UNIVERSIDAD NACIONAL AUTONOMA

DE MEXICO

UNIDAD ACADEMICA DE LOS CICLOS
PROFESIONAL Y DE POSGRADO
DEL
COLEGIO DE CIENCIAS Y HUMANIDADES

INSTITUTO DE INVESTIGACIONES EN
MATEMATICAS APLICADAS Y SISTEMAS

TESIS QUE PARA OBTENER EL GRADO DE MAESTRO

EN CIENCIAS DE LA COMPUTACION

PRESENTA

ARTURO ANTONIO RAMIREZ GALLEGOS

México, D. F., 1991

UNAM – Dirección General de Bibliotecas Tesis Digitales Restricciones de uso

G E M O L I

UN SISTEMA PARA LA GENERACION E INTEGRACION DE MODELOS LINEALES

CONTENIDO

1 INTRODUCCION

1.1 ANTECEDENTES

A partir de la generalización del uso del método SIMPLEX para la resolución de problemas de programación lineal, se han realizado múltiples esfuerzos encaminados a facilitar el proceso de desarrollo de este tipo de modelos, así como para mejorar y generalizar los algoritmos de resolución.

Conforme se ha avanzado y extendido el uso de estas herramientas, gracias a la penetración de las computadoras, los problemas que se pueden desarrollar y resolver han crecido progresivamente en tamaño y complejidad.

Como elemento clave de esta evolución, además de la aparición del método SIMPLEX ya citado, se puede mencionar el grado de desarrollo y consistencia de los paquetes de programación lineal existentes.

A la fecha, estos paquetes tienen como objetivo fundamental encontrar la solución del modelo y están orientados primordialmente al manejo óptimo de grandes problemas. Los esfuerzos de mejoramiento en esta área, han estado dirigidos para acelerar los algoritmos de resolución, lograr un uso más eficiente de los recursos de cómputo, y ampliar las dimensiones de los modelos posibles de manejar.

Todos los paquetes de programación lineal disponibles para equipos mayores, se han estandarizado en el formato y contenido de la matriz de coeficientes (formato MPS). Esta estandarización ha definido tácitamente una interfase de comunicación, y a su vez, una frontera conceptual de división entre el proceso de modelación del problema, y el proceso para encontrar la solución del modelo.

En la actualidad, el rezago tecnológico del proceso de modelación respecto al proceso de solución se hace evidente si se observa que, para la mayoría de los modelos utilizados en la práctica, la tarea más demandante de esfuerzo es la integración de la matriz de coeficientes, tarea más laboriosa que el mismo desarrollo matemático del modelo.

Los sistemas realizados para apoyar el proceso de modelación se pueden agrupar en dos tipos básicos:

- Aquéllos cuya finalidad primordial es la generación de la matriz de coeficientes, comunmente llamados los Generadores de Matrices, cuyo formato o lenguaje de definición está orientada a integrar un archivo en formato MPS.

- Y los Sistemas Intérpretes cuya fortaleza radica en reconocer lenguajes especializados para la definición de estos modelos.

El desarrollo de los generadores de matrices se llevó a cabo durante la década pasada, sin embargo, siguen siendo en la actualidad la herramienta más comunmente utilizada en el proceso de modelación. Como elementos representativos de este enfoque se pueden mencionar los siguientes sistemas: MaGen [3], DATAMAT [4], GAMMA [5] y MGRW [6].

Las limitaciones y desventajas de los Generadores de Matrices se superan ampliamente con el enfoque utilizado por los sistemas intérpretes. Una amplia exposición sobre este punto la realizó ROBERT FOURER en 1983 [1].

Como el prototipo más avanzado a la fecha de los sistemas intérpretes se puede mencionar al AMPL [2] (A Mathematical Programming Language) difundido a principios de 1987, en este sistema resalta el grado de desarrollo del lenguaje, ya que su sintaxis es muy similar a la forma y notación algebraica comunmente utilizada en la definición de estos modelos. Como esfuerzos anteriores de este mismo enfoque se pueden mencionar los sistemas: GAMS [7], [8] y ALPS [9], [10].

1.2 PREMISAS DE DISEÑO DEL SISTEMA

A efecto de disponer con un sistema intérprete: que aproveche el amplio desarrollo de la paquetería comercial disponible, tanto en computadoras personales como en equipos mayores; que fortalezca el proceso de modelación mediante un lenguaje de definición ADHOC para modelos de programación lineal, y que mediante archivos ASCII facilite la intercomunicación entre diversos modelos; se desarrolló el sistema GEMOLI bajo las premisas de diseño que se describen a continuación.

La fortaleza de todo sistema intérprete radica primordialmente en las características del lenguaje que interpreta, por tal motivo, especial atención se puso en la definición de las premisas del lenguaje, las cuales se enuncian a continuación:

- El lenguaje deberá estar orientado especialmente a definir modelos de programación lineal.

- El lenguaje deberá describir el modelo, no el procedimiento o las etapas de la resolución del mismo.

- Todos y cada uno de los elementos de un modelo deberán ser

definidos y descritos en la forma más conceptual, directa y concisa posible.

- El lenguaje deberá permitir definiciones de arriba hacia abajo y de lo general a lo particular.

Estas premisas asegurarán el poder enfrentar modelos extremadamente complejos, con una metodología de definición que permitirá respuestas perentorias.

Las premisas de diseño que se refieren a la implementación propiamente del sistema, son las siguientes:

El sistema deberá manejar eficientemente grandes modelos, deberá poder manipular matrices de coeficientes con vectores hasta de 30,000 elementos, además deberá asegurar un eficiente manejo de parámetros, en particular, se deberá contar con un algoritmo que optimice el cálculo de expresiones, eliminando la duplicidad de subexpresiones y maximizando el uso de parámetros en memoria, a efecto de eliminar lecturas repetitivas.

Con el objeto de aprovechar el desarrollo alcanzado en los paquetes comerciales de programación lineal, el sistema deberá ser capaz de integrar las interfases de entrada requeridas por los paquetes más comunes. Asimismo, el sistema deberá ser capaz de extraer de estos mismos paquetes la información solicitada sobre la solución óptima y las características de la misma.

Para asegurar la conectividad y encadenamiento entre modelos, se requiere el manejo de archivos tipo ASCII en disco, desde donde se pueda leer la información del modelo, y a través de los cuales se pueda comunicar a un sistema envolvente, además de los valores de la solución óptima, cualquier información conocida del modelo como pueden ser: las características de variable y restricción en la solución y/o los aspectos descriptivos (significado de variable e índices) que se hayan definido para el modelo.

En todo modelo de programación lineal, los coeficientes y los límites de las restricciones (parámetros del modelo) están sujetos a un rango válido de valores. Cuando un parámetro está fuera de su rango válido provoca que el sistema de ecuaciones definido ya no representa al problema modelado, o en algunos casos críticos, provoca que el sistema de ecuaciones sea inconsistente y sin solución. Existen también planteamientos que exigen, para llegar a una solución válida, que se cumplan ciertas relaciones lineales entre algunos parámetros del modelo. Por estas razones y debido también a la total libertad permitida para la definición y cálculo de los parámetros del modelo, ya que éstos pueden incluso provenir de un proceso automatizado, como última premisa se establece que tanto el

lenguaje como el sistema deberán permitir supervisar la validez de los valores introducidos, y verificar la integridad de los parámetros a través de expresiones de relación que representen los supuestos básicos.

1.3 ESTRUCTURA DE LA DESCRIPCION DEL SISTEMA

En la presente tesis se describen los conceptos y elementos básicos asociados a los sistemas intérpretes en general y a la implementación particular del sistema GEMOLI.

Al inicio de la presente introducción se mencionó la motivación para el desarrollo del sistema y los avances que se pretenden alcanzar con el mismo.

En el segundo capítulo se describen los conceptos básicos y las particularidades del álgebra utilizada en el lenguaje GEMOLI. Mismo que se describe al detalle y en su totalidad en el tercer capítulo.

El cuarto capítulo describe la problemática para el cálculo de las expresiones, así como los procedimientos y el algoritmo desarrollados para la optimización de dicho cálculo.

En el quinto capítulo se describe la implementación realizada para interpretar el lenguaje GEMOLI. En él se describe al sistema en base a los programas que lo integran, y a los archivos del sistema y su contenido genérico.

La evaluación del sistema se realiza en el sexto capítulo, se presentan los principales parámetros de comportamiento, además de conclusiones sobre el lenguaje y el sistema desarrollado.

La información de apoyo para el uso del sistema se presenta en dos apéndices:

El apéndice A lo constituye la definición en notación BNF, de la sintaxis del lenguaje GEMOLI.

En el apéndice B se presentan seis ejemplos con problemas típicos de programación lineal. Los primeros cinco ejemplos son sumamente sencillos y de inmediata comprensión con el objeto de enfocar la atención en la utilización del lenguaje para describirlos, así en cada uno de ellos se describe el problema, su especificación algebraica, su definición en lenguaje GEMOLI y la matriz generada por el sistema, que será la entrada a los diversos paquetes de programación lineal. Con el ejemplo 6 del apéndice B se pretende demostrar todo el potencial del lenguaje, por tal razón es el ejemplo más complejo y por lo mismo, únicamente se describe el problema a resolver y con mayor amplitud el modelo utilizado.

1.4 REFERENCIAS

[1] Fourer, Robert
Modeling Languages versus Matrix Generators for Linear Programming.
ACM Transactions on Mathematical Software
1983
Vol. 9, No. 2, págs. 143-183

[2] Fourer, Robert; Gay, David; Kernighan, Brian
AMPL: A Mathematical Programming Language.
Computing Science Technical Report No. 133
AT&T Bell Laboratories.
Murray Hill N.J.
1987

[3] MaGen: Reference Manual
Haverly Systems, Inc.
Denville, N.J.
1977

[4] DATAMAT Reference Manual
Center for Computational Research in Economics and Management Science.
M.I.T.
Cambridge, Mass.
1975.

[5] GAMMA 3.4 Procedures Summary
Sperry Univac Computer Systems
St. Paul, Minn.
1977

[6] Matrix Generator and Report Writer Program
Reference Manual
I.B.M. World Trade Corporation
New York, N.Y.
1972

[7] GAMS: General Algebraic Modeling System
Development Research Center
The World Bank
Washington, D.C.

[8] Bisschop J.; Meeraus A.
Toward Succesful Modeling Applications in Strategic Planning Environment.
Contenido en:
Dantzing, G.B.; Dempster, M.A.H.; Dallio, M.J. (Editores)
Large-Scale Linear Programming
International Institute for Applied System Analysis
Laxenburg, Austria
1981

[9] ALPS: Advanced Linear Programming System
United Computed Systems Inc.
Kansas City Mo.

[10] Steinberg, D.L.
ALPS (Advanced Linear Programming System)
ORSA/TIMS Joint National Meeting
Atlanta Ga.
1977

2 CONCEPTOS BASICOS DEL LENGUAJE

2.1 ASPECTOS INTEGRADOS AL LENGUAJE.

Existen varias formas para definir los modelos de programación lineal, la utilización de alguna de ellas está determinada por aquellos aspectos del modelo que se desean resaltar, así la forma descriptiva normalmente se utiliza en la enunciación inicial, posteriormente como resultado del proceso de análisis y formulación del modelo, en notación algebraica se establece el sistema de ecuaciones que lo definen. [1], [3], [6], [8], [10], [12] y [13].

Las notaciones matricial y vectorial se utilizan cuando se está analizando o describiendo el método simplex, y en general, estas notaciones son particularmente útiles cuando se describen los métodos de optimización lineal. [2], [4], [5], [7], [9], [11] y [14].

Adicional a estas formas generales, los paquetes de cómputo para la definición y solución de modelos lineales tienen también sus formas particulares de definición, tal es el caso del sistema que nos ocupa.

El lenguaje GEMOLI fue desarrollado EX-PROFESSO para la definición, descripción y manipulación automatizada de modelos lineales. En el lenguaje se integran los siguientes aspectos fundamentales:

- El rompimiento conceptual del modelo en sus partes constitutivas, lo cual asegura una disminución de la complejidad de la definición.

- La formulación algebraica del sistema de ecuación, lo cual permite definir al modelo en una forma ampliamente conocida, sucinta y sin ambigüedades.

- La integración de la descripción del modelo en la definición, objetivo que se logra asociando a cada elemento del modelo su significado, y permitiendo la intercalación a lo largo de la definición, de todo comentario que se juzgue necesario para darle sentido y claridad a la definición.

- Para asegurar una comunicación automatizada con otros sistemas y/o modelos, el lenguaje permite la definición de archivos y sus respectivos campos, así como la posibilidad de definir criterios de validación sobre la información de entrada.

La integración alcanzada en el Lenguaje GEMOLI proporciona un medio rápido y eficiente para la integración y GEneración de MOdelos de programación LIneal, objetivo último del sistema y de donde se sintetizó su nombre.

2.2 FORMULACION ALGEBRAICA DEL SISTEMA DE ECUACIONES

Como antecedente necesario para la descripción del lenguaje, es conveniente definir algunas particularidades de la notación y del álgebra utilizada en la definición de expresiones y ecuaciones.

Si se analizan diversas formulaciones algebraicas de este tipo de modelos, tanto en la literatura: [1], [3], [6], [8], [10], [12] y [13], como en los ejemplos contenidos en el apéndice B, se observará en todas ellas, que en la notación algebraica se hace un amplio uso del concepto de índices. Mediante índices se distinguen los diversos tipos de actividad en los modelos económicos, en donde también se usan éstos, para identificar períodos de tiempo; en modelos de transporte se utilizan para definir los orígenes y destinos de los tramos de las redes. En general: los índices se utilizan para diferenciar los diversos tipos de elementos del mundo físico que conforman el problema, y los valores que se asignan a cada índice, identifican inequívocamente a los elementos de un mismo tipo.

El lenguaje GEMOLI considera que cada variable, parámetro y restricción, definida e invocada en el modelo, es una función, en donde, como variables independientes, sólo se aceptan índices definidos para el modelo.

Para las variables y las restricciones, el sistema genera automáticamente valores únicos, que identifican inequívocamente las columnas y los renglones de las matrices MPSX, a partir del identificador de cada variable y/o restricción y de los índices asociados a las mismas.

En el caso de los parámetros, el lenguaje GEMOLI permite, a través de los postulados de <VALOR> y/o <DATO>, definir simultáneamente tanto el dominio como el rango de estas funciones; así al establecer los diferentes valores que puede tomar el parámetro, y asociar cada uno de estos a una combinación de valores válidos de los índices que definen el parámetro (dominio de la función), se está definiendo plenamente la función que representa al parámetro.

En el uso de funciones en base a índices, es común subdividir el rango de la función en subconjuntos, para hacerlos corresponder perfectamente con los criterios que determinan las ecuaciones en los que son invocados. A estos subconjuntos del dominio de las funciones los hemos denominado espacios de aplicación de la función.

La componente más importante que perfila las características del lenguaje GEMOLI, es sin duda, la notación para definir y manejar los espacios de aplicación de las variables, parámetros y restricciones del modelo.

2.2.1 INDICES Y SUBINDICES

El Lenguaje GEMOLI requiere, que en el postulado <IND> (índice), se defina para cada índice, además de la información asociada con su significado, el valor máximo que puede tomar, este valor establece el dominio teórico del índice en el modelo. Por definición de diseño, el dominio teórico de cada índice es el conjunto de valores enteros comprendidos entre 1 y el valor máximo declarado.

Al definirse para cada índice un dominio teórico, se está implicando: que cada valor del dominio teórico es un elemento del mundo físico del modelo, que es válido, totalmente identificable y que tiene un sentido lógico en el modelo.

En la actividad cotidiana de modelación, con mucha frecuencia, especialmente en modelos complicados, se requieren agrupar ciertos valores del dominio teórico de un índice, normalmente por que los elementos asociados cumplen con una cierta limitación o característica. GEMOLI permite hacer subconjuntos del dominio teórico de un índice mediante la declaración de subíndices.

La definición de un subíndice además del postulado <IND>, requiere del postulado complementario <SC> (Subconjunto de), en donde se especifica el índice padre y la definición explícita de los valores que integran el dominio teórico del subíndice. Así, cuando se invoca un subíndice, funcionalmente se está invocando un subdominio (dominio restringido) del índice padre.

Es válido definir subíndices de subíndices, sin existir limitación para este anidamiento.

En resumen, el dominio teórico de un índice o de un subíndice es el conjunto de valores que puede tomar el índice o el subíndice a todo lo largo del modelo. Este dominio queda establecido en los postulados <IND> (postulados de definición de índices y subíndices).

En el ejemplo 1 del apéndice B se puede observar que el índice I representa a las fábricas de envolturas; los distribuidores son representados por el índice J.

En el mismo ejemplo se observa que el dominio teórico de I es el conjunto de valores {1,2}; que I=1 representa lógicamente a la fábrica de Toluca y que I=2 representa a la fábrica de Querétaro.

En el ejemplo 2, I representa los lotes de materia prima, y J representa a los diversos metales que conformarán la aleación; también se puede observar que el dominio de J es el conjunto { 1,2,3,4,5,6 }, y que cada valor de J tiene asociado un metal, así:

J=1 representa al fierro,
J=2 representa al cobre,
:
:
J=6 representa al silicón

2.2.2 VARIABLES Y PARAMETROS

Para el sistema, como ya se ha mencionado, las variables, los parámetros y las restricciones son funciones de los índices del modelo, en donde el dominio o espacio teórico de aplicación para cada parámetro (variable o restricción) es el conjunto generado por el producto cartesiano de los dominios teóricos de los índices asociados al parámetro (variable o restricción).

Esta regla implica que el orden en que los índices se establecen en la definición del parámetro (variable o restricción), debe ser el mismo que se utilice en sus invocaciones.

Si se analizan los parámetros del ejemplo 2, se comprenderán las equivalencias entre la notación algebraica y la notación GEMOLI, que a continuación se presentan.

Notación Algebraica	Notación GEMOLI	Representa:
$COSTO.U_i$	COSTO.U(I)	Al conjunto rango de la función: {(1,.03) (2,.08) (3,.17) (4,.12) (5,.15) (6,.21) (7,.38) } que define los costos unitarios de las materias primas. El primer elemento de los pares ordenados (dominio de la función) son los valores que toma el índice I (diversas materias primas), el segundo elemento de los pares ordenados (rango de la función) son los costos unitarios.
$CONT.ME_{i,j}$	CONT.ME(I,J)	Al conjunto rango de la función: {((1,1),.15) ((1,2),.03) ((1,3),.02) ((1,4),.02) ((1,5),.70) ((1,6),.02) ((2,1),.04) ((2,2),.05) ((2,3),.04) ((2,4),.03) ((2,5),.75) ((2,6),.06) ((3,1),.02) ((3,2),.08) ((3,3),.01) ((3,4), 0) ((3,5),.80) ((3,6),.08) ((4,1),.04) ((4,2),.02) ((4,3),.02) ((4,4), 0) ((4,5),.75) ((4,6),.12) ((5,1),.02) ((5,2),.06) ((5,3),.02) ((5,4),.01) ((5,5),.80) ((5,6),.02) ((6,1),.01) ((6,2),.01) ((6,3), 0)

```
((6,4), 0 ) ((6,5),.97) ((6,6), 0 )
((7,1),.03) ((7,2), 0 ) ((7,3),.01)
((7,4), 0 ) ((7,5), 0 ) ((7,6),.97)
}
```

En donde los dos primeros elementos de la tercia representan el dominio de la función, integrado por los valores del índice I (materia prima) y los valores del índice J (metal básico); el tercer elemento de la tercia es el rango de la función y representa el contenido unitario del metal J en la materia prima I.

En cada invocación a un parámetro o a una variable, el lenguaje permite definir espacios de aplicación particulares, que sólo podrán ser subconjuntos del espacio teórico o dominio de la función. Estas acotaciones se definen asignando a cada índice, el conjunto de valores que configuran el espacio deseado, para tal efecto, a continuación del índice, se indica el operador = (o <DIF>) y entre corchetes ({ y }), se enuncian los valores válidos para el índice. Este conjunto de valores, también se puede establecer enunciando dentro de paréntesis cuadrados ([y]) elementos que son consecutivos, indicando los límites inferior y superior del rango de valores. (Dentro de corchetes es válido utilizar paréntesis cuadrados para definir elementos consecutivos).

En las invocaciones a los parámetros y a las variables, si a un índice se le asigna un sólo valor (I=1), el sistema considera que se está eliminando una dimensión del espacio de aplicación del parámetro, por lo que a todo índice en esta situación se le llama un índice eliminado; esta operación es equivalente a sustituir una variable por una constante en una función multivariable, dando como resultado una función sobre el resto de las variables.

Si a un índice se le asigna un conjunto de un solo valor (I={1} se considera un conjunto de un elemento) o no se le asigna explícitamente ningún valor, (en cuyo caso el índice puede tomar todos los valores del dominio teórico), se dice que el índice es un índice libre, ya que puede tomar distintos valores.

Los conceptos de índice libre e índice eliminado serán utilizados en la definición de espacio equivalente, cuando se analicen expresiones y ecuaciones bien formuladas.

El espacio vigente del parámetro en una invocación dada es el conjunto de n-tuples generado por el producto cartesiano de los

conjuntos particulares de valores asignados a los índices en la invocación. Dado lo anterior, se entenderá fácilmente que cada invocación de un parámetro genera su propio espacio vigente y que éste será un subconjunto del dominio de la función.

Continuando con los parámetros del ejemplo 2, a continuación se presentan ejemplos de definiciones de diversos espacios de aplicación.

Notación Algebraica	Notación GEMOLI	Representa:
$COSTO.U_i$ para i = 1	COSTO.U(I=1)	A la constante .03 que es el costo unitario del "material de desperdicio 1" (I=1).
$COSTO.U_i$ para i = 6	COSTO.U(I=6)	A la constante .21 que es el costo unitario del aluminio puro (I=6).
$CONT.ME_{ij}$ para i = 1	CONT.ME(I=1;J)	Al rango de la función: {((1,1),.15) ((1,2),.03) ((1,3),.02) ((1,4),.02) ((1,5),.70) ((1,6),.02)} que representa el contenido del metal J para la materia prima "material de desperdicio 1" (I=1). En esta invocación el conjunto { (1,1), (1,2), (1,3), (1,4), (1,5), (1,6) } representa el espacio vigente.
$CONT.ME_{ij}$ para j = 6	CONT.ME(I;J=6)	Al rango de la función: {((1,6),.02) ((2,6),.06) ((3,6),.08) ((4,6),.12) ((5,6),.02) ((6,6),.01) ((7,6),.97) } que establece el contenido de silicón (J=6) para cada una de las materias primas (I=[1,7]) respectivamente. El espacio vigente para esta invocación es: { (1,6) (2,6) (3,6) (4,6) (5,6) (6,6) (7,6) }
$CONT.ME_{ij}$ para i = {4,5,6} y j = 6	CONT.ME(I={4,5,6};J=6) o CONT.ME(I=[4,6];J=6)	Al rango de la función: {((4,6),.12) ((5,6),.02) ((6,6),.01)} que representa al contenido de silicón en las materias primas: "Material de desperdicio 4" (I=4), "Material de desperdicio 5" (I=5) y "Aluminio" (I=6).

Notación Algebraica	Notación GEMOLI	Representa:
		El espacio vigente en esta invocación es el: { (4,6) (5,6) (6,6) }

Si en el ejemplo 2 (ejemplo que hemos estado analizando) se hubiese definido IP como un subíndice del índice I, con un dominio teórico igual al subconjunto { 2,4,5,6 }, se podrían utilizar las siguientes invocaciones:

Notación Algebraica	Notación GEMOLI	Representa:
$CONT.ME_{i,j}$ para i={2,4,5,6} j = 6	CONT.ME(IP;J=6) o CONT.ME(I={2,4,5,6};J=6) o CONT.ME(I={2,[4,6]};J=6)	Al rango de la función: {((2,6),.06) ((4,6),.12) ((5,6),.02) ((6,6),.01)}

En todos estos casos, el espacio vigente es el conjunto de parejas: { (2,6) (4,6) (5,6) (6,6) }.

Este mismo subconjunto se puede invocar por medio del operador <DIF>.

Notación Algebraica	Notación GEMOLI	Representa:
$CONT.ME_{i,j}$ para i diferente de { 1,3,7 } y j = 6	CONT.ME(I <DIF> {1,3,7};J=6)	Al rango de la función: {((2,6),.06) ((4,6),.12) ((5,6),.02) ((6,6),.01)}

I <DIF> { 1,3,7 } genera el conjunto resultante de restarle al dominio teórico del índice I, el conjunto especificado a continuación de <DIF>, así en este caso:

I <DIF> { 1,3,7 } = { 1,2,3,4,5,6,7 } - { 1,3,7 }
= { 2,4,5,6 }

El conjunto de valores de un parámetro, que interviene en el cálculo de una expresión específica, depende del espacio vigente definido para el parámetro en la expresión, y depende también del conjunto real de valores del parámetro que fueron leídos o calculados.

Cualquier parámetro puede no tener valores definidos (leídos y calculados) para ciertos n-tuples del espacio teórico de aplicación, en estos casos, el sistema los reconoce como valores inexistentes y por lo tanto, no son expandidos en el modelo.

2.2.4 RESTRICCIONES

GEMOLI considera cada restricción definida (postulado <RES>), como un conjunto de ecuaciones que modelan una característica o limitación del mundo físico, y a diferencia de las variables y los parámetros, las restricciones pueden no llevar índices asociados cuando la limitación se puede establecer con una sola ecuación.

El lenguaje utiliza tres postulados para definir las restricciones del modelo; con el postulado <RES> se identifican al nombre y a los índices asociados a cada restricción, así como su significado o descripción.

Los índices asociados al identificador de la restricción, además de etiquetar al conjunto de ecuaciones de ésta, definen el espacio teórico de aplicación de la restricción y por consecuencia, al espacio teórico de aplicación del conjunto de ecuaciones.

El postulado <DOM> sirve para definir un subespacio de aplicación para el conjunto de ecuaciones que le siguen inmediatamente a continuación; mediante <DOM>, se establecen los dominios de los índices que configuran el espacio deseado.

Si no es necesario subdividir el espacio teórico de una restricción en subespacios, no se requiere el postulado <DOM> y el sistema aplica al conjunto de ecuaciones, el espacio teórico de la restricción.

Con el postulado <EC> se formula el conjunto de ecuaciones que modelan la restricción. Como ya se ha indicado, las ecuaciones son formalmente ecuaciones lineales y desigualdades lineales entre funciones (parámetros y variables), cuyos elementos están definidos y acotados por un espacio de aplicación.

Todo conjunto de ecuaciones asociado a una restricción se puede subdividir en subconjuntos disjuntos. Esta característica del lenguaje GEMOLI es particularmente útil cuando una restricción asociada a una característica del mundo real, no se puede (o no se quiere) representar con una sola ecuación general; como suele suceder en los casos donde en el conjunto de ecuaciones existen igualdades y desigualdades, o en los casos donde se quiere hacer explícita y patente la participación de un sumando en un subconjunto del sistema de ecuaciones.

Si se analiza la restricción CMPM(J) del ejemplo 2, se observará que la restricción está asociada al contenido permitido (máximo o mínimo) de los diversos metales en la aleación de aluminio, y se notará que el contenido permitido de todos ellos menos dos es una cota superior, que para el aluminio es una cota inferior y que para el silicón existe una cota superior y una cota inferior.

En este ejemplo,la restricción CMPM(J) se subdividió en tres subconjuntos de ecuaciones:

$$\sum_i CONT.ME_{ij} * CARGA_i \leq CONTMP_j \quad \text{para toda } j < 5 \quad ,$$

$$\sum_i CONT.ME_{ij} * CARGA_i > CONTMP_j \quad \text{para } j = 5 \text{ (Aluminio)} \quad y$$

$$250 \geq \sum_i CONT.ME_{ij} * CARGA_i \leq CONTMP_j \quad \text{para } j = 6$$

cuya especificación en lenguaje GEMOLI es:

```
<RES> CMPM(J)"CONTENIDO MAXIMO/MINIMO PERMITIDO DEL METAL J"
<DOM> (J=[1,4])
<EC> SUM((I=[1,7]),CONT.ME(I;J) * CARGA(I)) <MEI> CONTMP(J)
<DOM> (J=5)
<EC> SUM((I=[1,7]),CONT.ME(I;J) * CARGA(I)) <MAI> CONTMP(J)
<DOM> (J=6)
<EC> SUM((I=[1,7]),CONT.ME(I;J) * CARGA(I)) <MEI> CONTMP(J); 50
```

Si se analiza la definición de la primera ecuación global:

```
<EC> SUM((I=[1,7]),CONT.ME(I;J) * CARGA(I)) <MEI> CONT.MP(J)
```

Se observará que representa al conjunto de ecuaciones:

A) $.15\ CARGA_1 + .04\ CARGA_2 + .02\ CARGA_3 + .04\ CARGA_4 + .05\ CARGA_5 + .01\ CARGA_6 + .03\ CARGA_7 \leq 60$

B) $.03\ CARGA_1 + .05\ CARGA_2 + .08\ CARGA_3 + .02\ CARGA_4 + .06\ CARGA_5 + .01\ CARGA_6 \leq 100$

C) $.02\ CARGA_1 + .04\ CARGA_2 + .01\ CARGA_3 + .02\ CARGA_4 + .02\ CARGA_5 \leq 40$

D) $.02\ CARGA_1 + .03\ CARGA_2 + .05\ CARGA_5 \leq 30$

Si se recuerda que el espacio de aplicación para esta ecuación global está definido para los valores J=[1,4], se verá claramente que la ecuación A corresponde al valor J=1; que la ecuación B corresponde al valor J=2, etc.

Se puede encontrar fácilmente para la ecuación A que:

$.15 = CONT.ME_{1,1}$ (I=1;J=1)
$.04 = CONT.ME_{2,1}$ (I=2;J=1)
$.02 = CONT.ME_{3,1}$ (I=3;J=1)
etc.

y que 60 es el valor de $CONT.MP_1$.

También se puede observar en el sistema de ecuaciones que:

100 es el valor de $CONT.MP_2$
40 es el valor de $CONT.MP_3$
30 es el valor de $CONT.MP_4$

Lo cual indica que en la ecuación A, sólo se usaron elementos del conjunto asociados con los valores de J=1 e I=[1,7] únicamente. En la ecuación B, sólo se usaron elementos del conjunto asociados con los valores J=2 e I=[1,7]; la ecuación C, está asociada con los valores J=3 e I={1,7} y la ecuación D, está asociada con J=4 e I=[1,7]. En las ecuaciones B, C y D, los sumandos que no aparecen son aquellos cuyo coeficiente CONT.ME(I;J) vale cero.

El lenguaje GEMOLI permite jerarquizar restricciones, subdordinando conceptualmente unas a otras. Esta subordinación se logra declarando una restricción como subrestricción de otra, mediante los postulados <RES> y <SC> (subconjunto de) simultáneamente. El sistema realiza como un proceso especial, la verificación de que el espacio teórico de aplicación definido para la subrestricción, sea un subconjunto del espacio teórico de la restricción padre; por todo lo demás, se procesa como cualesquier restricción.

2.2.5 OPERADORES ARITMETICOS DEL LENGUAJE

Dado que la notación de GEMOLI integra al álgebra lineal con el manejo de funciones con limitaciones en cuanto al tipo de dominio y rango permitido, así como al manejo explícito de los espacios de aplicación. A continuación se ejemplifican las características relevantes de los operadores que se manejan en este lenguaje.

Sean:

$\|P$ = { enteros positivos }

$\|R$ = { números reales }

Indices

$I = \{ i \varepsilon \| P \mid i \leq \text{Valor-máximo-I} \}$

$J = \{ j \varepsilon \| P \mid j \leq \text{Valor-máximo-J} \}$

$K = \{ k \varepsilon \| P \mid k \leq \text{Valor-máximo-k} \}$

Espacios Teóricos de Aplicación

$\text{Espacio-IJ} = I \times J$

$= \{ (i,j) \mid i \varepsilon I \text{ y } j \varepsilon J \}$

$\text{Espacio-IK} = I \times K$

$= \{ (i,k) \mid i \varepsilon I \text{ y } k \varepsilon K \}$

$\text{Espacio-JK} = J \times K$

$= \{ (j,k) \mid j \varepsilon J \text{ y } k \varepsilon K \}$

$\text{Espacio-IJK} = I \times J \times K$

$= \{ (i,j,k) \mid i \varepsilon I \text{ y } j \varepsilon J \text{ y } k \varepsilon K \}$

Funciones Continuas

$A(I) = \{ (i,\text{Valor-}A_i) \mid i \varepsilon I \text{ y } \text{Valor-}A_i \varepsilon \| R \}$

$B(I;J) = \{ ((i,j),\text{Valor-}B_{i,j}) \mid (i,j) \varepsilon \text{ Espacio-IJ y } \text{Valor-}B_{i,j} \varepsilon \| R \}$

$C(I;J) = \{ ((i,j),\text{Valor-}C_{i,j}) \mid (i,k) \varepsilon \text{ Espacio-IJ y } \text{Valor-}C_{i,j} \varepsilon \| R \}$

$D(I;K) = \{ ((i,k),\text{Valor-}D_{i,k}) \mid (i,k) \varepsilon \text{ Espacio-IK y } \text{Valor-}D_{i,k} \varepsilon \| R \}$

Funciones Discontinuas

$E(I;J) = \{ ((i,j),\text{Valor-}E_{i,j}) \mid (i,j) \varepsilon \text{ Espacio-IJ y } (i,j) \varepsilon \{ (1,1) (1,2) (2,2) (3,1) (3,3) \} \text{ y } \text{Valor-}E_{i,j} \varepsilon \| R \}$

Se define un operador binario Ω como:

1) $A(I) \ \Omega \text{ constante} = \{ (i,(\text{Valor-}A_i \ \Omega \text{ Constante})) \mid (i,\text{Valor-}A_i) \varepsilon A(I) \}$

2) $B(I=\{1\};J=\{2\})\ \Omega\ C(I:J)=$

$$\{\ ((i,j),(\text{Valor-}B_{i,2}\ \Omega\ \text{Valor-}C_{i,j}) \mid ((1,2),\ \text{Valor-}B_{i,2})\ \varepsilon\ B(I;J)\ y\ ((i,j),\ \text{Valor-}C_{i,j})\ \varepsilon\ C(I;J)\ \}$$

3) $A(I)\ \Omega\ B(I:J) = \{\ ((i,j),(\text{Valor-}A_i\ \Omega\ \text{Valor-}B_{i,j})) \mid (i,\text{Valor-}A_i)\ \varepsilon\ A(I)\ \ y\ ((i,j),\text{Valor-}B_{i,j})\ \varepsilon\ B(I;J)\ \}$

4) $B(I;J)\ \Omega\ C(I;J) = \{\ ((i,j),\ (\text{Valor-}B_{i,j}\ \Omega\ \text{Valor-}C_{i,j}) \mid ((i,j),\text{Valor-}B_{i,j})\ \varepsilon\ B(I;J)\ \ y\ ((i,j),\text{Valor-}C_{i,j})\ \varepsilon\ C(I;J)\ \}$

5) $B\ (I=\{2,3,4\}\ ;\ J=\{1,2,3\}\ )\ \Omega\ C(I;J)$
$= \{\ ((i,j),(\text{Valor-}B_{i,j}\ \Omega\ \text{Valor-}C_{i,j})) \mid ((i,j),\text{Valor-}B_{i,j})\ \varepsilon\ B(I;J)\ \ y\ ((i,j),\text{Valor-}C_{i,j})\ \varepsilon\ C(I;J)\ \ y\ i\ \varepsilon\ \{2,3,4\}\ \ y\ j\ \varepsilon\ \{1,2,3\}\ \}$

6) $B(I=\{1,2,3\};J)\ \Omega\ C(I=\{4,5\},J)$ = Conjunto Vacío.

7) $B(I;J)\ \Omega\ D(I;K) = \{\ ((i,j,k),(\text{Valor-}B_{i,j}\ \Omega\ \text{Valor-}D_{i,k})) \mid ((i,j),\ \text{Valor-}B_{i,j})\ \varepsilon\ B(I;J)\ \ y\ ((i,k),\ \text{Valor-}D_{i,k})\ \varepsilon\ D(I;K)\ \}$

8) $B(I;J)\ \Omega\ E(I;J) = \{\ ((i,j),(\text{Valor-}B_{i,j}\ \Omega\ \text{Valor-}E_{i,j})) \mid ((i,j),\text{Valor-}B_{i,j})\ \varepsilon\ B(I;J)\ \ y\ ((i,j),\text{Valor-}C_{i,j})\ \varepsilon\ E(I;J)\ \ \}$

$= \{\ ((i,j),(\text{Valor-}B_{i,j}\ \Omega\ \text{Valor-}E_{i,j}) \mid \text{Valor-}B_{i,j}\ \varepsilon\ B(I;J)\ \ y\ \text{Valor-}E_{i,j}\ \varepsilon\ E(I;J)\ \ \text{para}\ (i,j)\ \varepsilon\ \{\ (1,1)\ (1,2)\ (3,1)\ (3,3)\}\}$

Como una notación especial, que se utiliza para eliminar índices libres, se define:

9) $B(I=1;J=2)\ \Omega\ C(I;J) = B(I;J)\ \Omega\ C(I=1;J=2)$
$= \text{Constante}$
$= \text{Valor-}B_{1,2}\ \Omega\ \text{Valor-}C_{1,2}$

10) $B(I=1;J)\ \Omega\ C(I;J) = \{\ (j,(\text{Valor-}B_{1,j}\ \Omega\ \text{Valor-}C_{1,j}) \mid ((1,J),\ \text{Valor-}B_{1,j})\ \varepsilon\ B(I;J)\ y\ ((1,J),\ \text{Valor-}C_{1,j})\ \varepsilon\ C(I;J)\ \ \}$

En todas estas expresiones se puede observar que todo operador sobre estas funciones genera un conjunto de valores reales definido para un espacio de aplicación especifico, generado por el producto cartesiano del conjunto de valores de los índices libres involucrados en la expresión.

Esta característica es equivalente a las siguientes expresiones:

$$h(x,y,z) = f(x) \Omega g(y,z)$$
$$h(x,y,z) = f(x,y) \Omega g(y,z)$$
$$h(x,y,z) = f(x,y,z) \Omega g(x,y,z)$$

en las que se puede observar que toda suma, resta, multiplicación, etc., de dos funciones con distintas variables independientes, genera una nueva función cuyas variables independientes son la combinación de las variables de ambas funciones.

Las expresiones 9) y 10) son equivalentes a:

$$h(x,y) = f(x) \Omega g(y,z)$$

cuando en la función g(y,z) se sustituye z por una constante.

Como segunda propiedad se puede mencionar que estos operadores no son conmutativos, dado que el espacio resultante es un producto cartesiano.

Por la misma razón, cuando uno de los dos operadores (o ambos) no está definido, no se incluye ningún elemento en el conjunto resultante.

Cuando una expresión da como resultado una constante (porque se han eliminado todos los índices involucrados) y alguno de los dos operandos no existe, al resultado de la expresión, por definición, se le asigna el valor 0.

