BENÉMERITA UNIVERSIDAD AUTÓNOMA DE PUEBLA

FACULTAD DE CIENCIAS DE LA COMPUTACIÓN

"EVALUADOR DE UNA RED DE AFILIACIÓN"

T E S I S

Para obtener el título de:

LICENCIADO EN CIENCIAS DE LA COMPUTACIÒN

PRESENTA:

Alejandro Francisco Martínez Montes

Asesor:

M. C. Pedro Bello López

Puebla Pue. Junio 2016

Agradecimiento

A dios por haberme dado la oportunidad de cumplir con este pasó más de mi vida

A mi familia por su respaldo para ver cumplida esta meta.

Al M. C. Pedro Bello López por su apoyo para que este proyecto pudiera llevarse a cabo.

A la Red PRODEP: Algoritmos combinatorios y reconocimiento de patrones, por el apoyo económico brindado para el desarrollo de esta tesis.

Contenido

Introducción 4

Capítulo 1. Marco conceptual 6

1.1 Redes de Afiliación 6

1.2 Grafos bipartitos 8

1.3 Conceptos básicos 10

1.4 Grafos bipartitos 12

1.4.1 Recorrido a profundidad de un grafo 13

1.4.2 Algoritmo del recorrido a lo profundo 15

1. 5 Herramientas de desarrollo 16

Capítulo 2. Desarrollo del algoritmo 20

2.1 Entrada 23

2.2 Proceso 24

2.3 Salida 26

Capítulo 3. Implementación y pruebas del sistema 28

3.1. Inicio del sistema 28

3.2. Evaluación de una red de afiliación 29

Conclusiones y trabajo futuro 34

Bibliografía 36

Anexo 1. Código principal del sistema 37

Introducción

En el estudio y análisis de redes, aplicado a asuntos variados, desde hace décadas ha sido las tradiciones dominantes en las ciencias sociales, construir los modelos de explicación científica, bien a partir de modelos causales, que generalmente se han aplicado a estructuras macro-sociales o económicas, en los años sesentas se consolidó el esfuerzo por aplicar la formalización matemática a instituciones previamente expresada con metáforas desde entonces la teoría de grafos (Harary y Norman, 1953) se han convertido en el sustrato formalizado para el desarrollo de análisis de redes, el cual es un método, un conjunto de instrumentos para conectar el mundo de los actores (individuos, organizaciones etc.).

Cuando se representan las relaciones de los actores con determinados hechos y, a través de éstos, la relación entre los actores. A esta relación subordinada se le denomina filiación y a este tipo concreto de redes se conocen como redes de afiliación, que representa por tanto, la asociación de un conjunto de actores con un conjunto de acontecimientos.

En este proyecto se aplicara un método computacional para analizar una clase especial de redes sociales bimodales que representan la afiliación de un conjunto de actores con un conjunto de acontecimientos. Las redes de afiliación describen colecciones de actores en lugar de simplemente lazos entre pares de actores, es decir, las redes de afiliación son relacionales en tres modos: primero, muestran cómo están relacionados entre ellos los actores y los acontecimientos; segundo, los acontecimientos crean lazos entre los actores; y en tercer lugar, los actores crean lazos entre los acontecimientos.

Objetivo general:

Diseñar e implementar un algoritmo que dados los datos de entrada de un conjunto de actores y acontecimientos (grafo) determinar si es un grafo bipartito y por tanto se trata de una red de afiliación. Para determinar si el grafo es bipartito se obtendrá un recorrido a lo profundo, se generaran los ciclos fundamentales y si estos ciclos están formados con un número par de aristas, entonces el grafo es bipartito.

Objetivos Particulares:

- Determinar las estructura de datos a utilizar.
- Crear el recorrido a lo profundo de un grafo y obtener los ciclos fundamentales.
- Determinar si los ciclos fundamentales están formados por un nuevo par de aristas, entonces es bipartito.
- Diseñar el algoritmo completo de toda la aplicación.
- Implementar la aplicación en un lenguaje de alto nivel.
- Realizar un conjunto de pruebas con grafos de diferente configuración (estructura) para determinar si forman una red de afiliación.

Capítulo 1. Marco conceptual

La forma más directa de estudiar una estructura social es analizar los patrones de vínculos que ligan a sus miembros. El análisis de redes busca las estructuras profundas, La red es una construcción relacional, en el cual las descripciones se basan en los conceptos de vínculos (ligas), que unen actores (nodos) que pueden ser personas, grupos, organizaciones o personas un sistema social.

Dada la importancia que se tiene de las redes de filiación en el entorno académico y social, tomando como base fundamental para su estudio y aplicación el uso de grafos, que juegan un papel importante en el manejo y distribución de la información para determinar cuándo un grafo es o será bipartito y una red de afiliación podrá entonces representarse como un grafo bipartito mostrando a los actores y acontecimientos como nodos lo que darán que existan g actores y h acontecimientos y g+h nodos en el grafo bipartito [1].

1.1 Redes de Afiliación

Las redes de afiliación contienen información sobre colección de actores.

Los elementos básicos que definen una red son esencialmente dos, los actores que establecen las relaciones entre sí, y estas relaciones; los primeros son representados por puntos en la red o nodos y los segundos por líneas. si los actores se describen como nodos y sus relaciones como líneas entre pares de nodos, el concepto de red social pasa de ser una metáfora a una herramienta operativa analítica qué utiliza el lenguaje matemático de la teoría de grafos, de la matrices y del álgebra relacional se pueden construir

múltiples tipos de redes, las más se corresponden con redes de modo-uno es decir aquellas en las que todos los actores pertenecen a un único conjunto; un caso particular de redes de modo-dos son las denominadas redes de afiliación.

La representación más sencilla de una red de afiliación es una matriz que registre la afiliación de cada actor a cada acontecimiento, esta matriz, a la cual llamaremos matriz de afiliación, codifica para cada actor, los acontecimientos a los cuales cada actor está afiliado, de igual modo registra, para cada acontecimiento los actores afiliados a él por lo tanto la matriz es una sociomatriz bimodal en la cual las filas indexan a los actores y las columnas indexan a los acontecimientos. Dado que existen g actores y h acontecimientos, la matriz es g x h y se pondrá un 1 en las celdas (i,j)-ésimas si el actor de la fila i está afiliado con el acontecimiento de la columna j, y un 0 si el actor de la fila i no está afiliado al acontecimiento j,

En la tabla 1.1 se muestra un ejemplo de una matriz de filiación para 6 niños y 3 fiestas donde el actor es el niño y los acontecimientos son las fiestas así es como podemos ver que Antonio asistió a la fiesta 1 y a la fiesta 3, pero no a la fiesta 2.

	Acontecimientos		
Actor	Fiesta 1	Fiesta 2	Fiesta 3
Antonio	1	0	1
Pedro	0	1	0
María	0	1	1
Martha	0	0	1
Luis	1	1	1
Juan	1	1	0

Tabla 1.1 Matriz de afiliación

1.2 Grafos bipartitos

La teoría de grafos juega un papel importante en la fundamentación matemática de las ciencias de la computación. Los grafos constituyen una herramienta básica para modelar fenómenos discretos y son fundamentales para la comprensión de las estructuras de datos y análisis de algoritmos.

Historia: [2] tiene su origen en 1736 cuando Leonhard Euler publico el primer trabajo de lo que hoy llamamos teoría de grafos, en ese trabajo Euler desarrolla una teoría que resuelve el famoso problema de los puentes en Konigsberg.

El problema de los puentes, el río Pregel fluye a través de la ciudad de Konigsberg en Prusia (actualmente Kalinigrado, Rusia) rodeando una isla en el centro de la ciudad para luego bifurcarse en dos cursos. La ciudad quedaba por tanto dividida en cuatro partes, dos regiones a orilla del río Pregel, la isla de Knciphof y la región que quedaba entre ambos brazos del río, Siete puentes conectaban entre si estas regiones (ahora no pues la ciudad fue parcialmente destruida durante la segunda guerra mundial) ver figura 1.1.

Figura 1.1 Puente de Konigsberg.

En el siglo XVIII se hizo popular, como pasatiempo, averiguar si era posible cruzar los siete puentes de la ciudad pasando una sola vez por cada uno de ellos partiendo y llegando al mismo sitio. para tratar de resolver el problema. Euler represento esquemáticamente las áreas de tierra por puntos y los puentes por líneas que conectaban esos puntos. El resultado represento el primer ejemplo de grafos.

