DIVISION DE ESTUDIOS DE POSGRADO
Facultad de Ingeniería

PROGRAMACION LINEAL MIXTA-LOGICA

María Auxilio Osorio Lama

TESIS DOCTORAL

PRESENTADA A LA DIVISION DE ESTUDIOS DE POSGRADO DE LA
FACULTAD DE INGENIERIA
DE LA
UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

COMO REQUISITO PARA OBTENER EL GRADO DE

DOCTOR EN INGENIERIA

INVESTIGACION DE OPERACIONES
dirigida por el Dr. John Hooker

CIUDAD UNIVERSITARIA
1996

UNAM – Dirección General de Bibliotecas Tesis Digitales Restricciones de uso

DEDICATORIAS

A los asesores que hicieron posible este trabajo:

En la **Universidad Carnegie Mellon:**

Dr. John Hooker	Porque este trabajo ha sido el producto de un gran esfuerzo bajo la dirección de un gran hombre.
Dr. Ignacio Grossmann	Por haberme abierto las puertas a esta gran oportunidad y por sus oporturnos consejos durante el desarrollo de este trabajo.

En la **Universidad Nacional Autónoma de México:**

Dr. Sergio Fuentes Maya	Por el impulso y apoyo brindado en todo momento.

A las instituciones que me financiaron durante el doctorado:

Benemérita Universidad Autónoma de Puebla	Con profundo agradecimiento.
Comisión Nacional de Ciencia y Tecnología	

A mi familia:

Porque sin su ayuda hubiera sido imposible realizar mis sueños.

A mi hijo Victor José,	Porque las experiencias compartidas nos unan cada día más.
A mis papás Victor José (+) y Josefina (+),	Por el espíritu de lucha que sembraron en nosotros.
A mi querida tia Enriqueta:	Por la fuente de inspiración que ha sido en mi vida.
A mis hermanos: Victor José (+), Pepita, Arturo, Lupita y Juan,	Por el espíritu de solidaridad y ayuda mutua que siempre nos ha caracterizado, y nos ha ayudado a avanzar en la vida

A los maestros que me formaron:

A esos maestros especiales que a lo largo de mi vida lograron hacer de sus clases espacios de luz, y me hicieron soñar en ir más alto, porque sus vidas terminaron siendo fuente de inspiración de la mia.

A mis amigos y compañeros:

Porque este camino ha estado lleno de un sin fin de ayudas desinteresadas, de consejos oportunos, de muestras de aprecio y apoyo en momentos dificiles. A todos, muchas gracias.

PROGRAMACION LINEAL MIXTA-LOGICA

María Auxilio Osorio Lama

RESUMEN

La programación lineal mixta-lógica es un enfoque general para formular y resolver problemas de optimización que tienen ambos elementos, discretos y continuos y que permite, naturalmente, diferentes estrategias de ramificación, relajaciones y algoritmos de procesamiento lógico que o no pueden ajustarse o no son sugeridos desde el enfoque tradicional. En particular, cuenta las estrategias tradicionales entre sus opciones y debe ser vista como una extensión de la programación lineal mixta-entera en vez de como una alternativa a ella.

Representa los elementos discretos por medio de fórmulas lógicas y los elementos continuos por desigualdades lineales; por lo tanto tiene la opción de prescindir de las variables enteras a la hora de resolver el problema, y en vez de requerir que una solución factible satisfaga un conjunto fijo de desigualdades, provee varias conjuntos alternativos de desigualdades, donde el rol de los fórmulas lógicas consiste en gobernar cuáles alternativas son aceptables. A su vez, todo problema expresado en forma de programa lineal mixto-lógico es equivalente a un problema de programación disyuntiva y su conjunto factible es, por lo tanto una unión de muchos poliedros finitos en el espacio de las variables continuas.

Ya que es un enfoque general para resolver problemas mixtos continuos/discretos, comprende un conjunto de herramientas lógicas que pueden ser utilizadas de distinta manera para distintos problemas, y su efectividad, depende de que tan cuidadosamente se diseñan las relajaciones, los cortes, los esquemas de ramificación y las ecuaciones lógicas para resolver cada problema específico. El objetivo aquí fué el de proveer un amplio rango de opciones, y mostrar con ejemplos, como la mayoría de estas opciones son superiores a los métodos tradicionales para una amplia variedad de problemas. Las aplicaciones incluyen problemas de síntesis de procesos en ingeniería, de localización de almacenes, de calendarización de trabajos y el problema de la "fiesta progresiva", un problema de calendarización que originó un problema combinatorial tan complejo y grande que no se ha podido resolver hasta la fecha con programación lineal mixta-entera.

Entre las principales ventajas que la programación mixta-lógica ha mostrado en la mayoría de las aplicaciones ejemplificadas, están las siguientes: *separación de la regla de ramificación y la relajación del problema, modelos lineales más pequeños en los árboles de solución, procesamiento de las variables discretas más rápido y eficiente, soluciones factibles identificadas más rápidamente, relajaciones más fuertes, un enfoque alternativo para identificar cortes y un modelamiento más natural.*

MIXED LOGICAL / LINEAR PROGRAMMING

María Auxilio Osorio Lama

ABSTRACT

Mixed logical/linear programming (MLLP) is a general approach to formulate and solve optimization problems that have both discrete and continuous elements. It allows naturally for branching strategies, relaxations and logic processing algorithms that neither fit comfortably into nor are suggested by MILP. In particular it counts the traditional strategies among its options and should therefore be seen as an extension of MILP rather than an alternative to it.

It represents the discrete elements by logical formulas and only the continuous element by linear inequalities. It therefore has the option of dispensing with integer variables. Rather than require that a feasible solution satisfy a fixed set of inequalities, MLLP provides several alternative sets of inequalities. The role of the logical formulas is to govern which alternatives are acceptable. Also, every MLLP is equivalent to a disjunctive programming problem whose constraint is a disjunction of linear systems (*i.e.*, the solution must satisfy at least one of the systems). Its feasible set is therefore a union of finitely many polyhedra.

Because MLLP is not only a general approach to continuous/discrete problem solving but a framework within which several approaches can be used, its effectiveness depends on how
carefully one designs relaxations, cuts, and branching schemes to fit the problem at hand. The intent here is to provide a broader range of options and to show by example that they can be superior to the conventional ones. The examples include chemical engineering network synthesis problems, warehouse location problems, flow, shop scheduling problems, and the ``progressive party problem," which is a scheduling problem that has attracted a good deal of attention because its intractability with the traditional methodology.

The main advantages that the MLLP's broader framework has made evident in most of the examples used here, are the following: *separation of branching and relaxation, smaller linear programming problems in the search tree, faster and/or more effective processing of discrete variables, feasible solutions identified sooner, stronger relaxations, an alternative approach to identifying cuts, and a more natural modeling.*

INDICE

Capítulo 5: APLICACIONES

CAPITULO 1: INTRODUCCION

La programación lineal mixta-lógica es un enfoque general para formular y resolver problemas de optimización que tienen ambos elementos, discretos y continuos. Aunque el enfoque tradicional de programación lineal mixta-entera es efectivo en muchos casos, restringe innecesariamente el modelamiento y las opciones de solución disponible. En cambio, la programación lineal mixta-lógica permite, naturalmente, diferentes estrategias de ramificación, relajaciones y algoritmos de procesamiento lógico que o no pueden ajustarse o no son sugeridos desde el enfoque tradicional. En particular, cuenta las estrategias tradicionales entre sus opciones y debe ser vista como una extensión de la programación lineal mixta-entera en vez de como una alternativa a ella.

Los problemas mixtos con variables discretas y continuas se pueden concebir tradicionalmente como problemas continuos en los cuales algunas de las variables están restringidas a ser enteras. La programación lineal mixta-lógica tiene un punto de vista completamente diferente. En vez de ajustarse a los aspectos discretos de un problema mixto-entero, mismos que en ocasiones tienen que ser artificialmente fijados, representa los elementos discretos por medio de fórmulas lógicas y los elementos continuos por desigualdades lineales. Por lo tanto tiene la opción de prescindir de las variables enteras a la hora de resolver el problema, y en vez de requerir que una solución factible satisfaga un conjunto fijo de desigualdades, provee varias conjuntos alternativos de desigualdades, donde el rol de las fórmulas lógicas consite en gobernar cuáles alternativas son aceptables. A su vez, puede afirmarse que todo problema expresado en forma de programa lineal mixto-lógico es equivalente a un problema de programación disyuntiva cuyo conjunto de restricciones es un conjunto de disyunciones de sistemas lineales (*i.e.* la solución debe satisfacer al menos uno de los sistemas). Su conjunto factible es, por lo tanto una unión de muchos poliedros finitos en el espacio de las variables continuas.

El objetivo de esta disertación es el de explorar la programación lineal mixta-lógica como un enfoque general y práctico para resolver problemas con ambos elementos: continuos y discretos. A su vez se presentan los trabajos anteriores en el área, en un esfuerzo por exponer ordenadamente las ideas, unas antiguas y otras nuevas, asociadas con la programación lineal mixta-lógica, en un esquema coherente donde puedan exponerse y clarificarse las ventajas potenciales de este enfoque, a través de la presentación de los resultados computacionales obtenidos con dicho enfoque, en diferentes campos de aplicación.

Ya que la programación lineal mixta-lógica es un enfoque general para resolver problemas mixtos continuos/discretos, comprende un conjunto de herramientas lógicas que pueden ser utilizadas de distinta manera para distintos problemas, y que, como en la

programación lineal mixta-entera, su efectividad, en ocasiones, depende de que tan cuidadosamente se diseñan las relajaciones, los cortes, los esquemas de ramificación y las ecuaciones lógicas para resolver cada problema específico. El intento aquí es el de proveer un amplio rango de opciones, y mostrar con ejemplos, como la mayoría de estas opciones son superiores a los métodos tradicionales para los problemas ejemplificados en esta disertación.

Las aplicaciones incluyen problemas de síntesis de procesos en ingeniería, problemas de localización de almacenes, de calendarización de trabajos y el problema de la "fiesta progresiva" que es un problema de calendarización que surgió a raíz de la organización de una fiesta de yates el año pasado en Inglaterra, y que originó un problema combinatorial tan complejo y grande que no se ha podido resolver hasta la fecha con programación lineal mixta entera.

Aunque la programación disyuntiva puede considerarse como el antecedente teórico más importante de la programación lineal mixta-lógica, y si bien sus problemas comparten las mismas propiedades matemáticas básicas, ya que como se explicará ampliamente en el capítulo 3, se puede mostrar que cualquier modelo de programación lineal mixta-lógica puede expresarse como un modelo de programación disyuntiva, la programación lineal mixta-lógica incluye entre su metodología diferentes métodos de inferencia simbólica y de generación y expresión de planos de corte y de cortes lógicos que no se han utilizado hasta ahora en la programación disyuntiva, por lo que puede considerarse como una herramienta, diferente y en ocasiones mucho más poderosa.

También cabe hacer mención de que mientras que los otros enfoques contemporáneos que utilizan la lógica como herramienta, como es el caso de la programación "por restricciones" y algunos otros enfoques que se sitúan en el área de la inteligencia artificial utilizan modelos basados en los procedimientos y están implementados básicamente en PROLOG, la programación lineal mixta lógica se mantiene como un enfoque de programación matemática, basada en modelos declarativos. Además, en algunos casos donde la programación lineal mixta-entera ha mostrado ser insuficiente para resolver ciertos problemas con modelos muy complejos, la programación lineal mixta-lógica, gracias a la ayuda del procesamiento lógico ha encontrado soluciones que sólo pudieron ser encontradas por medio de los modelos basados en procedimientos que utiliza la inteligencia artificial, como es el caso de "la fiesta progresiva" que se explora en esta disertación.

A continuación se exponen las ventajas potenciales del amplio entorno de herramientas y posibilidades que componen la programación lineal mixta-lógica y que serán evidentes a lo largo de esta disertación, a través de los ejemplos escogidos para ilustrarlas.

- *Separación de la Regla de Ramificación y la Relajación del problema.* En la programación lineal mixta-entera, las variables enteras sirven para fijar la regla de ramificación, designando alguna de las variables enteras que tienen un valor fraccional en la solución relajada, en el método de ramificación y acotamiento y a su vez, sirven

para relajar el problema, removiendo la restricción de integralidad en las variables. Sin embargo, existen otros esquemas de relajación y de ramificación. La programación lineal mixta-lógica permite el uso de cualquier esquema de ramificación, en combinación con cualquiera de relajación, porque la relajación no necesita involucrar las variables discretas. La relajación es creada, generalmente, agregando cortes a la parte continua del modelo.

- *Modelos de programación lineal más pequeños.* Algunos problemas combinatorios están formulados con un gran número de variables enteras. La programación lineal mixta-entera fuerza a estas variables a estar incluídas en los problemas lineales resueltos en cada nodo del árbol de ramificación y acotamiento, mientras que el esquema de programación lineal mixta-lógica incluye sólo las variables continuas en los problemas lineales. Esta ventaja está dramáticamente ilustrada en el problema de la fiesta. En algunos casos, sin embargo, la ventaja de remover las variables enteras se pierde por la necesidad de agregar cortes para tener una buena relajación, como se verá en los ejemplos de localización de almacenes.

- *Procesamiento de las variables discretas más rápido o más eficiente.* Las variables lógicas, pueden, frecuentemente, ser fijadas en cada nodo, mediante la aplicación de procedimientos de inferencia en las restricciones lógicas. En muchos casos, las mismas variables deberían ser fijadas resolviendo la relajación convencional continua. Pero en estos casos, el procesamiento lógico es mucho más rápido, ya que puede ser realizado por medio de un procedimiento de resolución unitaria, lineal en tiempo. Esto es ilustrado en el problema de la fiesta. En otros casos, el procesamiento lógico es más poderoso que la programación lineal, porque fija variables con valores de verdadero o falso, mientras que recibe valores fraccionales en una relajación continua, lo cual puede ser ilustrado en los ejemplos de calendarización de trabajos y en el uso de variables semicontinuas en los problemas de síntesis de procesos.

- *Las soluciones factibles se identifican más pronto.* La relajación continua convencional de una formulación disyuntiva 0-1 puede tener soluciones fraccionales aún cuando la disyunción esté satisfecha. Esto significa que la programación lineal mixta entera puede seguir generando ramas en el árbol de ramificación y acotamiento aún cuando se haya encontrado una solución factible. La programación lineal mixta-lógica evita este defecto y puede, por lo tanto, producir un árbol de búsqueda más pequeño. Esto se ilustra en los problemas de calendarización de trabajos y en los casos en donde los problemas de síntesis de procesos tienen variables semicontinuas.

- *Relajaciones más fuertes.* La programación lineal mixta-lógica puede proporcionar relajaciones más fuertes de tres maneras:

    1. Algunas veces, los cortes válidos para las restricciones disyuntivas pueden ser más fuertes que la relajación continua de cualquier formulación obvia 0-1.

2. El procesamiento lógico puede generar restricciones lógicas (cortes lógicos) cuyas relajaciones lineales pueden ser agregadas al modelo lineal. A su vez, los mismos cortes lógicos pueden ser usados como planos en un algoritmo convencional de ramificación y acotamiento, pero el procesamiento lógico provee una forma sistemática de generarlos. Esto es ilustrado en los problemas de localización de almacenes y de la fiesta progresiva.

- *Un enfoque alternativo para identificar cortes.* El punto de vista lógico puede sugerir cortes lógicos que pueden derivarse fácilmente de un entendimiento intuitivo del problema pero que no son fácilmente revelados por el análisis poliédrico convencional. También puede sugerir cortes lógicos no-válidos, los cuales no cambian el valor óptimo pero son más fuertes porque excluyen algunas soluciones factibles que no tienen sentido. Los problemas de redes de proceso químico pueden ilustrar ambos puntos.

- *Modelamiento más natural.* Las variables enteras, particularmente las variables 0-1 intentan expresar lo que originalmente fué concebido como restricciones lógicas. En dichos casos, el enfoque de la programación lineal mixta-lógica permite una formulación más natural. En otros casos, los problemas pueden ser mejor expresados con variables discretas multivaluadas, como por ejemplo, en problemas en los cuales un valor de variable puede indicar la máquina a la cuál un trabajo es asignado o una posición de una operación en una secuencia. Las restricciones en aquellas variables que no pueden ser expresadas en forma de desigualdad pueden ser fácilmente formuladas con predicados que son más naturales para los algoritmos de procesamiento lógico. Un predicado particularmente útil es el predicado "todos-diferentes", el cuál requiere que un conjunto dado, todas las variables tengan valores diferentes. Las ventajas de las variables multivaluadas ha sido explotada en la literatura de la programación restringida, y los algoritmos desarrollados pueden ser usados en la fase de procesamineto lógico del entorno que constituye la programación lineal mixta-lógica.

El hábito de escribir modelos pensando en la conveniencia del paquete computacional utilizado para resolverlo es común en Investigación de Operaciones y puede llevarnos a olvidar que el principal objetivo del modelamiento en la ciencia es el explicativo. El proceso de modelamiento debería mejorar el entedimiento del problema y no supeditarse a la necesidad de escribir formulaciones compatibles con los paquetes de computación utilizados para resolverlos. De otra manera, si el modelo no es totalmente el adecuado para la solución de un problema, debería reformularse, ya sea por un procedimiento automático o por un experto humano. En algunos casos, puede ser difícil relacionar la solución con la formulación original. Con la programación lineal mixta-lógica se puede uno situar a la mitad del camino, ya que permite que el modelamiento discreto-continuo sea más natural, eliminando la necesidad de variables enteras e introduciendo restricciones lógicas y variables multivalentes, pero el enfoque de la solución está todavía relacionado con el problema original porque efectúa las ramificaciones en las variables proposicionales que aparecen en él. El modelo, generalmente, requiere una preparación

adicional antes de su solución, pero la preparación, generalmente involucra la adición de cortes y no la reformulación del mismo.

Las ventajas mencionadas de utilizar el amplio entorno de la programación lineal mixta-lógica, pueden ser atenuadas por al menos las tres consideraciones siguientes:

- *Las relajaciones pueden ser generadas explícitamente.* Un modelo de programación lineal mixta-entera ya contiene las relajaciones lineales, pero el entorno de programación lineal mixta-lógica nos obliga a hacer una selección consciente de la relajación, y algunas veces no es obvio como crear una relajación que sea, al menos, tan fuerte como la tradicional (sin reintroducir las variables enteras tradicionales).

- *Las relajaciones pueden agrandar el modelo.* Los cortes necesarios para construir relajaciones útiles pueden hacer el conjunto de restricciones lineales más grande que el modelo lineal mixto-entero, como en el caso de los problemas de localización de almacenes.

- *Se requiere más experiencia, y en ocasiones de la mano de un experto.* Utilizar el enfoque de programación lineal mixta-lógica, generalmente requiere de la mano de un experto, dado el rango más grande de opciones impuestas al usuario, haciendo, por lo tanto más reducido el círculo de personas que puede utilizarlo, a diferencia del enfoque tradicional de programación mixta-entera.

Uno puede responder a los dos primero puntos como sigue: Aunque una relajación no tradicional puede ser difícil de indentificar en algunos casos o puede agrandar mucho el problema en otros, existen otros casos en los que puede resolver problemas que en otros casos serían intratables. La clave consiste en ser capaz de identificar la relajación apropiada para un conjunto dado de restricciones. Se pueden distinguir los siguientes casos que son más rigurosamente caracterizados a lo largo de esta disertación.

1. *Las restricciones a ser relajadas pueden no tener una buena relajación lineal*, porque el envolvente convexo del conjunto factible ocupa completa o casi completamente el espacio de solución. En estos casos, no debería usarse ninguna relajación, como en el caso de los problemas que se presentan en la calendarización de trabajos, y en el uso de variables semicontinuas en la síntesis de procesos. La programación lineal mixta-lógica permite esto, mientras que la programación lineal mixta-entera nos fuerza a utilizar variables enteras, lo cual introduce en el modelo una sobrecarga totalmente inútil.

2. *Se puede encontrar un buena relajación, pero la programación lineal mixta-entera es débil o totalmente inútil.* Puede ser posible reemplazar los cortes de la programación lineal mixta-entera con un número razonable de cortes que produzcan una relajación más fuerte. Dos de los métodos utilizados, descritos en esta disertación, y que pueden ser útiles son el de reforzar los cortes obtenidos de la programación lineal mixta-entera y el de la generación de cortes de separación óptima.

3. *La relajación lineal mixta-entera es útil.* Puede ser fácil emular los efectos de la relajación lineal mixta-entera con pocos cortes. Si no, uno puede tener siempre la opción de agregar variables enteras para obtener la relajación clásica. Aún en este caso, se puede desear relajar sólo una parte del modelo de esta manera, y las variables enteras no necesitan ser utilizadas con el propósito de buscar variables de ramificación en el método de ramificación y acotamiento, así, se puede mantener la distinción entre la relajación del problema original y la regla de ramificación.

La objeción final, en el sentido de que la programación lineal mixta-lógica requiere la mano de un experto, puede ser parcialmente superada por la automatización de todas las opciones posibles e instalando opciones redundantes. Algunos de los paquetes computacionales de programación lineal mixta-entera más utilizados, suelen aplicar, por ejemplo un número de cortes y secciones de preprocesamiento que pueden o no ser útiles para un determinado problema.

Finalmente, sin embargo, una gran cantidad de problemas combinatorios, suelen requerir de la mano de un experto para su solución. El hecho es el de cuánta intervención es apropiada. Sin embrago, no parece razonable el restringirse a uno mismo a rutinas automáticas en paquetes computacionales de propósito general cuando algunas simples adiciones nos pueden ayudar a encontrar soluciones que de otra manera pueden estar inalcanzables. En el otro extremo, es impráctico invertir en cada problema nuevo, los años de investigación y esfuerzo que se le han dedicado el problema del viajero o al de la calendarización de trabajos. Por lo tanto, la programación lineal mixta-lógica está diseñada para presentar un compromiso entre estos dos extremos.

Después de esta breve discusión sobre programación lineal mixta-lógica, los siguientes capítulos se estructuran de la siguiente manera. En el capítulo 2 se presenta una descripción detallada del antecedente teórico más importante de la programación lineal mixta-lógica, la programación disyuntiva, campo que permitió el desarrollo posterior de la programación lineal mixta-lógica; en el capítulo 3 se presentan los antecedentes más cercanos de la programación lineal mixta-lógica, la formulación general de sus modelos, se resume el algoritmo básico utilizado y se presenta la descripción matemática detallada de las relajaciones y los planos de corte utilizados. En el capítulo 4 se presentan los diferentes procedimientos utilizados para el procesamiento lógico propuesto, incluída la generación de los cortes lógicos. En el capítulo 5 se presentan los modelos utilizados en cada una de las áreas de aplicación estudiadas y; finalmente, en el capítulo 6 se presentan los resultados computacionales obtenidos en cada una de las áreas estudiadas, así como las conclusiones que pudieron desprenderse de dicho estudio computacional. En el apéndice se incluye una presentación detallada de los ejemplos utilizados en este trabajo, presentando tablas con los datos de cada problema, y con los resultados de la función objetivo y de las principales variables continuas en las corridas realizadas; principalmente con propósitos didácticos y/o para proporcionar la información necesaria para reproducir estos resultados con otra metodología o con alguna variante de la aquí utilizada.

CAPITULO 2: ANTECEDENTE TEORICO: La Programación Disyuntiva

Hoy en día, el campo de la Programación Disyuntiva abarca la utilización de programas lineales (o no lineales) con restricciones total o parcialmente expresadas en forma de disyunciones que pudieran ser expresados en forma entera (o mixta-entera), pero que pueden ser expresadas en forma más natural utilizando esta formulación.

En el aspecto teórico, la Programación Disyuntiva, provee las caracterizaciones estructurales que han ofrecido nuevas líneas de investigación y desarrollo; y en el aspecto práctico, produce una variedad de planos de corte con propiedades deseables, y ofrece varias formas de combinar planos de corte con el método de ramificación y acotamiento, así como la posibilidad de trabajar con los modelos planteados en forma disyuntiva, lo cual nos puede llevar a modelos lineales considerablemente más pequeños en cada nodo del árbol de búsqueda.

Este capítulo, basado totalmente en los resultados obtenidos y publicados por Balas y Jeroslow en los últimos 15 años, se organiza de la siguiente manera. En la sección 1 se presentan los diferentes antecedentes que dieron origen a la programación disyuntiva, en la sección 2, se presentan los conceptos básicos y la terminología utilizada para formular los programas disyuntivos. La sección 3 contiene la caracterización básica de la familia de desigualdades válidas para un programa disyuntivo, y discute algunas de las implicaciones de este principio general para derivar planos de corte. En la sección 4 se presenta la extensión del teorema de dualidad de la programación lineal a la programación disyuntiva. En la sección 5 se presentan las propiedades básicas de los polares inversos y su utilización en la caracterización del envolvente convexo de un conjunto disyuntivo, y se muestra como las facetas del envolvente convexo del conjunto factible de un programa disyuntivo en n variables, definido por una disyunción con q términos, puede ser obtenida para un programa lineal con $q(n+1)$ restricciones.

Finalmente, en la sección 6 se aborda la importante cuestión, acerca de cuándo se puede generar secuencialmente el envolvente convexo de un conjunto disyuntivo, imponiendo, una por una, las disyunciones de la forma conjuntiva normal, y produciendo en cada paso, el envolvente convexo del conjunto disyuntivo con una disyunción elemental. Aunque la respuesta a esta pregunta es negativa para el caso de un programa disyuntivo general o un programa (mixto-) entero general, es afirmativa para una clase especial de programas disyuntivos llamados faciales, que comprenden los programas (puros o mixtos) cero-uno, el problema de complementariedad lineal, el programa cuadrático no convexo (restringido linealmente), etc.

