Universidad Autónoma Metropolitana
Unidad Azcapotzalco

División de Ciencias Básicas e Ingeniería

Proyecto Tecnológico

Licenciatura en Ingeniería en Computación

El algoritmo dual geométrico de Goemans-Williamson para acoplamiento en gráficas completas

Jeaneth Santos Platero 209331361

Asesores

Laura Elena Chávez Lomelí, Profesor Asociado,
Departamento de Ciencias Básicas.

Risto Fermin Rangel Kuoppa, Profesor Titular, Departamento de Sistemas.

Trimestre 2015 Invierno

22 de abril de 2015

Yo, Laura Elena Chávez Lomelí, declaro que aprobé el contenido del presente Reporte de Proyecto de Integración y doy mi autorización para su publicación en la Biblioteca Digital, así como en el Repositorio Institucional de UAM Azcapotzalco.

Laura E. Chávez L.

Yo, Risto Fermin Rangel Kuoppa, declaro que aprobé el contenido del presente Reporte de Proyecto de Integración y doy mi autorización para su publicación en la Biblioteca Digital, así como en el Repositorio Institucional de UAM Azcapotzalco.

Yo, Jeaneth Santos Platero, doy mi autorización a la Coordinación de Servicios de Información de la Universidad Autónoma Metropolitana, Unidad Azcapotzalco, para publicar el presente documento en la Biblioteca Digital, así como en el Repositorio Institucional de UAM Azcapotzalco.

Índice

ÍNDICE DE FIGURAS

Introducción

Una clase importante de problemas de optimización combinatoria se presentan en los problemas de acoplamiento.

La palabra acoplamiento significa literalmente formar parejas. En los problemas de acoplamiento en gráficas, la idea más general consiste en encontrar subconjuntos de aristas que cumplen ciertos criterios de optimización, donde los parámetros involucrados pueden ser el número de aristas en la solución, el número de vértices que son extremos del total de aristas o el peso asociado a las aristas.

Los problemas de acoplamiento se estudiaron por mucho tiempo llamando la atención de muchos investigadores interesados por sus múltiples aplicaciones tales como: sistema de representantes distintos, problema de cubrimiento, descomposición en cadenas, cuadrados latinos, problema del matrimonio, calendarización de vuelos, enrole de locomotoras, etc. y sus diversas variantes (Osuna, 1998).

Este trabajo estudia el problema de acoplamiento de cardinalidad máxima con costo mínimo en gráficas completas.

Objetivos

I. Extender el estudio del problema del acoplamiento máximo en gráficas bipartitas al problema de acoplamiento máximo a costo mínimo en gráficas completas.

II. Implementar el algoritmo de Goemans-Williamson para resolver el problema de acoplamiento máximo a costo mínimo considerando distancia euclidiana y distancia manhattan.

III. Diseñar y programar la interfaz gráfica para ilustrar el progreso del algoritmo.

Justificación

Los problemas de acoplamiento son de mucho interés en áreas como el Análisis Combinatorio o la Investigación de Operaciones, por sus múltiples aplicaciones.

El propósito de la implementación y diseño de la interfaz para el algoritmo de Goemans-Williamson es que forme parte del material de apoyo para las UEAs que abordan el tema de acoplamientos tales como: Investigación de Operaciones y Combinatoria, ayudando a los alumnos a entender de una manera visual el desempeño del algoritmo lo cual contribuirá al fomento del aprendizaje.

Marco Teórico

Problema de Acoplamiento de Cardinalidad Máxima

Dada una gráfica G con un número finito de vértices y aristas, se desea determinar el acoplamiento de mayor cardinalidad (número de aristas) posible. A este problema también se le conoce como acoplamiento máximo.

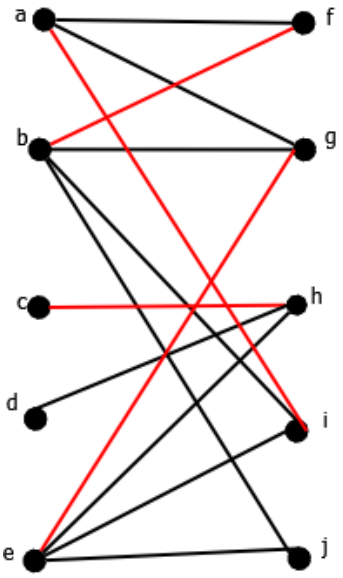

Figura 1.- Acoplamiento de cardinalidad máxima

La grafica de la figura 1 tiene un acoplamiento máximo, cubre todos los vértices con el máximo número de aristas posibles. Existen diversos problemas que pueden ser planteados de manera equivalente al problema de acoplamiento de cardinalidad máxima por ejemplo el problema de sistema de representantes distintos.

Problema de acoplamiento de peso mínimo

En este problema se tiene una gráfica G donde cada arista tiene asociado un peso. Lo que se busca es hallar un acoplamiento en donde la suma de los pesos de sus aristas sea mínimo.

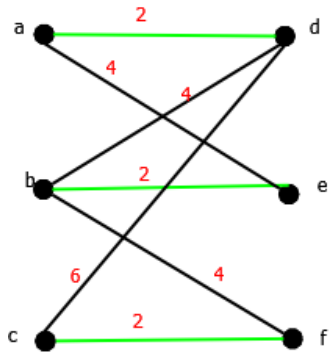

Figura 2.- Acoplamiento de peso mínimo

La gráfica de la figura 2 tiene un acoplamiento de peso mínimo, se cubren todos los vértices y el peso de las aristas en el acoplamiento es mínimo. Una de las aplicaciones del problema de acoplamiento con peso se le conoce como problema de asignación.