Utilizando propiedades básicas de grafos. euler dedujo que la respuesta a este problema es negativa un grafo es intuitivamente, un conjunto de puntos, llamados vértices, junto con una colección de líneas, llamadas aristas, cada una de las cuales une, o un par de puntos, o un punto consigo mismo.

Definición 1. Un grafo es un par G=(V,E) donde V es un conjunto finito no vacío cuyos elementos se llaman vértices o nodos, y E es un conjunto cuyos elementos se llaman aristas o ejes [5].

- Si las aristas son "pares no ordenados" de vertices de V entonces diremos que el grafo G es no dirigidos

$$G=(V,W) \text{ o } (W,V)$$

- Si las aristas son "pares ordenados" de vértices de V, Entonces diremos que el grafo G es dirigido.

$$G=(V,W) \neq (W,V)$$

1.3 Conceptos básicos

En matemáticas y en las ciencias de la computación [7] la teoría de grafos estudia las propiedades de los grafos, que son colecciones de objetos llamados vértices (o Nodos) conectados por líneas llamadas aristas (o arcos) que pueden tener orientación (dirección asignada), Típicamente, un grafo está diseñado por una serie de puntos (los vértices), conectado por líneas (las aristas).

Un grafo consiste de dos conjuntos finitos, un conjunto no vacío V(G) de vértices y un conjunto de aristas E(G) donde cada arista está asociada a un conjunto compuesto por uno o dos vértices llamados puntos extremos [8].

Para poder trabajar con grafos en una computadora hemos de representar en forma matricial. a todo grafo se le puede asociar una matriz cuadrada booleana M, donde las filas van representar el vértice inicial y las columnas el vértice final, de modo que en cada uno se colocara 0 o 1 [9].

Grafos Ponderados: Se presentan los pesos de cada arista y se puede determinar la longitud de una ruta que es la suma de todos los pesos de las aristas como se ve muestra en la figura 1.2

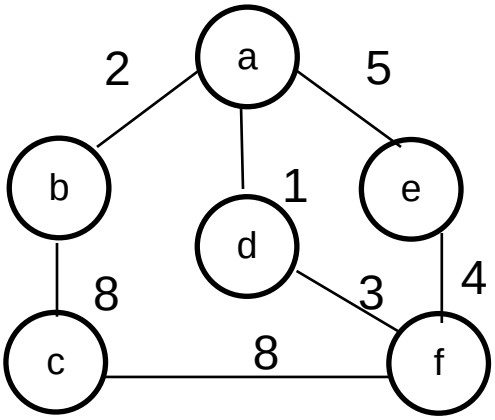

Figura 1.2 con sus pesos definidos en cada vértice

Grafos simples: no contienen ni lazos, ni aristas paralelas, ni aristas dirigidas, Figura 1.3.

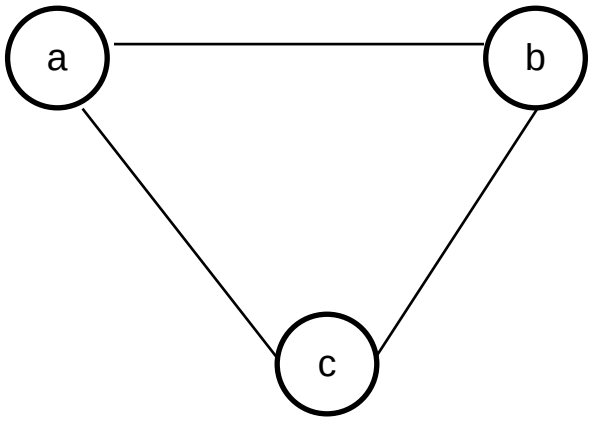

Figura 1.3 Grafo sin lazos y sin aristas paralelas ni dirigidas

Grafos completos: es aquel que tiene n vértices que tienen una arista entre cada par de vértices, no puede haber aristas paralelas. Figura 1.4

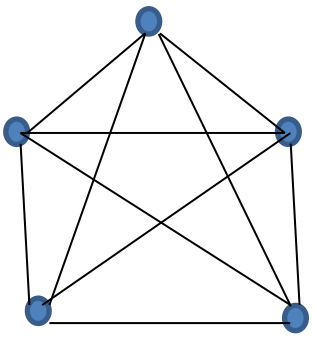

Figura 1.4 Grafo Completo

Grafos de similitud: Son aquellos en los que se puede derivar subgrafos.

Figura 1.5

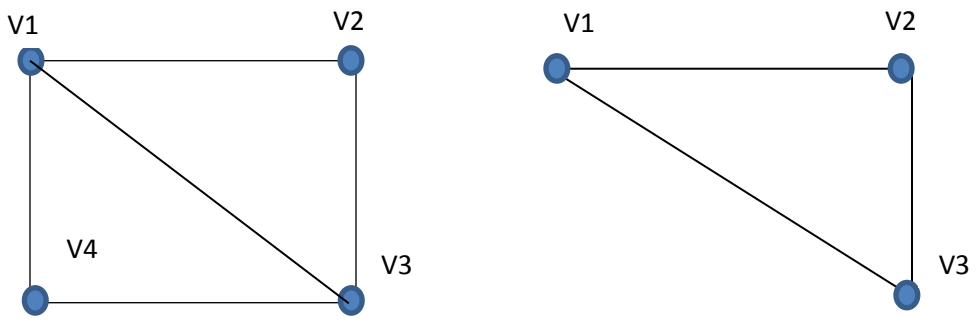

Figura 1.5 Grafo del lado izquierdo y un subgrafo derivado

Grado de un vértice: Comprende el número de aristas que inciden a ese vértice

1.4 Grafos bipartitos

Otra clase de grafos que se presenta con frecuencia es la de los Grafos Bipartidos. Figura 1.6

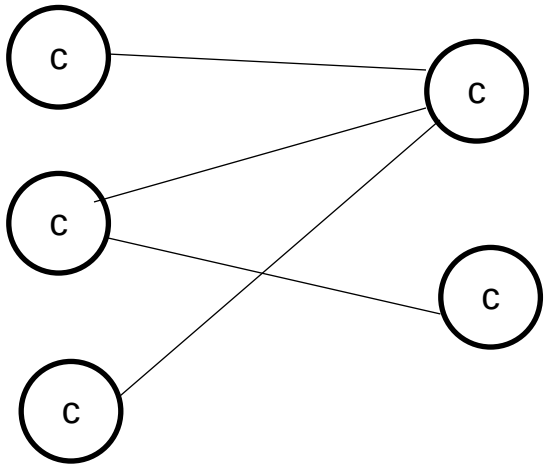

Figura 1.6 Grafo Bipartito

Un Grafo bipartito se denomina al grafo cuyos vértices se pueden separar en dos subconjuntos disjuntos V1(G) y V2(G) y las líneas siempre unen vértices de un subconjunto con vértices de otro subconjunto. Además se tiene que:

V1(G) U V2(G) = V(G).

La intersección de V1(G) y V2(G) es vacío.

1.4.1 Recorrido a profundidad de un grafo

La búsqueda o profundidad de un grafo recorre los nodos de un grafo localizando los recorridos posibles y en el caso de no poder continuar, vuelve al punto donde existe nuevo recorrido posible con el fin de visitar todos los nodos ver figura 1.7.

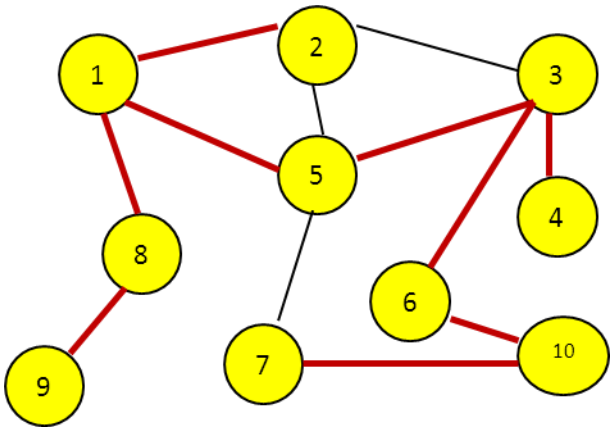

Figura 1.7 Recorrido a lo profundo para el nodo 2 (2-1-5-3-4-6-10-7-8-9)

Generando su matriz de adyacencia asignado un 1 en caso de que haya una afiliación entre nodos y 0 en caso contrario. Generando una matriz de adyacencia (tabla 1.2) la cual es simétrica.

