01149

24

UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA
DIVISIÓN DE ESTUDIOS DE POSGRADO

PROPUESTA DE UN ALGORITMO BASADO EN EL ALGORITMO DE DIJKSTRA QUE RESULEVE EL PROBLEMA DE RUTA MÁS CORTA MODELADO CON REDES CUYOS ARCOS TIENEN LONGITUDES NEGATIVAS.

T E S I S

QUE PARA OBTENER EL GRADO DE:
MAESTRO EN INGENIERÍA
ÁREA SISTEMAS
MÓDULO DE INVESTIGACIÓN DE OPERACIONES

PRESENTA:
GEORGINA LÓPEZ HERNÁNDEZ

DIRECTOR DE TESIS:
DR. MARCO ANTONIO MURRAY LASSO

CIUDAD UNIVERSITARIA, MÉXICO, D.F, ENERO DE 2002

Dedico esta tesis a mi madre y a la memoria de mi padre, a mis hermanos: Gabrie, Gigos, Grisos, Gordo y Alex; y a mi maravillosa familia, Alexis.

AGRADECIMIENTOS

A la Universidad Nacional Autónoma de México.

A la Facultad de Ingeniería y su División de Estudios de Posgrado.

Al Dr. Marco Antonio Murray Lasso por su gran apoyo para la realización de este trabajo.

A todos y cada uno de mis profesores por su tiempo y dedicación.

A todos por los que fue posible este trabajo.

ÍNDICE

I INTRODUCCIÓN

1.1 Resumen

El problema de ruta más corta es uno de los problemas fundamentales de la optimización de redes. Los algoritmos para este tipo de problemas se han estudiado por varios años. Sin embargo, se siguen realizando estudios para eficientar los algoritmos de ruta más corta.

El algoritmo de Dijkstra (Dijkstra, 1959), uno de los algoritmos más conocidos, es el algoritmo más eficiente para resolver el problema de ruta más corta modelado con redes cuyos arcos tienen longitudes positivas, según un estudio realizado por Cherkassky, Goldberg y Radzik en 1996. Sin embargo, al resolver el problema de ruta más corta en donde la red que modela el problema tiene arcos con longitudes negativas, el algoritmo de Dijkstra no llega a la solución.

En 1999 Cherkassky y Goldberg utilizaron una variante al algoritmo de Díjkstra que admite arcos negativos, se evaluó su desempeño junto con los algoritmos más eficientes conocidos hasta la fecha y se observó que la variante utilizada por Cherkassky y Goldberg es sumamente ineficiente.

En el presente trabajo se propone una modificación al algoritmo de Dijkstra, basada en un ordenamiento topológico, que resulta ser eficiente.

Cherkassky y Goldberg (1999) observaron que los algoritmos más eficientes en la solución del problema de ruta más corta en redes con arcos negativos son: el algoritmo de Goldberg-Radzik (Goldberg y Radzik, 1993), el algoritmo de Tarjan (Tarjan, 1981) y el Simplex (Cherkassky y Goldberg, 1999). El algoritmo propuesto en este trabajo se compara con estos algoritmos para observar su desempeño; además, se incluye la comparación con la variante del Dijkstra usada por Cherkassky y Goldberg (1999) y con el algoritmo clásico de Bellman-Ford-Moore (Bellman, 1958; Ford, 1962; Moore, 1959).

La evaluación de los algoritmos consiste en medir el tiempo que tardan en resolver el problema de ruta más corta sobre distintos tipos y tamaños de redes. Las redes utilizadas son las mismas que se utilizaron en el estudio previo realizado por Cherkassky y Goldberg en 1999.

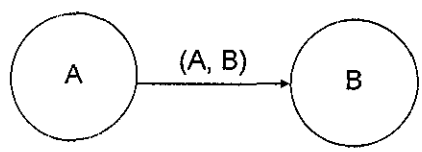
Los programas generadores de redes que se utilizaron en este trabajo y los algoritmos probados por Cherkassky y Goldberg (1999) se obtuvieron de internet (http://www.intertrust.com/star/goldberg/index.html) y se les hicieron pequeñas modificaciones para correrlos en una P.C. con Windows 98 y el compilador de Borland C++ 5.0.

Los resultados obtenidos por el algoritmo propuesto se compararon con los obtenidos por los algoritmos más eficientes, llegando a la conclusión de que la modificación propuesta en este trabajo es un algoritmo robusto y eficiente, cumpliendo así con el objetivo propuesto.

1.2 Redes

Las redes son importantes en nuestra vida diaria; por ejemplo, las redes telefónicas, las redes eléctricas, las redes de carreteras, trenes y aviones, las redes de distribución de productos, entre muchos otros ejemplos. En la mayoría de estos problemas domina el deseo de mover algo (información, electricidad, productos, personas, vehículos, etc.) de un punto a otro en una red de forma eficiente. Debido a esta necesidad de eficiencia se han estudiado y descubierto varios algoritmos para los diferentes problemas que se pueden modelar con una red.

Una red se representa mediante un conjunto de puntos y un conjunto de líneas que unen ciertos pares de puntos. Los puntos se llaman vértices o nodos, los cuales se representan por círculos, y las líneas se llaman arcos o ligaduras. Los arcos se etiquetan con el nombre de los nodos en sus puntos terminales; por ejemplo, (A, B) es el arco entre los vértices A y B. Otra forma de etiquetarlos es A → B.



Los arcos de una red pueden tener flujos de algún tipo que pasa por ellos, por ejemplo, el flujo de tranvías sobre los caminos de una ciudad. Sin embargo, existen modelos un poco más abstractos; por ejemplo, en ciertos problemas, los arcos pueden representar actividades de algún tipo y los valores asociados a cada uno pueden ser los costos de esas actividades.

Si el flujo a través de un arco se permite sólo en una dirección se dice que el arco es un arco dirigido. La dirección se indica agregando una cabeza de flecha al final de la línea que representa el arco. Al etiquetar un arco dirigido con el nombre de los vértices que une, siempre se pone primero el nodo de donde viene y después el nodo a donde va; esto es, un arco dirigido del nodo A al nodo B debe etiquetarse como (A, B) y no como (B, A). Una red que sólo tiene arcos dirigidos se llama red dirigida.

1.3 Problemas de ruta más corta

El problema de ruta más corta surge de forma natural en problemas que se modelan con redes. El objetivo del problema es encontrar la ruta más corta (la trayectoria con la mínima distancia total) que va desde un punto origen a un punto destino, el procedimiento es analizar toda la red a partir del punto origen, identificando sucesivamente la ruta más corta a cada uno de los nodos. El problema de ruta más corta se usa con frecuencia para resolver redes de transporte, en donde se desea conocer el mejor camino a seguir desde un punto origen a cada uno de los puntos destinos.

Es importante mencionar que el problema de ruta más corta no sólo se utiliza para minimizar la distancia de un punto origen a un punto destino, en realidad el problema de ruta más corta es el encontrar cuál es la ruta que conecta a dos nodos específicos que minimiza la suma de los valores de los arcos sobre esa ruta. Por ejemplo, los arcos pueden corresponder a actividades de algún tipo y los valores asociados a cada uno de ellos pueden representar el costo de esa actividad. Entonces, el problema sería encontrar qué secuencia de actividades logra el objetivo de minimizar el costo total relacionado. Otra posibilidad consiste en que el valor asociado a cada arco sea el tiempo requerido para realizar esa actividad, en este caso se desearía encontrar la secuencia de actividades que logra el objetivo específico de minimizar el tiempo total requerido. Así algunas de las aplicaciones más importantes del problema de la ruta más corta no tienen nada que ver con distancias.

El problema de ruta más corta atrae tanto a investigadores como a profesionales por varias razones:

- El poder plantear problemas modelados con una red, en donde los arcos pueden representar: distancias, tiempos, costos o flujos de datos (telefónicos, computacionales, etc.) entre dos puntos específicos.

- Al seguir estudiando este tipo de problemas, se continúan desarrollando los procedimientos de solución. El uso de estructuras de datos y la forma de agrupación de datos son algunas de las ideas que han surgido como desarrollo de estas investigaciones.

- Frecuentemente surge como un subproblema al resolver problemas combinatorios y de optimización de redes, por lo que su campo de aplicación es muy grande.

Algunos ejemplos de problemas que se resuelven con algoritmos de ruta más corta son: optimizar líneas de producción y distribución de productos (Ahuja *et al.*, 1991), determinar vías de comunicación (Ahuja *et al.*, 1991), determinar rutas de transporte (Ahuja *et. al*, 1991), determinar la duración mínima de un proyecto y la calendarización justo a tiempo (JIT) (Elmaghraby, 1978), estudios de alineación secuencial del DNA (Waterman, 1988), problemas de distribución (Glover y Klingman, 1977), asignación de

recursos para proyectos (Ahuja *et al.*, 1991), problema de congestión de tráfico (Zawack y Thompson, 1987), entre muchos otros.

Por lo que en general se puede decir que el problema de ruta más corta se presenta mucho en la industria, incluyendo la agricultura, comunicaciones, educación, energía, cuidado de la salud, manufactura, medicina y transporte, entre muchos otros.

Existen diferentes tipos de problemas a los que se les llama de ruta más corta, entre los principales podríamos mencionar el encontrar la ruta más corta de

a. Un vértice origen a todos los demás vértices.
b. Un vértice origen a un vértice destino.
c. desde todos los vértices a un vértice destino.
d. desde cada vértice a todos los demás.

En el presente trabajo se considera como el problema de ruta más corta el de encontrar, en una red, el camino más corto desde un vértice origen *a* todos los demás vértices; esto es, el problema definido en el inciso *a*.

Veamos un ejemplo ilustrativo: Un camión repartidor de pan dulce tiene que hacer un recorrido por cada una de las tiendas que hay dentro de un segmento de la ciudad. Su recorrido empieza en el almacén principal. Se desea encontrar, para el conductor del camión, la ruta más corta desde el punto de partida hasta cada una de las tiendas.

A continuación se presenta una red en donde se incluye al almacén A y las tiendas T_i. Los arcos representan las distancias entre los vértices de la red. Las distancias están en Km.

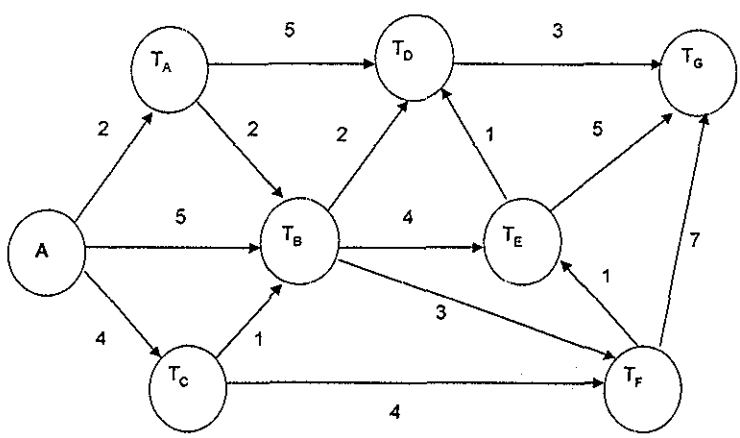


Figura 1.1

Después de realizar el análisis se determinó que la ruta más corta desde el almacén para realizar el recorrido por las tiendas es la que se muestra en la figura 1.2.

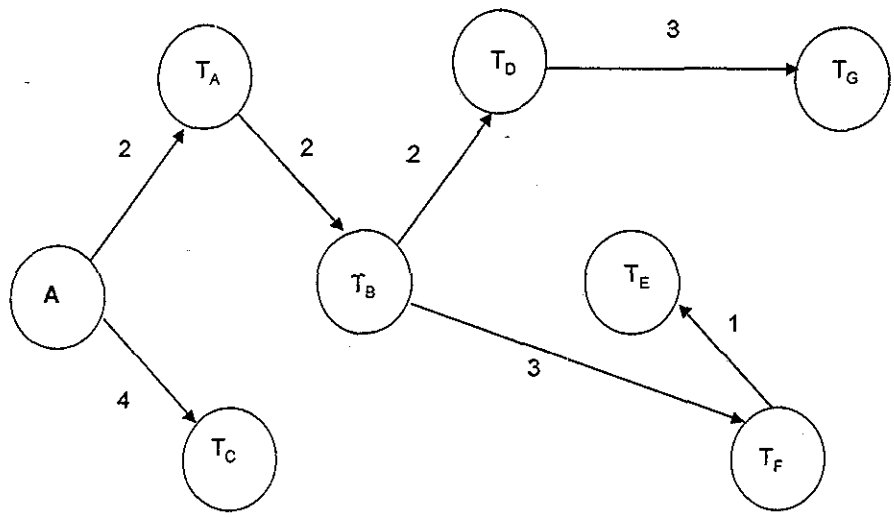


Figura 1.2

El ejemplo anterior se puede resolver analizando directamente la gráfica a través de un procedimiento simple, como podría ser el de prueba y error. Sin embargo, cuando el problema crece, es necesario utilizar algoritmos que nos den el resultado en un menor tiempo. Una herramienta fundamental para resolver el problema es la computadora ya que permite almacenar una gran cantidad de datos y operar algoritmos específicos en tiempos sumamente cortos.

Ya que las redes se usan para modelar diferentes tipos de problemas, hay ocasiones en que se presentan redes con longitudes de arcos arbitrarias, lo cual significa que pueden ser tanto positivas como negativas. En el ejemplo anterior se puede observar que las longitudes de los arcos representan la distancia entre un punto y otro, por lo cual las longitudes son todas positivas. Pero hay ciertos casos en donde la "longitud" de los arcos es negativa, como cuando representan los costos de los flujos (pérdidas o ganancias). Estos modelos surgen como un subproblema para resolver problemas de flujo de costo mínimo y pueden ser representados como problemas de ruta más corta. (Ahuja *et. al*, 1991).

Es importante mencionar que la necesidad de incluir arcos con longitudes negativas no es solamente para representar pérdidas o ganancias. Por ejemplo, el algoritmo que emplean Busacker y Gowen (T.C. Hu, 1969) para resolver problemas de flujo a costo mínimo, en los cuales el problema de ruta más corta aparece como un subproblema, los flujos o longitudes de algunos arcos se vuelven negativos como parte del procedimiento, a pesar de que en la red inicial los flujos de todos los arcos son positivos.

Para ejemplificar una de tantas aplicaciones de problemas en los que se presentan redes con longitudes de arcos arbitrarios, veamos el siguiente problema:

Supongamos que un vendedor de Michoacán planea venir a la Ciudad de México para ver a un cliente. El vendedor quiere utilizar las autopistas para aprovechar el viaje e ir a visitar a otros clientes que viven en pueblos que le quedan de paso y obtener una ganancia extra en la carretera. Él sabe, en promedio, cuanto ganará de comisión por cada uno de los clientes que visite durante su recorrido. El vendedor quiere saber cómo puede obtener la máxima ganancia durante su recorrido por la carretera desde Michoacán a la Ciudad de México. (Minieka, 1978)

Este problema se puede modelar con una red como la que se muestra en la figura 1.3, en la que los nodos representan las ciudades y los arcos al sistema de carreteras. La cantidad asociada con los arcos, a la que llamamos longitud del arco, no representa una longitud física sino el costo neto.

Costo neto = Costo total – Ganancia total, donde:

Costo total = costo de gasolina + costo de carretera

Ganancia total = ganancia por comisión.

Por lo que en esta red, las longitudes de los arcos que son negativas representan una ganancia por parte del vendedor y cuando las longitudes son positivas representan una pérdida para el vendedor.

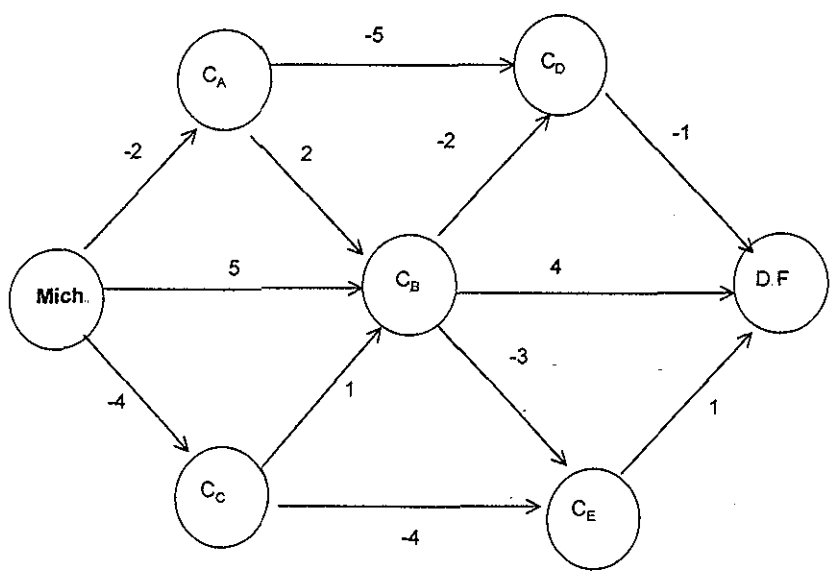


Figura 1.3

La solución del problema de la ruta más corta, tomando a Michoacán como nodo origen se muestra en la figura 1.4.

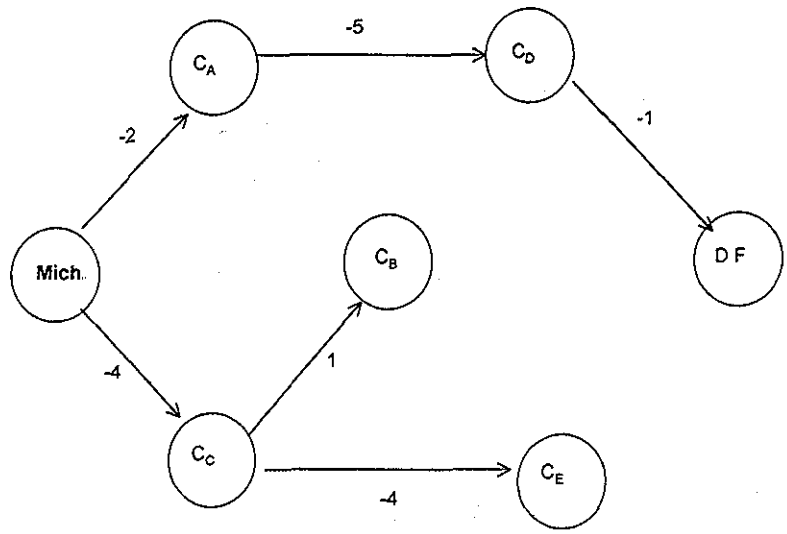


Figura 1.4

En esta solución se puede ver la ruta más adecuada que puede realizar el vendedor para llegar al D.F. con la menor pérdida visitando a varios clientes durante su recorrido por la carretera. Al visitar a los clientes A y D durante su recorrido hacia el D.F., el vendedor obtiene una ganancia total de 8. Como se mencionó anteriormente, al tener redes más grandes y complejas se necesita el uso de un algoritmo eficiente y una computadora para aplicarlo.

La mayoría de los problemas cuyos arcos representan costos de los flujos pueden presentar longitudes de arcos positivas, negativas o arbitrarias según el tipo de problema. En problemas del flujo a costo mínimo, como se comentó anteriormente, el problema de ruta más corta surge como un subproblema para llegar a la solución. Por ejemplo, si se quiere maximizar el flujo en una red considerando una capacidad para cada nodo y un costo asociado, es necesario encontrar la ruta más corta para minimizar los costos y que cubra la capacidad requerida. (Ahuja *et al.*, 1991)

Para problemas en los que las longitudes de los arcos son todas positivas, el algoritmo de Dijkstra, que será explicado en el siguiente capítulo, resulta ser el más eficiente (Cherkassky *et al.*, 1996). Sin embargo, cuando se presentan longitudes negativas, el algoritmo de Dijkstra no converge por lo que no se llega a una solución para este tipo de redes.

En el presente trabajo se propone una modificación al algoritmo de Dijkstra para que resuelva eficientemente problemas cuyas redes tengan arcos con longitudes negativas.

1.4 Objetivo

Modificar el algoritmo de Dijkstra para que resuelva eficientemente el problema de ruta más corta modelado con redes con longitudes de arco arbitrarias.

1.5 Descripción del trabajo

En el siguiente capítulo, *llamado Algoritmo de Dijkstra*, se da una explicación sobre la definición y notación usada para problemas de ruta más corta. También se explica detalladamente el algoritmo de Dijkstra, se da un ejemplo ilustrativo y un caso donde se presenta un problema de ruta más corta modelado con una red con arcos cuyas longitudes son negativas.

En el tercer capítulo, *Algoritmos para redes con longitud de arcos negativas*, se describen los algoritmos más eficientes, según un estudio realizado por Cherkassky y Goldberg (1999), así como un pequeño ejemplo para cada uno. Los algoritmos son: variante del algoritmo de Dijkstra utilizado por Goldberg y Cherkassky, el algoritmo de Bellman-Ford-Moore, el Simplex, el algoritmo de Tarjan y el algoritmo de Goldberg-Radzik.

El cuarto capítulo, *Algoritmo propuesto*, describe detalladamente el desarrollo del algoritmo propuesto en este trabajo, algoritmo de Dijkstra modificado, con un pseudocódigo y un ejemplo que se explica paso a paso.

El quinto capítulo, *Experimentación y resultados*, describe la estructura y el tipo de cada una de las redes utilizadas para la experimentación, la escala de evaluación utilizada y los resultados finales a los que se llegó.

El sexto capítulo da las conclusiones y recomendaciones para trabajos posteriores.

En el apéndice A se incluye el código del programa del algoritmo propuesto en lenguaje C.

En el apéndice B se incluye el formato para proporcionar los datos de la red a los programas resolvedores y los archivos de comandos que se utilizaron en los experimentos.

En el apéndice C se incluyen los resultados completos de los experimentos.

En el disco compacto (D.C.) anexo se incluyen todos los programas generadores de las redes, el código de los programas originales obtenidos de internet, el código de todos los algoritmos utilizados para este trabajo y los resultados obtenidos de los experimentos.

II ALGORITMO DE DIJKSTRA

Como se mencionó en el capítulo anterior, si las longitudes de los arcos en una red son no negativas, el algoritmo de Dijkstra logra el menor tiempo en obtener la solución del problema de ruta más corta, pero al tener redes con longitudes de arcos negativas, el algoritmo de Dijkstra no funciona, debido a que no cumple con las condiciones de terminación del método de etiquetamiento que veremos más adelante en este capítulo.

En este capítulo se explicará el algoritmo de Dijkstra y se ejemplificará su desarrollo cuando se presentan redes con longitudes de arcos no negativas. También se presentará un ejemplo con una red cuyas longitudes de arco son negativas y veremos cómo el algoritmo de Dijkstra no llega a la solución correcta.

Primero se describirá la definición y la notación para las redes que modelan problemas para encontrar la ruta más corta. (Cherkassky *et. al*, 1996, Cherkassky y Goldberg, 1999)

2.1 Definición y notación

Como se mencionó en el capítulo anterior, el problema de la ruta más corta se modela con una red, la cual denotaremos como G. La red se compone de un conjunto de vértices o nodos V, y un conjunto de arcos E, por lo que denotaremos a la red como un conjunto de nodos y arcos, G = (V, E). Además, el número de nodos se denotará con *n* y el número de arcos con *m*.

El problema de ruta más corta se compone de una red G, un nodo origen *s*, donde $s \in V$ y una función de longitud l; donde l: E→ $\mathbb{R}$, que proporciona las longitudes de cada uno de los arcos. El objetivo del problema de ruta más corta (G, *s*, l) es, por lo tanto, encontrar la ruta más corta, conociendo la longitud de los arcos a través de la función l, desde el vértice origen *s* hasta cualquier otro vértice de G; asumiendo, sin perder la generalidad, que todos los vértices son alcanzables desde *s* en G.

Para poder resolver este tipo de problemas, se debe tener una etiqueta de longitud *d*, la cual es una función *d*: V → $\mathbb{R} \cup \{\infty\}$, que nos dirá la distancia calculada desde el nodo origen *s* hasta el vértice en cuestión. Al principio de cada algoritmo la etiqueta de longitud asignada a cada uno de los nodos, exceptuando al nodo origen, será de ∞.

Dada una etiqueta de longitud *d*, definimos la función de costo reducido $l_d : E \to \mathbb{R} \cup \{\infty\}$ como :

$$l_d [(v,w)] = l [(v,w)] + d(v) - d(w)$$

Si $\ell_d [(v,w)] \leq 0$, decimos que el arco (v,w) es admisible. Al conjunto de arcos admisibles se denota como E_d. Por lo que la gráfica admisible se define por $G_d = (V,E_d)$. Note que si $d(v) < \infty$ y $d(w) = \infty$, el arco (v,w) es admisible. Si $d(v) = d(w) = \infty$, se define $\ell_d(v,w) = \ell(v,w)$; y en este caso, el arco (v,w) es admisible sólo si $\ell(v,w) \leq 0$.

El árbol de la ruta más corta de G es un árbol con raíz en *s* tal que, para cualquier $v \in V$, el camino de *s* a *v* en el árbol es el camino más corto desde *s* hasta *v*.

Dado un árbol y un vértice *v* en el árbol, se entiende por profundidad de *v* al número de arcos desde la raíz hasta *v*.

A lo largo de un algoritmo se dice que $d(v)$, la cual va cambiando durante la solución, es exacta, si la distancia mínima de *s* a *v* en G es igual a $d(v)$; de otra forma es inexacta.

Método de etiquetamiento

En esta sección se describe el método de etiquetamiento (Ford, 1956; Ford y Fulkerson, 1962, Cherkassky y Goldberg, 1999) para resolver problemas de ruta más corta. Muchos algoritmos se basan en este método.

Para cada vértice *v*, el método mantiene la etiqueta de longitud $d(v)$ y el padre del vértice, $\pi(v)$. Inicialmente para cada vértice *v*, $d(v) = \infty$ y $\pi(v) =$ nulo.

El método inicia al poner $d(s) = 0$ y $\pi(s) = s$., donde *s* es el nodo origen.

En cada paso, el método selecciona un arco (u, v) tal que:

$$d(u) < \infty \text{ y } d(u) + l(u,v) < d(v)$$

y coloca

$$d(v) = d(u) + l(u,v),$$

$$\pi(v) = u.$$

Si no existe algún arco que cumpla con las condiciones de selección, el algoritmo termina.

Lema 2.1.- El método de etiquetamiento mantiene la invariante de que si d(v) es finita, existe un camino desde s hasta v de longitud d(v). (Tarjan, 1983)

Lema 2.2.- Si p es cualquier camino desde s a cualquier nodo v, entonces *distancia*(p) $\geq$ d(v) cuando el método termina. (Tarjan, 1983)

Teorema 2.1.- El método de etiquetamiento termina si y sólo si G no contiene ciclos negativos. Si el método termina, entonces d nos da la longitud correcta y los arcos que apuntan a los padres nos dan el árbol con las longitudes correctas de la ruta más corta. (Tarjan, 1983)

Método de escaneo

El método de escaneo (Cherkassky y Goldberg, 1999) es una variante del método de etiquetado, basada en la operación de escaneo. El método mantiene para cada vértice v, además de d(v) y $\pi(v)$, el estado S(v) que puede ser: "no alcanzado", "etiquetado" o "escaneado". Al iniciar, s, el nodo origen, tiene un estado S(s) = "etiquetado", una etiqueta de longitud d(s) = 0 y un padre $\pi(s)$ = s y todos los demás vértices tienen d(v) = ∞, $\pi(v)$ = nulo y S(v) = "no alcanzado".

En cada paso el método selecciona un nodo v con S(v) = "etiquetado" para ser escaneado. La operación de escaneo de un nodo v consiste en: analizar todos los arcos (v,w) que parten de v, verificando si

$$d(v) + l(v,w) < d(w) ;$$

en caso afirmativo, la etiqueta de longitud de w se actualiza con:

$$d(w) = d(v) + l(v,w),$$

y el estado de w, S(w), se cambia por "etiquetado". Además el padre de w se cambia por $\pi(w) = v$.

Ya que terminó el análisis de cada uno de los arcos que parten de v, se cambia el estado de v por S(v) = "escaneado".

El método de escaneo termina cuando ya no hay vértices etiquetados para ser escaneados; por lo que, para cada nodo v, d(v) nos da la longitud del camino más corto de s hasta v.

El método de escaneo es correcto

La demostración de que el método de escaneo es correcto, se basa en el conocimiento de que el método de etiquetamiento es correcto: (Tarjan, 1983)

Para que un nodo v sea escaneado, éste debe tener un estado "etiquetado" por lo tanto $d(v)<\infty$; entonces, se cumple la primera condición que impone el método de etiquetamiento para seleccionar un arco. Ya que al escanear un nodo v se analizan todos los arcos que parten de v que cumplen con la condición $d(v) + l(v,w) < d(w)$, se están cumpliendo las dos condiciones que impone el método de etiquetado para seleccionar un arco. Entonces, el método de escaneo no es más que el mismo método de etiquetamiento, sólo que en este caso se seleccionan a los arcos que parten de un mismo nodo en forma consecutiva, por lo tanto, con base en el teorema 2.1, el método de escaneo es correcto.

2.2 Algoritmo de Dijkstra

El algoritmo de Dijkstra (Dijkstra, 1959) encuentra la ruta más corta desde el nodo origen s hasta los demás vértices en una red con arcos no negativos. Este algoritmo se basa en el método de escaneo, en donde los estados de los vértices serán: "no alcanzado", "etiquetado" y "permanente".

Se ha cambiado el estado "escaneado" del método de escaneo por el de "permanente"; esto se debe a que un nodo que ya haya sido escaneado no podrá ser etiquetado nuevamente, su etiqueta de longitud no podrá ser modificada y tampoco su padre. Como consecuencia, un nodo con estado "permanente" no podrá ser escaneado nuevamente.

Al principio del algoritmo, el estado del vértice origen s es de "etiquetado", su etiqueta de longitud igual a cero y su padre él mismo:

$$d(s) = 0,\ \pi(s) = s \text{ y } S(s) = \text{ "etiquetado"}.$$

Para el resto de los vértices,

$$d(v) = \infty,\ \pi(v) = \text{nulo y } S(v) = \text{"no alcanzado"}.$$

A diferencia del método de escaneo, el método Dijkstra tiene un criterio para seleccionar el siguiente nodo a escanear. Obviamente tendrá que cumplir con $S(v)$ = "etiquetado", pero deberá ser el nodo etiquetado con la mínima $d(v)$.

Una vez que termina la operación de escaneo, el estado del nodo pasa a "permanente". El algoritmo termina cuando ya no hay nodos "etiquetados" por escanear. Al terminar todos los nodos tendrán un estado "permanente".

Al escanear al nodo v cuya etiqueta de longitud es la mínima, se tiene la certeza de que $d(v)$ es exacta; siempre y cuando, los arcos sean no negativos. Ya que el algoritmo escanea sólo a vértices con $d(v)$ exacta, se considera el mejor algoritmo para encontrar la ruta más corta en redes con arcos no negativos. (Cherkassky *et al.*, 1996; Denardo y Fox, 1979).

Para ejemplificar el funcionamiento del algoritmo de Dijkstra, se usará la red que se muestra en la figura 2.1, tomando como el nodo origen *s*, al vértice 1.

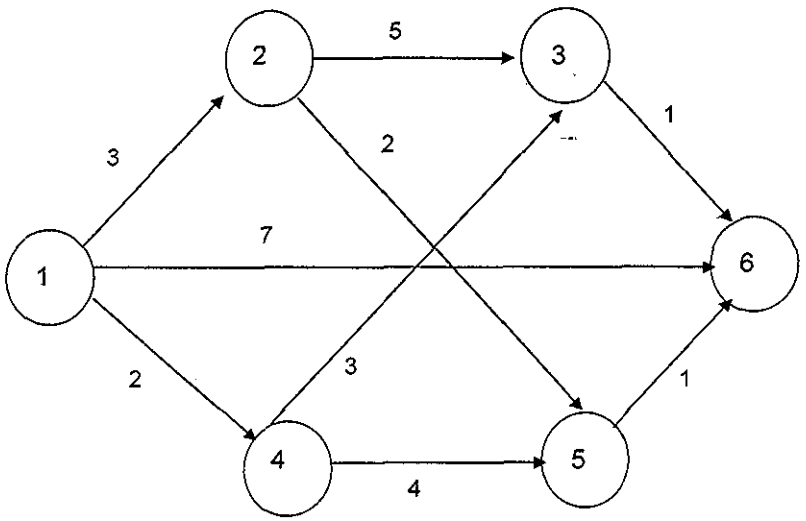


Figura 2 1

Al principio se procede a etiquetar al nodo origen, por lo que su estado es de "etiquetado" ("e"). A los demás nodos se les asignan padres nulos, etiquetas de longitudes iguales a infinito y su estado es "no alcanzado" ("na"). Los datos de los nodos que se muestran en la figura 2.2, se conforman de: [padre ($(\pi(v)$), etiqueta de longitud (d(v)), estado (S(v))]

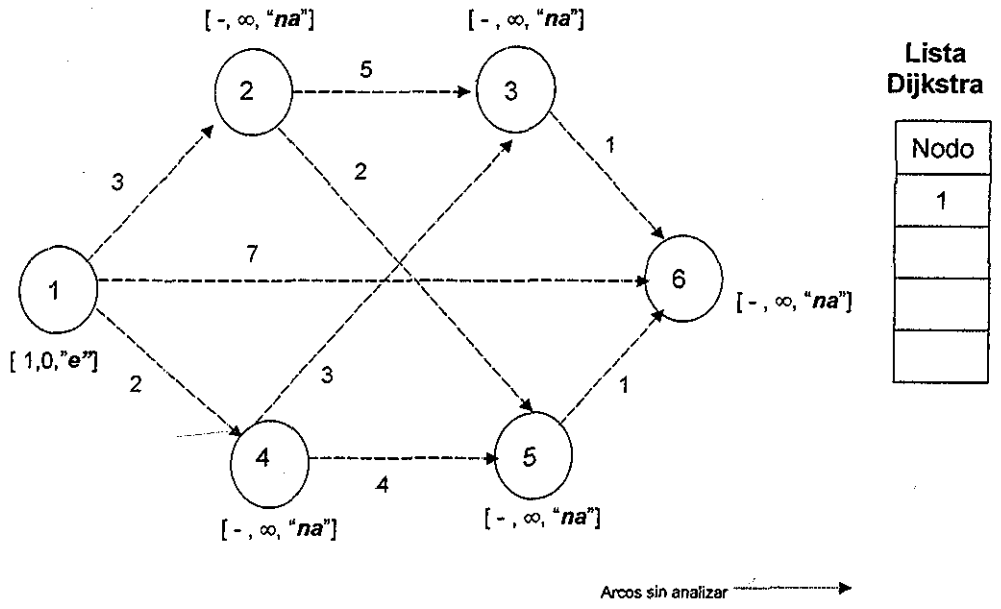


Figura 2 2

Se escoge el vértice etiquetado con la etiqueta de longitud d(v) menor, que como se puede observar en la figura 2.2, el único nodo con estado "etiquetado" es el nodo origen s, por lo que el vértice 1 se escanea, quedando:

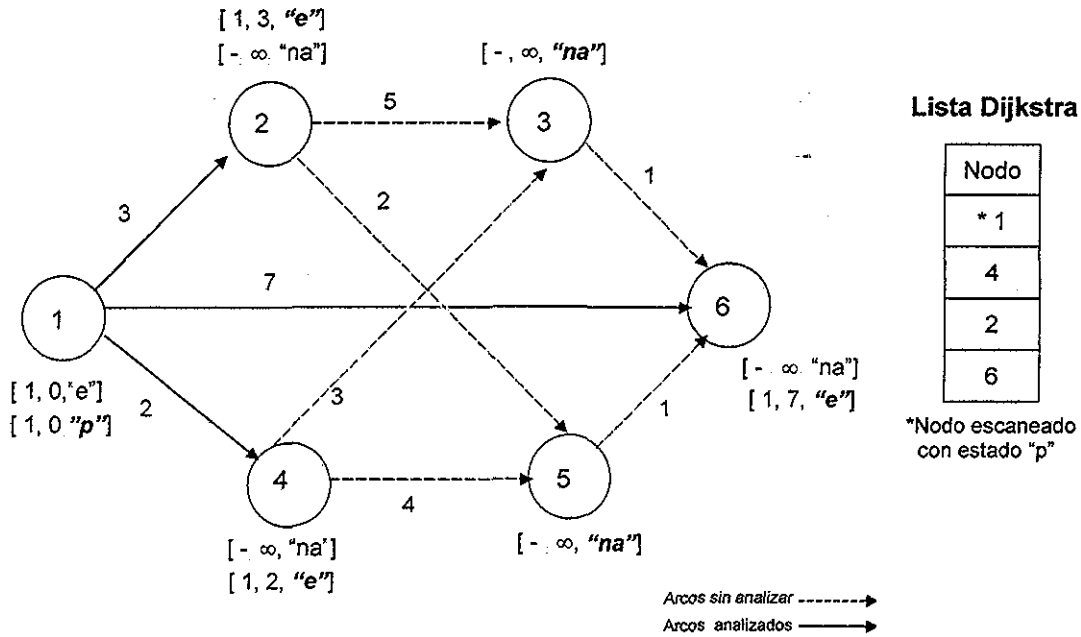


Figura 2.3

Como se muestra en la figura 2.3, el estado del vértice origen, una vez que fue escaneado, su estado cambia a "permanente" ("p"). Posteriormente se escoge el vértice etiquetado con d(v) mínima (nodo 4) y se escanea, como se muestra en la figura 2.4.

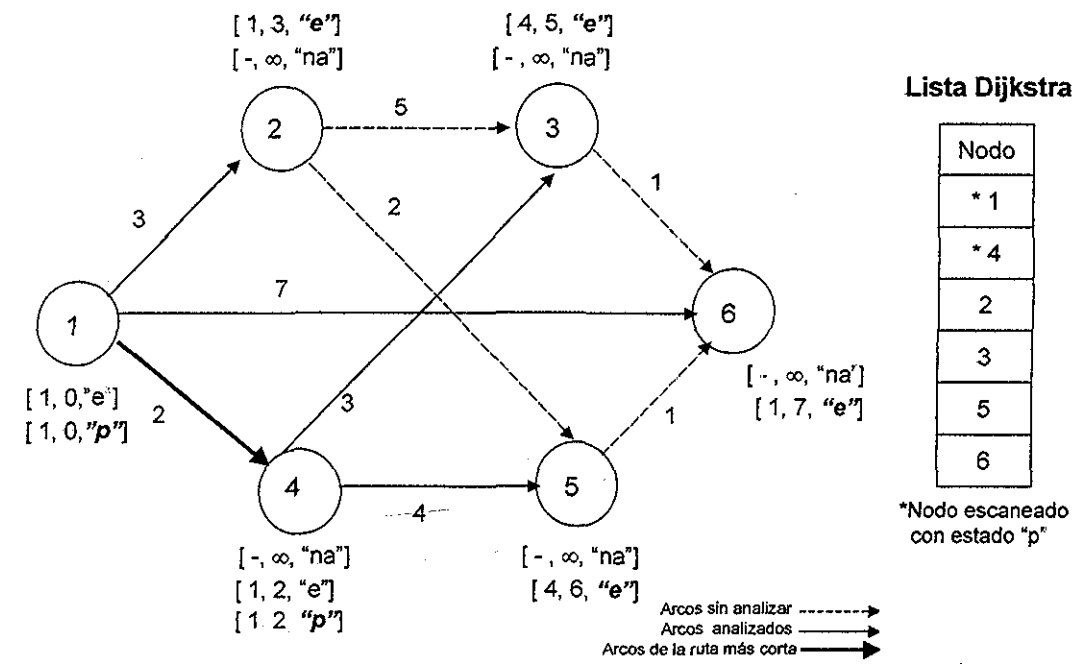


Figura 2.4

En los siguientes pasos, se continúa el mismo procedimiento, hasta que no haya nodos "etiquetados", como se muestra en la figura 2.5. Observe cómo un vértice puede cambiar su etiqueta de longitud y su padre varias veces, siempre y cuando su estado no sea "permanente".

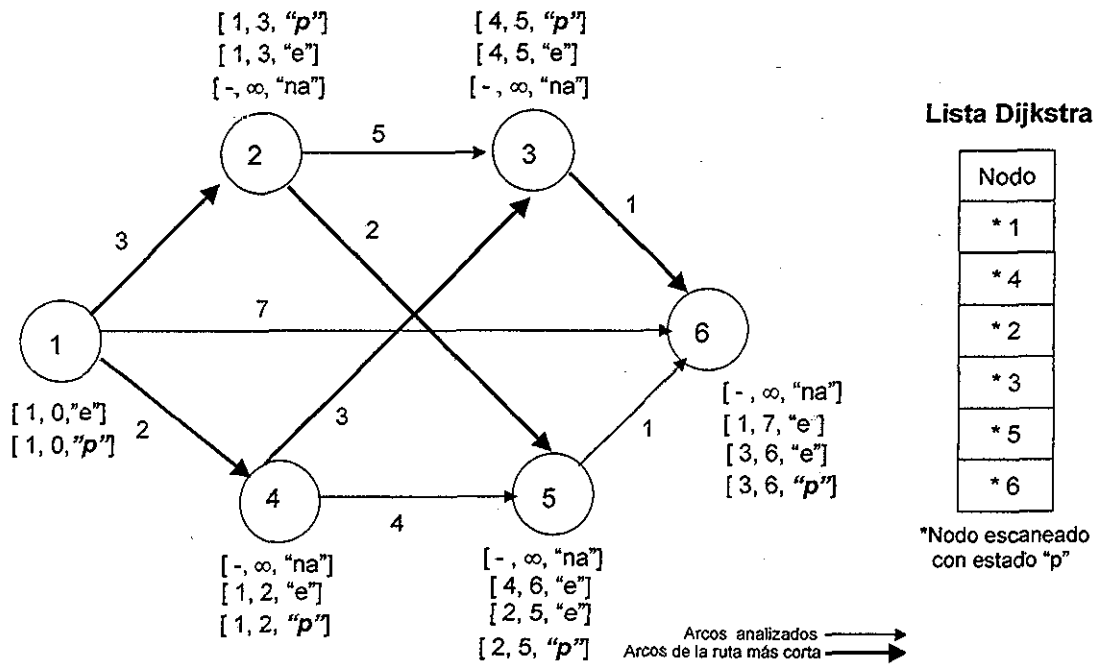


Figura 2.5

Al tener todos los nodos un estado diferente al "etiquetado" se llega a la solución. El árbol resultante se forma con los arcos que vienen de los padres de cada uno de los nodos con estado "permanente" como se muestra en la figura 2.6.

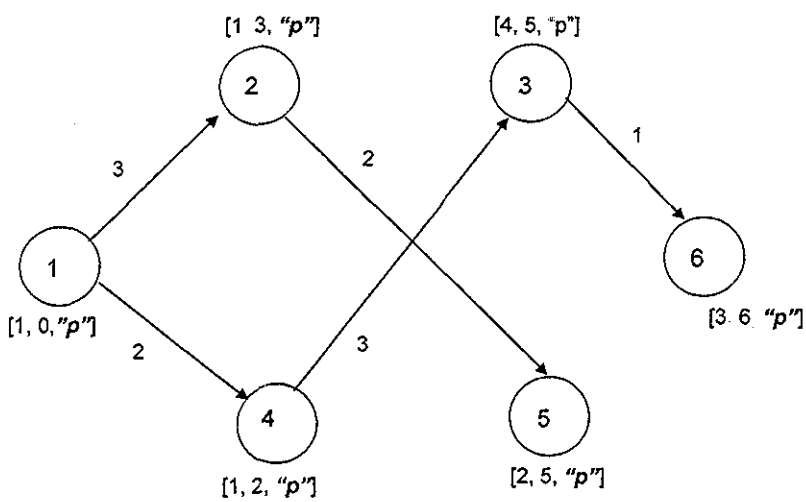


Figura 2.6

La funcionalidad del algoritmo se basa en la forma de ordenar los vértices etiquetados para escoger de forma eficiente al vértice con la etiqueta de longitud d(*v*) mínima. Cuando se tienen redes más grandes, el método de ordenamiento es muy importante para que el algoritmo sea eficiente.

2.2.1 Redes cuyos arcos tengan longitudes negativas

Para resolver redes cuyos arcos tienen longitudes negativas, el algoritmo de Dijkstra no funciona. Para entender el por qué, retomemos el ejemplo anterior y veamos qué sucede si a la red se le agrega el arco (2, 4) con una longitud de –4, como se muestra en la figura 2.7

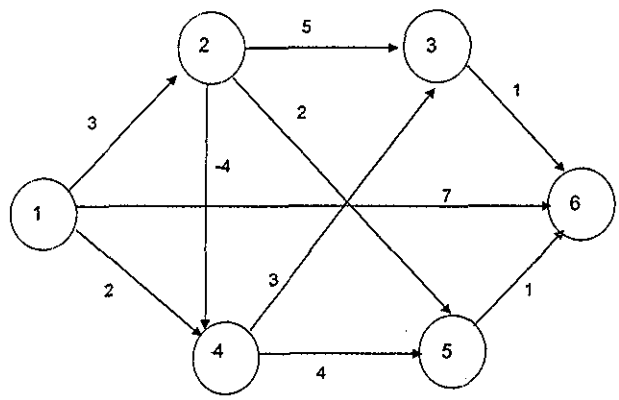


Figura 2.7

Aplicando el algoritmo de Dijkstra, el segundo vértice con estado "permanente" es el vértice 4, como se muestra en la figura 2.8.

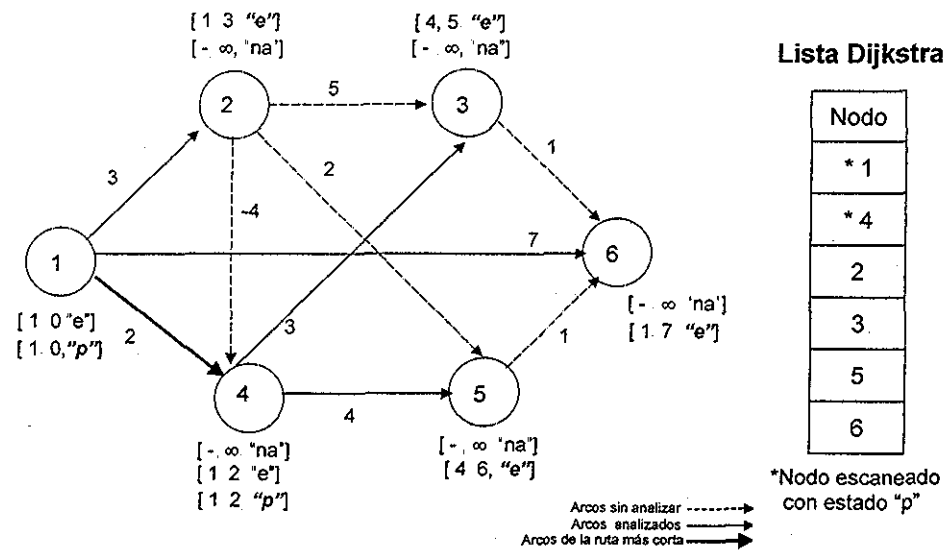


Figura 2.8

Sin embargo, la etiqueta de longitud con la cual quedó el vértice 4, d(4)=2, no es exacta. Es claro que la ruta de longitud mínima para llegar al nodo 4 es 1->2->4 y el nodo debería de quedar con una etiqueta de longitud d(4)=-1. Al continuar con el algoritmo de Dijkstra, el resultado final no nos entregará la solución correcta del problema de ruta más corta.

Al tener longitudes de arco negativas, ya no se tiene la certeza de que el nodo con etiqueta de longitud d(v) mínima, que se selecciona para ser escaneado, tenga una d(v) exacta. Ya que el algoritmo no permite que un nodo con etiqueta "permanente" pueda ser escaneado nuevamente, el algoritmo de Dijkstra no funciona para redes cuyos arcos tengan longitudes negativas.

El algoritmo de Dijkstra es incorrecto para redes con arcos negativos

Al tener un estado *"permanente"* y tener arcos con longitudes negativas pueden quedar nodos que deberían de ser *"etiquetados"*, y por lo tanto reescaneados, sin escanearse. De modo que, al terminar el algoritmo, podría no cumplirse la condición de terminación que marca el método de escaneo.

El error también podría verse desde el punto de vista del método de etiquetamiento, ya que al terminar el algoritmo de Dijkstra, podrían quedar arcos que cumplen con las condiciones de selección sin analizarse; incumpliéndose la condición de terminación del método de etiquetamiento. (Tarjan, 1983)

La variación típica al algoritmo de Dijkstra, usada por Goldberg y Cherkassky en 1999, permite que se resuelvan redes con arcos negativos. Consiste en eliminar el estado "permanente" y en su lugar usar el estado "escaneado", permitiendo que un nodo pueda ser escaneado más de una vez. Con esta variación, el algoritmo de Dijkstra entrega una solución correcta; sin embargo, esta variación resulta en un algoritmo sumamente ineficiente. (Goldberg y Cherkassky 1999)

En el siguiente capítulo se comentarán los algoritmos más eficientes según el estudio realizado por Goldberg y Cherkassky en 1999, para el problema de ruta más corta en redes con longitudes de arco negativas. Y en el capítulo 4 se muestra la modificación propuesta al algoritmo de Dijkstra, para resolver, de forma eficiente este tipo de problemas.

III ALGORITMOS PARA REDES CON LONGITUD DE ARCOS NEGATIVAS

En este capítulo, se describen los algoritmos más eficientes que se conocen para resolver el problema de ruta más corta en donde las longitudes de los arcos pueden ser negativas.

Uno de los algoritmos más conocidos para resolver este tipo de problemas, se debe a Bellman (Bellman, 1958), Ford (Ford, 1962) y Moore (Moore, 1959). Este algoritmo se toma como base para la implementación de nuevos algoritmos.

Un algoritmo que se basa en el algoritmo de Bellman-Ford-Moore es el propuesto por Tarjan (Tarjan, 1981), que es una combinación del algoritmo de Bellman-Ford–Moore y una estrategia de disgregación del subárbol. La estrategia de disgregación del subárbol nace de la necesidad de detectar ciclos negativos en una red.

El algoritmo de Goldberg y Radzik (Goldberg y Radzik, 1993), es una mejora del algoritmo de Bellman-Ford-Moore. Goldberg y Radzik proponen utilizar un ordenamiento de los vértices, el cual llaman ordenamiento topológico, para seleccionar el nodo a ser escaneado.

Otra mejora al algoritmo de Bellman-Ford-Moore es una especialización del método Simplex para redes (Dantzing, 1951), el cual actualiza la etiqueta de longitud de los descendientes de un nodo recién etiquetado mediante un procedimiento conocido como pivoteo.