1. Antecedentes

La línea de investigación que dió como resultado el enfoque de la programación disyuntiva se origina en el trabajo sobre intersección o cortes convexos de Young[40], Balas[2], Glover[23], Owen[36] y otros ([13], [24], [42]). Este enfoque geométricamente motivado puede ser descrito en términos de las intersecciones de los límites del cono que se origina en el óptimo x del programa lineal, con los límites de algún conjunto convexo S, cuyo interior contiene x, pero ningún punto entero factible, y usando el hiperplano definido por la intersección de los puntos como un plano de corte.

En los años 70's, la investigación sobre intersecciones o cortes convexos se dió en dos direcciones, principalmente. Una, tipificada en ([23],[18],[4]) y llevada a cabo por Balas, Glover y Jeroslow fué dirigida a la obtención de cortes más fuertes, incluyendo dentro de S, el espacio de restricciones, algunos puntos factibles enteros, implícita o explícitamente enumerados. La otra también llevada a cabo por Balas [3] introdujo la utilización de la polaridad y los polares exteriores (*i.e.*, los polares de los conjuntos factibles, escalados de tal manera que contengan todos los puntos factibles 0-1 en sus fronteras), y conceptos relacionados con el análisis convexo, como las extensiones máximas convexas, funciones de soporte ([5],[15],[19]). Además, de los planos de corte, esta segunda dirección también produjo ([5], [6]) el método de la "restricción activada". Esta investigación produjo algunos resultados relevantes y algunos planos de corte razonablemente fuertes, pero estos procedimientos resultaron demasiado costosos, computacionalmente hablando, para ser prácticamente útiles.

Posteriormente, Glover [25] descubrió que los cortes de intersección derivados de un poliedro convexo S pueden ser reforzados rotando las facetas de S en ciertas formas, un procedimiento que él llamó la "anexión poliédrica". Este fué un paso importante hacia el desarrollo de las técnicas que ayudaron a caracterizar esta corriente disyuntiva. Las mismas conclusiones fueron alcanzadas independientemente, en un contexto diferente por Balas [7]. El nuevo contexto introducido por él, fué el reconocimiento de que los cortes de intersección pueden ser vistos como derivados de una disyunción, lo cual originó la consideración de los programas disyuntivos en su generalidad [8], [9] y la caracterización de todas las desigualdades válidas para un programa (mixto-)entero. Por la misma razóm, ofreció la nueva posibilidad de generar cortes, especialmente derivados de una estructura dada. Además, ofreció una perspectiva teórica unificada sobre planos de corte y enumeración, así comouna forma práctica de combinar diferentes enfoques en la solución de un programa mixto-entero.

2. Definición Formal

La programación disyuntiva describe el conjunto de programas de optimización con restricciones disyuntivas, programas lineales enteros y mixtos-enteros y otros problemas de programación no-convexos, como el programa general cuadrático, el problema lineal de complementariedad, los programas separables no-lineales, etc., que pueden expresarse

como programas disyuntivos, esto es, expresando total o parcialmente sus restricciones en forma conjuntiva normal. En este contexto las restricciones de tipo disyuntivo, incluyen conjuntos de desigualdades lineales que pueden llamarse "condiciones lógicas". Estas condiciones lógicas son declaraciones con desigualdades lineales, que involucran las operaciones lógicas "Y" ($\wedge$, conjunción -que algunas veces se denota como yuxtaposición), "O" ($\vee$, disyunción), y "complemento de " ($\neg$, negación). La operación "si ... entonces" ($\Rightarrow$, implicación) puede expresarse como una disyunción. La negación de un conjunto conjuntivo de desigualdades es una disyunción cuyos términos son las mismas desigualdades. La operación de conjunción aplicada a desigualdades lineales origina conjuntos poliédricos (convexos). Las disyunciones son así, los elementos cruciales de una condición lógica (los que hacen el conjunto de restricciones no-convexo), y por eso se le llama a este tipo de problema, *programa disyuntivo.*

Un programa disyuntivo es entonces, un problema de la forma

$$\min \{cx \mid Ax \geq a_0,\ x \geq 0, x \in L\},$$

donde A es una matriz mxn, a_0 es un vector-m, y L es un conjunto de condiciones lógicas. Ya que este problema puede ser expresado de muchas formas, existen muchas formas diferentes en las cuales un problema disyuntivo puede ser establecido. Dos de ellas son fundamentales.

El conjunto de restricciones de un programa disyuntivo (y el problema en sí mismo) se dice que está en forma normal disyuntiva si está definido por una disyunción cuyos términos no contienen otras disyunciones; y se dice que está en forma normal conjuntiva, si está definido por una conjunción cuyos términos no contienen otras conjunciones. La forma normal conjuntiva, es entonces

$$\bigvee_{h \in Q} (A^h x \geq a^h{}_0,\ x \geq 0) \qquad (2.1)$$

mientras que la forma normal disyuntiva es

$$Ax \geq a_0,\ x \geq 0,$$
$$\bigvee_{i \in Q_j} (d^i x \geq d_{i0}), \qquad j \in S \qquad (2.2)$$

o, alternativamente,

$$\begin{pmatrix} Ax \geq a_0 \\ x \geq 0 \end{pmatrix} \wedge \left[\bigvee_{i \in Q_1} \left(d^i x \geq d_{i0} \right) \right] \wedge \ldots \wedge \left[\bigvee_{i \in Q_s} \left(d^i x \geq d_{i0} \right) \right] \qquad (2.2')$$

Aquí, cada d^i es un vector-n y cada d_{i0} es un escalar, mientras que los conjuntos Q y Q_j, $j \in S$, pueden o no ser finitos. La conección entre las dos formas es que cada término de la disyunción (2.1) tiene, además de $m + n$ desigualdades del sistema $Ax \geq a_0$, $x \geq 0$, precisamente una desigualdad $d^i x \geq d_{i0}$, $i \in Q_j$ de cada disyunción $j \in S$ de (2.2), y que todos

los sistemas $A^h x \geq a^h{}_0$, $x \geq 0$ distintos con esta propiedad están presentes entre los términos de la ecuación (2.1); así que, si Q (y entonces cada Q_j, $j \in S$) es finita, entonces lo es $|Q|=|\Pi_{j\in S} Q_j|$, donde Π representa el producto cartesiano. Ya que las operaciones $\wedge$ y $\vee$ son distributivas con respecto a la otra [*i.e.*, si A, B, C son desigualdades, $A\wedge(B\vee C) = AB\vee AC$, y $A\vee(BC) = (A\vee B)\wedge(A\vee C)$], cualquier condición lógica que involucre estas operaciones puede ser expresada en cualquiera de las dos formas fundamentales, y cada una de ellas puede ser obtenida de la otra.

Se puede ilustrar el significado de estas dos formas en el caso en que el programa disyuntivo es un programa cero-uno en n variables. Entonces, la forma normal disyuntiva (2.1) es

$$\bigvee_{h\in Q} (Ax \geq a_0,\ x \geq 0, x = x^h)$$

donde $x_1, \ldots, x_{|Q|}$ es el conjunto de todos los puntos 0-1, y $|Q| = 2^n$; mientras que la forma normal conjuntiva es

$$Ax \geq 0, \qquad x \geq 0, \qquad (x_j = 0) \vee (x_j = 1), \qquad j = 1, \ldots, n.$$

Una vez que las desigualdades que ocurren en las conjunciones y/o disyunciones de un programa disyuntivo son dadas, las formas disyuntiva y conjuntiva normal son únicas. Es un hecho de crucial importancia práctica, sin embargo, que las desigualdades que expresan las condiciones de un problema dado pueden ser escogidas en más de una forma. Por ejemplo, el conjunto de restricciones

$$\begin{aligned} &3\,x_1 + x_2 - 2\,x_3 + x_4 \leq 1,\\ &x_1 + x_2 + x_3 + x_4 \leq 1,\\ &x_j = 0 \text{ ó } 1, \quad j = 1, \ldots, 4, \end{aligned}$$

cuando son puestas en forma normal disyuntiva, se convierten en una disyunción con 2^4=16 términos; pero el mismo conjunto de restricciones puede también ser expresado como

$$3\,x_1 + x_2 - 2\,x_3 + x_4 \leq 1,$$

$$\bigvee_{i=1} (x_i = 1\ , x_j = 0,\ j \neq i) \vee (x_j = 0,\ \forall j),$$

lo cual nos lleva a una disyunción con sólo 5 términos.

3. El Principio Básico de la Programación Disyuntiva

Una restricción B se dice que es una *consecuencia de*, o *implicada por*, una restricción A, si cada x que satisface A, también satisface B. Estamos interesados en la familia de desigualdades implicadas por el conjunto de restricciones de un programa general disyuntivo. Todos los planos de corte válidos para el programa disyuntivo pertenecen, por

supuesto a esta familia. Además, el conjunto de puntos que satisface todas las desigualdades en la familia es precisamente el envolvente convexo del conjunto de soluciones factibles del programa disyuntivo lineal. Una caracterización de esta familia viene dada por el siguiente teorema, el cual es una generalización fácil pero importante de un resultado clásico.

Sea $x \in \mathbf{R}^n$, $\alpha \in \mathbf{R}^n$, $\alpha_0 \in \mathbf{R}$, $A^h \in \mathbf{R}^{m_h \times n}$, $a_0{}^h \in \mathbf{R}^{m_h}$, $h \in Q$ (no necesariamente finito) y sea $a_j{}^h$ la j-ésima columna de A^h, $h \in Q, j \in N = \{1, \ldots, n\}$.

Teorema 3.1 *La desigualdad* $\alpha x \geq \alpha_0$ *es una consecuencia de la restricción*

$$\bigvee_{h \in Q} (A^h x \geq a^h{}_0,\ x \geq 0) \tag{3.1}$$

si y sólo si existe un conjunto de $\theta^h \in \mathbf{R}^{m_h}$, $\theta^h \geq 0$, $h \in Q^*$, *que satisface*

$$\alpha \geq \theta^h A^h \text{ y } \alpha_0 \leq \theta^h a_0{}^h, \ \forall h \in Q^*, \tag{3.2}$$

donde A^* *es el conjunto de aquellas* $h \in Q$ *tal que el sistema* $A^h x \geq a_0$, $x \geq 0$ *es consistente.*

Prueba. Es una consecuencia de (3.1) si y sólo si es una consecuencia de cada término de (3.1). Este resultado es una consecuencia directa del Lemma de Farkas, donde se establecen las condiciones para encontrar las consecuencias homogéneas de un sistema homogéneo.

Comentario 3.1 Si la desigualdad i-ésima $h \in Q^*$ de un sistema (3.1) es reemplazada por una ecuación, el i-ésimo componente de θ^h se va a hacer irrestringido. Si se permite que la variable x_i en (3.1) sea irrestringida, la desiguadad j-ésima de cada sistema $\alpha \geq \theta^h A^h$, $h \in Q^*$, va a ser reemplazada por la ecuación correspondiente en la parte "si" de la cláusula.

Con estos cambios, el Teorema 3.1 permanece verdadero.

Una forma alternativa de expresar la ecuación (3.2) es:

$$\alpha \geq \sup_{h \in Q^*} \theta^h a_j{}^h, \qquad j \in N,$$

$$\alpha \geq \inf_{h \in Q^*} \theta^h a_0{}^h. \tag{3.3}$$

Ya que $Q \subseteq Q^*$, la parte "si" del Teorema permanece válida si Q^* se reemplaza por Q.

Ya que(3.3) define *todas* las desigualdades válidas para (3.1), cada plano de corte válido para un problema disyuntivo puede ser obtenido de la ecuación (3.3), escogiendo los

multiplicadores θ^h, convenientes. Si pensamos que el sistema (3.1) está siendo expresado en términos de las variables no básicas de una solución básica óptima del programa lineal asociado con el programa disyuntivo, entonces, una desigualdad válida $\alpha x \geq \alpha_o$ corta la solución óptima del programa lineal (correspondiente a $x_j = 0$, $j \in N$) si y sólo si $\alpha_o > 0$; entonces α_o tendrá que ser fijada en un valor positivo. Las desigualdades con $\alpha_o \leq 0$ todavía pueden cortar partes del conjunto factible del programa lineal, pero no la solución óptima $x=0$.

El caso especial cuando cada sistema $A^h x \geq a^h{}_o$, $h \in Q$, consiste sólo de la desigualdad $a^h x \geq a_{ho}$ (donde a^h es un vector, y a_{ho} es un escalar positivo) merece atención especial. En este caso, escogiendo los multiplicadores $\theta^h = 1/a_{ho}$, $h \in Q$, se obtiene la desigualdad

$$\sum_{j \in J} (\max a^h{}_j / a_{ho})\, x_j \geq 1 \qquad (3.4)$$

la cual, para Q finita, es la ecuación del corte de Owen[36]. También puede ser mirada como una versión ligeramente mejorada del corte de interseción del conjunto convexo

$$S = \{x \mid a^h x \leq a_{ho},\ h \in Q\},$$

el cual tiene los mismos coeficientes de (3.4), excepto para aquellos (si alguno) $j \in J$ tal que $a^h{}_j < 0$, $\forall h \in Q$. Más aún, el corte de intersección de S tiene coeficientes cero cuando los coeficientes correspondientes de (3.4) son negativos.

Siempre que todos los coeficientes de (3.4) son positivos (en términos de cortes de intersección, esto corresponde al caso en que S está acotado), (3.4) es la desigualdad más fuerte implicada por la disyunción $\vee_{h \in Q}(a^h x \leq a_{ho})$; en la presencia de coeficientes negativos; sin embargo, (3.4) puede, algunas veces, ser reforzado todavía.

Debido a la generalidad de la familia de desigualdades definida por (3.3), no sólo los cortes anteriores descritos en la literatura, pueden ser fácilmente recuperados por una selección apropiada de los multiplicadores θ^h (ver [8]), sino que, también, poniéndolos en la forma (3.3) se puede, por la misma razón, ver la forma en que dichos cortes pueden ser reforzados, con una selección apropiada de los multiplicadores.

Otra característica importante del principio expresado en el Teorema 3.1 para generar planos de corte, es el hecho de que al formular un problema (mixto-)entero como un problema disyuntivo, uno puede sacar provecho de cualquier estructura particular que tenga el problema.

4. Dualidad en los Problemas Lineales Disyuntivos

El teorema de dualidad de la programación lineal puede ser generalizado para los problemas disyuntivos. Considere el problema disyuntivo:

$$z_0 = \min cx,$$
$$\text{s.a} \quad \bigvee_{h\in Q} (A^h x \geq b^h,\ x \geq 0) \qquad \text{(P)}$$

donde A^h es una matriz y b^h un vector, $\forall h \in Q$. De acuerdo a Balas [11] , se define el dual del problema (P), como:

$$w_0 = \max w,$$
$$\text{s.a} \bigwedge_{h\in Q} \left\{ \begin{array}{c} w - u^h b^h \leq 0 \\ u^h A^h \leq c \\ u^h \geq 0 \end{array} \right\} \qquad \text{(D)}$$

El conjunto de restricciones del problema (D), requiere que cada u^h, $h \in Q$ satisfaga el sistema correspondiente a cada miembro de la disyunción y que w satisfaga cada uno de ellos.

Si:

$$X_h = \{x \mid A^h x \geq b^h,\ x \geq 0\}, \qquad X^*_h = \{x \mid A^h x \geq 0,\ x \geq 0\};$$
$$U_h = \{u^h \mid u^h A^h \leq c,\ u^h \geq 0\}, \qquad U^*_h = \{u^h \mid u^h A^h \leq 0,\ u^h \geq 0\}.$$

Además, si dejamos que:

$$Q^* = \{h \in Q \mid X_h \neq \mathbf{0}\}, \quad Q^{**} = \{h \in Q \mid U_h \neq \mathbf{0}\}.$$

Se puede asumir lo siguiente

Condición de Regularidad:

$$(Q^* \neq \mathbf{0},\ Q \setminus Q^{**} \neq \mathbf{0}) \Rightarrow Q^* \setminus Q^{**} \neq \mathbf{0}\ ;$$

i.e. si (P) es factible y (D) es infactible, entonces, existe $h \in Q$ tal que $X_h \neq \mathbf{0}$, $U_h \neq \mathbf{0}$

Teorema 4.1 Asuma que (P) y (D) satisfacen la condición de regularidad. Entonces exactamente una de las dos siguientes situaciones se satisface.

(1) *Ambos problemas son factibles; cada uno tiene una solución óptima y* $z_0 = w_0$.
(2) *Uno de los problemas es infactible; el otro o es infactible o no tiene un óptimo finito.*

Prueba. (1) Asuma que ambas, (P) y (D) son factibles. Si (P) tiene un óptimo no finito, entonces, existe $h \in Q$ tal que $X^*_h \neq \mathbf{0}$ y $x^* \in X^*_h$ tal que $cx^* < 0$. Pero, entonces $U_h \neq \mathbf{0}$, *i.e.*, (D) es infactible, lo cual es una contradicción.

Ahora supongamos que(P) tiene una solución óptima, digamos x. Entonces la desigualdad $cx \geq z_0$ es una consecuencia del conjunto de restricciones de (P), *i.e.*, $x \in X_h$ implica $cx \geq z_0$,

$\forall h \in Q$. Pero entonces, para toda $h \in Q^*$, existe $u^h \in U_h$ tal que $u^h b^h \geq z_0$. Más aún, ya que (D) es factible, para cada $h \in Q \setminus Q^*$ existe $u^{*h} \in U^*_h$, tal que $u^{*h} b^h > 0$, $\forall h \in Q \setminus Q^*$. Pero entonces, definiendo

$$u^h(\lambda) = u^{*h} + \lambda\, u^{*h} \quad , h \in Q \setminus Q^*,$$

para λ suficientemente grande, $u^h(\lambda) \in U^*_h$, $u^h(\lambda) b^h \geq z_0$, $\forall h \in Q \setminus Q^*$.

Entonces, para toda , $h \in Q$, existen vectores u^h que satisfacen las restricciones de (D) para $w = z_0$. Para mostrar que este es el valor máximo de w, debemos notar que ya que x^* es el valor óptimo para (P), existe $h \in Q$ tal que

$$cx^* = \min \{cx \mid x \in X_h\}.$$

Pero entonces, por la dualidad de la programación lineal,

$$\begin{aligned} cx^* &= \max \{u^h b^h \mid u^h \in U_h\}, \\ &= \max \{w \mid w - u^h b^h \leq 0 \mid u^h \in U_h\} \\ &\geq \max \{w \mid \underset{h \in Q}{\wedge} (w - u^h b^h \leq 0 \mid u^h \in U_h)\} \end{aligned}$$

i.e. $w \leq z_0$, entonces, el valor máximo de x es $w_0 = z_0$.

(2) Asuma que al menos uno de los dos modelos (P) o (D) es infactible. Si (P) es infactible, $X_h = \mathbf{0}$, $\forall h \in Q$; entonces, para toda $h \in Q$, existe $u^{*h} \in U^*_h$ tal que $u^{*h} b^h > 0$.

Si (D) es también infactible, estamos satisfaciendo el terorema. De otra manera, para cada $h \in Q$, existe $\hat{u}^h \in U_h$. Entonces, definiendo

$$u^h(\lambda) = \hat{u}^h + \lambda\, u^{*h}, \quad h \in Q,$$

$u^h(\lambda) \in U_h$, $h \in Q$, para toda $\lambda > 0$, y ya que $u^{*h}(\lambda) b^h \geq 0 \geq w$ puede ser hecha arbitrariamente grande incrementando λ; *i.e.*, (D) no tiene un óptimo finito.

A la inversa, si (D) es infactible, entonces o (P) es infactible y el teorema se satisface, o además, de la condición de regularidad, $Q^* \setminus Q^{**} \neq \mathbf{0}$ y para $h \in Q^* \setminus Q^{**}$ existe $x^o \in X_h$ y $x^* \in X^*_h$ tal que $cx^* < 0$. Pero entonces

$$x(\mu) = x^o + \mu x^*$$

es una solución factible para (P) para cualquiera $\mu > 0$, y ya que $cx^* < 0$, z puede ser hecho arbitrariamente pequeño incrementando μ; *i.e.* (P) no tiene un óptimo finito.

Este teorema asegura que la situación 1 ó 2 se satisface para (P) y (D) si la condición de regularidad se satisface. El siguiente corolario muestra que la condición no sólo es suficiente sino que también es necesaria.

Corolario 4.1 Si la condición de regularidad no se satisface, entonces, si (P) es factible y (D) es infactible, (P) tiene un mínimo finito (*i.e.* existe "diferencia dual").

Prueba. Dejemos que (P) sea factible, (D) infactible, y $Q^* \setminus Q^{**} \neq \mathbf{0}$, *i.e.*, para cada $h \in Q^*$, deje que $U_h \neq \mathbf{0}$. Entonces, para cada $h \in Q^*$, $\min\{cx \mid x \in X_h\}$es finito, entonces (P) tiene un mínimo finito.

Comentario. El teorema permanece verdadero si algunas de las variables de (P) [de (D)] no están restringidas, y las restricciones correspondientes de (D) [de (P)] son igualdades.

Puede esperarse que la condición de regularidad se aplique en todas las situaciones, menos en algunas especialmente peculiares. En la dualidad de la programación lineal, el caso cuando ambos problemas, el primal y el dual son infactibles, sólo ocurre para problemas cuyos coeficientes de la matriz A tienen la propiedad especial de que existe $x \neq 0$, $u \neq 0$, satisfaciendo el sistema homogéneo

$$\begin{aligned} Ax &\geq 0, & x &\geq 0; \\ uA &\leq 0 & u &\geq 0. \end{aligned}$$

En este contexto, esta condición de regularidad requiere que, si el problema primal es factible y el dual es infactible, entonces, al menos una de las matrices A_h cuyos conjuntos asociados U_h son infactibles, no tenga la propiedad especial mencionada.

5. El Envolvente Convexo de un Conjunto Disyuntivo

Habiendo descrito la familia de todas las desigualdades válidas, el siguiente interés se centrará en la identificación de las desigualdades más fuertes entre las últimas, *i.e.*, las caras del envolvente convexo de los puntos factibles para un programa disyuntivo.

Si se denota el conjunto factible de un programa disyuntivo por

$$F = \left\{ x \in R^n \,\middle|\, \bigvee_{h \in Q} \begin{pmatrix} A^h x \geq a_0^h \\ x \geq 0 \end{pmatrix} \right\},$$

entonces, para un escalar α_o dado, la familia de desigualdades $\alpha x \geq \alpha_o$ satisfecha por todas las $x \in F$, *i.e.*, la familia de desigualdades válidas para el programa disyuntivo, es obviamente isomórfico a la familia de vectores $\alpha \in F^{\#}_{(\alpha o)}$, donde

$$F^{\#}_{(\alpha o)} = \{y \in \mathbf{R}^n \mid yx \geq \alpha_o, \forall x \in F\},$$

en el sentido de que $\alpha x \geq \alpha_o$ es una desigualdad válida si y sólo si $\alpha \in F^{\#}_{(\alpha o)}$.

En vista de su relación con un conjunto ordinario polar, se puede llamar a $F^{\#}_{(\alpha_o)}$ el *inverso polar* de F. De hecho, el conjunto ordinario polar de F es

$$F^o = \{y \in \mathbf{R}^n \mid yx \leq 1, \forall x \in F\},$$

y si se denota $F^o_{(\alpha o)}$ como el polar de F a escala α_o, (*i.e.*, el conjunto obtenido reemplazando 1 con α_o en F), entonces $F^{\#}_{(\alpha o)} = - F^o_{(-\alpha o)}$.

El tamaño (lo opuesto del signo) de α_o no es de interés aquí. Por lo tanto se distinguirá sólo entre los 3 casos $\alpha_o > 0$ (o $\alpha_o = 1$), $\alpha_o = 0$ y $\alpha_o < 0$ ($\alpha_o = -1$). (cuando el signo de α_o no hace ninguna diferencia o es claro, dado el contexto, simplemente se escribirá $F^{\#}$). Para $\alpha_o \leq 0$, como se mencionó antes, $F^{\#}_{(\alpha o)}$ es (el negativo de) un cojunto polar ordinario, cuyas propiedades están descritas en la literatura. El caso más interesante para los modelos disyuntivos, sin embargo, es cuando $\alpha_o > 1$, ya que es el único caso cuando la desigualdad $\alpha x \geq \alpha_o$ corta el punto $x = 0$, y esta es la razón por la que se necesita el concepto de inversos polares.

Para un conjunto arbitrario $S \subseteq \mathbf{R}^n$ se denotará como **cl** S, el espacio cerrado de S, como **conv** X, el envolvente convexo, como **cono** de S, el envolvente cónico, como **int** S, el conjunto interior de S y como **Dim** S, la dimensión de S. Para un conjunto poliédrico $S \subseteq \mathbf{R}^n$, se denotará por *vert* S y **dir** S el conjunto de vértices (puntos extremos) y el conjunto de vectores de direcciones extremas de S, respectivamente.

Balas mostró [10] que mientras que algunas de las propiedades básicas de los conjuntos polares se aplican a los polares inversos, como las siguientes, que se pueden derivar de la definición:

(a) $(\lambda S)^{\#} = (1/\lambda) S^{\#}$;
(b) $S \subseteq T \Rightarrow S\# \supseteq T\#$
(c) $(S \cup T)\# = S\# \cap T\#$

otras sólo pueden ser recuperadas en una forma modificada, y serán presentadas dentro de los siguientes teoremas, enunciados por Balas en 1979[10].