Los operadores aritméticos definidos en el lenguaje son:

+	para la suma aritmética
-	para la resta aritmética
*	para la multiplicación
/	para la división
^	para la exponenciación ($A^B = A^B$)

Todos ellos, se calculan en artimética de punto flotante, aún cuando ambos operandos se hallan declarado enteros. Sin embargo, cuando el resultado de una expresión se asigna a un parámetro entero, el lenguaje permite definir si el valor entero se obtiene mediante un truncamiento o un redondeo.

Además de los operadores mencionados, el lenguaje cuenta con la función SUM ((índices-a-sumarizarse), expresión-de-parámetros). En índices a sumarizarse, se indican los índices que controlarán el proceso de sumarización, éstos pueden ser desde 1 hasta 6, y a cada índice se le debe definir un dominio particular, utilizando los corchetes y los paréntesis cuadrados, en la misma forma que la utilizada en las invocaciones a parámetros y variables.

El proceso de sumarización genera un espacio de aplicación con menos índices que la expresión que se está sumarizando, los índices que controlan este proceso son los eliminados.

Como resultado de la sumatoria se puede tener una constante, si todos los índices libres de la expresión fueron definidos en los índices a sumarizarse.

Si se analiza la función objetivo del ejemplo 1, podremos comprobar que el resultado de la expresión

SUM ((I=[1,2];J=[1,3]),COSTO.U(I;J) * EMBARQ(I;J))

es una constante generada por el siguiente conjunto de sumandos:

$$14 * EMBARQ_{1,1} + 10 * EMBARQ_{1,2} + 12 * EMBARQ_{1,3} +$$

$$+ 15 * EMBARQ_{2,1} + 13 * EMBARQ_{2,2} + 8 * EMBARQ_{2,3}$$

Si la expresión en GEMOLI hubiese sido:

SUM((I=[1,2,];J=[2,3])COSTO.U(I;J) * EMBARQ(I;J))

El resultado seguiría siendo una constante, pero su valor estaría determinado por un conjunto distinto de sumandos; a saber:

$$10 * EMBARQ_{1,2} + 12 * EMBARQ_{1,3}$$

$$+ 13 * EMBARQ_{2,2} + 8 * EMBARQ_{2,3}$$

Este mismo resultado se obtiene con las siguientes expresiones alternativas:

SUM((I=[1,2];J={2,3}),COSTO.U(I;J) * EMBARQ(I;J))

SUM((I={1,2};J={2,3}),COSTO.U(I;J) * EMBARQ(I;J))

Finalmente, cabe mencionar que los valores asignados a los índices a sumarizarse, pueden ser valores no consecutivos, e inclusive, puede asignarse un sólo valor, así:

SUM((I=1;J{2,3}),COSTO.U(I;J) * EMBARQ(I;J))

representa

$$10 * EMBARQ_{1,2} + 12 * EMBARQ_{1,3}$$

2.2.6 EXPRESIONES DE PARAMETRO BIEN FORMULADAS

En el lenguaje GEMOLI, se puede utilizar indistintamente la invocación a un parámetro, o a una expresión sobre parámetros,

siempre y cuando la expresión esté comprendida entre paréntesis "(" y ")" y esté bien formada.

Una expresión de parámetros bien formada implica que:

- como operandos se utilicen constantes y parámetros,
- cuando menos exista un parámetro en la expresión,
- como operadores se utilicen los operadores aritméticos
- y se cumplan las reglas de expansión siguientes (ver apéndice A):

«expresión-parámetros» ::=

SUM ((dominio-sumatoria), «expresión-parámetros») |

(«expresión-parámetros»)

«expresión-parámetros» «operador-aritmético» «expresión-parámetros»)

«parámetro») |

«constante)

«parámetro» ::= «identificador-parámetro» («dominio-parámetro»)

«dominio-parámetro» ::= «dominio-tl»

Tanto el conjunto resultante de la expresión, como el espacio de aplicación de la misma, se generan aplicando cada operador de la expresión, en la secuencia establecida por las siguientes prioridades:

Prioridad	Operador	
1	( )	Agrupamiento de operaciones mediante paréntesis
2	^	Exponenciación
3	* y /	Multiplicación y división
4	+ y -	Suma y resta

2.2.7 CONJUNTOS DE N-TUPLES

Como ya se ha mencionado anteriormente, el control de los elementos de una función (variable, parámetro o restricción) que participan en una expresión, se realiza mediante la definición del espacio de aplicación, vía la invocación de la función; con este método, el no asignar un valor a un índice, implica la eliminación de un conjunto de elementos, todos ellos integrantes del plano (del espacio de aplicación) asociado con el valor no asignado al índice; por lo tanto, mediante la definición del espacio de aplicación, no es posible detallar elementos particulares del espacio.

El lenguaje GEMOLI permite mediante el postulado <CONJ> (conjuntos de n-tuples) definir conjuntos de combinaciones particulares de índices y sus valores válidos o inválidos, así, mediante conjuntos se pueden manejar excepciones tales como: para toda i > 5; para toda i,j tal que i <> j; etc.

Con base en estos conjuntos, se puede condicionar para cada sumando del sistema de ecuaciones (y/o desigualdades) del modelo, la participación de ciertos elementos de la función (variable/parámetro) invocada.

Existen varias formas para definir un conjunto, la más simple es enunciando los valores de las n-tuples válidas, como es el caso de los conjuntos BUQ_LT y BUQ_PT del ejemplo 6; la segunda manera de definir conjuntos es leyendo los valores que conforman los n-tuples válidos; finalmente, un conjunto se puede definir mediante una expresión de conjuntos, en el ejemplo 6, en los conjuntos TVV, OR_D_DT, Y B_D_L, se utiliza este método.

Nótese que en todos los casos hasta ahora citados, los conjuntos fueron definidos mediante el postulado <CONJ>, el sistema GEMOLI permite también la definición de conjuntos mediante expresiones enunciadas directamente en las ecuaciones de las restricciones, tal es el caso del conjunto (ORIGEN <DIF> DESTINO) en la restricción TC del ejemplo 6.

Si en el ejemplo 6 se observan las definiciones de los conjuntos, se encontrará que:

- El conjunto OR_D_DT -ORIGEN DIFERENTE DE DESTINO- representa al conjunto de parejas (origen,destino) que cumplen con la propiedad de que el puerto de origen es diferente al puerto de destino.

- El conjunto OR_I_DT -ORIGEN IGUAL A DESTINO- es el conjunto de pareja (origen,destino) tales que el puerto de origen es igual que el puerto destino.

- El conjunto BUQ_LT es el conjunto de parejas (BUQUE,LOTE) que representan los tamaños de lote que puede manejar cada BUQUE.

- El conjunto BUQ_PT es el conjunto de parejas (BUQUE,PUERTO) que representan los puertos en los que puede atracar el buque.

Como ejemplos de la utilización de estos conjuntos, se puede analizar:

- En la función objetivo del propio ejemplo 6, se observa que el costo de la variable RTR(BUQUE;ORIGEN;DESTINO;PER_FIN) (ruta recorrida por el buque) varía si la pareja de índices (ORIGEN,DESTINO) pertenece al conjunto OR_D_DT -puerto de

origen <> puerto destino- o al conjunto OR_I_DT -puerto de origen = puerto destino-; en el primer caso el costo asociado es el costo de la ruta navegada; en el segundo caso, el costo se refiere al costo de mantener al buque en el muelle.

Si se analiza la restricción TC -TIEMPO DE CARGA EN REFINERIAS-, del mismo ejemplo 6, se verá que los elementos del primer sumando de la ecuación que se sumarán, serán aquellos cuyos índices ORIGEN y DESTINO tienen valores diferentes.

En el mismo ejemplo, en la restricción LLD, se puede observar que se restarán sólo aquellos elementos de DSC(BUQUE;DESTINO; PER_FIN) cuya tríada de identificación sea elemento del conjunto B_D_L.

Finalmente, y para cubrir los aspectos más importantes sobre conjuntos, cabe mencionar que los conjuntos de n-tuples se pueden utilizar como funciones de índices o reglas de transformación de índices como se explica en la sección 2.2.8.

2.2.8 EXPRESIONES DE CONJUNTOS

La forma básica para definir un conjunto mediante una expresión es que ésta sea una expresión de relación, en donde el conjunto queda definido sobre todos los índices que participan en la expresión, y los n-tuples, elementos del conjunto resultante, se integran con los valores de los índices, para los cuales se cumplió con la expresión de relación. Los operandos de relación son:

<IG>	para igual que,
<DIF>	para diferente de,
<MA>	para mayor que,
<MAI>	para mayor e igual que,
<ME>	para menor que y
<MEI>	para menor e igual que.

La forma para generar un conjunto en base a la combinación de otros conjuntos es mediante una expresión de unión, intersección y/o resta de conjuntos (operadores <O> <Y> <MENOS> respectivamente).

Tanto los operadores de relación como los operadores sobre conjuntos, al igual que los operadores numéricos, generan un espacio de aplicación que es el producto cartesiano del conjunto de valores de los índices libres involucrados en la expresión.

Las características generales de los operadores de relación se ejemplifican a continuación.

Sean:

$\|P$ = { enteros positivos }

$\|R$ = { números reales }

Indices

$I = \{ i \in \|P \mid i \leq 5 \}$

$J = \{ j \in \|P \mid j \leq 4 \}$

$K = \{ k \in \|P \mid k \leq 3 \}$

Espacios Teóricos de Aplicación

Espacio-IJ = I x J

$= \{ (i,j) \mid i \in I \text{ y } j \in J \}$

Espacio-JK = J x K

$= \{ (j,k) \mid j \in J \text{ y } k \in K \}$

Espacio-IK = I x K

$= \{ (i,k) \mid i \in I \text{ y } k \in K \}$

Espacio-IJK = I x J x K

$= \{ (i,j,k) \mid i \in I \text{ y } j \in J \text{ y } k \in K \}$

Parámetros

A(I) = { (1,10) (2,20) (3,25)
(4,15) (5,12) }

$= \{ (i, \text{Valor-}A_i) \mid i \in I \text{ y Valor-}A_i \in \|R \}$

B(I,J) = { ((1,1),15) ((1,2),16) ((1,3),12)
((1,4),17)
((2,1),18) ((2,2),19) ((2,3),20)
((2,4),15)
((3,1),21) ((3,2),22) ((3,3),18)
((3,4),12)
((4,1),22) ((4,2),25) ((4,3),27)
((4,4),40)
((5,1),12) ((5,2),15) ((5,3),24)
((5,4),30) ((5,5),27) }

$= \{ ((i,j), \text{Valor-}B_{i,j}) \mid (i,j) \in \text{Espacio-IJ y Valor-}B_{i,j} \in \|R \}$

C(K) = { (1,2) (2,4) (3,6) }
= { (i, Valor-C_k) | i ε I y
Valor-C_k ε ||R }

Como ejemplos de la formación de conjuntos a través de expresiones se tiene:

EXPRESION C O N J U N T O

(i <MA> 3) = { i ε I | i > 3 }

= { 4,5 }

(i <IG> j) = { (i,j) | (i,j) ε Espacio-IJ y i = j }

= { (1,1) (2,2) (3,3) (4,4) }

((i-j) <MAI> 3) = { (i,j) | (i,j) ε Espacio-IJ y (i-j) ≥ 3 }

= { (4,1) (5,1) (5,2) }

(i <MA> C(K)) = { (i,k) | (i,k) ε Espacio-IK y
i > Valor-C_k }

= { (3,1) (4,1) (5,1) (5,2) }

Nótese que i=3, i=4 e i=5 son todos mayores que C(1)=2 por lo que las parejas (i=3,k=1), (i=4,k=1) e (i=5,k=1) son elementos del conjunto resultante, así mismo, i=5 es mayor que C(2)=4 por lo que (i=5,k=2) es también elemento del conjunto resultante.

(B(I;J) <MEI> A(I)) = { (i,j) | (i,j) ε Espacio-IJ
y Valor-$B_{i,j}$ ≤ Valor-A_i }

= { (1,3) (2,1) (2,2) (2,3) (3,1)
(3,2) (3,3) (3,4) (5,1) }

(A(I) <MA> C(K)) = { (i,k) | (i,k) ε Espacio-IK
y Valor-A_i > Valor-C_k }

= { (1,1) (1,2) (1,3)
(2,1) (2,2) (2,3)
(3,1) (3,2) (3,3)
(4,1) (4,2) (4,3)
(5,1) (5,2) (5,3) }

En todos estos ejemplos se puede observar, que el conjunto

resultante de una expresión de relación está definido para un espacio de aplicación que incluye a todos los índices libres que intervienen en la expresión, ya sea que aparezcan en la expresión como operandos directamente o como índices de un parámetro invocado en la expresión.

Los n-tuples que integran los elementos del conjunto, son todas aquellas combinaciones de valores que corresponden respectivamente a los índices del espacio de aplicación, para las cuales, se cumple la expresión de relación.

Los operadores de unión, intersección y resta de conjuntos generan un espacio de aplicación de la misma forma que todos los operadores ya descritos.

El conjunto generado por una unión de conjuntos contendrá como elementos aquellas combinaciones de valores de índices, para los cuales existe el elemento asociado en cualesquiera de los dos conjuntos.

El conjunto generado por el operador intersección, contendrá como elementos aquella combinación de valores de índices para los cuales existe un elemento asociado en cada uno de los conjuntos operandos.

El conjunto resultante de una resta la integran aquellas combinaciones de índices para las cuales existe el elemento asociado en el conjunto invocado como primer operando y no existe el elemento asociado en el conjunto invocado como segundo operando. Continuando con las definiciones y ejemplos recién establecidos, se tiene:

EXPRESION	C O N J U N T O
((i <IG> j) <Y> (i > 3))	= { (i,j) | (i,j) ε Espacio-IJ y (i = j) y (i > 3) }
	= { (4,4) }
((i <IG> j) <O> (i > 3))	= { (i,j) | (i,j) ε Espacio-IJ y ((i = j) o (i > 3)) }
	= { (1,1) (2,2) (3,3) (4,1) (4,2) (4,3) (4,4) (5,1) (5,2) (5,3) (5,4) }
	en este caso la regla (i > 3) es la que mayor número de elementos aporta al conjunto resultante.

((i <IG> j) <MENOS> (i > 3))= { (i,j) | (i,j) ε Espacio-IJ y
(i,j) ε { (i,j) | i = j } y
(i,j) ∉ { (i,j) | i > 3 } }

= { (1,1) (2,2) (3,3) }

2.2.9 EXPRESIONES DE CONJUNTOS BIEN FORMULADAS

Una expresión sobre conjuntos bien formulada implica que:

- como operandos se utilicen constantes, índices y/o parámetros.
- cuando menos exista un conjunto en la expresión, o una expresión de relación, que incluya un índice o un parámetro como operando y cualquier operador de relación.
- y que se cumplan las siguientes reglas de expansión.

«exp-conjunto» ::=

(«exp-conjunto») |

«exp-conjunto» «operador-conjunto» «exp-conjunto» |

«exp-relación» |

«conjunto»

«exp-relación» ::=

«exp-aritmética» «operador-relación» «exp-aritmética»

«exp-aritmética» ::=

(«exp-aritmética») |

«exp-aritmética» «operador-aritmético» «exp-aritmética»

SUM((«dominio-sumatoria»), «exp-aritmética») |

«parámetro» |

«índice» |

«constante» |

«operador-conjunto» ::= <Y> | <O> | <MENOS> |

«operador-relación» ::= <IG> | <DIF> | <MA> | <MAI> | <ME> | <MEI>

«operador-aritmético» ::= (|) | ^ | / | * | + | -

«conjunto» ::= «identificador-conjunto» («dominio-conjunto»)

«parámetro» ::= «identificador-parámetro» («dominio-parámetro»)

«dominio-conjunto» ::= «dominio-teórico»

«dominio-parámetro» ::= «dominio-tl»

Tanto el conjunto resultante, como el espacio de aplicación de la expresión, se generan aplicando cada operador de la expresión, en la secuencia establecida por las siguientes prioridades.

Prioridad	
1	paréntesis de agrupamiento
2	operadores aritméticos
3	operadores de relación
4	operadores de conjunto

2.2.10 ECUACIONES Y DESIGUALDADES BIEN FORMULADAS

Como ya se mencionó en la sección 2.2.4, el sistema de ecuaciones/desigualdades asociado a una restricción debe ser consistente con el espacio teórico de aplicación definido por los índices asociados a la restricción.

Esta consistencia se debe dar para cada uno de los elementos de las ecuaciones: así, cada sumando de la ecuación/desigualdad debe tener un espacio de aplicación equivalente con el espacio teórico de aplicación definido para la restricción; lo mismo se aplica para el límite de la ecuación/desigualdad ("right hand side"), en donde cualquier parámetro definido como límite de la restricción deberá tener un espacio de aplicación consistente con el de la restricción.

La equivalencia entre espacios de aplicación se da, si el conjunto de índices libres que conforman el espacio de aplicación del elemento de la ecuación, es el mismo que el conjunto de índices que definió el espacio de aplicación para la restricción.

En los espacios equivalentes importan los índices libres que conforman los espacios de aplicación, sin importar el orden de los mismos, asimismo, los subíndices de un índice son tratados como si fueran exactamente este último.

En el ejemplo 1 del apéndice B se puede observar que la restricción COSTO, que genera un espacio de aplicación vacío, es equivalente al único sumando de la ecuación:

SUM((I = [1,2] ; J = [1,3]), COSTO.U(I;J) * EMBARQ(I;J))

ya que COSTO.U(I;J) * EMBARQ(I;J) generan un espacio de aplicación igual a I x J, que al sumarizarse sobre I y sobre J, generan como resultado de la sumarización un espacio de aplicación vacío.

En la restricción EXTF(I) del mismo ejemplo se observará que el espacio de aplicación es I, que el límite de la ecuación es el parámetro EXTNC(I), definido sobre el mismo espacio de aplicación I, y que el único sumando SUM((J=[1,3]),EMBARQ(I;J)) genera un espacio de aplicación sobre I también.

En el ejemplo 6 del mismo apéndice B se puede observar que la restricción COSTO tiene un espacio de aplicación vacío por ser la función objetivo, asimismo, si se analiza el espacio de aplicación generado para el sumando que representa el costo de otros medios, se notará que:

COM(DESTINO) * OMD(DESTINO;PER_FIN)

genera un espacio de aplicación igual a DESTINO x PER_FIN, que al sumarizarse sobre los índices DESTINO y PER_FIN el resultado es un espacio de aplicación vacío, mismo que es consistente con el espacio definido para la restricción COSTO.

En el mismo ejemplo 6 se puede observar que la restricción VIU(BUQUE;PER_FIN) genera un espacio de aplicación igual a BUQUE x PER_FIN y que el único sumando de la restricción:

SUM((ORIGEN=[1,4]; DESTINO=[1,4]), RTR(BUQUE; ORIGEN ; DESTINO; PER_FIN))

genera el mismo espacio al sumarizarse sobre los índices ORIGEN y DESTINO.

Cuando se está utilizando un conjunto de índices válidos para restringir el espacio de aplicación corriente de un sumando, se presentan dos posibilidades de espacio equivalente para el sumando en cuestión.

En el primer caso, el espacio de aplicación del conjunto es un subespacio del espacio de aplicación del sumando, en esta situación, el espacio equivalente está determinado por el espacio de aplicación del sumando.

Ejemplos de estos casos, extraídos del ejemplo 6, son:

SUM((BUQUE=[1,2];ORIGEN=[1,4];DESTINO=[1,4];PER_FIN=[1,6]),
CTR(BUQUE;ORIGEN;DESTINO)+RTR(BUQUE;ORIGEN;DESTINO;PER_FIN)
<ELE>OR_D_DT(ORIGEN;DESTINO))

que es un sumando de la restricción COSTO, en donde sólo se sumarizarán aquellos productos CTR(BUQUE;ORIGEN;DESTINO) * RTR(BUQUE;ORIGEN;DESTINO; PER_FIN) que sean elementos del conjunto OR_D_DT(ORIGEN ; DESTINO).

El espacio de aplicación de esta multiplicación es igual al producto cartesiano de BUQUE x ORIGEN x DESTINO x PER_FIN ya que el conjunto al que están asociados, esta definido para un espacio de aplicación que es un subespacio de éste.

Como otro ejemplo del mismo caso se puede citar:

```
-SUM((LOTE=[1,4]),
    DSC(BUQUE;DESTINO;PER_FIN;LOTE)
    <ELE>B_D_L(BUQUE;DESTINO;LOTE))
```

de la restricción LLD(BUQUE,DESTINO,PER_FIN), en donde sólo se sumarán aquellos elementos del conjunto DSC(BUQUE,DESTINO, PER_FIN;LOTE) cuyas tercias (BUQUE,DESTINO,LOTE) existan en el conjunto B_D_L.

En este caso, el espacio de aplicación del sumando es el conjunto BUQUE x DESTINO x PER_FIN ya que el índice LOTE fue eliminado por la sumarización.

El segundo caso se presenta cuando el conjunto asociado a un sumando de la ecuación contiene uno o varios índices no contenidos en el sumando, en este caso se considera al conjunto de índices válidos como una función de transferencia; tal es el caso del elemento

```
-SUM((DESTINO=[1,4];PER_FIN=[1,6]),
 RTR(BUQUE;ORIGEN=PUERTO;DESTINO;PER_FIN)
<ELE> TVV(ORIGEN;DESTINO;PER_IN=PERIODO;PER_FIN))
```

de la restricción MV(BUQUE,PUERTO,PERIODO), en donde con ORIGEN, DESTINO y PER_IN se encuentran los valores válidos para PER_FIN, con lo cual quedan perfectamente definidos los elementos del conjunto RTR(BUQUE;ORIGEN;DESTINO;PER_FIN) que se deben sumarizar para cada tercia (BUQUE,PUERTO,PERIODO) del espacio de la restricción.

En estos casos, ciertos índices se utilizan para definir los valores válidos de otros, con lo cual queda perfectamente definida la participación de los elementos del conjunto sumando, en cada elemento del conjunto de ecuaciones de la restricción.

Si al conjunto de índices que definen el espacio teórico de la restricción se le denomina el conjunto R; y si al conjunto de índices libres que definen el espacio de aplicación del sumando se le llama el conjunto S; y si al conjunto de índices libres que definen el espacio de aplicación de los n-tuples válido se

le denomina el conjunto C; se cumple siempre que el espacio de aplicación del sumando es consistente (no existen ambigüedades) con el espacio de aplicación de la restricción si se cumplen las tres condiciones siguientes:

1) $C \subseteq R \cup S$
2) $R \subseteq C \cup S$
3) $S \subseteq C \cup R$

2.3 GENERACION DE HILERAS Y COLUMNAS PARA LA MATRIZ MPSX

Como hileras de la matriz para el simplex, el sistema integra las restricciones definidas en el modelo, para los cuales exista su valor límite (valor del "right hand side").

Si para una restricción se restringió el dominio teórico mediante el postulado <DOM>, los elementos del conjunto integrados como hileras serán aquellos que estén contenidos en el dominio restringido y para los cuales se haya definido el valor límite de la ecuación.

El conjunto real de elementos de cada variable que el sistema integra como columnas de la matriz del simplex, está determinado por el espacio teórico definido para la variable y por la condición necesaria que obliga la existencia en el sistema total de ecuaciones, de cuando menos un coeficiente explícitamente declarado para cada elemento del conjunto.

Estas condiciones permiten definir los modelos con ecuaciones sobre funciones, con lo cual se representan los aspectos generales de las restricciones, sin perder el control de la participación individual de cada elemento.

Así, si un elemento del conjunto de una variable no tiene asociado ningún coeficiente en ninguna restricción o una ecuación/desigualdad de una restricción no tiene un valor límite, es evidente que el n-tuple que identifica al elemento, no tiene un significado lógico en el modelo, y por lo tanto no debe expandirse a la salida.

Como un ejemplo general de este caso se puede mencionar que en las redes, los coeficientes asociados a tramos definidos por la pareja de nodos <origen, destino> son generalmente inexistentes para valores de origen y destino iguales.

2.4 REFERENCIAS

[1] Dorfman, Robert; Samuelson, Paul A.; Solow, Robert M.
Linear Programming and Economic Analysis
McGraw-Hill
New York, N.Y.
1958
págs. 39-40, 64-65

[2] Ibídem
págs. 499-501

[3] Dantzing, George B.
Linear Programming and Extensions
Princeton, N.J.
1963
págs. 60-62, 573-574

[4] Ibídem
págs. 195-202

[5] Intriligator, Michael D.
Mathematical Optimization and Economic Theory
Prentice-Hall Inc.
Englewood Cliffs, N.J.
1971
págs. 8-12

[6] Taha, Hamdy A.
Operations Research
An Introduction
McMillan Publishing Co., Inc.
New York, N.Y.
1971
págs. 24-30

[7] Ibídem
págs. 226-228

[8] Simonnard, M.
Programación Lineal
Paraninfo
Madrid
1972
págs. 27-28, 291-292, 378-380, 408-409

[9] Ibídem
Págs. 29-35, 294-295, 459

[10] Prawda, Juan
Métodos y Modelos de Investigación de Operaciones
Vol. I: Modelos Determinísticos
Limusa
México
1976
págs. 37-55

[11] Ibídem
págs. 67-68

[12] Mills, Gordon
Optimization in Economic Analysis
George Allen & Unwin (Publishers) Ltd.
London
1984
págs. 18-20

[13] Pike, Ralph W.; Guerra G., Lautaro
Optimización en Ingeniería
Alfaomega
México
1989
págs. 50-53

[14] Ibídem
págs. 73-88

3 DEFINICION DETALLADA DEL LENGUAJE

Es importante mencionar que como complemento referencial a este capítulo, en el apéndice A se presenta la sintaxis general del lenguaje, en donde se describen la visión general y el máximo detalle de todas las alternativas de expresión permitidas para cada postulado.

3.1 DESCRIPCION FUNCIONAL DE LOS POSTULADOS DEL LENGUAJE

El lenguaje GEMOLI está constituido por un conjunto de postulados, cada uno de ellos especializado en la definición de uno de los distintos elementos que constituyen este tipo de modelos.

POSTULADO	SIRVE PARA DEFINIR:
<MOD>	El nombre con el que se reconocerá el modelo, los diferentes archivos asociados al mismo y el paquete SIMPLEX que se utilizará para la solución del problema.
<IND>	Los índices y subíndices manejados en el modelo, sus respectivos dominios teóricos, el significado general (<SG>) de los mismos, así como el significado particular (<SP>) asociado a cada valor que toma el índice o subíndice.
<VAR>	Las variables del modelo, cuyos valores en la solución óptima se desean conocer, su significado general, el tipo de variable (real, entera o binaria) y las fronteras a las que se deben sujetar sus valores. También se establece la asociación entre el conjunto de cada variable (nombre de la variable) con los índices que definen y distinguen a los individuos pertenecientes al mismo.
<PARAM>	Los parámetros del modelo. Se define el nombre genérico del conjunto, los índices asociados y el significado general del parámetro. Los valores de los parámetros se definen mediante los postulados <VALOR> o <DATO>.
<CONJ>	Los conjuntos válidos de n-tuples. Se definen los nombres de conjuntos, su significado general y los índices que los constituyen.

POSTULADO	SIRVE PARA DEFINIR:
<RES>	Las familias de restricciones. Se define el nombre del conjunto, los índices asociados a él y el significado general de la familia.
<DOM>	Subespacios del espacio teórico de aplicación de la restricción. Este postulado es opcional, si no está presente, el espacio de aplicación de la restricción lo establece todo el rango de valores que pueden tomar los índices asociados a la restricción.
<EC>	Las ecuaciones de la restricción. Se definen las variables que se suman/restan, el límite de la restricción y la relación de magnitud (<IG>, <MAI>, <MEI>, etc.) entre la suma de variables y el valor límite de la restricción.
<ARCH_E>	Archivos de donde se leerán algunos datos para el modelo. Se define el nombre del archivo, el número máximo de campos que contiene y la secuencia del archivo.
<DATO>	Define los parámetros o vectores frontera, cuyos valores serán leídos del archivo, así como la forma en que se deberán buscar en el mismo.
<CONS>	Las reglas de consistencia. Permite definir para los parámetros y las variables, las relaciones que deben cumplir sus valores, para asegurar la consistencia del modelo.
<VALOR>	Establece el valor de los parámetros y/o de los valores frontera de las variables. Para un parámetro dado, el valor del mismo se puede establecer mediante uno de los dos postulados <DATO> o <VALOR>, pero no por ambos.
<FIN>	Indica el fin de la definición del modelo y debe ser el último postulado.
$	Sirve para intercalar comentarios en cualquier punto de la definición del modelo. Todo texto a continuación del signo de $ y hasta el fin de la línea, se considerará un comentario.

3.2 ESTRUCTURA DE LA DEFINICION DEL MODELO

Como complemento a esta descripción se debe consultar la expansión del término <<MODELO>> en la sintaxis del lenguaje (apéndice A).

a) En primer lugar, mediante el postulado <NOM> se define el nombre del modelo y del paquete que se usará para resolverlo.

b) A continuación se definen los índices y subíndices del modelo.

Se define en primera instancia el índice, con el postulado <IND>. En forma optativa se pueden definir los significados asociados a los valores del índice, mediante la opción <SP> (significado particular).

Opcionalmente se pueden definir subíndices del índice definido, en cuyo caso y a continuación de éste se definirán todos sus subíndices mediante el postulado <IND> y la opción <SC> (subconjunto).

c) Una vez definidos todos los índices y sus respectivos subíndices, se definen todas las variables del modelo mediante el postulado <VAR>.

d) Una vez definidas todas las variables, se procede a definir todos los parámetros del modelo. Esta definición se realiza mediante el postulado <PARAM>.

e) A continuación de la definición de parámetros, si se requieren se deben definir los conjuntos de n-tuples válidos del modelo, para lo cual se utiliza el postulado <CONJ>.

f) La definición de la función objetivo es el siguiente paso. Esta función se define como una restricción por lo que se usan los postulados <RES> y <EC>.

g) Una vez definida la función objetivo se especifican todas las restricciones y sus respectivas subrestricciones.

Cada conjunto de restricciones estará compuesta de un encabezado de restricción, definido mediante el postulado <RES>.

Si se requiere, el espacio de aplicación de las restricciones se puede dividir en subespacios disjuntos mediante el postulado <DOM>.

Para cada subespacio definido, a continuación del mismo, debe ir el postulado <EC> con la ecuación de la restricción que se aplicará en el subespacio de aplicación especificado.

Si la restricción no se subdivide en diversos subespacios bastará el postulado de <RES> y el postulado <EC>.

El proceso de definición se repite hasta cubrir todas las restricciones del modelo.

h) A continuación de las restricciones, se definen todos los archivos de entrada que se usarán en la integración del modelo.

Por cada archivo de entrada se requiere un postulado <ARCH_E> y un postulado <SEC>, a continuación se requieren tantos postulados <DATO> como atributos diferentes se obtendrán del archivo.

Todo parámetro que vaya a ser leído del archivo requiere de un postulado <DATO>. Con el postulado <DATO> también se puede establecer la lectura de archivos para los valores frontera de las variables, los valores de los índices de un conjunto, así como los significados particulares asociados a los valores de los índices del modelo.

Si el modelo no requiere de archivos de entrada, los postulados <ARCH_E> y <DATO> se deben omitir.

i) Si se requiere, a continuación de los archivos de entrada se deberán definir las reglas de consistencia mediante el postulado <CONS>. Cada regla de consistencia requiere un postulado <CONS>.

j) A continuación se definen los valores de los parámetros mediante el postulado <VALOR>, requiriéndose un postulado <VALOR> por cada parámetro que se desee definir.

Con el postulado <VALOR> se pueden establecer al igual que con <DATO> los valores para la frontera de las variables.

k) Finalmente se declara el fin de la definición del modelo mediante el postulado <FIN>.

NOTAS IMPORTANTES

Para un parámetro dado, el valor del mismo se puede establecer mediante uno de los dos postulados <DATO> o <VALOR> pero no por ambos.

Para que un modelo sea expansible, se requiere que las variables válidas tengan, como coeficiente, cuando menos un parámetro explícitamente definido con <VALOR> o con <ARCH> y <DATO>, por lo que, todo modelo deberá contar con cuando menos un postulado de <VALOR> o de un postulado <ARCH> y un postulado <DATO>.

3.3 POSTULADO <MOD> (nombre del modelo)

El modelo se inicia con el postulado <MOD>, en él se define el nombre del modelo y el nombre del paquete que se usará para

resolver el modelo. Aquí mismo se pueden definir opcionalmente los nombres para los vectores: objetivo, restricciones, rangos y fronteras, con los cuales se almacenarán estos vectores y podrán ser recuperados por los paquetes: MPSX, APEX o TEMPO.

FORMATO:

<MOD> Palabra reservada que se debe escribir tal cual se presenta.

Nombre-del-modelo

El nombre del modelo debe respetar las reglas del MS-DOS para nombrar archivos, además, el nombre no debe incluir ningún sufijo ni referencias a directorios. Con el nombre del modelo indicado y diferentes sufijos asignados automáticamente por el sistema, se identificarán los archivos que constituyen el modelo.

<PAQ> Palabra reservada que se debe escribir tal cual se presenta.

Nombre-del-paquete

Nombre del paquete de resolución que se utilizará, las opciones válidas son:

MPSX, IBM, APEX, TEMPO, LP88, MILP88, XA

En el apéndice B se pueden observar los diferentes archivos generados por el sistema, dependiendo de la definición del paquete que se realice; así en la sección B.1.5 se describe el archivo correspondiente al paquete MILP88, la sección B.2.5 contiene el archivo de entrada para el paquete XA, la sección B.3.5 contiene el archivo generado cuando se define como paquete IBM y finalmente, en la sección B.4.5 se presenta el archivo correspondiente al paquete MPSX.

Las siguientes parejas de elementos del postulado son opcionales:

<FO> Palabra reservada que se debe escribir tal cual se presenta.

Identificador-vector-objetivo

Nombre con el que se desea almacenar el vector de coeficientes de la función objetivo. Si se omite, el sistema automáticamente genera OBJETIVO.

<RS> Palabra reservada que se debe escribir tal cual se presenta.

Identificador-vector-restricciones

Nombre con el que se desea almacenar los límites de las restricciones del modelo. Si se omite, el sistema automáticamente genera RESTRCS.

<RG> Palabra reservada que se debe escribir tal cual se presenta.

Identificador-vector-rangos

Nombre con el cual se almacenará el vector de rangos de las restricciones. Si se omite, el sistema automáticamente genera RANGOS.

<FR> Palabra reservada que se debe escribir tal cual se presenta.

Identificador-vector-fronteras

Nombre con el cual se almacenarán los valores de frontera superior e inferior de las variables del modelo. Si se omite, el sistema automáticamente genera FRONTERS.

Los identificadores de los vectores descritos deberán ser nombres que inicien con una letra y no deben ser mayores de ocho caracteres.

Estos identificadores son utilizados para la creación del programa que controla el proceso del MPSX, particularmente en los postulados "MOVE" de estos programas (ver apartado B.3.5 del apéndice B).

EJEMPLOS:

En los apartados del apéndice B referidos a continuación, se pueden observar las siguientes definiciones.

B.1.3
<MOD> TRANSP
<PAQ> MILP88

B.2.3
<MOD> MEZCLA
<PAQ> XA

B.3.3
<MOD> FLUJO_MX
<PAQ> IBM

B.4.3
<MOD> ASIGN
<PAQ> XA

```
B.5.3       <MOD> VIAJE
            <PAQ> XA

B.6.3       <MOD> COMBUS65
            <PAQ> IBM
            <RS>  RESTRCS
            <RG>  RANGOS
            <FR>  FRONTERS
```

3.4 POSTULADO <IND> (definición de índices y subíndices)

Mediante este postulado se definen los índices y subíndices utilizados en el modelo.

Este postulado está constituido por tres secciones o partes:

-La definición del índice y su significado general <IND>.
-La definición de los significados particulares <SP>.
-La definición del índice como un subíndice o subconjunto de otro índice <SC>.

3.4.1 NOMBRE Y SIGNIFICADO GENERAL DEL INDICE <IND>

Esta sección del postulado es obligatoria, en ella se define el nombre del índice o subíndice y el dominio teórico del mismo.

El dominio teórico del índice es el conjunto de valores que puede tomar el índice, que por definición irá de 1 al dominio-máximo indicado en este postulado.

El dominio corregido de un subíndice se define en la sección <SC>.

FORMATO:

<IND> Palabra reservada que debe escribirse tal cual se presenta.

Identificador-índice o Identificador-subíndice

Nombre del índice o subíndice que se está definiendo.

El nombre debe iniciar con letra y no debe ser mayor de 7 caracteres, debe estar constituido de letras y números y como caracteres especiales sólo se aceptan (.) punto y (_) subrayado.

,"Significado-General"

En esta parte se describe el significado del índice.

El significado-general puede ser cualquier leyenda delimitada por comillas y no mayor a 250 caracteres. No se aceptan ni (") ni ($) como parte de la leyenda.

,Dominio-máximo

Se indica el valor máximo que puede tomar el índice o subíndice.

EJEMPLOS:

En los apartados del apéndice B referidos a continuación, se pueden observar las siguientes definiciones:

B.1.3) <IND>I,"FABRICA",2

En el ejemplo 1, la I representa la fábrica de donde pueden salir los embarques de envolturas.

Los valores válidos que puede tomar I son 1 y 2 que representan las dos fábricas.

<IND>J,"DETALLISTA",3

En el mismo ejemplo, la J representa a los detallistas, a donde irán los embarques, el dominio teórico definido para J es 1, 2 y 3 únicamente.

B.2.3) <IND>I,"LOTES DE CARGA",7

En el ejemplo 2, la I representa los diversos lotes que pueden ser carga para la aleación de aluminio.

Existen 7 lotes, por lo que el dominio teórico del índice se define como 1,2,3,4,5,6 y 7.

<IND>J,"METALES BASICOS",6

En el mismo ejemplo, la J representa los 6 diferentes metales básicos que interesan.

3.4.2 SIGNIFICADO PARTICULAR <SP>

A continuación del significado general del índice, se permiten establecer los significados particulares correspondientes a los valores que puede tomar el índice.

Esta sección del postulado no es obligatoria y su propósito es documental. Esta sección no se permite para subíndices, ya que los significados de los valores de un subíndice serán iguales a los significados de los valores del índice padre.

FORMATO:

<SP>{ Símbolos reservados, se escriben tal cual se presentan.

Valor-del-índice

El valor del índice cuyo significado particular se pretende detallar.

, "Significado-particular"

El significado particular asociado al valor del índice establecido en valor-del-índice.

El significado-particular puede ser cualquier leyenda delimitada por comillas y no mayor a 250 caracteres. No se aceptan ni (") ni ($) como parte de la leyenda.

; Se pone punto y coma para indicar que siguen más valores.

Entre cada par valor-significado debe ir como separador un ; .

} Este símbolo da por terminada la definición de los significados particulares.