Es importante mencionar que en problemas con longitudes de arco negativas pueden surgir ciclos con longitud total negativa. Si una red tiene ciclos negativos, no se puede encontrar una solución, ya que la supuesta solución sería recorrer el ciclo negativo un infinito número de veces. Debido a esto, se debe contar con estrategias para la detección de ciclos negativos. El algoritmo de Tarjan, el Simplex y el de Goldberg y Radzik cuentan con su propia estrategia para la detección de estos ciclos. (Cherkassky *et al.*, 1999)

Las tres mejoras del algoritmo de Bellman-Ford-Moore mencionadas anteriormente, resultaron ser los tres algoritmos más eficientes en la evaluación reportada por Cherkassky y Goldberg (1999).

3.1 Algoritmos

A continuación se hace una descripción de los algoritmos de: Bellman-Ford-Moore, Tarjan, Goldberg-Radzik y por último el Simplex.

3.1.1 Algoritmo de Bellman-Ford-Moore

Este algoritmo, es el más conocido para aplicarse cuando existen redes con longitudes de arcos negativas, se debe a Bellman, Ford y Moore (Bellman, 1958; Ford y Fulkerson, 1962; Moore 1959). El algoritmo se basa en el método de escaneo en donde los estados de los nodos serán "no alcanzado", "etiquetado" y "escaneado".

Al principio del algoritmo, el estado del vértice origen *s* es "etiquetado", su etiqueta de longitud es igual a cero y él es su mismo padre; los demás nodos tienen estado "no alcanzado", etiqueta de longitud igual a infinito y padre nulo.

Este algoritmo mantiene al conjunto de vértices con estado "etiquetado" en una cola FIFO (First In First Out), en la cual el primer nodo en llegar es el primero en ser escaneado. El siguiente vértice a ser escaneado, se obtiene del principio de la cola. Al inicio del algoritmo, el primer y único elemento de la cola es el nodo origen *s*.

Si durante la operación de escaneo de un nodo *u* se analiza a un arco (u,v) con costo reducido negativo ($l_d = l(u,v) + d(u) - d(v) < 0$), se provoca una mejora en la etiqueta de longitud del nodo destino, de modo que *v* se *actualiza* ($d(v) = l(u,v) + d(u)$, $\pi(v) = u$), se marca con $S(v)$="etiquetado" y se agrega al final de la cola, si es que no se encuentra en la misma.

Si algún nodo *v* con estado "escaneado" mejora su etiqueta de longitud $d(v)$, se puede "etiquetar" nuevamente y se agrega al final de la cola. El algoritmo termina cuando la cola queda vacía.

El límite teórico del tiempo de ejecución del algoritmo de Bellman-Ford-Moore es $O(nm)$ (Tarjan, 1983)

El defecto que tiene este algoritmo, al igual que la modificación típica del algoritmo de Dijkstra mencionada al final del capítulo anterior, es que puede escanear muchas veces a un mismo vértice antes de llegar a su etiqueta de longitud exacta; aunque es mucho más eficiente que la modificación típica del Dijkstra (Cherkassky *et al.*, 1996). Al tener redes grandes, el tiempo de ejecución puede ser muy tardado. Para prevenir escanear muchas veces a un sólo vértice, existen algunos criterios heurísticos, los cuales son la base de las mejoras al algoritmo de Bellman-Ford- Moore.

Este algoritmo no cuenta con una estrategia de detección de ciclos negativos.

3.1.1.1 Ejemplo del algoritmo de Bellman-Ford–Moore

Para ejemplificar cada uno de los algoritmos de este capítulo se toma como ejemplo la misma red, los datos de la red se presentan a continuación al igual que su gráfica.

Datos de la red para el ejemplo

Vértice	Arcos	Longitud del arco
1	(1,2) (1,3) (1,4)	-6 -4 3
2	(2,7)	-6
3	(3,2) (3,5) (3,6) (3,7)	-3 2 -2 -4
4	(4,3) (4,5)	-8 -5
5	(5,6)	2
6	(6,7)	1
7	———	———

Tabla 4.1.1

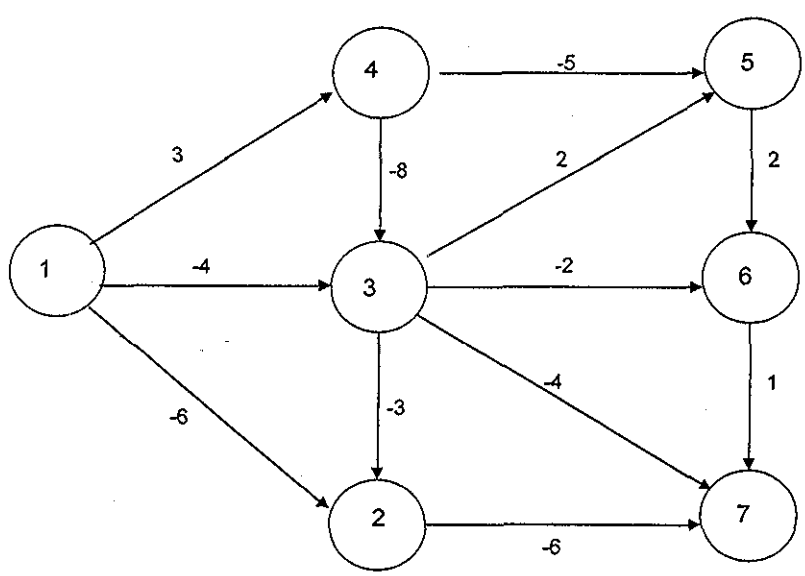


Figura 3.1.1.1

Al inicio del algoritmo se marca a todos los vértices con padre nulo, etiqueta de longitud igual a infinito y estado "no alcanzado" ("na"), excepto al vértice origen que para nuestro ejemplo es el vértice 1. Como se muestra en la figura 3.1.1.2.

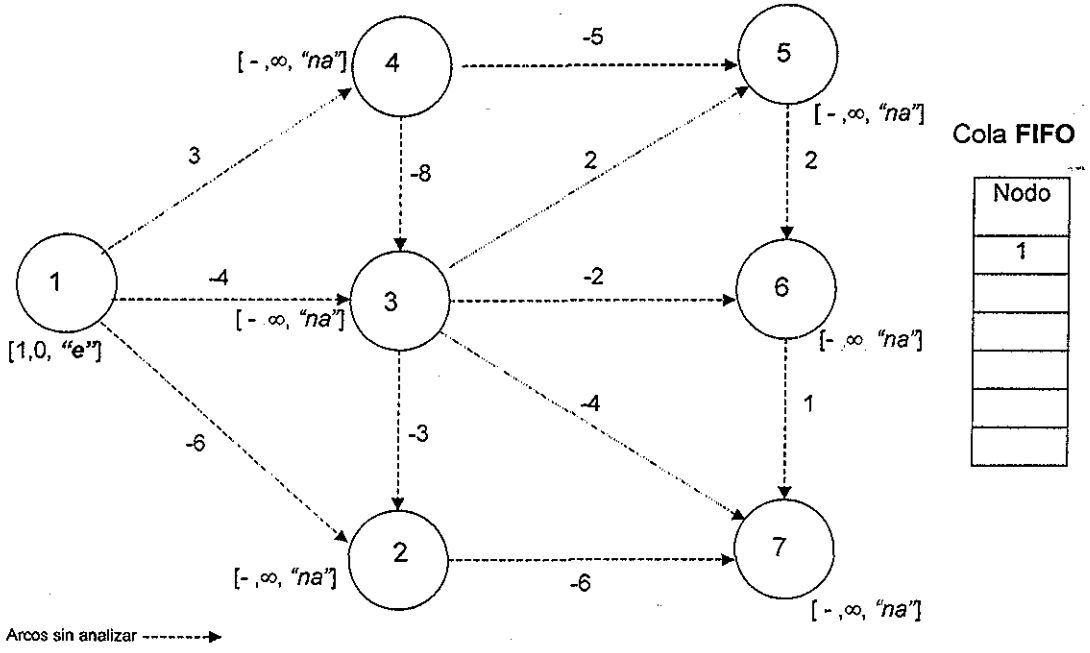


Figura 3.1.1.2

Al inicio del algoritmo el único vértice con estado "etiquetado" ("e") es el nodo origen y es el primer elemento que forma parte de la cola FIFO y se procede a escanearlo. Una vez que el nodo se escanea su estado cambia a "escaneado" ("s") como se muestra en la siguiente figura.

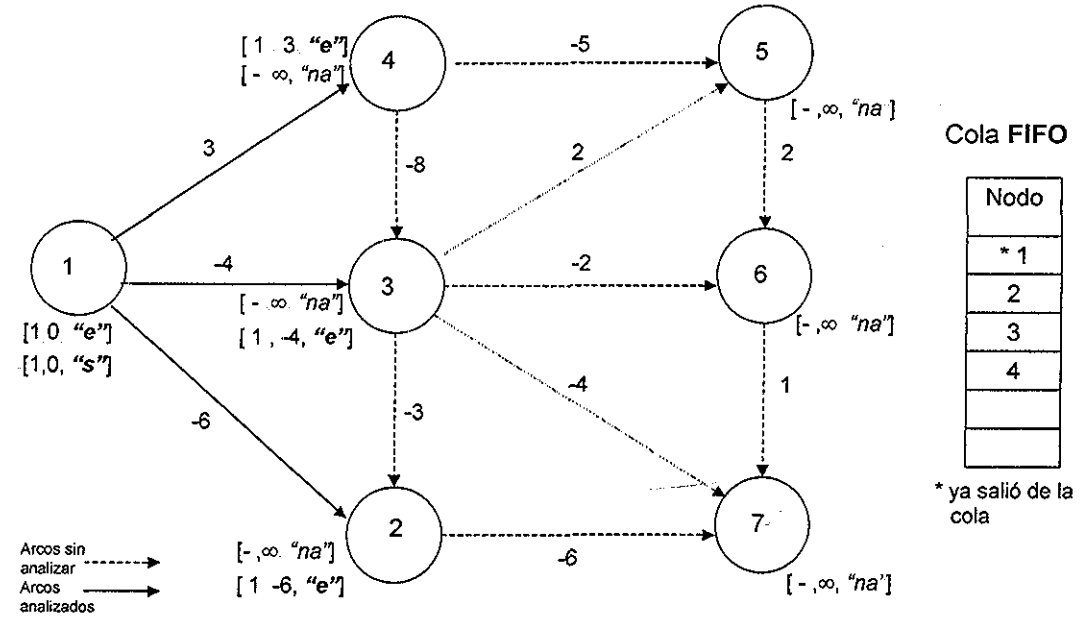


Figura 3.1.1.3

Al escanearse el nodo origen los nuevos elementos cuyo estado cambió a "etiquetado" entran a la cola. Siendo así el primer elemento de la cola el nodo 2 seguido por el nodo 3 y por último el nodo 4 (esto debido al orden en el cual se leen los arcos de la lista original: *tabla 4.1.1*). Se escoge el primer elemento de la cola para ser el siguiente nodo a escanearse, para nuestro ejemplo es el nodo 2. Una vez escaneado su estado cambia a "escaneado" y sale de la cola. Como se muestra en la figura 3.1.1.4.

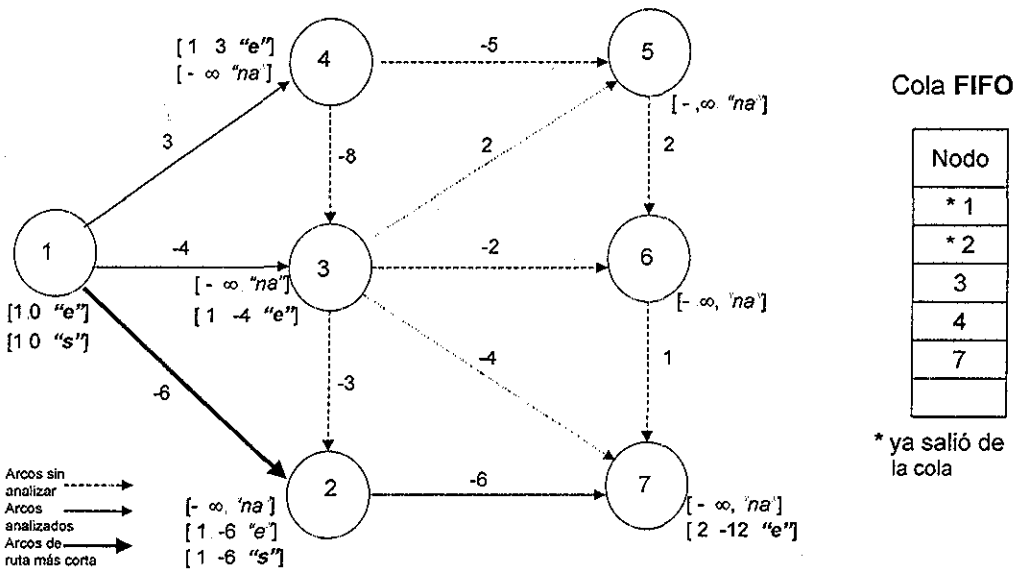


Figura 3.1.1.4

Si el estado de un vértice es "escaneado" y llega a él un arco con costo reducido negativo, el estado del vértice cambiará a "etiquetado". En el ejemplo, al escanear al nodo 3 con d(*3*) = -4, provoca que d(*2*) mejore. El nodo 2 se *actualiza*, su estado cambia a "etiquetado" y entra como último elemento de la cola. Ver figura 3.1.1.5.

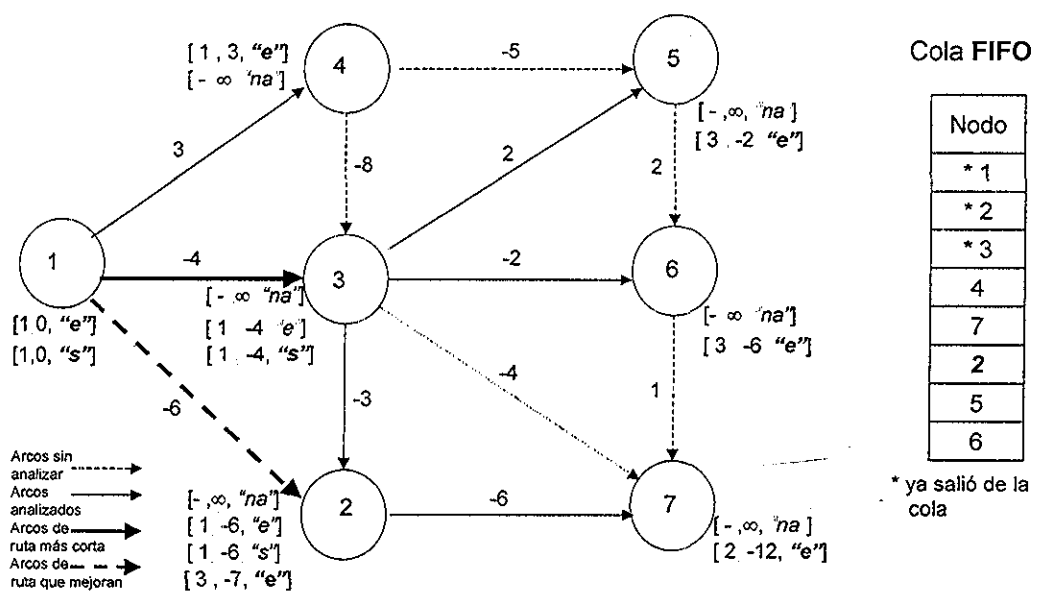


Figura 3.1.1.5

En la figura 3.1.1.6. se muestran los diferentes estados que tomaron los nodos hasta obtener el resultado final. También se muestra la cola FIFO formada desde el inicio hasta el final del algoritmo. El algoritmo termina hasta que la cola FIFO quede vacía.

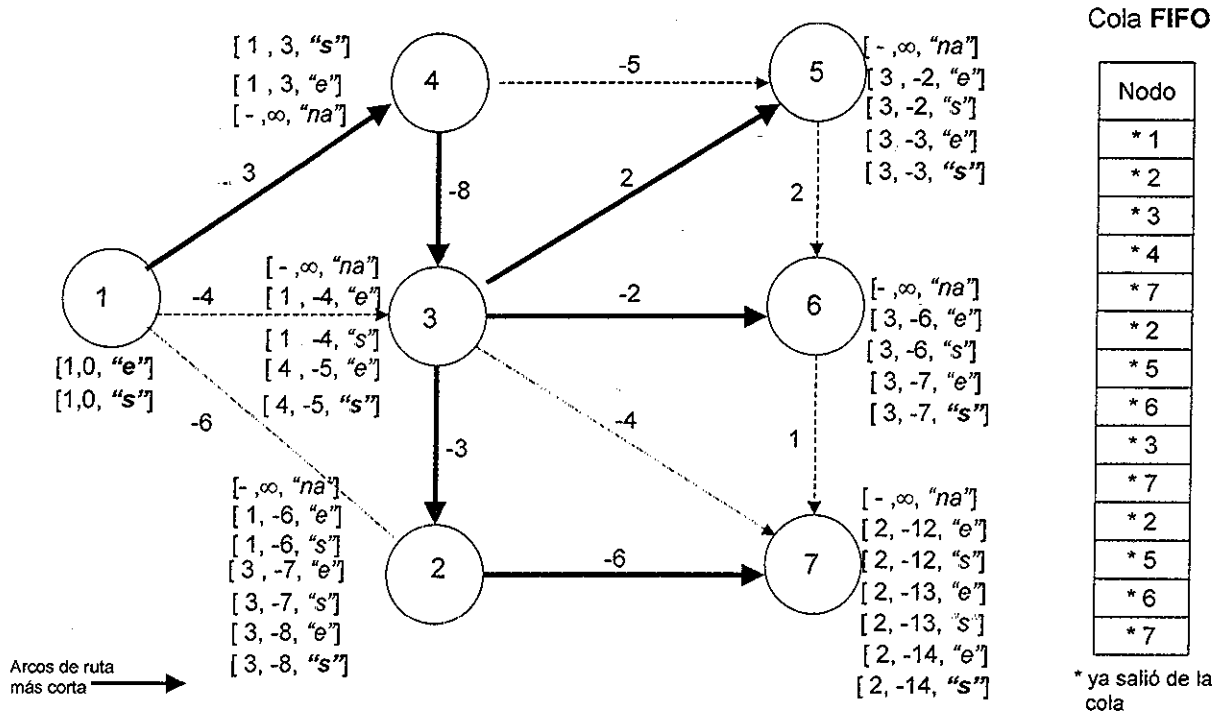


Figura 3.1.1.6

El resultado final se muestra en la figura 3.1.1.7.

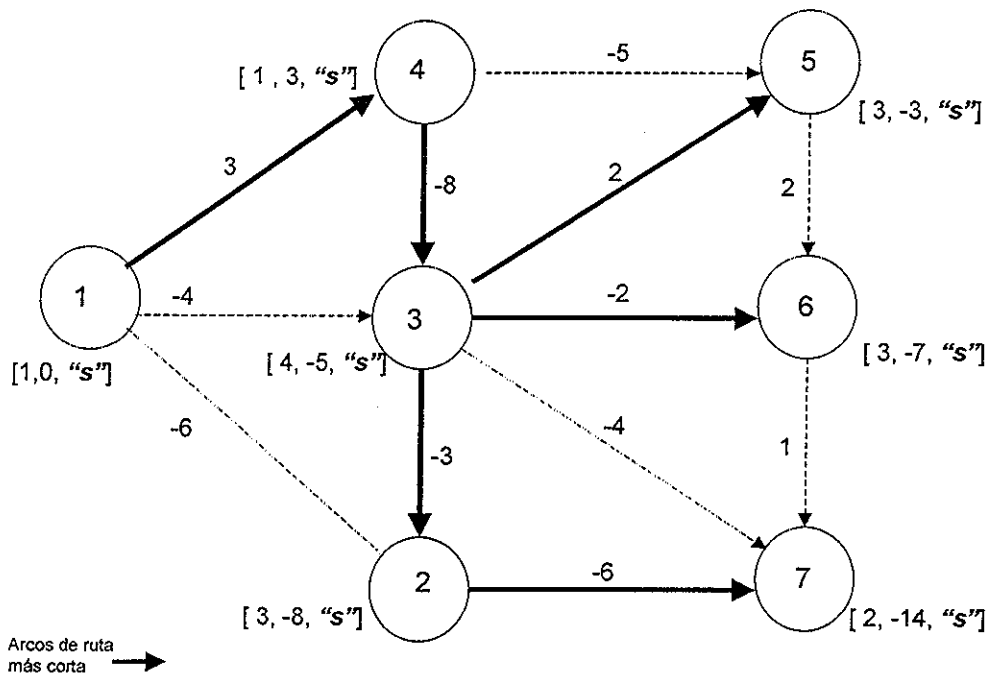


Figura 3.1.1.7

3.1.2 Algoritmo de Tarjan

Este algoritmo propuesto por Tarjan (Tarjan, 1981) es una variación del algoritmo de Bellman-Ford-Moore, donde algunos vértices con estado "etiquetado" o "escaneado" cambian su estado a "no alcanzado"; de modo qué, existirán vértices "no alcanzados" con etiquetas de longitud finitas pero padres nulos. Las etiquetas de longitud de estos vértices, eran inexactas de cualquier manera.

Al principio del algoritmo, el estado del vértice origen *s* es "etiquetado", su etiqueta de longitud es igual a cero y él es su mismo padre; los demás nodos tienen estado "no alcanzado", etiqueta de longitud igual a infinito y padre nulo.

Este algoritmo también mantiene al conjunto de vértices con estado "etiquetado" en una cola FIFO. El siguiente vértice a ser escaneado, se obtiene del principio de la cola. Al inicio del algoritmo, el primer y único elemento de la cola es el nodo origen *s*.

Si durante la operación de escaneo de un nodo *u* se analiza a un arco (u,v) con costo reducido negativo, se *actualiza* el nodo *v* y se marca con $S(v)$="etiquetado" agregándose al final de la cola, si es que no se encuentra en la misma. Además, se recorre **todo** el subárbol con raíz en *v* modificando las características de cada uno de sus descendientes a: S = "no alcanzado", π = nulo y se sacan de la cola, si es que se encuentran en ella. A esto se le llama *disgregación del subárbol de v*. Observe que todos los descendientes de *v* quedan huérfanos.

El algoritmo termina cuando la cola queda vacía. El límite teórico del tiempo de ejecución es de $O(nm)$. (Tarjan, 1983)

A pesar de que los nodos también pueden ser escaneados varias veces, al disgregar el subárbol del nodo mejorado disminuye el número de escaneos. La disminución se debe a que los descendientes no se volverán a escanear hasta que su ancestro, que ha sido mejorado, se escanee y provoque que entren a la cola nuevamente.

Si durante la disgregación del subárbol de un nodo *v*, se encuentra que *v* pertenece a su propio subárbol, entonces existe un ciclo negativo en la red y el algoritmo termina indicando que encontró un ciclo negativo.

3.1.2.1 Ejemplo del algoritmo de Tarjan

Como se comentó anteriormente para ejemplificar este algoritmo se usará la misma red que para el algoritmo anterior, figura 3.1.1.1.

Al inicio del algoritmo se marcan a todos los vértices con padre nulo, etiqueta de longitud igual a infinito y estado "na", excepto al vértice origen. Como se observa en la figura 3.1.2.1

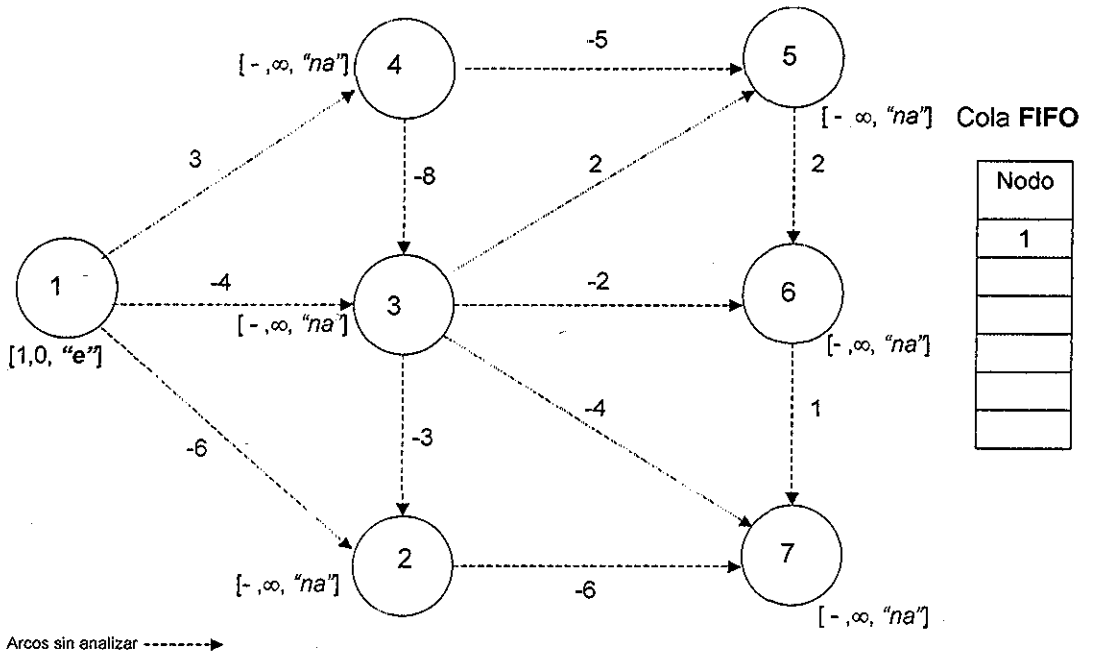


Figura 3.1.2.1

Al igual que el algoritmo de Bellman-Ford-Moore, el algoritmo de Tarjan utiliza una cola, en la cual el único elemento que se encuentra en ella al inicio del algoritmo es el nodo origen por tener un estado "etiquetado". Se procede a escanearlo. Una vez escaneado su estado cambia a "escaneado" y sale de la cola como se muestra en la siguiente figura:

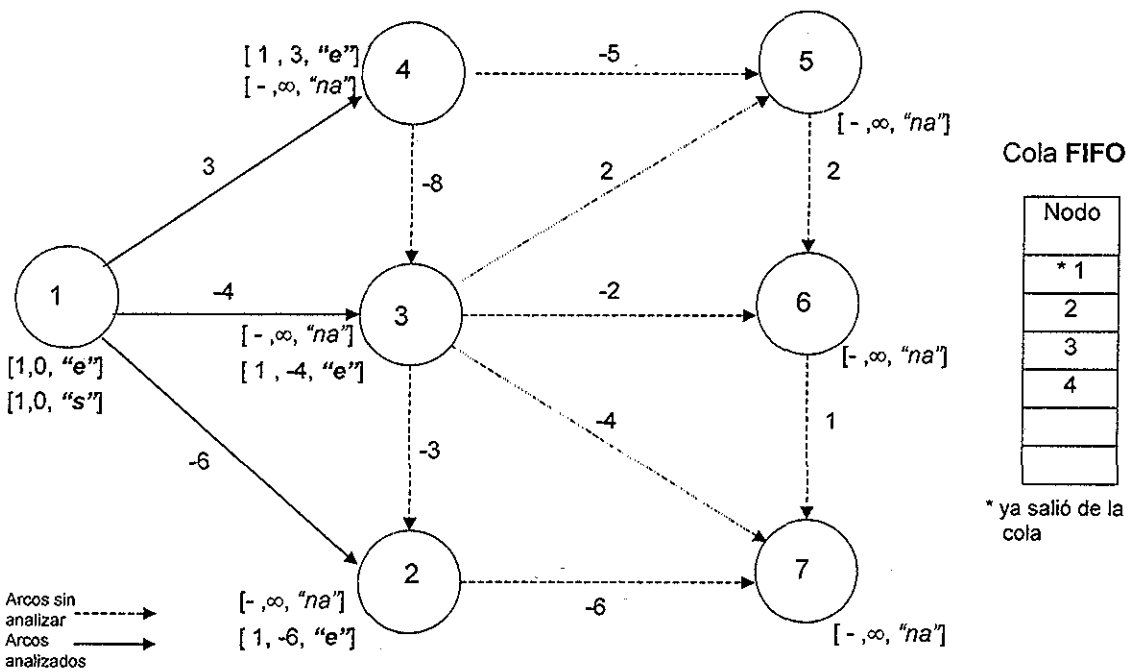


Figura 3.1.2.2

El siguiente nodo a escanear es el nodo 2 por ser el primero de la cola seguido por el nodo 3 y el nodo 4. Al escanear el nodo 3, se encuentra que el arco (3,2) tiene costo reducido negativo por lo que se *actualiza* al nodo 2, su estado cambia a "etiquetado" y se agrega al final de la cola. Además, se recorre el subárbol del nodo 2, que para el ejemplo

solo se conforma por el nodo 7, se saca de la cola, se cambia su estado a *"no alcanzado"* y se le pone padre *nulo*, como se muestra en la figura 3.1.2.3.

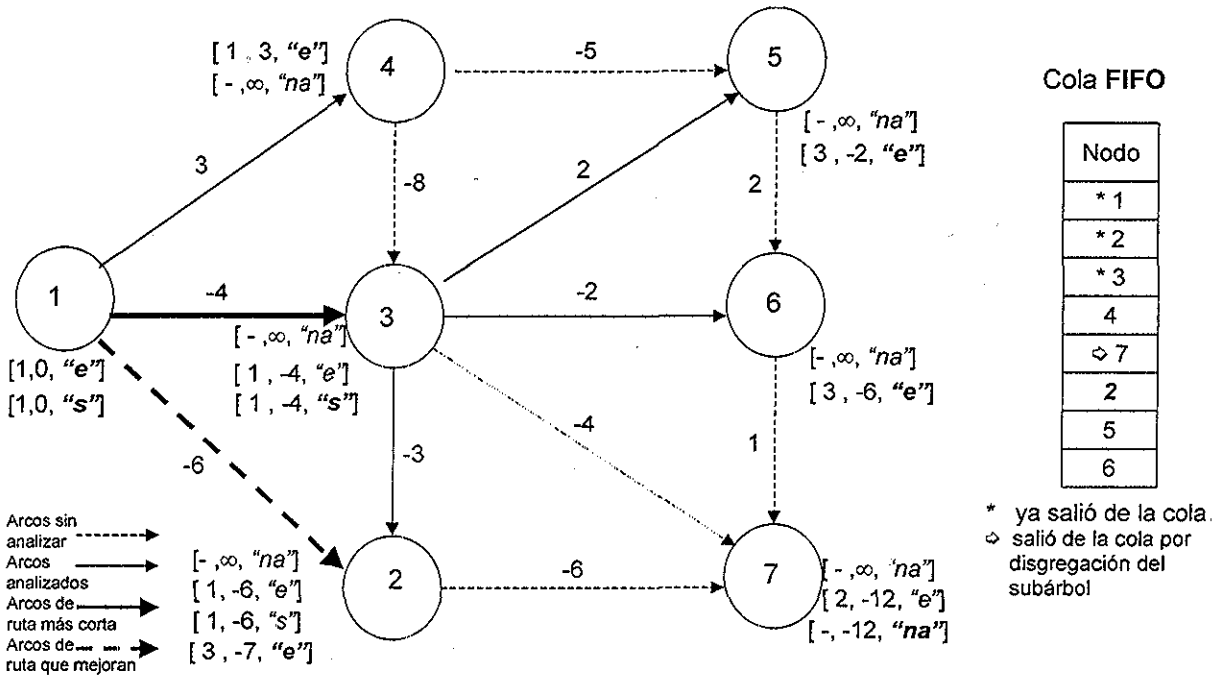


Figura 3.1.2.3

Al escanear el nodo 4 se presenta el mismo caso anterior. Se debe *actualizar* al nodo 3 debido que el arco (4,3) tiene costo reducido negativo y deben buscarse todos los descendientes del nodo 3 (nodo 2, 5 y 6) para sacarlos de la cola, cambiar sus estados a "no alcanzado" y ponerles padres nulos, como se muestra en la figura 3.1.2.4.

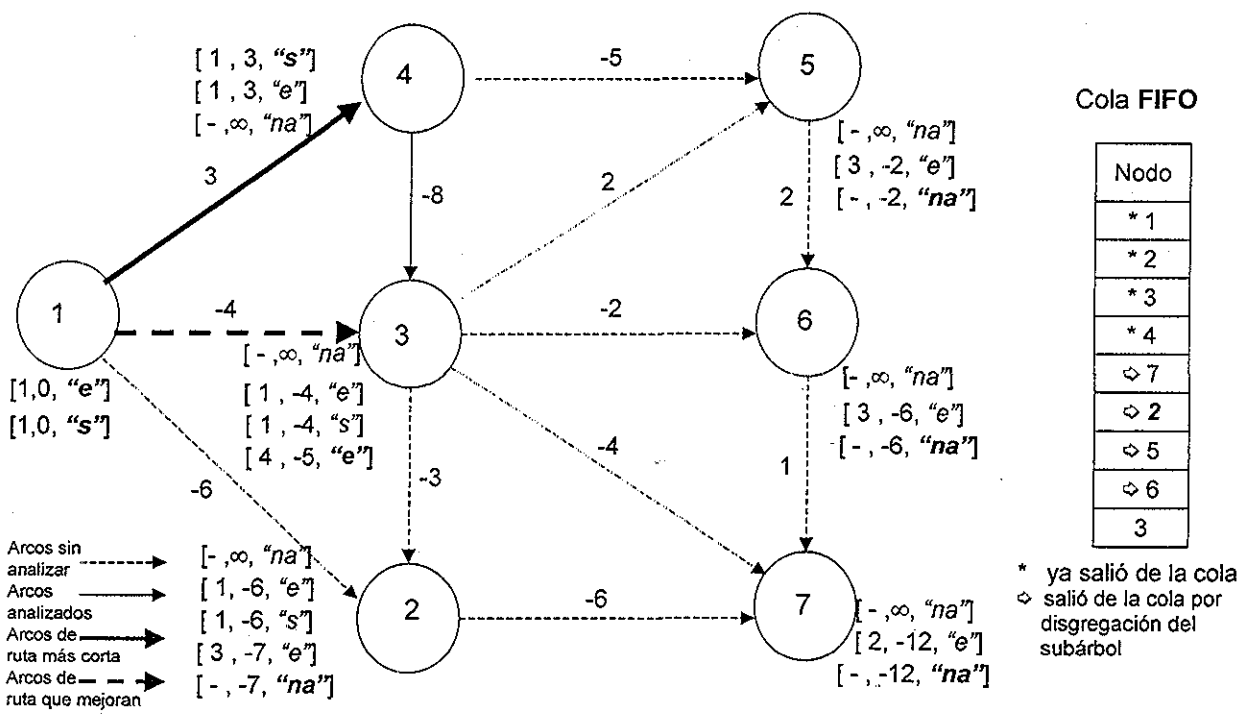


Figura 3.1.2.4

En la figura 3.1.2.5. se muestran los diferentes estados que tomaron los nodos en el algoritmo para obtener el resultado final. También se muestra la cola FIFO formada desde el inicio hasta el final del algoritmo. El algoritmo termina hasta que la cola FIFO este vacía. Se observa que el número de escaneos se redujo en comparación al Bellman-Ford-Moore.

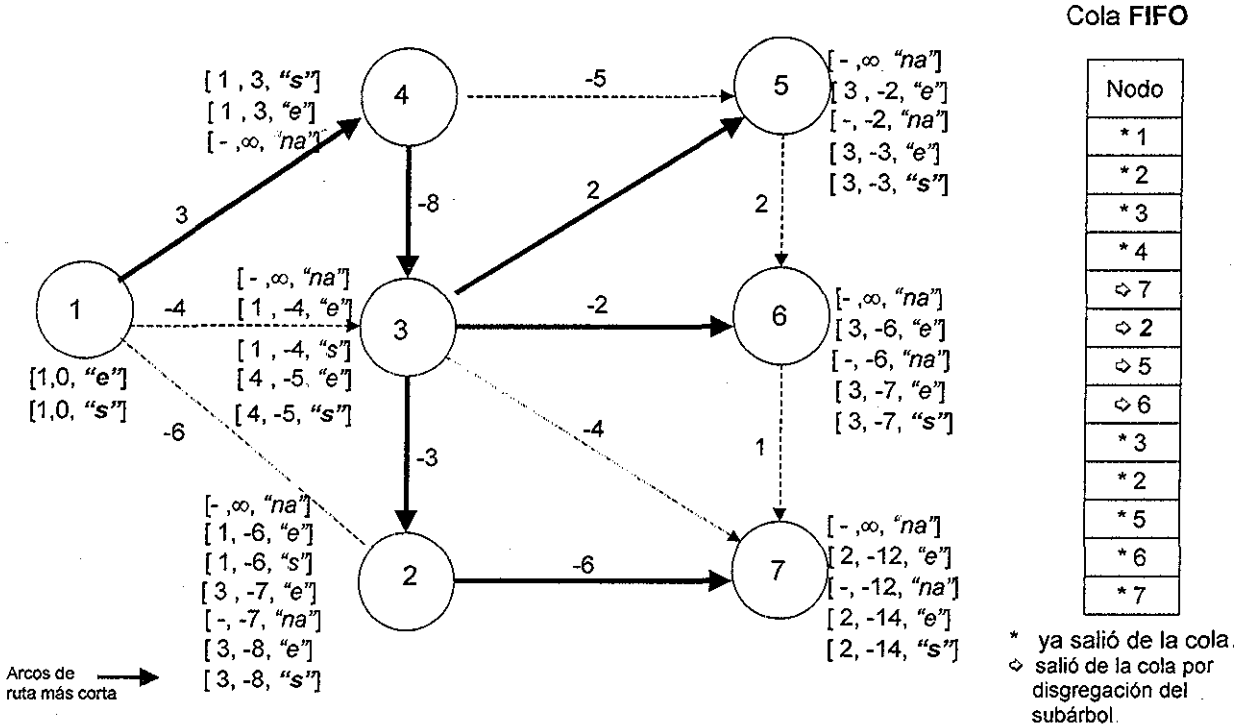


Figura 3 1.2 5

La gráfica resultante se muestra en la figura 3.1.2.6.

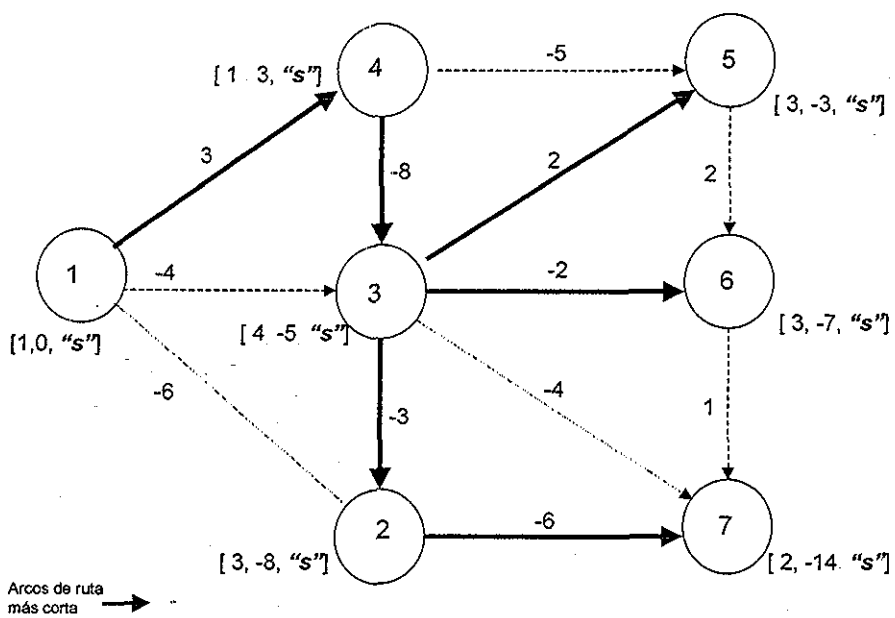


Figura 3 1 2.6

3.1.3 Algoritmo de Goldberg–Radzik

Este algoritmo se debe a Goldberg y Radzik (Goldberg y Radzik, 1993), quienes sugirieron una mejora al algoritmo de Bellman-Ford-Moore.

El algoritmo mantiene al conjunto de vértices etiquetados en dos subconjuntos, A y B. Inicialmente A = ∅ y B = {*s*} y nuevamente S(*s*)="etiquetado", $\pi(s)=s$ y d(*s*)=0.

Al principio de cada *paso*, el algoritmo utiliza al subconjunto B para calcular al subconjunto A, en donde quedarán los nodos que serán escaneados durante un *paso* y B quedará vacío.

El subconjunto A se calcula a partir de B, de la siguiente forma:

1.- Para cada $v \in B$ que no tenga algún arco con costo reducido negativo, se saca al nodo *v* de B y su estado se cambia a "escaneado".

2.- Se colocan en A todos los vértices que se puedan alcanzar desde los nodos que quedaron en B por medio de arcos admisibles, es decir, a través de la gráfica admisible G_d. El estado de todos los vértices que queden en A se cambia a "etiquetado".

3.- Se aplica un ordenamiento topológico al subconjunto A, de modo que, para cada par de nodos *v* y *w* para los que exista un arco (*v,w*) admisible, *v* quede antes que *w*. Así *v* será escaneado antes que *w*.

Durante un *paso*, se sacan los elementos de A siguiendo el orden y se escanean. Los vértices que sean *actualizados* y se encuentren fuera de A, se marcan como "etiquetados" y se agregan al subconjunto B. Un *paso* termina cuando A quede vacío. El algoritmo termina cuando B quede vacío al final de un *paso*.

El límite teórico del tiempo de ejecución del algoritmo de Goldberg-Radzik es de O(*nm*). (Goldberg y Radzik, 1993)

Si durante el ordenamiento de los nodos del subconjunto A se observa que un nodo *u* deberá estar antes que un nodo *v* y también después, teniendo un costo reducido negativo en algún arco en la ruta de *u* hacia sí mismo, entonces existe un ciclo negativo en la red y el algoritmo termina indicando que encontró un ciclo negativo.

3.1.3.1 Ejemplo del algoritmo de Goldberg-Radzik

Como se comentó anteriormente para ejemplificar este algoritmo se usará la misma red que para el algoritmo de Bellman-Ford-Moore y Tarjan, como se muestra en la figura 3.1.3.1.

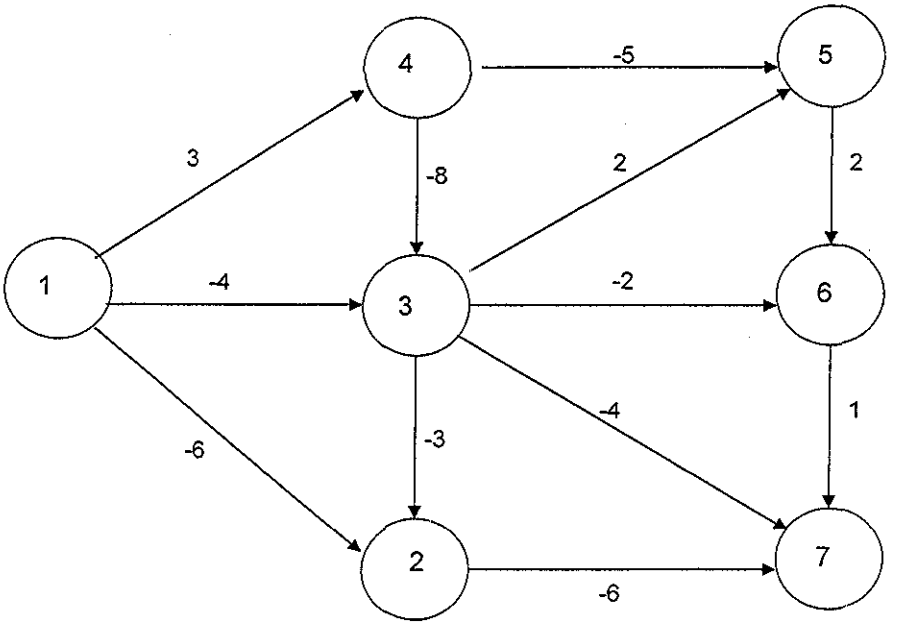


Figura 3.1.3. 1

Al inicio del algoritmo se marcan a todos los vértices con padre nulo, etiqueta de longitud igual a infinito y estado "na", excepto al vértice origen. Como se observa en la figura 3.1.3.2.

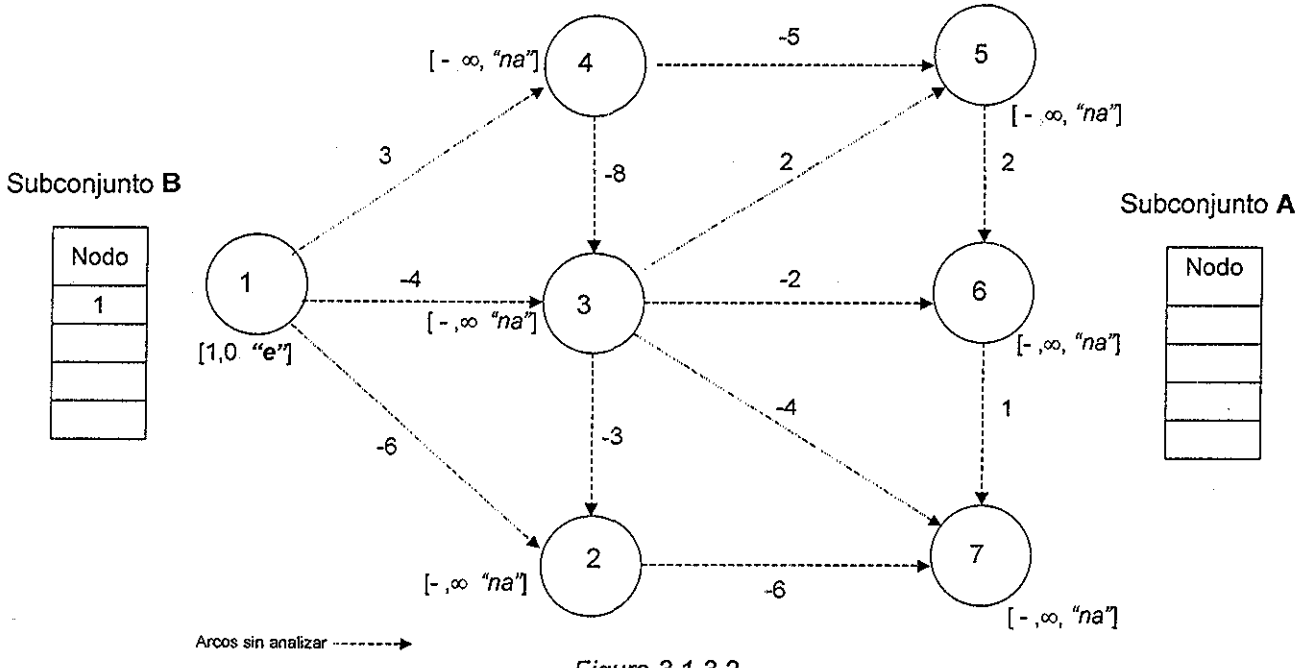


Figura 3.1.3.2

Al principio del algoritmo el nodo origen es el único elemento que forma parte del subconjunto B, el cual se utiliza para determinar los elementos del subconjunto A.

A partir del nodo origen se toma al primero de sus descendientes cuyo arco sea admisible; para nuestro ejemplo, el primer descendiente con arco admisible es el nodo 2.

El siguiente paso es verificar si el nodo 2 tiene descendientes con arcos admisibles, si es el caso, se escoge al primero de ellos, para nuestro caso se escoge al nodo 7 y así sucesivamente. Este procedimiento tiene el objetivo de formar al subconjunto A.

En la siguiente figura se muestra aquellos vértices que se alcanzaron desde el nodo origen por medio de arcos admisibles y forman parte del subconjunto A. Es importante mencionar que el estado de todos los vértices que forman parte de A se cambia a "etiquetado".

Una vez que se establecen los nodos que forman parte del subconjunto A, se procede a ordenarlos. Al terminar el ordenamiento, el nodo 7 debe quedar después del nodo 2 y el nodo 6 después del nodo 3. Esto provoca una disminución al número de escaneos.

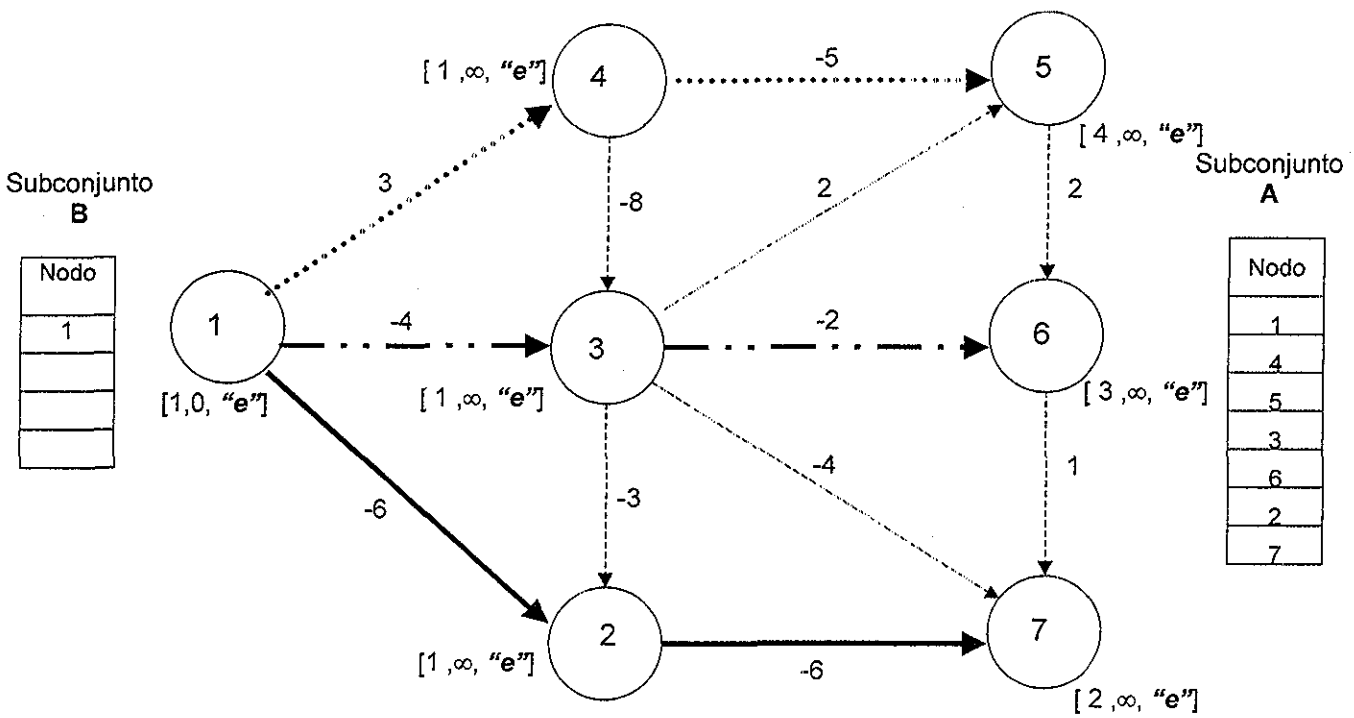


Figura 3.1.3.3. Los arcos con líneas gruesas representan los arcos admisibles desde el nodo origen, se usan distintos tipos de línea para las tres distintas familias que surgen.

