Capítulo 4 Simulación de Algoritmos de Ruteo

En este capítulo se muestran ejemplos de topologías simuladas con el *Simulador de Algoritmos Dijkstra y Bellman-Ford.* Se analiza cada paso de ambos algoritmos para una topología de seis nodos. Así podemos comprobar que por ambos algoritmos llegamos a la misma solución.

4.1 Introducción

El *Simulador de Algoritmos Dijkstra y Bellman-Ford* tiene la capacidad de simular ambos algoritmos en una topología generada por el usuario y topologías previamente establecidas. A continuación se presentan dos topologías generadas en este modo y se dará la explicación pertinente de cada algoritmo.

En este capítulo se muestran las simulaciones obtenidas con el *Simulador de algoritmos Dijkstra y Bellman-Ford V1.0*. Se tomará como ejemplo la simulación de ambos algoritmos en su *Topología 1,* que consta de seis *routers* enlazados como se muestra en la Figura 4.1. Las flechas representan a los enlaces entre cada nodo. Los números representan el *peso* del enlace entre cada nodo.

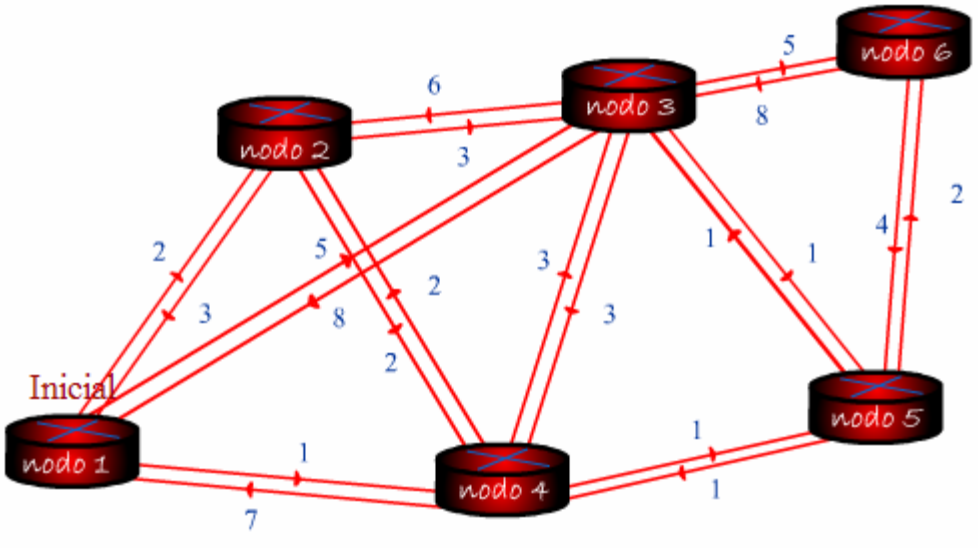

Figura 4.1 Topología 1

En el simulador es posible cambiar el *peso* del enlace. Recordemos que para ello basta seleccionar el enlace que se desee cambiar y asignarle el nuevo valor.

4.2 Simulación de Algoritmo de *Dijkstra*

A continuación se presenta el análisis del algoritmo *Dijkstra* de la manera en la que el simulador lo hace en su modalidad *paso a paso*. Se hace el análisis para una red de seis nodos enlazados como se muestra en la Figura 4.1, topología 1.

4.2.1 Primera Iteración:

El nodo inicial (nodo 1) procede a medir el *peso* de los nodos vecinos. Al presionar el botón *paso a paso* se visualiza en los enlaces que unen a los nodos vecinos con el nodo inicial una serie de paquetes que son enviados. Los nodos vecinos al nodo inicial cambian de color y muestran en la parte superior el *peso* del enlace como se muestra en la Figura 4.2

Al mismo tiempo en el primer renglón de la tabla observamos como los campos han sido llenados con la información que se muestra en la Tabla 4.1.

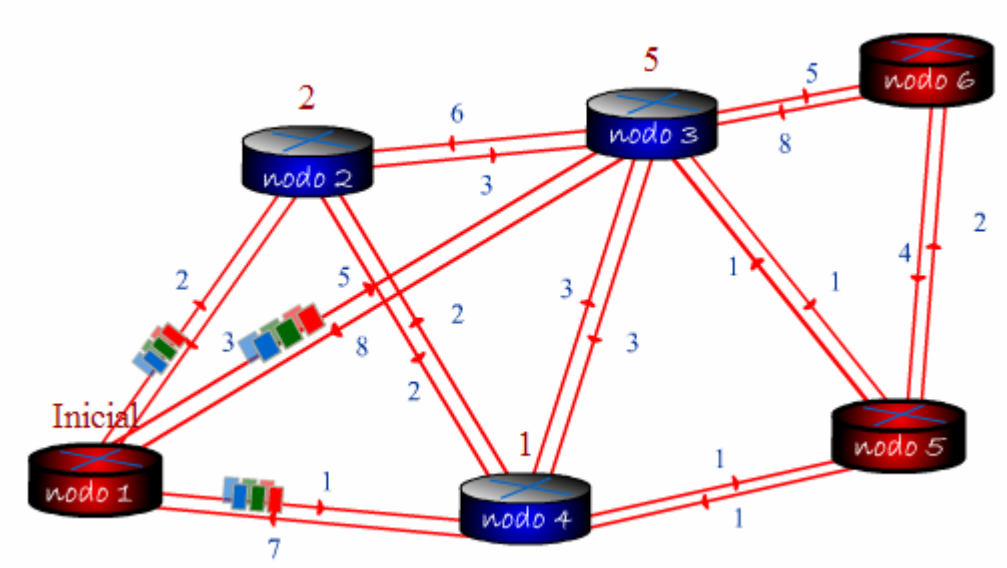

Figura 4.2 Iteración 1 Dijkstra

Tabla 4.1 Iteración 1 Dijkstra

Iteracion	Nodes Calculated	Node2	Path	Node3	Path	Node4	Path	Node5	Path	Node6	Path
1	1	2	1,2	5	1,3	1	1,4	99		99	

En la tabla podemos observar bajo la columna con el respectivo nombre de cada nodo el *peso del camino* (el peso de la ruta) desde el nodo 1 hasta cada nodo correspondiente. En la columna *camino* se despliega el camino (o ruta) más corta desde el nodo 1 hasta el nodo correspondiente. En la columna *Nodos Calculados* obtendremos los nodos que ya tienen un *peso* y un camino establecido.