	1	2	3	4	5	6	7	8	9	10
1	0	1	0	0	1	0	0	1	0	0
2	1	0	1	0	1	0	0	0	0	0
3	0	1	0	1	1	1	0	0	0	0
4	0	0	1	0	0	0	0	0	0	0
5	1	1	1	0	0	0	1	0	0	0
6	0	0	1	0	0	0	0	0	0	1
7	0	0	0	0	1	0	0	0	0	1
8	1	0	0	0	0	0	0	0	1	0
9	0	0	0	0	0	0	0	1	0	0
10	0	0	0	0	0	1	1	0	0	0

Tabla 1.2 Matriz de adyacencia

En la figura 1.2 se marca el recorrido a lo profundo donde las aristas que no fueron tocadas, representan ciclos, así del nodo 2 al nodo 3 hay un clico, del nodo 2 al nodo 5 existe otro ciclo y del nodo 5 al nodo 7 hay otro ciclo.

En la tabla 1.3 se muestra la tabla de adyacencias de la figura 1.2 donde se marca con “2” las aristas visitadas en el recorrido a lo profundo, de donde las aristas que quedaron con “1” representan los ciclos fundamentales.

	1	2	3	4	5	6	7	8	9	10
1	0	2	0	0	2	0	0	2	0	0
2	2	0	1	0	1	0	0	0	0	0
3	0	1	0	2	2	2	0	0	0	0
4	0	0	2	0	0	0	0	0	0	0
5	2	1	2	0	0	0	1	0	0	0
6	0	0	2	0	0	0	0	0	0	2
7	0	0	0	0	1	0	0	0	0	2
8	2	0	0	0	0	0	0	0	2	0
9	0	0	0	0	0	0	0	2	0	0
10	0	0	0	0	0	2	2	0	0	0

Tabla 1.3 Matriz de adyacencia marcada con las aristas visitadas

Gráficamente veríamos el recorrido a lo profundo en la figura 1.8

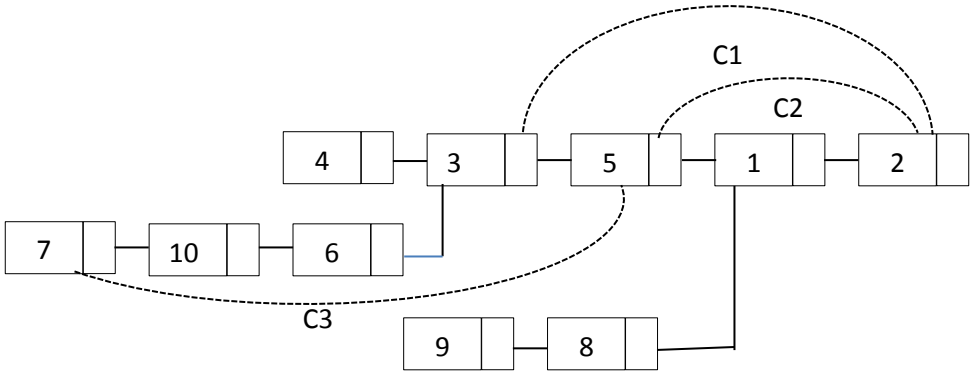

Figura 1.8 Recorrido a lo profundo para el nodo 2 (2-1-5-3-4-6-10-7-8-9)

1.4.2 Algoritmo del recorrido a lo profundo

Algoritmo del recorrido a lo profundo [10]

- Inicializar todos los nodos del grafo en el estado **en espera**.
- Para cada nodo del grafo:
    - Si el estado es **en espera** entonces:
        - Insertar el nodo en una pila (cambiar su estado a **listo**)
            - Mientras la pila de nodos listos no esté vacía:
                - Sacar un nodo de la pila y procesarlo.
                - Cambiar su estado a **Procesado.**
                - Meter en la pila todos los nodos vecinos del nodo

recién sacado cuyo estado sea **en espera** o **listo**.

Cambiar el estado de estos nodos a **listo**.

- Si el nodo ya existe en la pila (con estado de **listo**) se deja sólo el que llegó al último (el más reciente).

1. 5 Herramientas de desarrollo

Para la realización de este proyecto se utilizó el lenguaje de programación PHP Hypertext Pre-processor, el cual fue desarrollado puntualmente para diseñar páginas web dinámicas programando scripts del lado del servidor. El lenguaje PHP siempre va incrustado dentro del HTML.

Historia

Originalmente diseñado por el programador danés-canadiense Rasmus Lerdorf, en el año 1994 en base a la escritura de un grupo de CGI binarios escritos en el lenguaje C. En un comienzo, PHP sólo estaba compuesto por algunas macros que permitían trabajar más fácilmente en la creación de páginas web.

En el año de 1995 Rasmus Lerdorf le añadió el analizador sintáctico y se llamó PHP/F1 Versión 2, sólo reconocía texto HTML y algunas directivas de mSQL. Después de esta fecha la contribución al código fue pública.

Por qué utilizar PHP como lenguaje de programación para este sistema, una de las características por ser un lenguaje gratuito y multiplataforma. Además de su posibilidad de acceso a muchos tipos de bases de datos, también es importante destacar su capacidad de crear páginas dinámicas, así como la posibilidad de separar el diseño del contenido de una web.

PHP era la solución para la construcción de Webs con independencia de la Base de Datos y del servidor Web, válido para cualquier plataforma.

Velocidad: PHP no solo es rápido al ser ejecutado sino que no genera retrasos en la máquina, por esto no requiere grandes recursos del sistema. PHP se integra muy bien junto a otras aplicaciones, especialmente bajo ambientes Unix.

Estabilidad: PHP utiliza su propio sistema de administración de recursos y posee, un sofisticado método de manejo de variables, conformando un sistema robusto y estable.

La similitud con lenguajes C y C++ podrán utilizar PHP rápidamente. Además PHP dispone de una amplia gama de librerías, y permite la posibilidad de agregarle extensiones. Esto le permite su aplicación en múltiples áreas, tales como encriptado, gráficos, XML y otras.

Permite la interacción con gran cantidad de motores de bases de datos tales como MySQL, MS SQL, Oracle, etc.

PHP es Open Source, (Código abierto) esto significa que no depende de ninguna compañía comercial y que no requiere de licencias. Librería grafica[1] utilizada.

A demás por su versatilidad para trabajar con la programación Orientada a Objetos, [12] es una metodología de programación avanzada y bastante extendida, en la que los sistemas se modelan creando clases, que son un

[1] http://code.tutsplus.com/tutorials/an-introduction-to-the-raphael-js-library--net-7186

conjunto de datos y funcionalidades. Las clases son definiciones, a partir de las que se crean objetos. Los objetos son ejemplares de una clase determinada y como tal, disponen de los datos y funcionalidades definidos en la clase.

Conceptos de programación orientada a objetos

Clase

Representa características, atributos y métodos de un conjunto de objetos del mismo tipo.

Atributo

Son las propiedades que describen a los objetos pueden ser datos.

Método

Son las acciones las operaciones, los procedimientos y funciones los cuales los tenemos que aplicar a los atributos de los objetos que ocupamos.

Objeto

Como ya sabemos un objeto es la instancia de una clase donde los objetos pueden contener atributos y métodos que manejan la solución al problema que queremos resolver.

En la programación orientada a objetos utilizamos tres aspectos muy importantes

Abstracción

La abstracción se refiere a acciones que solo ocuparemos aunque nuestro programa tenga muchas funciones solo se utilizan las necesarias.

Herencia

Esto se refiere a que un objeto hereda las características de la clase padre que es de la que proviene aunque una clase de igual forma puede heredar características de otra clase.

La herencia define la relación entre clases “es un”, donde una subclase hereda de una o más superclases.

Polimorfismo

En el polimorfismo nos referimos que muchas veces una operacion tiene el mismo nombre pero tienen diferentes clases y funciones siendo distintas en cada una de ellas.

Encapsulamiento

Esto se refiere a que varios objetos al programarlos ocultan sus funcionalidades internas al exterior ya que reduce errores que pueden ocurrir cuando estamos ejecutando nuestros programas. La ventaja es que puede tener acceso a varias acciones son mostrar sus detalles.