Una vez que el subconjunto A es ordenado, se toma al primer elemento de A para ser escaneado. Para nuestro ejemplo, el primer elemento a ser escaneado es el nodo origen como se muestra en la figura 3.1.3.4.

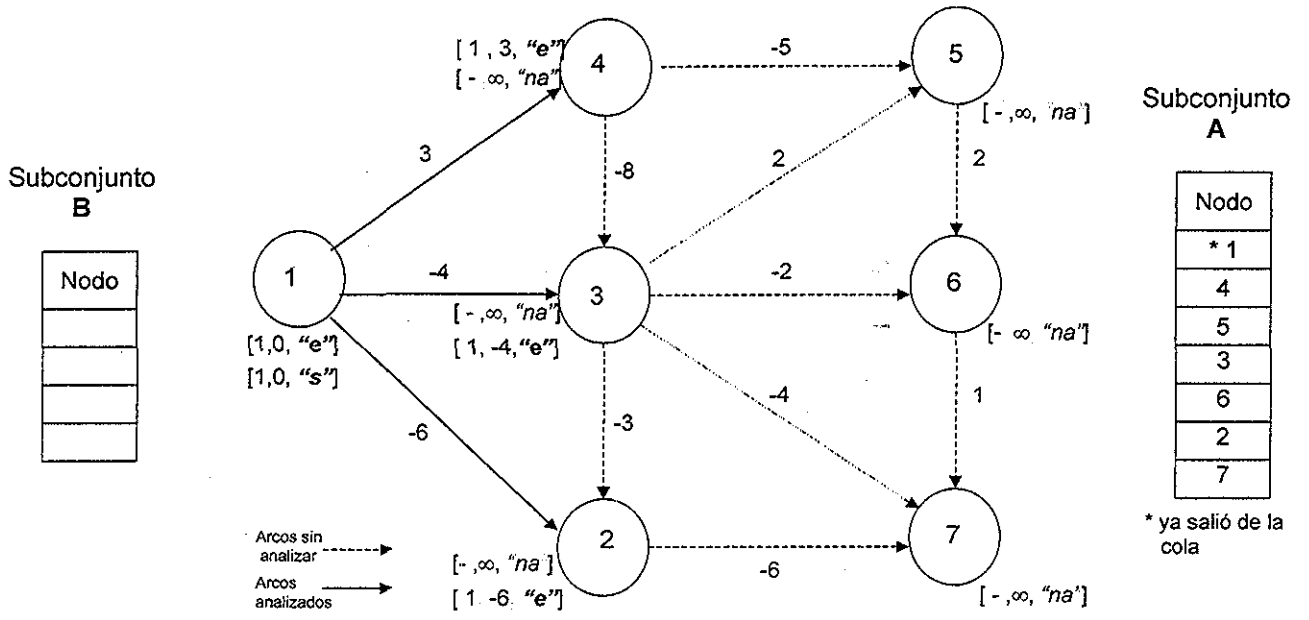


Figura 3.1.3.4

Una vez escaneado el nodo origen se toma el siguiente elemento del subconjunto A, para nuestro ejemplo es el nodo 4, seguido por el nodo 5, como se muestra en la figura 3.1.3.5.

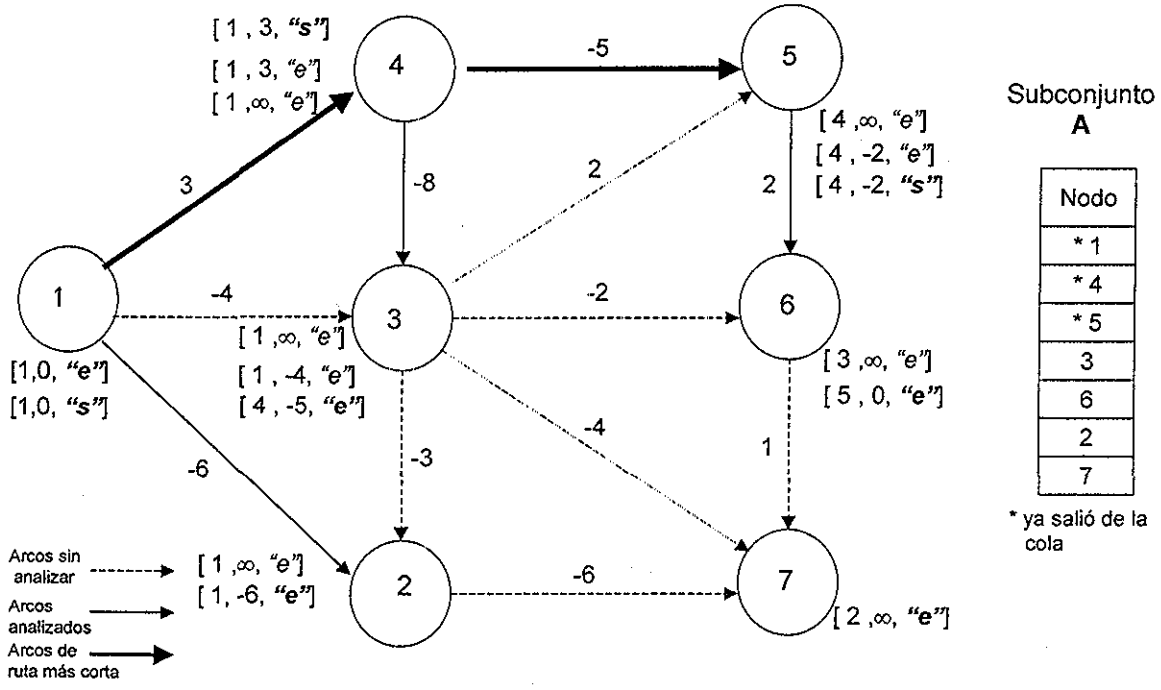


Figura 3.1.3.5

El siguiente nodo a ser escaneado según el orden establecido en el subconjunto A, es el nodo 3 seguido por el nodo 6, nodo 2 y por último el nodo 7. En la figura 3.1.3.6 se muestran los diferentes estados que tomaron los nodos hasta llegar al resultado final. La gráfica resultante es la misma que la de la figura 3.1.1.7.

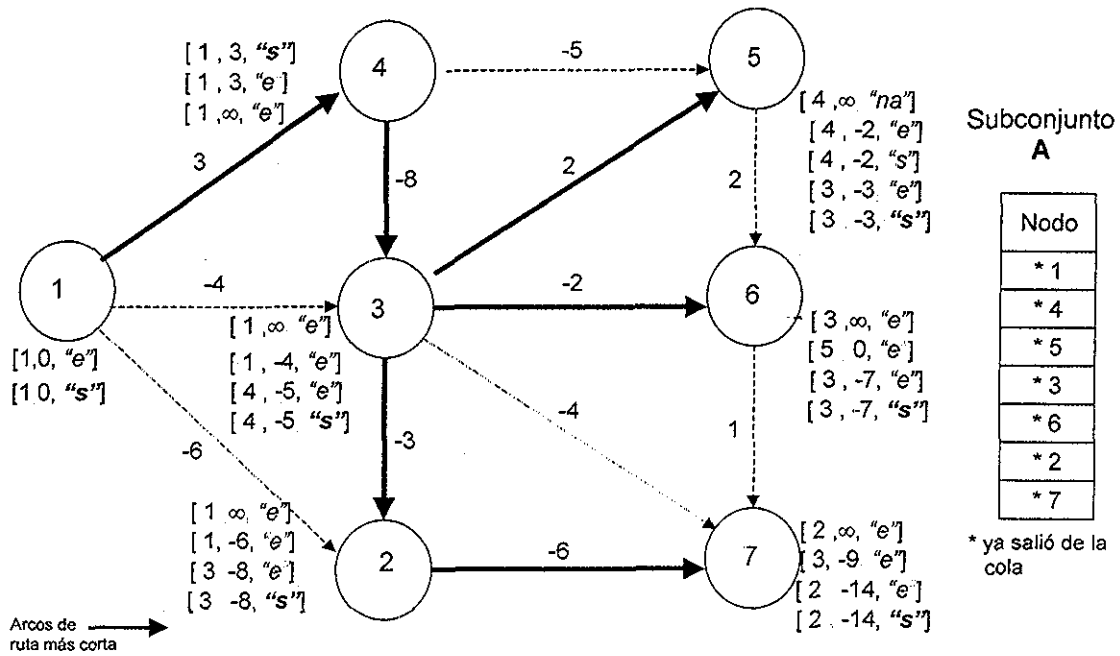


Figura 3.1.3 6

3.1.4 Algoritmo Simplex

Este algoritmo es una especialización del método Simplex (Dantzig, 1951) que se utiliza en redes para problemas de ruta más corta. Cuando la etiqueta de longitud d(*v*) de un vértice *v* se reduce tras haber sido *actualizado*, el algoritmo reduce las etiquetas de longitud de todos los vértices en el subárbol con raíz en *v* por la misma cantidad, lo cual es equivalente a recorrer el subárbol y aplicar la operación de etiquetado a los arcos que están en el árbol.

Varios códigos del método Simplex para redes mantienen a los vértices del árbol en una lista. Las implementaciones difieren en cómo se encuentra el siguiente arco para iniciar un pivoteo.

La implementación de Cherkassky y Goldberg (1999) mantiene al conjunto de vértices con estado "etiquetado" en una cola FIFO, al igual que los algoritmos de Tarjan y Bellman-Ford-Moore. El siguiente vértice a ser escaneado, se obtiene del principio de la cola.

Al principio del algoritmo, el estado del vértice origen *s* es "etiquetado", su etiqueta de longitud es igual a cero y él es su mismo padre; los demás nodos tienen estado "no alcanzado", etiqueta de longitud igual a infinito y padre nulo. Por lo que al inicio del algoritmo el primer y único elemento de la cola es el nodo origen *s*.

Si durante la operación de escaneo de un nodo *u* se analiza a un arco (*u,v*) con costo reducido negativo, se *actualiza* y se marca con S(*v*)="etiquetado" agregándose al final de la cola, si es que no se encuentra en la misma. Además, se recorre **todo** el subárbol con

raíz en *v* actualizando a las etiquetas de longitud de los descendientes por la misma diferencia con la que fue actualizada d(*v*), lo que se conoce como pivoteo y se sacan de la cola, si es que se encuentran en ella. Al arco (*u,v*) se le llama arco pivote.

El algoritmo termina cuando la cola queda vacía. El límite teórico del tiempo de ejecución del algoritmo Simplex es de $O(n^2m)$. (Cherkassky y Goldberg, 1999)

A pesar de que los nodos también pueden ser escaneados varias veces, al hacer el pivoteo se realizan menos operaciones de escaneos, debido a que aquellos vértices cuyas etiquetas de longitud mejoraron se actualizan.

Si durante el pivoteo de un nodo *v*, se encuentra que *v* pertenece a su propio subárbol, entonces existe un ciclo negativo en la red y el algoritmo termina indicando que encontró un ciclo negativo.

3.1.4.1 Ejemplo del algoritmo Simplex

Como se comentó anteriormente, para ejemplificar este algoritmo se utilizará la misma red que para los algoritmos anteriores, figura 3.1.4.1.

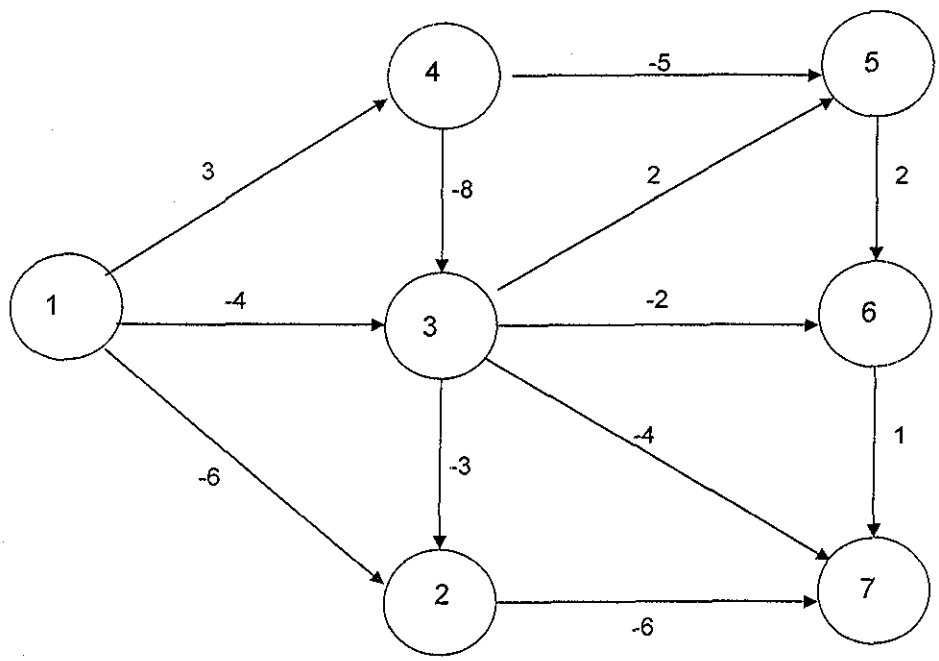


Figura 3.1.4.1

Al inicio del algoritmo se marcan todos los vértices con padre nulo, etiqueta de longitud igual a infinito y estado "na", excepto al vértice origen, como en los algoritmos anteriores y como se muestra en la figura 3.1.4.2.

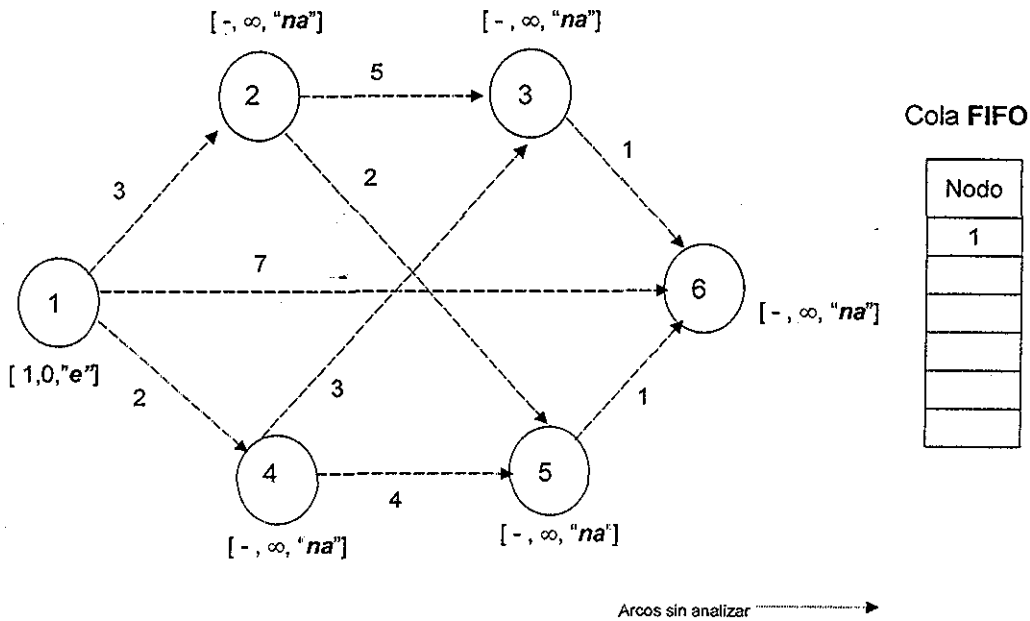


Figura 3.1.4.2

Al igual que en los algoritmos de Bellman-Ford-Moore y Tarjan, este algoritmo maneja una cola FIFO.

Al inicio del algoritmo el único elemento que se encuentra en la cola es el nodo origen, por lo que es el primer nodo a escanear. Ver figura 3.1.4.3.

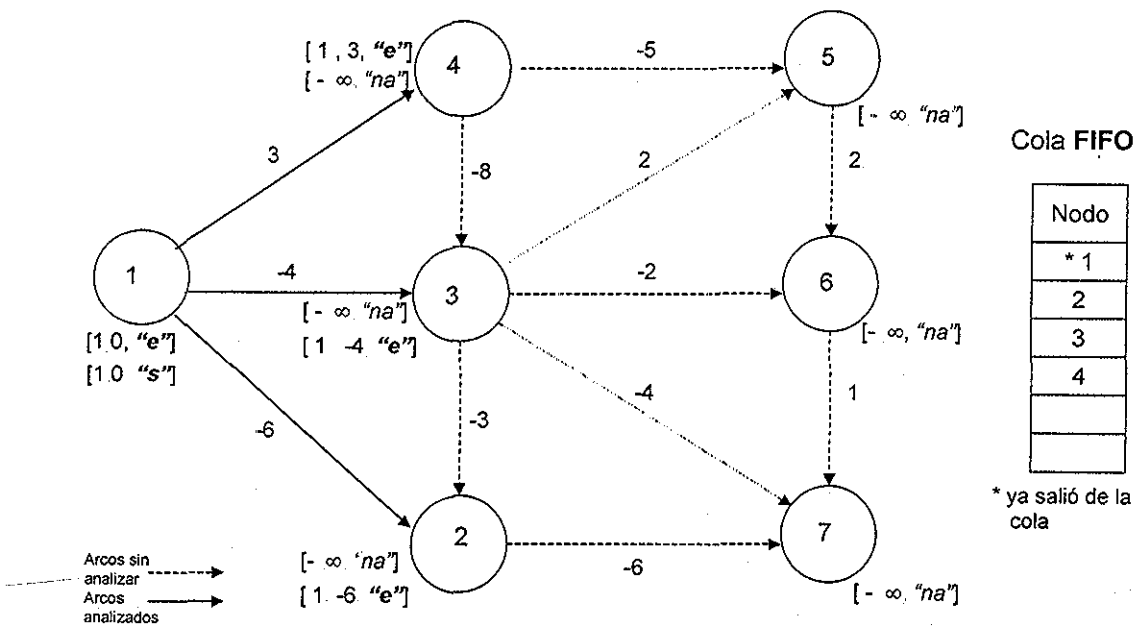


Figura 3.1.4.3

Al igual que en el ejemplo de los algoritmos de Bellman-Ford-Moore y Tarjan, el siguiente nodo a escanear es el nodo 2, como se muestra en la figura 3.1.4.4.

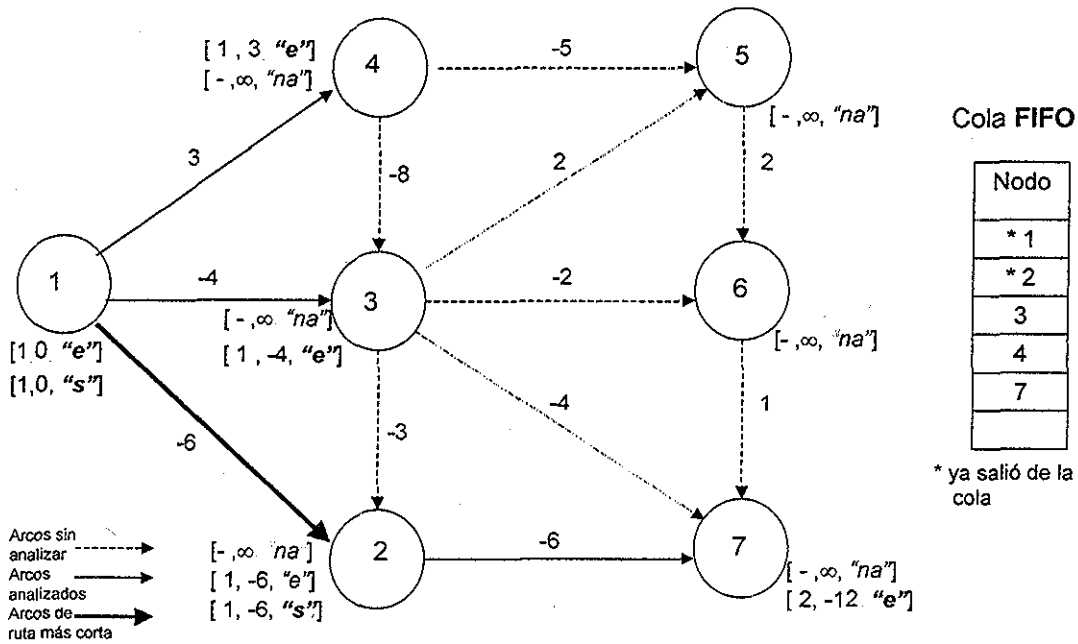


Figura 3.1.4.4

Una vez escaneado se procede con el siguiente de la cola, el nodo 3. Durante el escaneo del nodo 3 se tiene que el arco que va al nodo 2 tiene un costo reducido negativo; por lo que, el nodo 2 se *actualiza,* su estado cambia a "etiquetado" y se agrega al final de la cola. El algoritmo recorre el subárbol del nodo 2 para *actualizar* a sus descendientes y sacarlos de la cola. Para nuestro caso su único descendiente es el nodo 7; por lo que, se disminuye d(7) con la misma diferencia con la que fue a*ctualizada* d(*2*), -1, y se saca de la cola, como se muestra en la figura 3.1.4.5.

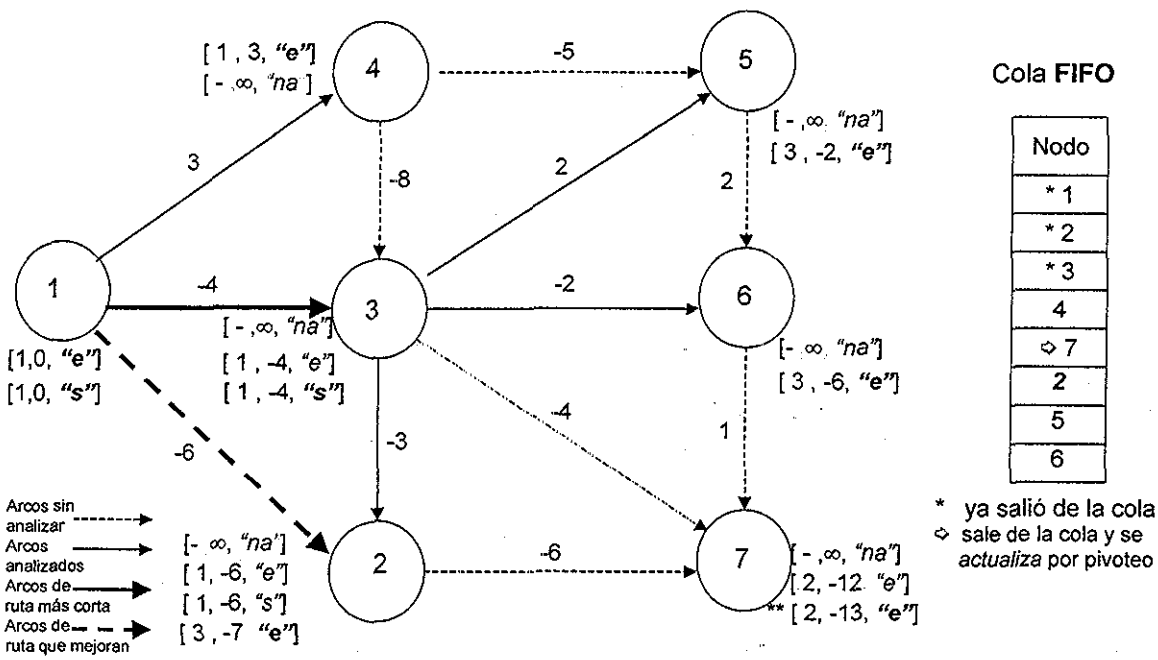


Figura 3.1.4.5. El doble asterisco indica que el nodo fue sacado de la cola

Durante el escaneo del nodo 4 se tiene que el arco que va al nodo 3 tiene un costo reducido negativo; por lo que, el nodo 3 se *actualiza,* su estado cambia a "etiquetado" y se agrega al final de la cola. El algoritmo recorre el subárbol del nodo 3 para *actualizar* a sus descendientes y sacarlos de la cola. Para nuestro caso sus descendientes son los nodos 2, 5 y 6; por lo que, se disminuye su d(*v*) con la misma diferencia con la que fue *actualizada* d(3), -1, y se sacan de la cola, como se muestra en la figura 3.1.4.6.

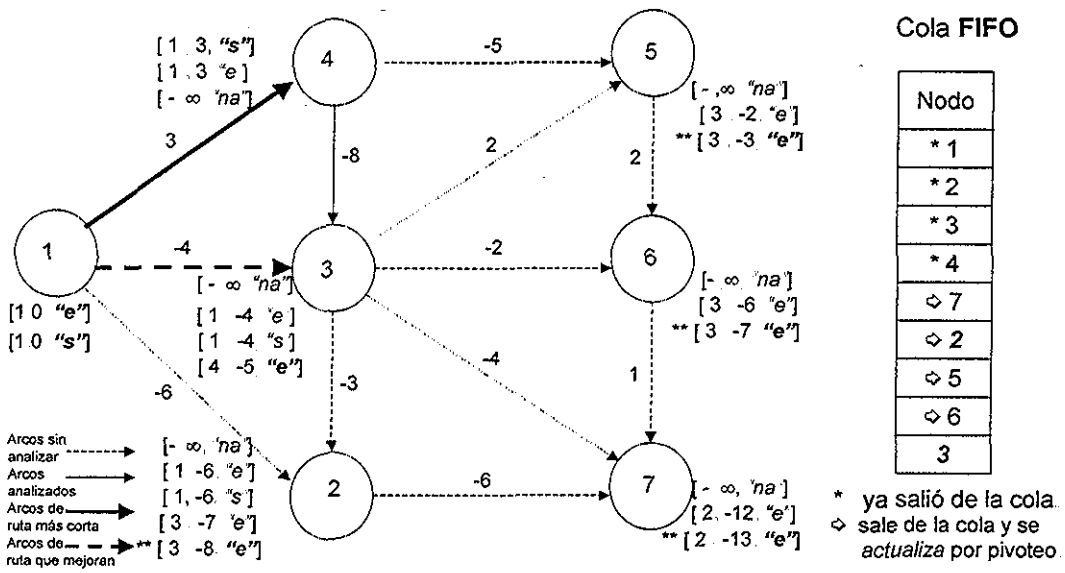


Figura 3.1.4.6. El doble asterisco indica que el nodo fue sacado de la cola

El algoritmo termina cuando la cola queda vacía. En la figura 3.1.4.7 se muestran los diferentes estados que tuvieron los nodos hasta obtener el resultado final, también se muestra la cola FIFO generada. Como se comento anteriormente la gráfica resultante se encuentra en la sección 3.1.1 siendo la figura 3.1.1.7.

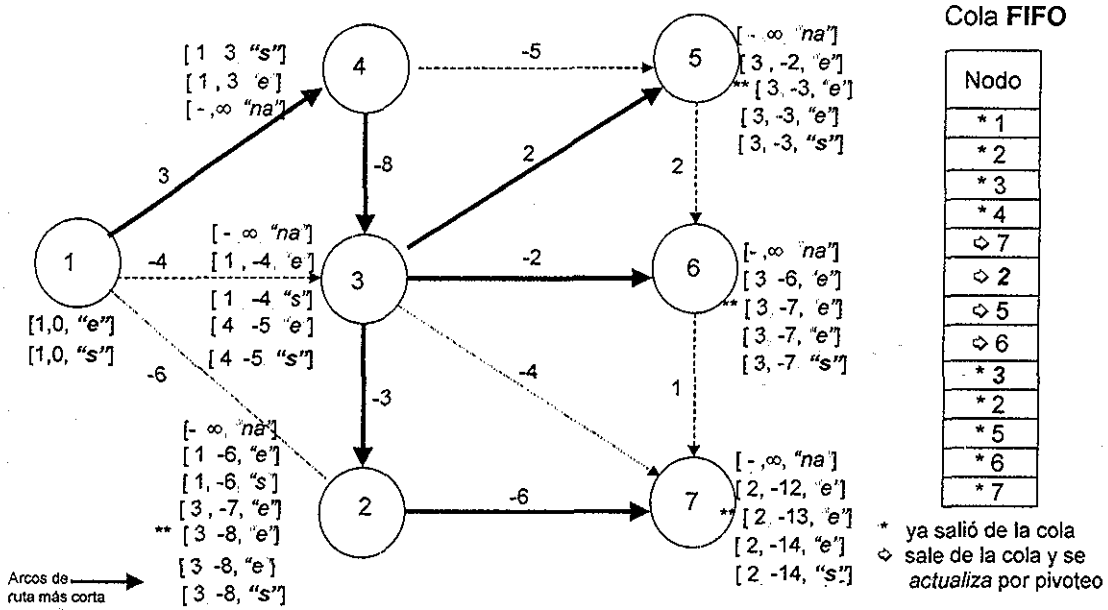


Figura 3.1.4.7

En el siguiente capítulo se describirá una modificación al algoritmo de Dijkstra, el cual resuelve el problema de ruta más corta de forma eficiente en redes cuyos arcos tengan longitudes arbitrarias.

IV ALGORITMO PROPUESTO

Como se comentó al final del capitulo II, la variación utilizada por Cherkassky y Goldberg (1999) del algoritmo de Dijkstra es sumamente ineficiente en redes con arcos negativos. La modificación consiste en no utilizar el estado "permanente", lo cual permite que un vértice sea escaneado varias veces y así llegar a la solución del problema. Sin embargo, esta variación provoca que el número de escaneos crezca en forma exponencial (Cherkassky *et al.*, 1996).

Debido a que el algoritmo de Dijkstra sigue siendo el algoritmo más eficiente para resolver el problema de ruta más corta con arcos no negativos, sería conveniente encontrar una modificación al mismo que permita resolver el problema en redes con arcos negativos de forma más eficiente.

En la siguiente sección se explica la modificación propuesta al algoritmo de Dijkstra para resolver en forma eficiente redes con arcos cuyas longitudes son negativas.

4.1 Dijkstra modificado

El algoritmo propuesto consta de dos etapas. La primera, llamada de Dijkstra, utiliza el mismo principio que el algoritmo de Dijkstra; y la segunda, llamada de reescaneo, realiza un reescaneo de los vértices de los subárboles generados durante la primera etapa. Los estados utilizados en este algoritmo son: "no alcanzado" (na), "etiquetado" (e), "*permanente*" (*p*), "*permanente actualizado*" (*pa*) y "*reescaneado*" (*rs*).

Inicio del algoritmo

1. Al principio del algoritmo, el vértice origen *s* tendrá:
   etiqueta de longitud $d(s) = 0$, padre $\pi(s) = s$ y estado $S(s) =$ etiquetado,
   y el resto de los vértices
   $d(v) = \infty$, $\pi(v) =$ nulo y $S(v) =$ "no alcanzado".

Primera etapa del algoritmo o etapa de Dijkstra

El procedimiento a seguir en esta etapa es muy similar al algoritmo de Dijkstra original, explicado en el capítulo II. La diferencia consiste en que para el algoritmo modificado existe un estado "permanente actualizado", que se asigna a nodos con estado "permanente" cuya etiqueta de longitud d(v) no es exacta. Si al término de esta etapa existe algún nodo con estado "permanente actualizado" se procede a la segunda etapa llamada de reescaneo. Los pasos a seguir en esta etapa de Dijkstra son:

2. Se escanea el vértice v con estado "etiquetado" que tenga etiqueta de longitud d(v) mínima. El proceso de escaneo de v consiste en:
   2.1 Analizar todos los arcos (v,w) que parten de v, verificando si $d(v) + l(v,w) < d(w)$; en caso afirmativo, la etiqueta de longitud de w se *actualiza* ($d(w) = d(v) + l(v,w)$ y $\pi(w) = v$).
   2.2 Si el estado de w es "permanente" se cambia a "permanente actualizado". Si el estado de w no es "permanente" se cambia a "etiquetado". (En este punto estriba la diferencia de este algoritmo con respecto al de Dijkstra.)
   2.3 El estado de v se cambia a "permanente".

3. Si existe algún nodo v con estado "etiquetado" se regresa al punto 2. En caso contrario se avanza al siguiente punto.

4. Si existe algún nodo con estado "permanente actualizado" se procede a la segunda etapa (punto 5), en caso contrario el algoritmo ha terminado con la solución correcta.

Observe que los nodos con estado "etiquetado" son los únicos que se escanean, de modo que ningún nodo es escaneado más de una vez en esta primera etapa.

Segunda etapa del algoritmo o etapa de reescaneo

Si al final de la primera etapa del algoritmo se tienen nodos con estado "permanente actualizado" quiere decir que no se ha llegado a la solución correcta, debido a que existen nodos con etiqueta de longitud d(v) inexacta y debe procederse a un reescaneo. En esta etapa se reescanean los nodos de los subárboles generados en la etapa anterior. Los pasos a seguir son:

5. Se genera una lista ordenada llamada lista de reescaneo formada por los nodos que pertenecen a los subárboles generados en la etapa anterior. Los subárboles mencionados consisten en todos los nodos que fueron marcados con estado "permanente" o "permanente actualizado" en la etapa anterior. El orden de la lista consiste en que para un par de vértices v y w, que se encuentren en algún subárbol, en donde v es padre de w, v quede antes que w. Si durante el ordenamiento se descubre que un nodo v con estado "permanente actualizado" es descendiente de sí mismo entonces hay un ciclo negativo en la red y el algoritmo se detiene.

6. Se reescanea el nodo v que se encuentre al principio de la lista ordenada. El proceso de reescaneo consiste en:

6.1 Analizar todos los arcos (v,w) que parten de v, verificando si $d(v)+ l(v,w) < d(w)$,en caso afirmativo, la etiqueta de longitud de w se *actualiza*.

6.2 Si w esta fuera de la lista de reescaneo su estado cambia a "etiquetado" no importando el estado que tenga.

6.3 El estado de v se cambia a "reescaneado" y se saca de la lista.

7. Si existe algún nodo en la lista de reescaneo se regresa al punto 6. En caso contrario se avanza al siguiente punto.

8. Si existe algún nodo con estado "etiquetado" se procede a la primera etapa (punto 2), en caso contrario el algoritmo ha terminado con la solución correcta.

Dado que el algoritmo de Dijkstra modificado impide el reescaneo de nodos en la etapa de Dijkstra, su tiempo de ejecución no crece exponencialmente. En el peor de los casos, el método heurístico de selección de los nodos a escanear, puede no ayudar, de modo que su elección se convertiría en arbitraria. Dado que el algoritmo de Bellman-Ford-Moore selecciona los nodos a escanear en forma arbitraria y su límite teórico de tiempo de ejecución es $O(nm)$, el límite de tiempo de ejecución del algoritmo de Dijkstra modificado es $O(nm)$.

Las operaciones del algoritmo se describen a continuación en la figura 4.1

```
Inicio
  /* Se inicializa al nodo origen s */
  d(s) = 0,
  π(s) = s,
  S(s) = "etiquetado"
  /* Se inicializa al resto de los nodos */
  Para todo (v ≠ s ) haz {
     d(v) = ∞,
     π(v) = nulo,
     S(v) = "no alcanzado"
  }
  /* Primera etapa o etapa de Dijkstra */
  Mientras no se llegue a TERMINA EL ALGORITMO haz
     Si existe algún v con S(v) = "etiquetado", entonces {
        u = nodo con etiqueta de longitud d(v) mínima
        Para todo arco (u,w) con inicio en u haz {
           Si d(u) + l(u,w) < d(w), entonces {
              /* Se actualiza w */
              d(w) = d(u) + l(u,w),
              π(w) = u,
              Si S(w) = "permanente", entonces {
```

```
                S(w) = "permanente actualizado",
            } si no {
                S(w) = "etiquetado",
            }
        }
    }
    S(u) = "permanente"
}

    /* Segunda etapa o etapa de reescaneo */
    Si existe algún v con S(v) = "permanente actualizado", entonces {
        Se genera la lista de reescaneo,
        Si se encontró un ciclo negativo, entonces {
            TERMINA EL ALGORITMO,
        } sino {
            Si existe algún nodo v en la lista de reescaneo haz{
                Para todo arco (v,w) con inicio en v haz {
                    Si d(v) + l(v,w) < d(w), entonces {
                        /* Se actualiza w */
                        d(w) = d(v) + l(v,w),
                        π(w) = v,
                        Si w ∉ a la lista de reescaneo, entonces {
                            S(w) = "etiquetado",
                        }
                    }
                }
                S(v) = "reescaneado",
            }
        }
    } sino {
        TERMINA EL ALGORITMO con solución correcta,
    }
} /* Fin del mientras */
```

Figura 4.1

4.1.1 Ejemplo del algoritmo

Para entender mejor el procedimiento se presenta un ejemplo. Es importante mencionar que el orden con el cual el algoritmo toma a los arcos de un vértice cualquiera, depende del orden como fueron introducidos los datos.

Se tienen los siguientes datos de la red:

Datos de la red para el ejemplo

Vértice	Arcos	Longitud del arco
1	(1,2) (1,3) (1,4)	-6 -4 3
2	(2,5)	-6
3	(3,5) (3,2) (3,6) (3,7)	-4 -3 -2 2
4	(4,3) (4,7)	-8 -5
5	(5,9) (5,8)	3 1
6	(6,5) (6,9) (6,10)	1 3 7
7	(7,6) (7,9)	2 -8
8	(8,9) (8,6)	4 2
9	------	------
10	(10,9) (10,7)	-6 0

Tabla 4.1.1

Su gráfica se muestra en la figura 4.1.1

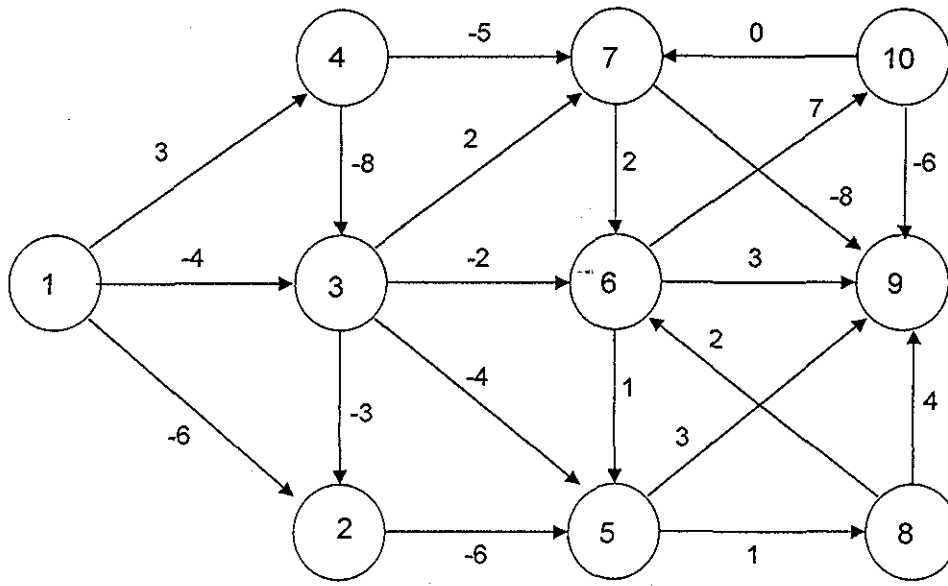


Figura 4.1.1

Al inicio del algoritmo se marca a todos los vértices con padre nulo y con etiqueta de longitud igual a infinito, excepto al vértice origen que en este caso es el vértice 1. Como se observa en la figura 4.1.2

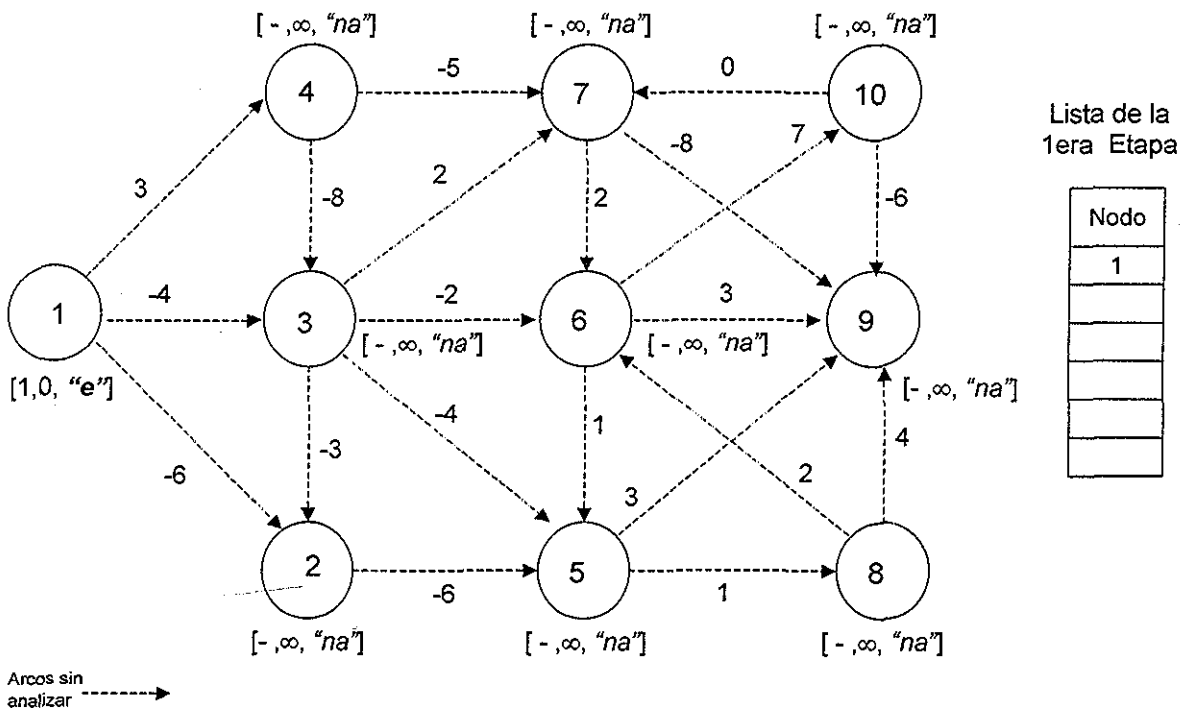


Figura 4.1.2

Al inicio del algoritmo el único vértice con estado "etiquetado" es el nodo origen por lo que se procede a escanearlo. Una vez que el nodo se escanea su estado cambia a "permanente". Como se muestra en la figura 4.1.3

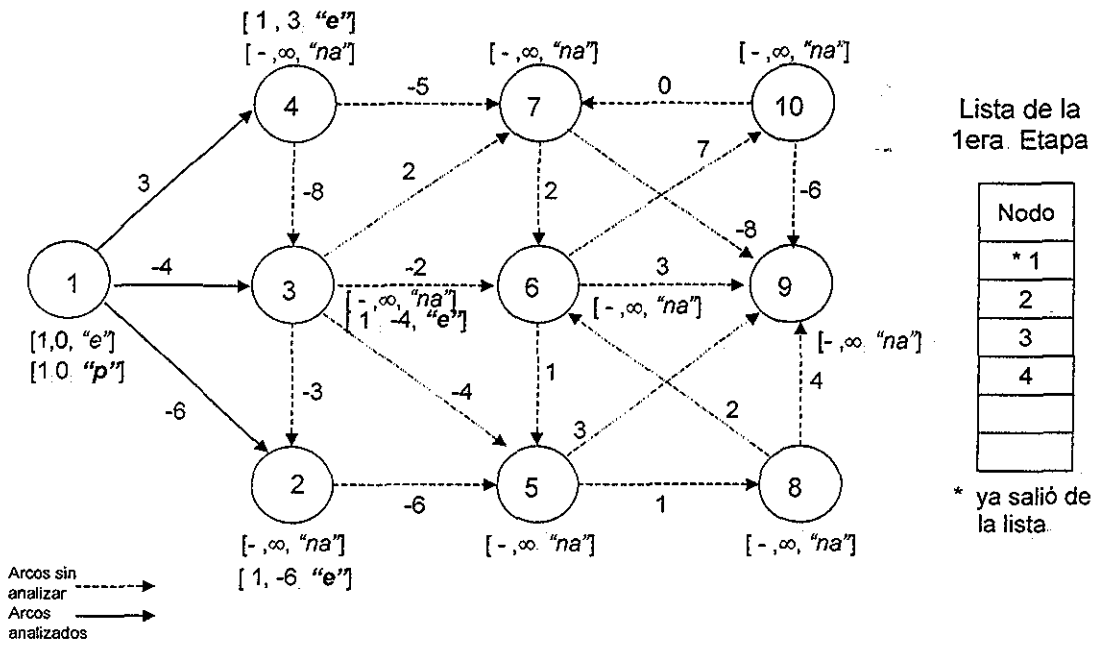


Figura 4.1.3

Se escoge el vértice con estado "etiquetado" cuya d(v) sea la mínima para ser el siguiente nodo a escanear. Una vez que el nodo se escanea su estado cambia a "permanente". Como se muestra en la figura 4.1.4

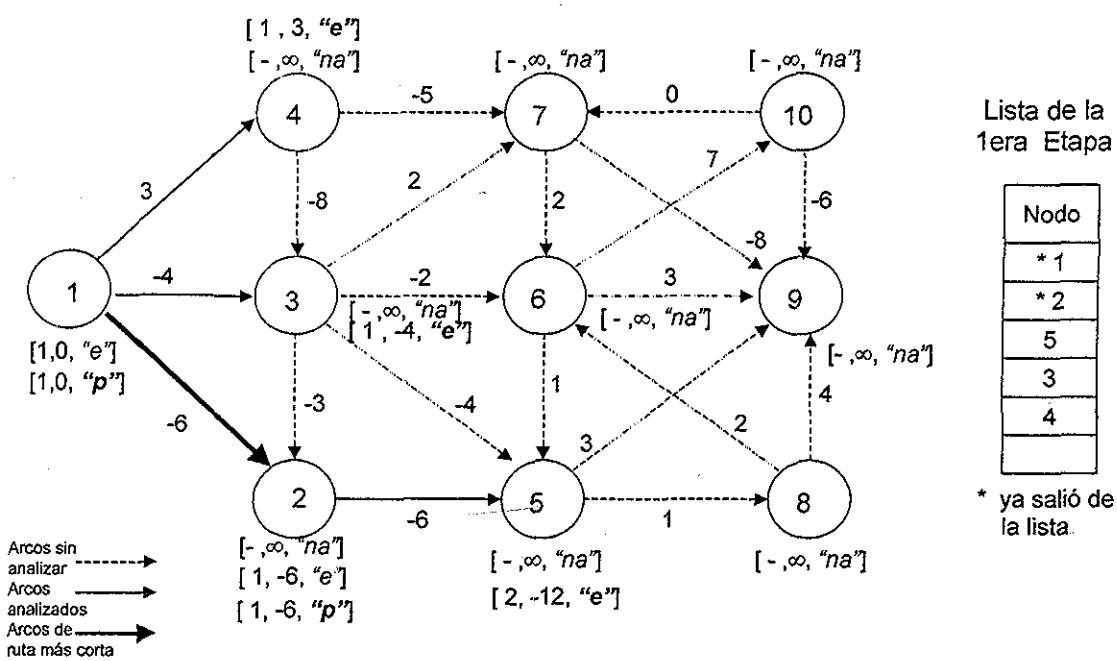


Figura 4.1.4

Falta

pag

45

En la figura 4.1.7 se muestra el subárbol generado en la primera etapa del algoritmo, el cual se forma con los arcos que vienen de los padres que tienen los nodos con estado "permanente" o "permanente actualizado".

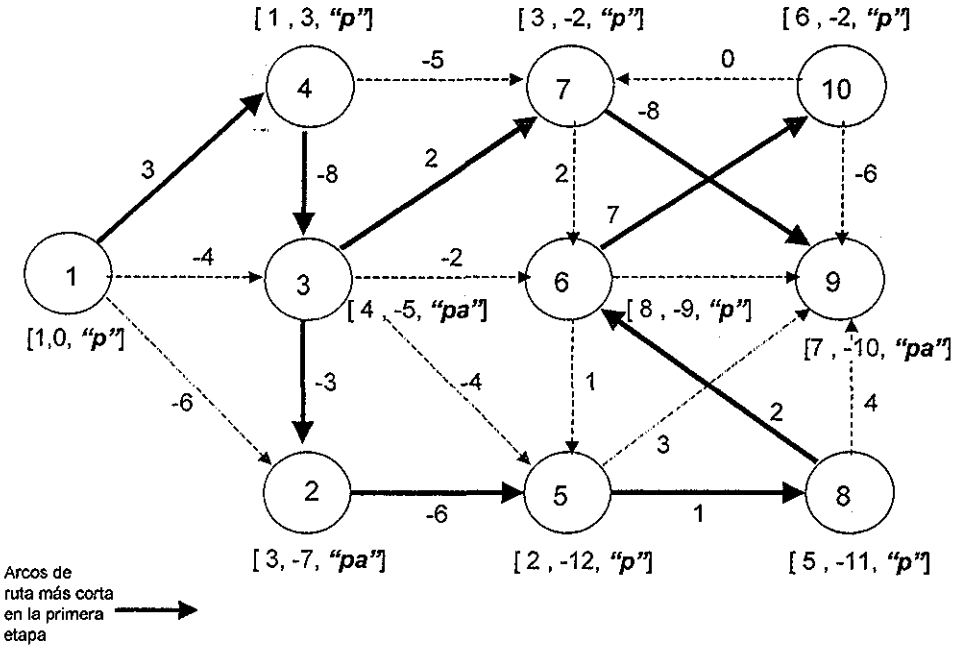


Figura 4.1.7

Si al terminar la primera etapa hay algún nodo con estado "permanente actualizado" entonces se pasa a la segunda etapa o etapa de reescaneo, donde los nodos serán reescaneados de acuerdo al orden del subárbol generado en la figura 4.1.7

Como se puede observar en la figura anterior, se tienen vértices con estado "permanente actualizado", por lo que se pasa a la segunda etapa. En la segunda etapa se ordenan los vértices, de modo que la raíz del subárbol generado, el vértice 1, será el primer vértice a ser escaneado, el vértice 4 al ser el único sucesor del nodo 1 será el siguiente en la lista y así sucesivamente hasta ordenar todos los vértices que se encuentran en el subárbol.