4.2.2 Segunda Iteración

El algoritmo elige al nodo vecino con menor peso y lo agrega a la lista de *Nodos Calculados*. En este caso en particular, ha elegido al *nodo 4* con un *peso* de *1* como se muestra en la Figura 4.3

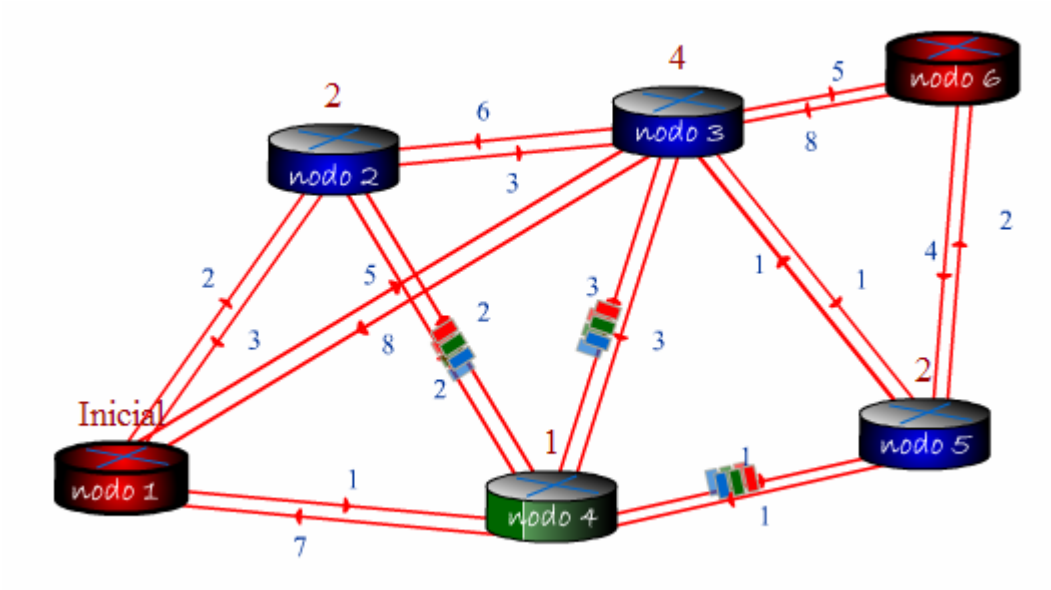

Figura 4.3 Iteración 2 Dijkstra

Ahora establecemos al *nodo 4* como *nodo de partida* o "inicial". Y procedemos a medir el *peso* hacia los nodos vecinos al *nodo 4*. Para representar esto en el Tutorial el *nodo 4* cambia de color a verde y manda *paquetes* a sus nodos vecinos. El algoritmo procede a analizar si el *peso parcial* de los nodos que se encuentran en azul (los nodos vecinos) es menor a la sumatoria del *peso final* del *nodo 4* más el *peso del enlace* entre el *nodo 4* y

los respectivos nodos vecinos. En caso de que sea menor le asigna este nuevo valor. En caso de que dicha sumatoria exceda el valor del *peso parcial* entonces deja este valor. Arriba de cada nodo azul podemos apreciar el valor del *peso parcial* a cada nodo. Estos valores también se registran en la tabla como se muestra en la Tabla 4.2.

Tabla 4.2 Iteración 2 Dijkstra

Iteracion	Nodes Calculated	Node2	Path	Node3	Path	Node4	Path	Node5	Path	Node6	Path
1	1	2	1,2	5	1,3	1	1,4	99		99	
2	1,4	2	1,2	4	1,4,3	1	1,4	2	1,4,5	99	-

En la columna *Nodos Calculados* podemos ver que el *nodo 4* ha sido registrado. En otras palabras podemos ahora afirmar que el camino más corto del *nodo1* al *nodo4* ya ha sido determinado. Y el *peso* de este camino lo podemos ver directamente arriba del *nodo4* que se encuentra en verde o en la tabla en su respectiva columna. El camino por el cuál se obtiene la ruta más corta se muestra en la columna *camino*.

4.2.3 Tercera Iteración

El algoritmo procede a elegir al nodo con menor *peso* que todavía no este definido como *nodo establecido* (que no se encuentre en la columna *nodos calculados).* En este caso en particular podemos observar que dos nodos tienen el mismo peso (*nodo 2 y nodo5)*. El algoritmo elegirá al nodo más pequeño *nodo2* en este caso y como en el paso anterior procede a pintar el nodo de color verde.