Problema de acoplamiento máximo y peso mínimo

El problema de acoplamiento máximo de peso mínimo es un problema dual, busca que el número de aristas en el acoplamiento sea máximo (acoplamiento de cardinalidad máxima) y que el peso del acoplamiento sea mínimo (acoplamiento de peso mínimo).

Tanto para el problema de acoplamiento de cardinalidad máxima como el de peso mínimo se han encontrado varios algoritmos de solución. El algoritmo más conocido para resolver problemas de acoplamientos es el algoritmo Húngaro (Kuhn, 1955).

El algoritmo Húngaro es intuitivo y de los más sencillos, pero solo funciona para el caso bipartito. Cabe mencionar que es ilustrativo de principios centrales de la solución conocida para el problema de acoplamiento máximo en gráficas no bipartitas que se resuelve vía el algoritmo de capullos de Edmons y Karp (Edmonds & Karp, 1972), sin embargo una condición de este algoritmo es que la gráfica debe de ser bipartita.

Una gráfica bipartita se define como:

Definición 1.- *Una gráfica G = (V, A) de dice bipartita si el conjunto de vértices V puede separarse en dos conjuntos disjuntos V = S ∪ T de tal manera que las aristas de la gráfica unen siempre un vértice de S con uno de T (es decir, no hay aristas entre los vértices de S ni entre los vértices de T)* (Willson, 1996).

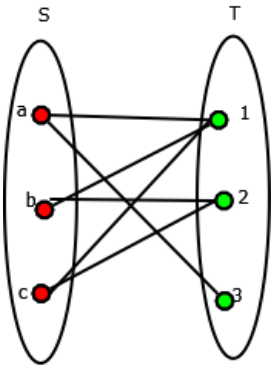

Figura 4.- Gráfica Bipartita

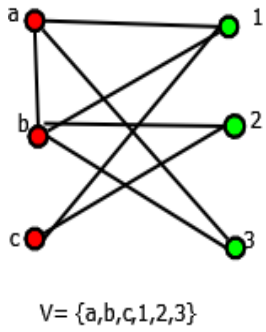

Figura 3.- Gráfica no Bipartita

Este trabajo estudia un algoritmo de acoplamiento máximo de peso mínimo para en gráficas completas.

Una gráfica completa se define como:

Definición 2.-*Se dice que una gráfica es completa cuando todos sus vértices son adyacentes a todos los vértices de la gráfica, es decir, cuando cada par de vértices son los extremos de una arista. La gráfica completa con n vértices es denotada por Kn. Kn tiene exactamente* $\frac{1}{2}n(n-1)$ *aristas* (Willson, 1996).

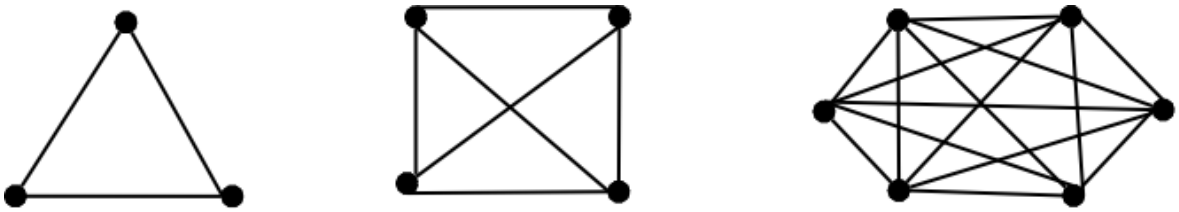

Figura 5.- Graficas Completas K3, K4 y K5

El algoritmo de Goemans-Williamson es un algoritmo de aproximación, que presenta una solución al problema de acoplamiento máximo a costo mínimo en gráficas completas.

Algoritmo de Goemans-Williamson

Este algoritmo trabaja bajo ciertas condiciones específicas:

- Gráficas completas
- Los pesos en las aristas satisfacen la desigualdad del triangulo

Definición 3.- *Desigualdad del triángulo: En todo triángulo la suma de las longitudes de dos lados cualesquiera es siempre mayor a la longitud del lado restante* (Weisstein, 2013).

Para asegurar que los pesos en las aristas satisficieran la desigualdad del triángulo, el valor asociado al peso de cada arista está dado por la distancia euclidiana o por la distancia manhattan entre sus vértices extremos.

Definición 4.-*La distancia euclidiana entre dos puntos P1 y P2, de coordenadas cartesianas (x1, y1) y (x2, y2) es la longitud del camino más corto entre ambos* (Arguello, 2005). *La cual está dada por la siguiente expresión:*

$$d(P1, P2) = \sqrt{(x2 - x1)^2 + (y2 - y1)^2}$$

Definición 5.- *La distancia manhattan entre dos puntos P1 y P2, de coordenadas cartesianas (x1, y1) y (x2, y2) está dada por:*

$$d(P1, P2) = |x1 - y1| + |x2 - y2|$$

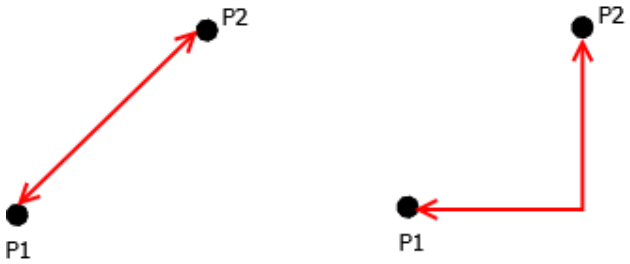

Figura 6.- Representación de la distancia euclidiana y la distancia manhattan

El algoritmo de Goemans –Williamson (William J. Cook, 1998) utiliza la idea de zonas de control para ir construyendo el acoplamiento en una gráfica. Estas zonas de control son representadas por círculos (en el caso de la distancia euclidiana) y cuadrados (en el caso de la distancia manhattan) cuyos centros son los vértices de la gráfica. El tamaño de las zonas está dado por la distancia más pequeña entre vértices. Si ***dm*** es la distancia más pequeña de un vértice a otro, se dibuja un círculo o un cuadrado de tamaño dm (diámetro del circulo o lado del cuadrado) para cada vértice de la gráfica. Cuando se unen dos zonas (círculos o cuadrados) los vértices a los que pertenecen estos son candidatos a vértices del acoplamiento y la arista que los une entra a ser parte del acoplamiento de la gráfica.

