UNIVERSIDAD AUTONOMA DE NUEVO LEON
FACULTAD DE INGENIERIA MECANICA
Y ELECTRICA
DIVISION DE ESTUDIOS DE POSTGRADO

UN SISTEMA PARA LA DETECCION Y ELIMINACION DE INFACTIBILIDAD EN PROBLEMAS DE REDES

POR
ING. ROSA MARIA DE LA CRUZ FERNANDEZ

COMO REQUISITO PARCIAL
PARA OBTENER EL GRADO DE
MAESTRO EN CIENCIAS
DE LA ADMINISTRACION CON ESPECIALIDAD
EN SISTEMAS

DICIEMBRE, 1998

UN SISTEMA PARA LA DETECCION Y ELIMINACION DE INFACTIBILIDAD EN PROBLEMAS DE REDES

UNIVERSIDAD AUTONOMA DE NUEVO LEON
FACULTAD DE INGENIERIA MECANICA
Y ELECTRICA
DIVISION DE ESTUDIOS DE POSTGRADO

UN SISTEMA PARA LA DETECCION Y ELIMINACION DE INFACTIBILIDAD EN PROBLEMAS DE REDES

POR
ING. ROSA MARIA DE LA CRUZ FERNANDEZ

COMO REQUISITO PARCIAL
PARA OBTENER EL GRADO DE
MAESTRO EN CIENCIAS
DE LA ADMINISTRACION CON ESPECIALIDAD
EN SISTEMAS

DICIEMBRE. 1998

UNIVERSIDAD AUTÓNOMA DE NUEVO LEÓN

FACULTAD DE INGENIERÍA MECÁNICA Y ELÉCTRICA

DIVISIÓN DE ESTUDIOS DE POSTGRADO

UN SISTEMA PARA LA DETECCIÓN Y ELIMINACIÓN DE INFACTIBILIDAD EN PROBLEMAS DE REDES

Por

ING. ROSA MARÍA DE LA CRUZ FERNÁNDEZ

Como requisito parcial para obtener el Grado de MAESTRO EN CIENCIAS DE LA ADMINISTRACIÓN CON ESPECIALIDAD EN SISTEMAS

Diciembre, 1998

UNIVERSIDAD AUTÓNOMA DE NUEVO LEÓN
FACULTAD DE INGENIERÍA MECÁNICA Y ELÉCTRICA
DIVISIÓN DE ESTUDIOS DE POSTGRADO

Los miembros del comité de tesis recomendamos que la tesis UN SISTEMA PARA LA DETECCIÓN Y ELIMINACIÓN DE INFACTIBILIDAD EN PROBLEMAS DE REDES realizada por el Ing. Rosa María de la Cruz Fernández sea aceptada para su defensa como opción al grado de Maestro en Ciencias de la Administración con especialidad en Sistemas.

El Comité de Tesis

Asesor
Dra. Ada Margarita Álvarez Socarrás

Coasesor
Dr. Victoriano F. Alatorre González

Coasesor
Dr. Ernesto Vázquez Martínez

Vo. Bo.
M.C. Roberto Villarreal Garza
División de Estudios de Postgrado

San Nicolás de los Garza, Nuevo León, a Julio de 1998

DEDICATORIAS

A Dios . . .

Por darme la oportunidad de seguir desarrollándome tanto personal como profesionalmente.

A mi Esposo . . .

Arturo del Angel Ramírez, por su amor y su apoyo incondicional.

A mis Padres . . .

Sr. Ramón de la Cruz Carreón y Sra. María Santos Fuentes de de la Cruz, por su amor y apoyo moral como espiritual para concluir con esta meta.

A mis Hermanos . . .

Juan Ramón y Ana Catalina de la Cruz Fernández, por su comprensión.

A mis Amigos y Compañeros . . .

Por su valiosa amistad y comprensión en los buenos y malos momentos de mi vida.

Y a todas aquellas personas que de alguna u otra forma contribuyeron para la terminación de esta meta.

AGRADECIMIENTOS

Quiero expresar mi más sincero agradecimiento a la Dra. Ada Margarita Álvarez Socarrás, asesora de mi tesis, por toda la paciencia, confianza y dedicación que puso en este trabajo para poder concluir con esta meta. Quiero agradecer a mis coasesores, el Dr. Ernesto Vázquez Martínez y el Dr. Victoriano Alatorre González, por su ayuda incondicional en la realización de esta meta. También, al Ing. Apolinar Zapata, por sus correcciones y recomendaciones para el término de la tesis

Al Consejo de Ciencia y Tecnología por el apoyo económico para la realización de mis estudios.

Agradezco también a todos mis maestros que compartieron sus conocimientos y su experiencia en las horas clases, además de su amistad y consejo para mejorar personal como profesionalmente.

RESUMEN

Rosa María de la Cruz Fernández **Fecha de Graduación: 1998**

Universidad Autónoma de Nuevo León

Facultad de Ingeniería Mecánica y Eléctrica

Título del Estudio: **UN SISTEMA PARA LA DETECCIÓN Y ELIMINACIÓN DE INFACTIBILIDAD EN PROBLEMAS DE REDES**

Número de Páginas: 81

Candidato para el grado de Maestría en Ciencias de la Administración con Especialidad en Sistemas

Área de Estudio: **Ingeniería**

Propósito y Método del Estudio: Actualmente, en toda empresa que hace uso diario de los términos de ofertas y demandas, y quienes juegan el papel de proveedor, están atentos en satisfacer las demandas que el cliente requiere, y si es necesario, las condiciones que él impone. Un ejemplo claro, es referente a un cliente que decide qué proveedor deberá surtir su producto. Este caso, representado por ofertas, demandas y restricciones, se refiere a un modelo de red bipartita incompleta, ya que existen celdas prohibidas en la red. Cuando se habla de la solución óptima que puede tomar un problema de programación lineal, se espera que el problema tenga solución factible para proceder a la toma de decisiones; pero cuando el problema no tiene solución factible, se dice que el problema es *infactible*. Uno de los problemas que se presenta en la modelación de redes, es la existencia de infactibilidad en la red. El propósito de este trabajo es presentar un sistema para la detección y eliminación de la infactibilidad en los modelos de redes. La detección de la infactibilidad en la red se realizará con el apoyo del algoritmo del Flujo Máximo, y la eliminación de esta infactibilidad, se hará por medio de un algoritmo propuesto es esta tesis, la cual recibe el nombre de algoritmo heurístico *De la Cruz*. Este algoritmo realiza ajustes mínimos solamente en los valores de las ofertas, con el fin de satisfacer completamente las demandas requeridas por el cliente. Cuando el problema es convertido en factible por medio del algoritmo *De la Cruz*, esta red puede ser solucionada por cualquier algoritmo de programación lineal.

Contribuciones y Conclusiones: De acuerdo al análisis de numerosos problemas de redes bipartitas incompletas, resueltos por métodos que resuelven la infactibilidad de problemas de redes [Currin,Klingman,Roodman; 1986,1988,1979] y por el algoritmo que se presenta en esta tesis, se concluye que el algoritmo heurístico *De la Cruz,* ajusta mínimamente los valores de las ofertas en la red en comparación con las otros métodos. Con este nuevo algoritmo, se contribuye a eliminar la infactibilidad en problemas de flujos sobre redes bipartitas incompletas.

FIRMA DEL ASESOR: _____

Dra. Ada Margarita Álvarez Socarrás

TABLA DE CONTENIDO

Capítulo **Página**

LISTA DE FIGURAS

Figura **Página**

Figura **Página**

NOMENCLATURA

:	Tal que
$\mathbf{R}$	Conjunto de los números reales
$\mathbf{R}^{+}$	Conjunto de números reales no negativos
$\mathbf{R}_{e}^{+}$	$\{\mathbf{R}^{+} \cup \infty\}$
$\lceil x \rceil$	Menor entero mayor o igual que x, $x \in \mathbf{R}$
$\lfloor x \rfloor$	Mayor entero menor o igual que x, $x \in \mathbf{R}$
Δ	Igualdad por definición
(x,y)	El arco del nodo inicial x al nodo final y
N	Conjunto de nodos
A	Conjunto de arcos
(N,A)	Red con nodos en N y arcos en A
S	Conjunto de fuentes
T	Conjunto de sumideros
R	Conjunto de intermedios
$a(x)$	Oferta en el nodo x
$b(x)$	Demanda en el nodo x
$A(x)$	$\{y \in N: (x,y) \in A\}$
$B(x)$	$\{y \in N: (y,x) \in A\}$
$\&$	Grafo
$\&(N,A)$	Grafo (dirigido) con nodos en N y arcos en A
$c(x,y)$	Capacidad del arco (x,y)
$f(x,y) = f_{xy}$	Flujo (uniproducto) sobre el arco (x,y)

CAPÍTULO 1

INTRODUCCIÓN

1.1 *Establecimiento del problema*

Donde quiera que miremos en nuestras vidas diarias aparecen las redes: las redes eléctricas y de potencia nos traen luz y entretenimiento a nuestros hogares; las redes telefónicas nos permiten comunicarnos con personas lejanas; las redes de carretera y ferrocarril nos permiten atravesar grandes distancias geográficas y las redes de manufactura y distribución nos dan acceso a productos de consumo.

De igual forma, los conceptos de ofertas y demandas son manejados en todas partes del mundo, ya sea en las grandes empresas, microempresas, miniempresas, negocios y hasta en los hogares. Quienes juegan el papel de proveedores, se preocupan por satisfacer las demandas que el cliente requiere, pero no siempre pueden cumplir con el cliente por diversos motivos.

El problema que enfrentan los proveedores hacia los clientes, es que el cliente pone las restricciones, por ejemplo, para el caso de una empresa, el cliente puede decidir qué proveedor y sucursal (bajo la misma empresa) le surtirá el producto.

Lo anterior lleva en ciertos casos a la representación de una red incompleta, ya que algunos de los enlaces entre proveedores y clientes estarán prohibidos debido a que no todos los proveedores de la misma empresa deben de surtir el producto que el cliente requiere, ya que él mismo tomó la decisión de qué proveedor le sería más conveniente.

Al modelarse una red para este tipo de problemas se tiene el riesgo de que el modelo sea *infactible*, es decir, que no se puedan satisfacer completamente los requerimientos de los demandantes.

En la actualidad, existen diversos métodos que dan solución a la infactibilidad de problemas de redes, pero estos métodos se fundamentan en la filosofía de "*ofrecer lo que tenemos*" y esto implica quedar mal con algún cliente.

Estos métodos tienen la desventaja de no satisfacer completamente con lo que el cliente requiere, ya que la manera en que ellos resuelven el problema es modificando las demandas de los clientes y en ocasiones, también las ofertas de los proveedores. Lo más conveniente sería respetar la demanda del cliente y así lograr la satisfacción al 100%.

Basándose en lo anterior, consideramos que sería interesante y útil realizar el presente estudio con el objetivo de diseñar una técnica que elimine la infactibilidad de este tipo de redes bajo la filosofía de "satisfacer las demandas que el cliente requiere".

1.2 *Objetivos*

Los objetivos de este trabajo son los siguientes:

1) Estudiar los diferentes métodos para eliminar la infactibilidad en problemas de redes, con el fin de establecer una comparación con el método propuesto en esta tesis.

2) Establecer un nuevo método que elimine el problema de la infactibilidad en una red, ajustando mínimamente los valores de las ofertas para satisfacer completamente las demandas del cliente.

3) Desarrollar un sistema que detecte la existencia de infactibilidad en una red, y que la elimine mediante el algoritmo *De la Cruz*, generando un nuevo modelo de red factible.

1.3 *Estructura de la tesis*

La tesis cuenta con 6 capítulos y un apéndice. El primer capítulo se dedicó al establecimiento del problema, la justificación y los objetivos del estudio.

En el capítulo 2 introducen a los conceptos básicos sobre redes y flujos, la clasificación de los problemas de flujos en redes, y algunos tipos de redes bipartitas. Se muestran algunos problemas de optimización sobre redes bipartitas y se define el problema de investigación.

En el capítulo 3 se formula una condición necesaria y suficiente para la infactibilidad en una red y se analizan algunos métodos que eliminan la infactibilidad en problemas de redes bipartitas incompletas propuestos por distintos investigadores.

El capítulo 4 se dedica fundamentalmente a exponer el algoritmo heurístico creado para eliminar la infactibilidad en una red bipartita incompleta, mientras que en el capítulo cinco, se describe el sistema diseñado y los resultados obtenidos con él.

Por último, en el capítulo 6 se presentan las conclusiones de la investigación y recomendaciones para investigaciones futuras en esta área.

La codificación del sistema *Rosma* que detecta y elimina la infactibilidad en problemas de redes, se presenta en el Apéndice A.

C A P Í T U L O 2

CONCEPTOS BÁSICOS DE FLUJO EN REDES

2.1 *Introducción*

En este capítulo se exponen algunos de los conceptos y resultados de la teoría de flujos en redes necesarios para fundamentar las soluciones del problema de investigación planteadas en la presente tesis.

2.2 *Conceptos básicos*

Un *grafo dirigido*, o simplemente *grafo*, está formado por un conjunto finito N cuyos elementos son llamados *nodos*, y un conjunto finito A de parejas ordenadas de nodos, cuyos elementos son llamados *arcos dirigidos*, o simplemente *arcos*. Denotaremos el grafo anterior como $\&(N, A)$, los nodos como $x, y,$ y el arco que va del nodo x al nodo y como (x, y). Los nodos x, y son llamados extremos del arco, o bien, *nodo inicial y nodo terminal* respectivamente.

$$\sum_{y \in A(x)} f(x,y) - \sum_{y \in B(x)} f(y,x) = \begin{cases} F, \text{ si } x = s \\ 0, \text{ si } x \neq s, t \\ -F, \text{ si } x = t \end{cases} \quad \textbf{(2.1)}$$

$$0 \leq f(x,y) \text{ para cada } (x,y) \in A, \quad \textbf{(2.2)}$$

$$f(x,y) \leq c(x,y) \text{ para cada } (x,y) \in A, \quad \textbf{(2.3)}$$

donde s es llamado *fuente*, t *sumidero*, y los restantes nodos son llamados *intermedios* (ver la figura 2.2).

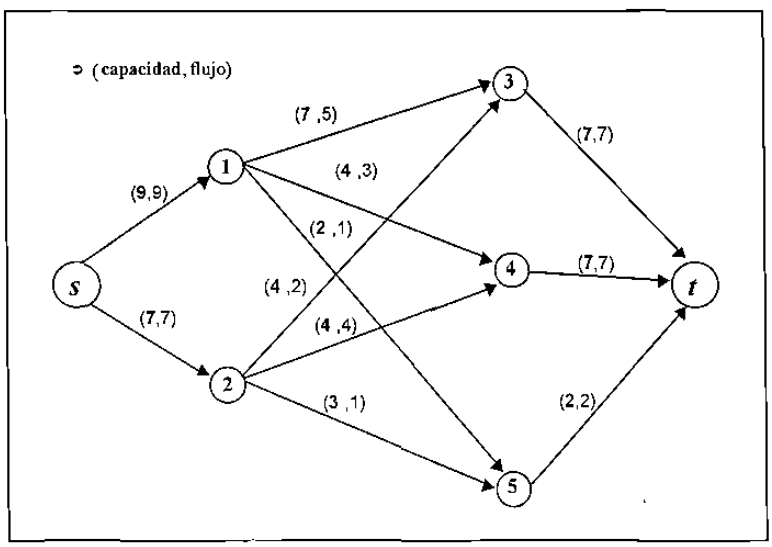

Figura 2.2. Red capacitada con flujo asociado en cada arco.

La ecuación (2.1) es llamada de *conservación de flujo*, la desigualdad (2.2) es llamada *restricción de no-negatividad* y la (2.3) es llamada *restricción de capacidades.*

Si $(x, y) \in A$, el valor de $f(x, y) = f_{xy}$ puede ser interpretado como la cantidad de algún producto que circula a través del arco (x,y). Si las capacidades de los arcos son

infinitas, esto es, si eliminamos la restricción (2.3) en la definición de flujo, se dice que el flujo es *sin capacidades*.

La definición anterior se puede generalizar cuando existen en la red múltiples fuentes y sumideros [Ford and Fulckerson,1962]. Sea (N,A) una red arbitraria, donde N es particionado en tres subconjuntos: S (fuentes), R (nodos intermedios), y T (sumideros). Un *flujo con capacidades de* S a T es una función real no negativa f definida sobre A que satisface:

$$\sum_{y\in A(x)} f(x,y) - \sum_{y\in B(x)} f(y,x) = 0 \qquad \text{para } x \in R \tag{2.4}$$

$$0 \le f(x,y) \le c(x,y) \qquad \text{para } (x,y) \in A \tag{2.5}$$

con valor de flujo

$$F = \sum_{(x,y)\in(S,A(x))} f(x,y) - \sum_{(y,x)\in(B(x),S)} f(y,x) \tag{2.6}$$

donde:

$$(S,A(x)) \triangleq \{(x,y): x \in S,\ y \in A(x)\} \text{ y}$$
$$(B(x),S) \triangleq \{(y,x): x \in S,\ y \in B(x)\}. \tag{2.7}$$

No obstante esta generalización, es posible transformar cualquier problema de flujos sobre una red con dos conjuntos de fuentes y sumideros, S y T respectivamente, en un problema de flujos donde se tenga solamente una fuente y un sumidero [Bazaraa, 1990].

2.3 *Clasificación de los problemas de flujos en redes*

En general se distinguen dos clases de problemas de flujos en redes [Assad, 1978]:

1) *Problemas en análisis de redes.* En este tipo de problemas se analiza una configuración de red específica para encontrar el patrón de flujo óptimo con respecto a alguna función objetivo. Por ejemplo, maximizar el flujo de un producto en un problema de flujo máximo, o minimizar los costos de satisfacer un conjunto dado de requerimientos en problemas de costo mínimo.

2) *Problemas en diseño de redes.* En este tipo de problemas se busca una configuración de red óptima que satisfaga un conjunto dado de requerimientos. Por ejemplo, satisfacer ciertos requerimientos instalando capacidades sobre la red, y minimizar el costo total de instalación.

2.4 *Algunos tipos de redes*

2.4.1 Redes bipartitas completas

Sea (N,A) una red de tal forma que $N=\{x_1, x_2,\ldots,x_m, x_{m+1},\ldots,x_n\}$ y $A=\{(x_i, x_j): i=1,\ldots,m; j=m+1,\ldots n\}$, entonces (N,A) es llamada *red bipartita completa.* La red (N,A) es bipartita en el sentido que N está dividido en subconjuntos $S=\{x_1, x_2,\ldots,x_m\}$ de fuentes y $T=\{x_{m+1}, x_{m+2},\ldots,x_n\}$ de sumideros (note que $N=S\cup T$ y $R=\varnothing$). La red (N,A) es completa en el sentido de que todos los arcos posibles de S a T están presentes, tal y como se muestra en la figura 2.3.