Se procede medir los *pesos* de los nodos vecinos que no tengan un *peso final*. En este caso el *nodo3* es su único vecino que no tiene un peso establecido. Como podemos observar de la Figura 4.4 la suma del *peso del nodo2* más la suma del enlace entre el *nodo2* y el *nodo4* es superior al *peso parcial* que el *nodo4* ya tiene. Entonces no cambiará el *peso* del *nodo4.*

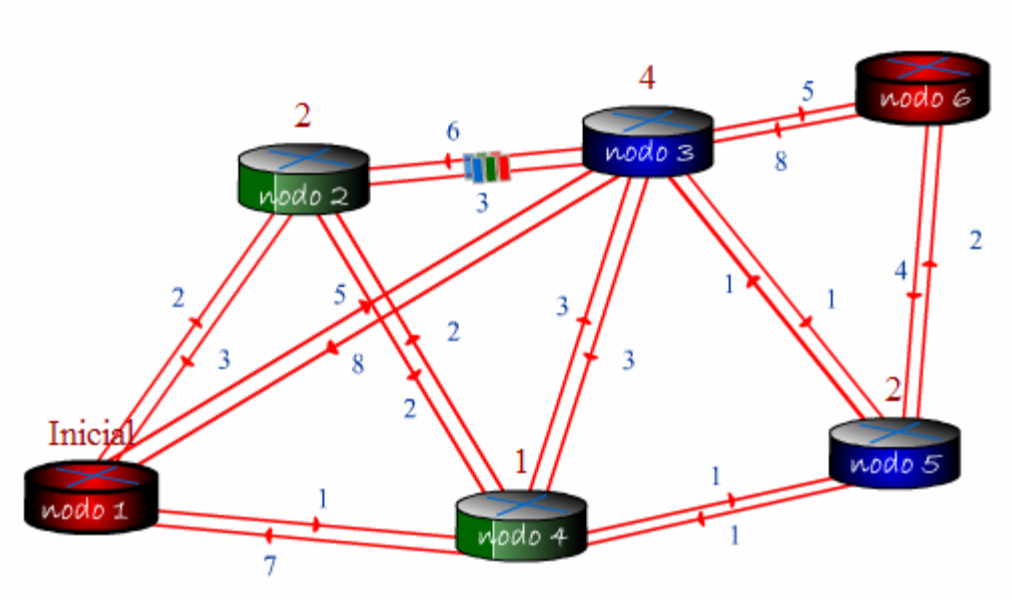

Figura 4.4 Iteración 3 Dijkstra

La tabla generada se muestra en la Tabla 4.3. Como podemos observar el *nodo2* ha sido agregado a la columna de *Nodos Calculados*. Se afirma ahora que el camino más corto al *nodo2* es a través del *nodo1(inicial)* como se muestra en la tabla.

Tabla 4.3 Tabla Iteración 3 Dijkstra

Iteracion	Nodes Calculated	Node2	Path	Node3	Path	Node4	Path	Node5	Path	Node6	Path
1	1	2	1,2	5	1,3	1	1,4	99		99	
2	1,4	2	1,2	4	1,4,3	1	1,4	2	1,4,5	99	-
3	1,4,2	2	1,2	4	1,4,3	1	1,4	2	1,4,5	99	-

4.2.4 Cuarta Iteración

Entre el *nodo5* y el *nodo3* el algoritmo elegirá al *nodo5* como y le asignará un *peso final* pues es el que presenta el menor *peso parcial* de los dos nodos. Procede a medir el *peso* hacia sus nodos vecinos que no tengan un *peso final* definido. En este caso en particular al *nodo6* y al *nodo3*. En este caso la sumatoria del *peso del canal* más el *peso final* del *nodo5* es menor al *peso parcial* contenidos en el *nodo3*. Por lo tanto cambiará su *peso parcial* por el resultado de esta sumatoria.

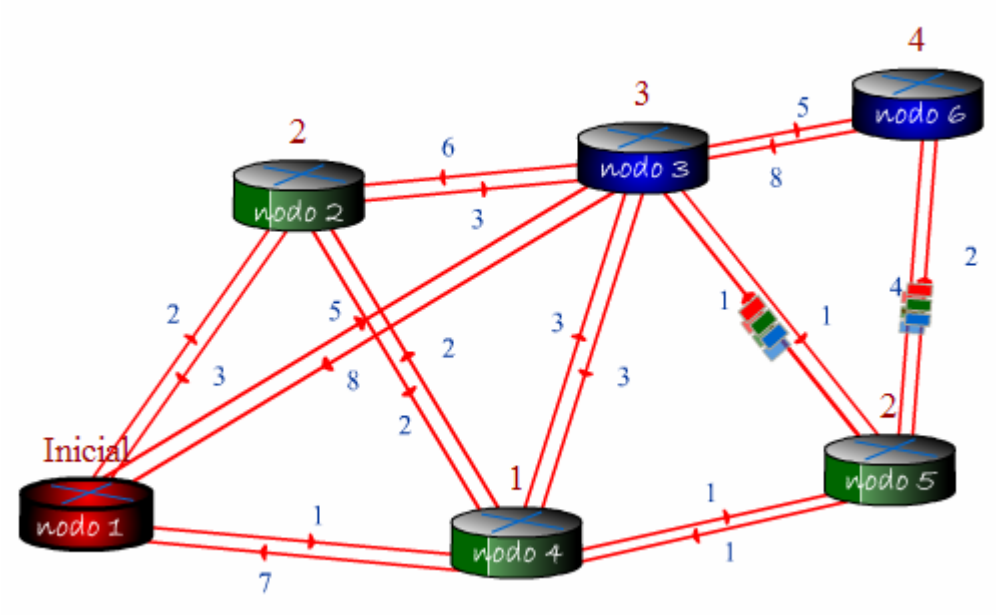

Figura 4.5 Iteración 4 Dijkstra

La tabla ahora se encuentra como lo muestra la Tabla 4.4.

Tabla 4.4 Tabla Iteración 4 Dijkstra

Iteracion	Nodes Calculated	Node2	Path	Node3	Path	Node4	Path	Node5	Path	Node6	Path
1	1	2	1,2	5	1,3	1	1,4	99		99	
2	1,4	2	1,2	4	1,4,3	1	1,4	2	1,4,5	99	-
3	1,4,2	2	1,2	4	1,4,3	1	1,4	2	1,4,5	99	-
4	1,4,2,5	2	1,2	3	1,4,5,3	1	1,4	2	1,4,5	4	1,4,5,6

Con la tabla ahora podemos ver que el *nodo5* ha sido agregado a los *Nodos Calculados,* es ahora que podemos afirmar que el camino más corto desde el *nodo1* al *nodo5* es conocido. La ruta más corta en este caso es a través de los nodos *1 y 4* con un *peso final* de 2.