En nuestro ejemplo para el caso de los vértices 2 y 7, los cuales tienen como padre al nodo 3, el primer nodo que sube a la lista es el último de los hijos del nodo 3, según el orden de entrada de los datos, en este caso es el nodo 7.

En la lista de reescaneo, la cual se muestra en la tabla 4.1.2, los datos para cada nodo se conforman por $(v, \pi(v), d(v), S(v))$. En esta tabla se muestra el estado de cada uno de los nodos al iniciar la segunda etapa del algoritmo. Es importante mencionar que el estado que tengan los nodos en la lista de reescaneo siempre será diferente al estado "etiquetado" no importando el estado que tengan.

Lista de reescaneo

Nodo	π	d	S
1	1	0	"permanente"
4	1	3	"permanente"
3	4	-5	" permanente actualizado"
7	3	-2	"permanente"
9	7	-10	"permanente"
2	3	-7	" permanente actualizado"
5	2	-12	"permanente"
8	5	-11	"permanente"
6	8	-9	"permanente"
10	6	-2	" permanente actualizado"

Tabla 4.1.2

Durante la segunda etapa del algoritmo, se escanean y se sacan los elementos de la lista de reescaneo siguiendo el orden establecido, empezando por el primer nodo de la lista. Al escanear los nodos, su estado cambia a "reescaneado". Los vértices cuyas etiquetas de longitud mejoren y se encuentren fuera de la lista de reescaneo, se marcan como "etiquetados", aunque tengan estado "reescaneado".

En la figura 4.1.8 se muestran los diferentes estados que toman los nodos hasta llegar al final de la segunda etapa. En nuestro ejemplo, al reescanear al nodo 10 se mejora la etiqueta de longitud del nodo 7, cuyo estado era "reescaneado" y se encontraba fuera de la lista de reescaneo, por lo que se *actualiza* y su estado cambia a *"etiquetado"*.

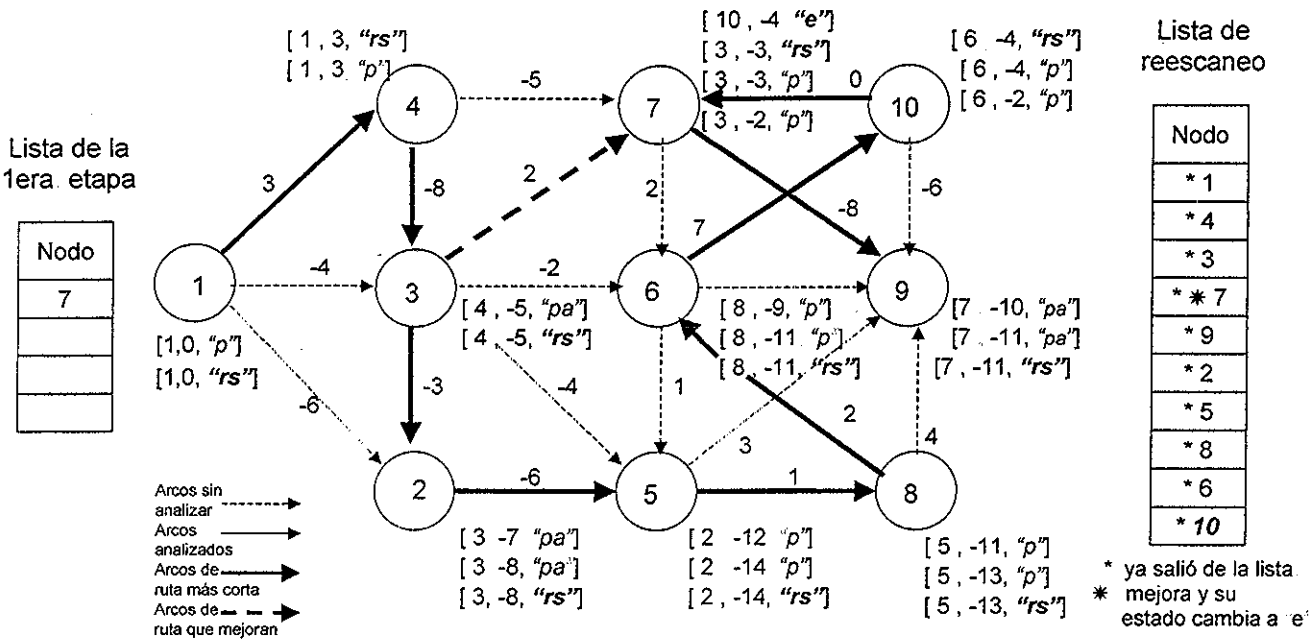


Figura 4.1.8

Dado que existen nodos con estado "etiquetado" al final de la segunda etapa, se continúa con la primera etapa nuevamente. En este caso, el único nodo con estado "etiquetado", al inicio de la etapa de Dijkstra, es el nodo 7, como se muestra en la figura 4.1.9. Es importante mencionar que, en caso de que se tuviera que pasar a la segunda etapa nuevamente, los subárboles a tomarse en cuenta, para la generación de la lista de reescaneo, tendrían su raíz en los vértices con estado "etiquetado" al inicio de la etapa de Dijkstra; como el nodo 7 en este caso.

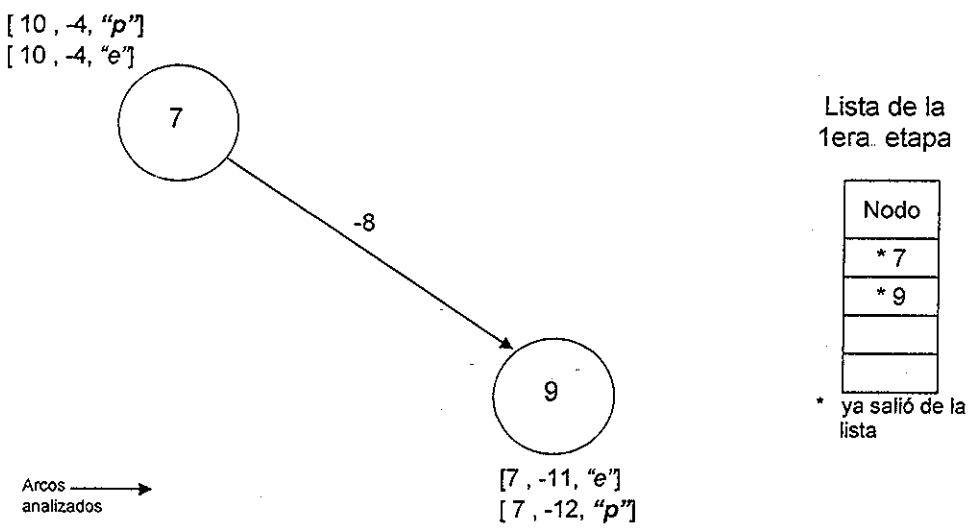


Figura 4.1.9

En la figura 4.1.10 se muestran los estados por los que pasaron los nodos del subárbol generado en esta etapa hasta llegar al fin de la misma.

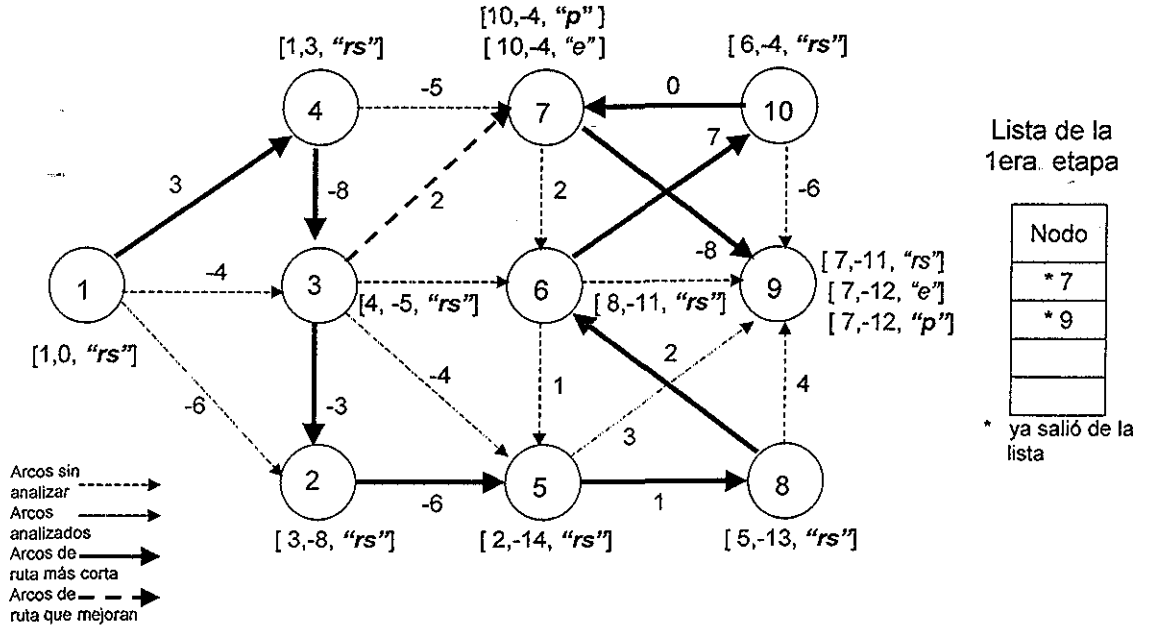


Figura 4.1.10

Al no quedar vértices con estado "permanente actualizado" en la primera etapa del algoritmo, el algoritmo termina.

En la Figura 4.1.11 se muestra el resultado final.

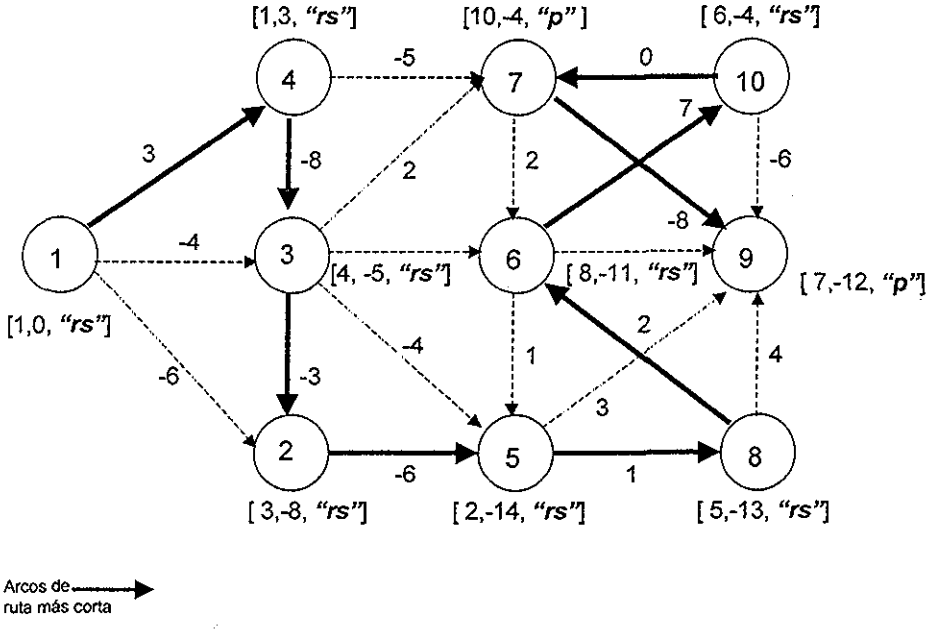


Figura 4.1.11

El algoritmo de Dijkstra modificado es correcto

En el capítulo 2 se mostró mediante un ejemplo que el algoritmo de Dijkstra es incorrecto para redes con longitudes de arco negativas, ya que al marcar a los nodos con estado *"permanente"* pueden quedar nodos que deberían de ser *"etiquetados"*, y por lo tanto reescaneados, sin escanearse. De modo que, al terminar el algoritmo, podría no cumplirse la condición de terminación que marca el método de escaneo.

El error también podría verse desde el punto de vista del método de etiquetamiento; ya que, al terminar el algoritmo de Dijkstra, podrían quedar arcos sin analizarse pero que cumplen con las condiciones de selección, incumpliéndose la condición de terminación del método de etiquetamiento.

La primera etapa del algoritmo de Dijkstra modificado tiene el mismo error, pero la segunda etapa lo corrige; ya que, reescanea **todos** los nodos de los subárboles generados en la etapa de Dijkstra. Y es justamente en estos subárboles donde se encuentran los arcos seleccionables (desde el punto de vista del método de etiquetamiento) o todos los nodos etiquetables (desde el punto de vista del método de escaneo). Ya que la etapa de reescaneo coloca el estado de *"etiquetado"* a todo nodo *actualizado,* **sin importar el estado que tenga,** no se violan los principios del método de escaneo y, por lo tanto, tampoco los del método de etiquetamiento. Entonces, con base en el teorema 2.1, el algoritmo de Dijkstra modificado es correcto.

En el siguiente capítulo se presentan los resultados obtenidos tras comparar el desempeño de los algoritmos con distintos tipos y tamaño de redes. Las redes que se usaron para la etapa de experimentación se basan en el estudio realizado por Cherkassky *et al.* (1996) y por Cherkassky y Goldberg (1999).

V EXPERIMENTACIÓN Y RESULTADOS

En este capítulo se describen los experimentos realizados para evaluar el desempeño del nuevo algoritmo en distintos tamaños y tipos de redes. Se escribe una breve explicación sobre las redes utilizadas y los criterios de evaluación. Finalmente se reportan los resultados obtenidos.

5.1 Experimentación

Para la experimentación del trabajo se generaron aleatoriamente diferentes tipos de redes, las cuales modelan el problema de ruta más corta. La característica común de estas redes es la de tener al menos un arco con longitud negativa.

Los programas para generar las redes se obtuvieron de internet (http://www.intertrust.com/star/goldberg/index.html). Los códigos de los programas originales también se obtuvieron de internet (http://www.intertrust.com/star/goldberg/index.html); vienen escritos en C y listos para compilarse en una versión de Solaris (Cherkassky *et al.*, 1999). Para realizar los experimentos se hicieron algunas modificaciones a los programas originales, debido a la necesidad de adaptarlos a Borland C de Windows 95 para una PC. Tanto los programas modificados como los originales se encuentran en el disco compacto (D.C.) anexo. El código del programa para el *Dijkstra modificado* se incluye en el apéndice A y también en el D.C. Los archivos de comandos utilizados se incluyen en el apéndice B y los resultados completos están en el apéndice C.

En la siguiente sección se describen los tipos de redes que se emplearon para el estudio.

5.1.1 Tipo de redes

Grid

Para explicar mejor cómo es que se generan este tipo de redes, llamemos capa *x* a una "línea" horizontal de nodos, esto es, a todos los vértices con la misma *y* y capa *y* a

una "línea" vertical de nodos en donde todos los vértices tengan la misma *x*. Al referirnos a los arcos intercapas nos referimos a aquellos arcos que conectan una capa con otra.

La red tipo Grid es una red con esqueleto de cuadrícula. En los casos más simples todos los arcos pertenecen a la cuadrícula. Sin embargo, para redes más complicadas hay arcos adicionales.

Las redes pueden ser cuadradas o rectangulares. Las redes cuadradas tienen el mismo número de nodos en las capas *x* que en las capas *y*. Las redes rectangulares tienen más nodos en las capas *x* que en las capas *y* o viceversa.

Para todas las redes generadas hay un vértice adicional, llamado nodo fuente. El nodo fuente se conecta a todos los vértices de la primera capa *y*. Para experimentos con redes tipo Grid, la longitud de los arcos entre capas *y* se selecciona uniformemente al azar en un intervalo de ℓ = [0, L], donde L puede tomar cualquier valor real. Los demás arcos toman valores aleatorios en un intervalo de [0, *U*], donde *U* puede tomar cualquier valor real.

En la siguiente figura se muestra ejemplos de redes tipo Grid cuadradas y rectangulares.

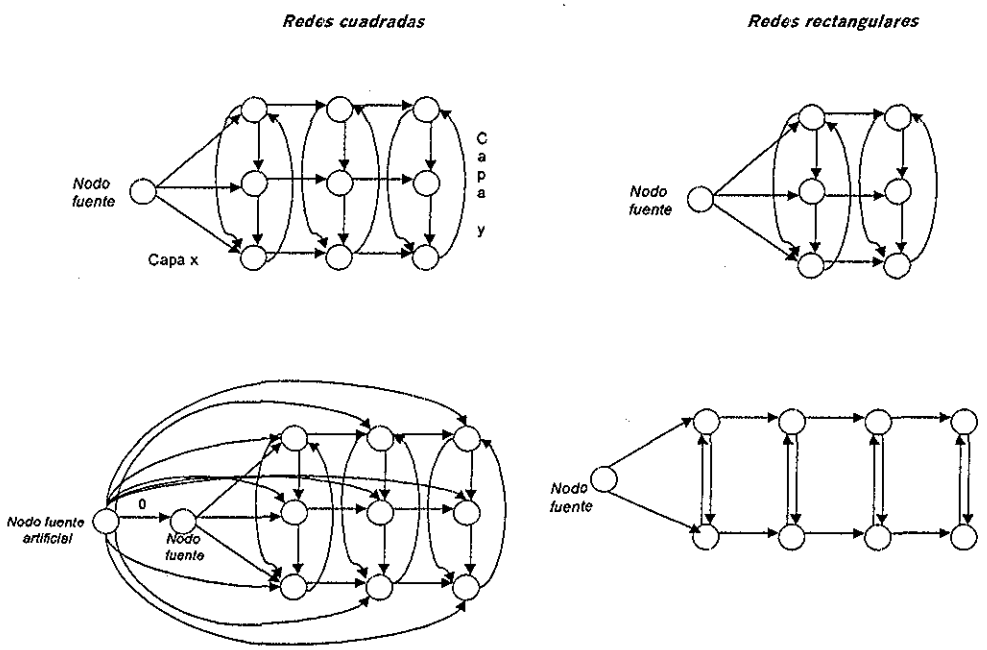


Figura 5.1.1.1

Para nuestro experimento se utilizaron redes Grid un poco más complicadas, Grid-NHard. Las redes Grid-NHard pueden ser rectangulares o cuadradas. Este tipo de redes, al igual que las mencionadas anteriormente tienen un nodo fuente; y en cada capa *y* forman un ciclo simple con un mismo valor positivo y pequeño, que denotaremos como *c*.

Además, cada capa *y* tiene un conjunto de arcos conectando parejas de vértices al azar; la longitud de estos arcos se escoge aleatoriamente de un rango [0, *U*], donde *U* puede tomar cualquier valor positivo.

Hay arcos que van de una capa *y* a la siguiente, como en las redes simples, pero además hay arcos que avanzan más de una capa *y*. Los arcos entre capas *y* tienen longitudes negativas y su valor se escoge aleatoriamente de un intervalo [L_1 , L_2], donde L_1 y L_2 son negativos.

En la figura 5.1.2. se muestra una red Grid-NHard para una cuadrícula con 3 nodos en la capa *x* y 4 nodos en la capa *y*, la longitud de los arcos intercapas *y* se tomaron aleatoriamente de un intervalo [L_1=-20,L_2= -5]. La longitud para los arcos del ciclo simple en cada capa *y* se fijó en *c*=1. El resto de los arcos para las capas *y* tomaron valores entre [0,*U*] donde *U* =100.

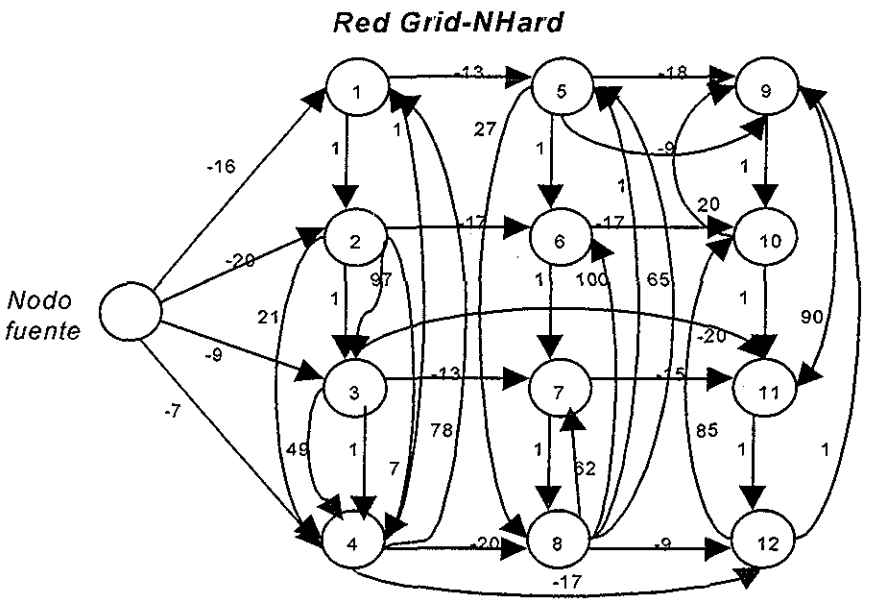


Figura 5.1.1.2

En este tipo de red es muy común encontrar que el camino más corto entre dos nodos contenga muchos arcos en vez de pocos. Esto hace que los algoritmos sean forzados a realizar muchos reescaneos.

Ciclicas

Las redes cíclicas se crean a partir de un ciclo básico. En Cherkassky *et al.* (1996) se realizan experimentos con redes positivas, las cuales nos servirán para explicar las redes que se utilizan en el presente trabajo. En las redes positivas utilizadas en Cherkassky *et al.* (1996) se fija un valor pequeño y positivo, *c*, para todos los arcos que forman el ciclo básico. Además, las redes tienen arcos que conectan pares de nodos aleatoriamente y cuya longitud se escoge aleatoriamente de un intervalo [0, *U*].

Para nuestro estudio se usa la red Rand-P, este tipo de red sí tiene arcos negativos. Los arcos negativos se generan a partir de una red positiva de la siguiente manera: se asigna a cada nodo v (sólo para la generación de la red) una etiqueta de longitud d(v) aleatoria. El valor de d(v) se toma de un intervalo [0,P]; posteriormente se calculan los costos reducidos de los arcos y se dejan esos costos reducidos como la longitud de los arcos para la red. Entre más pequeño sea P, la red generada tendrá menor número de arcos con longitud negativa.

En la figura 5.1.1.3 se muestra una red tipo Rand-P generada con los siguientes parámetros: n = 5, m = 20, donde n es el número de nodos y m el número de arcos, la longitud de los arcos que forman el ciclo tomaron un valor inicial de c = 1 y el resto de los arcos tomaron un valor inicial entre [0, U=10000]; posteriormente, d(v) tomó valores aleatorios entre [0, P=10000] para calcular la longitud final de los arcos, a partir del costo reducido auxiliar.

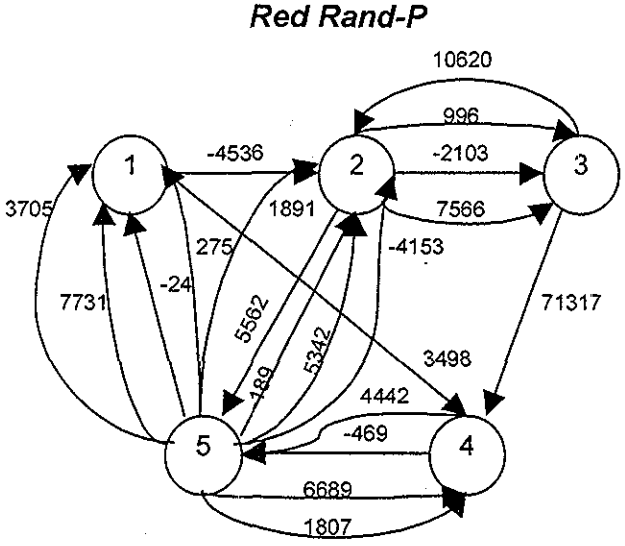


Figura 5.1.1.3

Acíclicas

La característica principal de estas redes es no tener ciclos. La red se forma de la siguiente manera: los nodos son numerados de 1 a n, y se forma un camino con arcos (i, i+1), $1 \leq i < n$. Estos arcos se llaman *arcos centrales*. Adicionalmente se generan arcos (i, j), $i < j$ aleatoriamente. Las longitudes de los arcos aleatorios y de los *arcos centrales* se escogen bajo diferentes criterios, como se explicará enseguida.

Una de las redes que se utilizó para la experimentación es la tipo Acyc-Neg, cuya característica es que todos sus arcos son no positivos. Los *arcos centrales* tienen una longitud de $c < 0$ pequeño y la longitud del resto de los arcos se escoge aleatoriamente de un intervalo entre [L, 0], donde $L < 0$.

En la figura 5.1.1.4. se muestra una red Acyc-Neg, generada con los siguientes parámetros: n = 5, m = 20, la longitud de los *arcos centrales* tomaron un valor de c = -1 y las longitudes de los demás arcos tomaron valores aleatorios entre [L=-20, 0].

Red Acyc-Neg

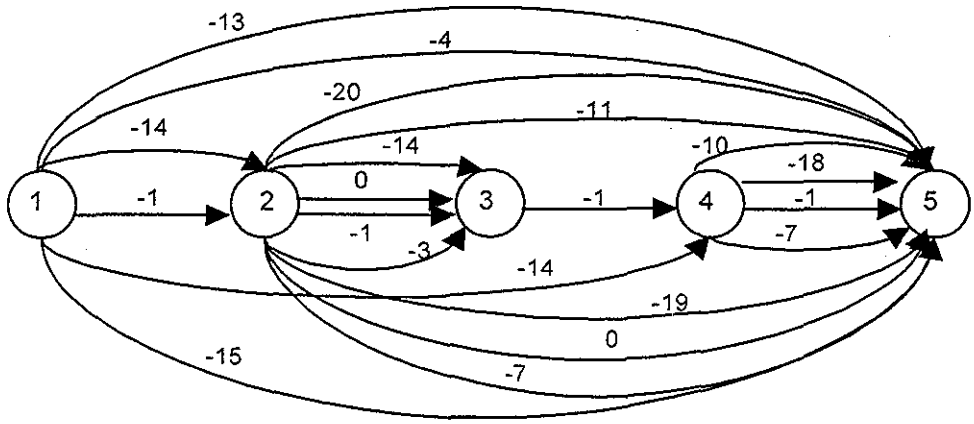


Figura 5.1.1.4

La otra red acíclica que se usó para el experimento es la tipo Acyc-P2N; esta red puede tener arcos negativos y positivos. Las longitudes de todos los arcos, incluyendo a los *arcos centrales,* se escogen aleatoriamente de un intervalo entre [*L*, *U*], donde *L* y *U* pueden tomar cualquier valor real. La idea usada en el experimento es controlar el porcentaje de arcos negativos por medio de los valores de *L* y *U*.

En la figura 5.1.1.5. se muestra una red tipo Acyc-P2N generada con los siguientes parámetros: n = 5 y m = 20. Al querer generar una red que tuviera aproximadamente el 25% de los arcos con longitud negativa, se asigna un valor L = -7 y U = 21.

Red Acyc-P2N

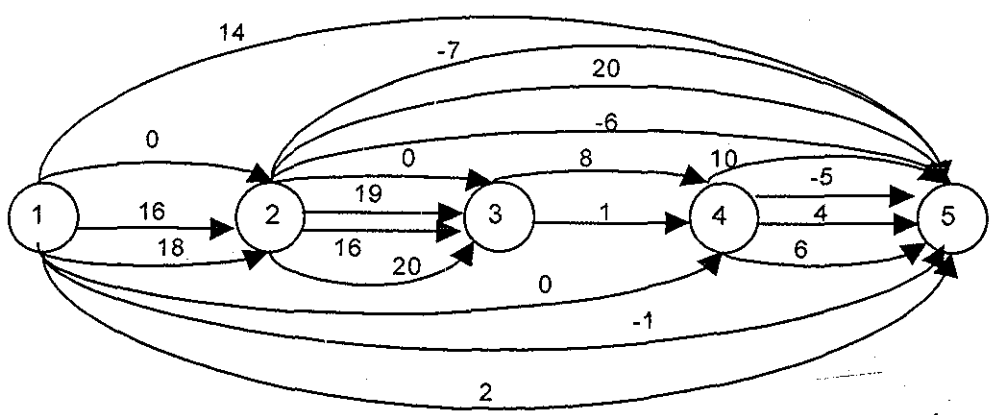


Figura 5.1.1.5

5.1.2 Evaluación

Se evalúa el algoritmo de **Dijkstra modificado** comparándolo con los algoritmos de **Tarjan**, **Simplex** y **Goldberg-Radzik**, los cuales, según el estudio realizado por Cherkassky y Goldberg (1999), son los algoritmos más eficientes para resolver el problema de ruta más corta modelado con redes cuyos arcos tengan longitudes negativas. Además, para observar la mejora que produce el nuevo algoritmo propuesto, se incluye en el estudio la variación del Dijkstra que se puede aplicar a redes con arcos negativos, mencionada en el capítulo II. Los resultados de la variación del Dijkstra se reportan con el nombre: **Variante Dijkstra**.

Es importante mencionar que la lista ordenada de los vértices etiquetados, necesaria para los algoritmos de *Variante Dijkstra* y *Dijkstra modificado* (en su primera etapa), se mantuvo con el uso de una "heap" de orden K, con K = 3 (Ahuja *et al.*, 1991).

Basándose en la metodología empleada por Cherkassky y Goldberg (1999) para calificar a los diferentes algoritmos, se tomaron como parámetros de comparación el tiempo de ejecución y el número de escaneos por nodo realizados por los algoritmos en las diferentes redes.

El tiempo que se reporta en la tabla de resultados está en segundos; y es el tiempo promedio obtenido a partir de cinco experimentos sobre redes aleatorias distintas generadas con los mismos parámetros (*n*, *m*, L, U, etc.). En la tabla de resultados se muestra el promedio del tiempo y el promedio del número de escaneos por nodo que realizaron los algoritmos con las cinco redes generadas. En caso de que el tiempo de ejecución excediera el límite (15 minutos) se interrumpió la ejecución y se reportan las siglas ELT (excedió el límite de tiempo).

El número de escaneos es una medida útil para determinar la eficiencia del algoritmo de forma independiente a la computadora usada. Por ejemplo, con este dato se puede observar el impacto que tienen las ideas implantadas en los algoritmos con la finalidad de reducir el número de escaneos.

El desempeño de cada uno de los algoritmos se calificó con los tiempos realizados en las redes más grandes de cada familia; y se usó la siguiente escala, la cual se basa en el estudio previo realizado por Cherkassky y Goldberg (1999):

bueno (○), regular (◉), pobre (⊗), malo (•),

si el código más rápido corrió en x segundos, un código que corrió en un intervalo de tiempo de [x, 4x] segundos se evaluó como bueno, de [4x, 16x] como regular, de [16x, 64x] como pobre y en un tiempo de más de 64x segundos se evaluó como malo.

5.2 Resultados

En esta sección se describen los resultados obtenidos para cada una de las redes generadas. En la siguiente tabla se reporta la calificación obtenida por cada uno de los algoritmos al resolver los diferentes tipos de redes.

Calificación de los algoritmos

Familia generadora de redes	Variante Dijkstra	Tarjan	Simplex	Dijkstra modificado	Goldberg-Radzik
Grid-NHard	•	○	○	○	○
Acyc-Neg	•	•	⊗	◉	○
Acyc-P2N	•	⊗	○	○	○
Rand P	◉	○	○	○	○

Tabla 5.2. Donde ○ significa bueno, ◉ regular, ⊗ pobre y • malo.

El algoritmo más robusto y eficiente fue el algoritmo *Goldberg-Radzik*, ya que obtuvo las más altas calificaciones. Este algoritmo también resultó ser el más eficiente en el trabajo de Cherkassky y Goldberg (1999).

El nuevo algoritmo propuesto, algoritmo de *Dijkstra modificado*, también resulta ser un algoritmo robusto y eficiente, a pesar de que para redes Acyc-Neg su desempeño fue regular.

La mejora que se logra con respecto a la *Variante* del algoritmo de *Dijkstra* es clara y contundente. Como podemos observar, el algoritmo *Variante Dijkstra*, tuvo el funcionamiento más pobre e ineficiente. En la mayoría de las redes excedió el límite de tiempo y esto se debe a la gran cantidad de escaneos que realiza por nodo.

El nuevo algoritmo propuesto cumple con el objetivo, ya que se logra un buen funcionamiento y eficiencia al resolver el problema de ruta más corta con arcos negativos. El ordenar los vértices a partir del subárbol generado para ser reescaneados resulta de gran ayuda para eficientar al algoritmo.

Por otro lado, el algoritmo *Simplex* a pesar de su mal desempeño en las redes tipo Acy-Neg resulta ser un algoritmo robusto. El algoritmo *Tarjan* resultó ser cuarto en la prueba.

En las siguientes secciones se muestran los resultados obtenidos para cada tipo de red.

5.2.1 Experimentos con la red Grid-NHard

En las tablas de resultados se presenta el promedio del tiempo (en "**negritas**") y de escaneos al resolver el problema en cinco redes distintas y aleatorias generadas con los mismos parámetros, como se comentó anteriormente.

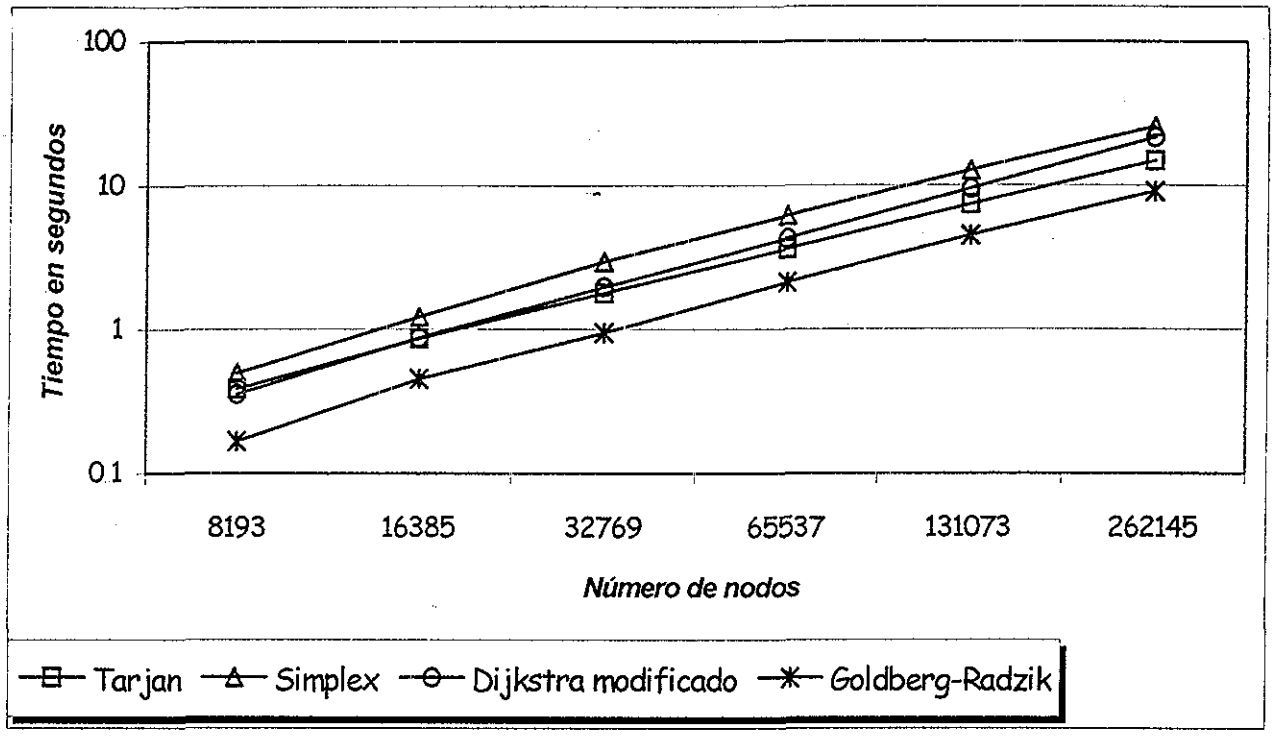
Los parámetros que se establecieron para generar todas las redes del tipo Grid-NHard se describen a continuación. El intervalo que se usó en todas las redes para obtener aleatoriamente las longitudes de los arcos que unen a las capas *y* fue [L_1=-10000, L_2=-1000]. Para el ciclo simple en las capas *y*, se tomó un valor de c = 1; y el intervalo para los valores de las longitudes de los arcos restantes fue de [0, U=100]. Los parámetros que variaron fueron el número de nodos *n* y el número de arcos *m*.

Tabla de resultados de la familia Grid-NHard

Nodos / Arcos	Variante Dijkstra	Tarjan	Simplex	Dijkstra modificado	Goldberg-Radzik
8193	ELT	**0.384**	**0.494**	**0.354**	**0.166**
63808		27.007	8.683	15.689	18.253
16385	ELT	**0.856**	**1.218**	**0.868**	**0.452**
129344		29.108	8 940	18 034	19 830
32769	ELT	**1.778**	**2.934**	**1.948**	**0.942**
260416		30.655	9.081	18.834	20.433
65537	ELT	**3.626**	**6.186**	**4.284**	**2.120**
522560		30 872	9.157	19.782	21.698
131073	ELT	**7.392**	**12.742**	**9.48**	**4.492**
1046846		31 315	9 091	21 139	22.557
262145	ELT	**14.808**	**25.718**	**21.554**	**9.126**
2095424		31.307	9.132	23 294	22.770

Tabla 5.2.1

Gráfica del tiempo de ejecución con la familia Grid-NHard



Gráfica 5.2.1

Como podemos observar en la tabla 5.2.1, el algoritmo *Variante Dijkstra* excedió el tiempo límite desde la primera red generada; lo cual confirma que este algoritmo es ineficiente para redes con arcos negativos.

Como se puede apreciar en la gráfica 5.2.1, el algoritmo de *Dijkstra modificado* y el de *Tarjan* tuvieron tiempos similares para las redes más pequeñas, pero al generar redes más grandes el algoritmo de *Tarjan* empleó menos tiempo en llegar a la solución, a pesar de que realizó un mayor número de escaneos. Estos dos algoritmos hicieron en promedio el doble de tiempo que el algoritmo de *Goldberg-Radzik*. El tiempo que realizó el algoritmo *Simplex* fue en promedio de un factor de 2.5 con respecto al mejor. En la gráfica se puede apreciar que los tiempos de ejecución de los cuatro algoritmos antes mencionados no están muy dispersos unos de otros.

En la tabla de resultados se observa que el algoritmo que realizó menos escaneos por nodo fue el *Simplex*. Sin embargo, hizo más tiempo que los algoritmos de *Tarjan, Dijkstra modificado y Goldberg-Radzik*; lo cual se debe a que el *Simplex,* al usar el método de pivoteo, requiere más tiempo para realizar cada pivote.

Por otra parte, el algoritmo de *Dijkstra modificado* realizó casi el mismo número de escaneos que el *Goldberg-Radzik*, pero se tardó el doble de tiempo, esto debido al costo de encontrar el nodo con la etiqueta de longitud mínima en su primera etapa.

El algoritmo de *Tarjan* fue el que realizó el mayor número de escaneos por nodo, pero con un tiempo similar al realizado por el algoritmo de *Dijkstra modificado*. Por lo que, se puede decir que al usar la disgregación del subárbol en una red con estructura compleja se provoca un mayor número de escaneos, pero cada operación se realiza rápidamente.

Para las redes tipo Grid-NHard el algoritmo que obtuvo el mejor tiempo fue el de *Goldberg-Radzik,* siendo 9.126 [s] su tiempo de ejecución en la red más grande. Por lo tanto, el rango que se especificó para que los algoritmos se calificaran como buenos va de 9.126 [s] a 36.504 [s]. Como podemos observar, todos los algoritmos, exceptuando la *variante Dijkstra,* cayeron dentro del rango; lo cual se reporta en la tabla 5.2.

5.2.2 Experimentos con la red Rand P

Los parámetros que se establecieron para generar todas las redes tipo Rand P se escriben a continuación. Se fijó el número de nodos a n = 131072 y el número de arcos a m = 524288.

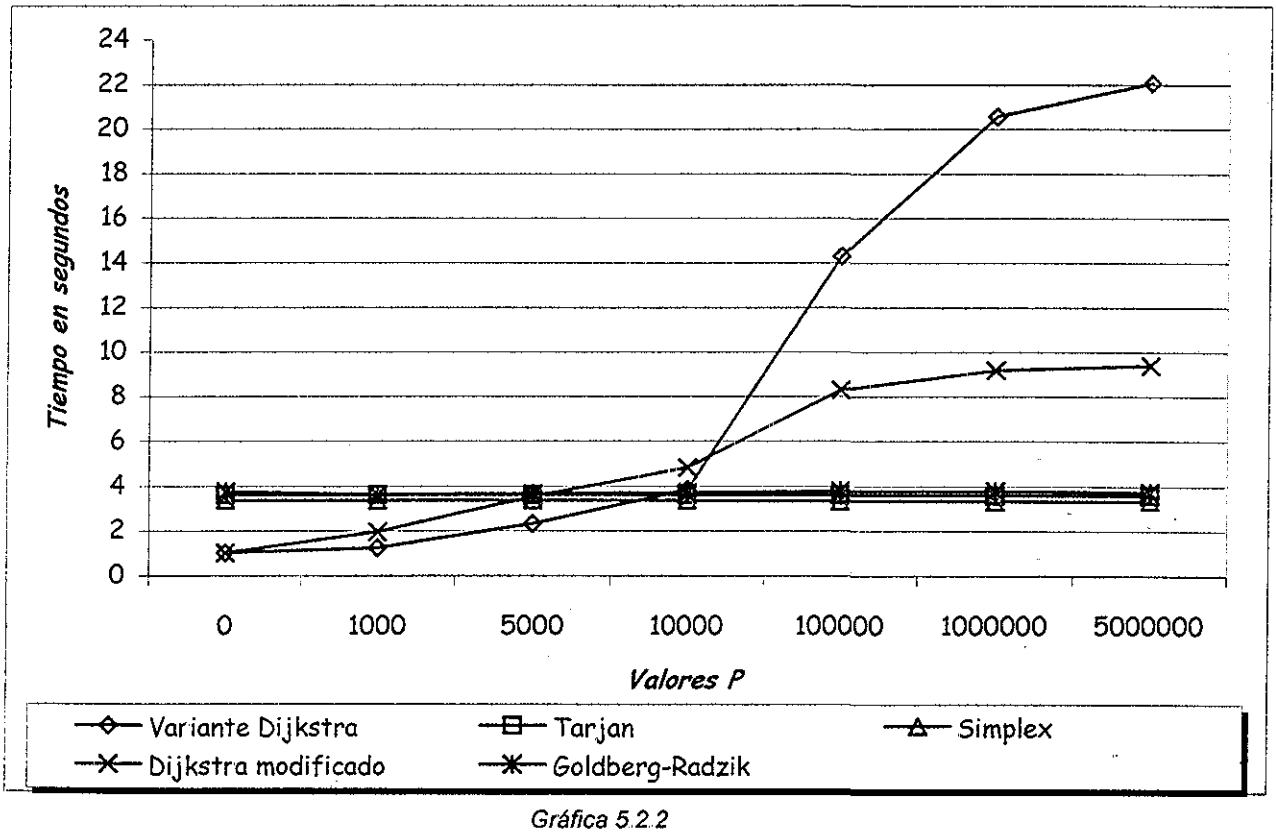
Como se comentó anteriormente, estas redes se generaron a partir de una red con arcos positivos. La red positiva contiene un ciclo básico que une a todos los nodos con arcos de longitud c =1. La longitud del resto de los arcos de la red inicial positiva se escoge aleatoriamente de un intervalo [0, U=10000]. En la etapa final de la generación se asigna una d(v) aleatoria tomada del intervalo [0, P] a cada nodo, donde P varía como se indica en la tabla 5.2.2. Posteriormente, se calculan los costos reducidos de los arcos y se dejan esos costos reducidos como la longitud de los arcos para la red; de modo que, entre mayor sea P, la red generada tendrá un mayor número de arcos con longitud negativa.

Tabla de resultados de la familia Rand-P

P	Nodos / Arcos	Variante Dijkstra	Tarjan	Simplex	Dijkstra modificado	Goldberg-Radzik
0	131072	1.012	3.580	3.350	0.998	3.734
	524288	1.000	10.477	8.974	1.000	19.479
1000	131072	1.228	3.602	3.34	1.944	3.592
	524288	1.280	10.477	8 974	3.462	18.892
5000	131072	2.294	3.588	3.352	3.506	3.646
	524288	2.992	10.477	8.974	7.090	19.212
10000	131072	3.822	3.582	3.348	4.810	3.678
	524288	5 554	10.477	8.974	9.831	19.417
100000	131072	14.292	3.59	3.328	8.294	3.776
	524288	23 723	10.477	8 974	16.087	19 760
1000000	131072	20.556	3.59	3.318	9.184	3.788
	524288	33 801	10.477	8 974	17 050	19 782
5000000	131072	22.07	3.582	3.338	9.392	3.746
	524288	35.931	10.477	8.974	17.263	19.532

Tabla 5.2.2

Gráfica del tiempo de ejecución con la familia Rand-P



Gráfica 5.2.2

Para este tipo de red, la *Variante Dijkstra* sí logra terminar en todas las pruebas realizadas. Este algoritmo es el mejor algoritmo cuando la cantidad de arcos negativos es baja; sin embargo es el peor algoritmo cuando la cantidad de arcos negativos es alta.

El algoritmo de *Dijkstra modificado* tiene un comportamiento similar al *Variante Dijkstra*: tiene un desempeño excelente en redes con pocos arcos negativos, y aunque su tiempo de ejecución se incrementa cuando se tienen más arcos negativos, se logra una mejora notable con respecto al algoritmo *Variante Dijkstra*, y se califica como bueno según los criterios de tabla 5.2.

El tiempo de ejecución del algoritmo *Dijkstra modificado* en el experimento con P = 5,000,000 fue de un factor de 3 comparado con el mejor tiempo realizado, el cual fue obtenido por el algoritmo *Simplex*. Observe que los algoritmos *Tarjan* y *Goldberg-Radzik* obtuvieron tiempos similares al *Simplex*.

En cuanto al número de escaneos, que se muestran en la tabla 5.2.2, los algoritmos de *Tarjan, Simplex* y *Goldberg-Radzik* tienen, cada uno, el mismo número de escaneos por nodo para todas las redes generadas. Lo anterior se debe a que estos algoritmos son invariantes a un cambio potencial (Cherkassky *et al.*, 1996); y en la generación de estas redes se realiza un cambio potencial en la etapa final. En una red generada a partir de un cambio de potencial, la diferencia que existe entre dos caminos distintos entre el mismo par de nodos no cambia (Cherkassky *et al.*, 1996).

Observe que para P = 0, tanto el algoritmo *Variante Dijkstra* como el *Dijkstra modificado* realizaron un solo escaneo por nodo, ya que la red tiene puros arcos positivos. Al incrementarse P se incrementó el número de escaneos, observándose que estos algoritmos no son invariantes a un cambio de potencial. El nuevo algoritmo realiza menos escaneos que la variante clásica del Dijkstra, mostrando nuevamente las ventajas del nuevo algoritmo.

5.2.3 Experimentos con la red Acyc-Neg

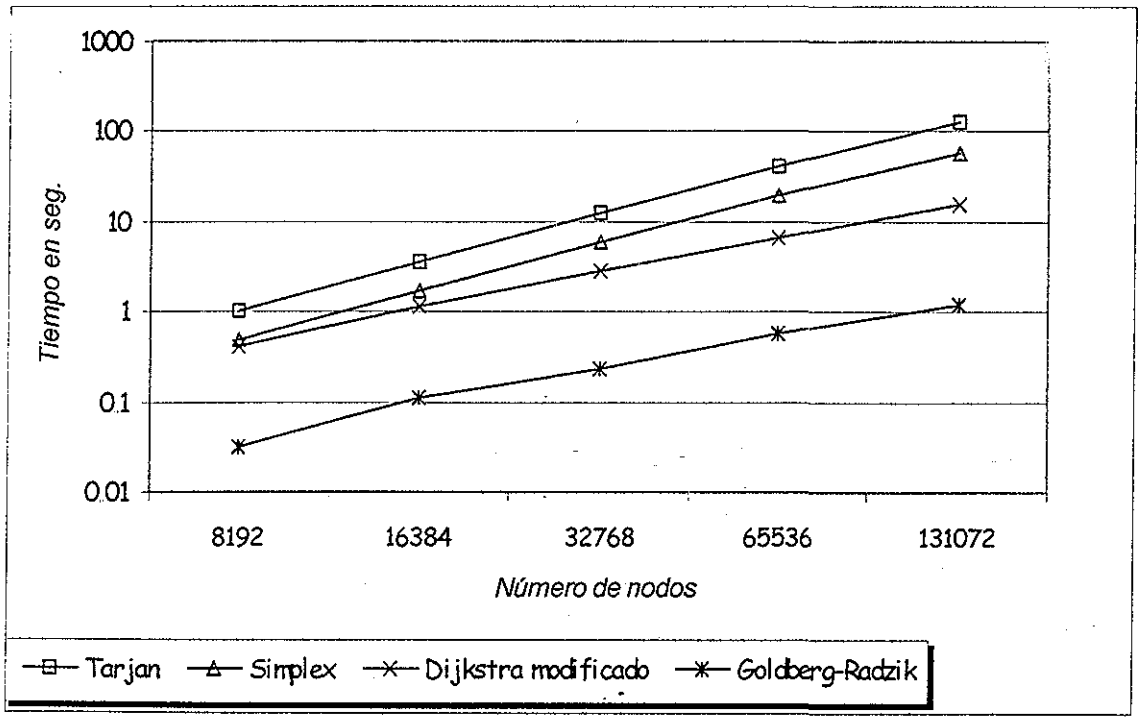
Los parámetros que se establecieron para generar todas las redes tipo Acyc-Neg se escriben a continuación. El número de nodos y arcos utilizados se muestra en la tabla 5.2.3. El valor de la longitud de los *arcos centrales* es *c* = -1 y el resto de los arcos toman una longitud aleatoria del intervalo [L = -10000, 0].

