Scientia et Technica Año XIV, No 40, Diciembre de 2008. Universidad Tecnológica de Pereira.

DISEÑO DE APLICACIÓN PARA MANIPULACION DE GRAFOS CON ALGORITMOS HEURISTICOS DE ANALISIS

Design of application for graph`s handling with heuristic algorithms of analysis

RESUMEN

El siguiente artículo muestra la manera de desarrollar una sencilla aplicación de entorno grafico sobre la cual se puede experimentar diversas técnicas, desde algoritmos de resolución de grafos hasta heurísticas empleadas en inteligencia artificial.

Palabras Clave: Grafos, algoritmo, heurística, aplicación, C#

CARLOS ANDRÉS LÓPEZ.
Ingeniero de Sistemas
Docente programa de Ingeniería de sistemas.
Universidad Tecnológica de Pereira

WILLIAM ARDILA URUEÑA
Msc. Instrumentación Física
Docente: Universidad Tecnológica de Pereira
Correo: williamar@utp.edu.co

Abstract

The next section shows how to develop a simple graphical application environment on which to experiment with various techniques, from algorithms resolution graph until heuristics used in artificial intelligence.

Key words: graphs, algorithms, heuristic, application, C#.

1. INTRODUCCION

Los grafos fueron creados por Leonard Euler como método para encontrar respuesta a problemas asociados con la conexión entre vías de comunicación y la posibilidad que encierran las posibles rutas cuando existen diversas alternativas.

Este mismo modelo aparece en el campo de las estructuras de datos ya que los grafos pueden considerarse como una forma de árbol eneario, en este articulo se expone la manera como a través de la programación orientada a objetos es posible diseñar modelos fácilmente implementables en un ambiente de desarrollo como C# que permite además adicionar técnicas de resolución de problemas en esta área tales como los caminos de Euler o de Hamilton.

2. MARCO TEORICO

Un grafo es una colección de puntos llamados vértices unidos por líneas llamadas aristas, donde cada arista une dos vértices. Cuando las aristas tienen dirección estamos hablando de grafos dirigidos.

Existe una variedad de ejemplos, ejercicios y problemas dentro de la teoría de grafos que requieren metodologías específicas, ejemplo de ello son [1]:

- Trayectoria de Euler: es aquella que recorre todas las aristas de un grafo conexo.
- Circuito de Euler: trayecto que recorre todas las aristas de un grafo conexo donde el punto inicial coincide con el punto final.
- Circuito Hamiltonianos: trayecto que recorre cada vértice de un grafo exactamente una vez.

Las siguientes son dos definiciones necesarias:

Grado de un vértice: cantidad de aristas que inciden en el.

Arista: segmento que une dos vértices (ver figura 1).

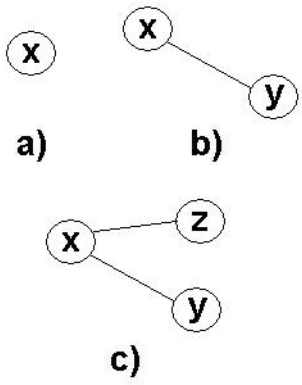

Figura 1: a) Vértice x de grado 0 (0 aristas). b) Vértices x e y de grado 1 (ambos están unidos por una arista). c) Vértices z e y de grado 1 (tocados por una arista), vértice x de grado 2 (tocado por dos aristas).

Fecha de Recepción: 13 de Septiembre de 2008.
Fecha de Aceptación: 15 de Diciembre de 2008.

Heurística: capacidad de encontrar soluciones para un problema específico empleando razonamientos similares a los empelados por los seres humanos.

Agentes inteligentes: entidades que interactúan con un ambiente conociéndolo (a través de sensores) y actuando sobre él (a través de actuadores).

Los agentes pueden dotarse de memoria, en forma de base de conocimiento, aprendizaje a través de algoritmos de diseño heurístico, y puede atribuírseles funcionalidad a través de código siendo estos los actuadores del agente [2].

