UNIVERSIDAD MICHOACANA DE SAN NICOLÁS DE HIDALGO
FACULTAD DE INGENIERÍA QUÍMICA

"ANÁLISIS ESTRUCTURAL DE PROCESOS QUÍMICOS"

POR

FABRICIO NÁPOLES RIVERA

TESIS PRESENTADA A LA FACULTAD DE INGENIERÍA QUÍMICA COMO REQUISITO PARCIAL PARA OBTENER EL TITULO DE INGENIERO QUÍMICO

ASESORES
DR. MEDARDO SERNA GONZÁLEZ
M.C. JOSÉ MARÍA PONCE ORTEGA

MORELIA, MICH. **JULIO DE 2006**

CONTENIDO

LISTA DE FIGURAS

LISTA DE TABLAS

CAPITULO 1: INTRODUCCIÓN

En un proceso químico, la transformación de las materias primas en los productos deseados usualmente no se puede lograr en una etapa única. En vez de ello, la transformación global es el resultado de la combinación de transformaciones químicas y/o físicas intermedias que se llevan a cabo en varias etapas interconectadas, entre las que se tienen las reacciones químicas, las separaciones, el mezclado, el calentamiento, el enfriamiento y el cambio de presión de las corrientes de proceso. Una etapa de transformación en un diagrama de flujo puede ser representada por un equipo o una combinación de equipos de proceso.

1.1 Diagramas de flujo de procesos químicos

Los **diagramas de flujo** son representaciones gráficas de los procesos químicos que muestran, en esencia, todos y cada uno de los equipos a ser usados en las diversas etapas del proceso, así como sus interconexiones mediante flechas que representan los distintos flujos de materia, energía y/o información. Como se muestra en la Figura 1.1, un diagrama de flujo de un proceso químico continuo moderadamente complejo está constituido por un número significativo de equipos de proceso individuales tales como reactores, separadores, intercambiadores de calor, mezcladores y divisores de flujo, entre otros. Además, la mayoría de los procesos se caracterizan por la presencia de corrientes de recirculación debido a que los reactores químicos no operan al 100% de conversión. Por lo tanto, por razones económicas y ambientales es preciso recuperar y recircular los reactivos no consumidos. En algunos casos, la operabilidad de los procesos requiere la selección de corrientes de purga para evitar la acumulación de sustancias indeseables. Por consiguiente, los diagramas de flujo definen completamente las complejas estructuras que forman las diversas etapas de proceso, las cuales son requeridas para

lograr las transformaciones físicas y químicas de las materias primas en los productos deseados.

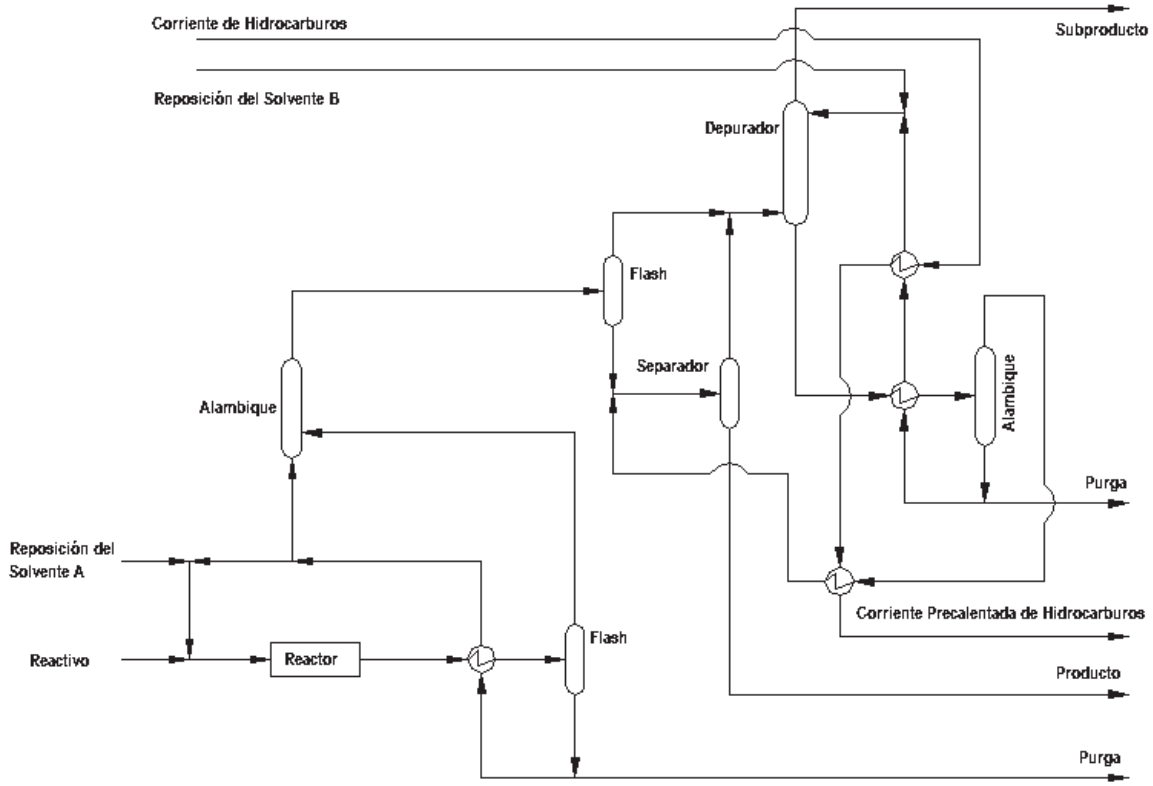


Figura 1.1 Diagrama de Flujo de un Proceso Químico.

1.2 Simulación de procesos químicos y diagramas modulares

La operación de cada equipo de un proceso puede ser descrita por un modelo matemático, que es un conjunto de ecuaciones independientes que permite calcular las variables de las corrientes de salida una vez especificadas las variables independientes de la(s) corriente(s) de entrada y los parámetros del equipo. Un modelo matemático se construye comúnmente usando principios fundamentales de la física y la química, por lo que está constituido por balances de materia y energía junto con aquellos datos físicos y termodinámicos, así como correlaciones que son necesarias para los cálculos.

La simulación de un equipo de proceso individual se refiere a la solución numérica de su modelo matemático. Esta tarea se simplifica al ordenar sistemáticamente las ecuaciones que definen cada equipo para obtener rutinas de cálculo preestablecidas. La reformulación de las ecuaciones del modelo de un equipo como un algoritmo hace posible su simulación asistida por computadora. Esto implica que los algoritmos de los equipos deben ser expresados como programas o subrutinas de cómputo, los cuales son comúnmente conocidos como módulos de simulación. Generalmente, es muy grande el número de diferentes módulos de equipos de proceso, por lo que su colección puede dar lugar a una gran biblioteca de módulos de simulación.

El enfoque de cálculos por computadora más comúnmente utilizado para simular procesos químicos estacionarios es la estrategia modular secuencial, la cual requiere que cada unidad de proceso sea representada en términos de uno o más módulos de simulación. En esta aproximación también es necesario representar formalmente como módulos de simulación a los equipos virtuales, tales como mezcladores y divisores de corrientes, y consignar los nombres de los módulos de simulación (rutinas de cálculo) en los bloques de los diagramas en lugar de los nombres de los equipos de proceso. Todos los flujos y módulos son numerados y las conexiones entre los módulos describen la transferencia de información entre los módulos y también la transferencia de materia y energía. Un ejemplo de esta aplicación se muestra en la Figura 1.2, que es el diagrama de simulación del diagrama de flujo de la Figura 1.1. Por lo tanto, en la estrategia modular secuencial un proceso completo se puede modelar en términos de un conjunto de módulos de simulación conectados por los flujos de materia y energía que existen entre ellos. De esta manera se obtiene el **diagrama modular** para la simulación de un proceso.

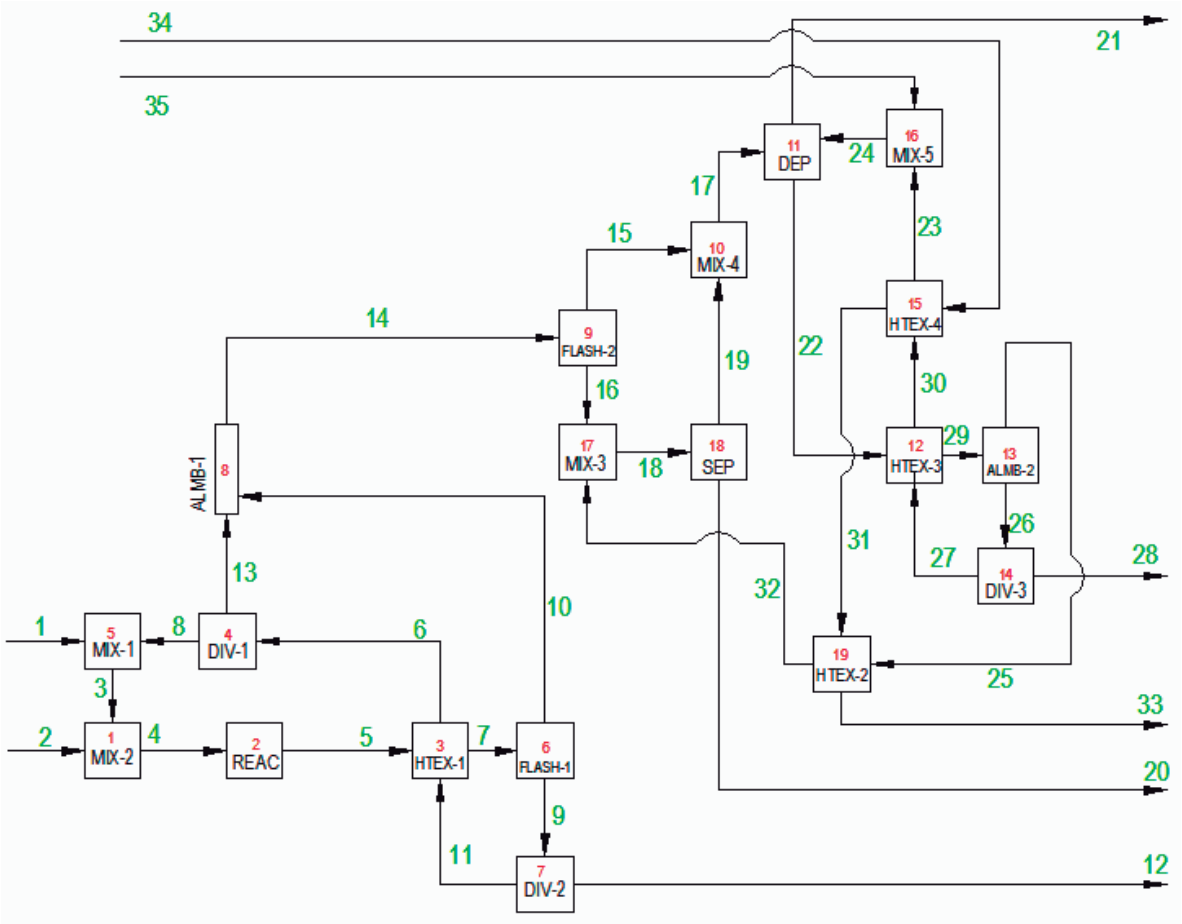


Figura 1.2. Diagrama Modular Para la Simulación del Proceso de la Figura 1.1.

Cuando un diagrama modular no contiene corrientes de recirculación, la complejidad del problema de simulación mediante la estrategia modular secuencial es muy simple; los cálculos comienzan en la primera alimentación y prosiguen módulo por módulo en la secuencia que fija el flujo físico del proceso, hasta que los productos son obtenidos. Sin embargo, la mayoría de los procesos químicos se caracterizan por la presencia de corrientes de recirculación. El diagrama de flujo de la Figura 1.1 es un ejemplo representativo de este tipo de procesos. La presencia de recirculaciones, como se puede apreciar claramente en el diagrama

modular de la Figura 1.2, impide resolver directamente en algún orden los módulos de simulación, dado que las salidas de unos equipos son las entradas a los equipos previos en la secuencia. Para estos casos, la descomposición y el rasgado de diagramas de flujo modulares son dos técnicas ampliamente utilizadas para desarrollar una estrategia ordenada de cálculo del problema de simulación de procesos.

1.3 Descomposición y rasgado de los procesos químicos

La descomposición de los diagramas modulares tiene por objeto identificar los bloques de módulos que deben resolverse simultáneamente y por separado, con el propósito de simplificar la solución del problema de simulación de procesos (Norman, 1962; Sargent y Westerberg, 1964). Para lograr este objetivo se buscan, en primer lugar, los conjuntos más pequeños de módulos interconectados por corrientes de recirculación y, en segundo lugar, se establece el orden de resolución de tales bloques independientes. Para ilustrar la conveniencia de la descomposición de un sistema en subsistemas para facilitar su análisis, considere el diagrama modular mostrado en la Figura 1.2. Este parece ser bastante complejo, debido a que a primera vista se observa un sólo bloque de módulos sumamente interconectados. Sin embargo, en la Figura 1.3 se muestra que este diagrama modular se puede dividir en cuatro bloques de módulos o subsistemas que deben resolverse independientemente y de acuerdo a una secuencia dada, lo que produce una simplificación sustancial de la simulación del proceso global. Las fronteras de tales subsistemas se señalan claramente en la Figura 1.3, debido a que una vez identificados los bloques de módulos independientes se representan como bloques únicos en el diagrama modular.

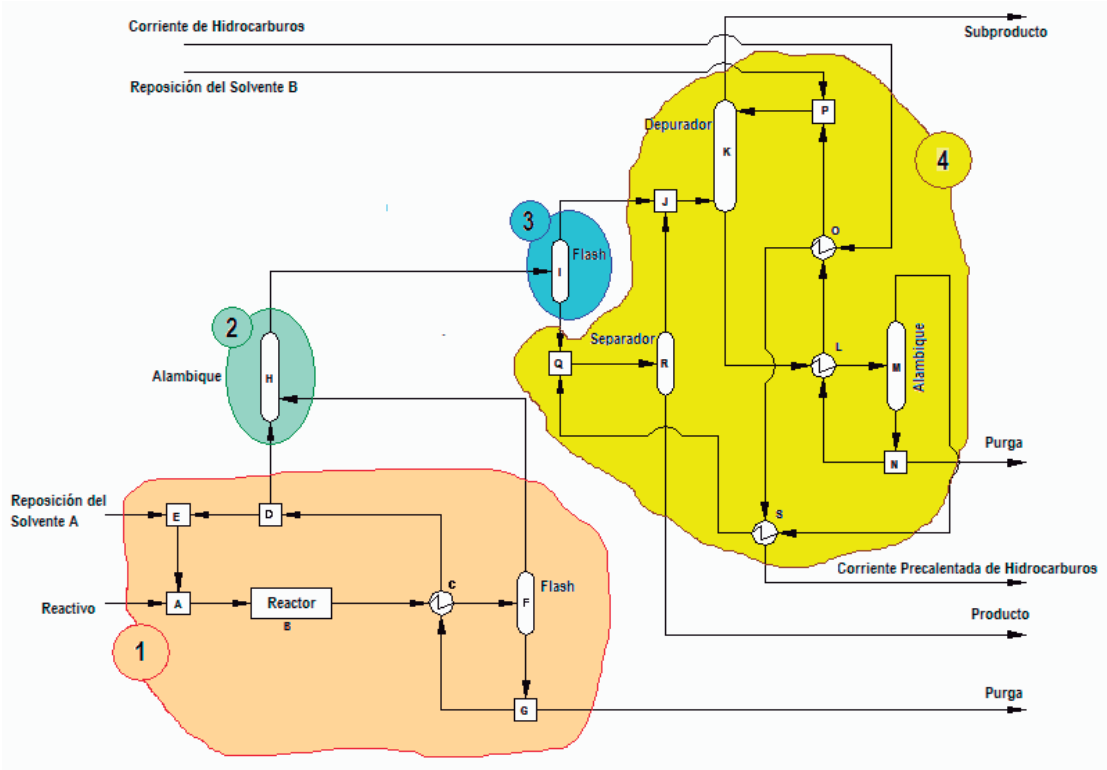


Figura 1.3. Identificación de Bloques de Módulos o Subsistemas.

Por otro lado, para resolver por separado un bloque de módulos involucrados en uno o más lazos de recirculación es necesario determinar el número mínimo de corrientes de corte sobre las cuales se deberá iterar y seleccionar las corrientes de corte (Barkeley y Motar, 1972; Upandhye y Grens, 1974). En la Figura 1.4 se localizan las dos corrientes de corte que rompen los lazos del cuarto bloque de módulos independientes del diagrama modular de la Figura 1.2. Se dan estimados iniciales de las variables independientes de las corrientes de corte seleccionadas para romper los lazos de recirculación, a fin de realizar en forma secuencial los cálculos de cada módulo en la dirección del flujo. Estos estimados son actualizados en el transcurso de pasos repetidos a través del diagrama del bloque de módulos. Los cálculos terminan cuando se ha alcanzado la concordancia deseada entre los valores de las corrientes supuestas en dos iteraciones sucesivas.

El número de corrientes de corte y su selección se establece de forma que el número de cálculos iterativos se minimice; desafortunadamente, es muy difícil predecir si alguna selección en particular es mejor que otra sin tener que resolver los balances de materia y energía para comprobarlo. Por otro lado, el establecimiento del orden en que deben actualizarse las corrientes de corte es una tarea para la cual tampoco existe un método riguroso o infalible que asegure una disminución en el esfuerzo de cálculo.

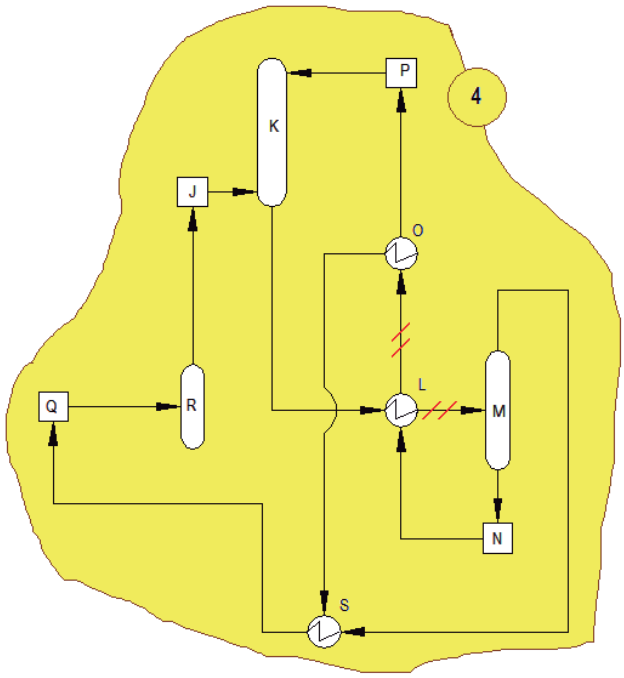


Figura 1.4. Selección de Corrientes de Corte del Subsistema 4.

1.4 Justificación del presente trabajo de tesis

De las secciones anteriores es obvio que, si la estrategia modular secuencial es usada para simular procesos con un gran número de recirculaciones es necesario, antes de iniciar los cálculos, resolver cuatro problemas no triviales que todavía son tema de continua investigación:

1. Descomposición de los diagramas de flujo[1].
2. Determinación exacta del número mínimo de corrientes de corte[2] sobre las cuales se deberá iterar.
3. Selección de las corrientes de corte o iteración.
4. Especificación del orden en que las corrientes de corte deberán actualizarse[3].

En este trabajo se presentan e implementan en computadora algunos algoritmos reportados en la literatura, con el propósito de llevar a cabo automáticamente la descomposición y el rasgado de procesos químicos altamente interconectados. Los procedimientos automáticos de descomposición y rasgado son necesarios si el usuario no tiene tiempo o experiencia para hacer estas tareas manualmente o si el proceso a ser simulado incluye un gran número de módulos y flujos, así como muchas corrientes de recirculación.

Conviene destacar que los algoritmos de descomposición y rasgado son apropiados para el caso de simulación en modo análisis de procesos químicos, esto es, cuando para la resolución secuencial de módulos de simulación independientes, según el flujo físico de la planta, se especifican como datos las variables independientes de las corrientes de entrada y los parámetros de los equipos, y como incógnitas todas las variables de las corrientes de salida.

Por otro lado, es importante tener presente que es posible utilizar los **diagramas de flujo de información** para representar tanto la estructura de un diagrama de flujo, que establece la presencia de diferentes equipos junto con sus interconexiones, así como la estructura del sistema de ecuaciones algebraicas de un modelo matemático de un equipo o de todo

[1]El término original para esta operación es *Partitioning.*

[2]Llamadas corrientes de corte o *Tear Streams.* Al proceso de selección de éstas suele llamársele *Tearing* o *Rasgado.*

[3]A este proceso se le llama *Sequencing* en la bibliografía especializada.

un proceso químico. Por consiguiente, los algoritmos de descomposición y rasgado de procesos químicos también se pueden aplicar a sistemas de ecuaciones algebraicas no lineales, con el propósito de tomar ventaja de la forma de las ecuaciones y resolver el sistema de forma más eficiente.

Este trabajo esta dirigido a dos sectores muy importantes de la Ingeniería Química: los estudiantes y los ingenieros en activo. En el primer grupo se resalta la utilidad del material desarrollado como una referencia de apoyo en la enseñanza y aprendizaje de los temas de rasgado y particionado de procesos químicos, principalmente para los alumnos de Ingeniería de Procesos II de la Facultad de Ingeniería Química de la Universidad Michoacana de San Nicolás de Hidalgo. Estos temas son obligatorios todos los programas de licenciatura de Ingeniería Química; sin embargo, no se dispone de material didáctico y bibliográfico adecuado, por lo que se espera que este esta tesis contribuya para eliminar dicha deficiencia. Por otro lado, este trabajo tiene ciertos alcances en la práctica industrial, debido a que las herramientas presentadas pueden ser usadas por los ingenieros de procesos para comprender de mejor manera las dependencias entre las unidades de un proceso y, por consiguiente, adquirir una visión global de los sistemas bajo su responsabilidad y proponer estrategias adecuadas para la mejora de los mismos. Tanto los alumnos como los ingenieros interesados en estos temas dispondrán de un programa de cómputo que implementa los correspondientes algoritmos. Las principales virtudes de este software son: la facilidad de manejo e interpretación de resultados, la simplificación en la introducción de los datos con respecto a programas similares y la posibilidad de acceder a él gratuitamente.

1.5 Objetivo

De las secciones anteriores se infiere la importancia de automatizar los algoritmos de descomposición y rasgado de procesos químicos con un número apreciable de módulos y corrientes de recirculación, con el fin de

facilitar su simulación. Por lo tanto, el **objetivo** de este proyecto es el desarrollo de un software de gran calidad que implemente algunos algoritmos de descomposición y rasgado reportados previamente en la literatura. Este software se caracteriza por ser interactivo y con salidas gráficas y/ó tabulares de los resultados, características que facilitan los estudios de análisis estructural de los procesos químicos.

1.6 Contenido de la Tesis

La tesis está organizada en cuatro capítulos y tres apéndices. En el primer capítulo se presentan la problemática y la justificación, así como los objetivos de este trabajo. En el capítulo 2 se describen los conceptos básicos de la teoría de grafos, la asociación entre los grafos y matrices, el algoritmo de descomposición de diagramas modulares basado en la búsqueda en profundidad de grafos, el algoritmo de rasgado de diagramas modulares y el algoritmo de ordenación de ecuaciones. En el capítulo 3 se presentan varios casos de estudios para ilustrar la aplicación de los algoritmos de descomposición y rasgado. En el capítulo 4 se exponen las principales conclusiones de este trabajo. Finalmente, en los apéndices se presentan los listados de los programas de cómputo que implementan los algoritmos presentados en el capítulo 2.

CAPITULO 2: METODOLOGIA Y ALGORITMOS

La división de un sistema de gran dimensión en subsistemas más pequeños, que se deben resolver por separado en un orden determinado, es una estrategia acertada para simplificar la solución del problema de simulación de procesos químicos en estado estacionario empleando la técnica modular secuencial.

En la primera sección de este capítulo se presentan aspectos básicos de la teoría de grafos, debido a que éstos se pueden emplear para facilitar la comprensión de los algoritmos de descomposición y rasgado de los diagramas modulares de procesos. Se otorga especial atención a la representación de dígrafos por medio de matrices, debido a que éstas permiten la implementación en computadora de los algoritmos de grafos. En las subsecuentes secciones se presentan los algoritmos de descomposición, rasgado y ordenamiento, así como su aplicación para obtener la secuencia de resolución de los distintos módulos o bloques de módulos de diagramas de flujo de procesos altamente interconectados.

2.1 Conceptos básicos de grafos

La teoría de grafos puede ser usada para representar sistemas compuesto de objetos discretos y las relaciones entre estos objetos (Mah, 1990). Los objetos son comúnmente conocidos como *vértices* o *nodos* y las relaciones como *arcos* o *aristas*.

Un grafo G consiste de un conjunto discreto de *n nodos* V relacionados por un conjunto discreto de *m arcos* A, de modo tal que un arco dado conecta sólo un par de nodos dando lugar a asociaciones binarias entre nodos y arcos. Cuando las relaciones entre dos nodos tienen una dirección (por ejemplo, flujos de materia y flujos de calor que conectan dos módulos de simulación en un proceso), las representaciones gráficas de sistemas se denominan grafos dirigidos o dígrafos.

2.1.1 Representación de un grafo dirigido

La Figura 2.1 muestra la forma general de un dígrafo con 8 nodos y 11 arcos. Tal como se indica, un dígrafo se representa empleando círculos[4] para especificar los nodos y flechas (líneas dirigidas) para trazar los arcos; las cabezas de las flechas indican la dirección de los arcos. Esta característica permite representar comúnmente a los arcos por medio de pares ordenados de nodos $a_k = (v_i, v_j)$, donde se dice que v_i es el **nodo de origen** (la cabeza) y v_j el **nodo de destino** (la cola) del arco a_k. Por lo tanto, los arcos muestran que entre un par de vértices existe una relación unívoca; por ejemplo, el arco a_k parte de v_i y llega a v_j, pero no lo contrario.

$G = \{V, A\}$ con:

$V = \{1,2,3,4,5,6,7,8\}$

$A = \{(1,8),(2,1),(3,2),(3,4),(4,5),(5,3)$
$(5,6),(5,7),(5,8),(6,7),(8,2)\}$

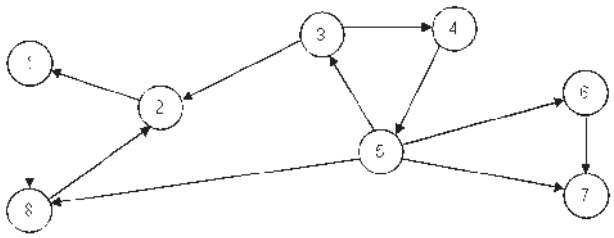


Figura 2.1 Representación de un dígrafo.