Otro aspecto que tenemos que saber es que cada mensaje está asociado a un método ya que cuando un objeto recibe un mensaje la respuesta es una ejecución de un método.

Para controlar el acceso a los miembros de una clase utilizaremos tres tipos diferentes

- public
- private
- protected.

Capítulo 2. Desarrollo del algoritmo

En este proyecto aplicaremos un método computacional para analizar una clase especial de redes sociales bimodales que representan la afiliación de un conjunto de actores con un conjunto de acontecimientos. Las redes de afiliación describen colecciones de actores en lugar de simplemente lazos entre pares de actores, es decir, las redes de afiliación son relacionales en tres modos: primero, muestran cómo están relacionados entre ellos los actores y los acontecimientos; segundo, los acontecimientos crean lazos entre los actores; y en tercer lugar, los actores crean lazos entre los acontecimientos.

	Acontecimiento		
Investigadores	*Congreso 1*	*Congreso 2*	*Congreso 3*
Fred	1	0	1
Andrew	0	1	0
John	1	1	1
Eliot	0	1	1
Ross	1	0	0

Tabla 2.1.Matriz de una red de afiliación

En la tabla 2.1 se muestra un ejemplo de la representación de una red de afiliación de 5 actores (investigadores) y 3 acontecimientos (congresos), donde cada aij = 1 si el actor i está afiliado al acontecimiento j. Además se pueden obtener algunos elementos que caracterizan a la red de afiliación como la sociomatriz donde se relacionan los actores y los acontecimientos todos juntos en una matriz extendida, así también se puede obtener el

hipergrafo que corresponde a mapear con conjuntos quienes van al congreso 1, quienes al congreso 2 y quienes al congreso 3, según el ejemplo de la tabla 2.1.

El método computacional para evaluar si una red de afiliación está bien formada es utilizando la teoría de grafos y en particular por medio de un grafo bipartito. Como ya se ha definido anterior mente Un grafo bipartito es un grafo en el cual los nodos pueden dividirse en dos subconjuntos, y todas las líneas están entre pares de nodos pertenecientes a diferentes subconjuntos. Una red de afiliación puede representarse por un grafo bipartito (figura 2.1) donde los actores y los acontecimientos son nodos, y asignando a los actores a un subconjunto de nodos y los acontecimientos al otro subconjunto.

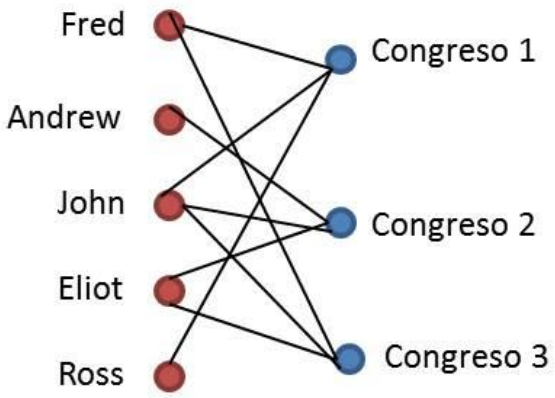

Figura 2.1. Grafo bipartito de actores y acontecimientos de la tabla 2.1

Para determinar si un grafo es bipartito utilizaremos el siguiente:

Teorema 1. Sea G conexo, G es bipartito si y sólo si no tiene ciclos de longitud impar [12].

Por lo que en este proyecto se desarrollara un algoritmo para determinar si el grafo es bipartito o no y en caso de que sea bipartito se obtendrá la su sociomatriz correspondiente (tabla 2.2) y el hipergrafo (figura 2.2).

						Acontecimiento		
	Fred	*Andrew*	*John*	*Eliot*	*Ross*	*Congreso 1*	*Congreso 2*	*Congreso 3*
Fred	0	0	0	0	0	1	0	1
Andrew	0	0	0	0	0	0	1	0
John	0	0	0	0	0	1	1	1
Eliot	0	0	0	0	0	0	1	1
Ross	0	0	0	0	0	1	0	0
Congreso 1	1	0	1	0	1	0	0	0
Congreso 2	0	1	1	1	0	0	0	0
Congreso 3	1	0	1	1	0	0	0	0

Tabla 2.2.Sociomatriz correspondiente al grafo bipartito de la red de afiliación

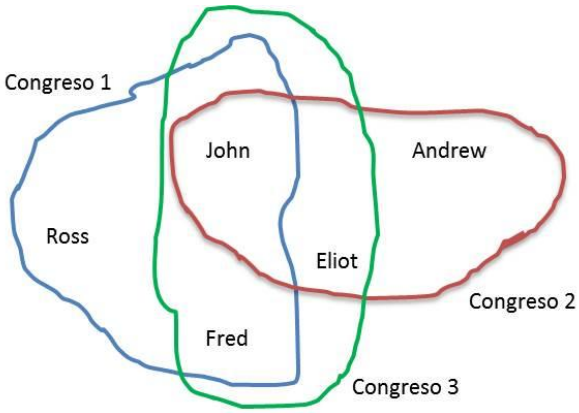

Figura 2,2. Hipergrafo generado a partir de los datos de la tabla 2.1

2.1 Entrada

La propuesta algorítmica (figura 2.3) para determinar si un grafo es bipartito se basa en el Teorema 1. Cuyos pasos básicos son:

- Leer el grafo de entrada
- Obtener el recorrido a lo profundo a partir de cualquier nodo
- Generar el árbol de recorrido a lo profundo
- Obtener los ciclos fundamentales del grafo
- Determinar las aristas de retroceso para los ciclos
- Verificar si los ciclos están formados por un número impar de aristas. Si existe entonces el grafo no es bipartito, en caso contrario si es biparito (ver ejemplo figura 2.3) y se podrán obtener la sociomatriz, el hipergrafo y el hipergrafo dual.

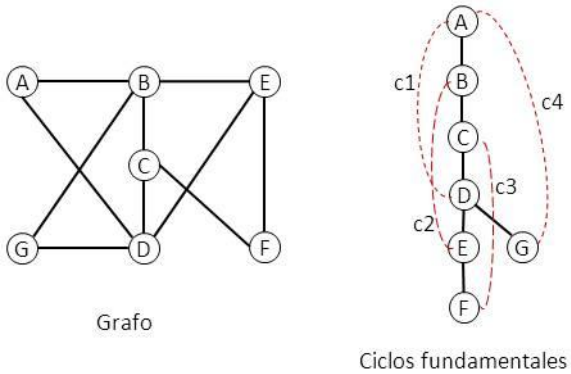

Figura 2.3. De lado izquierdo se tiene un grafo y en la parte derecha el recorrido a lo profundo y sus ciclos fundamentales, como son de longitud par, entonces el grafo si es bipartito.

El estudio de soluciones para grafos con ciertas características constituye un tópico de investigación y aplicación en ciertas áreas como lo son en redes computacionales en donde puede variar la topología y un algoritmo especifico llegará a ser más eficiente, sin embargo la necesidad de tener una solución general es igualmente útil para cuando se desconoce la estructura.

El desarrollo del algoritmo para el sistema mencionado consiste en dado un archivo de entrada de forma matricial con los nodos y las aristas de la red o grafo, se le aplica el recorrido a lo profundo ordenando y transformando en una lista ligada, identificando los ciclos fundamentales, y evaluándolos para determinar su longitud y si esta es de longitud par entonces el grafo es bipartito y forma una red de afiliación.
Para ello se tienen las variables usadas con su determinado tipo de datos.

2.2 Proceso

Una aplicación computacional con un algoritmo que reciba como entrada un grafo cualquiera y determine si es bipartito o no y en caso de que si sea, se tendrá una red de afiliación y se mostraran sus elementos: sociomatriz, hipergrafo e hipergrafo dual.

Como primer paso se escribirá una matriz de entrada, la cual se escribirá en formato modo texto, y tendrá los siguientes elementos, En el primer renglón corresponde al número de nodos, y los renglones a partir del renglón 2, son las aristas de (grafo) figura 2.4.