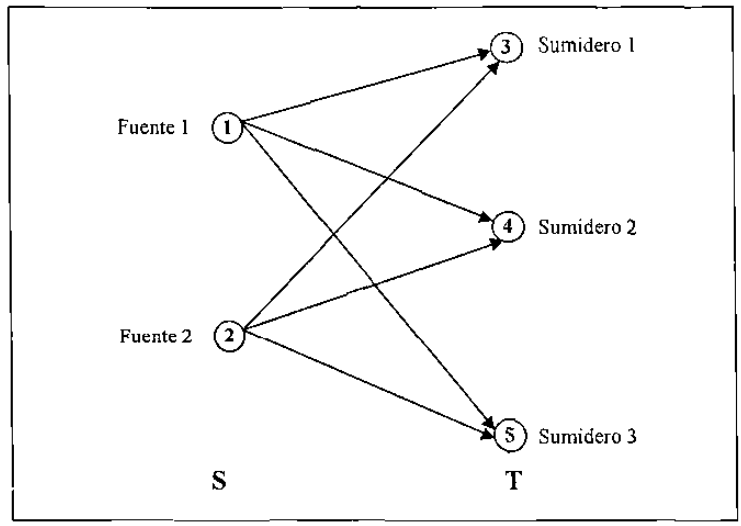

Figura 2.3. Red bipartita completa.

En este tipo de red usualmente se asocia a cada $x \in S$ un número entero no negativo $a(x)$, que significa la *oferta* de un producto en x, y a cada $x \in T$ un número entero no negativo $b(x)$, que significa la *demanda* del mismo producto en x. A este tipo de red se le conoce como *red bipartita completa* con ofertas-demandas. Un ejemplo de este tipo de red se muestra en la figura 2.4.

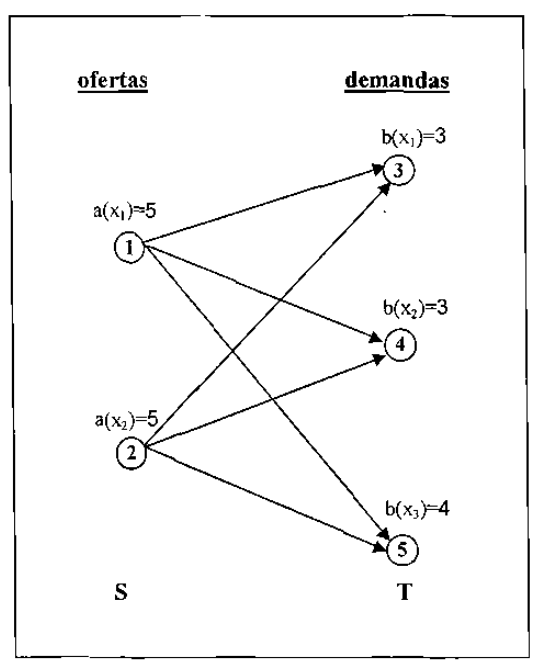

Figura 2.4. Red bipartita completa con ofertas-demandas.

Se define que una red es *balanceada* si cumple que la demanda total del producto sea igual a la oferta total, esto es, si satisface la condición:

$$\sum_{x \in T} b(x) = \sum_{x \in S} a(x) \qquad \textbf{(2.8)}$$

La red representada en la figura 2.4 es un ejemplo de red balanceada.

Sea (N,A) una red bipartita con ofertas-demandas, en donde $N = S \cup T$, y *s*upongamos que se satisface la ecuación de balance (2.8).

Definamos sobre A una función entera no negativa $f(x_i, x_j)$ que representa el flujo de dicho producto que va de x_i a x_j por el arco (x_i, x_j), de tal forma que se cumplan las siguientes restricciones:

$$\sum_{xj \in T} f(x_i, x_j) = a(x_i)\text{, para cada } x_i \in S \qquad \textbf{(2.9)}$$

$$\sum_{xi \in S} f(x_i, x_j) = b(x_j)\text{, para cada } x_j \in T \qquad \textbf{(2.10)}$$

$$0 \leq f(x_i, x_j) \text{ para cada } x_i \in S \text{ y } x_j \in T \qquad \textbf{(2.11)}$$

A una asignación de flujos sobre la red que cumpla las restricciones (2.9), (2.10) y (2.11) se denomina flujo factible.

Nótese que las condiciones anteriores significan lo siguiente:

(2.9) Expresa que para cada nodo fuente x_i debe cumplirse que el flujo total que sale por todos los arcos desde x_i hacia todos los sumideros debe ser igual a la oferta de dicho nodo fuente $(a(x_i))$.

(2.10) Expresa que para cada nodo sumidero x_j debe cumplirse que el flujo total que llega a x_j desde todas las fuentes debe ser igual a la demanda de dicho nodo sumidero ($b(x_j)$).

(2.11) Expresa que para cada arco de la red, el valor del flujo que circula por él debe ser mayor o igual a cero.

La condición (2.8) es necesaria y suficiente para que exista una asignación de flujos que satisfaga todas las demandas a partir de las ofertas existentes en la red [Ahuja, Magnanti and Orlin, 1993], esto es, para que exista una solución factible.

2.4.2 Redes bipartitas incompletas

Sea una red $R=(N,A)$ donde N denota el conjunto de vértices y A el conjunto de arcos, donde

$$N = S \cup T,\ |N| = n,\ |S| = m,\ S \cap T = \varnothing$$

$$A \subset S \times T \quad \text{y}$$

$$(x_i, x_j) \in A$$

A este tipo de red se le denomina *red bipartita incompleta*. El conjunto de nodos también se encuentra particionado en dos subconjuntos, pero no están presentes todos los arcos posibles de S a T; es decir, cuando algún nodo fuente sólo se une con algunos nodos sumideros, no con todos, se le conoce como "*red bipartita incompleta*" porque algunos de los arcos de la red están prohibidos. Este tipo de red es el objeto de estudio en este trabajo (ver la figura 2.5).

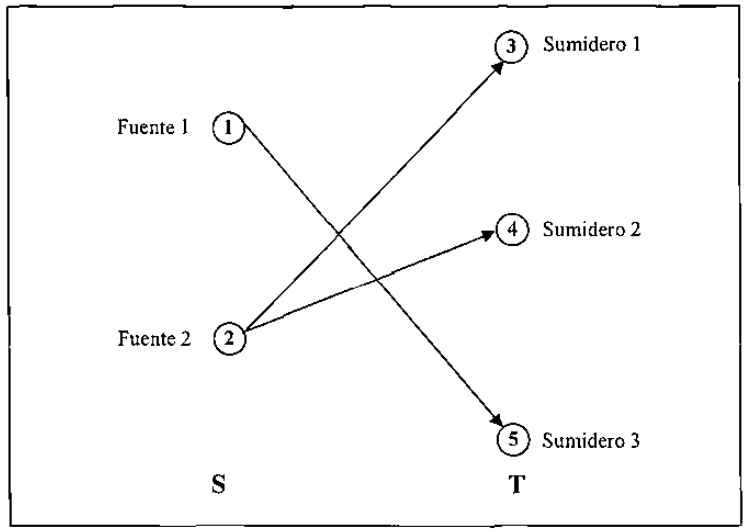

Figura 2.5. Red bipartita incompleta.

Al igual que en las redes bipartitas completas, en estas redes también pueden asociarse valores enteros no negativos *a(x)* a cada $x \in S$, que significa la *oferta* de un producto en x, y a cada $x \in T$ un número entero no negativo *b(x)*, que significa la *demanda* del mismo producto en x.

Ahora bien, a diferencia de las redes bipartita completas, la condición de balance $\sum_{x \in T} b(x) = \sum_{x \subset S} a(x)$, no es una condición suficiente para garantizar la existencia de una distribución de flujos que satisfaga las demandas a partir de las ofertas existentes.

2.5 *Algunos problemas de optimización sobre redes bipartitas*

Algunos problemas de optimización sobre redes bipartitas son:

- Buscar la asignación de flujos al menor costo posible (esto es cuando sobre los arcos también hay definida una función de costo) [Bazaraa, Jarvis and Sherali, 1990]. Este tipo de problema se le considera de análisis.

- Buscar la mínima capacidad que pueden tener los arcos para que exista una distribución de flujos que satisfaga las demandas a partir de las ofertas [Álvarez, Martínez; 1996]. Este tipo de problema se le considera de diseño.

- Buscar una red que sujeta a la capacidad mínima, tenga un número mínimo de arcos [Álvarez, Martínez; 1996]. Este problema es considerado de diseño óptimo.

2.6 *Definición del problema de investigación*

Una red (N, A) puede ser representada matricialmente de la siguiente forma:

- A cada fila de la matriz le corresponde un nodo fuente de la red y a cada columna un nodo sumidero.

- Si el arco $(i, j) \notin A$, donde $i \in S$ y $j \in T$, se tachará la correspondiente celda de la matriz (intersección de fila i y la columna j).

- Los valores de las ofertas de los nodos, se colocarán en el extremo izquierdo de la tabla y los valores de las demandas de los nodos, se colocarán en la parte superior de la tabla.

Por ejemplo, la red de la figura 2.5 se representa matricialmente de la siguiente forma:

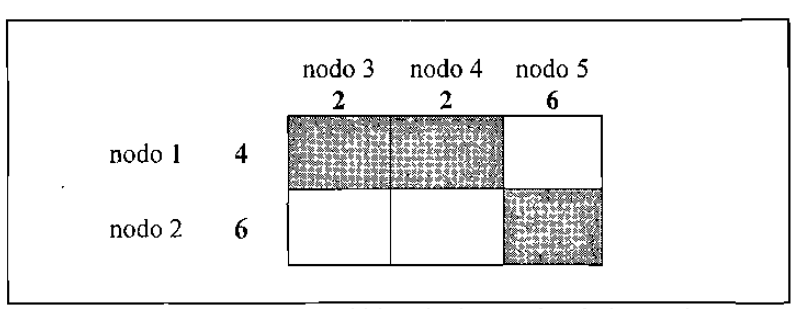

Figura 2.6. Red bipartita incompleta balanceada.

En las redes bipartitas incompletas, la condición de balance $\sum_{x \in T} b(x) = \sum_{x \in S} a(x)$, no asegura que exista solución factible, esto es, un flujo factible. Cuando el problema no tiene solución factible, se dice que es *infactible*.

Por ejemplo, la red de la figura 2.6 está balanceada, sin embargo es imposible satisfacer la demanda del nodo 5 ya que él está enlazado únicamente al nodo 1 cuya oferta es solamente de 4 unidades.

Con este ejemplo se concluye que la ecuación de balance no es una condición suficiente para que un problema de red bipartita incompleta sea factible.

La mayoría de los códigos existentes para resolver problemas de redes, cuando detectan infactibilidad en un problema, se limitan a informar al usuario de la infactibilidad, sin suministrar información adicional.

En ocasiones la infactibilidad se debe a errores en la modelación, pero en otros casos no y sería bueno suministrar al usuario la siguiente información:

- La localización de errores de entrada durante la modelación.

- Suministrar un reporte de las violaciones que se presentaron durante el proceso de la red.

Es útil para el usuario no sólo conocer que el problema es infactible, sino también conocer métodos para investigar por qué existe la infactibilidad y qué pudiera hacerse para eliminarla.

Se han desarrollado trabajos que eliminan la infactibilidad en las redes [Klingman, 1988; Roodman, 1979; Currin, 1986], pero como se verá en el siguiente capítulo, se les puede señalar como desventaja que para obtener factibilidad en la red, se

tiene que realizar ajustes tanto en las demandas, como también en algunos casos, en las ofertas de dicha red.

En este trabajo se presenta un algoritmo heurístico que se elimina la infactibilidad de una red realizando ajustes mínimos en los valores de las ofertas, sin afectar las demandas.

C A P Í T U L O 3

MÉTODOS QUE ELIMINAN LA INFACTIBILIDAD EN PROBLEMAS DE REDES BIPARTITAS INCOMPLETAS

3.1 Introducción

En este capítulo se describen algunos métodos utilizados para la eliminación de infactibilidades en problemas de redes. Algunos métodos que se presentan hacen posible la factibilidad de un problema, modificando las demandas de los nodos sumideros sin afectar las ofertas de los nodos fuentes, o modificando las ofertas y demandas de los nodos fuentes y sumideros; otros, identifican las restricciones que pudieran ser "relajadas" para llegar a una factibilidad y poder estimar los cambios a realizarse. En este capítulo se explicarán las principales características de estos métodos y se discutirán sus desventajas.

3.2 Condición necesaria y suficiente para la factibilidad

Cuando en una red bipartita algún nodo fuente sólo se une con algunos nodos sumideros, no con todos, se le conoce como "*red bipartita incompleta*", ya que algunos de los arcos de la red están prohibidos. Este tipo de red es el objeto de estudio en este trabajo de tesis.

Sea (N, A) una red bipartita incompleta con fuentes S y sumideros T, donde se asocia a cada $x \in S$ un número entero no negativo $a(x)$, que significa la *oferta* de un producto en x, y a cada $x \in T$ un número entero no negativo $b(x)$, que significa la *demanda* del mismo producto en x (ver la figura 3.1).

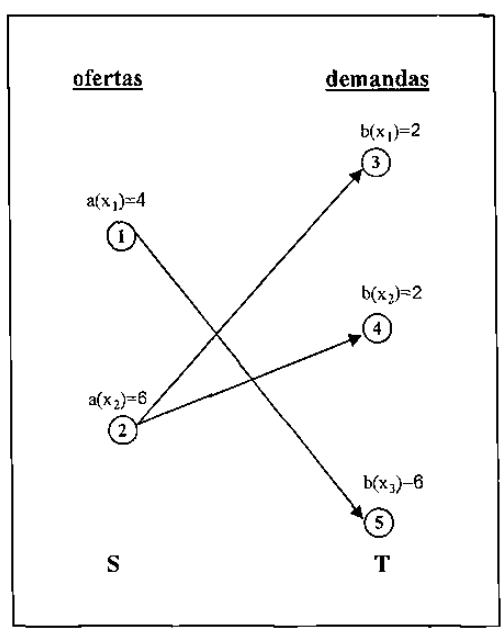

Figura 3.1. Red Bipartita incompleta con ofertas y demandas.

Como se ilustró anteriormente, en las redes bipartitas incompletas, la condición de balance $\sum_{x \in T} b(x) = \sum_{x \in S} a(x)$ no asegura que exista una solución factible. Cuando el problema no tiene solución factible, se dice que es *infactible*.

En diversos artículos [Emelichev, 1981; Kravtsov, 1982-1990] se enuncian criterios para la comprobación de la factibilidad en los casos en que el conjunto de arcos prohibidos cumple cierta regularidad prefijada.

[Guardado, 1986], demuestra una condición necesaria y suficiente para la existencia de solución factible en una red bipartita incompleta, sin embargo, impone la condición de que cada vértice tenga asociados β arcos, donde β es una constante positiva tal que $\beta < |T| - |S|$.

Por su parte, Dantzing [Dantzing, 1963] no presenta ninguna hipótesis de factibilidad al problema "*con celdas inadmisibles*", solo aplica el Método de las Dos fases utilizando una función con variables artificiales en la primera fase del Simplex; si al concluir la primera fase del método, la función toma valor cero, el problema tiene solución factible. Alvarez y Martínez [Alvarez y Martínez, 1995] estudiaron este problema y obtuvieron una condición necesaria y suficiente para la existencia de una solución factible. Esta condición es obtenida a partir del Teorema de Gale [Simonnnard, 1972] para redes arbitrarias, y se expresa de la siguiente forma:

$$\sum_{x_i \in \Gamma(T_1)} a(x_i) \geq \sum_{x_j \in T_1} b(x_j) \qquad \forall \quad T_1 \subseteq T \qquad \textbf{(3.1)}$$

donde $\Gamma(T_1)$ es el conjunto de nodos enlazados con los nodos de T_1.

Esto es, la condición necesaria y suficiente para que el problema sea factible expresa que para cualquier subconjunto de sumideros que se seleccione, la suma de sus demandas no debe exceder la suma de las ofertas de las fuentes enlazadas con ellos.

Una forma de verificar esta condición es utilizando el algoritmo de Flujo Máximo [Ford and Fulkerson, 1962]. Para ello, se procede como sigue:

- Formar una red aumentada (N^*, A^*) añadiendo vértices s, t y arcos (s, S), (T, t) (ver la figura 3.2).

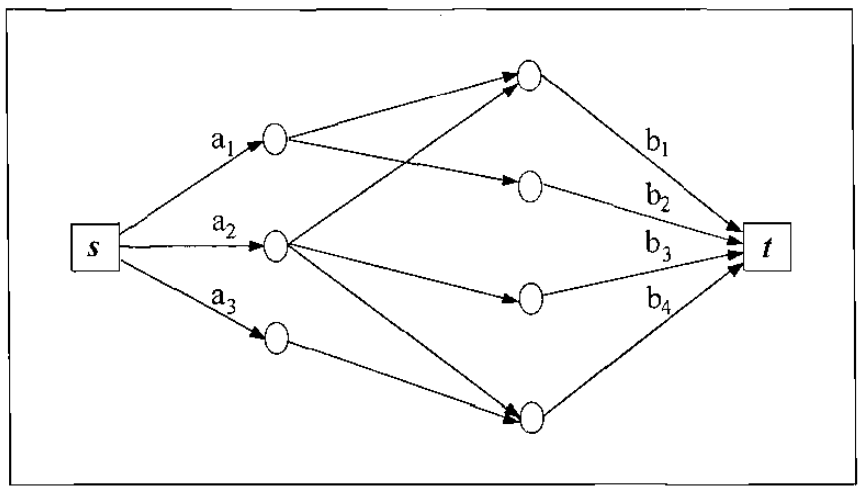

Figura 3.2. Red bipartita incompleta aumentada.

- Definir una función capacidad en N^*.

$$c^*(s, x_i) = a_i \qquad \forall x_i \in S$$

$$c^*(x_j, t) = b_j \qquad \forall x_j \in T$$

$$c^*(x_i, x_j) = \infty \qquad \forall (x_i, x_j) \in A$$

- Aplicar el algoritmo de Flujo Máximo del nodo s al nodo t.

Si el valor del flujo máximo que llega a t es $\sum_{x_j \in T} b_j$ el problema es factible; si es menor, es *infactible*.

3.3 Métodos para eliminar la infactibilidad

Existen programas de computadoras que proporcionan información al usuario cuando se detecta una infactibilidad en un problema de flujo en red o de programación lineal. Dicha información no se refiere a cómo solucionar la infactibilidad, sino más bien, proporciona:

- Un resumen de la solución terminada y encontrada, indicando las variables o restricciones infactibles.

- Información acerca de los errores de entrada que tiene el modelo.

No obstante, actualmente no existe una metodología que proporcione una guía discernida al usuario para reexaminar el modelo de red. Al existir infactibilidad en un modelo de red, se desea conocer cuál es el motivo de tal infactibilidad y cómo eliminar esta infactibilidad para poder obtener una solución factible y óptima de acuerdo a ciertos criterios.