EJEMPLOS:

En los apartados del apéndice B referidos a continuación, se pueden observar las siguientes definiciones:

B.1.3) <SP>{1,"FABRICA EN TOLUCA";2,"FABRICA EN QUERETARO"}

En el ejemplo 1 se está asociando al valor de I=1 el significado FABRICA EN TOLUCA y a I=2 el significado FABRICA EN QUERETARO.

```
<SP>{1,"ENVOLTURAS ELEGANTES";
     2,"PAQUETERIA FINA";
     3,"REGALOS DISTINGUIDOS"}
```

En el mismo ejemplo se está asociando a los valores de: J=1 el significado de ENVOLTURAS ELEGANTES; a J=2 el significado de PAQUETERIA FINA y a J=3 el significado de REGALOS DISTINGUIDOS.

```
B.2.3)  <SP>{1 "LOTE 1";2 "LOTE 2";3 "LOTE 3";4 "LOTE 4";
         5 "LOTE 5";6 "LOTE 6";7 "LOTE 7"}
```

En el ejemplo 2 se están asociando los nombres LOTE N a los valores N del índice I.

```
<SP>{1 "FIERRO"; 2 "COBRE"; 3 "MANGANESO";
     4 "MAGNESIO"; 5 "ALUMINIO"; 6 "SILICON"}
```

En el mismo ejemplo 2 se están asociando los nombres de los metales básicos a los valores de J que los representan.

3.4.3 INDICE SUBCONJUNTO DE OTRO INDICE <SC>

Esta sección del postulado <IND> sirve para definir un índice como un subconjunto de otro. Esto quiere decir que el dominio del subíndice será un subconjunto del dominio del índice padre. Antes de esta sección, debe ir el postulado <IND>, donde se define el nombre del subindice y su significado general.

En esta sección se define la relación padre-hijo y el dominio particular del subíndice.

El dominio particular que se defina para el subíndice, deberá ser un subconjunto del dominio del índice padre.

Un subíndice puede ser declarado como un subconjunto de otro subíndice.

FORMATO:

<SC> Palabra reservada que se debe escribir tal cual.

Identificador-índice-padre

Nombre del índice padre del cual es un subconjunto el subíndice que se está definiendo.

Formación-dominio-subíndice

Existen varias alternativas para definir el dominio del subíndice, a continuación se enuncian todas ellas.

En todas estas alternativas, cuando se solicita un índice-equivalente se está refiriendo indistintamente a un índice o a un subíndice.

Alternativa 1

<INC> {lista-valores-dominio}

El dominio se define por enumeración de todos los elementos del subdominio.

Alternativa 2

<EXC> {lista-valores-dominio no pertenecientes al subconjunto}

El dominio se define restando al conjunto padre los elementos enunciados que no pertenecen al subconjunto.

Alternativa 3

<INC> <IGD> Identificador-índice-equivalente

El dominio se define por referencia a un índice con un dominio equivalente, donde los valores del dominio del índice equivalente serán el dominio del subíndice que estamos definiendo.

Alternativa 4

<EXC> <IGD> Identificador-índice-equivalente

El dominio se define por referencia a un índice cuyo dominio restado al dominio del índice padre, nos da el dominio deseado.

Alternativa 5

<INC> <IGD> Identificador-índice-equivalente +
{lista-valores-dominio pertenecientes al dominio del subíndice}

Se define el dominio del subíndice como el conjunto resultante de sumarle al dominio de un índice (índice-equivalente) un conjunto explícito de valores.

Alternativa 6

<INC> <IGD> Identificador-índice-equivalente -
{lista-valores-dominio no pertenecientes al dominio del subíndice}

Se define el dominio del subíndice como el conjunto resultante de restarle al dominio del índice equivalente, un conjunto explícito de valores.

Alternativa 7

<EXC> <IGD> Identificador-índice-equivalente +
{lista-valores-dominio pertenecientes al dominio del subíndice}

Se define el dominio del subíndice, como el conjunto resultante de restarle al dominio del padre la suma del dominio del índice equivalente y un conjunto explícito de valores.

Alternativa 8

<EXC> <IGD> Identificador-índice-equivalente -
{lista-valores-dominio no pertenecientes al dominio del subíndice}

Se define el dominio del subíndice como el conjunto resultante de restarle al dominio del padre el dominio del índice equivalente al cual se le restó previamente el conjunto explícito de valores.

La lista-valores-dominio es un conjunto de números separados por coma. En esta lista de valores se permiten definir rangos consecutivos, tales como [1,4] que define al conjunto de valores 1,2,3,4.

EJEMPLOS:

Para los ejemplos a continuación, se debe suponer que con anterioridad se definieron:

Un índice I con el siguiente dominio: 1,2,3,4,5,6,7,8

Un subíndice JP con el siguiente dominio: 4,5

a) <IND> IP,"subíndice de I",4
<SC> I <INC>{3,4}

Se está definiendo el índice IP como un subíndice de I, el dominio de IP es: 3,4

b) <IND> IP,"subíndice de I",5
<SC> I <EXC>{3,[6,8]}

Se está definiendo al índice IP como un subíndice de I con un dominio = 1,2,4,5

c) <IND> IP,"subíndice de I",5
<SC> I <INC><IGD> JP

Se está definiendo como dominio de IP el conjunto 4,5

d) <IND> IP,"IP subíndice de I",8
<SC> I <EXC><IGD> JP

Se está definiendo como dominio de IP el conjunto: 1,2,3,6,7,8

e) <IND> IP,"IP subíndice de I",7
<SC> I <INC><IGD> JP + {2,7}

Se está definiendo como dominio de IP el conjunto 2,4,5,7

```
f)  <IND>     IP,"IP subíndice de I",4
        <SC> I <INC><IGD> JP - {5}
```

Se está definiendo como dominio de IP el valor 4 únicamente

```
g)  <IND>     IP,"IP subíndice de I",8
        <SC> I <EXC><IGD> JP + {6}
```

Se está definiendo como dominio de IP el conjunto 1,2,3,7,8

```
h)  <IND>     IP,"IP subíndice de I",8
        <SC> I <EXC><IGD> JP - {5}
```

Se está definiendo como dominio de IP el subconjunto 1,2,3,5,6,7,8

```
i)  <IND>     IP,"IP subíndice de I",7
        <SC> I <INC> {1,2,[4,7]}

    <IND>     IPP,"IPP es subíndice de IP",4
        <SC> IP, <EXC>{5,6,7}
```

Se está definiendo a IP como un subíndice de I, con un dominio =1,2,4,5,6,7

También se está definiendo a IPP como un subíndice de IP; IPP tiene como dominio: 1,2,4

En el apartado B.6.3 del apéndice B, se pueden observar las siguientes definiciones:

```
B.6.3)      <IND> PUERTO,"PUERTO",4
            <IND> ORIGEN,"PUERTO ORIGEN",4
                  <SC> PUERTO <INC> {[1,4]}
            <IND> DESTINO,"PUERTO DESTINO",4
                  <SC> PUERTO <IGD> ORIGEN
```

En donde se están definiendo los índices ORIGEN y DESTINO, como subíndices de PUERTO, asimismo se define que ambos tienen un dominio igual al conjunto {1,2,3,4,}.

3.5 POSTULADO <VAR> (definición de conjuntos de variables)

Mediante este postulado se definen los conjuntos de variables, los índices asociados a los mismos, así como la descripción del significado general de cada variable.

Por la naturaleza misma del lenguaje, no se aceptan conjuntos de variables sin índices asociados.

La secuencia en que se definen los índices de las variables se deberá respetar en todas las referencias posteriores que se hagan a las mismas.

FORMATO:

<VAR> Palabra reservada, se debe escribir tal cual.

<ENT> o <BIN> o <REAL>

Opcionalmente las variables se pueden declarar como variables enteras, variables binarias, o variables reales.

Las variables se declaran enteras incluyendo la palabra reservada <ENT>; las variables se declaran binarias si se incluye la palabra reservada <BIN>.

Si ninguna de estas dos opciones se indican, o explícitamente se usa la palabra reservada <REAL> se está declarando una variable real.

Para aquellas variables declaradas como enteras, cuando el paquete indicado en <NOM> no acepta este tipo de variables, el sistema las interpreta y maneja automáticamente como reales.

Identificador-variable

Nombre del conjunto variable que se está definiendo.

El nombre debe iniciar y terminar con letra y no debe ser mayor de siete caracteres, debe estar constituido de letras y números y como caracteres especiales sólo se aceptan (.) punto y (_) subrayado.

(dominio-teórico)

El dominio-teórico o espacio teórico de aplicación es la lista ordenada de los índices (y/o subíndices) asociados al conjunto variable, éstos deben ir entre paréntesis y separados por coma. La secuencia de definición de estos índices es importante y se debe respetar cada vez que se haga referencia al conjunto variable.

,"Significado-general"

El sentido o significado general de la variable.

El significado general puede ser cualquier leyenda delimitada por comillas y no mayor a 250 caracteres. No se aceptan ni (") ni ($) como parte de la leyenda.

,<FRON_INF> Identificador-vector-frontera-interior

Opcionalmente se puede declarar el nombre del vector donde se encontrarán los valores de los límites inferiores asociados a cada variable del conjunto, para lo cual es necesario indicar <FRON_INF> precediendo al nombre del vector.

,<FRON_SUP> Identificador-vector-frontera-superior

Al igual que las fronteras inferiores, la definición de las fronteras superiores es opcional; cuando se declara, se utiliza el prefijo <FRON_SUP> seguido del nombre del vector donde se definen los valores de los límites superiores.

Para una misma variable, se puede definir un tipo de frontera (inferior o superior), ambos tipos, o ninguna restricción de límite. Si se definen ambos tipos de frontera, se deberá definir en primer lugar la frontera inferior.

EJEMPLOS:

En los apartados del apéndice B referidos a continuación se pueden observar las siguientes definiciones.

B.1.3) <VAR> EMBARQ(I,J),"EMBARQUE DE LA FABRICA I AL DETALLISTA J"

B.2.3) <VAR> CARGA(I),"CARGA DEL LOTE I", <FRON_INF> FINF, <FRON_SUP> FSUP

B.4.3) <VAR> PRO.A (I.J),"COMPAÑIA I ASIGNADA AL PROYECTO J"

B.5.3) <VAR> TRAMO(I,J,S),"TRAMO RECORRIDO AL MOMENTO S"

B.6.3) <VAR> <BIN> RTR(BUQUE,ORIGEN,DESTINO,PER_FIN),"RUTA RECORRIDA"
<VAR> <BIN> DSC(BUQUE,DESTINO,PER_FIN,LOTE), "DESCARGA"

3.6 POSTULADO <PARAM> (definición de parámetros)

Mediante este postulado se definen los conjuntos parámetro del modelo.

Cuando en una restricción dada, se requieran diferentes coeficientes para un mismo conjunto variable, entonces se requiere declarar un conjunto parámetro para poder definir los diferentes valores requeridos.

Mediante parámetros se pueden definir los valores límites de las restricciones. ("RIGHT HAND SIDE").

Todo parámetro puede ser calculado en base a otros parámetros mediante la evaluación de una expresión.

La secuencia en que se definen los índices de los parámetros se deberá respetar a lo largo de la definición del modelo.

FORMATO:

<PARAM> Nombre del conjunto parámetro.

El nombre debe iniciar con letra y no debe ser mayor de siete caracteres y debe estar constituido de letras, números y como caracteres especiales sólo se aceptan (.) punto y (_) subrayado.

<ENT> Opcionalmente se puede definir que un parámetro se debe manejar en forma entera, esta definición requiere de la inclusión de la palabra reservada <ENT>.

Que un parámetro se maneje como un entero implica: que si el parámetro se calcula, se lee de un archivo, o se establecen sus valores mediante el postulado <VALOR>, el sistema no usará los decimales aún en el caso de que los valores definidos para el parámetro los incluyan.

<REDONDEADO>

Si opcionalmente se indica que un parámetro es entero y redondeado, se está definiendo: que el parámetro si trae decimales en sus valores o, que si el parámetro se calcula, se deberán obtener los valores enteros del parámetro mediante el redondeo.

La opción de <REDONDEADO> sólo es válida para parámetros enteros.

(dominio-teórico)

El dominio teórico o espacio teórico de aplicación del parámetro es la lista ordenada de índices (y/o subíndices) asociados al conjunto parámetro, éstos deben ir entre paréntesis y separados por comas, la secuencia de definición es importante y se debe respetar cada vez que se haga referencia al parámetro.

,"Significado-general"

Significado general del conjunto parámetro.

El significado general puede ser cualquier leyenda delimitada por comillas y no mayor a 250 caracteres. No se aceptan ni (") ni ($) como parte de la leyenda.

Hasta este punto llega la definición de aquellos parámetros cuyos valores se establecerán explícitamente mediante los postulados <VALOR> o <DATO>. Si el parámetro se calculara en base a otros parámetros, a continuación del "Significado-general" se deberá incluir, entre paréntesis y precedida del símbolo =, la expresión que define este cálculo.

= (expresión-parámetro)

La formulación correcta de expresiones de parámetros se describió en la sección 2.2.6. Sin embargo, es necesario en este punto resaltar que: el espacio de aplicación generado por la expresión de parámetros deberá ser equivalente al espacio teórico del parámetro que se está definiendo.

EJEMPLOS:

En los apartados del apéndice B referidos a continuación se pueden observar las siguientes definiciones:

B.1.3) <PARAM> COSTO.U(I,J),"COSTO UNITARIO DE EMBARQUE DE LA FABRICA I AL DETALLISTA J"

B.2.3) <PARAM> COSTO.U(I),"COSTO UNITARIO DEL LOTE I"

<PARAM> CONT.ME(I,J),"CONTENIDO DEL METAL J EN EL LOTE I"

B.3.3) <PARAM> FICT(ORG,DST),"OBLIGA LA DEFINICION EXPLICITA DEL COEFICIENTE"

B.4.3) <PARAM> COSTO.U(I,J),"COSTO DEL PROYECTO J EN LA COMPAÑIA I"

B.5.3) <PARAM> DIST(I,J),"DISTANCIA DE I A J"

B.6.3) <PARAM> <ENT> <REDONDEADO> DURV(ORIGEN,DESTINO), "DURACION VIAJE" = (DIST(ORIGEN;DESTINO) / 864 + TDSC(DESTINO))

<PARAM> FICT1(PUERTO,PER_FIN), "OBLIGA LA DEFINICION EXPLICITA DEL COEFICIENTE"

El parámetro FICT del ejemplo 3 se definió para que la variable FLUJO(ORG,DST) tuviera un coeficiente explícitamente definido; con la definición de este parámetro y los valores que se le asociaron en el postulado <VALOR>, se está definiendo el conjunto válido de variables para la familia FLUJO(ORG,DST).

Asimismo el parámetro FICT1 del ejemplo 6 está definiendo el conjunto válido de variables para EX_P(PUERTO,PER_FIN).

Es importante hacer notar en el cálculo del parámetro DURV(ORIGEN,DESTINO) que el espacio de aplicación de la expresión calculada es idéntico al espacio de aplicación del parámetro.

3.7 POSTULADO <CONJ> (definición de conjuntos)

Mediante este postulado se definen los conjuntos de n-tuples válidos. Este postulado no es obligatorio, sólo se usará si existe un conjunto que definir.

Como ya fue mencionado, los conjuntos en el lenguaje GEMOLI sirven para asociar un grupo de índices a un conjunto válido de combinaciones particulares de valores que pueden tomar estos índices.

FORMATO:

<CONJ> Palabra reservada, se debe escribir tal cual.

Identificador-conjunto

Nombre del conjunto de n-tuples que se está definiendo. El nombre debe iniciar con letra y no debe ser mayor de siete caracteres, como caracteres subsecuentes se pueden utilizar letras, números y como caracteres especiales se aceptan (.) punto y (_) subrayado.

(dominio-teórico)

El dominio teórico o espacio teórico de aplicación del conjunto es la lista ordenada de los índices (y/o subíndices) asociados al conjunto, éstos deben ir entre paréntesis y separados por comas. La secuencia de definición de estos índices es importante y se debe respetar cada vez que se haga referencia al conjunto.

,"significado-general"

El sentido o significado general del conjunto.
El significado general puede ser cualquier leyenda delimitada por comillas y no mayor a 250 caracteres. No se aceptan como parte de la leyenda ni (") ni ($).

Formación-del-conjunto

Los n-tuples elementos del conjunto se pueden definir de siete formas, dependiendo si se definen en base al

conjunto válido o si se definen como el complemento de un conjunto inválido, en combinación con las tres formas básicas de definir estos conjuntos (válido/inválido) que son: enunciando explícitamente el conjunto de n-tuples, indicando la entrada de datos por archivo o definiendo una expresión de conjunto donde se establecen las reglas de formación del mismo.

Alternativa 1

= { lista-de-n-tuples }

Mediante esta alternativa se enlistan los n-tuples que conforman el conjunto de valores válidos. El separador entre cada tuple es (;) punto y coma y los valores asociados a los índices que conforman cada tuple deben separarse por coma.

Alternativa 2

= nombre-vector-tuples

Mediante esta alternativa se está definiendo que el conjunto de n-tuples deberá ser leído de disco, y el nombre del vector de los tuples aquí definido deberá aparecer además en un postulado <DATO>, en donde se definen los campos del archivo que contienen los valores de los índices.

Alternativa 3

= (expresión-conjuntos)

Como se ha mencionado en la sección 2.2.8, la regla básica de formación de un conjunto es una expresión de relación formada por un parámetro o un índice comparado, mediante un operador de relación, con otro parámetro, índice o constante. Todos los casos donde la relación se cumple, los valores de los índices involucrados en la expresión conforman el conjunto de n-tuples válido.

El otro tipo de expresiones sobre conjuntos lo constituyen la unión o intersección de conjuntos.

La expresión de conjuntos debe estar comprendida entre paréntesis, su espacio de aplicación resultante debe ser equivalente al espacio de aplicación del conjunto que se está definiendo, y finalmente debe ser una expresión bien formulada (ver sección 2.2.9).

Para las alternativas que se describen a continuación, en donde

se maneja el complemento (<COMPL>) de conjuntos se deberá entender que:

<COMPL> A(I,J) = U - A(I,J)

en donde U es el conjunto formado por el producto cartesiano de los conjuntos de valores teóricos que pueden tomar todos los índices del conjunto A (ver espacio teórico de aplicación en la sección 2.2), así:

```
Si: I =      { 1,2,3,4 }
    J =      { 1,2,3 }
    A(I,J) = { (1,1) (1,3)
               (2,1) (2,2)
               (3,2) (3,3)
               (4,1) (4,3) }

    U(I,J) = { (1,1) (1,2) (1,3)
               (2,1) (2,2) (2,3)
               (3,1) (3,2) (3,3)
               (4,1) (4,2) (4,3) }
```

entonces:

<COMPL> A(I,J) = { (1,2) (2,3) (3,1) (4,2) }

Alternativa 4

<COMPL> Conjunto-base (dominio-teórico-conjunto-base).

Mediante esta alternativa, el conjunto se está definiendo como el complemento del conjunto-base indicado.

El dominio-teórico o espacio teórico de aplicación del conjunto que se está definiendo, deberá ser el mismo que el dominio teórico del conjunto-base invocado.

Alternativa 5

<COMPL> { lista-de-n-tuples }

Mediante esta alternativa se está definiendo el conjunto como el complemento de los n-tuples enlistados o inválidos.

La lista de n-tuples inválidos tiene como separador de tuple el (;) punto y coma y como separador de cada valor del tuple a la coma.

Alternativa 6

<COMPL> nombre-vector-tuples

Esta alternativa define el conjunto de n-tuples como el complemento del vector que se leerá de archivo.

El nombre-vector-tuples, aquí invocado deberá aparecer posteriormente en un postulado <DATO>, en donde se definirá, en qué campos del archivo se encuentran los valores de los índices que conforman el n-tuple.

Alternativa 7

<COMPL> (expresión-conjuntos)

Mediante esta alternativa se está definiendo un conjunto que es el complemento del conjunto resultante de la expresión entre paréntesis.

El espacio de aplicación resultante de la expresión debe ser equivalente al espacio de aplicación del conjunto que se está definiendo.

La expresión deberá cumplir con las reglas inherentes a expresiones bien formuladas (ver sección 2.2.9).

EJEMPLOS:

En el apartado B.6.2 del apéndice B, se describen las definiciones que se enuncian a continuación y que pertenecen al apartado B.6.3.

```
<CONJ> BUQ_LT(BUQUE,LOTE),"TAMAÑO LOTE VALIDO/BUQUE"
       = { 1,1; 1,2; 1,3; 1,4; 2,1; 2,2 }

<CONJ> BUQ_PT(BUQUE,DESTINO),"PUERTOS VALIDOS/BUQUE"
       = { 1,1; 1,3; 1,4; 2,1; 2,2; 2,3; 2,4 }

<CONJ> B_D_L(BUQUE,DESTINO,LOTE),"TRIPLE VALIDO"
       =(BUQUE_PT(BUQUE;DESTINO) <Y> BUQUE_LT(BUQUE;LOTE))

<CONJ> OR_D_DT,"ORIGEN <DIF> DESTINO"
       =(ORIGEN <DIF> DESTINO)

<CONJ> OR_I_DT,"ORIGEN IGUAL DESTINO"
       <COMPL> OR_D_DT(ORIGEN;DESTINO)

<CONJ> TVV(ORIGEN,DESTINO,PER_IN,PER_FIN),"TIEMPOS DE VIAJE
                                           VALIDOS"
       = ( ((PER_FIN-PER_IN) <IG> DURV(ORIGEN;DESTINO)) <Y>
                                  (ORIGEN <DIF> 4) )
```

3.8 POSTULADO <RES> (definición de restricciones y subrestricciones)

Mediante este postulado se define una familia de restricciones, los índices asociados a la misma y el sentido general de la familia.

El postulado <RES> representa la primera parte, obligatoria siempre, de la definición del conjunto restricción. El postulado <DOM> ó <EC> deben seguir a continuación.

En una familia se deben agrupar restricciones que conceptualmente representen la misma condición ó limitación. Así, deberá haber un conjunto para los balances de material, otro conjunto para los balances de energía; un mismo tipo de limitación física, como puede ser el flujo máximo en cada tramo de un ducto, debe constituir un conjunto restricción.

Un conjunto restricción no debe llevar índices asociados si la familia está formada por una sola condición, tal es el caso de la función objetivo que siempre es única en todo modelo.

FORMATO:

<RES> Palabras reservadas que deben escribirse tal como se presentan.

Nombre-de-la-restricción

Nombre de la restricción que se está definiendo.

El nombre debe iniciar y terminar con letra, no debe ser mayor de siete caracteres, debe estar constituido de letras y números, y como caracteres especiales sólo se aceptan (.) punto y (_) subrayado.

(Lista-ordenada-de-índices)

El conjunto de índices asociados al conjunto restricción debe ir entre paréntesis y separado por coma (,). Esta lista ordenada de índices y los paréntesis que la contienen, no se deben incluir si la familia de restricciones está formada por una sola ecuación.

"Significado-general"

El sentido o significado general de la familia de restricciones.

El significado general puede ser cualquier leyenda delimitada por comillas y no mayor a 250 caracteres. No se aceptan ni (") ni ($) como parte de la leyenda.

EJEMPLOS:

2)<RES> COSTO "COSTO DE LA ALEACION"

En el ejemplo 2 (apéndice B) se define la función objetivo como una restricción formada por una sola ecuación por lo que no requiere de índices.

<RES> PROD.T,"TOTAL DE ALEACION PRODUCIDA"

En el mismo ejemplo 2, el total de aleación producida representa sólo una ecuación por lo que esta familia no requiere de índices.

<RES> CONT.MP(J) "CONTENIDO PERMITIDO DEL METAL J"

Finalmente, en el mismo ejemplo la restricción CONT.MP es una familia de restricciones que representa el contenido máximo permitido del metal J en la aleación; cada valor de J define un miembro de la familia y por lo tanto una ecuación para cada metal J.

3.9 POSTULADO <DOM> (definición de subdominios de la restricción)

Mediante este postulado se divide el espacio de aplicación del conjunto restricción en subespacios. Cada subespacio lleva su propia ecuación.

Este postulado es opcional, si no está presente, el espacio de aplicación para la ecuación, lo establece el espacio teórico de aplicación definido por los índices en el postulado <RES>.

Cuando se requiere subdividir el dominio de una familia de restricciones en subdominios, se tendrán que usar tantos postulados <DOM> como subdominios sean requeridos, debiendo respetar que todos los subdominios definidos sean conjuntos disjuntos entre sí y sean subconjuntos del dominio total de la familia.

La necesidad de dividir el dominio de una familia de restricciones en subdominios se presenta cuando la relación de magnitud (<IG>,<MAI>,<MEI>) de las ecuaciones de la familia no es única, esto es, cuando existen ecuaciones de la familia que requieren una relación, y también existen ecuaciones que requieren otra relación distinta.

Se puede subdividir el dominio de una familia en subdominios cuando en ciertos subdominios no intervienen algunos sumandos de la ecuación. En estos casos es preferible subdividir el dominio que declarar explícitamente parámetros con valor cero.

FORMATO:

<DOM> Palabra reservada que debe escribirse tal cual.

(subdominio-de-la-restricción)

El subdominio de la restricción o subespacio de aplicación se define asignando a todos y cada uno de los índices de la restricción, el conjunto de valores válido para el subdominio que se está definiendo.

El subdominio de la restricción debe ir entre paréntesis, cada índice con sus valores debe ir separado de los siguientes por el signo ;.

Los valores asignados a los índices deberán ser elementos del dominio teórico del índice.

Existen varias alternativas para definir el subdominio de cada índice, todas ellas se describen a continuación.

Existen dos formas generales para asignar valores a un índice: la forma enumerativa en donde el subdominio es una lista de valores separados por una coma (,); o por diferencia, donde el subdominio es el conjunto de valores resultante de restarle al dominio teórico del índice un conjunto de valores separados por una coma (,). Ambas formas básicas generan las siguientes alternativas.

Alternativa 1

Nombre-del-índice = valor

Se define al subdominio del índice como el valor especificado

Alternativa 2

Nombre-del-índice = [valor-inferior-del-rango,
valor-superior-del-rango]

Se define al subdominio del índice como el conjunto de valores comprendidos dentro del rango especificado, esto es, el conjunto de valores consecutivos que inician en el valor- inferior- del-rango y terminan en el valor-superior-del-rango.

Alternativa 3

Nombre-del-índice = {lista-de-valores del subdominio}

Se define el subdominio del índice mediante una lista-de-valores, esto es, un conjunto de números separados por una coma (,). En esta lista-de-valores se permite definir como un elemento, rangos consecutivos tales como [1,4] que define al conjunto de valores 1,2,3,4.

Alternativa 4

Nombre-del-índice <DIF> valor

Se define el subdominio del índice como el conjunto de valores resultante de restarle al dominio teórico del índice el valor especificado.

Alternativa 5

Nombre-del-índice <DIF> [valor-inferior-del-rango,
valor-superior-del-rango]

Se define el subdominio del índice como el conjunto de valores resultante de restarle al dominio teórico del índice, los valores comprendidos dentro del rango especificado, esto es, restarle al dominio teórico del índice el conjunto de valores consecutivos que incian en el valor-inferior-del-rango y terminan en el valor-superior-del-rango.

Alternativa 6

Nombre-del-índice <DIF> {lista-de-valores no contenidos en el dominio}

Se define el subdominio del índice como el conjunto de valores resultante, de restarle al dominio teórico del índice, el conjunto de valores definidos mediante la lista-de-valores no contenidos en el dominio, esto es, un conjunto de números separados por una coma (,). (En esta lista de valores se permite definir rangos consecutivos tal como [1,4] que define el conjunto de valores 1,2,3,4).

EJEMPLOS:

2) <DOM> (J=[1,4])
<DOM> (J=5)
<DOM> (J=6)

En el ejemplo 2 (del apéndice B) el dominio de j del conjunto restricción CONT.MP, se divide en tres subconjuntos disjuntos entre sí, el primero para j=[1,4] contiene las restricciones que indican la cantidad máxima permitida de fierro, cobre, manganeso y magnesio; el segundo j=5 indica la restricción que limita la cantidad mínima permitida de aluminio; el último j=6, define la restricción y el rango para el silicón.

3.10 POSTULADO <EC> (definición de ecuaciones)

Este postulado se utiliza para definir las ecuaciones de las restricciones. En él se establecen los sumandos, la relación de magnitud (=, ≤, ≥), el valor límite de la restricción y opcionalmente el rango de éstas.

Este postulado sigue inmediatamente al postulado de <DOM> si éste existe.

Si el dominio o espacio de aplicación de la restricción no se tiene que subdividir, entonces este postulado debe seguir del postulado <RES> para restricciones y del postulado <SC> para las subrestricciones.

El espacio de aplicación para la ecuación será igual al dominio definido en el postulado inmediato anterior ya sea <DOM>, <RES> o <SC>.

FORMATO:

<EC> Palabra reservada que se debe escribir tal cual.

Sumandos-de-la-ecuación

A continuación de <EC> se deben definir los sumandos de la restricción. Puede haber más de un sumando en una ecuación, cada uno debe iniciar con el símbolo + cuando se deba sumar, o con el símbolo - cuando se deba restar.

Para el primer sumando puede omitirse el símbolo de + y se interpreta como si hubiese sido éste.

Las alternativas para definir los sumandos se discuten en la siguiente sección.

Relación-de-magnitud

Las relaciones de magnitud de las ecuaciones / desigualdades válidas son:

<MAX> implica MAXIMO

<MIN>	"	MINIMO
<IG>	"	=
<MA>	"	>
<MAI>	"	≥
<ME>	"	<
<MEI>	"	≤

Las relaciones <MAX> y <MIN> se permiten sólo en la función objetivo, misma que debe ser la primera restricción que se declare en el modelo.

Valor-límite-de-la-restricción

A continuación de la relación de magnitud se define el conjunto de valores límites de la restricción.

Si se declara una constante como valor límite de la restricción, todos los elementos del espacio de aplicación de la ecuación, tendrán como valor límite la constante definida.

Si se especifica un conjunto parámetro, el sistema buscará los valores asociados al parámetro en un postulado <VALOR> o en un postulado <DATO> y los utilizará como los valores límite del conjunto de ecuaciones.

Para que exista una equivalencia biunívoca entre el conjunto de valores definidos por el parámetro y el conjunto de ecuaciones, es necesario que el espacio teórico de aplicación del parámetro sea equivalente al espacio teórico de aplicación para las ecuaciones.

Como última alternativa, el valor-límite-de-la-restricción se puede definir mediante una expresión de parámetros que deberá ir entre paréntesis; en este último caso el sistema calculará los elementos de la expresión y los asociará al conjunto de ecuaciones definida, por lo que se requiere que el espacio de aplicación resultante del cálculo de la expresión sea equivalente al espacio de aplicación de las ecuaciones.

En el caso particular de la función objetivo, como valor-límite, invariablemente se pondrá el nombre de la función objetivo con lo cual se termina la definición de ésta.

;Valor-del-rango

El rango de una ecuación es opcional; si existe, se debe especificar a continuación del valor límite de la restricción un ; y a continuación el valor-del-rango.

Como valor-del-rango se puede definir: una constante,

un parámetro, o una expresión de parámetros delimitada por paréntesis. Las reglas y comentarios expuestas para el valor-límite-de-la-restricción se aplican de igual manera para la definición del valor-del-rango.

3.10.1 Sumandos de la ecuación

Los sumandos de toda ecuación están constituidos por tres elementos básicos: la variable que se suma o resta, el coeficiente de la variable y la condición que respecto a un conjunto de n-tuples deben cumplir los índices de la variable y el coeficiente, para que el sumando sea seleccionado como válido y sea considerado en la expansión de la ecuación (ver sección 2.2.7).

La variable es el único elemento obligatorio. Como coeficiente se puede definir una constante, un parámetro, o una expresión de parámetros; si no se declara un coeficiente explícitamente, el sistema automáticamente le asigna el valor de 1.

La condición de selección del sumando se utiliza cuando se quiere restringir el espacio de aplicación del sumando, tal como se describió en la sección 2.2.10.

La condición de selección del sumando está conformada por el prefijo <ELE> y un conjunto de n-tuples.

El conjunto de n-tuples puede ser la invocación a un conjunto ya definido, o una expresión de conjuntos delimitada entre paréntesis.

En resumen los sumandos de una ecuación se pueden definir mediante las siguientes reglas de expansión:

«sumandos» ::= + «sumando» | - «sumando»

«sumando» ::= SUM((dominio-sumatoria), «coeficiente-variable»)
«coeficiente-variable»

«coeficiente-variable» ::=
«coeficiente»? «variable» «condición-selección»?

«coeficiente» ::= («expresión-parámetros») | «parámetro» |
«constante»

«condición-selección» ::= <ELE> («exp-conjunto») | <ELE> «conjunto»

«variable» ::= «identificador-variable» («dominio-variable»)

«parámetro» ::= «identificador-parámetro» («dominio-parámetro»)

«conjunto» ::= «identificador-conjunto» («dominio-conjunto»)

«dominio-sumatoria» ::= «dominio-t2»

«dominio-variable» ::= «dominio-t1»

«dominio-parámetro» ::= «dominio-t1»

«dominio-conjunto» ::= «dominio-t1»

De las cuales, la expresión-parámetros ya fue definida en la sección 2.2.6, y la expresión-conjunto fue definida en la sección 2.2.8.

Los dominios de las variables, parámetros y conjuntos se describirán con todo detalle en el próximo apartado, en este lugar lo importante es resaltar el formato general de las ecuaciones.

EJEMPLOS:

En los apartados del apéndice B referidos a continuación se pueden observar las siguientes definiciones.

FUNCIONES OBJETIVO

B.1.3)
```
<RES> COSTO, "COSTO MINIMO"
<EC>  SUM ((I=[1,2]; J=[1,3]), COSTO.U(I;J)*EMBARQ(I;J)
      <MIN> COSTO
```

B.2.3)
```
<RES> COSTO, "COSTO DE LA ALEACION"
<EC>  SUM((I=[1,7]), COSTO.U(I) * CARGA (I))
      <MIN> COSTO
```

B.3.3)
```
<RES> FLU_OP, "FLUJO OPTIMO"
<EC>  SUM((DST=[1,6]), FLUJO (ORG=1;DST))
      <MAX> FLU_OP
```

B.4.3)
```
<RES> COSTO, "MINIMO COSTO DE LOS PROYECTOS"
<EC>  SUM((I=[1,4]; J=[1,3]), COSTO.U(I;J)*PRO.A(I;J))
      <MIN> COSTO
```

B.5.3)
```
<RES> DIST.REC, "DISTANCIA RECORRIDA"
<EC>  SUM((I=[1,4];J=[1,4];S=[1,4]),
         DIST(I;J)*TRAMO(I;J;S))
      <MIN> DIST.REC
```

```
B.6.3)    <RES> COSTO, "COSTO DE EMBARQUES"
          <EC>  SUM((BUQUE=[1,2];ORIGEN=[1,4];DESTINO=[1,4];
                     PER_FIN=[1,12]),
                        CTR(BUQUE;ORIGEN;DESTINO)*
                        RTR(BUQUE;ORIGEN;DESTINO;PER_FIN)
                     <ELE> OR_D_DT(ORIGEN;DESTINO))
                + SUM((BUQUE=[1,2];ORIGEN=[1,4];DESTINO=[1,4];
                     PER_FIN=[1,12]),CTM(BUQUE;DESTINO)*
                        RTR(BUQUE;ORIGEN,DESTINO;PER_FIN)
                     <ELE> OR_I_DT(ORIGEN;DESTINO))
                + SUM((DESTINO=[2,3*;PER_FIN[2,12],
                        COM(DESTINO)*OMD(DESTINO;PER_FIN))
                  <MIN> COSTO


RESTRICCIONES

B.1.3     <RES> ORDE(J), "ORDEN DEL DETALLISTA"
          <EC>  SUM((I=[1,2]),EMBARQ(I,J))
                <MAI> DEM(J)

B.4.3     <RES> COM(I), "PROYECTOS ASIGNADOS POR COMPAÑIA"
          <EC>  SUM((J=[1,3]),PROY.A(I;J))
                <MEI> 1

B.6.3     <RES> LA_P(PUERTO,PER_FIN), "LIMITE DE ALMACENAMIENTO"
          <DOM> (PUERTO <DIF> {1,4};PER_FIN=[1,12])
          <EC>  FICT1(PUERTO,PER_FIN)*EX_P(PUERTO;PER_FIN)
                <MAI> (0.2*CAP_AL(PUERTO;PER_FIN);
                       (0.6*CAP_AL(PUERTO;PER_FIN)
```

3.10.2 Definición de los dominios del sumando.

En el dominio-sumatoria se indican los índices que controlarán el proceso de sumarización, estos índices como ya fue mencionado en el apuntador 2.2.5, pueden llegar a ser seis y a cada uno de ellos se les debe definir un dominio particular, es decir, se les debe siempre asignar un conjunto de valores.

La sintaxis para la formación del dominio-sumatoria es la siguiente:

```
«dominio-sumatoria» ::= «índice» «formación-dominio-índice»
                          ┤;«dominio-sumatoria»├*

«formación-dominio-índice» ::=
                    = «dominio-índice» | <DIF> «dominio-índice»

«dominio-índice» ::= «valor-entero positivo» |
                     [«rango-inferior», «rango-superior»] |
                     {«lista-valores-dominio»}
```

«lista-valores-dominio» ::=

«valor-entero-positivo»┤,«lista-valores-dominio»├*
[«rango-inferior»,«rango-superior»] ┤,«lista-valores-dominio»├*

En el dominio-conjunto se acepta definir equivalencias entre índices y dominios, no se aceptan desfasamientos ni dominios particulares, términos que se comentan a continuación.

A través del dominio-variable y del dominio-parámetro se logran definir tres operaciones básicas. La primera es definir un desfasamiento del elemento invocado, esto es, definir un retraso o adelanto del valor del índice de la variable o parámetro con respecto al valor del mismo índice de la restricción.

La segunda operación es definir una equivalencia entre el índice de la variable y/o parámetro con un índice (diferente) de la restricción.

Finalmente la tercera operación es definir el espacio de aplicación de la variable y/o del parámetro en la ecuación, esto es definir los elementos del conjunto variable o del conjunto parámetro que participarán en la ecuación que se está definiendo. (Ver sección 2.2.10).

Es importante recordar que cada sumando con un coeficiente explícito, genera un espacio de aplicación definido por las reglas del operador * (multiplicación) y que cualquier dominio particular o equivalencia que se defina en el coeficiente o en la variable, afecta a ambos por igual.

Como ya fue establecido en la sección 3.10.1, la sintaxis para la definición de estos dominios es la siguiente:

«dominio-variable» ::= «dominio-tl»

«dominio-parámetro» ::= «dominio-tl»

«dominio-tl» ::=
«equivalencia-dominio-índice» ┤;«dominio-tl»├*

«equivalencia-dominio-índice» ::=
«índice»
«desplazaiento-índice»?
«equivalencia-índice»?
«información-dominio-índice»

«índice» ::= «identificador-índice» |
«identificador-subíndice»

```
«desplazamiento-índice» ::=
        + «desplazamiento» |
        - «desplazamiento»

«desplazamiento» ::= «valor-entero-positivo»

«equivalencia-índice» ::=
        = «identificador-índice-equivalente»

«identificador-índice-equivalente» ::=
        «identificador-índice» |
        «identificador-subíndice»
```

en donde formación-dominio-índice ya se definió al inicio de esta sección.