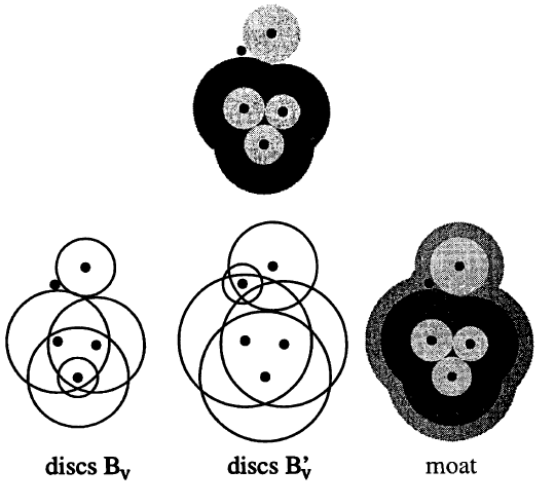

Figura 7.- Construcción de las zonas de control (William J. Cook, 1998)

Ejemplo de ejecución del algoritmo

En las siguientes imágenes se muestra el comportamiento del algoritmo de Goemans-Williamson, tomando en cuenta la distancia manhattan y la distancia euclidiana, para una gráfica K4. La figura 8 muestra una gráfica K4 con las respectivas distancias entre sus vértices.

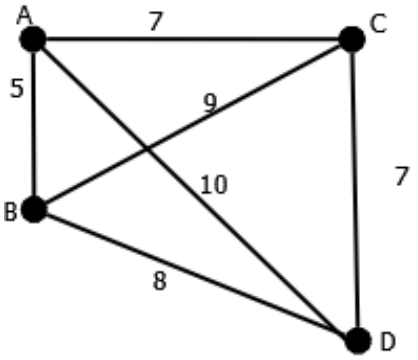

Figura 8.- Representación de K4

Se toma la mínima distancia entre vértices, 5 en este caso, y se construyen círculos o cuadrados de tamaño 5 (figura 9).

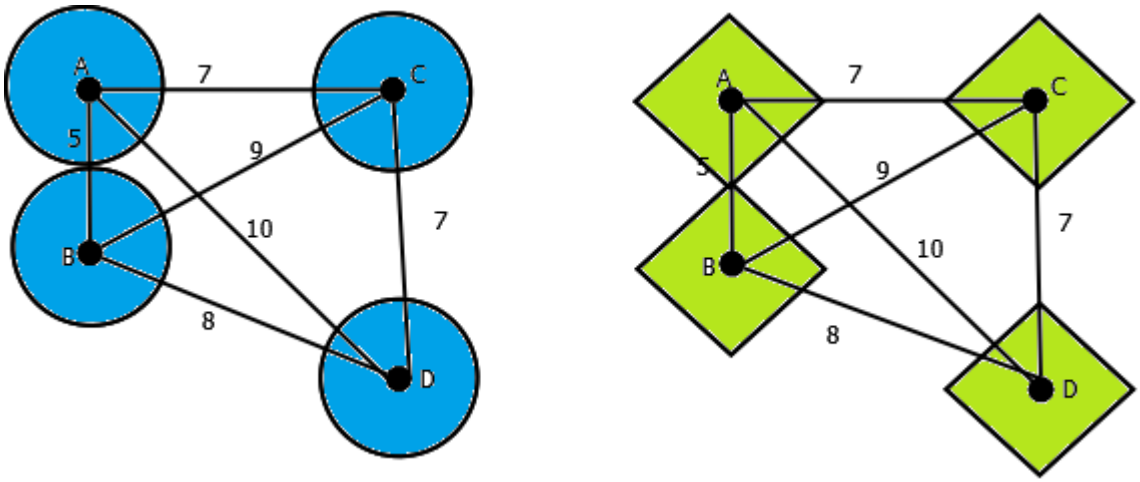

Figura 9.- Etapa 1 de la construcción del acoplamiento

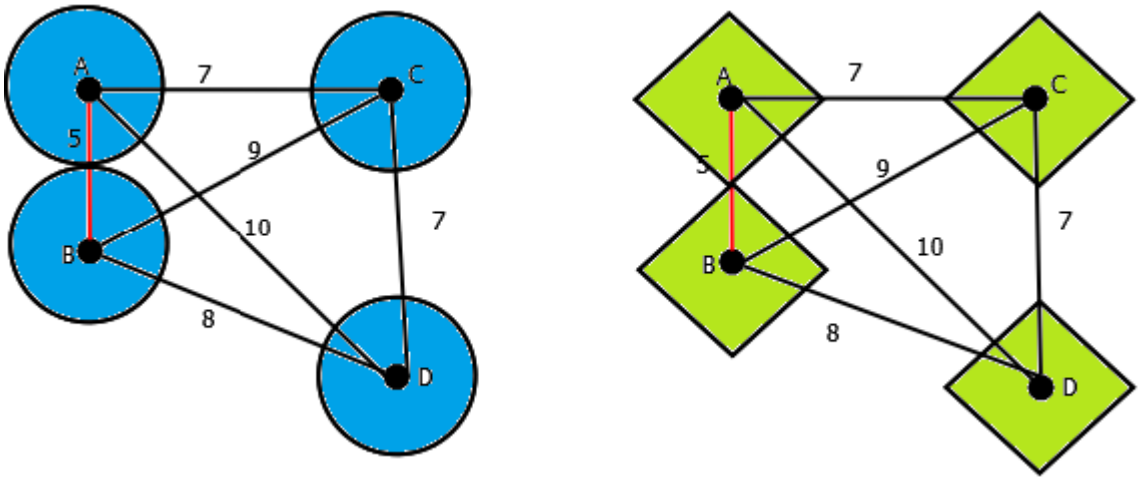

Figura 10.- Etapa 2 de la construcción del acoplamiento

Las zonas (círculos o cuadrados) que contiene a los vértices A y B se unen así es que la arista entre estos vértices es una buena candidata para formar parte del acoplamiento ya que es la de menor distancia que hay entre dos vértices.