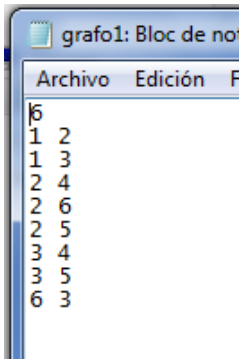

Figura 2.4 Datos de la matriz de entrada

Al momento de ejecutar el programa, leerá el archivo y una vez mediante un recorrido a lo profundo transformándolo en una lista ligada donde se identifican los ciclos fundamentales, se evalúan los ciclos si todos son de longitud par, entonces el grafo es bipartito y por lo tanto forma una red de afiliación Figura 2.5.

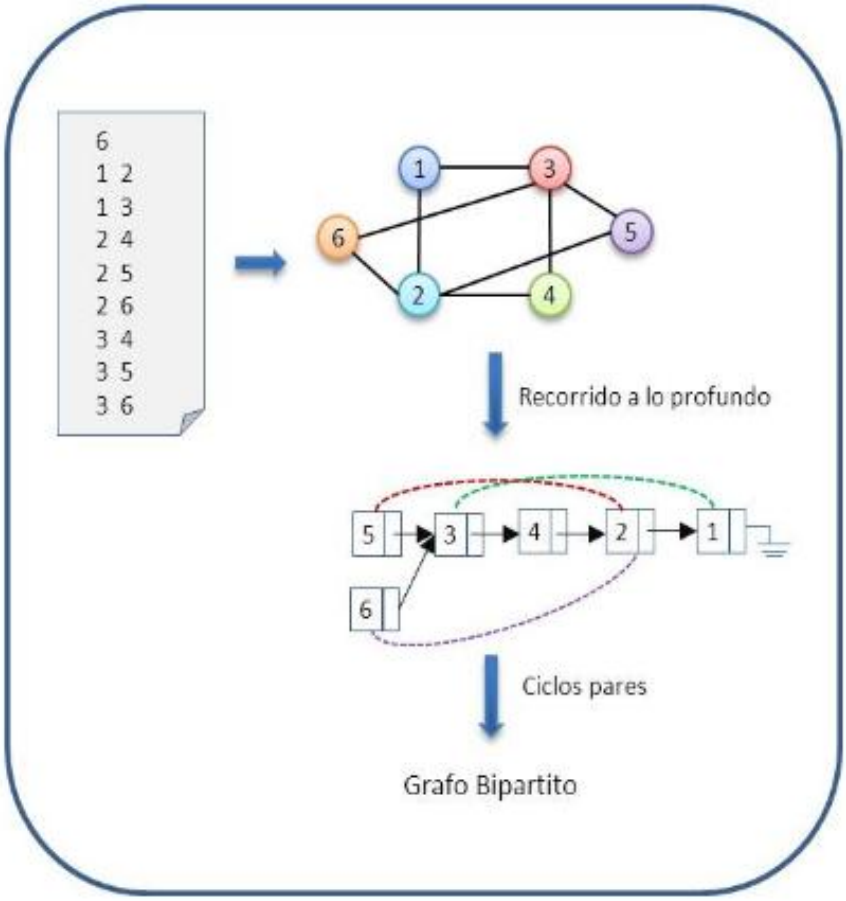

Figura 2.5 Flujo del modelo propuesto

La estructura de datos utilizada incluye una pila, una lista ligada, la clase para dibujar el grafo en javaescript y la clase principal denominado grafo, la clase pila se utiliza para el recorrido a lo profundo y la clase lista para

transformar el grafo en un árbol y determinar los ciclos fundamentales y su longitud. Como se muestra en la figura 2.6.

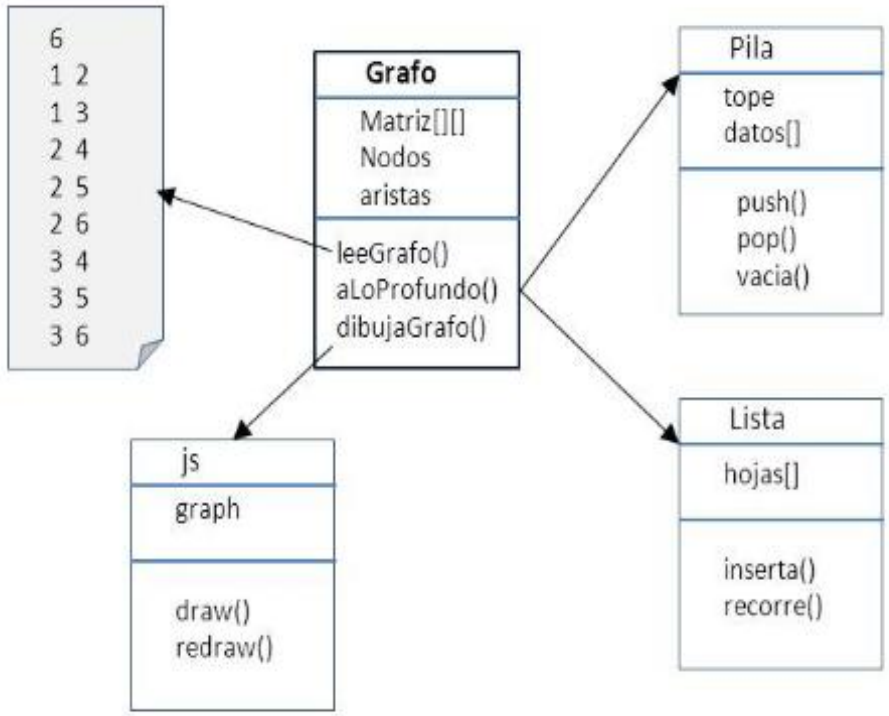

Figura 2.6. Relación de las clases utilizadas en el desarrollo del sistema

2.3 Salida

El resultado de salida obtenida será una red de afiliación con sus elementos en forma gráfica ver figura 2.6.

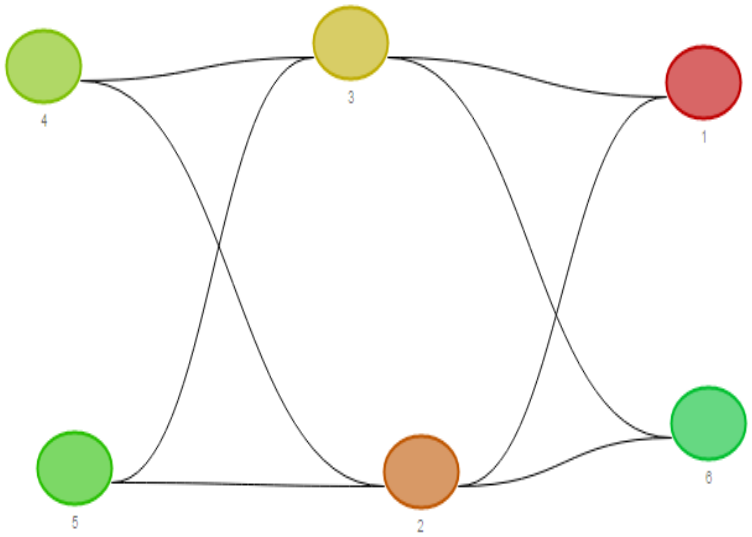

Figura 2.6. Salida del sistema de n grafo bipartito

Capítulo 3. Implementación y pruebas del sistema

En este capítulo se presenta un prototipo en php de la aplicación desarrollada para determinar si la red de afiliación está bien definida o no, es decir dado un grafo se determina si es bipartito o no.

3.1. Inicio del sistema

A continuación se describe en general como se presenta el sistema que permite evaluar una red de afiliación.

Figura 3.1. Pantalla inicial del sistema

En la figura 3.1 se presenta la pantalla inicial del sistema donde se describe brevemente que es una red de afiliación, en esta pantalla se tienen las opciones:

- Inicio: es la pantalla inicial del sistema.
- Evaluar: evalúa la red de afiliación para determinar si está bien formada o no dicha red.
- Modelo: Aquí se muestra el modelo utilizado en el sistema.
- Acerca de: muestra los participantes del proyecto.

3.2. Evaluación de una red de afiliación

Las pantallas siguientes muestran el funcionamiento del sistema al evaluar un grafo que representa la red de afiliación.