Un grafo dirigido puede representarse simbólicamente como $G = \{V, A\}$, donde $V = \{v_1, v_2, \cdots, v_n\}$, $A = \{a_1, a_2, \cdots, a_m\}$ y $a_k = (v_i, v_j)$ tal que $v_i, v_j \in V$. En dicho dígrafo se entiende que $(v_i, v_j) \neq (v_j, v_i)$ y, en muchos casos, sólo existe uno de los pares ordenados de vértices.

2.1.2 Nodo fuente y sumidero

Un nodo que sólo tiene arcos saliendo de él se denomina fuente y un nodo que sólo tiene arcos dirigidos hacia él se denomina sumidero.

2.1.3 Relaciones de adyacencia

Sea $G = \{V, A\}$ un grafo dirigido. Un nodo v_j se dice que es **adyacente** al nodo v_i si existe el arco (v_i, v_j) en dicho grafo. Este concepto es de

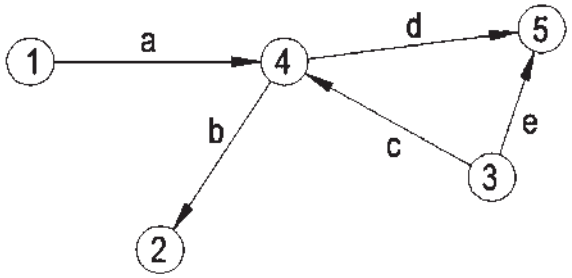
[4] En vez de círculos se pueden emplear rectángulos para representar los nodos.

particular importancia, dado que los grafos suelen representarse en la computadora por medio de matrices de adyacencias de nodos o de arcos.

2.1.3.1 Matriz de adyacencia de nodos. La matriz de adyacencia de nodos de *G* es la matriz $A = (a_{ij})$ de orden $n \times n$ definida por:

$$a_{ij} = \begin{cases} 1, & \text{si } v_j \text{ es adyacente en } v_i \\ 0, & \text{en caso contrario} \end{cases} \qquad (2.1)$$

En esta matriz los nodos están representados tanto por las filas como por las columnas, por lo que es una matriz cuadrada con dimensión igual al número de nodos del dígrafo. La Figura 2.2 es un ejemplo de la correspondencia entre un grafo dirigido de 5 nodos y su matriz de adyacencia 5 por 5. Una forma simple de ver la información guardada en la matriz de adyacencia de nodos es que los renglones de la misma representan el origen y las columnas el destino de cada arco en el grafo. De acuerdo a la ecuación (2.1), como se muestra en la Figura 2.2, se pone un cero en la celda del renglón *i*, columna *j* de la matriz cuando no hay conexión entre los nodos *i* y *j*; por otro lado, se introduce un uno cuando existe un arco en el grafo en la dirección *i* a *j*. Por esta característica, la matriz de adyacencia de nodos también se denomina matriz de conexión del dígrafo.



Nodos – Origen \ Nodos - Destino	1	2	3	4	5
1	0	0	0	1	0
2	0	0	0	0	0
3	0	0	0	1	1
4	0	1	0	0	1
5	0	0	0	0	0

Figura 2.2. Grafo Dirigido y su Matriz de Adyacencia de Nodos

El renglón *i* de la matriz de adyacencia *A* contiene todos los nodos que son alcanzables al trazar un camino[5] de una sola etapa a partir del

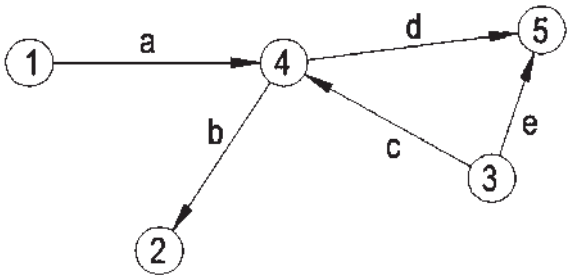
[5] En la sección 2.1.6 se presenta la definición de "camino" en un grafo.

nodo *i*; por tanto, una fila contiene todos los nodos que son los sucesores inmediatos (descendientes de primera generación) del nodo *i*. De igual manera, la columna *j* contiene todos los nodos que preceden de forma inmediata al nodo *j*. Si en la matriz de adyacencia *A* aparece una columna que sólo contiene ceros, el nodo correspondiente a esa columna no tiene precursores y, por consiguiente, no tiene arcos de entrada. Ese nodo sólo podría tener arcos de salida. De igual forma, si en la matriz de adyacencia *A* se encuentra un renglón con ceros solamente, entonces no hay arcos que salgan del nodo correspondiente a ese renglón a otros nodos.

2.1.3.2 Matriz de adyacencia de arcos (Mah, 1990). La matriz de adyacencia de arcos de *G* es la matriz $B = (b_{ij})$ de orden $m \times m$ definida por:

$$b_{ij} = \begin{cases} 1, & \text{si } a_i \text{ entra a un nodo y } a_j \text{ sale del mismo nodo} \\ 0, & \text{en caso contrario} \end{cases} \qquad (2.2)$$

Las filas y las columnas de la matriz representan los arcos del grafo. Al elemento b_{ij} de la matriz se le asigna un valor de uno si el arco *i* es incidente (entra) a un nodo y el arco *j* sale de ese mismo nodo. Por ejemplo, la Figura 2.3 muestra un dígrafo y su matriz de adyacencia de arcos. La principal ventaja de esta matriz es que permite la representación de autociclos y múltiples arcos entre dos nodos (Mah, 1990).



Arcos – Origen \ Arcos - Destino	a	b	c	d	e
a	0	1	0	1	0
b	0	0	0	0	0
c	0	1	0	1	0
d	0	0	0	0	0
e	0	0	0	0	0

Figura 2.3 Grafo Dirigido y su correspondiente Matriz de Adyacencia de Arcos

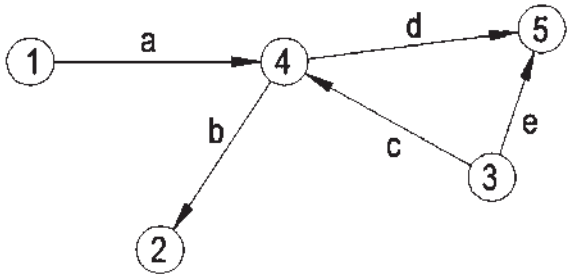
2.1.4 Relaciones de incidencia

Sea $G = \{V, A\}$ un grafo dirigido. Si $a_k = (v_i, v_j) \in A$, se dice que el arco a_k **incide desde** v_i (sale de ...) e **incide en** v_j. (llega a ...).

2.1.4.1 Matriz de incidencia. También se acostumbra representar a un grafo por medio de su matriz de incidencia asociada. En esta matriz cada nodo es representado por un renglón *i* y cada arco por una columna *j*. La matriz de incidencia de *G* es la matriz $C = (c_{ij})$ de orden $n \times m$ definida por

$$c_{ij} = \begin{cases} +1, & \text{si el arco } j \text{ es una entrada (incide en) al nodo } i \\ -1, & \text{si el arco } j \text{ es una salida (incide desde) del nodo } i \\ 0, & \text{en los demás casos o si el arco es un auto - ciclo} \end{cases} \qquad (2.3)$$

En la Figura 2.4 se representa un dígrafo y su correspondiente matriz de incidencia. La representación +1 y -1 a lo largo de los renglones se refiere, respectivamente, a los arcos que entran al y salen del nodo *i*.



Nodos	Arcos a	b	c	d	e
1	-1	0	0	0	0
2	0	+1	0	0	0
3	0	0	-1	0	-1
4	+1	-1	+1	-1	0
5	0	0	0	+1	+1

Figura 2.4 Grafo Dirigido y su Matriz de Incidencia

La matriz de incidencia también se utiliza para representar la información estructural contenida en un modelo matemático, esto es, para identificar cuáles variables están asociadas con cada una de las ecuaciones. En este caso, la matriz de incidencia de un modelo estructural coloca en las filas los números de las ecuaciones y en las columnas las variables independientes.

2.1.5 Grado de un vértice y de un grafo.

El grado de un vértice v en un grafo dirigido G es la suma de:

- *Grado de salida,* $d_o(v)$. El número de arcos que salen de él.
- *Grado de salida,* $d_i(v)$. El número de arcos que entran en él.

El nodo con mayor grado en el grafo le otorga el grado a dicho grafo.

2.1.6 Caminos

Un camino de longitud k desde u a u' en un grafo $G = \{V, A\}$ es una secuencia finita de nodos consecutivos $\langle v_0, v_1, ..., v_k \rangle$ tal que:

- $v_0 = u$ y $v_k = u'$.
- $\forall\, i : 1 \ldots k : (v_{i-1}, v_i) \in A$ (v_i es adyacente a v_{i-1})
- La longitud k del camino es el número de arcos.

Si hay un camino P desde u hasta u', se dice que u' es alcanzable desde u vía P.

2.1.7 Caminos simples y ciclos

Un **ciclo** es un camino $\langle v_0, v_1, ..., v_k \rangle$ que:

- Empieza y acaba en el mismo vértice ($v_0 = v_k$).
- Contiene al menos un arco.

Un camino es **simple** si todos sus nodos son distintos.

Un **bucle** es un ciclo de longitud 1.

Un grafo es **acíclico** si no contiene ciclos.

2.1.8 Subgrafo, subgrafo fuertemente conectado y componente fuerte

Un **subgrafo** SG de G es un grafo constituido por un subconjunto SV de V y un subconjunto SA de A que conectan los nodos de SV.

Se dice que un dígrafo G es **fuertemente conectado** si existe un camino desde u a v y un camino desde v a u para cada par distinto de

nodos u y v, con $u, v \in V$. Si un dígrafo no está fuertemente conectado entonces puede ser partido en subgrafos fuertemente conectados.

Un subgrafo es un **componente fuerte** de G si es fuertemente conectado y no puede ser agrandado a otro subgrafo mediante la adición de nodos extras y sus correspondientes arcos. Cada nodo puede pertenecer solamente a un componente fuerte (que puede estar constituido por un nodo simple) y todos los nodos de cualquier ciclo de un grafo deben estar en el mismo componente fuerte. En consecuencia, los componentes fuertes definen una partición del grafo. Al menos debe haber un componente fuerte en un grafo de modo tal que no haya un camino de cualquiera de sus nodos a cualquier nodo de otro componente fuerte.

2.2 Diagrama de flujo de información de un proceso

Los dígrafos resultan apropiados para visualizar la estructura de interconexión existente entre distintos objetos o tareas de procesamiento y, por tanto, se utilizan para representar procesos químicos continuos.

El diagrama modular de un proceso químico se puede transformar en un grafo dirigido, donde los nodos representan los distintos módulos de simulación, los arcos representan las corrientes y los ciclos (componentes fuertes) representan los lazos de recirculación. El dígrafo de un proceso químico también se conoce como **diagrama de flujo de información** (DFI) del proceso, en virtud de que transforma los módulos en nodos hacia donde llegan y de donde salen corrientes de información (antes de proceso), que comúnmente son líneas dirigidas de los flujos de materia y energía del proceso. Es necesario tener presente que el diagrama modular y el diagrama de flujo de información son idénticos sólo en el caso especial en que se especifica la simulación en modo análisis del proceso.

La Figura 2.5 muestra un ejemplo de un proceso simple y su correspondiente dígrafo.

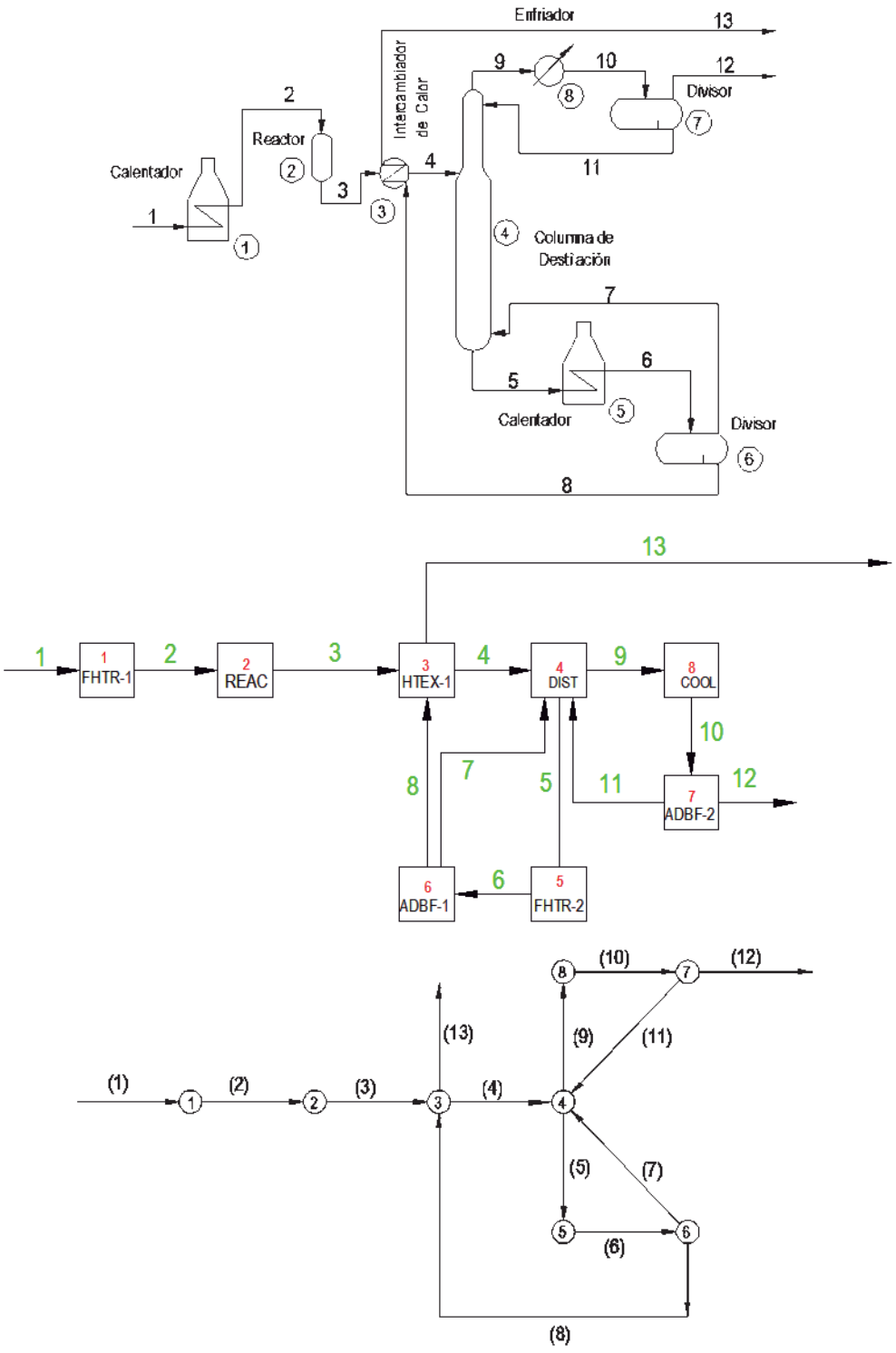


Figura 2.5 Diagrama de Flujo de Información de un Proceso Simple.

Para problemas donde el número de nodos y arcos es pequeño, la representación gráfica del proceso resulta adecuada para llevar a cabo las

tareas de descomposición y rasgado de procesos. Sin embargo, cuando un DFI tiene un número de nodos y arcos muy grande, es necesario representar los dígrafos de tal forma que puedan ser manipulados por una computadora. Para satisfacer este requerimiento, según se consideró en las secciones 2.1.3 y 2.1.4, se recurre a la estrecha relación que existe entre los grafos y la teoría de matrices, especialmente ralas o dispersas, la cual se basa en el carácter binario de los arcos. Por lo tanto, la información de los DFI del proceso se puede almacenar en diversos tipos de matrices, cuyos elementos contienen los valores de las variables denominadas estructurales.

2.3 Pre-procesamiento de procesos químicos

El pre-procesamiento del diagrama modular tiene como propósito la determinación del orden de resolución de los distintos módulos del diagrama de simulación. Como se muestra en la Figura 2.6, el procedimiento requerido para encontrar tal orden de resolución se puede realizar en tres fases.

Para iniciar el procedimiento se proporciona la información del DFI del proceso en forma gráfica o matricial. Luego, si existen corrientes de recirculación, se procede a la descomposición del diagrama de simulación. Para ello se requiere un método de búsqueda de ciclos que implica el recorrido de los nodos del dígrafo. La mayoría de los algoritmos de grafos requieren de un método sistemático para visitar sus nodos. En particular, la **búsqueda de profundidad**, conocida en inglés como **Depth - First Search** (**DFS**), es una técnica ampliamente usada que posee características que contribuyen a obtener algoritmos muy eficientes (Sargent y Westerberg, 1964; Tarjan, 1972). La técnica DFS sobre grafos es el algoritmo seleccionado en este trabajo para determinar los componentes fuertes o los ciclos de un dígrafo.

Una vez que se ha llevado a cabo el particionado se encuentran los pseudonodos, los cuales agrupan módulos conectados por lazos de

recirculación y que, por consiguiente, deben ser resueltos por separado en forma iterativa. Esto implica suponer una o más corrientes para poder establecer los ciclos iterativos necesarios para resolver todo el diagrama modular del proceso. El propósito de esta segunda tarea de la fase de pre-procesamiento, conocida comúnmente como **rasgado**, es encontrar las corrientes de corte que minimicen el tiempo requerido para simular los pseudonodos. En este trabajo se utiliza el algoritmo de rasgado de Ollero y Amselem (1983), que es un algoritmo sencillo, rápido y que garantiza el menor número de corrientes de corte (Mah, 1990).

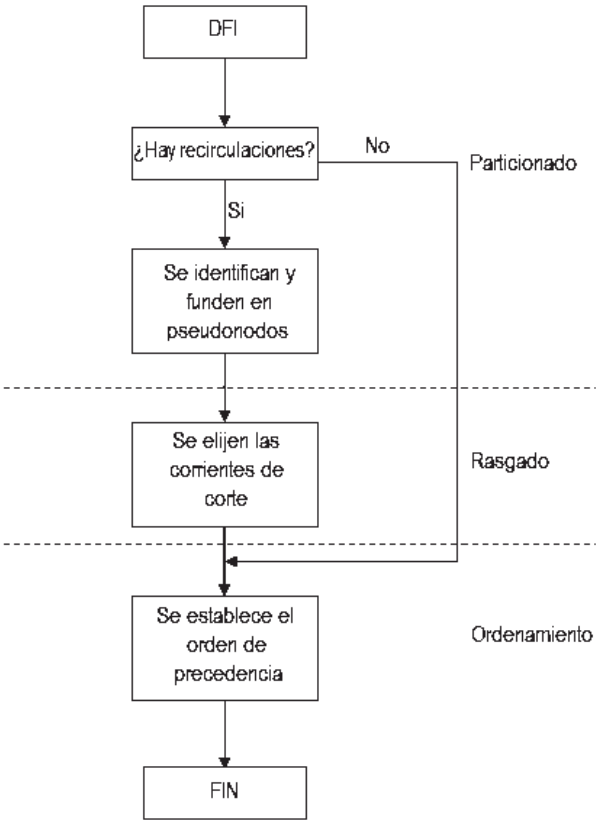


Figura 2.6. Pre-procesamiento del Diagrama Modular.

2.4 Descomposición y Ordenamiento de DFIs: Algoritmo DFS

El algoritmo DFS es el método más simple y efectivo disponible para la descomposición y ordenamiento de los DFI de procesos. La aplicación de este algoritmo se ilustra utilizando el DFI de la Figura 2.7a, que está

formado por 12 nodos (módulos de simulación) y cinco corrientes de recirculación. Como una regla, es preciso transformar, en primer lugar, el DFI del proceso en su correspondiente *dígrafo reducido*. Como se muestra en la Figura 2.7b, en éste sólo se retienen las corrientes intermedias entre los nodos y, por tanto, no se incluyen las corrientes de entrada al y salida del proceso.

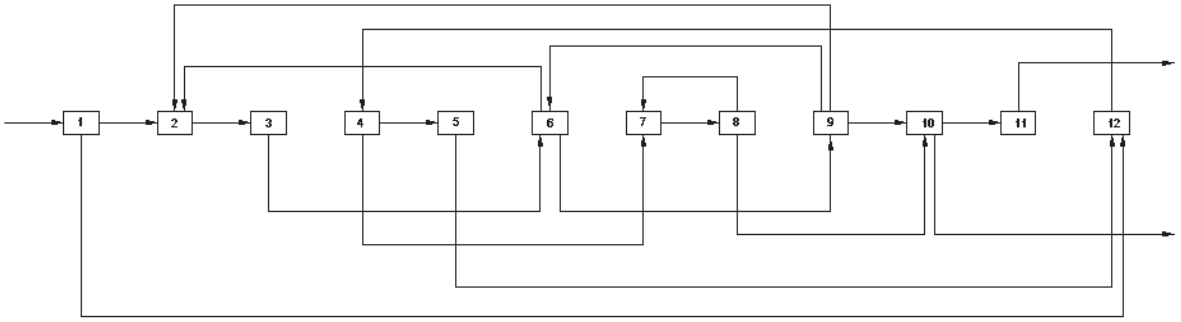


Figura 2.7a. DFI de un Proceso Hipotético.

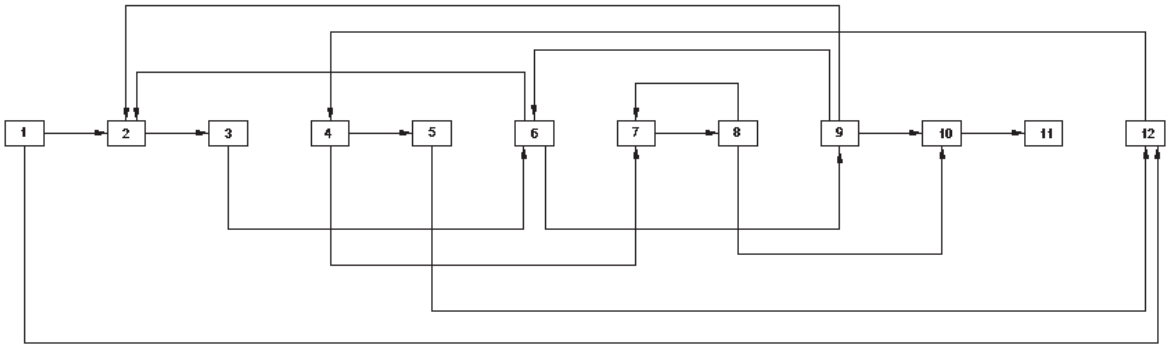


Figura 2.7b. Dígrafo Reducido del DFI del Proceso Hipotético de la Fig. 2.5a.

El algoritmo DFS se aplica al dígrafo reducido de un proceso y consiste en los siguientes pasos (Duff y Reid, 1978; Sargent y Westerberg, 1964; Westerberg y col., 1979):

Paso 1. Seleccionar un nodo de forma arbitraria.

Paso 2. Trazar un camino desde el nodo seleccionado siguiendo arcos no recorridos en la dirección de flujo del proceso. De esta manera se va formando una secuencia finita de nodos consecutivos, hasta que:

a) Se encuentra un nodo que ya está en el conjunto de nodos del camino. En este caso, todos los nodos que se encuentran entre los nodos repetidos junto con el nodo repetido forman un ciclo que los agrupa. Por lo tanto, los nodos del ciclo son reemplazados por un sólo "nodo compuesto" o pseudonodo, cuyas entradas y salidas son las mismas que conectan a los nodos agrupados con otros nodos del DFI. El trazado del camino se continúa desde el pseudonodo así formado.

b) Se encuentra un nodo o bloque de nodos (pseudonodo) que no tiene arcos de salida. Tal nodo o pseudonodo y sus arcos de entrada son borrados del DFI. El nombre del nodo o pseudonodo borrado se traslada del camino a la parte superior de una lista de bloques, lo que equivale a su total eliminación del problema. El recorrido del DFI se continúa para explorar los arcos no recorridos, comenzando desde el nodo o pseudonodo anterior al nodo eliminado.

El proceso de eliminación y agrupación de nodos continúa hasta que el camino haya quedado vacío. Cuando esto suceda, ir al paso 3.

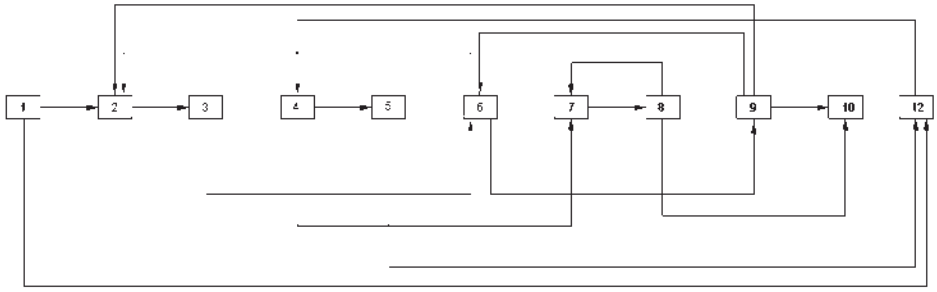
Paso 3. Si queda algún nodo en el dígrafo remanente, entonces trazar un nuevo camino repitiendo el procedimiento desde el paso uno. En caso contrario, el algoritmo de búsqueda de profundidad concluye.

Solución.

De acuerdo al Paso 1 del algoritmo DFS, el nodo 1 es elegido para trazar el siguiente camino:

Camino: $1 \rightarrow 2 \rightarrow 3 \rightarrow 6 \rightarrow 9 \rightarrow 10 \rightarrow 11$ (caso b)

El camino termina en el nodo 11, debido a que no tiene arcos de salida. Por lo tanto, el nodo 11 debe borrarse del DFI, de conformidad a lo establecido en el inciso b del Paso 2 del algoritmo DFS, y su nombre se traslada a la lista de bloques. Estas acciones dan lugar a la Figura 2.8, que muestra la parte remanente del DFI del proceso bajo estudio.



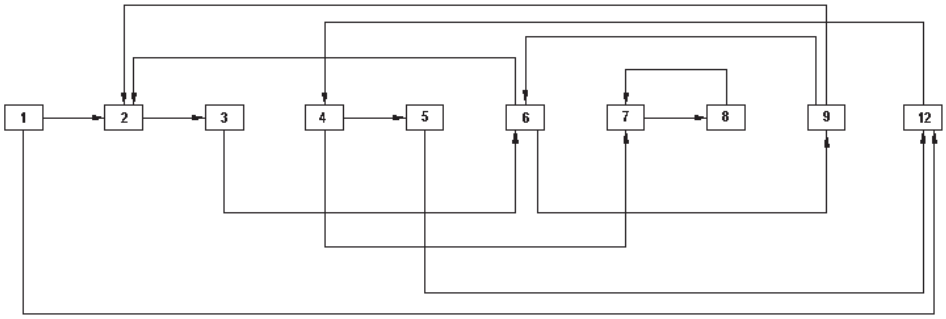
Lista de Bloques
11

Figura 2.8. Dígrafo Resultante de la Eliminación del Nodo 11.