EJEMPLOS:

En los apartados del apéndice B referidos a continuación se pueden observar las siguientes definiciones:

B.3.3)
```
<RES> LIM_FL(ORG,DST),"LIMITE DE FLUJO EN EL TRAMO"
<EC>  FICT(ORG;DST)*FLUJO(ORG;DST)
      <MEI> FLU_MX(ORG;DST)
```

En esta restricción se puede observar que todos los elementos tienen el mismo espacio de aplicación, el cual está definido por el producto cartesiano del conjunto de valores teóricos de los índices ORG y DST.

```
<RES> EN_SL(NODO),"BALANCE ENTRADA/SALIDA"
<DOM> (NODO=[2,5])
<EC>  SUM((ORG=[1,6]),FLUJO(ORG;DST=NODO))
     -SUM((DST=[1,6]),FLUJO(ORG=NODO;DST))
      <IG> 0
```

En esta restricción se observa que el espacio de aplicación sobre la ecuación está definido por NODO=[2,5]; que el primer sumando lo constituye la sumatoria de la variable FLUJO, controlada por el índice ORG y por lo tanto eliminado del dominio de aplicación de este primer sumando, también se puede observar que la consistencia del espacio de aplicación de esta sumarización con el espacio de la restricción, se logra haciendo la equivalencia explícita entre el índice DST de la variable con el índice NODO de la restricción.

En el segundo restando, el índice DST de la variable FLUJO se elimina mediante la sumarización, y la

consistencia del espacio de aplicación se logra haciendo una equivalencia explícita entre el índice ORG de la variable FLUJO con el índice NODO de la restricción.

B.5.3)
```
<RES> SALE(N,S),"NODO DEL QUE SALE EN LA SECUENCIA S"
<DOM> (N=[1,4]; S=[2,4]
<EC>  SUM((I=[1,4]),TRAMO(I;J=N;S-1))
     -SUM((J=[1,4]),TRAMO(I=N;J;S))
      <IG> 0
```

En esta restricción se puede observar la equivalencia explícita J=N y I=N, con lo cual se logra la equivalencia entre los espacios de aplicación de los sumandos y la restricción. Asimismo en el primer sumando se puede observar un desplazamiento negativo del índice S, esto quiere decir que la variable TRAMO cuando S=1 se sumará en la restricción SALE para S=2; que TRAMO cuando S=2 se sumará en SALE para S=3; etc.

Nótese también en esta restricción, que se ha declarado un dominio restringido para el índice S definido como S=[2,4], lo cual es necesario para ser consistente con el desfasamiento ya comentado en el párrafo anterior.

B.6.3)
```
<RES> CAR_R(BUQUE,PER_FIN),"CARGA EN REFINERIAS"
<EC>  RTR(BUQUE;ORIGEN=1;DESTINO=1;PER_FIN)
     -SUM((LOTE[1,4]),FICT2(BUQUE;PER_FIN)*
                        CARGA(BUQUE;PER_FIN;LOTE))
      <IG> 0
```

En esta restricción se puede observar que el espacio de aplicación de la variable RTR se ha limitado para ORIGEN=1 y DESTINO=1, esto quiere decir que del conjunto variable RTR, sólo se sumarán aquellos elementos cuyos índices ORIGEN y DESTINO valgan 1.

3.11 POSTULADO <ARCH_E> (definición de archivos)

A continuación de las restricciones, se definen todos los archivos que se usarán en la integración del modelo.

Por cada archivo se requiere un postulado <ARCH_E> y a continuación tantos postulados <DATO> como atributos diferentes se obtendrán del archivo.

Toda familia de parámetros que vaya ha ser leída de archivo requiere de un postulado <DATO>.

Si el modelo no requiere de archivos, los postulados <ARCH_E> y <DATO> se deben omitir.

Los archivos que se pueden leer son archivo tipo ASCII. Los separadores de los diversos campos o valores de un registro son las comas y los espacios. Una literal alfanumérica puede llevar como parte de ella a estos separadores, siempre y cuando esté delimitada por comillas.

El lenguaje sólo reconoce los diez primeros valores (campos) del archivo, mismos que por definición se identificarán como:

```
<CAMPO>(1)  el primer valor del registro
<CAMPO>(2)  el segundo  "          "
<CAMPO>(3)  el tercer   "          "
<CAMPO>(4)  el cuarto   "          "
<CAMPO>(5)  el quinto   "          "
<CAMPO>(6)  el sexto    "          "
<CAMPO>(7)  el séptimo  "          "
<CAMPO>(8)  el octavo   "          "
<CAMPO>(9)  el noveno   "   del registro.
<CAMPO>(10) el décimo   "          "
```

FORMATO:

<ARCH_E> Palabra reservada que debe escribirse tal cual.

"Nombre-del-archivo"

Nombre del archivo que se está definiendo.

El nombre debe respetar las reglas del MS-DOS para la definición de archivos.

No se acepta ninguna mención a directorios o dispositivos como parte del nombre.

,Número-máximo-de-campos

Se indica el número máximo de campos que contiene el archivo. El número máximo de campos identificables por el sistema es de diez.

<SEC> La palabra reservada <SEC> inicia la definición de la secuencia en que debe estar el archivo. Esta definición es necesaria para optimizar la búsqueda de los distintos valores en el archivo.

<CAMPO>(n) <CAMPO>(m) ...<CAMPO>(s)

A continuación se deben enunciar los campos del

archivo (n,m,..s) que establecen la secuencia del mismo.

La referencia a los campos deberá ser de acuerdo a sus precedencias. El más general en primer lugar, el de mayor detalle al final.

Todo campo de donde se lea el valor de un índice debe enunciarse en esta sección, en el lugar que le corresponda.

EJEMPLOS:

B.5.4) <ARCH> "ARCVIAJE",3 <SEC> <CAMPO>(1) <CAMPO>(2)

En el ejemplo 5 del apéndice B, se define el archivo ARCVIAJE compuesto de tres campos, en donde los dos primeros establecen la secuencia del archivo.

3.12 POSTULADO <DATO> (definición de los datos del archivo)

Por medio del postulado <DATO> se definen los elementos del modelo que se leerán del archivo y la forma de encontrarse.

De un archivo se pueden leer: los parámetros del modelo, los vectores frontera superior e inferior asociados a las variables mediante los postulados <VAR>, los significados particulares asociados a los índices y los tuples que conforman un conjunto.

El postulado DATO está integrado por tres secciones:

1) La cláusula de selección.

   Con ella, es posible seleccionar del archivo, sólo aquellos registros que interesan para el parámetro o vector que se está definiendo.

2) La definición del elemento y su dominio.

   Esta sección permite definir, a través del dominio, los elementos del conjunto que se obtendrán del archivo.

   La definición del dominio deberá ser en base al contenido del archivo, así, cuando menos uno de los índices que conforman el dominio, tomará su valor de los datos del archivo.

3) Expresión para calcular el valor del elemento.

   El valor del parámetro o vector, deberá abstenerse de la

información del archivo.

En principio, el valor del elemento deberá ser un dato contenido en alguno de los diez primeros campos del archivo, sin embargo, mediante una expresión numérica, que contenga cuando menos un campo del archivo como operando, se puede calcular el valor del elemento.

FORMATO:

<DATO> Palabra reservada que debe escribirse tal cual.

Cláusula de selección.

<SI><CAMPO>(n) = Identificador-del-registro

Esta sección es opcional. Si se define, se seleccionarán del archivo, sólo aquellos registros en donde el campo n sea igual a la constante entera definida como identificador-del-registro.

Definición del elemento y su dominio

identificador-parámetro (dominio-archivo) ó
identificador-vector-frontera-inferior (dominio-archivo) ó
identificador-vector-frontera-superior (dominio-archivo) ó
identificador-vector-significados-particulares (índice-campo) ó
identificador-vector-tuples (lista-índice-campo)

En esta sección se define el parámetro o vector que se buscará en el archivo, así como el dominio de los elementos del conjunto que se deberá buscar.

El conjunto de índices asociado al elemento que se leerá y sus respectivos subdominios debe ir entre paréntesis. Cada par índice-subdominio debe separarse por punto y coma (;). La secuencia de definición de los índices es importante y debe respetarse la especificada en la definición original del parámetro o vector.

En el siguiente apartado se describen las diversas alternativas para definir estos dominios (dominio-archivo, índice-campo y lista-índice-campo).

Valor del elemento

=<CAMPO>(n) o
=expresión

En el primer caso, al parámetro o vector definido en este postulado, se le asignarán, secuencialmente, los datos encontrados en el campo n del archivo.

En del segundo caso, el valor del elemento se establece mediante una expresión numérica que contenga forzosamente a <CAMPO>(n) como un operando.

La expresión se puede formar con:

- referencias a otros campos del archivo y/o constantes explícitas como operandos.

- como operadores, se pueden utilizar:

  + suma
  - resta
  * multiplicación
  / división
  ^ elevar a
  (inicio de agrupamiento
) fin de agrupamiento.

Los vectores de tuples no requieren de esta sección; los vectores de significados particulares, sólo aceptan la primera opción, ya que en ella se define el campo donde se encuentra el significado particular del índice.

Definición de dominios

La definición de los dominios, como ya se ha mencionado en repetidas ocasiones, se establece, asignando los valores correctos a cada uno de los índices asociados al elemento que se leerá del archivo.

Cuando se está definiendo un vector de significados particulares, en el dominio o espacio de aplicación del vector (índice-campo) sólo se acepta igualar el índice a un campo del archivo (alternativa 4).

Cuando se está definiendo un vector de tuples, en el dominio sólo se acepta que a cada índice del tuple se le asigne un número de campo del archivo. La invocación a los índices deberá respetar la definición original del conjunto.

El dominio de los parámetros (dominio-archivo), deberá respetar la secuencia de los índices, establecida en la definición original.

Los vectores frontera deberán respetar la misma secuencia de invocación de índices, a la establecida en la definición de la variable asociada a la frontera.

En todos los casos de este postulado, cuando menos a un índice del dominio, se le deberá asociar un campo del archivo (ver alternativa 4).

Las alternativas para asignar a los índices, los valores válidos del dominio son:

Alternativa 1

Nombre-del-índice = valor

Se define al subdominio del índice como el valor especificado.

Alternativa 2

Nombre-del-índice = [valor-inferior-del-rango,
valor-superior-del-rango]

Se define al subdominio del índice como el conjunto de valores comprendidos dentro del rango especificado, esto es, el conjunto de valores consecutivos que inician en el valor-inferior-del--rango y terminan en el valor-superior-del-rango.

Alternativa 3

Nombre-del-índice = {lista-de-valores del subdominio}

Se define al subdominio del índice mediante una lista de valores, esto es, un conjunto de enteros separados por una coma (,). En esta lista de valores se permite definir como un elemento rangos consecutivos tales como [1,4] que define al conjunto de valores 1,2,3,4.

Alternativa 4

Nombre-del-índice=<CAMPO>(n)

Al índice especificado se le asignarán, secuencialmente, los datos encontrados en el campo n del archivo.

Si en un registro, el dato en el campo especificado no pertenece al dominio teórico del índice, ese registro del archivo será ignorado.

EJEMPLOS:

B.5.4) <DATO> DIST(I=<CAMPO>(1);J=<CAMPO>(2);S=[1,4])=<CAMPO>(3)

Este postulado del ejemplo 5 (apéndice B) establece que de cada registro del archivo se deben obtener:

el valor de I del campo 1 del registro,

el valor de J del campo 2 del registro,
los valores de DIST(I,J,1),
DIST(I,J,2),
DIST(I,J,3),
y DIST(I,J,4),

serán iguales al valor encontrado en el campo 3 del registro.

3.13 POSTULADO <CONS> (definición de reglas de consistencia)

A continuación de las restricciones se definen las reglas de consistencia. Este postulado es optativo, se usará siempre que se desee verificar ciertas reglas que deban cumplir los parámetros y/o las variables del modelo.

Existen tres tipos de consistencia que se pueden definir, la primera se utiliza cuando se quiere verificar que todos los elementos de un conjunto parámetro cumplen con una relación (operador de relación) respecto a una constante; el segundo tipo de consistencia que se puede verificar se refiere a que una expresión de parámetros (más de un parámetro) cumpla con una relación respecto a una constante; finalmente, el tercer tipo de consistencia que se puede definir, se refiere a verificar la existencia obligatoria como sumandos, de todos los elementos de un conjunto variable en un conjunto restricción.

El sistema avisa de todos aquellos elementos que no cumplieron con la relación definida en las reglas de consistencia.

FORMATO:

<CONS> Palabra reservada que se define tal cual.

"significado general"

Sirve para darle una etiqueta, nombre o explicación a la regla que se está definiendo.

Puede ser cualquier leyenda delimitada por comillas y no mayor a 250 caracteres. No se aceptan ni comillas (" ") ni signos de pesos ($) como parte de la leyenda.

Alternativa 1 Consistencia de un parámetro

<TODO> Palabra reservada opcional, si se define se establece que todos y cada uno de los elementos del espacio de aplicación teórico del conjunto parámetro deberán existir y deberán cumplir con la relación que se está definiendo en esta regla de consistencia.

Si se omite esta palabra reservada, sólo aquellos parámetros existentes del conjunto deberán cumplir con la regla de consistencia que se está definiendo.

identificador-parámetro

Se define el conjunto parámetro para el cual se está definiendo la regla de consistencia.

operador-relación

Puede ser cualquiera de los operadores de relación: <IG>, <DIF>, <MA>, <MAI>, <ME>, <MEI>.

constante Constante contra la cual se compararán todos los elementos del conjunto parámetro.

Alternativa 2 Consistencia entre parámetros

(expresión-parámetros)

Se declara una expresión de parámetros entre paréntesis, misma que se verifica cumpla con la relación que a continuación se expresa.

operador-relación

Puede ser cualquiera de los operadores de relación: <IG>, <DIF>, <MA>, <MAI>, <ME>, <MEI>.

constante Constante contra la cual se comparará el resultado de la expresión definida.

Alternativa 3 Consistencia de variables

<SI> Palabra reservada que se escriben tal cual.

identificador-variable

Nombre del conjunto variable para el cual se está definiendo la regla de consistencia.

<ELE> <identificador-restricción-selección>
<NOELE> <identificador-restricción-selección>

Esta sección de selección es opcional, si no se define, la regla de consistencia se aplica para todos y cada uno de los elementos del conjunto variable.

Si esta sección se define, sólo aquellas variables

que son sumandos (<ELE>) de la restricción de selección (o aquellas variables que no son sumandos <NOELE> de esta restricción) se les verifica la regla de consistencia que se está definiendo.

<IMPLICA> <ELE>

Palabras reservadas que se escriben tal cual.

identificador-restricción-consecuente

Sirve para definir la restricción en la cual, los elementos del conjunto variable declarado, deben existir como sumandos.

En resumen, las tres posibilidades para verificar que cada elemento del conjunto variable exista como sumando en un conjunto restricción dado, son:

- Si la variable existe como elemento del modelo, ésta debe ser un sumando de la restricción definida como consecuente.

- Si la variable es sumando de la restricción definida en la sección de selección, deberá ser también sumando de la restricción consecuente.

- Si la variable no es sumando de la restricción de selección, entonces deberá ser sumando de la restricción consecuente.

EJEMPLOS:

B.6.3)
```
<CONS>"CARGA INICIAL ≤ LIMITE DE CAPACIDAD"
      (LIM_C(BUQUE;PER_FIN=1)
      - CR_INC(BUQUE))
      <MAI>  0

<CONS>"EXISTENCIA INICIAL ≤ LIMITE SUPERIOR DE
       OPERACION"
       ( (.8 * CAP_AL(PUERTO;PER_FIN))
       - EX_IN_P(PUERTO;PER_FIN))
       <MAI>  0

<CONS>"EXISTENCIA INICIAL ≥ LIMITE INFERIOR DE
       OPERACION
       ( EX_IN_P(PUERTO;PER_FIN)
       - (.2 * CAP_AL(PUERTO;PER_FIN)) )
       <MAI>  0
```

La primera regla de consistencia asegura que la carga inicial del buque -CR_INC(BUQUE)- sea menor que el límite de carga del mismo.

En el segundo y tercer ejemplo, se asegura que la existencia inicial en puerto -EX_IN_P(PUERTO;PER_FIN)- esté dentro de los límites de operación de la tanquería.

3.14 POSTULADO <VALOR> (definición de valores para parámetros, límites y rangos de restricciones)

Permite definir el valor de parámetros, de vectores frontera-inferior y de vectores frontera-superior.

FORMATO:

<VALOR> Palabra reservada que debe escribirse tal cual.

Nombre Nombre del parámetro, nombre del vector frontera-inferior o nombre del vector frontera-superior que se está definiendo.

Este nombre debe ser igual al dado en la definición original.

(dominio) El conjunto de índices asociado al conjunto de parámetros, (o vector frontera) y sus respectivos subdominios debe ir entre paréntesis. Cada par índice-subdominio debe separarse por punto y coma (;). La secuencia de definición de los índices es importante y debe respetarse la especificada en la definición original del parámetro.

La secuencia de definición de los índices de frontera deberá ser la misma que la definida para la variable asociada al vector frontera que se define.

Existen varias alternativas para definir el subdominio del índice, todas ellas se describen en el siguiente apartado.

={lista-de-valores}

Lista de valores enteros ó reales separados por una coma (,) o por uno o más blancos. Estos valores se asignan a los parámetros, (vectores), uno a uno, en correspondencia al subdominio definido en la primera parte del postulado.

En esta lista de valores se permite definir valores consecutivos tales como 3,3,3,3 con la expresión 4*3, donde el número cuatro es el factor de repetición.

Como un caso especial, cuando la lista de valores se

componga de un solo valor, éste, debe ir también encerrado entre llaves {valor}.

Cuando se requiere definir la inexistencia de un valor, se deberá escribir la palabra NULO, misma que puede ir precedida de un factor de repetición.

Alternativa 1

Nombre-del-índice = valor

Se define al subdominio del índice como el valor especificado

Alternativa 2

Nombre-del-índice = [valor-inferior-del-rango,
valor-superior-del-rango]

Se define al subdominio del índice como el conjunto de valores comprendidos dentro del rango especificado, esto es, el conjunto de valores consecutivos que inician en el valor inferior del rango y terminan en el valor superior del rango.

Alternativa 3

Nombre-del-índice = {lista-de-valores del subdominio}

Se define al subdominio del índice mediante una lista de valores, esto es, un conjunto de números separados por una coma (,). En esta lista de valores se permite definir como un elemento, rangos consecutivos tales como [1,4] que define al conjunto de valores 1,2,3,4.

EJEMPLOS:

B.1.3) <VALOR> COSTO.U (I=[1,2];J=[1,3])={14,10,12,15,13,8}

Este postulado establece que el valor del costo unitario del embarque de la fabrica I al detallista J, es:

COSTO.U(1,1)=14, COSTO.U(1,2)=10, COSTO.U(1,3)=12
COSTO.U(2,1)=15, COSTO.U(2,2)=15, COSTO.U(2,3)=8.

<VALOR> EXTNC(I=[1,2])={1200, 1000}

Este postulado establece que el valor límite de existencias en la fabrica es:

EXTNC (1)=1200, EXTNC (2)=1000.

B.2.3) <VALOR> FINF(I={3,4} = {400,100}

Este postulado establece que los valores frontera inferior son:

FINF(3) = 400 y
FINF(4) = 100

B.3.3) <VALOR> FLU_MX(ORG=[1,6];DST=[1,6]=

```
{   NULO     4     6          3*NULO
  3*NULO     6     2            NULO
    NULO     3     2*NULO     4 NULO
  5*NULO     6
  3*NULO     1       NULO     2
  6*NULO }
```

Postulado que establece el siguiente conjunto de valores para el parámetro FLU_MX(ORG,DST).

```
FLU MX(1,2)=4          FLU MX(1,3)=6
FLU MX(2,4)=6          FLU MX(2,5)=2
FLU MX(3,2)=3          FLU MX(3,5)=4
FLU MX(4,6)=6
FLU MX(5,4)=1          FLU MX(5,6)=2
```

3.15 POSTULADO <FIN> (fin del modelo)

Mediante este postulado se indica el fin de la definición del modelo y debe ser el último postulado.

3.16 PALABRAS Y SIMBOLOS RESERVADOS DEL LENGUAJE

PALABRA/SIMBOLO	SIGNIFICADO
<ARCH_E>	Inicio de la definición de un archivo de entrada.
<BIN>	<VAR><BIN> Inicia la definición de una variable binaria.
<CAMPO>(n)	Referencia al campo enésimo del archivo.
<COMPL>	Se utiliza para definir un conjunto de n-tuples en función del complemento de otro.
<CONS>	Inicia la definición de las reglas de consistencia.

PALABRA/SIMBOLO	SIGNIFICADO
<DATO>	Inicio del postulado <DATO> donde se establecen los elementos del modelo cuyos datos se deben buscar en un archivo.
<DIF>	Se interpreta como "el conjunto diferente a ", en la definición del subdominio de un subíndice.
<DOM>	Inicio de la definición del subdominio de una restricción.
<EC>	Inicio de la definición de la ecuación de la restricción.
<ELE>	Conector para condicionar el sumando de una restricción a ser elemento de un conjunto de n-tuples.
<ENT>	<VAR><ENT> define una variable entera.
<EXC>	En la definición del subdominio de un subíndice (o de una subrestricción) se debe leer como "del índice padre (o restricción padre),excluye como elementos del dominio a los siguientes valores".
<FIN>	Fin de la definición del modelo
<FO>	Prefijo para definir el nombre en donde MPSX encontrará el vector de coeficientes de la Función Objetivo.
<FR>	Prefijo para definir el nombre en donde MPSX encontrará el vector de fronteras de las variables.
<FRON-INF>	Prefijo para definir el nombre del vector donde GEMOLI encontrará los valores frontera inferior de una variable.
<FRON-SUP>	Prefijo para definir el nombre del vector donde GEMOLI encontrará las variables frontera superior de una variable

PALABRA/SIMBOLO	SIGNIFICADO
<IG>	Operador de relación igual que.
<IGD>	En la definición del subdominio de un subíndice (o de una subrestricción), se debe leer como "igual dominio que".
<IMPLICA>	Prefijo para definir la restricción consecuente en las reglas de consistencia.
<INC>	En la definición del subdominio de un subíndice (o de una subrestricción), se debe leer como "incluye como elementos del dominio los valores que están a continuación".
<IND>	Inicio de la definición de un índice.
<MA>	Operador de relación mayor que.
<MAI>	Operador de relación mayor o igual que.
<MAX>	La suma de las variables de la ecuación deberá ser un máximo.
<ME>	Operador de relación menor que.
<MEI>	Operador de relación menor o igual que.
<MENOS>	Operador para resta de conjuntos.
<MIN>	La suma de las variables de la ecuación deberá ser un mínimo.
<MOD>	Inicio de la definición del modelo.
<NOELE>	Conector para condicionar el sumando de una restricción a no ser elemento de un conjunto de n-tuples.
<O>	Operador de unión de conjuntos.
<PAQ>	Prefijo para definir el paquete de programación lineal que se utilizará.

PALABRA/SIMBOLO	SIGNIFICADO
<PARAM>	Inicio de la definición de un parámetro.
<REAL>	<VAR> <REAL> Sirve para definir que la variable es de tipo real.
<REDONDEADO>	Prefijo de una expresión de parámetros para pasar el valor de la misma a entero, mediante un redondeo.
<RES>	Inicio de la definición de la restricción.
<RG>	Prefijo para definir el nombre del vector en donde MPSX encontrará los valores de los rangos de las restricciones.
<RS>	Prefijo para definir el nombre del vector en donde MPSX encontrará los límites ("RIGHT HAND SIDE") de las restricciones.
<SC>	Define un índice o una restricción como un subconjunto de otro índice o de otra restricción respectivamente.
<SEC>	Inicia la definición de la secuencia de un archivo.
<SG>	Significado general.
<SI>	Inicio de la sección de selección o condición de un postulado.
<SP>	Significado particular.
<TODO>	Prefijo para indicar que la regla de consistencia se debe aplicar a todo el espacio de aplicación teórico de un parámetro.
<VALOR>	Inicia la asignación de valores a: parámetros y vectores frontera de variables.
<VAR>	Inicia la definición de una variable.

PALABRA/SIMBOLO	SIGNIFICADO
<Y>	Operador de intersección de conjuntos.
SUM	Σ sumatoria
+	suma
-	resta
*	multiplicación
/	división
^	elevar al exponente
=	igual "conjunto igual que"
,	separador de elementos del lenguaje.
;	En la definición de subdominios, sirve para separar las definiciones del subdominio de cada índice.
{	Inicia una lista de valores.
}	Termina una lista de valores.
[	Inicia la definición de un rango de valores.
]	Termina la definición de un rango de valores.
(	Inicia un agrupamiento de elementos.
)	Termina un agrupamiento de elementos.

4 CALCULO DE LAS EXPRESIONES DEL MODELO

4.1 CARACTERISTICAS DE LAS EXPRESIONES

En la definición de los modelos pueden existir dos tipos de expresiones: aquellas que se refieren al cálculo de parámetros y las que se utilizan para definir conjuntos de combinaciones de valores válidos para los índices del modelo.

Las expresiones para el cálculo de parámetros se encuentran en la definición de los parámetros y en la definición de las ecuaciones de las restricciones; las expresiones para la definición de conjuntos se pueden especificar en la definición misma de los conjuntos y también, en la definición de las expresiones de las restricciones.

Al analizar en conjunto todas las expresiones de un modelo, se puede encontrar que existen parámetros invocados en más de una expresión y en algunos casos, pueden existir subexpresiones que se repiten en las ecuaciones del modelo, a este tipo de elementos los mencionaremos en forma genérica como parámetros multicitados.

El énfasis especial en el cálculo de expresiones es consecuencia directa de las características de los parámetros utilizados en este tipo de modelos. Toda invocación a un parámetro se puede traducir en el proceso de un conjunto de valores, cuya cardinalidad varía de parámetro a parámetro y que por diseño se ha limitado a 30,000 elementos para cada parámetro. Cualquier operación binaria sobre parámetros puede llegar a requerir teóricamente 90,000 valores almacenados simultáneamente en memoria (30,000 para cada operando y 30,000 para el resultado).

Otra característica importante de estos parámetros es que los conjuntos de valores normalmente son muy porosos; en muchos casos, en especial aquellos parámetros con un número alto de índices asociados, el conjunto real de valores es bastante menor que la mitad del número de valores teóricos.

Como resultado de estas características, si se desea que el cálculo de expresiones sea un proceso eficiente, es necesario que todo parámetro multicitado residente en memoria, ya sea por haberse leído o por haberse calculado, se aproveche al máximo; con él se deberán satisfacer en lo posible, todas las invocaciones que se hagan del mismo en el conjunto de ecuación del modelo.

Dependiendo del tamaño del modelo y de las limitaciones de memoria, en la práctica, se trata de minimizar el recálculo de subexpresiones multicitadas, así como de evitar múltiples lecturas de un mismo parámetro.

4.2 DESCRIPCION GLOBAL DEL PROCESO

Los elementos multicitados son los vectores que competirán por permanecer en memoria hasta haberse utilizado en todas las expresiones que los invocan, por lo que su manejo difiere radicalmente de los parámetros y expresiones que son invocados y calculados una sola vez. Si bien estos últimos se procesan mediante una pila de ejecución, (son almacenados en la parte superior de la pila cuando van a ser utilizados y, una vez ejecutada la operación, los operandos se eliminan de la parte superior de la pila). Los parámetros multicitados requieren un proceso más complejo. En primer lugar, habrá que escoger qué parámetros deberán permanecer en memoria, en qué momento se deberán descargar de la misma y por lo tanto cuáles se leerán dos o más veces.

Una vez determinada la permanencia de cada parámetro a lo largo de la evaluación de todas y cada una de las expresiones del modelo, se les deberá asignar una localización en memoria. El proceso de asignación de memoria deberá minimizar espacios no utilizados, para asegurar que la evaluación de las expresiones cuente con espacio suficiente.

La necesidad de facilitar y hacer más eficientes los procesos recién expuestos, hizo que se tomaran las siguientes decisiones de diseño:

1) Todo conjunto de valores de cada parámetro tiene asociados dos vectores: el primero, donde residirán los valores reales del parámetro, ocupará un vector de variables reales tan grande como el dominio teórico del parámetro; el segundo es un vector de bits donde posicionalmente se indica la existencia o ausencia del valor respectivo en el vector de valores. Este vector ocupa un arreglo de variables enteras cuya longitud será una quinceaba parte del tamaño del arreglo de vectores.

   Esta condición de diseño evita limpiar las áreas de valores y permite la determinación inmediata, mediante un 'and', de la existencia de los operandos correspondientes de toda operación.

2) A efecto de localizar rápida e inequívocamente el valor y/o el bit de existencia de un parámetro particular, el cálculo del ordinal de tal elemento, se realiza en función de los valores de los índices asociados al parámetro, según la siguiente fórmula:

   Ordinal = (Valor del Ultimo Indice - 1) *6

   (Factor de Ponderación del Ultimo Indice) +

   (Valor del Penúltimo Indice - 1) *

(Factor de Ponderación del Penúltimo Indice)+
.
.
.
(Valor del Primer Indice - 1) *
(Factor de Ponderación del Primer Indice) + 1

En donde el primer índice es el de la extrema izquierda y el factor de ponderación del i-ésimo índice (f_i) es:

f_i = 1

f_{i-1} = Valor máximo declarado que puede tomar el índice i

f_{i-2} = f_{i-1} * Valor máximo declarado para el índice i

f_i = f_2 * Valor máximo declarado para el índice 2

Así, el ordinal para A(1,1,1) = 1
A(1,3,1) = 5
A(2,2,2) = 10
cuando se declara que el valor máximo del primer índice es 2, del segundo índice es 3 y del tercer índice es 2.

3) La memoria disponible para la evaluación de las expresiones se ha dividido en dos partes: la parte inferior se utiliza para almacenar los parámetros multicitados; la parte superior se utiliza para almacenar la pila de ejecución de la expresión.

   El inicio de la memoria para la pila de ejecución varía de expresión a expresión, depende de la memoria ocupada por los parámetros multicitados que deban estar presentes durante la ejecución de cada expresión.

4) Al inicio del cálculo de toda expresión, los valores de todos los parámetros multicitados utilizados en la expresión ya deberán estar presentes en la parte baja de la memoria, (salvo el valor que se calcula como resultado de la expresión). Esta consideración permite disociar el proceso de asignación de direcciones de memoria a los parámetros multicitados, del proceso de localización de la pila de evaluación de la expresión.

El cálculo de expresiones del modelo está constituido por seis etapas secuenciales.

4.3 IDENTIFICACION DE LAS EXPRESIONES DEL MODELO

En la primera etapa se identifican y reagrupan todas las expresiones del modelo (expresiones de parámetro y de conjunto); se identifican y marcan aquellos parámetros cuyos valores serán leídos de disco (parámetros básicos). Para los parámetros calculados se identifica el inicio y fin de la expresión asociada, y se marcan todos los parámetros y conjuntos que son invocados en las ecuaciones de las restricciones (parámetros permanentes), finalmente se identifican los índices asociados a cada parámetro, así como el tamaño de la memoria requerida para su almacenamiento. Todas estas características se integran en una tabla ya que serán utilizadas a lo largo del procedimiento.

4.4 INTEGRACION DE LA GRAFICA DAG

La segunda etapa la constituye la integración de la gráfica acíclica dirigida (DAG) con el conjunto de todas las expresiones del modelo, así como la detección de subexpresiones repetidas y la identificación del número de invocaciones para cada parámetro básico y para cada una de las subexpresiones repetidas.

La integración de la gráfica DAG y la búsqueda de subexpresiones se realiza con base en los algoritmos enunciados en Aho [1], con las siguientes modificaciones:

1) Se utiliza un operador virtual de secuenciación, para incluir en una sola gráfica todas las expresiones del modelo y calcularlas (salvo las totalmente independientes) en el orden en el cual fueron declaradas.

2) Se considera que dos subexpresiones son equivalentes si tienen el mismo operador, los mismos operandos y el mismo dominio declarado, para todos los índices de cada operando. Así:

A(i;j) * B(i)	es equivalente a	A(i;j) * B(i)
A(i={2,3};j)*B(i)	es equivalente a	A(i={2,3};j)*B(i)
A(i={2,3};j)*B(i)	no es equivalente a	A(i=3;j)*B(i)

Con base en la gráfica generada, se integra una matriz que relaciona a los parámetros y subexpresiones multicitadas, renglón i de la matriz, con cada una de las expresiones a calcularse, columna j de la matriz; como elementos de la matriz M(i,j), se tienen:

- Un número que indica el número de invocaciones que del parámetro i se realizan en la expresión j. Este número indica también que el parámetro deberá ser leído a memoria antes de iniciar la evaluación de la expresión.

- Un carácter 'E', para indicar que el parámetro/expresión i es el resultado final de la expresión j.

- Un carácter '-' indicará que el parámetro/expresión i no será utilizado por la expresión j, sin embargo, se requiere que el parámetro/expresión permanezca en la memoria, porque existe una expresión posterior que lo requerirá.

En esta etapa se realiza, en base a la matriz generada, la identificación y eliminación de expresiones que no se utilizan como paso intermedio para el cálculo de otras expresiones, y que tampoco son invocadas en las expresiones de las restricciones.

Finalmente, se reagrupan las expresiones interrelacionadas y las totalmente independientes. Las primeras son aquellas que utilizan los elementos multicitados o que generan un resultado que es invocado por alguna otra expresión; las expresiones totalmente independientes, al no requerir una secuencia de cálculo y/o un elemento multicitado, se pueden calcular en cualquier orden.

4.5 DEFINICION DE LA PERMANENCIA EN MEMORIA DE LOS PARAMETROS CRITICOS

En la tercera etapa se determina, para cada expresión, la secuencia de eventos de lectura de los parámetros básicos y su permanencia en memoria, así como la escritura a disco de los parámetros permanentes y/o de las expresiones intermedias que recién se han calculado.

Cuando el tamaño del modelo no permite que todos los parámetros requeridos en más de una ocasión permanezcan en memoria todo el tiempo necesario, la secuencia de eventos obtenida mostrará ciertos reprocesos (más de una lectura para un parámetro, y/o escritura y lectura de alguna subexpresión) que permitan liberar momentáneamente memoria ocupada.

La finalidad de la tercera etapa es encontrar la secuencia de eventos que minimice el reproceso.

La naturaleza y características de este proceso se clarifican si se analiza una sola expresión a la vez y posteriormente se integra el conjunto. Si se toma cualquier expresión intermedia, se tiene que el conjunto de parámetros multicitados que pueden permanecer en memoria una vez terminado el cálculo de la expresión son:

- Los parámetros multicitados que dejó en memoria la expresión anterior.

- Los parámetros que se leyeron por ser requeridos para el cálculo de la expresión.

- El resultado del cálculo de la expresión.

Si la memoria requerida por todos estos parámetros, más la memoria mínima requerida para evaluar la expresión, fuera menor a la memoria disponible, la solución en esta situación sería dejar en memoria todos los parámetros requeridos. Cuando éste no es el caso, se deberán seleccionar, mediante un criterio de optimización, aquellos parámetros que deberán permanecer en memoria y aquéllos que deben liberar espacio.

Para llevar a cabo esta selección, primero se tienen que analizar y establecer las alternativas factibles, las cuales son las combinaciones de los parámetros que sí caben en el remanente resultante de restarle a la memoria disponible, la memoria mínima requerida para evaluar todos los operadores de la expresión.

4.5.1 ETIQUETACION DE LA GRAFICA DAG

En el cálculo de la memoria mínima requerida para evaluar la expresión, se utiliza el algoritmo de etiquetación de la gráfica acíclica dirigida expuesto en Aho [2], con modificaciones para determinar la memoria requerida para el cálculo de cada operador de la expresión. Las reglas que se utilizan para esta determinación se sumarizan en las tablas de decisión 1 y 2 a continuación.

TABLA 1

REGLAS PARA EL CALCULO DE LA MEMORIA REQUERIDA EN CADA OPERACION

¿El resultado de la operación es un parámetro multicitado?	S	N	N
¿El operador es una sumatoria?		N	S
Memoria Requerida = 0	X		
Memoria Requerida = memoria para almacenar el resultado de la operación		X	
Memoria Requerida = memoria para almacenar el resultado de la operación + memoria para almacenar el operando izquierdo			X
EJECUTA TABLA 2	X	X	
REGRESAR	X	X	X

Nota: Al ser la sumatoria un operador unario, el único operando existente es el izquierdo.

TABLA 2

(continuación)

REGLAS PARA EL CALCULO DE LA MEMORIA REQUERIDA EN CADA OPERACION

¿El operando izquierdo es un parámetro multicitado?	S	N	N	N	N	N	N	N	N	N	N	N	N
¿El operando izquierdo es una expresión?		S	S	S	S	S	S	N	N	N	N	N	N
¿El dominio del operando izquierdo es equivalente al dominio del resultado de la operación?		S	S	S	N	N	N						
¿El operando derecho es un parámetro multicitado?	S	S	N	N	S	N	N	S	N	N	N	N	N
¿El operando derecho es una expresión?			N	S		S	S		S	S	N	N	N
¿El dominio del operando derecho es equivalente al dominio del resultado de la operación?						S	N		S	N			
¿El dominio del parámetro izquierdo es mayor o igual al dominio del parámetro derecho?											S	N	N
Memoria Requerida = Memoria Requerida + memoria para almacenar el resultado de la expresión izquierda.					X	X	X						
Memoria Requerida = Memoria Requerida + memoria para almacenar el resultado de la expresión derecha.				X			X			X			
Memoria Requerida = Memoria Requerida + memoria para almacenar el parámetro derecho.											X		
Memoria Requerida = Memoria Requerida + memoria para almacenar el parámetro izquierdo.													X
REGRESAR	X	X	X	X	X	X	X	X	X	X	X	X	X

Una vez etiquetada la gráfica acíclica dirigida con los valores de memoria requeridos, ésta se recorre, conforme al algoritmo expuesto en Aho [2], para determinar la secuencia en que se evaluará la expresión y detectar la simultaneidad entre los parámetros multicitados de entrada y el resultado de la expresión que se está evaluando. Con base en esta simultaneidad se corrige el valor de memoria requerida para los parámetros multicitados.

Para que una alternativa sea factible, la suma de la memoria mínima requerida para evaluar la expresión, más la memoria simultánea requerida para los parámetros multicitados, deberá ser menor que la memoria disponible.

4.5.2 ALGORITMO DE OPTIMIZACION

Para determinar las alternativas que en cada expresión nos conducen a un óptimo global, se utilizó el algoritmo A* descrito ampliamente en Nilsson [3], Tanimoto [5], Barr [6] y Schutzer [9].

El algoritmo A* es particularmente sensible a los valores que toma la función heurística, de estos valores dependen tanto la eficiencia del algoritmo como la certeza de que el óptimo encontrado es un óptimo global.

La función heurística del algoritmo implementado en este sistema representa al máximo aprovechamiento de los parámetros en memoria, es decir: para toda alternativa donde se leyó o calculó un parámetro que debería permanecer en memoria y que no es posible dejarlo, la función heurística asociada reflejará esta falta de aprovechamiento restando a la función heurística resultante de la decisión anterior, los costos de reproceso incurridos para el parámetro que se eliminará de memoria. Por cada parámetro que se elimina de memoria se puede incurrir en tres reprocesos básicos:

1) Si el parámetro existe en disco, se tendrá que leer nuevamente, para satisfacer la próxima invocación. Este caso se penaliza con 4 unidades por cada elemento del conjunto de valores del parámetro.