Tomando este criterio de optimización se marca la arista como lo muestra la figura 10 y se continúa con la siguiente distancia más pequeña, de tamaño 7 para esta gráfica.

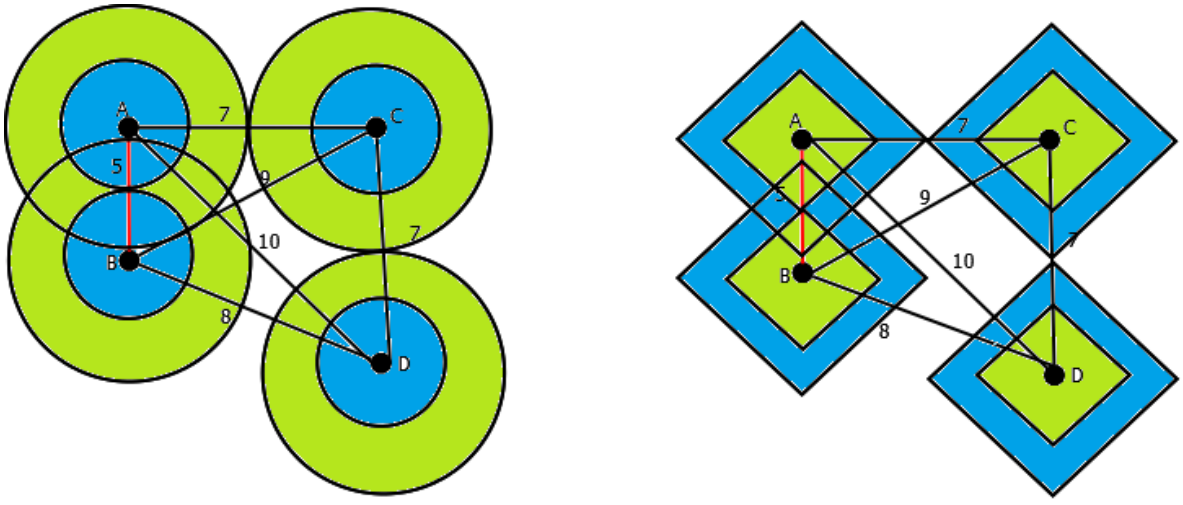

Figura 11.- Etapa 3 de la construcción del acoplamiento

Se construyen zonas para cada vértice de tamaño 7. En la figura 11 se muestra que las zonas de los vértices A-C y C-D se unen, sin embargo la arista de A-C ya no es candidata a estar en el acoplamiento porque A ya forma parte de una arista que está en el acoplamiento con la garantía que es de menor peso que cualquier otra. De manera que entra al acoplamiento la arista de C-D (figura 12).

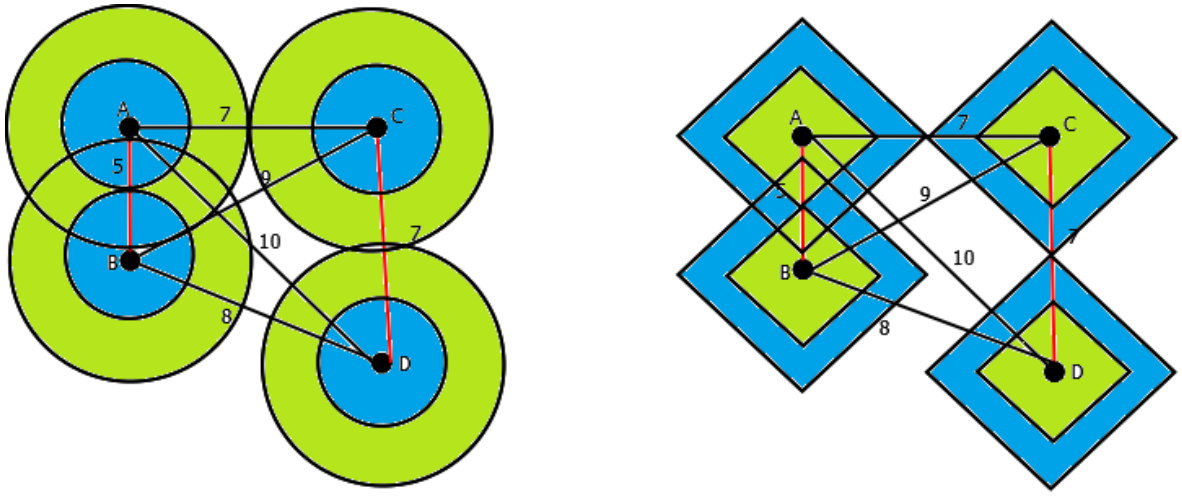

Figura 12 Etapa 4 de la construcción del acoplamiento

La figura 12 muestra las aristas marcadas (A-B y C-D) y como ya no hay mas vértices finaliza la construcción del acoplamiento cubriendo todos los vértices de la gráfica con una distancia de tamaño 12 unidades. En la figura 13 se puede observar el acoplamiento óptimo