3. ANALISIS DEL PROBLEMA

Se requiere aplicación que permita definir los grafos de la siguiente manera:

a) Un grafo tiene varios vértices.
b) Un grafo tiene varias aristas.

Para cumplir con este requisito diseñamos un objeto llamado Grafo el cual consta de los siguientes atributos:

Nombre: almacena el nombre del vértice.
Grado: cantidad de aristas que parten del vértice.
Lista: almacena una lista de destinos.

Tienen definido el tipo conexiones como sigue:

Arista: almacena la arista de una conexión.
Valor: almacena el valor de la conexión.
Conectado: establece si el camino hacia ese destino se encuentra activo.

Los métodos de la clase son:

Adicionar: permite adicionar una conexión a la lista de conexiones del grafo.

Existe: verifica si un vértice se encuentra en la lista de destinos.

Eliminar: permite sacar un vértice de la lista de destinos.

Conectar: permite activar la conexión de un vértice en particular de la lista de destinos.

Desconectar: permite desactivar la conexión de un vértice en particular de la lista de destinos.

GradoVertice: retorna el grado del vértice.

Para realizar una implementación se deben establecer algunas condiciones que permitan al usuario interactuar cómodamente con la aplicación, por lo que deben considerarse las siguientes especificaciones:

- Entorno gráfico.
- El usuario podrá agregar vértices.
- El usuario podrá agregar artistas.
- El usuario podrá establecer o quitar conexiones entre los vértices.

La figura 2 muestra un pantallazo de la aplicación en proceso de desarrollo que cumple con las especificaciones descritas y que implementa los algoritmos a continuación descritos y otros que se derivan de ellos.

Tal clase debe adaptarse a una aplicación que permita al usuario manipular información a través de una interfaz gráfica que pueda soportar el trazo de los grafos, y además es necesaria una clase que permita administrar la información que provea las soluciones, por eso se requiere la clase vértice y la clase agente.

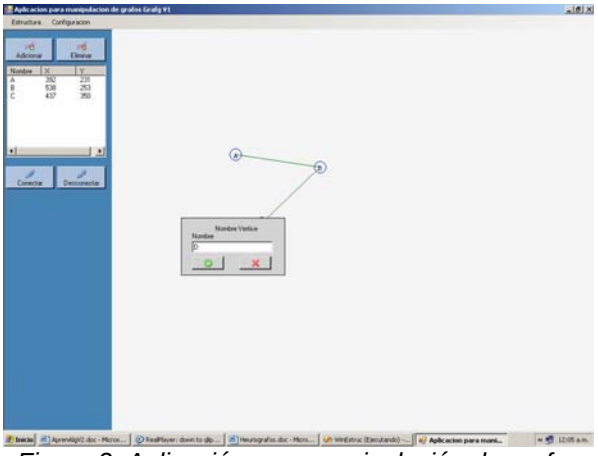

Figura 2: Aplicación para manipulación de grafos Grafg versión .1

La clase vértice hereda de la clase grafo y posee los siguientes atributos:

X: posición en la coordenada x del vértice en el área de trabajo.

Y: posición en la coordenada x del vértice en el área de trabajo.

Completo: booleano que indica si faltan aristas asociadas a el.

Disponibles: cantidad de aristas posibles.

La clase agente consta de los siguientes atributos:

Lista: listado de los vértices en el área de trabajo.

Los métodos de esta clase son:

Adicionar: adicionar vértice a la lista de vértices.

RetVertices: retorna el número de vértices del grafo.

RetAristas: retorna el número de aristas en el grafo.

CaminoEuler: encuentra, si existe, un camino euleriano en el grafo.

CircuitoEuler: encuentra, si existe, un circuito euleriano en el grafo.

CircuitoHamilton: encuentra, si existe, un circuito Hamiltoniano en el grafo.

AlmacenarGrafo: almacena las soluciones encontradas en un grafo.

BuscarCaso: busca si un grafo ha sido almacenado.

Reconstruir: toma un grafo codificado, lo reconstruye de ser posible y lo dibuja.