2) Cuando el parámetro recién se ha calculado, (y por lo tanto se tendrá que escribir a disco si se desean conservar sus valores), si por necesidades del modelo distintas al cálculo de parámetros, se requiere en disco, el volver a leerlo en el proceso de evaluación de expresiones, se penaliza con 4 unidades.

3) Si el parámetro se requiere únicamente para evaluar otros parámetros, el hecho de tenerlo que escribir y leer nuevamente se penaliza con 12 unidades por cada valor del conjunto. 8 unidades corresponden a la escritura en disco y 4 unidades son las relativas a la lectura.

Por lo tanto, si un parámetro se tiene que eliminar de la memoria, el valor de la función heurística se reducirá conforme a las reglas enunciadas a continuación, en donde:

CR_i es el costo de reproceso para el parámetro i
DT_i es el número de valores teóricos del parámetro i
FN es un factor de normalización que se describe más adelante.

1) Si el parámetro es básico, el costo de reproceso es igual a:

$$CR_i = 4 * DT_i/FN$$

2) Si el parámetro es una expresión calculada pero es un parámetro permanente,

$$CR_i = 4 * DT_i/FN$$

En este caso la penalización es igual que en el caso anterior, a pesar de ser una expresión recién calculada cuyos valores tendrán que escribirse en disco. En este caso, la escritura en disco no se penaliza, por ser requerida la información en este medio para el proceso posterior de las restricciones.

3) Si el parámetro es una expresión intermedia que una vez calculada tiene que salir y volver a memoria, se penaliza una sola vez con 8 unidades por la escritura a disco asociada, y cada regreso a memoria se penaliza con las 4 unidades asociadas a la lectura.

Los costos unitarios mencionados son arbitrarios, con ellos se pretende representar las diferencias en tiempo de máquina entre: calcular un parámetro, leerlo de disco o escribirlo en disco. Los valores de 4 para la lectura y 8 para la escritura, son el resultado de dividir respectivamente, los tiempos promedios de ejecución de la lectura y de la escritura de 10,000 valores, entre la suma del tiempo del proceso de cálculo de 5,000 valores producto de una multiplicación, más el tiempo de proceso de 5,000 sumas.

Si pudieran permanecer en memoria durante el tiempo necesario todos los parámetros requeridos, el valor de la función heurística sería el aprovechamiento máximo posible y es el valor de inicio que toma la función heurística en el algoritmo implementado. Este valor se calcula sumando el aprovechamiento máximo para cada uno de los parámetros multicitados del modelo, el cual a su vez se calcula:

1) Si el parámetro es permanente,
$AM_i = 4 * NI_i * DT_i/FN$

2) Si el parámetro es una expresión intermedia,
$AM_i = (4 * NI_i + 8) * DT_i/FN$
en donde:

AM_i es el aprovechamiento máximo para el parámetro i

NI_i es el número de expresiones que invocan como operando al parámetro i

DT_i es el número de valores teóricos del parámetro i

FN es un factor de normalización cuyo valor se calcula para asegurar que el aprovechamiento máximo total sea un valor menor o igual a 64,000; con lo cual se logra almacenar el resultado de estos cálculos en variables enteras sin signo (lo cual ocupará sólo dos bytes de memoria).

El hecho de que la función heurística se inicie con el máximo valor posible de aprovechamiento, y que la función se penalice con costos de reproceso perfectamente determinables en cada alternativa, asegura que los valores que toma la función heurística siempre estarán en el rango límite de frontera que guía al algoritmo inequívocamente al óptimo global.

En el caso de un proceso de maximización como el presentado, los valores que toma la función heurística en las alternativas que pertenecen a la ruta óptima, siempre serán mayores o iguales que los valores de las alternativas que no conforman la solución óptima, con esta condición satisfecha, se asegura que el algoritmo A* es "admisible" para el problema y función heurística planteados. Ver Aho [2].

Los nodos de la gráfica que representan los diversos estados del proceso de decisión contienen:

- La expresión que se está evaluando.
- Un vector de bits que representa a los parámetros que están presentes en memoria al inicio de la evaluación de la expresión.
- Un vector de bits que representa a los parámetros que permanecerán en memoria, al término de la evaluación de la expresión.
- Un apuntador al estado antecesor.
- El valor de la función heurística.

4.5.3 PROCEDIMIENTO DE OPTIMIZACION

El procedimiento para determinar qué parámetros permanecerán en memoria al término de la evaluación de cada expresión, se puede resumir en los siguientes pasos:

PASO 1 Se inicializan las siguientes variables de control del proceso.

Función a Evaluarse	<-	1
Estado Padre	<-	NULO
Vector Salida del Padre	<-	Ceros
Función Heurística Padre	<-	Máximo Valor de Aprovechamiento

2 Se apunta la primera expresión.

Para la expresión apuntada:

3 Se expanden todas las combinaciones posibles de todos los parámetros que deberían permanecer en memoria, eliminando aquellos que no están en el Vector Salida del Padre y que no son invocados como operandos de la expresión.

4 Se eliminan aquellas alternativas que no caben en la memoria.

5 Se calcula la función heurística para cada alternativa resultante y éstas se clasifican en orden descendente.

6 Para cada una de las alternativas:

se arma la descripción del estado con los siguientes valores:

- La expresión que se está evaluando	<-	Función a Evaluarse
- Vector de Parámetros de Entrada	<-	Vector Salida del Padre OR Vector de Parámetros multicitados, invocados por la función a evaluarse
- Vector de Parámetros de Salida	<-	Alternativa Seleccionada
- Apuntador al Estado Antecesor	<-	Estado Padre
- Valor de la Función Heurística	<-	La calculada en el paso 5

y se intercalan en la <cola de nodos a expandirse>, en el orden definido en el paso 5.

7 Se extrae de la <cola de nodos a expandirse> el primer nodo.

Si la expresión del nodo extraído es igual a la última expresión del modelo, se va al paso 8 (se encontró el último nodo del recorrido óptimo).

Se actualizan los valores del Vector Salida del Padre, y Función Heurística del Padre, con los valores del nodo extraído. En Estado Padre se almacena el apuntador al nodo extraído. La Nueva Función a Evaluarse = Función a Evaluarse del nodo extraído + 1.

Se incluye el nodo extraído en el <vector de nodos expandidos>.

Se apunta a la expresión contenida en Función a Evaluarse.

Se va al paso 3.

8 Iniciando con el nodo extraído de la <Cola de nodos a expandirse>, se integra la ruta óptima, recorriendo aquellos nodos ancestros, (almacenados en el <vector de nodos expandidos>), que son señalados mediante el apuntador que para tal efecto se definió como parte de los nodos.

9 Finalmente se integra nuevamente la matriz de relaciones entre parámetros/expresiones multicitados y las expresiones del modelo, con las permanencias de los parámetros definidas en la solución óptima. La estructura de la matriz es la ya expuesta en la segunda etapa.

4.6 ASIGNACION DE DIRECCIONES DE MEMORIA

La cuarta etapa tiene como objetivo el asignar una localización de memoria a cada uno de los parámetros multicitados, tratando de que ésta siempre sea la misma a lo largo del proceso de cálculo, si esto no es posible, la finalidad de la presente etapa es minimizar el proceso de reacomodo de dichos parámetros.

Como resultado de la etapa, se completa la definición de la secuencia de eventos para los parámetros multicitados, con la localización de memoria donde residirán en la evaluación de cada una de las expresiones.

El procedimiento se inicia identificando para cada parámetro, las secuencias consecutivas de expresiones en donde el parámetro

debe permanecer en memoria. Cada secuencia se etiqueta con el número de expresiones contenidas.

A cada combinación parámetro-secuencia consecutiva, se le va asignando en un orden descendente, respecto al número de expresiones contenidas, una localización que cumple con las siguientes propiedades:

1) En la dirección asignada se inicia un espacio no ocupado, cuyo tamaño permite almacenar al conjunto de valores teóricos del parámetro.

2) La dirección asignada es la dirección más baja posible, que permite la misma localización para el parámetro, en las evaluaciones de cada una de todas las expresiones que lo invocan en la secuencia.

Debido a que este procedimiento va dejando espacios no ocupados de memoria, se hace un proceso de eliminación de éstos, únicamente para aquellas expresiones donde se rebasa la memoria permitida.

4.7 ARMADO DE LA PILA DE EJECUCION DE LAS EXPRESIONES

En la quinta etapa se establece la secuencia de cálculo de las operaciones que integran cada una de las expresiones. La secuencia que se define es aquella que requiere la mínima cantidad de memoria para la evaluación de cada expresión.
Asimismo, en esta etapa se asignan las localizaciones de memoria para el resto de parámetros, (aquellos que son invocados una sola vez, en el conjunto de expresiones del modelo).

Como resultado de esta etapa se obtiene una pila de ejecución, donde se indica la secuencia de lectura y almacenamiento de parámetros, el cálculo de operandos, y la escritura a disco de los resultados.

La gráfica acíclica, integrada en la segunda etapa y etiquetada en la tercera con la memoria requerida para evaluar cada rama de la gráfica, es recorrida evaluando en primer lugar las ramas que requieren mayor memoria, conforme al multicitado algoritmo en Aho [2].

La pila de ejecución para la evaluación de la expresión, así como la asignación de las direcciones de los parámetros no multicitados, se definen en este mismo recorrido:

El procedimiento se inicia asignando al resultado de la expresión (nodo raíz), la primera dirección disponible de la pila de memoria si la expresión es una de las independientes, si la expresión es una de las interrelacionadas, se asigna como dirección de la expresión, la calculada en la cuarta etapa.

TABLA 3

RECORRIDO Y PROCESO DE CADA NODO

¿El nodo corriente representa a: ?	M	P	P	E	E	E	E	E	E	E	E	E	E	E	E	E
¿La dirección asignada al nodo = cero?		N	S													
¿El operando es una sumatoria?				S	S	S	N	N	N	N	N	N	N	N	N	N
¿El operando izquierdo es:?				M	P	E	E	E	E	M	P	M	M	P	P	P
¿El operando derecho es:?							M	P	E	E	E	M	M	P	P	P
¿La memoria requerida para evaluar la rama izquierda es igual o mayor que la memoria requerida para evaluar la rama derecha?								S	N							
¿El dominio teórico del parámetro izquierdo es igual o mayor que el dominio teórico del parámetro derecho?														S	N	
Incluye en la pila de ejecución la orden de lectura del parámetro.	X	X														
Asigna como dirección del nodo izq.:																
la dirección calculada en la 4a etapa										X		X	X			
la dirección disponible en la pila de memoria.					X	X	X	X	X							X
un cero.				X							X			X	X	
Actualizar la dirección disponible en la pila de memoria sumando el dominio teórico del operando izquierdo.					X	X	X	X	X							
Asigna como dirección del nodo der.:																
la dirección calculada en la 4a etapa							X					X		X		
la dirección disponible en la pila de memoria.									X	X	X	X			X	
un cero.								X					X			X
Actualizar la dirección disponible en la pila de memoria sumando el dominio teórico del operando derecho.									X	X	X	X				
Asigna como dirección del nodo izq.:																
la dirección calculada en la 4a etapa																
la dirección disponible en la pila de memoria.										X						
un cero.																
Actualizar la dirección disponible en la pila de memoria sumando el dominio teórico del: operando izquierdo.										X						
operando derecho.																
Recorre Rama Izquierda.				X	X	X	X	X	X				X	X	X	X	X
Recorre Rama Derecha.							X	X	X	X	X	X	X	X	X	X	X
Recorre Rama Izquierda.										X	X	X					
Actualiza la dirección disponible en la pila de memoria restando el dominio teórico del: operando izquierdo.					X	X	X	X	X								
operando derecho.									X	X	X	X					
operando izquierdo.										X							
Incluye en la pila de ejecución:																	
La definición del operando izquierdo.				X	X	X	X	X	X	X	X	X	X	X	X	X	X
La definición del operando derecho.							X	X	X	X	X	X	X	X	X	X	X
La ejecución del operador.				X	X	X	X	X	X	X	X	X	X	X	X	X	X
Regresar.	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X

El recorrido y proceso de cada nodo se resume en la Tabla 3, en donde el símbolo M indica que el resultado del nodo es un parámetro multicitado ya calculado, el símbolo P indica que el nodo representa un parámetro básico, y el símbolo E indica que el nodo es una expresión intermedia. Los nodos tipo M y P son nodos terminales.

La pila de ejecución la constituyen las siguientes instrucciones básicas:

Lectura de parámetros, instrucción que ordena la lectura, desde disco, de un parámetro y su almacenamiento en una posición definida de memoria.

Definición del operando izquierdo, mediante esta instrucción se definen las posiciones de memoria ocupadas por el operando izquierdo.

Definición del operando derecho, define las posiciones de memoria del operando derecho.

Escribe a disco, define el nombre del parámetro que se escribirá a disco y la posición de memoria en donde se encuentra.

Ejecuta operación, define la operación a ejecutarse sobre los operandos anteriormente definidos (+,-,*,/, etc.) y define las direcciones de memoria donde se almacenará el resultado de la operación.

Mover Parámetro define un reacomodo en memoria del parámetro indicado en la operación.

4.8 CALCULO DE LAS EXPRESIONES

Finalmente, en la sexta etapa se realiza el cálculo de las expresiones, respetando la pila de ejecución desarrollada en la etapa anterior.

4.9 REFERENCIAS

[1] Aho, Alfred; Sethi, Ravi; Ullman, Jeffrey
Compilers; principles, techniques and tools
Addison Wesley Publishing Co.
1985
págs. 546-552

[2] Ibídem, págs. 559-562

[3] Nilsson J., Nils
Problem-Solving Methods in Artificial Intelligence
McGraw-Hill
New York, New York
1971
págs. 53-71

[4] Ibídem, págs. 59-61

[5] Tanimoto L., Steven
The Elements of Artificial Intelligence
Computer Science Press
Rockville, Maryland
1987
págs. 148-164

[6] Barr, Avron; Feigenbaum A., Edward
The Handbook of Artificial Intelligence
Addison Wesley
1981
Vol. 1, págs. 58-67

[7] Ibídem, pág. 65

[8] Gelperin, D.
On the optimality of A*
Artificial Intelligence
1977
Vol. 8, págs. 69-76

[9] Schutzer, Daniel
Artificial Intelligence
An Applications-Oriented Approach
Van Nostrand Reinhold
New York
1987
págs. 72-76

5 PROGRAMAS Y ARCHIVOS DEL SISTEMA

En el presente capítulo se describen, en forma resumida, los programas y archivos que constituyen el sistema, con el objeto de complementar la visión total del mismo.

5.1 DESCRIPCION DE LOS PROGRAMAS.

El sistema está constituído por veintidós módulos agrupados en nueve programas que se describen a continuación.

5.1.1 PROGRAMA GEMOLI01

El objetivo del programa GEMOLI01 es verificar la sintaxis de los postulados que definen el modelo e integrar la definición de los elementos del mismo.

El GEMOLI01 está integrado por los "overlays" GEMOLI01 y GMLI01_1.

En el módulo GEMOLI01 se lee la definición del modelo y se realiza el rompimiento de los postulados en sus elementos atómicos ("TOKENS"); con estos elementos se genera el archivo de trabajo ELEMENTO, el cual es leído en la segunda parte del programa.

En el módulo GMLI01_1 se analiza la sintaxis de los postulados que integran la definición del modelo, se informan los errores encontrados y se integran los archivos <modelo>.MOD, <modelo>.VAL y <modelo>.TXT con los elementos del modelo y sus características, los valores numéricos y los valores alfanuméricos de la definición, respectivamente.

Se genera además el archivo MENSAJE, el cual es una bitácora de los mensajes de error desplegados en la pantalla. Este archivo contiene también una impresión de los tres archivos anteriormente descritos.

5.1.2 PROGRAMA GEMOLI02

El objetivo del programa GEMOLI02 es obtener una secuencia de cálculo de las expresiones del modelo, que minimice las lecturas y escrituras a disco de los parámetros y/o cálculos intermedios requeridos por las expresiones.

El GEMOLI02 está integrado por el módulo raíz GEMOLI02, el módulo GMLI02VG y los "overlays", GMLI02_0, GMLI02_1, GMLI02_2 y GMLI02_3.

En el módulo raíz GEMOLI02 se controla la ejecución del proceso.

El módulo GMLI02VG contiene las definiciones de las variables globales del programa y los procedimientos comunes.

El módulo GMLI02_0 es el proceso de inicialización de la carga de "overlays".

En el módulo GMLI02_1 se identifican todos los parámetros, conjuntos y expresiones del modelo, y se integra la gráfica acíclica dirigida con todas estas expresiones.

El módulo GMLI02_2 lo constituye el algoritmo que optimiza la secuencia de cálculo de las expresiones, asignando los espacios de memoria para realizar los cálculos, así como la decisión de cargar y/o descargar de memoria, cada parámetro involucrado en el modelo.

El módulo GMLI02_3 arma los archivos: <modelo>.PRM con las características de los parámetros básicos y de las expresiones; así como el archivo <modelo>.STK con el stack de ejecución de las expresiones.

5.1.3 PROGRAMA GEMOLI03

El objetivo del programa GEMOLI03 es realizar, en base al "stack" de ejecución generado en el programa GEMOLI02, el cálculo de las expresiones involucradas en el modelo.

El programa está integrado por el módulo raíz GEMOLI03, el módulo GMLI03VG, y los "overlays" GMLI03_0, GMLI03_1, GMLI03_2 y GMLI03_3.

El GEMOLI03 es el módulo raíz en donde se lleva el control de la ejecución del programa.

El módulo GMLI03VG contiene las definiciones de las variables globales del programa y de los procedimientos comunes.

El módulo GMLI03_0 es de el proceso de inicialización de carga de "overlays".

En el módulo GMLI03_1 están los procesos de expansión y comparación de los espacios de aplicación de los operandos de las expresiones, así como las rutinas de escritura a disco de los valores de los parámetros calculados.

En el módulo GMLI03_2 están los proceso de lectura de parámetros básicos.

El módulo GMLI03_3 está integrado por todo el proceso de cálculo de operadores.

En este programa se generan los archivos: <modelo>.EXP que contiene información relativa a las expresiones y a los parámetros del modelo, el archivo <modelo>.VCi (en donde i puede ir de 1 a 6) contiene los valores de los parámetros calculados, finalmente el archivo <modelo>.VXi (i puede ir de 1 a 6) contiene el vector de bits que indican la existencia o no del valor de cada elemento de los conjuntos parámetro.

5.1.4 PROGRAMA GEMOLI04

El objetivo del programa GEMOLI04 es identificar y expandir los elementos de la matriz del modelo.

El programa GEMOLI04 está integrado por el módulo raíz GEMOLI03 y los "overlays" GMLI04_1 y GMLI04_2.

En el módulo GEMOLI04 se expanden los dominios de cada elemento del modelo, analizando simultáneamente la consistencia entre dominios y subdominios, el resumen de estas expansiones se almacena en el archivo <modelo>.CMP.

En el módulo GMLI04_1 se resuelve y se define la equivalencia entre las restricciones y las hileras de la matriz.

Las funciones que se realizan para lograrlo son: el análisis de la consistencia entre las restricciones, las subrestricciones y los dominios de las mismas; encontrar los valores límites y los rangos asociados a cada restricción; expandir los elementos de las restricciones.

Como resultado de este módulo se genera el archivo <modelo>.RST con todos los datos de las restricciones necesarias para identificar las hileras asociadas.

En el módulo GMLI04_2 se resuelve y se definen las equivalencias entre variables, coeficientes, restricciones e hileras, para lo cual: se analiza la consistencia de las ecuaciones, se avisa de los errores encontrados, y se buscan todos los coeficientes explícitos.

Como resultado se genera el archivo <modelo>.MTZ con los datos relevantes de la relación Variable-Coeficiente-Restricción.

5.1.5 PROGRAMA GEMOLI05

El programa GMOLI05 es un proceso de clasificación de los archivos <modelo>.RST y <modelo>.MTZ. Los archivos clasificados se graban en los mismos archivos de entrada.

5.1.6 PROGRAMA GEMOLI06

El programa GEMOLI06 imprime el informe de equivalencias entre restricciones e hileras.

5.1.7 PROGRAMA GEMOLI07

El programa GEMOLI07 imprime el informe de las equivalencias entre variables y columnas, así mismo, se genera el archivo <modelo>.MDL con la matriz de entrada al simplex, grabada en formato MPS.

Cuando el paquete de uso definido es IBM o MPSX, este programa genera, además, el programa de control para el paquete MPSX; y en el caso de que se declare como paquete IBM, se transcribe e intercala al principio del archivo mencionado, el contenido del archivo <modelo>.TCI, en donde se deben encontrar las tarjetas de control para ejecutar el trabajo.

5.1.8 PROGRAMA GEMOLI09

El programa GEMOLI09 lee el archivo con la solución del modelo obtenida del paquete XA y decodifica los índices de las variables y restricciones del modelo, intercalando en el archivo, campos conteniendo el significado particular de estos índices.

5.2 MATRIZ ARCHIVOS-PROGRAMAS

ARCHIVOS	PROGRAMAS GEMOLI								PAQUETE DE PROGRAMACION LINEAL
	01	02	03	04	05	06	07	09	
DEF. DEL MODELO	C								
ARCHIVOS DE DATOS			C	C					
ELEMENTO	T								
MODELO.CMP				G		C	C	C	
.DMR				T					
.EXP			G						
.MOD	G	C	C	C		C	C	C	
.MDL							T		
.MPS							G		C
.MSG	G								
.MTZ				G	A		C		
.PRM		G	C						
.RNG									
.RST				G	A	C	C		
.STK		G	C						
.TCI							C		
.TXT	G	C	C	A		C	C	C	
.VAL	G	C	C	C		C	C	C	
.VCi			G	C					
.VXi			G	C					
SOLUCION DEL MODELO								C	G
SOLUCION DEL MODELO DECODIFICADA								G	

SIMBOLOGIA:
A Consulta y actualiza el archivo.
C Consulta el archivo.
G Genera el archivo.
T Archivo de trabajo.

6 EVALUACION DEL SISTEMA Y CONCLUSIONES

El presente capítulo presenta una evaluación del comportamiento del sistema, así como algunas características cuantificables del mismo y de los modelos que se han utilizado como ejemplos y como base para obtener las mediciones que se analizan.

6.1 DIMENSIONES DE LOS PROGRAMAS Y DE LOS MODELOS EJEMPLO

En la Tabla 4 se presentan para cada programa, la suma de las líneas de código que conforman todos los módulos y/o "overlays", así como la suma del tamaño en bytes de los archivos ejecutables de estos mismos módulos.

A pesar de que los módulos de los programas GEMOLI01 y GEMOLI04, que están en lenguaje BASIC, y por lo tanto no son comparables directamente con el resto de programas que están en TURBO PASCAL, el tamaño de los archivos ejecutables representan fielmente la complejidad de las funciones que se realizan en cada uno de ellos.

En la Tabla 5 se presentan el número de instrucciones, variables y coeficientes válidos, que conforman cada uno de los modelos.

Para efecto de la evaluación, se encontró que el número de coeficientes representa satisfactoriamente la complejidad y tamaño de los modelos, por tal motivo, se escogió este parámetro para compararlo con los tiempos de proceso (ver GRAFICA 1).

TABLA 4

DIMENSIONES DE LOS PROGRAMAS

PROGRAMA	TOTAL DE LINEAS DE CODIGO	TAMAÑO EN BYTES DE LOS ARCHIVOS EJECUTABLES
GEMOLI01	3012	98,226
GEMOLI02	3525	185,216
GEMOLI03	4937	265,840
GEMOLI04	4260	126,739
GEMOLI05	234	14,576
GEMOLI06	528	16,560
GEMOLI07	1702	36,544
GEMOLI09	1078	23,456

TABLA 5

DIMENSIONES DE LOS EJEMPLOS

EJEMPLO	MODELO	RESTRICCIONES	VARIABLES	COEFICIENTES
1	TRANSP	6	6	18
2	MEZCLA	8	7	48
3	FLUJO_MX	14	9	25
4	ASIGN	8	12	36
5	VIAJE	22	48	219
6	COMBUS65	403	771	3043

6.2 TIEMPOS DE EJECUCION DEL SISTEMA

En la Tabla 6 se presenta bajo el rubro "SISTEMA GEMOLI" un resumen con la suma de los tiempos que tardaron los programas del sistema, en generar la matriz de entrada al paquete XA, para cada uno de los ejemplos del apéndice B; bajo el rubro "GEMOLI+XA" se incluye además, el tiempo que tardó el paquete XA en encontrar la solución del modelo.

Es importante señalar que estos tiempos fueron obtenidos al ejecutar el sistema en un computador personal HP con procesador Intel 80386 de 16 MHz.

En la Tabla 7 se presenta el tiempo de proceso de cada programa, como porcentaje del tiempo total requerido para generar la matriz y resolver el modelo (rubro GEMOLI+XA).

En esta distribución de tiempos se puede observar:

- que los programas GEMOLI06 y GEMOLI07 presentan tiempo de proceso mucho mayores que el resto de programas, lo cual es debido a la impresión que se realiza en estos programas, de las equivalencias entre restricciones e hileras, y entre variables y columnas.

- que la distribución de tiempos es homogénea en los diversos ejemplos, a excepción del ejemplo 6, en donde a diferencia de los anteriores, se realizan cálculos de parámetros.

Para poder comparar los tiempos de proceso requeridos por cada modelo ejemplo, en la Tabla 8 y en la Gráfica 1 se presentan los tiempos normalizados de proceso del sistema GEMOLI por coeficiente generado.

Los coeficientes considerados fueron aquellos cuyo valor fue diferente de cero, en la matriz MPS generada.

Con estos consumos unitarios se puede observar claramente en la Gráfica 1, que a mayor tamaño del modelo, el proceso unitario se va reduciendo.

La flecha, en la parte inferior derecha de la Gráfica, representa el tiempo de proceso por coeficiente generado, requerido para el modelo del ejemplo 6, en donde además de que la matriz del modelo contiene 3043 coeficientes válidos, lo cual lo hace el modelo más grande, también es el más complejo, ya que la generación de este modelo incluye el cálculo de expresiones de parámetros y de conjuntos, así como un amplio uso de estos últimos.

El tiempo unitario de proceso para el ejemplo 6, confirma que el sistema mejora su comportamiento conforme aumenta el tamaño y complejidad del modelo que se está resolviendo, con lo cual se cumple ampliamente la premisa principal de diseño, enunciada en el Capítulo 1, referente a que "El sistema deberá manejar eficientemente grandes modelos, ...".

TABLA 6

TIEMPOS DE EJECUCION DE LOS EJEMPLOS (SEGUNDOS)

EJEMPLO	1	2	3	4	5	6
SISTEMA GEMOLI	73	85	75	69	94	776
GEMOLI+XA	94	112	97	90	131	1276

TABLA 7

DISTRIBUCION DE LOS TIEMPOS DE EJECUCION DE LOS EJEMPLOS

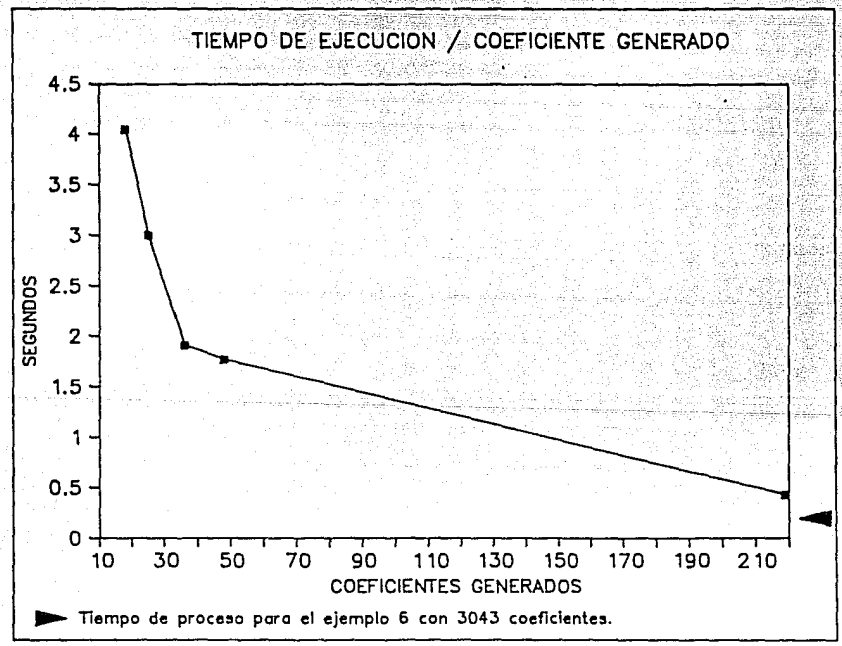
	(CIFRAS EN PORCENTAJES)					
EJEMPLO	1	2	3	4	5	6
PROGRAMA						
GEMOLI01	22.3	20.5	21.6	24.4	17.5	4.4
GEMOLI02	1.0	.8	1.0	2.2	1.5	.2
GEMOLI03	-	-	-	-	-	.6
GEMOLI04	6.3	6.2	7.2	7.7	7.6	8.2
GEMOLI05	6.3	7.1	7.2	7.7	5.3	2.6
GEMOLI06	12.7	13.3	13.4	12.2	13.7	18.6
GEMOLI07	26.0	24.5	24.0	20.0	24.4	25.8
PAQUETE XA	22.3	24.1	22.6	23.6	28.5	39.2
GEMOLI09	3.1	3.5	3.0	2.2	1.5	.4
TOTAL %	100.0	100.0	100.0	100.0	100.0	100.0

TABLA 8

TIEMPOS DE EJECUCION POR COEFICIENTE GENERADO (SEGUNDOS)

EJEMPLO	MODELO	NUMERO DE COEFICIENTES	TIEMPO DE EJECUCION DE GEMOLI TOTAL	TIEMPO DE EJECUCION DE GEMOLI POR COEFICIENTE
1	TRANSP	18	73	4.05
2	MEZCLA	48	85	1.77
3	FLUJO_MX	25	75	3.00
4	ASIGN	36	69	1.91
5	VIAJE	219	94	.42
6	COMBUS65	3043	776	.25

GRAFICA 1



6.3 EFICIENCIA DEL CALCULO DE EXPRESIONES.

Para evaluar el algoritmo de optimización del cálculo de expresiones (Capítulo 4), se procedió a definir un modelo en el cual se invocara en múltiples ocasiones la expresión (A(I)+B(I))/C(I).

Se ejecutó el sistema y se tomaron los tiempos del proceso de optimización (T_o) y del proceso propio de cálculo de las expresiones (T_c), para 2, 3 y 4 invocaciones de la expresión ya mencionada, variando simultáneamente los límites del índice I a los siguientes valores:

10, 50, 500, 1000, 3000, 5000, 7000

El proceso se repitió obligando al sistema a calcular las expresiones (T_{cno}) sin ejecutar el algoritmo de optimización.

Para cada instancia se tomaron tres mediciones y el promedio de éstas se presenta en la Tabla 9.

El ahorro de tiempo debido al proceso de optimización se definió como la diferencia entre el tiempo requerido para calcular las expresiones cuando se inhibió el algoritmo de optimizaciones (T_{cno}) y el tiempo requerido por el algoritmo de optimización (T_o), más el tiempo requerido para el cálculo de los parámetros, cuando el "stack" de ejecución fue optimizado (T_c).

$$\text{Ahorro(tiempo)} = T_{cno} - (T_o + T_c)$$

Para hacer comparables los diferentes casos, se tomó como unidad de evaluación

$$\text{Ahorro(\%)} = \frac{T_{cno} - (T_o + T_c)}{T_{cno}} * 100$$

que es el ahorro en términos porcentuales del tiempo requerido para los cálculos, cuando éstos se realizaron inhibiendo la utilización del algoritmo de optimización.

Los ahorros de tiempo en porcentaje están representados en la Gráfica 2 para las diversas instancias medidas. En esta gráfica se pueden observar dos zonas de comportamiento del ahorro, la zona de respuesta rápida, comprendida entre 0 y 500 elementos por conjunto parámetro, en donde rápidamente se pasa de pagar un sobreprecio (ahorro negativo) por utilizar el algoritmo de optimización, en modelos con pocos elementos por parámetro, a ahorros significativos para modelos con un número modesto de elementos por conjunto parámetro; en la zona de lenta respuesta se puede observar que la variación del ahorro de tiempo va disminuyendo conforme aumenta el número de elementos de los parámetros calculados.

Este mismo comportamiento se puede observar si se analiza la variación del ahorro con respecto al número de invocaciones, el incremento logrado al pasar de 3 a 4 invocaciones es menor que el incremento logrado al pasar de 2 a 3 invocaciones.

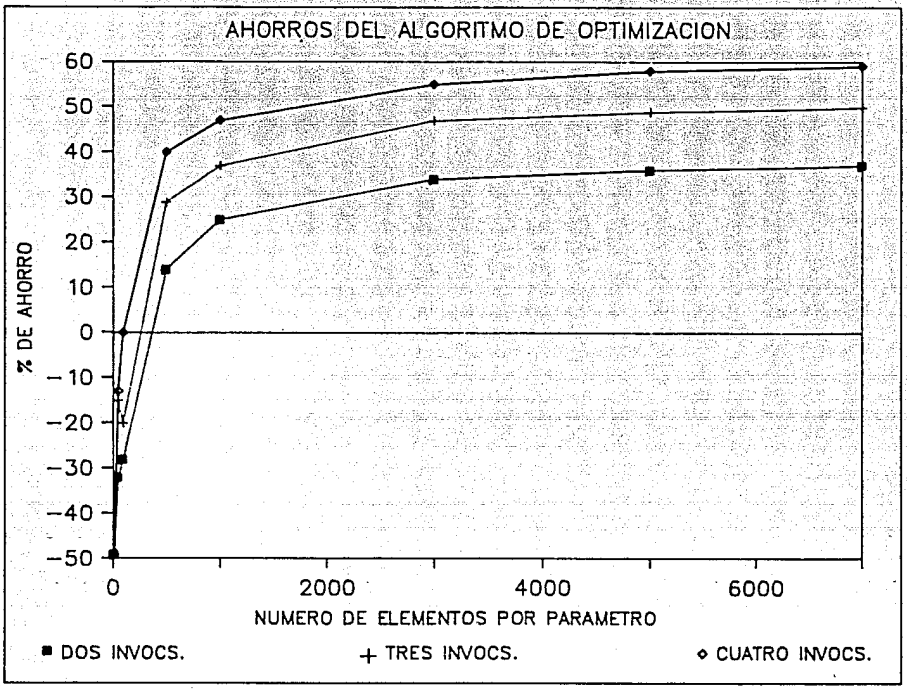
Tanto el ahorro porcentual como el tiempo de ejecución / coeficiente generado tienden, cada uno, a un límite. Este límite representa el tiempo directamente asociable al tamaño del modelo; en este punto extremo, los tiempo "fijos" consumidos en el análisis del modelo y generación de matrices asociadas, así como en el análisis de expresiones, optimización de su cálculo y generación del "stack" de ejecución se vuelven despreciables, con lo cual nuevamente se confirma que el sistema mejora su rendimiento conforme es mayor el modelo analizado.

TABLA 9

TIEMPOS DE PROCESO DEL CALCULO DE PARAMETROS
(SEGUNDOS)

NUMERO DE INVOCACIONES DEL PARAMETRO	NUMERO DE ELEMENTOS POR PARAMETRO	(T_o)	(T_c)	(T_o+T_c)	(T_{cso})	AHORRO (%)
2	10	1.87	2.01	3.88	2.23	-74
2	50	1.43	2.40	3.83	2.89	-32
2	100	2.20	2.53	4.73	3.70	-28
2	500	2.23	7.12	9.35	10.88	14
2	1000	2.22	12.07	14.29	19.10	25
2	3000	2.14	32.46	34.60	52.16	34
2	5000	2.12	52.68	54.80	86.18	36
2	7000	2.18	73.53	75.71	120.30	37
3	10	1.92	1.77	3.69	2.08	-77
3	50	1.36	2.27	3.63	3.17	-15
3	100	2.25	3.04	5.29	4.42	-20
3	500	2.27	8.07	10.34	14.48	29
3	1000	2.63	14.13	16.76	26.55	37
3	3000	2.22	37.70	39.92	75.67	47
3	5000	2.25	61.98	64.23	125.75	49
3	7000	2.20	85.38	87.58	175.56	50
4	10	1.92	2.23	4.15	2.78	-49
4	50	1.50	2.80	4.30	3.82	-13
4	100	2.27	3.59	5.86	5.88	0
4	500	2.27	9.35	11.62	19.44	40
4	1000	2.31	16.37	18.68	35.47	47
4	3000	2.23	43.73	45.96	101.15	55
4	5000	2.25	70.83	73.08	175.95	58
4	7000	2.27	97.97	100.24	245.72	59

GRAFICA 2



6.4 DIMENSIONES MAXIMAS DE MODELOS SOPORTADOS

El sistema GEMOLI puede manejar todo aquel sistema que no exceda las siguientes dimensiones:

30,000 elementos por función; este límite se refiere al número máximo de elementos del espacio teórico de aplicación de cada parámetro, variable o restricción.

La presente versión soporta hasta 200 postulados por modelo, es decir, el número de índices definidos, más el número de variables declaradas, más el número de restricciones, etc., no debe exceder de 200.

Asimismo el sistema, como su nombre lo indica, maneja únicamente ecuaciones lineales; la presente versión no tiene instrumentado ningún proceso de linearización.

Hemos estimado que estas características son suficientes para resolver la mayoría de los modelos requeridos en el ejercicio profesional.

6.5 CONCLUSIONES

6.5.1 USO DEL LENGUAJE GEMOLI

Este sistema se ha estado utilizando en Petróleos Mexicanos en las Gerencias de Planeación de las divisiones Comercial y Petroquímica. En la primera ha servido para evaluar diversas alternativas de inversión en el sistema de distribución de la empresa. En el área de Petroquímica se ha utilizado para evaluar múltiples alternativas de localización de plantas productoras de petroquímicos básicos.

En ambas áreas se ha comprobado que las características y la especialización del lenguaje GEMOLI para definir modelos lineales han permitido desarrollar modelos en cuestión de semanas, cuando esta actividad anteriormente requería de meses.

El notable incremento en la capacidad de desarrollo de este tipo de modelos se ha debido a varias causas:

La primera, y tal vez la más importante, es la facilidad del lenguaje para manejar modelos complejos, lo cual queda plenamente demostrado si se analiza el ejemplo 6 del apéndice B.

Como segunda causa se puede mencionar que el lenguaje GEMOLI se ha convertido en el medio sintético de comunicación de ideas y cambios sobre los modelos, entre los participantes de los grupos desarrolladores, inclusive entre equipos de trabajo que manejan diferentes paquetes de programación lineal.

Finalmente es importante resaltar que la capacidad de GEMOLI para la creación de prototipos de modelos mediante la manipulación de los límites de los índices, (se genera el conjunto completo de ecuaciones pero sólo se utiliza un subconjunto de datos), ha permitido definir y afinar los modelos en tiempos de proceso muy reducidos.