En la literatura se han reportado trabajos como los de Currin [Currin, 1986], Klingman [Klingman, 1988] y Roodman [Roodman, 1979] en los que se presentan distintos métodos para eliminar la infactibilidad. A continuación se describen estos algoritmos.

3.3.1 Algoritmo de Currin

Currin [Currin, 1986] resuelve una sucesión de problemas con el objetivo de minimizar la máxima insatisfacción en los sumideros, para lo cual se realiza un ajuste en la demanda de los mismos.

El método de Currin tiene la desventaja de no satisfacer completamente las demandas originales de la red, ya que después de aplicar el método, el flujo total que se asigna a algunos sumideros, es a veces menor o mayor a la cantidad de unidades que demandan; y en la práctica esto no es conveniente, ya que un cliente nunca comprará más de lo que requiere.

Esta desventaja es dada porque Currin asigna en primer instancia un flujo en la red basándose en los valores de las ofertas; posteriormente, analiza las demandas en donde el flujo que se le asignó es mayor a la demanda requerida y, realiza un ajuste sobre estas demandas, aumentando aquellas ofertas en donde se realizó dicho cambio.

Por otra parte, cuando la asignación total de flujo a algunos sumideros es menor de lo que se demanda, se realiza una disminución en las ofertas afectadas. Esto significa que en algunos modelos de redes, Currin efectúa cambios tanto en las ofertas como en las demandas.

A pesar del objetivo de Currin de minimizar los ajustes realizados en las demandas, no se obtiene una satisfacción total en las mismas.

3.3.2 Algoritmo de Klingman

Klingman [Klingman, 1988] considera que la infactibilidad en una red puede ser debida a que la cantidad total de ofertas no es igual al total de las demandas, o a la existencia de rutas (arcos) inadmisibles. Por tal motivo, él muestra que el problema puede ser modelado y resuelto como un problema de capacidad de transporte, multicriterio. Es decir, que puede obtenerse una solución óptima resolviendo un problema de transporte uni-criterio (seleccionando pesos suficientemente grandes para agregar el criterio) o mediante una sucesión de problemas de transporte uni-criterio.

El objetivo del algoritmo de Klingman es minimizar la máxima desviación entre la fracción baja-oferta de los nodos sumideros.

A continuación se comparan los métodos de Currin y Klingman:

- El método de Currin requiere un proceso de solución especial, el cual es similar al del Flujo Máximo con restricciones, mientras que el algoritmo de Klingman únicamente requiere de un código estándar para resolver el problema de transporte capacidad o un problema de transbordo.

- El proceso de Klingman requiere de la solución de uno, dos o tres problemas de transporte capacitados, donde la solución de cada problema es utilizada como la solución inicial al siguiente problema. El algoritmo de Currin consiste en resolver problemas de flujo máximo con restricciones para cada nodo insatisfecho para ajustar las demandas y entonces tiene que resolver el problema original de transporte.

- El proceso de Klingman tiene una ventaja computacional específicamente para problemas grandes, donde los tiempos de solución mediante el algoritmo de Currin pudieran deteriorarse substancialmente.

En conclusión, el investigador Klingman elimina la infactibilidad en redes resolviendo un problema apropiado de transporte con capacidades multicriterio.

3.3.3 Algoritmo de Roodman

Otro trabajo que analiza la infactibilidad en redes es el desarrollado por Garay M. Roodman [Roodman, 1979], el cual propone un algoritmo para el análisis de post-infactibilidad sobre los problemas de programación lineal en general. El objetivo del algoritmo propuesto por Roodman es identificar las restricciones que pueden ser "relajadas" para poder llegar a una factibilidad y estimar las magnitudes de los cambios requeridos.

Roodman hace uso del Método Simplex y del Método Dual Simplex utilizando variables artificiales (bajo ciertas restricciones) para dar solución a la infactibilidad de problemas y establecer una solución factible.

El propósito del algoritmo de Roodman es desarrollar un método de Análisis de Post-Infactibilidad, similar en el propósito y el diseño a los métodos tradicionales de Análisis Post-Óptimo. El algoritmo tiene las siguientes características:

- Identifica restricciones o grupos de restricciones cuyos lados derechos puedan ser "relajados" hasta llegar a una condición de factibilidad.

- Estima las magnitudes de los cambios requeridos. Esta información es similar a la información suministrada por un análisis elemental de post-optimalidad.

La ventaja del algoritmo de Roodman es que computacionalmente es barato, y la desventaja es que al eliminar la infactibilidad de un problema de red, realiza ajustes en las ofertas y en las demandas y, generalmente el número de ajustes que realiza son demasiado grandes, en comparación con el algoritmo de Currin.

3.4 *Conclusiones del capítulo*

La mayoría de los programas existentes no proporcionan la información necesaria para dar solución a un problema de infactibilidad; normalmente quedan las siguientes interrogantes: ¿Cuál fue el motivo de la infactibilidad?, ¿Existen problemas en el modelo de la red?, si hay infactibilidad ¿Cómo poder solucionar el problema?; se puede continuar con una serie de interrogantes en las que muchas veces se concluye que la infactibilidad es debida a la modelación, por la cual se procede a remodelar la red; no obstante, el problema no siempre está en la modelación.

Diversos investigadores han propuesto varias técnicas para dar solución a la infactibilidad en los problemas de redes. Entre ellos:

- Currin, propone minimizar la máxima insatisfacción en los sumideros;

- Klingman soluciona la infactibilidad por medio de un problema de transporte con capacidades multicriterio;

- Roodman propone un análisis de post-infactibilidad sobre los problemas de programación lineal.

Los algoritmos mencionados anteriormente son un medio para eliminar la infactibilidad en redes, pero algunos con la desventaja de modificar las demandas de la red y en ocasiones las ofertas de la misma red, provocando una insatisfacción a los demandantes.

En el próximo capítulo se desarrollará un algoritmo heurístico que elimina la infactibilidad realizando ajustes mínimos solamente en las ofertas de los nodos fuentes.

C A P Í T U L O 4

ALGORITMO HEURÍSTICO De la Cruz

4.1 *Introducción*

Como se indicó en el capítulo anterior, los algoritmos empleados para la eliminación de infactibilidad en redes, se llevan a cabo mediante la modificación de las demandas en dicha red, de tal manera que no satisfacen completamente con las demandas requeridas originalmente, o envían a un sumidero más flujo que la demanda que él requiere. Esto último no es recomendable, ya que no siempre puede obligarse a un cliente a comprar más de lo que requiere. Otros métodos están diseñados para problemas de programación lineal, por lo cual no resultan muy eficientes en el caso de redes bipartitas incompletas.

El objetivo de este capítulo el desarrollo de un algoritmo heurístico que permita eliminar la infactibilidad en problemas de redes bipartitas incompletas, afectando solamente los nodos fuentes de la red.

4.2 *Utilización del algoritmo de Flujo Máximo en la detección de la infactibilidad*

Cuando estamos ante un problema de flujo representado mediante una red bipartita incompleta, lo primero que hay que preguntarse es si existe un flujo que satisfaga las demandas, es decir, si la red es factible.

Para determinar lo anterior puede utilizarse el algoritmo de Flujo Máximo de la siguiente forma:

Se diseña una red auxiliar agregando un nodo inicial y un nodo final (ver la figura 4.2). Del nodo inicial saldrá un arco hacia cada nodo fuente con capacidad igual a la oferta de ese nodo fuente y de cada nodo sumidero saldrá un arco hacia el nodo final con capacidad igual a la demanda del respectivo nodo sumidero. Los arcos presentes en la red original se mantienen en la nueva red, y como la red original es no capacitada, se pondrá como capacidad, un número suficientemente grande. Es este caso, fm representa el flujo máximo que llegó al nodo final. Si el flujo máximo que llega al nodo final es menor que $\sum_{x_j \in T} b_j$ *, el problema es infactible.*

Una vez que ha sido detectada la infactibilidad, podemos obtener información valiosa para su posterior eliminación, a partir de la red final obtenida por el algoritmo de Flujo Máximo.

Esta información es la siguiente:

1. Número mínimo total de unidades ofertas a cambiar en la red.
2. Indicación del nodo fuente que asignará las unidades sobrantes a otro nodo.
3. Indicación de los nodos sumideros a los que se asignarán las unidades faltantes.

Para ilustrar lo anterior, se analizará el siguiente ejemplo (ver la figura 4.1).

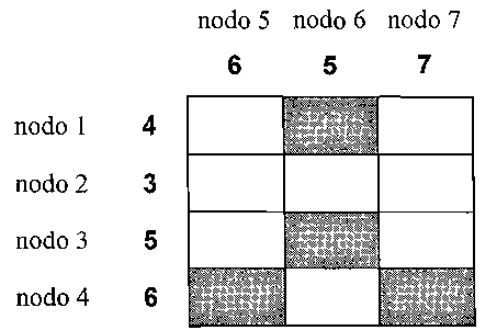

Figura 4.1. Red bipartita incompleta infactible.

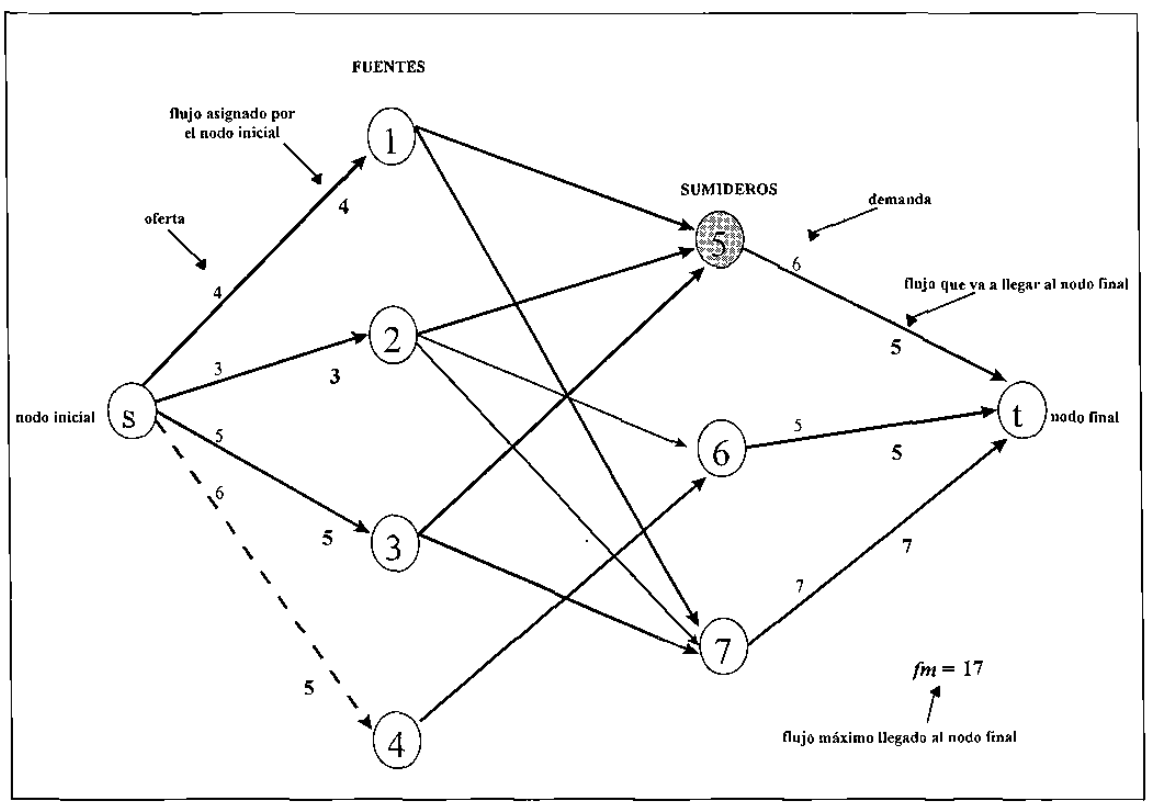

Figura 4.2. Aplicación del algoritmo de Flujo Máximo.

La figura 4.2, muestra la red final obtenida en la aplicación del algoritmo del Flujo Máximo. Se puede observar que al nodo **t** llegó un flujo total de 17 unidades, es decir, una unidad menos que la suma total de las demandas, $\sum b_j=18$. Con este análisis, la cantidad mínima de unidades a cambiar en la red es de *una unidad.*

Puede observarse en la figura 4.2 que el nodo 4 asignará la unidad restante a otro nodo, ya que el flujo asignado al arco fue de 5 unidades y su capacidad es de 6 unidades, por lo tanto, le sobra una unidad. Esta unidad sobrante, será asignada al nodo 5, ya que la capacidad de este nodo es de 6 unidades, y al aplicarse el algoritmo del Flujo Máximo, se le asignó un flujo de 5 unidades, por lo cual, podrá recibir una unidad más. Ahora bien, no podemos saber cuál es el otro nodo interceptor para especificar exactamente entre qué nodos se le asignará el valor.

Aunque el algoritmo de Flujo Máximo proporciona información valiosa, no precisa cómo realizar los ajustes necesarios para obtener un problema factible.

En este trabajo, se propone un algoritmo que *elimina la infactibilidad en problemas de redes*, cumpliendo con el objetivo de satisfacer el 100% de las demandas, realizando ajustes solamente en las ofertas. Esta es su principal ventaja sobre los trabajos anteriores, donde para obtener una red factible, se ajustaban los valores de las demandas, lo cual es poco recomendable en un problema real.

El algoritmo que se describe en esta tesis, realiza ajustes mínimos en los valores de las ofertas, con el fin de convertir un problema infactible a factible y además, satisfacer las demandas requeridas en la red.

El algoritmo desarrollado está basado en técnicas heurísticas, y es denominado "*Algoritmo Heurístico De La Cruz*".

4.3 *Algoritmo heurístico De la Cruz*

Las ventajas del algoritmo heurístico sobre los métodos anteriores son:

- Las demandas de los nodos sumideros no son modificadas durante el proceso.

- Los cambios realizados en los nodos fuentes son menores, en comparación con otras aplicaciones utilizadas, entre ellas, la de Roodman.

Como se explicó anteriormente, para determinar si existe infactibilidad en la red, se utiliza el algoritmo de Flujo Máximo. La infactibilidad es detectada cuando el flujo máximo que llega al nodo final de la red, es menor a la suma total de las demandas de los sumideros, indicando que el problema es *infactible*.

Una vez que se ha detectado que un problema es infactible, la red puede ser procesada por el algoritmo *De la Cruz,* para transformar el problema de red infactible a un problema de red factible, mediante ajustes mínimos en los valores de las ofertas en la red.

Las características básicas del algoritmo heurístico *De la Cruz* son:

- Solo se puede aplicar a problemas definidos sobre redes bipartitas incompletas.

- El algoritmo considera que la red es balanceada.

La manera de trabajar con este algoritmo es sobre una tabla en donde las ofertas se asocian a las filas y las demandas a las columnas.

Antes de describir el algoritmo se definirá el concepto "*oferta truncada*". Durante el proceso de asignación, puede surgir el caso en donde una oferta podrá satisfacer totalmente una demanda de la red, por consiguiente, se asignarán ceros en la columna de dicha demanda satisfecha. Si al realizar la asignación de ceros en la columna, se detecta la existencia de una oferta que aún no ha sido distribuida y que ya cuenta con flujos asignados en sus celdas, con valores de ceros y/o celdas prohibidas, se dice que se encontró una *oferta truncada*. Este término es utilizado en el algoritmo heurístico *De la Cruz*.

Los indicadores utilizados para describir el algoritmo son los siguientes:

A, indica que la oferta fue distribuida sin ningún problema.

&, indica el exceso que hubo en la oferta distribuida.

*, indica que un nodo oferta está en espera de unidades por parte de otro(s) nodo(s) oferta(s).

#, indica la eliminación de una fila de la tabla.

E, indica que la fila fue eliminada.

La idea central del algoritmo es analizar cada oferta de la red en orden decreciente, anteponiendo un indicador a la oferta que ya fue analizada; el indicador indica el estado en que quedó la oferta. Primero se selecciona una fuente a analizar, y se procede a buscar el sumidero al que se asignará un cierto flujo, basándose en el sumidero que tenga un mayor número de celdas prohibidas. Una vez asignado el flujo a la celda correspondiente, se antepondrá un indicador a la fuente que indicará en qué estado quedó. Así, se procederá con cada fuente, hasta que los indicadores de las fuentes analizadas sean un conjunto de indicadores A, o un conjunto de indicadores A y/o E. Finalmente, se obtendrá la nueva red factible a partir de la sumatoria total de los flujos asignados en la misma. A continuación se describe el algoritmo heurístico *De la Cruz*.

4.3.1 Descripción del algoritmo heurístico *De la Cruz*

La entrada de datos al algoritmo es:

- Ofertas de los nodos fuentes,
- Demandas de los nodos sumideros,

- Arcos prohibidos.

Dada una red bipartita incompleta infactible balanceada:

1) Observar si hay una nueva oferta mayor que las demás ofertas de la tabla por analizar que no contenga indicador:

1.1) Sí: Si el valor de la oferta es único, ir al paso 2. Si hay más valores iguales de ofertas con celdas prohibidas:

1.1.1) Seleccionar la oferta que tenga mayor número de celdas prohibidas (si lo hay) para esa fila y verificar si:

1.1.1.1) El número de celdas prohibidas es único, ir al paso 2.

1.1.1.2) Si hay iguales números de celdas prohibidas, tomar cualquier oferta, e ir al paso 2.

1.1.2) Si no hay celdas prohibidas, seleccionar cualquier oferta, e ir al paso 2.

1.2) No: Si todas las ofertas analizadas tienen indicadores *A* o combinación de indicadores *A* y/o *E*, el algoritmo termina y se procede al paso 5. En caso contrario, ir al paso 4.

2) Seleccionar la demanda que tenga mayor número de celdas prohibidas (si lo hay) entre aquellas celdas vacías que pertenecen a la oferta que se analiza:

2.1) Si es la única demanda con el mayor número de celdas prohibidas, ir al paso 3.

2.2) Existe más de una demanda que contiene el mismo número de celdas prohibidas:

2.2.1) Si los valores de las demandas son diferentes, tomar la demanda mayor e ir al paso 3.

2.2.2) Si los valores de las demandas son iguales, tomar cualquier demanda mayor-igual e ir al paso 3.

2.3) Si no hay celdas prohibidas, ir al paso 3 en cualquiera de los 3 casos siguientes: **a)** Seleccionar la única demanda para esa fila; **b)** si hay demandas con diferentes valores, seleccionar la demanda mayor; **c)** si hay varias demandas iguales, seleccionar cualquier demanda-mayor-igual.

3) Se asigna un valor de flujo basándose en:

3.1) Si la demanda es menor que la oferta, si es la última celda de la fila por analizar:

3.1.1) Sí: Se satisface la demanda y se marca a la oferta analizada con el indicador *&*, se disminuye la oferta y la demanda, se colocan ceros en las celdas de la columna satisfecha, y se coloca el indicador # en las ofertas truncadas que resulten al hacer ceros la columna. Posteriormente se procede al paso 1.