AlmacenarGrafo y BuscarGrafo tienen como objetivo almacenar información sobre casos de grafos resueltos, de esa forma en casos similares, por ejemplo cuando se trata de grafos isomórficos o problemas de subgrafos, se evita procesar información obtenida con anterioridad, optimizando la resolución de los problemas planteados por el usuario.

Por otro lado, esta base de conocimiento puede servir como insumo para desarrollos en donde esta información puede ser útil y que escapan al alcance de este articulo, como lo son la comparación de grafos.

El ambiente, que llamaremos de ahora en adelante la interfaz de usuario, debe permitir adicionar o retirar vértices, conectarlos entre si y desde luego activar el agente para que realice su trabajo y aprenda de los casos que se le presentan.

4. MODELO PROPUESTO

Se plantea el ambiente con la capacidad de manipular las siguientes variables:

Lista	Listav
Vertice	V
Agente	Ag
Pila	Pvertice

Donde Lista es el tipo de dato Arraylist de Visual Studio .Net, el cual se comporta como un vector dinámico que almacena objetos, Pila es una variable de tipo Stack de Visual Studio .Net la cual se comporta como una pila LIFO o FIFO, Vértice es la clase definida anteriormente la cual hereda de la clase Grafo y Agente es la clase definida anteriormente.

El usuario agrega vértices, y selecciona la resolución de un camino o circuito, el agente actúa de la siguiente manera:

a) Procesa la información sobre los vértices.
b) Codifica la información obtenida y define el caso para el grafo.
c) Toma el caso y lo busca en su base del conocimiento, si este se encuentra lo carga en la memoria y salta al paso e), de lo contrario salta al paso d).
d) Llama al método de resolución de camino o circuito correspondiente, lo almacena, después va al paso e).
e) Se muestra al usuario el ejercicio resuelto.

El paso referente a la codificación se plantea de la siguiente manera: sea un grafo con vértices y aristas entre si, las cuales solo pueden conectar un vértice a la vez (grafo simple).

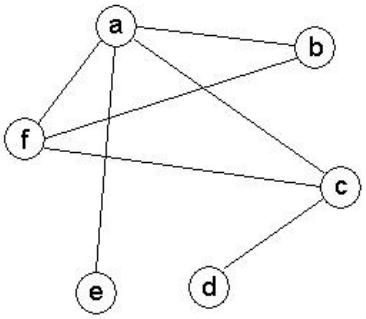

Figura 3: grafo ejemplo.

En la figura 3 se muestra un grafo con las siguientes características:

Numero de vértices: 6
Numero de aristas: 7

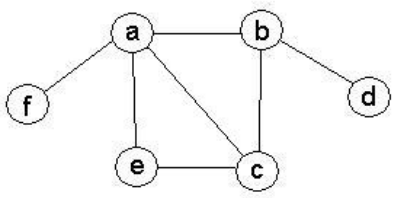

Figura 4: grafo ejemplo.

Como vemos en la figura 4 este también tiene seis vértices y siete aristas, la diferencia entre ambas es la nomenclatura entre los nombres de los vértices, en este caso se presenta lo que se llama isomorfismo, y ocurre cuando existe correspondencia entre los grados de los vértices de los grafos y pero puede diferir la configuración de la conexión de las aristas.

Por lo tanto es una forma correcta de identificar los casos de grafos de forma inequívoca si asumimos la codificación de estos con las siguientes reglas:

a) El número de caracteres de la codificación representan el número de vértices del grafo.
b) Cada número representa el grado del vértice.
c) Cada numero debe ir ordenado de mayor a menor.

Del ejemplo anterior podemos afirmar que el código para este grafo es 433211, con esta nomenclatura es posible descartar algunos grafos que son imposibles de construir, si hacemos uso de la siguiente información:

Lrema: la suma de los grados de los vértices es igual al doble de las aristas.

Del ejemplo anterior:

Sgrados = 4+3+3+2+1+1= 14

Numero de aristas = Sgrados/2 = 14/2 = 7

Puede observarse que el numero de aristas del grafo es exactamente siete, que existe un vértice que se encuentra conectado con cuatro vértices teniendo así cuatro aristas cosa que le define el grado, dos vértices conectados con tres aristas, un vértice conectado con dos aristas y dos vértices conectados con una arista.