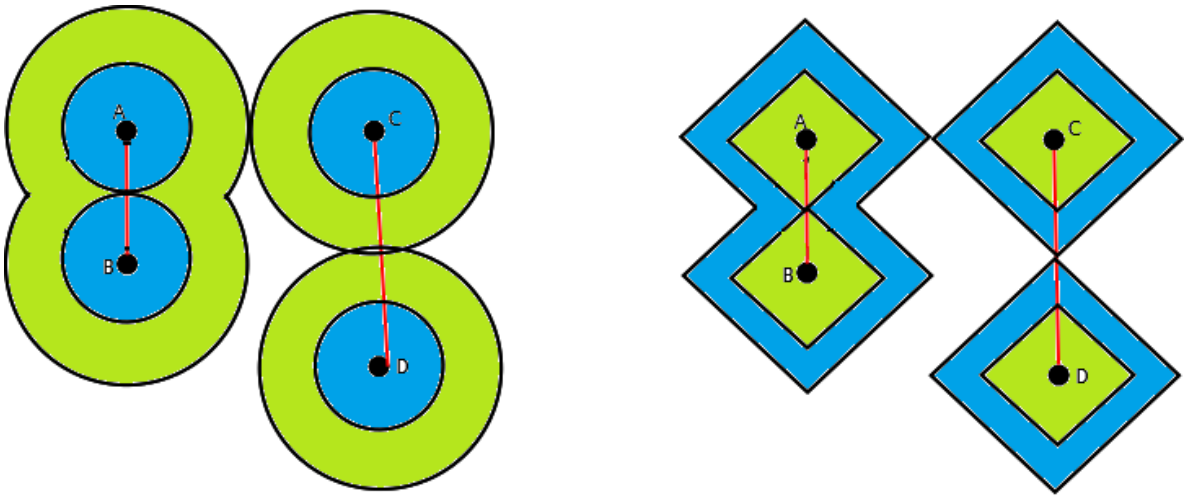

Figura 13.- Acoplamiento óptimo

Desarrollo del proyecto

A fin de cumplir el objetivo principal de este proyecto se llevó a cabo la siguiente metodología.

Para extender el estudio del problema del acoplamiento máximo en gráficas bipartitas, al problema de acoplamiento máximo a costo mínimo en gráficas completas. Se hizo un estudio del algoritmo Húngaro (Kuhn, 1955) el cual modela un problema de asignación como una matriz de costes n×m, donde cada elemento representa el coste de asignar el enésimo trabajador al emésimo trabajo.

Algoritmo Húngaro:

(1) Dada la matriz de costes$C$, se construye C' encontrando el valor mínimo de cada fila y restando ese valor a cada elemento de la fila.

$$\Rightarrow \ C'_{i,j} = C_{i,j} - \min C_{i,j}$$

(2) Se encuentra el valor mínimo de cada columna y se resta a cada elemento de la columna.

$$\Rightarrow \ C'_{i,j} = C'_{i,j} - \min C_{i,j}$$

(3) A partir de C' se considera "gráfica de las igualdades" a $G = (N1, N2, A)$ tal que A está constituido por todas las copias (i, j) tales que $C'_{i,j} = 0$. En otras palabras, verificamos si para todas las filas existe una columna con costo 0 que no ha sido asignada a otra fila.

Determinar sobre G un acoplamiento M de cardinalidad máxima.

si $|M| = |N_1| = |N_2| \Rightarrow \ STOP$

Para el diseño e implementación de la interfaz de la aplicación se utilizó Windows Presentation Foundation (WPF) con el lenguaje de programación C#. Y con el objetivo de dibujar gráficas completas se hizo uso de un software de código abierto para la visualización de gráficas llamado graphviz (AT&T, 2015). Graphviz utiliza el lenguaje dot de descripción de gráficas.

El siguiente código muestra algunos aspectos de la programación.

```
MiProceso.StartInfo.UseShellExecute = true;
MiProceso.StartInfo.FileName = (@"C:\Program Files (x86)\Graphviz2.38\bin\neato.exe"); // el compilador de GraphViz ... graphviz.exe MiGrafo.dat MiIma
MiProceso.StartInfo.CreateNoWindow = true;
MiProceso.StartInfo.Arguments = "-Tjpg " + ex1 + " -o grafo1.jpg"; // <argumentos>
MiProceso.Start();
System.Threading.Thread.Sleep(300);
BitmapImage myBitmapImage = new BitmapImage();
miimagen.Source = new BitmapImage(new Uri(@"C:\Users\Pc\Documents\Visual Studio 2012\Projects\GraficaCompleta\GraficaCompleta\bin\Debug\grafo1.jpg"));
```

Figura 14.- Código para conectar C# con graphviz

En la figura 14 se puede observar el proceso para hacer una llamada al compilador de Graphviz. Se le da como parámetro un archivo de texto en lenguaje dot, graphviz ejecuta ese archivo y devuelve la imagen de la gráfica en un formato jpg. Se carga la imagen y se muestra en pantalla.

Graphviz tiene varios compiladores de lenguaje dot: dot, neato, twopi, fdp, sfdp y circo. Se empezó el proyecto utilizando el compilador dot, pero algunos de los problemas encontrados fueron que el compilador dot solo permite dibujar graficas jerárquicas y no es posible controlar el cruce de entre aristas así es que dibujar un grafica completa se volvía imposible. Teniendo en cuenta estos detalles del compilador se descubrió que neato era un compilador para poder dibujar graficas

no dirigidas con la posibilidad de tener más control en las aristas con respecto a la distancia entre vértices y al tipo de línea, permitiendo el cruce de aristas.

