del algoritmo de Dijkstra. Establecer distancia

Si hace clic sobre un vértice candidato, podrá determinar la distancia. Para ello, deberá escribirla en el recuadro y pulsar la tecla <Intro>. Si introdujo la distancia correcta, el algoritmo continuará avanzando, y dicho vértice ya no será candidato para el vértice actual. Se establecerá la distancia mínima para dicho vértice. En caso contrario, la aplicación mostrará un error:

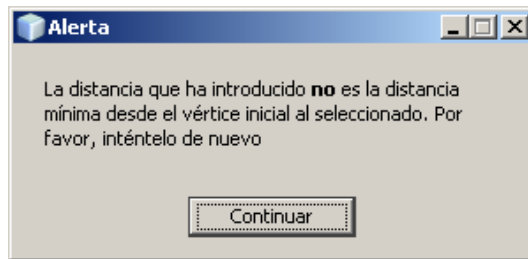


Figura 94. Ventana de Alerta: Distancia incorrecta

El usuario realizará esta operación, determinando la distancia mínima al origen, para cada vecino (y a su vez para cada vértice, en el orden correcto), definiendo la búsqueda de caminos mínimos de Dijkstra.

Según vayamos seleccionando vértices, en el cuadro de información del algoritmo, se irá mostrando información relativa a la ejecución.

5. ESTUDIO ESTADISTICO

Hemos podido ver en secciones anteriores diversos algoritmos de coloración de grafos. Además ya se ha comentado que no se conoce un algoritmo eficiente para colorear los vértices de un grafo.

Debido a este motivo, se ha decidido realizar un estudio estadístico de los resultados de los diversos algoritmos de coloración implementados en la aplicación, sobre un conjunto de grafos.

Para este estudio comparativo, se ha decidido ejecutar los algoritmos sobre 10 grafos de 100, 200, 300, 400 y 500 vértices. Para ello, se ha utilizado la herramienta Estadística de la aplicación AIG, que nos muestra diversa información de cada algoritmo. La información más relevante que vamos a mostrar es el número de colores utilizado por cada algoritmo.

Debido a que se han realizado diez ejecuciones sobre distintos grafos con un mismo número de grafos, no vamos a representar datos numéricos, sino que comentaremos los resultados.

Cabe destacar que se ha establecido, que a la hora de generar los grafos, la probabilidad de que exista una arista entre dos nodos del grafo sea del 20%. Es probable que si se hubiese decidido otra probabilidad de existencia de aristas, los resultados pudiesen variar ligeramente, pero no creemos que lo hiciesen significativamente.

A continuación vamos a comentar los resultados obtenidos.

Para todas las ejecuciones que se han realizado, el algoritmo que presenta peores resultados es el Secuencial Básico. Evidentemente, este algoritmo es el que mayor número de colores utiliza para colorear un grafo, ya que lo único que se tiene en cuenta es el orden en que se colorean los vértices.

En líneas generales, el algoritmo que presenta mejores resultados es el algoritmo de Brelaz, pero según vamos aumentando el número de vértices, la coloración por independencia (MCI), se va acercando al nivel de colores utilizados por Brelaz, incluso mejorando sus resultados en alguna ejecución con grafos de 500 vértices.

Brelaz utiliza de media entre uno y dos colores menos que los algoritmos de Welsh y de Matula, en todos los conjuntos de grafos. Esta situación también se repite con el

algoritmo de MCI, pero como hemos comentado, según vamos aumentando el número de vértices del grafo, el algoritmo MCI va mejorando sus prestaciones.

En estudios realizados en Teoría de Grafos, se expone que los algoritmos de independencia terminan ofreciendo mejores resultados que los algoritmos secuenciales en grafos grandes.

Por este motivo, es de suponer que según se vaya aumentando el número de vértices de los grafos, el algoritmo MCI, vaya siendo el que presente mejores resultados.

6. CONCLUSIONES

En este proyecto, se ha desarrollado una aplicación que permite manejar cualquier tipo de grafo. Además de esto, se puede aplicar al grafo generado una serie de algoritmos de coloración o de búsqueda.

La realización de este trabajo de fin de carrera, ha llevado mucho tiempo, el compaginar la vida laboral, con la familiar y a la vez tener tiempo para realizar el proyecto, suele ser complicado.

La aplicación se ha desarrollado, con la intención de que pueda servir tanto de material docente, por parte del profesorado, como de material de estudio, por parte del alumnado. Por este motivo, durante la ejecución de los algoritmos, se puede ver una explicación de cada uno de ellos. A parte de esto, el usuario podrá ejecutar interactivamente cada algoritmo, gracias a esto, el usuario podrá comprobar el grado de conocimiento que tiene sobre el funcionamiento de cada algoritmo.

Cabe destacar que se ha intentado que el código fuente, este lo más estructurado posible, para que en el futuro, si algún alumno desea continuar el trabajo realizado e incorporar nuevos tipos de algoritmos a la aplicación, lo haga sin ningún tipo de problemas. Para ello la fase de análisis y modelado de clases fue bastante larga.

No hay que olvidar que la realización de esta aplicación, ha sido una tarea compartida con otro compañero, con lo que uno, no solamente he aprendido/recordado cosas relacionadas con la coloración de grafos, sino de temas de búsquedas, recorrido de grafos, etc. El transito de información entre los dos componentes del proyecto ha sido bidireccional, con lo que supongo que mi compañero de proyecto, habrá aprendido también temas relacionados con la coloración. Eso que ganamos.

Espero sinceramente, que haya gente a la que la aplicación le resulte interesante y que exista algún alumno que decida continuar y mejorar nuestro trabajo.

7. BIBLIOGRAFÍA

Libros:

- G. Blanco, J. J. Carreño, A. de la Cruz, M. Foulquié, J. García, F. Gómez, A. I. Lías. *Matemática Discreta*. Servicio de Publicaciones, E. U. de Informática, UPM, 1995.
- T. Dinski, Z. Xuding: “*A bound for the game chromatic number of graphs*”
- G. Hernández. *Grafos: Teoría y Algoritmos*. Servicio de Publicaciones, Facultad de Informática, UPM.

Webs:

- <http://www.wikipedia.org>
- http://yalma.fime.uanl.mx/~maribel/proyecto_progra.html
- <http://docs.oracle.com/javase/6/docs/api/>
- www.scribd.com/doc/56985886/coloracion-de-grafos