6.5.2 COMPARACION CON OTROS LENGUAJES

Algunos sistemas de programación lineal que integran tanto

facilidades para definir el modelo, como el algoritmo simplex para la solución de los mismos, (LINDO [1], EUREKA [2], XA [3]), aceptan directamente, como una alternativa de definición del modelo, las ecuaciones desarrolladas del mismo. Estas ecuaciones están integradas por cada una de las variables, por los coeficientes y por las limitantes del modelo, razón por la cual, son eficientes sólo para modelos muy pequeños, ya que para modelos grandes estas ecuaciones se vuelven inmanejables por su longitud. GEMOLI, por el contrario, utiliza ecuaciones en donde cada elemento de las ecuaciones es un conjunto que puede tener, en la matriz expandida, hasta 30,000 elementos .

En comparación con el AMPL [4], existen cuatro diferencias fundamentales:

- Manejo de archivos. En AMPL no se define la lectura de datos de archivos ni la escritura en disco de los resultados.

- Manejo de reglas de consistencia. AMPL tampoco define un mecanismo para verificar la consistencia entre los coeficientes, los límites y los rangos.

- AMPL es un sistema que integra tanto los medios para la definición del modelo, como los algoritmos para la solución de los mismos. GEMOLI sólo produce la entrada a los paquetes de solución.

- Finalmente, aún cuando la notación es distinta, ambos cumplen satisfactoriamente con las capacidades y facilidades requeridas para una definición apropiada de los modelos.

GEMOLI es una buena alternativa cuando ya se cuenta con algunos de los paquetes para los cuales GEMOLI genera entrada.

6.5.3 RESUMEN

En resumen, se puede afirmar que GEMOLI es un lenguaje poderoso que permite en forma conceptual y directa, describir concisamente modelos lineales.

Que las facilidades del lenguaje permiten: resolver modelos cada vez más complejos, así como reducir los tiempos de desarrollo, mismos que resultan bastante menores a los tradicionales.

Asimismo, se puede afirmar que el sistema implementado para la interpretación del lenguaje es eficiente y de fácil uso.

6.6 REFERENCIAS

[1] LINDO
Linear Interactive and Discrete Optimizer
LINDO Systems, Inc.

[2] EUREKA: THE SOLVER
Borland International
Scotts Valley, California 95066

[3] XA
PROFESSIONAL LINEAR PROGRAMMING SYSTEM
Sunset Software Technology
San Marino, California 91108

[4] Fourer, Robert; Gay David; Kernighan, Brian
AMPL: A Mathematical Programming Language
Computing Science Technical Report No. 133
AT&T Bell Laboratories
Murray Hill, N. J.
1987

A SINTAXIS DEL LENGUAJE GEMOLI

A.1 SIMBOLOS DEL METALENGUAJE BNF

« » Paréntesis que delimitan un símbolo NO TERMINAL.

┤ ├ Paréntesis que delimitar una concatenación de símbolos.

| Separador de las diferentes producciones posibles para un símbolo no terminal.

::= Define las producciones posibles de un símbolo no terminal.

« »? ┤ ├? Define que el símbolo o concatenación de símbolos entre los parentesís pueden no ocurrir u ocurrir una sola vez.

« »* ┤ ├* Define que el símbolo o concatenación de símbolos entre los paréntesis pueden no ocurrir u ocurrir multiples veces.

« »+ ┤ ├+ Define que el símbolo o concatenación de símbolos entre los paréntesis deben ocurrir cuando menos una vez.

A.2 SINTAXIS

«MODELO» ::=

«nombres»

┤ «índices» «subíndices»* ├+

«variables»+

«parámetros»+

«conjuntos»*

«función-objetivo»

┤ «restricciones» «subrestricciones»* ├+

«archivos-entrada»*

«reglas-consistencia»*

«valores»*

<FIN>

«nombres» ::=

<MOD> «nombre-modelo»

<PAQ> «nombre-del-paquete»

┤ <FO> «identificador-vector-objetivo» ├*

┤ <RS> «identificador-vector-restricciones» ├*

┤ <RG> «identificador-vector-rangos» ├*

┤ <FR> «identificador-vector-fronteras» ├*

«nombre-del-paquete» ::=

MPSX | APEX | TEMPO |

LP88 | MILP88 | XA

«índices» ::=

<IND> «identificador-índice», "«significado-general»",

«dominio-máximo»

┤ <SP> {«lista-significados-particulares»} |

<SP> «identificador-vector-significados-particulares» ├*

«lista-significados-particulares» ::=

«valor-del-índice», "«significado-particular»"

┤ ; «lista-significados-particulares» ├*

«subíndices» ::=

<IND> «identificador-subíndice», "«significado general»",

«dominio-máximo»

<SC> «identificador-índice-padre» «formación-dominio-subíndice»

«formación-dominio-subíndice» ::=

<INC> «elementos-dominio-subíndice» |

<EXC> «elementos-dominio-subíndice»

«elementos-dominio-subíndice» ::=

<IGD> «identificador-índice-equivalente» + {«lista-valores-dominio»} |

<IGD> «identificador-índice-equivelente» - {«lista-valores-dominio»} |

<IGD> «identificador-índice-equivalente» |

{«lista-valores-dominio»}

«variables» ::=

<VAR> «tipo-variable»⁷ «identificador-variable» («dominio-teórico»),

"«significado-general»"

┤ , <FRON_INF> «identificador-vector-frontera-inferior» ├⁷

┤ , <FRON_SUP> «identificador-vector-frontera-superior» ├⁷

«tipo-variable» ::= <ENT>| <BIN>| <REAL>

«parámetros» ::=

<PARAM> ┤ <ENT> ├⁷ ┤ «REDONDEADO» ├⁷

«identificador-parámetro» («dominio-teórico»),

"«significado-general»" ┤= («expresión-parámetros») ├⁷

«expresión-parámetros» ::=

SUM ((«dominio-sumatoria»), «expresión-parámetros») |

(«expresión-parámetros») |

«expresión-parámetros» «operador-aritmético» «expresión-parámetros»

|

«parámetro» |

«constante»

«dominio-sumatoria» ::= «dominio-t2»

```
«conjuntos» ::=
    <CONJ> «identificador-conjunto» («dominio-teórico»),
           "«significado-general»" «formación-conjunto»

«formación-conjunto» ::=
          = { «lista-n-tuplos» }   |
          = «identificador-vector-tuplos»   |
          = («exp-conjunto») |
     <COMPL> «conjunto» |
     <COMPL> { «lista-n-tuplos» }   |
     <COMPL> «identificador-vector-tuplos»   |
     <COMPL> «exp-conjunto»

«lista-n-tuplos» ::=
         «tuplo»+ ┤; «lista-n-tuplos» ├+

«tuplo» ::= «valor-del-índice» ┤, «valor-del-índice» ├+

«exp-conjunto» ::=
         («exp-conjunto»)   |
         «exp-conjunto» «operador-conjunto» «exp-conjunto» |
         «exp-relación» |
         «conjunto»   |

«exp-relación» ::=
         «exp-aritmética» «operador-relación» «exp-aritmética»

«exp-aritmética» ::=
         («exp-aritmética»)   |
         «exp-aritmética» «operador-aritmético» «exp-aritmética» |
```

SUM((«dominio-sumatoria»), «exp-aritmética») |

« parámetro» |

«índice» |

«constante»

«función-objetivo» ::= «restricciones»

«restricciones» ::=

<RES> «identificador-restricción» ┤ («dominio-teórico») ├?

"«significado-general»" «dominio-ecuación»

«dominio-ecuación» ::= «dominio-restricción» «ecuación» | «ecuación»

«dominio-restricción» ::= <DOM> («dominio-t2»)

«ecuación» ::=

<EC> «primer-sumando» «sumandos» «relación-limite»; «rango»?

«relación-límite» ::=

<MAX> «identificador-función-objetivo» |

<MIN> «identificador-función-objetivo» |

«relación» «límite»

«identificador-función-objetivo» ::=

«identificador-restricción»

«relación» ::= <IG> | <MAI> | <MEI>

«límite» ::= «constante» | «parámetro» | («expresión-parámetros»)

«rango» ::= «constante» | «parámetro» | («expresión-parámetros»)

«primer-sumando» ::= «sumando» | + «sumando» | - «sumando»

«sumandos» ::= + «sumando» | - «sumando»

«sumando» ::= SUM((«dominio-sumatoria»), «coeficiente-variable») |
«coeficiente-variable»

«coeficiente-variable» ::=
(«expresión-parámetros») * «variable» ┤ <ELE> «exp-conj» ├* |
«parámetros» * «variable» ┤ <ELE> «exp-conj» ├* |
«constante» * «variable» ┤ <ELE> «exp-conj» ├* |
«variable» ┤ <ELE> «exp-conj» ├*

«exp-conj» ::= «conjunto» | («exp-conjunto»)

«subrestricciones» ::=
<RES> «identificador-subrestricción» («dominio-teórico»),
"«significado general»"
<SC> «identificador-restricción-padre»
«formación-dominio-subrestricción» «ecuación»

«formación-dominio-subrestricción» ::=
<INC> «elementos-dominio-subrestricción» |
<EXC> «elementos-dominio-subrestricción»

«elementos-dominio-subrestricción» ::=
<IGD> «identificador-restricción-equivalente» + «dominio-restricción» |
<IGD> «identificador-restricción-equivalente» - «dominio-restricción» |
«dominio-restricción»

```
«archivos-entrada» ::=
        <ARCH_E> "«nombre-del-archivo»", «número-máximo-campos»
        <SEC> «lista-campos»+
              «datos-archivo»+

«lista-campos» ::=
              <CAMPO> («número-campo»)
              ┤, «lista-campos» ├ *

«datos-archivo» ::=
    <DATO>  «condicion-existencia» ?
            «identificador-elemento» («dominio-archivo»)
          = «exp-campos»     |
    <DATO>  «condición-existencia» ?
            «identificador-vector-significados-particulares»
            («índice-campo») = <CAMPO> («campo-de-significado»)    |
    <DATO>  «condición-existencia»?
            «identificador-vector-tuplos» («lista-indices-campos»)

«condición-existencia» ::=
    <SI> <CAMPO> («número-campo») <IG> «valor-en-campo»

«identificador-elemento» ::=
    «identificador-parámetro»    |
    «identificador-vector-frontera-inferior»    |
    «identificador-vector-frontera-superior»

«dominio-archivo» ::=  «lista-indices-campos-dominios»
```

«lista-índices-campos-dominios» ::=

«identificador-índice» = «índice-campo-dominio»

┤ ; «lista-índices-campos-dominio» ├*

«índice-campo-dominio» ::=

<CAMPO> («número-campo») | «dominio-índice»

«lista-índices-campos» ::=

«índice-campo» ┤ ; «lista-índices-campos» ├*

«índice-campo» ::=

«identificador-índice» = <CAMPO> («número-campo»)

«exp-campos» ::=

(«exp-campos») |

«exp-campos» «operador-aritmético» «exp-campos» |

<CAMPO> («número-campo») |

«constante»

«reglas-consistencia» ::=

«consistencia-variables» | «consistencia-parámetro-individual»

«consistencia-parámetros»

«consistencia-variables» ::=

<CONS> "«significado-general» " <SI> «identificador-variable»

┤ <ELE> «identificador-restricción» |

<NOELE> «identificador-restricción» ├?

<IMPLICA> <ELE> «identificador-restricción»*

```
«consistencia-parámetro-individual» ::=
  <CONS> "«significado-general»" <TODO>* «identificador-parámetro»
         «operador-relación»  «constante»

«consistencia-parámetros»::=
  <CONS> "«significado-general»" («expresión-parámetros»)
         «operador-relación»
         «constante»

«valores» ::=
  <VALOR> <elemento-t2-modelo> («dominio-t3») = {«lista-valores»}

«elemento-t2-modelo» ::=
        «parámetro»    |
        «identificador-vector-frontera-inferior»    |
        «identificador-vector-frontera-superior»

«lista-valores» ::=
        «lista-valores» ┤, «lista-valores» ├*
        «factor-repetición» * («lista-valores»)    |
        «factor-repetición» * «constante»    |
        «factor-repetición» NULO
        «constante»    |
        NULO

«parámetro» ::=
        «identificador-parámetro» («dominio-t1»)
```

«conjunto» ::=

«identificador-conjunto» («dominio-tl»)

«variable» ::=

«identificador-variable» («dominio-tl»)

«dominio-teórico» ::= «lista-ordenada-índices»

«lista-ordenada-índices» ::=

«índice» ┤ , «lista-ordenada-índices» ├*

«dominio-tl» ::= «lista-ordenada-dominiol-índice»

«lista-ordenada-dominiol-índice» ::=

«equivalencia-dominio-índice» ┤; «lista-ordenada-dominiol-índice» ├*

«equivalencia-dominio-índice» ::=

«índice»

«desplazamiento-índice»?

«equivalencia-índice»?

«formación-dominio-índice»

«desplazamiento-índice» ::=

+ «desplazamiento» | - «desplazamiento»

«equivalencia-índice» ::=

= «identificador-índice-equivalente»

«formación-dominio-índice» ::=

= «dominio-índice» | <DIF> «dominio-índice»

«dominio-índice» ::= «valor-entero-positivo» |
[«rango-inferior», «rango-superior»] |
{ «lista-valores-dominio» }

«dominio-t2» ::= «lista-ordenada-dominio2-índice»

«lista-ordenada-dominio2-índice» ::=
«índice» «formación-dominio-índice»
┤; «lista-ordenada-dominio2-índice» ├*

«lista-valores-dominio» ::=
[«rango-inferior», «rango-superior»] ┤, «lista-valores-dominio» ├* |
«valor-entero-positivo» ┤, «lista-valores-dominio» ├*

«dominio-t3» ::= «lista-ordenada-dominio3-índice»

«lista-ordenada-dominio3-índice» ::=
«índice» = «dominio-índice»
┤; «lista-ordenada-dominio3-índice» ├*

«operador-aritmético» ::= ^ | * | / | + | -

«operador-conjunto» ::= <Y> | <O> | <MENOS>

«operador-relación» ::= <IG> | <DIF> | <MA> | <MAI> | <ME> | <MEI>

«operador-lógico» ::= <Y> | <O>

«índice» ::= «identificador-índice» | «identificador-subíndice»

«identificador-índice» ::= «identificador»

«identificador-subíndice» ::= «identificador»

«identificador-índice-padre» ::= «índice»

«identificador-índice-equivalente» ::= «índice»

«identificador-variable» ::= «identificador»

«identificador-parámetro» ::= «identificador»

«identificador-conjunto» ::= «identificador»

«identificador-restricción» ::= «identificador»

«identificador-subrestricción» ::= «identificador»

«identificador-restricción-padre ::= «identificador»

«identificador-restricción-equivalente» ::= «identificador»

«identificador-vector-significados-particulares»::= «identificador»

«identificador-vector-objetivo» ::= «identificador»

«identificador-vector-restricciones» ::= «identificador»

«identificador-vector-rangos» ::= «identificador»

«identificador-vector-fronteras» ::= «identificador»

«identificador-vector-frontera-inferior» ::= «identificador»

«identificador-vector-frontera-superior» ::= «identificador»

«identificador-vector-tuplos» ::= «identificador»

«identificador» ::=

«letra» «resto-identificador»*

«resto-identificador» ::= «letra» | «dígito» | «símbolos-especiales»

«letra» ::= a | b | c | d | f | g | h | i | j | k | l | m | n | o |
p | q | r | s | t | u | v | w | x | y | z |
A | B | C | D | F | G | H | I | J | K | L | M | N | O |
P | Q | R | S | T | U | V | W | X | Y | Z |

«dígito» ::= 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 0

«símbolos-especiales» ::= . | _ | @ | # | % | & | !

«nombre-modelo» ::= «inicio-nombre»

«nombre-del-archivo» ::= «inicio-nombre» [. «sufijo-nombre»]?

«inicio-nombre» ::= «letra» «resto-nombre»*

«sufijo-nombre» ::= «letra» «resto-nombre»*

«resto-nombre» ::= «letra» | «dígito» | «símbolos-especiales-archivo»

«simbolos-especiales-archivo» ::= _ | @ | # | % | & | !

«dominio-máximo» ::= «valor-entero-positivo»

«desplazamiento» ::= «valor-entero-positivo»

«valor-del-índice» ::= «valor-entero-positivo»

«número-máximo-campos» ::= «valor-entero-positivo»

«número-campo» ::= «valor-entero-positivo»

«campo-de-significado» ::= «valor-entero-positivo»

«longitud-campos» ::= «valor-entero-positivo»

«número-decimales» ::= «valor-entero-positivo»

«factor-repetición» ::= «valor-entero-positivo»

«rango-inferior» ::= «valor-entero-positivo»

«rango-superior» ::= «valor-entero-positivo»

«valor-en-campo» ::= «constante»

«constante» ::= ┤ - ├? «dígito»+ ┤ . «dígito»+ ├? |
┤ - ├? «dígito»+

«valor-entero» ::=
«valor-entero-positivo» | - «valor-entero-positivo»

«valor-entero-positivo» ::= «dígito»+

«significado-general» ::= «descripción»

«significado-particular» ::= «descripción»

«etiqueta» ::= «descripción»

«descripción» ::= «caracteres»+

«caracteres» ::=
«letra» | «número» | «símbolos especiales» | «resto-símbolos»

«resto-símbolos» ::=

«espacio» | = | - | , | ; | : | / | * | ^ | (|) | [|] | { | }

A.3 INDICE DE DEFINICIONES

B.1 EJEMPLO # 1

B.1.1 DESCRIPCION DEL PROBLEMA

Un fabricante de plásticos tiene en existencia en su fábrica de Toluca, 1200 cajas de envoltura transparente y otras 1000 cajas en su fábrica de Querétaro. El fabricante tiene órdenes para este producto por parte de tres diferentes detallistas: Envolturas Elegantes, Paquetería Fina y Regalos Distinguidos en cantidades de 1000, 700 y 500 cajas, respectivamente.

Los costos unitarios de envío de las fábricas a los detallistas son los siguientes:

	Envolturas Elegantes	Paquetería Fina	Regalos Distinguidos
Toluca	14	10	12
Querétaro	15	13	8

Se desea encontrar la distribución que minimice el costo.

B.1.2 FORMULACION ALGEBRAICA DEL MODELO

Sean:

Indices

$i \in \{1,2\}$ El índice asociado al conjunto de fábricas { Toluca, Querétaro } respectivamente.

$j \in \{1,2,3\}$ El índice asociado al conjunto de detallistas { Envolturas Elegantes, Paquetería Fina, Regalos Distinguidos } respectivamente.

Variables

$EMBARQ_{ij}$ La cantidad embarcada de la fábrica i al detallista j.

Parámetros

$COSTO.U_{ij}$ El costo de transportar cada caja de la fábrica i al detallista j.

$EXTNC_i$ La existencia de cajas en la fábrica i.

$DEMANDA_j$ La demanda de cajas del detallista j.

Minimizar:

Función Objetivo

$$COSTO = \sum_i \sum_j COSTO.U_{ij} * EMBARQ_{ij}$$

Costo total de los embarques.

sujeto a:

Restricciones

$$\sum_j EMBARQ_{ij} \leq EXTNC_i$$

Asegura que el total de embarques de la fábrica i no sea mayor a su existencia.

$$\sum_i EMBARQ_{ij} = DEMANDA_j$$

Asegura que el total de embarques recibidos por el detallista j satisfagan la cantidad demandada

Dados:

Datos

i	$EXTNC_i$
1	1200
2	1000

J	$DEMANDA_j$
1	1000
2	700
3	500

i	j	$COSTO.U_{ij}$
1	1	14
1	2	10
1	3	12
2	1	15
2	2	13
2	3	8

B.1.3 ESPECIFICACION EN EL LENGUAJE GEMOLI

```
<MOD> TRANSP

<PAQ> MILP88


     $ DEFINICION DE INDICES

<IND>      I,"FABRICA",2

<SP>  {1,"FABRICA EN TOLUCA";2,"FABRICA EN QUERETARO"}
<IND>      J,"DETALLISTA",3
<SP>  {1,"ENVOLTURAS ELEGANTES";2,"PAQUETERIA FINA";3,
        "REGALOS DISTINGUIDOS"}


     $ DEFINICION DE VARIABLES

<VAR> EMBARQ (I,J), "EMBARQUE DE LA FABRICA I AL DETALLISTA J"


     $ DEFINICION DE PARAMETROS

 <PARAM> COSTO.U (I,J), "COSTO UNITARIO DE EMBARQUE DE LA
                         FABRICA I AL DETALLISTA J"

 <PARAM> EXTNC(I), "EXISTENCIAS EN LA FABRICA"

 <PARAM> DEM(J), "DEMANDA DEL DETALLISTA"

     $ FUNCION OBJETIVO

<RES> COSTO "COSTO MINIMO DEL PEDIDO"
<EC>  SUM ((I=[1,2];J=[1,3]),COSTO.U (I;J)*EMBARQ(I;J))
<MIN> COSTO


     $ DEFINICION DE RESTRICCIONES

     $ ASEGURA QUE LOS EMBARQUES SEAN IGUALES O MENORES A LA
     $            EXISTENCIA EN LA FABRICA

<RES> EXTF (I) "EXISTENCIAS EN LA FABRICA"
<EC>  SUM ((J=[1,3]),EMBARQ(I;J)) <MEI> EXTNC(I)


     $ ASEGURA QUE CADA DETALLISTA RECIBA AL MENOS LO SOLICITADO

<RES> ORDE (J) "ORDEN DEL DETALLISTA"
<EC>  SUM ((I=[1,2]),EMBARQ(I;J)) <MAI> DEM(J)
     $ DEFINICION DE VALORES
```

```
<VALOR>  COSTO.U (I=[1,2];J=[1,3])= {14,10,12,15,13,8}
<VALOR>  EXTNC (I=[1,2])= {1200,1000}
<VALOR>  DEM (J=[1,3])= {1000,700,500}
<FIN>
```

B.1.4 EQUIVALENCIAS HILERAS/RESTRICCIONES Y COLUMNAS/VARIABLES
M O D E L O TRANSP.

EQUIVALENCIAS ENTRE HILERAS Y RESTRICCIONES

HILERAS	RESTRICCION		NOMBRES
	FUNCION OBJETIVO		
0	MINIMO COSTO		COSTO
	EXTF(I)		
	EXISTENCIAS EN LA FABRICA		
1	<=	1200	EXTF(1)
2	<=	1000	EXTF(2)
	ORDE(J)		
	ORDEN DEL DETALLISTA		
3	>=	1000	ORDE(1)
4	>=	700	ORDE(2)
5	>=	500	ORDE(3)

EQUIVALENCIAS ENTRE COLUMNAS Y VARIABLES

COLUMNA	VARIABLE	NOMBRES
	EMBARQ(I,J)	
1	EMBARQ(1,1)	
2	EMBARQ(1,2)	
3	EMBARQ(1,3)	
4	EMBARQ(2,1)	
5	EMBARQ(2,2)	
6	EMBARQ(2,3)	

B.1.5 ARCHIVO GENERADO PARA MILP88

```
NEW PROBLEM,TRANSP,MIN,5,0,6  EDIT,0,1,14,1,1,1,3,1,1
EDIT,0,2,10,1,2,1,4,2,1       EDIT,0,3,12,1,3,1,5,3,1
EDIT,0,4,15,2,4,1,3,4,1       EDIT,0,5,13,2,5,1,4,5,1
EDIT,0,6,8,2,6,1,5,6,1
EDIT,1,REL,<=,2,REL,<=,3,REL,>=,4,REL,>=,5,REL,>=
EDIT,1,RHS,1200,2,RHS,1000,3,RHS,1000,4,RHS,700,5,RHS,500
SAVE PROBLEM,TRANSP
```

B.1.6 SOLUCION DEL MODELO

FUNCION OBJETIVO

COSTO = 25 500

VARIABLES

COLUMNA	VARIABLE	VALOR	DESCRIPCION
1	EMBARQ(1,1)	500	EMBARQUE DE LA FABRICA EN TOLUCA AL DETALLISTA ENVOLTURAS ELEGANTES
2	EMBARQ(1,2)	700	EMBARQUE DE LA FABRICA EN TOLUCA AL DETALLISTA PAQUETERIA FINA
4	EMBARQ(2,1)	500	EMBARQUE DE LA FABRICA EN QUERETARO AL DETALLISTA ENVOLTURAS ELEGANTES
6	EMBARQ(2,3)	500	EMBARQUE DE LA FABRICA EN QUERETARO AL DETALLISTA REGALOS DISTINGUIDOS

RESTRICCIONES

HILERA	RESTRICCION	VALOR	DESCRIPCION
1	EXTF(1)	1200	EXISTENCIA UTILIZADA DE LA FABRICA EN TOLUCA
2	EXTF(2)	1000	EXISTENCIA UTILIZADA DE LA FABRICA EN QUERETARO.
3	ORDE(1)	1000	ORDENES RECIBIDAS POR ENVOLTURAS ELEGANTES
4	ORDE(2)	700	ORDENES RECIBIDAS POR PAQUETERIA FINA
5	ORDE(3)	500	ORDENES RECIBIDAS POR REGALOS DISTINGUIDOS

B.2 EJEMPLO #2

B.2.1 DESCRIPCION DEL PROBLEMA

Una fundidora de aleación de aluminio debe producir 2000 tons. de una aleación particular al mínimo costo. La aleación, sin embargo, debe satisfacer ciertas restricciones químicas. La fundidora tiene a su disposición para la aleación cinco materiales de desperdicio de conocida composición química, además, requiere de aluminio y silicón industrialmente puros (97%), de los cuales puede comprar lo que necesite. Se considera, que durante el proceso las substancias procedentes de las materias primas no aumentarán o disminuirán sus cantidades.

La información relevante del problema se proporciona en forma tabular en las siguientes tablas.

Contenido permitido de metales por cada 2000 tons. de aleación.

(T o n e l a d a s)

Metales Básicos		Máximo	Mínimo
Fierro	(Fe)	60	
Cobre	(Cu)	100	
Manganeso	(Mn)	40	
Magnesio	(Mg)	30	
Aluminio	(Al)		1500
Silicón	(Si)	300	250

Inventario de material de desperdicio

Lotes de carga	Inventario (tons.)	Cantidades mínimas a ser usadas
1	200	
2	750	
3	800	400
4	700	100
5	1500	

Contenido de Metal por Unidad de Materia Prima

Materia Prima	Metales Básicos					
	Fe	Cu	Mn	Mg	Al	Si
Material de Desperdicio 1	.15	.03	.02	.02	.70	.02
Material de Desperdicio 2	.04	.05	.04	.03	.75	.06
Material de Desperdicio 3	.02	.08	.01	0	.80	.08
Material de Desperdicio 4	.04	.02	.02	0	.75	.12
Material de Desperdicio 5	.02	.06	.02	.01	.80	.02
Aluminio	.01	.01	0	0	.97	.01
Silicón	.03	0	0	0	0	.97

Costo por Lote de Carga

lotes de carga	costo por tonelada
1	.03
2	.08
3	.17
4	.12
5	.15
6	.21
7	.38

Se desea la mezcla de materia prima que minimice el costo de la aleación.

B.2.2 FORMULACION ALGEBRAICA DEL PROBLEMA

Sean:

Indices

$i \varepsilon \{1,2,3,4,5,6,7\}$
Los índices asociados al conjunto de materias primas { material de desperdicio 1, material de desperdicio 2, material de desperdicio 3, material de desperdicio 4, material de desperdicio 5, aluminio, silicón } respectivamente.

$j \varepsilon \{1,2,3,4,5,6\}$
Los índices asociados al conjunto de metales { fierro, cobre, manganeso, magnesio, aluminio, silicio } respectivamente.

Variables

$CARGA_i$ La carga de la materia prima i para producir la aleación.

Parámetros

$COSTO.U_i$ El costo unitario de la materia prima i.

$CONT.ME_{ij}$ El contenido del metal j por cada unidad de materia prima i.

$CONT.MP_j$ El contenido máximo permitido del metal j en la aleación que se desea producir.

$CARGA.LT_i$ El límite inferior para los valores de $carga_i$.

$CARGA.LS_i$ El límite superior para los valores de $carga_i$.

Minimizar:

Función objetivo

$$COSTO = \sum_i COSTO.U_i * CARGA_i$$

Costo total de producir la aleación.

Sujeto a:

Restricciones

$$\sum_i CARGA_i = 2000$$

Asegura que se produzca la cantidad de aleación deseada.

$$\sum_j CONT.ME_{ij} * CARGA_i \leq CONT.MP_i \quad \text{para } i \varepsilon \{1,2,3,4,6\}$$

$$\sum_j CONT.ME_{ij} * CARGA_i \geq CONT.MP_i \quad \text{para } i=5$$

$$\sum_j CONT.ME_{ij} * CARGA_i \geq 250 \quad \text{para } i=6$$

Aseguran que la aleación producida cumpla con los contenidos de metales especificados.

y a:

Valores de Frontera

$$CARGA.LT_i \leq CARGA_i \leq CARGA.LS_i$$

Asegura que los valores que se asignen a la variable $CARGA_i$ estén dentro del rango válido.

y a:

Reglas de consistencia

$CONT.ME_{ij} \leq 1.0$ para toda i, para toda j

Aseguran que el contenido del metal j en la materia prima i sea una fracción de la misma.

Dados:

Datos

i	$COSTO.U_i$	$CARGA.LI_i$	$CARGA.LS_i$	j	$CONT.MP_j$
1	.03		200	1	60
2	.08		750	2	100
3	.17	400	800	3	40
4	.12	100	700	4	30
5	.15		1500	5	1500
6	.21			6	300
7	.38				

$CONT.ME_{ij}$

i	j 1	2	3	4	5	6
1	.15	.03	.02	.02	.70	.02
2	.04	.05	.04	.03	.75	.06
3	.02	.08	.01	0	.80	.08
4	.04	.02	.02	0	.75	.12
5	.02	.06	.02	.01	.80	.02
6	.01	.01	0	0	.97	.01
7	.03	0	0	0	0	.97

B.2.3 ESPECIFICACION EN EL LENGUAJE GEMOLI

```
<MOD> MEZCLA

<PAQ> XA

     $ DEFINICION DE INDICES

<IND>       I,"LOTES DE CARGA",7
      <SP> {1 "LOTE 1"; 2 "LOTE 2"; 3 "LOTE 3"; 4 "LOTE 4";
            5 "LOTE 5"; 6 "LOTE DE ALUMINIO";
            7 "LOTE DE SILICON"}

<IND>       J,"METALES BASICOS",6
      <SP> {1 "FIERRO"; 2 "COBRE"; 3 "MANGANESO";
            4 "MAGNESIO"; 5 "ALUMINIO"; 6 "SILICON"}
```

```
     $ DEFINICION DE VARIABLES

<VAR> CARGA(I),"CARGA DEL LOTE I",<FRON_INF> FINF,<FRON_SUP> FSUP

     $ DEFINICION DE PARAMETROS

<PARAM> COSTO.U (I),"COSTO UNITARIO DEL LOTE I"

<PARAM> CONT.ME (I,J),"CONTENIDO DEL METAL J EN EL LOTE I"
<PARAM> CONTMP (J),"CONTENIDO MAXIMO PERMITIDO DEL METAL J"

     $ DEFINICION DE FUNCION OBJETIVO

<RES> COSTO "COSTO DE LA ALEACION"
<EC>  SUM ((I=[1,7]),COSTO.U(I)* CARGA (I)) <MIN> COSTO

     $ DEFINICION DE RESTRICCIONES

     $ ASEGURA EL VOLUMEN DE PRODUCCION

<RES> PROD.T "TOTAL DE ALEACION PRODUCIDA"
<EC>  SUM ((I=[1,7]), CARGA (I)) <IG> 2000

     $ ASEGURAN LA CANTIDAD MAXIMA Y MINIMA PERMITIDA DE METAL

<RES>  CMPM (J) "CONTENIDO MAXIMO/MINIMO PERMITIDO DEL METAL J"
<DOM> (J=[1,4])
<EC>    SUM  ((I=[1,7]),CONT.ME (I;J)*CARGA(I)) <MEI> CONTMP(J)
<DOM> (J=5)
<EC>    SUM((I=[1,7]),  CONT.ME (I;J)*CARGA(I)) <MAI> CONTMP(J)
<DOM> (J=6)
<EC>  SUM((I=[1,7]), CONT.ME (I;J)*CARGA(I)) <MEI> CONTMP(J) ; 50

     $ DEFINICION DE VALORES

<VALOR> COSTO.U(I=[1,7]) = {0.03, 0.08, 0.17, 0.12, 0.15,
                            0.21, 0.38}
<VALOR> CONTMP (J=[1,6]) = {60, 100, 40, 30, 1500, 300}
<VALOR> CONT.ME (I=[1,7]; J=[1,6])=
        {0.15, 0.03, 0.02, 0.02, 0.70, 0.02,
         0.04, 0.05, 0.04, 0.03, 0.75, 0.06,
         0.02, 0.08, 0.01, 0, 0.80, 0.08,
         0.04, 0.02, 0.02, 0, 0.75, 0.12,
         0.02, 0.06, 0.02, 0.01, 0.80, 0.02,
         0.01, 0.01, 0, 0, 0.97, 0.01,
         0.03, 0, 0, 0, 0, 0.97 }

<VALOR> FSUP(I=[1,5])={200, 750, 800, 700, 1500}
<VALOR> FINF(I={3,4})={400,100}

<FIN>
```

B.2.4 EQUIVALENCIA HILERAS/RESTRICCIONES Y COLUMNAS/VARIABLES

M O D E L O MEZCLA

EQUIVALENCIAS ENTRE HILERAS Y RESTRICCIONES

HILERAS	RESTRICCION	RANGOS	NOMBRES
	F U N C I O N O B J E T I V O		
COSTO	MINIMO COSTO		
	R E S T R I C C I O N E S		
	PROD.T TOTAL DE ALEACION PRODUCIDA		
PROD.T	= 2000		PROD.T
	CMPM(J,I) CONTENIDO MAXIMO PERMITIDO DEL METAL J		
CMPM1	<= 60		CMPM(1)
CMPM2	<= 100		CMPM(2)
CMPM3	<= 40		CMPM(3)
CMPM4	<= 30		CMPM(4)
CMPM5	>= 1500		CMPM(5)
CMPM6	<= 300	250	CMPM(6)
	F R O N T E R A S U P E R I O R		
	CARGA(I) CARGA DEL LOTE I		
CARGA1	<= 200		CARGA(1)
CARGA2	<= 750		CARGA(2)
CARGA3	<= 800		CARGA(3)
CARGA4	<= 700		CARGA(4)
CARGA5	<= 1500		CARGA(5)
	F R O N T E R A I N F E R I O R		
	CARGA(I) CARGA DEL LOTE I		
CARGA 3	>= 400		CARGA(3)
CARGA 4	>= 100		CARGA(4)

EQUIVALENCIAS ENTRE COLUMNAS Y VARIABLES

COLUMNA	VARIABLE
	CARGA(I)
CARGA1	CARGA(1)
CARGA2	CARGA(2)
CARGA3	CARGA(3)
CARGA4	CARGA(4)
CARGA5	CARGA(5)
CARGA6	CARGA(6)
CARGA7	CARGA(7)

B.2.5 ARCHIVO GENERADO PARA XA

```
NAME            MEZCLA
ROWS
 N   COSTO
 E   PROD.T
 L   CMPM1
 L   CMPM2
 L   CMPM3
 L   CMPM4
 G   CMPM5
 L   CMPM6
COLUMNS
    CARGA1    COSTO     0.03          PROD.T    1
    CARGA1    CMPM1     0.15          CMPM2     0.03
    CARGA1    CMPM3     0.02          CMPM4     0.02
    CARGA1    CMPM5     0.70          CMPM6     0.02
    CARGA2    COSTO     0.08          PROD.T    1
    CARGA2    CMPM1     0.04          CMPM2     0.05
    CARGA2    CMPM3     0.04          CMPM4     0.03
    CARGA2    CMPM5     0.75          CMPM6     0.06
    CARGA3    COSTO     0.17          PROD.T    1
    CARGA3    CMPM1     0.02          CMPM2     0.08
    CARGA3    CMPM3     0.01          CMPM5     0.80
    CARGA3    CMPM6     0.08
    CARGA4    COSTO     0.12          PROD.T    1
    CARGA4    CMPM1     0.04          CMPM2     0.02
    CARGA4    CMPM3     0.02          CMPM5     0.75
    CARGA4    CMPM6     0.12
    CARGA5    COSTO     0.15          PROD.T    1
    CARGA5    CMPM1     0.02          CMPM2     0.06
    CARGA5    CMPM3     0.02          CMPM4     0.01
    CARGA5    CMPM5     0.80          CMPM6     0.02
    CARGA6    COSTO     0.21          PROD.T    1
    CARGA6    CMPM1     0.01          CMPM2     0.01
```

```
    CARGA6      CMPM5       0.97          CMPM6       0.01
    CARGA7      COSTO       0.38          PROD.T      1
    CARGA7      CMPM1       0.03          CMPM6       0.97
RHS
    RESTRICS    PROD.T      2000          CMPM1       60
    RESTRICS    CMPM2       100           CMPM3       40
    RESTRICS    CMPM4       30            CMPM5       1500
    RESTRICS    CMPM6       300
RANGES
    RANGOS      CMPM6       250
BOUNDS
 UP FRONTERA    CARGA1      200
 UP FRONTERA    CARGA2      750
 UP FRONTERA    CARGA3      800
 UP FRONTERA    CARGA4      700
 UP FRONTERA    CARGA5      1500
 LO FRONTERA    CARGA3      400
 LO FRONTERA    CARGA4      100
ENDATA
```

B.2.6 SOLUCION DEL MODELO

FUNCION OBJETIVO

COSTO = 296.22

VARIABLES

COLUMNA	VARIABLE	VALOR	DESCRIPCION
CARGA2	CARGA(2)	665.34	CARGA DEL LOTE 2
CARGA3	CARGA(3)	490.25	CARGA DEL LOTE 3
CARGA4	CARGA(4)	424.19	CARGA DEL LOTE 4
CARGA6	CARGA(6)	299.64	CARGA DEL LOTE DE ALUMINIO
CARGA7	CARGA(7)	120.57	CARGA DEL LOTE DE SILICON

RESTRICCIONES

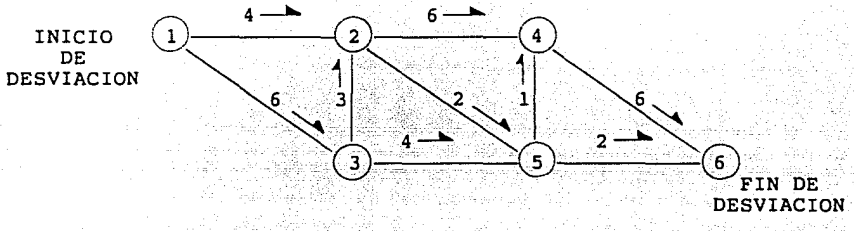
HILERA	RESTRICCION	VALOR	DESCRIPCION
PROD.T	PROD.T	2000.00	TOTAL DE ALEACION PRODUCIDA
CMPM1	CMPM(1)	60.00	CONTENIDO DE FIERRO
CMPM2	CMPM(2)	83.97	CONTENIDO DE COBRE
CMPM3	CMPM(3)	40.00	CONTENIDO DE MANGANESO
CMPM4	CMPM(4)	19.96	CONTENIDO DE MAGNESIO
CMPM5	CMPM(5)	1500.00	CONTENIDO DE ALUMINIO
CMPM6	CMPM(6)	250.00	CONTENIDO DE SILICON

B.3 EJEMPLO # 3

B.3.1 DESCRIPCION DEL PROBLEMA

Por la construcción de un puente sobre una vía rápida, se tiene que desviar el tráfico de la misma en un punto anterior al sitio donde irá el puente y reconectarla en un punto posterior.