Teorema 5.1 (*i*) *Si* $\alpha_o \leq 0$, *entonces* , *y* $S^{\#} \neq \mathbf{0}$

$0 \in$ **int cl conv** S $\Leftrightarrow$ $S^{\#}$ *está acotado*

(*ii*) *Si* $\alpha_o > 0$, *entonces*

$0 \in$ **cl conv** S $\Leftrightarrow$ $S^{\#} = \mathbf{0} \Leftrightarrow$ $S^{\#}$ *está acotado*

Prueba. (*i*) Se sigue de la propiedad correspondiente del polar ordinario S^o de S y del hecho de que $S^{\#}_{(\alpha o)} = - S^o_{(\alpha o)}$.

(ii) Para, $\alpha_o > 0$, si $S^{\#} \neq \mathbf{0}$ existe $y \in \mathbf{R}^n$ tal que $xy \geq \alpha_o$, $\forall x \in$ **cl conv** S. Pero $0.y < \alpha_o$, por lo tanto $0 \notin$ **cl conv** S. Así, si $0 \in$ **cl conv** S, entonces $S^{\#} = \mathbf{0}$; y por lo tanto $S^{\#}$ está acotado. Contrariamente, si $0 \notin$ **cl conv** S, existe un hiperplano $ax = \alpha_o$ que separa 0 de

cl conv S , *i.e.* de tal manera que, ($\alpha_o > 0$ y) $ax \geq \alpha_o$, $\forall x \in$ **cl conv** S; lo cual implica $a \in S^{\#}$, *i.e.* $S^{\#} \neq \mathbf{0}$. También implica $\lambda a \in S^{\#}$, $\forall \lambda > 1$, *i.e.*, $S^{\#}$ no está acotado

Desde este punto de vista, se puede restringir la atención a los conjuntos cuya envoltura convexa es poliédrica. Para el conjunto disyuntivo F, esta condición se satisface si Q es finito. La mayoría de los resultados se pueden aplicar a los casos generales, pero las pruebas se pueden simplificar con las suposiciones anteriores.

Teorema 5.2 *Si* **cl conv** S *es poliédrico, también lo es* $S^{\#}$.

Prueba. Deje que $u_1, \ldots, u_p$ y $v_1, \ldots, v_q$ sean los vértices y los vectores de direcciones extremas, respectivamente, de **cl conv** S. Entonces, para cada $y \in S$, existen escalares $\lambda_i \geq 0$, $i = 1, \ldots, p$, $\mu_j \geq 0, j = 1, \ldots, q$, con $\sum_{i=1}^{p} \lambda = 1$, tal que

$$y = \sum_{i=1}^{p} u_i \lambda_i + \sum_{j=1}^{q} v_j \mu_j,$$

de lo cual puede obtenerse que para una x arbitraria, $xy \geq \alpha_o$, $\forall y \in S$, si y sólo si $xu_i \geq \alpha_o$, $i = 1, \ldots, p$, y , $j = 1, \ldots, q$. Así

$$S^{\#} = \{x \in \mathbf{R}^n \mid xu_i \geq \alpha_o, i = 1, \ldots, p \ \wedge \ xv_i \geq 0, j = 1, \ldots, q\},$$

i.e., $S^{\#}$ es poliédrico.

El siguiente resultado describe la propiedad involutoria crucial de los conjuntos polares e inversos polares.

Teorema 5.3 Asuma que $S^{\#} \neq \mathbf{0}$. Entonces

$$S^{\#\#} = \begin{cases} \textbf{cl conv } S + \textbf{cl cono } S, & \text{si } \alpha_o > 0 \\ \textbf{cl cono } S, & \text{si } \alpha_o = 0 \\ \textbf{cl conv } (S \cup \{0\}), & \text{si } \alpha_o < 0 \end{cases}$$

Prueba. $S^{\#\#} = \{x \in \mathbf{R}^n \mid xy \geq \alpha_o, \forall y \in S^{\#}\}$

$$= \left\{ x \in R^n \middle| \begin{array}{ll} xy \geq \alpha_0, & \forall y \in S \\ u_i y \geq \alpha_0 & i = 1, \ldots, p \\ v_j y \geq 0 & j = 1, \ldots, q \end{array} \right\}$$

pero $xy \geq \alpha_o$ es una consecuencia del sistema $u_i y \geq \alpha_o$, $i = 1, \ldots, p$ y $v_i y \geq \alpha_o$, $i = 1, \ldots, q$ (consistente, ya que $S^{\#} \neq \mathbf{0}$), si y sólo si existe un conjunto de $\theta_i \geq 0$, $i = 1, \ldots, p$, $\sigma_i \geq 0$, $i = 1, \ldots, q$, tal que

$$x = \sum_{i=1}^{p} \theta_i u_i + \sum_{i=1}^{q} \sigma_i v_i, \qquad (2.1)$$

con

$$\sum_{i=1}^{p} \theta_i \alpha_o \geq \alpha_o,$$

Ya que $S^{\#}$ es poliédrico, también $S^{\#\#}$ lo es, así que $S^{\#\#}$ es el conjunto cerrado de puntos $x \in \mathbf{R}^n$ de la forma (5.1) con $\theta_i \geq 0$, $i = 1, \ldots, p$, $\sigma_i \geq 0$, $i = 1, \ldots, q$, y los tres conjuntos que son equivalentes

$$\sum_{i=1}^{p} \theta_i \begin{cases} \geq 1, & si \quad \alpha_0 > 0 \\ \geq 0, & si \quad \alpha_0 = 0 \\ \leq 1, & si \quad \alpha_0 < 0 \end{cases}$$

Pero éstas son precisamente las expresiones para los tres conjuntos que según el teorema son iguales a $S^{\#\#}$ en los casos respectivos.

Corolario 5.3.1 cl conv $S = S^{\#\#}_{(1)} \cap S^{\#\#}_{(-1)}$.

Prueba. Se sigue inmediatamente de la prueba del Teorema 5.3, donde $x \in S^{\#\#}_{(1)} \cap S^{\#\#}_{(-1)}$ corresponde a $\Sigma^p_{i=1} \theta_i = 1$.□

Ejemplo 5.1 Considere el siguiente conjunto disyuntivo

$$F = \left\{ x \in R^2 \middle| \begin{array}{c} -x_1 - x_2 \geq -2 \\ x_1 \geq 0 \\ x_2 \geq 0 \\ x_1 \geq 1 \vee x_2 \geq 1 \end{array} \right\}$$

ilustrado en la Fig. 2.1 (a). Sus polares inversos para $\alpha_o = 1$ y $\alpha_o = -1$ son los conjuntos

$$F^{\#}_{(1)} = \left\{ y \in R^2 \middle| \begin{array}{c} 2y_1 \geq 1 \\ 2y_2 \geq 1 \\ y_1 \geq 1 \\ y_2 \geq 1 \end{array} \right\} = \left\{ y \in R^2 \middle| \begin{array}{c} y_1 \geq 1 \\ y_2 \geq 1 \end{array} \right\}$$

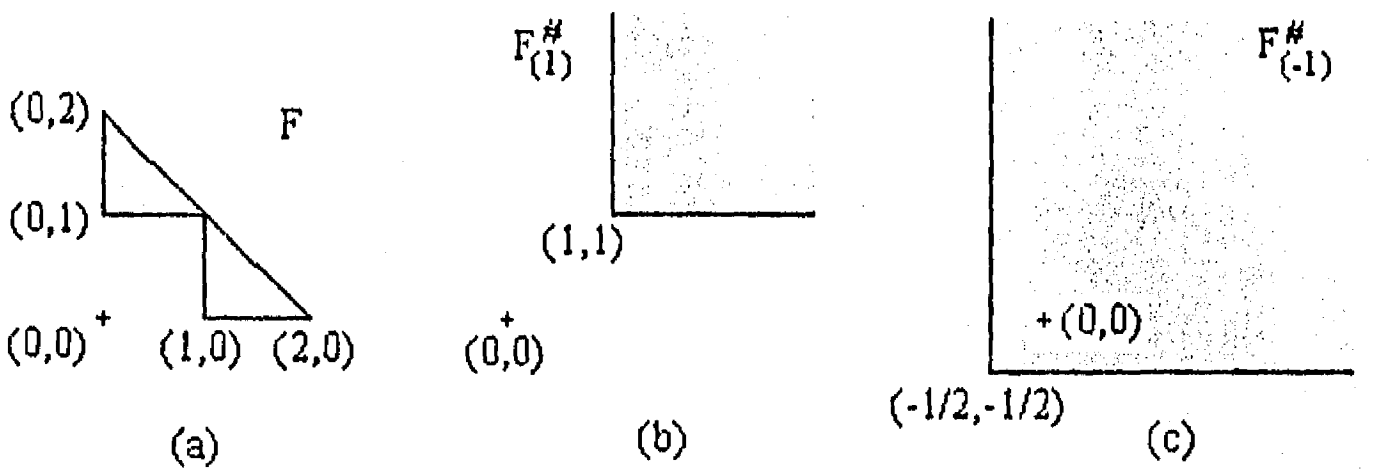
y

$$F^{\#}_{(-1)} = \left\{ y \in R^2 \middle| \begin{array}{c} 2y_1 \geq -1 \\ 2y_2 \geq -1 \\ y_1 \geq -1 \\ y_2 \geq -1 \end{array} \right\} = \left\{ y \in R^2 \middle| \begin{array}{c} 2y_1 \geq -1 \\ 2y_2 \geq -1 \end{array} \right\}$$

mostrados en la Fig. 2.1 (b) y (c).

Finalmente, los conjuntos $F^{\#\#}$ correspondientes a $\alpha_o = 1$ y $\alpha_o = -1$ (mostrados en la Fig. 2.2 (a) y (b)) son:

$$F^{\#\#}_{(1)} = \left\{ x \in R^2 \middle| \begin{array}{c} x_1 + x_2 \geq 1 \\ x_1 \geq 0 \\ x_2 \geq 0 \end{array} \right\}, \qquad F^{\#\#}_{(-1)} = \left\{ x \in R^2 \middle| \begin{array}{c} -x_1 - x_2 \geq -2 \\ x_1 \geq 0 \\ x_2 \geq 0 \end{array} \right\}$$

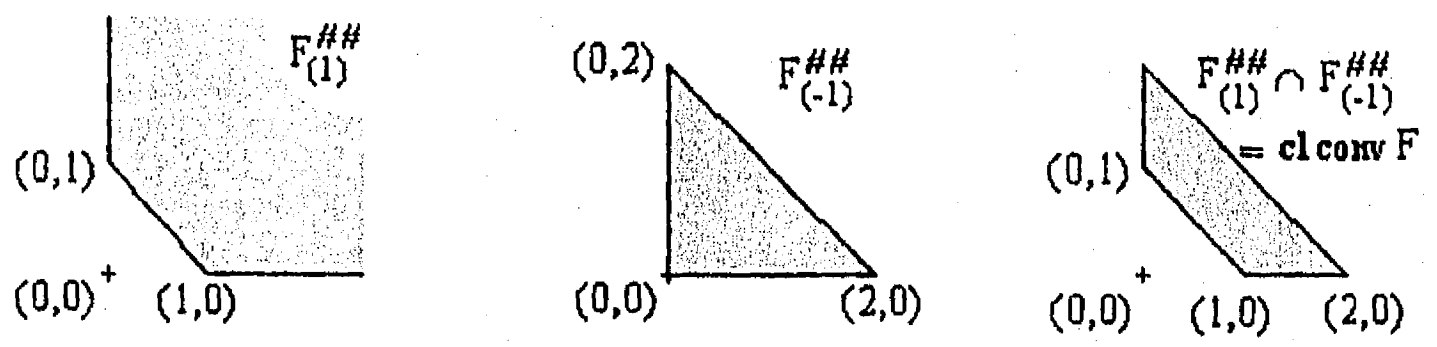


(a) (b) (c)

Fig. 2.1

y su intersección que se muestra en la Fig. 2.1(c) es

$$F^{\#\#}_{(1)} \cap F^{\#\#}_{(-1)} = cl \quad conv \quad F = \left\{ x \in R^2 \middle| \begin{array}{c} -x_1 - x_2 \geq -2 \\ x_1 + x_2 \geq 1 \\ x_1 \geq 0 \\ x_2 \geq 0 \end{array} \right\}$$



(a) (b) (c)

Fig. 2.2

Teorema 5.4 $S^{\#\#\#} = S^{\#}$.

Prueba. Si $\alpha_o \leq 0$, el Teorema se sigue de la propiedad correspondiente de los polares ordinarios. Si $\alpha_o > 0$ y $0 \in$ **cl conv** S, entonces $S^{\#} = \mathbf{0}$, $S^{\#\#} = \mathbf{R}^n$, y $S^{\#\#\#} = \mathbf{0} = S^{\#}$. Finalmente, si $\alpha_o > 0$ y $0 \notin$ **cl conv** S, entonces

$$\begin{aligned} S^{\#\#\#} &= (\textbf{cl conv } S + \textbf{cl cono } S)^{\#} && \text{(del Teorema 2.3)} \\ &= \{y \in \mathbf{R}^n \mid xy \geq \alpha_o, \forall x \in (\textbf{cl conv } S + \textbf{cl cono } S)\} \\ &= \{y \in \mathbf{R}^n \mid xy \geq \alpha_o, \forall x \in S\} = S^{\#}. \end{aligned}$$

Estos resultados pueden ser usados para caracterizar las facetas de **cl conv** S.

Se tiene que una desigualdad del tipo $\pi x \geq \pi_o$ define una faceta de un conjunto poliédrico convexo C si $\pi x \geq \pi_o$, $\forall x \in C$, y $\pi x = \pi_o$ para exactamente d puntos x independientes de C, donde d = dim C. La faceta de C definida por $\pi x \geq \pi_o$ es entonces $\{x \in C \mid \pi x = \pi_o\}$; aunque por brevedad, se puede considerar $\pi x \geq \pi_o$ en sí mismo una faceta. Se procederá en dos pasos, el primero de los cuales sirve para definir las facetas de $S^{\#\#}$.

Teorema 5.5 Sea dim S = n, y $\alpha_o \neq 0$. Encontes $\alpha x \geq \alpha_o$ es una faceta de $S^{\#\#}$ si y sólo si $\alpha \neq 0$ es un vértice de $S^{\#}$.

Prueba. Del Teorema 5.4, $\alpha \in \mathbf{R}^n$ es un vértice de $S^{\#}$ si y sólo si

$$\alpha \in S^{\#} = \left\{ y \in R^n \middle| \begin{array}{lll} uy \geq \alpha_0, & \forall u \in vert & S^{\#\#} \\ vy \geq 0, & \forall v \in dir & S^{\#\#} \end{array} \right\}$$

y α satisface con igualdad un subconjunto de rango n del sistema que define $S^{\#}$. más aún, $\alpha_o \neq 0$ si y sólo si este subconjunto de desigualdades no es homogéneo (*i.e.*, al menos un coeficiente del lado derecho es $\alpha_o \neq 0$).

De otra manera, $\alpha x \geq \alpha_o$ es una faceta de $S^{\#\#}$ si y sólo si (*i*) $\alpha x \geq \alpha_o$, $\forall x \in S^{\#\#}$, *i.e.* $\alpha \in S^{\#}$; y (*ii*) $\alpha x = \alpha_o$ para exactamente n puntos independientes de $S^{\#\#}$. Pero (*ii*) se ajusta si y sólo si $\alpha u = \alpha_o$ para r vértices u de $S^{\#\#}$, y $\alpha v = 0$ para s vectores de dirección extrema v de $S^{\#\#}$, con $r \geq 1$ (ya que $\alpha \neq 0$) y $r + s \geq n$, tal que el sistema de estas ecuaciones es de rango n.

Así, los dos conjuntos de condiciones (para que α_o sea una faceta de $S^{\#\#}$ y para que α sea un vértice de $S^{\#}$) son idénticos.

Por argumentos similares a esta prueba, se puede mostrar que $\alpha x \geq 0$ es una faceta de $S^{\#\#}$ si y sólo si α es un vector de dirección extrema de $S^{\#}$. De diferente manera, para $\alpha_o \neq 0$, la desigualdad homogénea es una faceta de $S^{\#\#}_{(1)}$ si y sólo si es también una faceta de $S^{\#\#}_{(-1)}$ [de $S^{\#\#}_{(0)}$], debido al hecho de que cada vector de dirección extrema de $S^{\#}_{(1)}$ es también un vector de dirección extrema de $S^{\#}_{(-1)}$ [de $S^{\#}_{(0)}$], y vice-versa.

Teorema 5.6 Sea dim S = n, y $\alpha_o \neq 0$: Entonces $\alpha x \geq \alpha_o$ es una faceta de **cl con** S si y sólo si es una faceta de $S^{\#\#}_{(\alpha_o)}$.

Prueba. *(i)* Si $\alpha_o > 0$, el semiespacio $\alpha x \geq \alpha_o$ contiene **cl conv** S si y sólo si contiene **cl con** S + **cl cono** S. Si $\alpha_o < 0$, el semiespacio $\alpha x \geq \alpha_o$ contiene **cl conv** S si y sólo si contiene **cl con** (S $\cup$ {0}). Del Teorema 5.3, en ambos casos $\alpha x \geq \alpha_o$ es un semiespacio de soporte para **cl conv** S si y sólo si es un semiespacio de soporte para $S^{\#\#}_{(\alpha_o)}$.

(ii) A continuación se mostrará que

$$\{x \in \textbf{cl conv } S \mid \alpha x = \alpha_o \} = \{ x \in S^{\#\#}{}_{(\alpha o)} \mid \alpha x = \alpha_o \}$$

La relación $\subseteq$ se sigue de que **cl conv** $S \subseteq S^{\#\#}{}_{(\alpha o)}$ (Teorema 5.3). Para mostrar el inverso, asuma que es falso y deje que $x \in S^{\#\#}{}_{(\alpha o)} \setminus$ **cl conv** S satisfaga $\alpha x = \alpha_o$. Del Teorema 5.3, $x = \lambda u$ para alguna $u \in$ **cl conv** S , y $\lambda > 1$ si $\alpha_o > 0$, $0 < \lambda < 1$ si $\alpha_o < 0$. En cada caso, $\alpha x = \alpha_o$ implica $\alpha u = (1/\lambda)\, \alpha x < \alpha_o$ para alguna $u \in$ **cl conv** $S \subseteq S^{\#\#}{}_{(\alpha o)}$, contrario a la suposición de que $\alpha x \geq$. $\forall x \in S^{\#\#}{}_{(\alpha o)}$.

Por un argumento similar a esta prueba, se puede mostrar que si $\alpha x \geq 0$ es una faceta **de cl conv** S, entonces es una faceta de $S^{\#\#}{}_{(\alpha o)}$ para ambas $\alpha_o = 1$ y $\alpha_o =$ - El contrario;sin embargo, no es verdad, *i.e.*, $\alpha x \geq 0$ puede ser una faceta de ambas $S^{\#\#}{}_{(1)}$ y $S^{\#\#}{}_{(-1)}$, sin ser una faceta de $S^{\#\#}{}_{(1)} \cap S^{\#\#}{}_{(-1)}$.

Ahora, estamos listos para caracterizar las facetas del envolvente convexo del conjunto disyuntivo

$$F = \left\{ x \in R^n \,\middle|\, \bigvee_{h \in Q} \begin{pmatrix} A^h x \geq a_0^h \\ x \geq 0 \end{pmatrix} \right\}$$

donde se asume que Q es finito, y F es completamente dimensional.

Teorema 5.7 $\alpha x \geq \alpha_o$ con $\alpha_o \neq 0$, es una faceta de **cl conv** F si y sólo si $\alpha \neq 0$ es un vértice del poliedro

$$F^{\#}_{(\alpha,0)} = \left\{ y \in R^n \,\middle|\, \begin{array}{c} y \geq \theta^h A^h, \quad h \in Q^* \\ para \quad alguna \quad \theta^h \geq 0, \quad h \in Q^* \\ que \quad satisface \quad \theta^h a_0^h \geq \alpha_0 \end{array} \right\}$$

donde Q* es el conjunto de $h \in$ Q aquellas para las cuales el sistema $A^h x \geq a^h{}_o$, $x \geq 0$ es consistente.

Prueba. Del Teorema 5.3, el conjunto $F^{\#}{}_{(\alpha o)} = \{y \in \mathbf{R}^n \mid xy \geq \alpha_o, \forall x \in F\}$, es de la forma expresada arriba. El resto es una aplicación directa de los Teoremas 5.5 y 5.6.

Similar al caso $\alpha_o = 0$, de estos comentarios se sigue que si $\alpha x \geq 0$ es una faceta de **cl conv** F, entonces $\alpha \neq 0$ es un vector de dirección extrema de $F^{\#}{}_{(\alpha o)}$ para toda α_o. Lo contrario no es cierto, pero si $\alpha \neq 0$ es un vector de dirección extrema de $F^{\#}{}_{(\alpha o)}$ para alguna α_o (entonces para toda α_o), entonces $\alpha x \geq 0$ o es una faceta de **cl conv** F, o es la intersección de las dos facetas $\alpha^1 x \geq \alpha_o{}^1$ y $\alpha^2 x \geq \alpha_o{}^2$ con $\alpha^1 + \alpha^2 = \alpha$ y $\alpha_o{}^1 = - \alpha_o{}^2 \neq 0$.

Ya que las facetas de **el conv** F son vértices (en el caso nohomogéneo) o vectores de dirección extrema (en el caso homogéneo) del poliedro convexo $F^{\#}$, dichas facetas pueden ser encontradas maximizando o minimizando algunas funciones lineales en $F^{\#}$, escogidas apropiadamente, *i.e.*, resolviendo un programa lineal de la forma

$$\min gy, \qquad P_1^*(g,\alpha_o)$$
$$y - \theta^h A^h \geq 0,$$
$$\theta^h a_o{}^h \geq \alpha_o, \qquad h \in Q^*,$$
$$\theta^h \geq 0,$$

o su dual

$$\max \sum_{h\in Q^*} \alpha_o \xi_o{}^h \qquad P_2^*(g,\alpha_o)$$
$$\sum_{h\in Q^*} \xi^h = g$$
$$a_o{}^h \xi_o{}^h - A^h \xi^h \leq 0, \qquad h \in Q^*,$$
$$\xi_o{}^h \geq 0, \xi^h \geq 0,$$

Del Teorema 5.2, si $\alpha_o \leq 0$ entonces $F^{\#}_{(\alpha_o)} \neq \mathbf{0}$, *i.e.*, $P_1^*(g,\alpha_o)$ es siempre factible; mientras que $\alpha_o > 0$, entonces $P_1^*(g,\alpha_o)$ es factible si y sólo si $0 \notin$ **cl conv** F. Esta última condición expresa el hecho obvio de que una desigualdad que corte el punto de origen puede sólo ser derivada de una disyunción que por sí misma corte el origen.

Dos problemas aparecen en conección con el uso de este programa lineal mencionado para generar facetas de **cl conv** F. El primero es que algunas veces, sólo Q es conocido, pero Q* no lo es. Se puede tener cuidado de esto, trabajando con Q en vez de Q*. Si se deja que $P_k(g,\alpha_o)$ denote el problema obtenido cuando se reemplaza Q* con Q en $P_k^*(g,\alpha_o)$, $k = 1,2$. Fué mostrado en [10] que si $P_2(g,\alpha_o)$, tiene una solución óptima ξ tal que

$$(\xi_o{}^h = 0, \xi^h \neq 0) \Rightarrow h \in Q^*,$$

entonces cada solución óptima de $P_1(g,\alpha_o)$ es una solución óptima de $P_1^*(g,\alpha_o)$. Así, uno puede empezar resolviendo $P_2(g,\alpha_o)$. Si la condición mencionada arriba es violada por alguna $h \in Q \setminus Q^*$, entonces h puede ser removida de Q y $P_2(g,\alpha_o)$ resuelta para Q redefinida de esta forma. Cuando sea necesario, este procedimiento puede ser repetido.

El segundo problema es que, ya que las facetas de **cl conv** F de interés primario son las no homogéneas (en particular aquellas con $\alpha_o > 0$, ya que son las que pueden cortar el origen), sería deseable identificar la clase de vectores g para los cuales $P_1^*(g,\alpha_o)$ tiene un mínimo finito. Balas demostró en 1974 [10], que $P_1^*(g,\alpha_o)$ tiene un mínimo finito si y sólo si $\lambda g \in$ **cl conv** F para alguna $\lambda > 0$; y que, g debería satisfacer esta condición, sólo si $\alpha = y^*$ para cada solución óptima (y^*,θ^*) de $P_1^*(g,\alpha_o)$.

Como resultado de estas caracterizaciones las facetas del envolvente convexo del conjunto factible F pueden ser calculadas resolviendo el programa lineal $P_1(g,\alpha_o)$ o su dual. Si la disyunción que define F tiene muchos términos, como en el caso en el que F viene de la formulación de programación disyuntiva de un problema 0-1 con un número grande de condiciones 0-1, $P_1(g,\alpha_o)$ es demasiado grande para que valga la pena resolverlo. Si, sin embargo, F se hace corresponder a una relajación del programa original cero-uno, involucrando condiciones cero-uno sólo para una cuantas variables bien escogidas, entonces $P_1(g,\alpha_o)$ o su dual puede ser resuelto de una manera práctica y provee los cortes más fuertes posibles que pueden ser obtenidos de esas condiciones particulares cero-uno.