La figura 15 muestra el código para describir una gráfica completa con 4 vértices (K4) en lenguaje dot.

```
graph G
{
  graph [splines=line splines=false ]
   node [shape=point]; node1; node2; node3,  node4;


  node1 -- node2  [len=1  ];
  node1 -- node3  [len=1  ];
  node1 -- node4  [len=1  ];
  node2 -- node3  [len=1  ];
  node2 -- node4  [len=1  ];
  node3 -- node4  [len=1  ];
}
```

Figura 15.- Código de K4

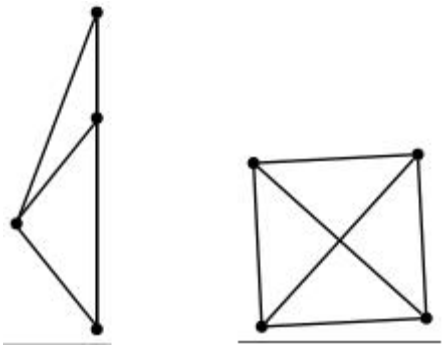

Figura 16.- K4 imagen generada por el compilador dot y por neato

La figura 16 muestra las imágenes generadas al ejecutar el mismo código (figura 15) con dos diferentes compiladores (dot y neato). La imagen generada por el compilador dot ignora una de las aristas pues no permite aristas encimadas, una

falla importante en el contexto de gráficas ya que son importantes todos los vértices y aristas. En contraste neato dibuja una K4 con todos sus vértices y aristas.

Para fines de la visualización del algoritmo era necesario poder dibujar círculos o cuadrados para cada vértice de la gráfica. En el lenguaje dot la función *ranksep* (Graphviz, 2014) permite dibujar y controlar el crecimiento de los círculos o cuadrados, desafortunadamente esta función puede ser utilizada con el compilador dot y twopi únicamente.

A fin de poder dibujar las zonas de control se pensó en utilizar C# y que teniendo conocimiento la ubicación de los vértices generados por neato, desde la interfaz en WPF se dibujaran las zonas de control. Sin embargo la ubicación de los vértices generada por neato es automática y el usuario no tiene conocimiento de la ubicación de estos, ningún archivo generado durante la compilación da información de la ubicación de las gráficas o los elementos de esta (vértices y aristas).

A pesar de que se buscó otra solución para este inconveniente no fue posible encontrar una solución óptima al problema lo cual limita la parte visual del algoritmo.

Resultados

Se obtuvo una aplicación que muestra una gráfica completa, bajo ciertos parámetros introducidos por el usuario la aplicación muestra la gráfica con las distancias entre vértices especificadas también por el usuario.

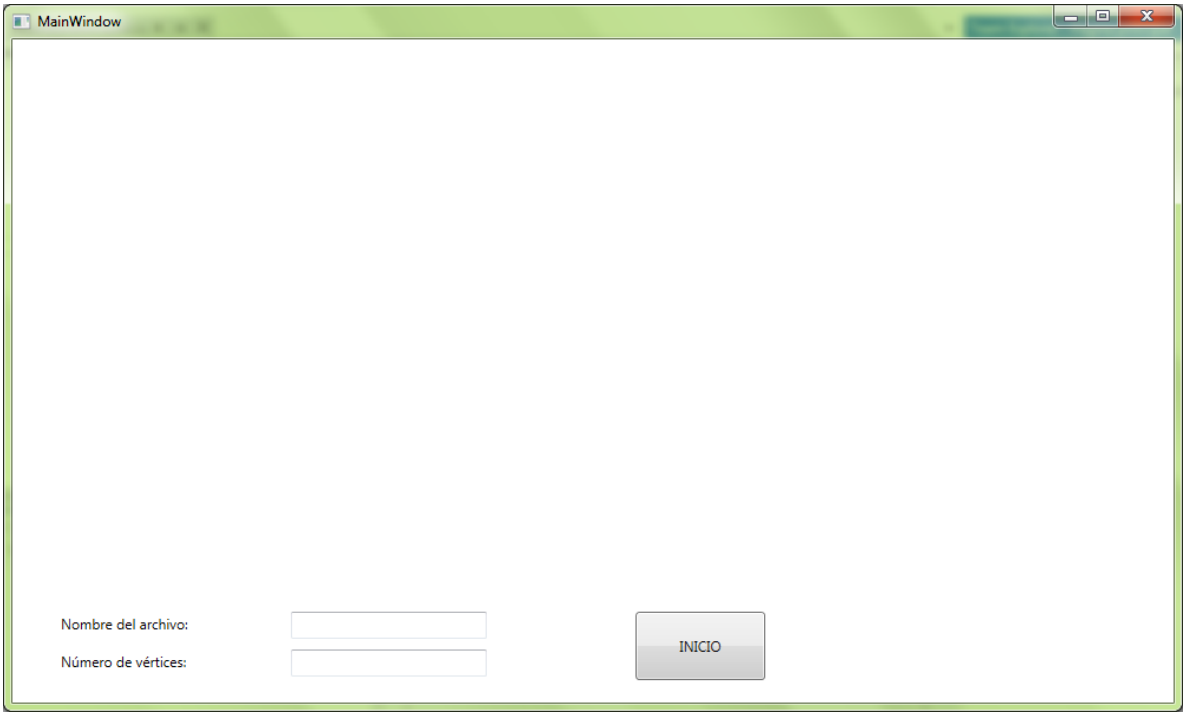

Figura 17.- Interfaz

La figura 17 muestra la interfaz principal con la que interactúa el usuario. Los campos nombre de archivo y numero de vértices son introducidos por el usuario. La aplicación lee un archivo de texto con las siguientes especificaciones: debe de tener como datos el vértice de inicio, vértice final y la distancia entre ellos. Debido a que es una gráfica completa (cada vértice es adyacente a todos los vértices de la gráfica) es fácil saber que de acuerdo al número de vértices, el número de aristas de la gráfica está dado por la siguiente formula:$\frac{1}{2}\ vèrtices(vèrtices-1)$ de

ahí la importancia de que el usuario introduzca el número de vértices en la gráfica pues el programa hace una validación de que el número de vértices en la gráfica sea congruente con los datos proporcionados.