Tabla de resultados de la familia Acyc-Neg

Nodos / Arcos	Variante Dijkstra	Tarjan	Simplex	Dijkstra modificado	Goldberg-Radzik
8192	**180.794**	**1.012**	**0.47**	**0.418**	**0.032**
131072	9038.748	45.211	12 381	13.609	2.000
16384	**ELT**	**3.516**	**1.668**	**1.122**	**0.11**
262144		64.834	14 527	16.030	2 000
32768	**ELT**	**12.436**	**6.042**	**2.798**	**0.232**
524288		92.765	16.187	17 047	2 000
65536	**ELT**	**41.258**	**19.74**	**6.534**	**0.56**
1048576		130.204	18.417	18.165	2.000
131072	**ELT**	**130.998**	**59.066**	**15.554**	**1.208**
2097152		185.733	20.559	19.502	2.000

Tabla 5.2.3

Gráfica del tiempo de ejecución con la familia Acyc-Neg



Gráfica 5.2.3

Como podemos observar en la tabla 5.2.3, el algoritmo *Variante Dijkstra* excedió el límite de tiempo a partir de la segunda red generada. Mostrando nuevamente su ineficiencia al tratar de resolver redes con arcos cuyas longitudes son negativas.

Como podemos apreciar en la gráfica 5.2.3, el algoritmo que logró el mejor tiempo de ejecución por encima de los demás algoritmos fue el *Goldberg-Radzik*.

Como se indica en la tabla 5.2 y la tabla 5.2.3, es en este tipo de redes donde se tiene el peor desempeño de los algoritmos *Tarjan, Simplex* y *Dijkstra modificado*. Sin embargo, es importante notar que el nuevo algoritmo obtuvo el segundo lugar, siendo superado únicamente por el *Goldberg-Radzik*.

En cuanto al número de escaneos, el algoritmo de *Goldberg-Radzik* para todas las redes generadas realizó dos escaneos por vértice, con lo que se comprueba que al usar un ordenamiento topológico en redes acíclicas se minimiza el trabajo de escaneo y se eficienta el algoritmo (Cherkassky *et al.*, 1996).

El número de escaneos por nodo que realizaron los algoritmos de *Dijkstra modificado* y *Simplex* fue muy similar; sin embargo, el tiempo de ejecución para el primer algoritmo resultó mucho menor que el del segundo. Esto muestra que la técnica de pivoteo es mucho más costosa que el mantenimiento de una lista ordenada para escoger el nodo etiquetado con d(v) mínima (primera etapa *Dijkstra modificado*) y la generación de una lista de reescaneo (segunda etapa *Dijkstra modificado*).

El algoritmo *Tarjan* obtuvo una calificación *pobre*, mostrando que la disminución del número de escaneos que resulta de la disgregación del subárbol no es suficiente para tener un buen desempeño en este tipo de redes.

5.2.4 Experimentos con la red Acyc-P2N

En el experimento previo se observó que el desempeño de todos los algoritmos, excepto el de *Goldberg-Radzik,* fue bajo para redes acíclicas con arcos negativos. En este experimento se estudiará el desempeño de los algoritmos en redes acíclicas, pero variando la fracción de arcos negativos.

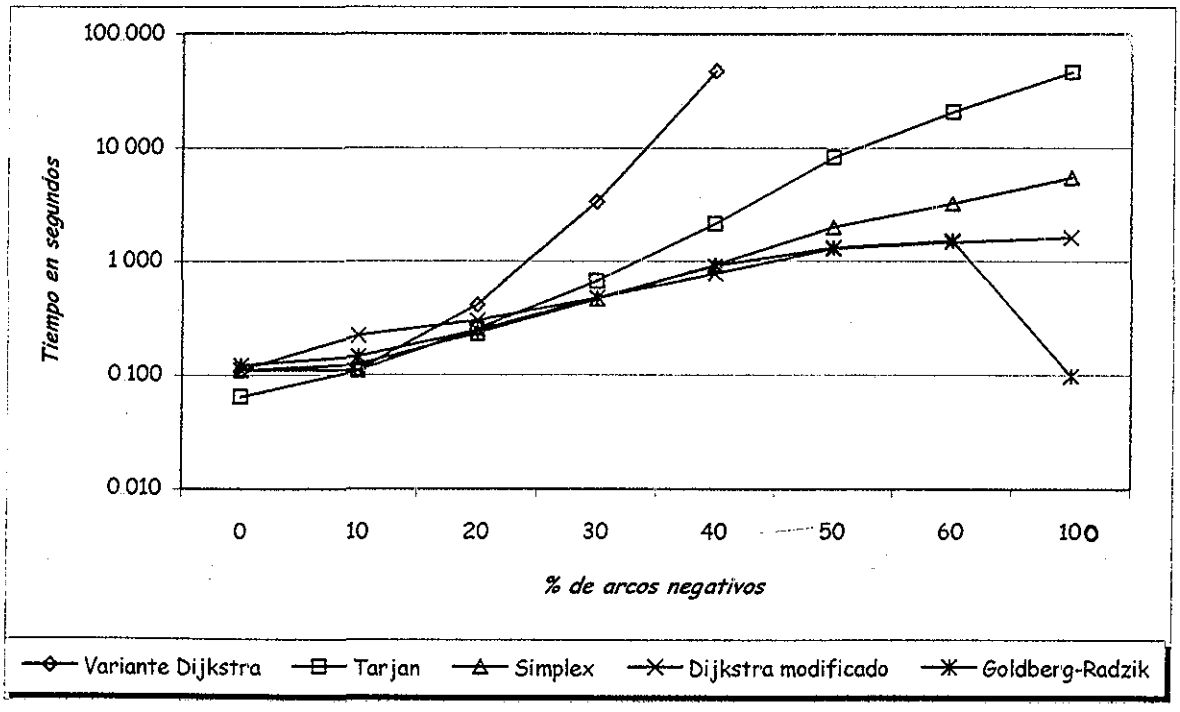
Los parámetros que se establecieron para generar todas las redes del tipo Acyc-P2N se describen a continuación. En todos las redes se fijaron el número de nodos n = 16384 y el número de arcos m = 262144. La longitud de los arcos (incluyendo los *arcos centrales*) se tomó aleatoriamente del intervalo [L, U], donde L y U se variaron para controlar el porcentaje f de arcos negativos en la red. El valor mínimo de L fue de -10000 y el valor máximo de U fue de 10000. Las calificaciones otorgadas en la tabla 5.2 se tomaron del experimento con f = 50%, ya que con f = 100% se caen en la familia anterior.

Tabla de resultados de la familia Acyc-P2N

f(%)	Nodos / Arcos	Variante de Dijkstra	Tarjan	Simplex	Dijkstra modificado	Goldberg-Radzik
0	**16384**	**0.110**	**0.064**	**0.110**	**0.110**	**0.120**
	262144	1.000	1.641	1.613	1.000	2.606
10	**16384**	**0.110**	**0.108**	**0.122**	**0.220**	**0.144**
	262144	1.140	2.145	2.086	2.040	3.014
20	**16384**	**0.408**	**0.250**	**0.230**	**0.298**	**0.244**
	262144	6.582	5.388	4.525	3.653	6.294
30	**16384**	**3.338**	**0.658**	**0.460**	**0.472**	**0.472**
	262144	63.556	14.432	8.011	6.615	12.486
40	**16384**	**47.216**	**2.124**	**0.920**	**0.770**	**0.900**
	262144	978.743	43.655	13.208	11.278	22.810
50	**16384**	**ELT**	**8.228**	**1.974**	**1.264**	**1.306**
	262144		153.537	21.250	17.945	30.810
60	**16384**	**ELT**	**20.55**	**3.206**	**1.452**	**1.504**
	262144		371.177	30.986	20.587	33.520
100	**16384**	**ELT**	**46.312**	**5.482**	**1.622**	**0.098**
	262144		723.441	70.367	22.234	2.000

Tabla 5.2.4

Gráfica del tiempo de ejecución con la familia Acyc-P2N



Gráfica 5.2.4

El algoritmo *Variante Dijkstra* logró el mejor desempeño en redes cuyo porcentaje de arcos negativos es menor al 10%. Sin embargo, al aumentar el porcentaje de arcos negativos su desempeño empobreció drásticamente e incluso excedió el límite de tiempo en redes con más de 50% de arcos negativos, como se muestra en la tabla 5.2.4.

Como se puede observar en la tabla 5.2.4 y en la gráfica 5.2.4, el algoritmo *Dijkstra modificado* mejora por mucho a la *Variante* clásica de *Dijkstra*, logrando tiempos muy bajos en todas las pruebas; incluso es el mejor algoritmo en tres de ellas (40% - 60%). Cuando se tienen un 100% de arcos negativos, el algoritmo propuesto, *Dijkstra modificado*, es sólo superado por el algoritmo de *Goldberg-Radzik*.

Todos los algoritmos degradan su desempeño conforme se aumenta el porcentaje de arcos negativos. Por ejemplo, el *Tarjan* pasa de ser el mejor algoritmo al penúltimo en la prueba. El algoritmo de *Dijkstra modificado*, también degrada su desempeño pero conserva buenos tiempos de ejecución.

Como se comprobó en el experimento anterior, el mejor algoritmo para redes acíclicas con el 100% de arcos negativos es el *Goldberg-Radzik*; incluso en este experimento se repite la marca de dos escaneos por nodo. Para los experimentos con ***f*** menor al 60%, el algoritmo con el menor número de escaneos fue el algoritmo de *Dijkstra modificado*; mostrando la mejora otorgada por la segunda etapa del algoritmo, en la que se reescanean en forma ordenada los subárboles generados en la primera etapa.

El algoritmo que realizó más escaneos por nodo, sin tomar en cuenta al algoritmo *Variante Dijkstra* es el algoritmo de *Tarjan*, el cual utiliza una técnica de disgregación del subárbol.

VI CONCLUSIONES Y RECOMENDACIONES

6.1 Conclusiones

El algoritmo de *Dijkstra modificado* supera por mucho los tiempos realizados por el algoritmo *Variante Dijkstra*. Así, **se cumplió con el objetivo establecido, al obtenerse un nuevo algoritmo basado en el algoritmo de Dijkstra cuyo desempeño, para resolver el problema de ruta más corta en redes con arcos de longitud arbitraria, es muy bueno**.

La idea de reescanear en forma ordenada al subárbol generado en la etapa de Dijkstra reduce considerablemente el número de escaneos por nodo con respecto a la *Variante* clásica de *Dijkstra* para redes con arcos negativos. Incluso, para la mayoría de las pruebas fue el segundo mejor algoritmo en los tiempos realizados y en el número de escaneos por nodo.

El algoritmo más robusto para la solución de redes con arcos negativos, de acuerdo a las pruebas realizadas, es el *Goldberg-Radzik*. El segundo mejor algoritmo es el *Dijkstra modificado* seguido del *Simplex* y el *Tarjan*, los cuales ocuparon el segundo y tercer lugar en las pruebas realizadas por Cherkassky y Goldberg (1999). El algoritmo con el peor desempeño, como era de esperarse, fue el *Variante Dijkstra* y se utilizó simplemente para observar la mejora que otorga el nuevo algoritmo propuesto, la cual resultó notable.

De acuerdo a los resultados obtenidos, se observa que las técnicas de reescaneo basadas en un cierto orden son muy útiles en la reducción de escaneos cuando se tienen redes con arcos de longitud negativa. El ordenamiento topológico usado en el *Goldberg-Radzik* (Goldberg y Radzik, 1993) es la técnica más robusta, seguida de la etapa de reescaneo ordenado de subárboles del *Dijkstra modificado*. Los algoritmos de *Tarjan* y *Simplex* no usan ningún tipo de ordenamiento y su desempeño es un poco más bajo, a pesar de que las técnicas usadas en estos algoritmos reducen notablemente el número de escaneos con respecto a su algoritmo padre: el Bellman-Ford-Moore (Cherkassky y Goldberg, 1999). De hecho, el número de escaneos por nodo del *Simplex* resultó ser bajo; sin embargo, el método de pivoteo es muy costoso y provoca que el tiempo empleado se

incremente. En el caso del *Tarjan* la técnica de disgregación del subárbol también es costosa y además, el número de escaneos no baja tanto como en el *Simplex*.

A pesar de que el algoritmo más robusto en las pruebas resultó ser el *Goldberg-Radzik*, el *Dijkstra modificado* lo superó en redes con un porcentaje bajo de arcos negativos, como puede observarse en los experimentos de las familias Rand-P y Acyc-P2N. Esto se debe a que el *Dijkstra modificado* conserva la alta eficiencia del algoritmo de Dijkstra original para resolver problemas en redes no negativas. De modo que, se recomienda el uso del algoritmo de *Dijkstra modificado* cuando se sepa que el porcentaje de arcos negativos en la red es bajo.

6.2 Recomendaciones

Se sabe que el algoritmo de Dijkstra original, aplicado a redes con arcos de longitud no negativa, varía su desempeño dependiendo de la estructura utilizada para ordenar los nodos etiquetados (Cherkassky, Goldberg y Silverstein, 1999). En el presente trabajo se utilizó una "heap" de orden 3 para mantener la lista de vértices etiquetados en la primera etapa del *Dijkstra modificado*; sin embargo, no se probó el uso de otras estructuras. Como un siguiente paso de la investigación, se recomienda evaluar el algoritmo propuesto usando distintas estructuras para el mantenimiento de la lista de vértices etiquetados.

El algoritmo propuesto puede detectar ciclos de longitud negativa, como se mencionó en el capítulo IV. Sin embargo, en el presente trabajo no se investigó la eficiencia de los algoritmos en la detección de ciclos negativos. Como un siguiente paso en la investigación, se recomienda evaluar la eficiencia del algoritmo propuesto en la detección de ciclos negativos.

APÉNDICE A

En este apéndice se incluye el código en C del algoritmo *Dijkstra modificado*. Es importante mencionar que los programas de los algoritmos: *Variante Dijkstra, Simplex, Tarjan* y *Goldberg-Radzik* se obtuvieron de la internet en la dirección www.intertrust.com/star/goldberg/index. Estos programas se agregan en el D.C. anexo.

A.1 Programa del algoritmo *Dijkstra modificado*

Archivo lectura.c

Este archivo lee los datos de entrada y los guarda para posteriormente trabajar con ellos al ejecutar el algoritmo.

```
/*****************************************************************************/
/*  parse (...) :                                                            */
/*      1. Lee los problemas de ruta más corta en formato DIMACS   */
/*      2. Prepara la representación de los datos internos #1.      */
/*****************************************************************************/

/* Archivos que deben incluirse :
#include <stdlib.h>
#include <string.h>
#include <stdio.h>
#include "deftypes h""/

/* ---------------------------------------------------------- */
int parse( n_ad, m_ad, nodos_ad, arcos_ad, source_ad, nodo_min_ad )

/* Todos los parámetros son de salida */
long   *n_ad;                          /* dirección del número de nodos */
long   *m_ad;                          /* dirección del número de arcos */
nodo   **nodos_ad;                     /* dirección del arreglo de nodos*/
arco   **arcos_ad;                     /* dirección del arreglo de arcos*/
nodo   **source_ad;                    /* dirección del apuntador al nodo fuente */
long   *nodo_min_ad;                   /* dirección del nodo mínimo */
{

#define MAXLINE          100           /* longitud máxima del archivo de entrada */
#define ARC_FIELDS       3             /* no de campos en el arco de entrada  */
#define P_FIELDS         3             /* no de campos en la línea del problema */
#define PROBLEM_TYPE     "sp"          /* nombre del tipo de problema */

long        n,                         /* número interno de nodos */
            nodo_min,                  /* no mínimo de nodos  */
            nodo_max,                  /* no máximo de nodos */
            *primer_arco               /* arreglo interno que contiene
                                              - grado del nodo
```

```
                                                    - posición del primer arco de salida */
        *arco_tail,                 /* arreglo interno: colas de los arcos */
        source,                     /* no del nodo fuente */
        head, tail, i;              /* Variables temporales que llevan el no de nodos */

long    m,                          /* número interno de arcos */
        ultimo, arco_num,
        arco_nuevo_num;             /* Variables temporales que llevan el no de arcos */

nodo    *nodos,                     /* apuntador a la estucutura nodo */
        *head_p;

arco    *arcos,                     /* apuntador a la estructura arc */
        *arco_actual,
        *arco_nuevo;
long    longitud;                   /* longitud del arco actual */

long    no_lines=0,                 /* no de la linea de entrada actual */
        no_plines=0                 /* no de lineas del problema */
        no_nlines=0,                /* no de lineas del nodo(fuente)*/
        no_alines=0;                /* no de lineas de los arcos */

char    in_line[MAXLINE],           /* para leer la linea de entrada */
        pr_type[3];                 /* para leer el tipo de problema */

int     k,                          /* temporal */
        err_no;                     /* no de error detectado */

/* ---------- Mensajes de error y el número de error ---------- */
#define EN1  0
#define EN2  1
#define EN3  2
#define EN4  3
#define EN6  4
#define EN8  7
#define EN9  8
#define EN11 9
#define EN12 10
#define EN14 12
#define EN16 13
#define EN15 14
#define EN17 15
#define EN18 16
#define EN21 17
#define EN19 5
#define EN20 6
#define EN22 11

static char *err_mensaje[] ={
/* 0*/   "más de una línea para el problema",
/* 1*/   "El número de parámetros en la línea del problema esta mal",
/* 2*/   "No es un problema de ruta más corta",
/* 3*/   "Mal valor en un parámetro en la línea del problema",
/* 4*/   "No se puede resolver el problema por falta de memoria",
/* 5*/   "No hay los suficientes arcos en la entrada",
/* 6*/   "El nodo fuente no tiene arcos de salida",
/* 7*/   "La descripción del problema debe estar antes que la del nodo fuente",
/* 8*/   "Se tiene más de un nodo fuente",
/* 9*/   "Esta mal el número de parametros en la línea del nodo fuente",
/*10*/   "El valor de los parámetros esta mal en la línea del nodo fuente"
/*11*/   "El archivo de entrada esta vacío",
/*12*/   "La descripción del nodo fuente debe estar antes que la de los arcos",
/*13*/   "Se tienen muchos arcos en la entrada",
/*14*/   "El número de parámetros en la línea de arcos esta mal",
/*15*/   "El valor de los parámetros en la línea de arcos esta mal",
/*16*/   "No se conoce el tipo de línea en la entrada",
/*17*/   "Error en la lectura de datos" };
/* ------------------------------------------------------- */

/* El ciclo principal:
```

Se lee la línea de entrada, se analiza el tipo de línea, se checa que los parámetros esten correctos, se meten los datos en los arreglos
*/

```
while ( gets ( in_line ) != NULL ){
no_lines ++;
    switch (in_line[0]){
        case 'c':                   /* Saltate líneas comentadas */
        case '\n':                  /* Saltate líneas vacias     */
        case '\0':                  /* Saltate líneas vacias al final del archivo*/
        case 't':                   /* nombre del problema       */
        break;
        case 'p':                   /* Descripción del problema  */
            /* Si se tiene más de una línea para definir el problema */
            if ( no_plines > 0 ){
                err_no = EN1 ;
                goto error;
            }
            no_plines = 1;
            /* Leyendo la línea del problema: tipo de problema, no de nodos, no de arcos */
            /* El número de parámetros es incorrecto en la línea del problema */
            if (sscanf(in_line, "%*c %2s %ld %ld",pr_type, &n, &m )!= P_FIELDS ){
                err_no = EN2;
                goto error;
            }
            /* El tipo de problema esta mal */
            if ( strcmp ( pr_type, PROBLEM_TYPE ) ){
                err_no = EN3;
                goto error;
            }
            /* El no de arcos o nodos tiene un valor que no puede ser */
            if ( n <= 0  || m <= 0 ){
                err_no = EN4;
                goto error;
            }
            /* Reservando memoria para 'nodos', 'arcs' y arreglos internos */
            nodos           = (nodo*) calloc ( n+2, sizeof(nodo) );
            arcos           = (arco*) calloc ( m+1, sizeof(arco) );
            arco_tail   = (long*) calloc ( m,   sizeof(long) );
            primer_arco = (long*) calloc ( n+2, sizeof(long) );

            /* primer_arco [ 0   n+1 ] = 0 - Se inicializa por calloc */   /* No se pudo reservar memoria */
            if(nodos == NULL||arcos==NULL||primer_arco== NULL||arco_tail==NULL){
                err_no = EN6;
                goto error;
            }

            /* Apuntador al arco actual */
            arco_actual = arcos;
        break;

        case 'n':                   /* La descripción del nodo fuente(s) */
            /* No existe línea del problema */
            if ( no_plines == 0 ){
                err_no = EN8;
                goto error;
            }
            /* Se tiene más de una línea para el nodo fuente */
            if ( no_nlines != 0){
                err_no = EN9;
                goto error;
            }
            no_nlines = 1;
            /* Leyendo el nodo fuente */
            k = sscanf ( in_line,"%*c %ld %ld", &source );
            /* El número del nodo fuente no se leyó */
            if ( k < 1 ){
                err_no = EN11;
                goto error;
            }
            /* El nodo fuente tiene un valor que no esta bien */
```

```
                if ( source < 0 || source > n ){
                        err_no = EN12;
                        goto error;
                }
                nodo_max = 0;
                nodo_min = n;
        break;

        case 'a':               /* Descripción del arco */
                /* No se tiene la descripción del nodo fuente */
                if ( no_nlines == 0 ){
                        err_no = EN14;
                        goto error;
                }
                /* Hay demasiados arcos en la entrada */
                if ( no_alines >= m ){
                        err_no = EN16;
                        goto error;
                }
                /* Leyendo la descripción de los arcos */ /* La descripción de los arcos no es correcta */
                if ( sscanf ( in_line,"%*c %ld %ld %ld" &tail, &head, &longitud)!= 3){
                        err_no = EN15;
                        goto error;
                }
                /* Esta mal el valor de los nodos */
                if ( tail < 0  ||  tail > n  || head < 0  ||  head > n ){
                        err_no = EN17;
                        goto error;
                }
                primer_arco[tail + 1] ++;  /* no de arcos de salida de la cola que se almacenan en arc_first[tail+1] */
                /* Guardando la información de los arcos */
                arco_tail[no_alines]= tail;
                arco_actual -> head = nodos + head;
                arco_actual -> lon  = longitud;
                /* Buscando el nodo mínimo y máximo */
                if ( head < nodo_min ) nodo_min = head;
                if ( tail < nodo_min ) nodo_min = tail;
                if ( head > nodo_max ) nodo_max = head;
                if ( tail > nodo_max ) nodo_max = tail;
                no_alines ++;
                arco_actual ++;
        break;
        default:
        /* No se conoce el tipo de línea */
        err_no = EN18;
        goto error;
        break;
        } /* Fin del switch */
  }  /* Fin del loop */

  /* —— Todo se leyó o se tuvo error mientras se leía —— */
  if ( feof (stdin) == 0 ){ /* error de lectura */
        err_no=EN21;
        goto error;
  }
  if ( no_lines == 0 ){ /* Entrada vacía */
        err_no = EN22;
        goto error;
  }
  if ( no_alines < m ){ /* No hay suficientes arcos */
        err_no = EN19;
        goto error;
  }

/********** Ordenando los arcos - Tiempo del algoritmo **********/
  /* El primer arco del primer nodo */
  ( nodos + nodo_min ) -> primero = arcos;
  for ( i = nodo_min + 1; i <= nodo_max + 1; i ++ ){
        primer_arco[i]        += primer_arco[i-1];
        ( nodos + i ) -> primero = arcos + primer_arco[i];
```

```
        }
        /* Escaneando todos los nodos excepto el último */
        for ( i = nodo_min; i < nodo_max; i ++ ){
                ultimo = ( ( nodos + i + 1 ) -> primero ) - arcos;
                /*los arcos que salen de i deben ser llamados de la posición arc_first[i] a la posición igual al valor inicial de arc_first[i+1]-1*/
                for ( arco_num = primer_arco[i]; arco_num < ultimo; arco_num ++ ){
                        tail = arco_tail[arco_num];
                        /* El no del arco arc_num no esta en su lugar porque arc que es llamado debe salir de i, se pone en su lugar y se
                        continua con el proceso  hasta que un arco en esta posición sale de i */
                        while ( tail != i ){
                                arco_nuevo_num  = primer_arco[tail];
                                arco_actual     = arcos + arco_num;
                                arco_nuevo      = arcos + arco_nuevo_num;
                                /* arc_current debe ser llamado en la posición arc_new cambiando estos arcos: */
                                head_p                          = arco_nuevo -> head;
                                arco_nuevo -> head      = arco_actual -> head;
                                arco_actual -> head     = head_p;
                                longitud                = arco_nuevo -> lon;
                                arco_nuevo -> lon       = arco_actual -> lon;
                                arco_actual -> lon      = longitud;
                                arco_tail[arco_num]     = arco_tail[arco_nuevo_num];
                                /*Se incrementa arc_first[tail] pero se etiqueta la posición previa*/
                                arco_tail[arco_nuevo_num]       = tail;
                                primer_arco[tail] ++ ;
                                tail                            = arco_tail[arco_num];
                        }
                }
        /* Todos los arcos que salen de i están en su lugar */
        }
        /* ---------- Los arcos están ordenados ---------- */
        /* Asignando valores de salida */
        *m_ad = m;
        *n_ad = nodo_max - nodo_min + 1;
        *source_ad = nodos + source;
        *nodo_min_ad = nodo_min;
        *nodos_ad = nodos + nodo_min;
        *arcos_ad = arcos;

        /* Error en el valor del nodo fuente */
        if ( source < nodo_min || source > nodo_max ){
                err_no = EN20;
                goto error;
        }
        /* Ningun arco sale del nodo fuente */
        if ( (*source_ad) -> primero == ((*source_ad) + 1) -> primero ){
                err_no = EN20;
                goto error;
        }
        /* Memoria interna liberada */
        free ( primer_arco ); free ( arco_tail );
        /* Todo esta hecho */
        return (0);
  /* ---------- */
        error:  /* error encontrado al leer el archivo de entrada */
        printf ( "\nLine %d of input - %s\n", no_lines, err_mensaje[err_no] );
        exit (1);
}
/* ----------  FIN DE PARSER  ----------*/

/*La siguiente función es practicamente identica a parse, solo que se modificó para que el programa pudiera leer directamente de un
archivo sin redireccionamiento. Esto sirve para poder debuggear los programas desde Borland c++ 5.0 */
/* Las diferencias se encuentran donde aparezca el apuntador a archivo:entrada*/

int fparse( n_ad, m_ad, nodos_ad, arcos_ad  source_ad, nodo_min_ad,entrada )

FILE *entrada;
/* Todos los parámetros son de salida */
long   *n_ad;                   /* dirección del número de nodos  */
long   *m_ad;                   /* dirección del número de arcos  */
nodo  **nodos_ad;               /* dirección del arreglo de nodos */
```

```
arco  **arcos_ad;                    /* dirección del arreglo de arcos */
nodo  **source_ad;                   /* dirección del apuntador al nodo fuente */
long  *nodo_min_ad;                  /* dirección del nodo mínimo     */
{

#define MAXLINE         100          /* longitud máxima del archivo de entrada */
#define ARC_FIELDS      3            /* no de campos en el arco de entrada  */
#define P_FIELDS        3            /* no de campos en la línea del problema */
#define PROBLEM_TYPE "sp"            /* nombre del tipo de problema */

long        n,                       /* número interno de nodos */
            nodo_min,                /* no mínimo de nodos  */
            nodo_max,                /* no máximo de nodos */
            *primer_arco             /* arreglo interno que contiene
                                                - nodo degree
                                                - posición del primer arco de salida */
            *arco_tail,              /* arreglo interno: colas de los arcos */
            source,                  /* no del nodo fuente */
            head, tail, i;           /* Variables temporales que llevan el no de nodos */

long        m,                      /* número interno de arcos */
            ultimo, arco_num,
            arco_nuevo_num;          /* Variables temporales que llevan el no de arcos */

nodo        *nodos,                  /* apuntador a la estucutura nodo */
            *head_p;

arco        *arcos,                  /* apuntador a la estructura arc */
            *arco_actual,
            *arco_nuevo;

long        longitud;                /* longitud del arco actual */

long        no_lines=0,              /* no de la linea de entrada actual */
            no_plines=0,             /* no de lineas del problema */
            no_nlines=0,             /* no de lineas del nodo(fuente)*/
            no_alines=0;             /* no de lineas de los arcos */

char        in_line[MAXLINE],        /* para leer la linea de entrada */
            pr_type[3];              /* para leer el tipo de problema */

int         k,                       /* temporal */
            err_no;                  /* no de error detectado */
 /* ---------- Número de error y mensajes de error ---------- */
#define EN1  0
#define EN2  1
#define EN3  2
#define EN4  3
#define EN6  4
#define EN8  7
#define EN9  8
#define EN11 9
#define EN12 10
#define EN14 12
#define EN16 13
#define EN15 14
#define EN17 15
#define EN18 16
#define EN21 17
#define EN19 5
#define EN20 6
#define EN22 11
static char *err_mensaje[] = {
/* 0*/  "más de una línea para el problema",
/* 1*/  "El número de parámetros en la línea del problema esta mal",
/* 2*/  "No es un problema de ruta más corta",
/* 3*/  "Mal valor en un parámetro en la línea del problema",
/* 4*/  "No se puede resolver el problema por falta de memoria",
/* 5*/  "No hay los suficientes arcos en la entrada",
/* 6*/  "El nodo fuente no tiene arcos de salida",
```

```
/* 7*/   "La descripción del problema debe estar antes que la del nodo fuente",
/* 8*/   "Se tiene más de un nodo fuente",
/* 9*/   "Esta mal el número de parametros en la línea del nodo fuente",
/*10*/   "El valor de los parámetros esta mal en la línea del nodo fuente",
/*11*/   "El archivo de entrada esta vacío",
/*12*/   "La descripción del nodo fuente debe estar antes que la de los arcos",
/*13*/   "Se tienen muchos arcos en la entrada",
/*14*/   "El número de parámetros en la línea de arcos esta mal",
/*15*/   "El valor de los parámetros en la línea de arcos esta mal",
/*16*/   "No se conoce el tipo de línea en la entrada",
/*17*/   "Error en la lectura de datos" };

/* ---------------------------------------------------------- */
/* El ciclo principal: Se lee la llínea de entrada, se analiza el tipo de línea, se checa que los parámetros esten correctos y se meten los
datos en los arreglos. */
        while ( fgets ( in_line, 500,entrada ) != NULL ){
        no_lines ++;
                switch (in_line[0]){
                        case 'c':               /* Saltate líneas comentadas */
                        case \n':               /* Saltate líneas vacias     */
                        case '\0':              /* Saltate líneas vacias al final del archivo*/
                        case 't':              /* nombre del problema       */
                        break;
                        case 'p':               /* Descripción del problema  */
                                /* Si se tiene más de una línea para definir el problema */
                                if ( no_plines > 0 ){
                                        err_no = EN1 ;
                                        goto error;
                                }
                                no_plines = 1;
                                /* Leyendo la linea del problema:tipo de problema, no. nodos, no  arcos */
                                /* El número de parámetros es incorrecto en la línea del problema */
                                if (sscanf(in_line, "%*c %2s %ld %ld",pr_type, &n, &m )!= P_FIELDS ){
                                        err_no = EN2;
                                        goto error;
                                }
                                /* El tipo de problema esta mal */
                                if ( strcmp ( pr_type, PROBLEM_TYPE ) ){
                                        err_no = EN3;
                                        goto error;
                                }
                                /* El no de arcos o nodos tiene un valor que no puede ser */
                                if ( n <= 0  || m <= 0 ){
                                        err_no = EN4;
                                        goto error;
                                }
                                /* Reservando memoria para 'nodos', 'arcs' y arreglos internos */
                                nodos           = (nodo*) calloc ( n+2, sizeof(nodo) );
                                arcos           = (arco*)  calloc ( m+1, sizeof(arco) );
                                arco_tail       = (long*) calloc ( m,   sizeof(long) );
                                primer_arco     = (long*) calloc ( n+2, sizeof(long) );
                                /* arc_first [ 0 .. n+1 ] = 0 - Se inicializa por calloc */ /* No se pudo reservar memoria */
                                if(nodos == NULL||arcos == NULL||primer_arco== NULL||arco_tail == NULL){
                                        err_no = EN6;
                                        goto error;
                                }
                                /* Apuntador al arco actual */
                                arco_actual = arcos;
                        break;

                        case 'n':               /* La descripción del nodo fuente(s) */
                        /* No existe línea del problema */
                        if ( no_plines == 0 ){
                                err_no = EN8;
                                goto error;
                        }
                        /* Se tiene más de una línea para el nodo fuente */
                        if ( no_nlines != 0){
                                err_no = EN9;
                                goto error;
```

```
                }
                no_nlines = 1;
                /* Leyendo el nodo fuente */
                k = sscanf ( in_line,"%*c %ld %ld", &source );
                /* El número del nodo fuente no se leyó */
                if ( k < 1 ){
                        err_no = EN11;
                        goto error;
                }
                /* El nodo fuente tiene un valor que no esta bien */
                if ( source < 0 || source > n ){
                        err_no = EN12;
                        goto error;
                }
                nodo_max = 0;
                nodo_min = n;
        break;

        case 'a':               /* Descripción del arco */
        /* No se tiene la descripción del nodo fuente */
                if ( no_nlines == 0 ){
                        err_no = EN14;
                        goto error;
                }
                /* Hay demasiados arcos en la entrada */
                if ( no_alines >= m ){
                        err_no = EN16;
                        goto error;
                }
                /* Leyendo la descripción de los arcos */
                /* La descripción de los arcos no es correcta */
                if ( sscanf ( in_line,"%*c %ld %ld %ld",&tail  &head, &longitud)!= 3){
                        err_no = EN15;
                        goto error;
                }
                /* Esta mal el valor de los nodos */
                if ( tail < 0  ||  tail > n  || head < 0  ||  head > n ){
                        err_no = EN17;
                        goto error;
                }
                primer_arco[tail + 1] ++;/* no de arcos de salida de la cola que se almacenan en primer_arco[tail+1] */
                /* Guardando la información de los arcos */
                arco_tail[no_alines]        = tail;
                arco_actual -> head         = nodos + head;
                arco_actual -> lon          = longitud;
                /* Buscando el nodo mínimo y máximo */
                if ( head < nodo_min ) nodo_min = head;
                if ( tail < nodo_min ) nodo_min   = tail;
                if ( head > nodo_max ) nodo_max = head;
                if ( tail > nodo_max ) nodo_max = tail;
                no_alines ++;
                arco_actual ++;
        break;
        default:
        /* No se conoce el tipo de línea */
        err_no = EN18;
        goto error;
        break;
        } /* Fin del switch */
}   /* Fin del ciclo loop */

/* —— Todo se leyó o se tuvo error mientras se leía —— */
if ( feof (entrada) == 0 ){ /* error de lectura */
        err_no=EN21;
        goto error;
}
if ( no_lines == 0 ){ /* Entrada vacía */
        err_no = EN22;
        goto error;
}
```

```
        if ( no_alines < m ){ /* No hay suficientes arcos */
                err_no = EN19;
                goto error;
        }

    /********** Ordenando los arcos - Tiempo del algoritmo **********/
        /* El primer arco del primer nodo */
        ( nodos + nodo_min ) -> primero = arcos;
        for ( i = nodo_min + 1; i <= nodo_max + 1; i ++ ){
                primer_arco[i]                  += primer_arco[i-1];
                ( nodos + i ) -> primero    = arcos + primer_arco[i];
        }
        /* Escaneando todos los nodos excepto el último */
        for ( i = nodo_min; i < nodo_max; i ++ ){
                ultimo = ( ( nodos + i + 1 ) -> primero ) - arcos;
                /* los arcos que salen de i deben ser llamados de la posición primer_arco[i] a la posición igual al valor inicial de
                primer_arco[i+1]-1  */
                for ( arco_num = primer_arco[i]; arco_num < ultimo; arco_num ++ ){
                        tail = arco_tail[arco_num];
                        /* El no del arco arco_num no esta en su lugar porque arc que es llamado debe salir de i, se pone en su lugar y se
                        continua con el proceso  hasta que un arco en esta posición sale de i */
                        while ( tail != i ){
                                arco_nuevo_num  = primer_arco[tail];
                                arco_actual     = arcos + arco_num;
                                arco_nuevo      = arcos + arco_nuevo_num;
                                /* arc_actual debe ser llamado en la posición arc_new cambiando estos arcos: */
                                head_p          = arco_nuevo -> head;
                                arco_nuevo  -> head     = arco_actual -> head;
                                arco_actual -> head     = head_p;
                                longitud                = arco_nuevo -> lon;
                                arco_nuevo -> lon       = arco_actual -> lon;
                                arco_actual -> lon      = longitud;
                                arco_tail[arco_num]     = arco_tail[arco_nuevo_num];
                                /*Se incrementa primer_arco[tail] pero se etiqueta la posición previa*/
                                arco_tail[arco_nuevo_num] = tail;
                                primer_arco[tail] ++ ;
                                tail = arco_tail[arco_num];
                        }
                }
                /* Todos los arcos que salen de i están en su lugar */
        }
/* ---------- Los arcoos están ordenados ---------- */
        /* Asignando valores de salida */
        *m_ad = m;
        *n_ad = nodo_max - nodo_min + 1;
        *source_ad = nodos + source;
        *nodo_min_ad = nodo_min;
        *nodos_ad = nodos + nodo_min;
        *arcos_ad = arcos;

        /* Error en el valor del nodo fuente */
        if ( source < nodo_min || source > nodo_max ){
                err_no = EN20;
                goto error;
        }
        /* Ningun arco sale del nodo fuente */
        if ( (*source_ad) -> primero == ((*source_ad) + 1) -> primero ){
                err_no = EN20;
                goto error;
        }
        /* Memoria interna liberada */
        free ( primer_arco ); free ( arco_tail );
        /* Todo esta hecho */
        return (0);
        /* ---------- */
        error:  /* error encontrado al leer el archivo de entrada */
        printf ( "\nLine %d of input - %s\n"  no_lines, err_mensaje[err_no] );
        exit (1);
}
    /* ---------- FIN DE FPARSER ----------*/
```

Archivo def_globales.h

En este archivo se encuentran las variables globales que se utilizan en el algoritmo.

```
/* ---- Definición usadas para el código Dijkstra modificado ---- */
/*  Lo que significa cada variable en nuestro código
long   n                - el número de nodos en la red;
long   m                - el número de arcos;
long   nmin             - el número del nodo mínimo;
nodo*  primer_nodo      - apuntador al primer nodo de la red;
nodo*  ultimo_nodo      - ficción: nodo despues del último de la red;
arco*  primer_arco      - apuntador al primer arco;
arco*  ultimo_arco      - el siguiente arco despues del último de salida del node actual */

#define NNULL         (nodo*)NULL
#define INFINITO      1073741823
#define NUMERO_NODO(nodo) (long)( ( (nodo) - primer_nodo ) + nmin )
#define NUMERO_ARCO(arco) (long)( (arco) - primer_arco )
#define PARA_TODO_NODO( nodo )\
    ultimo_nodo = primer_nodo + n ;\
    for ( nodo  = primer_nodo; nodo != ultimo_nodo; nodo++ )
#define PARA_TODO_ARCO_DESDE_EL_NODO( nodo, arco )\
    ultimo_arco = ( ( nodo ) + 1 ) -> primero;\
    for ( arco = ( nodo ) -> primero; arco != ultimo_arco; arco++ )
#define PARA_TODO_ARCO_DESDE_EL_NODO_ORDENA( nodo, arco, empieza )\
    ultimo_arco = ( ( nodo ) + 1 ) -> primero;\
    for ( arco = empieza; arco != ultimo_arco; arco++ )
```

Archivo deftypes.h

En este archivo se define el tipo de estructura para los nodos y los arcos.

```
/* Tipo de estructuras utilizadas */

typedef  /* arco */
    struct arco_st{
        long            lon;      /* longitud del arco */
        struct nodo_st  *head;    /* nodo al que llega el arco */
    }arco;

typedef  /* nodo*/
    struct nodo_st{
        int             temp;          /* para etiquetas temporales */
        int             status;        /* status del nodo en el orden topológico */
        long            dist;          /* long. tentativa del camino más corto  */
        long            heap_pos;      /* número de posición en la heap */
        long            B_pos;         /* numero de posición en B*/
        struct nodo_st  *padre;        /* apuntador al padre */
        arco            *primero;      /* el primer arco de salida */
        arco            *actual;       /* arco actual en DFS del orden topológico */
        long            negs;          /* numero de arcos neg. en camino al nodo */
    } nodo;
```

Archivo Funcion_heap_2.h

Este archivo se usa para ordenar los nodos en la primera etapa o etapa de Dijkstra del algoritmo. Selecciona el nodo con la d(v) mínima.

```
/*******************  Definiciones de heap  *****************/

typedef /* heap */
```

```
  struct heap_st{
    long          size;   /* El número del último elemento de la heap */
    nodo          **nd;   /* heap de los apuntadores a nodos          */
}heap;

heap  d_heap;

long   h_actual_pos,
       h_new_pos,
       h_pos,
       h_ultimo_pos;

nodo  *nodo_j,
      *nodo_k;

long   dist_k,
       dist_min;

int    h_grado;

#define HEAP_POWER  2
#define NILL       -1
#define FIJO       -2
#define TRATO_MEJORAR  -3
#define NODO_EN_HEAP( nodo_i ) ( nodo_i -> heap_pos > NILL )
#define SIN_SUBIR( nodo_i ) ( nodo_i -> heap_pos == NILL )
#define NODO_FIJO( nodo_i ) ( nodo_i -> heap_pos == FIJO )
#define NODO_TRATO_MEJORAR( nodo_i ) ( nodo_i -> heap_pos == TRATO_MEJORAR )
#define CON_ELEMENTOS_EN_HEAP  ( d_heap.size > 0 )
#define PON_A_POS_EN_LA_HEAP( nodo_i, pos ){\
  d_heap.nd[pos]    = nodo_i;\
  nodo_i -> heap_pos = pos;\
}
#define EN_NEW_PASS  1

void Init_heap ( n  source )

nodo* source;
long  n;
{
       h_grado   = 1 << HEAP_POWER;
       printf("ddeg %ld\n", h_grado );
       d_heap.size = 1;
       d_heap.nd   = (nodo**) calloc ( (n+1), sizeof(nodo*) );
       PON_A_POS_EN_LA_HEAP( source, 0 )
}

void Heap_decrease_key ( nodo_i, dist_i )

nodo* nodo_i;
long  dist_i;

{
       for(h_actual_pos=nodo_i -> heap_pos;h_actual_pos>0;h_actual_pos=h_new_pos){
              h_new_pos  = ( h_actual_pos - 1 ) >> HEAP_POWER;
              nodo_j             = d_heap.nd[h_new_pos];
              if ( dist_i  >=  nodo_j -> dist )
       break;
       PON_A_POS_EN_LA_HEAP ( nodo_j  h_actual_pos )
       }
       PON_A_POS_EN_LA_HEAP ( nodo_i, h_actual_pos )
}

void Inserta_a_heap ( nodo_i )

nodo* nodo_i;

{
       PON_A_POS_EN_LA_HEAP ( nodo_i, d_heap.size )
       d_heap.size ++;
```

```
        Heap_decrease_key( nodo_i, nodo_i -> dist);
}

nodo* Extrae_min ( ){

nodo* nodo_0;

nodo_0                  = d_heap.nd[0];
nodo_0 -> heap_pos = FIJO;
d_heap.size -- ;

if ( d_heap.size > 0 ){
        nodo_k =  d_heap.nd [ d_heap.size ];
        dist_k =  nodo_k -> dist;
        h_actual_pos = 0;
        while ( 1 ){
                h_new_pos = ( h_actual_pos << HEAP_POWER ) +  1;
                if ( h_new_pos >= d_heap.size )
                break;
                dist_min                        = d_heap.nd[h_new_pos] -> dist;
                h_ultimo_pos        = h_new_pos + h_grado;
                if ( h_ultimo_pos > d_heap.size )
                        h_ultimo_pos = d_heap.size;
                        for ( h_pos = h_new_pos + 1; h_pos < h_ultimo_pos; h_pos ++ ){
                                if ( d_heap.nd[h_pos] -> dist < dist_min ){
                                        h_new_pos = h_pos;
                                        dist_min  = d_heap.nd[h_pos] -> dist;
                                }
                        }
                        if ( dist_k <= dist_min )
                        break;
                        PON_A_POS_EN_LA_HEAP ( d_heap.nd[h_new_pos], h_actual_pos )
                        h_actual_pos = h_new_pos;
                }
                PON_A_POS_EN_LA_HEAP ( nodo_k, h_actual_pos )
        }
        return nodo_0;
}
```

Archivo Funcion_stack.h

Este archivo incluye las funciones que se realizan en la segunda etapa (etapa de reescaneo) del algoritmo.

```
/* ----- definiciones para las stacks ----- */

typedef struct str_stack {
  long        top;
  nodo        **arr;
}stack;

#define INICIALIZA_STACK( stack ){\
        stack.arr = (nodo**) calloc ( n  sizeof (nodo*) );\
        stack.top = -1;\
        if ( stack.arr == (nodo**) NULL )\
        exit ( 1 );\
}
#define CON_ELEMENTOS_STACK( stack )    ( stack.top >= 0 )
#define POP( stack, nodo ){\
        nodo = stack.arr [ stack.top ];\
        stack.top -- ;\
}
#define TRAE( stack, nodo ){\
        stack.top ++ ;\
        stack.arr [ stack.top ] = nodo;\
}\
```

Archivo ejecutable Dijkstra_modificado.c

El archivo ejecutable se llama dijkstra_modificado.c, en este archivo se manda llamar al archivo donde se encuentra el algoritmo y varias de sus funciones. Este archivo nos imprime los resultados finales.

```
                                        /*Algoritmo de Dijkstra modificado*/
#include <stdio.h>
#include <stdlib.h>
#include <assert.h>
#include "def_globales.h"

/* Variables estáticas */
long n_scans;
long n_pass;

/* types_### - estructura de nodos y arcos */
/* queue.h   - operaciones con queues  *
/* stack.h   - operaciones con stacks */

#define dikmod 1
#define RUN 1
/* #define XRUN 1*/
#define PRINT_CYCLE 1
/*#define DBG 1*/

#ifdef dikmod
#include "deftypes.h"
#include "funcion_heap_2.h"
#include "funcion_stack.h"
#include "dijkstra_modificado_sp.c"
#endif

/* Función para leer los datos de entrada en un formato específico */
#include "lectura.c"

/* Función 'timer()' para medir el tiempo de ejecución */
#include "timer.c"

void main (argc,argv)
int argc;
char *argv[];

{
FILE    *entrada;                   /*archivo de entrada de la red, cuando haya*/
double  t;                          /* tiempo de ejecución */

arco    *primer_arco,               /* primer arco */
        *ultimo_arco,               /* el último arco para escanear */
        *ta;                        /* arco actual */

nodo    *primer_nodo,               /* primer nodo */
        *ultimo_nodo,               /* después del último nodo */
        *source,                    /* raíz del árbol */
        *k,                         /* nodo actual */
        *i,                         /* nodo actual */
        *point;                     /* NULL si no se detectan ciclos negativos o un nodo en un ciclo negativo */

long    n,                          /* número de nodos */
        m,                          /* número de arcos */
        nmin,                       /* número mínimo de nodos */
        l_ciclo = 0;                /* número de arcos en un ciclo negativo */

double  sum_d = 0,                  /* suma de las longitudes para la solución */
        lon_c = 0,                  /* longitud del ciclo negativo */
        l_min;                      /* la longitud mínima de los arcos actual */
```

```
char   titulo[40];                    /* titulo del programa de ejecución */
nodo  *dummyNodo;                     /* para uso diverso */

#ifdef RUN
printf("SPC version 1 1\n");
printf ( "\nEmpieza la lectura de datos de entrada...\n" );
#endif

dummyNodo = (nodo *) calloc(1, sizeof(nodo));
if (argc==1)
        parse( &n, &m, &primer_nodo, &primer_arco, &source, &nmin );
        else{
                entrada=fopen(argv[1],"rt");
                fparse( &n, &m, &primer_nodo, &primer_arco, &source, &nmin ,entrada);
                fclose(entrada);
        }

#ifdef RUN
printf ( "Se leyeron datos de entrada  Empieza el calculo del problema ...\n" );
#endif

t = timer();
point = spc ( n, primer_nodo, source );
t = timer() -t;
/*printf("pasos %15ld \n",n_pass); */
if ( point == NNULL ){
        PARA_TODO_NODO ( k ){
        if ( k -> padre != NNULL )
        sum_d += (double) (k -> dist);
        }
}else{
        k = point;
        do {
                l_ciclo ++ ;
                i = k -> padre;
                l_min = INFINITO;
                PARA_TODO_ARCO_DESDE_EL_NODO ( i, ta ){
                        if ( ( ta -> head ) == k && ( ta -> lon ) < l_min )
                                l_min = ta -> lon;
                }
                lon_c += l_min;
                k = i;
        } while ( k != point );
        sum_d = lon_c;
}
#ifdef dikmod
strcpy ( titulo, "Dijkstra modificado" );
#endif

#ifdef XRUN
printf ("%8ld %8ld %15.0f %8ld %9ld %13.4f\n",
        n, m, sum_d, l_ciclo, n_scans, t );
#endif

#ifdef RUN
printf ("\n%s\n\
Nodos: %9ld\n\
Arcos: %9d\n\n\
Numero de escaneos:    %15ld\n\
Suma de las distancias: %15.0f\n\n\
Tiempo de ejecucion:   %17.4f\n", titulo, n, m, n_scans, sum_d, t );

if ( point != NNULL ){
        printf ("Arcos que conforman el ciclo negativo:   %10ld\n", l_ciclo );
        printf ("Longitud del ciclo negativo: %10.0f\n\n", lon_c );

        #ifdef PRINT_CYCLE
        printf ("Ciclo Negativo:\n");
        k = point;
```

```
        do {
                printf ("%7ld\n", NUMERO_NODO ( k ) );
                k = k -> padre;
        } while ( k != point );
        #endif
}else
        printf ("No se detectaron ciclos negativos\n\n");
#endif

#ifdef DBG
if ((n<=20)&&(m<=30)) {
        printf ( "Red inicial:\nn= %ld, m= %ld, nmin= %ld, source = %ld\n",n, m, nmin, NUMERO_NODO(source) );
        printf ("\nArcos ordenados:\n");
        PARA_TODO_NODO ( k ){
                PARA_TODO_ARCO_DESDE_EL_NODO ( k, ta )
                printf ( " %ld %ld %ld\n" NUMERO_NODO( k ), NUMERO_NODO( ta -> head )  ta -> lon);
        }
        printf ( "\nArbol:\n" );
        PARA_TODO_NODO ( k )
        printf ("%ld %ld %ld\n",
        NUMERO_NODO ( k ), NUMERO_NODO ( k -> padre ), k -> dist );
}
#endif

}
```