4.2.5 Quinta Iteración

El algoritmo elige al *nodo3* pues es el que tiene el menor *peso parcial.* Y lo agrega a la lista de *Nodos Calculados.* A estas alturas solamente el *nodo6* no tiene un *peso final* por lo cual se mide el *peso* hacia el *nodo6.* Esta iteración se muestra en la Figura 4.6

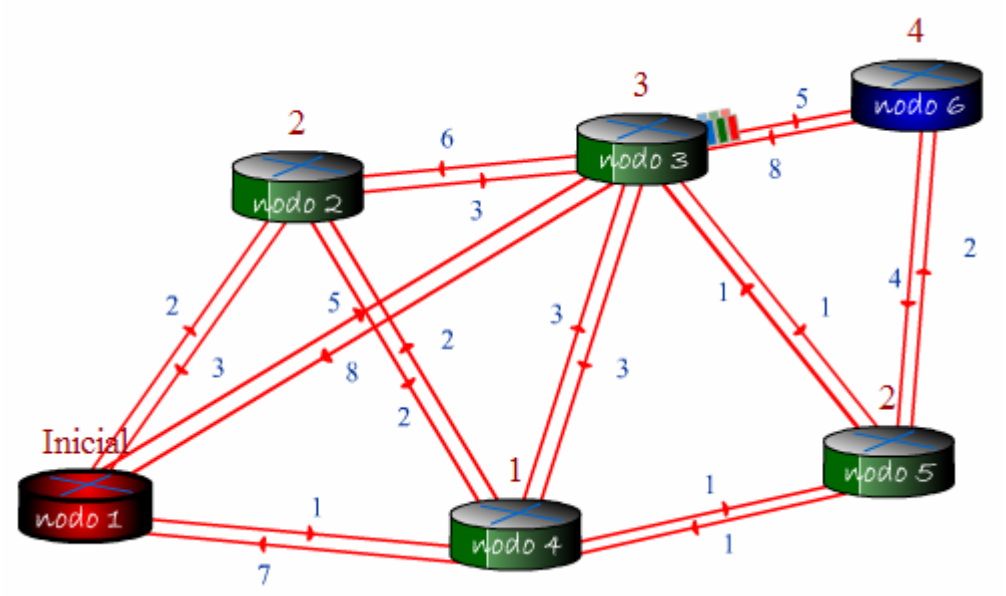

Figura 4.6 Iteración 5 Dijkstra

Finalmente los datos obtenidos por la quinta iteración se muestran en la Tabla 4.5

Tabla 4.5 Tabla Iteración 5 Dijkstra

Iteracion	Nodes Calculated	Node2	Path	Node3	Path	Node4	Path	Node5	Path	Node6	Path
1	1	2	1,2	5	1,3	1	1,4	99		99	
2	1,4	2	1,2	4	1,4,3	1	1,4	2	1,4,5	99	-
3	1,4,2	2	1,2	4	1,4,3	1	1,4	2	1,4,5	99	-
4	1,4,2,5	2	1,2	3	1,4,5,3	1	1,4	2	1,4,5	4	1,4,5,6
5	1,4,2,5,3	2	1,2	3	1,4,5,3	1	1,4	2	1,4,5	4	1,4,5,6

4.3 Simulación de Algoritmo de *Bellman-Ford*

A continuación se presenta el análisis del algoritmo *Bellman-Ford* de la manera en la que el simulador lo hace en su modalidad *paso a paso*. Cada iteración equivale a presionar una vez el botón de *paso a paso* en el simulado.

Se realizará el análisis para una red de seis nodos enlazados como se muestra en la Figura 4.1, esta misma topología fue analizada en la sección anterior para el algoritmo *Dijkstra* así logra comprobar que por ambos algoritmos llegamos a la misma conclusión. Los datos que presentan las tablas generadas por el simulador corresponden al vector distancia del nodo *Start*.

4.3.1 Primera Iteración

Comencemos con la primera iteración del algoritmo *Bellman-Ford*, en esta iteración se realizará la inicialización del algoritmo. Recordemos que el algoritmo establece como nodo inicial al nodo *Start*. La inicialización del algoritmo, como se describió en el capítulo 2 es la siguiente:

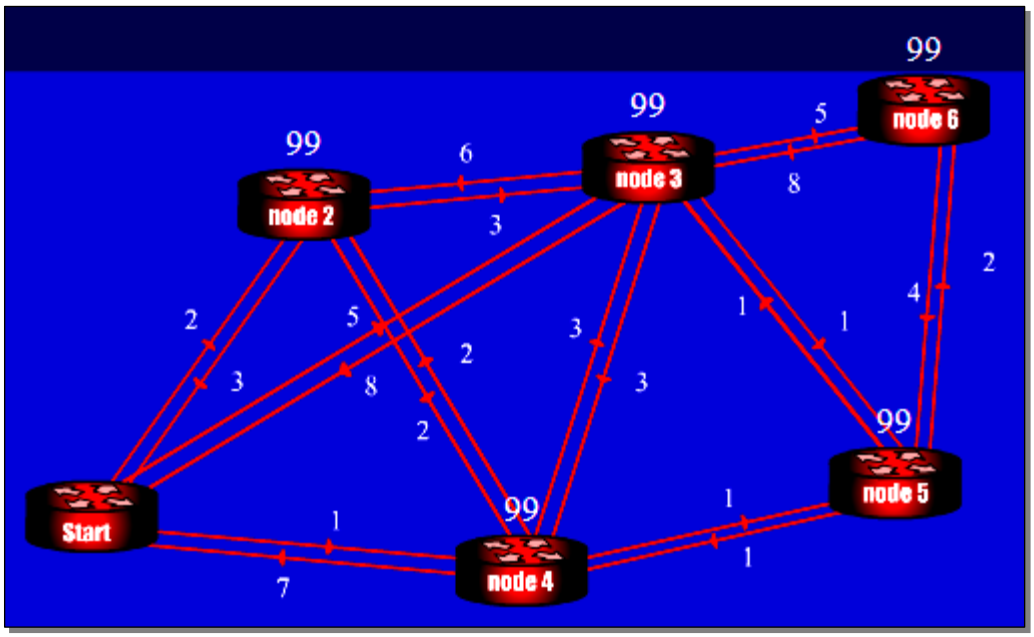

Figura 4.7 Primera Iteración de Bellman-Ford

Para todos los nodos (sin tomar en cuenta a *Start*) la distancia del nodo *Start* a cada uno de los nodos es infinita. Esto se representa con el escalar 99. La distancia de l nodo *Start* hacia sí mismo es cero

Tabla. 4.6 Tabla de la Primera Iteración de Bellman-Ford

h	Nodo2	Camino	Nodo3	Camino	Nodo4	Camino	Nodo5	Camino	Nodo6	Camino
0	99	--	99	--	99	--	99	--	99	--

La inicialización del algoritmo *Bellman-Ford* la podemos visualizar en la Figura 4.7. La tabla correspondiente que se genera automáticamente al presionar el botón de *paso a paso* es la que se muestra en la Tabla 4.6. Recordemos que esta tabla corresponde al vector distancia del nodo *Start.*

4.3.2 Segunda Iteración

Una vez realizada la inicialización del algoritmo, este mismo procede a actualizar su vector distancia. Para ello recordemos la ecuación de *Bellman-Ford* presentada en el capítulo 2 (ecuación 2.1).

$$d_x(y) = \min_v [c(x,v) + d_v(y)]$$

Donde $d_x(y)$ es el costo del camino con menor coste del nodo x al nodo y

Donde $c(x,v)$ es el coste del predecesor hacia el nodo *x.*

Donde $d_v(y)$ es el coste del predecesor hacia el nodo *Start.*

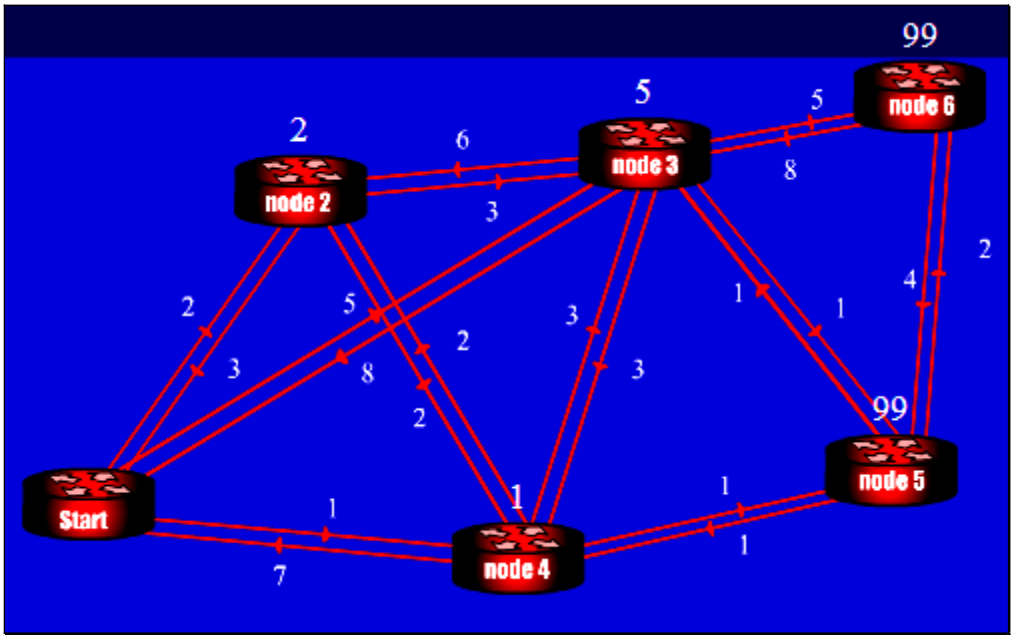

Figura 4.8 Segunda Iteración de Bellman-Ford

Analicemos ahora la ecuación de *Bellman-Ford* para encontrar el camino hacia el *Nodo2.* Para ello, tenemos que analizar quién es vecino directo del *Nodo 2*. En este caso los nodos vecinos son: *Nodo Start, Nodo 3,* y *Nodo 4.* Ahora vamos a sustituir los valores de la red en la ecuación de *Bellman-Ford.*

$$d_x(y) = \min_v [c(x,v) + d_v(y)] \; = \min_v [0+2,\; \infty+5,\; \infty+1] = \min_v [2,\; \infty,\; \infty] = 2$$

Así el algoritmo de *Bellman-Ford* continúa evaluando la ecuación para cada uno de los nodos. Veamos en la siguiente tabla un análisis similar al que acabamos de realizar.