3.1.2) No: Se satisface la demanda, se disminuye la oferta y la demanda, se colocan ceros en las celdas de la columna satisfecha (si es necesario), y se coloca el indicador # en las ofertas truncadas que resulten al hacer ceros la columna. Finalmente se procede al paso 2.

3.2) Si la demanda es mayor que la oferta, se verifica si es la única celda por asignar en esa columna:

3.2.1) Sí: Checar si la oferta tiene marcado un indicador de *:

3.2.1.1) Sí: ir al paso 4.

3.2.1.2) No: se marca la oferta analizada con el indicador * y se reinicializan los flujos que fueron asignados a la oferta (si es que hubo); ir al paso 1.

3.2.2) No: Se satisface completamente la demanda en base a las unidades restantes de la oferta; se disminuye la oferta y demanda, se colocan ceros en las celdas restantes de la fila (si es necesario) y se coloca el indicador *A* al lado de la oferta analizada; ir al paso 1.

3.3) Si la demanda es igual a la oferta, se verifica si es la única celda de la fila por asignar, o si la fila tiene demandas que pueden ser satisfechas por otras ofertas; si este es el caso, se asigna la demanda totalmente y se marca la oferta con el indicador *A*, se disminuye la oferta y demanda, y se colocan ceros en las celdas de la columna satisfecha (si es necesario), verificando si existen ofertas truncadas. Si no se cumple las condiciones anteriores, se marca la oferta con el

indicador *, y se procede al paso 1. En el proceso, las ofertas truncadas, se marcan con el indicador #.

4) Existe alguna oferta marcada con el indicador #:

4.1) Sí: Asignar el valor de la oferta que contiene el indicador # a la oferta con el indicador "*"; posteriormente, se elimina la fila y se cambia el indicador # por una *E*. Si no hay ofertas con el indicador *, se cambia la primera oferta con indicador # por el indicador * y se procede al paso 1.

4.2) No: Hay filas marcadas con el indicador *&*:

4.2.1) Sí: Se asignan sus unidades sobrantes a la primera oferta con indicador *; posteriormente, se cambian las ofertas asignantes (indicador *&*) por el indicador *A*, eliminando el indicador * de la oferta. Se procede al paso 1.

4.2.2) No: Se elimina el indicador * y se procede al paso 1.

5) La nueva red se obtiene aplicando los siguientes criterios:

- La nueva oferta de cada nodo fuente será igual a la suma de los flujos asignados en su fila correspondiente.

- La nueva demanda de cada nodo sumidero será igual a la suma de los flujos asignados en su columna correspondiente.

4.3.2 Diagrama del algoritmo heurístico *De la Cruz*

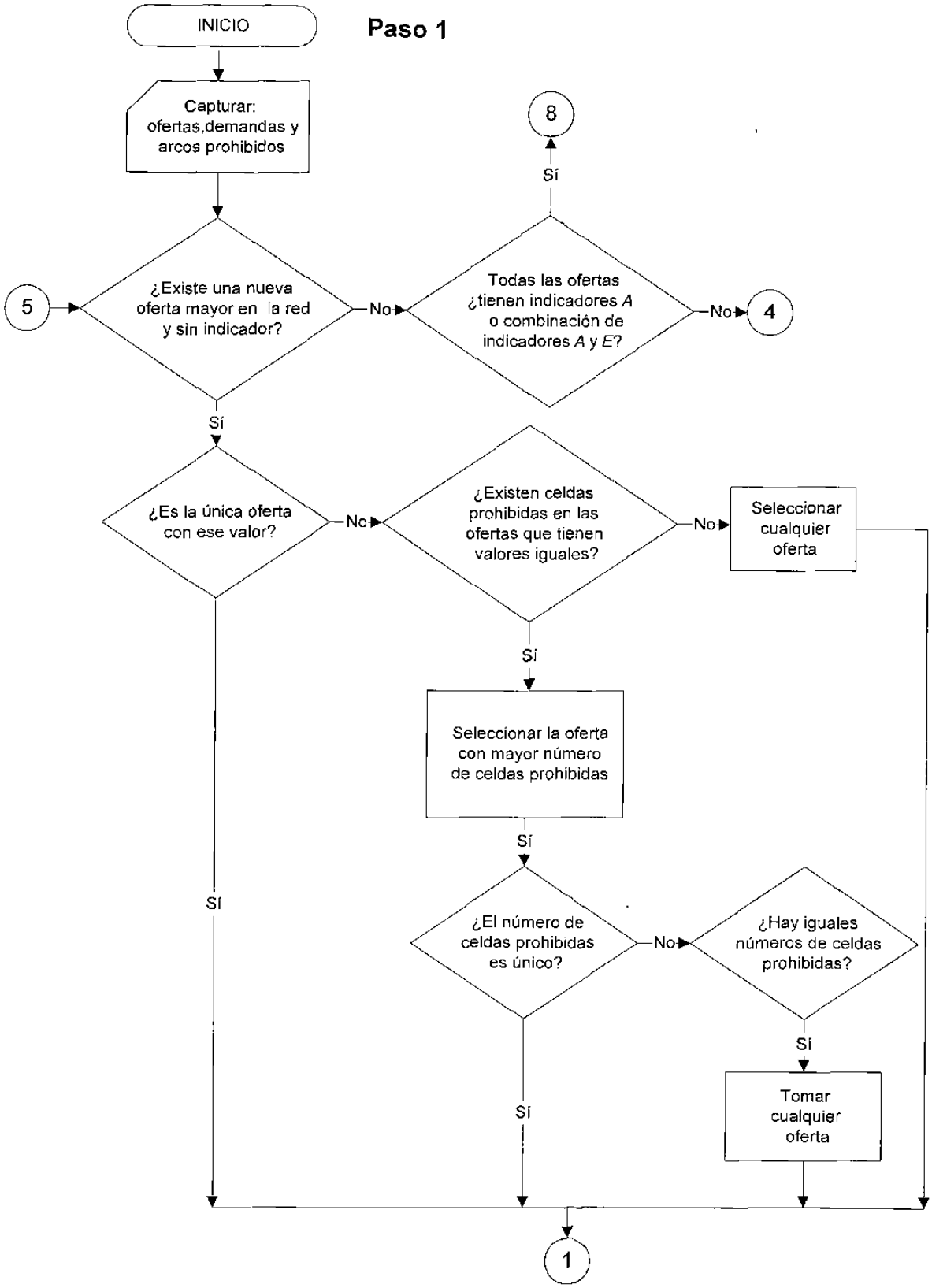

Figura 4.3. Diagrama del algoritmo heurístico *De la Cruz* en el paso 1.

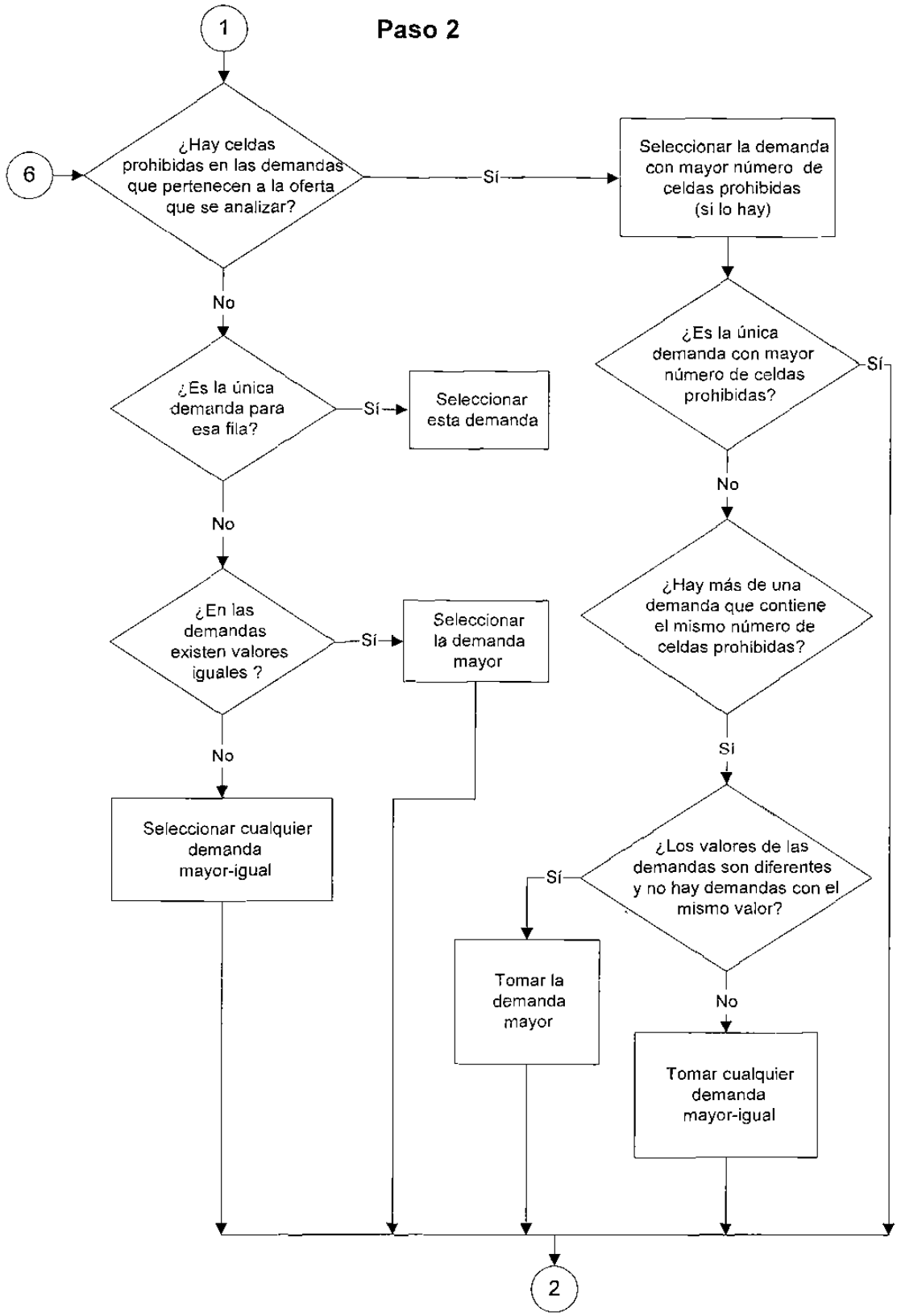

Figura 4.4. Diagrama del algoritmo heurístico *De la Cruz* en el paso 2.

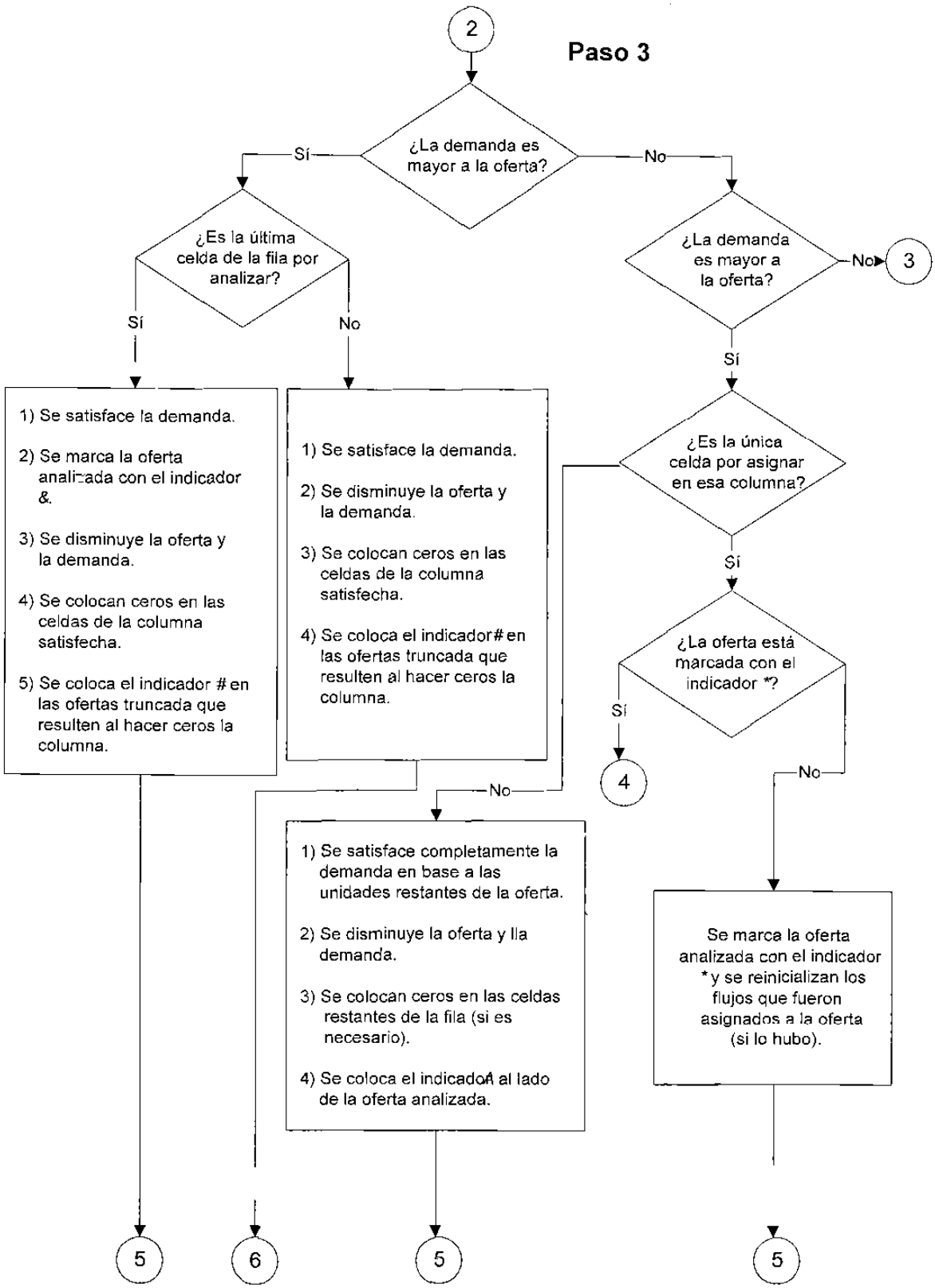

Figura 4.5. Diagrama del algoritmo heurístico *De la Cruz* en el paso 3.

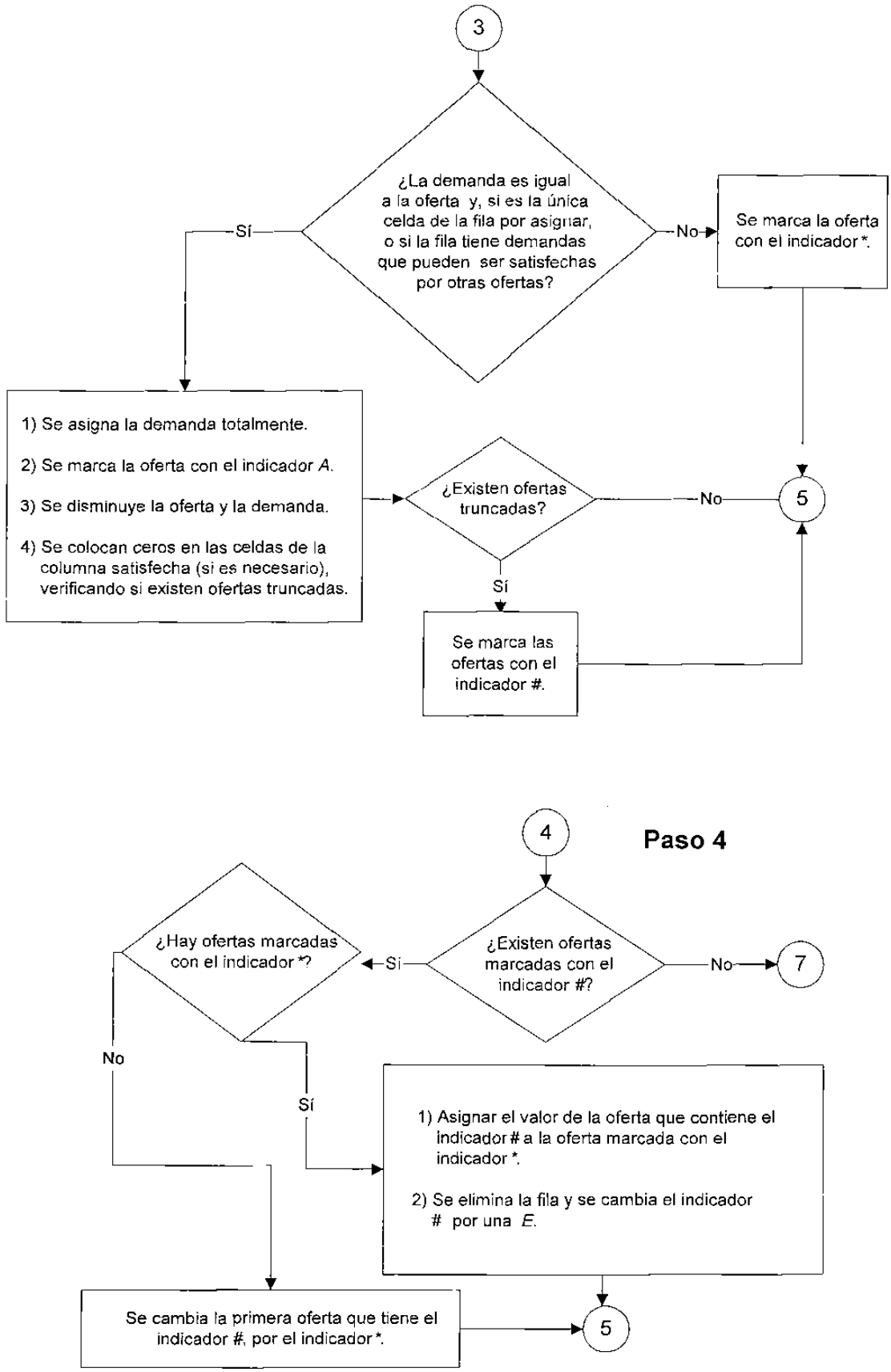

Figura 4.6. Diagrama del algoritmo heurístico *De la Cruz* entre los pasos 3 y 4.