De otra forma, ya que el conjunto de restricciones de $P_2(g,\alpha_o)$ consiste de $|Q|$ subsistemas más o menos pobremente conectados, uno está tentado a tratar de aproximar una solución óptima de $P_2(g,\alpha_o)$ -y por lo tanto de $P_1(g,\alpha_o)$- resolviendo los subsistemas independientemente. Las primeras experiencias computacionales indican que estas aproximaciones son bastante buenas.

A continuación se presenta un ejemplo numérico del cálculo de una faceta del envolvente convexo de un conjunto factible F.

Ejemplo 5.2 Encuentre todas las facetas de **cl conv** F que corten el origen (*i.e.*, todas las facetas de la forma $\alpha x \geq 1$), donde $F \in \mathbf{R}^2$ es el conjunto disyuntivo

$$F = F_1 \vee F_2 \vee F_3 \vee F_4,$$

con

$$F_1 = \{ x \mid -x_1 + 2x_2 \geq 6, \quad 0 \leq x_1 \leq 1, x_2 \geq 0\},$$
$$F_2 = \{ x \mid 4x_1 + 2x_2 \geq 11, \quad 1 \leq x_1 \leq 2.5, x_2 \geq 0\},$$
$$F_3 = \{ x \mid -x_1 + x_2 \geq -2, \quad 2.5 \leq x_1 \leq 4, x_2 \geq 0\},$$
$$F_4 = \{ x \mid x_1 + x_2 \geq 6, \quad 4 \leq x_1 \leq 6, x_2 \geq 0\},$$

(ver Fig. 2.3)

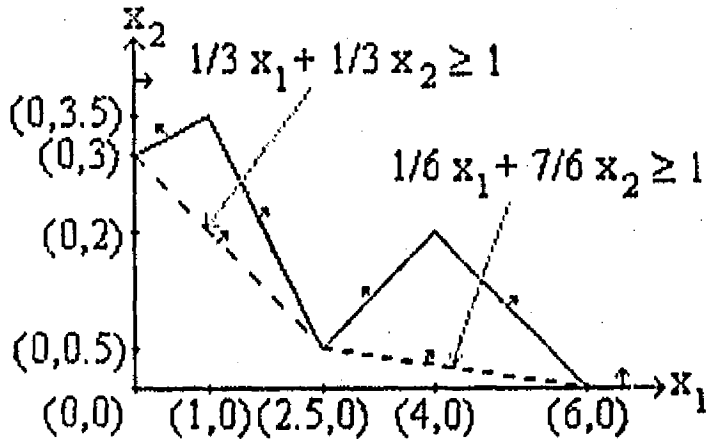


Fig. 2.3

Después de remover algunas restricciones redundantes, F puede ser reescrito como el conjunto de aquellas $x \in \mathbf{R}^2$ que satisfacen

$$\{-x_1 + 2x_2 \geq 6\} \vee \begin{Bmatrix} 4x_1 + 2x_2 \geq 11 \\ -x_1 + x_2 \geq -2 \end{Bmatrix} \vee \{x_1 + x_2 \geq 6\}$$

y el problema $P_1(g,1)$ correspondiente es

$$
\begin{array}{lllllllll}
\min & g_1 y_1 + g_2 y_2 & & & & & & & \\
 & y_1 & & + \theta^1_1 & & & & \geq & 0 \\
 & & y_2 & - 2\theta^1_1 & & & & \geq & 0 \\
 & y_1 & & & - 4\theta^2_1 + \theta^2_2 & \geq & 0 & & \\
 & & y_2 & & - 2\theta^2_1 - \theta^2_2 & \geq & 0 & & \\
 & y_1 & & & & & - \theta^3_1 & \geq & 0 \\
 & & y_2 & & & & - \theta^3_1 & \geq & 1 \\
 & & & 6\theta^1_1 & & & & \geq & 1 \\
 & & & & 11\theta^2_1 - 2\theta^2_2 & \geq & 1 & & \\
 & & & & & & 6\theta^3_1 & \geq & 1 \\
 & & & \theta^1_1, \theta^2_1, \theta^2_2, \theta^3 \geq 0. & & & & &
\end{array}
$$

Resolviendo este programa lineal pra $g = (1,1)$, se tienen los siguientes puntos óptimos

$$(y\,;\,0) = (1/3,1/3;\ 1/6,1/9,1/9,1/6), \qquad (y\,;\,0) = (1/3,1/3;\ 1/6,1/9,1/9,1/3),$$

los cuales tienen el mismo componente-y: (1/3,1/3). Estos puntos son óptimos (y la y asociada es única) para toda $g > 0$ tal que $g_1 < 5\ g_2$. Para $g_1 = 5\ g_2$, en adición a los puntos de arriba; los cuales todavía son óptimos, los puntos

$$(y\,;\,0) = (1/6,7/6;\ 1/6,2/9,13/18,1/6), \qquad (y\,;\,0) = (1/6,7/6;\ 7/12,2/9,13/18,1/6),$$

que nuevamente tienen el mismo componente-y $y = (1/6,7/6)$, también se convierten en óptimos; y son las únicas soluciones óptimas para toda $g > 0$ tal que $g_1 > 5\ g_2$.

Así, se ha encontrado que el envolvente convexo de F tiene dos facetas que cortan el origen y que corresponden a los dos vértices $v_1 = (1/3,1/3)$ y $v_2 = (1/6,7/6)$ de $F^{II}_{(1)}$.

$$
\begin{aligned}
1/3\ x_1 + 1/3\ x_2 &\geq 1, \\
1/6\ x_1 + 7/6\ x_2 &\geq 1.
\end{aligned}
$$

6. Programas Disyuntivos Faciales

En esta sección se presenta el siguiente problema explorado por Balas en 1974 [10]: Dado un problema disyuntivo en la forma normal conjuntiva (2.2), ¿es posible generar el envolvente convexo de los puntos factibles, imponiendo las disyunciones $j \in S$ una por una, en cada paso y calculando el envolvente convexo "parcial", *i.e.*, el envolvente convexo del conjunto definido por las desigualdades generadas antes, más una de las disyunciones?

Por ejemplo, en el caso de un problema entero, ¿es posible generar el envolvente convexo de los puntos factibles, primero produciendo todas las facetas del envolvente convexo de

los puntos que satisfacen las desigualdades lineales, más las condicionesde integralidad sobre, por decir algo, x_1; y entonces, agregando todas las desigualdades correspondientes a las facetas del conjunto de restricciones y generando las facetas del envolvente convexo de los puntos que satisfacen este conjunto corregido de destigualdades, más la condición de integralidad en x_2; etc.? Esta cuestión tiene una importancia práctica obvia. ya que calcular las facetas del envolvente convexo de puntos que satisfacen una disyunción es una tarea considerablemente más fácil, como se mostró en la sección anterior. que calcular las facetas del envolvente convexo del conjunto disyuntivo completo.

La respuesta a las preguntas anteriores es negativa en general, pero positiva para una clase muy importante de problemas disyuntivos, que se pueden denominar faciales. Esta clase incluye programas (puros o enteros) 0-1.

El hecho, de que en el caso general, este procedimiento no produce el envolvente convexo de los puntos factibles puede ser ilustrado en el siguiente problema de 2 variables.

Ejemplo 6.1 Dado el conjunto

$$F_0 = \{x \in \mathbf{R}^2 \mid -2x_1 + 2x_2 \le 1,\ 2x_1 - 2x_2 \le 1,\ 0 \le x_1 \le 2,\ 0 \le x_2 \le 2\}$$

encuentre $F =$ **cl conv** $(F_0 \cap \{x \mid x_1, x_2 \text{ enteros}\})$. Denotando

$$F_1 = \textbf{cl conv}\ (F_0 \cap \{x \mid x_1 \text{ entero}\}), \quad F_2 = \textbf{cl conv}\ (F_1 \cap \{x \mid x_2 \text{ entero}\}),$$

la pregunta ahora es, cuando $F_2 = F$. Como se muestra en la Fig. 2.4, la respuesta es no, ya que

$$F_2 = \{x \mid 2x_1 - x_2 \ge 0, \quad -2x_1 + 3x_2 \ge 0, -2x_1 + x_2 \ge -2, 2x_1 - 3x_2 \ge -2\ \}$$

mientras que $F = \{x \mid -x_1 + x_2 = 0,\ 0 \le x_1 \le 2,\ 0 \le x_2 \le 2\}$,

Si se cambia el orden en el que las restricciones de integralidad se impusieron, la salida será la misma.

Considera un problema disyuntivo expresado en su forma normal conjuntiva (2.2), y denote

$$F_0 = \{x \in \mathbf{R}^n \mid Ax \ge a_0,\ x \ge 0\}$$

El problema disyuntivo (y su conjunto de restricciones) es llamado *facial* si cada desigualdad $d^i x \ge d_{i0}$, que aparece en una disyunción de (2.2) define una cara de F_0; *i.e.* si para toda $i \in Q_j$, $j \in S$, el conjunto

$$F_0 \cap \{x \mid d^i x \ge d_{i0}\ \}$$

es una cara de F_0. (Una cara de un poliedro P es la intersección de P con algunos de sus planos frontera).

Claramente, el programa disyuntivo es *facial* si y sólo si para cada $i \in Q_j$, $j \in S$, $F_0 \subseteq \{x \mid d^i x \geq d_{i0}\}$.

La clase de programas disyuntivos que tienen la propiedad facial incluyen los casos más importantes de la programación disyuntiva, como la programación 0-1 (pura o mixta), la programación cuadrática noconvexa, la programación separable, los problemas de complementariedad lineal, etc.; pero no el problema general de programación entera. En todos los casos mencionados, las desigualdades $d^i x \geq d_{i0}$ de cada disyunción, actualmente definen facetas, *i.e.*, $(d - 1)$-dimensionales de F_0, donde d es la dimensión de F_0.

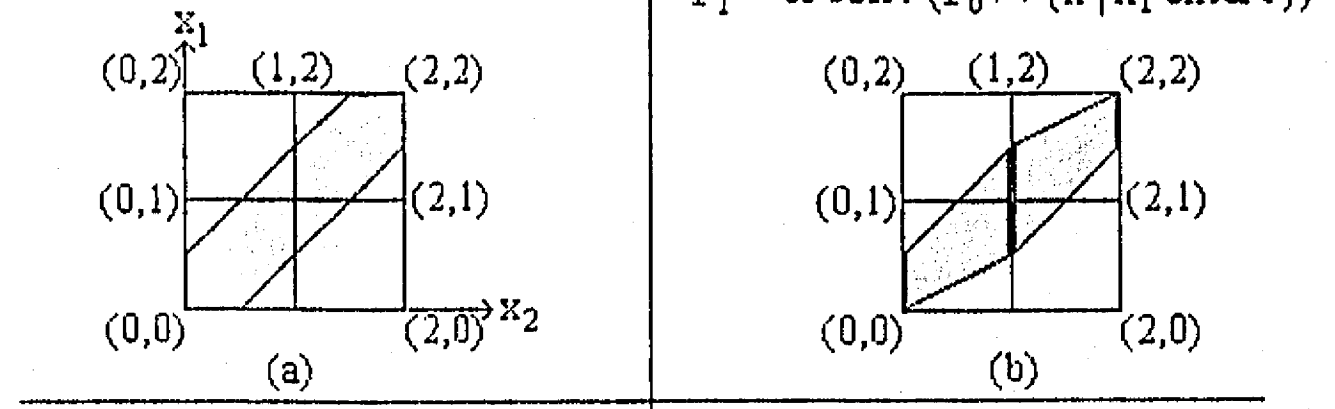


Fig. 3.4

Teorema 6.1 Sea el conjunto de restricciones de un programa disyuntivo representado como

$$F = \{ x \in F_0 \mid \bigvee_{i \in Q_j} (d^i x \geq d_{i0}), j \in S \}$$

donde $$F_0 = \{x \in \mathbf{R}^n \mid Ax \geq a_0,\ x \geq 0\}$$

Para un ordenamiento arbitrario de S, defina recursivamente

$$F_j = \text{conv} \left[\bigvee_{i \in Q_j} (F_{j-1} \cap \{x \mid d^i x \geq d_{i0} \}) \right], \qquad j = 1, \ldots, |S|$$

Si F es facial y F_0 está acotado, entonces $F_{|S|} = \mathbf{conv}\ F$.

La prueba de este teorema usa el siguiente resultado auxiliar [10].

Lemma 6. Sea $P_1, \ldots, P_r$ un conjunto finito de polítopos (poliedros acotados), *y* $P=\bigcup^r_{h=1} P_h$.

Sea $H^+ = \{x \in \mathbf{R}^n \mid d^i x \le d_{i0}\}$ un semiespacio arbitrario, y $H^- = \{x \in \mathbf{R}^n \mid d^i x \ge d_{i0}\}$. Si $P \subseteq H^+$, entonces

$$H^- \cap \mathbf{conv}\ P = \mathbf{conv}\ (H^- \cap P)$$

Prueba. Sea $H^- \cap \mathbf{conv}\ P \neq \mathbf{0}$ (de otra manera el Lemma sería trivial). Claramente, $(H^- \cap P) \subseteq (H^- \cap \mathbf{conv}\ P)$, y por lo tanto

$$\mathbf{conv}\ (H^- \cap P) \subseteq \mathbf{conv}\ (H^- \cap \mathbf{conv}\ P) = H^- \cap \mathbf{conv}\ P.$$

Para probar $\supseteq$, sean $u_1, \ldots, u_p$ los vértices de todos los polítopos P_h, $h = 1, \ldots, r$. Obviamente, p es finito, **conv** P es cerrado, y **vert conv** $P \subseteq (\bigcup^r_{h=1}$ **vert** $P_h)$. Entonces

$$x \in H^- \cap \mathbf{conv}\ P \Rightarrow d^i x \ge d_{i0} \quad \text{y} \qquad x = \sum_{i=1}^{p} \lambda_k u_k$$

con $$\sum_{i=1}^{p} \lambda_k = 1, \quad \lambda \ge 0, \qquad k = 1, \ldots, p.$$

Más aún, $P \subseteq H^+$ implica $d^i u_k \le d_{i0}$, $k = 1, \ldots, p$. Si asumimos que en esta expresión, para x, si $\lambda_k > 0$ entonces $d^i u_k \ge d_{i0}$. Para mostrar esto, se asume que existe $\lambda_k > 0$ tal que $d^i u_k < d_{i0}$. Entonces

$$d^i x = d^i \left(\sum_{i=1}^{p} \lambda_k u_k\right) < d_{i0} \left(\sum_{i=1}^{q} \lambda_k\right)$$

$$= d_{i0},$$

una contradicción. Aquí, x es la combinación convexa de los puntos $u^k \in H^- \cap P$, o $x \in$ **conv** $(H^- \cap P)$.

Otra relación que se usa en la prueba de este teorema es el hecho que para cada arbitraria $S_1, S_2 \subseteq \mathbf{R}^n$,

$$\mathbf{conv}\ (\mathbf{conv}\ S_1 \cup \mathbf{conv}\ S_2) = \mathbf{conv}\ (S_1 \cup S_2).$$

Prueba del Teorema 6.1. Para $j = 1$, el postulado es verdadero, de acuerdo a las definiciones y a la relación obvia

$$\bigvee_{i \in Q_j} (F_0 \cap \{x \mid d^i x \ge d_{i0}\}) = \{x \in F_0 \mid \bigvee_{i \in Q_j} (d^i x \ge d_{i0})\}. \qquad (6.1)$$

Para probar el teorema por inducción en j, suponga que el postulado es verdadero para $j = 1, \ldots, k$. Entonces

$$F_{k+1} = \mathbf{conv}\ [\bigvee_{i \in Q_{k+1}} (F_k \cap \{x \mid d^i x \ge d_{i0}\})\ ], \qquad \text{por definición}$$

$$= \mathbf{conv}\,[\bigvee_{i\in Q_{k+1}}(\{x \mid d^i x \geq d_{i0}\} \cap \mathbf{conv}\{x\in F_0 \mid \bigvee_{i\in Q_j}(d^i x \geq d_{i0}), j=1,\ldots,k\})]$$

(de la suposición)

$$= \mathbf{conv}\,[\bigvee_{i\in Q_{k+1}}\mathbf{conv}(\{x \mid d^i x \geq d_{i0}\} \cap \{x\in F_0 \mid \bigvee_{i\in Q_j}(d^i x \geq d_{i0}),\ j=1,\ldots,k\})]$$

(del Lemma 6.1.1, ya que este programa disyuntivo es facial, por lo tanto $F_0 \subseteq \{x \mid d^i x \leq d_{i0}\}$, y F_0 está acotado.)

$$= \mathbf{conv}\,[(\bigvee_{i\in Q_{k+1}}\{x \mid d^i x \geq d_{i0}\}) \cap \{x\in F_0 \mid \bigvee_{i\in Q_j}(d^i x \geq d_{i0}), j=1,\ldots,k\}]$$

(de (6.1) y de la distributividad de $\cap$ con respecto a $\vee$)

$$= \mathbf{conv}\,\{x\in F_0 \mid \bigvee_{i\in Q_j}(d^i x \geq d_{i0}), j=1,\ldots,k+1\},$$

i.e., el postulado es también verdadero para $j = k + 1$.

El Teorema 6.1 implica que para un programa disyuntivo facial acotado con un conjunto factible F, el envolvente convexo de F puede ser generado en |S| etapas, (donde S tiene la forma (2.2)), generando en cada etapa un envolvente convexo "parcial", llamado el envolvente convexo de un programa disyuntivo con una sola disyunción.

En términos de un programa 0-1, por ejemplo este resultado significa que el problema

$$\min \{cx \mid Ax \geq b, 0 \leq x \leq e,\ x_j = 0 \text{ ó } 1, j = 1, \ldots, n\},$$

donde $e = (1, \ldots, 1)$, es equivalente a (tiene el mismo envolvente convexo de sus puntos factibles, que)

$$\min \{cx \mid Ax \geq b, 0 \leq x \leq e, \alpha^i x \geq \alpha_{i0}, i\in H_1, x_j = 0 \text{ ó } 1, j = 2, \ldots, n\}, \quad (6.2)$$

donde $\alpha^i x \geq \alpha_{i0}$, $i\in H_1$ son las facetas de

$$F_1 = \text{conv}\,\{x \mid Ax \geq b, 0 \leq x \leq e,\ x_j = 0 \text{ ó } 1, j = 1, \ldots, n\},$$

En otras palabras, se garantiza que x_1 tenga un valor entero en una solución de (6.2). Un programa 0-1 en n variables puede, así, ser reemplazado por uno en n - 1 variables al costo de introducir nuevas desigualdades lineales. Las desigualdades por sí mismas no son difíciles de generar, ya que la disyunción que le da origen ($x_1 = 0 \vee x_1 = 1$) tiene sólo dos términos. La dificultad está, más bien en el número de facetas que uno tendría que generar, utilizando este enfoque para resolver problemas 0-1. Sin embargo, usando alguna información como la de cuáles desigualdades (facetas de un envolvente convexo "parcial") son las que nos van a acercar al óptimo, uno puede ser capaz de hacer este procedimiento eficiente, generando sólo unas cuantas facetas del envolvente convexo "parcial" en cada iteración.

CAPITULO 2: Bibliografía en Programación Disyuntiva

[1] J. A. Araoz Durand, *Polyhedral Neopolarities*, Ph. D. Dissertation, University of Waterloo (November **1973**).

[2] E. Balas, *Intersection cuts - A new type of cutting planes for integer programming*. Operations Res. 19 (**1971**) 19-39.

[3] E. Balas, *Integer programming and convex analysis: Intersection cuts from outer polars*, Math. Programming 2 (**1972**) 330-382.

[4] E. Balas, *Ranking the facets of the octahedron*, Discrete Math. 2 (1972) 1-15.

[5] E. Balas, *Noncovex quadratic programming via generalized polars*, SIAM J. Appl. Math. 28 (1975) 335-349.

[6] E. Balas, *A constraint-activating outer polar method for pure and mixed integer 0-1 programs*, in: P. L. Hammer and G. T. Zoputendijk, eds., Mathematical Programming in Theory and Practice, (North-Holland, kamsterdam, 1974 275-310).

[7] E. Balas, *On the use of intersection cuts in branch and bound*. Paper presented at the 8th International Symposium on Mathematical Programming, Stanford, CA (August 27-31, 1973).

[8] E. Balas, *Intersection cuts from disjunctive constraints*, MSRR No. 330, Carnegie-Mellon University, (February 1974).

[9] E. Balas, *Disjunctive programming: Properties of the convex hull of feasible points*, MSSR NO. 348, Carnegie-Mellon University (July 1974).

[10] E. Balas, *A Note on duality in disjunctive programming*, J. Optimization Theory Appl. 15 (1977).

[11] E. Balas, *Set covering with cutting planes from conditional bounds*, MSRR No. 339, Carnegie-Mellon University (July 1977).

[12] E. Balas, *Set covering with cutting planes from conditional bounds*, MSRR No. 399, Carnegie-Mellon University (July 1976).

[13] E. Balas, V.J. Bowman, F. Glover and D. Sommer, *An intersection cut from the dual of the unit hypercube*, Operations Res. 19 (1971) 40-44.

[14] E. Balas and R. Jeroslow, *Strengthening cuts for mixed integer programs*, MSSR No. 359, Carnegie-Mellon University (February 1975).

[15] E. Balas and A. Zoltners, *Intersection cuts from outer polars of truncated cubes*, Naval Res. Logist. Quart. 22 (1975) 477-496.

[16] M. Bellmore and H. D. Ratliff, *Set covering and involutory bases*, Management Sci. 18 (1971) 194-206.

[17] C. E. Blair and R. G. Jeroslow, *A converse for disjunctive constraints*, MSRR No. 393, Carnegie-Mellon University (June 1976).

[18] J. Borwein, *A strong duality theory for the minimum of a family of convex programs*, Dalhousie University, Halifax (June 1978).

[19] C. A. Burdet, Polaroid: *A new tool in nonconvex and integer programming*, Naval Res. Logist. Quart. 20 (1973) 13-24.

[20] D. R. Fulkerson, Clocking Polyhedra, in: B. Harris, ed., **Graph Theory and Its Applications** (Academic Press, New York, 1972) 93-112.
[21] ID. R. Fulkerson, *Anti-blocking polyhedra*, J. Combinatorial Theory 12 (1972) 50-71.
[22] D. R. Fulkerson, *Blocking and anti-blocking pairs of polyhedra*, Math. Programming 1 (1971) 168-194.
[23] F. Glover, *Convexity cuts and cut search*, Operations Res. 21 (1973) 123-134.
[24] F. Glover, *Convexity cuts for multiple choice problems, Discrete Math.* 6 (1973) 221-234.
[25] F. Glover, *Polyhedral annexation in mixed integer and combinatorial programming*, Math. Programming 9 (1975) 161-188.
[26] F. Glover and D. Klingman and J. Stutz, *The disjunctive facet problem: formulation and solution techniques*, Operations Res. 22 (1974) 582-601.
[27] J.J. Greenberg and W. P. Pierskalla, *Stability theorems for infinitely constrained mathematical programs*. J. Optimization Theory Appl. 16 (1975) 409-428.
[28] B. Grünbaum, **Convex Polytopes** (J. Wiley, New York, 1967).
[29] Hoang Tuy, *Concave programming under linear constraints*. Soviet Math. Dokl. (1964) 1437-1440.
[30] R.G. Jeroslow, *The principles of cutting plane theory: part I.* Carnegie Mellon University (February 1974).
[31] R. G. Jeroslow, *Cutting planes for relaxations of integer programs*, MSRR No. 347, Carnegie Mellon University (July 1974).
[32] R. G. Jeroslow. *Cutting plane theory: Disjunctive methods*. Ann. Discrete Math., vol. 1: Studies in Integer Programming, (1977) 293-330.
[33] R. G. Jeroslow, *Cutting planes for complementarity constraints*, SIAM J. Control 16 (1978).
[34] R. G. Jeroslow, *A cutting plane game and its algorithms*, CORE Discussion Paper 7724 (June 1977).
[35] E. L. Johnson. *The group problem for mixed integer programming*, Math. Programming Study 2 (1974) 137-179.
[36] B. Owen, *Cutting planes for programs with disjunctive constraints*, J. Optimization Theory Appl. 11 (1973) 49-55.
[37] R. T. Rockafellar., **Convex Analysis** (Princeton University Press, Princeton, N.J. (1970).
[38] J. Stoer and C. Witzgall, **Convexity and Optimization in Finite Dimensions I**, (Springer, Berlin, 1970).
[39] J. Tind, *Blocking and anti-blocking sets*, Math. Programming 6 (1974) 157-166.
[40] R.D. Young, *Hypercylindrically deduced cuts in zero-one integer programs*. Perations Lrese. 19 (1971) 1393-1405.
[41] E. Zemel, *On the facial structure of 0-1polytopes*, Ph. D. Dissertation, GSIA, Carnegie Mellon University (May 1976).
[42] P. Zwart, *Intersection cuts for separable programming*, Washington University, St. Louis (January 1972).

CAPITULO 3: LA PROGRAMACION LINEAL MIXTA-LOGICA

En este capítulo se presentan los conceptos, algoritmos y definiciones que constituyen la programación lineal mixta-lógica y que junto con el procesamiento lógico descrito en el siguiente capítulo constituyeron la serie de herramientas aplicadas en el desarrollo de esta investigación, mismas que estan por publicarse en el artículo: Hooker, Osorio: *Programación Lineal Mixta-Lógica*. Empezando por los antecedentes, se presentan la forma general y la interpretación disyuntiva de la programación lineal mixta-lógica, así como su algoritmo básico. Posteriormente se discute la serie de relajaciones y cortes que puede aplicarse a los problemas resueltos por medio de la programación mixta-lógica.