La figura 18 muestra un ejemplo del archivo de texto introducido por el usuario. El primer número está relacionado con el vértice de inicio seguido por un delimitador el cual pude ser (`'-',',',' ',';', ':'`), el vértice final, otro delimitador y la distancia entre los vértices.

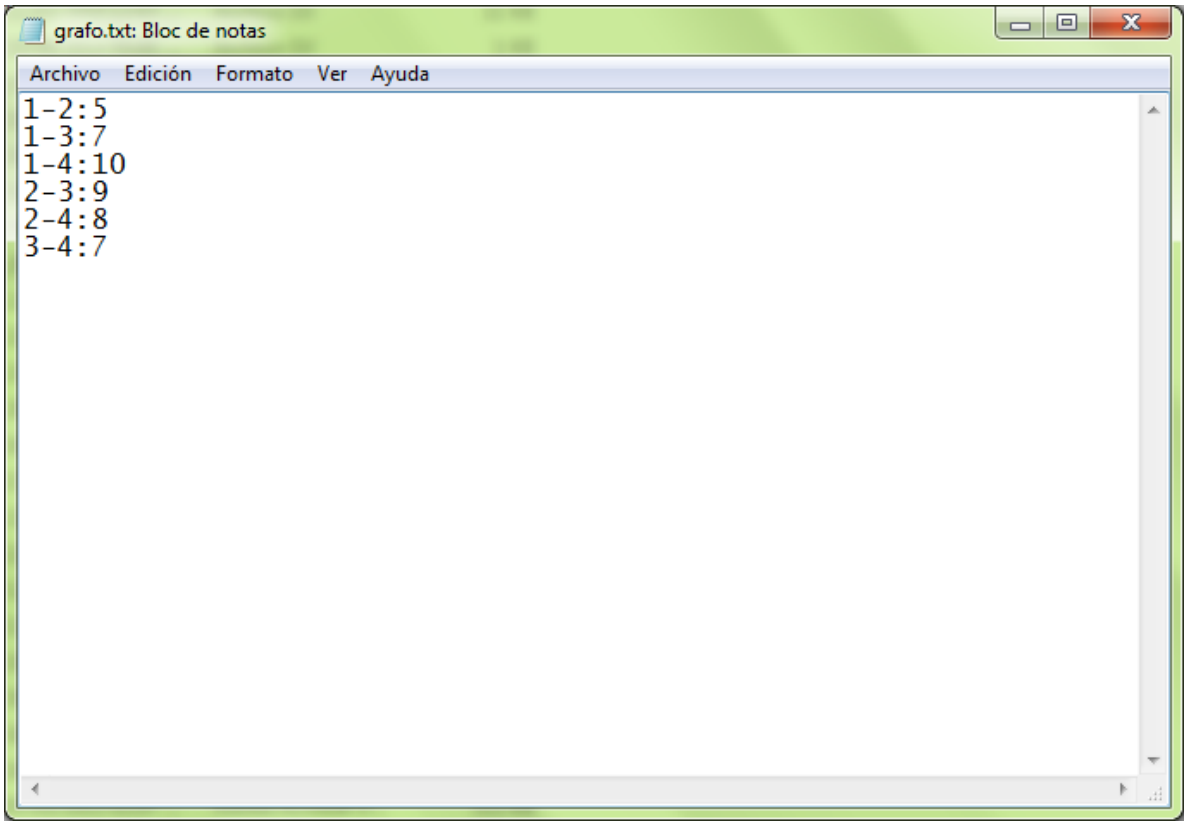

Figura 18.- Ejemplo de archivo de texto de entrada para la aplicación

Si el número de vértices no corresponde al número de datos que son necesarios para dibujar la gráfica completa la aplicación manda un mensaje como el que se puede ver en la figura 19.