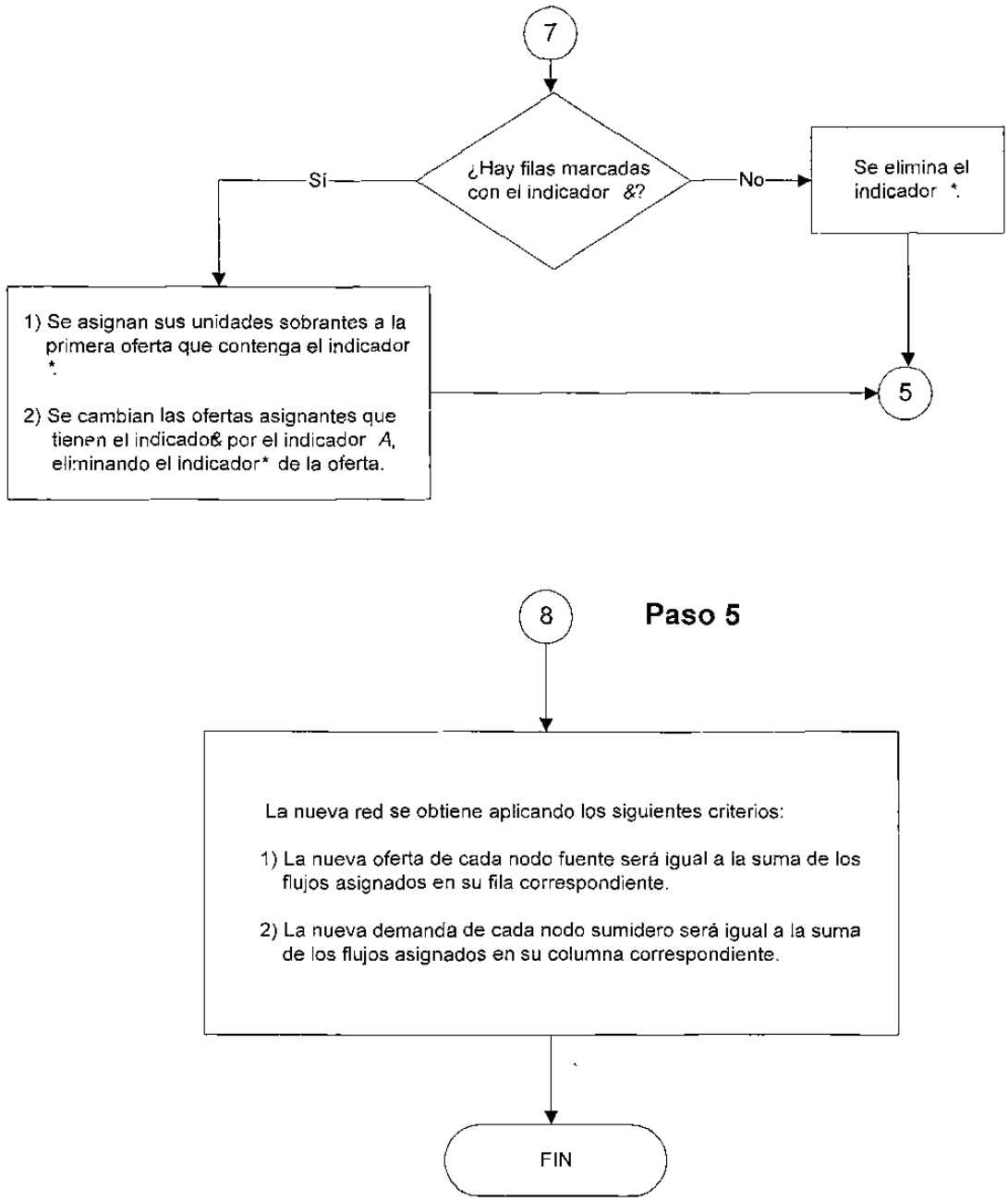

Figura 4.7. Diagrama del algoritmo heurístico *De la Cruz* entre los pasos 4 y 5.

4.3.3 Caso de análisis

A continuación se describe un problema de flujo en redes para ejemplificar la aplicación del algoritmo heurístico *De la Cruz*. Este ejemplo ilustra las ventajas y desventajas que tiene el método de Currin en comparación con el algoritmo propuesto.

- Considere la siguiente red bipartita incompleta, cuya infactibilidad ya fue detectada por el algoritmo de Flujo Máximo (ver la figura 4.2).

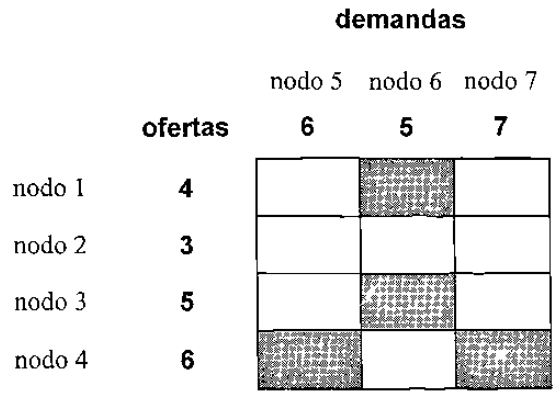

Figura 4.8. Red bipartita incompleta infactible.

- Se selecciona el nodo con la oferta mayor (nodo 4) y se asigna el flujo en la celda con mayor número de celdas prohibidas (nodo 6). El flujo asignado entre el nodo 4 y 6 es de 5 unidades, ya que la demanda requiere de sólo 5 unidades; por lo tanto, el sobrante de la oferta es de una unidad y se representa con el indicador de &.

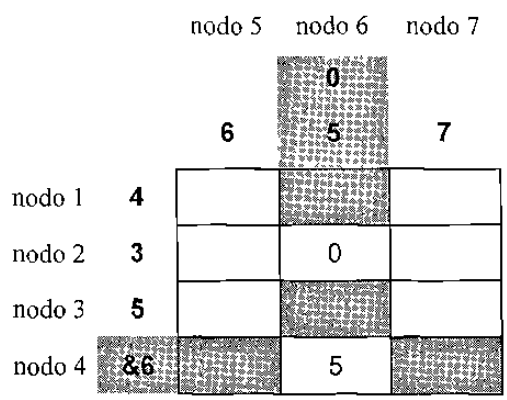

Figura 4.9. Asignación de unidades por parte del nodo 4 al nodo 6.

- Después, se toma la siguiente oferta mayor (nodo 3) y se asigna el flujo en la celda con mayor número de celdas prohibidas, que en este caso serían los nodos 5 y 7. El criterio para seleccionar un nodo en el caso de que existan igual demandas con el mismo número de celdas prohibidas, es seleccionar la demanda mayor (nodo 7). Se

asignamos el flujo entre los nodos 3 y 7 basándose en la demanda requerida; en este caso la demanda es mayor a la oferta y tiene celdas desocupadas, por lo que se asigna el valor total de la oferta. Entonces, el nodo 7 recibe un flujo de 5 unidades y se marca la oferta analizada con el indicador *A*.

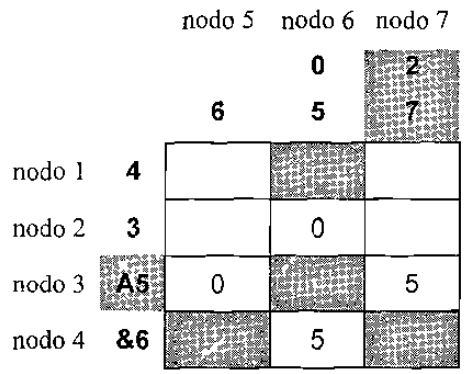

Figura 4.10. Asignación de unidades por parte del nodo 3 a los nodos 5 y 7.

- Se toma el siguiente nodo con la oferta mayor (nodo 1) y se asigna el flujo en la celda con mayor número de celdas prohibidas (nodos 5 y 7); en este caso se selecciona el nodo 5. Se asigna como flujo entre los nodos 1 y 5 el valor de la demanda requerida, pero como la demanda es mayor a la oferta y además tiene celdas desocupadas en la columna de la demanda, se le asigna el valor total de la oferta, se marca la oferta con un indicador *A* y se procede a analizar la siguiente oferta mayor.

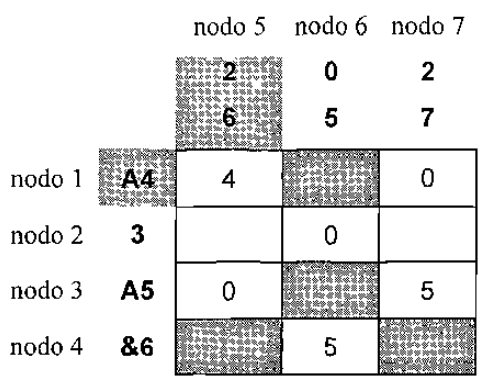

Figura 4.11. Asignación de unidades por parte del nodo 1 a los nodos 5 y 7.

- La siguiente oferta mayor es el nodo 2 y se asigna el flujo en la celda con mayor número de celdas restringidas; en este caso se selecciona el nodo 7, ya que cuando existe un empate en los valores de las demandas, se selecciona cualquier demanda mayor-igual. A continuación, se debe asignar el flujo entre los nodos 2 y 7 basándose en la demanda requerida, sin embargo, como aún se debe asignar el flujo para el nodo 5, existe el problema de que la oferta no puede cubrir con la demanda especificada; por tal motivo, se marca la oferta con indicador *.

		nodo 5	nodo 6	nodo 7
		2	**0**	**2**
		6	**5**	**7**
nodo 1	**A4**	4		0
nodo 2	*3		0	
nodo 3	**A5**	0		5
nodo 4	**&6**		5	

Figura 4.12. Asignación de unidades por parte del nodo 2 a los nodos 5 y 7.

- Una vez que todas las ofertas han sido analizadas, se asignan las unidades sobrantes, o aquellas ofertas con indicador *&*, a las ofertas con el indicador *; cambiando el indicador *&* por *A*, y eliminando el indicador * de la oferta. El nodo 2 recibe una unidad del nodo 4 completando un total de unidades de 4. Analizando nuevamente la oferta sin indicador del nodo 2, se asigna el flujo correspondiente a cada demanda de la red, colocando el indicador *A* a la oferta analizada.

		nodo 5	nodo 6	nodo 7
		0	**0**	**0**
		6	**5**	**7**
nodo 1	**A4**	4		0
nodo 2	**A4**	2	0	2
nodo 3	**A5**	0		5
nodo 4	**A5**		5	

Figura 4.13. Red bipartita incompleta factible con flujos asignados.

En este punto se ha eliminado el problema de infactibilidad en la red, ya que todas las ofertas tienen asignado un indicador *A*. El algoritmo termina definiendo un nuevo modelo de red (ver la figura 4.14).

		nodo 5	nodo 6	nodo 7
		6	**5**	**7**
nodo 1	**4**			
nodo 2	**4**			
nodo 3	**5**			
nodo 4	**5**			

Figura 4.14. Nuevo modelo de red solucionado por el algoritmo heurístico *De la Cruz*.

Analizando el nuevo modelo de red, se puede observar que solo una unidad de oferta fue modificada; además, las demandas quedaron invariantes.

Si este ejemplo fuera solucionado por el método de Currin, el número de unidades modificadas serían de 2 (ver la figura 4.15).

Si se aplicara el método de Currin al ejemplo anterior, el número de unidades cambiadas serían de 2 (véase la figura 10).

		nodo 5	nodo 6	nodo 7
		6	**5**	**5**
nodo 1	**3**	3		0
nodo 2	**3**	3	0	0
nodo 3	**5**	0		5
nodo 4	**5**		5	

Figura 4.15. Modelo de red final solucionado por el método de Currin.

En este ejemplo se puede observar la ventaja que tiene el algoritmo heurístico *De la Cruz,* en comparación con el método de Currin. La ventaja de este nuevo algoritmo es

que sólo afecta las ofertas, cumpliendo con el objetivo de satisfacer completamente las demandas requeridas.

4.4 *Conclusiones del capítulo*

La aplicación del algoritmo heurístico *De la Cruz* lleva a la eliminación de la infactibilidad en un problema de flujos sobre redes bipartitas incompletas infactible. Su objetivo es satisfacer totalmente los requerimientos de las demandas de la red, lo cual solo requiere realizar un ajuste mínimo en los valores de las ofertas.

Antes de aplicar el algoritmo heurístico *De la Cruz*, es necesario detectar un problema de infactibilidad en la red. En el desarrollo de este trabajo se utilizó el algoritmo de Flujo Máximo para realizar esta función.

CAPÍTULO 5

PRESENTACIÓN DEL SISTEMA Rosma

5.1 *Introducción*

En éste capítulo se describe el sistema *Rosma*, que detecta y elimina el problema de infactibilidad en una red bipartita incompleta. El sistema *Rosma* aplica el algoritmo de Flujo Máximo para la detección de infactibilidades, e implementa el algoritmo heurístico *De la Cruz* para proponer un nuevo modelo de red factible.

Se indican las limitaciones actuales del sistema y se propone alternativas para mejorar el desempeño del sistema.

Se muestran los resultados obtenidos de la utilización del sistema en la detección y eliminación de infactibilidades en problemas de redes bipartitas incompletas infactibles utilizando varios ejemplos que ilustren el funcionamiento del sistema. Finalmente, se hace mención de las ventajas y desventajas que se encontraron al aplicar técnicas alternativas para eliminar la infactibilidad en problemas bipartitas incompletas infactibles en comparación con el algoritmo heurístico *De la Cruz*.

5.2 *Descripción de los módulos del sistema Rosma*

El sistema ROSMA fue desarrollado en Lenguaje C, en ambiente DOS. La programación se realizó en base a operaciones matriciales.

Puede ser ejecutado bajo cualquier estándar de computadora, ya que se cuenta con el archivo ejecutable. El archivo se llama *delacruz*.

El sistema consta de dos módulos, el primero detecta la infactibilidad y el segundo la elimina:

1) El módulo que detecta la infactibilidad (llamado **MDI**) se basa en la aplicación del algoritmo del Flujo Máximo, implementado bajo el software Tora [Taha, 1990]. Los datos de entrada son las ofertas de los nodos fuentes, las demandas de los nodos sumideros y los arcos prohibidos de la red; el análisis se realiza asumiendo que la red es balanceada:

   a) Si el problema es factible, el Flujo Máximo será igual a la sumatoria total de ofertas y demandas, y la distribución de flujo puede determinarse en forma directa.

   b) Si el Flujo Máximo es menor a la suma total de las demandas significa que el problema es infactible y se procede a aplicar el algoritmo heurístico *De la Cruz*, implementado en el módulo 2.

2) El módulo que elimina la infactibilidad (llamado **MEI**) se basa en el algoritmo heurístico *De la Cruz* descrito en el capítulo 4, cuyo objetivo es realizar ajustes mínimos en los valores de las ofertas, y respetando las demandas.

5.3 *Limitaciones*

El sistema *Rosma* solo puede solucionar problemas de redes que sean representados por matrices con una dimensión de 100 x 100, ya que la versión del lenguaje C está restringída a 16 bits por carácter. No obstante, esta limitación puede ser resuelta emigrando el código fuente del sistema *Rosma* a una versión de Lenguaje C de 32 bits, con lo cual la única restricción sería la memoria disponible cuando se ejecute el sistema.

5.4 *Caso de análisis*

A continuación se presentan tres casos de redes bipartitas incompletas utilizadas para analizar el desempeño del sistema *Rosma.*

Problema No.1

Modificar la siguiente red bipartita incompleta para obtener su factibilidad:

		demandas			
		nodo 4	nodo 5	nodo 6	nodo 7
	ofertas	**3**	**13**	**36**	**14**
nodo 1	**42**				
nodo 2	**22**				
nodo 3	**2**				

Figura 5.1. Red bipartita incompleta infactible.

El sistema se ejecuta con el nombre de "delacruz" y aparece la siguiente ventana:

SISTEMA "ROSMA"

Seleccione el módulo :

1. **DETECCIÓN DE INFACTIBILIDAD EN LA RED (MDI)**
2. **ELIMINACIÓN DE INFACTIBILIDAD EN LA RED (MEI)**

Figura 5.2. Menú principal del sistema *Rosma*.

Para detectar la infactibilidad de la red, se selecciona el módulo correspondiente a la "Detección de infactibilidad en la red", ejecutádose el programa Tora. La secuencia del análisis de infactibilidad utilizando el programa Tora se describe en la figura 5.3 a la figura 5.11.

Figura 5.3. Pantalla de presentación del programa Tora.

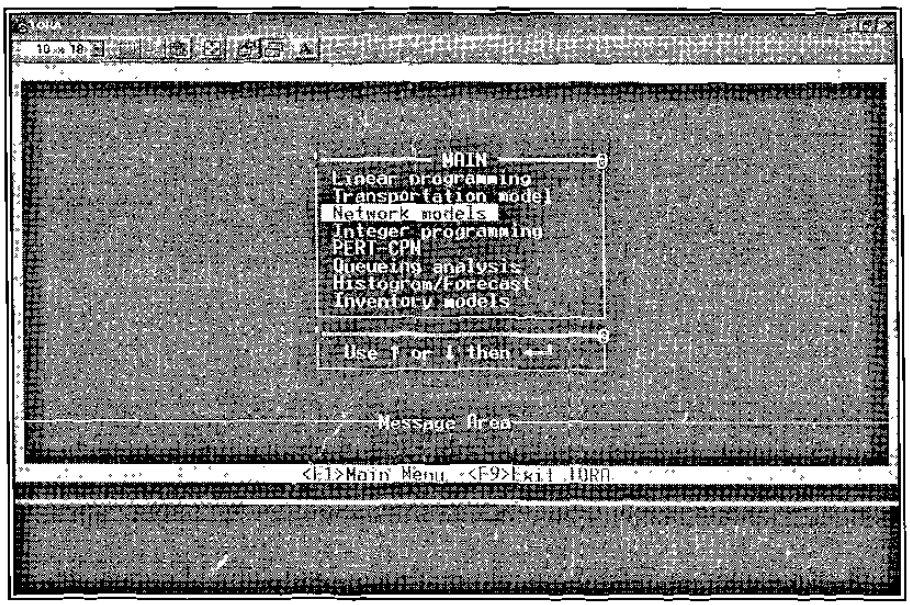

Figura 5.4. Menú principal del programa Tora.

- De acuerdo a nuestro problema, se selecciona la opción "Modelos de Redes".

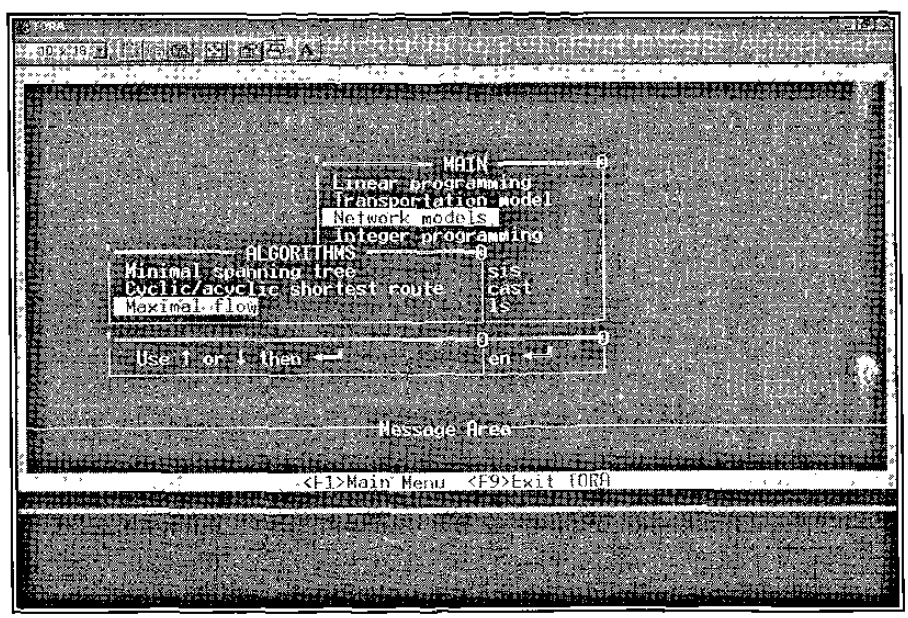

Figura 5.5. Submenú de "Modelos de Redes" del programa Tora.