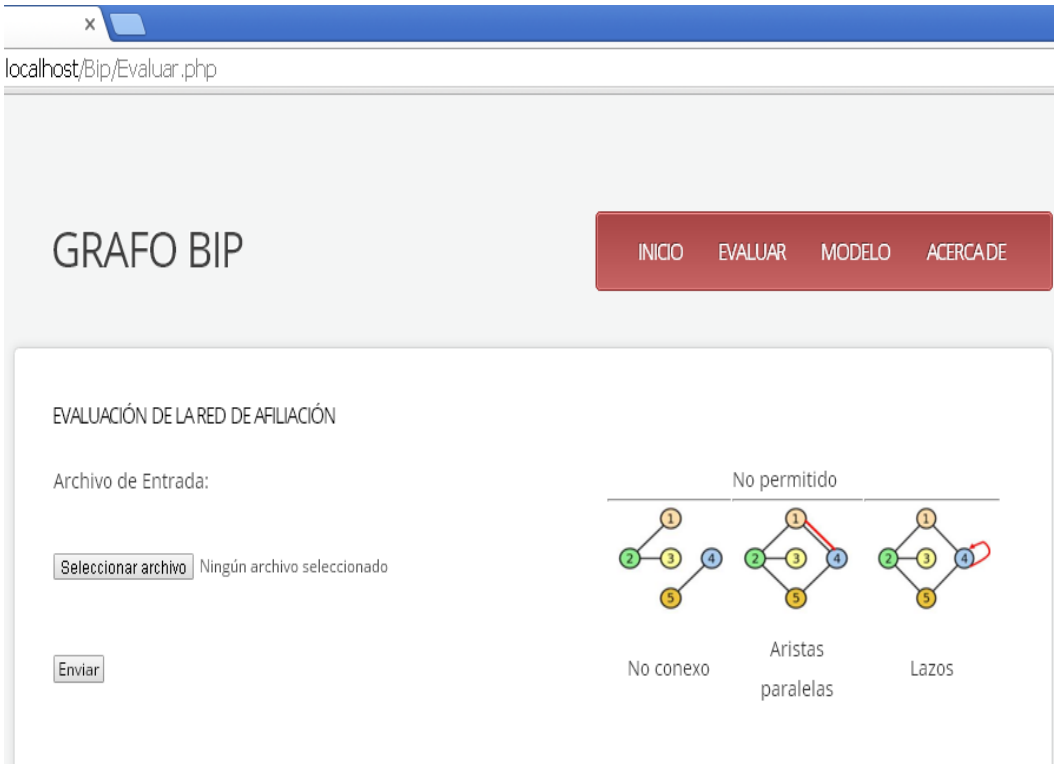

Figura 3.2. Pantalla de evaluación de la red de afiliación

En la figura 3.2 se muestra la pantalla para evaluar la red e afiliación, donde se solicita seleccionar el archivo de entrada donde se tiene almacenada la red a evaluar, además se indica que para evaluar correctamente la red es necesario que el grafo que representa la red de afiliación sea conexo, no contenga aristas paralelas ni lazos.

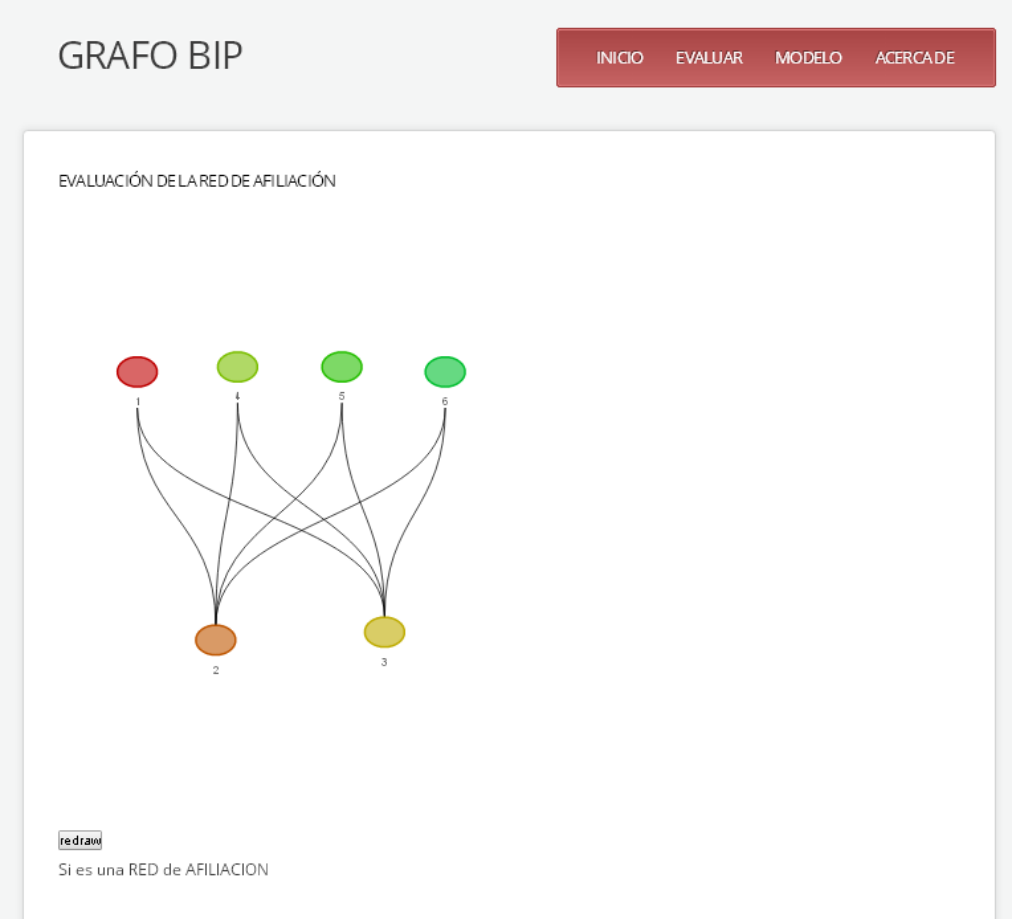

Figura 3.3. Resultado de la evaluación de la red

En la figura 3.3 se muestra una red de afiliación que fue evaluada con el sistema donde nos indica que si es una red de afiliación, en esta figura los nodos de la parte superior denotan los acontecimientos y los nodos de la parte inferior los recursos o eventos, aunque también se pueden considerar en sentido opuesto. El sistema permite redibujar el grafo con la opción “redraw” de forma automática, o utilizando el mouse de forma manual.

En la figura 3.4 se muestra otro ejemplo evaluado con el sistema donde el resultado es que la red de afiliación no está bien formada debido a que el

grafo que la representa no es bipartito, en la figura podemos observar por ejemplo que el nodo 5 no puede ser parte de los acontecimientos ya que es adyacente al 1 o parte de los recursos ya que es adyacente al 2.