1. Antececentes

Entre los trabajos precursores que presentaron un enfoque general basado en la lógica aplicada a la Investigación de Operaciones se encuentra el tratado sobre métodos booleanos publicado por Hammer y Rudeanu[25] en 1968. También Granot y Hammer[23] sugirieron en 1971 la posibilidad de utilizar métodos booleanos para la programación entera.

Las ideas teóricas más importantes del enfoque de programación lineal mixta-lógica descrito aquí fueron articuladas primero, por Jeroslow [41] quien estaba interesdo primariamente en asuntos de representabilidad. El vió las variables discretas como herramientas para representar un subconjunto factible de un espacio continuo, como es el caso de un modelo de programación lineal mixta-lógica o de programación lineal mixta-entera. De esto se sigue que los modelos de programación lineal mixta-lógica y de programación lineal mixta-entera son esencialmente modelos de programación disyuntiva con todas las propiedades descritas en el capítulo anterior. En un trabajo conjunto con Lowe [42], Jeroslow probó que un modelo de programación lineal mixta entera representa una unión finita de muchos poliedros si y sólo si tienen el mismo cono de recesión.

Al mismo tiempo, Williams [66,67, 68,70], Blair [8,9] y Hooker [28, 29, 30, 31] exploraron las conecciones entre la lógica y la optimización. El trabajo de Beaumont en 1990 [6] fué aparentemente el primer estudio sistemático de programación lineal mixta-lógica como una técnica de solución para problemas de optimización. Basado en el trabajo seminal de Balas en programación disyuntiva [2,3,4], describe familias de cortes válidos que pueden ser utilizados para crear relajaciones de restricciones disyuntivas.

Más recientemente, Hooker argumentó en 1994 [32] que un enfoque basado en la lógica aplicado a la optimización, incluyendo la programación lineal mixta-lógica, puede explotar la estructura del problema en formas paralelas a las técnicas poliédricas tradicionales. Wilson [71, 72, 73] estudió los cortes lógicos y las formulaciones basadas en la lógica.

Es crucial demostrar el valor práctico de la programación lineal mixta-lógica en el dominio de un problema. Esto fué realizado ampliamente por Grosmann en el área de la síntesis de procesos en Ingeniería Química, en una serie de artículos donde tiene como coautores a Hooker, Turkay, Yan y particularmente a Raman [37, 49, 50, 51, 52, 64]. Estos trabajos desarrollaron algunas de los conceptos claves de programación lineal mixta-lógica discutidos aquí. Bollapragada, Ghattas y Hooker también obtuvieron resultados alentadores en el área de diseño estructural [10].

Es instructivo constrastar la programación lineal mixta-lógica con otros enfoques que combinan elementos discretos y continuos.

El enfoque de programación lineal mixta-lógica de McAloon y Tretkoff [44,45], combina la programación por procedimientos con la programación declarativa. El elemento discreto es representado por una rutina proporcionada por el usuario que controla la formulación y la solución de modelos lineales que representan el elemento continuo. Esto contrasta con el enfoque de programación lineal mixta-lógica descrito aquí, en el cual ambos elementos son modelados en una forma declarativa. Los dos enfoques no son incompatibles, aunque el enfoque de McAloon proporciona un ambiente en el cual se pueden implementar la mayoría de las técnicas presentadas aquí.

Aún cuando los problemas de optimización puramente binarios tienen un elemento continuo en el sentido de que las restricciones se representan por medio de desigualdades lineales, y no es obvio como aplicarles los métodos basados en la lógica, un enfoque desarrollado por Barth [5] es el de derivar fórmulas de las desigualdades que pueden ser procesadas con métodos de inferencia lógica. Algunas de las técnicas de Barth pueden mejorar la fase del procesamiento lógico de los algoritmos de programación lineal mixta-lógica.

El trabajo de McAloon, Tretkoff y Barth está influenciado por varias corrientes de investigación que históricamente se han orientado a problemas discretos pero que están exprerimentando con diferentes formas de incorporar variables continuas. Los modelos de programación lógica introducidos por Colmerauer [16] y Kowalski [43] permiten formular un problema en un subconjunto de lógica de primer orden (cláusulas lógicas de Horn). Las versiones recientes del lenguaje de programación lógica PROLOG [11,61], tal como PROLOG III [17] (y pronto IV), incorporan programación lineal.

La integración de la solución por restricciones con la programación lógica fué formalizada en el esquema de *programación lógica por restricciones* de Jaffar y Lassez

[39]. Generaliza el paso de "unificación" de los métodos de inferencia lógica para abarcar la solución por restricciones en general [40].

La programación lineal por restricciones proporciona un ambiente para integrar métodos de satisfacción de restricciones desarrollados en la comunidad de inteligencia artificial (y en todas partes) con las ideas de programación lógica [20, 63, 65]. Un número de sistemas se ha desarrollado a lo largo de esta línea, en adición al Prolog III, incluyendo CLP® [39], CAL[1], CHIP [9,59], el resolvedor ILOG [46], y otros paquetes [12,57,53]. La programación lineal tiene un lugar en varios de estos sistemas. A diferencia de la programación lineal mixta-lógica, estos métodos descansan en el modelamiento por procedimientos. También carecen del énfasis de la programación lineal mixta-lógica en la explotación de la estrucutra del problema para la generación de cortes y relajaciones, aunque la literatura de la programación por restricciones ha mostrado algún interés en explotar la estructura (p. ej. [21]).

1. Forma General

Un modelo de programación lineal mixta-lógica tiene la forma

$$\begin{array}{ll} & \min cx \\ \text{s.a.} & g_i(y,h),\ i\in I \\ & y_j \rightarrow (A^j x \geq a^j), j\in J \end{array} \qquad (2.1)$$

El modelo tiene una parte lógica y una parte continua. La parte lógica consiste de fórmulas $g_i(y,h)$ que involucran proposiciones atómicas $y = (y_1,\ldots,y_n)$, las cuales son o verdaderas o falsas. Dicha fórmula puede ser $y_1 \vee y_2$, la cual siginica que y_1 o y_2 (o ambas) deben ser verdaderas. También debe haber algunas variables $h = (h_1,\ldots,h_m)$ que tomen varios valores discretos. La parte continua asocia cada proposición atómica y_j con un sistema $A^j x \geq a^j$ de desigualdades lineales. El sistema se fuerza cuando y_j es verdadero. Así la fórmula $y_1 \vee y_2$ en efecto, requiere de cualquier solución x que satisfaga $A^1 x \geq a^1$ o $A^2 x \geq a^2$ (o ambas). En general, x es factible si satisface los sistemas lineales forzados por al menos un valor (y,h) que satisfaga las fórmulas lógicas.

El problema puede ser resuelto en un árbol de búsqueda de ramificación y acotamiento, ramificando sobre los valores verdaderos de las y_j y los valores discretos de las h_j. En cada nodo del árbol de búsqueda, uno resuelve un problema de programación lineal que contiene las restricciones que corresponden a las y_j que son verdaderas, más cualquiera de los cortes agregados para reforzar la relajación. Un elemento clave de la programación lineal mixta-lógica es el de aplicar un algoritmo de inferencia lógica a las fórmulas lógicas antes de resolver el problema lineal. Esto puede generar futuras restricciones lógicas y puede fijar algunas y_j o h_j adicionales. Existen preprocesadores que realizan esta función de una manera limitada y *ad hoc*.

Como se comentó desde la introducción y se profundiza en el siguiente apartado, es fácil mostrar que cualquier problema de programación lineal mixta-lógica es equivalente a un problema de programación disyuntiva cuyo conjunto de restricciones en una disyunción de sistemas lineales (*i.e.*, la solución debe satisfacer al menos uno de los sistemas). Su conjunto factible es, por lo tanto, una unión finita de muchos poliedros en el espacio de las variables continuas.

3. Interpretación Disyuntiva

Cualquier modelo de programación lineal mixta-lógica (2.1) puede ser escrito en forma de un problema de programación disyuntiva,

$$\begin{aligned} &\min cx \\ \text{s.a.} \quad &\bigvee_{t\in T} A^t x \geq a^t, \end{aligned} \tag{3.1}$$

donde T es un conjunto finito de índices. Esto es hecho, dejando que la disyunción (3.1) sea

$$\bigvee_{y} \{A^j x \geq a^j, \quad | \, y_j \text{ es verdadero, } j = 1,\ldots,n\} \tag{3.2}$$

Aquí, y tiene un rango que comprende todos los valores que satisfacen las restricciones lógicas; *i.e.*, cada valor y para el cual existe una h que haga cada $g_i(y,h)$ verdadera.

Porque cada disyunción de (3.1) representa un poliedro en el espacio-x, el conjunto factible de (3.1), y por lo tanto de cualquier modelo de programación lineal mixta-lógica es una unión finita de muchos poliedros. Jeroslow mostró que esta unión puede ser representada por un conjunto de restricciones de programación lineal mixta-entera si todos los poliedros tienen el mismo cono de recesión. El cono de recesión de un poliedro P es el conjunto de todas las direcciones d tal que para alguna $x' \in P$, $x' + \alpha d \in P$ para toda $\alpha \geq 0$. Esta restricción sobre los conos de recesión es de poca importancia práctica, porque si uno pone límites superiores grandes sobre todas las variables, el cono de recesión de todos los poliedros es el origen.

Por el contrario, cada modelo de programación lineal mixta-lógica puede ser escrito como un problema de disyuntivo, porque puede ser transformado a un problema 0-1,

$$\begin{aligned} &\min cx \\ \text{s.a.} \quad &Ax + By \geq a \\ &y_j \in \{0,1\}, \quad \forall j, \end{aligned} \tag{3.3}$$

lo cual a su vez, es equivalente a un modelo disyuntivo cuyo conjunto de restricciones es

$$\bigvee_{y} Ax \geq a - By,$$

donde el rango de y contiene todos lo vectores 0-1.

Una fórmula lógica individual $g_i(y,h)$ puede también tener una interpretación disyuntiva. Su conjunto factible puede ser mirado como (3.1), donde el rango de y contiene todos los valores que hacen $g_i(y,h)$ verdadera para alguna h. El conjunto factible de $g_i(y,h)$ puede, por lo tanto, ser mirado como la unión finita de muchos poliedros en el espacio-x.

4. Algoritmo Básico

El algoritmo básico de programación lineal mixta-lógica ramifica sobre los valores verdaderos de las variables proposicionales y_j y sobre los valores de las variables discretas h_j. Cuando ramifica, fija y_j a verdadero o falso, y la fórmula y_j o $\neg y_j$ se convierte en una de las fórmulas lógicas de $g_i(y,h)$. Cuando h_j se fija a v, el dominio D_j de h_j (*i.e.*, el conjunto de sus valores posibles) se reduce en $\{v\}$.

A continuación, se aplica un algoritmo de procesamiento lógico a las fórmulas. Esto puede generar fórmulas adicionales (cortes lógicos). También puede fijar los valores de las y_j adicionales o remover elementos de algunas D_j. Si las fórmulas son insatisfechas, esto puede o no ser descubierto, dependiendo de la fuerza del algoritmo, pero si es descubierto, la búsqueda regresa al nodo padre del presente nodo. En el siguiente capítulo se discuten los algoritmos de procesamiento lógico en mayor detalle.

Si se desea, se pueden generar ahora, cortes lineales, como se discute en la siguiente sección de este capítulo. Se formula un problema de programación lineal cuyas restricciones son las desigualdades $A^j x \geq a^j$ que corresponden a los valores verdaderos de las y_j, más cualquier corte adicional. Si el problema lineal es infactible, el algoritmo regresa al nodo padre del presente nodo. De otra manera, la solución x' del problema lineal, satisfacerá, en general, ciertos conjuntos de restricciones $A^j x \geq a^j$ y no otros. Si no se ha fijado aún la proposión y_j a verdadera o falsa, temporalmente se asume que es verdadera si x' satisface $A^j x \geq a^j$ y falsa en otro caso. Si una y_j no fija corresponde a un conjunto vacío de restricciones, puede tomar cualquier valor asignado que aplique hasta que se fije de otra manera.

En este punto, las variables y_j son verdaderas o falsas y las variables discretas h_j pueden tener dominios de varios tamaños. Si estos valores pueden hacer todas las fórmulas verdaderas, x' es una solución factible. Si alguna $g_i(y,h)$ es falsa o tiene un valor verdadero no determinado porque h no está totalmente determinada), uno puede tratar de generar un corte de separación; *i.e.*, una desigualdad válida para $g_i(y,h)$ que x' viole. La generación de cortes de separación se trata en la siguiente sección. Si no se genera dicha desigualdad, entonces $g_i(y,h)$ se mira como una fórmula no satisfecha. Debería notarse que si x' cae dentro del envolvente convexo del conjunto factible de $g_i(y,h)$ pero no dentro del conjunto factible en sí mismo, entonces ninguna desigualdad lineal puede cortarla, y $g_i(y,h)$ será inevitablemente clasificada como no satisfecha.

Deje que G sea un conjunto de fórmulas lógicas, inicialmente las fórmulas $g_i(y,h)$ en (2.1)
Deje que L sea un conjunto de desigualdades lineales, inicialmente vacío,
Deje que V,F,I indiquen verdadero, falso o indefinido.
Deje que y' sea un vector de valores verdaderos para y, inicialmente $y'=(I,\ldots,I)$
Deje que $D = (D_1,\ldots,D_m)$ sean los dominios de $h_1,\ldots,h_m$.
Deje que z' sea un límite superior sobre el valor óptimo,inicialmente ∞.
Deje que A sea el conjunto de nodos activos, inicialmente con $A=\{(G,L,y',D)\}$.
Mientras que A no esté vacío:
  Remueva un tuple (G,L,y',D) de A.
  Aplique un procesamiento lógico a G, posiblemente cambiando los
    contenidos de G, posiblemente cambiando algunas de las y_j de I
    a V o F, y posiblemente removiendo elementos de algunas de las D_j.
  Si no se detecta contradicción lógica, **entonces**
  Para cada y'_j cambiada a V, adicione $A^j x \geq d^j$ a L.
  Genere cortes de desigualdad para las fórmulas en G y agréguelos a L.
  Deje que x' minimice cx sujeto a L.
  Si $cx' < z'$ **entonces**
    Para cada y_j:
      Si $y'_j \in \{V,F\}$ entonces deje que $y''_j = y'_j$.
      si no deje que $y''_j = V$ si $A^j x' \geq d^j$ y $y''_j = F$ de otra manera.
    Deje que C, inicialmente vacío, sea el conjunto de fórmulas no satisfechas.
    Para cada $g_i(y, h) \in G$:
      Si $g_i(y'',h')$ es F o I **entonces**
        Si se desea, trate de generar un corte de separación para
          $g_i(y,h)$ con respecto a (y'',h'').
        Si se genera un corte de separación,**entonces** agréguelo a L.
        si no, adicione $g_i(y,h)$ a C.
      Si C está vacío, **entonces**
        Si no se generaron cortes de separación, **entonces**
          x' es factible; deje $x^* = x'$ y $z' = cx'$.
        si no adicione (G,L,y',D) a A.
      si no
        Escoja una variable y_j con $y'_j = I$ o una variable h_j con $|D_j| > 1$,
          tal que se pueda hacer y_j V o F, o haga h_j igual a uno de sus valores
          discretos, que satisfaga o tienda a satisfacer una de las fórmulas en C.
          Si y_j se escoge **entonces**
            Adicione $(G \cup \{y_j\},L,y',D)$ y $(G \cup \{\neg y_j\},L,y',D)$ a A.
          si no, si h_j se escoge **entonces**
            Para cada $v \in D_j$:
              Haga D' = D, haga $D'_j = \{v\}$, y adicione (G,L,y',D) a A.
Si $z' < \infty$ entonces x^* es una solución óptima.
si no el problema es infactible.

Fig. 3.1 *Algoritmo genérico de ramificación para la Programación Lineal Mixta-Lógica*

Finalmente, se escoge para ramificación alguna de las variables y_j o h_j a las que no se les haya asignado valor. Esta variable debe escogerse de tal manera que cuando tome algún valor, al menos se satisfaga una fórmula no satisfecha, o quizás, la acerque a la satisfacción en algún sentido. Una presentación más precisa de este algoritmo aparece en la Fig. 3.1

5. Relajaciones y Planos de Corte

El problema de programación lineal resuelto en cada nodo de un árbol de búsqueda de programación lineal mixta-lógica provee un límite inferior para el valor óptimo en cada nodo. Sin embargo, el problema lineal contiene sólo aquéllas restricciones que son forzadas por los valores verdaderos de las variables proposicionales. Las fórmulas lógicas que no fijen cualquiera de sus variables no están representadas en la relajación lineal. Esto, por lo tanto proveerá un límite débil,y cuando sea posible es importante mejorarlo con cortes adicionales que puedan representar las fórmulas lógicas.

Esta sección presenta algunas de las técnicas para obtener relajaciones lineales de las fórmulas lógicas por medio de la generación de cortes válidos en variables continuas. Algunos de estos cortes emulan el efecto de la relajación continua tradicional de un modelo 0-1, aunque la fuerza y naturaleza de la relajación tradicional es muy pobremente entendida, tomando en cuenta el grado de utilización que tiene, por lo que, un análisis de dicha relajación formará una parte importante de esta sección.

5.1 Envolvente Convexo

Se enfatizó antes que cualquier fórmula lógica $g_i(y,h)$ es equivalente a una disyunción de conjuntos de restricciones lineales,

$$\bigvee_{t\in T} A^t x \geq a^t, \tag{5.1}$$

Su conjunto factible es, por lo tanto una unión finita de muchos poliedros, y una descripción del envolvente convexo de esta unión es la mejor relajación lineal posible de la fórmula.

En algunos casos, el envolvente convexo es tan grande que aún la mejor relajación posible es pobre o inútil. Si por ejemplo, x está acotado a $0 \leq x \leq m$, no es poco común para el envolvente convexo de (5.1) abarcar la mayor parte o toda la caja descrita por $0 \leq x \leq m$. Un ejemplo notorio de este caso, aparece en los problemas de calendarización. Si las operaciones 1 y 2 empiezan en los tiempos x_1 y x_2 y duran 2 minutos, uno puede imponer la restricción disyuntiva

$$(x_2 \geq x_1 + 2) \vee (x_1 \geq x_2 + 2)$$

para asegurar que una ocurre después de la otra. Los límites superiores m representan el último tiempo en el cual una operación m puede ser calendarizada y son, por lo tanto,

mucho más grandes que 2. La línea punteada de la Fig. 3 encierra el envolvente convexo cuando $m = (10,10)$. En este caso, la mejor relajación posible viene dada por $x_1 + x_2 \geq 2$. $x_1 + x_2 \geq 18$ y $0 \leq x_j \leq 10$. Esto no es demasiado diferente de $0 \leq x_j \leq 10$ y es probablemente muy poco útil en la práctica.

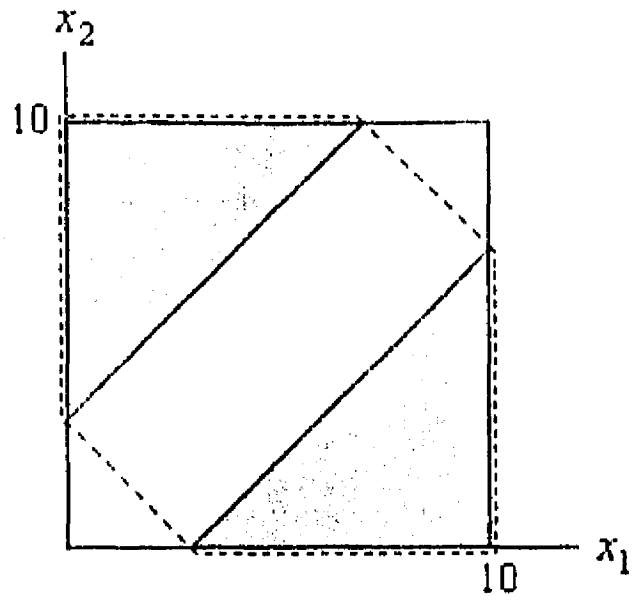


Fig. 3.2 *Envolvente convexo del conjunto factible de una disyunción para un problema de calendarización*

Un ejemplo aún más patético es el de las variables semicontinuas. Si $0 \leq x_j \leq 4$ y se impone y la disyunción

$$(0 \leq x_j \leq 1) \vee (3 \leq x_j \leq 4)$$

el envolvente convexo es el intervalo completo [0,4] y cualquier relajación posible es, por lo tanto, totalmente inútil.

5.2 Cortes Disyuntivos y Duales

Una relajación de (5.1) puede ser obtenida generando cortes válidos que parcial o completamente describan el envolvente convexo. Balas [4] caracterizó los cortes válidos para (5.1) como sigue. Primero, note que $bx \geq \beta$ es un corte válido para una disyunción factible $A^l x \geq a^l$ si y sólo si está dominado por una combinación lineal no negativa (o *surrogada*) de $A^l x \geq a^l$. Una surrogada dominante puede ser escrita como $uAx \geq ua$, donde $b \geq uA$, $\beta \leq ua$ y $u \geq 0$. Pero $bx \geq \beta$ es un corte válido para la disyunción si y sólo si es válido para cada elemento de la disyunción; *i.e.* para cada elemento de la disyunción se puede encontrar una surrogada que domine $bx \geq \beta$.

Teorema 5.1 (Balas) La desigualdad $bx \geq \beta$ es un corte válido para (5.1) si y sólo si para cada sistema factible $A^l x \geq a^l$ existe una $u^l \geq 0$ tal que $b \geq u^l A^l$ y $\beta \leq u^l a^l$.

Dado cualquier conjunto de surrogadas $u^l A^l x \geq u^l a^l$, si $x \geq 0$, uno puede, inmediatamente, escribir el corte disyuntivo válido

$$(\max_{t\in T}\{\, u^t A^t\})x \geq \min_{t\in T}\{\, u^t a^t\} \qquad (5.2)$$

para (5.1), donde el máximo es uno de los mismos componentes. El teorema 5.1 claramente implica que si $x \geq 0$, cada corte válido está dominado por un corte disyuntivo (5.2).

La fuerza y utilidad de un corte disyuntivo (5.2) depende radicalmente de la selección de las surrogadas. Uno puede, en principio, generar cortes disyuntivos para definir cada faceta del envolvente convexo, pero esto es frecuentemente impráctico. La tarea de obtener una buena relajación para (5.1) es en esencia la tarea de escoger los multiplicadores u^t juiciosamente.

Una selección inicialmente atractiva para u^t viene dada por la solución de un problema dual. Cada surrogada debería, idealmente, dar el mejor límite posible sobre la función objetivo cx. Esto es, u^t debería ser escogido de tal manera que el valor mínimo de cx sujeto a $u^t A^t x \geq u^t a^t$ sea maximizado. La u^t deseada se puede ver fácilmente como la solución óptima del problema lineal dual de $\min\{cx \mid A^t x \geq a^t\}$, donde u^t es el vector de las variables duales. (Para poner esto de manera diferente, la surrogada dual para un problema de programación lineal es idéntica al dual del problema lineal [22]).

La dificultad con este enfoque es que ya que $A^t x \geq a^t$ es sólo una pequeña parte del conjunto original de restricciones, puede no acoplarse con la función objetivo. Esto es, las variables x_j que tienen coeficientes no cero en cx pueden tener coeficientes cero en $A^t x \geq a^t$, y viceversa. Esto significa que cx no provee información para guiar la selección de u^t, una situación que de hecho es común en la práctica.

Un remedio posible es el de incluir más restricciones en el problema cuyo dual se resuelve, de tal manera que capture el lazo entre cx y $A^t x \geq a^t$. Esto puede ser hecho como sigue. En cualquier nodo del árbol de búsqueda se fuerza un sistema $Ax \geq a$ con ciertas restricciones lineales fijadas por los valores verdaderos de las variables proposicionales. Si $Ax \geq a$ es incluído en cada término de la disyunción (5.1), esta se convierte en

$$\bigvee_{t\in T} \{A^t x \geq a^t,\ Ax \geq a\}$$

Para cada t, uno resuelve el dual de

$$\begin{array}{lll} & \min cx & \\ \text{s.a} & A^t x \geq a^t & (u^t) \\ & Ax \geq a & (u) \end{array} \qquad (5.3)$$

donde (u^t,u), como se mostró son las variables duales. Una solución óptima del dual proporciona un conjunto razonable de multiplicadores u^t para el corte disyuntivo (5.1).

Desafortunadamente este enfoque parece ser impráctico, porque (5.3) es generalmente, un problema lineal grande y toma tiempo resolver el dual de (5.3) para cada elemento de la

disyunción. De hecho, si uno ramifica sobre la disyunción, forzando cada elemento en turno, (5.3) es precisamente el problema lineal que uno resolvería en cada nodo hijo del nodo actual. Así, uno podría ramificar sobre la disyunción en vez de relajarla. Podría haber alguna ventaja en relajar varias disyunciones simultáneamente, pero el tiempo invertido es bastante grande y probablemente, difícil de justificar. La discusión restante va, por lo tanto a enfocarse en mecanismos mucho más rápidos para escoger multiplicadores efectivos u^t.