- Se selecciona la opción Flujo Máximo ya que la infactibilidad de la red se detecta por medio de este algoritmo.

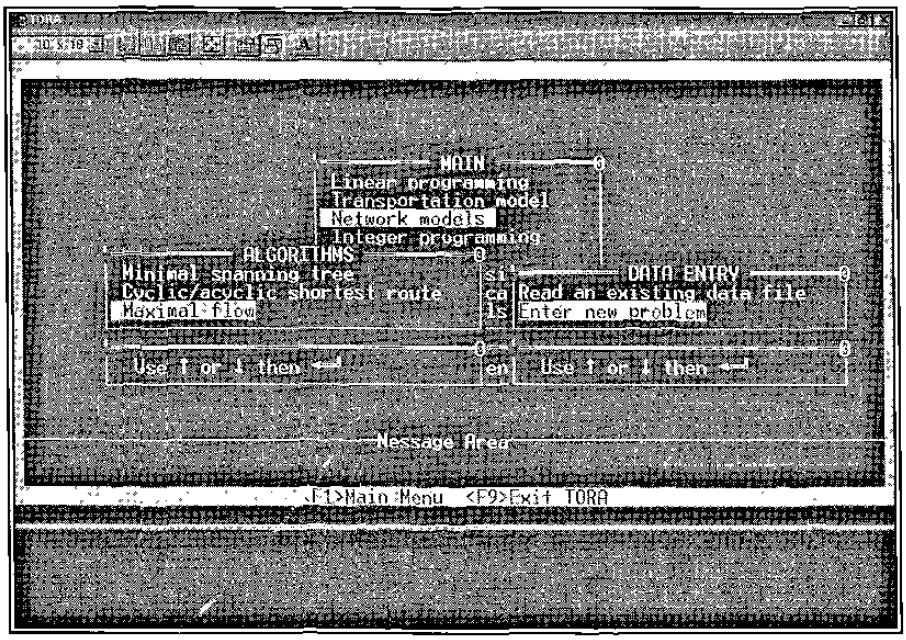

Figura 5.6. Submenú del algoritmo de "Flujo Máximo".

- Se diseña un nuevo problema de red, por lo cual se selecciona la opción "Realizar un nuevo problema".

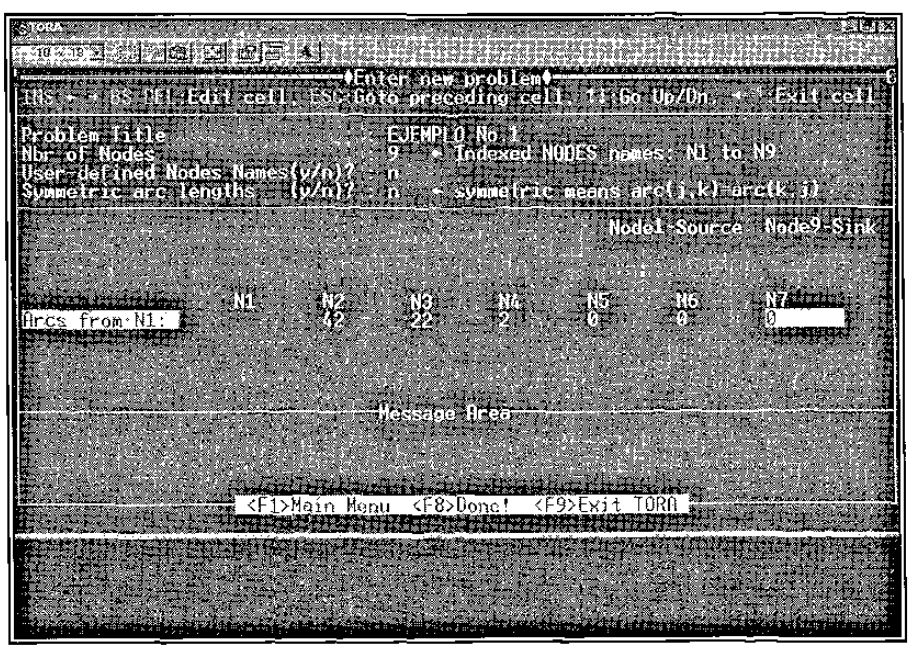

Figura 5.7. Área de trabajo para la modelación de redes.

- Se capturan los datos necesarios; entre ellos, el número de nodos que conforma la red y sus conexiones con los demás nodos.

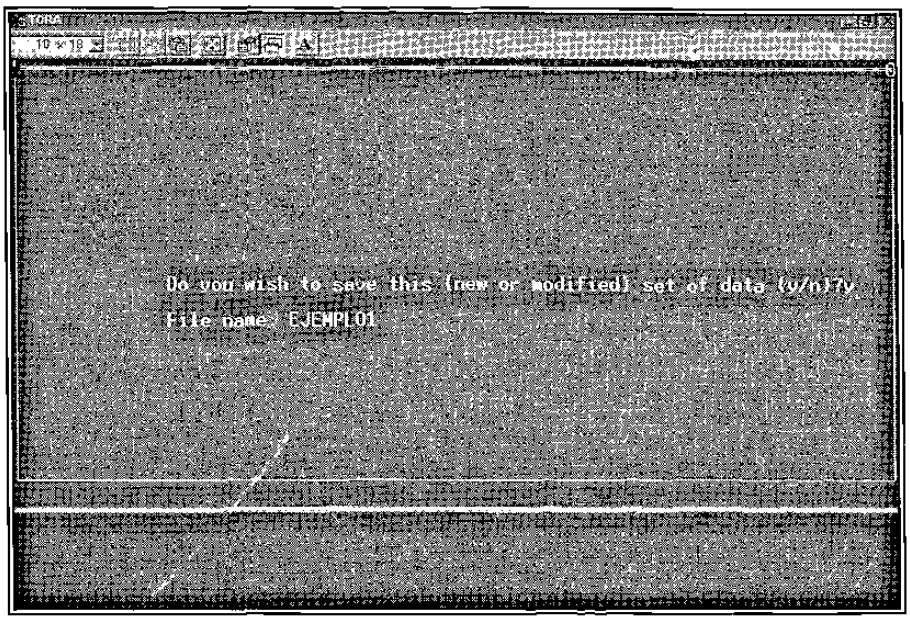

Figura 5.8. Pantalla para guardar el nuevo diseño de red.

- Se guarda el diseño de la red con algún nombre de archivo.

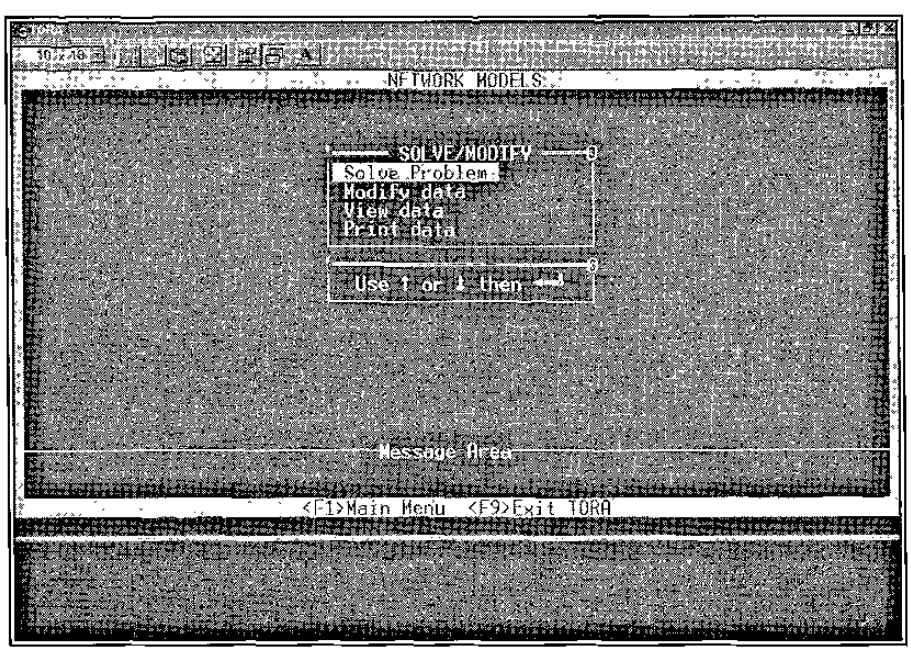

Figura 5.9. Menú para solucionar el modelo de red.

- Por medio de la opción "Resolver el problema", se da la orden para ejecutar el algoritmo del Flujo Máximo.

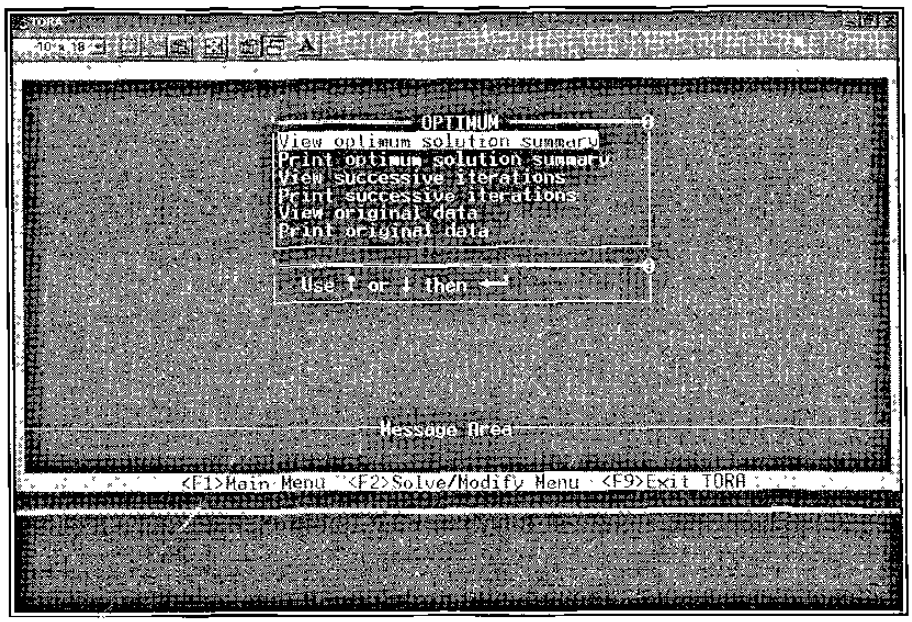

Figura 5.10. Menú de Resultados

- Para observar si la red es factible o infactible, se selecciona la opción "Ver resumen de la solución óptima".

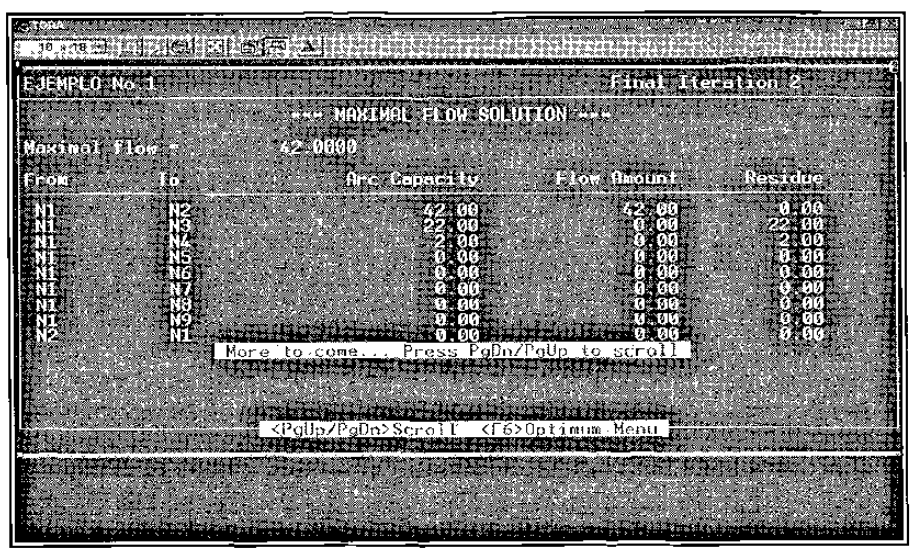

Figura 5.11. Solución de la red.

La red es infactible, ya que la suma de los nodos demandas es de 66 y en base al algoritmo del Flujo Máximo, solo se cuenta con un flujo máximo de 42. Para salir al Menú Óptimo, presione la tecla F6.

Una vez detectada la infactibilidad en la red, se ejecuta el módulo de "Eliminación de infactibilidad en la red" con el fin de proporcionar un nuevo diseño de red factible.

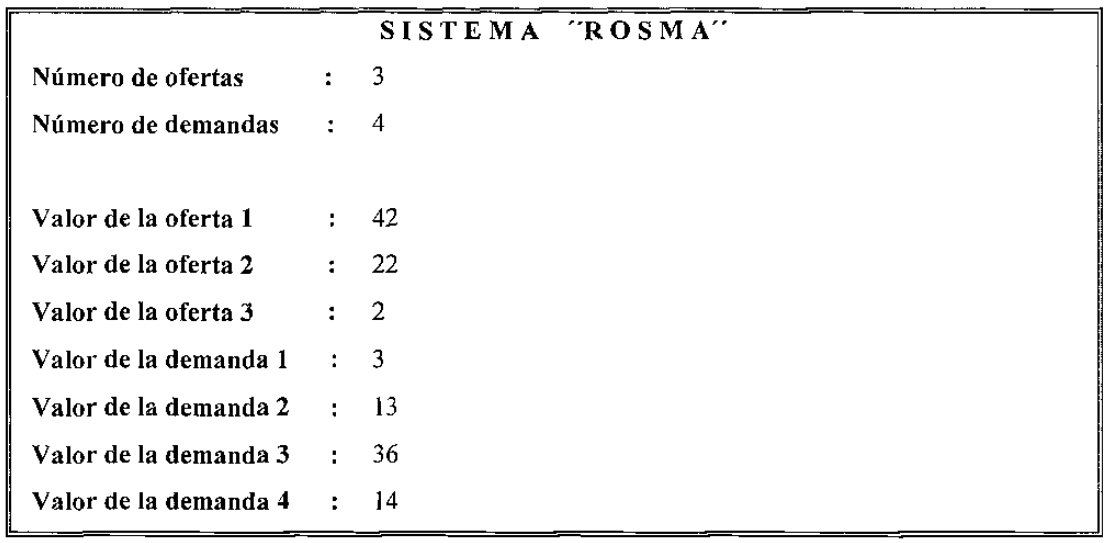

Figura 5.12. Captura de valores a los nodos de la red.

- Se proporcionan los datos necesarios para el diseño de la red infactible.

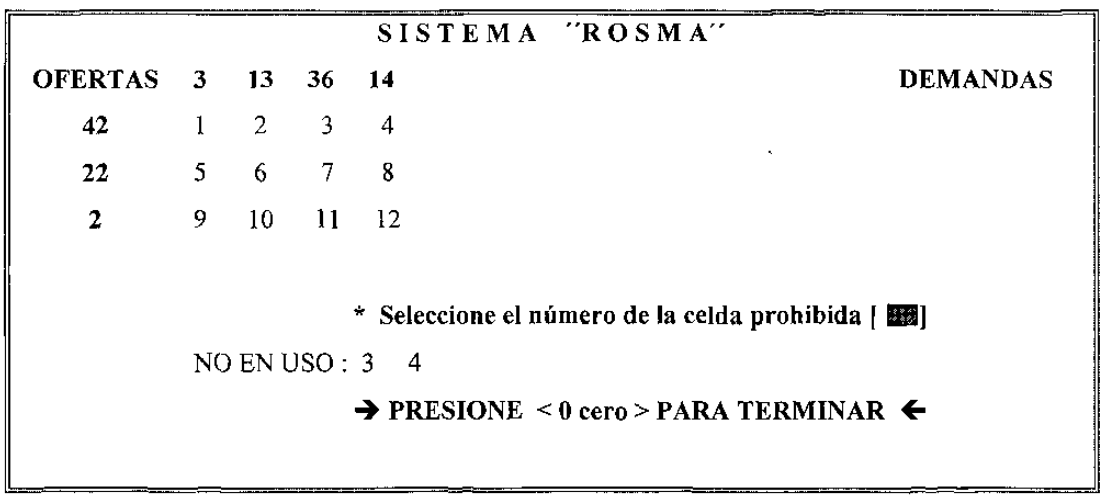

Figura 5.13. Modelo de red sin celdas prohibidas.

- Se indican los números que corresponden a las celdas prohibidas en la red.

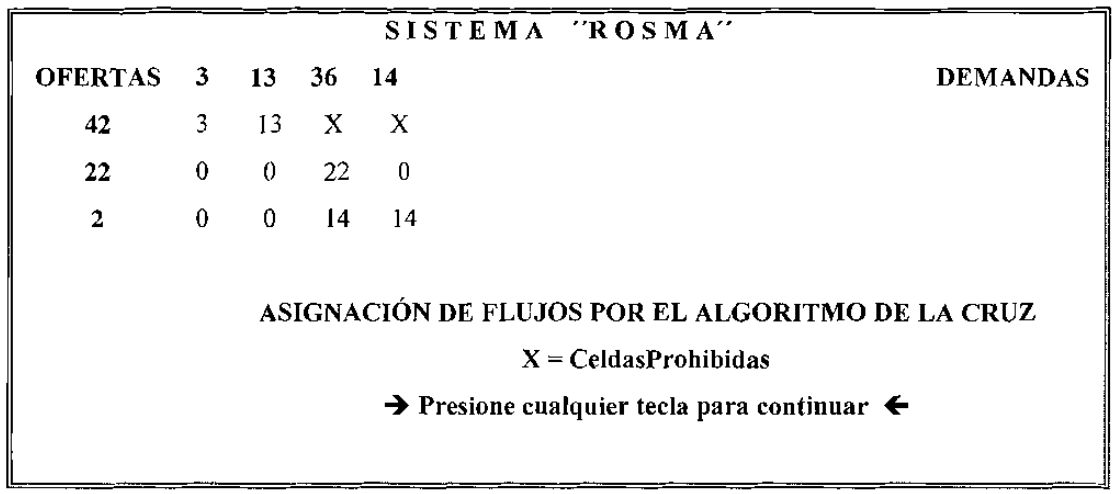

Figura 5.14. Asignación de flujos por el algoritmo *De la Cruz*.

- En la figura anterior, se muestran los flujos asignados en la red por medio del algoritmo *De la Cruz*.