Tabla 4.7 Desglose de la segunda iteración

Para el Nodo 3
Nodos vecinos: Nodo Start, Nodo2, Nodo4, Nodo5, Nodo6 $d_x(y) = \min_v[c(x,v) + d_v(y)] = \min_v[0+5, \infty+6, \infty+3, \infty+1, \infty+8] = 5$
Para el Nodo 4
Nodos vecinos: Nodo Start, Nodo2, Nodo3, Nodo5 $d_x(y) = \min_v[c(x,v) + d_v(y)] = \min_v[0+1, \infty+2, \infty+3, \infty+1] = \min_v[1, \infty, \infty, \infty] = 1$
Para el Nodo 5
Nodos vecinos: Nodo3, Nodo4, Nodo6 $d_x(y) = \min_v[c(x,v) + d_v(y)] = \min_v[\infty+1, \infty+1, \infty+4] = \min_v[\infty, \infty, \infty] = \infty$
Para el Nodo 6
Nodos vecinos: Nodo3, Nodo5 $d_x(y) = \min_v[c(x,v) + d_v(y)] = \min_v[\infty+5, \infty+2] = \min_v[\infty, \infty] = \infty$

Como podemos apreciar, los resultados obtenidos de la ecuación *Bellman-Ford* mostrados en la Tabla 4.8 corresponden a los valores mostrados por la Figura 4.8 obtenida del simulador de algoritmos.

Tabla 4.8 Segunda Iteración de Bellman-Ford

h	Nodo2	Camino	Nodo3	Camino	Nodo4	Camino	Nodo5	Camino	Nodo6	Camino
0	99	--	99	--	99	--	99	--	99	--
1	2	1,2	5	1,3	1	1,4	99	-,-	99	-,-

4.3.3 Tercera Iteración

Una vez actualizada la tabla por la iteración dos, el algoritmo procede de manera idéntica para actualizar los valores de las tablas. Una vez más recordemos que algoritmo tiene su funcionamiento basado en la actualización y distribución de las tablas de los nodos. Veamos la tabla 4.017 que es similar a la que realizamos en la segunda iteración, en esta tabla se muestran los nodos vecinos directos de cada nodo y su correspondiente ecuación de *Bellman-Ford.*

Tabla 4.9 Tabla Desglosada de la Tercera Iteración

Para el Nodo 2
Nodos vecinos: Nodo Start, Nodo3, Nodo4 $d_x(y) = \min_v[c(x,v) + d_v(y)] = \min_v[0+2,\ 5+6,\ 1+2] = 2$
Para el Nodo 3
Nodos vecinos: Nodo Start, Nodo2, Nodo4, Nodo5,   ***Nodo6*** $d_x(y) = \min_v[c(x,v) + d_v(y)] = \min_v[0+5,\ 2+6,\ 1+3,\ \infty+1,\ \infty+8] = \min_v[5,8,4,\infty,\infty] = 4$
Para el Nodo 4
Nodos vecinos: Nodo Start, Nodo2, Nodo3, Nodo5 $d_x(y) = \min_v[c(x,v) + d_v(y)] = \min_v[0+1,\ 2+2,\ 5+3,\ \infty+1] = \min_v[1,\ 4,\ 8,\infty] = 1$
Para el Nodo 5
Nodos vecinos: Nodo3, Nodo4, Nodo6 $d_x(y) = \min_v[c(x,v) + d_v(y)] = \min_v[1+1,\ 5+1,\ \infty+4] = \min_v[2,\ 6,\ \infty] = 2$
Para el Nodo 6
Nodos vecinos: Nodo3, Nodo5 $d_x(y) = \min_v[c(x,v) + d_v(y)] = \min_v[5+5,\ \infty+2] = \min_v[5,\ \infty] = 10$

Una vez realizadas las actualizaciones en la iteración tres, los nodos se presentan como se muestra en la Tabla 4.12. Los resultados obtenidos por el simulador se muestran en la Tabla 4.12. Una vez más si son comparados con los resultados obtenidos de la tabla desglosada (Tabla 4.7) podemos observar que los resultados son actualizados por el algoritmo cada vez que se corre el programa, esto es, cada vez que los nodos intercambian la información de sus tablas.

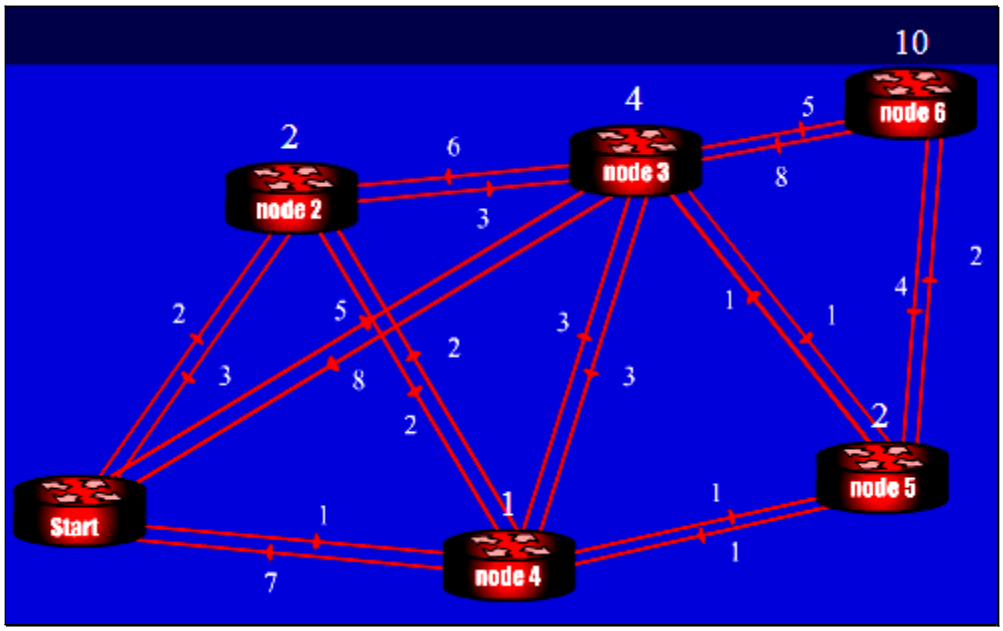

Figura 4.9 Tercera Iteración de Bellman-Ford

Si analizamos los datos detalladamente podemos notar que tanto el *Nodo 2* como el *Nodo 4* no cambiaron con respecto a los valores obtenidos anteriormente. Esto es porque en esta iteración no se encontró un camino que superara la distancia ya calculada anteriormente. En caso contrario, para el *Nodo 5* se encontró un camino más corto, note que anteriormente el camino hasta el *Nodo 5* era a través de los nodos 1 y 3. Sin embargo el algoritmo encontró que si primero hacemos un salto al *Nodo 4* y posteriormente un salo al *Nodo 3* el camino es menor. Para los nodos 5 y 6 se encontró finalmente un camino, pues anteriormente permanecía a una distancia infinita del *Nodo Start*. Esto es porque los nodos 5 y 6 se encuentran a una distancia de tres saltos del *Nodo Start.* Como posiblemente habrá supuesto el lector, el algoritmo necesita de *n* iteraciones para encontrar un camino, no necesariamente el menor, hacia un nodo que se encuentra a *n* saltos del nodo inicial.