5.3 Cortes Elementales

La clase más común de restricción disyuntiva (5.1) es una en la cual, cada elemento de la disyunción es una simple desigualdad.

$$\bigvee_{t\in T} a^t x \geq \alpha^t \tag{5.4}$$

Beaumont[6] mostró como generar un corte para para (5.4) que es equivalente a la relajación continua de la formulación tradicional 0-1 de (5.4). Esto es

$$\begin{aligned} & a^t x \geq \alpha^t - M_t(1 - y_t), \quad t\in T \\ & \Sigma_{t\in T}\, y_t = 1 \\ & 0 \leq x \leq m \\ & y_t \in \{0,1\}, \quad t\in T \end{aligned} \tag{5.5}$$

Cada M_t se escoge de tal manera que $\alpha^t - M_t$ es un límite inferior sobre el valor de $a^t x$. Los límites $0 \leq x \leq m$ se imponen para asegurar que dicho límite inferior exista. Puede ser asumido, sin pérdida de generalidad que $M_t > 0$, porque de otra manera la desigualdad en (5.5) carece de sentido y puede ser desechada. Beaumont obtiene un corte tomando una combinación lineal de las desigualdades en (5.5), donde cada desigualdad t recibe un peso $1/M_t$. Esto origina lo que Beaumont llama el *corte elemental* para (5.4)

$$\left(\sum_{t\in T}\frac{a^t}{M_t}\right)x \geq \sum_{t\in T}\frac{\alpha_t}{M_t} - |T| + 1 \tag{5.6}$$

Teorema 5.2 (Beaumont) El corte elemental (5.6) es equivalente a la relajación continua de (5.5). Esto es, el conjunto factible de (5.6) y $0 \leq x \leq m$ es igual a la proyección del conjunto factible de la relajación continua de (5.5) sobre el espacio-x.

Uno puede también probar equivalencia aplicando la eliminación de Fourier a (5.9) con el objeto de eliminar y. Es fácil mostrar que (5.6) y $0 \leq x \leq m$ son las desigualdades resultantes.

Una técnica similar obtiene cortes elementales para todas las fórmulas lógicas que se pueden expresar como restricciones de "maleta".

$$
\begin{aligned}
& dy \geq \delta \\
& y_t \rightarrow (a^t x \geq \alpha^t), \quad t \in \mathrm{T} \\
& 0 \leq x \leq m,
\end{aligned}
\tag{5.7}
$$

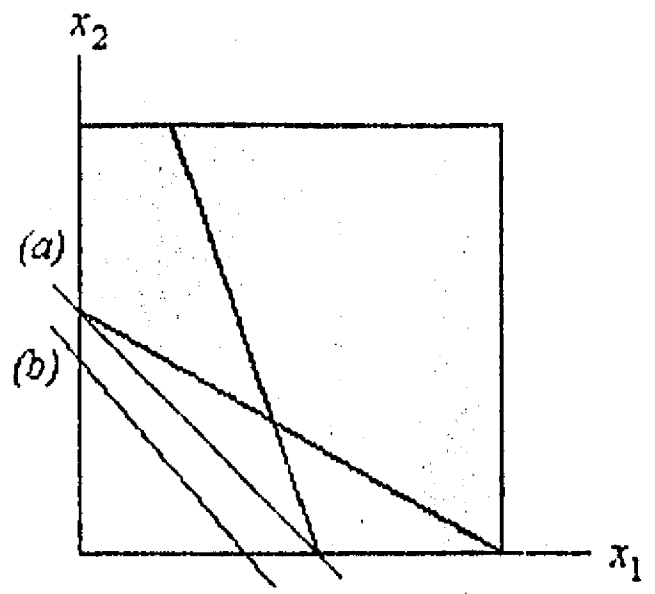


Fig. 3.3 *(a) Un corte elemental de soporte y (b) un corte elemental no de soporte.*

donde $d \geq 0$. Es verdad que (5.7) puede ser puesta en forma disyuntiva usando el esquema (2.1). pero esto puede requerir un gran número de elementos en la disyunción. (Las disyunciones son, por supuesto un caso especial de $dx \geq \delta$ en donde $d = 1$ y cada $d_j \in \{0,1\}$). La representación 0-1 de (5.7) es

$$
\begin{aligned}
& a^t x \geq \alpha_t - M_t(1 - y_t), \quad t \in \mathrm{T} \\
& 0 \leq x \leq m \\
& dy \geq \delta \\
& y_t \in \{0,1\}, \quad t \in \mathrm{T}
\end{aligned}
\tag{5.8}
$$

Una combinación lineal de las desigualdades, usando pesos d_t/M_t, nos da el corte elemental

$$
\left(\sum_{t \in T} a^t \frac{d_t}{M_t}\right) x \geq \sum_{t \in T} \alpha_t \frac{d_t}{M_t} - \sum_{t \in T} d_t + \delta \tag{5.9}
$$

Esto es en general, más débil que la relajación continua de (5.8). Si $\Sigma_t d_t = \delta$, por ejemplo, (5.8) fuerza todos los elementos de la disyunción que representa, mientras que (5.9) sólo fuerza una combinación lineal de estos elementos.

Beaumont obtiene M_t sólo de los límites $0 \leq x \leq m$ haciendo

$$
\alpha_t - M_t = \sum_j \min\{0, a^t_j\} m_j. \tag{5.10}
$$

En muchos casos, se puede obtener un mejor límite inferior para $a^t x$, que resulte en un corte más fuerte. Un método es el de minimizar $a^t x$ sujeto a cada uno de los demás elementos de la disyunción y $0 \le x \le m$ y escoger el más pequeño de los valores mínimos. Por lo tanto, M_t se escoge de tal manera que

$$\alpha_t - M_t = \min_{t' \neq t} \{ \min_{x} \{ a^t x \mid a^{t'} x \ge \alpha_{t'}, 0 \le x \le m \} \} \qquad (5.11)$$

Considere por ejemplo, el siguiente conjunto de restricciones, cuyo conjunto factible es el área sombreada en la Fig. 3.3

$$(x_1 + 2\,x_2 \ge 2) \vee (3\,x_1 + x_2 \ge 3)$$
$$0 \le x_j \le 2.$$

La formulación 0-1 es

$$x_1 + 2\,x_2 \ge 2 - M_1(1 - y_1)$$
$$3\,x_1 + x_2 \ge 3 - M_2(1 - y_2)$$
$$y_1 + y_2 = 1$$
$$0 \le x_j \le 2, \qquad y_j \in \{0,1\}$$

Beaumont hace $(M_1,M_2) = (2,3)$ lo cual resulta en el corte $3/2\,x_1 + 4/3\,x_2 \ge 1$. Por contraste, (5.11) pone $(M_1,M_2) = (1,2)$, lo cual da el corte más fuerte $x_1 + x_2 \ge 1$. Esto es un *corte de soporte* en el sentido de que define un hiperplano de soporte para el conjunto factible.

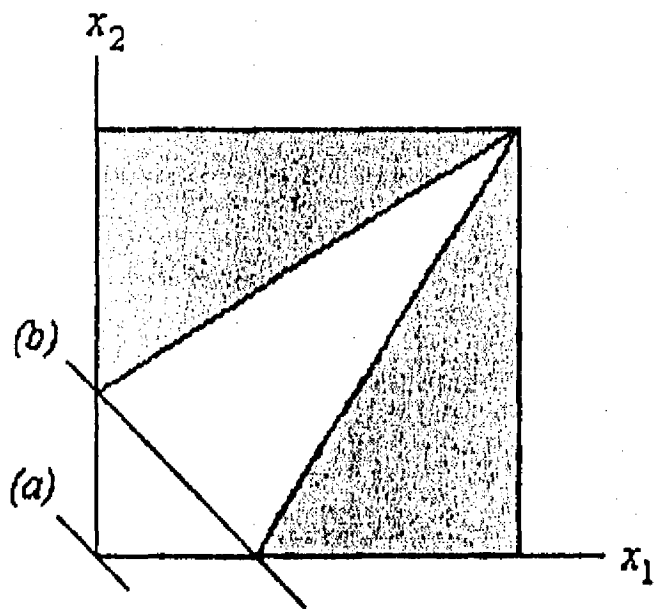


Fig. 3.4 *(a) Un corte elemental y (b) un corte elemental reforzado*

Aún cuando (5.11) es usado para calcular M_t, el corte resultante puede fallar en ser de soporte. Considere las restricciones (Fig. 3.4),

$$(-x_1 + 2\,x_2 \ge 2) \vee (2\,x_1 - x_2 \ge 2)$$
$$0 \le x_j \le 2.$$

La ecuación (5.10) hace $(M_1,M_2) = (4,4)$, lo cual resulta en el corte $x_1 + x_2 \ge 0$ que es totalmente inútil. El corte puede, obviamente ser reforzado a $x_1 + x_2 \ge 1$.

Cuando las desigualdades $a^i x \geq \alpha_i$ en (5.7) son reemplazadas por sistemas de desigualdades $A^i x \geq a^i$, se requieren de muchos cortes elementales para efectuar el efecto de la relajación tradicional. Deje que cada sistema $A^i x \geq a^i$ consista de desigualdades $A^{ii} x \geq a_i^i$ para $i \in I_i$. La formulación 0-1 es

$$\begin{aligned} & A^i x \geq a_i - M^i(1 - y_i), \quad i \in T \\ & 0 \leq x \leq m \\ & dy \geq \delta \\ & y_i \in \{0,1\}, \quad i \in T. \end{aligned} \tag{5.12}$$

Aquí M^i es un arreglo tal que para $i \in I_i$, $a_i^i - M_i^i$ es un límite inferior sobre $A^{ii} x$. Aplicaciones repetidas de la eliminación de Fourier revelan que la proyección del conjunto factible de (5.12) sobre el espacio-x es descrita por el conjunto de desigualdades de la forma,

$$\left(\sum_{i \in T} A^{ii_i} \frac{d_i}{M_{i_i}^i}\right) x \geq \sum_{i \in T} a_{i_i}^i \frac{d_i}{M_{i_i}^i} - \sum_{i \in T} d_i + \delta$$

para todos los posibles vectores $(i_1, \ldots, i_{|T|}) \in I_1 \times \ldots \times I_{|T|}$.

Los cortes elementales pueden, por lo tanto, ser imprácticos cuando las y_i correspondan a sistemas de desigualdades. En esos casos uno puede usar los cortes de separación óptima (descritos a continuación) o la relajación tradicional.

5.4 Cortes Elementales de soporte

El ejemplo de la Fig. 3.4 muestra que un corte elemental puede fallar en ser de soporte. En esos casos sólo se trata de incrementar el lado derecho hasta que soporte el conjunto factible, para obtener así, un corte elemental *reforzado*. De hecho, existe una fórmula de forma cerrada para obtener el mejor lado derecho posible. La fórmula permite checar fácilmente cuando un corte elemental dado es de soporte, y cuando no lo es, para mejorar sobre la relajación continua tradicional que el corte representa.

Las figuras 3.3 y 3.4 pueden sugerir que los dos elementos de la disyunción $a^1 x \geq \alpha_1$, $a^2 x \geq \alpha_2$ producen un corte elemental de soporte si y sólo si los vectores a_1 y a_2 forman un ángulo agudo, y que se puede descubrir una relación similar para más de dos elementos de la disyunción. Un tercer ejemplo revela que la situación es más complicada que esto. En la Fig. 3.5 se muestra el conjunto factible para

$$\begin{aligned} & (-3\, x_1 + x_2 \geq -3) \vee (-x_2 \geq -1) \\ & 0 \leq x_j \leq 3 \end{aligned}$$

El corte elemental $3\, x_1 + 2\, x_2 \leq 12$, es de soporte aún cuando (-3,1) y (0,-1) formen un ángulo obtuso.

En un análisis más adecuado, deje que $bx \geq \beta$ sea el corte elemental reforzado, donde bx es el lado izquierdo del corte elemental (5.9). Ya que $bx \geq \beta$ define un hiperplano de soporte para el conjunto factible de (5.4), β es el más pequeño de los valores mínimos obtenidos minimizando bx sujeto a cada uno de los elementos $a^t x \geq \alpha_t$ de la disyunción. Esto es,

$$\beta = \min_{t \in T} \beta_t, \tag{5.13}$$

donde

$$\beta_t = \min\{bx \mid a^t x \geq \alpha_t, 0 \leq x \leq m\}. \tag{5.14}$$

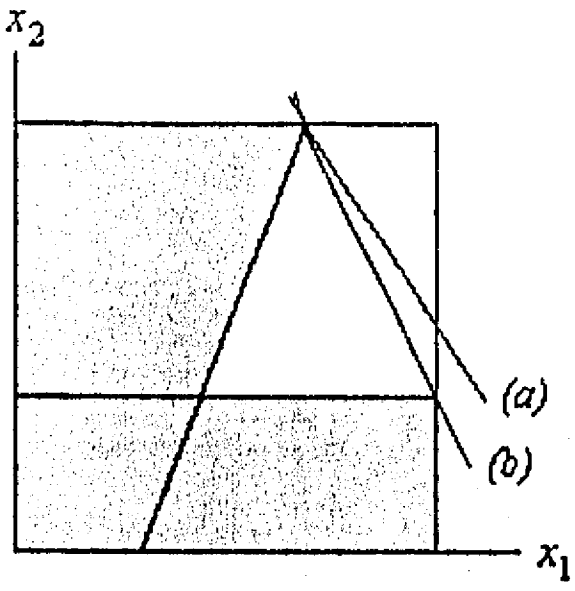


Fig. 3.5 *(a) Un corte elemental de soporte y (b) un corte que define una faceta*

El cálculo de β_t es simplificado si $b \geq 0$, porque en este caso los límites superiores $x \leq m$ pueden ser ignorados. Para finalizar, se puede introducir el cambio de variable,

$$\hat{x} = \begin{cases} x_j & si \quad b_j \geq 0 \\ m_j - x_j & en \quad otro \quad caso \end{cases}$$

El corte elemental reforzado en términos de $\hat{x}$, digamos $\hat{b}\hat{x} \geq \hat{\beta}$, puede ahora, ser calculado, donde $\hat{b} = |b_j|$. El lado derecho de $bx \geq \beta$ puede ser recuperado de (5.13), haciendo

$$\beta_t = \hat{\beta}_t + \sum_{\substack{j \\ b_j < 0}} m_j b_j \,. \tag{5.15}$$

Falta por calcular

$$\hat{\beta}_t = \min\left\{\hat{b}x \mid \hat{a}^t \hat{x} \geq \hat{\alpha}, \hat{x} \geq 0\right\}, \tag{5.16}$$

donde

$$\hat{a}'_j = \begin{cases} a'_j & si \quad b'_j \geq 0 \\ -a'_j & en \quad otro \quad caso \end{cases} \tag{5.17}$$

y

$$\hat{\alpha}_t = \alpha_t - \sum_{\substack{j \\ h_j<0}} m_j a^t_j \tag{5.18}$$

Porque $\hat{b} \geq 0$, la dualidad del problema lineal aplicada a (5.16) da

$$\hat{\beta} = \min_{\substack{j \\ \hat{a}^t_j>0}} \left\{ \frac{\hat{b}_j}{\hat{a}^t_j} \right\} \max\{\hat{\alpha}_t, 0\}. \tag{5.19}$$

Esto prueba el,

Teorema 5.3 El corte elemental (5.9) para la disyunción (5.4) es un corte de soporte si y sólo si su lado derecho es igual a β, como se define por (5.13), (5.15) y (5.19).

5.5 Representaciones Integrales 0-1

La relajación tradicional continua de una restricción disyuntiva, actualmente tiene dos debilidades. Puede ser débil, como en los casos mencionados, pero aún cuando es fuerte, permite soluciones fraccionales cuando la disyunción original es satisfecha. Esto significa que un método tradicional de ramificación y acotamiento puede seguir ramificando aún cuando se haya descubierto una solución factible. Por lo tanto, es mejor checar las disyunciones (así como otras restricciones lógicas) directamente buscando factibilidad, como se hace en la programación lineal mixta-lógica.

La formulación 0-1 de la disyunción (5.1) es la siguiente

$$\begin{aligned} & A^t x \geq a^t - M_t(1 - y_t), \quad t \in \mathrm{T} \\ & 0 \leq x \leq m \\ & \textstyle\sum_{t \in \mathrm{T}} y_t = 1 \\ & y_t \in \{0,1\}, \quad t \in \mathrm{T}, \end{aligned} \tag{5.20}$$

donde M_t viene dada por (5.11). La afirmación aquí es que cuando x se fija a algún valor x', un punto extremo de solución $y = y'$ de (5.20) puede ser no entero, aún cuando x' satisfaga (5.1). Un ejemplo de esta situación se presenta en un caso sencillo de variable semicontinua, $x \in \{0\} \cup [s_1, s_2]$, o

$$\begin{aligned} & (-x \geq 0) \vee (x \geq s_1) \\ & 0 \leq x \leq s_2. \end{aligned} \tag{5.21}$$

La relajación continua de (5.20) es

$$\begin{aligned} & -x \geq -s_2(1 - y) \\ & x \geq s_1 - s_1 y \\ & 0 \leq x \leq s_2. \\ & 0 \leq y \leq 1. \end{aligned} \tag{5.22}$$

Si x se fija a x' y (5.22) se proyecta sobre y, el resultado es

$$1-\frac{x'}{s_1} \le y \le 1-\frac{x'}{s_2}, \quad 0 \le y \le 1. \tag{5.23}$$

Si $0 \le x' \le s_2$, $y' = 1 - x'/s_2$ es un punto de solución extrema de (5.23) y por lo tanto de (5.22) y no es entero cuando $0 \le x' \le s_2$. Así, (5.22) puede tener soluciones de puntos extremos con valores fraccionales de y aún cuando $x' \in [s_1, s_2]$, y aún cuando (5.22) es la mejor relajación posible (el envolvente convexo) de (5.20). Las soluciones de punto etremo para $x' \in [s_1, s_2]$ está garantizado que tienen una y entera sólo cuando $s_1 = s_2$; *i.e.*, cuando x es esencialmente un variable binaria reescalada.

La idea puede ser definida en general como sigue. Deje $P_{x'}$ ser el conjunto de puntos y que satisfacen (5.20) cuando x se fija a x'. Deje la relajación continua de (5.20) ser entera si para cada (x', y') que satisface (5.20) tal que y' es un punto extremo de $P_{x'}$, y' es entera.

Lo siguiente caracteriza las relajaciones enteras. Un elemento de la disyunción (5.1) es redundante cuando su conjunto factible cae dentro de otra disyunción. Obviamente, las disyunciones redundantes pueden ser deshechadas sin ningún efecto.

Teorema 5.4 Suponga que la disyunción (5.1) contiene elementos no redundantes en la disyunción. Para $t, t' \in T$ con $t \neq t'$ defina

$$y^*_t(t') = \max \{y_t \mid M_t y_t \le A^t x - a^t + M_t,\ A^{t'} x \ge a^{t'},\ 0 \le x \le m,\ y_t \le 1\}.$$

Entonces, la relajación continua de (5.20) es entera si y sólo si $y^*_t(t') = 0$ para cada par $t, t' \in T$ con $t \neq t'$.

Prueba. Es claro que $y^*_t(t')$ puede ser escrita,

$$y^*_t(t') = \max \{y_t \mid A^t x \ge a^t - M_t(1 - y_t),\ A^{t'} x \ge a^{t'},\ 0 \le x \le m,\ y_t \le 1\}. \tag{5.14}$$

Es conveniente dejar $S_t = \{x \mid A^t x \ge a^t,\ 0 \le x \le m\}$ para $t \in T$.

Afirmación. Para cualquier $x' \in S_{t'}$ y cualquier $t \neq t'$,

$$y^*_t(t') = \max_y \{y_t \mid y \in P_{x'}\}. \tag{5.15}$$

Prueba de la afirmación. Es suficiente mostrar que cualquier y_t que es factible en (5.15), es factible (5.14), y viceversa. Primero suponga que y_t es factible en (5.15). Entonces, dejando $x = x'$ sea factible en (5.14), porque por virtud del hecho de que $x' \in S_{t'}$. Por el contrario, deje que y_t sea factible en (5.14). Para ver que es factible en (5.15), deje $y_{t'} = 1 - y_t$ y $y_{t''} = 0$ para $t'' \neq t, t'$. Es suficiente mostrar que

$$A^{t''} x' \ge a^{t''} - M_{t''}(1 - y_{t''}) \tag{5.16}$$

, (5.16) ajusta para $t'' = t$ por estipulación. Se ajusta para $t'' \neq t, t'$ por definición de $M_{t''}$. Esto prueba la afirmación.

ponga que $y^*_t(t') > 0$ para alguna t,t' con $t \neq t'$. Porque el elemento t' de la unción no es redundante, uno puede escoger $x' \in S_{t'} \setminus S_t$. Esto implica que $y^*_t(t') < 1$, lo cual junto con $y^*_t(t') > 0$ significa que $y^*_t(t')$ no es entera. También (5.25) implica que alguna y' con $y'_t = y^*_t(t')$ es un punto extremo de $P_{x'}$. Se sigue que (5.20) no es entera.

Por el contrario, suponga que $y^*_t(t') = 0$ para todos los pares t,t' con $t \neq t'$. Es suficiente mostrar que para cualquier x' que satisfaga (5.1), cualquier punto extremo dado y' de $P_{x'}$ es entero. Si se supone que $x' \in S_{t'}$, se puede establecer lo siguiente.

$$\max\{y_t \mid y \in P_{x'}\} = 1 \qquad (5.27)$$
$$\max\{y_t \mid y \in P_{x'}\} = 0, \quad t \neq t'.$$

El primero es debido simplemente al hecho de que $x' \in S_{t'}$. Por la afirmación de arriba, el segundo es equivalente a $y^*_t(t') = 0$, lo cual es dado. Pero (5.27) implica que $P_{x'}$ es un segmento de línea de longitud unitaria que se extiende desde el origen en una dirección positiva a lo largo del eje $y_{t'}$. Entonces, cualquier punto extremo $y' \in P_{x'}$ es entero, lo cual significa que (5.20) es entera. □

Corolario 5.1 Considere una disyunción (5.4) con una desigualdad por disyunción y límites $0 \leq x \leq m$. Si (5.4) contiene elementos de la disyunción no redundantes, entonces (5.20) es entera si y sólo si

$$\max\{a^t x \mid a^{t'} x \geq \alpha_{t'}, 0 \leq x \leq m\} = \alpha_t - M_t \qquad (5.28)$$

para cada $t,t' \in T$ con $t \neq t'$.

Las condiciones del Teorema 5.4 y el Corolario 1 son estrictas. De hecho,

Corolario 5.2 La relajación continua de (5.20) es entera si y sólo si los conjuntos factibles descritos por los elementos de la disyunción (5.1) están separados.

Prueba. Suponga que dos de los conjuntos factibles se intersectan, p. ej. aquellos que corresponden a los elemntos t y t' de la disyunción. Entonces $y^*_t(t') = 1$, lo cual viola la condición del teorema. □

Ni aún los conjuntos factibles separados son condición suficiente para la integralidad, como se mostró en el ejemplo antes mencionado.

5.6 Cortes de Beaumont

Beaumont [6] identifica una clase de cortes que bajo ciertas condiciones, definen facetas del envolvente convexo del conjunto factible para restricciones disyuntivas en las cuales cada elemento de la disyunción consiste de una simple desigualdad, como en (5.4).

Desafortunadamente, las condiciones son frecuentemente insatisfechas, lo cual limita la utilidad de estos cortes.

El enfoque de Beaumont es esencialmente un método razonable para escoger multiplicadores u^t así como para generar un corte disyuntivo (5.2). El primero incorpora los límites $x \leq m$ dentro de la disyunción (5.4) para obtener

$$\bigvee_{t\in T} \begin{bmatrix} -I \\ a^t \end{bmatrix} x \geq \begin{bmatrix} -m \\ \alpha_t \end{bmatrix}, \quad t \in T \tag{5.29}$$

El vector de multiplicadores no negativos para cada elemento de la disyunción es $u^t=(v^t,w_t)$, donde w_t corresponde a la última desigualdad en el elemento de la disyunción. El objeto es derivarun corte disyuntivo $bx \geq \beta$ que satisfaga

$$b \geq w_t a^t - v^t$$
$$\beta \leq w_t \alpha_t - v^t m$$

para toda t. Para una w_t dada (aún indeterminada), es razonable hacer los componentes de b tan pequeños como sea posible para obtener una restricción más ajustada. Así, deje que

$$b = \min_t \{u_t a^t\}, \tag{5.30}$$

donde el mínimo es tomado entre los componentes. Uno puede ahora hacer

$$v^t = u_t a^t - b, \qquad t\in T, \tag{5.31}$$

porque (5.30) implica $v^t \geq 0$. Para hacer el lado derecho del corte, tan ajustado como sea posible, haga

$$\beta = \min_{t\in T} \{u_t \alpha_t - v^t m\}. \tag{5.32}$$

Todavía falta escoger los valores para las w_t. La opción de Beaumont es equivalente a hacer $w_t = M_t$ cuando M_t se deriva de los límites de la variable como en (5.10) y $a^t \leq 0$. Entonces

$$w_t = \frac{1}{\alpha_t - a^t m}. \tag{5.33}$$

Este enfoque no puede aplicarse cuando el denominador es negativo, caso para el cual. Beaumont sugiere dejar

$$w_t = \frac{1}{\alpha_t - \min\{a^t, 0\} m}. \tag{5.34}$$

Teorema 5.5 (Beaumont) El corte dado por (5.30)-(5.33) define una faceta de (5.4) si $\alpha_t - a^t m > 0$ para toda $t\in T$.