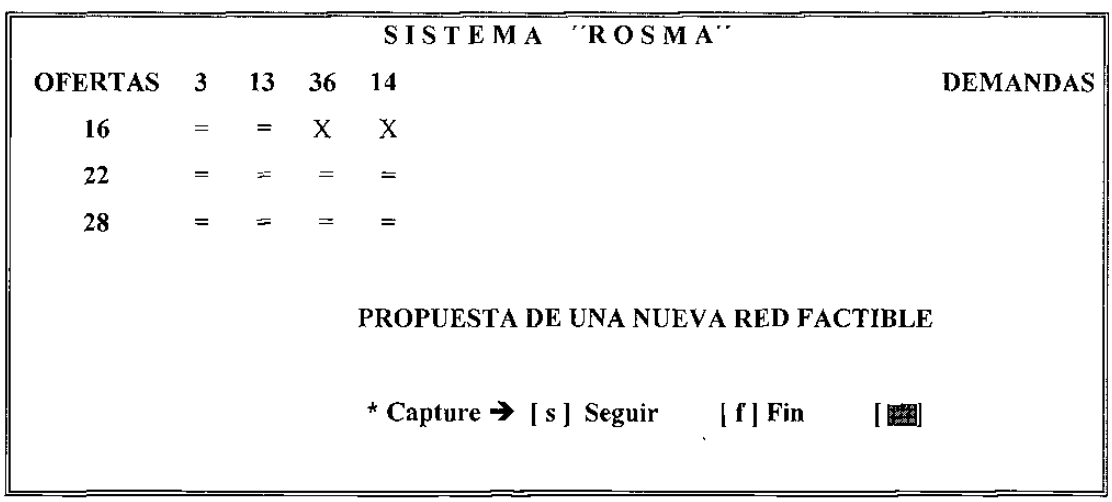

Figura 5.15. Nueva red bipartita incompleta factible.

Finalmente, puede observarse que en la nueva red factible (ver la figura 5.15) solamente las ofertas fueron modificadas y las demandas fueron respetadas en comparación con la red original (ver la figura 5.3).

Con esto, se concluye la eficiencia del algoritmo heurístico para eliminar la infactibilidad en redes bipartitas incompletas.

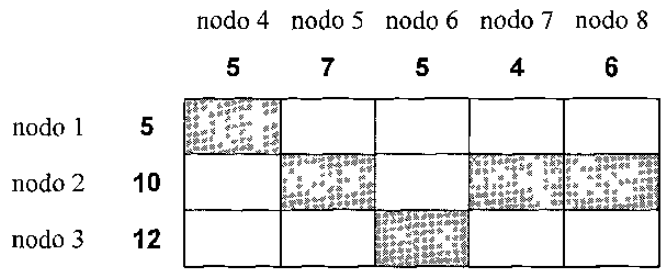

Figura 5.18. Red bipartita incompleta factible.

En la figura 5.17 se muestra una red bipartita incompleta infactible, donde al aplicarse el algoritmo heurístico propuesto en esta tesis, proporciona la nueva red factible mostrada en la figura 5.18 respetándose sus demandas.

5.5 *Resultados*

Para la verificación y comprobación del objetivo del sistema, se resolvieron 568 problemas de redes bipartitas incompletas infactibles, obteniéndose nuevos diseños factibles.

El tiempo que tarda el sistema *Rosma* en modificar una red infactible a factible una vez que se han proporcionado todos los datos de la red, depende de la cantidad total de nodos que está formada dicha red. Por ejemplo, si se tratara de modificar una red de 4 x 4, se ejecutará en menos de un segundo; en cambio, cuando se modelan redes de 10 x 10, el tiempo de ejecución es de 2 a 3 segundo, y así sucesivamente.

5.5.1 Comparación con otros métodos

En cuanto al módulo MEI, los resultados obtenidos fueron muy significativos, ya que el número de unidades cambiadas en las ofertas fue mínimo en comparación con los

otros métodos que también eliminan dicha infactibilidad, entre ellos los desarrollados por Currin, Klingman y Roodman.

En la aplicación del método de Currin en estos mismos ejemplos, se observó que el número de unidades cambiadas fue de 1 unidad más.

Otra desventaja del método Currin es que efectúa cambios tanto en las demandas como en las ofertas (en algunas redes) para obtener la factibilidad de la red. Sin embargo, el algoritmo propuesto de este trabajo satisface al 100% las demandas requeridas por el cliente.

También, se analizó el método propuesto por Roodman, y se observó que éste requiere modificar un gran número de unidades en comparación con los algoritmos de Currin y *De la Cruz*. El número de unidades a cambiar es 3 veces mayor que las unidades que se deben cambiar por *De la Cruz* y por Currin.

Otra desventaja importante del algoritmo de Roodman es que al igual que el de Currin, realiza ajustes tanto en las ofertas como en las demandas de la red para resolver el problema de infactibilidad.

La aplicación del método de Klingman para eliminar la infactibilidad en redes, se basa en realizar ajustes en las demandas de la red minimizando las modificaciones que puedan surgir en las ofertas. Para lograrlo, se enfoca en tres criterios (en orden de prioridad):

1) Minimizar la suma de las rutas infactibles.

2) Minimizar la máxima desviación fraccional entre la baja-oferta para los nodos demandas.

3) Minimizar los costos totales de transportación.

Bajo estos criterios, el método de Klingman tiene ventaja sobre el algoritmo de Currin, ya que proporciona más alternativas de soluciones óptimas en un tiempo corto y con costos mínimos.

En cambio, las desventajas del algoritmo de Klingman en comparación con el algoritmo *De la Cruz* son: en primer lugar, el ajuste que hace en las demandas de la red para lograr la factibilidad del problema; y en segundo lugar, el tiempo que tarda en dar una solución factible a la red.

Por lo tanto, el algoritmo *De la Cruz* presenta importantes ventajas en solución de infactibilidades en redes bipartitas incompletas en comparación de otros algoritmos.

5.6 *Conclusiones del capítulo*

El sistema *Rosma* puede desarrollarse bajo cualquier ambiente y lenguaje de programación, siempre y cuando se esté consciente de los cambios de restricciones o condiciones que esto implique.

Este sistema puede ser mejorado; por ejemplo, el módulo MDI que implementa el algoritmo del flujo máximo, mediante el programa Tora, puede programarse en forma independiente, con la opción de seleccionar diferentes algoritmos para la detección de infactibilidades.

El sistema permitió evaluar el algoritmo *De la Cruz* mediante la solución de 568 casos de redes bipartitas incompletas infactibles, obteniendo resultados óptimos en cada uno de ellos, es decir, los modelos de redes factibles generados, satisfacían las demandas con un número mínimo de modificaciones en las ofertas de la red en comparación con Currin, Klingman y Roodman.

En los ejemplos desarrollados, se evidenció un mejor comportamiento del algoritmo *De la Cruz* en comparación con el algoritmo de Currin, Klingman y Roodman, ya que el número de unidades afectadas fue menor y además no se realizaron cambios en las demandas.

CAPÍTULO 6

CONCLUSIONES Y RECOMENDACIONES

En base al enfoque de "*satisfacer las demandas requeridas*", los métodos encontrados en la literatura presentan serias desventajas; no obstante, en base al enfoque "*ofrecer lo que tenemos*", es un criterio más flexible para establecer un nuevo modelo de red factible, pero con la desventaja de afectar las demandas de los usuarios.

El algoritmo heurístico *De la Cruz* tiene una mayor ventaja sobre los métodos analizados al convertir un problema de red bipartita infactible a factible, ya que ninguno de estos métodos tiene por objetivo el satisfacer completamente las demandas requeridas en la red. Currin, Roodman, Klingman, realizan ajustes en las demandas y ofertas, mientras que *De la Cruz*, realiza sólo ajustes en los valores de las ofertas, sin afectar los valores de las demandas de la red.

Es conocido que al algoritmo de Flujo Máximo puede utilizarse para detectar infactibilidad en una red pero además, proporciona información valiosa y eficiente para

eliminar dicha infactibilidad. Por ejemplo, la cantidad mínima de unidades a cambiar, que es precisamente la cantidad de unidades cambiadas en el algoritmo *De la Cruz*. Esto se obtiene verificando cuántas unidades faltaron para que el problema llegara a ser factible, es decir, las unidades que faltaron para cumplir la condición $fm=\sum b_j$. Por tanto, el algoritmo *De la Cruz* tendrá que cambiar $\sum b_j - fm$ unidades.

El algoritmo de Flujo Máximo también muestra el nodo que proporcionará su sobrante a un nodo que lo necesita; este nodo coincide con el nodo que asigna su sobrante en el algoritmo *De la Cruz*, además, el flujo máximo proporciona información acerca de los nodos a los cuales que les serán asignados las unidades faltantes de otros nodos, aspecto que coincide con el algoritmo *De la Cruz*. No obstante, el algoritmo del flujo máximo no determina el nodo o nodos interceptores para realizar la asignación del flujo.

En base a los resultados obtenidos, se recomienda continuar investigando en los siguientes aspectos, a fin de mejorar el desempeño del algoritmo heurístico *De la Cruz* y del sistema *Rosma*:

1. Demostrar la optimalidad del algoritmo *De la Cruz*, en el sentido de que realiza un número mínimo de variaciones.

2. Adaptar el algoritmo en el caso en que haya capacidades en los arcos o costos asociados.

La aportación que se efectuó en este trabajo de tesis es el desarrollo de un nuevo método que elimina la infactibilidad en problemas de redes bipartitas incompletas infactibles, ajustando mínimamente las ofertas de la red con el fin de respetar sus demandas y satisfacer completamente con sus demandantes.

REFERENCIAS

Ahuja, V., *Design and Analysis of Computer Communication Network,* McGraw-Hill International Editions Computer Science series, 1985.

Álvarez, A. M., *"Algunos Problemas de Síntesis de Redes del Tipo de Transporte"*, Tesis Doctoral, Universidad Central de las Villas, Cuba, 1993.

Álvarez y Martínez, XXVIII Congreso Nacional de la Sociedad Matemática Mexicana, Colima, Colima, México, 1995.

Assad, A.A., "Multicommodity Network Flows -A Survey", Networks, Vol. 8, pp. 37-91, 1978.

Bazaraa, M.S., Jarvis, J.J. and Sherali, H. D., *Linear Programming and Network Flows*, second edition, John Wiley & Sons, 1990.

Currin, D. C., *J. Operation Research Society*, National Research Institute for Mathematical Sciences, South África, Vol. 37, No. 4, pp 387-396, 1986..

Currin, D., "Transportation Problem with Inadmisible Routes". *Journal of the Operational Research Society.* 37, 383-396, 1986.

Dantzing, G., "Linear Programming and Extensions". *Editorial Pricenton,* New Jersey, 1963.

Emelichev, V., "Poliedros de Transporte de Estructura Especial". *Voporosi Kiberetiki.* 64, 29-47, 1980.

Ford L.R., and Fulckerson, *Flow in Networks,* Princeton University Press, Princeton, N. J., 1962.

Guardado, M., "Síntesis de Redes de Tipo de Transporte con Restricciones del Número de Aristas de Salida". *Investigación Operacional.* 1, 3-12, 1986.

Klingman Darwin and Phillips Nancy V, *J. Operation Research Society*, Vol. 39, No. 8, pp 735-742, 1988.

Klinfman, D., "Equitable Demand Adjustment for Infeasible Transportation Problems". *Journal of the Operations Research Society*. 8, 735-742, 1988.

Kravtsor, M., "Problemas de Combinatoria Poliédrica en Problemas de Transporte con Prohibiciones". *Kibernética*. 6, 63-71, 1990.

Kravstor, M., "Sobre la Estructura del Conjunto de Soluciones Factibles del Problema de Transporte con Prohibiciones". *Vestnik Bieloruskogo Universitieta*. Serie I. 1, 64-68, 1982.

Martínez, J.L., "*Algoritmos que Minimizan el Número de Arcos en Redes Bipartitas Completas con Flujos Balanceados*", Tesis Doctoral, Universidad Autónoma de Nuevo León, 1996.

Roodman Gary, *Analysis of Post-Infeasibility in Linear Programming*. Management Science, Vol 25, No. 9, 1979.

Simonnnard, M., *Computer Communication Network Design and Analysis*, Prentice Hall, Englewood Cliffs, N.J., 1977.

Taha Hamdy A., "*Tora Optimization System*", Investigación de Operaciones, SIMNET II IPC, 1990.

A P É N D I C E A

Codificación del sistema "Rosma"

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <conio.h>

/*    TECNICA HEURISTICA DE LA CRUZ IDEADO POR ING. ROSA MARÍA DE LA
CRUZ FERNÁNDEZ. */

void presenta();
void captura();
void diseno();
void inicializa();
void proceso();
void imprimirviejo();
void imprimirnuevo();
int continuar();
void buscarigualesofertas();
void buscarcrdmay();
void buscarigualesdemandas();
void asignarcelda();
void checarofertavalida();
void buscarcol0();
void buscarfil0();
void buscarcasvaciascol();
void checarcastruncadas();

char  op,signo[10];
int   s=0,cuenta1=0,sumA=0,sumb=0,h,p,k,nn,m,n,r,c,i,j,aux,w,acum,sw;
int     o=0,cu=0;
int   a[50],b[50],ax[50],bx[50],rest[100],x[100],temp[100],M[50][50];
int   crd[50],cro[50],num[50],vec1[50],vecigualof[50],vecigualde[50];
int         corte[25],acumula[25];

void main ()                          /* ---------------------------------
---- */
       {                                   /*          PROGRAMA PRINCIPAL
*/
            do
              {
                        presenta();
                        captura();
        inicializa();
                        diseno();
                        proceso();
                        imprimirviejo();
                        imprimirnuevo();
                        continuar();
                  }while(op=='s');
              clrscr();
       }

void presenta()                       /* ---------------------------------
-------- */
      {                                    /* SOLAMENTE DA LA PRESENTACION
DEL PROGRAMA */
            clrscr();
            textcolor(WHITE);
            for(r=1;r<=23;r++)
                  {
                        gotoxy(1,r);
                        cprintf("²");
```

```
                    gotoxy(79,r);
                    cprintf("²");
                }
            for(c=1;c<=79;c++)
                {
                    gotoxy(c,1);
                    cprintf("²");
                    gotoxy(c,24);
                    cprintf("²");
                }
            textcolor(GREEN);
            gotoxy(24,3);
            cprintf("TECNICA HEURISTICA  ïïDE LA CRUZïï ");
            gotoxy(23,4);
            textcolor(WHITE);
            cprintf("-----------------------------------");
    textcolor(GREEN);
            gotoxy(14, 23);
            cprintf("Dise¤ado por : Ing. Rosa Mar¡a de la Cruz
Fern ndez");
            textcolor(WHITE);
            gotoxy(2,22);

      cprintf("²²²²²²²²²²²²²²²²²²²²²²²²²²²²²²²²²²²²²²²²²²²²²²²²²²²²²²²²²²
²²²²²²²²²²²²²²²²²²²²²²²");
      }

void captura()                        /* -------------------------------------
------ */
            {                                 /* CAPTURA DE OFERTAS Y
DEMANDAS DE LA RED */

            /* gotoxy(14, 23);
             cprintf("Dise¤ado por : Ing. Rosa María de la Cruz
Fernández");
             textcolor(WHITE);
             gotoxy(2,22);

cprintf("²²²²²²²²²²²²²²²²²²²²²²²²²²²²²²²²²²²²²²²²²²²²²²²²²²²²²²²²²²²²²²
²²²²²²²²²²²²²²²²");*/
     textcolor(GREEN);
             gotoxy(5,7);
             cprintf("* Número de ofertas          : ");
             scanf("%d",&m);
             gotoxy(5,8);
             cprintf("* Número de demandas         : ");
             scanf("%d",&n);
             gotoxy(2,9);
             textcolor(WHITE);
             cprintf("-------------------------------------------------
---------------------------");
             for(i=1;i<=m;i++)
                       {
                        gotoxy(5,11);
                        cprintf("* Valor de la oferta %d       : ",i);
                        scanf("%d",&a[i]);
                        num[i]=i;
                        gotoxy(34,11);
                        cprintf("         ");
                        ax[i]=a[i];
                        sumA=sumA+a[i];
```

```
                    }
            for(j=1;j<=n;j++)
                    {
                     gotoxy(5,12);
                     cprintf("* Valor de la demanda %d     : ",j);
                     scanf("%d",&b[j]);
                     gotoxy(34,12);
                     cprintf("        ");
                     bx[j]=b[j];
                     sumb=sumb+b[j];
                    }
            if(sumA==sumb)
                  {
                        return;
                  }
            else
                        {
                         gotoxy(27,15);
                         textcolor(RED+BLINK);
                         cprintf(" ! ! !   Error  ! ! !");
                         gotoxy(17,20);
                         textcolor(WHITE);
                         printf(" == Presione cualquier tecla para
COMENZAR ==");
                         getch();
                         exit(1);
                        }
          return;
          }

void diseno()                          /* ---------------------------------
---- */
      {                                    /* EL USUARIO PROPORCIONA LAS
CASILLAS   */
          for (i=1;i<=(m*n);i++)               /* RESTRINGIDAS Y RELLENA LA
MATRIZ     */
          {                                    /* CON -2 Y -1 LAS CR
*/
                x[i]=i;
                rest[i]=-2;
          }
          presenta();
          gotoxy(2,7);
          cprintf(" OFERTAS ");
          gotoxy(68,6);
          cprintf(" DEMANDAS ");
          c=11; r=9;  k=s=0;
          for(i=1;i<=m;i++)
                {
                      gotoxy(6,r);
                      cprintf("%d",a[i]);
                      for(j=1;j<=n;j++)
                            {
                                  gotoxy(c+1,7);
                                  cprintf("%d",b[j]);
                                  gotoxy(11,8);
                                  cprintf("-------------------------
----------------------------------------");
                                  gotoxy(c,r);
                                  k++;
                                  cprintf(" %d ",x[k]);
```

```
                                    c=c+4;
                                }
                                c=11;
                                r=r+1;
                                if (r==21)
                                    {
                                                gotoxy(15,22);printf("==>
Presione cualquier tecla para continuar ...");
                                                getch();
            presenta();
                                                gotoxy(3,7);
                                                cprintf(" OFERTAS ");
                                                gotoxy(67,6);
                                                cprintf(" DEMANDAS ");
                                                c=11; r=9; k=0;
                                    }
                    }
            h=cuenta1=0;
    r++;
            do
            {
                    h++;
                    textcolor(GREEN);gotoxy(15,r+4);
                    cprintf("==>    PRESIONE < 0 cero > PARA TERMINAR
<==");
                    do
                    {
                            textcolor(GREEN);gotoxy(14,r);
                            cprintf("* Seleccione el número de la celda
prohibida [²²]");
                            gotoxy(65,r);
                            scanf("%d",&temp[h]);
                    }while(temp[h]>(m*n));
                    textcolor(WHITE);gotoxy(4,r+2);
                    cprintf("NO EN USO: ");
                    s=s+2;
                    if (temp[h]!=0)
                    {
                            gotoxy(13+s,r+2);
                            cprintf("%d",temp[h]);
                            s=s+2;
                            cuenta1=cuenta1+1;
                    }
      if (s>60)
                    {
            textcolor(WHITE);gotoxy(4,r+2);
                            cprintf("NO EN USO:
");
                            s=0;
                    }
            }while (temp[h] != 0);

            for(i=1;i<=cuenta1;i++)
                    rest[temp[i]]=-1;
            h=1;
    for (i=1;i<=m;i++)
                    for (j=1;j<=n;j++)
                            M[i][j]=rest[h++];
      return;
      }
```