Tabla 4.10 Tabla de la Tercera Iteración de Bellman-Ford

h	Nodo2	Camino	Nodo3	Camino	Nodo4	Camino	Nodo5	Camino	Nodo6	Camino
0	99	--	99	--	99	--	99	--	99	--
1	2	1,2	5	1,3	1	1,4	99	-,-	99	-,-
2	2	1,2	4	1,4,3	1	1,4	2	1,4,5	10	1,3,6

4.3.4 Cuarta Iteración

Continuando con el algoritmo, analicemos lo acontecido en la cuarta iteración. Los datos desglosados se muestran a continuación en la tabla 4.020.

Tabla 4.11 Tabla de Datos Desglosados de la Cuarta Iteración

Para el Nodo 2
Nodos vecinos: Nodo Start, Nodo3, Nodo4 $d_x(y) = \min_v[c(x,v) + d_v(y)] = \min_v[0+2,\ 4+6,\ 1+2] = 2$
Para el Nodo 3
Nodos vecinos: Nodo Start, Nodo2, Nodo4, Nodo5, Nodo6 $d_x(y) = \min_v[c(x,v) + d_v(y)] = \min_v[0+5,\ 2+6,\ 1+3,\ 2+1,\ 10+8] = \min_v[5,8,4,3,18] = 4$
Para el Nodo 4
Nodos vecinos: Nodo Start, Nodo2, Nodo3, Nodo5 $d_x(y) = \min_v[c(x,v) + d_v(y)] = \min_v[0+1,\ 2+2,\ 4+3,\ 2+1] = \min_v[1,\ 4,\ 7,3] = 1$
Para el Nodo 5
Nodos vecinos: Nodo3, Nodo4, Nodo6 $d_x(y) = \min_v[c(x,v) + d_v(y)] = \min_v[1+1,\ 4+1,\ 10+4] = \min_v[2,\ 5,\ 14] = 2$
Para el Nodo 6
Nodos vecinos: Nodo3, Nodo5 $d_x(y) = \min_v[c(x,v) + d_v(y)] = \min_v[4+5,\ 2+2] = \min_v[5,\ 4] = 4$

En esta iteración como pudimos observar solamente se encontraron dos caminos más cortos. Estos caminos son los correspondientes al *Nodo 3* y al *Nodo 6*. Para el *Nodo 3* se encontró una ruta más corta a través del *Nodo 5*, anteriormente el camino era directamente desde el *Nodo 4,* sin embargo a través del *Nodo 5* obtenemos un coste de camino una unidad menor a la obtenida anteriormente.

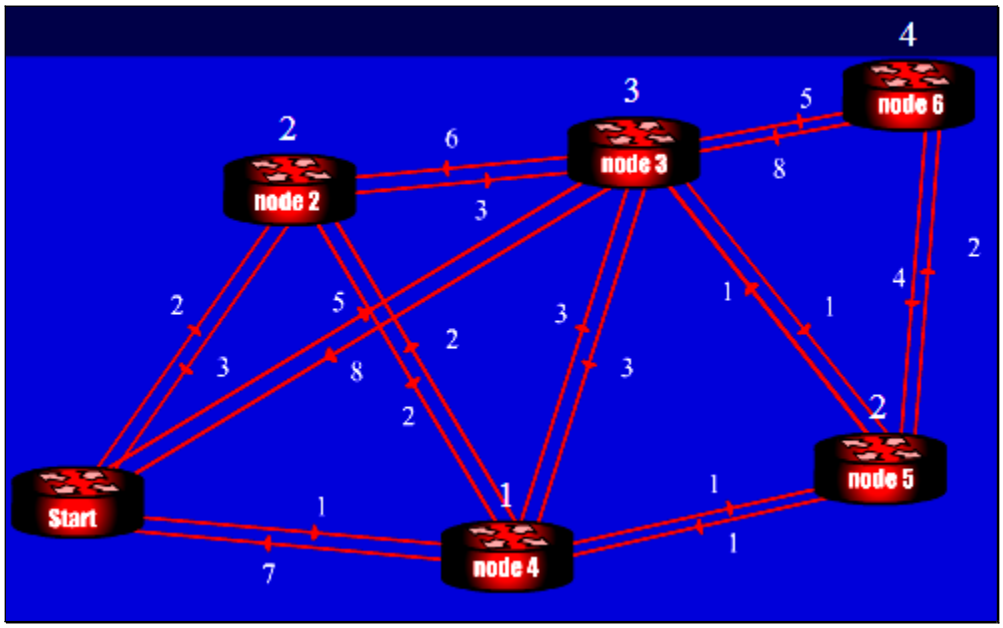

Figura 4.10 Cuarta Iteración de Bellman-Ford

En cuanto a lo referente al *Nodo 6*, el coste del camino se redujo significativamente al encontrar una ruta más corta a través de un número mayor de saltos. Anteriormente para la ruta más corta al *Nodo 6* era a través de los Nodos *1* y *3*. En esta iteración se encontró que si en vez de hacer dos saltos se hacen tres saltos se obtiene un menor coste, así comprobamos que el número de saltos no tiene relevancia en encontrar el camino de menor coste.