Una forma de construir el grafo 433211 es la siguiente, (método Reconstruir):

a) Calculamos sumatoria de grado de los vértices, si es impar terminar, si no calcular aristas e ir a paso b).
b) Inicializamos los vértices con conexiones en 0 (grado 0) y disponibles iguales al grado esperado, luego los introducimos en una pila FIFO (primero en entrar, primero en salir) e ir a paso c).
c) Tomamos vértice de la pila y lo conectamos con los próximos n vértices que tengan espacio disponible, siempre que haya espacio disponible en los otros vértices y hayan aristas disponibles, de ser así ir a paso d) de lo contrario terminar y mensaje grafo no dibujable. En este ejemplo de n=4 de 7 aristas disponibles, tomamos el vértice y lo conectamos con otros cuatro vértices con espacio disponible que se encuentren en la pila y continuamos.
d) Descontamos el número de aristas que se emplearon en el paso anterior e ir a paso e). Existían disponibles 7 aristas, 7-4=3, ahora solo se dispone de tres.
e) Analizamos grado actual de los vértices y lo comparamos con grado esperado de todo el conjunto, si coinciden exactamente terminar y mensaje de grafo reconstruido, de lo contrario ir a paso f). En este caso el grado esperado del conjunto es {4,3,3,2,1,1} y el actual es {4,1,1,1,1,0}

f) Marcamos como grafo completo, ya que no se puede conectar con ningún otro y vamos a paso c).

c) Tomamos próximo vértice de la pila, verificamos aristas que le faltan para completar grado, en este caso 2, en relación con las disponible, es decir 3, verificamos espacios disponibles de los vértices al interior de la pila, teniendo en cuenta que los grados actuales son {4,3,2,2,1,0}, los que quedan disponibles son {0,0,1,0,0,1}, en este caso existe espacio disponible, ir a paso d).

d) Descontamos el número de aristas que se emplearon en paso anterior e ir a paso e). Existían disponibles 3 aristas, 3-2=1, ahora solo se dispone de una, ir a paso e).

e)Grado actual {4,3,2,2,1,0} contra esperado {4,3,3,2,1,1} ir a paso f).

f) marcamos como grafo completo el segundo vértice y vamos a c).

c) Tomamos vértice cuyo grado actual es 2, el necesario es 3, como existen aristas disponibles suficientes, verificamos que exista espacio en el vértice y en otro dentro de la pila, para el tercer vértice hay uno disponible y el próximo disponible se encuentra en el vértice de la ultima posición, por lo tanto se pueden conectar, ir a paso d).

d) Descontamos el número de aristas que se emplearon en paso anterior e ir a paso e). En este caso 1-1=0.

e) Analizamos grado actual de los vértices {4,3,3,2,1,1} con esperado {4,3,3,2,1,1} y vemos que coinciden, por lo tanto terminar y grafo reconstruido.

El algoritmo de reconstrucción solo se emplea para casos de grafos en donde no se conoce la configuración de los vértices con sus respectivas aristas, esta información es almacenada en la base del conocimiento como la relación vértice grafo que es la que finalmente se almacena. En los casos donde no es posible construir el grafo se almacena la información de casos no posibles para evitar perdida de recursos de procesamiento.

Debe mencionarse que los conjuntos de caso actual y caso esperado requieren vectores de tipo entero y que para llenarlos deben emplearse los métodos creados para retornar la información de grado del GradoVertice.

5. ALGORITMOS DE RESOLUCION

Para resolver problemas de caminos de Euler, circuitos de Euler y circuitos de Hamilton es necesario implementar algoritmos como el de Fleury, con las estructuras de control que posee el diseño descrito esto puede hacerse sin problema, a continuación se muestra un ejemplo de resolución de de caminos de Euler a nivel de algoritmo general.