El corte de Beaumont puede, por lo tanto, ser superior a un corte elemental de soporte. Esto se ilustra en la Fig. 5.5, donde el corte de Beaumont es el corte que define la faceta $2x_1 + x_2 \leq 7$.

Asumir que $\alpha_t - a^t m > 0$ es equivalente a asumir que el punto $x = m$ es infactible, en cuyo caso tiene sentido separar este punto del conjunto factible. Sin embargo, $x = m$ es frecuentemente factible, como en el ejemplo de la Fig. 3.3. Aquí $(w_1, w_2) = (-\frac{1}{4}, -1/5)$, y uno debe volver a (5.34) lo cual da el corte $3x_1 + 2x_2 \geq -2$ que no sirve para nada.

La dificultad subyacente es que el enfoque de Beaumont no tiene un mecanismo para detectar cuál esquina de la caja $0 \leq x \leq m$ debe cortarse del conjunto factible. Una esquina apropiada podría, en efecto, ser identificada usando un cambio de variable similar al discutido antes, digamos

$$\hat{x}_j = \begin{cases} m_j - x_j & cuando \quad sea \quad conveniente \\ x_j & en \quad otro \quad caso \end{cases}$$

Una transformación "conveniente" debería ser aquella que haga $\hat{\alpha}_t - \hat{a}^t m > 0$ para tantos elementos t de la disyunción como sea posible, donde $\hat{a}^t$ y $\hat{\alpha}_t$ están dados por (5.17) y (5.18). Esto plantea un problema de programación entera que puede ser resuelto heurísticamente. Sin embargo, debido a la cantidad de cálculos involucrados, esta opción no será aplicada posteriormente.

5.7 Cortes de Separación Optima

Una forma de identificar un punto apropiado para ser cortado por un corte disyuntivo es simplemente tratar de cortar la solución de la relajación actual. Esto es también un mecanismo para utilizar información acerca de la función objetivo, ya que la solución actual fué obtenida minimizando la función objetivo. Afortunadamente es directo el formular un pequeño problema de programación lineal cuya solución identifique un corte de separación si y sólo si éste existe. Entonces, si no se encuentra ningún corte, se sabe que la solución actual cae dentro del envolvente convexo del conjunto factible, y se hace necesaria la ramificación para obtener una solución factible -a menos que la solución actual ya sea infactible. El corte es óptimo en el sentido de que es escogido para maximizar la cantidad porla cual la solución actual lo viola.

Otra atracción de los cortes de separación óptima es que pueden ser generados para el caso en el cual existen varias desigualdades en cada elemento de la disyunción. Sin embargo, una restricción lógica $g_t(y,h)$ diferente a una disyunción, debe ser puesta en forma disyuntiva.

Suponga que la solución x' de la relajación actual va a ser separada del conjunto factible de la restricción disyuntiva (5.1). Cualesquiera límites superiores $x \leq m$ deberían ser incorporados dentro de cada elemento de la disyunción (5.1). Ya que cualquier corte

disyuntivo se defina por una selección de multiplicadores u^t, se puede formular un modelo de programación lineal para encontrar un conjunto de valores u^t que definan un corte $bx \geq \beta$ que es violado máximamente por x'. Dicho modelo es,

$$\begin{aligned} & \max \beta - bx' \\ \text{s.a} \quad & \beta \leq u^t a^t, \qquad t \in T \\ & b \geq u^t A^t, \qquad t \in T \\ & -e \leq x \leq e \\ & u^t \geq 0,\ t \in T \ \text{ y } \ \beta, b \text{ irrestrictas.} \end{aligned} \tag{5.35}$$

Note que las variables en el modelo son β, b y u. La función objetivo mide la cantidad por la cual x' viola el corte. Si el valor de la función objetivo es cero, no existe corte de separación. La restricción $-e \leq x \leq e$, donde e es un vector de unos, asegura que la solución esté acotada. No resulta en pérdida de generalidad porque un corte óptimo siempre puede ser reescalado para satisfacer la restricción.

El modelo (5.35) tiene un dual interesante.

$$\begin{aligned} & \min (s + t)e \\ \text{s.a} \quad & x' - \textstyle\sum_{t \in T} x^t = s - t \ (b) \\ & A^t x^t \geq a^t y_t, \quad t \in T \quad (u^t) \\ & \textstyle\sum_{t \in T} y_t = 1 \qquad (\beta) \\ & s, t, x^t, y_t \geq 0,\ t \in T \end{aligned} \tag{5.36}$$

Si $s - t$ se fija a cero y x' es una variable, el conjunto de restricciones es la representación de Balas del envolvente convexo para la disyunción (5.1) [4]. Esto es, cuando $s - t = 0$. la proyección del conjunto factible de (5.36) sobre el espacio-x' es el envolvente convexo del conjunto factible de (5.1). (Esto está relacionado al hecho, observado por Williams [69], de que el dual del dual de un problema de programación disyuntiva es la representación del envolvente convexo del problema.) El problema (5.36) por lo tanto. busca un punto $\sum_{t \in T} x^t$ en el envolvente convexo que sea el más cercano a x'. medido en una distancia rectilínea.

Un corte de separación óptima puede ser superior a un corte elemental de soporte. Considere el ejemplo de la Fig. 5.5 , el cual se convierte en

$$\begin{pmatrix} -3x_1 + x_2 \geq -3 \\ -x_1 \geq -3 \\ -x_2 \geq -3 \end{pmatrix} \vee \begin{pmatrix} -x_2 \geq -1 \\ -x_1 \geq -3 \\ -x_2 \geq -3 \end{pmatrix}$$

La solución de (5.37) es $\beta = -7/2$, $b = (-1, -\frac{1}{2})$, $u^1 = (1/3,0,5/6)$, $u^2 = (\frac{1}{2},1,0)$ lo cual produce el corte $2x_1 + x_2 \leq 7$ que define una faceta del envolvente convexo. Sin embargo, el corte de separación óptima no necesita definir una faceta del envolvente convexo. Si el envolvente convexo de la disyunción es la caja definida por $0 \leq x_j \leq 1$ para $j = 1,2$, el corte de separación óptima para $x' = (2,2)$ es $x_1 + x_2 \leq 2$.

CAPITULO 4: PROCESAMIENTO LOGICO

El aspecto lógico de la programación lineal mixta-lógica gira primariamente alrededor de dos cuestiones: ¿Qué repertorio de fórmulas lógicas están disponibles para expresar las restricciones discretas?, ¿Qué algoritmos de procesamiento lógico pueden ser usados para derivar implicaciones de un conjunto de fórmulas lógicas? Después de una breve discusión de estas dos preguntas, las secciones de este capítulo seran dedicadas a varias clases de fórmulas lógicas de creciente generalidad. También se presentan los algoritmos utilizados para cada una, siguiendo lo expuesto en el artículo de Hooker, Osorio de *Programación Lineal Mixta-Lógica.*

Los cortes lógicos pueden también ser desarrollados manualmente, basados completamente en la estructura del problema. Esto es discutido en la sección 7.

1. Fórmulas Lógicas

En teoría, una fórmula lógica $g_i(y,h)$ puede representar cualquier función de (y,h) que tome varios valores verdadero y falso. Pero hay ciertas formas sintácticas que han probado ser útiles para expresar las restricciones. Las más básicas serán discutidas aquí.

Cláusulas lógicas. Una cláusula lógica es una disyunción de *literales*, que son proposiciones atómicas y_j o sus negaciones $\neg y_j$. Entonces la expresión $y_1 \vee \neg y_2 \vee y_3$ es una cláusula, donde $\vee$ es un "o inclusivo" lógico. En teoría, las conjunciones de cláusulas lógicas puden expresar cualquier función sólo de y (i.e., cualquier función booleana), pero puede ser conveniente usar otras formas también. Las implicaciones (p.ej. $y_1 \rightarrow y_2$) y las equivalencias ($y_1 \equiv y_2$) son fácilmente definidas en términos de cláusulas.

Cláusulas extendidas. Estas tienen la forma, "al menos k de $L_1, \ldots, L_p$ son verdaderas", donde las L_j son literales. Ellas, también expresan funciones booleanas, pero frecuentemente son más convenientes que las cláusulas porque tienen un aspecto *quasi*-aritmético además de su aspecto lógico.

Restricciones tipo "maleta". La familiar restricción de "la maleta" 0-1 $by \geq \beta$, donde cada $y_j \in \{0,1\}$, puede también ser mirada como una fórmula lógica que es verdadera cuando la suma sobre las b_j para las cuales y_j es verdadera es al menos β. Las funciones booleanas de esta forma se llaman funciones de *umbral* y son estudiadas en la literatura de ingeniería eléctrica [58]. Aunque son difíciles de procesar

lógicamente pueden ser utilizadas para generar cortes lógicos en la forma de cláusulas y cláusulas extendidas, las cuales son fácilmente manipuladas.

Cláusulas multivalentes. Estas generalizan las cláusulas para acomodar variables multivalentes y son adecuadas para expresar cualquier función bivalente de (y,h). Son disyunciones de términos que tienen la forma $h_j \in H$, donde H es un subconjunto del dominio de h_j.

El predicado todos-diferentes. Como en el caso de las cláusulas bivalentes, es útil proporcionar cláusulas multivalentes con otros tipos de sintáxis. Una fórmula particularmente útil establece que en un conjunto de variables h_j, todos los elementos del conjunto tienen diferentes valores.

En este trabajo se trataron las variables proposicionales y_j de manera diferente que las variables discretas multivalentes h_j. Mientras que las y_j representan proposiciones que son verdaderas o falsas, las h_j no son proposiciones. Para obtener un proposición, se debe establecer algo acerca de las h_j, como que $h_j \in H$. Otra distinción es que las y_j corresponden a conjuntos de desigualdades lineales $A^j x \geq d^j$ y las h_j no. En teoría, uno puede forzar un conjunto de restricciones para cada posible valor de h_j, pero el mismo efecto se consigue escribiendo $y_j \equiv (h_j = v)$ para cada valor v y asociando restricciones con las y_j. Las y_j y las h_j se tratan aquí de manera diferente, principalmente porque existen técnicas conocidas para escribir relajaciones lineales de varias clases de fórmulas que involucren sólo variables y_j, y no se puede decir lo mismo de las variables h_j.

Por corrección lógica se debe notar que las y_j son "bivalentes" en un sentido diferente al que las h_j son "multivalentes"; mientras que las variables y_j pueden tomar dos valores lógicos, como verdadero o falso, las variables h_j pueden tomar varios valores discretos que no son verdaderos. Todas las fórmulas lógicas discutidas aquí, que incluyen $h_j \in H$, pertenecen a la lógica bivalente.

2. Algoritmos de Procesamiento lógico

El objetivo del procesamiento lógico es el de extraer información que es implícita en las fórmulas lógicas $g_i(y,h)$. Es esencialmente un proceso de inferencia que deriva cortes lógicos, o implicaciones, de las fórmulas. Puede ser útil de tres maneras: *a*) los cortes lógicos pueden dar origen a cortes elementales o a otras desigualdades que pueden reforzar el problema lineal relajado; *b*) algunos cortes lógicos peuden fijar variables y_j o h_j, reduciendo así el árbol de búsqueda; y *c*) el procesamiento lógico puede detectar inconsistencia, reduciendo nuevamente la búsqueda. La solución del problema lineal relajado de un modelo de programación 0-1 puede, algunas veces, realizar (*b*) o (*c*), pero sólo algunas veces, y aún así, mucho más lentamente que un algoritmo de inferencia lógica apropiado.

Como se mencionó antes, un gran número de algoritmos de procesamiento lógico se han desarrollados por las comunidades de investigación asociadas con la inferencia lógica, la satisfacción de restricciones, la programación de restricciones y la programación lógica. Por lo mismo, la discusión acá se limitará a dos tipos básicos de algoritmos que pueden formar la base lógica de un programa computacional de programación lineal mixta-lógica. Uno es un algoritmo *completo* que deriva todas las inferencias posibles, y otro que es un algoritmo mucho más rápido y es *incompleto*.

El algoritmo incompleto es una simple técnica de propagación que toma la forma de *resolución unitaria* ("encadenamiento hacia adelante") en el caso de las cláusulas lógicas. Es probablemente adecuado para la mayoría de las aplicaciones, pero cuando se requiere de un algoritmo de inferencia más poderoso, se puede usar el algoritmo de resolución. Es un método clásico de inferencia para cláusulas lógicas y puede ser generalizada para cláusulas extendidas y multivalentes. La resolución puede ser un poco lenta y es inadecuada para problemas con un gran número de variables proposicionales. Existen muchas aplicaciones, sin embargo, en las cuales, el número de variables discretas es relativamente pequeño, comparado con la parte continua del problehma. En esos casos, puede valer la pena extraer tanta información como sea posible de las fórmulas lógicas con el objeto de evitar resolver problemas lineales grandes.

3. Cláusulas Lógicas

En principio, una fórmula lógica que contiene sólo variables bivalentes y_j siempre puede ser escrita como una conjunción de cláusulas, *i.e.*, en forma normal conjuntiva (CNF). Por ejemplo, la fórmula $(y_1 \wedge y_2) \vee y_3$, donde $\wedge$ significa "y", y puede ser escrita como $(y_1 \vee y_3) \wedge (y_2 \vee y_3)$. También la implicación $y_1 \rightarrow y_2$ (o $y_1 \supset y_2$) puede ser escrita como $\neg y_1 \vee y_2$, y la equivalencia $y_1 \equiv y_2$ se puede reescribir como $(\neg y_1 \vee y_2) \wedge (y_1 \vee \neg y_2)$.

La forma conjuntiva normal (CNF) es completamente expresiva porque cualquier fórmula en variables bivalentes puede ser mirada como una función booleana (verdadera-falsa) $f(y) = f(y_1, \ldots, y_m)$. Si $y = v^1, \ldots v^N$ son los valores de y que hacen $f(y)$ falsa, entonces la siguiente fórmula es equivalente a $f(y)$,

$$\bigwedge_{i=1}^{N} \bigvee_{j=1}^{m} y_j(v^i_j),$$

donde $y_j(v^i_j)$ es $\neg y_j$ si v^i_j = verdadera y es y_j si v^i_j= falsa.

Esta conversión a la forma conjuntiva normal requiere tiempo y espacio exponencial en el peor de los casos. Sin embargo, una fórmula que involucra las conectivas $\neg, \vee, \wedge, \rightarrow, \equiv$ (y otras conectivas con tiempo-lineal de transformación a CNF) puede ser convertida a la forma conjuntiva normal en tiempo lineal agregando nuevas variables. El algoritmo va como sigue. Si una fórmula dada F tiene la forma A $\wedge$ B, donde A y B son subfórmulas, entonces se aplica el algoritmo a A y a B separadamente, y se juntan los resultados. Si F

tiene la forma A → B entonces hay que reescribirla como ¬ A ∨ B y aplicar el algoritmo a la fórmula resultantes, y similarmente para A≡B. Si F tiene la forma A ∨ B, entonces se escribe como $(y_{m+1} \equiv A) \wedge (y_{m+2} \equiv B) \wedge (y_{m+1} \vee y_{m+2})$, donde y_{m+1}, y_{m+2} son variables que no ocurren en F, y se aplica el algoritmo a la fórmula resultantes (ver [71] para refinamientos). El algoritmo termina cuando se alcanza la forma conjuntiva normal.

En la práctica, un programa computacional de programación lineal mixta-entera aceptaría varias conectivas que se puedan convertir rápidamente a la forma conjuntiva normal y haría la conversión. Normalmente sería innecesario introducir nuevas variables, porque generalmente,las fórmulas son lo suficientemente cortas como para que un equivalente en CNF sin ellas sea también una fórmula corta.

Un algoritmo simple de *resolución* [47, 48, 54] deriva todas las implicaciones de un conjunto de cláusulas. Deje que la cláusula C_1 *absorba* la cláusula C_2 cuando todas las literales de C_1 ocurran en C_2; C_1 implica C_2 si y sólo si C_1 absorbe C_2. Dos cláusulas tienen un *resolvente* (único) cuando exactamente una variable y_j ocurre positivamente en una y negativamente en la otra. El resolvente es una disyunción de todas las literales que ocurren en cualquier cláusula excepto y_j y $\neg y_j$. Por ejemplo, $y_2 \vee \neg y_3$ es el resolvente de $y_1 \vee y_2$ y $y_2 \vee y_3$. Dado un conjunto S de cláusulas, el algoritmo de resolución escoge un par de cláusulas en S que tengan un resolvente que es absorbido por ninguna cláusula en S, y agregan el resolvente a S. Se repite hasta que no existe otro par, lo cual ocurre en un número grande y finito de iteraciones.

Teorema 3.1 (Quine [47,48]) Un conjunto de cláusulas S implica la cláusula C si y sólo si el algoritmo de resolución aplicado a S genera una cláusula que absorba C. En particular, S no es satisfacible si y sólo si la resolución genera una cláusula vacía.

Este teorema seguirá de un resultado más general que se probará en la sección 6 de este capítulo.

Las relajaciones lineales pueden ser generadas para las cláusulas derivadas por resolución, si se desea. Un cláusula es simplemente una disyunción. Así, si la cláusula contiene todas las literales positivas, y cada una de sus variables corresponden a un conjunto de restricciones lineales, un corte elemental o de otro tipo pueden ser generado como se discute en la sección 5. Si una cláusula derivada es una cláusula unitaria, *i.e.*, contiene una literal simple y_j o $\neg y_j$, entonces y_j puede ser fijada de acuerdo al valor que tenga en ella.

La resolución no sólo tiene una complejidad exponencial en el peor de los casos [24] pero puede ser lenta en la práctica [27]. Un algoritmo de inferencia mucho más rápido que sacrifica el hecho de ser un algoritmo completo, es el algoritmo de *resolución unitaria*. Es el mismo que el de resolución total excepto que uno de los padres de un resolvente es siempre una cláusula unitaria. El hecho de que no sea completo puede ser visto en el ejemplo,

$$y_1 \vee \quad y_2 \vee \quad y_3$$

$$y_1 \vee \neg y_2 \vee \quad y_3$$
$$y_1 \vee \quad y_2 \vee \neg y_3$$
$$y_1 \vee \neg y_2 \vee \neg y_3$$

La resolución fija y_1 a verdadero, pero la resolución unitaria no hace nada porque no existen cláusulas unitarias para empezar. Sin embargo, la resolución unitaria es eficiente, ya que corre en un tiempo $O(nL)$, si existen n variables y L literales, y tiende a ser muy rápida en la práctica. Un algoritmo preciso que se adapta a un caso más general de cláusulas extendidas aparece en la Fig. 4.1.

La resolución unitaria es un algoritmo de inferencia completo para ciertas clases de cláusulas, como las cláusulas de Horn, la cláusulas renombrables de Horn, etc. [13, 14, 15, 56, 62]. Ninguna propiedad estructural conocida para un conjunto de cláusulas es una condición necesaria y suficiente para que el algoritmo de resolución unitaria sea completo.

La resolución unitaria tiene el mismo poder inferencial que la programación lineal, en el siguiente sentido. Suponga que las cláusulas de S se escriben como un sistema $Ay \geq a$ de desigualdades 0-1 en el estilo usual; *i.e.*, una cláusula $\vee_{j\in J} L_j$ es escrita como $\sum_{j\in J} y_j(L_j) \geq 1$, donde $y_j(L_j)$ es y_j si $L_j = y_j$ y es $1 - y_j$ si $L_j = \neg y_j$.

Teorema 3.2 (**Blair, Jeroslow, Lowe** [9]) La resolución unitaria encuentra una contradicción en el conjunto de cláusulas S si y sólo si la relajación lineal del sistema correspondiente $Ay \geq a$ de desigualdades 0-1 es infactible.

$Ay \geq a$ es infactible cuando la resolución unitaria encuentra una contradicción porque la resolución unitaria (a diferencia de la resolución en general) simplemente agrega las representaciones de desigualdad de las cláusulas. Así, el derivar una cláusula vacía es equivalente a obtener $0 \geq 1$ de una combinación lineal no negativa de $Ay \geq a$. Por el contrario, si la resolución unitaria no detecta contradicción, entonces las desigualdades que representan las cláusulas remanentes pueden ser satisfechas haciendo cada $y_j = ½$.

4. Cláusulas Extendidas

Las cláusulas extendidas parecen ser un compromiso particularmente útil entre la aritmética y la lógica porque expresan las nociones de "al menos" y "a lo más" pero pueden ser procesadas eficientemente como fórmulas lógicas. De hecho, el programa basado en las restricciones para problemas de optimización 0-1 de Barth [5] trabaja con desigualdades 0-1, sólo después de haberlas convertido en cláusulas extendidas.

Una cláusula extendida de grado k puede ser escrita como

$$\sum_{j\in J} L_j \geq k,$$

donde cada L_j es una literal. Aquí la suma no es una suma aritmética, pues simplemente cuenta el número de literales que son verdaderas. Las cláusulas ordinarias tienen grado 1. Para decir que al menos k son verdaderas, se puede escribir

$$\sum_{j \in J} \neg L_j \geq |J| - k,$$

y se pueden usar dos cláusulas extendidas pra decir que exactamente k son verdaderas.

Un algoritmo de inferencia completo ("resolución generalizada") para cláusulas extendidas es presentado en [27,30] y es refinado por Barth en [5]. Usa resolución así como una *sumatoria diagonal*, donde la suma se define como sigue. Una cláusula extendida $\sum_{j \in J} L_j \geq k + 1$ es la suma diagonal del conjunto de desigualdaes extendidas $\{\sum_{j \in J_i} L_j \geq k \mid i \in J\}$ si $J = \bigcup_{i \in J} J_i$ pero para cada $i \in J$, $i \notin J_i$. El algoritmo de [27] se aplica a un conjunto S de cláusulas extendidas como sigue. Si existen dos cláusulas C_1, C_2 de grado 1 con un resolvente C que sea implicado por una cláusula no extendida en S, tal que C_1 sea implicado por una cláusula extendida en S y similarmente para C_2, entonces agregue C a S. Si existe un conjunto E de cláusulas extendidas con una suma diagonal D que sea implicada por una cláusula no extendida en S, tal que cada cláusula en E es implicada por alguna cláusula en S, entonces agregue D a S. El algoritmo continúa hasta que no existen más cláusulas que puedan ser agregadas a S.

Teorema 4.1 (Hooker [27],[30]) Un conjunto S de cláusulas extendidas implica la cláusula C si y sólo si el algoritmo de resolución generalizada, aplicado a S genera una cláusula que implique C.

La implementación del algoritmo requiere el reconocimiento de cuando una cláusula extendida implica otra. $\sum_{j \in J1} L_{1j} \geq k_1$ implica $\sum_{j \in J2} L_{2j} \geq k_2$ si y sólo si

$$|J_1| - |\{j \in J_1 \cap J_2 \mid L_{1j} = L_{2j}\}| \leq k_1 - k_2.$$

Cuando todas las literales de una cláusula extendida derivada son positivos y corresponden a conjuntos de desigualdades, se puede formular una relajación lineal usando uno delosmétodos descritos en la sección 5. Un algoritmo de resoluci|n unitaria para cláusulas extendidas aparece en la Fig. 4.1

La programación lineal es un algoritmo de inferencia más fuerte que la resolución lineal para cláusulas extendidas. Por ejemplo, el problema lineal detectala infactibilidad de las siguientes desigualdades, pero la resolución unitaria no puede hacer nada con las cláusulas extendidas correspondientes.

$$y_1 + y_2 + y_3 \geq 2$$
$$(1 - y_1) + (1 - y_2) + (1 - y_3) \geq 2$$

Deje que S sea un conjunto $\{\sum_{j\in J_i} L_{ij} \geq k_i \mid i \in I\}$ de cláusulas extendidas, donde cada L_{ij} es y_j o $\neg y_j$.
Deje que U sea una pila de cláusulas unitarias, inicialmente vacía.
Para cada $i \in I$ con $|J_i| = k_i$:
    Para cada $j \in J_i$ agregue L_{ij} a U.
    Deje $J_i = \varnothing$.
Mientras U no esté vacía
    Remueva L'_t de U.
    Para cada $i \in I$ con $t \in J_i$:
        Si $L_{it} = L'_t$ **entonces**
            Deje $k_i = k_i - 1$, $J_i = J_i \backslash \{t\}$.
        si no
            Si $k_i = |J_i|$ **entonces pare**; S es insatisfacible.
        si no
            Si $k_i = |J_i| + 1$ **entonces**
                Para cada $j \in J_i \backslash \{t\}$ agregue L_{ij} a U.
                Deje $J_i = \varnothing$.
            si no
                Deje $J_i = J_i \backslash \{t\}$.

Fig. 4.1 *Un algoritmo de resolución unitaria para cláusulas extendidas*

Ningún algoritmo conocido de inferencia tiene exactamente el mismo efecto de un problema lineal sobre las cláusulas extendidas, hasta que se ven los algoritmos de programación lineal como algoritmos de inferencia. La resolución generalizada es, por lo tanto, más fuerte que la programación lineal.

5. Restricciones tipo "maleta"

Un algoritmo de inferencia completo para restricciones tipo "maleta" aparece en [30], y se puede derivar fácilmente un algoritmo análogo de resolución unitaria para estas restricciones. Pero quizás pueden usarse mejor como una fuente de cortes lógicos que son procesados más fácilmente, como las cláusulas y las cláusulas extendidas. Las cláusulas implicadas, por ejemplo, son idénticas a las bien conocidas "desigualdades de cubrimiento" para la restricción tipo "maleta", y su derivación es directa (*p. ej.* [23]).