Archivo Dikjstra_modificado_sp.c

Este archivo realiza el procedimiento del algoritmo

```
/* Algoritmo de Dijkstra modificado con detección de ciclos  */

nodo *spc ( n, primer_nodo  source )

long   n;                   /* Número del node_first */
nodo   *primer_nodo,        /* apuntador al primer nodo */
       *source;             /* apuntador al nodo fuente */

{
int    cuenta;              /*contador para etiquetar a los nodos en su parte temp*/
nodo   *i,                  /* Nodo que se escanea */
       *j,                  /* Nodo en la cabeza del arco */
       *k,                  /* Nodo en ciclo negativo     */
       *ultimo_nodo,        /* ficticio: nodo después del último en la red */
       *nodo_last,
       *nodo_desde,
       *nodo_a,
       *nodos;

arco   *arco_ij,            /* arco analizandose */
       *ultimo_arco,        /* el siguiente arco despues del último que sale del nodo analizandose */
       *arco_last;

long   dist_new,            /* distancia del node_to vía node_from */
       dist_i,              /* distancia del node_from */
       r_cost,              /* costo reducido del arco */
       dist_from;

stack  new_pass,            /* stack para el new pass */
       top_sort,            /* stack para el orden topológico */
       pass;                /* stack para el paso Bellmann-Ford  */

int topol;

/* estatus del nodo en la lista ordenada stacks */
#define FUERA_DE_STACKS  0
#define EN_NEW_PASS    1
#define EN_TOP_SORT    2
```

```
#define EN_PASS      3

long   num_scans = 0,num_pass=0;

/* Inicialización */
nodos=primer_nodo;
nodo_last = nodos + n ;
cuenta=1;
PARA_TODO_NODO ( i ){
   i -> padre          = NNULL;
   i -> dist           = INFINITO;
   i -> status         = FUERA_DE_STACKS;
   i -> heap_pos       = NILL;
   i -> B_pos          = NILL;
   i -> temp           = cuenta++;
}

source -> padre     = source;
source -> dist      = 0;

/* Inicialización de la pila y la lista ordenada */
Init_heap (  n, source );
INICIALIZA_STACK ( new_pass )
TRAE ( new_pass, source )
source->status = EN_NEW_PASS;
INICIALIZA_STACK ( top_sort )
INICIALIZA_STACK ( pass )

        /* Ciclo principal */
        while ( CON_ELEMENTOS_EN_HEAP ){
        topol=0;
        /*Ciclo de Dijkstra*/
                while ( CON_ELEMENTOS_EN_HEAP ){
                        nodo_desde = Extrae_min (  );
                        arco_last = ( nodo_desde + 1 ) -> primero;
                        dist_from = nodo_desde -> dist;
                        num_scans ++;
                        for ( arco_ij = nodo_desde -> primero; arco_ij != arco_last; arco_ij ++ ){
                                nodo_a  = arco_ij -> head;
                                dist_new = dist_from  + ( arco_ij -> lon );
                                if ( dist_new <  nodo_a -> dist ){
                                        nodo_a -> dist  = dist_new;
                                        nodo_a -> padre = nodo_desde;
                                        if (NODO_FIJO (nodo_a)){
                                                nodo_a -> heap_pos = TRATO_MEJORAR;
                                                TRAE(new_pass,nodo_a)
                                                nodo_a -> status = EN_NEW_PASS;
                                                 topol++;
                                        }else{
                                                if ( SIN_SUBIR (nodo_a) ){
                                                        Inserta_a_heap ( nodo_a );
                                                }else{
                                                        Heap_decrease_key (  nodo_a, dist_new );
                                                }
                                        }
                                }
                        }
                }
        /* Ordenamiento de la lista de reescaneo */
        cuenta=0;
        if (topol){
                while( CON_ELEMENTOS_STACK(new_pass)){
                        cuenta=1;
                        POP(new_pass,i)
                        /* Búsqueda de ordenamiento */
                        if (i->status == EN_NEW_PASS){
                                i->status       = EN_TOP_SORT;
                                i -> actual     = i -> primero;   /* de arriba*/
                                while ( 1 ){
                                        PARA_TODO_ARCO_DESDE_EL_NODO_ORDENA ( i, arco_ij, i -> actual){
```

```
                                        j = arco_ij -> head;
                                        if (j->padre == i){
                                                if ( j -> status < EN_TOP_SORT ){
                                                        /* Siguiente nodo en el ordenamiento */
                                                        i -> actual = arco_ij + 1;
                                                        TRAE ( top_sort, i )
                                                        j -> status   = EN_TOP_SORT;
                                                        j -> actual   = j -> primero;
                                                        j -> negs     = i -> negs;
                                                        i = j;
                                                break;
                                                }
                                                if ( j -> status == EN_TOP_SORT ){
                                                        /* Se detecto ciclo negativo o de cero */
                                                        if ( NODO_TRATO_MEJORAR(j) ){
                                                                /* Se detecto un ciclo negativo */
                                                                j -> padre = i;
                                                                /* Marcando el ciclo negativo */
                                                                while ( i != j ){
                                                                        POP ( top_sort, k )
                                                                        i -> padre = k;
                                                                        i = k;
                                                                }
                                                                n_scans = num_scans;
                                                                return i;
                                                        }
                                                }
                                        }
                                }
                                if ( arco_ij == ultimo_arco ){
                                        i -> status    = EN_PASS;
                                        i -> heap_pos= NILL;
                                        TRAE ( pass, i )
                                        if ( CON_ELEMENTOS_STACK ( top_sort ) )
                                        POP ( top_sort, i )
                                else
                                        break;
                                }
                        } /* Fin del ordenamiento */
                }       /* Fin de la búsqueda de ordenamiento */
        }               /* Fin del while */
}                       /* Fin del orden topológico */
/* Paso de Bellman - Ford */
while ( CON_ELEMENTOS_STACK ( pass ) ){
        num_scans ++;
        POP ( pass  i )
        i -> status = FUERA_DE_STACKS;
        dist_i  = i -> dist;
        PARA_TODO_ARCO_DESDE_EL_NODO ( i  arco_ij ){ /* escaneando arcos que salen de i */
                j           = arco_ij -> head;
                dist_new = dist_i + ( arco_ij -> lon );
                if ( dist_new <  j -> dist ){
                        j -> dist   = dist_new;
                        j -> padre = i;
                        if ( j -> status == FUERA_DE_STACKS  ){
                                Inserta_a_heap(j);
                                TRAE (new_pass,j)
                                j -> status = EN_NEW_PASS;
                        }
                }
        } /* Fin del escaneo de i */
}       /* Fin de un paso */
}
num_pass+=topol;
}               /* Fin del ciclo principal */
n_scans = num_scans;
n_pass=num_pass;
return NNULL;
}
```

Archivo timer.c

Este archivo calcula el tiempo de ejecución del algoritmo para calcular el problema de ruta más corta en una red cualquiera.

```
                                        /* La función tiempo */
/*#include <sys/time.h>
#include <sys/resource.h>
#include <unistd.h>
float timer ()*/
 /* Modificacion para que funcione en Borland c++  5.0 */
#include <sys/timeb.h>
double timer ()
{
      /*struct rusage r;*/ /*modificacion para Borland c++ 5.0*/
      struct timeb r;

      /*getrusage(0, &r);*//*cambia tambien la función*/
      ftime(&r);
      /*return (float)(r.ru_utime.tv_sec+r.ru_utime.tv_usec/(float)1000000);*/
      return (double)r.time + (double)r.millitm/1000.0;
}
```

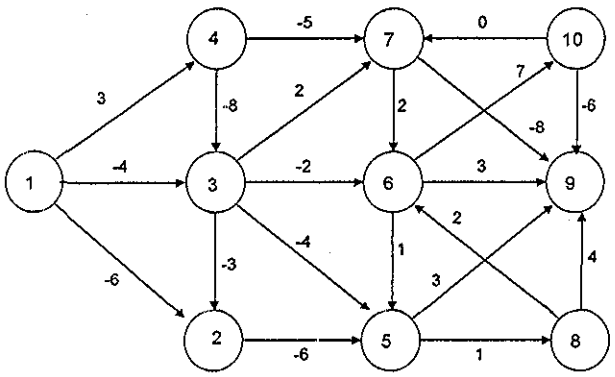
APÉNDICE B

En este apéndice se muestran los archivos de comandos con los que se corrieron los experimentos y el formato en que deben proporcionarse los datos de las redes a los programas resolvedores. Los generadores de redes proporcionan sus resultados en este formato.

B.1 Formato de datos de las redes.

El formato utilizado es el que fijó DIMACS en el primer concurso de implementación de algoritmos (Johnson y McGeoch, 1993).

A continuación se muestra un ejemplo de una red pequeña y el formato que debe tener para poder ser leído por l os programas resolvedores.



Los comentarios que se encuentran dentro de paréntesis no deben incluirse en el archivo de datos.

Archivo de datos:

```
c ejemplo tesis                (c = línea comentada)
c
t red_con_arcos_negativos      (t = línea para el titulo del problema)
c
```

```
p sp  10  20   (p = línea donde se especifica el tipo de problema, sp = shortest path, no. de nodos=10 y no. de arcos=20)
c
n  1           (n = línea para poner el nodo fuente, en este caso es el nodo 1)
c
a    1   2   -6        (a = línea para especificar los arcos. El primero dato es de donde sale el arco, el
a    1   3   -4        segundo a donde llega y por último la longitud del arco)
a    1   4    3
a    2   5   -6
a    3   5   -4
a    3   2   -3
a    3   6   -2
a    3   7    2
a    4   3   -8
a    4   7   -5
a    5   9    3
a    5   8    1
a    6   5    1
a    6   9    3
a    6   10   7
a    7   6    2
a    8   9   -8
a    8   9    4
a    10  9   -6
a    10  7    0
```

B.2 Archivos de comandos

La estructura de los directorios en la P.C. es la siguiente.
El directorio a partir de donde se encuentran todos los programas es:

c:\shortest

El directorio en donde se encuentran los programas generadores de redes es:

c:\shortest\generadores\bin

Los programas generadores usados son: spgrid.exe (genera la red tipo Grid-NHard), sprand.exe (genera la red tipo Rand-P) y spacyc.exe (genera las redes tipo Acyc-Neg y Acyc-P2N)

El programa resolvedor del algoritmo *Dijkstra modificado* es:

c:\shortest\dikmod\bin\dikmod.exe

El programa resolvedor del algoritmo *Simplex* es:

c:\shortest\simplex\bin\simplex.exe

El programa resolvedor del algoritmo *Tarjan* es:

c:\shortest\tarjan\bin\tarjan.exe

El programa resolvedor del algoritmo *Goldberg-Radzik* es:

c:\shortest\goldberg\bin\goldberg.exe

El programa resolvedor del algoritmo *Variante Dijkstra* es:
c:\shortest\dikh\bin\dikh.exe

El directorio en donde se encuentran los archivos de comandos para los experimentos está en el directorio particular de cada algoritmo, en el subdirectorio scripts:
c:\shortest*nombre del algoritmo*\scripts

Cada línea en los archivos de comandos tiene la siguiente lógica de ejecución:
(Generador) (Parámetros del geñerador) | (Resolvedor) >> (Archivo de resultados)
donde | es el caracter de concatenación de ms-dos y >> la orden de salida acumulada a un archivo. La documentación de los parámetros necesarios para los generadores se encuentra en el conjunto de programas originales incluido en el D.C. anexo.

A continuación se incluyen los archivos de comandos utilizados.

tesis_dijkstra_modificado.bat

```
.\ .\generadores\bin\spgrid 256 32 111 -cc -cl1 -cm1 -ax64 -im-10000 -il-1000 -ix5 -ih5 -in0 | .\bin\dijkstra_modificado > .\results\grid_nhard_dijkstra_modificado.res
.\ .\generadores\bin\spgrid 256 32 113 -cc -cl1 -cm1 -ax64 -im-10000 -il-1000 -ix5 -ih5 -in0 | .\bin\dijkstra_modificado >> .\results\grid_nhard_dijkstra_modificado.res
.\ .\generadores\bin\spgrid 256 32 115 -cc -cl1 -cm1 -ax64 -im-10000 -il-1000 -ix5 -ih5 -in0 | .\bin\dijkstra_modificado >> .\results\grid_nhard_dijkstra_modificado.res
.\ .\generadores\bin\spgrid 256 32 117 -cc -cl1 -cm1 -ax64 -im-10000 -il-1000 -ix5 -ih5 -in0 | .\bin\dijkstra_modificado >> .\results\grid_nhard_dijkstra_modificado.res
.\ .\generadores\bin\spgrid 256 32 119 -cc -cl1 -cm1 -ax64 -im-10000 -il-1000 -ix5 -ih5 -in0 | .\bin\dijkstra_modificado >> .\results\grid_nhard_dijkstra_modificado.res

.\ .\generadores\bin\spgrid 512 32 211 -cc -cl1 -cm1 -ax64 -im-10000 -il-1000 -ix5 -ih5 -in0 | .\bin\dijkstra_modificado >> .\results\grid_nhard_dijkstra_modificado.res
.\ .\generadores\bin\spgrid 512 32 213 -cc -cl1 -cm1 -ax64 -im-10000 -il-1000 -ix5 -ih5 -in0 | .\bin\dijkstra_modificado >> .\results\grid_nhard_dijkstra_modificado.res
.\ .\generadores\bin\spgrid 512 32 215 -cc -cl1 -cm1 -ax64 -im-10000 -il-1000 -ix5 -ih5 -in0 | .\bin\dijkstra_modificado >> .\results\grid_nhard_dijkstra_modificado.res
.\ .\generadores\bin\spgrid 512 32 217 -cc -cl1 -cm1 -ax64 -im-10000 -il-1000 -ix5 -ih5 -in0 | .\bin\dijkstra_modificado >> .\results\grid_nhard_dijkstra_modificado.res
.\ .\generadores\bin\spgrid 512 32 219 -cc -cl1 -cm1 -ax64 -im-10000 -il-1000 -ix5 -ih5 -in0 | .\bin\dijkstra_modificado >> .\results\grid_nhard_dijkstra_modificado.res

.\ .\generadores\bin\spgrid 1024 32 311 -cc -cl1 -cm1 -ax64 -im-10000 -il-1000 -ix5 -ih5 -in0 | .\bin\dijkstra_modificado >> .\results\grid_nhard_dijkstra_modificado.res
.\ .\generadores\bin\spgrid 1024 32 313 -cc -cl1 -cm1 -ax64 -im-10000 -il-1000 -ix5 -ih5 -in0 | .\bin\dijkstra_modificado >> .\results\grid_nhard_dijkstra_modificado.res
.\ .\generadores\bin\spgrid 1024 32 315 -cc -cl1 -cm1 -ax64 -im-10000 -il-1000 -ix5 -ih5 -in0 | .\bin\dijkstra_modificado >> .\results\grid_nhard_dijkstra_modificado.res
.\ .\generadores\bin\spgrid 1024 32 317 -cc -cl1 -cm1 -ax64 -im-10000 -il-1000 -ix5 -ih5 -in0 | .\bin\dijkstra_modificado >> .\results\grid_nhard_dijkstra_modificado.res
.\ .\generadores\bin\spgrid 1024 32 319 -cc -cl1 -cm1 -ax64 -im-10000 -il-1000 -ix5 -ih5 -in0 | .\bin\dijkstra_modificado >> .\results\grid_nhard_dijkstra_modificado.res

.\ .\generadores\bin\spgrid 2048 32 411 -cc -cl1 -cm1 -ax64 -im-10000 -il-1000 -ix5 -ih5 -in0 | .\bin\dijkstra_modificado >> .\results\grid_nhard_dijkstra_modificado.res
.\ .\generadores\bin\spgrid 2048 32 413 -cc -cl1 -cm1 -ax64 -im-10000 -il-1000 -ix5 -ih5 -in0 | .\bin\dijkstra_modificado >> .\results\grid_nhard_dijkstra_modificado.res
.\ .\generadores\bin\spgrid 2048 32 415 -cc -cl1 -cm1 -ax64 -im-10000 -il-1000 -ix5 -ih5 -in0 | .\bin\dijkstra_modificado >> .\results\grid_nhard_dijkstra_modificado.res
.\ .\generadores\bin\spgrid 2048 32 417 -cc -cl1 -cm1 -ax64 -im-10000 -il-1000 -ix5 -ih5 -in0 | .\bin\dijkstra_modificado >> .\results\grid_nhard_dijkstra_modificado.res
.\ .\generadores\bin\spgrid 2048 32 419 -cc -cl1 -cm1 -ax64 -im-10000 -il-1000 -ix5 -ih5 -in0 | .\bin\dijkstra_modificado >> .\results\grid_nhard_dijkstra_modificado.res
```

```
\ \generadores\bin\spgrid 4096 32 511 -cc -cl1 -cm1 -ax64 -im-10000 -il-1000 -ix5 -ih5 -in0 | \bin\dijkstra_modificado >>
\results\grid_nhard_dijkstra_modificado.res
\ \generadores\bin\spgrid 4096 32 513 -cc -cl1 -cm1 -ax64 -im-10000 -il-1000 -ix5 -ih5 -in0 | \bin\dijkstra_modificado >>
\results\grid_nhard_dijkstra_modificado.res
\ \generadores\bin\spgrid 4096 32 515 -cc -cl1 -cm1 -ax64 -im-10000 -il-1000 -ix5 -ih5 -in0 | \bin\dijkstra_modificado >>
\results\grid_nhard_dijkstra_modificado.res
\ \generadores\bin\spgrid 4096 32 517 -cc -cl1 -cm1 -ax64 -im-10000 -il-1000 -ix5 -ih5 -in0 | \bin\dijkstra_modificado >>
\results\grid_nhard_dijkstra_modificado.res
\ \generadores\bin\spgrid 4096 32 519 -cc -cl1 -cm1 -ax64 -im-10000 -il-1000 -ix5 -ih5 -in0 | \bin\dijkstra_modificado >>
\results\grid_nhard_dijkstra_modificado res

\ \generadores\bin\spgrid 8192 32 611 -cc -cl1 -cm1 -ax64 -im-10000 -il-1000 -ix5 -ih5 -in0 | \bin\dijkstra_modificado >>
\results\grid_nhard_dijkstra_modificado.res
\ \generadores\bin\spgrid 8192 32 613 -cc -cl1 -cm1 -ax64 -im-10000 -il-1000 -ix5 -ih5 -in0 | \bin\dijkstra_modificado >>
\results\grid_nhard_dijkstra_modificado.res
\ \generadores\bin\spgrid 8192 32 615 -cc -cl1 -cm1 -ax64 -im-10000 -il-1000 -ix5 -ih5 -in0 | \bin\dijkstra_modificado >>
\results\grid_nhard_dijkstra_modificado.res
\ \generadores\bin\spgrid 8192 32 617 -cc -cl1 -cm1 -ax64 -im-10000 -il-1000 -ix5 -ih5 -in0 | \bin\dijkstra_modificado >>
\results\grid_nhard_dijkstra_modificado.res
\ \generadores\bin\spgrid 8192 32 619 -cc -cl1 -cm1 -ax64 -im-10000 -il-1000 -ix5 -ih5 -in0 | \bin\dijkstra_modificado >>
\results\grid_nhard_dijkstra_modificado.res

\ \generadores\bin\sprand 131072 524288 961 -cl1 | \bin\dijkstra_modificado >. \results\rand_p_dijkstra_modificado.res
\ \generadores\bin\sprand 131072 524288 963 -cl1 | \bin\dijkstra_modificado >> \results\rand_p_dijkstra_modificado.res
\ \generadores\bin\sprand 131072 524288 965 -cl1 | \bin\dijkstra_modificado >> \results\rand_p_dijkstra_modificado.res
\ \generadores\bin\sprand 131072 524288 967 -cl1 | \bin\dijkstra_modificado >> \results\rand_p_dijkstra_modificado.res
\ \generadores\bin\sprand 131072 524288 969 -cl1 | \bin\dijkstra_modificado >> \results\rand_p_dijkstra_modificado.res

\ \generadores\bin\sprand 131072 524288 961 -pl1000 -cl1 | \bin\dijkstra_modificado >> \results\rand_p_dijkstra_modificado.res
\ \generadores\bin\sprand 131072 524288 963 -pl1000 -cl1 | \bin\dijkstra_modificado >> \results\rand_p_dijkstra_modificado.res
\ \generadores\bin\sprand 131072 524288 965 -pl1000 -cl1 | \bin\dijkstra_modificado >> \results\rand_p_dijkstra_modificado.res
\ \generadores\bin\sprand 131072 524288 967 -pl1000 -cl1 | \bin\dijkstra_modificado >> \results\rand_p_dijkstra_modificado.res
\ \generadores\bin\sprand 131072 524288 969 -pl1000 -cl1 | \bin\dijkstra_modificado >> \results\rand_p_dijkstra_modificado.res

\ \generadores\bin\sprand 131072 524288 961 -pl5000 -cl1 | \bin\dijkstra_modificado >> \results\rand_p_dijkstra_modificado.res
\ \generadores\bin\sprand 131072 524288 963 -pl5000 -cl1 | \bin\dijkstra_modificado >> \results\rand_p_dijkstra_modificado.res
\ \generadores\bin\sprand 131072 524288 965 -pl5000 -cl1 | \bin\dijkstra_modificado >> \results\rand_p_dijkstra_modificado.res
\ \generadores\bin\sprand 131072 524288 967 -pl5000 -cl1 | \bin\dijkstra_modificado >> \results\rand_p_dijkstra_modificado.res
\ \generadores\bin\sprand 131072 524288 969 -pl5000 -cl1 | \bin\dijkstra_modificado >> \results\rand_p_dijkstra_modificado.res

\ \generadores\bin\sprand 131072 524288 961 -pl10000 -cl1 | \bin\dijkstra_modificado >> \results\rand_p_dijkstra_modificado res
\ \generadores\bin\sprand 131072 524288 963 -pl10000 -cl1 | \bin\dijkstra_modificado >> \results\rand_p_dijkstra_modificado.res
\ \generadores\bin\sprand 131072 524288 965 -pl10000 -cl1 | \bin\dijkstra_modificado >> \results\rand_p_dijkstra_modificado res
\ \generadores\bin\sprand 131072 524288 967 -pl10000 -cl1 | \bin\dijkstra_modificado >> \results\rand_p_dijkstra_modificado res
\ \generadores\bin\sprand 131072 524288 969 -pl10000 -cl1 | \bin\dijkstra_modificado >> \results\rand_p_dijkstra_modificado res

\ \generadores\bin\sprand 131072 524288 961 -pl100000 -cl1 | \bin\dijkstra_modificado >> \results\rand_p_dijkstra_modificado res
\ \generadores\bin\sprand 131072 524288 963 -pl100000 -cl1 | \bin\dijkstra_modificado >> \results\rand_p_dijkstra_modificado.res
\ \generadores\bin\sprand 131072 524288 965 -pl100000 -cl1 | \bin\dijkstra_modificado >> \results\rand_p_dijkstra_modificado res
\ \generadores\bin\sprand 131072 524288 967 -pl100000 -cl1 | \bin\dijkstra_modificado >> \results\rand_p_dijkstra_modificado.res
\ \generadores\bin\sprand 131072 524288 969 -pl100000 -cl1 | \bin\dijkstra_modificado >> \results\rand_p_dijkstra_modificado.res

\ \generadores\bin\sprand 131072 524288 961 -pl1000000 -cl1 | \bin\dijkstra_modificado >> \results\rand_p_dijkstra_modificado.res
\ \generadores\bin\sprand 131072 524288 963 -pl1000000 -cl1 | \bin\dijkstra_modificado >> \results\rand_p_dijkstra_modificado.res
\ \generadores\bin\sprand 131072 524288 965 -pl1000000 -cl1 | \bin\dijkstra_modificado >> \results\rand_p_dijkstra_modificado.res
\ \generadores\bin\sprand 131072 524288 967 -pl1000000 -cl1 | \bin\dijkstra_modificado >> \results\rand_p_dijkstra_modificado.res
\ \generadores\bin\sprand 131072 524288 969 -pl1000000 -cl1 | \bin\dijkstra_modificado >> \results\rand_p_dijkstra_modificado.res

\ \generadores\bin\sprand 131072 524288 961 -pl5000000 -cl1 | \bin\dijkstra_modificado >> \results\rand_p_dijkstra_modificado.res
\ \generadores\bin\sprand 131072 524288 963 -pl5000000 -cl1 | \bin\dijkstra_modificado >> \results\rand_p_dijkstra_modificado.res
\ \generadores\bin\sprand 131072 524288 965 -pl5000000 -cl1 | \bin\dijkstra_modificado >> \results\rand_p_dijkstra_modificado.res
\ \generadores\bin\sprand 131072 524288 967 -pl5000000 -cl1 | \bin\dijkstra_modificado >> \results\rand_p_dijkstra_modificado.res
\ \generadores\bin\sprand 131072 524288 969 -pl5000000 -cl1 | \bin\dijkstra_modificado >> \results\rand_p_dijkstra_modificado.res

\ \generadores\bin\spacyc 8192 131072 561 -lm-10000 -ll0 -cl-1 | \bin\dijkstra_modificado > \results\acyc_neg_dijkstra_modificado.res
\ \generadores\bin\spacyc 8192 131072 563 -lm-10000 -ll0 -cl-1 | \bin\dijkstra_modificado >> \results\acyc_neg_dijkstra_modificado.res
\ \generadores\bin\spacyc 8192 131072 565 -lm-10000 -ll0 -cl-1 | \bin\dijkstra_modificado >> \results\acyc_neg_dijkstra_modificado.res
\ \generadores\bin\spacyc 8192 131072 567 -lm-10000 -ll0 -cl-1 | \bin\dijkstra_modificado >> \results\acyc_neg_dijkstra_modificado.res
\ \generadores\bin\spacyc 8192 131072 569 -lm-10000 -ll0 -cl-1 | \bin\dijkstra_modificado >> \results\acyc_neg_dijkstra_modificado.res

\ \generadores\bin\spacyc 16384 262144 661 -lm-10000 -ll0 -cl-1 | \bin\dijkstra_modificado >> \results\acyc_neg_dijkstra_modificado.res
```

```
\ \generadores\bin\spacyc 16384 262144 663 -lm-10000 -ll0 -cl-1 | \bin\dijkstra_modificado >> \results\acyc_neg_dijkstra_modificado res
\ \generadores\bin\spacyc 16384 262144 665 -lm-10000 -ll0 -cl-1 | \bin\dijkstra_modificado >> \results\acyc_neg_dijkstra_modificado res
\ \generadores\bin\spacyc 16384 262144 667 -lm-10000 -ll0 -cl-1 | \bin\dijkstra_modificado >> \results\acyc_neg_dijkstra_modificado res
\ \generadores\bin\spacyc 16384 262144 669 -lm-10000 -ll0 -cl-1 | \bin\dijkstra_modificado >> \results\acyc_neg_dijkstra_modificado res

\ \generadores\bin\spacyc 32768 524288 761 -lm-10000 -ll0 -cl-1 | \bin\dijkstra_modificado >> \results\acyc_neg_dijkstra_modificado res
\ \generadores\bin\spacyc 32768 524288 763 -lm-10000 -ll0 -cl-1 | \bin\dijkstra_modificado >> \results\acyc_neg_dijkstra_modificado res
\ \generadores\bin\spacyc 32768 524288 765 -lm-10000 -ll0 -cl-1 | \bin\dijkstra_modificado >> \results\acyc_neg_dijkstra_modificado res
\ \generadores\bin\spacyc 32768 524288 767 -lm-10000 -ll0 -cl-1 | \bin\dijkstra_modificado >> \results\acyc_neg_dijkstra_modificado res
\ \generadores\bin\spacyc 32768 524288 769 -lm-10000 -ll0 -cl-1 | \bin\dijkstra_modificado >> \results\acyc_neg_dijkstra_modificado res

\ \generadores\bin\spacyc 65536 1048576 861 -lm-10000 -ll0 -cl-1 | \bin\dijkstra_modificado >> \results\acyc_neg_dijkstra_modificado res
\ \generadores\bin\spacyc 65536 1048576 863 -lm-10000 -ll0 -cl-1 | \bin\dijkstra_modificado >> \results\acyc_neg_dijkstra_modificado res
\ \generadores\bin\spacyc 65536 1048576 865 -lm-10000 -ll0 -cl-1 | \bin\dijkstra_modificado >> \results\acyc_neg_dijkstra_modificado res
\ \generadores\bin\spacyc 65536 1048576 867 -lm-10000 -ll0 -cl-1 | \bin\dijkstra_modificado >> \results\acyc_neg_dijkstra_modificado res
\ \generadores\bin\spacyc 65536 1048576 869 -lm-10000 -ll0 -cl-1 | \bin\dijkstra_modificado >> \results\acyc_neg_dijkstra_modificado res

\ \generadores\bin\spacyc 131072 2097152 961 -lm-10000 -ll0 -cl-1 | \bin\dijkstra_modificado >> \results\acyc_neg_dijkstra_modificado res
\ \generadores\bin\spacyc 131072 2097152 963 -lm-10000 -ll0 -cl-1 | \bin\dijkstra_modificado >> \results\acyc_neg_dijkstra_modificado res
\ \generadores\bin\spacyc 131072 2097152 965 -lm-10000 -ll0 -cl-1 | \bin\dijkstra_modificado >> \results\acyc_neg_dijkstra_modificado res
\ \generadores\bin\spacyc 131072 2097152 967 -lm-10000 -ll0 -cl-1 | \bin\dijkstra_modificado >> \results\acyc_neg_dijkstra_modificado res
\ \generadores\bin\spacyc 131072 2097152 969 -lm-10000 -ll0 -cl-1 | \bin\dijkstra_modificado >> \results\acyc_neg_dijkstra_modificado res

\ \generadores\bin\spacyc 16384 262144 461 -lm0 -ll10000 | \bin\dijkstra_modificado > \results\acyc_p2n_dijkstra_modificado res
\ \generadores\bin\spacyc 16384 262144 463 -lm0 -ll10000 | \bin\dijkstra_modificado >> \results\acyc_p2n_dijkstra_modificado res
\ \generadores\bin\spacyc 16384 262144 465 -lm0 -ll10000 | \bin\dijkstra_modificado >> \results\acyc_p2n_dijkstra_modificado res
\ \generadores\bin\spacyc 16384 262144 467 -lm0 -ll10000 | \bin\dijkstra_modificado >> \results\acyc_p2n_dijkstra_modificado res
\ \generadores\bin\spacyc 16384 262144 469 -lm0 -ll10000 | \bin\dijkstra_modificado >> \results\acyc_p2n_dijkstra_modificado res

\ \generadores\bin\spacyc 16384 262144 761 -lm-1000 -ll9000 | \bin\dijkstra_modificado >> \results\acyc_p2n_dijkstra_modificado res
\ \generadores\bin\spacyc 16384 262144 763 -lm-1000 -ll9000 | \bin\dijkstra_modificado >> \results\acyc_p2n_dijkstra_modificado res
\ \generadores\bin\spacyc 16384 262144 765 -lm-1000 -ll9000 | \bin\dijkstra_modificado >> \results\acyc_p2n_dijkstra_modificado res
\ \generadores\bin\spacyc 16384 262144 767 -lm-1000 -ll9000 | \bin\dijkstra_modificado >> \results\acyc_p2n_dijkstra_modificado res
\ \generadores\bin\spacyc 16384 262144 769 -lm-1000 -ll9000 | \bin\dijkstra_modificado >> \results\acyc_p2n_dijkstra_modificado res

\ \generadores\bin\spacyc 16384 262144 861 -lm-2000 -ll8000 | \bin\dijkstra_modificado >> \results\acyc_p2n_dijkstra_modificado res
\ \generadores\bin\spacyc 16384 262144 863 -lm-2000 -ll8000 | \bin\dijkstra_modificado >> \results\acyc_p2n_dijkstra_modificado res
\ \generadores\bin\spacyc 16384 262144 865 -lm-2000 -ll8000 | \bin\dijkstra_modificado >> \results\acyc_p2n_dijkstra_modificado res
\ \generadores\bin\spacyc 16384 262144 867 -lm-2000 -ll8000 | \bin\dijkstra_modificado >> \results\acyc_p2n_dijkstra_modificado res
\ \generadores\bin\spacyc 16384 262144 869 -lm-2000 -ll8000 | \bin\dijkstra_modificado >> \results\acyc_p2n_dijkstra_modificado res

\ \generadores\bin\spacyc 16384 262144 961 -lm-3000 -ll7000 | \bin\dijkstra_modificado >> \results\acyc_p2n_dijkstra_modificado res
\ \generadores\bin\spacyc 16384 262144 963 -lm-3000 -ll7000 | \bin\dijkstra_modificado >> \results\acyc_p2n_dijkstra_modificado res
\ \generadores\bin\spacyc 16384 262144 965 -lm-3000 -ll7000 | \bin\dijkstra_modificado >> \results\acyc_p2n_dijkstra_modificado res
\ \generadores\bin\spacyc 16384 262144 967 -lm-3000 -ll7000 | \bin\dijkstra_modificado >> \results\acyc_p2n_dijkstra_modificado res
\ \generadores\bin\spacyc 16384 262144 969 -lm-3000 -ll7000 | \bin\dijkstra_modificado >> \results\acyc_p2n_dijkstra_modificado res

\ \generadores\bin\spacyc 16384 262144 261 -lm-4000 -ll6000 | \bin\dijkstra_modificado >> \results\acyc_p2n_dijkstra_modificado res
\ \generadores\bin\spacyc 16384 262144 263 -lm-4000 -ll6000 | \bin\dijkstra_modificado >> \results\acyc_p2n_dijkstra_modificado res
\ \generadores\bin\spacyc 16384 262144 265 -lm-4000 -ll6000 | \bin\dijkstra_modificado >> \results\acyc_p2n_dijkstra_modificado res
\ \generadores\bin\spacyc 16384 262144 267 -lm-4000 -ll6000 | \bin\dijkstra_modificado >> \results\acyc_p2n_dijkstra_modificado res
\ \generadores\bin\spacyc 16384 262144 269 -lm-4000 -ll6000 | \bin\dijkstra_modificado >> \results\acyc_p2n_dijkstra_modificado res

\ \generadores\bin\spacyc 16384 262144 461 -lm-5000 -ll5000 | \bin\dijkstra_modificado >> \results\acyc_p2n_dijkstra_modificado res
\ \generadores\bin\spacyc 16384 262144 463 -lm-5000 -ll5000 | \bin\dijkstra_modificado >> \results\acyc_p2n_dijkstra_modificado res
\ \generadores\bin\spacyc 16384 262144 465 -lm-5000 -ll5000 | \bin\dijkstra_modificado >> \results\acyc_p2n_dijkstra_modificado res
\ \generadores\bin\spacyc 16384 262144 467 -lm-5000 -ll5000 | \bin\dijkstra_modificado >> \results\acyc_p2n_dijkstra_modificado res
\ \generadores\bin\spacyc 16384 262144 469 -lm-5000 -ll5000 | \bin\dijkstra_modificado >> \results\acyc_p2n_dijkstra_modificado res

\ \generadores\bin\spacyc 16384 262144 661 -lm-6000 -ll4000 | \bin\dijkstra_modificado >> \results\acyc_p2n_dijkstra_modificado res
\ \generadores\bin\spacyc 16384 262144 663 -lm-6000 -ll4000 | \bin\dijkstra_modificado >> \results\acyc_p2n_dijkstra_modificado res
\ \generadores\bin\spacyc 16384 262144 665 -lm-6000 -ll4000 | \bin\dijkstra_modificado >> \results\acyc_p2n_dijkstra_modificado res
\ \generadores\bin\spacyc 16384 262144 667 -lm-6000 -ll4000 | \bin\dijkstra_modificado >> \results\acyc_p2n_dijkstra_modificado res
\ \generadores\bin\spacyc 16384 262144 669 -lm-6000 -ll4000 | \bin\dijkstra_modificado >> \results\acyc_p2n_dijkstra_modificado res

\ \generadores\bin\spacyc 16384 262144 761 -lm-10000 -ll0 | \bin\dijkstra_modificado >> \results\acyc_p2n_dijkstra_modificado res
\ \generadores\bin\spacyc 16384 262144 763 -lm-10000 -ll0 | \bin\dijkstra_modificado >> \results\acyc_p2n_dijkstra_modificado res
\ \generadores\bin\spacyc 16384 262144 765 -lm-10000 -ll0 | \bin\dijkstra_modificado >> \results\acyc_p2n_dijkstra_modificado res
\ \generadores\bin\spacyc 16384 262144 767 -lm-10000 -ll0 | \bin\dijkstra_modificado >> \results\acyc_p2n_dijkstra_modificado res
\ \generadores\bin\spacyc 16384 262144 769 -lm-10000 -ll0 | \bin\dijkstra_modificado >> \results\acyc_p2n_dijkstra_modificado res
```

tesis_simplex.bat

```
\ \generadores\bin\spgrid 256  32 111 -cc -cl1 -cm1 -ax64 -im-10000 -il-1000 -ix5 -ih5 -in0 | \bin\simplex > \results\grid_nhard_simplex.res
\ \generadores\bin\spgrid 256  32 113 -cc -cl1 -cm1 -ax64 -im-10000 -il-1000 -ix5 -ih5 -in0 | \bin\simplex >> \results\grid_nhard_simplex res
\ \generadores\bin\spgrid 256  32 115 -cc -cl1 -cm1 -ax64 -im-10000 -il-1000 -ix5 -ih5 -in0 | \bin\simplex >> \results\grid_nhard_simplex res
```

```
\ \generadores\bin\spgrid 256  32 117 -cc -cl1 -cm1 -ax64 -im-10000 -il-1000 -ix5 -ih5 -in0 | \bin\simplex >> \results\grid_nhard_simplex.res
\ \generadores\bin\spgrid 256  32 119 -cc -cl1 -cm1 -ax64 -im-10000 -il-1000 -ix5 -ih5 -in0 | \bin\simplex >> \results\grid_nhard_simplex.res

\ \generadores\bin\spgrid 512  32 211 -cc -cl1 -cm1 -ax64 -im-10000 -il-1000 -ix5 -ih5 -in0 | \bin\simplex >> \results\grid_nhard_simplex.res
\ \generadores\bin\spgrid 512  32 213 -cc -cl1 -cm1 -ax64 -im-10000 -il-1000 -ix5 -ih5 -in0 | \bin\simplex >> \results\grid_nhard_simplex.res
\ \generadores\bin\spgrid 512  32 215 -cc -cl1 -cm1 -ax64 -im-10000 -il-1000 -ix5 -ih5 -in0 | \bin\simplex >> \results\grid_nhard_simplex.res
\ \generadores\bin\spgrid 512  32 217 -cc -cl1 -cm1 -ax64 -im-10000 -il-1000 -ix5 -ih5 -in0 | \bin\simplex >> \results\grid_nhard_simplex.res
\ \generadores\bin\spgrid 512  32 219 -cc -cl1 -cm1 -ax64 -im-10000 -il-1000 -ix5 -ih5 -in0 | \bin\simplex >> \results\grid_nhard_simplex.res

\ \generadores\bin\spgrid 1024 32 311 -cc -cl1 -cm1 -ax64 -im-10000 -il-1000 -ix5 -ih5 -in0 | \bin\simplex >> \results\grid_nhard_simplex.res
\ \generadores\bin\spgrid 1024 32 313 -cc -cl1 -cm1 -ax64 -im-10000 -il-1000 -ix5 -ih5 -in0 | \bin\simplex >> \results\grid_nhard_simplex.res
\ \generadores\bin\spgrid 1024 32 315 -cc -cl1 -cm1 -ax64 -im-10000 -il-1000 -ix5 -ih5 -in0 | \bin\simplex >> \results\grid_nhard_simplex.res
\ \generadores\bin\spgrid 1024 32 317 -cc -cl1 -cm1 -ax64 -im-10000 -il-1000 -ix5 -ih5 -in0 | \bin\simplex >> \results\grid_nhard_simplex.res
\ \generadores\bin\spgrid 1024 32 319 -cc -cl1 -cm1 -ax64 -im-10000 -il-1000 -ix5 -ih5 -in0 | \bin\simplex >> \results\grid_nhard_simplex.res

\ \generadores\bin\spgrid 2048 32 411 -cc -cl1 -cm1 -ax64 -im-10000 -il-1000 -ix5 -ih5 -in0 | \bin\simplex >> \results\grid_nhard_simplex.res
\ \generadores\bin\spgrid 2048 32 413 -cc -cl1 -cm1 -ax64 -im-10000 -il-1000 -ix5 -ih5 -in0 | \bin\simplex >> \results\grid_nhard_simplex.res
\ \generadores\bin\spgrid 2048 32 415 -cc -cl1 -cm1 -ax64 -im-10000 -il-1000 -ix5 -ih5 -in0 | \bin\simplex >> \results\grid_nhard_simplex.res
\ \generadores\bin\spgrid 2048 32 417 -cc -cl1 -cm1 -ax64 -im-10000 -il-1000 -ix5 -ih5 -in0 | \bin\simplex >> \results\grid_nhard_simplex.res
\ \generadores\bin\spgrid 2048 32 419 -cc -cl1 -cm1 -ax64 -im-10000 -il-1000 -ix5 -ih5 -in0 | \bin\simplex >> \results\grid_nhard_simplex.res

\ \generadores\bin\spgrid 4096 32 511 -cc -cl1 -cm1 -ax64 -im-10000 -il-1000 -ix5 -ih5 -in0 | \bin\simplex >> \results\grid_nhard_simplex.res
\ \generadores\bin\spgrid 4096 32 513 -cc -cl1 -cm1 -ax64 -im-10000 -il-1000 -ix5 -ih5 -in0 | \bin\simplex >> \results\grid_nhard_simplex.res
\ \generadores\bin\spgrid 4096 32 515 -cc -cl1 -cm1 -ax64 -im-10000 -il-1000 -ix5 -ih5 -in0 | \bin\simplex >> \results\grid_nhard_simplex.res
\ \generadores\bin\spgrid 4096 32 517 -cc -cl1 -cm1 -ax64 -im-10000 -il-1000 -ix5 -ih5 -in0 | \bin\simplex >> \results\grid_nhard_simplex.res
\ \generadores\bin\spgrid 4096 32 519 -cc -cl1 -cm1 -ax64 -im-10000 -il-1000 -ix5 -ih5 -in0 | \bin\simplex >> \results\grid_nhard_simplex.res

\ \generadores\bin\spgrid 8192 32 611 -cc -cl1 -cm1 -ax64 -im-10000 -il-1000 -ix5 -ih5 -in0 | \bin\simplex >> \results\grid_nhard_simplex.res
\ \generadores\bin\spgrid 8192 32 613 -cc -cl1 -cm1 -ax64 -im-10000 -il-1000 -ix5 -ih5 -in0 | \bin\simplex >> \results\grid_nhard_simplex.res
\ \generadores\bin\spgrid 8192 32 615 -cc -cl1 -cm1 -ax64 -im-10000 -il-1000 -ix5 -ih5 -in0 | \bin\simplex >> \results\grid_nhard_simplex.res
\ \generadores\bin\spgrid 8192 32 617 -cc -cl1 -cm1 -ax64 -im-10000 -il-1000 -ix5 -ih5 -in0 | \bin\simplex >> \results\grid_nhard_simplex.res
\ \generadores\bin\spgrid 8192 32 619 -cc -cl1 -cm1 -ax64 -im-10000 -il-1000 -ix5 -ih5 -in0 | \bin\simplex >> \results\grid_nhard_simplex.res

\ \generadores\bin\sprand 131072 524288 961 -cl1 | \bin\simplex >> \results\rand_p_simplex.res
\ \generadores\bin\sprand 131072 524288 963 -cl1 | \bin\simplex >> \results\rand_p_simplex.res
\ \generadores\bin\sprand 131072 524288 965 -cl1 | \bin\simplex >> \results\rand_p_simplex.res
\ \generadores\bin\sprand 131072 524288 967 -cl1 | \bin\simplex >> \results\rand_p_simplex.res
\ \generadores\bin\sprand 131072 524288 969 -cl1 | \bin\simplex >> \results\rand_p_simplex.res

\ \generadores\bin\sprand 131072 524288 961 -pl1000 -cl1 | \bin\simplex >> \results\rand_p_simplex.res
\ \generadores\bin\sprand 131072 524288 963 -pl1000 -cl1 | \bin\simplex >> \results\rand_p_simplex.res
\ \generadores\bin\sprand 131072 524288 965 -pl1000 -cl1 | \bin\simplex >> \results\rand_p_simplex.res
\ \generadores\bin\sprand 131072 524288 967 -pl1000 -cl1 | \bin\simplex >> \results\rand_p_simplex.res
\ \generadores\bin\sprand 131072 524288 969 -pl1000 -cl1 | \bin\simplex >> \results\rand_p_simplex.res

\ \generadores\bin\sprand 131072 524288 961 -pl5000 -cl1 | \bin\simplex >> \results\rand_p_simplex.res
\ \generadores\bin\sprand 131072 524288 963 -pl5000 -cl1 | \bin\simplex >> \results\rand_p_simplex.res
\ \generadores\bin\sprand 131072 524288 965 -pl5000 -cl1 | \bin\simplex >> \results\rand_p_simplex.res
\ \generadores\bin\sprand 131072 524288 967 -pl5000 -cl1 | \bin\simplex >> \results\rand_p_simplex.res
\ \generadores\bin\sprand 131072 524288 969 -pl5000 -cl1 | \bin\simplex >> \results\rand_p_simplex.res

\ \generadores\bin\sprand 131072 524288 961 -pl10000 -cl1 | \bin\simplex >> \results\rand_p_simplex.res
\ \generadores\bin\sprand 131072 524288 963 -pl10000 -cl1 | \bin\simplex >> \results\rand_p_simplex.res
\ \generadores\bin\sprand 131072 524288 965 -pl10000 -cl1 | \bin\simplex >> \results\rand_p_simplex.res
\ \generadores\bin\sprand 131072 524288 967 -pl10000 -cl1 | \bin\simplex >> \results\rand_p_simplex.res
\ \generadores\bin\sprand 131072 524288 969 -pl10000 -cl1 | \bin\simplex >> \results\rand_p_simplex.res

\ \generadores\bin\sprand 131072 524288 961 -pl100000 -cl1 | \bin\simplex >> \results\rand_p_simplex.res
\ \generadores\bin\sprand 131072 524288 963 -pl100000 -cl1 | \bin\simplex >> \results\rand_p_simplex.res
\ \generadores\bin\sprand 131072 524288 965 -pl100000 -cl1 | \bin\simplex >> \results\rand_p_simplex.res
\ \generadores\bin\sprand 131072 524288 967 -pl100000 -cl1 | \bin\simplex >> \results\rand_p_simplex.res
\ \generadores\bin\sprand 131072 524288 969 -pl100000 -cl1 | \bin\simplex >> \results\rand_p_simplex.res

\ \generadores\bin\sprand 131072 524288 961 -pl1000000 -cl1 | \bin\simplex >> \results\rand_p_simplex.res
\ \generadores\bin\sprand 131072 524288 963 -pl1000000 -cl1 | \bin\simplex >> \results\rand_p_simplex.res
\ \generadores\bin\sprand 131072 524288 965 -pl1000000 -cl1 | \bin\simplex >> \results\rand_p_simplex.res
\ \generadores\bin\sprand 131072 524288 967 -pl1000000 -cl1 | \bin\simplex >> \results\rand_p_simplex.res
\ \generadores\bin\sprand 131072 524288 969 -pl1000000 -cl1 | \bin\simplex >> \results\rand_p_simplex.res

\ \generadores\bin\sprand 131072 524288 961 -pl5000000 -cl1 | \bin\simplex >> \results\rand_p_simplex.res
\ \generadores\bin\sprand 131072 524288 963 -pl5000000 -cl1 | \bin\simplex >> \results\rand_p_simplex.res
\ \generadores\bin\sprand 131072 524288 965 -pl5000000 -cl1 | \bin\simplex >> \results\rand_p_simplex.res
\ \generadores\bin\sprand 131072 524288 967 -pl5000000 -cl1 | \bin\simplex >> \results\rand_p_simplex.res
\ \generadores\bin\sprand 131072 524288 969 -pl5000000 -cl1 | \bin\simplex >> \results\rand_p_simplex.res
```

```
\ \generadores\bin\spacyc 8192 131072 561 -lm-10000 -ll0 -cl-1 | \bin\simplex > \results\acyc_neg_simplex.res
\ \generadores\bin\spacyc 8192 131072 563 -lm-10000 -ll0 -cl-1 | \bin\simplex >> \results\acyc_neg_simplex.res
\ \generadores\bin\spacyc 8192 131072 565 -lm-10000 -ll0 -cl-1 | \bin\simplex >> \results\acyc_neg_simplex.res
\ \generadores\bin\spacyc 8192 131072 567 -lm-10000 -ll0 -cl-1 | \bin\simplex >> \results\acyc_neg_simplex.res
\ \generadores\bin\spacyc 8192 131072 569 -lm-10000 -ll0 -cl-1 | \bin\simplex >> \results\acyc_neg_simplex.res

\ \generadores\bin\spacyc 16384 262144 661 -lm-10000 -ll0 -cl-1 | \bin\simplex >> \results\acyc_neg_simplex.res
\ \generadores\bin\spacyc 16384 262144 663 -lm-10000 -ll0 -cl-1 | \bin\simplex >> \results\acyc_neg_simplex.res
\ \generadores\bin\spacyc 16384 262144 665 -lm-10000 -ll0 -cl-1 | \bin\simplex >> \results\acyc_neg_simplex.res
\ \generadores\bin\spacyc 16384 262144 667 -lm-10000 -ll0 -cl-1 | \bin\simplex >> \results\acyc_neg_simplex.res
\ \generadores\bin\spacyc 16384 262144 669 -lm-10000 -ll0 -cl-1 | \bin\simplex >> \results\acyc_neg_simplex.res

\ \generadores\bin\spacyc 32768 524288 761 -lm-10000 -ll0 -cl-1 | \bin\simplex >> \results\acyc_neg_simplex.res
\ \generadores\bin\spacyc 32768 524288 763 -lm-10000 -ll0 -cl-1 | \bin\simplex >> \results\acyc_neg_simplex.res
\ \generadores\bin\spacyc 32768 524288 765 -lm-10000 -ll0 -cl-1 | \bin\simplex >> \results\acyc_neg_simplex.res
\ \generadores\bin\spacyc 32768 524288 767 -lm-10000 -ll0 -cl-1 | \bin\simplex >> \results\acyc_neg_simplex.res
\ \generadores\bin\spacyc 32768 524288 769 -lm-10000 -ll0 -cl-1 | \bin\simplex >> \results\acyc_neg_simplex.res

\ \generadores\bin\spacyc 65536 1048576 861 -lm-10000 -ll0 -cl-1 | \bin\simplex >> \results\acyc_neg_simplex.res
\ \generadores\bin\spacyc 65536 1048576 863 -lm-10000 -ll0 -cl-1 | \bin\simplex >> \results\acyc_neg_simplex.res
\ \generadores\bin\spacyc 65536 1048576 865 -lm-10000 -ll0 -cl-1 | \bin\simplex >> \results\acyc_neg_simplex.res
\ \generadores\bin\spacyc 65536 1048576 867 -lm-10000 -ll0 -cl-1 | \bin\simplex >> \results\acyc_neg_simplex.res
\ \generadores\bin\spacyc 65536 1048576 869 -lm-10000 -ll0 -cl-1 | \bin\simplex >> \results\acyc_neg_simplex.res

\ \generadores\bin\spacyc 131072 2097152 961 -lm-10000 -ll0 -cl-1 | \bin\simplex >> \results\acyc_neg_simplex.res
\ \generadores\bin\spacyc 131072 2097152 963 -lm-10000 -ll0 -cl-1 | \bin\simplex >> \results\acyc_neg_simplex.res
\ \generadores\bin\spacyc 131072 2097152 965 -lm-10000 -ll0 -cl-1 | \bin\simplex >> \results\acyc_neg_simplex.res
\ \generadores\bin\spacyc 131072 2097152 967 -lm-10000 -ll0 -cl-1 | \bin\simplex >> \results\acyc_neg_simplex.res
\ \generadores\bin\spacyc 131072 2097152 969 -lm-10000 -ll0 -cl-1 | \bin\simplex >> \results\acyc_neg_simplex.res

\ \generadores\bin\spacyc 16384 262144 461 -lm0 -ll10000 | \bin\simplex > \results\acyc_p2n_simplex.res
\ \generadores\bin\spacyc 16384 262144 463 -lm0 -ll10000 | \bin\simplex >> \results\acyc_p2n_simplex.res
\ \generadores\bin\spacyc 16384 262144 465 -lm0 -ll10000 | \bin\simplex >> \results\acyc_p2n_simplex.res
\ \generadores\bin\spacyc 16384 262144 467 -lm0 -ll10000 | \bin\simplex >> \results\acyc_p2n_simplex.res
\ \generadores\bin\spacyc 16384 262144 469 -lm0 -ll10000 | \bin\simplex >> \results\acyc_p2n_simplex.res

\ \generadores\bin\spacyc 16384 262144 761 -lm-1000 -ll9000 | \bin\simplex >> \results\acyc_p2n_simplex.res
\ \generadores\bin\spacyc 16384 262144 763 -lm-1000 -ll9000 | \bin\simplex >> \results\acyc_p2n_simplex.res
\ \generadores\bin\spacyc 16384 262144 765 -lm-1000 -ll9000 | \bin\simplex >> \results\acyc_p2n_simplex.res
\ \generadores\bin\spacyc 16384 262144 767 -lm-1000 -ll9000 | \bin\simplex >> \results\acyc_p2n_simplex.res
\ \generadores\bin\spacyc 16384 262144 769 -lm-1000 -ll9000 | \bin\simplex >> \results\acyc_p2n_simplex.res

\ \generadores\bin\spacyc 16384 262144 861 -lm-2000 -ll8000 | \bin\simplex >> \results\acyc_p2n_simplex.res
\ \generadores\bin\spacyc 16384 262144 863 -lm-2000 -ll8000 | \bin\simplex >> \results\acyc_p2n_simplex.res
\ \generadores\bin\spacyc 16384 262144 865 -lm-2000 -ll8000 | \bin\simplex >> \results\acyc_p2n_simplex.res
\ \generadores\bin\spacyc 16384 262144 867 -lm-2000 -ll8000 | \bin\simplex >> \results\acyc_p2n_simplex.res
\ \generadores\bin\spacyc 16384 262144 869 -lm-2000 -ll8000 | \bin\simplex >> \results\acyc_p2n_simplex.res

\ \generadores\bin\spacyc 16384 262144 961 -lm-3000 -ll7000 | \bin\simplex >> \results\acyc_p2n_simplex.res
\ \generadores\bin\spacyc 16384 262144 963 -lm-3000 -ll7000 | \bin\simplex >> \results\acyc_p2n_simplex.res
\ \generadores\bin\spacyc 16384 262144 965 -lm-3000 -ll7000 | \bin\simplex >> \results\acyc_p2n_simplex.res
\ \generadores\bin\spacyc 16384 262144 967 -lm-3000 -ll7000 | \bin\simplex >> \results\acyc_p2n_simplex.res
\ \generadores\bin\spacyc 16384 262144 969 -lm-3000 -ll7000 | \bin\simplex >> \results\acyc_p2n_simplex.res

\ \generadores\bin\spacyc 16384 262144 261 -lm-4000 -ll6000 | \bin\simplex >> \results\acyc_p2n_simplex.res
\ \generadores\bin\spacyc 16384 262144 263 -lm-4000 -ll6000 | \bin\simplex >> \results\acyc_p2n_simplex.res
\ \generadores\bin\spacyc 16384 262144 265 -lm-4000 -ll6000 | \bin\simplex >> \results\acyc_p2n_simplex.res
\ \generadores\bin\spacyc 16384 262144 267 -lm-4000 -ll6000 | \bin\simplex >> \results\acyc_p2n_simplex.res
\ \generadores\bin\spacyc 16384 262144 269 -lm-4000 -ll6000 | \bin\simplex >> \results\acyc_p2n_simplex.res

\ \generadores\bin\spacyc 16384 262144 461 -lm-5000 -ll5000 | \bin\simplex >> \results\acyc_p2n_simplex.res
\ \generadores\bin\spacyc 16384 262144 463 -lm-5000 -ll5000 | \bin\simplex >> \results\acyc_p2n_simplex.res
\ \generadores\bin\spacyc 16384 262144 465 -lm-5000 -ll5000 | \bin\simplex >> \results\acyc_p2n_simplex.res
\ \generadores\bin\spacyc 16384 262144 467 -lm-5000 -ll5000 | \bin\simplex >> \results\acyc_p2n_simplex.res
\ \generadores\bin\spacyc 16384 262144 469 -lm-5000 -ll5000 | \bin\simplex >> \results\acyc_p2n_simplex.res

\ \generadores\bin\spacyc 16384 262144 661 -lm-6000 -ll4000 | \bin\simplex >> \results\acyc_p2n_simplex.res
\ \generadores\bin\spacyc 16384 262144 663 -lm-6000 -ll4000 | \bin\simplex >> \results\acyc_p2n_simplex.res
\ \generadores\bin\spacyc 16384 262144 665 -lm-6000 -ll4000 | \bin\simplex >> \results\acyc_p2n_simplex.res
\ \generadores\bin\spacyc 16384 262144 667 -lm-6000 -ll4000 | \bin\simplex >> \results\acyc_p2n_simplex.res
```

```
\ \generadores\bin\spacyc 16384 262144 669 -lm-6000 -ll4000 | \bin\simplex >> \results\acyc_p2n_simplex.res

\ \generadores\bin\spacyc 16384 262144 761 -lm-10000 -ll0 | \bin\simplex >> \results\acyc_p2n_simplex.res
\ \generadores\bin\spacyc 16384 262144 763 -lm-10000 -ll0 | \bin\simplex >> \results\acyc_p2n_simplex.res
\ \generadores\bin\spacyc 16384 262144 765 -lm-10000 -ll0 | \bin\simplex >> \results\acyc_p2n_simplex.res
\ \generadores\bin\spacyc 16384 262144 767 -lm-10000 -ll0 | \bin\simplex >> \results\acyc_p2n_simplex.res
\ \generadores\bin\spacyc 16384 262144 769 -lm-10000 -ll0 | \bin\simplex >> \results\acyc_p2n_simplex.res
```