El método CaminoEuler contiene los siguientes pasos:

a) Decodificar grafo e ir a b).
b) Verificar si existen grados impares, si existen ir a c) de lo contrario ir a e)
c) Si más de dos vértices poseen grado impar entonces terminar y mensaje de no es posible encontrar camino, de lo contrario ir a paso d).
d) Crear Pila LIFO donde el primer y el último vértice son los impares luego ir a f).
e) Crear Pila LIFO con los vértices luego ir a f).
f) Extraer elemento de la pila, e ir a g).
g) Si existen elementos en la pila conectarlo con el siguiente elemento de la pila e ir a f) de lo contrario terminar.

Analicemos el caso 433211.
a) Decodificar grafo, resulta el conjunto {4,3,3,2,1,1} e ir a b).
b) Número de vértices con grado impar: 2, ir a paso c).
c) Existen más de 2 vértices impares?. No, entonces ir a paso d).
d) Crear pila LIFO primer y ultimo elementos son vértices de grado impar: {1,4,3,3,2,1}. Ir a f).
f) Extraer elemento de la pila, vértice de grado 1 e ir a próximo. Pila {4,3,3,2,1}
g) Como existen más elementos en la pila se conecta vértice actual (de grado 1) con el primeros de la pila (vértice de grado 4) y se conectan. Pila actual {4,3,3,2,1}, ir a paso f).
f) Extraer elemento de la pila, vértice de grado 4 e ir a próximo. Pila {3,3,2,1}
g) Conectar con primer elemento de la pila (vértice de grado 3) e ir a paso f). Pila actual {3,3,2,1}.
f) Extraer elemento de la pila, vértice de grado 3 e ir a próximo. Pila {3,2,1}
g) Conectar con primer elemento de la pila (vértice de grado 3) e ir a paso f). Pila actual {3,2,1}.
f) Extraer elemento de la pila, vértice de grado 3 e ir a próximo. Pila {2,1}
g) Conectar con primer elemento de la pila (vértice de grado 2) e ir a paso f). Pila actual {2,1}.
f) Extraer elemento de la pila, vértice de grado 2 e ir a próximo. Pila {1}
g) Conectar con primer elemento de la pila (vértice de grado 1) e ir a paso f). Pila actual {1}.
f) Extraer elemento de la pila, vértice de grado 1 e ir a próximo. Pila {}
g) No existen elementos en la pila entonces terminar.

6. CONCLUSIONES

Los grafos son herramientas importantes para analizar problemas de alta complejidad como los son aquellos relacionados con la búsqueda de rutas.

Es importante explorar soluciones informáticas que permitan incorporar alternativas que encuentren soluciones basadas en criterios lógicos ya que este tipo de soluciones son preferibles a aquellas que requieren el empleo de fuerza bruta, término que se aplica a los procedimientos informáticos que encuentran soluciones expandiendo árboles de posibilidades donde el costo de procesamiento y la viabilidad de encontrar la solución dependen de la capacidad de la máquina para realizar el algoritmo.

Las bases de conocimiento son información valiosa equiparable a la experiencia adquirida por un ser humano, la posibilidad de retroalimentar otros sistemas con información hallada por otro durante un proceso aprendizaje posibilita que estos comiencen su propia experiencia del mundo sin rehacer tareas y por lo tanto centrándose en un problema nuevo y particular.

Los algoritmos aquí expuestos son base importante para la incorporación de nuevas soluciones asociadas al problema de grafos.

Los algoritmos propuestos están en capacidad de resolver problemas de construcción de grafos los cuales son un área de estudio de esta disciplina.

Los algoritmos podrían ajustarse sin cambiar la estructura de la aplicación, para resolver problemas de comparación de grafos, siendo este otro caso de estudio del campo de los grafos.

La información recolectada por el agente permitirá a versiones mejoradas resolver circuitos de Euler y de Hamilton.

BIBLIOGRAFIA

[1] Teoría de grafos. Manuel Moreno Valencia. Bogota. Editorial Universidad Inca de Colombia. 2004.

[2] Inteligencia Artificial, un enfoque moderno. Stuart Russell. Peter Norvig. Prentice-Hall