El recorrido de arcos se continúa, comenzando en el nodo 10 de la Figura 2.8 desde el cual el nodo 11 fue descubierto:

Camino: $1 \rightarrow 2 \rightarrow 3 \rightarrow 6 \rightarrow 9 \rightarrow 10$ (caso b)

El nodo 10 no tiene arcos de salida, por lo que también debe eliminarse del DFI y colocarse en la parte superior de la lista de bloques (caso b del Paso 2). La Figura 2.9 muestra el dígrafo resultante.



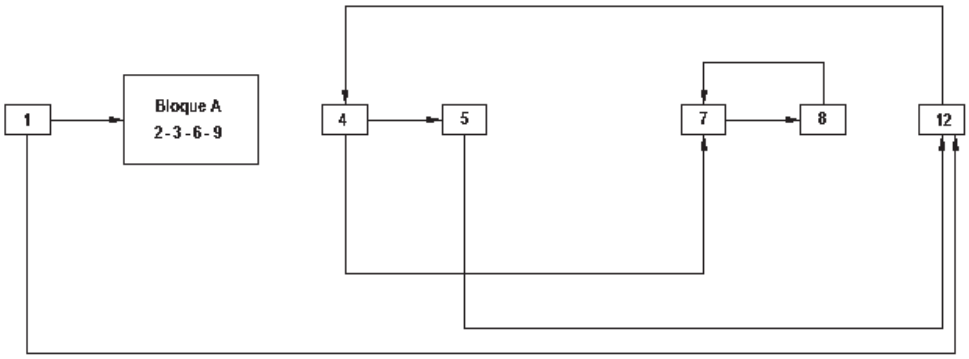
Lista de Bloques
10
11

Figura 2.9. Dígrafo Resultante de la Eliminación del Nodo 10.

La exploración de arcos se reinicia en el nodo 9 de la Figura 2.9:

Camino: $1 \rightarrow 2 \rightarrow 3 \rightarrow 6 \rightarrow 9 \rightarrow 2$ (caso a)

En este camino reaparece la unidad 2, por lo que se ha identificado un ciclo constituido por los nodos 2, 3, 6 y 9, los cuales se agrupan para formar un pseudonodo o bloque A (caso a del Paso 2). Este pseudonodo no tiene arcos de salida y como arco de entrada al mismo sólo se retiene el arco que se dirige desde el nodo 1 al nodo 2. El correspondiente dígrafo se muestra en la Figura 2.10.



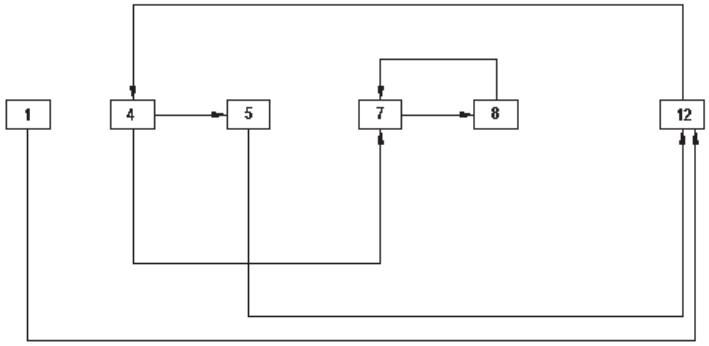
Lista de Bloques
10
11

Figura 2.10. Agrupación de los Nodos 2-3-6-9.

El recorrido se continúa en el grafo de la Figura 2.10 desde el pseudonodo encontrado:

Camino: 1 → A (caso b)

El bloque A debe ser borrado del grafo debido a que no tiene arcos de salida (caso b del Paso 2). La Figura 2.11 muestra el dígrafo resultante. La siguiente entrada a la parte superior de la lista de bloques es el bloque A.



Lista de Bloques
A
10
11

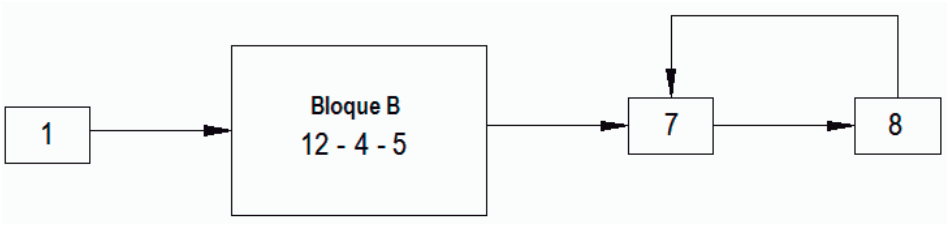
Figura 2.11. Dígrafo Resultante de la Eliminación del Bloque A.

De acuerdo al algoritmo DFS, el recorrido debe continuar en el nodo 1 de la Figura 2.11, dando lugar al siguiente camino:

Camino: 1 → 12 → 4 → 5 → 12 (caso a)

donde el nodo 12 repite. Por lo tanto, se ha encontrado un ciclo constituido por los nodos 12, 4 y 5. Estos nodos se agrupan para formar el pseudonodo denominado bloque B en la Figura 2.12, donde se puede

observar que sólo los arcos que se dirigen o salen de alguno de los nodos pertenecientes al bloque B desde o hacia nodos externos se conservan como arcos de entrada o salida del mismo.



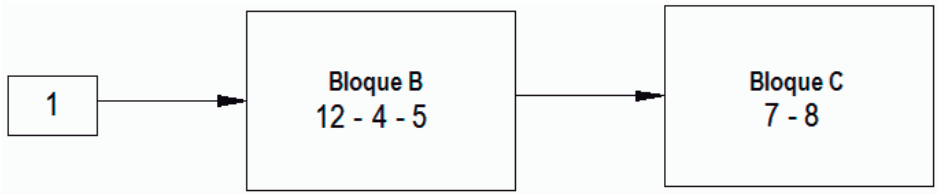
Lista de Bloques
A
10
11

Figura 2.12. Agrupación de los Nodos 12-4-5.

El recorrido se continúa en el bloque B, para obtener el siguiente camino:

Camino: 1 → B → 7 → 8 → 7 (caso a)

La unidad 7 repite en el camino, por lo que se ha encontrado un nuevo ciclo formado por los nodos 7 y 8. La agrupación de estos nodos en un nuevo pseudonodo proporciona la Figura 2.13.



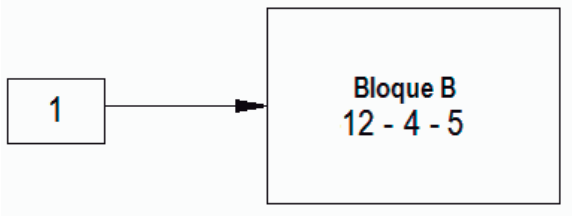
Lista de Bloques
A
10
11

Figura 2.13. Agrupación de los Nodos 7-8.

La exploración de los arcos continúa en el pseudonodo creado o bloque C de la Figura 2.13:

Camino: 1 → B → C (caso b)

Como se puede ver en la Figura 2.13, el bloque C no tiene arcos de salida, por lo que se suprime del grafo y se coloca en la parte superior de la lista de bloques. El resultado de estas manipulaciones se presenta en la Figura 2.14.

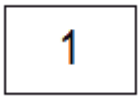


Lista de Bloques
C
A
10
11

Figura 2.14. Dígrafo Resultante de la Eliminación del Bloque C.

Camino: 1 → B (caso b)

El bloque B tampoco tiene arcos de salida; en consecuencia, se elimina del DFI remanente y pasa a encabezar la lista de bloques, como se muestra en la Figura 2.15.



Lista de Bloques
B
C
A
10
11

Figura 2.15. Dígrafo Resultante de la Eliminación del Bloque B.

Finalmente, se continúa el trazado desde el nodo 1 de la Figura 2.15:
Camino: 1 (caso b)

El nodo 1 es borrado del grafo porque no tiene salidas y su nombre se registra en la parte superior de la lista de bloques.

Lista de Bloques
1
B
C
A
10
11

El procedimiento de búsqueda en profundidad ha terminado, debido a que el camino ha quedado vacío y no hay más nodos.

A partir de la lista final de bloques se puede preparar la Figura 2.16, que muestra el resultado de la descomposición del diagrama modular de la

Figura 2.7a y la secuencia global de cálculo de los bloques encontrados. Se puede observar que no es necesario resolver simultáneamente los 12 módulos y las 5 corrientes de recirculación del DFI original, en virtud de que el proceso se ha dividido en 6 subsistemas más pequeños y manejables para facilitar su análisis, tres de los cuales constan de un sólo módulo y los otros tres constan de pequeños bloques de módulos. El más grande de estos bloques contiene cuatro módulos y tres corrientes de recirculación. En consecuencia, el esfuerzo de cálculo se puede reducir sustancialmente al resolver los subsistemas en forma independiente en el orden claramente definido, de izquierda a derecha, por la Figura 2.16: 1 → 12 4 5 (B) → 7 8 (C) → 2 3 6 9 (A) → 10 → 11. Conviene destacar que el orden de los cálculos de los bloques encontrados es opuesto al orden en el que fueron borrados del DFI.

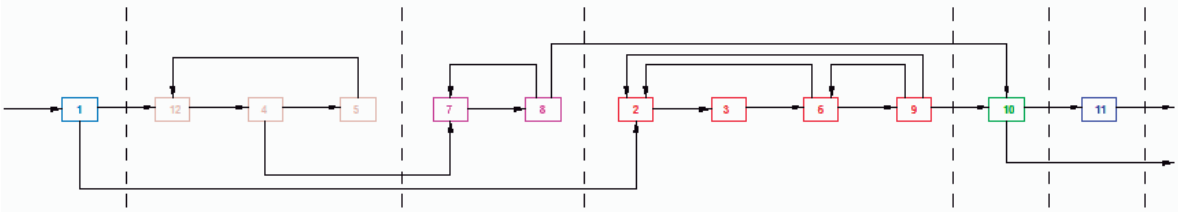


Figura 2.16. Diagrama Modular Particionado.

Es importante tener presente que la descomposición del DFI es siempre única. Sin embargo, a diferencia del número y conformación de los bloques, el orden de resolución no es siempre único debido a que puede depender del orden en el que los arcos son recorridos. Por ejemplo, para este proceso también se puede obtener la siguiente secuencia de cálculo: 1 → 2 3 6 9 (A) → 12 4 5 (B) → 7 8 (C) → 10 → 11.

De la lista de bloques resultante también es posible obtener la estructura de un DFI acíclico (ver Figura 2.17), donde los pseudonodos son tratados como nodos simples. En consecuencia, el procedimiento de descomposición DFS transforma un DFI con ciclos en uno sin ciclos, es decir, produce un DFI linealizado constituido por nodos y/o pseudonodos.

A esta estructura acíclica le corresponde una estrategia de cálculo secuencial en el orden establecido por la lista final de bloques, teniendo presente que todos los nodos y arcos dentro de un pseudonodo tienen que ser resueltos juntos antes de proceder al siguiente subsistema de la lista de bloques.

Figura 2.17 DFI Acíclico.

A manera de conclusión de esta sección se puede indicar que la lista final de bloques proporciona:

- Los subsistemas o conjuntos de nodos independientes, que pueden estar constituidos por un sólo nodo o por bloques de nodos que integran los pseudonodos.
- El orden global de resolución o precedencia de cálculos de los conjuntos de módulos independientes, de arriba hacia abajo.

El algoritmo DFS tiene la gran virtud de que cada arco del grafo original tiene que ser explorado sólo una vez. En términos de la teoría de grafos, el algoritmo DFS identifica los componentes fuertes de un dígrafo y reemplaza cada componente fuerte por un nodo simple. El dígrafo acíclico obtenido (ver Figura 2.17), comúnmente conocido como dígrafo condensado, tiene una matriz de adyacencia de nodos superior triangular. Esta característica define completamente la secuencia de cálculo de los bloques del dígrafo condensado (Mah, 1990).

2.5 Rasgado de Pseudonodos: Algoritmo de Ollero–Amselem (OA)

Si ninguno de los bloques encontrado por el algoritmo DFS contiene más de un nodo, entonces el DFI no tiene ciclos y su simulación es simple. Si este no es el caso, cada uno de los pseudonodos o bloques conteniendo

dos o más nodos deben ser resueltos simultáneamente, debido a que cada nodo en el bloque influencia a todos los demás nodos en ese bloque porque están conectados por corrientes de recirculación. Por consiguiente, no hay modo de dividir el problema de simulación en subproblemas más pequeños. La resolución simultánea de este problema aparentemente es sencilla usando las computadoras y técnicas numéricas actualmente disponibles; no obstante, implica un esfuerzo de cálculo considerable si se compara con una estrategia de cálculo secuencial, debido a que la dificultad de resolver un conjunto de ecuaciones aumenta proporcionalmente con el cubo del número de ellas que deben manejarse simultáneamente (Rudd y Watson, 1968; Mah, 1990). Además, si los valores de los estimados iniciales están lejos de la solución, entonces el método numérico utilizado puede sufrir problemas de divergencia. Por lo tanto, es deseable encontrar un conjunto mínimo de corrientes de corte que permitan el rompimiento de los ciclos y, en consecuencia, la simulación secuencial e iterativa de los bloques de nodos. El proceso de rompimiento de ciclos es conocido comúnmente como **rasgado**.

En este trabajo se utiliza el algoritmo de rasgado de Ollero y Amselem (1983), el cual opera sobre el *grafo dual* propuesto por Barkley y Motard (1972). La Figura 2.18 muestra el diagrama modular de un proceso simple y la Figura 2.19 muestra su correspondiente grafo dual, que también es un dígrafo. En el DFI los nodos representan módulos y los arcos representan corrientes de proceso. En el grafo dual, los nodos representan corrientes de proceso y los arcos representan flujos de información.

La matriz de adyacencia de arcos, descrita en la Sección 2.1.2.3, es la matriz de adyacencia de nodos-corrientes asociada a un grafo dual que muestra la relación entre las corrientes de un proceso. Su principal ventaja es que permite representar tanto arcos paralelos y autociclos. En esta matriz, como se muestra en la Tabla 2.1 para el grafo dual de la Figura 2.19, los nodos-corrientes son representados por las filas y las

columnas. Para denotar la existencia de un arco del nodo-corriente *i* al nodo-corriente *j* en el grafo dual se pone un uno en la posición *i,j* de la matriz de adyacencia de nodos-corrientes. De esta manera se indica que en el DFI la corriente *j* es una salida física de un módulo para el cual la corriente *i* es una entrada física y, en consecuencia, que los valores de la corriente *i* son necesarios para determinar los valores de la corriente *j* en la simulación en modo análisis del correspondiente proceso. Por lo tanto, para cada nodo *v* del DFI habrá $d_i(v)$ x $d_o(v)$ arcos en el grafo dual.

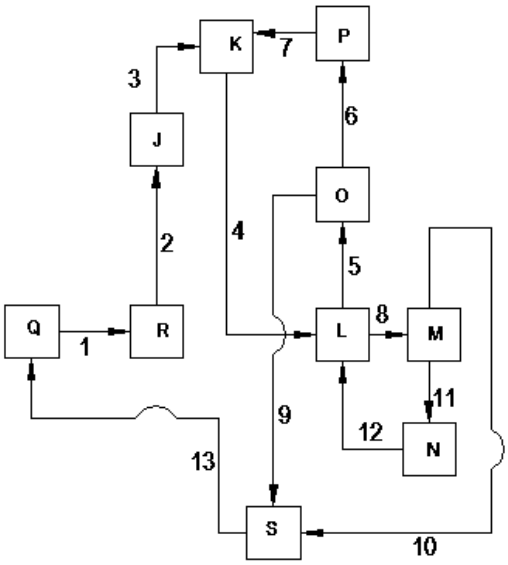


Figura 2.18 Diagrama Modular de un Proceso con las Corrientes Numeradas.

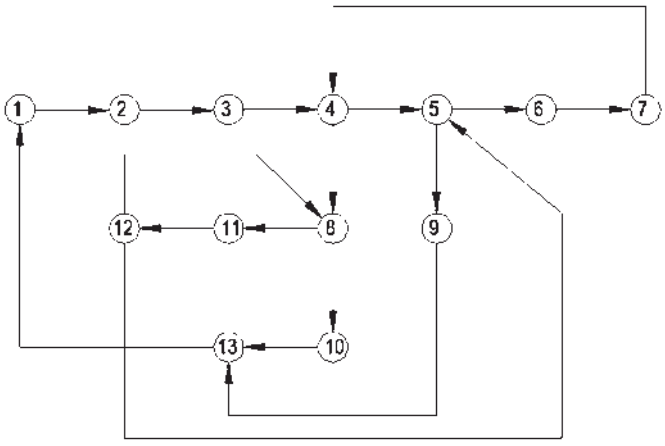


Figura 2.19 Grafo Dual del DFI de la Figura 2.18

Tabla 2.1 Matriz de Adyacencia de Nodos del Grafo Dual

	1	2	3	4	5	6	7	8	9	10	11	12	13	d_o
1		1												1
2			1											1
3				1										1
4					1			1						2
5						1			1					2
6							1							1
7				1										1
8										1	1			2
9													1	1
10													1	1
11												1		1
12					1			1						2
13	1													1
d_i	1	1	1	2	2	1	1	2	1	1	1	1	2	

El algoritmo OA simplifica el grafo dual mediante la eliminación de corrientes que pueden ser excluidas del conjunto de corte óptimo (corrientes inelegibles) y de corrientes que deben ser incluidas en el conjunto de corte óptimo (corrientes esenciales). Para lograrlo supone dos tipos de simplificaciones:

a).- Reducción de nodos-corrientes esenciales o eliminación de nodos. En esta simplificación un nodo candidato v y todos sus arcos asociados son eliminados del grafo. Es equivalente a borrar la fila y la columna asociadas con el nodo v en la matriz de adyacencia.

b).- Reducción de nodos-corrientes no esenciales o contracción. En esta simplificación, el nodo candidato v es borrado, pero se agrega un nuevo bloque de nodos que conectan a los **ancestros (nodos predecesores) con los descendientes (nodos sucesores) del nodo borrado.** En términos de la matriz de adyacencia de nodos-corrientes, es equivalente a borrar la columna v y la fila v, pero haciendo el elemento $x_{i,j}$ igual a "1" siempre que $x_{i,v}$ y $x_{v,j}$ sean diferentes de cero.

La eliminación de nodos se aplica a dos tipos de nodos: (i) Nodos que no tienen corrientes de entrada ó corrientes de salida, (ii) Nodos con auto

ciclos resultantes de operaciones previas en el grafo. Los nodos del tipo (i) no forman parte del conjunto de corrientes de corte, pero los del tipo (ii) sí.

La contracción se inicia con nodos que tienen una sólo grado de entrada $d_i(v)$ o un sólo grado de salida $d_o(v)$, y continúa con nodos con $d_i(v)$ ó $d_o(v)$ = 2, 3, 4…. Después de cada contracción se examina el grafo para ver si es posible llevar a cabo una eliminación de nodo adicional. El algoritmo reduce el grafo dual a un grafo nulo alternando la contracción y la eliminación de nodos.

Antes de comenzar con el algoritmo se debe de realizar el grafo dual, donde los arcos del DFI debidamente numerados se transforman en nodos-corrientes y los arcos que los unen son los módulos. A partir del grafo dual podemos obtener la matriz de adyacencia de nodos.

Los pasos que conforman el algoritmo OA son los siguientes:

Paso 1. Se calculan el grado de entrada $d_i(v)$ y salida $d_o(v)$ de cada uno de los nodos v. Esto se hace sumando los elementos no nulos de las columnas y filas de la matriz; la suma total de cada columna corresponde al grado de entrada $d_i(v)$ y la de las filas al grado de salida $d_o(v)$ del respectivo nodo. Otra opción es contar directamente en el grafo dual los arcos que entran y salen de cada nodo.

Paso 2. Empezando con el grafo dual G o la matriz de adyacencia, se hace ES = 1. El parámetro ES especifica el grado de entrada o el grado de salida de un nodo.

Paso 3. Se examina un nodo v en G.

a) Si $d_i(v)$ o $d_o(v)$ = 0, borrar el nodo v y avanzar al paso *4*.

b) Si tiene un auto-ciclo, seleccionar el nodo v como una corriente de corte, borrar v de G y proceder al paso *4*.

c) Si $d_i(v)$ o $d_o(v)$ = ES, aplicar una contracción al nodo v. De lo contrario ir al paso *5*.

Paso 4. Parar si la matriz está vacía.

Paso 5. Ir al paso *3* hasta que todos los nodos hayan sido examinados para el valor actual de ES.

Paso 6. Si al menos un vértice ha sido borrado en el paso *3(b)*, hacer ES = 1 y regresar al paso *3*. De lo contrario hacer ES = ES + 1 y volver al paso *3*.

Para ilustrar el procedimiento considere el DFI de la Figura 2.18, en la cual se observa que a cada corriente de proceso se le ha asignado un número (si es el caso, es necesario numerar las corrientes de proceso). El grafo dual de ese DFI se muestra en la 2.19 y la matriz de adyacencia de nodos-corrientes del grafo dual se presenta en la Tabla 2.1, en la cual se han agregado el grado de salida $d_i(v)$ y el grado de entrada $d_i(v)$ de cada nodo, en la última columna y fila, respectivamente.

Nodo 1

$d_i = ES = 1$

Aplicamos una contracción al nodo "1", para hacerlo ubicamos la posición de los elementos no nulos tanto en la columna "1" como en la fila "1" este paso lo podemos observar en la Tabla 2.2a.

Tabla 2.2a. Ubicación de Elementos No Nulos Para la Contracción del Nodo 1

	1	2	3	4	5	6	7	8	9	10	11	12	13	d_o
1		1												1
2			1											1
3				1										1
4					1			1						2
5						1			1					2
6							1							1
7				1										1
8										1	1			2
9													1	1
10													1	1
11												1		1
12					1			1						2
13	1													1
d_i	1	1	1	2	2	1	1	2	1	1	1	1	2	

Columna

$x_{i,v} = 13,1$

Fila

$x_{v,j} = 1,2$

Los valores anteriores hacen referencia a la posición de los elementos no nulos en la matriz de adyacencia de nodos del grafo dual; con dichos valores podemos generar las coordenadas para una nueva posición que en este caso es:

$x_{i,j} = 13,2$

Con el valor encontrado ubicamos las nuevas coordenadas, si el nuevo elemento es **no nulo** en la matriz de adyacencia se hace igual a "1".

Borramos la fila y la columna 1 y calculamos nuevamente los grados de entrada y salida, este paso se puede observar en la Tabla 2.2b.

Tabla 2.2b. Contracción del Nodo 1

	2	3	4	5	6	7	8	9	10	11	12	13	d_o
2		1											1
3			1										1
4				1			1						2
5					1			1					2
6						1							1
7			1										1
8									1	1			2
9												1	1
10												1	1
11											1		1
12				1			1						2
13	1												1
d_i	1	1	2	2	1	1	2	1	1	1	1	2	

El elemento marcado en verde es el elemento que resultó de la contracción del nodo 1.

Nodo 2

$d_i = ES = 1$

Aplicamos la contracción al nodo 2; para ello ubicamos los elementos no nulos en la Fila y Columna 2 (Tabla 2.3a).

Tabla 2.3a. Ubicación de Elementos No Nulos Para la Contracción del Nodo 2

	2	3	4	5	6	7	8	9	10	11	12	13	*do*
2		1											1
3			1										1
4				1			1						2
5					1			1					2
6						1							1
7			1										1
8									1	1			2
9												1	1
10												1	1
11											1		1
12				1			1						2
13	1												1
di	1	1	2	2	1	1	2	1	1	1	1	2	

Columna

$x_{i,j} = 13,2$

Fila

$x_{v,j} = 2,3$

Las coordenadas del nuevo elemento son:

$x_{i,j} = 13,3$

El nuevo elemento se hace igual a “1”; borramos la fila y la columna 2 y recalculamos los grados de entrada y salida como se muestra en la Tabla 2.3b.

Tabla 2.3b. Contracción del Nodo 2

	3	4	5	6	7	8	9	10	11	12	13	*do*
3		1										1
4			1			1						2
5				1			1					2
6					1							1
7		1										1
8								1	1			2
9											1	1
10											1	1
11										1		1
12			1			1						2
13	1											1
di	1	2	2	1	1	2	1	1	1	1	2	

Nodo 3

$d_i = ES = 1$

Aplicamos la contracción al nodo 3. La tabla 2.4a nos muestra la ubicación de los elementos no nulos.

Tabla 2.4a. Ubicación de Elementos No Nulos Para la Contracción del Nodo 3

	3	4	5	6	7	8	9	10	11	12	13	*do*
3		1										1
4			1			1						2
5				1			1					2
6					1							1
7		1										1
8								1	1			2
9											1	1
10											1	1
11										1		1
12			1			1						2
13	1											1
di	1	2	2	1	1	2	1	1	1	1	2	

Columna

$x_{i,j} = 13,3$

Fila

$x_{v,j} = 3,4$

Las coordenadas del nuevo elemento son:

$x_{i,j} = 13,4$

El nuevo elemento se hace igual a "1". Se borran la fila y la columna 3 y se calculan nuevamente los grados de entrada y de salida como se muestra en la Tabla 2.4b.

Tabla 2.4b. Contracción del Nodo 3

	4	5	6	7	8	9	10	11	12	13	*do*
4		1			1						2
5			1			1					2
6				1							1
7	1										1
8							1	1			2
9										1	1
10										1	1
11									1		1
12		1			1						2
13	1										1
di	2	2	1	1	2	1	1	1	1	2	

Nodo 4

$d_o = 2$

$d_i = 2$

No se puede reducir este nodo, ya que no hay auto-ciclos; además los grados de entrada y salida son diferentes de cero y de ES, por lo que procedemos al análisis del nodo 5.

Nodo 5

$d_o = 2$

$d_i = 2$

No se puede reducir este nodo.

Nodo 6

$d_i = ES = 1$

Se le aplica la contracción al nodo 6; la tabla 2.5a nos muestra la ubicación de los elementos no nulos.