tesis_tarjan.bat

```
\ \generadores\bin\spgrid 256  32 111 -cc -cl1 -cm1 -ax64 -im-10000 -il-1000 -ix5 -ih5 -in0 | \bin\tarjan > \results\grid_nhard_tarjan.res
\ \generadores\bin\spgrid 256  32 113 -cc -cl1 -cm1 -ax64 -im-10000 -il-1000 -ix5 -ih5 -in0 | \bin\tarjan >> \results\grid_nhard_tarjan.res
\ \generadores\bin\spgrid 256  32 115 -cc -cl1 -cm1 -ax64 -im-10000 -il-1000 -ix5 -ih5 -in0 | \bin\tarjan >> \results\grid_nhard_tarjan.res
\ \generadores\bin\spgrid 256  32 117 -cc -cl1 -cm1 -ax64 -im-10000 -il-1000 -ix5 -ih5 -in0 | \bin\tarjan >> \results\grid_nhard_tarjan.res
\ \generadores\bin\spgrid 256  32 119 -cc -cl1 -cm1 -ax64 -im-10000 -il-1000 -ix5 -ih5 -in0 | \bin\tarjan >> \results\grid_nhard_tarjan.res

\ \generadores\bin\spgrid 512  32 211 -cc -cl1 -cm1 -ax64 -im-10000 -il-1000 -ix5 -ih5 -in0 | \bin\tarjan >> \results\grid_nhard_tarjan.res
\ \generadores\bin\spgrid 512  32 213 -cc -cl1 -cm1 -ax64 -im-10000 -il-1000 -ix5 -ih5 -in0 | \bin\tarjan >> \results\grid_nhard_tarjan.res
\ \generadores\bin\spgrid 512  32 215 -cc -cl1 -cm1 -ax64 -im-10000 -il-1000 -ix5 -ih5 -in0 | \bin\tarjan >> \results\grid_nhard_tarjan.res
\ \generadores\bin\spgrid 512  32 217 -cc -cl1 -cm1 -ax64 -im-10000 -il-1000 -ix5 -ih5 -in0 | \bin\tarjan >> \results\grid_nhard_tarjan.res
\ \generadores\bin\spgrid 512  32 219 -cc -cl1 -cm1 -ax64 -im-10000 -il-1000 -ix5 -ih5 -in0 | \bin\tarjan >> \results\grid_nhard_tarjan.res

\ \generadores\bin\spgrid 1024  32 311 -cc -cl1 -cm1 -ax64 -im-10000 -il-1000 -ix5 -ih5 -in0 | \bin\tarjan >> \results\grid_nhard_tarjan.res
\ \generadores\bin\spgrid 1024  32 313 -cc -cl1 -cm1 -ax64 -im-10000 -il-1000 -ix5 -ih5 -in0 | \bin\tarjan >> \results\grid_nhard_tarjan.res
\ \generadores\bin\spgrid 1024  32 315 -cc -cl1 -cm1 -ax64 -im-10000 -il-1000 -ix5 -ih5 -in0 | \bin\tarjan >> \results\grid_nhard_tarjan.res
\ \generadores\bin\spgrid 1024  32 317 -cc -cl1 -cm1 -ax64 -im-10000 -il-1000 -ix5 -ih5 -in0 | \bin\tarjan >> \results\grid_nhard_tarjan.res
\ \generadores\bin\spgrid 1024  32 319 -cc -cl1 -cm1 -ax64 -im-10000 -il-1000 -ix5 -ih5 -in0 | \bin\tarjan >> \results\grid_nhard_tarjan.res

\ \generadores\bin\spgrid 2048  32 411 -cc -cl1 -cm1 -ax64 -im-10000 -il-1000 -ix5 -ih5 -in0 | \bin\tarjan >> \results\grid_nhard_tarjan.res
\ \generadores\bin\spgrid 2048  32 413 -cc -cl1 -cm1 -ax64 -im-10000 -il-1000 -ix5 -ih5 -in0 | \bin\tarjan >> \results\grid_nhard_tarjan.res
\ \generadores\bin\spgrid 2048  32 415 -cc -cl1 -cm1 -ax64 -im-10000 -il-1000 -ix5 -ih5 -in0 | \bin\tarjan >> \results\grid_nhard_tarjan.res
\ \generadores\bin\spgrid 2048  32 417 -cc -cl1 -cm1 -ax64 -im-10000 -il-1000 -ix5 -ih5 -in0 | \bin\tarjan >> \results\grid_nhard_tarjan.res
\ \generadores\bin\spgrid 2048  32 419 -cc -cl1 -cm1 -ax64 -im-10000 -il-1000 -ix5 -ih5 -in0 | \bin\tarjan >> \results\grid_nhard_tarjan.res

\ \generadores\bin\spgrid 4096  32 511 -cc -cl1 -cm1 -ax64 -im-10000 -il-1000 -ix5 -ih5 -in0 | \bin\tarjan >> \results\grid_nhard_tarjan.res
\ \generadores\bin\spgrid 4096  32 513 -cc -cl1 -cm1 -ax64 -im-10000 -il-1000 -ix5 -ih5 -in0 | \bin\tarjan >> \results\grid_nhard_tarjan.res
\ \generadores\bin\spgrid 4096  32 515 -cc -cl1 -cm1 -ax64 -im-10000 -il-1000 -ix5 -ih5 -in0 | \bin\tarjan >> \results\grid_nhard_tarjan.res
\ \generadores\bin\spgrid 4096  32 517 -cc -cl1 -cm1 -ax64 -im-10000 -il-1000 -ix5 -ih5 -in0 | \bin\tarjan >> \results\grid_nhard_tarjan.res
\ \generadores\bin\spgrid 4096  32 519 -cc -cl1 -cm1 -ax64 -im-10000 -il-1000 -ix5 -ih5 -in0 | \bin\tarjan >> \results\grid_nhard_tarjan.res

\ \generadores\bin\spgrid 8192  32 611 -cc -cl1 -cm1 -ax64 -im-10000 -il-1000 -ix5 -ih5 -in0 | \bin\tarjan >> \results\grid_nhard_tarjan.res
\ \generadores\bin\spgrid 8192  32 613 -cc -cl1 -cm1 -ax64 -im-10000 -il-1000 -ix5 -ih5 -in0 | \bin\tarjan >> \results\grid_nhard_tarjan.res
\ \generadores\bin\spgrid 8192  32 615 -cc -cl1 -cm1 -ax64 -im-10000 -il-1000 -ix5 -ih5 -in0 | \bin\tarjan >> \results\grid_nhard_tarjan.res
\ \generadores\bin\spgrid 8192  32 617 -cc -cl1 -cm1 -ax64 -im-10000 -il-1000 -ix5 -ih5 -in0 | \bin\tarjan >> \results\grid_nhard_tarjan.res
\ \generadores\bin\spgrid 8192  32 619 -cc -cl1 -cm1 -ax64 -im-10000 -il-1000 -ix5 -ih5 -in0 | \bin\tarjan >> \results\grid_nhard_tarjan.res

\ \generadores\bin\sprand 131072 524288 961 -cl1 | \bin\tarjan > \results\rand_p_tarjan.res
\ \generadores\bin\sprand 131072 524288 963 -cl1 | \bin\tarjan >> \results\rand_p_tarjan.res
\ \generadores\bin\sprand 131072 524288 965 -cl1 | \bin\tarjan >> \results\rand_p_tarjan.res
\ \generadores\bin\sprand 131072 524288 967 -cl1 | \bin\tarjan >> \results\rand_p_tarjan.res
\ \generadores\bin\sprand 131072 524288 969 -cl1 | \bin\tarjan >> \results\rand_p_tarjan.res

\ \generadores\bin\sprand 131072 524288 961 -pl1000 -cl1 | \bin\tarjan >> \results\rand_p_tarjan.res
\ \generadores\bin\sprand 131072 524288 963 -pl1000 -cl1 | \bin\tarjan >> \results\rand_p_tarjan.res
\ \generadores\bin\sprand 131072 524288 965 -pl1000 -cl1 | \bin\tarjan >> \results\rand_p_tarjan.res
\ \generadores\bin\sprand 131072 524288 967 -pl1000 -cl1 | \bin\tarjan >> \results\rand_p_tarjan.res
\ \generadores\bin\sprand 131072 524288 969 -pl1000 -cl1 | \bin\tarjan >> \results\rand_p_tarjan.res

\ \generadores\bin\sprand 131072 524288 961 -pl5000 -cl1 | \bin\tarjan >> \results\rand_p_tarjan.res
\ \generadores\bin\sprand 131072 524288 963 -pl5000 -cl1 | \bin\tarjan >> \results\rand_p_tarjan.res
\ \generadores\bin\sprand 131072 524288 965 -pl5000 -cl1 | \bin\tarjan >> \results\rand_p_tarjan.res
\ \generadores\bin\sprand 131072 524288 967 -pl5000 -cl1 | \bin\tarjan >> \results\rand_p_tarjan.res
\ \generadores\bin\sprand 131072 524288 969 -pl5000 -cl1 | \bin\tarjan >> \results\rand_p_tarjan.res

\ \generadores\bin\sprand 131072 524288 961 -pl10000 -cl1 | \bin\tarjan >> \results\rand_p_tarjan.res
\ \generadores\bin\sprand 131072 524288 963 -pl10000 -cl1 | \bin\tarjan >> \results\rand_p_tarjan.res
\ \generadores\bin\sprand 131072 524288 965 -pl10000 -cl1 | \bin\tarjan >> \results\rand_p_tarjan.res
\ \generadores\bin\sprand 131072 524288 967 -pl10000 -cl1 | \bin\tarjan >> \results\rand_p_tarjan.res
\ \generadores\bin\sprand 131072 524288 969 -pl10000 -cl1 | \bin\tarjan >> \results\rand_p_tarjan.res

\ \generadores\bin\sprand 131072 524288 961 -pl100000 -cl1 | \bin\tarjan >> \results\rand_p_tarjan.res
\ \generadores\bin\sprand 131072 524288 963 -pl100000 -cl1 | \bin\tarjan >> \results\rand_p_tarjan.res
\ \generadores\bin\sprand 131072 524288 965 -pl100000 -cl1 | \bin\tarjan >> \results\rand_p_tarjan.res
```

```
.\.\generadores\bin\sprand 131072 524288 967 -pl100000 -cl1 | .\bin\tarjan >> .\results\rand_p_tarjan.res
.\.\generadores\bin\sprand 131072 524288 969 -pl100000 -cl1 | .\bin\tarjan >> .\results\rand_p_tarjan.res

.\.\generadores\bin\sprand 131072 524288 961 -pl1000000 -cl1 | .\bin\tarjan >> .\results\rand_p_tarjan.res
.\.\generadores\bin\sprand 131072 524288 963 -pl1000000 -cl1 | .\bin\tarjan >> .\results\rand_p_tarjan.res
.\.\generadores\bin\sprand 131072 524288 965 -pl1000000 -cl1 | .\bin\tarjan >> .\results\rand_p_tarjan.res
.\.\generadores\bin\sprand 131072 524288 967 -pl1000000 -cl1 | .\bin\tarjan >> .\results\rand_p_tarjan.res
.\.\generadores\bin\sprand 131072 524288 969 -pl1000000 -cl1 | .\bin\tarjan >> .\results\rand_p_tarjan.res

.\.\generadores\bin\sprand 131072 524288 961 -pl5000000 -cl1 | .\bin\tarjan >> .\results\rand_p_tarjan.res
.\.\generadores\bin\sprand 131072 524288 963 -pl5000000 -cl1 | .\bin\tarjan >> .\results\rand_p_tarjan.res
.\.\generadores\bin\sprand 131072 524288 965 -pl5000000 -cl1 | .\bin\tarjan >> .\results\rand_p_tarjan.res
.\.\generadores\bin\sprand 131072 524288 967 -pl5000000 -cl1 | .\bin\tarjan >> .\results\rand_p_tarjan.res
.\.\generadores\bin\sprand 131072 524288 969 -pl5000000 -cl1 | .\bin\tarjan >> .\results\rand_p_tarjan.res

.\.\generadores\bin\spacyc 8192 131072 561 -lm-10000 -ll0 -cl-1 | .\bin\tarjan > .\results\acyc_neg_tarjan.res
.\.\generadores\bin\spacyc 8192 131072 563 -lm-10000 -ll0 -cl-1 | .\bin\tarjan >> .\results\acyc_neg_tarjan.res
.\.\generadores\bin\spacyc 8192 131072 565 -lm-10000 -ll0 -cl-1 | .\bin\tarjan >> .\results\acyc_neg_tarjan.res
.\.\generadores\bin\spacyc 8192 131072 567 -lm-10000 -ll0 -cl-1 | .\bin\tarjan >> .\results\acyc_neg_tarjan.res
.\.\generadores\bin\spacyc 8192 131072 569 -lm-10000 -ll0 -cl-1 | .\bin\tarjan >> .\results\acyc_neg_tarjan.res

.\.\generadores\bin\spacyc 16384 262144 661 -lm-10000 -ll0 -cl-1 | .\bin\tarjan >> .\results\acyc_neg_tarjan.res
.\.\generadores\bin\spacyc 16384 262144 663 -lm-10000 -ll0 -cl-1 | .\bin\tarjan >> .\results\acyc_neg_tarjan.res
.\.\generadores\bin\spacyc 16384 262144 665 -lm-10000 -ll0 -cl-1 | .\bin\tarjan >> .\results\acyc_neg_tarjan.res
.\.\generadores\bin\spacyc 16384 262144 667 -lm-10000 -ll0 -cl-1 | .\bin\tarjan >> .\results\acyc_neg_tarjan.res
.\.\generadores\bin\spacyc 16384 262144 669 -lm-10000 -ll0 -cl-1 | .\bin\tarjan >> .\results\acyc_neg_tarjan.res

.\.\generadores\bin\spacyc 32768 524288 761 -lm-10000 -ll0 -cl-1 | .\bin\tarjan >> .\results\acyc_neg_tarjan.res
.\.\generadores\bin\spacyc 32768 524288 763 -lm-10000 -ll0 -cl-1 | .\bin\tarjan >> .\results\acyc_neg_tarjan.res
.\.\generadores\bin\spacyc 32768 524288 765 -lm-10000 -ll0 -cl-1 | .\bin\tarjan >> .\results\acyc_neg_tarjan.res
.\.\generadores\bin\spacyc 32768 524288 767 -lm-10000 -ll0 -cl-1 | .\bin\tarjan >> .\results\acyc_neg_tarjan.res
.\.\generadores\bin\spacyc 32768 524288 769 -lm-10000 -ll0 -cl-1 | .\bin\tarjan >> .\results\acyc_neg_tarjan.res

.\.\generadores\bin\spacyc 65536 1048576 861 -lm-10000 -ll0 -cl-1 | .\bin\tarjan >> .\results\acyc_neg_tarjan.res
.\.\generadores\bin\spacyc 65536 1048576 863 -lm-10000 -ll0 -cl-1 | .\bin\tarjan >> .\results\acyc_neg_tarjan.res
.\.\generadores\bin\spacyc 65536 1048576 865 -lm-10000 -ll0 -cl-1 | .\bin\tarjan >> .\results\acyc_neg_tarjan.res
.\.\generadores\bin\spacyc 65536 1048576 867 -lm-10000 -ll0 -cl-1 | .\bin\tarjan >> .\results\acyc_neg_tarjan.res
.\.\generadores\bin\spacyc 65536 1048576 869 -lm-10000 -ll0 -cl-1 | .\bin\tarjan >> .\results\acyc_neg_tarjan.res

.\.\generadores\bin\spacyc 131072 2097152 961 -lm-10000 -ll0 -cl-1 | .\bin\tarjan >> .\results\acyc_neg_tarjan.res
.\.\generadores\bin\spacyc 131072 2097152 963 -lm-10000 -ll0 -cl-1 | .\bin\tarjan >> .\results\acyc_neg_tarjan.res
.\.\generadores\bin\spacyc 131072 2097152 965 -lm-10000 -ll0 -cl-1 | .\bin\tarjan >> .\results\acyc_neg_tarjan.res
.\.\generadores\bin\spacyc 131072 2097152 967 -lm-10000 -ll0 -cl-1 | .\bin\tarjan >> .\results\acyc_neg_tarjan.res
.\.\generadores\bin\spacyc 131072 2097152 969 -lm-10000 -ll0 -cl-1 | .\bin\tarjan >> .\results\acyc_neg_tarjan.res

.\.\generadores\bin\spacyc 16384 262144 461 -lm0 -ll10000 | .\bin\tarjan > .\results\acyc_p2n_tarjan.res
.\.\generadores\bin\spacyc 16384 262144 463 -lm0 -ll10000 | .\bin\tarjan >> .\results\acyc_p2n_tarjan.res
.\.\generadores\bin\spacyc 16384 262144 465 -lm0 -ll10000 | .\bin\tarjan >> .\results\acyc_p2n_tarjan.res
.\.\generadores\bin\spacyc 16384 262144 467 -lm0 -ll10000 | .\bin\tarjan >> .\results\acyc_p2n_tarjan.res
.\.\generadores\bin\spacyc 16384 262144 469 -lm0 -ll10000 | .\bin\tarjan >> .\results\acyc_p2n_tarjan.res

.\.\generadores\bin\spacyc 16384 262144 761 -lm-1000 -ll9000 | .\bin\tarjan >> .\results\acyc_p2n_tarjan.res
.\.\generadores\bin\spacyc 16384 262144 763 -lm-1000 -ll9000 | .\bin\tarjan >> .\results\acyc_p2n_tarjan.res
.\.\generadores\bin\spacyc 16384 262144 765 -lm-1000 -ll9000 | .\bin\tarjan >> .\results\acyc_p2n_tarjan.res
.\.\generadores\bin\spacyc 16384 262144 767 -lm-1000 -ll9000 | .\bin\tarjan >> .\results\acyc_p2n_tarjan.res
.\.\generadores\bin\spacyc 16384 262144 769 -lm-1000 -ll9000 | .\bin\tarjan >> .\results\acyc_p2n_tarjan.res

.\.\generadores\bin\spacyc 16384 262144 861 -lm-2000 -ll8000 | .\bin\tarjan >> .\results\acyc_p2n_tarjan.res
.\.\generadores\bin\spacyc 16384 262144 863 -lm-2000 -ll8000 | .\bin\tarjan >> .\results\acyc_p2n_tarjan.res
.\.\generadores\bin\spacyc 16384 262144 865 -lm-2000 -ll8000 | .\bin\tarjan >> .\results\acyc_p2n_tarjan.res
.\.\generadores\bin\spacyc 16384 262144 867 -lm-2000 -ll8000 | .\bin\tarjan >> .\results\acyc_p2n_tarjan.res
.\.\generadores\bin\spacyc 16384 262144 869 -lm-2000 -ll8000 | .\bin\tarjan >> .\results\acyc_p2n_tarjan.res

.\.\generadores\bin\spacyc 16384 262144 961 -lm-3000 -ll7000 | .\bin\tarjan >> .\results\acyc_p2n_tarjan.res
.\.\generadores\bin\spacyc 16384 262144 963 -lm-3000 -ll7000 | .\bin\tarjan >> .\results\acyc_p2n_tarjan.res
.\.\generadores\bin\spacyc 16384 262144 965 -lm-3000 -ll7000 | .\bin\tarjan >> .\results\acyc_p2n_tarjan.res
.\.\generadores\bin\spacyc 16384 262144 967 -lm-3000 -ll7000 | .\bin\tarjan >> .\results\acyc_p2n_tarjan.res
.\.\generadores\bin\spacyc 16384 262144 969 -lm-3000 -ll7000 | .\bin\tarjan >> .\results\acyc_p2n_tarjan.res

.\.\generadores\bin\spacyc 16384 262144 261 -lm-4000 -ll6000 | .\bin\tarjan >> .\results\acyc_p2n_tarjan.res
```

```
\ \generadores\bin\spacyc 16384 262144 263 -lm-4000 -ll6000 | \bin\tarjan >> \results\acyc_p2n_tarjan.res
\ \generadores\bin\spacyc 16384 262144 265 -lm-4000 -ll6000 | \bin\tarjan >> \results\acyc_p2n_tarjan.res
\ \generadores\bin\spacyc 16384 262144 267 -lm-4000 -ll6000 | \bin\tarjan >> \results\acyc_p2n_tarjan.res
\ \generadores\bin\spacyc 16384 262144 269 -lm-4000 -ll6000 | \bin\tarjan >> \results\acyc_p2n_tarjan.res

\ \generadores\bin\spacyc 16384 262144 461 -lm-5000 -ll5000 | \bin\tarjan >> \results\acyc_p2n_tarjan.res
\ \generadores\bin\spacyc 16384 262144 463 -lm-5000 -ll5000 | \bin\tarjan >> \results\acyc_p2n_tarjan.res
\ \generadores\bin\spacyc 16384 262144 465 -lm-5000 -ll5000 | \bin\tarjan >> \results\acyc_p2n_tarjan.res
\ \generadores\bin\spacyc 16384 262144 467 -lm-5000 -ll5000 | \bin\tarjan >> \results\acyc_p2n_tarjan.res
\ \generadores\bin\spacyc 16384 262144 469 -lm-5000 -ll5000 | \bin\tarjan >> \results\acyc_p2n_tarjan.res

\ \generadores\bin\spacyc 16384 262144 661 -lm-6000 -ll4000 | \bin\tarjan >> \results\acyc_p2n_tarjan.res
\ \generadores\bin\spacyc 16384 262144 663 -lm-6000 -ll4000 | \bin\tarjan >> \results\acyc_p2n_tarjan.res
\ \generadores\bin\spacyc 16384 262144 665 -lm-6000 -ll4000 | \bin\tarjan >> \results\acyc_p2n_tarjan.res
\ \generadores\bin\spacyc 16384 262144 667 -lm-6000 -ll4000 | \bin\tarjan >> \results\acyc_p2n_tarjan.res
\ \generadores\bin\spacyc 16384 262144 669 -lm-6000 -ll4000 | \bin\tarjan >> \results\acyc_p2n_tarjan.res

\ \generadores\bin\spacyc 16384 262144 761 -lm-10000 -ll0 | \bin\tarjan >> \results\acyc_p2n_tarjan.res
\ \generadores\bin\spacyc 16384 262144 763 -lm-10000 -ll0 | \bin\tarjan >> \results\acyc_p2n_tarjan.res
\ \generadores\bin\spacyc 16384 262144 765 -lm-10000 -ll0 | \bin\tarjan >> \results\acyc_p2n_tarjan.res
\ \generadores\bin\spacyc 16384 262144 767 -lm-10000 -ll0 | \bin\tarjan >> \results\acyc_p2n_tarjan.res
\ \generadores\bin\spacyc 16384 262144 769 -lm-10000 -ll0 | \bin\tarjan >> \results\acyc_p2n_tarjan.res
```

tesis_Goldberg.bat

```
\ \generadores\bin\spgrid 256  32 111 -cc -cl1 -cm1 -ax64 -im-10000 -il-1000 -ix5 -ih5 -in0 | \bin\goldberg >  \results\grid_nhard_goldberg.res
\ \generadores\bin\spgrid 256  32 113 -cc -cl1 -cm1 -ax64 -im-10000 -il-1000 -ix5 -ih5 -in0 | \bin\goldberg >> \results\grid_nhard_goldberg.res
\ \generadores\bin\spgrid 256  32 115 -cc -cl1 -cm1 -ax64 -im-10000 -il-1000 -ix5 -ih5 -in0 | \bin\goldberg >> \results\grid_nhard_goldberg.res
\ \generadores\bin\spgrid 256  32 117 -cc -cl1 -cm1 -ax64 -im-10000 -il-1000 -ix5 -ih5 -in0 | \bin\goldberg >> \results\grid_nhard_goldberg.res
\ \generadores\bin\spgrid 256  32 119 -cc -cl1 -cm1 -ax64 -im-10000 -il-1000 -ix5 -ih5 -in0 | \bin\goldberg >> \results\grid_nhard_goldberg.res

\ \generadores\bin\spgrid 512  32 211 -cc -cl1 -cm1 -ax64 -im-10000 -il-1000 -ix5 -ih5 -in0 | \bin\goldberg >> \results\grid_nhard_goldberg.res
\ \generadores\bin\spgrid 512  32 213 -cc -cl1 -cm1 -ax64 -im-10000 -il-1000 -ix5 -ih5 -in0 | \bin\goldberg >> \results\grid_nhard_goldberg.res
\ \generadores\bin\spgrid 512  32 215 -cc -cl1 -cm1 -ax64 -im-10000 -il-1000 -ix5 -ih5 -in0 | \bin\goldberg >> \results\grid_nhard_goldberg.res
\ \generadores\bin\spgrid 512  32 217 -cc -cl1 -cm1 -ax64 -im-10000 -il-1000 -ix5 -ih5 -in0 | \bin\goldberg >> \results\grid_nhard_goldberg.res
\ \generadores\bin\spgrid 512  32 219 -cc -cl1 -cm1 -ax64 -im-10000 -il-1000 -ix5 -ih5 -in0 | \bin\goldberg >> \results\grid_nhard_goldberg.res

\ \generadores\bin\spgrid 1024  32 311 -cc -cl1 -cm1 -ax64 -im-10000 -il-1000 -ix5 -ih5 -in0 | \bin\goldberg >> \results\grid_nhard_goldberg.res
\ \generadores\bin\spgrid 1024  32 313 -cc -cl1 -cm1 -ax64 -im-10000 -il-1000 -ix5 -ih5 -in0 | \bin\goldberg >> \results\grid_nhard_goldberg.res
\ \generadores\bin\spgrid 1024  32 315 -cc -cl1 -cm1 -ax64 -im-10000 -il-1000 -ix5 -ih5 -in0 | \bin\goldberg >> \results\grid_nhard_goldberg.res
\ \generadores\bin\spgrid 1024  32 317 -cc -cl1 -cm1 -ax64 -im-10000 -il-1000 -ix5 -ih5 -in0 | \bin\goldberg >> \results\grid_nhard_goldberg.res
\ \generadores\bin\spgrid 1024  32 319 -cc -cl1 -cm1 -ax64 -im-10000 -il-1000 -ix5 -ih5 -in0 | \bin\goldberg >> \results\grid_nhard_goldberg.res

\ \generadores\bin\spgrid 2048  32 411 -cc -cl1 -cm1 -ax64 -im-10000 -il-1000 -ix5 -ih5 -in0 | \bin\goldberg >> \results\grid_nhard_goldberg.res
\ \generadores\bin\spgrid 2048  32 413 -cc -cl1 -cm1 -ax64 -im-10000 -il-1000 -ix5 -ih5 -in0 | \bin\goldberg >> \results\grid_nhard_goldberg.res
\ \generadores\bin\spgrid 2048  32 415 -cc -cl1 -cm1 -ax64 -im-10000 -il-1000 -ix5 -ih5 -in0 | \bin\goldberg >> \results\grid_nhard_goldberg.res
\ \generadores\bin\spgrid 2048  32 417 -cc -cl1 -cm1 -ax64 -im-10000 -il-1000 -ix5 -ih5 -in0 | \bin\goldberg >> \results\grid_nhard_goldberg.res
\ \generadores\bin\spgrid 2048  32 419 -cc -cl1 -cm1 -ax64 -im-10000 -il-1000 -ix5 -ih5 -in0 | \bin\goldberg >> \results\grid_nhard_goldberg.res

\ \generadores\bin\spgrid 4096  32 511 -cc -cl1 -cm1 -ax64 -im-10000 -il-1000 -ix5 -ih5 -in0 | \bin\goldberg >> \results\grid_nhard_goldberg.res
\ \generadores\bin\spgrid 4096  32 513 -cc -cl1 -cm1 -ax64 -im-10000 -il-1000 -ix5 -ih5 -in0 | \bin\goldberg >> \results\grid_nhard_goldberg.res
\ \generadores\bin\spgrid 4096  32 515 -cc -cl1 -cm1 -ax64 -im-10000 -il-1000 -ix5 -ih5 -in0 | \bin\goldberg >> \results\grid_nhard_goldberg.res
\ \generadores\bin\spgrid 4096  32 517 -cc -cl1 -cm1 -ax64 -im-10000 -il-1000 -ix5 -ih5 -in0 | \bin\goldberg >> \results\grid_nhard_goldberg.res
\ \generadores\bin\spgrid 4096  32 519 -cc -cl1 -cm1 -ax64 -im-10000 -il-1000 -ix5 -ih5 -in0 | \bin\goldberg >> \results\grid_nhard_goldberg.res

\ \generadores\bin\spgrid 8192  32 611 -cc -cl1 -cm1 -ax64 -im-10000 -il-1000 -ix5 -ih5 -in0 | \bin\goldberg >> \results\grid_nhard_goldberg.res
\ \generadores\bin\spgrid 8192  32 613 -cc -cl1 -cm1 -ax64 -im-10000 -il-1000 -ix5 -ih5 -in0 | \bin\goldberg >> \results\grid_nhard_goldberg.res
\ \generadores\bin\spgrid 8192  32 615 -cc -cl1 -cm1 -ax64 -im-10000 -il-1000 -ix5 -ih5 -in0 | \bin\goldberg >> \results\grid_nhard_goldberg.res
\ \generadores\bin\spgrid 8192  32 617 -cc -cl1 -cm1 -ax64 -im-10000 -il-1000 -ix5 -ih5 -in0 | \bin\goldberg >> \results\grid_nhard_goldberg.res
\ \generadores\bin\spgrid 8192  32 619 -cc -cl1 -cm1 -ax64 -im-10000 -il-1000 -ix5 -ih5 -in0 | \bin\goldberg >> \results\grid_nhard_goldberg.res

\ \generadores\bin\sprand 131072 524288 961 -cl1 | \bin\goldberg >  \results\rand_p_goldberg.res
\ \generadores\bin\sprand 131072 524288 963 -cl1 | \bin\goldberg >> \results\rand_p_goldberg.res
\ \generadores\bin\sprand 131072 524288 965 -cl1 | \bin\goldberg >> \results\rand_p_goldberg.res
\ \generadores\bin\sprand 131072 524288 967 -cl1 | \bin\goldberg >> \results\rand_p_goldberg.res
\ \generadores\bin\sprand 131072 524288 969 -cl1 | \bin\goldberg >> \results\rand_p_goldberg.res

\ \generadores\bin\sprand 131072 524288 961 -pl1000 -cl1 | \bin\goldberg >> \results\rand_p_goldberg.res
\ \generadores\bin\sprand 131072 524288 963 -pl1000 -cl1 | \bin\goldberg >> \results\rand_p_goldberg.res
\ \generadores\bin\sprand 131072 524288 965 -pl1000 -cl1 | \bin\goldberg >> \results\rand_p_goldberg.res
\ \generadores\bin\sprand 131072 524288 967 -pl1000 -cl1 | \bin\goldberg >> \results\rand_p_goldberg.res
\ \generadores\bin\sprand 131072 524288 969 -pl1000 -cl1 | \bin\goldberg >> \results\rand_p_goldberg.res
```

```
\ \generadores\bin\sprand 131072 524288 961 -pl5000 -cl1 | \bin\goldberg >> \results\rand_p_goldberg res
\ \generadores\bin\sprand 131072 524288 963 -pl5000 -cl1 | \bin\goldberg >> \results\rand_p_goldberg res
\ \generadores\bin\sprand 131072 524288 965 -pl5000 -cl1 | \bin\goldberg >> \results\rand_p_goldberg res
\ \generadores\bin\sprand 131072 524288 967 -pl5000 -cl1 | \bin\goldberg >> \results\rand_p_goldberg res
\ \generadores\bin\sprand 131072 524288 969 -pl5000 -cl1 | \bin\goldberg >> \results\rand_p_goldberg res

\ \generadores\bin\sprand 131072 524288 961 -pl10000 -cl1 | \bin\goldberg >> \results\rand_p_goldberg res
\ \generadores\bin\sprand 131072 524288 963 -pl10000 -cl1 | \bin\goldberg >> \results\rand_p_goldberg res
\ \generadores\bin\sprand 131072 524288 965 -pl10000 -cl1 | \bin\goldberg >> \results\rand_p_goldberg res
\ \generadores\bin\sprand 131072 524288 967 -pl10000 -cl1 | \bin\goldberg >> \results\rand_p_goldberg res
\ \generadores\bin\sprand 131072 524288 969 -pl10000 -cl1 | \bin\goldberg >> \results\rand_p_goldberg res

\ \generadores\bin\sprand 131072 524288 961 -pl100000 -cl1 | \bin\goldberg >> \results\rand_p_goldberg res
\ \generadores\bin\sprand 131072 524288 963 -pl100000 -cl1 | \bin\goldberg >> \results\rand_p_goldberg res
\ \generadores\bin\sprand 131072 524288 965 -pl100000 -cl1 | \bin\goldberg >> \results\rand_p_goldberg res
\ \generadores\bin\sprand 131072 524288 967 -pl100000 -cl1 | \bin\goldberg >> \results\rand_p_goldberg res
\ \generadores\bin\sprand 131072 524288 969 -pl100000 -cl1 | \bin\goldberg >> \results\rand_p_goldberg res

\ \generadores\bin\sprand 131072 524288 961 -pl1000000 -cl1 | \bin\goldberg >> \results\rand_p_goldberg res
\ \generadores\bin\sprand 131072 524288 963 -pl1000000 -cl1 | \bin\goldberg >> \results\rand_p_goldberg res
\ \generadores\bin\sprand 131072 524288 965 -pl1000000 -cl1 | \bin\goldberg >> \results\rand_p_goldberg res
\ \generadores\bin\sprand 131072 524288 967 -pl1000000 -cl1 | \bin\goldberg >> \results\rand_p_goldberg res
\ \generadores\bin\sprand 131072 524288 969 -pl1000000 -cl1 | \bin\goldberg >> \results\rand_p_goldberg res

\ \generadores\bin\sprand 131072 524288 961 -pl5000000 -cl1 | \bin\goldberg >> \results\rand_p_goldberg res
\ \generadores\bin\sprand 131072 524288 963 -pl5000000 -cl1 | \bin\goldberg >> \results\rand_p_goldberg res
\ \generadores\bin\sprand 131072 524288 965 -pl5000000 -cl1 | \bin\goldberg >> \results\rand_p_goldberg res
\ \generadores\bin\sprand 131072 524288 967 -pl5000000 -cl1 | \bin\goldberg >> \results\rand_p_goldberg res
\ \generadores\bin\sprand 131072 524288 969 -pl5000000 -cl1 | \bin\goldberg >> \results\rand_p_goldberg res

\ \generadores\bin\spacyc 8192 131072 561 -lm-10000 -ll0 -cl-1 | \bin\goldberg > \results\acyc_neg_goldberg res
\ \generadores\bin\spacyc 8192 131072 563 -lm-10000 -ll0 -cl-1 | \bin\goldberg >> \results\acyc_neg_goldberg res
\ \generadores\bin\spacyc 8192 131072 565 -lm-10000 -ll0 -cl-1 | \bin\goldberg >> \results\acyc_neg_goldberg res
\ \generadores\bin\spacyc 8192 131072 567 -lm-10000 -ll0 -cl-1 | \bin\goldberg >> \results\acyc_neg_goldberg res
\ \generadores\bin\spacyc 8192 131072 569 -lm-10000 -ll0 -cl-1 | \bin\goldberg >> \results\acyc_neg_goldberg res

\ \generadores\bin\spacyc 16384 262144 661 -lm-10000 -ll0 -cl-1 | \bin\goldberg >> \results\acyc_neg_goldberg res
\ \generadores\bin\spacyc 16384 262144 663 -lm-10000 -ll0 -cl-1 | \bin\goldberg >> \results\acyc_neg_goldberg res
\ \generadores\bin\spacyc 16384 262144 665 -lm-10000 -ll0 -cl-1 | \bin\goldberg >> \results\acyc_neg_goldberg res
\ \generadores\bin\spacyc 16384 262144 667 -lm-10000 -ll0 -cl-1 | \bin\goldberg >> \results\acyc_neg_goldberg res
\ \generadores\bin\spacyc 16384 262144 669 -lm-10000 -ll0 -cl-1 | \bin\goldberg >> \results\acyc_neg_goldberg res

\ \generadores\bin\spacyc 32768 524288 761 -lm-10000 -ll0 -cl-1 | \bin\goldberg >> \results\acyc_neg_goldberg res
\ \generadores\bin\spacyc 32768 524288 763 -lm-10000 -ll0 -cl-1 | \bin\goldberg >> \results\acyc_neg_goldberg res
\ \generadores\bin\spacyc 32768 524288 765 -lm-10000 -ll0 -cl-1 | \bin\goldberg >> \results\acyc_neg_goldberg res
\ \generadores\bin\spacyc 32768 524288 767 -lm-10000 -ll0 -cl-1 | \bin\goldberg >> \results\acyc_neg_goldberg res
\ \generadores\bin\spacyc 32768 524288 769 -lm-10000 -ll0 -cl-1 | \bin\goldberg >> \results\acyc_neg_goldberg res

\ \generadores\bin\spacyc 65536 1048576 861 -lm-10000 -ll0 -cl-1 | \bin\goldberg >> \results\acyc_neg_goldberg res
\ \generadores\bin\spacyc 65536 1048576 863 -lm-10000 -ll0 -cl-1 | \bin\goldberg >> \results\acyc_neg_goldberg res
\ \generadores\bin\spacyc 65536 1048576 865 -lm-10000 -ll0 -cl-1 | \bin\goldberg >> \results\acyc_neg_goldberg res
\ \generadores\bin\spacyc 65536 1048576 867 -lm-10000 -ll0 -cl-1 | \bin\goldberg >> \results\acyc_neg_goldberg res
\ \generadores\bin\spacyc 65536 1048576 869 -lm-10000 -ll0 -cl-1 | \bin\goldberg >> \results\acyc_neg_goldberg res

\ \generadores\bin\spacyc 131072 2097152 961 -lm-10000 -ll0 -cl-1 | \bin\goldberg >> \results\acyc_neg_goldberg res
\ \generadores\bin\spacyc 131072 2097152 963 -lm-10000 -ll0 -cl-1 | \bin\goldberg >> \results\acyc_neg_goldberg res
\ \generadores\bin\spacyc 131072 2097152 965 -lm-10000 -ll0 -cl-1 | \bin\goldberg >> \results\acyc_neg_goldberg res
\ \generadores\bin\spacyc 131072 2097152 967 -lm-10000 -ll0 -cl-1 | \bin\goldberg >> \results\acyc_neg_goldberg res
\ \generadores\bin\spacyc 131072 2097152 969 -lm-10000 -ll0 -cl-1 | \bin\goldberg >> \results\acyc_neg_goldberg res

\ \generadores\bin\spacyc 16384 262144 461 -lm0 -ll10000 | \bin\goldberg > \results\acyc_p2n_goldberg res
\ \generadores\bin\spacyc 16384 262144 463 -lm0 -ll10000 | \bin\goldberg >> \results\acyc_p2n_goldberg res
\ \generadores\bin\spacyc 16384 262144 465 -lm0 -ll10000 | \bin\goldberg >> \results\acyc_p2n_goldberg res
\ \generadores\bin\spacyc 16384 262144 467 -lm0 -ll10000 | \bin\goldberg >> \results\acyc_p2n_goldberg res
\ \generadores\bin\spacyc 16384 262144 469 -lm0 -ll10000 | \bin\goldberg >> \results\acyc_p2n_goldberg res

\ \generadores\bin\spacyc 16384 262144 761 -lm-1000 -ll9000 | \bin\goldberg >> \results\acyc_p2n_goldberg res
\ \generadores\bin\spacyc 16384 262144 763 -lm-1000 -ll9000 | \bin\goldberg >> \results\acyc_p2n_goldberg res
\ \generadores\bin\spacyc 16384 262144 765 -lm-1000 -ll9000 | \bin\goldberg >> \results\acyc_p2n_goldberg res
\ \generadores\bin\spacyc 16384 262144 767 -lm-1000 -ll9000 | \bin\goldberg >> \results\acyc_p2n_goldberg res
```

```
\ \generadores\bin\spacyc 16384 262144 769 -lm-1000 -ll9000 |  \bin\goldberg >>  \results\acyc_p2n_goldberg.res

\ \generadores\bin\spacyc 16384 262144 861 -lm-2000 -ll8000 |  \bin\goldberg >>  \results\acyc_p2n_goldberg.res
\ \generadores\bin\spacyc 16384 262144 863 -lm-2000 -ll8000 |  \bin\goldberg >>  \results\acyc_p2n_goldberg.res
\ \generadores\bin\spacyc 16384 262144 865 -lm-2000 -ll8000 |  \bin\goldberg >>  \results\acyc_p2n_goldberg.res
\ \generadores\bin\spacyc 16384 262144 867 -lm-2000 -ll8000 |  \bin\goldberg >>  \results\acyc_p2n_goldberg.res
\ \generadores\bin\spacyc 16384 262144 869 -lm-2000 -ll8000 |  \bin\goldberg >>  \results\acyc_p2n_goldberg.res

\ \generadores\bin\spacyc 16384 262144 961 -lm-3000 -ll7000 |  \bin\goldberg >>  \results\acyc_p2n_goldberg.res
\ \generadores\bin\spacyc 16384 262144 963 -lm-3000 -ll7000 |  \bin\goldberg >>  \results\acyc_p2n_goldberg.res
\ \generadores\bin\spacyc 16384 262144 965 -lm-3000 -ll7000 |  \bin\goldberg >>  \results\acyc_p2n_goldberg.res
\ \generadores\bin\spacyc 16384 262144 967 -lm-3000 -ll7000 |  \bin\goldberg >>  \results\acyc_p2n_goldberg.res
\ \generadores\bin\spacyc 16384 262144 969 -lm-3000 -ll7000 |  \bin\goldberg >>  \results\acyc_p2n_goldberg.res

\ \generadores\bin\spacyc 16384 262144 261 -lm-4000 -ll6000 |  \bin\goldberg >>  \results\acyc_p2n_goldberg.res
\ \generadores\bin\spacyc 16384 262144 263 -lm-4000 -ll6000 |  \bin\goldberg >>  \results\acyc_p2n_goldberg.res
\ \generadores\bin\spacyc 16384 262144 265 -lm-4000 -ll6000 |  \bin\goldberg >>  \results\acyc_p2n_goldberg.res
\ \generadores\bin\spacyc 16384 262144 267 -lm-4000 -ll6000 |  \bin\goldberg >>  \results\acyc_p2n_goldberg.res
\ \generadores\bin\spacyc 16384 262144 269 -lm-4000 -ll6000 |  \bin\goldberg >>  \results\acyc_p2n_goldberg.res

\ \generadores\bin\spacyc 16384 262144 461 -lm-5000 -ll5000 |  \bin\goldberg >>  \results\acyc_p2n_goldberg.res
\ \generadores\bin\spacyc 16384 262144 463 -lm-5000 -ll5000 |  \bin\goldberg >>  \results\acyc_p2n_goldberg.res
\ \generadores\bin\spacyc 16384 262144 465 -lm-5000 -ll5000 |  \bin\goldberg >>  \results\acyc_p2n_goldberg.res
\ \generadores\bin\spacyc 16384 262144 467 -lm-5000 -ll5000 |  \bin\goldberg >>  \results\acyc_p2n_goldberg.res
\ \generadores\bin\spacyc 16384 262144 469 -lm-5000 -ll5000 |  \bin\goldberg >>  \results\acyc_p2n_goldberg.res

\ \generadores\bin\spacyc 16384 262144 661 -lm-6000 -ll4000 |  \bin\goldberg >>  \results\acyc_p2n_goldberg.res
\ \generadores\bin\spacyc 16384 262144 663 -lm-6000 -ll4000 |  \bin\goldberg >>  \results\acyc_p2n_goldberg.res
\ \generadores\bin\spacyc 16384 262144 665 -lm-6000 -ll4000 |  \bin\goldberg >>  \results\acyc_p2n_goldberg.res
\ \generadores\bin\spacyc 16384 262144 667 -lm-6000 -ll4000 |  \bin\goldberg >>  \results\acyc_p2n_goldberg.res
\ \generadores\bin\spacyc 16384 262144 669 -lm-6000 -ll4000 |  \bin\goldberg >>  \results\acyc_p2n_goldberg.res

\ \generadores\bin\spacyc 16384 262144 761 -lm-10000 -ll0 |  \bin\goldberg >>  \results\acyc_p2n_goldberg.res
\ \generadores\bin\spacyc 16384 262144 763 -lm-10000 -ll0 |  \bin\goldberg >>  \results\acyc_p2n_goldberg.res
\ \generadores\bin\spacyc 16384 262144 765 -lm-10000 -ll0 |  \bin\goldberg >>  \results\acyc_p2n_goldberg.res
\ \generadores\bin\spacyc 16384 262144 767 -lm-10000 -ll0 |  \bin\goldberg >>  \results\acyc_p2n_goldberg.res
\ \generadores\bin\spacyc 16384 262144 769 -lm-10000 -ll0 |  \bin\goldberg >>  \results\acyc_p2n_goldberg.res
```