```
void inicializa()                          /* --------------------------------
---- */
      {                                        /* INICIALIZA VARIABLES
*/
            for(i=1;i<=m;i++)
            {
                  cro[i]=0;
                  signo[i]=' ';
                  for(j=1;j<=n;j++)
                  {
                        crd[j]=0;
                        M[i][j]=0;
                  }
            }
            return;
      }

/* PROCESO DEL PROGRAMA, ANALIZAR CADA DETALLE */
void proceso()
      {                                    /* ----------------------------
-------- */
            for(i=1;i<=m;i++)                  /* CONTEO DE CASILLAS
RESTRINGIDAS       */
            {
                  acum=0;
                  for(j=1;j<=n;j++)            /* CRO(ofertas) Y
CRD(demandas)          */
                   if (M[i][j]==-1)
                   {
                              crd[j]=crd[j]+1;
                              acum=acum+crd[j];
                   }
                  cro[i]=acum;
            }
            for(i=1;i<=m-1;i++)              /* -------------------------
------------ */
                  for(j=i+1;j<=m;j++)          /* ORDENAMIENTO DE LA >
OFERTA A ANALIZAR */
                   if(ax[j]>ax[i])               /* POR MEDIO DE UN
APUNTADOR LLAMADO NUM  */
                        {
                              aux=num[j];
                              num[j]=num[i];
                              num[i]=aux;
                        }
            for(i=1;i<=m;i++)                /* -------------------------
----------- */
                  {                                /* CICLO/CONTROL DE
OFERTAS > A ANALIZAR */
                  if(signo[num[i]]==' '|| signo[num[i]]=='*')
                  {
                        buscarigualesofertas();  /* BUSCA SI HAY
OFERTAS CON IGUAL VALOR  */

                         /* y SI HAY = OFERTAS CON EL MISMO # CRO
*/
                        buscarcrdmay();          /* BUSCA LA
CELDA-DEMANDA MAYOR A ASIGNAR*/

                         /* En BASE A LA CELDA DEMANDA > SE
ASIGNA*/
```

```
                              asignarcelda();                    /*
DEPENDIENDO DEL VALOR DE LA DEMANDA   */
                        }
                        else
                              break;
                  }
            w=0;
            for(i=1;i<=m;i++)
                  if((signo[i]=='A')||(signo[i]=='E'))
                        w++;
            if (w==m)
                  {
      for(i=1;i<=m;i++)
                              for(j=1;j<=n;j++)
                                    if ((M[i][j]!=-2)&&(M[i][j]!=-1))
                                          ax[i]=ax[i]+M[i][j];

                  }
            else
                  {
                        clrscr(); cprintf("ERROR");sleep(7);
                  }
      }

void buscarigualesofertas()
      {
            s=1;
           /* ------------------------------------ */
            for(j=i+1;j<=m;j++)            /* VERIFICAR SI HAY OFERTAS
CON EL MISMO  */
                  if(ax[num[j]]==ax[num[i]])    /* VALOR
*/
                        {
                              vecigualof[++s]=num[j];
                              if(s==2)
                                    vecigualof[1]=num[i];
                        }
            if(s!=1)                          /* BUSCAR LA OFERTA CON >#
DE CRO          */
                  {
                        for(j=1;j<=s-1;j++)
                              for(k=j+1;k<=s;k++)

      if(cro[vecigualof[k]]>cro[vecigualof[j]])
                                          {
                                                aux=vecigualof[k];

      vecigualof[k]=vecigualof[j];
                                                vecigualof[j]=aux;
                                          }
                        h=1;
                        for(j=i;j<=s+1;j++)
                              num[j]=vecigualof[h++];
                  }
            return;
      }

void buscarcrdmay()                  /* ---------------------------------
--- */
      {                                 /* BUSCO LOS NODOS DEMANDAS DE
CRD DE    */
```

```
            w=0;                                    /* >A< A ANALIZAR PARA EL
NODO OFERTA   */
            for(j=1;j<=n;j++)
                  vec1[j]=0;
            for(j=1;j<=n;j++)
                  {
                        if (M[num[i]][j]==-2)
                              vec1[++w]=j;
                  }
            if (w>1)
            {
                  sw=1;
                  for(j=1;j<=w-1;j++)                    /* -------------
------------------------- */
                        for(k=j+1;k<=w;k++)                 /*
ORDENAMIENTO DE LA > DEMANDA ANALIZAR  */
                        {                                       /* DE
ACUERDO AL >#  DE CRD                  */
                         sw=0;
                     /* SI NO HAY CRD, SE EVALUA POR DEMANDA    */
                         if(crd[vec1[k]]>crd[vec1[j]])
                              {
                                    aux=vec1[k];
                                    vec1[k]=vec1[j];
                                    vec1[j]=aux;
                              }
                        }
                  if(sw==0)
                        buscarigualesdemandas();

                  if(sw==1)
                        {
                        for(j=1;j<=w-1;j++)               /* -----
---------------------------------- */
                              for(k=j+1;k<=w;k++)             /*
ORDENAMIENTO DE LA > DEMANDA ANALIZAR  */
                              if(bx[vec1[k]]>bx[vec1[j]])
/* DE ACUERDO AL > DE DEMANDA, Y TAMBIEN  */
                              {
                                    /* ANALIZA CUANDO LAS
DEMANDAS > SON =     */
                                    aux=vec1[k];
                                    vec1[k]=vec1[j];
                                    vec1[j]=aux;
                              }
                        }
            }
      return;
      }

void buscarigualesdemandas()
      {
            for(j=1;j<=w;j++)
                  corte[j]=0;
            cuenta1=1;h=0;
             /* ------------------------------------- */
            for(j=1;j<=w-1;j++)             /* VERIFICAR SI HAY DEMANDAS
CON EL MISMO */
                  {                               /* VALOR y UN CJTO. DE
DEMANDAS IGUALES    */
```

```
                        s=1;                            /* CON EL MISMO CRD
*/
                        for(k=j+1;k<=w;k++)
                              if(crd[vec1[k]]==crd[vec1[j]])
                                    {
                                          corte[cuenta1++]=++s;
                                          h=h+s;
                                    }
                        j=h;
                  }
            if(cuenta1!=1)
            {
                  cu=0;
                  for(o=1;o<=cuenta1;o++)          /* COMPARAR SI DE
ESAS DEMANADAS =,EXISTE */
                        {
                              for(j=1+cu;j<=corte[o]+cu-1;j++)     /*
UNA DEMANDA > PARA ORDENARLAS          */
                                    for(k=j+1;k<=corte[o]+cu;k++)
                                          if(bx[vec1[k]]>bx[vec1[j]])
                                                {
                                                      aux=vec1[k];
                                                      vec1[k]=vec1[j];
                                                      vec1[j]=aux;
                                                }
                              cu=cu+corte[o];
                        }
            }
            return;
      }

void asignarcelda()
      {
            for(j=1;j<=w;j++)              /* ASIGNACION DE FLUJOS A LA
MATRIZ      */
                  {
                        if (bx[vec1[j]]<ax[num[i]])            /*
SI DEMANDA < OFERTA */
                                                {

      M[num[i]][vec1[j]]=bx[vec1[j]];

      ax[num[i]]=ax[num[i]]-bx[vec1[j]];
                                                      bx[vec1[j]]=0;
                                                      buscarcol0();

      checarcastruncadas();
                                                      if(j==w)
                                                      {
                                                            acum=0;

      for(k=1;k<=w;k++)

      acum=acum+M[num[i]][vec1[k]];

      if((a[num[i]]-acum)>0)    /*(w==1))*/
                                                                  {

      signo[num[i]]='&';
                                                                        break;
                                                                  }
```

```
                                                    else
                                                    {
      signo[num[i]]='A';
                                                          break;
                                                    }
                                              }
                                        }
                                        else
                                              if
(bx[vec1[j]]==ax[num[i]])       /* SI DEMANDA = OFERTA */
                                              {
      checarofertavalida();
                                                    if(sw==0)
                                                          {
      M[num[i]][vec1[j]]=bx[vec1[j]];
      bx[vec1[j]]=0;
      ax[num[i]]=0;
      buscarcol0();
      checarcastruncadas();
      buscarfil0();
      signo[num[i]]='A';
      break;
                                                          }
                                                    else
                                                          break;
                                              }
                                              else
                                                    {
                                                          if
(bx[vec1[j]]>ax[num[i]])   /* SI DEMANADA > OFERTA */
                                                          {
      buscarcasvaciascol();
      if (h>1)                      /* SI HAY MAS CASILLAS  */
      {                             /* POR ASIGNAR          */
            M[num[i]][vec1[j]]=ax[num[i]];
            bx[vec1[j]]=bx[vec1[j]]-ax[num[i]];
            ax[num[i]]=0;
            buscarfil0();
            signo[num[i]]='A';
      }
      else                          /* SOLO HAY UNA CASILLA */
```

```
{                          /* POR ASIGNAR          */
      sw=1;acum=0;
      if(signo[num[i]]=='*')  /* CORRESPONDE AL PASO  */
            {
                  for(k=1;k<=m;k++)    /* 3.2.1 DEL ALGORITMO. */
                        if(signo[k]=='#') /* BUSCO UN # EN ALGUNA
*/
                              {
      /* OFERTA               */
                                    ax[num[i]]=ax[num[i]]+ax[k];
                                    ax[k]=0;
                                    signo[k]='E';
                                    h=i;  i=k;
                                    buscarfil0();
                                    i=h;
                                    sw=0;
                              }
                  if (sw==1)
                        {
                              for(k=1;k<=m;k++)       /*
CORRESPONDE AL PASO  */
                                    if(signo[k]=='&')     /* 8.2
DEL ALGORITMO.   */
                                          {                  /*
BUSCO UN & EN ALGUNA */
                                                acum=acum+ax[k];
/* OFERTA               */
                                                ax[k]=0;
                                                signo[k]='A';
                                                sw=0;
                                          }
                              if (sw==0)
                                    {
```

```
      ax[num[i]]=ax[num[i]]+acum; /* ACUMULO VALORES DE & */
                                                signo[num[i]]=' ';
                                                i--;
                                    /*          acum=0;
                                                      for(k=1;k<=w;k++)
CHECO SI LO ACUMULADO

      acum=acum+bx[vec1[k]];        SOBRA
                                                if(ax[num[i]]>acum)
                                                      signo[num[i]]='&';
                                                if(ax[num[i]]==acum)
                                                      signo[num[i]]='A';
                                                else
                                                      {

      clrscr();cprintf("NO PUEDE SER MENOR");sleep(7);
                                                      } */
                                          }
                                          else
                                          {
                                                buscarcrdmay();   j=1;
                                          }
                                    }
                              }
                              else
                                    signo[num[i]]='*';
                                    if (i==m)
                                    {
                                          i--;
                                          break;
                                    }
```

```
                                else
                                                    break;
        }
                                                                    }
                                                                }
                    }
              return;
        }

void buscarcol0()                       /* ---------------------------------
---- */
        {                                   /* CICLO QUE HACE 0ïs LA COLUMNA
AFECTA- */
              for(k=1;k<=m;k++)                 /* DA POR SATISFACER DEMANDA
*/
                    if(M[k][vec1[j]]==-2)
                                M[k][vec1[j]]=0;
        return;
        }

void buscarfil0()                       /* ---------------------------------
---- */
        {                                   /* CICLO QUE HACE 0ïs LA FILA
AFECTADA    */
              for(k=1;k<=n;k++)                 /* POR SATISFACER OFERTA
*/
                    if(M[num[i]][k]==-2)
                    {
                                M[num[i]][k]=0;
                                j++;
                    }
        return;
        }

void buscarcasvaciascol()               /* ---------------------------------
---- */
        {                                   /* CICLO QUE BUSCA CASILLAS
VACIAS EN LA */
              h=0;                              /* COLUMNA PARA CHECAR SI ES
LA UNICA     */
              for(k=1;k<=m;k++)                 /* CASILLA POR ASIGNAR FLUJO
*/
                    {
                                if (M[k][vec1[j]]==-2)
                                            h++;
                    }
        return;
        }

void checarcastruncadas()               /* ---------------------------------
---- */
        {                                   /* CHECA CASILLAS TRUNCADAS PARA
ASIGNALE*/
              for(h=1;h<=m;h++)                 /* A LA OFERTA UN SIGNO DE #
*/
                    {
                                acum=0;
                                for(k=1;k<=n;k++)
```

```
                                    if ((M[h][k]==0)||(M[h][k]==-1))
                                          acum++;
                                    if (acum==n)
                                          signo[h]='#';
                              }
      return;
      }

void checarofertavalida()
      {
            for(h=1;h<=w;h++)
                  s=1;
                  for(k=1;k<=m;k++)
                        if(M[k][vec1[h]]==-2)
                              acumula[h]=s++;
                  acum=0;
      for(h=1;h<=w;h++)
                        if(acumula[h]==1)
                              acum=acum+bx[vec1[h]];
                        if(acum>ax[i])
                                    {
                                          signo[i]='*';
                                          sw=1;
                                          j=w+1;
                                    }
                        else
                              sw=0;
      return;
      }
/* ------------------------------------------------------------------
------ */
void imprimirviejo()                  /* ---------------------------------
-------- */
      {                                     /* IMPRIME RESULTADOS DE LA
MATRIZ RESULTATE */
            presenta();
            gotoxy(2,7);
            cprintf(" OFERTAS ");
            gotoxy(68,6);
            cprintf(" DEMANDAS ");
            c=11;
            r=9;
            for(i=1;i<=m;i++)
                  {
                        gotoxy(6,r);
                        cprintf("%d",a[i]);
                        for(j=1;j<=n;j++)
                              {
                                    gotoxy(c+1,7);
                                    cprintf("%d",b[j]);
                                    gotoxy(11,8);
                                    cprintf("-------------------------
 ----------------------------------------");
                                    gotoxy(c,r);
                                    if (M[i][j]==-1)
                                                cprintf(" X");
                                          else
                                                cprintf(" %d
",M[i][j]);
                                    c=c+4;
                              }
```

```
                              c=11;
                              r=r+1;
                              if (r==21)
                                    {
                                                gotoxy(15,22);cprintf("==>
Presona cualquier tecla para continuar ...");
                                                getch();
                presenta();
                                                gotoxy(3,7);
                                                cprintf(" OFERTAS ");
                                                gotoxy(67,6);
                                                cprintf(" DEMANDAS ");
                                                c=11;
                                                r=9;
                                    }
                        }
              gotoxy(17,r+1);textcolor(GREEN);
              cprintf(" ASIGNACION DE FLUJOS POR LA T□CNICA DE LA CRUZ
");
              gotoxy(17,r+2);
              cprintf("           X = Casillas Restringidas");
              gotoxy(17,r+3);
              cprintf(" == Presione cualquier tecla para  continuar ==");
              getche();
              return;
        }

void imprimirnuevo()                  /* ---------------------------------
-------- */
        {                                   /* IMPRIME NUEVA RED  */
              presenta();
              gotoxy(2,7);
              cprintf(" OFERTAS ");
              gotoxy(68,6);
              cprintf(" DEMANDAS ");
              c=11;
              r=9;
              for(i=1;i<=m;i++)
                    {
                          gotoxy(6,r);
                          cprintf("%d",ax[i]);
                          for(j=1;j<=n;j++)
                              {
                                    gotoxy(c+1,7);
                                    cprintf("%d",b[j]);
                                    gotoxy(11,8);
                                    cprintf("-------------------------
 ----------------------------------------");
                                    gotoxy(c,r);
                                    if (M[i][j]==-1)
                                                cprintf(" X");
                                          else
                                                cprintf(" ="); /*
 %d ",M[i][j]);*/
                                    c=c+4;
                              }
                          c=11;
                          r=r+1;
                          if (r==21)
                                {
```

```
                                        gotoxy(15,22);cprintf("==>
Presona cualquier tecla para continuar ...");
                                        getch();
            presenta();
                                        gotoxy(3,7);
                                        cprintf(" OFERTAS ");
                                        gotoxy(67,6);
                                        cprintf(" DEMANDAS ");
                                        c=11;
                                        r=9;
                                    }
                }
          gotoxy(25,r+1);textcolor(GREEN);
          cprintf(" PROPUESTA DE UNA NUEVA RED FACTIBLE ");
          return;
     }

int continuar()                     /* ---------------------------------
---- */
{                                   /* CONTINUAR SI O NO
*/
     do
          {
           gotoxy(19,r+4);
           printf(" * Capture ==>  [s] Seguir  [f] Fin  [ ² ]");
           gotoxy(58,r+4);
           op=tolower(getche());
           if(op!='s' && op!='f')
                {
                     gotoxy(58,r+5);
                     printf("Solo oprima S o F");
                     sleep(1);
                     gotoxy(58,r+5);
                     printf("                  ");
                }
          }while(op!='s' && op!='f');
 return(op);
}_
```

RESUMEN AUTOBIOGRÁFICO

Rosa María de la Cruz Fernández

Candidato para el Grado de
Maestro en Ciencias de la Administración con Especialidad en Sistemas

Tesis	:	UN SISTEMA PARA LA DETECCIÓN Y ELIMINACIÓN DE INFACTIBILIDAD EN PROBLEMAS DE REDES.
Campo de estudio	:	Ingeniería.
Biografía	:	Nacida en el municipio de Torreón, Coahuila el 15 de Enero de 1973, hija del Sr. Ramón de la Cruz Carreón y de la Sra. María Santos Fernández de de la Cruz; hermana de Juan Ramón de la Cruz Fernández y Ana Catalina de la Cruz Fernández. Actualmente esposa del Sr. Arturo del Angel Ramírez.
Educación	:	Egresada de la Universidad Autónoma de Nuevo León, con el grado de Ingeniero Administrador de Sistemas en el año de 1994.

Experiencia Profesional : Ayudante general en el área de sistema y capacitación en la empresa COACTOR en 1994, Auxiliar en el proceso de empaque y embarque en VITRO FLEX, S.A. de C.V. en 1995, Ingeniero en Sistemas en IMPULSORA ELIZONDO en 1996, Maestro de horas en la Facultad de Ingeniería Mecánica y Eléctrica, en la Facultad de Derecho y Ciencias Sociales y, Cursos al Exterior, en la Universidad Autónoma de Nuevo en 1997; Trabajo Independiente en Capacitación y Desarrollo de Sistemas.

Grado al que se aspira : Maestro en Ciencias de la Administración con Especialidad en Sistemas.