El diagrama de calles que se pueden utilizar para la desviación se presenta a continuación, en él se indican el sentido de las calles y el aforo vehicular máximo permitido (en miles).



Se desea encontrar la estructura que nos proporcione el flujo máximo.

B.3.2 FORMULACION ALGEBRAICA DEL PROBLEMA

Sean:

Indices

nodo ε {1,2,3,4,5,6}
Los nodos de la red.

org ε {1,2,3,4,5,6}
Los índices asociados a los nodos origen de cada tramo de desviación.

dst ε {1,2,3,4,5,6}
Los índices asociados a los nodos destino de cada tramo de desviación.

Variables

$FLUJO_{org\ dst}$
El flujo vehicular en la solución óptima, del tramo definido por los índices org y dst.

Parámetros

$FLU\_MX_{org\ dst}$

El flujo máximo permitido en el tramo definido por los índices org y dst.

Maximizar:

Función objetivo

$$FLU\_OP = \sum_{dst} FLUJO_{org\ dst} \quad \text{para org} = \{ 1 \}$$

Maximiza el flujo vehicular que puede entrar a la desviación.

Sujeto a:

Restricciones

$$FLUJO_{org\ dst} \leq FUJO\_MX_{org\ dst}$$

Asegura que en cada tramo no se exceda el aforo límite.

$$\sum_{org} FLUJO_{org\ (dst=nodo)} - \sum_{dst} FLUJO_{(org=nodo)\ dst} = 0$$

para todo nodo ε { 2,3,4,5 }

Asegura que en todo nodo intermedio el flujo de entrada sea igual al flujo de salida.

Dados:

Datos

org	dst	$FLU\_MX_{org\ dst}$
1	2	4
1	3	6
2	4	6
2	5	2
3	2	3
3	5	4
4	6	6
5	4	1
5	6	2

B.3.3 ESPECIFICACION EN EL LENGUAJE GEMOLI

```
<MOD> FLUJO_MX

<PAQ> IBM

          $ DEFINICION DE INDICES

<IND> NODO,"NODO",6

<IND> ORG,"ORIGEN DEL TRAMO",6

<IND> DST,"DESTINO DEL TRAMO",6

          $ DEFINICION DE VARIABLES

<VAR> FLUJO(ORG,DST), "FLUJO VEHICULAR EN EL TRAMO"

          $ DEFINICION DE PARAMETROS

<PARAM> FLU_MX(ORG,DST), "FLUJO MAXIMO EN EL TRAMO"

<PARAM> FICT(ORG,DST),"OBLIGA LA DEFINICION EXPLICITA DEL
                       COEFICIENTE"

          $ FUNCION OBJETIVO

<RES> FLU_OP,"FLUJO OPTIMO"

<EC> SUM((DST=[1,6]),FLUJO(ORG=1;DST))

<MAX> FLU_OP

      $ DEFINICION DE RESTRICCIONES

<RES> LIM_FL(ORG,DST),"LIMITE DE FLUJO EN EL TRAMO"

<EC> FICT(ORG;DST)*FLUJO(ORG;DST)

<MEI> FLU_MX(ORG;DST)

<RES> EN_SL(NODO),"BALANCE ENTRADA/SALIDA"

<DOM> (NODO=[2,5])

<EC> SUM((ORG=[1,6]),FLUJO(ORG;DST=NODO)) -
     SUM((DST=[1,6]),FLUJO(ORG=NODO;DST))

<IG> 0
```

```
      $ DEFINICION DE VALORES
<VALOR> FLU_MX(ORG=[1,6];DST=[1,6])=
      { NULO  4  6  3*NULO
        3*NULO  6  2  NULO
        NULO  3  2*NULO  4  NULO
        5*NULO  6
        3*NULO  1  NULO  2
        6*NULO }
<VALOR> FICT(ORG=[1,6];DST=[1,6])={36*1}
<FIN>
```

B.3.4 EQUIVALENCIA HILERAS/RESTRICCIONES Y COLUMNAS/VARIABLES

M O D E L O FLUJO_MX

EQUIVALENCIAS ENTRE HILERAS Y RESTRICCIONES

HILERAS	RESTRICCION	NOMBRES
	FUNCION OBJETIVO	
FLU_OP	MAXIMO	

R E S T R I C C I O N E S

	LIM_FL(ORG,DST) LIMITE DE FLUJO EN EL TRAMO	
LIM_FL02	<= 4	LIM_FL(1,2)
LIM_FL03	<= 6	LIM_FL(1,3)
LIM_FL10	<= 6	LIM_FL(2,4)
LIM_FL11	<= 2	LIM_FL(2,5)
LIM_FL14	<= 3	LIM_FL(3,2)
LIM_FL17	<= 4	LIM_FL(3,5)
LIM_FL24	<= 6	LIM_FL(4,6)
LIM_FL28	<= 1	LIM_FL(5,4)
LIM_FL30	<= 2	LIM_FL(5,6)

```
                EN_SL(NODO)
            BALANCE ENTRADA/SALIDA

EN_SL2                  = 0             EN_SL(2)
EN_SL3                  = 0             EN_SL(3)
EN_SL4                  = 0             EN_SL(4)
EN_SL5                  = 0             EN_SL(5)

              EQUIVALENCIAS ENTRE COLUMNAS Y VARIABLES

COLUMNA         VARIABLE

                FLUJO(ORG,DST)
        FLUJO VEHICULAR  EN EL TRAMO

FLUJO02         FLUJO(1,2)
FLUJO03         FLUJO(1,3)
FLUJO10         FLUJO(2,4)
FLUJO11         FLUJO(2,5)
FLUJO14         FLUJO(3,2)
FLUJO17         FLUJO(3,5)
FLUJO24         FLUJO(4,6)
FLUJO28         FLUJO(5,4)
FLUJO30         FLUJO(5,6)
```

B.3.5 ARCHIVO GENERADO PARA IBM

```
/PMX020  JOB (PXMGPC,MPSX),'SISTEMAGEMOLI',
//         CLASS=A,MSGCLASS=T,MSGLEVEL=(1,1),NOTIFY=PMX020
//*UQ ALLOW
//JOBLIB   DD DSN=DPL.MPSX370,DISP=SHR,UNIT=3380,VOL=SER=D3380D
//MPSXCOMP EXEC PGM=DPLCOMP
//SYSPRINT DD SYSOUT=T
//SYSMLCP  DD UNIT=SYSDA,DISP=(,PASS),SPACE=(CYL,(1,1))
//SCRATCH1 DD UNIT=SYSDA,SPACE=(CYL,(1,1)),DISP=(NEW,DELETE)
//SCRATCH2 DD UNIT=SYSDA,SPACE=(CYL,(1,1)),DISP=(NEW,DELETE)
//SCRATCH3 DD UNIT=SYSDA,SPACE=(CYL,(1,1)),DISP=(NEW,DELETE)
//SCRATCH4 DD UNIT=SYSDA,SPACE=(CYL,(1,1)),DISP=(NEW,DELETE)
//SYSIN    DD *
        PROGRAM
        INITIALZ
        TITLE('M O D E L O  FLUJO_MX')
        MOVE(XDATA,'FLUJO_MX')
        MOVE(XPBNAME,'PBFILE')
        MOVE(XMINMAX,'MAX')
        CONVERT
        SETUP
        MOVE(XOBJ,'FLU_OP')
        MOVE(XRHS,'RESTRICS')
        WRITE('1H1')
        OPTIMIZE
```

```
       SOLUTION('FILE','SOLUCION')
       EXIT
       PEND
/*
//MPSXEXEC EXEC PGM=DPLEXEC,REGION=5M,COND=(0,NE,MPSXCOMP)
//ETA1     DD UNIT=SYSDA,SPACE=(CYL,(15,5)),DISP=(NEW,DELETE)
//ETA2     DD UNIT=SYSDA,SPACE=(CYL,(10,5)),DISP=(NEW,DELETE)
//ETA3     DD UNIT=SYSDA,SPACE=(CYL,(10,5)),DISP=(NEW,DELETE)
//ETA4     DD UNIT=SYSDA,SPACE=(CYL,(10,5)),DISP=(NEW,DELETE)
//MATRIX1  DD UNIT=SYSDA,SPACE=(CYL,(15,15)),DISP=(NEW,DELETE)
//MATRIX2  DD UNIT=SYSDA,SPACE=(CYL,(15,15)),DISP=(NEW,DELETE)
//MATRIX3  DD UNIT=SYSDA,SPACE=(CYL,(10,5)),DISP=(NEW,DELETE)
//MATRIX4  DD UNIT=SYSDA,SPACE=(CYL,(10,5)),DISP=(NEW,DELETE)
//SCRATCH1 DD UNIT=SYSDA,SPACE=(CYL,(10,4),RLSE),DISP=(NEW,DELETE)
//SCRATCH2 DD UNIT=SYSDA,SPACE=(CYL,(10,5),RLSE),DISP=(NEW,DELETE)
//SCRATCH3 DD UNIT=SYSDA,SPACE=(CYL,(10,4),RLSE),DISP=(NEW,DELETE)
//SCRATCH4 DD UNIT=SYSDA,SPACE=(CYL,(10,4),RLSE),DISP=(NEW,DELETE)
//*SOLUCION DD DSN=SOLMOD,DISP=(NEW,KEEP),UNIT=3380,VOL=SER=E3380E,
//*          SPACE=(CYL,(50,20)),DCB=(RECFM=VBS,LRECL=204,BLKSIZE=1024)
//SOLUCION DD DSN=SOLMOD,DISP=OLD,UNIT=3380,VOL=SER=E3380E
//PROBFILE DD UNIT=SYSDA,SPACE=(CYL,(50,20)),DISP=(NEW,DELETE)
//REPWORK  DD UNIT=SYSDA,SPACE=(CYL,(20,1)),DISP=(NEW,DELETE)
//REPFILE  DD UNIT=SYSDA,SPACE=(CYL,(20,1)),DISP=(NEW,DELETE)
//SYSMLCP  DD DSN=*.MPSXCOMP.SYSMLCP,DISP=(OLD,PASS)
//REVIFILE DD UNIT=SYSDA,SPACE=(CYL,(5)),DISP=(NEW,DELETE)
//SYSPRINT DD SYSOUT=T
//SYSIN    DD *
NAME          FLUJO_MX
ROWS
 N  FLU_OP
 L  LIM_FL02
 L  LIM_FL03
 L  LIM_FL10
 L  LIM_FL11
 L  LIM_FL14
 L  LIM_FL17
 L  LIM_FL24
 L  LIM_FL28
 L  LIM_FL30
 E  EN_SL2
 E  EN_SL3
 E  EN_SL4
 E  EN_SL5
COLUMNS
    FLUJO02   FLU_OP    1             LIM_FL02  1
    FLUJO02   N_SL2     1
    FLUJO03   FLU_OP    1             LIM_FL03  1
    FLUJO03   EN_SL3    1
    FLUJO10   LIM_FL10  1             EN_SL2    -1
    FLUJO10   EN_SL4    1
    FLUJO11   LIM_FL11  1             EN_SL2    -1
    FLUJO11   EN_SL5    1
```

```
    FLUJO14    LIM_FL14   1              EN_SL2     1
    FLUJO14    EN_SL3     -1
    FLUJO17    LIM_FL17   1              EN_SL3     -1
    FLUJO17    EN_SL5     1
    FLUJO24    LIM_FL24   1              EN_SL4     -1
    FLUJO28    LIM_FL28   1              EN_SL4     1
    FLUJO28    EN_SL5     -1
    FLUJO30    LIM_FL30   1              EN_SL5     -1
RHS
    RESTRICS   LIM_FL02   4              LIM_FL03   6
    RESTRICS   LIM_FL10   6              LIM_FL11   2
    RESTRICS   LIM_FL14   3              LIM_FL17   4
    RESTRICS   LIM_FL24   6              LIM_FL28   1
    RESTRICS   LIM_FL30   2              EN_SL2     0
    RESTRICS   EN_SL3     0              EN_SL4     0
    RESTRICS   EN_SL5     0
ENDATA
/*
```

B.3.6 SOLUCION DEL MODELO

FUNCION OBJETIVO

FLU_OP = 8

VARIABLES

COLUMNA	VARIABLE	VALOR	DESCRIPCION
FLUJO02	FLUJO(1,2)	4	FLUJO VEHICULAR DEL NODO 1 AL NODO 2
FLUJO03	FLUJO(1,3)	4	FLUJO VEHICULAR DEL NODO 1 AL NODO 3
FLUJO10	FLUJO(2,4)	6	FLUJO VEHICULAR DEL NODO 2 AL NODO 4
FLUJO14	FLUJO(3,2)	2	FLUJO VEHICULAR DEL NODO 3 AL NODO 2
FLUJO17	FLUJO(3,5)	2	FLUJO VEHICULAR DEL NODO 3 AL NODO 5
FLUJO24	FLUJO(4,6)	6	FLUJO VEHICULAR DEL NODO 4 AL NODO 6
FLUJO30	FLUJO(5,6)	2	FLUJO VEHICULAR DEL NODO 5 AL NODO 6

RESTRICCIONES

HILERA	RESTRICCION	VALOR	DESCRIPCION
LIM_FL02	LIM_FL(1,2)	4	FLUJO VEHICULAR DEL NODO 1 AL NODO 2
LIM_FL03	LIM_FL(1,3)	4	FLUJO VEHICULAR DEL NODO 1 AL NODO 3
LIM_FL10	LIM_FL(2,4)	6	FLUJO VEHICULAR DEL NODO 2 AL NODO 4
LIM_FL11	LIM_FL(2,5)	0	FLUJO VEHICULAR DEL NODO 2 AL NODO 5
LIM_FL14	LIM_FL(3,2)	2	FLUJO VEHICULAR DEL NODO 3 AL NODO 2
LIM_FL17	LIM_FL(3,5)	2	FLUJO VEHICULAR DEL NODO 3 AL NODO 5
LIM_FL24	LIM_FL(4,6)	6	FLUJO VEHICULAR DEL NODO 4 AL NODO 6
LIM_FL28	LIM_FL(5,4)	0	FLUJO VEHICULAR DEL NODO 5 AL NODO 4

HILERA	RESTRICCION	VALOR	DESCRIPCION
LIM_FL30	LIM_FL(5,6)	2	FLUJO VEHICULAR DEL NODO 5 AL NODO 6
EN_SL2	EN_SL(2)	0	ENTRADAS MENOS SALIDAS DEL NODO 2
EN_SL3	EN_SL(3)	0	ENTRADAS MENOS SALIDAS DEL NODO 3
EN_SL4	EN_SL(4)	0	ENTRADAS MENOS SALIDAS DEL NODO 4
EN_SL5	EN_SL(5)	0	ENTRADAS MENOS SALIDAS DEL NODO 5

B.4 EJEMPLO #4

B.4.1 DESCRIPCION DEL PROBLEMA

El gobierno va a construir tres proyectos, agua potable, drenaje profundo, y tren bala. Cuatro compañías constructoras están compitiendo por los proyectos: IMA, DER, CON y ACE. La siguiente tabla resume la cotización de cada compañía por proyecto.

Cotizaciones

	Agua Potable	Drenaje Profundo	Tren Bala
IMA	5	13	19
DER	13	10	15
CON	11	15	27
ACE	15	9	6

Por motivos políticos y de un reparto más equitativo de las erogaciones se decidió que cada empresa tendrá cuando más un proyecto, y que cada proyecto será realizado por una sola compañía. Se desea conocer la asignación óptima de los proyectos que generen el costo mínimo.

B.4.2 FORMULACION ALGEBRAICA DEL PROBLEMA

Sean:

Indices

$i \in \{1,2,3,4\}$ Los índices asociados al conjunto de compañías { IMA, DER, CON, ACE } respectivamente.

$j \in \{1,2,3,\}$ Los índices asociados al conjunto de proyectos { Agua Potable, Drenaje Profundo, Tren Bala } respectivamente.

Variables

$PRO.A_{ij}$ La variable binaria que indicará la compañía i que realizará el proyecto j.

Parámetros

$COSTO.U_{ij}$ El costo del proyecto j si lo realiza la compañía i.

Minimizar:

Función Objetivo

$$COSTO = \sum_i \sum_j COSTO.U_{ij} * PRO.A_{ij}$$

El costo total de realizar todos los proyectos.

Sujeto a:

Restricciones

$$\sum_j PRO.A_{ij} \leq 1 \quad \text{para toda } i$$

Asegura que a cada compañía i, se le asigne como máximo un proyecto j.

$$\sum_i PRO.A_{ij} = 1 \quad \text{para toda } j$$

Asegura que todos los proyectos sean asignados.

B.4.3 ESPECIFICACION EN EL LENGUAJE GEMOLI

```
<MOD> ASIGN

<PAQ> XA

   $ DEFINICION DE INDICES

<IND>     I,"COMPAÑIA",4
     <SP> {1,"IMA";2,"DER";3,"CON";4,"ACE"}

<IND>     J,"PROYECTO",3
     <SP> {1,"AGUA POTABLE";2,"DRENAJE PROFUNDO";3,"TREN BALA"}

   $ DEFINICION DE VARIABLES

<VAR> PRO.A(I,J),"COMPAÑIA I ASIGNADA AL PROYECTO J"

   $ DEFINICION DE PARAMETROS

<PARAM> COSTO.U(I,J),"COSTO DEL PROYECTO J EN LA COMPAÑIA I"

   $ FUNCION OBJETIVO MINIMO COSTO POR LOS PROYECTOS

<RES>     COSTO,"MINIMO COSTO DE LOS PROYECTOS"

<EC> SUM((I=[1,4];J=[1,3]),COSTO.U(I;J)*PRO.A(I;J)) <MIN> COSTO

   $ DEFINICION DE RESTRICCIONES
```

```
   $ ASEGURA QUE EL NUMERO DE PROYECTOS ASIGNADOS POR COMPAÑIA
   $ SEA IGUAL A UNO

<RES>     COM(I),"PROYECTOS ASIGNADOS POR COMPAÑIA"
<EC> SUM((J=[1,3]),PRO.A(I;J)) <MEI> 1

   $ ASEGURA QUE TODOS LOS PROYECTOS SEAN ASIGNADOS

<RES>     PROY(J),"UN PROYECTO ASIGNADO A UNA SOLA COMPAÑIA"
<EC> SUM((I=[1,4]),PRO.A(I;J)) <IG> 1

   $ DEFINICION DE VALORES
<VALOR> COSTO.U(I=[1,4];J=[1,3])=
                { 5,13,19,
                  13,10,15,
                  11,15,27,
                  15,9,6  }
<FIN>
```

B.4.4 EQUIVALENCIA HILERAS/RESTRICCIONES Y COLUMNAS/VARIABLES

M O D E L O ASIGN

EQUIVALENCIAS ENTRE HILERAS Y RESTRICCIONES HOJA 1

HILERAS	RESTRICCION	RANGOS	NOMBRES
	F U N C I O N O B J E T I V O		
COSTO	MINIMO COSTO		
	R E S T R I C C I O N E S		
	COM(I) PROYECTOS ASIGNADOS POR COMPAÑIA		
COM1	<= 1		COM(1)
COM2	<= 1		COM(2)
COM3	<= 1		COM(3)
COM4	<= 1		COM(4)
	PROY(J) UN PROYECTO ASIGNADO A UNA SOLA COMPAÑIA		
PROY1	= 1		PROY(1)
PROY2	= 1		PROY(2)
PROY3	= 1		PROY(3)

EQUIVALENCIAS ENTRE COLUMNAS Y VARIABLES

COLUMNA	VARIABLE
	PRO.A(I,J)
PRO.A01	PRO.A(1,1)
PRO.A02	PRO.A(1,2)
PRO.A03	PRO.A(1,3)
PRO.A04	PRO.A(2,1)
PRO.A05	PRO.A(2,2)
PRO.A06	PRO.A(2,3)
PRO.A07	PRO.A(3,1)
PRO.A08	PRO.A(3,2)
PRO.A09	PRO.A(3,3)
PRO.A10	PRO.A(4,1)
PRO.A11	PRO.A(4,2)
PRO.A12	PRO.A(4,3)

B.4.5 ARCHIVO GENERADO PARA MPSX

```
        TITLE('M O D E L O  ASIGN')
        MOVE(XDATA,'ASIGN')MOVE(XPBNAME,'PBFILE')
        MOVE(XMINMAX,'MIN')
        CONVERT
        SETUP
        MOVE(XOBJ,'COSTO')
        MOVE(XRHS,'RESTRICS')
        WRITE('1H1')
        OPTIMIZE
        SOLUTION('FILE','SOLUCION')
        EXIT
        PEND
NAME            ASIGN
ROWS
 N  COSTO
 L  COM1
 L  COM2
 L  COM3
 L  COM4
 E  PROY1
 E  PROY2
 E  PROY3
COLUMNS
    PRO.A01   COSTO     5              COM1       1
    PRO.A01   PROY1     1
    PRO.A02   COSTO     13             COM1       1
    PRO.A02   PROY2     1
    PRO.A03   COSTO     19             COM1       1
    PRO.A03   PROY3     1
```

```
    PRO.A04   COSTO     13          COM2      1
    PRO.A04   PROY1     1
    PRO.A05   COSTO     10          COM2      1
    PRO.A05   PROY2     1
    PRO.A06   COSTO     15          COM2      1
    PRO.A06   PROY3     1
    PRO.A07   COSTO     11          COM3      1
    PRO.A07   PROY1     1
    PRO.A08   COSTO     15          COM3      1
    PRO.A08   PROY2     1
    PRO.A09   COSTO     27          COM3      1
    PRO.A09   PROY3     1
    PRO.A10   COSTO     15          COM4      1
    PRO.A10   PROY1     1
    PRO.A11   COSTO     9           COM4      1
    PRO.A11   PROY2     1
    PRO.A12   COSTO     6           COM4      1
    PRO.A12   PROY3     1
RHS
    RESTRICS  COM1      1           COM2      1
    RESTRICS  COM3      1           COM4      1
    RESTRICS  PROY1     1           PROY2     1
    RESTRICS  PROY3     1
ENDATA
/*
```

B.4.6 SOLUCION DEL MODELO

FUNCION OBJETIVO

COSTO = 21

VARIABLES

COLUMNA	VARIABLE	VALOR	DESCRIPCION
PRO.A01	PRO.A(1,1)	1	SE ASIGNA A LA COMPAÑIA IMA EL PROYECTO DE AGUA POTABLE
PRO.A05	PRO.A(2,2)	1	SE ASIGNA A LA COMPAÑIA DER EL PROYECTO DEL DRENAJE PROFUNDO
PRO.A12	PRO.A(4,3)	1	SE ASIGNA A LA COMPAÑIA ACE EL PROYECTO DEL TREN BALA

RESTRICCIONES

HILERA	RESTRICCION	VALOR	DESCRIPCION
COM1	COM(1)	1	NUMERO DE PROYECTOS ASIGNADOS A LA COMPAÑIA IMA
COM2	COM(2)	1	NUMERO DE PROYECTOS ASIGNADOS A LA COMPAÑIA DER
COM3	COM(3)	0	NUMERO DE PROYECTOS ASIGNADOS A LA COMPAÑIA CON
COM4	COM(4)	1	NUMERO DE PROYECTOS ASIGNADOS A LA COMPAÑIA ACE
PROY1	PROY(1)	1	NUMERO DE VECES QUE FUE ASIGNADO EL PROYECTO AGUA POTABLE
PROY2	PROY(2)	1	NUMERO DE VECES QUE FUE ASIGNADO EL PROYECTO DRENAJE PROFUNDO
PROY3	PROY(3)	1	NUMERO DE VECES QUE FUE ASIGNADO EL PROYECTO TREN BALA

B.5 EJEMPLO #5

B.5.1 DESCRIPCION DEL PROBLEMA

El servicio de parques nacionales planea desarrollar una zona campestre para el turismo. Se han señalado cuatro sitios para llegar en automóvil. Estos sitios y las distancias entre ellos, se presentan en la siguiente tabla.

Distancias en Kilómetros

	Entrada al Parque	Cascada	Formación Rocosa	Mirador
Entrada al parque	--	7	10	10
Cascada	7	--	6	4
Formación rocosa	9	6	--	3
Mirador	10		3	--

A efecto de completar la información para el folleto de presentación del parque, se requiere calcular el recorrido de mínima distancia, que iniciando en la entrada, visite todos los puntos de interés y finalice en la misma entrada.

B.5.2 FORMULACION ALGEBRAICA DEL PROBLEMA

Sean:

Indices

$n \in \{1,2,3,4\}$

Los índices asociados al conjunto de lugares turísticos { entrada al parque, cascada, formación rocosa, mirador } respectivamente.

$i \in \{1,2,3,4\}$

Los índices asociados al conjunto de lugares turísticos que definen el origen de un recorrido o tramo.

$j \in \{1,2,3,4\}$

Los índices asociados al conjunto de lugares turísticos que representan el destino de un recorrido o tramo

$s \in \{1,2,3,4\}$

Los índices asociados a la secuencia de recorrido de los tramos.

Variables

$TRAMO_{ijs}$ La variable binaria que representa al tramo recorrido del origen i al destino j en el momento s.

Parámetros

$DIST_{ij}$ La distancia del origen i al destino j.

Minimizar:

Función Objetivo

$$DIS.REC = \sum_i \sum_j \sum_s DIST_{ij} * TRAMO_{ijs}$$

Distancia total del recorrido.

Sujeto a:

Restricciones

$\sum_i \sum_j TRAMO_{ijs}=1$ para toda s

Asegura que en cada secuencia de decisión sólo se recorra un tramo.

$\sum_j TRAMO_{ijs}=1$ para i = 1 y s = 1

Asegura que cualquier recorrido inicie en la entrada al parque (i=1) (s=1).

$\sum_i \sum_s TRAMO_{ijs} \geq 1$ para toda j

Asegura que todos los lugares turísticos sean visitados.

$\sum_i TRAMO_{i(j-n)(s-1)} - \sum_j TRAMO_{(i-n)js} = 0$ para toda s, para toda n > 1

Asegura que se salga del último nodo al que se llegó.

Dados:

Datos

i	j	$DIST_{i,j}$
1	2	7
1	3	10
1	4	10
2	1	7
2	3	6
2	4	4
3	1	9
3	2	6
3	4	3
4	1	10
4	2	4
4	3	3

B.5.3 ESPECIFICACION EN EL LENGUAJE GEMOLI

```
<MOD> VIAJE

<PAQ> XA

     $ DEFINICION DE INDICES

<IND>       N,"NODO",4
      <SP> { 1,"ENTRADA AL PARQUE";2,"CASCADA";
             3,"FORMACION ROCOSA";4,"MIRADOR"}
<IND>       I,"ORIGEN",4
      <SP> { 1,"ENTRADA AL PARQUE";2,"CASCADA";
             3,"FORMACION ROCOSA";4,"MIRADOR"}
<IND>       J,"DESTINO",4
      <SP> { 1,"ENTRADA AL PARQUE";2,"CASCADA";
             3,"FORMACION ROCOSA";4,"MIRADOR"}
<IND>       S,"SECUENCIA DE DECISION",4

     $  DEFINICION DE VARIABLES

<VAR>   TRAMO(I,J,S),"TRAMO RECORRIDO AL MOMENTO S"

     $  DEFINICION DE PARAMETROS

<PARAM> DIST(I,J),"DISTANCIA DE I A J "

     $  FUNCION OBJETIVO MINIMA DISTANCIA RECORRIDA

<RES>       DIS.REC,"DISTANCIA RECORRIDA "
<EC>  SUM((I=[1,4];J=[1,4];S=[1,4]),
            DIST(I;J)* TRAMO(I;J;S)) <MIN> DIS.REC
```

```
     $  DEFINICION DE RESTRICCIONES

     $  CONDICION DE INICIO

<RES>       INICIO,"PRIMERA DECISION"
<EC> SUM((I=1;J=[1,4];S=1),TRAMO(I;J;S)) <IG> 1

     $  ASEGURA UN SOLO TRAMO POR SECUENCIA

<RES>        TRAMUN(S),"TRAMO UNICO EN CADA SECUENCIA"
<EC> SUM((I=[1,4];J=[1,4]),TRAMO(I;J;S)) <IG> 1

     $  ASEGURA QUE TODOS LOS NODOS SEAN TOCADOS

<RES>          DESTNOS(J),"NODOS A LOS QUE LLEGA "
<EC> SUM((I=[1,4];S=[1,4]),TRAMO(I;J;S)) <MAI> 1

     $   ASEGURA QUE SE SALGA DEL ULTIMO NODO AL QUE SE LLEGO

<RES>           SALE(N,S),"NODO DEL QUE SALE EN LA SECUENCIA S"
<DOM> (N=[1,4];S=[2,4])
<EC> SUM((I=[1,4]),TRAMO(I;J=N;S-1))
   - SUM((J=[1,4]),TRAMO(I=N;J;S)) <IG> 0

     $ DEFINICION DE ARCHIVO

<ARCH_E> "ARCVIAJE",3 <SEC> <CAMPO>(1) <CAMPO>(2)
<DATO>  DIST(I=<CAMPO>(1);J=<CAMPO>(2))=<CAMPO>(3)
<FIN>
```

B.5.4 EQUIVALENCIA HILERAS/RESTRICCIONES Y COLUMNAS/VARIABLES

M O D E L O VIAJE

EQUIVALENCIAS ENTRE HILERAS Y RESTRICCIONES

HILERAS	RESTRICCIONES	RANGOS	NOMBRES
F U N C I O N O B J E T I V O			
DIS.REC	MINIMO DIS.REC		
R E S T R I C C I O N E S			
	INICIO PRIMERA DECISION		
INICIO	= 1		INICIO

TRAMUN(S)
TRAMO UNICO EN CADA SECUENCIA

TRAMUN1	= 1	TRAMUN(1)
TRAMUN2	= 1	TRAMUN(2)
TRAMUN3	= 1	TRAMUN(3)
TRAMUN4	= 1	TRAMUN(4)

DESTNOS(J)
NODOS A LOS QUE LLEGA

DESTNOS1	>= 1	DESTNOS(1)
DESTNOS2	>= 1	DESTNOS(2)
DESTNOS3	>= 1	DESTNOS(3)
DESTNOS4	>= 1	DESTNOS(4)

SALE(N,S)
NODO DEL QUE SALE EN LA SECUENCIA S

SALE05	= 0	SALE(2,1)
SALE06	= 0	SALE(2,2)
SALE07	= 0	SALE(2,3)
SALE08	= 0	SALE(2,4)
SALE09	= 0	SALE(3,1)
SALE10	= 0	SALE(3,2)
SALE11	= 0	SALE(3,3)
SALE12	= 0	SALE(3,4)
SALE13	= 0	SALE(4,1)
SALE14	= 0	SALE(4,2)
SALE15	= 0	SALE(4,3)
SALE16	= 0	SALE(4,4)

EQUIVALENCIAS ENTRE COLUMNAS Y VARIABLES

TRAMO(I,J,S)
TRAMO RECORRIDO AL MOMENTO S

TRAMO05	TRAMO(1,2,1)
TRAMO06	TRAMO(1,2,2)
TRAMO07	TRAMO(1,2,3)
TRAMO08	TRAMO(1,2,4)
TRAMO09	TRAMO(1,3,1)
TRAMO10	TRAMO(1,3,2)
TRAMO11	TRAMO(1,3,3)
TRAMO12	TRAMO(1,3,4)
TRAMO13	TRAMO(1,4,1)
TRAMO14	TRAMO(1,4,2)
TRAMO15	TRAMO(1,4,3)
TRAMO16	TRAMO(1,4,4)
TRAMO17	TRAMO(2,1,1)

TRAMO18	TRAMO(2,1,2)
TRAMO19	TRAMO(2,1,3)
TRAMO20	TRAMO(2,1,4)
TRAMO25	TRAMO(2,3,1)
TRAMO26	TRAMO(2,3,2)
TRAMO27	TRAMO(2,3,3)
TRAMO28	TRAMO(2,3,4)
TRAMO29	TRAMO(2,4,1)
TRAMO30	TRAMO(2,4,2)
TRAMO31	TRAMO(2,4,3)
TRAMO32	TRAMO(2,4,4)
TRAMO33	TRAMO(3,1,1)
TRAMO34	TRAMO(3,1,2)
TRAMO35	TRAMO(3,1,3)
TRAMO36	TRAMO(3,1,4)
TRAMO37	TRAMO(3,2,1)
TRAMO38	TRAMO(3,2,2)
TRAMO39	TRAMO(3,2,3)
TRAMO40	TRAMO(3,2,4)
TRAMO45	TRAMO(3,4,1)
TRAMO46	TRAMO(3,4,2)
TRAMO47	TRAMO(3,4,3)
TRAMO48	TRAMO(3,4,4)
TRAMO49	TRAMO(4,1,1)
TRAMO50	TRAMO(4,1,2)
TRAMO51	TRAMO(4,1,3)
TRAMO52	TRAMO(4,1,4)
TRAMO53	TRAMO(4,2,1)
TRAMO54	TRAMO(4,2,2)
TRAMO55	TRAMO(4,2,3)
TRAMO56	TRAMO(4,2,4)
TRAMO57	TRAMO(4,3,1)
TRAMO58	TRAMO(4,3,2)
TRAMO59	TRAMO(4,3,3)

B.5.5 SOLUCION DEL MODELO

FUNCION OBJETIVO

DIS.REC = 23

VARIABLES

COLUMNA	VARIABLE	VALOR	DESCRIPCION
TRAMO05	TRAMO(1,2,1)	1	RECORRE DEL NODO 1 AL NODO 2 EN EL MOMENTO 1
TRAMO30	TRAMO(2,4,2)	1	RECORRE DEL NODO 2 AL NODO 4 EN EL MOMENTO 2

COLUMNA	VARIABLE	VALOR	DESCRIPCION
TRAMO59	TRAMO(4,3,3)	1	RECORRE DEL NODO 4 AL NODO 3 EN EL MOMENTO 3
TRAMO36	TRAMO(3,1,4)	1	RECORRE DEL NODO 3 AL NODO 1 EN EL MOMENTO 4

RESTRICCIONES

HILERA	RESTRICCION	VALOR	DESCRIPCION
INICIO	INICIO	1	ASEGURA QUE SE ARRANQUE DEL NODO 1 AL MOMENTO 1
TRAMUN1	TRAMUN(1)	1	ASEGURA QUE EN EL MOMENTO 1 SOLO SE RECORRA UN TRAMO
TRAMUN2	TRAMUN(2)	1	ASEGURA QUE EN EL MOMENTO 2 SOLO SE RECORRA UN TRAMO
TRAMUN3	TRAMUN(3)	1	ASEGURA QUE EN EL MOMENTO 3 SOLO SE RECORRA UN TRAMO
TRAMUN4	TRAMUN(4)	1	ASEGURA QUE EN EL MOMENTO 4 SOLO SE RECORRA UN TRAMO
DESTNOS1	DESTNOS(1)	1	ASEGURA QUE EL NODO 1 FUE VISITADO
DESTNOS2	DESTNOS(2)	1	ASEGURA QUE EL NODO 2 FUE VISITADO
DESTNOS3	DESTNOS(3)	1	ASEGURA QUE EL NODO 3 FUE VISITADO
DESTNOS4	DESTNOS(4)	1	ASEGURA QUE EL NODO 4 FUE VISITADO
SALE05	SALE(2,1)	0	LLEGADAS MENOS SALIDAS EN EL NODO 2 AL MOMENTO 1
SALE06	SALE(2,2)	0	LLEGADAS MENOS SALIDAS EN EL NODO 2 MOMENTO 2
SALE07	SALE(2,3)	0	LLEGADAS MENOS SALIDAS EN EL NODO 2 AL MOMENTO 3
SALE08	SALE(2,4)	0	LLEGADAS MENOS SALIDAS EN EL NODO 2 AL MOMENTO 4
SALE09	SALE(3,1)	0	LLEGADAS MENOS SALIDAS EN EL NODO 3 AL MOMENTO 1
SALE10	SALE(3,2)	0	LLEGADAS MENOS SALIDAS EN EL NODO 3 AL MOMENTO 2
SALE11	SALE(3,3)	0	LLEGADAS MENOS SALIDAS EN EL NODO 3 AL MOMENTO 3
SALE12	SALE(3,4)	0	LLEGADAS MENOS SALIDAS EN EL NODO 3 AL MOMENTO 4
SALE13	SALE(4,1)	0	LLEGADAS MENOS SALIDAS EN EL NODO 4 AL MOMENTO 1
SALE14	SALE(4,2)	0	LLEGADAS MENOS SALIDAS EN EL NODO 4 AL MOMENTO 2
SALE15	SALE(4,3)	0	LLEGADAS MENOS SALIDAS EN EL NODO 4 AL MOMENTO 3
SALE16	SALE(4,0)	0	LLEGADAS MENOS SALIDAS EN EL NODO 4 AL MOMENTO 4

B.6 EJEMPLO #6

B.6.1 DESCRIPCION DEL PROBLEMA

Se tienen dos buquetanques (BUQUE_1, BUQUE_2) dedicados a transportar combustóleo desde una refinería (REFINERIA) en la costa, a dos puertos estratégicos (PUERTO_2, PUERTO_3) por la amplitud de sus áreas de influencia y por el volumen de la demanda que esto representa.

El primero de los buquetanques (BUQUE_1) es el más grande y puede transportar 200,000 barriles de producto, el segundo buque (BUQUE_2) puede transportar solamente 100,000 barriles. Ambos buques recorren en promedio 864 millas náuticas por periodo.

Los costos relativos de los buques navegando en cada ruta son los siguientes:

Para el BUQUE_1:

DESTINO:	REFINERIA	PUERTO_2	PUERTO_3	INTERMEDIO
ORIGEN:				
REFINERIA	---	5.2	10.4	2.6
PUERTO_2	3.9	---	5.2	2.6
PUERTO_3	9.1	5.2	---	2.6
INTERMEDIO	2.6	2.6	2.6	---

Para el BUQUE_2:

DESTINO:	REFINERIA	PUERTO_1	PUERTO_2	INTERMEDIO
ORIGEN:				
REFINERIA	---	4.0	8.0	2.0
PUERTO_1	3	---	4	2
PUERTO_2	7	4	+	2
INTERMEDIO	2	2	2	-

El PUERTO "INTERMEDIO" es un puerto ficticio, equidistante de cada puerto en el equivalente a un periodo, necesario para iniciar o cerrar el modelo a un número fijo de periodos.

Los costos relativos de los buques en puerto para cada periodo, ya sea cargando, descargando o en espera, son en promedio de 1.3 y 1.0 para el BUQUE_1 y el BUQUE_2 respectivamente.