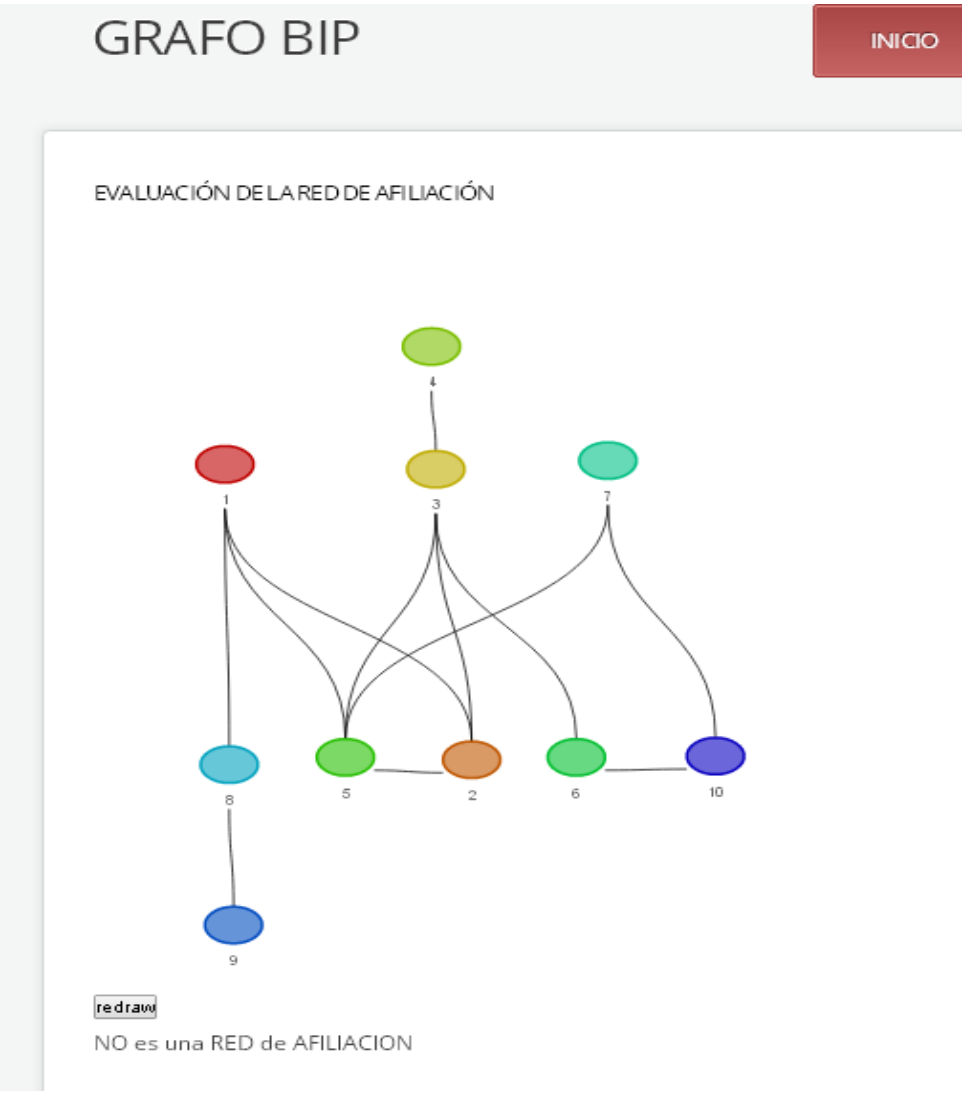

Figura 3.4. Ejemplo donde el grafo no representa una red de afiliación (no es bipartito)

En la figura 3.5 se muestra un ejemplo con 40 nodos y 58 aristas, este grafo fue evaluado con el sistema dando como resultado que no forma una red de afiliación, es decir no es un grafo bipartito.

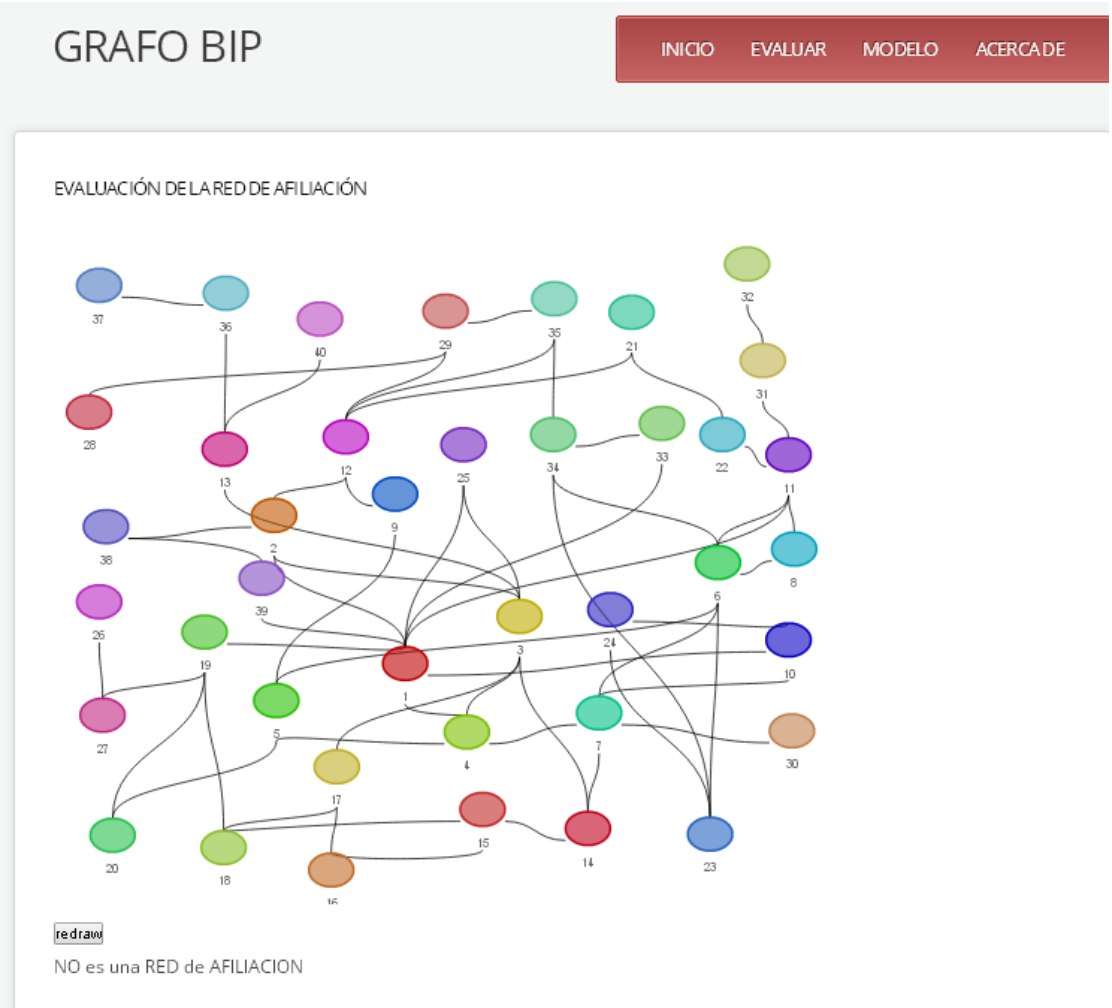

Figura 3.5. Ejemplo de grafo con 40 nodos y 58 aristas

Conclusiones y trabajo futuro

Como resultado del presente trabajo se obtuvo:

- En la investigación sobre grafos se han presentado algoritmos que resuelven ciertos problemas sobre diversas áreas de la ciencia y aquí se presenta un algoritmo para tratar un tipo especial de redes sociales como lo es una red de afiliación, en general mostramos como la teoría de grafos permite modelar este tipo de problemas de la vida real. En particular se desarrolló un trabajo donde se modela una red de afiliación utilizando la teoría de grafos. Se creó un algoritmo que permite evaluar si la red de afiliación está bien formada, es decir si corresponde a un grafo bipartito.

- Se creó una aplicación computacional que permite en general evaluar si un grafo es bipartito o no, es decir para el caso de que sea bipartito significa que se tiene una red de afiliación.

- De igual forma la aplicación desarrollada puede ser usada como ayuda didáctica en materias de matemáticas discretas, análisis y diseño de algoritmos, estructuras de datos y complejidad de algoritmos.

- Se utilizó la tecnología de php, javascript y la programación orientada a objetos para ejemplificar el algoritmo desarrollado.

Trabajo a futuro:

En el desarrollo de este trabajo se sugiere considerar algunas mejoras y perspectivas como las siguientes:

- Como la aplicación desarrollada está en PHP, se puede montar en un servidor lo que proveerá el acceso masivo de los alumnos para conocer el trabajo elaborado y realizar pruebas.

- En el trabajo solo se consideró solo modelar grafos con un número reducido de nodos aproximadamente 100 nodos y 200 aristas, debido al tiempo y recursos que necesita el sistema para su ejecución, por lo que resulta conveniente mejorar el código para modelar grafos de mayor tamaño.

- El análisis de la complejidad del algoritmo queda como trabajo posterior ya que no fue considerado dentro de los objetivos planteados.

Bibliografía

[1] Stanley Wasserman, Katherine Faust, Análisis de redes métodos y aplicaciones, Primera edición, ESPAÑA 2003.

[2] Ana María Vieites Rodríguez, Teoría de Grafos, Ejercicios resueltos y propuestos. Segunda Edición. Ediciones Paraninfo

[3] Luis Sanz Menéndez Análisis de redes sociales: o cómo representar las estructuras sociales subyacentes Julio 2003 Madrid España.

[4] KATHERINE FAUST Análisis de redes sociales Métodos y aplicaciones primera edición Madrid España.

[5] Ramón Espinoza Armenta, Matemáticas discretas
Primera Edición, grupo editor AlfaOmega.