Tabla 2.5a. Ubicación de Elementos No Nulos Para la Contracción del Nodo 6

	4	5	6	7	8	9	10	11	12	13	*do*
4		1			1						2
5			1			1					2
6				1							1
7	1										1
8							1	1			2
9										1	1
10										1	1
11									1		1
12		1			1						2
13	1										1
di	2	2	1	1	2	1	1	1	1	2	

Columna

$x_{i,j} = 5,6$

Fila

$x_{v,j} = 6,7$

Las coordenadas del nuevo elemento son:

$x_{i,j} = 5,7$

Marcamos el nuevo elemento, borramos tanto la Fila como la Columna 6 y calculamos los grados de entrada y salida (Tabla 2.5b)

Tabla 2.5b. Contracción del Nodo 6

	4	5	7	8	9	10	11	12	13	*do*
4		1		1						2
5			1		1					2
7	1									1
8						1	1			2
9									1	1
10									1	1
11								1		1
12		1		1						2
13	1									1
di	2	2	1	2	1	1	1	1	2	

Nodo 7

$d_i = ES = 1$

La Tabla 2.6b nos muestra la ubicación de los elementos no nulos de la Fila y Columna 7

Tabla 2.6a. Ubicación de Elementos No Nulos Para la Contracción del Nodo 7

	4	5	7	8	9	10	11	12	13	*do*
4		1		1						2
5			1		1					2
7	1									1
8						1	1			2
9									1	1
10									1	1
11								1		1
12		1		1						2
13	1									1
di	2	2	1	2	1	1	1	1	2	

Columna

$x_{i,v} = 5,7$

Fila

$x_{v,j} = 7,4$

Las coordenadas del nuevo elemento son:

$x_{i,j} = 5,4$

Borramos la fila y la columna 7 y calculamos nuevamente los grados de entrada y salida como se muestra en la Tabla 2.6b.

Tabla 2.6b. Contracción del Nodo 7

	4	5	8	9	10	11	12	13	*do*
4		1	1						2
5	1			1					2
8					1	1			2
9								1	1
10								1	1
11							1		1
12		1	1						2
13	1								1
di	2	2	2	1	1	1	1	2	

Nodo 8

$d_o = 2$

$d_i = 2$

No se puede reducir este nodo.

Nodo 9

$d_i = ES = 1$

Ubicamos los elementos no nulos en la Fila y la Columna 9, esto lo podemos ver en la Tabla 27.a

Tabla 2.7a. Ubicación de Elementos No Nulos Para la Contracción del Nodo 9

	4	5	8	9	10	11	12	13	*do*
4		1	1						2
5	1			1					2
8					1	1			2
9								1	1
10								1	1
11							1		1
12		1	1						2
13	1								1
di	2	2	2	1	1	1	1	2	

Columna

$x_{i,v} = 5,9$

Fila

$x_{v,j} = 9,13$

Las coordenadas del nuevo elemento son:

$x_{i,j} = 5,13$

Ubicamos el nuevo elemento, borramos la fila y la columna 9 y calculamos nuevamente los grados de entrada y salida como se muestra en la Tabla 2.7b.

Tabla 2.7b. Contracción del Nodo 9

	4	5	8	10	11	12	13	*do*
4		1	1					2
5	1						1	2
8				1	1			2
10							1	1
11						1		1
12		1	1					2
13	1							1
di	2	2	2	1	1	1	2	

Nodo 10

$d_i = ES = 1$

Ubicamos los elementos no nulos en la fila y columna 10 (Tabla 2.8a).

Tabla 2.8a. Ubicación de Elementos No Nulos Para la Contracción del Nodo 10

	4	5	8	10	11	12	13	*do*
4		1	1					2
5	1						1	2
8				1	1			2
10							1	1
11						1		1
12		1	1					2
13	1							1
di	2	2	2	1	1	1	2	

Columna

$x_{i,v} = 8{,}10$

Fila

$x_{v,j} = 10{,}13$

Las coordenadas del nuevo elemento son:

$x_{i,j} = 8{,}13$

Se borran la fila y columna 10, se ubica el nuevo elemento y se calculan los grados de entrada y salida

Tabla 2.8b. Contracción del Nodo 10

	4	5	8	11	12	13	*do*
4		1	1				2
5	1					1	2
8				1		1	2
11					1		1
12		1	1				2
13	1						1
di	2	2	2	1	1	2	

Nodo 11

$d_i = ES = 1$

En la tabla 2.9a se pueden ver los elementos no nulos de la fila y columna 11.

Tabla 2.9a. Ubicación de Elementos No Nulos Para la Contracción del Nodo 11

	4	5	8	11	12	13	*do*
4		1	1				2
5	1					1	2
8				1		1	2
11					1		1
12		1	1				2
13	1						1
di	2	2	2	1	1	2	

Columna

$x_{i,v} = 8{,}11$

Fila

$x_{v,j} = 11{,}12$

Las coordenadas del nuevo elemento son:

$x_{i,j} = 8{,}12$

Se ubica el nuevo elemento, se borran las fila y columna 11 y se recalculan los grados de entrada y salida, este paso lo observamos a la Tabla 2.9b.

Tabla 2.9b. Contracción del Nodo 11

	4	5	8	12	13	do
4		1	1			2
5	1				1	2
8				1	1	2
12		1	1			2
13	1					1
di	2	2	2	1	2	

Nodo 12

$d_i = ES = 1$

La Tabla 2.10 nos muestra los elementos no nulos en la fila y columna 12.

Tabla 2.10a. Ubicación de Elementos No Nulos Para la Contracción del Nodo 12

	4	5	8	12	13	do
4		1	1			2
5	1				1	2
8				1	1	2
12		1	1			2
13	1					1
di	2	2	2	1	2	

Columna

$x_{i,v} = 8,12$

Fila

$x_{v,j1} = 12,5$

$x_{v,j2} = 12,8$

En este caso existen dos elementos no nulos en la fila examinada, por lo que también tendremos dos nuevos elementos que son:

$x_{i,j1} = 8,5$

$x_{i,j2} = 8,8$

Se ubican los nuevos elementos, se borran las fila y columna 12 y se recalculan los grados de entrada y salida, este paso lo observamos a la Tabla 2.10b.

Tabla 2.10b. Contracción del Nodo 12

	4	5	8	13	*do*
4		1	1		2
5	1			1	2
8		1	1	1	3
13	1				1
di	2	2	2	2	

En este paso se ha encontrado un auto-ciclo. Esto lo podemos ver como un elemento no nulo sobre la diagonal en la Tabla 2.10b; sin embargo, dicho auto–ciclo se encuentra en un nodo que ya ha sido examinado así que continuamos examinando los nodos restantes sin eliminarlo.

Nodo 13

$d_o = ES = 1$

Los elementos no nulos de la fila y columna 13 se muestran en la Tabla 2.11a

Tabla 2.11a. Ubicación de Elementos No Nulos Para la Contracción del Nodo 13

	4	5	8	13	*do*
4		1	1		2
5	1			1	2
8		1	1	1	3
13	1				1
di	2	2	2	2	

Columna

$x_{i1,v} = 5,13$

$x_{i2,v} = 8,13$

Fila

$x_{v,j} = 13,4$

En este caso existen dos elementos no nulos en la columna examinada, por lo que también tendremos dos nuevos elementos que son:

$x_{i,j1} = 5,4$

$x_{i,j2} = 8,4$

En la Tabal 2.11b observamos la eliminación la fila y columna 13, así como los nuevos elementos.

Tabla 2.11b. Contracción del Nodo 13

	4	5	8	*do*
4		1	1	2
5	1			1
8	1	1	1	3
di	2	2	2	

Una vez que todos los nodos han sido examinados hacemos ES = ES + 1 ya que no se eliminó ningún vértice en el paso 3(b) y comenzamos desde el primer nodo.

ES = 1 + 1 = 2

Nodo 4

$d_i = ES = 2$

Los elementos no nulos se muestran en la Tabla 2.12a:

Tabla 2.12a. Ubicación de Elementos No Nulos Para la Contracción del Nodo 4

	4	5	8	*do*
4		1	1	2
5	1			1
8	1	1	1	3
di	2	2	2	

Columna

$x_{i1,v} = 5,4$

$x_{i2,v} = 8,4$

Fila

$x_{v,j1} = 4,5$

$x_{v,j2} = 4,8$

En este caso se presentan dos elementos no nulos tanto en la fila como en la columna lo que genera cuatro nuevos elementos.

$x_{i,j1} = 5,5$

$x_{i,j2} = 5,8$

$x_{i,j3} = 8,5$

$x_{i,j4} = 8,8$

La eliminación de la fila y columna 4 y la ubicación de los nuevos elementos se muestran en la Tabla 2.12b.

Tabla 2.12b. Contracción del Nodo 4

	5	8	*do*
5	1	1	2
8	1	1	2
di	2	2	

Nodo 5

El nodo 5 presenta un Auto – Ciclo se agrega a las corrientes de corte y se borran la fila y la columna 5, esto lo podemos ver en la Tabla 2.13.

Tabla 2.13. Eliminación del Nodo 5

	8	*do*
8	1	1
di	1	

Nodo 5

El nodo 8 presenta un Auto – Ciclo se agrega a las corrientes de corte y se borran la fila y la columna 8, hemos llegado a una matriz vacía por lo que terminamos.

Las corrientes de corte son la 5 y 8, que coinciden con las corrientes de corte mostradas en la Figura 1.4.

2.6 Descomposición y rasgado de sistemas de ecuaciones

Las técnicas de particionado y rasgado también pueden ser aplicadas para resolver sistemas de ecuaciones. En este trabajo se utilizarán los algoritmos de ordenamiento de ecuaciones (OE) y agrupación de variables (AV), propuestos por Ramírez (1997), los cuales trabajan con la matriz de funcionalidad. Dicha matriz se forma a partir del conjunto de ecuaciones que conforman el modelo matemático del proceso; las filas

corresponden a las ecuaciones y las columnas a cada una de las variables. Al elemento (i,j) de la matriz se le asigna un valor diferente de cero si la ecuación *"i"* contiene la variable *"j"*.

La Tabla 2.14 es la matriz correspondiente a las ecuaciones que forman el modelo del proceso de evaporación súbita (flash) isotérmica de una mezcla binaria.

Ecuaciones

$$x_1 + x_2 = 1 \quad [1]$$

$$P = p_1 + p_2 \quad [2]$$

$$p_1^0 = p_1^0(t) \quad [3]$$

$$p_2^0 = p_2^0(t) \quad [4]$$

$$\gamma_1 = \gamma_1(x_1, x_2, t) \quad [5]$$

$$\gamma_2 = \gamma_2(x_1, x_2, t) \quad [6]$$

$$p_1 = \gamma_1 x_1 p_1^0 \quad [7]$$

$$p_2 = \gamma_2 x_2 p_2^0 \quad [8]$$

$$y_1 = p_1 / P \quad [9]$$

$$y_2 = p_2 / P \quad [10]$$

Tabla 2.14. Matriz de Funcionalidad Flash Isotérmico

	X_1	x_2	P	p_1	p_2	p_1^0	p_2^0	t	γ_2	γ_1	y_1	Y_2
	1	2	3	4	5	6	7	8	9	10	11	12
1	A	A										
2			A	A	A							
3						A		A				
4							A	A				
5	A	A						A	A			
6	A	A						A		A		
7	A			A		A			A			
8		A			A		A			A		
9			A	A							A	
10			A		A							A

El algoritmo de ordenamiento consiste en los siguientes pasos:

Paso 1. Se calculan los grados de libertad de las variables que, en la matriz de funcionalidad, corresponde a la cantidad de elementos no nulos para cada una de las columnas.

Paso 2. Se examina cada una de las columnas:

a) Si tiene un sólo grado de libertad, la fila que contiene el elemento no nulo es eliminada de la matriz.

b) Si no existe ninguna columna con un sólo grado de libertad, se escoge la columna con el menor número de grados de libertad; se eliminan las filas que contienen los elementos no nulos.

Paso 3. Después de cada eliminación se calculan nuevamente los grados de libertad y se vuelve al paso 2.

El algoritmo de agrupación de variables consiste en los siguientes pasos:

Paso 1. Se reconstruye la matriz de funcionalidad acomodando las filas en orden inverso a como fueron eliminadas con el algoritmo anterior.

Paso 2. Se calcula la frecuencia de las ecuaciones, esto es, el número de elementos no nulos en cada fila.

Paso 3. Se examina cada una de las filas:

a) Si se encuentra una fila con frecuencia igual a 1, se elimina la columna que contiene el elemento no nulo.

b) Si no existe ninguna fila con frecuencia igual a 1, se selecciona la fila que está en la parte superior de la matriz y se eliminan las columnas que contienen elementos no nulos; la fila seleccionada inicia un subgrupo.

Paso 4. Después de cada eliminación se calcula nuevamente la frecuencia de las filas y se vuelve al paso 2.

A continuación se resolverá el problema del flash isotérmico; primero se calculan los grados de libertad. Buscamos una columna con grado de libertad igual a 1. En este caso las columnas 11 y 12 cumplen dicha condición; se selecciona la 11 y se borra la fila que contiene el elemento no nulo, es decir, la fila 9. Se calculan nuevamente los grados de libertad y se examina la matriz; ahora seleccionamos la columna 11 y borramos la fila 10. Posteriormente se escoge la columna 4 para eliminar la fila 7, continuamos con la columna 5 para eliminar la fila 8 y la columna 6 para eliminar la fila 3. La Tabla 2.15 nos muestra todos los pasos.

Tabla 2.15. Aplicación del Algoritmo de Agrupación de Variables.

	X_1	x_2	P	p_1	p_2	p_1^0	P_2^0	t	γ_2	γ_1	y_1	y_2	
	1	2	3	4	5	6	7	8	9	10	11	12	
1	A	A											
2			A	A	A								
3						A		A					
4							A	A					
5	A	A						A	A				
6	A	A						A		A			
7	A			A		A			A				
8		A			A		A			A			
9			A	A							A		
10			A		A							A	
GL_0	4	4	3	3	3	2	2	4	2	2	1	1	Se elimina fila 9
GL_1	4	4	2	2	3	2	2	4	2	2	0	1	Se elimina fila 10
GL_2	4	4	1	2	2	2	2	4	2	2	0	0	Se elimina fila 2
GL_3	4	4	0	1	1	2	2	4	2	2	0	0	Se elimina fila 7
GL_4	3	4	0	0	1	1	2	4	1	2	0	0	Se elimina fila 8
GL_5	3	3	0	0	0	1	1	4	1	1	0	0	Se elimina fila 3
GL_6	3	3	0	0	0	0	1	3	1	1	0	0	Se elimina fila 4
GL_7	3	3	0	0	0	0	0	2	1	1	0	0	Se elimina fila 5
GL_8	2	2	0	0	0	0	0	1	0	1	0	0	Se elimina fila 6
GL_9	1	1	0	0	0	0	0	0	0	0	0	0	Se elimina fila 1
GL_{10}	0	0	0	0	0	0	0	0	0	0	0	0	

ente la matriz de fu

Una vez que se han eliminado todas las filas se forma nuevamente la matriz de funcionalidad pero con las filas en orden inverso a como fueron borradas. La Tabla 2.16 es la matriz resultante, después calculamos la frecuencia de cada una de las filas. Como ninguna fila tiene frecuencia 1,

se selecciona la primera fila y se borran las columnas que presenten elementos no nulos; dicha fila se marca como el inicio del subgrupo 1. Se recalcula la frecuencia de las filas. Nuevamente encontramos que ninguna fila tiene frecuencia 1; por lo tanto, seleccionamos la fila que queda más arriba en la matriz, que en este caso es la fila 6. Borramos las columnas 8 y 9 y marcamos dicha fila como inicio del subgrupo 2. Todos los pasos se ilustran en la Tabla 2.16.

Tabla 2.16. Matriz de Funcionalidad Para Algoritmo Agrupación.

	x_1	x_2	P	p_1	p_2	p_1^0	p_2^0	t	γ_2	γ_1	y_1	y_2											
	1	2	3	4	5	6	7	8	9	10	11	12	F_1	F_2	F_3	F_4	F_5	F_6	F_7	F_8	F_9	F_{10}	
1	A	A											2	0	0	0	0	0	0	0	0	0	0
6	A	A						A		A			4	2	0	0	0	0	0	0	0	0	0
5	A	A						A	A				4	2	1	0	0	0	0	0	0	0	0
4							A	A					2	2	1	1	0	0	0	0	0	0	0
3						A		A					2	2	1	1	1	0	0	0	0	0	0
8		A			A		A			A			4	3	2	2	1	1	0	0	0	0	0
7	A			A		A			A				4	3	3	2	2	1	1	0	0	0	0
2			A	A	A								3	3	3	3	3	3	2	1	0	0	0
10			A		A							A	3	3	3	3	3	3	2	2	1	0	0
9			A	A							A		3	3	3	3	3	3	3	2	1	1	0
													Borrar Columnas 1 y 2	Borrar Columnas 8 y 10	Borrar Columna 9	Borrar Columna 7	Borrar Columna 6	Borrar Columna 5	Borrar Columna 4	Borrar Columna 3	Borrar Columna 12	Borrar Columna 11	

La Tabla 2.17 es la matriz de funcionalidad rearreglada. En las filas nos muestra las ecuaciones en orden inverso a como fueron eliminadas por el primer algoritmo y en las columnas nos muestra las variables en el mismo orden en el que fueron eliminadas por el segundo algoritmo. Dicha matriz nos proporciona información sobre la estrategia de solución más sencilla para el modelo en estudio.

Tabla 2.17. Matriz de Funcionalidad Rearreglada.

		x_1	x_2	t	γ_2	γ_1	p_2^0	p_1^0	p_2	p_1	P	y_2	y_1
		1	**2**	**8**	**10**	**9**	**7**	**6**	**5**	**4**	**3**	**12**	**11**
Subgrupo 1	**1**	A	A										
Subgrupo 2	**6**	A	A	A	A								
	5	A	A	A		A							
	4			A			A						
	3			A				A					
	8		A		A		A		A				
	7	A				A		A		A			
	2								A	A	A		
	10								A		A	A	
	9									A	A		A

La Tabla 2.18 es la estrategia de solución obtenida a partir de la matriz de funcionalidad rearreglada.

Tabla 2.18. Estrategia de Solución Equilibrio Líquido – Vapor.

Paso	Acción	Variable	Ecuación
1	Seleccionar variable de diseño	(x_1 ó x_2)	
2	Calcular	x_2	1
3	Seleccionar variable de diseño	(t ó γ_2)	
4	Calcular	γ_2	6
5	Calcular	γ_1	5
6	Calcular	P_2^0	4
7	Calcular	P_1^0	3
8	Calcular	P_2	8
9	Calcular	P_1	7
10	Calcular	P	2
11	Calcular	y_2	10
12	Calcular	y_1	9

2.7 Software desarrollado en el presente trabajo.

El programa de cómputo a desarrollar deberá tener los siguientes atributos:

a) Dará respuestas correctas. Esta característica es obvia, sin embargo, hay un gran número de software comercial cuyos métodos, lógica y limitaciones son inciertos. Por lo tanto, para que el software a desarrollar sea un buen programa deberá imprimir mensajes de precaución cuando los datos de entrada excedan los límites dentro de los cuales el programa es aplicable y terminará de correr cuando los errores sean detectados.

b) Será fácil de usar. La entrada y edición de datos se hará a través de menús usando el formato de una hoja de cálculo universal. Los resultados generados se presentarán en forma gráfica y/o tabular.

c) Estará bien documentado. Los tres tipos principales de documentación que se elaborarán son:

- Estarán bien especificados y soportados teóricamente los métodos usados por el programa.
- Se elaborará un manual del usuario del programa, que describirá las instrucciones que el usuario deberá seguir para utilizar el software en su computadora, preparar las entradas e interpretar los resultados. Asimismo, este manual contendrá las capacidades del programa, las instrucciones de entrada, el listado de errores, la lógica del programa, las modificaciones, los casos de estudio y la nomenclatura.
- El listado del programa fuente será el elemento primario de la documentación del mismo, para permitir al usuario examinar en detalle los contenidos del programa. Este software se desarrollará en el lenguaje Fortran y C++ siguiendo las buenas prácticas de programación estructurada y la programación orientada a objetos.

d) Soporte técnico. Este atributo es esencial para que el software desarrollado sea usado continuamente por los usuarios. Esto es particularmente importante cuando se anticipan cambios en los métodos de cómputo para mejorar el programa.

Todas las actividades estarán apoyadas en una revisión continua de la literatura.

2.8 Programas.

Los tres programas propuestos para este trabaja (DFS, Ollero – Amselem y OE – AV) fueron desarrollados en FORTRAN, pero para proporcionar al usuario una interfase gráfica que facilitara la introducción de datos se utilizo el programa Visual C++.

La estructura de los programas se presenta en la Figura 2.20:

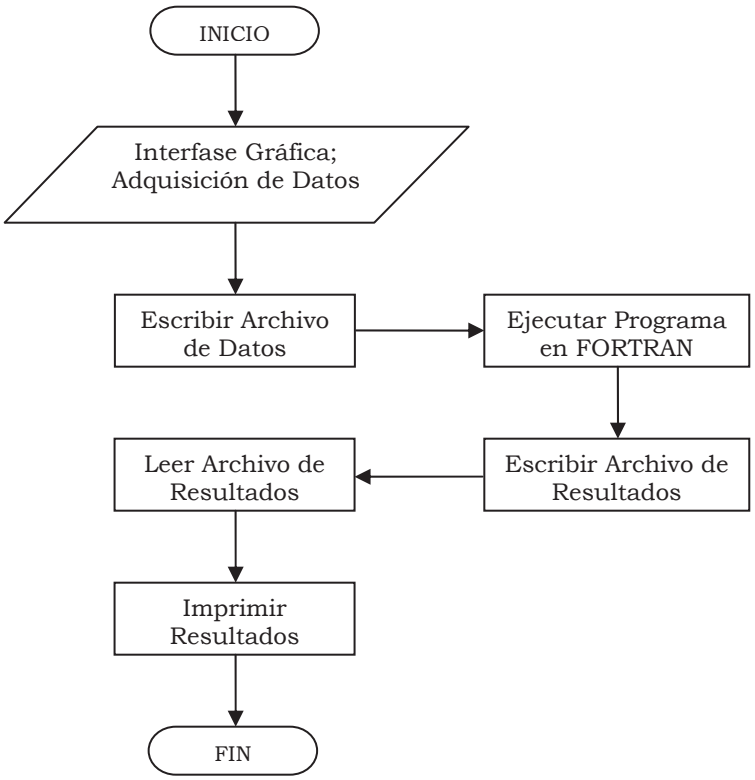


Figura 2.20. Estructura de los Programas.

CAPITULO 3: CASOS DE ESTUDIO

3.1 Casos de Estudio Algoritmos DFS y Ollero – Amselem

Los casos de estudio seleccionados son los ejemplos más representativos encontrados en la literatura, el primero de ellos es el segundo ejemplo presentado por Christensen y Rudd (1969) Figura 3.1, el segundo corresponde al grafo de Sargent y Westerberg (1964) Figura 3.2, el tercero es el grafo correspondiente a una planta de ácido sulfúrico y fue presentado por Shannon y Jonson (1966) Figura 3.3; el cuarto es una planta de sulfonal y fue propuesta por Ki-Won Cho, (1999) Figura 3.4; y finalmente se tiene una planta de tratamiento de agua dura presentado por Gundersen y Hertzberg (1982) Figura 3.6.

Ya que la matriz de adyacencia de corrientes del grafo original es equivalente a la matriz de adyacencia de nodos del grafo dual, trabajaremos con la primera para evitar la tarea de crear el grafo dual.

Caso 1: Segundo Grafo de Christensen y Rudd

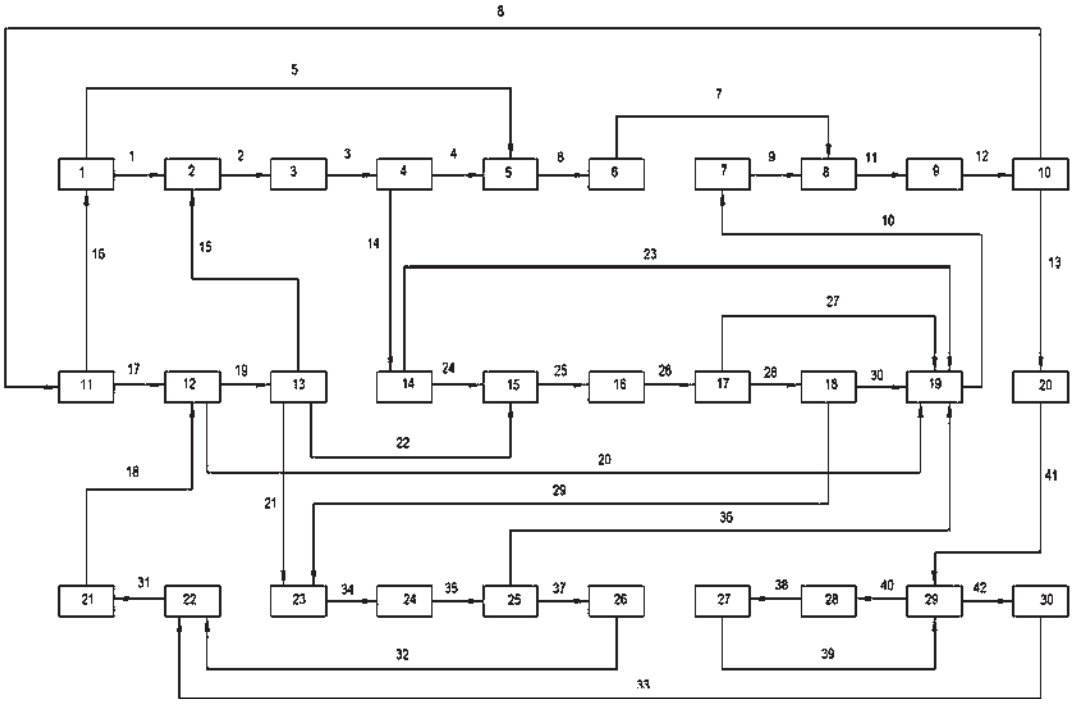


Figura 3.1. Grafo de Christensen y Rudd (1969).

El primer paso es obtener los datos para la implementación del software, la Tabla 3.1 contiene los datos para el programa DFS y la Tabla 3.2 la matriz de adyacencia con los datos para el programa de rasgado.

Datos DFS.

NVER = 30

NEDGE = 42

Tabla 3.1. Datos DFS Para Grafo C y R.

Corriente	RI	CI
1	1	2
2	2	3
3	3	4
4	4	5
5	1	5
6	5	6
7	6	8
8	10	11
9	7	8
10	19	7
11	8	9
12	9	10
13	10	20
14	4	14
15	13	2
16	11	1
17	11	12
18	21	12
19	12	13
20	12	19
21	13	23

Corriente	RI	CI
22	13	15
23	14	19
24	14	15
25	15	16
26	16	17
27	17	19
28	17	18
29	18	23
30	28	29
31	22	21
32	26	22
33	30	22
34	23	24
35	24	25
36	25	19
37	25	26
38	28	27
39	27	29
40	29	28
41	20	29
42	29	30

El programa nos indica que sólo existe un componente fuerte, es decir, todo el sistema debe de resolverse simultáneamente. En casos como éste, el programa no proporciona información sobre el orden de resolución del problema.