Para cada puerto habrá que hacer llegar cargamentos de combustóleo tales que satisfagan la demanda regional y se mantengan los niveles de existencia dentro de la banda de operación definida por .2 * (Capacidad física de almacenamiento)

como el límite inferior y .8 * (Capacidad física de almacenamiento) como límite superior; siendo:

	MILES DE BARRILES	
	PUERTO_2	PUERTO_3
DEMANDA PROMEDIO POR PERIODO	20	50
CAPACIDAD FISICA DE ALMACENAMIENTO	400	500

Si con los recibos de buque no se satisfacen estas condiciones, se pueden complementar con recibos por otros medios cuyo costo relativo en cada puerto para cada 1000 barriles es el siguiente:

	PUERTO_2	PUERTO_3
COSTO DE OTROS MEDIOS	20	60

B.6.2 DESCRIPCION DEL MODELO

B.6.2.1 INDICES

El índice PUERTO representa los tres puertos reales y uno ficticio.

El índice ORIGEN se utiliza para representar el puerto de origen de un tramo o trayecto recorrido por el buque.

El índice DESTINO representa el puerto destino de un trayecto.

Por necesidades de simplificación del modelo, PERIODO en este modelo representa un periodo de 72 horas o 3 días calendario.

PER_IN representa el periodo de inicio de las operaciones.

PER_FIN representa el periodo terminal de las operaciones.

BUQUE es el índice que representa a los buquetanques.

LOTE representa el tamaño de las cargas y/o descargas de los buques.

B.6.2.2 VARIABLES

La variable RTR (Ruta recorrida) es una variable binaria que indica para cada buque (BUQUE) los tramos de recorrido (ORIGEN, DESTINO) y el periodo en el que llega al puerto destino (PER FIN). Variable que en la solución óptima conforma el recorrido óptimo.

Una forma más simple y evidente hubiera sido asociar a la variable RTR (Ruta recorrida) además de los índices ya mencionados, el periodo en que se inició la ruta (PER_FIN) sin embargo, se prefirió la manera ya descrita para ejemplificar en todo su potencial, el manejo de conjuntos, en especial, el manejo que se hace del conjunto TVV (Tiempos de viaje válidos) en la restricción MV (Movimiento de buques), y que se describe al detalle posteriormente.

La variable DSC (Descarga de combustóleo en puerto) es una variable binaria que indica para cada buque (BUQUE) el tamaño de la descarga (LOTE) el puerto en que se realiza (DESTINO) y el periodo en que se efectúa (PER_FIN).

CARGA es una variable binaria que representa el tamaño de las cargas (LOTE) que cada buquetanque (BUQUE) realiza en la refinería, así como el periodo en que se realiza (PER_FIN).

La variable CR_D (Carga disponible) representa la carga existente en cada buquetanque (BUQUE) al inicio del periodo indicado por PER_FIN.

OMD (Suministro por otros medios) representa la cantidad de combustóleo que habrá que hacer llegar a cada puerto (DESTINO) en el periodo indicado por PER_FIN para completar los envíos por barco y cumplir con las restricciones de demanda y de nivel de existencia.

La variable EX_P (Existencia en puerto) indica la cantidad de producto en existencia al inicio de periodo (PER_FIN) en cada puerto (PUERTO).

Por necesidades de la restricción BMP (Balance de material en puertos), en el manejo y signo de la demanda, se definió la variable DEM_P. A esta variable se le asignan los valores de la demanda mediante la restricción DEM (Demanda en cada puerto) y el parámetro DEMANDA.

B.6.2.3 CONJUNTOS

El conjunto OR_D_DT (Origen diferente de destino) define los pares ordenados de puertos, tales que, se cumple que el puerto ORIGEN es diferente al puerto DESTINO.

El conjunto OR_I_DT es el conjunto de pares en donde el puerto ORIGEN es igual al puerto DESTINO.

Estos dos conjuntos sirven para identificar si un BUQUE está navegando o está en puerto, en el primer caso el ORIGEN de la ruta debe ser distinto al DESTINO de la misma.

El conjunto BUQ_LT lo conforman las parejas BUQUE_LOTE que identifican los tamaños de lote que cada buque puede descargar. Nótese en este caso, que la combinación (BUQUE=2, LOTE=3) y (BUQUE=2, LOTE=4) no existen en el conjunto, por ser la capacidad máxima del buque 2 solamente de 100 mil barriles, y el tamaño de los lotes 3 y 4 son de 150 y 200 mil respectivamente.

En el conjunto BUQ_PT se declara la pareja (BUQUE=1, DESTINO=2) como inválida, ya que esta ausencia representa la imposibilidad del BUQUE_1 de atracar en el PUERTO_2.

El conjunto B_D_L determina la tercia válida de Buque-Destino-Lote, es decir, indica los destinos válidos y los tamaños de lote que puede manejar el buque en tales destinos.

El conjunto TVV (Tiempos de viaje válidos) es la cuarteta en donde se establece para cada ruta (ORIGEN,DESTINO), la duración del viaje en función del periodo donde se inicia el viaje (PER

IN) y el periodo en que llega a puerto (PER_FIN).

Si se observa el cálculo del parámetro DURV (Duración de viaje) para cada ruta se podrá comprobar que éste se calculó en función de la distancia entre puertos, de la velocidad promedio de los buques y del tiempo promedio requerido para descargar el producto en el destino de la ruta.

B.6.2.4 FUNCION OBJETIVO

La función objetivo representa el costo relativo de la distribución del combustible, y está constituido por tres componentes:

- El costo de las rutas navegadas, calculado en base a la ruta recorrida (variable RTR) y el costo de la ruta navegada (parámetro CTR)

- El costo de los buques en puerto, calculado en base a las rutas recorridas (variable RTR) tales que el puerto origen es igual al destino, multiplicado por el costo de buque cuando está atracado (parámetro CTM).

- El costo de llevar combustóleo por otros medios diferentes al buque, calculándose multiplicando el suministro por otros medios (variable OMD) por el costo de otros medios (parámetro COM).

B.6.2.5 RESTRICCIONES

La restricción RUI (Ruta inicial) sirve para establecer la posición de inicio de los buques, definiendo el ORIGEN, el DESTINO y período de arribo (PER_FIN) del primer recorrido de cada buque, mediante la asignación del valor 1 al parámetro RTI (Ruta inicial) correspondiente.

La restricción VIU (Viaje único) asegura que cada BUQUE viaje sobre una ruta y sólo una en cada periodo.

La restricción MV (Asegura el movimiento del buque) establece que cada buque que llega a un puerto, inicie al siguiente periodo una nueva ruta, con lo cual se mantiene el buque en "movimiento"; mediante el conjunto TVV se asegura que la nueva ruta sea consistente en tiempo con la ruta inmediata anterior.

Dado que el parámetro DURV (Duración de viaje) incluye el tiempo de navegación más el tiempo de descarga en el puerto destino, es necesario considerar por separado el tiempo requerido para cargar el barco en la refinería, por esta razón, toda vez que un buque llega a la refinería, se le obliga a permanecer un periodo en la refinería, tiempo suficiente para cargar el barco. La restricción TC (Tiempo de carga) asegura que el buque permanecerá cargando en la refinería, en el siguiente periodo a su arribo a la misma.

La restricción CAR_R (Carga en refinerías) obliga a los buques a cargar, (la variable CARGA toma un valor) cuando el buque permanece un periodo en la refinería.

Las restricciones LLDP (Arriba para descargar) obliga a los buques a descargar (la variables DSC toma un valor) cada vez que arriban a PUERTO_2 o PUERTO_3, y la restricción LLD asegura que la variable DSC tome un valor consistente con el conjunto de triadas válidas BUQUE_DESTINO_LOTE.

La restricción BMI (Carga inicial del barco) obliga a que la carga disponible (CR_D) en el barco para el primer periodo, sea igual a la carga inicial del mismo (parámetro CR_INC).

La restricción CND_D (Condición para descargar) asegura que el tamaño del lote indicado para la descarga (variable DSC) sea menor que la carga disponible en el barco (CR_D).

La restricción BMD (Balance de material del barco descargando) calcula la carga disponible en el buque (CR_D) al inicio de cada periodo, restando a la carga disponible en el periodo anterior, el lote descargado.

La restricción CMB (Carga máxima en el buque) asegura que el valor de la carga disponible en el buque (CR_D) siempre sea menor o igual al límite de carga de buque (parámetro LIM_C).

Como ya se mencionó anteriormente, la demanda de producto por periodo-puerto introducido al modelo mediante el parámetro DEMANDA se requiere como variable en la restricción BMP (Balance de material en puertos), por esta razón la restricción DEM (Demanda en cada puerto) lo único que realiza es asignarle a la variable DEM_P (Demanda en puerto) los valores del parámetro DEMANDA.

La restricción XIP (Existencia inicial en puerto) le asigna a la variable EX_P (Existencia en puerto) para el primer periodo, los valores del parámetro EX_IN_P (Existencia inicial en puerto).

La restricción BMP (Balance de material en puertos) calcula la existencia en puertos al inicio de cada periodo (mayor que 1) a partir de la existencia del periodo anterior, de todas las llegadas ya sean por buque (TLOTE * DSC) o por otros medios (OMD) y de las salidas para satisfacer la demanda (DEM_P).

La restricción LA_P (Límite de almacenamiento en puerto) asegura que la existencia en puerto (EX_P) permanezca dentro de los rangos de operación de la tanquería.

B.6.2.6 Reglas de Consistencia

La regla de consistencia "CARGA INICIAL ≤ LIMITE DE CAPACIDAD" asegura que la carga inicial del buque (parámetro CR_INC) sea menor o igual que la capacidad límite de carga del barco.

Si esta condición no se cumple, la restricción CMB (Carga máxima en buque) nunca se cumplirá y el modelo no encontrará solución factible.

Las reglas de consistencia "EXISTENCIA INICIAL ≤ LIMITE SUPERIOR DE OPERACION" y "EXISTENCIA INICIAL ≥ LIMITE INFERIOR DE OPERACION" aseguran que el valor del parámetro EX_IN_P (Existencia inicial en puerto) sea consistente con la restricción LA_P (Límite de almacenamiento en puerto).

B.6.3 ESPECIFICACION EN LENGUAJE GEMOLI

```
<MOD>  COMBUS65

<PAQ>  IBM
<FO>   OBJETIVO
```

```
        <RS>   RESTRCS
        <RG>   RANGOS
        <FR>   FRONTERS

                $  I   N   D   I   C   E   S

<IND>  PUERTO, "PUERTO", 4

       <SP>    {1,"REFINERIA"; 2,"PUERTO_2"; 3,"PUERTO_3";
                4,"INTERMEDIO"}

<IND>  ORIGEN,"PUERTO ORIGEN",4
       <SC> PUERTO <INC> {[1,4]}

<IND>  DESTINO,"PUERTO DESTINO",4
       <SC> PUERTO <INC> <IGD> ORIGEN

<IND>  PERIODO,"PERIODO",12

<IND>  PER_IN,"PERIODO INICIAL",12

<IND>  PER_FIN,"PERIODO FINAL",12

<IND>  BUQUE,"BUQUETANQUE",2

       <SP>    {1,"BUQUE_1";2,"BUQUE_2"}

<IND>  LOTE,"TAMAÑO DEL LOTE",4

                $ V A R I A B L E S

<VAR> <BIN> RTR(BUQUE,ORIGEN,DESTINO,PER_FIN),"RUTA RECORRIDA"

<VAR> <BIN> DSC(BUQUE,DESTINO,PER_FIN,LOTE),"DESCARGA"

<VAR> <BIN> CARGA(BUQUE,PER_FIN,LOTE),"CARGA EN REFINERIAS"

<VAR> CR_D(BUQUE,PER_FIN),"CARGA DISPONIBLE EN EL BUQUE AL
                           INICIO DEL PERIODO"

<VAR> OMD(DESTINO,PER_FIN),"SUMINISTRO POR OTROS MEDIOS"

<VAR> EX_P(PUERTO,PER_FIN),"EXISTENCIA EN PUERTO AL
                            INICIO DEL PERIODO"

<VAR> DEM_P(PUERTO,PER_FIN),"DEMANDA EN PUERTO"

                $ P A R A M E T R O S

<PARAM> <ENT> RTI(BUQUE,ORIGEN,DESTINO,PER_FIN),"RUTA INICIAL"

<PARAM> CR_INC(BUQUE),"CARGA INICIAL"
```

```
<PARAM> TLOTE(LOTE),"TAMAÑO DEL LOTE DE DESCARGA"

<PARAM> T_LT_C(BUQUE,LOTE),"TAMAÑO DEL LOTE DE CARGA"

<PARAM> CTR(BUQUE,ORIGEN,DESTINO),"COSTO DE LA RUTA"

<PARAM> CTM(BUQUE,DESTINO),"COSTO EN MUELLE"

<PARAM> COM(DESTINO),"COSTO DE OTROS MEDIOS"

<PARAM> LIM_C(BUQUE,PER_FIN),"LIMITE CARGA"

<PARAM> DIST(ORIGEN,DESTINO),"DISTANCIA"

<PARAM> TDSC(DESTINO),"TIEMPO DESCARGA"

<PARAM> <ENT> <REDONDEADO> DURV(ORIGEN,DESTINO),"DURACION VIAJE"
              = (DIST(ORIGEN;DESTINO) / 864 + TDSC(DESTINO))

<PARAM> DEMDA(PUERTO,PER_FIN),"DEMANDA POR PERIODO"

<PARAM> EX_IN_P(PUERTO,PER_FIN),"EXISTENCIA INICIAL EN PUERTOS"

<PARAM> CAP_AL(PUERTO,PER_FIN),"CAPACIDAD DE ALMACENAMIENTO EN
                                EL PUERTO"

<PARAM> FICT1(PUERTO,PER_FIN),"OBLIGA LA DEFINICION EXPLICITA
                               DEL COEFICIENTE"

<PARAM> FICT2(BUQUE,PER_FIN),"OBLIGA LA DEFINICION EXPLICITA DEL
                              COEFICIENTE"

<PARAM> FICT3(BUQUE,ORIGEN,DESTINO,PER_FIN),"OBLIGA LA
                         DEFINICION EXPLICITA DEL COEFICIENTE"

<PARAM> FICT4(BUQUE,DESTINO,PER_FIN,LOTE),"OBLIGA LA DEFINICION
                                    EXPLICITA DEL COEFICIENTE"
```

$ C O N J U N T O S

```
<CONJ> TVV(ORIGEN,DESTINO,PER_IN,PER_FIN),"TIEMPOS DE VIAJE
                                           VALIDOS"
       = (((PER_FIN - PER_IN) <IG> DURV(ORIGEN;DESTINO)) <Y>
                                  (ORIGEN <DIF> 4))

<CONJ> OR_D_DT(ORIGEN,DESTINO),"ORIGEN <DIF> DESTINO"
             = (ORIGEN <DIF> DESTINO)

<CONJ> OR_I_DT(ORIGEN,DESTINO),"ORIGEN IGUAL DESTINO"
            <COMPL> OR_D_DT(ORIGEN;DESTINO)
```

```
<CONJ> BUQ_LT(BUQUE,LOTE),"TAMAÑO LOTE VALIDO/BUQUE"
            = {1,1; 1,2; 1,3; 1,4; 2,1; 2,2}

<CONJ> BUQ_PT(BUQUE,DESTINO),"PUERTOS VALIDOS/BUQUE"

            = {1,1; 1,3; 1,4; 2,1; 2,2; 2,3; 2,4}

<CONJ> B_D_L(BUQUE,DESTINO,LOTE),"TRIPLE VALIDO"

          = (BUQ_PT(BUQUE;DESTINO) <Y> BUQ_LT(BUQUE;LOTE))


                  $ F U N C I O N   O B J E T I V O


                  $ COSTO TOTAL DE EMBARQUES

<RES> COSTO,"COSTO DE EMBARQUES"

                  $ COSTO DE BUQUES NAVEGANDO

<EC>  SUM((BUQUE=[1,2]; ORIGEN=[1,4]; DESTINO = [1,4];
                                      PER_FIN = [1,12]),
      CTR(BUQUE; ORIGEN; DESTINO)*RTR(BUQUE;ORIGEN;DESTINO;
                                      PER_FIN)
      <ELE> OR_D_DT(ORIGEN;DESTINO)  )


                  $ COSTO DE BUQUES EN PUERTO

   +  SUM((BUQUE = [1,2]; ORIGEN = [1,4]; DESTINO = [1,4];
                                         PER_FIN = [1,12]),
      CTM(BUQUE;DESTINO) * RTR(BUQUE; ORIGEN; DESTINO; PER_FIN)
      <ELE> OR_I_DT(ORIGEN; DESTINO)  )


                  $ COSTO DE OTROS MEDIOS

   +  SUM((DESTINO=[2,3]; PER_FIN=[2,12]),
           COM(DESTINO)*OMD(DESTINO;PER_FIN))

   <MIN> COSTO


                  $ R E S T R I C C I O N E S

                  $  RUTA INICIAL DE BUQUES


<RES> RUI(BUQUE,ORIGEN,DESTINO,PER_FIN),"RUTA AL INICIO DE LA
                                        SIMULACION"
```

```
<EC>  RTR(BUQUE;ORIGEN;DESTINO;PER_FIN)
<IG>  RTI(BUQUE;ORIGEN;DESTINO;PER_FIN)

                $ ASEGURA QUE PARA TODO BUQUE_PERIODO
                $  SOLO EXISTA UN VIAJE

<RES> VIU(BUQUE,PER_FIN),"VIAJE UNICO"

<EC>  SUM((ORIGEN=[1,4]; DESTINO=[1,4]),
        RTR(BUQUE;ORIGEN;DESTINO;PER_FIN))
<MEI> 1

        $ ASEGURA QUE CADA BUQUE PERMANEZCA EN MOVIMIENTO

<RES> MV(BUQUE,PUERTO,PERIODO),"ASEGURA MOVIMIENTO DEL BUQUE"

<DOM> (BUQUE=[1,2]; PUERTO=[1,4]; PERIODO=[1,11])

<EC>  SUM((ORIGEN=[1,4]),RTR(BUQUE;ORIGEN;DESTINO=PUERTO;
                        PER_FIN=PERIODO))

    - SUM((DESTINO=[1,4]; PER_FIN = [1,12]),
        RTR(BUQUE;ORIGEN=PUERTO;DESTINO;PER_FIN)
      <ELE>  TVV(ORIGEN;DESTINO;PER_IN=PERIODO;PER_FIN))

 <IG> 0

                $ ASEGURA CUANDO MENOS UN PERIODO DE
                $        CARGA EN REFINERIAS

<RES> TC(BUQUE,PUERTO,PERIODO),"TIEMPO DE CARGA EN REFINERIAS"
      <SC> MV <INC> <DOM> (BUQUE=[1,2]; PUERTO=1;
                        PERIODO=[1,11])

<EC>  SUM((ORIGEN = [1,4]),RTR(BUQUE;ORIGEN;DESTINO=PUERTO;
                                    PER_FIN=PERIODO)
      <ELE>(ORIGEN <DIF> DESTINO))

    - SUM ((DESTINO=[1,4]),RTR(BUQUE;ORIGEN=PUERTO;DESTINO;
                            PER_FIN+1=PERIODO)
      <ELE> OR_I_DT(ORIGEN;DESTINO))

 <IG> 0

                $ OBLIGA A TODO BARCO A CARGAR EN REFINERIAS

<RES> CAR_R(BUQUE,PER_FIN),"CARGA EN REFINERIAS"

<EC>  RTR(BUQUE;ORIGEN=1;DESTINO=1;PER_FIN)
```

```
    - SUM((LOTE=[1,4]),FICT2(BUQUE;PER_FIN) * CARGA(BUQUE;
                              PER_FIN;LOTE))

<IG> 0

                $ OBLIGA A TODO BARCO A DESCARGAR CUANDO
                $      LLEGA A UN PUERTO DIFERENTE A
                $         REFINERIA O "INTERMEDIO"

<RES> LLD(BUQUE,DESTINO,PER_FIN), "ARRIVA PARA DESCARGAR"

<DOM> (BUQUE=[1,2]; DESTINO=[2,3]; PER_FIN=[1,12])

<EC>   SUM((ORIGEN=[1,4]),

       FICT3(BUQUE;ORIGEN;DESTINO;PER_FIN) * RTR(BUQUE;ORIGEN;
                                                 DESTINO;PER_FIN))

     - SUM((LOTE=[1,4]),
       FICT4(BUQUE;DESTINO;PER_FIN;LOTE) * DSC(BUQUE;DESTINO;
                                                 PER_FIN;LOTE)
       <ELE> B_D_L(BUQUE;DESTINO;LOTE) )

<IG>   0

<RES> LLDP(BUQUE,DESTINO,PER_FIN), "ARRIVA PARA DESCARGAR"

<DOM> (BUQUE=[1,2]; DESTINO=[2,3]; PER_FIN=[1,12])

<EC>  SUM((ORIGEN=[1,4]),

      FICT3(BUQUE;ORIGEN;DESTINO;PER_FIN) * RTR(BUQUE;ORIGEN;
                                                 DESTINO;PER_FIN))

    - SUM((LOTE=[1,4]),
      FICT4(BUQUE;DESTINO;PER_FIN;LOTE) * DSC(BUQUE;DESTINO;
                                                 PER_FIN;LOTE))

<IG>  0

                $ BALANCE DE MATERIAL EN EL BARCO
                $  PARA CADA PERIODO

                $ CONDICIONES INICIALES

<RES> BMI(BUQUE),"CARGA INICIAL DEL BARCO"

<EC>  CR_D(BUQUE;PER_FIN=1)

<IG>  CR_INC(BUQUE)
```

```
                $ D E S C A R G A N D O

        $C O N D I C I O N   P A R A   D E S C A R G A R

<RES> CND_D(BUQUE,PER_FIN),"CONDICION PARA DESCARGAR"

<DOM> (BUQUE=[1,2];PER_FIN=[1,12])

<EC>  CR_D(BUQUE;PER_FIN)

    - SUM((DESTINO=[2,3];LOTE=[1,4]),
          TLOTE(LOTE)*DSC(BUQUE;DESTINO;PER_FIN;LOTE))

<MAI> 0

                $B A L A N C E

<RES> BMD(BUQUE,PER_FIN),"BALANCE DE MATERIAL DESCARGANDO"

<DOM> (BUQUE=[1,2];PER_FIN=[1,11])

<EC>  CR_D(BUQUE;PER_FIN+1)

    - CR_D(BUQUE;PER_FIN)

    - SUM((LOTE=[1,4]),T_LT_C(BUQUE;LOTE)*CARGA(BUQUE;PER_FIN;
                                                LOTE))

    + SUM((DESTINO=[2,3];LOTE=[1,4]),
          TLOTE(LOTE)*DSC(BUQUE;DESTINO;PER_FIN;LOTE))

<IG>  0

        $ CAPACIDAD MAXIMA DE CARGA DEL BUQUE

<RES> CMB(BUQUE,PER_FIN),"CARGA MAXIMA EN BUQUE"

<EC>  FICT2(BUQUE;PER_FIN) * CR_D(BUQUE;PER_FIN)

<MEI> LIM_C(BUQUE;PER_FIN)

        $ D E M A N D A   E N   P U E R T O S

<RES> DEM(PUERTO,PER_FIN),"DEMANDA EN CADA PUERTO"

<EC>  FICT1(PUERTO;PER_FIN) * DEM_P(PUERTO;PER_FIN)

<IG>  DEMDA(PUERTO;PER_FIN)
```

```
                    $ BALANCE DE MATERIAL EN PUERTOS

                    $ CONDICIONES INICIALES

<RES>  XIP(PUERTO,PER_FIN),"EXISTENCIAS INICIALES EN PUERTOS"
<DOM> (PUERTO = [1,4];PER_FIN ={1})

<EC>   FICT1(PUERTO;PER_FIN) * EX_P(PUERTO;PER_FIN)
<IG>   EX_IN_P(PUERTO;PER_FIN)

                    $ BALANCE

<RES> BMP(PUERTO,PER_FIN),"BALANCE DE MATERIAL EN PUERTOS"

<DOM> (PUERTO <DIF> {1,4}; PER_FIN = [1,11])

<EC>   FICT1(PUERTO;PER_FIN) * EX_P(PUERTO;PER_FIN+1)

     - EX_P(PUERTO;PER_FIN)

     - SUM((BUQUE=[1,2];LOTE=[1,4]),
           TLOTE(LOTE) * DSC(BUQUE;DESTINO=PUERTO;PER_FIN;LOTE))

     - FICT1(PUERTO;PER_FIN) * OMD(DESTINO=PUERTO;PER_FIN)
     + DEM_P(PUERTO;PER_FIN)

<IG>   0

              $ LIMITES DE ALMACENAMIENTO EN PUERTOS

<RES> LA_P(PUERTO,PER_FIN),"LIMITE DE ALMACENAMIENTO"

<DOM> (PUERTO <DIF> {1,4} ; PER_FIN = [1,12])

<EC>   FICT1(PUERTO;PER_FIN) * EX_P(PUERTO;PER_FIN)

<MAI> (0.2 * CAP_AL(PUERTO;PER_FIN));(.6 * CAP_AL(PUERTO;
                                          PER_FIN))

          $ R E G L A S   D E   C O N S I S T E N C I A

<CONS>"CARGA INICIAL ≤ LIMITE DE CAPACIDAD"
       (LIM_C(BUQUE;PER_FIN=1)
     - CR_INC(BUQUE))

<MAI>  0

<CONS>"EXISTENCIA INICIAL ≤ LIMITE SUPERIOR DE OPERACION"
       ( (.8 * CAP_AL(PUERTO;PER_FIN))
     - EX_IN_P(PUERTO;PER_FIN) )

<MAI>  0
```

```
<CONS> "EXISTENCIA INICIAL ≥LIMITE INFERIOR DE OPERACION"

       ( EX_IN_P(PUERTO;PER_FIN) -

       (.2 * CAP_AL(PUERTO;PER_FIN)) )

<MAI>  0

                    $ V A L O R E S

<VALOR> RTI(BUQUE = [1,2]; ORIGEN = [1,4]; DESTINO = [1,4];
                                            PER_FIN = [1,12]) =
          {168*NULO,1,179*NULO,1,35*NULO}

<VALOR> CR_INC(BUQUE = [1,2]) = {200,100}

<VALOR> TLOTE(LOTE = [1,4]) = { 50,100,150,200 }

<VALOR> T_LT_C(BUQUE=[1,2];LOTE=[1,4]) =
                      { 200,150,100,0, 100,50,0,0 }

<VALOR> CTR(BUQUE = [1,2]; ORIGEN = [1,4]; DESTINO = [1,4]) =
            { NULO    5.2      10.4     2.6
              3.9     NULO     5.2      2.6
              9.1     5.2      NULO     2.6
              2.6     2.6      2.6      NULO
              NULO    4.0      8.0      2.0
              3       NULO     4        2
              7       4        NULO     2
              2       2        2        NULO}

<VALOR> CTM(BUQUE = [1,2]; DESTINO = [1,4]) =
            { 1.3     1.3      1.3      1.3
              1.      1.       1.       1.   }

<VALOR> COM(DESTINO = [2,3]) = { 20,60 }

<VALOR> LIM_C(BUQUE = [1,2]; PER_FIN = [1,12]) =
            { 12*200,12*100 }

<VALOR> DIST(ORIGEN = [1,4]; DESTINO = [1,4]) =
            { 864   864 3600  864
              864 NULO 1800   864
             3600 1800 NULO   864
              864  864 864   NULO  }

<VALOR> TDSC(DESTINO = [1,4]) = { 0,      .4,      .6,      0 }

<VALOR> DEMDA(PUERTO = [2,3]; PER_FIN = [1,12]) =
            { 12*20       12*50 }
```

```
<VALOR> EX_IN_P(PUERTO = [1,4];PER_FIN = {1}) =
          { 400, 150, 300, 0  }

<VALOR> CAP_AL(PUERTO = [1,4];PER_FIN=[1,12]) =
          { 12*NULO, 12*400, 12*500, 12*NULO }

<VALOR> FICT1(PUERTO = [1,4];PER_FIN = [1,12]) = {48*1}

<VALOR> FICT2(BUQUE = [1,2];PER_FIN = [1,12]) = {24*1}

<VALOR> FICT3(BUQUE = [1,2];ORIGEN = [1,4];DESTINO = [1,4];
                                  PER_FIN = [1,12]) = {384*1}

<VALOR> FICT4(BUQUE = [1,2];DESTINO = [1,4];PER_FIN = [1,12];
                                  LOTE = [1,4]) = {384*1}

<FIN>
```

B.6.4 SOLUCION DEL MODELO

FUNCION OBJETIVO

COSTO = 3047

VARIABLES

COLUMNA	VARIABLE	VALOR	DESCRIPCION
RTR169	RTR(1,4,3,1)	1	EL BUQUE 1 RECORRE DE UN PUNTO INTERMEDIO AL PUERTO 3, LLEGANDO Y DESCARGANDO EN ESTE EN EL PERIODO 1
RTR101	RTR(1,3,1,5)	1	EL BUQUE 1 RECORRE DEL PUERTO 3 A LA REFINERIA, LLEGANDO A ESTA EN EL PERIODO 5
RTR006	RTR(1,1,1,6)	1	EL BUQUE 1 PERMANECE EN LA REFINERIA DURANTE EL PERIODO 6 (CARGANDO)
RTR035	RTR(1,1,3,11)	1	EL BUQUE 1 RECORRE DE LA REFINERIA AL PUERTO 3, LLEGANDO Y DESCARGANDO EN ESTE EN EL PERIODO 11
RTR144	RTR(1,3,4,12)	1	EL BUQUE 1 RECORRE DEL PUERTO 3 A UN PUNTO INTERMEDIO, LLEGANDO A ESTE EN EL PERIODO 12
RTR349	RTR(2,4,2,1)	1	EL BUQUE 2 RECORRE DE UN PUNTO INTERMEDIO AL PUERTO 2, LLEGANDO Y DESCARGANDO ESTE EN EL PERIODO 1
RTR242	RTR(2,2,1,2)	1	EL BUQUE 2 RECORRE DEL PUERTO 2 A LA REFINERIA, LLEGANDO A ESTA EN EL PERIODO 2
RTR195	RTR(2,1,1,3)	1	EL BUQUE 2 PERMANECE EN LA REFINERIA DURANTE EL PERIODO 3 (CARGANDO)

COLUMNA	VARIABLE	VALOR	DESCRIPCION
RTR224	RTR(2,1,3,8)	1	EL BUQUE 2 RECORRE DE LA REFINERIA AL PUERTO 3, LLEGANDO Y DESCARGANDO EN ESTE EN EL PERIODO 8
RTR300	RTR(2,3,1,12)	1	EL BUQUE 2 RECORRE DEL PUERTO 3 A LA REFINERIA, LLEGANDO A ESTA EN EL PERIODO 12
DSC099	DSC(1,3,1,3)	1	EL BUQUE 1 DESCARGA EN EL PUERTO 3 EN EL PERIODO 1 UN LOTE TIPO 3 (150)
DSC137	DSC(1,3,11,1)	1	EL BUQUE 1 DESCARGA EN EL PUERTO 3 EN EL PERIODO 11 UN LOTE TIPO 1 (50)
DSC241	DSC(2,2,1,1)	1	EL BUQUE 2 DESCARGA EN EL PUERTO 2 EN EL PERIODO 1 UN LOTE TIPO 1 (50)
DSC318	DSC(2,3,8,2)	1	EL BUQUE 2 DESCARGA EN EL PUERTO 3 EN EL PERIODO 8 UN LOTE TIPO 2 (100)
CARGA24	CARGA(1,6,4)	1	EL BUQUE 1 CARGA EN EL PERIODO 6 UN LOTE TIPO 4 (0)
CARGA58	CARGA(2,3,2)	1	EL BUQUE 2 CARGA EN EL PERIODO 3 UN LOTE TIPO 2 (50)
CR_D01	CR_D(1,1)	200	CARGA DISPONIBLE EN EL BUQUE 1 AL INICIO DEL PERIODO 1
CR_D02	CR_D(1,2)	50	CARGA DISPONIBLE EN EL BUQUE 1 AL INICIO DEL PERIODO 2
CR_D03	CR_D(1,3)	50	CARGA DISPONIBLE EN EL BUQUE 1 AL INICIO DEL PERIODO 3
CR_D04	CR_D(1,4)	50	CARGA DISPONIBLE EN EL BUQUE 1 AL INICIO DEL PERIODO 4
CR_D05	CR_D(1,5)	50	CARGA DISPONIBLE EN EL BUQUE 1 AL INICIO DEL PERIODO 5
CR_D06	CR_D(1,6)	50	CARGA DISPONIBLE EN EL BUQUE 1 AL INICIO DEL PERIODO 6
CR_D07	CR_D(1,7)	50	CARGA DISPONIBLE EN EL BUQUE 1 AL INICIO DEL PERIODO 7
CR_D08	CR_D(1,8)	50	CARGA DISPONIBLE EN EL BUQUE 1 AL INICIO DEL PERIODO 8
CR_D09	CR_D(1,9)	50	CARGA DISPONIBLE EN EL BUQUE 1 AL INICIO DEL PERIODO 9
CR_D10	CR_D(1,10)	50	CARGA DISPONIBLE EN EL BUQUE 1 AL INICIO DEL PERIODO 10
CR_D11	CR_D(1,11)	50	CARGA DISPONIBLE EN EL BUQUE 1 AL INICIO DEL PERIODO 11
CR_D12	CR_D(1,12)	0	CARGA DISPONIBLE EN EL BUQUE 1 AL INICIO DEL PERIODO 12

COLUMNA	VARIABLE	VALOR	DESCRIPCION
CR_D13	CR_D(2,1)	100	CARGA DISPONIBLE EN EL BUQUE 2 AL INICIO DEL PERIODO 1
CR_D14	CR_D(2,2)	50	CARGA DISPONIBLE EN EL BUQUE 2 AL INICIO DEL PERIODO 2
CR_D15	CR_D(2,3)	50	CARGA DISPONIBLE EN EL BUQUE 2 AL INICIO DEL PERIODO 3
CR_D16	CR_D(2,4)	100	CARGA DISPONIBLE EN EL BUQUE 2 AL INICIO DEL PERIODO 4
CR_D17	CR_D(2,5)	100	CARGA DISPONIBLE EN EL BUQUE 2 AL INICIO DEL PERIODO 5
CR_D18	CR_D(2,6)	100	CARGA DISPONIBLE EN EL BUQUE 2 AL INICIO DEL PERIODO 6
CR_D19	CR_D(2,7)	100	CARGA DISPONIBLE EN EL BUQUE 2 AL INICIO DEL PERIODO 7
CR_D20	CR_D(2,8)	100	CARGA DISPONIBLE EN EL BUQUE 2 AL INICIO DEL PERIODO 8
CR_D21	CR_D(2,9)	0	CARGA DISPONIBLE EN EL BUQUE 2 AL INICIO DEL PERIODO 9
CR_D22	CR_D(2,10)	0	CARGA DISPONIBLE EN EL BUQUE 2 AL INICIO DEL PERIODO 10
CR_D23	CR_D(2,11)	0	CARGA DISPONIBLE EN EL BUQUE 2 AL INICIO DEL PERIODO 11
CR_D24	CR_D(2,12)	0	CARGA DISPONIBLE EN EL BUQUE 2 AL INICIO DEL PERIODO 12
OMD13	OMD(2,1)	140	EL PUERTO 2 RECIBE EN EL PERIODO 1 140 POR OTROS MEDIOS
OMD26	OMD(3,2)	50	EL PUERTO 3 RECIBE EN EL PERIODO 2 50 POR OTROS MEDIOS
EX_P13	EX_P(2,1)	150	EXISTENCIA DISPONIBLE EN EL PUERTO 2 AL INICIO DEL PERIODO 1
EX_P14	EX_P(2,2)	320	EXISTENCIA DISPONIBLE EN EL PUERTO 2 AL INICIO DEL PERIODO 2
EX_P15	EX_P(2,3)	300	EXISTENCIA DISPONIBLE EN EL PUERTO 2 AL INICIO DEL PERIODO 3
EX_P16	EX_P(2,4)	280	EXISTENCIA DISPONIBLE EN EL PUERTO 2 AL INICIO DEL PERIODO 4
EX_P17	EX_P(2,5)	260	EXISTENCIA DISPONIBLE EN EL PUERTO 2 AL INICIO DEL PERIODO 5
EX_P18	EX_P(2,6)	240	EXISTENCIA DISPONIBLE EN EL PUERTO 2 AL INICIO DEL PERIODO 6
EX_P19	EX_P(2,7)	220	EXISTENCIA DISPONIBLE EN EL PUERTO 2 AL INICIO DEL PERIODO 7
EX_P20	EX_P(2,8)	200	EXISTENCIA DISPONIBLE EN EL PUERTO 2 AL INICIO DEL PERIODO 8
EX_P21	EX_P(2,9)	180	EXISTENCIA DISPONIBLE EN EL PUERTO 2 AL INICIO DEL PERIODO 9
EX_P22	EX_P(2,10)	160	EXISTENCIA DISPONIBLE EN EL PUERTO 2 AL INICIO DEL PERIODO 10

COLUMNA	VARIABLE	VALOR	DESCRIPCION
EX_P23	EX_P(2,11)	140	EXISTENCIA DISPONIBLE EN EL PUERTO 2 AL INICIO DEL PERIODO 11
EX_P24	EX_P(2,12)	120	EXISTENCIA DISPONIBLE EN EL PUERTO 2 AL INICIO DEL PERIODO 12
EX_P25	EX_P(3,1)	300	EXISTENCIA DISPONIBLE EN EL PUERTO 3 AL INICIO DEL PERIODO 1
EX_P26	EX_P(3,2)	400	EXISTENCIA DISPONIBLE EN EL PUERTO 3 AL INICIO DEL PERIODO 2
EX_P27	EX_P(3,3)	400	EXISTENCIA DISPONIBLE EN EL PUERTO 3 AL INICIO DEL PERIODO 3
EX_P28	EX_P(3,4)	350	EXISTENCIA DISPONIBLE EN EL PUERTO 3 AL INICIO DEL PERIODO 4
EX_P29	EX_P(3,5)	300	EXISTENCIA DISPONIBLE EN EL PUERTO 3 AL INICIO DEL PERIODO 5
EX_P30	EX_P(3,6)	250	EXISTENCIA DISPONIBLE EN EL PUERTO 3 AL INICIO DEL PERIODO 6
EX_P31	EX_P(3,7)	200	EXISTENCIA DISPONIBLE EN EL PUERTO 3 AL INICIO DEL PERIODO 7
EX_P32	EX_P(3,8)	150	EXISTENCIA DISPONIBLE EN EL PUERTO 3 AL INICIO DEL PERIODO 8
EX_P33	EX_P(3,9)	200	EXISTENCIA DISPONIBLE EN EL PUERTO 3 AL INICIO DEL PERIODO 9
EX_P34	EX_P(3,10)	150	EXISTENCIA DISPONIBLE EN EL PUERTO 3 AL INICIO DEL PERIODO 10
EX_P35	EX_P(3,11)	100	EXISTENCIA DISPONIBLE EN EL PUERTO 3 AL INICIO DEL PERIODO 11
EX_P36	EX_P(3,12)	100	EXISTENCIA DISPONIBLE EN EL PUERTO 3 AL INICIO DEL PERIODO 12