[6] Jonhsonbaugh, Richard. Matemáticas Discretas sexta edición Pearson Educación, México 2005

[7] Jiménez José A, Matemáticas para la computación Primera Edición Alfaomega, segunda Edición, México

[8] Susanna S. Epp Matemáticas discretas con aplicaciones, Cuarta edición México.

[9] J. Luis García Valle, Matemáticas Especiales para computación

[10] Román Martínez, Elda Quiroga. Estructuras de datos: referencia práctica con orientación a objetos, Alfaomega, México 2002.

[11] http://www.ma.uva.es/~antonio/Industriales/Apuntes_05-06/LabM/A_Grafos2006-2(2).pdf, consultado 15 de Mayo de 2015.

[12] http://www.desarrolloweb.com/articulos/1540.php consultado 11 de Mayo 2015

Anexo 1. Código principal del sistema

A continuación se exponen las clases principales utilizadas en el sistema desarrollado.

Programa Principal para evaluar la red de afiliación

```
<?PHP
  require_once("Grafo.php");  // Inluye la clase Grafo
  $G=new Grafo(); // Crea un objeto G de la clase grafo
  $nom=$_FILES['archivo']['name'];
  $G->leeGrafo($nom); // Lee el archivo de entrada
  echo '<div id="canvas"></div>';
  echo '<button id="redraw" onClick="redraw();">redraw</button>';
  $G->dibujo(); // Dibuja el grafo
  $G->Aloprofundo(1); //Evalua la red(grafo) con el recorrido a lo profundo
?>
```

Clase Grafo

```
<?PHP
class Grafo
{
 private $NN; // numero de nodos
 private $NA; // numero de aristas
 private $M; // mmatriz de adyacencias
 public $NI; // nodos iniciales
 public $NF; // nodos finales

 public function leeGrafo($arch)
 {
  $fp = fopen($arch,"r");
  $this->NN = fgets($fp);
//   echo "Numero de Nodos: $this->NN <br>";

  for($i=0;$i<=$this->NN;$i++)
    for($j=0;$j<=$this->NN;$j++)
          $this->M[$i][$j]=0;
  $k=0;
  while(!feof($fp))
  {
          $linea = fgets($fp);
          $t1=strtok($linea,' ');
          $t2=strtok(' ');
          $this->NI[$k]=$t1;
          $this->NF[$k]=$t2;
          //        escribe los datos del archivo
  // echo "$t1 $t2 $k<br>";
```

```
            $k=intval($k);
            $k=$k+1;
            $x=intval($t1);
            $y=intval($t2);
            $this->M[$x][$y]=1;
            $this->M[$y][$x]=1;
    }
    fclose($fp);
            $this->NA=$k;
 }


public function dibujo()
 {
 //echo "numero de aristas= $this->NA <br>";
echo "<script>";
echo "var redraw, g, renderer;";

echo "window.onload = function() {

    var width = 700;
    var height = 600;

    g = new Graph();";

for($i=0; $i<$this->NA; $i++)
{
$x=intval($this->NI[($i)]);
$y=intval($this->NF[($i)]);
echo "g.addEdge('$x', '$y');";
}

   echo " var layouter = new Graph.Layout.Spring(g);
    renderer = new Graph.Renderer.Raphael('canvas', g, width, height);

    redraw = function() {
        layouter.layout();
        renderer.draw();
    };

};
</script>";
}


public function Aloprofundo($inicio)
 {
    require_once("pila.php");
    require_once("lista.php");
    $P=new pila(5000);
    $PP=new pila(5000);
```

```
  $L=new lista();
  $nodos=array();
  $contador=0;
  $nd=$this->NN;
  for($j=0;$j<=$nd;$j++) $nodos[$j]=0;//marcamos nodos como no visitados

  $P->push($inicio);//metemos el primer dato a la pila
  $PP->push(0);
  $ancestro=0;
  while($contador<$this->NN)
  {
    $n=$P->pop(); //sacamos un dato de la pila
    $na=$PP->pop();
    if($nodos[$n]==0){
                        //echo "Dato: $n $na<br>";
                        $L->inserta($n,$na,$L->r); //creacion del arbol
                 $this->M[$n][$na]=2;
                            $this->M[$na][$n]=2;
                        $contador++;
                        $nodos[$n]=1; // marcamos nodo visitado

                 for($k=$nd; $k>=1;$k=$k-1)
                  {
                    //echo "K=$k";
                    $x=intval($k);
                   if($this->M[$n][$x]==1)
                    {
                     //  echo "ADyacente=$k ";
                      if($nodos[$x]==0)
                      {
                       $P->push($x);
                                        $PP->push($n);
                      }
                    }
                  }

                        $ancestro=$n;
                      }
   }
   $L->fin();

      $res=$this->ciclos($L);
      if($res==0) echo "<br>Si es una RED de AFILIACION <br>";
      if($res==1) echo "<br>NO es una RED de AFILIACION <br>";
 }

public function ciclos($L)
{

  for($i=1;$i<=$this->NN; $i++)
  {
```

```
            for ($j=$i;$j<=$this->NN;$j++)
        {
          if($this->M[$i][$j]==1)
            {
            // echo "ciclo $i con $j <br>";
             $x=$this->tipoCiclo($i,$j,$L);
             if ($x==1) return 1;
            }
        }
   }
   return 0;
 }

 public function tipoCiclo($ini,$fin,$L)
 {
  for($i=1;$i<=$L->r;$i++)
  {
   $aux=$L->rama[$i];
   $con=0;
   $band=0;
   $pos1=0;
   $pos2=0;
   $tam=0;
   while($aux!=null && $band==0)
   {
      $con=$con+1;
      if($aux->info==$ini) $pos1=$con;
      if($aux->info==$fin) $pos2=$con;
      $aux=$aux->sig;
   }
   if($pos1>0 && $pos2>0)
    {
      $tam=abs($pos1-$pos2);
    //  echo " Contador = $con pos1=$pos1 pos2=$pos2 Tamaño = $tam<br>";
      $tam=$tam+1;
      if($tam%2==0) return 0;
      else return 1;
    }
   }
 }

}
?>
```

Clase Pila

```
<?PHP
class pila
{
```

```
    private $tope;
    private $tam;
    private $P=array();

            public function __construct($t)
            {
                        $this->tam=$t;
                        $this->tope=0;
            }

            public function push($dato)
            {
                        $this->P[$this->tope]=$dato;
                        $this->tope++;
            }

            public function pop()
            {
                        $this->tope--;
                        $n=$this->P[$this->tope];
                        return $n;
            }


            public function vacia()
            {
                        if($this->tope==0) return 1;
                        else return 0;
            }

            public function llena()
            {
                        if($this->tope==$this->tam) return 1;
                        else return 0;
            }

        public function muestra()
        {
          for($i=0; $i<$this->tope; $i++)
          echo $this->P[$i]." ";
        }
}
?>
```

Clase Lista Ligada

```
<?PHP
class nodo
{
  public $info;
```

```
  public $sig;
  public function __construct($x)
  {
   $this->info=$x;
   $this->sig=null;
  }
}

class lista
{

    private $raiz;
    public $r;
    public $rama=array();

            public function __construct()
            {
                        $this->raiz=null;
               $this->r=1;
               $this->rama[0]=$this->raiz;
            }

            public function inserta($dato, $padre, $r)
            {
              $temp=new nodo($dato);

         if($this->raiz==null) $this->raiz=$temp;
         else {
               $aux=$this->raiz;
           //     echo "$aux->info - <br>";
               while($aux!=null && $aux->info!=$padre)
                {
             //      echo "$aux->info <br>";
                  $aux=$aux->sig;
                }

              // echo "PAdre $aux->info - <br>";
             if($this->raiz->info!=$padre)
               {
                 $x=$this->r;
                 $this->rama[$x]=$this->raiz;
                // echo "<br> Nueva Rama: $x Hoja:";
                 //echo $this->rama[$x]->info." <br>";
                 $this->r=$this->r+1;
               }
               $temp->sig=$aux;
               $this->raiz=$temp;

             }
        }
```

```
        public function recorre($x)
        {
          $aux=$this->rama[$x];

          while($aux!=null)
          {
            echo $aux->info." ";
            $aux=$aux->sig;
          }
         echo "<br>";
        }

        public function fin()
        {
          $x=$this->r;
       //   echo "fin: $x";
          $this->rama[$x]=$this->raiz;
        }
}
?>
```