Datos Ollero – Amselem

Numero de corrientes = 42

Elementos no nulos de la matriz = 56

Tabla 3.2. Matriz de Adyacencia de Corrientes Grafo de C y R.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42
1		1																																								
2			1																																							
3				1										1																												
4						1																																				
5						1																																				
6							1																																			
7											1																															
8																1	1																										
9											1																															
10									1																																	
11												1																														
12								1					1																														
13																																									1	
14																							1	1																			
15		1																																								
16	1				1																																				
17																			1	1																						
18																			1	1																						
19															1						1	1																				
20										1																																
21																																		1								
22																									1																	
23										1																																
24																									1																	
25																										1																
26																											1	1														
27										1																																
28																													1	1												
29																																		1								
30										1																																
31																			1																							
32																															1											
33																															1											
34																																			1							
35																																				1	1					
36										1																																
37																																1										
38																																							1			
39																																								1		1
40																																						1				
41																																								1		1
42																																	1									

La Tabla 3.3 nos muestra un comparativo de los resultados obtenidos mediante la aplicación de diversos algoritmos:

Tabla 3.3. Comparación de Resultados.

Algoritmo	# Corrientes	Corrientes
Barkley y Motard	3	11 – 34 – 40
Zhou Li	3	11 – 19 – 40
BTA[6]	3	12 – 35 – 40
SJR[7]	3	11 – 34 – 40
Presente Estudio	3	12 – 35 – 40

[6] Algoritmo de Rasgado Básico.

[7] Algoritmo de búsqueda, prueba y ajuste.

Caso 2: Grafo de Sargent y Westerberg

La Figura 3.2 es el grafo de Sargent y Westerberg. Las Tablas 3.4 y 3.5 contienen los datos necesarios para ejecutar los programas de particionado y rasgado.

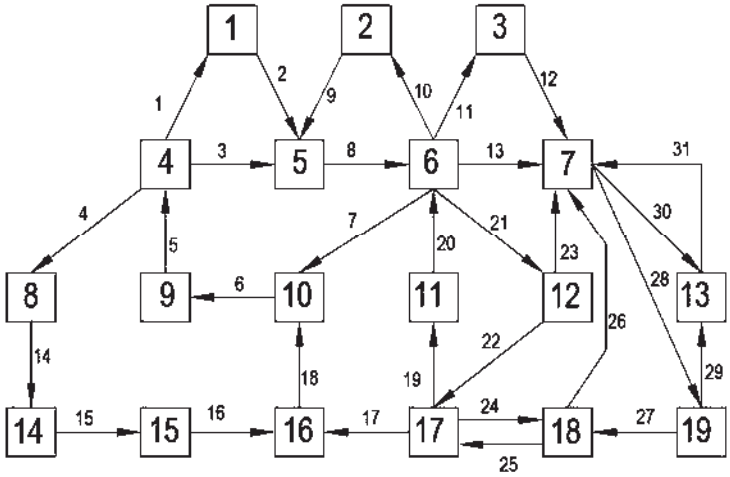


Figura 3.2. Grafo de S y W.

Datos DFS.

NVER = 19

NEDGE = 31

Tabla 3.4. Datos DFS Para Problema S y W.

Corriente	RI	CI
1	4	1
2	1	5
3	4	5
4	4	8
5	9	4
6	10	9
7	6	10
8	5	6
9	2	5
10	6	2
11	6	3
12	3	7
13	6	7
14	8	14
15	14	15
16	15	16

Corriente	RI	CI
17	17	16
18	16	10
19	17	11
20	11	6
21	6	12
22	12	17
23	12	7
24	17	18
25	18	17
26	18	7
27	19	18
28	7	19
29	19	13
30	7	13
31	13	7

Existe sólo un componente fuerte, el sistema se resuelve simultáneamente.

Datos Ollero – Amselem

Número de corrientes = 31

Elementos no nulos de la matriz = 54

Tabla 3.5. Matriz de Adyacencia del Grafo de S y W.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
1		1																													
2								1																							
3								1																							
4														1																	
5	1		1	1																											
6					1																										
7						1																									
8							1			1	1		1								1										
9								1																							
10									1																						
11												1																			
12																												1		1	
13																												1		1	
14															1																
15																1															
16																		1													
17																		1													
18						1																									
19																				1											
20							1			1	1		1									1										
21																						1	1								
22																	1		1					1							
23																												1		1	
24																									1	1					
25																	1		1					1							
26																												1		1	
27																									1	1					
28																											1		1		
29																															1
30																															1
31																												1		1	

La Tabla 3.6 nos muestra un comparativo de los resultados obtenidos mediante la aplicación de diversos algoritmos:

Tabla 3.6. Comparación de Resultados.

Algoritmo	# Corrientes	Corrientes
Barkley y Motard	6	6 – 8 – 21 – 24 -28 – 30
Zhou Li	6	6 – 8 – 19 – 25 – 28 -30
BTA	6	8 – 18 – 22 – 25 – 27 – 31
SJR	6	7 – 10 – 18 – 25 – 28 – 30
Presente Estudio	6	6 – 8 – 20 – 25 – 28 – 31

Caso 3: Planta de Acido Sulfúrico

La Figura 3.3 nos muestra el grafo de una planta de ácido sulfúrico; las Tablas 3.7 y 3.8 nos muestran los datos de los programas.

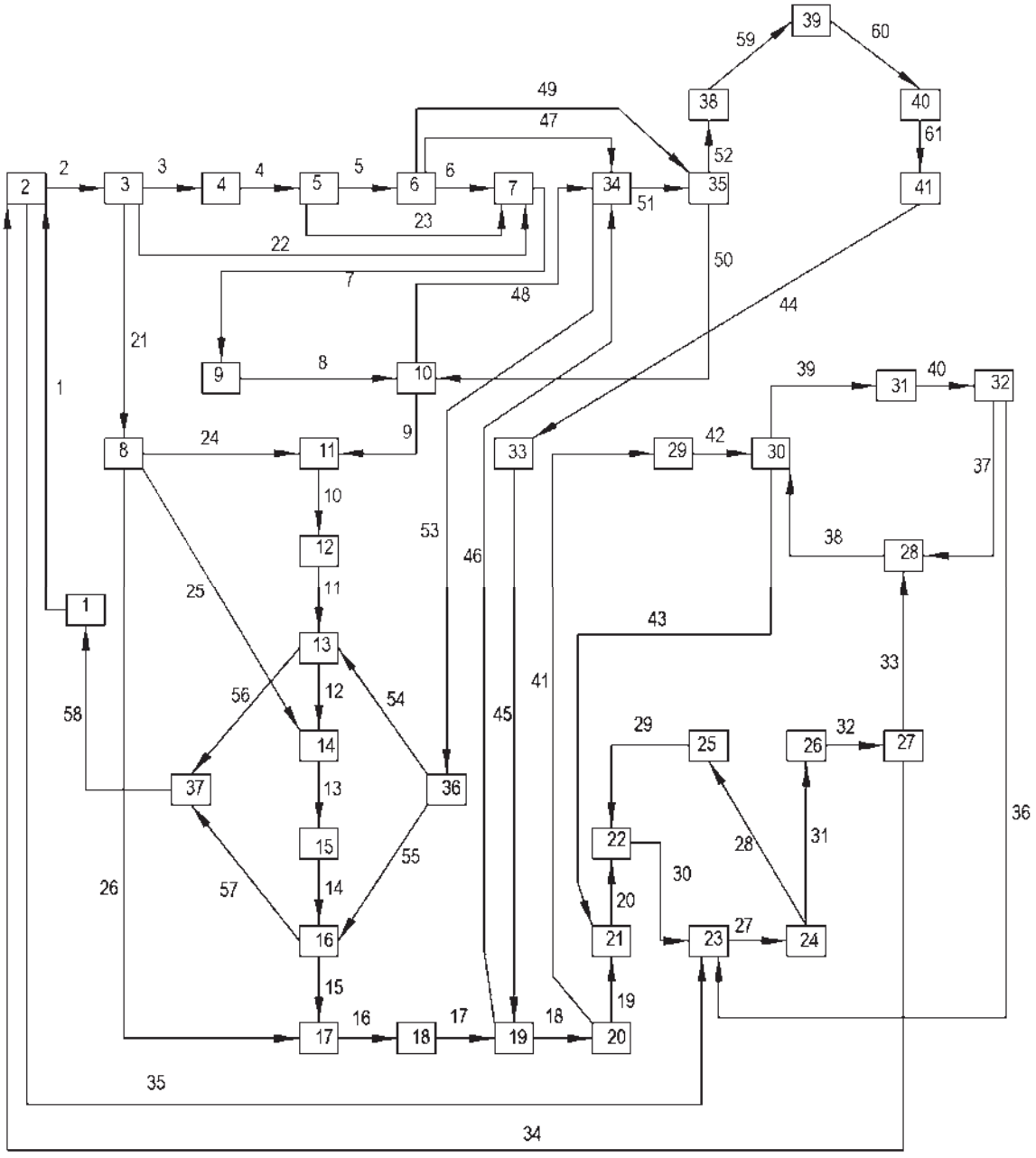


Figura 3.3. Planta de Ácido Sulfúrico.

Datos DFS.

NVER = 41

NEDGE = 61

Tabla 3.7. Datos DFS Para Grafo Planta Ácido Sulfúrico.

Corriente	RI	CI
1	1	2
2	2	3
3	3	4
4	4	5
5	5	6
6	6	7
7	7	9
8	9	10
9	10	11
10	11	12
11	12	13
12	13	14
13	14	15
14	15	16
15	16	17
16	17	18
17	18	19
18	19	20
19	20	21
20	21	22
21	3	8
22	3	7
23	5	7
24	8	11
25	8	14
26	8	17
27	23	24
28	24	25
29	25	22
30	22	23
31	24	26
32	26	27

Corriente	RI	CI
33	27	28
34	27	2
35	2	23
36	32	23
37	32	28
38	28	30
39	30	31
40	31	32
41	20	29
42	29	30
43	30	21
44	41	33
45	33	19
46	19	34
47	6	34
48	10	34
49	35	6
50	35	10
51	34	35
52	35	38
53	34	36
54	36	13
55	36	16
56	13	37
57	16	37
58	37	1
59	38	39
60	39	40
61	40	41

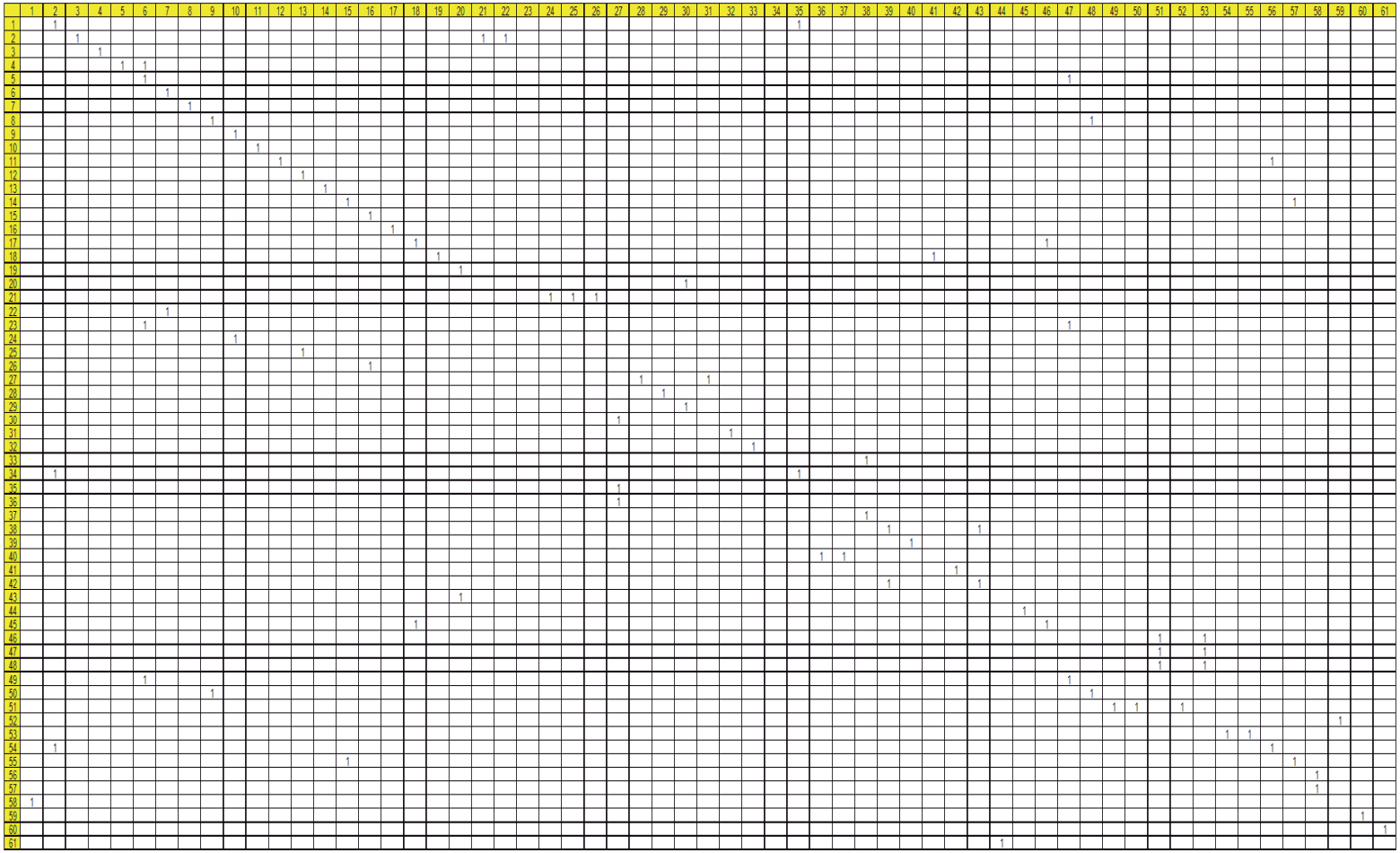
El sistema debe de resolverse simultáneamente, ya que sólo existe un componente fuerte.

Datos Ollero – Amselem

Número de corrientes = 61

Elementos no nulos de la matriz = 90

Tabla 3.8. Matriz de Adyacencia Planta de Ácido Sulfúrico.



La Tabla 3.9 nos muestra un comparativo de los resultados obtenidos mediante la aplicación de diversos algoritmos:

Tabla 3.9. Comparación de Resultados.

Algoritmo	# Corrientes	Corrientes
Barkley y Motard	5	27 – 39 – 51 – 53 – 58
Zhou Li	5	1 – 27 – 38 – 51 – 53
BTA	9	15 – 21 – 28 – 33 – 35 -37 – 51 – 56 – 57
SJR	5	11 – 27 – 38 – 51 – 53
Presente Estudio	5	27 – 40 – 51 – 53 – 58

Caso 4: Planta de Extracción de Sulfonal

La Figura 3.4 nos muestra la Planta de Extracción de Sulfonal presentada por Ki-Won y col. (1999). La Figura 3.5 es el dígrafo correspondiente. Las Tablas 3.10 y 3.11 nos muestran los datos para los programas.

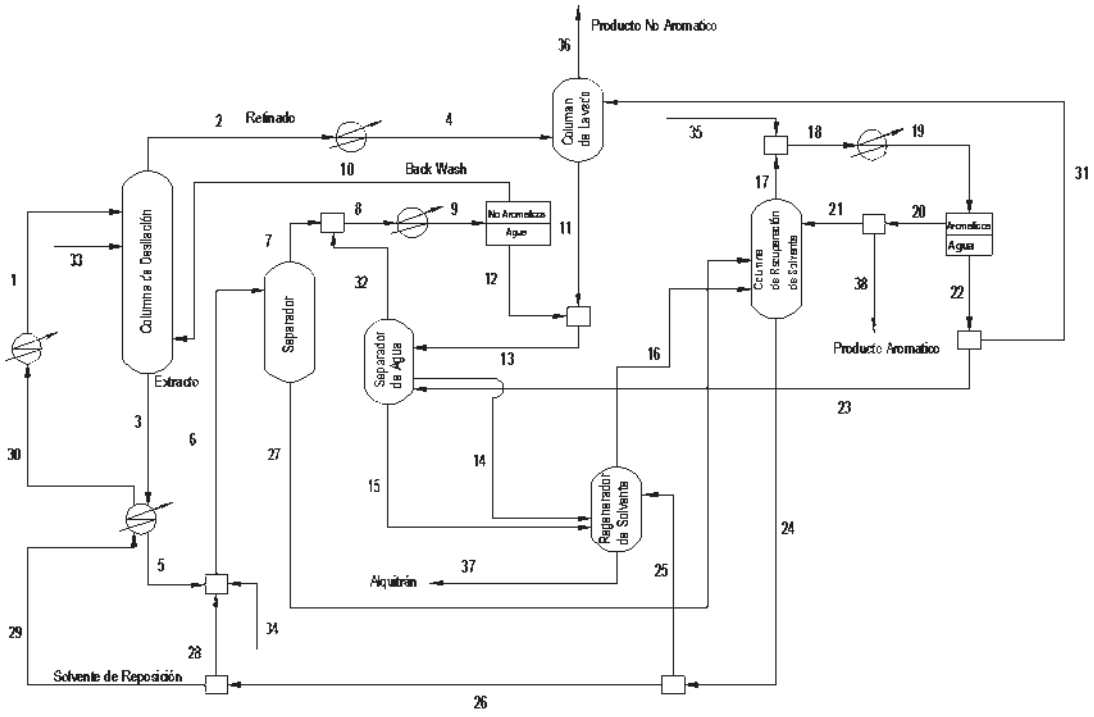


Figura 3.4 Planta de Extracción de Sulfonal.

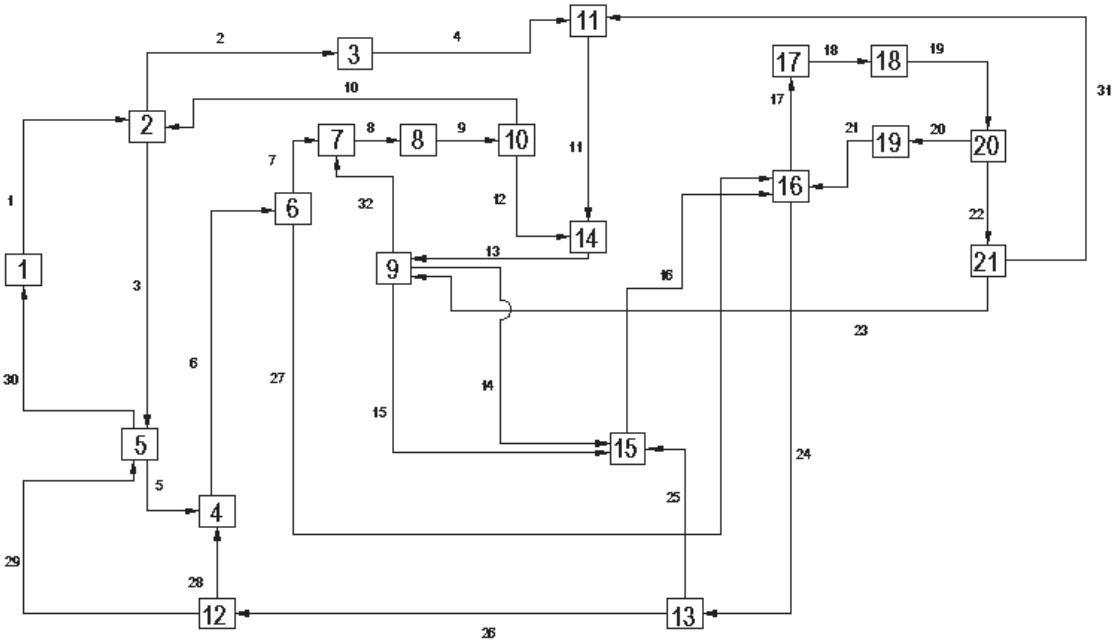


Figura 3.5 Dígrafo Asociado a la Planta de Extracción de Sulfonal.

Datos DFS.

NVER = 21

NEDGE = 32

Tabla 3.10. Datos DFS Para Grafo Planta Extracción de Sulfonal.

Corriente	RI	CI
1	1	2
2	2	3
3	3	4
4	4	5
5	5	6
6	6	7
7	7	9
8	9	10
9	10	11
10	11	12
11	12	13
12	13	14
13	14	15
14	15	16
15	16	17
16	17	18

Corriente	RI	CI
17	18	19
18	19	20
19	20	21
20	21	22
21	3	8
22	3	7
23	5	7
24	8	11
25	8	14
26	8	17
27	23	24
28	24	25
29	25	22
30	22	23
31	24	26
32	26	27

Existe sólo un componente fuerte, por lo que el sistema se resuelve simultáneamente. Este resultado coincide con el que fue reportado por Ki-Won y col. (1999).

Datos Ollero – Amselem

Número de corrientes = 32

Elementos no nulos de la matriz = 49

Tabla 3.11. Matriz de Adyacencia Planta de Extracción de Sulfonal.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
1		1	1																													
2				1																												
3					1																									1		
4											1																					
5						1																										
6							1																				1					
7								1																								
8									1																							
9										1		1																				
10		1	1																													
11													1																			
12													1																			
13														1	1																	1
14																1																
15																1																
16																	1							1								
17																		1														
18																			1													
19																				1		1										
20																					1											
21																	1							1								
22																							1								1	
23														1	1																	1
24																									1	1						
25																1																
26																												1	1			
27																	1							1								
28						1																										
29					1																									1		
30	1																															
31											1																					
32								1																								

El programa nos indica que el mínimo número de corrientes de corte es 4 y que éstas son:

19 – 24 – 30 – 9

Ki-Won y col., (1999) también encuentran 4 corrientes de corte y sugiere que éstas sean:

3 – 13 – 19 – 24

Caso 5: Planta de Tratamiento de Agua Dura

La Figura 3.5 nos muestra el grafo de una planta de tratamiento de agua dura. La Tabla 3.13 nos muestra los datos para el programa de particionado. No se presenta la matriz de adyacencia para este grafo debido a que es una matriz extremadamente grande (163 x 163).

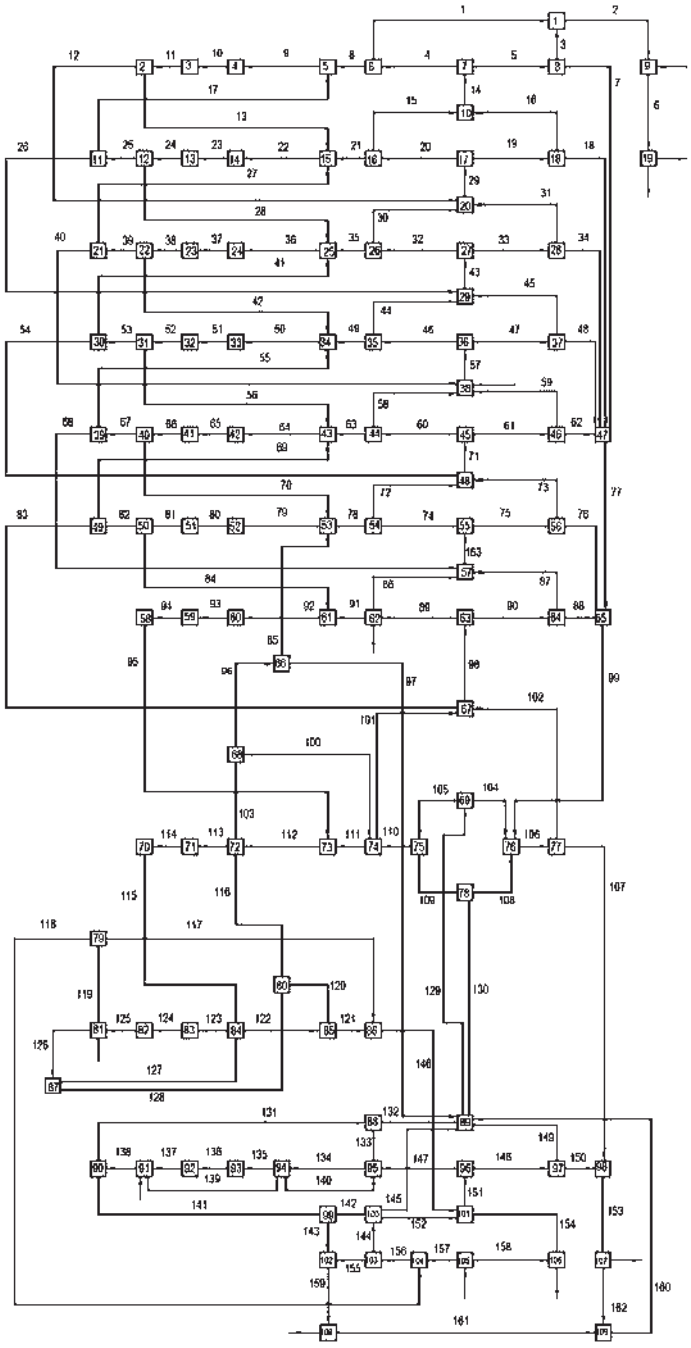


Figura 3.6. Planta de Tratamiento de Agua Dura.

Datos DFS.

NVER = 109

NEDGE = 163

Tabla 3.12. Datos DFS Para Planta de Tratamiento de Agua Dura.