tesis_Variantedikh.bat

```
\ \generadores\bin\sprand 131072 524288 961 -cl1 |  \bin\dikh >  \results\rand_p_dikh.res
\ \generadores\bin\sprand 131072 524288 963 -cl1 |  \bin\dikh >>  \results\rand_p_dikh.res
\ \generadores\bin\sprand 131072 524288 965 -cl1 |  \bin\dikh >>  \results\rand_p_dikh.res
\ \generadores\bin\sprand 131072 524288 967 -cl1 |  \bin\dikh >>  \results\rand_p_dikh.res
\ \generadores\bin\sprand 131072 524288 969 -cl1 |  \bin\dikh >>  \results\rand_p_dikh.res

\ \generadores\bin\sprand 131072 524288 961 -pl1000 -cl1 |  \bin\dikh >>  \results\rand_p_dikh.res
\ \generadores\bin\sprand 131072 524288 963 -pl1000 -cl1 |  \bin\dikh >>  \results\rand_p_dikh.res
\ \generadores\bin\sprand 131072 524288 965 -pl1000 -cl1 |  \bin\dikh >>  \results\rand_p_dikh.res
\ \generadores\bin\sprand 131072 524288 967 -pl1000 -cl1 |  \bin\dikh >>  \results\rand_p_dikh.res
\ \generadores\bin\sprand 131072 524288 969 -pl1000 -cl1 |  \bin\dikh >>  \results\rand_p_dikh.res

\ \generadores\bin\sprand 131072 524288 961 -pl5000 -cl1 |  \bin\dikh >>  \results\rand_p_dikh.res
\ \generadores\bin\sprand 131072 524288 963 -pl5000 -cl1 |  \bin\dikh >>  \results\rand_p_dikh.res
\ \generadores\bin\sprand 131072 524288 965 -pl5000 -cl1 |  \bin\dikh >>  \results\rand_p_dikh.res
\ \generadores\bin\sprand 131072 524288 967 -pl5000 -cl1 |  \bin\dikh >>  \results\rand_p_dikh.res
\ \generadores\bin\sprand 131072 524288 969 -pl5000 -cl1 |  \bin\dikh >>  \results\rand_p_dikh.res

\ \generadores\bin\sprand 131072 524288 961 -pl10000 -cl1 |  \bin\dikh >>  \results\rand_p_dikh.res
\ \generadores\bin\sprand 131072 524288 963 -pl10000 -cl1 |  \bin\dikh >>  \results\rand_p_dikh.res
\ \generadores\bin\sprand 131072 524288 965 -pl10000 -cl1 |  \bin\dikh >>  \results\rand_p_dikh.res
\ \generadores\bin\sprand 131072 524288 967 -pl10000 -cl1 |  \bin\dikh >>  \results\rand_p_dikh.res
\ \generadores\bin\sprand 131072 524288 969 -pl10000 -cl1 |  \bin\dikh >>  \results\rand_p_dikh.res

\ \generadores\bin\sprand 131072 524288 961 -pl100000 -cl1 |  \bin\dikh >>  \results\rand_p_dikh.res
\ \generadores\bin\sprand 131072 524288 963 -pl100000 -cl1 |  \bin\dikh >>  \results\rand_p_dikh.res
\ \generadores\bin\sprand 131072 524288 965 -pl100000 -cl1 |  \bin\dikh >>  \results\rand_p_dikh.res
\ \generadores\bin\sprand 131072 524288 967 -pl100000 -cl1 |  \bin\dikh >>  \results\rand_p_dikh.res
\ \generadores\bin\sprand 131072 524288 969 -pl100000 -cl1 |  \bin\dikh >>  \results\rand_p_dikh.res
```

```
.\ \generadores\bin\sprand 131072 524288 961 -pl1000000 -cl1 | .\bin\dikh >> .\results\rand_p_dikh.res
.\ \generadores\bin\sprand 131072 524288 963 -pl1000000 -cl1 | .\bin\dikh >> .\results\rand_p_dikh.res
.\ \generadores\bin\sprand 131072 524288 965 -pl1000000 -cl1 | .\bin\dikh >> .\results\rand_p_dikh.res
.\ \generadores\bin\sprand 131072 524288 967 -pl1000000 -cl1 | .\bin\dikh >> .\results\rand_p_dikh.res
.\ \generadores\bin\sprand 131072 524288 969 -pl1000000 -cl1 | .\bin\dikh >> .\results\rand_p_dikh.res

.\ \generadores\bin\sprand 131072 524288 961 -pl5000000 -cl1 | .\bin\dikh >> .\results\rand_p_dikh.res
.\ \generadores\bin\sprand 131072 524288 963 -pl5000000 -cl1 | .\bin\dikh >> .\results\rand_p_dikh.res
.\ \generadores\bin\sprand 131072 524288 965 -pl5000000 -cl1 | .\bin\dikh >> .\results\rand_p_dikh.res
.\ \generadores\bin\sprand 131072 524288 967 -pl5000000 -cl1 | .\bin\dikh >> .\results\rand_p_dikh.res
.\ \generadores\bin\sprand 131072 524288 969 -pl5000000 -cl1 | .\bin\dikh >> .\results\rand_p_dikh.res

.\ \generadores\bin\spacyc 8192 131072 561 -lm-10000 -ll0 -cl-1 | .\bin\dikh > .\results\acyc_neg_dikh.res
.\ \generadores\bin\spacyc 8192 131072 563 -lm-10000 -ll0 -cl-1 | .\bin\dikh >> .\results\acyc_neg_dikh.res
.\ \generadores\bin\spacyc 8192 131072 565 -lm-10000 -ll0 -cl-1 | .\bin\dikh >> .\results\acyc_neg_dikh.res
.\ \generadores\bin\spacyc 8192 131072 567 -lm-10000 -ll0 -cl-1 | .\bin\dikh >> .\results\acyc_neg_dikh.res
.\ \generadores\bin\spacyc 8192 131072 569 -lm-10000 -ll0 -cl-1 | .\bin\dikh >> .\results\acyc_neg_dikh.res

.\ \generadores\bin\spacyc 16384 262144 461 -lm0 -ll10000 | \bin\dikh > .\results\acyc_p2n_dikh.res
.\ \generadores\bin\spacyc 16384 262144 463 -lm0 -ll10000 | .\bin\dikh >> .\results\acyc_p2n_dikh.res
.\ \generadores\bin\spacyc 16384 262144 465 -lm0 -ll10000 | .\bin\dikh >> .\results\acyc_p2n_dikh.res
.\ \generadores\bin\spacyc 16384 262144 467 -lm0 -ll10000 | .\bin\dikh >> .\results\acyc_p2n_dikh.res
.\ \generadores\bin\spacyc 16384 262144 469 -lm0 -ll10000 | \bin\dikh >> .\results\acyc_p2n_dikh.res

.\ \generadores\bin\spacyc 16384 262144 761 -lm-1000 -ll9000 | .\bin\dikh >> .\results\acyc_p2n_dikh.res
.\ \generadores\bin\spacyc 16384 262144 763 -lm-1000 -ll9000 | .\bin\dikh >> .\results\acyc_p2n_dikh.res
.\ \generadores\bin\spacyc 16384 262144 765 -lm-1000 -ll9000 | \bin\dikh >> .\results\acyc_p2n_dikh.res
.\ \generadores\bin\spacyc 16384 262144 767 -lm-1000 -ll9000 | \bin\dikh >> .\results\acyc_p2n_dikh.res
.\ \generadores\bin\spacyc 16384 262144 769 -lm-1000 -ll9000 | \bin\dikh >> .\results\acyc_p2n_dikh.res

.\ \generadores\bin\spacyc 16384 262144 861 -lm-2000 -ll8000 | .\bin\dikh >> .\results\acyc_p2n_dikh.res
.\ \generadores\bin\spacyc 16384 262144 863 -lm-2000 -ll8000 | \bin\dikh >> .\results\acyc_p2n_dikh.res
.\ \generadores\bin\spacyc 16384 262144 865 -lm-2000 -ll8000 | .\bin\dikh >> .\results\acyc_p2n_dikh.res
.\ \generadores\bin\spacyc 16384 262144 867 -lm-2000 -ll8000 | .\bin\dikh >> .\results\acyc_p2n_dikh.res
.\ \generadores\bin\spacyc 16384 262144 869 -lm-2000 -ll8000 | .\bin\dikh >> .\results\acyc_p2n_dikh.res

.\ \generadores\bin\spacyc 16384 262144 961 -lm-3000 -ll7000 | .\bin\dikh >> .\results\acyc_p2n_dikh.res
.\ \generadores\bin\spacyc 16384 262144 963 -lm-3000 -ll7000 | \bin\dikh >> .\results\acyc_p2n_dikh.res
.\ \generadores\bin\spacyc 16384 262144 965 -lm-3000 -ll7000 | .\bin\dikh >> .\results\acyc_p2n_dikh.res
.\ \generadores\bin\spacyc 16384 262144 967 -lm-3000 -ll7000 | .\bin\dikh >> .\results\acyc_p2n_dikh.res
.\ \generadores\bin\spacyc 16384 262144 969 -lm-3000 -ll7000 | .\bin\dikh >> .\results\acyc_p2n_dikh.res

.\ \generadores\bin\spacyc 16384 262144 261 -lm-4000 -ll6000 | .\bin\dikh >> .\results\acyc_p2n_dikh.res
.\ \generadores\bin\spacyc 16384 262144 263 -lm-4000 -ll6000 | .\bin\dikh >> .\results\acyc_p2n_dikh.res
.\ \generadores\bin\spacyc 16384 262144 265 -lm-4000 -ll6000 | \bin\dikh >> .\results\acyc_p2n_dikh.res
.\ \generadores\bin\spacyc 16384 262144 267 -lm-4000 -ll6000 | .\bin\dikh >> .\results\acyc_p2n_dikh.res
.\ \generadores\bin\spacyc 16384 262144 269 -lm-4000 -ll6000 | \bin\dikh >> .\results\acyc_p2n_dikh.res
```

APÉNDICE C

En este apéndice se incluyen los resultados completos obtenidos en los experimentos.

C.1 Tablas de resultados para los diferentes tipos de redes

A continuación se muestra la tabla de resultados de cada uno de los algoritmos para las diferentes familias de redes utilizadas.

Redes tipo Grid-NHard

Algoritmo de Dijkstra Modificado

Nodos	*Arcos*	*Suma de la distancia*	*No. de escaneo*	*Tiempo*
8193	63808	-10202712340	125748	0.39
8193	63808	-10246924614	119598	0.33
8193	63808	-10247924342	120948	0.33
8193	63808	-10255959657	140774	0.39
8193	63808	-10235833185	135652	0.33
			15.689	**0.354**
16385	129344	-40739702729	306717	0.88
16385	129344	-40823828614	311078	0.88
16385	129344	-40844834252	288773	0.87
16385	129344	-40859616970	262509	0.77
16385	129344	-40781015944	308325	0.94
			18.034	**0.868**
32769	260416	-163137802589	623152	1.98
32769	260416	-163349629789	582181	1.87
32769	260416	-163278242461	621054	1.92
32769	260416	-163428031650	650026	2.04
32769	260416	-163360204742	609449	1.93
			18.834	**1.948**
65537	522560	-652215310630	1267851	4.23
65537	522560	-653213616144	1379077	4.51
65537	522560	-652928092983	1255804	4.17
65537	522560	-653108697571	1267916	4.23
65537	522560	-652077867875	1311751	4.28
			19.782	**4.284**

131073	1046848	-2610065428415	2777290	9.45
131073	1046848	-2609624287706	2855616	9.77
131073	1046848	-2608680747788	2766769	9.44
131073	1046848	-2611670850022	2725519	9 4
131073	1046848	-2609678725248	2728301	9.34
			21.139	**9.48**
262145	2095424	-10435849308363	6198683	21 86
262145	2095424	-10435763881065	5844700	20.71
262145	2095424	-10431621586787	5593858	19 77
262145	2095424	-10436055904027	6723989	23.62
262145	2095424	-10431371432094	6170855	21 81
			23.294	**21.554**

Algoritmo Simplex

Nodos	***Arcos***	***Suma de la distancia***	***No. de escaneo***	***Tiempo***
8193	63808	-10202712340	68807	0 49
8193	63808	-10246924614	70626	0 44
8193	63808	-10247924342	73883	0 6
8193	63808	-10255959657	72427	0 5
8193	63808	-10235833185	69975	0 44
			8.683	**0.494**
16385	129344	-40739702729	146164	1.37
16385	129344	-40823828614	146033	1.21
16385	129344	-40844834252	145624	1.21
16385	129344	-40859616970	145932	1.15
16385	129344	-40781015944	148639	1 15
			8.940	**1.218**
32769	260416	-163137802589	306038	3.35
32769	260416	-163349629789	296413	2.96
32769	260416	-163278242461	299079	2.86
32769	260416	-163428031650	295364	2.53
32769	260416	-163360204742	291016	2.97
			9.081	**2.934**
65537	522560	-652215310630	595635	6 15
65537	522560	-653213616144	608171	6 32
65537	522560	-652928092983	603314	6.15
65537	522560	-653108697571	594741	5.72
65537	522560	-652077867875	598912	6 59
			9.157	**6.186**
131073	1046848	-2610065428415	1218018	12.75
131073	1046848	-2609624287706	1173754	12.35
131073	1046848	-2608680747788	1203769	13.29
131073	1046848	-2611670850022	1163128	12 69
131073	1046848	-2609678725248	1199426	12 63
			9.091	**12.742**
262145	2095424	-10435849308363	2411275	26 59
262145	2095424	-10435763881065	2362357	25 38
262145	2095424	-10431621586787	2394088	25.54
262145	2095424	-10436055904027	2383979	24.77
262145	2095424	-10431371432094	2417353	26 31
			9.132	**25.718**

Algoritmo de Tarjan

Nodos	***Arcos***	***Suma de la distancia***	***No. de escaneo***	***Tiempo***
8193	63808	-10202712340	228028	0 39
8193	63808	-10246924614	216156	0.39

Nodos	Arcos	Suma de la distancia	No. de escaneo	Tiempo
8193	63808	-10247924342	227967	0 38
8193	63808	-10255959657	217597	0 38
8193	63808	-10235833185	216610	0.38
			27.007	**0.384**
16385	129344	-40739702729	478557	0 88
16385	129344	-40823828614	467491	0.82
16385	129344	-40844834252	475532	0.82
16385	129344	-40859616970	475272	0 88
16385	129344	-40781015944	487790	0 88
			29.108	**0.856**
32769	260416	-163137802589	999239	1 81
32769	260416	-163349629789	997305	1 75
32769	260416	-163278242461	1016923	1.81
32769	260416	-163428031650	1009608	1.76
32769	260416	-163360204742	999570	1.76
			30.655	**1.778**
65537	522560	-652215310630	2018126	3 63
65537	522560	-653213616144	2007559	3.57
65537	522560	-652928092983	2001421	3.57
65537	522560	-653108697571	2034596	3 68
65537	522560	-652077867875	2054696	3.68
			30.872	**3.626**
131073	1046848	-2610065428415	4044072	7.25
131073	1046848	-2609624287706	4113485	7.41
131073	1046848	-2608680747788	4151966	7.47
131073	1046848	-2611670850022	4128207	7 47
131073	1046848	-2609678725248	4085184	7.36
			31.315	**7.392**
262145	2095424	-10435849308363	8243343	14.83
262145	2095424	-10435763881065	8204346	14.83
262145	2095424	-10431621586787	8181124	14 78
262145	2095424	-10436055904027	8182982	14.72
262145	2095424	-10431371432094	8222933	14.88
			31.307	**14.808**

Algoritmo de Goldberg-Radzik

Nodos	Arcos	Suma de la distancia	No. de escaneo	Tiempo
8193	63808	-10202712340	146662	0 17
8193	63808	-10246924614	158092	0.16
8193	63808	-10247924342	145904	0 17
8193	63808	-10255959657	146398	0 16
8193	63808	-10235833185	150692	0.17
			18.253	**0.166**
16385	129344	-40739702729	343364	0.44
16385	129344	-40823828614	324510	0.44
16385	129344	-40844834252	323146	0.44
16385	129344	-40859616970	291966	0 44
16385	129344	-40781015944	341622	0 5
			19.830	**0.452**
32769	260416	-163137802589	686202	0 94
32769	260416	-163349629789	663450	0.93
32769	260416	-163278242461	697300	0.98
32769	260416	-163428031650	638824	0.93
32769	260416	-163360204742	662040	0 93
			20.433	**0.942**

Nodos	Arcos	Suma de la distancia	No. de escaneo	Tiempo
65537	522560	-652215310630	1516178	2.25
65537	522560	-653213616144	1297062	1.92
65537	522560	-652928092983	1385148	2.09
65537	522560	-653108697571	1370814	2.03
65537	522560	-652077867875	1540766	2.31
			21.698	**2.12**
131073	1046848	-2610065428415	2812758	4.28
131073	1046848	-2609624287706	2988700	4.56
131073	1046848	-2608680747788	3103418	4.67
131073	1046848	-2611670850022	2944492	4.45
131073	1046848	-2609678725248	2933388	4.5
			22.557	**4.492**
262145	2095424	-10435849308363	5755006	8.78
262145	2095424	-10435763881065	5570842	8.51
262145	2095424	-10431621586787	6110246	9.34
262145	2095424	-10436055904027	6235642	9.55
262145	2095424	-10431371432094	6172862	9.45
			22.770	**9.126**

Redes tipo Rand P

Algoritmo de Dijkstra Modificado

P	Nodos	Arcos	Suma de la distancia	No. de escaneo	Tiempo
0	131072	524288	58778073	131072	0.99
0	131072	524288	59031006	131072	0.99
0	131072	524288	54913132	131072	1.04
0	131072	524288	61545397	131072	0.98
0	131072	524288	66920392	131072	0.99
				1.000	**131072**
1000	131072	524288	40459439	467655	1.97
1000	131072	524288	114420816	445733	2.04
1000	131072	524288	52969199	440344	1.87
1000	131072	524288	2277821	458493	1.92
1000	131072	524288	81490065	456396	1.92
				3.462	**1.944**
5000	131072	524288	-32814892	921492	3.46
5000	131072	524288	335979396	901565	3.41
5000	131072	524288	45062325	902536	3.46
5000	131072	524288	-234399699	988998	3.68
5000	131072	524288	139507082	931599	3.52
				7.090	**3.506**
10000	131072	524288	-124407941	1359757	5
10000	131072	524288	612928242	1356991	5.05
10000	131072	524288	35211801	1173128	4.45
10000	131072	524288	-530344636	1290318	4.83
10000	131072	524288	212093615	1262436	4.72
				9.831	**4.81**
100000	131072	524288	-1772820650	2067696	8.24
100000	131072	524288	5596824417	2099643	8.18
100000	131072	524288	-142101048	2036779	8.02
100000	131072	524288	-5856176510	2075130	8.3
100000	131072	524288	1518390866	2263291	8.73
				16.087	**8.294**

P	Nodos	Arcos	Suma de la distancia	No. de escaneo	Tiempo
1000000	131072	524288	-18257602809	2357316	9.61
1000000	131072	524288	55436835235	2151188	8.79
1000000	131072	524288	-1915228699	2131143	8.68
1000000	131072	524288	-59115542219	2230636	9.23
1000000	131072	524288	14582018013	2303330	9.61
				17.050	**9.184**
5000000	131072	524288	-91523126533	2193029	9.39
5000000	131072	524288	276947658889	2291924	9.34
5000000	131072	524288	-9795926460	2147231	8.85
5000000	131072	524288	-295823892451	2181950	9.06
5000000	131072	524288	72642014756	2499673	10.32
				17.263	**9.392**

Algoritmo Simplex

P	Nodos	Arcos	Suma de la distancia	No. de escaneo	Tiempo
0	131072	524288	58778073	1130038	3.18
0	131072	524288	59031006	1216976	3.46
0	131072	524288	54913132	1160473	3.35
0	131072	524288	61545397	1161141	3.3
0	131072	524288	66920392	1212775	3.46
				8.974	**3.35**
1000	131072	524288	40459439	1130038	3.19
1000	131072	524288	114420816	1216976	3.46
1000	131072	524288	52969199	1160473	3.35
1000	131072	524288	2277821	1161141	3.29
1000	131072	524288	81490065	1212775	3.41
				8.974	**3.34**
5000	131072	524288	-32814892	1130038	3.19
5000	131072	524288	335979396	1216976	3.46
5000	131072	524288	45062325	1160473	3.35
5000	131072	524288	-234399699	1161141	3.3
5000	131072	524288	139507082	1212775	3.46
				8.974	**3.352**
10000	131072	524288	-124407941	1130038	3.24
10000	131072	524288	612928242	1216976	3.46
10000	131072	524288	35211801	1160473	3.35
10000	131072	524288	-530344636	1161141	3.29
10000	131072	524288	212093615	1212775	3.4
				8.974	**3.348**
100000	131072	524288	-1772820650	1130038	3.19
100000	131072	524288	5596824417	1216976	3.46
100000	131072	524288	-142101048	1160473	3.3
100000	131072	524288	-5856176510	1161141	3.29
100000	131072	524288	1518390866	1212775	3.4
				8.974	**3.328**
1000000	131072	524288	-18257602809	1130038	3.14
1000000	131072	524288	55436835235	1216976	3.46
1000000	131072	524288	-1915228699	1160473	3.35
1000000	131072	524288	-59115542219	1161141	3.24
1000000	131072	524288	14582018013	1212775	3.4
				8.974	**3.318**
5000000	131072	524288	-91523126533	1130038	3.19
5000000	131072	524288	276947658889	1216976	3.46
5000000	131072	524288	-9795926460	1160473	3.35

P	Nodos	Arcos	Suma de la distancia	No. de escaneo	Tiempo
5000000	131072	524288	-295823892451	1161141	3.29
5000000	131072	524288	72642014756	1212775	3.4
				8.974	**3.338**

Algoritmo de Tarjan

P	Nodos	Arcos	Suma de la distancia	No. de escaneo	Tiempo
0	131072	524288	58778073	1306523	3.4
0	131072	524288	59031006	1426426	3.73
0	131072	524288	54913132	1361666	3.57
0	131072	524288	61545397	1346942	3.52
0	131072	524288	66920392	1424715	3.68
				10.477	**3.58**
1000	131072	524288	40459439	1306523	3.4
1000	131072	524288	114420816	1426426	3.73
1000	131072	524288	52969199	1361666	3.57
1000	131072	524288	2277821	1346942	3.57
1000	131072	524288	81490065	1424715	3.74
				10.477	**3.602**
5000	131072	524288	-32814892	1306523	3.4
5000	131072	524288	335979396	1426426	3.73
5000	131072	524288	45062325	1361666	3.62
5000	131072	524288	-234399699	1346942	3.51
5000	131072	524288	139507082	1424715	3.68
				10.477	**3.588**
10000	131072	524288	-124407941	1306523	3.41
10000	131072	524288	612928242	1426426	3.73
10000	131072	524288	35211801	1361666	3.57
10000	131072	524288	-530344636	1346942	3.52
10000	131072	524288	212093615	1424715	3.68
				10.477	**3.582**
100000	131072	524288	-1772820650	1306523	3.4
100000	131072	524288	5596824417	1426426	3.73
100000	131072	524288	-142101048	1361666	3.63
100000	131072	524288	-5856176510	1346942	3.51
100000	131072	524288	1518390866	1424715	3.68
				10.477	**3.59**
1000000	131072	524288	-18257602809	1306523	3.4
1000000	131072	524288	55436835235	1426426	3.74
1000000	131072	524288	-1915228699	1361666	3.57
1000000	131072	524288	-59115542219	1346942	3.51
1000000	131072	524288	14582018013	1424715	3.73
				10.477	**3.59**
5000000	131072	524288	-91523126533	1306523	3.41
5000000	131072	524288	276947658889	1426426	3.73
5000000	131072	524288	-9795926460	1361666	3.57
5000000	131072	524288	-295823892451	1346942	3.52
5000000	131072	524288	72642014756	1424715	3.68
				10.477	**3.582**

Algoritmo de Goldberg-Radzik

P	Nodos	Arcos	Suma de la distancia	No. de escaneo	Tiempo
0	131072	524288	58778073	2430752	3.57
0	131072	524288	59031006	2629168	3.84
0	131072	524288	54913132	2603006	3.79

P	Nodos	Arcos	Suma de la distancia	No. de escaneo	Tiempo
0	131072	524288	61545397	2452144	3 63
0	131072	524288	66920392	2650658	3 84
				19.479	**3.734**
1000	131072	524288	40459439	2380840	3 46
1000	131072	524288	114420816	2604428	3 79
1000	131072	524288	52969199	2513472	3 63
1000	131072	524288	2277821	2387294	3 46
1000	131072	524288	81490065	2494784	3 62
				18.892	**3.592**
5000	131072	524288	-32814892	2312738	3 35
5000	131072	524288	335979396	2747962	4 01
5000	131072	524288	45062325	2617500	3 79
5000	131072	524288	-234399699	2432292	3 51
5000	131072	524288	139507082	2480140	3.57
				19.212	**3.646**
10000	131072	524288	-124407941	2427464	3.51
10000	131072	524288	612928242	2649560	3.79
10000	131072	524288	35211801	2547884	3.73
10000	131072	524288	-530344636	2472788	3.57
10000	131072	524288	212093615	2627160	3.79
				19.417	**3.678**
100000	131072	524288	-1772820650	2496938	3 68
100000	131072	524288	5596824417	2710516	3 95
100000	131072	524288	-142101048	2583554	3 78
100000	131072	524288	-5856176510	2463224	3 57
100000	131072	524288	1518390866	2695598	3 9
				19.760	**3.776**
1000000	131072	524288	-18257602809	2531962	3 68
1000000	131072	524288	55436835235	2709948	3 95
1000000	131072	524288	-1915228699	2633574	3 84
1000000	131072	524288	-59115542219	2510054	3.68
1000000	131072	524288	14582018013	2578472	3.79
				19.782	**3.788**
5000000	131072	524288	-91523126533	2484544	3 62
5000000	131072	524288	276947658889	2755670	4 01
5000000	131072	524288	-9795926460	2574460	3 79
5000000	131072	524288	-295823892451	2412136	3.57
5000000	131072	524288	72642014756	2573834	3.74
				19.532	**3.746**

Algoritmo Variante Dijkstra utilizado por Goldberg y Cherkassky

P	Nodos	Arcos	Suma de la distancia	No. de escaneo	Tiempo
0	131072	524288	58778073	131072	0 99
0	131072	524288	59031006	131072	1.1
0	131072	524288	54913132	131072	0.99
0	131072	524288	61545397	131072	0.99
0	131072	524288	66920392	131072	0 99
				1.000	**1.012**
1000	131072	524288	40459439	167615	1.26
1000	131072	524288	114420816	168209	1.21
1000	131072	524288	52969199	167490	1 2
1000	131072	524288	2277821	166983	1.26
1000	131072	524288	81490065	168557	1.21
				1.280	**1.228**

P	Nodos	Arcos	Suma de la distancia	No. de escaneo	Tiempo
5000	131072	524288	-32814892	393189	2.31
5000	131072	524288	335979396	393601	2.3
5000	131072	524288	45062325	395169	2.3
5000	131072	524288	-234399699	387804	2.25
5000	131072	524288	139507082	390925	2.31
				2.992	**2.294**
10000	131072	524288	-124407941	720891	3.79
10000	131072	524288	612928242	724589	3.79
10000	131072	524288	35211801	724749	3.79
10000	131072	524288	-530344636	732183	3.84
10000	131072	524288	212093615	737739	3.9
				5.554	**3.822**
100000	131072	524288	-1772820650	3247186	15.11
100000	131072	524288	5596824417	2791341	12.9
100000	131072	524288	-142101048	3091248	14.17
100000	131072	524288	-5856176510	3088438	14.17
100000	131072	524288	1518390866	3328919	15.11
				23.723	**14.292**
1000000	131072	524288	-18257602809	4708887	22.08
1000000	131072	524288	55436835235	3872449	18.13
1000000	131072	524288	-1915228699	4422227	20.44
1000000	131072	524288	-59115542219	4482087	20.82
1000000	131072	524288	14582018013	4665968	21.31
				33.801	**20.556**
5000000	131072	524288	-91523126533	4996776	23.57
5000000	131072	524288	276947658889	4107743	19.38
5000000	131072	524288	-9795926460	4737958	22.14
5000000	131072	524288	-295823892451	4729307	22.3
5000000	131072	524288	72642014756	4976246	22.96
				35.931	**22.07**

Redes tipo Acyc-Neg

Algoritmo de Dijkstra Modificado

Nodos	Arcos	Suma de la distancia	No. de escaneo	Tiempo
8192	131072	-9921510130	121884	0.44
8192	131072	-10137936625	102974	0.39
8192	131072	-10468517619	114574	0.44
8192	131072	-10020203123	103911	0.38
8192	131072	-9947989615	114086	0.44
			13.609	**0.418**
16384	262144	-28537596853	273743	1.21
16384	262144	-28464254867	270157	1.16
16384	262144	-28300889601	258710	1.1
16384	262144	-29225407855	256195	1.04
16384	262144	-28406151792	254398	1.1
			16.030	**1.122**
32768	524288	-82329035120	538850	2.74
32768	524288	-81658271971	578438	2.86
32768	524288	-79099847275	576271	2.85
32768	524288	-81299759275	562703	2.85
32768	524288	-80793751804	536766	2.69
			17.047	**2.798**

Nodos	Arcos	Suma de la distancia	No. de escaneo	Tiempo
65536	1048576	-238256105591	1206280	6 75
65536	1048576	-227040970169	1179909	6 42
65536	1048576	-225349712932	1198972	6.48
65536	1048576	-221040047980	1211354	6.59
65536	1048576	-231299060104	1155766	6.43
			18.165	**6.534**
131072	2097152	-658751422203	2659690	16.75
131072	2097152	-650518421486	2627669	15 87
131072	2097152	-654566097938	2549480	15.33
131072	2097152	-643772117619	2507949	15.05
131072	2097152	-639019008560	2436194	14 77
			19.502	**15.554**

Algoritmo Simplex

Nodos	Arcos	Suma de la distancia	No. de escaneo	Tiempo
8192	131072	-9921510130	102784	0 44
8192	131072	-10137936625	96386	0 49
8192	131072	-10468517619	105469	0.49
8192	131072	-10020203123	101613	0.49
8192	131072	-9947989615	100855	0 44
			12.381	**0.47**
16384	262144	-28537596853	239670	1.59
16384	262144	-28464254867	243281	1 75
16384	262144	-28300889601	225225	1 64
16384	262144	-29225407855	246582	1.71
16384	262144	-28406151792	235292	1.65
			14.527	**1.668**
32768	524288	-82329035120	544342	6.26
32768	524288	-81658271971	523759	6.05
32768	524288	-79099847275	515931	5 88
32768	524288	-81299759275	559147	6.04
32768	524288	-80793751804	508942	5.98
			16.187	**6.042**
65536	1048576	-238256105591	1202429	20 37
65536	1048576	-227040970169	1198266	20.05
65536	1048576	-225349712932	1195389	19.44
65536	1048576	-221040047980	1226753	19.39
65536	1048576	-231299060104	1212113	19 45
			18.417	**19.74**
131072	2097152	-658751422203	2686832	60 09
131072	2097152	-650518421486	2618367	57 34
131072	2097152	-654566097938	2693874	60 26
131072	2097152	-643772117619	2779268	59 7
131072	2097152	-639019008560	2695121	57 94
			20.559	**59.066**

Algoritmo Tarjan

Nodos	Arcos	Suma de la distancia	No. de escaneo	Tiempo
8192	131072	-9921510130	361592	0.99
8192	131072	-10137936625	370306	1 05
8192	131072	-10468517619	383835	1 04
8192	131072	-10020203123	361604	0.99
8192	131072	-9947989615	374534	0 99
			45.212	**1.012**

Nodos	Arcos	Suma de la distancia	No. de escaneo	Tiempo
16384	262144	-28537596853	1074767	3.57
16384	262144	-28464254867	1062672	3.46
16384	262144	-28300889601	1049087	3.46
16384	262144	-29225407855	1067641	3.52
16384	262144	-28406151792	1057010	3 57
			64.834	**3.516**
32768	524288	-82329035120	3058822	12 64
32768	524288	-81658271971	3057279	12.52
32768	524288	-79099847275	2942739	11.97
32768	524288	-81299759275	3083884	12 63
32768	524288	-80793751804	3055820	12.42
			92.765	**12.436**
65536	1048576	-238256105591	8716813	42.56
65536	1048576	-227040970169	8472301	41 03
65536	1048576	-225349712932	8375539	40 64
65536	1048576	-221040047980	8436527	40.37
65536	1048576	-231299060104	8664103	41.69
			130.204	**41.258**
131072	2097152	-658751422203	24612555	133 47
131072	2097152	-650518421486	24488799	130.67
131072	2097152	-654566097938	24505683	131 77
131072	2097152	-643772117619	24028297	129 9
131072	2097152	-639019008560	24086406	129 18
			185.733	**130.998**

Algoritmo de Goldberg-Radzik

Nodos	Arcos	Suma de la distancia	No. de escaneo	Tiempo
8192	131072	-9921510130	16384	0.06
8192	131072	-10137936625	16384	0
8192	131072	-10468517619	16384	0.05
8192	131072	-10020203123	16384	0.05
8192	131072	-9947989615	16384	0
			2.000	**0.032**
16384	262144	-28537596853	32768	0.11
16384	262144	-28464254867	32768	0 11
16384	262144	-28300889601	32768	0 11
16384	262144	-29225407855	32768	0.11
16384	262144	-28406151792	32768	0 11
			2.000	**0.11**
32768	524288	-82329035120	65536	0.22
32768	524288	-81658271971	65536	0.28
32768	524288	-79099847275	65536	0 22
32768	524288	-81299759275	65536	0.22
32768	524288	-80793751804	65536	0.22
			2.000	**0.232**
65536	1048576	-238256105591	131072	0.55
65536	1048576	-227040970169	131072	0.55
65536	1048576	-225349712932	131072	0.6
65536	1048576	-221040047980	131072	0 55
65536	1048576	-231299060104	131072	0 55
			2.000	**0.56**
131072	2097152	-658751422203	262144	1.21
131072	2097152	-650518421486	262144	1.21
131072	2097152	-654566097938	262144	1.21

Nodos	Arcos	Suma de la distancia	No. de escaneo	Tiempo
131072	2097152	-643772117619	262144	1.21
131072	2097152	-639019008560	262144	1.2
			2.000	**1.208**

Algoritmo Variante Dijkstra utilizado por Goldberg y Cherkassky

Nodos	Arcos	Suma de la distancia	No. de escaneo	Tiempo
8192	131072	-9921510130	53086324	134.08
8192	131072	-10137936625	88598986	209.87
8192	131072	-10468517619	61016780	155.83
8192	131072	-10020203123	69242046	170.76
8192	131072	-9947989615	98282985	233.43
			9038.748	**180.794**

Redes tipo Acyc-P2N

Algoritmo de Dijkstra Modificado

f (%)	Nodos	Arcos	Suma de la distancia	No. de escaneo	Tiempo
0	16384	262144	218192840	16384	0.11
0	16384	262144	223263574	16384	0.11
0	16384	262144	257311340	16384	0.11
0	16384	262144	245631236	16384	0.11
0	16384	262144	273785016	16384	0.11
				1.000	**0.11**
10	16384	262144	139131014	33020	0.22
10	16384	262144	164619947	33211	0.22
10	16384	262144	188682579	33159	0.22
10	16384	262144	203081590	33501	0.22
10	16384	262144	121327442	34196	0.22
				2.040	**0.22**
20	16384	262144	-228269626	68591	0.33
20	16384	262144	-43260872	53454	0.28
20	16384	262144	-63178054	62870	0.33
20	16384	262144	-116469927	63548	0.33
20	16384	262144	-120765633	50779	0.22
				3.653	**0.298**
30	16384	262144	-1213787163	102894	0.44
30	16384	262144	-1201152423	106858	0.44
30	16384	262144	-1250284492	120266	0.55
30	16384	262144	-1310653109	109774	0.49
30	16384	262144	-1127370652	102102	0.44
				6.615	**0.472**
40	16384	262144	-6443825983	161826	0.66
40	16384	262144	-6335559288	188580	0.77
40	16384	262144	-7146978039	193235	0.82
40	16384	262144	-7147588316	192964	0.83
40	16384	262144	-6105552193	187324	0.77
				11.278	**0.77**
50	16384	262144	-37466903939	306529	1.32
50	16384	262144	-42007873744	301641	1.32
50	16384	262144	-39855704264	285441	1.21
50	16384	262144	-36559969059	299234	1.26
50	16384	262144	-36574564642	277235	1.21
				17.945	**1.264**

f (%)	Nodos	Arcos	Suma de la distancia	No. de escaneo	Tiempo
60	16384	262144	-139245453286	322116	1.37
60	16384	262144	-139108867520	367234	1.59
60	16384	262144	-136498060052	346702	1.54
60	16384	262144	-135776744072	327310	1.38
60	16384	262144	-136037074238	323109	1.38
				20.587	**1.452**
100	16384	262144	-670099761456	375114	1.59
100	16384	262144	-669918810693	363710	1.65
100	16384	262144	-673874707522	353954	1.59
100	16384	262144	-674384030267	362222	1.64
100	16384	262144	-669749094673	366410	1.64
				22.234	**1.622**

Algoritmo Simplex

f (%)	Nodos	Arcos	Suma de la distancia	No. de escaneo	Tiempo
0	16384	262144	218192840	27680	0.11
0	16384	262144	223263574	26889	0.11
0	16384	262144	257311340	26274	0.11
0	16384	262144	245631236	25697	0.11
0	16384	262144	273785016	25595	0.11
				1.613	**0.11**
10	16384	262144	139131014	34949	0.11
10	16384	262144	164619947	32797	0.11
10	16384	262144	188682579	32272	0.11
10	16384	262144	203081590	34428	0.11
10	16384	262144	121327442	36432	0.17
				2.086	**0.122**
20	16384	262144	-228269626	89279	0.27
20	16384	262144	-43260872	67865	0.22
20	16384	262144	-63178054	67348	0.22
20	16384	262144	-116469927	73259	0.22
20	16384	262144	-120765633	72909	0.22
				4.525	**0.23**
30	16384	262144	-1213787163	133286	0.49
30	16384	262144	-1201152423	136742	0.44
30	16384	262144	-1250284492	136644	0.5
30	16384	262144	-1310653109	126924	0.43
30	16384	262144	-1127370652	122706	0.44
				8.011	**0.46**
40	16384	262144	-6443825983	206966	0.88
40	16384	262144	-6335559288	206810	0.93
40	16384	262144	-7146978039	238366	0.98
40	16384	262144	-7147588316	226804	0.99
40	16384	262144	-6105552193	203017	0.82
				13.208	**0.92**
50	16384	262144	-37466903939	364177	1.98
50	16384	262144	-42007873744	355291	2.09
50	16384	262144	-39855704264	354198	1.97
50	16384	262144	-36559969059	343999	1.97
50	16384	262144	-36574564642	323164	1.86
				21.250	**1.974**
60	16384	262144	-139245453286	486795	3.02
60	16384	262144	-139108867520	511412	3.24
60	16384	262144	-136498060052	519565	3.35

f (%)	Nodos	Arcos	Suma de la distancia	No. de escaneo	Tiempo
60	16384	262144	-135776744072	507279	3 18
60	16384	262144	-136037074238	513340	3.24
				30.986	**3.206**
100	16384	262144	-670099761456	1139337	5 33
100	16384	262144	-669918810693	1124692	5.49
100	16384	262144	-673874707522	1169018	5 44
100	16384	262144	-674384030267	1170405	5.55
100	16384	262144	-669749094673	1161014	5.6
				70.367	**5.482**

Algoritmo de Tarjan

f (%)	Nodos	Arcos	Suma de la distancia	No. de escaneo	Tiempo
0	16384	262144	218192840	28171	0.05
0	16384	262144	223263574	27329	0.05
0	16384	262144	257311340	26676	0 05
0	16384	262144	245631236	26176	0 11
0	16384	262144	273785016	26061	0 06
				1.641	**0.064**
10	16384	262144	139131014	35987	0.11
10	16384	262144	164619947	33642	0.1
10	16384	262144	188682579	33288	0.11
10	16384	262144	203081590	35237	0.11
10	16384	262144	121327442	37570	0.11
				2.145	**0.108**
20	16384	262144	-228269626	108969	0.27
20	16384	262144	-43260872	77570	0.22
20	16384	262144	-63178054	77128	0.22
20	16384	262144	-116469927	90925	0.27
20	16384	262144	-120765633	86823	0.27
				5.388	**0.25**
30	16384	262144	-1213787163	234762	0.66
30	16384	262144	-1201152423	228883	0 61
30	16384	262144	-1250284492	240355	0 71
30	16384	262144	-1310653109	253322	0 71
30	16384	262144	-1127370652	224921	0 6
				14.432	**0.658**
40	16384	262144	-6443825983	694688	1.92
40	16384	262144	-6335559288	718361	2.15
40	16384	262144	-7146978039	752175	2.25
40	16384	262144	-7147588316	747807	2.37
40	16384	262144	-6105552193	663176	1.93
				43.655	**2.124**
50	16384	262144	-37466903939	2448044	8.08
50	16384	262144	-42007873744	2704437	8 9
50	16384	262144	-39855704264	2588220	8 63
50	16384	262144	-36559969059	2415011	7 63
50	16384	262144	-36574564642	2422025	7.9
				153.537	**8.228**
60	16384	262144	-139245453286	6196082	21.14
60	16384	262144	-139108867520	6144855	21 03
60	16384	262144	-136498060052	6032509	20.1
60	16384	262144	-135776744072	5988280	20 32
60	16384	262144	-136037074238	6045100	20 16
				371.177	**20.55**

f (%)	Nodos	Arcos	Suma de la distancia	No. de escaneo	Tiempo
100	16384	262144	-670099761456	11859207	46.52
100	16384	262144	-669918810693	11878075	46.08
100	16384	262144	-673874707522	11846854	46 41
100	16384	262144	-674384030267	11868617	46 14
100	16384	262144	-669749094673	11811569	46.41
				723.441	**46.312**

Algoritmo de Goldberg-Radzik

f (%)	Nodos	Arcos	Suma de la distancia	No. de escaneo	Tiempo
0	16384	262144	218192840	47626	0.16
0	16384	262144	223263574	43164	0 11
0	16384	262144	257311340	41762	0 11
0	16384	262144	245631236	40776	0.11
0	16384	262144	273785016	40188	0.11
				2.606	**0.12**
10	16384	262144	139131014	49214	0 16
10	16384	262144	164619947	49118	0 17
10	16384	262144	188682579	46496	0 11
10	16384	262144	203081590	48876	0.11
10	16384	262144	121327442	53242	0 17
				3.014	**0.144**
20	16384	262144	-228269626	126932	0 28
20	16384	262144	-43260872	88956	0 22
20	16384	262144	-63178054	88358	0 22
20	16384	262144	-116469927	112562	0.28
20	16384	262144	-120765633	98762	0 22
				6.294	**0.244**
30	16384	262144	-1213787163	207952	0.5
30	16384	262144	-1201152423	174536	0.38
30	16384	262144	-1250284492	237414	0 55
30	16384	262144	-1310653109	187700	0.44
30	16384	262144	-1127370652	215230	0.49
				12.486	**0.472**
40	16384	262144	-6443825983	353574	0 82
40	16384	262144	-6335559288	363598	0.88
40	16384	262144	-7146978039	411260	1.04
40	16384	262144	-7147588316	353442	0 83
40	16384	262144	-6105552193	386730	0.93
				22.810	**0.9**
50	16384	262144	-37466903939	433640	1.15
50	16384	262144	-42007873744	480410	1.2
50	16384	262144	-39855704264	545630	1 38
50	16384	262144	-36559969059	505124	1 32
50	16384	262144	-36574564642	559118	1 48
				30.810	**1.306**
60	16384	262144	-139245453286	539764	1 49
60	16384	262144	-139108867520	540246	1 48
60	16384	262144	-136498060052	537348	1 42
60	16384	262144	-135776744072	569884	1.59
60	16384	262144	-136037074238	558686	1 54
				33.520	**1.504**
100	16384	262144	-670099761456	32768	0 05
100	16384	262144	-669918810693	32768	0 11
100	16384	262144	-673874707522	32768	0 11
100	16384	262144	-674384030267	32768	0.11
100	16384	262144	-669749094673	32768	0.11
				2.000	**0.098**

Algoritmo Variante Dijkstra utilizado por Goldberg y Cherkassky

f (%)	*Nodos*	*Arcos*	*Suma de la distancia*	*No. de escaneo*	*Tiempo*
0	16384	262144	218192840	16384	0.11
0	16384	262144	223263574	16384	0 11
0	16384	262144	257311340	16384	0.11
0	16384	262144	245631236	16384	0 11
0	16384	262144	273785016	16384	0 11
				1.000	**0 11**
10	16384	262144	139131014	18155	0.11
10	16384	262144	164619947	18448	0.11
10	16384	262144	188682579	18050	0.11
10	16384	262144	203081590	18785	0 11
10	16384	262144	121327442	19959	0.11
				1.140	**0.11**
20	16384	262144	-228269626	139633	0 55
20	16384	262144	-43260872	78738	0 28
20	16384	262144	-63178054	114952	0 44
20	16384	262144	-116469927	127186	0 5
20	16384	262144	-120765633	78695	0.27
				6.582	**0.408**
30	16384	262144	-1213787163	1169932	3.73
30	16384	262144	-1201152423	789065	2 53
30	16384	262144	-1250284492	1216897	3 95
30	16384	262144	-1310653109	1127636	3.68
30	16384	262144	-1127370652	902957	2 8
				63.556	**3.338**
40	16384	262144	-6443825983	9211921	27 3
40	16384	262144	-6335559288	24598222	73 27
40	16384	262144	-7146978039	17669090	51 91
40	16384	262144	-7147588316	7680474	23 24
40	16384	262144	-6105552193	21018895	60 36
				978.743	**47.216**

REFERENCIAS

Ahuja Ravindra K., Magnanti T.L. y Orlin J.B., "Networks flows", Theory, algorithms and applications. (1991).

Bellman R.E., "On a routing problem", *Quart. Appl. Math.* 16 p. 87-90 (1958).

Cherkassky B.V., Goldberg A.V. y Radzik T., "Shortes path algorithm: Theory and experimental evaluation". *Math. Program. 73*, p. 129-174 (1996).

Cherkassky B.V., Goldberg A.V., "Negative-cycle detection algorithms" *Math. Program 85*, p. 277-311 (1999).

Cherkassky B.V., Goldberg A.V. y Silverstein C., "Buckets, heaps, lists and monotone priority queues", *SIAM J. Comput*, 28, 4, p. 1326 – 1346 (1999).

Dantzig, G.B., "Application of the Simplex Method of Transportation Problem. In: Koopmans, T.C., ed., *Activity Analysis and Production and Allocation*, p. 359-373. Wiley, New York (1951).

Denardo W.V. y Fox B.L, "Shortest-route methods: 1. Reaching, pruning, and buckets", *Operation Research* 27 p. 161-186 (1979).

Dijkstra E.W., "A note on two problems in connection with graphs" *Numer. Math.* 1, p. 269 - 271 (1959).

Elmaghraby S.E., "Activity Networks: Project planning and control by Network models" WileyInterscience, New York (1978).

Ford Jr. L.R., Network Flow Theory, *Paper P-923*, RAND Corp., Santa Monica, CA. (1956).

Ford Jr. L.R. y Fulkerson D.R., *Flows in networks*, Princeton Univ. Press. Princeton, NJ, (1962).

Glover F. y Klingman D., Network applications in industry and goverment, *AIIE Transactions 9*, p. 363 – 376 (1977).

Goldberg A.V.,Radzik T., "A heuristic improvement of the Bellman-Ford Algorithm" *Applied Math. Lett. 6*, p. 3-6 (1993)

Hillier F.S. y Lieberman G.J., Introducción a la Investigación de Operaciones, Mc Graw Hill sexta edición, p. 303 –404 (1997).

Minieka, E. Optimization Algorithms for Networks and Graphs, Marcel Dekker, Inc., New York 42 (1978).

Moore, E.F., The Shortest Path Through a Maze. In: *Proceedings of the Int. Symp. on the Theory of Switching*. (Harvard University Press , 1959) p. 285-292.

Tarjan, R.E., Shortest Paths. Technical report, *AT&T Bell Laboratories*, Murray Hill, NJ (1981).

Tarjan, R.E., Data Structures and Network Algorithms, *Bell Laboratiores*, Murray Hill, NJ (1983).

T.C. Hu, Programming and Network Flows, p. 168 – 173 (1969).

Waterman M.S., Mathematical Methods for DNA Sequences, CRC Press. Boca Raton. Fl. (1978).

Zawack D.J. y Thompson G.L., A dynamic space-time network flow model for city traffic congestion, *Transportation Science 21,* p. 153 – 162 (1987).