Tabla 4.12 Tabla de la Cuarta Iteración de Bellman-Ford

h	Nodo2	Camino	Nodo3	Camino	Nodo4	Camino	Nodo5	Camino	Nodo6	Camino
0	99	--	99	--	99	--	99	--	99	--
1	2	1,2	5	1,3	1	1,4	99	-,-	99	-,-
2	2	1,2	4	1,4,3	1	1,4	2	1,4,5	10	1,3,6
3	2	1,2	3	1,4,5,3	1	1,4	2	1,4,5	4	1,4,5,6

Como en ocasiones anteriores se puede comprobar el resultado de los análisis obtenidos en la Tabla 4.11 con los obtenidos por el simulador mostrados en la Tabla 4.12.

4.3.5 Quinta Iteración

Esta es la quinta y última iteración. Veamos que la tabla mostrada en la Tabla 4.13 es igual a la tabla mostrada en la Tabla 4.12 para la cuarta iteración.

Tabla 4.13 Tabla Desglosada de la Quitna Iteración

Para el Nodo 2
Nodos vecinos: Nodo Start, Nodo3, Nodo4 $d_x(y) = \min_v[c(x,v) + d_v(y)] = \min_v[0+2,\ 4+6,\ 1+2] = 2$
Para el Nodo 3
Nodos vecinos: Nodo Start, Nodo2, Nodo4, Nodo5, Nodo6 $d_x(y) = \min_v[c(x,v) + d_v(y)] = \min_v[0+5,\ 2+6,\ 1+3,\ 2+1,\ 10+8] = \min_v[5,8,4,3,18] = 4$
Para el Nodo 4
Nodos vecinos: Nodo Start, Nodo2, Nodo3, Nodo5 $d_x(y) = \min_v[c(x,v) + d_v(y)] = \min_v[0+1,\ 2+2,\ 4+3,\ 2+1] = \min_v[1,\ 4,\ 7,3] = 1$
Para el Nodo 5
Nodos vecinos: Nodo3, Nodo4, Nodo6 $d_x(y) = \min_v[c(x,v) + d_v(y)] = \min_v[1+1,\ 4+1,\ 10+4] = \min_v[2,\ 5,\ 14] = 2$
Para el Nodo 6
Nodos vecinos: Nodo3, Nodo5 $d_x(y) = \min_v[c(x,v) + d_v(y)] = \min_v[4+5,\ 2+2] = \min_v[5,\ 4] = 4$

Tabla 4.14 Tabla de la Quinta Iteración de Bellman-Ford

h	Nodo2	Camino	Nodo3	Camino	Nodo4	Camino	Nodo5	Camino	Nodo6	Camino
0	99	--	99	--	99	--	99	--	99	--
1	2	1,2	5	1,3	1	1,4	99	-,-	99	-,-
2	2	1,2	4	1,4,3	1	1,4	2	1,4,5	10	1,3,6
3	2	1,2	3	1,4,5,3	1	1,4	2	1,4,5	4	1,4,5,6
4	2	1,2	3	1,4,5,3	1	1,4	2	1,4,5	4	1,4,5,6

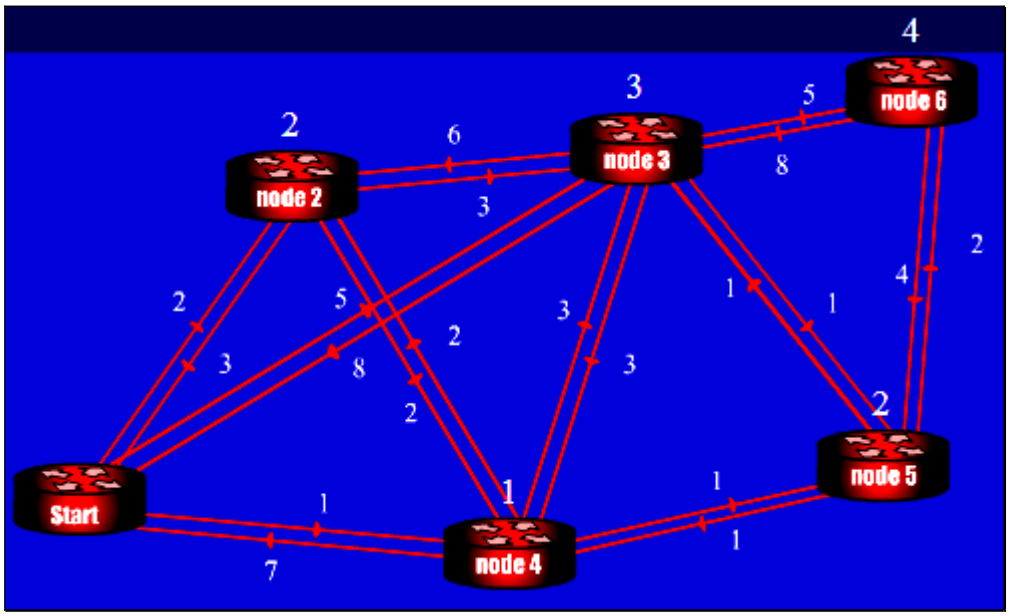

Figura 4.11 Quinta Iteración de Bellman-Ford

Los datos son exactamente iguales a los obtenidos en la cuarta iteración porque el algoritmo ha concluido con los cálculos del camino con menor coste. Estos datos son desplegados en la Tabla 4.14. Damos así por concluido el algoritmo de *Bellman-Ford*. Como se pude apreciar de tablas de las últimas iteraciones de ambos algoritmos se ha llegado a los mismos resultados tanto por el algoritmo de *Dijkstra* que por el algoritmo de *Bellman-Ford*.