Corriente	RI	CI
1	6	1
2	1	9
3	8	1
4	7	6
5	7	8
6	9	19
7	8	47
8	6	5
9	5	4
10	4	3
11	3	2
12	2	20
13	2	15
14	10	7
15	16	10
16	18	10
17	11	5
18	18	47
19	17	18
20	17	16
21	16	15
22	15	14
23	14	13
24	13	12
25	12	11
26	11	29
27	21	15
28	12	25
29	20	17
30	26	20
31	28	20
32	27	26
33	27	28
34	28	47
35	26	25
36	25	24
37	24	23
38	23	22

Corriente	RI	CI
39	22	21
40	21	38
41	30	25
42	22	34
43	29	27
44	35	29
45	37	29
46	36	35
47	36	37
48	37	47
49	35	34
50	34	33
51	33	32
52	32	31
53	31	30
54	30	48
55	39	34
56	31	43
57	38	36
58	44	38
59	46	38
60	45	44
61	45	46
62	46	47
63	44	43
64	43	42
65	42	41
66	41	40
67	40	39
68	39	57
69	49	43
70	40	53
71	48	45
72	54	48
73	56	48
74	55	54
75	55	56
76	56	65

Corriente	RI	CI
77	47	65
78	54	53
79	53	52
80	52	51
81	51	50
82	50	49
83	49	67
84	50	61
85	66	53
86	62	57
87	64	57
88	64	65
89	63	62
90	63	64
91	62	61
92	61	60
93	60	59
94	59	58
95	58	73
96	68	66
97	66	89
98	67	63
99	65	76
100	68	74
101	74	67
102	77	67
103	72	68
104	69	76
105	69	75
106	76	77
107	77	98
108	78	76
109	78	75
110	75	74
111	74	73
112	73	72
113	72	71
114	71	70

Corriente	RI	CI
115	70	84
116	80	72
117	79	86
118	79	104
119	81	79
120	85	80
121	85	86
122	84	85
123	84	83
124	83	82
125	82	81
126	81	87
127	87	84
128	87	80
129	89	69
130	89	78
131	90	88
132	88	89
133	95	88
134	95	94
135	94	93
136	93	92
137	92	91
138	91	90
139	91	94

Corriente	RI	CI
140	94	95
141	90	99
142	99	100
143	99	102
144	103	100
145	100	89
146	86	101
147	96	95
148	96	97
149	97	89
150	97	98
151	101	96
152	100	101
153	98	107
154	106	101
155	103	102
156	104	103
157	105	104
158	105	106
159	102	108
160	109	89
161	108	109
162	107	109
163	57	55

El programa nos indica que existen seis componentes fuertes y, por lo tanto, deben de resolverse independientemente, en el siguiente orden:

$105 \rightarrow 106 \rightarrow A \rightarrow 1 \rightarrow 9 \rightarrow 19$

Donde el Subgrupo A está formado por los siguientes nodos:

6	5	4	3	2	20
17	18	10	7	8	47
65	76	77	67	63	62
57	55	54	48	45	44
38	36	35	29	27	26
25	24	23	22	21	15
14	13	12	11	34	33
32	31	30	43	42	41

40	39	53	52	51	50
49	61	60	59	58	73
72	68	66	89	69	75
74	78	71	70	84	85
80	86	101	96	95	88
94	93	92	91	90	99
100	102	108	109	97	98
107	83	82	81	79	104
103	87	28	37	46	56
64	16				

Datos Ollero – Amselem

Número de corrientes = 163

Elementos no nulos de la matriz = 221

El programa nos indica que el conjunto de corrientes de corte es el siguiente:

104 – 112 – 125 – 137 – 134 – 24 – 29 – 52 – 81 – 106 – 43 – 163

Zhou Li (1987) reportó que para este problema existen 12 corrientes de corte; sin embargo, no es posible hacer una comparación con éstas, ya que no presenta el grafo para el cual son válidos sus resultados (recordemos que la numeración del grafo es arbitraria y, por lo tanto, los números de las corrientes pueden no coincidir).

3.2 Casos de Estudio OE – AV

Caso 1

El siguiente problema fue presentado por Tarifa y col. (2004):

Ecuación	Número
$f_1(x_1,x_2)=0$	1
$f_2(x_4)=0$	2
$f_3(x_3,x_6)=0$	3
$f_4(x_4,x_5)=0$	4
$f_5(x_1,x_6)=0$	5
$f_6(x_2,x_3,x_5)=0$	6
$f_7(x_2,x_7)=0$	7
$f_8(x_8,x_9,x_{10})=0$	8
$f_8(x_8,x_9,x_{10})=0$	9

Podemos observar que el sistema tiene 1 grado de libertad, ya que tenemos 10 variables y únicamente 9 ecuaciones. Tarifa y col. (2004) proponen fijar una de las variables externamente (x_{10}) para cumplir con los grados de libertad del sistema. Es obvio que nosotros también tendremos que fijar una de las variables, pero no lo haremos de manera arbitraria. En lugar de esa opción lo haremos en base a la estructura del modelo, haciendo la aclaración que la asignación de los grados de libertad antes de iniciar la solución de un modelo matemático es posible y, en muchos casos, recomendable cuando conocemos los valores aproximados de la variable, o bien cuando es una variable que podemos medir o no podemos cambiar (ej. T, P, A).

Datos OE – AV
Número de Ecuaciones = 9
Número de Variables = 10
Elementos No Nulos de la Matriz de Funcionalidad = 20

La matriz de funcionalidad para este problema se presenta en la Tabla 3.14.

Tabla 3.14. Matriz de Funcionalidad Problema de Tarifa y col.

	X_1	X_2	X_3	X_4	X_5	X_6	X_7	X_8	X_9	X_{10}
	1	2	3	4	5	6	7	8	9	10
1	a	a								
2				a						
3			a			a				
4				a	a					
5	a					a				
6		a	a		a					
7		a					a			
8								a	a	a
9								a	a	a

El programa nos da como resultado la matriz de funcionalidad rearreglada, que se presenta en la Tabla 3.15. El programa también nos indica que existen dos subgrupos.

SUB GRUPO 1 = FILA 9
SUB GRUPO 2 = FILA 3

Tabla 3.15. Matriz de Funcionalidad Rearreglada Problema de Tarifa y col.

	X_8	X_9	X_{10}	X_4	X_5	X_3	X_6	X_2	X_1	X_7
	8	9	10	4	5	3	6	2	1	7
9	a	a	a							
8	a	a	a							
2				a						
4				a	a					
3						a	a			
6					a	a		a		
5							a		a	
1								a	a	
7								a		a

La estrategia de solución se presenta en la Tabla 3.16:

Tabla 3.16. Estrategia de Solución Problema de Tarifa y col.

Paso	Acción		Variable	Ecuación
1	Seleccionar variable de diseño		X_8 , X_9 ó X_{10}	
2	¿Se pueden resolver simultáneamente las ecuaciones 8 y 9?	Si	X_8 y X_9	8 y 9
		No. Suponer una de las variables y resolver la otra con una de las ecuaciones	X_8 ó X_9	8 ó 9
3	Calcular		X_4	2
4	Calcular		X_5	4
5	Suponer		X_3 ó X_6	
6	Calcular		X_3	3
7	Calcular		X_2	6
8	Calcular		X_1	1
9	Calcular		X_7	7
10	Recalcular		X_6	5

Tarifa y col. (2004) reportan la secuencia de solución que se presenta en la Tabla 3.17:

Tabla 3.17. Estrategia de Solución Propuesta por Tarifa y col.

Paso	Acción	Variable	Ecuación
1	Calcular	X_4	2
2	Calcular	X_5	4
3	Suponer	X_1	
4	Calcular	X_6	5
5	Calcular	X_3	3
6	Calcular	X_2	6
7	Suponer	X_8	
8	Calcular	X_9	9

Como se puede observar, la estrategia obtenida a partir del programa OE – AV es más flexible que la presentada por Tarifa y col. (2004), ya que nos da la posibilidad de elegir entre las variables que pueden ser escogidas como variables de diseño y las que son elegidas como variables de corte (supuestas). Otra ventaja es la posibilidad de encontrar conjuntos de

ecuaciones que pueden resolverse simultáneamente, reduciendo así el número de corrientes de corte. Es preciso señalar que la distinción entre una variable de corte y una de diseño es la existencia de grados de libertad; en este problema había un sólo grado de libertad y, en consecuencia, tenemos únicamente una variable de diseño.

Caso 2

Considere el siguiente modelo matemático:

Ecuación	Número
$x_{1,3} + x_{2,3} = 1$	1
$x_{1,5} + x_{2,5} = 1$	2
$x_{3,6} + x_{4,6} = 1$	3
$x_{1,7} + x_{2,7} + x_{3,7} + x_{4,7} = 1$	4
$F_1 + F_2 = F_3$	5
$F_1 x_{3,1} + F_2 x_{1,2} = F_3 x_{1,3}$	6
$F_1 \overline{C}_{p1} T_1 + F_2 \overline{C}_{p2} T_2 = F_3 \overline{C}_{p3} T_3$	7
$S_1 = \tau_1 F_3 \big/ \tilde{\rho}_3$	8
$x_{1,3} = x_{1,5}$	9
$F_3 = F_5$	10
$F_4 = F_2$	11
$F_3 \overline{C}_{p3} T_3 = F_5 \overline{C}_{p5} T_5$	12
$F_4 \overline{C}_{p4} T_4 = F_2 \overline{C}_{p2} T_2$	13
$\Delta T_{lm} = \dfrac{(T_2 - T_3) - (T_4 - T_5)}{\ln \dfrac{(T_2 - T_3)}{(T_4 - T_5)}}$	14
$Q_2 = U_2 A_2 \Delta T_{lm}$	15
$F_5 + F_6 = F_7$	16
$F_5 x_{1,5} + F_6 x_{1,6} = F_7 x_{1,7}$	17
$F_5 x_{2,5} + F_6 x_{2,6} = F_7 x_{2,7}$	18
$F_5 x_{3,5} + F_6 x_{3,6} = F_7 x_{3,7}$	19
$F_5 \overline{C}_{p5} T_5 + F_6 \overline{C}_{p6} T_6 = F_7 \overline{C}_{p7} T_7$	20
$S_3 = \tau_3 F_5 \big/ \tilde{\rho}_5$	21

Suponga que $x_{1,1}$, $x_{2,2}$, $x_{2,4}$, T_5, $x_{3,6}$ son conocidas

Datos OE – AV

Número de Ecuaciones = 21

Número de Variables = 29

Elementos No Nulos de la Matriz de Funcionalidad = 69

La matriz de funcionalidad para el modelo matemático anterior se muestra en la Tabla 3.18:

Tabla 3.18. Matriz de Funcionalidad.

	T_1	F_1	T_2	F_2	T_3	F_3	$x_{1,3}$	$x_{2,3}$	T_4	F_4	F_5	$x_{1,5}$	$x_{2,5}$	T_6	F_6	$x_{4,6}$	T_7	F_7	$x_{1,7}$	$x_{2,7}$	$x_{3,7}$	$x_{4,7}$	S_1	τ_1	Q_2	ΔT_{lm}	A_2	S_3	τ_3
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29
1							a	a																					
2												a	a																
3																a													
4																			a	a	a	a							
5		a		a		a																							
6		a		a		a	a																						
7	a	a	a	a	a	a	a																						
8						a																	a	a					
9							a					a																	
10						a					a																		
11				a						a																			
12					a	a																							
13			a	a					a	a																			
14			a		a				a																	a			
15																									a	a	a		
16											a				a			a											
17											a	a			a			a	a										
18											a		a		a			a		a									
19											a				a			a			a								
20											a			a	a		a	a											
21																												a	a

El programa nos indica que existen 7 sub grupos, distribuidos de la siguiente forma:

SUB GROUPO 1 = FILA 21
SUB GROUPO 2 = FILA 15
SUB GROUPO 3 = FILA 11
SUB GROUPO 4 = FILA 13
SUB GROUPO 5 = FILA 8
SUB GROUPO 6 = FILA 16
SUB GROUPO 7 = FILA 20

Estos subgrupos se pueden observar en la matriz de funcionalidad rearreglada (Tabla 3.19)

Tabla 3.19. Matriz Rearreglada.

	$x_{4,6}$	S_3	τ_3	Q_2	ΔT_{lm}	A_2	F_2	F_4	T_2	T_4	T_3	F_3	F_5	F_1	$x_{1,3}$	$x_{1,5}$	$x_{2,5}$	$x_{2,3}$	T_1	S_1	τ_1	F_6	F_7	$x_{3,7}$	$x_{2,7}$	$x_{1,7}$	$x_{4,7}$	T_6	T_7
	16	28	29	25	26	27	4	10	3	9	5	6	11	2	7	12	13	8	1	23	24	15	18	21	20	19	22	14	17
3	a																												
21		a	a																										
15				a	a	a																							
11							a	a																					
13							a	a	a	a																			
14					a				a	a	a																		
12											a	a																	
10												a	a																
5							a					a		a															
6							a					a		a	a														
9															a	a													
2																a	a												
1															a			a											
7							a		a		a	a		a	a				a										
8												a								a	a								
16													a									a	a						
19													a									a	a	a					
18													a				a					a	a		a				
17													a			a						a	a			a			
4																								a	a	a	a		
20													a									a	a					a	a

Notamos que la ecuación 3 no pertenece a ningún subgrupo. Esto es debido a que sólo tiene una variable y puede ser resuelta desde un principio.

La estrategia de solución para este problema se resume en la Tabla 3.20.

Tabla 3.20. Estrategia de Solución.

Paso	Acción	Variable	Ecuación
1	Calcular	$x_{4,6}$	3
2	Seleccionar como variable de diseño	τ_3	
3	Calcular	S_3	21
4	Seleccionar como variable de diseño	Q_2	
5	Seleccionar como variable de diseño	A_2	
6	Calcular	ΔT_{lm}	15
7	Seleccionar como variable de diseño	F_4	
8	Calcular	F_2	11
9	Seleccionar como variable de diseño	T_4	
10	Calcular	T_2	13
11	Calcular	T_3	14
12	Calcular	F_3	12
13	Calcular	F_5	10
14	Calcular	F_1	5
15	Calcular	$x_{1,3}$	6
16	Calcular	$x_{1,5}$	9
17	Calcular	$x_{2,5}$	2
18	Calcular	$x_{2,3}$	1
19	Calcular	T_1	7
20	Seleccionar como variable de diseño	τ_1	
21	Calcular	S_1	8
22	Seleccionar como variable de diseño	F_6	
23	Calcular	F_7	16
24	Calcular	$x_{3,7}$	19
25	Calcular	$x_{2,7}$	18
26	Calcular	$x_{1,7}$	17
27	Calcular	$x_{4,7}$	4
28	Seleccionar como variable de diseño	T_6	
29	Calcular	T_7	20

CAPITULO 4: CONCLUSIONES

Las tareas de rasgado y particionado de un diagrama de flujo así como de un sistema de ecuaciones son tareas que se pueden realizar de manera sistemática aplicando varias técnicas bien definidas y reportadas en la literatura. Sin embargo, conforme aumenta el tamaño de un proceso y son mayores las interconexiones entre las unidades del mismo, también se incrementa la complejidad de los sistemas de estudio y, por consiguiente, la dificultad de análisis es mayor. Para estudiar adecuadamente este tipo de casos se hace necesario la automatización de tareas de descomposición y rasgado de sistemas, con el fin de hacerlas más rápidas y precisas. El presente trabajo representa un avance en esta dirección y nos muestra tres programas que fueron probados para diferentes casos de estudio.

El algoritmo DFS demostró su utilidad para sistemas pequeños o de gran dimensión, presentando tiempos de ejecución muy cortos y proporcionando información valiosa sobre la estructura de los diagramas de flujo y la relación entre sus componentes; la razón por la que los primeros tres diagramas analizados presentan sólo un componente fuerte es que originalmente estos problemas fueron seleccionados para probar diversas técnicas de rasgado. Por lo tanto, no es de extrañar el hecho de que tengan que resolverse como un todo.

Es sabido que mientras menor sea el número de corrientes de corte en un diagrama de flujo, más rápida será la convergencia del sistema (Delin Qu, 1991). Atendiendo a este criterio se deduce que el algoritmo de Ollero – Ameselem da muy buenos resultados encontrando para todos los casos de estudio el menor número de corrientes de corte. Otro aspecto importante a considerar es que el algoritmo funciona aún con los diagramas de flujo más complejos, caso en que muchos de los algoritmos existentes fallan.

El algoritmo de OE – AV es una herramienta valiosa para establecer sistemáticamente el orden de cálculo de las ecuaciones que conforman los modelos matemáticos. Un aspecto que es necesario atender, para lograr la máxima utilidad de este algoritmo, es la participación del Ingeniero en la apropiada selección de las variables que han de ser especificadas, ya sea como variables de diseño o como variables de corte.

CAPITULO 5: APENDICES

APENDICE A: Preparación de Datos.

1. Preparación de los datos para el uso de los programas DFS y Ollero – Amselem.

Numeramos de manera arbitraria tanto los nodos como las corrientes de un dígrafo, sin tomar en cuenta las corrientes de entrada o de salida del proceso (Figura A.1):

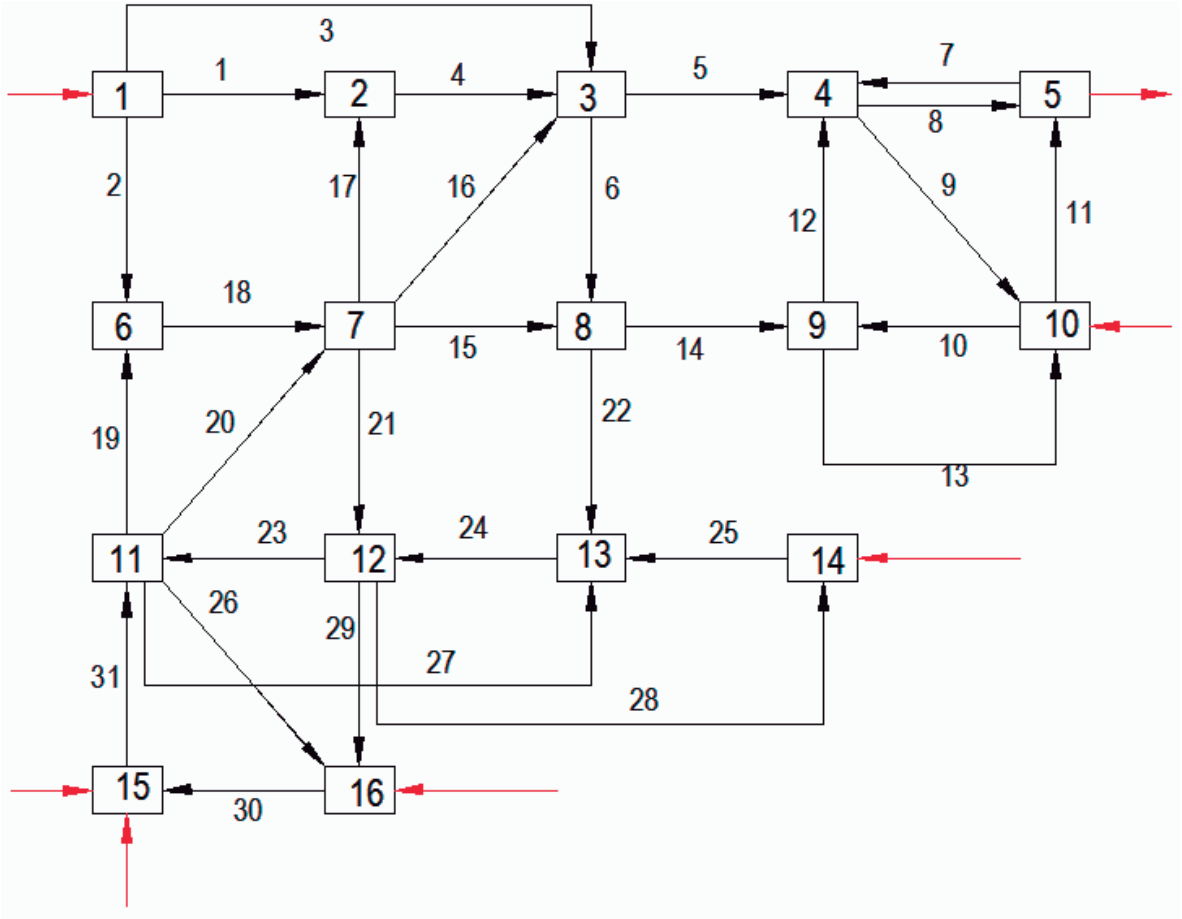


Figura A.1. Numeración del Dígrafo.

DFS

Los parámetros requeridos por el programa DFS son los siguientes:

- NVER = Número de nodos.
- NEDGE = Número de corrientes.

- RI se refiere a los nodos de los que sale una corriente en particular; se le conoce como cola de la corriente.
- CI es el nodo hacia el que se dirige una corriente en particular; se le conoce como cabeza de la corriente.

Los parámetros RI y CI deben de obtenerse para cada una de las corrientes, por ejemplo para la corriente 1:

RI = 1; CI = 2.

Para el dígrafo del ejemplo los datos son:

NVER = 16

NEDGE =31

Tabla A.1. RI y CI para las corrientes de la Figura A.1.

Corriente	RI	CI
1	1	2
2	1	6
3	1	3
4	2	3
5	3	4
6	3	8
7	5	4
8	4	5
9	4	10
10	10	9
11	10	5
12	9	4
13	9	10
14	8	9
15	7	8
16	7	3

Corriente	RI	CI
17	7	2
18	6	7
19	11	6
20	11	7
21	7	12
22	8	13
23	12	11
24	13	12
25	14	13
26	11	16
27	11	13
28	12	14
29	12	16
30	16	15
31	15	11

Ollero – Amselem

Los parámetros requeridos para este programa son:

- Número de Corrientes
- Elementos No Nulos de la Matriz de Adyacencia de Corrientes.

- Coordenadas de los Elementos No Nulos: se refiere al número de fila y de columna en que se encuentra el elemento no nulo.

La matriz de adyacencia se presenta en la Tabla A.2:
Número de Corrientes = 31
Elementos No Nulos de la Matriz = 58

Tabla A.2. Matriz de Adyacencia de Corrientes de la Figura A.1.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
1				1																											
2																		1													
3					1	1																									
4					1	1																									
5								1	1																						
6														1																	
7								1	1																						
8							1																								
9										1	1																				
10												1	1																		
11							1																								
12								1	1																						
13										1	1																				
14												1	1																		
15														1								1									
16					1	1																									
17				1																											
18															1	1	1				1										
19																		1													
20															1	1	1				1										
21																							1						1	1	
22																								1							
23																			1	1						1	1				
24																							1						1	1	
25																								1							
26																														1	
27																								1							
28																									1						
29																														1	
30																															1
31																			1	1						1	1				

2- Preparación de los datos para el programa de OE – AV.

El primer paso es numerar las ecuaciones y las variables que conforman el modelo matemático:

Ecuación	Número
$f_1(x_4, x_9) = 0$	[11]
$f_2(x_1, x_2, x_7, x_8) = 0$	[12]
$f_3(x_1, x_8) = 0$	[13]
$f_4(x_3, x_4, x_9) = 0$	[14]
$f_5(x_1, x_2, x_4, x_6, x_7) = 0$	[15]
$f_6(x_5, x_9) = 0$	[16]
$f_7(x_4, x_6, x_8) = 0$	[17]

Los parámetros requeridos para ejecutar el programa son:

- Número de Ecuaciones
- Número de Variables
- Número de Elementos No Nulos de la Matriz de Funcionalidad
- Coordenadas de los Elementos No Nulos de la Matriz, es decir, el número de fila y número de columna del elemento no nulo. Por ejemplo, el primer elemento no nulo de la matriz tiene coordenadas: (1,4).

La matriz de funcionalidad se presenta en la Tabla A.3.

Número de Ecuaciones = 7

Número de Variables = 9

Elementos No Nulos de la Matriz = 21

Tabla A.2. Matriz de Funcionalidad.

	1	2	3	4	5	6	7	8	9
1				a					a
2	a	a					a	a	
3	a							a	
4			a	a					a
5	a	a		a		a	a		
6					a				a
7				a		a		a	

APENDICE B: Uso del Software

El programa "Rasgado y Particionado" es la interfase a través de la cual podemos acceder a los programas DFS, Ollero-Amselem y OE – AV. La Figura B.1 nos muestra la ventana principal de este programa; para acceder a cada uno de los programas basta con hacer un clic sobre el nombre del programa deseado o bien presionando las teclas de acceso rápido (DFS = Alt + D; Ollero – Amselem = Alt + O; y OE – AV = Alt + E).

Figura B.1. Ventana Principal Programa Rasgado y Particionado.

DFS

La Figura B.2 nos muestra la ventana principal del programa DFS, en la cual debemos especificar los parámetros NVER y NEDGE. Al presionar el botón de "DATOS" o el comando de acceso rápido Alt + D aparecerá la pantalla de datos (Figura B.3), donde tenemos que introducir uno a uno los valores de RI y CI para cada una de las corrientes.

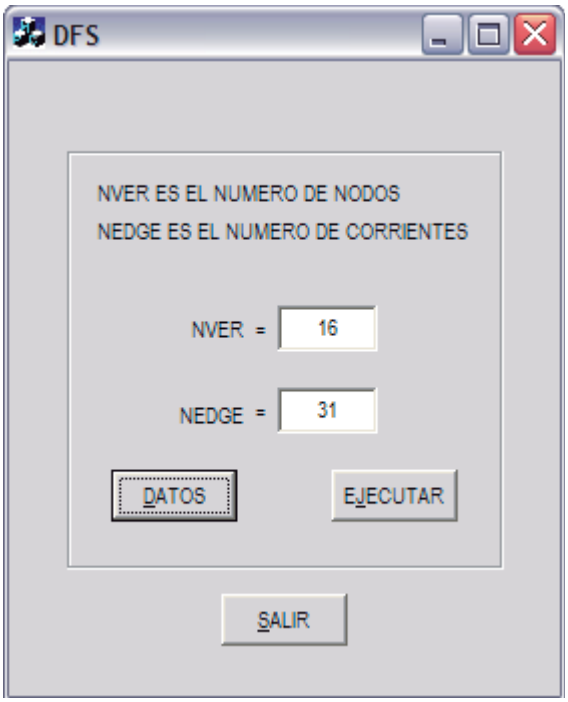


Figura B.2. Ventana Principal DFS.

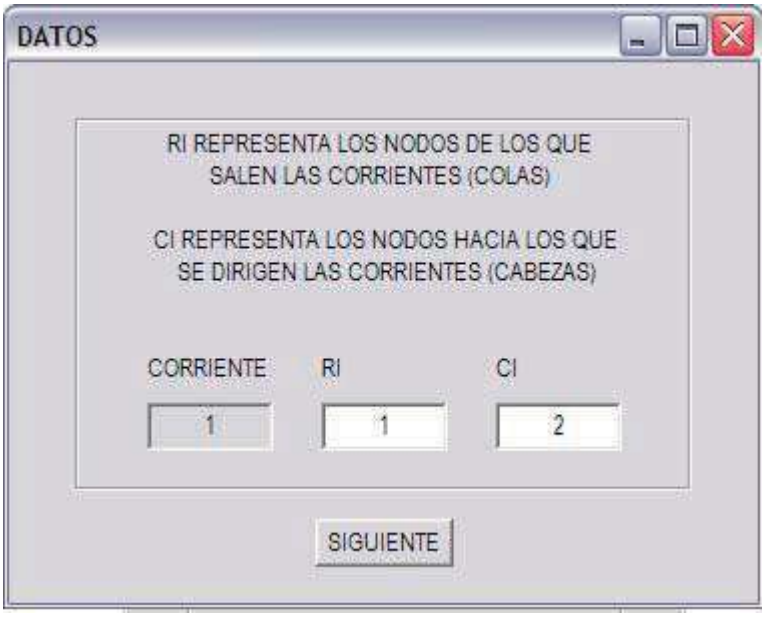


Figura B.3. Ventana de Datos DFS.