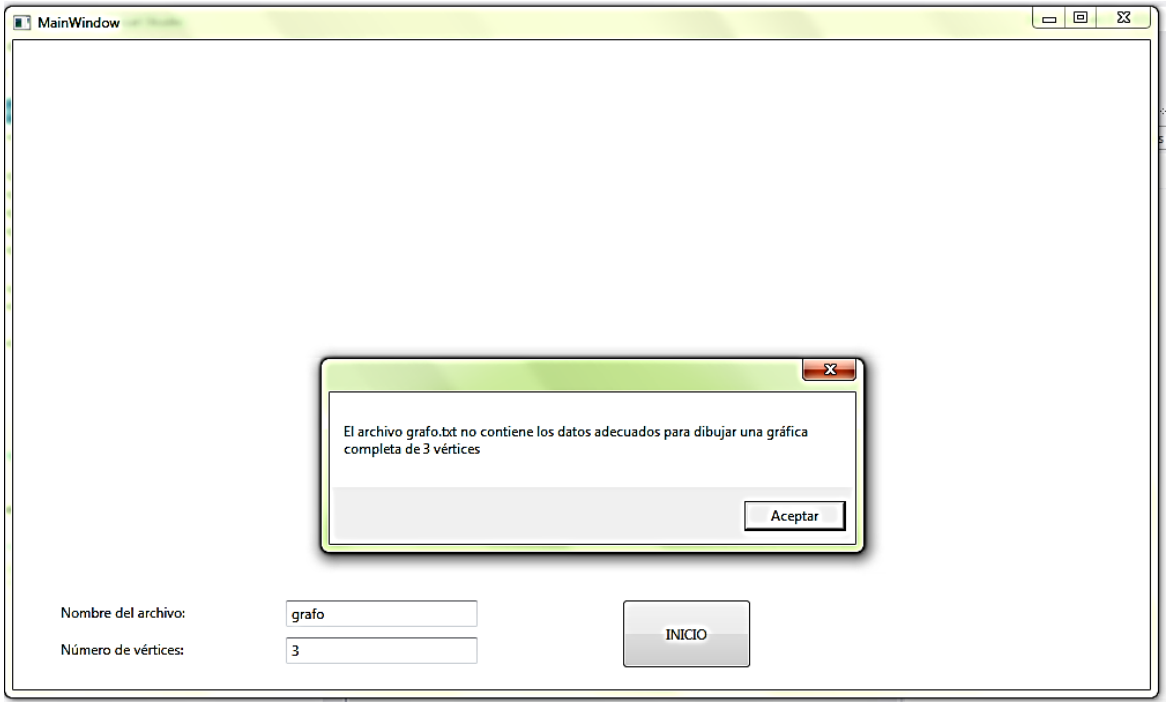

Figura 19.- Validación del número de vértices y los datos introducidos

Si es correcto tanto el archivo como el número de vértices la aplicación crea un archivo de texto en lenguaje dot que se envía al compilador de graphviz y este a su vez devuelve la imagen generada y se muestra en la aplicación como se puede observar en la figura 20.

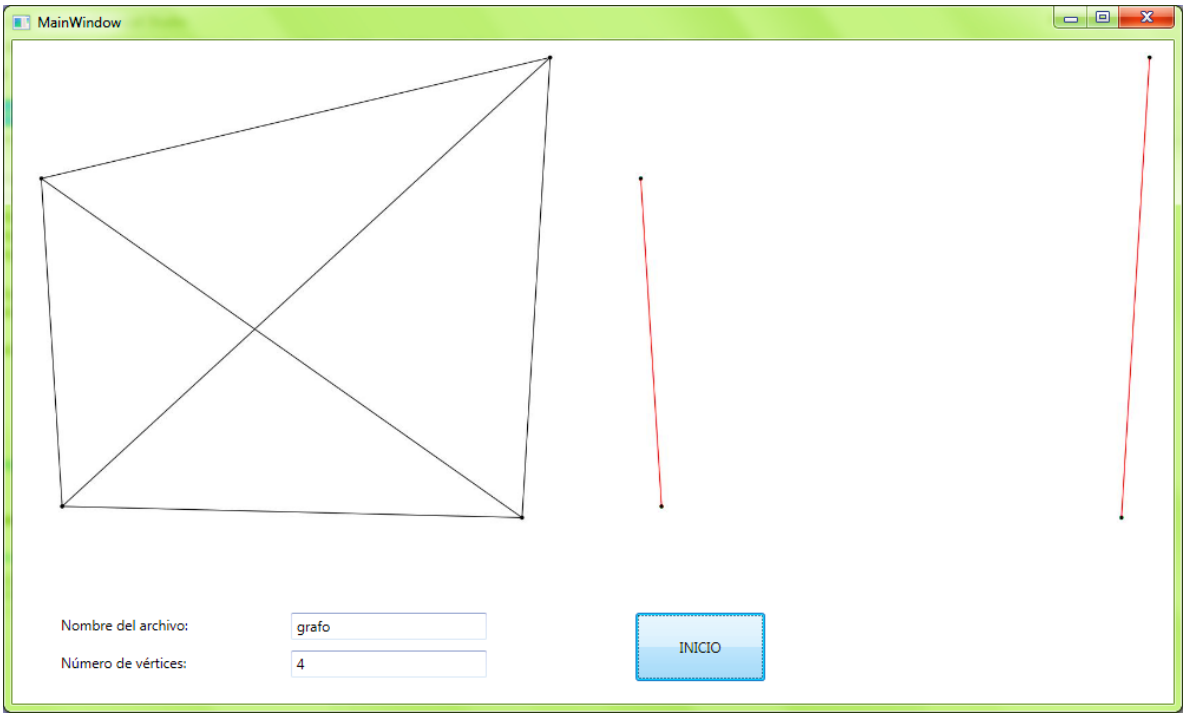

Figura 20.- Ejecución de la aplicación, acoplamiento óptimo

La figura 20 muestra no solo la gráfica dibujada en graphviz con las especificaciones introducidas por el usuario, sino el resultado de aplicar el algoritmo de Goemans-Williamson. Presenta los vértices y aristas que están en el acoplamiento.

Conclusiones

De acuerdo a los objetivos planteados al inicio del proyecto y con el visto bueno de los asesores se considera que se terminó el proyecto en el tiempo y forma.

Bibliografía

Arguello, J. F. (2005). *Introducciòn al càlculo.* EUNED.

AT&T. (2015). *Graphviz*. Obtenido de http://www.graphviz.org/

Edmonds, J., & Karp, R. M. (1972). *Theoretical Improvements in Algorithmic Effciency for Network Flow Problems.* J. ACM.

Graphviz. (2014). *Node, Edge and Graph attributes*. Obtenido de www.graphviz.org/doc/info/attrs.html

Kuhn, H. w. (1955). *The Hungarian metod for the assignment problem.* Naval Research Logistic Quartely.

Osuna, A. M. (1998). *El problema de acoplamiento en gràficas bipartitas.* Hermosillo, Sonora: Universidad de Sonora.

Weisstein, E. W. (Enero de 2013). *Triangle Inequality.* Recuperado el Enero de 2015, de http://mathworld.wolfram.com/TriangleInequality.html

William J. Cook, W. H. (1998). *Combinatorial Optimization.* Canada: John Wiley.

Willson, R. J. (1996). *Introduction to graph teory.* Inglaterra: Prentice Hall.