Al terminar de ingresar los valores de las colas y las cabezas, se tiene que regresar a la pantalla principal donde se presiona el botón "EJECUTAR" para ir a la pantalla de resultados.

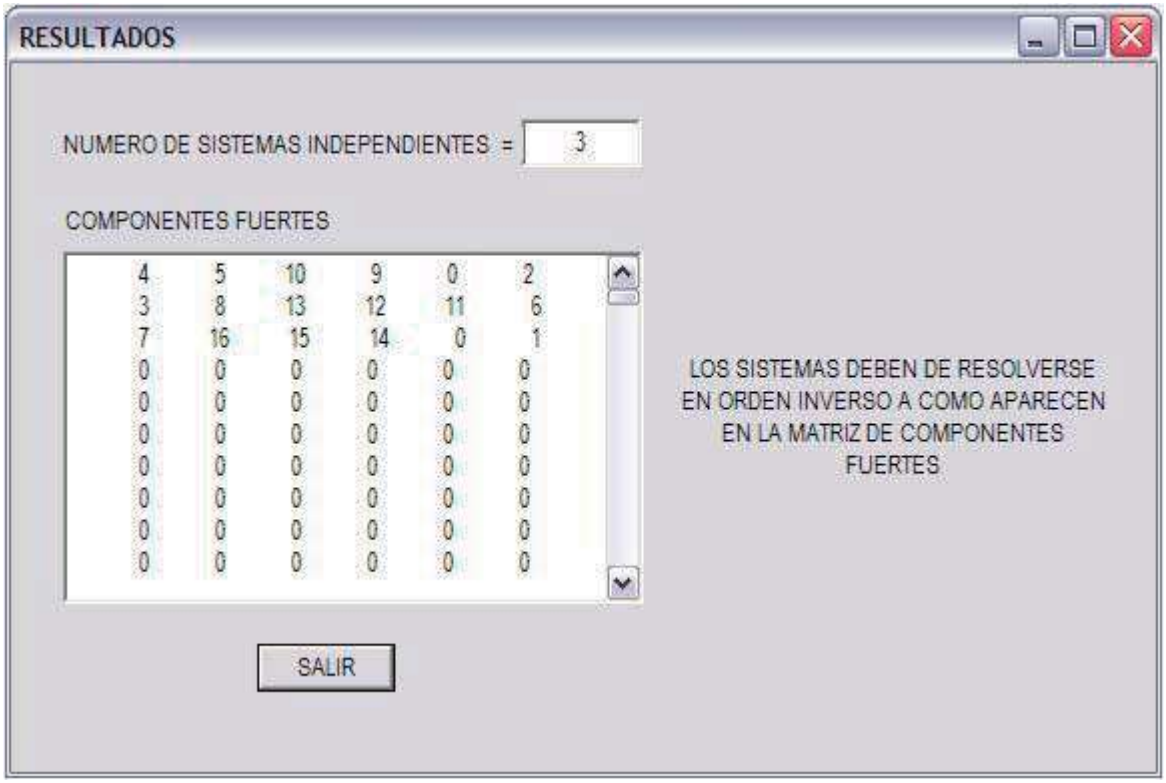


Figura B.4. Ventana de Resultados DFS.

La Figura B.4 nos muestra la ventana de resultados del programa DFS para el ejemplo presentado en la Figura A.1. Los valores mostrados en el campo de "COMPONENTES FUERTES" nos indican los subgrupos encontrados separados unos de otros por un "0" y la secuencia de resolución de los mismos en orden inverso, es decir, los subgrupos se tienen que resolver de abajo hacia arriba:

1 →2, 3, 8, 13, 12, 11, 6, 7, 16, 15, 14 → 4, 5, 10, 9

Ollero – Amselem

La Figura B.5 nos muestra la ventana principal del programa Ollero – Amselem. En esta pantalla se debe de especificar el número de corrientes y el número de elementos no nulos de la matriz de adyacencia de corrientes.

Una vez que hayamos especificado estos valores presionamos el botón de "MATRIZ".

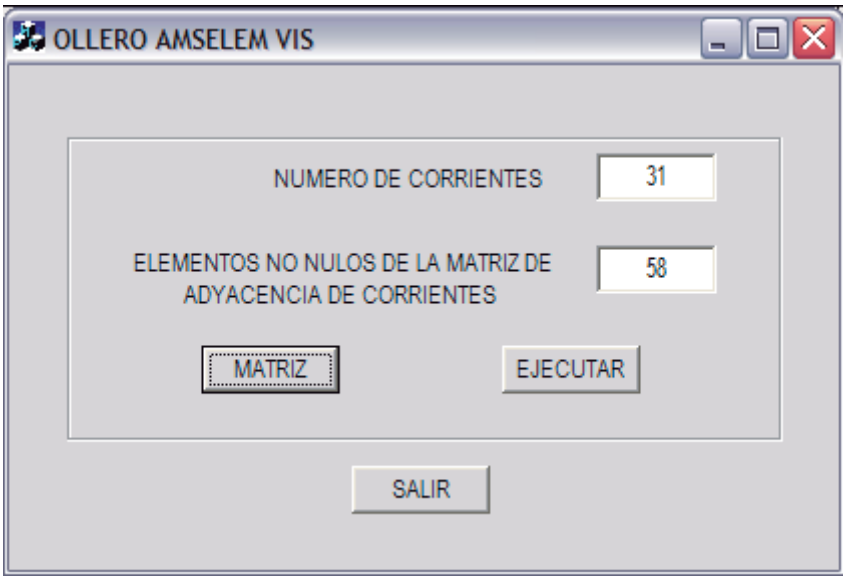


Figura B.5. Ventana Principal Ollero - Amselem.

Al presionar el botón "MATRIZ" se abrirá una nueva ventana (Figura B.6), en la cual tenemos que escribir una a una las coordenadas de los elementos no nulos de la matriz.

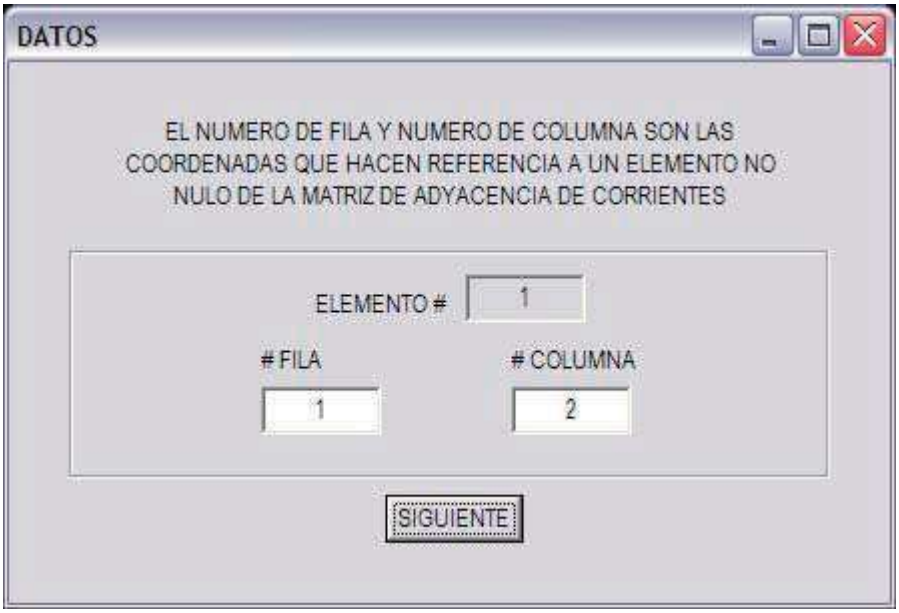


Figura B.6. Ventana de Datos Ollero – Amselem.

Al terminar de registrar las coordenadas regresaremos a la ventana principal y presionaremos el botón “EJECUTAR” para abrir la ventana de resultados.

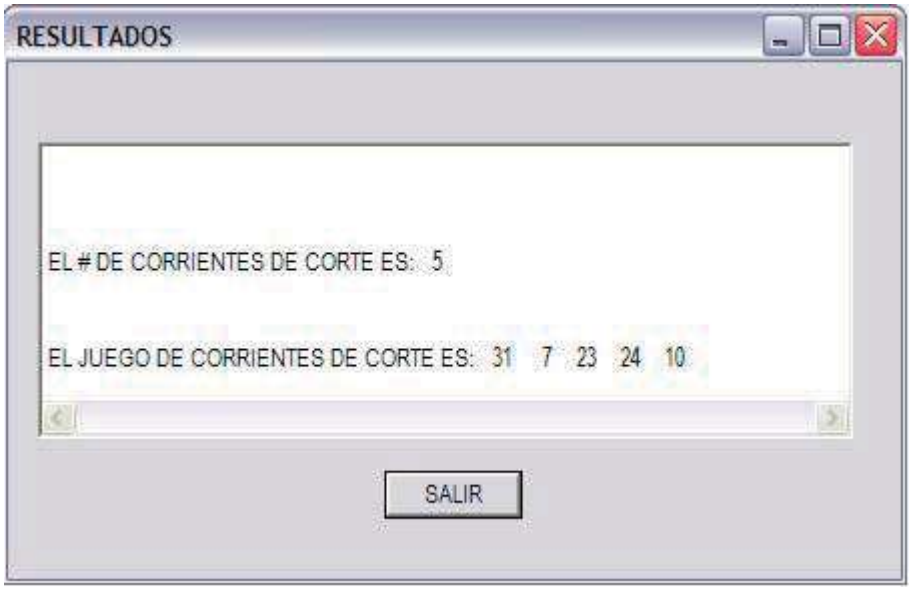


Figura B.7. Ventana de Resultados Ollero - Amselem.

La Figura B.7 nos muestra los resultados para el ejemplo de la Figura A.1.

En este caso resolvimos todo el sistema simultáneamente. Otra posibilidad es tomar los subsistemas encontrados por el programa de particionado y resolverlos por separado; el número de corrientes de corte y el conjunto de corrientes de corte debe de ser el mismo para ambos casos.

OE - AV

En la Figura B.8 podemos ver la ventana principal del programa OE – AV, en la cual tenemos que proporcionar el número de ecuaciones, el número de variables y el número de elementos no nulos de la matriz de funcionalidad.

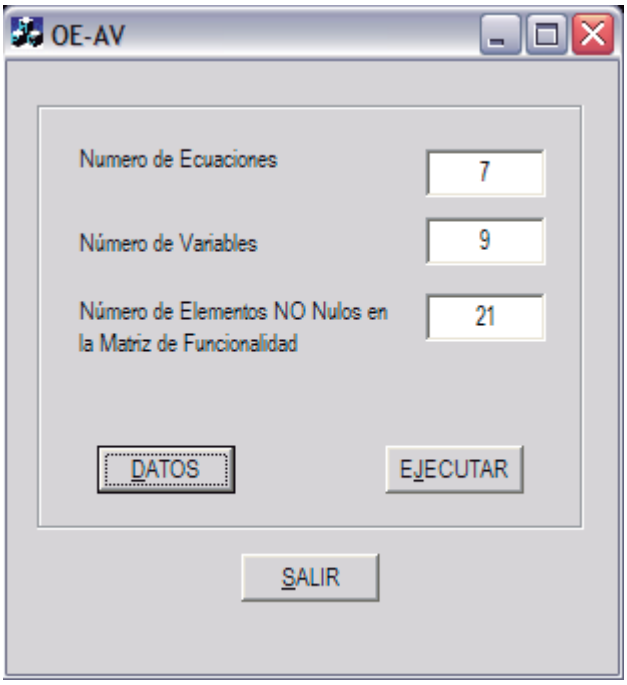


Figura B.8. Ventana Principal OE – AV.

Después de fijar los valores necesarios en la pantalla principal, hacemos clic sobre el botón de "DATOS" para abrir la pantalla donde tenemos que introducir las coordenadas de los elementos no nulos de la matriz de funcionalidad (Figura B.9).

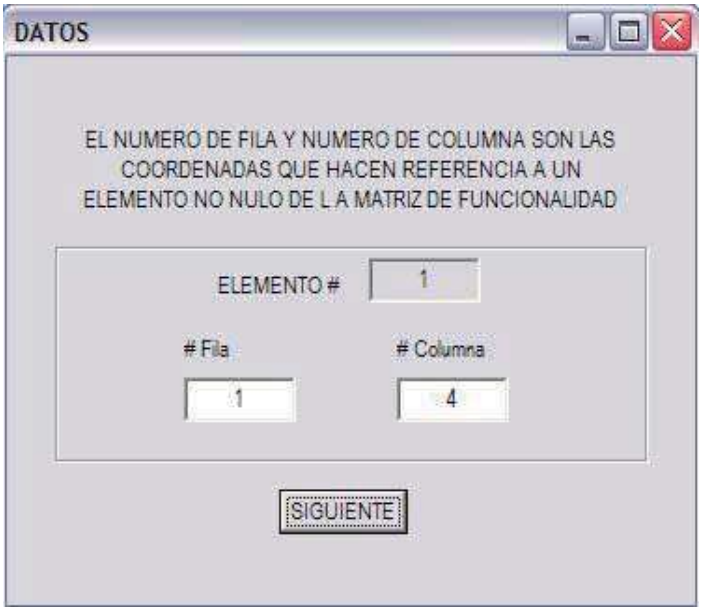


Figura B.9. Ventana de Datos OE – AV.

Al terminar de cargar los datos, regresamos a la pantalla principal y hacemos clic en el botón de “EJECUTAR”, con lo que aparecerá un archivo de texto con los resultados.

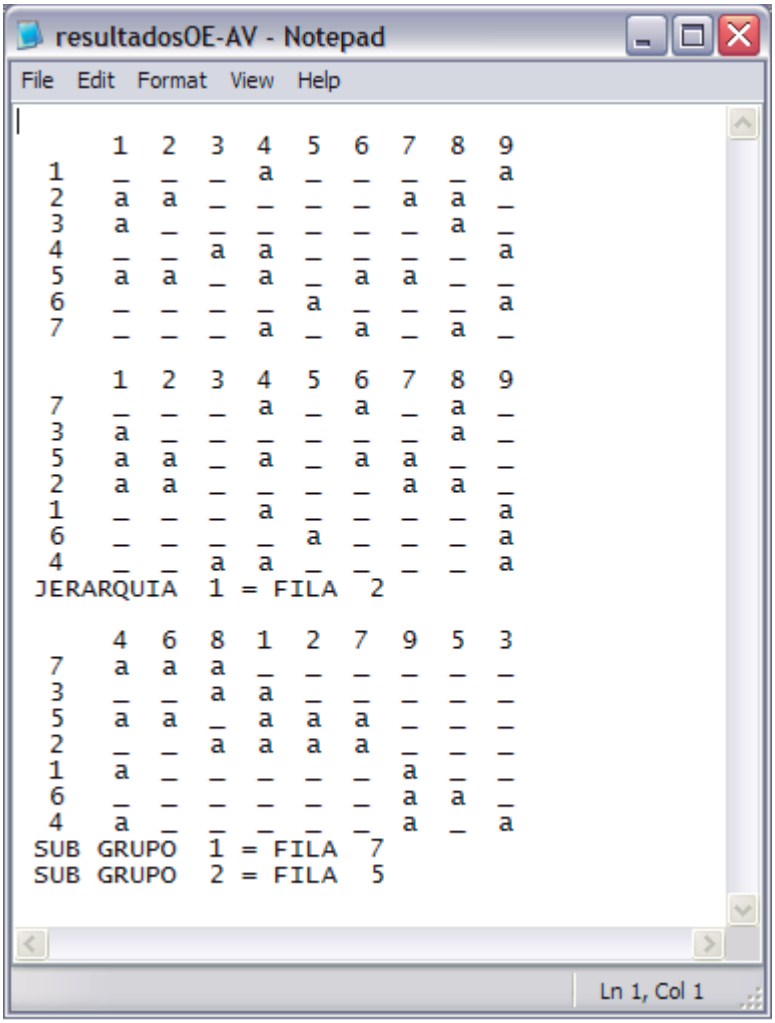


Figura B.10. Ventana de Resultados OE – AV.

La Figura B.10 nos muestra los resultados para el problema propuesto en el APENDICE A sección 2.

En el archivo de texto que almacena los resultados del programa OE – AV encontramos en primer lugar la matriz de funcionalidad, la siguiente matriz es la matriz resultante de aplicar el algoritmo OE y la tercera matriz es la matriz de funcionalidad rearreglada, es decir, la matriz que nos

proporciona la información sobre la estructura del problema. También se almacena información sobre los subgrupos encontrados.

A continuación se analiza la matriz de funcionalidad rearreglada de la Figura B.10 para dar al usuario una guía para su interpretación.

En primer lugar se observa que contiene dos subgrupos. El primero de ellos empieza en la Fila 7 y contiene todas las filas que le precedan hasta llegar a la fila 5; el subgrupo 2 está conformado por la fila 5 y todas las filas que le precedan.

Observamos que el subgrupo 1 está formado por dos ecuaciones y cuatro variables, por lo que es necesario seleccionar dos variables como variables de diseño; si escogemos las variables 4 y 6 podremos obtener la variable 8 de la ecuación 7 y la variable 1 de la ecuación 3. Al pasar al subgrupo 2 encontramos que las ecuaciones 2 y 5 contienen a las variables 2 y 7, es decir, se tienen dos ecuaciones con dos variables, por lo que se pueden resolver simultáneamente (en caso de no poderse resolver simultáneamente, se escoge una de las variables como variable de diseño o de corte y se resuelve la otra con una de las ecuaciones). Finalmente vemos que las ecuaciones 1, 6 y 4 contienen sólo una variable (9, 5 y 3, respectivamente); en consecuencia, podemos resolver cada una de las variables con su ecuación asociada.

Apéndice C: Código de los Programas en FORTRAN

DFS

```
      PROGRAM STRON
!
      INTEGER CI(1000),RI(1000),STRONG(2000)
      INTEGER NVER,NEDGE
      INTEGER I
!
!     Abrir Archivos de Datos y Resultados
!
      OPEN(5,FILE='DatosDFS.txt')
      OPEN(6,FILE='ResDFS.txt')
      READ(5,*)NVER,NEDGE

      DO 2 I=1,NEDGE
            READ(5,*)RI(I)

2     CONTINUE

      DO 3 I=1,NEDGE
            READ(5,*)CI(I)

3     CONTINUE
!
!     Encuentra los componentes fuertes del grafo
!
      CALL STRCOM(NVER,NEDGE,CI,RI,STRONG,ISTR)
      WRITE (6,*)STRONG
      STOP
      END
!
      SUBROUTINE STRCOM(NVER,NEDGE,CI,RI,STRONG,ISTR)
      INTEGER
      CI(1000),LLINK(1000),NUMB(1000),PRED(1000),RI(1000),STACK(1000),STR
      ONG(2000)
!
!     Inicializar  variables
!
      K=0
      INDEX=0
      ISTR=0
      DO 8 IA=1,NVER
            NUMB(IA)=-1
8     STACK(IA)=0
!
!     Encontrar un vértice no visitado y uno de sus arcos de salida
!
9     DO 10 II=1,NEDGE
            IF(RI(II).NE.0) GO TO 11
            GO TO 10
11          I=RI(II)
            IR=II
            RI(II)=0
```

```
                GO TO 12
        10      CONTINUE
                RETURN

        12      K=K+1
                INDEX=INDEX+1
                NUMB(I)=K
                LLINK(K)=K
                STACK(INDEX)=K
                IF(IR.EQ.0) GO TO 23
!
!               Revisar si el arco examinado es incidente a un vértice previamente visitado
!
        21      J=CI(IR)
                IF(NUMB(J).NE.-1) GO TO 17
                PRED(J)=I
!
!               Encontrar un arco de salida del vértice J que no haya sido visitado
!
                DO 13 IK=2,NEDGE
                        IF(RI(IK)-J)13,14,13
        14              IR=IK
                        RI(IK)=0
                        I=J
                        GO TO 12
        13      CONTINUE
                IF(NUMB(J).EQ.-1)GO TO 33
                GO TO 23
        33      I=J
                IR=0
                GO TO 12
!
!               Revisar si la cabeza del arco ha sido almacenada
!
        17      DO 18 IN=1,INDEX
                        IF (STACK(IN).EQ.NUMB(J)) GO TO 34
        18      CONTINUE
                GO TO 19
!
!               Revisar si el vértice considerado ha sido visitado antes
!
        34      IF(LLINK(NUMB(I)).LE.NUMB(J)) GO TO 19
                LLINK(NUMB(I))=NUMB(J)
!
!               Encontrar un arco no examinado que sea incidente al vértice considerad
!
        19      DO 22 IL=2,NEDGE
                        IF(RI(IL).NE.I) GO TO 22
                        RI(IL)=0
                        IR=IL
                        GO TO 21
        22      CONTINUE
!
!               Revisar si el vértice I es la raíz de un componente fuerte
!
```

```
      23    IF(LLINK(NUMB(I)).NE.NUMB(I)) GO TO 24
            GO TO 25
!
!           Regresar
!
      24    L=PRED(I)
            IF(LLINK(NUMB(L)).GT.LLINK(NUMB(I))) LLINK(NUMB(L))=LLINK(NUMB(I))
            I=L
            GO TO 19
!
!           Ubicar la posición de la raíz almacenada y borrar todos los vértices
!           debajo de este. Se ha identificado un componente fuerte.
!
      25    DO 27 IT=1,INDEX
                  IF(STACK(IT).NE.NUMB(I)) THEN
                        GO TO 27
                  ELSE
                        M=IT
                        GO TO 28
                  ENDIF
      27    CONTINUE
!
      28    M1=0
            DO 29 IM=M,INDEX
                  DO 30 IJ=1,NVER
                        IF(NUMB(IJ).NE.STACK(IM))THEN
                              GO TO 30
                        ELSE
                              ISTR=ISTR+1
                              STRONG(ISTR)=IJ
                              STACK(IM)=0
                        ENDIF
      30    CONTINUE
            M1=M1+1
      29    CONTINUE
!
            ISTR=ISTR+1
            STRONG(ISTR)=0
!
!           Iniciar desde el nodo predecesor a la raíz del componente fuerte
identificado
!
!
      31    INDEX=INDEX-M1
            IF(INDEX.EQ.0) GO TO 9
            I=PRED(I)
            GO TO 19
!
END
```

Ollero – Amselem

```
      PROGRAM TEAR
      COMMON/A1/ADJAC
      COMMON/A2/NOREM,REMAIN
      INTEGER ADJAC(1000,1000),ES,IS(1000),OUTDEG,REMAIN(1000),TEARST(1000)


      OPEN(5,FILE='DATOS OA.txt')
      OPEN(6,FILE='RESULTADOS OA.txt')

      READ(5,*)CONTA
      READ(5,*)((ADJAC(I,J),J=1,CONTA),I=1,CONTA)
      READ(5,*)NOREM


      DO 50 I=1,NOREM
50    REMAIN(I)=I
      NOESS=0
100   ES=1
      IFL=0
200   NX=1
300   N=REMAIN(NX)
      CALL DEGREE(N,INDEG,OUTDEG)
      IF(INDEG.NE.0.AND.OUTDEG.NE.0) GO TO 500
      CALL ESSENT(N)
400   CALL MISC(NX,ISF)
      IF(ISF-2)800,300,1000
500   IF(ADJAC(N,N).NE.1)GO TO 600
      CALL ESSENT(N)
      IFL=1
      NOESS=NOESS+1
      TEARST(NOESS)=N
      GO TO 400
600   IF(ES.NE.INDEG.AND.ES.NE.OUTDEG)GO TO 700
      CALL NONESS(N)
      GO TO 400
700   IF(NX.GE.NOREM) GO TO 800
      NX=NX+1
      GO TO 300
800   IF(IFL.EQ.0) GO TO 900
      IFL=0
      GO TO 100
900   ES=ES+1
      GO TO 200
1000  WRITE(6,*)
 10   FORMAT(1H1,//,20X,'EL GRAFO DUAL ESTA VACIO')
      WRITE(6,11)NOESS
 11   FORMAT(//,'EL # DE CORRIENTES DE CORTE ES:',I4)
      WRITE(6,12)(TEARST(I),I=1,NOESS)
 12   FORMAT(//,'EL JUEGO DE CORRIENTES DE CORTE ES: ',20(I4,2X))
      STOP
      END
```

```
      SUBROUTINE DEGREE(N,INDEG,OUTDEG)

!     Calcula los grados de entrada y salida del nodo-corriente N

      COMMON/A1/ADJAC
      COMMON/A2/NOREM,REMAIN
      INTEGER ADJAC(1000,1000),REMAIN(1000),OUTDEG
      INDEG=0
      OUTDEG=0
      DO 100 I=1,NOREM
            INDEG=INDEG+ADJAC(N,REMAIN(I))
100         OUTDEG=OUTDEG+ADJAC(REMAIN(I),N)
      RETURN
      END

      SUBROUTINE NONESS(N)
!     Reducción no esencial de un nodo-corriente
      COMMON/A1/ADJAC
      COMMON/A2/NOREM,REMAIN
      INTEGER ADJAC(1000,1000),REMAIN(1000),IS(1000),OUTDEG
      IJ=0
      DO 100 I=1,NOREM
      IF(ADJAC(N,REMAIN(I)).NE.1) GO TO 100
      IJ=IJ+1
      IS(IJ)=REMAIN(I)
      ADJAC(N,REMAIN(I))=0
100   CONTINUE
      OUTDEG=IJ
      DO 200 I=1,OUTDEG
            DO 200 J=1,NOREM
      IF(ADJAC(REMAIN(J),IS(I)).EQ.1.OR.ADJAC(REMAIN(J),N).EQ.1)
      ADJAC(REMAIN(J),IS(I))=1
200   CONTINUE
      RETURN
      END

      SUBROUTINE ESSENT(N)
!     Reducción esencial de un nodo-corriente
      COMMON/A1/ADJAC
      COMMON/A2/NOREM,REMAIN
      INTEGER ADJAC(1000,1000),REMAIN(1000)
      DO 100 I=1,NOREM
      ADJAC(N,REMAIN(I))=0
100   ADJAC(REMAIN(I),N)=0
      RETURN
      END

      SUBROUTINE MISC(NX,ISF)
      COMMON/A2/NOREM,REMAIN
      INTEGER REMAIN(1000)
      IF(NX.NE.NOREM) GO TO 100
      ISF=1
      NOREM=NOREM-1
      GO TO 300
```

```
100     N1=NOREM-1
        DO 200 I=NX,N1
200     REMAIN(I)=REMAIN(I+1)
        ISF=2
        NOREM=NOREM-1
300     IF(NOREM.EQ.0) ISF=3
        RETURN
        END
```

OE – AV

```
IMPLICIT NONE
INTEGER :: I, J, K, COUNT = 0, COUNT_HIER = 0
INTEGER :: MIN_DF_COLUMN, SUM_DF_COLUMN
INTEGER :: N, M, Y, Z, ELEMENTS
CHARACTER (LEN = 1), DIMENSION (:,:), ALLOCATABLE :: FUN, RAR_FUN_ROW, &
  RAR_FUN_COL
INTEGER, DIMENSION (:), ALLOCATABLE :: DF_COLUMN, VARIABLES
INTEGER, DIMENSION (:), ALLOCATABLE :: DF_ROW, HIER, EQNS
INTEGER, DIMENSION (:), ALLOCATABLE :: ROW_FREQ
INTEGER :: ROW_FREQ_MIN, HIGHEST_HIERARCHY, ROWS_ELIMINATED
INTEGER :: ROWNUM_MIN_FREQ, TOP_OF_HIER
CHARACTER (LEN = 30) :: MATRIX_FORMAT, COUNT_FORMAT, COL_DF_T_FORMAT,
HEADER
CHARACTER (LEN = 30) :: FINAL_FUN_FORMAT
INTEGER :: COUNT_ROW_ORDER, DIAG_ELEMENT, COUNT_SUB_GROUP
INTEGER, DIMENSION (:), ALLOCATABLE :: FINAL_ROW_ORDER, FINAL_EQNS,
SUB_GROUP
CHARACTER (LEN = 1), DIMENSION (:,:), ALLOCATABLE :: FINAL_FUN_MATRIX
CHARACTER (LEN = 50) :: HIERARCHY_FORMAT, SUB_GROUP_FORMAT,
READ_FORMAT
CHARACTER (LEN = 1) :: LABEL
INTEGER :: HIERARCHY_ROW_ALREADY_ELIMINATED = 0

READ_FORMAT = "( I2, 1X, I2, 1X, A)"

OPEN (UNIT = 1, FILE = 'functionality.matrix', STATUS = 'OLD')
OPEN (UNIT = 2, FILE = 'resultados ramirez.txt')
READ(1,*) N
READ(1,*) M
READ(1,*) ELEMENTS

ALLOCATE (FUN (N,M), RAR_FUN_ROW(N,M), RAR_FUN_COL(N,M))
ALLOCATE (DF_COLUMN(M), VARIABLES(M))
ALLOCATE (DF_ROW(N), HIER(N), EQNS(N), ROW_FREQ(N))
ALLOCATE (FINAL_ROW_ORDER(N), FINAL_EQNS(N), SUB_GROUP(N))
ALLOCATE (FINAL_FUN_MATRIX(N,M))

DO I = 1,N
  DO J = 1,M
    FUN(I,J) = '_'
    RAR_FUN_ROW(I,J) = '_'
```

```
   RAR_FUN_COL(I,J) = '_'
  END DO
END DO

DO I = 1,ELEMENTS
  READ(1,*) Y, Z, LABEL
  FUN(Y,Z) = LABEL
END DO

CLOSE(1)


DO I = 1,N
  EQNS(I) = I
END DO
DO I = 1,M
  VARIABLES(I) = I
END DO

HEADER = "(1X, /, 4X, 100(1X,I2)/)"
FINAL_FUN_FORMAT = "(1X, I2, 1X, 100(2X,A))"
MATRIX_FORMAT = "(1X, 4(2X,A))"
SUB_GROUP_FORMAT = "(1X, 'SUB GRUPO', I3, ' = FILA', I3)"
HIERARCHY_FORMAT = "(1X, 'JERARQUIA', I3, ' = FILA', I3)"

WRITE(*,HEADER) VARIABLES
WRITE(2,HEADER) VARIABLES
DO I = 1,N
 WRITE(*,FINAL_FUN_FORMAT) EQNS(I), (FUN(I,J), J = 1,M)
 WRITE(2,FINAL_FUN_FORMAT) EQNS(I), (FUN(I,J), J = 1,M)
END DO

! Buscar una columna con grado de libertad igual a 1

CALL CALC_COLUMN_DF(N, M, DF_COLUMN, FUN, MIN_DF_COLUMN,
SUM_DF_COLUMN)

DO WHILE(SUM_DF_COLUMN /= 0)
  DO WHILE(MIN_DF_COLUMN == 1)

   CALL ONE_DF_COLUMN_ARRANGE(N, M, COUNT, DF_COLUMN, FUN,
RAR_FUN_ROW, &
     EQNS)

   CALL CALC_COLUMN_DF(N, M, DF_COLUMN, FUN, MIN_DF_COLUMN,
SUM_DF_COLUMN)

  END DO

! Buscar una columna con grado de libertad mayora a 1

  CALL MANY_DF_COLUMN_ARRANGE(N, M, COUNT, DF_COLUMN, MIN_DF_COLUMN,
&
   FUN, RAR_FUN_ROW, COUNT_HIER, HIER, EQNS)
```

```
 CALL CALC_COLUMN_DF(N, M, DF_COLUMN, FUN, MIN_DF_COLUMN,
SUM_DF_COLUMN)

! Regresar a buscar una columna con grado de libertad igual a 1

END DO

COL_DF_T_FORMAT = "(1X, A, I3)"
COUNT_FORMAT = "(1X, A, I4)"


WRITE(*,HEADER) VARIABLES
WRITE(2,HEADER) VARIABLES
DO I = 1,N
 WRITE(*,FINAL_FUN_FORMAT) EQNS(I), (RAR_FUN_ROW(I,J), J = 1,M)
 WRITE(2,FINAL_FUN_FORMAT) EQNS(I), (RAR_FUN_ROW(I,J), J = 1,M)
END DO

COUNT = 0
COUNT_ROW_ORDER  = 0
COUNT_SUB_GROUP = 0
DIAG_ELEMENT = 1
HIGHEST_HIERARCHY = COUNT_HIER
TOP_OF_HIER = 1
IF (COUNT_HIER == 0) HIGHEST_HIERARCHY = 1

DO I = 1,N
  IF (HIER(I) == 0) HIER(I) = N
END DO

IF (COUNT_HIER /= 0) THEN
  DO I = COUNT_HIER, 1, -1
    WRITE(*,HIERARCHY_FORMAT) I, EQNS(HIER(I))
          WRITE(2,HIERARCHY_FORMAT) I, EQNS(HIER(I))
  END DO
ELSE
  WRITE(*,*) 'NO HIERARCHY POINTERS'
  WRITE(2,*) 'NO HIERARCHY POINTERS'
END IF


CALL CHECK_HIER_STATUS(N, M, RAR_FUN_ROW, HIER, COUNT_HIER, &
     TOP_OF_HIER, HIGHEST_HIERARCHY, HIERARCHY_ROW_ALREADY_ELIMINATED)

ROWS_ELIMINATED = 0

DO WHILE (ROWS_ELIMINATED < N)

! Calcular la frecuencia de las columnas, par la jerarquía mayor

  CALL CALC_ROW_FREQ(N, M, RAR_FUN_ROW, ROW_FREQ, ROW_FREQ_MIN, &
    ROWNUM_MIN_FREQ, TOP_OF_HIER, HIER, HIGHEST_HIERARCHY)

  CALL CHECK_HIER_STATUS(N, M, RAR_FUN_ROW, HIER, COUNT_HIER, &
     TOP_OF_HIER, HIGHEST_HIERARCHY, HIERARCHY_ROW_ALREADY_ELIMINATED)
```

```
! Borrar una fila si la frecuencia mínima en la jerarquía actual es 1


  IF (ROW_FREQ_MIN == 1) THEN

    CALL ELIMINATE_ONE_ROW(N, M, RAR_FUN_ROW, RAR_FUN_COL, &
      ROWNUM_MIN_FREQ, COUNT, VARIABLES, COUNT_ROW_ORDER, &
      FINAL_ROW_ORDER, DIAG_ELEMENT, HIER, HIGHEST_HIERARCHY, &
      HIERARCHY_ROW_ALREADY_ELIMINATED, TOP_OF_HIER)

  END IF

! Si no existe una fila con frecuencia uno borrar la primera fila


  IF (ROW_FREQ_MIN /= 1 .AND. ROW_FREQ_MIN /= 0) THEN

    CALL ELIMINATE_TOP_ROW(N, M, RAR_FUN_ROW, RAR_FUN_COL, ROW_FREQ,
COUNT, &
     VARIABLES, COUNT_ROW_ORDER, FINAL_ROW_ORDER, DIAG_ELEMENT, &
     COUNT_SUB_GROUP, SUB_GROUP, EQNS, TOP_OF_HIER, HIER, &
     HIGHEST_HIERARCHY, HIERARCHY_ROW_ALREADY_ELIMINATED)

  END IF

  ROWS_ELIMINATED = ROWS_ELIMINATED + 1

END DO

! Formar la matriz re-arreglada

CALL RAR_FINAL_ROW_ORDER(N, M, FINAL_ROW_ORDER, RAR_FUN_COL, &
  FINAL_FUN_MATRIX, FINAL_EQNS, EQNS)
WRITE(*,HEADER) VARIABLES
WRITE(2,HEADER) VARIABLES
DO I = 1,N
  WRITE(*,FINAL_FUN_FORMAT) FINAL_EQNS (I), (FINAL_FUN_MATRIX(I, J), J=1,M)
  WRITE(2,FINAL_FUN_FORMAT) FINAL_EQNS (I), (FINAL_FUN_MATRIX(I, J), J=1,M)
END DO


DO I = 1,COUNT_SUB_GROUP
  WRITE(*,SUB_GROUP_FORMAT) I, SUB_GROUP(I)
  WRITE(2,SUB_GROUP_FORMAT) I, SUB_GROUP(I)
END DO

END


SUBROUTINE CALC_COLUMN_DF(N, M, DF_COLUMN, FUN, MIN_DF_COLUMN,
SUM_DF_COLUMN)
IMPLICIT NONE
INTEGER :: I, J
INTEGER :: N, M
```

```
INTEGER, DIMENSION (M) :: DF_COLUMN
CHARACTER (LEN = 1), DIMENSION (N,M) :: FUN
INTEGER :: MIN_DF_COLUMN, SUM_DF_COLUMN

! Calcular los grados de libertad para cada fila

  DF_COLUMN(1:M) = 0

  DO I = 1,M
    DO J = 1,N
      IF (FUN(J,I) /= '_') DF_COLUMN(I) = DF_COLUMN(I) + 1
    END DO
  END DO

  MIN_DF_COLUMN = 1000

  DO I = 1,M
    IF (DF_COLUMN(I) < MIN_DF_COLUMN .AND. DF_COLUMN(I) /= 0)  &
       MIN_DF_COLUMN = DF_COLUMN(I)
  END DO

  SUM_DF_COLUMN = SUM(DF_COLUMN)


END


SUBROUTINE ONE_DF_COLUMN_ARRANGE(N, M, COUNT, DF_COLUMN, FUN, &
   RAR_FUN_ROW, EQNS)
IMPLICIT NONE
INTEGER :: I, J, COUNT
INTEGER :: N, M
INTEGER, DIMENSION (M) :: DF_COLUMN
CHARACTER (LEN = 1), DIMENSION (N,M) :: FUN, RAR_FUN_ROW
INTEGER, DIMENSION (N) :: EQNS

  DO I = 1,M
    IF(DF_COLUMN(I) == 1) THEN
      COUNT = COUNT + 1
      DO J = 1,N
        IF(FUN(J,I) /= '_') THEN
          RAR_FUN_ROW(N+1-COUNT,1:M) = FUN(J,1:M)
          FUN(J,1:M) = '_'
          EQNS(N+1-COUNT) = J
        END IF
      END DO
      EXIT
    END IF
  END DO

END


SUBROUTINE      MANY_DF_COLUMN_ARRANGE(N,      M,      COUNT,      DF_COLUMN,
MIN_DF_COLUMN, &
```

```
   FUN, RAR_FUN_ROW, COUNT_HIER, HIER, EQNS)
IMPLICIT NONE
INTEGER :: I, J, COUNT, MIN_DF_COLUMN, COUNT_HIER
INTEGER :: HIERARCHY
INTEGER :: N, M
CHARACTER (LEN = 1), DIMENSION (N,M) :: FUN, RAR_FUN_ROW
INTEGER, DIMENSION (M) :: DF_COLUMN
INTEGER, DIMENSION (N) :: HIER, EQNS

  HIERARCHY = 0
  DO I = 1,M
    IF(DF_COLUMN(I) == MIN_DF_COLUMN) THEN
      DO J = 1,N
        IF(FUN(J,I) /= '_') THEN
          HIERARCHY = HIERARCHY + 1
          COUNT = COUNT + 1
          RAR_FUN_ROW(N+1-COUNT,1:M) = FUN(J,1:M)
          FUN(J,1:M) = '_'
          EQNS(N+1-COUNT) = J
            IF (HIERARCHY == 1) THEN
               COUNT_HIER = COUNT_HIER + 1
               HIER(COUNT_HIER) = N+1-COUNT
            END IF
        END IF
      END DO
      EXIT
    END IF
  END DO

END

SUBROUTINE CALC_ROW_FREQ(N, M, RAR_FUN_ROW, ROW_FREQ, ROW_FREQ_MIN,
&
     ROWNUM_MIN_FREQ, TOP_OF_HIER, HIER, HIGHEST_HIERARCHY)
IMPLICIT NONE
INTEGER :: I, J, ROW_FREQ_MIN, ROWNUM_MIN_FREQ
INTEGER :: N, M, TOP_OF_HIER, HIGHEST_HIERARCHY, ken
INTEGER, DIMENSION (N) :: ROW_FREQ, HIER
CHARACTER (LEN = 1), DIMENSION (N,M) :: RAR_FUN_ROW

! Calcular el grado de libertad de la fila más arriba en la matriz

ROW_FREQ(TOP_OF_HIER:HIER(HIGHEST_HIERARCHY)) = 0

DO I = TOP_OF_HIER,HIER(HIGHEST_HIERARCHY)
  DO J = 1,M
    IF (RAR_FUN_ROW(I,J) /= '_') ROW_FREQ(I) = ROW_FREQ(I) + 1
  END DO
END DO

! Encontrar la fila con menor numero de grados de libertad

ROW_FREQ_MIN = 1000

DO I = TOP_OF_HIER,HIER(HIGHEST_HIERARCHY)
```

```
  IF (ROW_FREQ(I) /= 0 .AND. ROW_FREQ(I) < ROW_FREQ_MIN) THEN
    ROW_FREQ_MIN = ROW_FREQ(I)
    ROWNUM_MIN_FREQ = I
  END IF
END DO

END SUBROUTINE CALC_ROW_FREQ


SUBROUTINE ELIMINATE_TOP_ROW(N, M, RAR_FUN_ROW, RAR_FUN_COL,
ROW_FREQ, &
   COUNT, VARIABLES, COUNT_ROW_ORDER, FINAL_ROW_ORDER, DIAG_ELEMENT, &
   COUNT_SUB_GROUP, SUB_GROUP, EQNS, TOP_OF_HIER, HIER,
HIGHEST_HIERARCHY,&
   HIERARCHY_ROW_ALREADY_ELIMINATED)
IMPLICIT NONE
INTEGER :: N, M
INTEGER :: I, J, COUNT_ELIM_ROW, COUNT, L, TOP_OF_HIER, HIGHEST_HIERARCHY
INTEGER, DIMENSION (N) :: ROW_FREQ
CHARACTER (LEN = 1), DIMENSION (N,M) :: RAR_FUN_ROW, RAR_FUN_COL
INTEGER, DIMENSION (M) :: VARIABLES
INTEGER :: COUNT_ROW_ORDER, DIAG_ELEMENT, COUNT_SUB_GROUP,
ROW_SHUFFLED
INTEGER, DIMENSION (N) :: FINAL_ROW_ORDER, SUB_GROUP, EQNS, HIER
INTEGER :: FREQ_HIERARCHY_ROW, HIERARCHY_ROW_ALREADY_ELIMINATED

! Eliminar todas las columnas de la fila de la primera fila

COUNT_ELIM_ROW = 0

DO I = TOP_OF_HIER, HIER(HIGHEST_HIERARCHY)
  ROW_SHUFFLED = 0
  DO L = TOP_OF_HIER, HIER(HIGHEST_HIERARCHY)
    IF (FINAL_ROW_ORDER(L) == I) ROW_SHUFFLED = 1
  END DO
  IF (ROW_SHUFFLED /= 1) THEN
    IF (ROW_FREQ(I) /= 0) COUNT_ELIM_ROW = COUNT_ELIM_ROW + 1
    IF (COUNT_ELIM_ROW == 1) THEN
      DO J = 1,M
        IF (RAR_FUN_ROW(I,J) /= '_') THEN
          COUNT = COUNT + 1
          RAR_FUN_COL(1:N, COUNT) = RAR_FUN_ROW(1:N, J)
          RAR_FUN_ROW(1:N, J) = '_'
          VARIABLES(COUNT) = J
        END IF
      END DO
      COUNT_ROW_ORDER = COUNT_ROW_ORDER + 1
      FINAL_ROW_ORDER(COUNT_ROW_ORDER) = I
      DIAG_ELEMENT = I
      COUNT_SUB_GROUP = COUNT_SUB_GROUP + 1
      SUB_GROUP(COUNT_SUB_GROUP) = EQNS(I)
    END IF
  END IF

END DO
```

```
IF (HIERARCHY_ROW_ALREADY_ELIMINATED == 0) THEN

FREQ_HIERARCHY_ROW = 0

DO I = 1,M
  IF (RAR_FUN_ROW(HIER(HIGHEST_HIERARCHY), I) /= '_') THEN
    FREQ_HIERARCHY_ROW = FREQ_HIERARCHY_ROW + 1
  END IF
END DO

IF (FREQ_HIERARCHY_ROW == 0) THEN
  COUNT_ROW_ORDER = COUNT_ROW_ORDER + 1
  FINAL_ROW_ORDER(COUNT_ROW_ORDER) = HIER(HIGHEST_HIERARCHY)
END IF

END IF

END SUBROUTINE ELIMINATE_TOP_ROW



SUBROUTINE ELIMINATE_ONE_ROW(N, M, RAR_FUN_ROW, RAR_FUN_COL, &
  ROWNUM_MIN_FREQ,          COUNT,          VARIABLES,          COUNT_ROW_ORDER,
FINAL_ROW_ORDER, &
  DIAG_ELEMENT,                HIER,                                HIGHEST_HIERARCHY,
HIERARCHY_ROW_ALREADY_ELIMINATED, &
  TOP_OF_HIER)
IMPLICIT NONE
INTEGER :: N, M
INTEGER :: I, J, COUNT, COUNT_ELIM_ROW, ROWNUM_MIN_FREQ
CHARACTER (LEN = 1), DIMENSION (N,M) :: RAR_FUN_ROW, RAR_FUN_COL
INTEGER, DIMENSION (M) :: VARIABLES
INTEGER :: COUNT_ROW_ORDER, DIAG_ELEMENT, HIGHEST_HIERARCHY
INTEGER :: FREQ_HIERARCHY_ROW
INTEGER, DIMENSION (N) :: FINAL_ROW_ORDER, HIER
INTEGER :: HIERARCHY_ROW_ALREADY_ELIMINATED
INTEGER, DIMENSION (N) :: PRE_ROW_DF, POST_ROW_DF, POST_MINUS_PRE
INTEGER :: TOP_OF_HIER


PRE_ROW_DF(TOP_OF_HIER:HIER(HIGHEST_HIERARCHY)) = 0
DO I = TOP_OF_HIER, (HIER(HIGHEST_HIERARCHY)-1)
  DO J = 1,M
    IF (RAR_FUN_ROW(I,J) /= '_') PRE_ROW_DF(I) = PRE_ROW_DF(I) + 1
  END DO
END DO

COUNT_ELIM_ROW = 0

! Se encontró una fila con grado de libertad igual a 1, elimina esa fila

DO I = 1, M
  IF (RAR_FUN_ROW(ROWNUM_MIN_FREQ, I) /= '_') THEN
```

```
    COUNT_ELIM_ROW = COUNT_ELIM_ROW + 1
    IF (COUNT_ELIM_ROW == 1) THEN
      COUNT = COUNT + 1
      RAR_FUN_COL(1:N, COUNT) = RAR_FUN_ROW(1:N, I)
      RAR_FUN_ROW(1:N, I) = '_'
      VARIABLES(COUNT) = I
      COUNT_ROW_ORDER = COUNT_ROW_ORDER + 1

      IF (ROWNUM_MIN_FREQ >= DIAG_ELEMENT) THEN
        FINAL_ROW_ORDER(COUNT_ROW_ORDER) = ROWNUM_MIN_FREQ
        DIAG_ELEMENT = DIAG_ELEMENT + 1
      ELSE
        FINAL_ROW_ORDER(COUNT_ROW_ORDER) = I
      ENDIF

    END IF
  END IF
END DO

POST_ROW_DF(TOP_OF_HIER:HIER(HIGHEST_HIERARCHY)) = 0
DO I = TOP_OF_HIER, (HIER(HIGHEST_HIERARCHY)-1)
  DO J = 1,M
    IF (RAR_FUN_ROW(I,J) /= '_') POST_ROW_DF(I) = POST_ROW_DF(I) + 1
  END DO
END DO

POST_MINUS_PRE = POST_ROW_DF - PRE_ROW_DF
DO I = TOP_OF_HIER, (HIER(HIGHEST_HIERARCHY)-1)
IF (POST_MINUS_PRE(I) < 0 .AND. I /= ROWNUM_MIN_FREQ) THEN
  IF (POST_ROW_DF(I) == 0) THEN
    COUNT_ROW_ORDER = COUNT_ROW_ORDER + 1
    FINAL_ROW_ORDER(COUNT_ROW_ORDER) = I
  END IF
END IF
END DO

IF (ROWNUM_MIN_FREQ == HIER(HIGHEST_HIERARCHY)) THEN
  HIERARCHY_ROW_ALREADY_ELIMINATED = 1
END IF

IF (HIERARCHY_ROW_ALREADY_ELIMINATED == 0) THEN

FREQ_HIERARCHY_ROW = 0

DO I = 1,M
  IF (RAR_FUN_ROW(HIER(HIGHEST_HIERARCHY), I) /= '_') THEN
    FREQ_HIERARCHY_ROW = FREQ_HIERARCHY_ROW + 1
  END IF
END DO

IF (FREQ_HIERARCHY_ROW == 0) THEN
  COUNT_ROW_ORDER = COUNT_ROW_ORDER + 1
  FINAL_ROW_ORDER(COUNT_ROW_ORDER) = HIER(HIGHEST_HIERARCHY)
END IF
```

```
END IF

END SUBROUTINE ELIMINATE_ONE_ROW


SUBROUTINE CHECK_HIER_STATUS(N, M, RAR_FUN_ROW, HIER, COUNT_HIER, &
  TOP_OF_HIER, HIGHEST_HIERARCHY, HIERARCHY_ROW_ALREADY_ELIMINATED)
IMPLICIT NONE
INTEGER :: N, M
INTEGER :: I, J, HIGHEST_HIERARCHY
CHARACTER (LEN = 1), DIMENSION (N,M) :: RAR_FUN_ROW
INTEGER, DIMENSION (N) :: HIER
INTEGER :: COUNT_HIER, TOP_OF_HIER, HIER_STATUS
INTEGER :: HIERARCHY_ROW_ALREADY_ELIMINATED


HIER_STATUS = 0
DO I = TOP_OF_HIER, HIER(HIGHEST_HIERARCHY)
  DO J = 1, M
    IF (RAR_FUN_ROW(I, J) /= '_') HIER_STATUS = 1
  END DO
END DO

IF (HIER_STATUS == 0) THEN
  TOP_OF_HIER = HIER(HIGHEST_HIERARCHY) + 1
  COUNT_HIER = COUNT_HIER - 1
  HIGHEST_HIERARCHY = COUNT_HIER
  IF (COUNT_HIER == 0) HIGHEST_HIERARCHY = N
  HIERARCHY_ROW_ALREADY_ELIMINATED = 0
END IF

END SUBROUTINE CHECK_HIER_STATUS


SUBROUTINE RAR_FINAL_ROW_ORDER(N, M, FINAL_ROW_ORDER, RAR_FUN_COL, &
  FINAL_FUN_MATRIX, FINAL_EQNS, EQNS)
IMPLICIT NONE
INTEGER :: N, M, I, NEXT_ROW
CHARACTER (LEN = 1), DIMENSION (N,M) :: RAR_FUN_COL, FINAL_FUN_MATRIX
INTEGER, DIMENSION (N) :: FINAL_ROW_ORDER, EQNS, FINAL_EQNS


DO I = 1,N
  NEXT_ROW = FINAL_ROW_ORDER(I)
  FINAL_FUN_MATRIX(I, 1:M) = RAR_FUN_COL(NEXT_ROW, 1:M)
  FINAL_EQNS(I) = EQNS(NEXT_ROW)
END DO

END SUBROUTINE RAR_FINAL_ROW_ORDER
```

CAPITULO 6: BIBLIOGRAFIA

Barkley R. W. y Motard R. L. (1972). Decomposition of Nets. Chem Eng. J., 3, 265-275.

J. H. Christensen y D. F. Rudd (1969). Structuring design computations. AIChE Journal. 94 – 100.

Duff, I. S. y Reid, J. K. (1978). An Implementation of Tarjan's Algorithm for the Block Triangularization of a Matriz. ACM Transactions on Mathematical Software. 4(2), 137-147.

Ki-Won y col. (1999). Application of Partitioning and Tearing Techniques to Sulfolane Extraction Plant. Korean J. Chem. Eng. 16(4), 462-469.

Li y col. (1988). A New Strategy of Net Decomposition in Process Simulation. Comput. Chem. Eng., Vol. 12, No. 6, 581-588.

Mah, S.H. R. (1990). *Chemical Process Structures and Information Flows*. Butterworths (USA).

Norman, R.L. (1962). A Matrix Method for Location of Cycles of a Directed Graph. AIChE Journal, Vol. 11, No. 3. 450-452.

Ollero, P. y C. Amselem (1983). Decomposition Algorithm for Chemical Process Simulation. *Chem. Eng. Res. Des*., 61, 303-307.

Ramirez, W. F. (1997). *Computational Methods for Process Simulation*. Butterworths (USA).

Rudd, D.F. y Watson, C.C. (1968). *Strategy of Process Engineering*, John Wiley & Sons, Inc., New York, 34-49.

Sargent, R.W.H. y A.W. Westerberg (1964). SPEEDUP in Chemical Engineering Design. *Trans. Inst. Chem. Engrs.* Vol. 42, T190-197.

Tarifa y col. (2004). A New Method to Process Algebraic Equation Sistems Used to Model a MSF Desalination Plant. ELSEVIER Desalination 166 (2004) 113-121.

Tarjan, R. (1972). Depth-firs Search and Linear Graph Algorithms. *SIAM J. Compt.*, 1(2), 146-160.

Upadhye R.S. y Grens E.A., II (1972). An Efficient Algorithm for Optimum Decomposition of Recycle Systems. *AIChE J.*, 18, 533-439.

Westerberg y col. (1979). Process Flowsheeting. Cambridge University Press.