Puede ser más efectivo, sin embargo, inferir desigualdades extendidas. Mientras que es difícil derivar todas las desigualdades extendidas que son implicadas por una restricción, es fácil derival todos los *cortes contiguos*. Considere una desigualdad 0-1, $dy \geq \delta$ para la que se asume, sin pérdida de generalidad, que $d_1 \geq d_2 \geq \ldots \geq d_n > 0$; si $d_j < 0$, cambie su signo y agregue d_j a δ. Un corte coniguo para $dx \geq \delta$ tiene la forma,

$$\sum_{j=t}^{t+w+k-1} y_j \geq k, \qquad (5.1)$$

Deje $k = 1$, $s = \sum_{j=1}^{n} d_j$, $k_{\text{último}} = 0$.
Para $j = 1, \ldots, n$:
  Deje $s = s - d_j$.
  Si $s < \delta$ **entonces**
    Mientras $s + d_k < \delta$:
      Deje $s = s + d_k$,
      Deje $k = k + 1$.
    Si $k > k_{\text{último}}$ **entonces**
      Genere el corte $y_1 + \ldots + y_j \geq k$.
      Deje $k_{\text{último}} = k$.

Fig. 4.2 *Un algoritmo para generar todos los cortes-1 de una restricción tipo "maleta"* $d_j y_j \geq \delta$ *en la cual* $d_1 \geq d_2 \geq \ldots \geq d_n > 0$.

donde k es el grado del corte y $w < n$ su "debilidad") ($w = 0$ indica un corte que fija todas sus variables). En particular (5.1) es un corte-t porque el primer término es y_t. (5.1) es válido si y sólo si

$$\sum_{j=1}^{t+k-1} d_j + \sum_{t+w+k}^{n} d_j < \delta.$$

Más aún,

Lemma 5.1 (**Hooker** [35]) (5.1) es válido, si y sólo si,

$$d_1 + \ldots + d_{t+k-2} + d_{t+w+k} + \ldots + d_n < \delta \tag{5.2}$$

Una propiedad elemental de los cortes que será útil es,

Lemma 5.2 (**Hooker** [35]) Si (5.1) es válido, entonces el siguiente corte es válido para $k' \leq k$.

$$y_t + \ldots + y_{t+w+k'-1} \geq k'. \tag{5.3}$$

Más aún, si (5.1) es inválida, entonces (5.3) es inválida pra toda $k' \geq k$.

El siguiente hecho es la clave del resultado,

Lemma 5.3 (**Hooker** [35]). El corte

$$y_1 + \ldots + y_{w+k} \geq k \tag{5.4}$$

es válido si y sólo si el corte

$$y_t + \ldots + y_{w+k} \geq k - t + 1 \tag{5.5}$$

es válido.

Prueba. Debido al Lemma (5.1), (5.4) es válido si y sólo si

$$d_1 + \ldots + d_{k-1} + d_{k+w+1} + \ldots + d_n < \delta,$$

y lo mismo es cierto de (5.5). □

Ahora, el resultado principal

Teorema 5.1 (Hooker [35]) Cada corte-*t* de debilidad *w* para $dx \geq \delta$ es implicado por un corte-*l* de debilidad *w*.

Prueba. Deje que k_l sea la cardinalidad máxima de cortes-*l* válidos de debilidad *w*. Entonces

$$y_1 + \ldots + y_{w+k1} \geq k_1 \qquad (5.6)$$

es válido y

$$y_1 + \ldots + y_{w+k1+1} \geq k_1 + 1 \qquad (5.7)$$

es inválido. Basta con mostrar que el siguiente corte-t es o inválido o redundante.

$$y_t + \ldots + y_{t+w+kt-1} \geq k_t. \qquad (5.8)$$

Pero de (5.6) y del Teorema 5.1, sabemos que

$$y_t + \ldots + y_{w+k1} \geq k_1 - t + 1$$

es válido. Esto y el Lemma (5.2) implican que (5.8) es válido para $k_t \leq k_1 - t + 1$. En este caso (5.8) está dominado por (5.6) y por lo tanto es redundante. También de (5.7) y del Teorema 5.1, tenemos que

$$y_t + \ldots + y_{w+k1+1} \geq k_1 - t + 2$$

es inválido. Esto y el Lemma 2 implican que (5.8) es inválido para $k_t \geq k_1 - t + 2$.□

El poder de todos los cortes-*t* puede, por lo tanto, ser obtenido generando sólo cortes-*l*. El algoritmo de la Fig. 4.2, presentado por Hooker en [35], lo hace en un tiempo lineal. A manera de ejemplo, la restricción tipo "maleta"

$$13\,y_1 + 9\,y_2 + 8\,y_3 + 6\,y_4 + 5\,y_5 + 3\,y_6 \geq 30$$

da origen a los cortes-*l*,

$$y_1 + y_2 \geq 1$$
$$y_1 + y_2 + y_3 \geq 2$$
$$y_1 + y_2 + y_3 + y_4 + y_5 \geq 3.$$

El primero puede deshecharse si se desea, porque es redundante del segundo.

6. Cláusulas Multivalentes

Las cláusulas multivalentes proveen una sintáxis versátil y conveniente para expresar las fórmulas lógicas que involucran variables multivalentes. Una cláusula multivalente tiene la forma

$$\bigvee_{j=1}^{m} (h_j \in H_j), \tag{6.1}$$

donde cada H_j es un subconjunto del dominio D_j de h_j. Para simplicidad notacional, se asume que las variables bivalentes y_j se pueden mirar como variables h_j con dos valores que toman un valor de, se puede decir 1, cuando y_j es verdadero y 0 cuando y_j es falso. Si h_j está vacío, el término $(h_j \in H_j)$ puede ser omitido para (6.1), pero es conveniente suponer que (6.1) contiene un término para cada j.

Cualquier función verdadero-falsa $f(h) = f(h_1, \ldots, h_m)$ puede ser expresada como una conjunción de cláusulas multivalentes. En particular, $\neg(h_j \in H_j)$ puede ser escrita como $(h_j \in D_j \backslash H_j)$, y $(h_j \in H_j) \rightarrow (h_k \in H_k)$ puede ser escrita como $(h_j \in D_j \backslash H_j) \vee (h_k \in H_k)$. Una cláusula multivalente $\vee_j (h_j \in H_{1j})$ implica otra $\vee_j (h_j \in H_{2j})$ si y sólo si $H_{1j} \subset H_{2j}$ para cada j. Como ejemplo de cláusulas multivalentes, considere la siguiente fórmula del problema de la fiesta progresiva.

$$v_{ijt} \equiv (h_{it} = j)$$

Se puede expresar formalmente como dos cláusulas multivalentes,

$$(v_{ijt} \in \{0\}) \vee (h_{it} \in \{j\})$$
$$(v_{ijt} \in \{1\}) \vee (h_{it} \in D_j \setminus \{j\}).$$

La resolución puede ser generalizada para obtener un método completo de inferencia para cláusulas multivalentes. El algoritmo resultante se relaciona con el algoritmo de Cooper para obtener consistencia-k para un conjunto de restricciones [18]. Dado un conjunto de cláusulas multivalentes,

$$\bigvee_{j=1}^{n} (h_j \in H_{ij}), \quad i \in \mathrm{I} \tag{6.2}$$

el resolvente sobre h_k de estas cláusulas es

$$\left(h_k \in \bigcap_{i \in \mathrm{I}} H_{ik}\right) \vee \bigvee_{j \neq k} \left(h_j \in \bigcup_{i \in \mathrm{I}} H_{ij}\right)$$

La resolución ordinaria bivalente es un caso especial. Para aplicar el algoritmo de resolución a un conjunto S de cláusulas multivalentes, encuentre un subconjunto de S cuyo resolvente M no sea implicado por alguna cláusula en S, y agregue M a S. Continúe hasta que no existan más cláusulas que agregar a S.

Deje que S sea un conjunto $\{C_i \mid i \in I\}$ de cláusulas multivalentes, donde cada C_i tiene la forma $\vee^{m}_{j=1}(h_j \in H_{ij}) \vee \vee_{t \in Ti}$ *todos-diferentes*$(\{h_j | j \in J_t\})$
Deje n_i ser el número de términos $(h_j \in H_j)$ de C_i con H_j no vacío.
Deje U ser la pila de índices que representan los dominios activos;
inicialmente $U = \{1,\ldots,m\}$.
Deje A ser una lista de predicados *todos-diferentes* forzados, inicialmente vacía.
Para cada $i \in$ I:
Si $n_i = 0$ y $|T_i| = 1$ **entonces:**
Agregue el predicado a A y remueva i de I .
si no si $n_i = 1$ y $|T_i| = 0$ **entonces**
Deje H_{ij} ser no vacío.
Deje $D_j = D_j \cap H_{ij}$ y remueva i de I.
Mientras U no esté vacía:
Remueva k de U.
Si D_k está vacía **entonces pare**; S es insatisfacible.
Para toda $i \in$ I:
Si H_{ik} no está vacía entonces
Si $D_k \subset H_{ik}$ entonces remueva i de I.
si no
Deje $H_{ik} = H_{ik} \cap D_k$.
Si H_{ik} está vacía **entonces**
Deje $n_i = n_i - 1$.
Si $n_i = 1$ y $|T_i| = 0$ **entonces**
Deje H_{ij} ser no vacía y remueva i de I.
Si $D_j \not\subset H_{ij}$ **entonces**
Deje $D_j = D_j \cap H_{ij}$ y agregue j a U.
Si $n_i = 0$ y $|T_i| = 1$ **entonces**
Remueva i de I.
Agregue el predicado *todos-diferentes* en C_i a A.
Para cada predicado *todos-diferentes* haga *todos-diferentes*$(\{h_j | j \in J\})$ en A con $k \in K$:
Si $|D_k| = 1$ **entonces**
Para $j \in J \setminus \{k\}$:
Si $D_k \subset D_j$ **entonces Deje** $D_j = D_j \setminus D_k$ y agregue j a U.

Fig. 4.3 *Un algoritmo de resolución unitaria para cláusulas multivalentes*

El algoritmo de resolución multivalente es un algoritmo de inferencia completa para cláusulas multivalentes. La prueba del teorema usa la idea original de Quine para la resolución ordinaria.

Teorema 6.1 Un conjunto de cláusulas multivalentes implica una cláusula multivalente M si y sólo si el algoritmo de resolución multivalente aplicado a S genera una cláusula que implique M.

Prueba. La resolución multivalente deriva sólo implicaciones de S porque es claramente válido. Para probar lo contrario, deje que S' sea el resultado de aplicar el algoritmo a S. También defina la *longitud* de una cláusula (6.1) como $\Sigma_j |H_j|$. Suponga que el teorema es falso, y deje que (6.1) sea la cláusula más larga implicada por S pero por ninguna cláusula en S'.

Afirmación. Al menos una H_j en (6.1) está perdiendo por lo menos dos elementos: *i.e.*, $|D_j \backslash H_j| \geq 2$ para alguna j. Primero, es claro que no existe ninguna $H_j = D_j$, porque de otra manera (6.1) sería implicada por una (de hecho, todas) cláusula en S'. suponga, contrario a la afirmación que cada H_j está perdiendo exactamente un elemento, digamos v_j. Entonces $h = v = (v_1, \ldots, v_m)$ viola (6.1) y debe, por lo tanto violar alguna cláusula $\vee_j (h_j \in H'_j)$ en S', porque S' implica (6.1). Esto significa que cada $H'_j \subset D_j \backslash \{v_j\}$, así que $\vee_j (h_j \in H'_j)$ implica (6.1), contrario a la hipótesis. Esto prueba la afirmación.

Ahora suponga que v_k, v'_k están perdidas de H_k, y considere las siguientes cláusulas multivalentes

$$(h_k \in H_k \cup \{v_k\}) \vee \vee_{j \neq k} (h_j \in H_j), \quad (h_k \in H_k \cup \{v'_k\}) \vee \vee_{j \neq k} (h_j \in H_j), \qquad (6.3)$$

Ellas deben ser implicadas, respectivamente, por la cláusulas $M_1, M_2 \in S'$ porque son más largas que (6.1). Esto significa que el resolvente de M_1, M_2 sobre h_k implica (6.1). Así, por construcción del algoritmo de resolución, S' contiene una cláusula que implica (6.1), contrario a la hipótesis. □

La prueba del teorema muestra que basta, en principio, generar resolventes sólo de pares de cláusulas.

Un algoritmo de resolución para cláusulas multivalentes aparece en la Fig. 4.3. El algoritmo también acomoda los predicados *todos-diferentes*, los cuales pueden servir como disyunciones de cláusulas multivalentes a lo largo de términos laterales de la forma $h_j \in H_j$. Un ejemplo de este caso está en la siguiente restricción de la fiesta progresiva:

$$\delta_i \vee \delta_j \vee m_{ijt} \vee (h_{it} \neq h_{jt})$$

pero puede ser escrita formalmente como una cláusula multivalente como sigue.

$$(\delta_i \in \{1\}) \vee (\delta_j \in \{1\}) \vee (m_{ijt} \in \{1\}) \vee \textit{todos-diferentes}(h_{it}, h_{jt}).$$

7. Cortes Lógicos

Un conocimiento intuitivo del problema en estudio, puede sugerir cortes lógicos, ambos válidos y no válidos, aún cuando posteriormente no se pueda identificar fácilmente ningún corte poliédrico, de acuerdo a lo expuesto por Hooker, J.N., H. Yan, e I.

Grossmann en el artículo *Logic cuts for processing networks with fixed charges* [37], mismo que se seguirá prácticamente al pie de la letra, para presentar la siguiente discusión sobre cortes lógicos.

Un corte lógico para un modelo de programación lineal mixta-lógica ha sido caracterizado, por lo tanto como una implicación de las fórmulas lógicas en el modelo. Actualmente, cualquier fórmula lógica implicada por el conjunto de restricciones como un todo, es un corte lógico. Esto es, $g(y,h)$ es un corte lógico si es verdadero para cada (x,y,h) que satisface las restricciones. Por ejemplo, $\neg y_3$ es un corte lógico para el problema

$$\begin{aligned} & \min x_1 + x_2 \qquad (7.1) \\ \text{s.a} \quad & y_1 \vee y_2 \\ & y_1 \rightarrow (x_1 \geq 1) \\ & y_2 \rightarrow (x_2 \geq 1) \\ & y_3 \rightarrow (x_1 \leq 0) \end{aligned}$$

pero no es implicado por la fórmula $y_1 \vee y_2$.

Los cortes lógicos pueden ser definidos en un sentido aún más general que los permita ser no válidos. Deje (y,h) ser factible en el modelo original, si (x,y,h) es factible en el mismo modelo, para alguna x. Deje (y',h') dominar (y,h) si para cualquier (x',y',h') que es factible en el modelo, existe una (x,y,h) factible para la cual $cx \leq cx'$. Entonces $g(y,h)$ es un corte lógico si cualquier (y,h) factible que hace $g(y,h)$ falso es dominado por una (y',h') factible que hace $g(y',h')$ verdadero. El corte $g(y,h)$ puede ser agregado al modelo sin cambiar la solución óptima, pero puede excluir soluciones factibles.

Por ejemplo, las fórmulas y_1 y $\neg y_2$ son cortes lógicos (no válidos) para (7.1), porque cualquier $(0,y_2,y_3)$ factible es dominada por $(1,y_2,y_3)$ y cualquier $(y_1,1,y_3)$ factible es dominada por $(1,0,y_3)$. Son cortes lógicos no válidos porque excluyen los puntos factibles (0,1,0), (1,1,0).

7.1 Ilustración

Para ilustrar la idea de un corte lógico, consideramos un problema muy simple de optimización sobre una red de la Fig. 7.4.

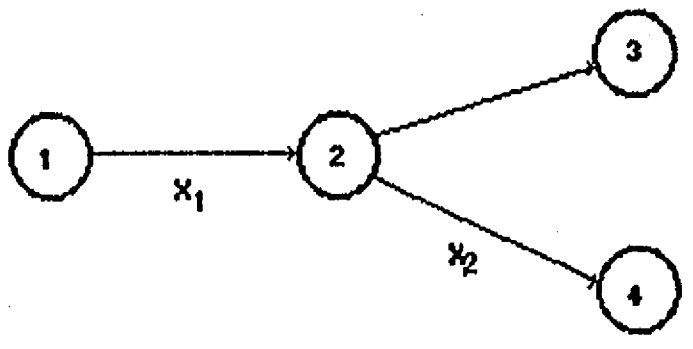


Fig. 7.4 *Ilustración de un corte lógico*

Un flujo con un valor superior de M está disponible en el nodo 1, los nodos 3 y 4 son descendientes, y el nodo 3 conserva el flujo. Los flujos en los arcos (1,2) y (2,4) son x_1 y x_2, y generan ganancias c_1x_1 y c_2x_2, respectivamente, donde c_j es cualquier número real. Sin embargo, los arcos (1,2) y (2,4) deben ser construidos a un costo $d_1>0$ y $d_2>0$, respectivamente, antes de que ellos puedan acarrear flujo. Esto sugiere el siguiente modelo de optimización lineal mixta-entera,

$$\begin{aligned} \max \quad & c_1 x_1 + c_2 x_2 - d_1 y_1 - d_2 y_2 \qquad (7.2) \\ \text{s.a} \quad & x_2 \le x_1 \\ & x_j \le M y_j \qquad j = 1,2 \\ & x_j \ge 0 \qquad y_j \in \{0,1\} \end{aligned}$$

Obviamente, no tiene sentido construir un arco que no acarree flujo. Aunque nada en el conjunto de reglas prohibe las soluciones con $(x_j, y_j) = (0,1)$. Una forma de excluir dichas soluciones es la de imponer restricciones de la forma $x_j \ge \varepsilon y_j$. Dichas restricciones requieren que cualquier flujo positivo x_j sea al menos tan grande como ε, pero esto es generalmente inocuo, para una ε suficientemente pequeña.

Se desea, por lo tanto, encontrar restricciones lógicas que puedan ser aplicadas simbólicamente. Esto no es posible con restricciones de la forma $x_j \ge \varepsilon y_j$, porque ellas contienen variables continuas. Se verá que las restricciones lógicas pueden eliminar soluciones que las restricciones de la forma $x_j \ge \varepsilon y_j$ no.

Por lo tanto, se buscarán restricciones que involucren sólo variables y_j, 0-1. Existen dichas restricciones que ponen fuera de juego, al menos algunas soluciones espurias de la forma $(x_j, y_j) = (0,1)$. Para ver esto, note que la restricción $x_1 \le My_1$ fuerza x_1 a cero cuando $y_1 = 0$. Pero $x_1 = 0$ implica $x_2 = 0$, en cuyo caso se puede suponer $y_1 = 0$. Se concluye que $y_1 = 0$ fuerza $y_2 = 0$. Esta es una relación lógica entre dos y_j:

$$y_1 . y_2 \ge 0 \qquad (7.3)$$

Si se considera y_j como una proosición que es verdadera cuando $y_j = 1$ en (7.2) y falsa cuando $y_j = 0$, (7.3) puede ser también escrita como una fórmula lógica:

$$y_1 \vee \neg y_2, \qquad (7.4)$$

Las restricciones (7.3)y (7.4) son formas alternas de formular el mismo *corte lógico*. No excluyen todas las soluciones en las cuales $(x_2,y_2) = (0,1)$, pero dejan afuera $(x_2,y_2) = (0,1)$ cuando $y_1 = 0$.

También se puede notar que el corte lógico de arriba tiene el efecto de forzar y_1 a un valor de uno si y_2 es fijado a uno. Si tal corte no estuviera presente, entonces, dependiendo del valor de M, y_1 puede tomar un valor fraccional cuando y_2 se fija a uno.

Si bien, un preprocesador puede generar el corte (7.3) como una implicación $y_1 \Rightarrow y_2$ en este problema simple, ningún preprocesador puede generar todos los cortes lógicos que se pueden identificar para una red de procesos.

Puede parecer que ya que una solución $(x_2,y_2) = (0,1)$ nunca es óptima en la relajación lineal del problema, dicha solución nunca aparecerá en un árbol de ramificación y acotamiento, de tal manera que no se necesita ningún corte para removerla. Sin embargo, dichas soluciones pueden ocurrir, y esos cortes lógicos, reducen sustancialmente el tamaño del árbol de búsqueda.

7.2 Definición

Se usarán los conceptos de epígrafe y su proyección para proveer una definición de un corte lógico. Considerando un problema de programación lineal mixto-entero,

$$\begin{array}{lll} \max & cx - dy & \\ \text{s.a} & Ax + By \le a & \\ & x_j \ge 0, & j = 1,\ldots,n, \\ & y_j \in \{0,1\}, & j = 1,\ldots,p, \end{array} \qquad (7.5)$$

donde A es m x n, B es m x p, y cada $d_j \ge 0$. (Si $d_j < 0$, reemplace y_j con $1 - y_j$.) La función objetivo puede ser vista como una función definida sobre el dominio D descrito por las restricciones. El grafo G de esta función, de acuerdo a la definición (7.5), es el conjunto $\{(x,y,z) \mid (x,y) \in D, z = cx - dy\}$. El epígrafe E es todo lo que se encuentre "*por debajo*" del grafo, definido como $\{(x,y,z) \mid (x,y) \in D, z \le cx\text{-}dy\}$. El valor óptimo del problema de optimización en (7.5), puede ser escrito como $\max \{z \mid (x,y,z) \in E\}$.

El grafo de (7.2) es la unión de los siguientes subconjuntos:

$$\begin{array}{l} \{(0,0,0,0,0)\} \\ \{(0,0,0,1,-d_2)\} \\ \{(x_1,0,1,0,c_1x_1 - d_1) \mid 0 \le x_1 \le M\} \\ \{(x_1,x_2,1,1,c_1x_1 + c_2x_2 - d_1 - d_2) \mid 0 \le x_2 \le x_1 \le M\} \end{array} \qquad (7.6)$$

El grafo proyectado G_p es la proyección de G sobre el espacio de variables continuas, de tal manera que $G_p = \{(x,z) \mid (x,y,z) \in G\}$. El epígrafe proyectado es $E_p = \{(x,z) \mid (x,y,z) \in E\}$. Claramente, el valor óptimo de (7.5) es $\max\{z \mid (x,z) \in E_p\}$. Esto significa que se pueden remover puntos del epígrafe E sin cambiar el valor de la solución óptima, dado que la proyeccion del epígrafe E_p no cambia.

La gráfica proyectada para (7.2) es la unión de los siguientes conjuntos.

$$\begin{aligned}&\{(0,0,0)\}\\&\{(0,0,-d_2)\}\\&\{(x_1,0,c_1x_1 - d_1)|0 \le x_1 \le M\}\\&\{(x_1,x_2,c_1x_1 + c_2x_2 - d_1 - d_2)|0 \le x_2 \le x_1 \le M\}\end{aligned} \tag{7.7}$$

El primer conjunto está mostrado en la Fig. 7.5 por el punto $y = (0,0)$, el segundo por el punto $y = (0,1)$, el tercero por el segmento lineal $y = (1,0)$, y el cuarto por el conjunto triangular $y = (1,1)$. El epígrafe proyectado es la unión de estos conjuntos con todos los puntos abajo de ellos.

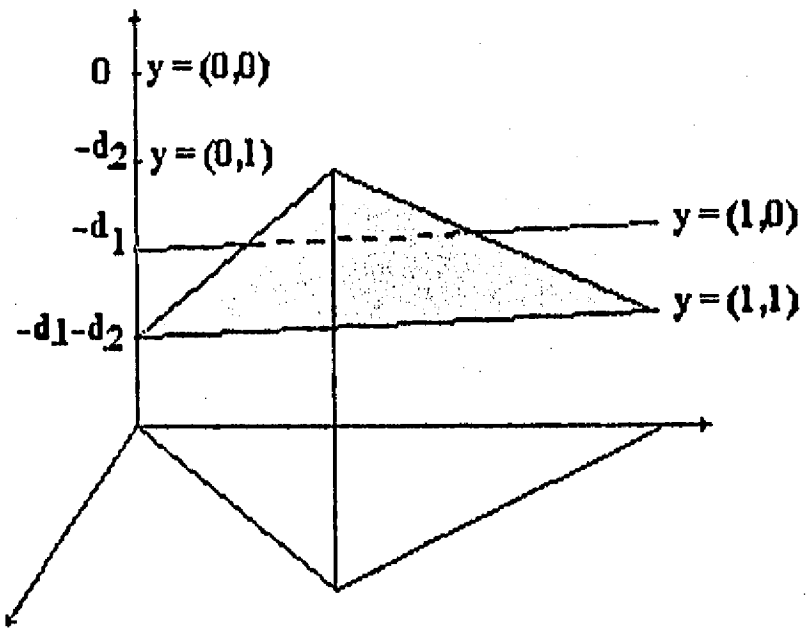


Fig. 7.5 *Corte Lógico*

Un *corte lógico* para (7.5) es una restricción en los valores posibles de y, tal que, cuando se aplique no tiene efecto en el epígrafe proyectado Ep de (7.5). Entonces, se tiene:

Lemma 7.1 La adición de un corte lógico al conjunto de restricciones en (7.5) no cambia el valor de la solución óptima.

Por ejemplo, el corte (3) remueve el punto $(0,0,0,1,-d_2)$ del grafo (7.2), entonces, ese corte remueve sólo el punto $(0,0,-d_2)$ etiquetado $y = (0,1)$ del grafo proyectado G_p en la Fig. 7.5. El epígrafe proyectado E_p no está cambiado, ya que $d_2 \ge 0$ implica que el origen cae directamente arriba de este punto. La ecuación simbólica (7.4) es, por lo tanto, un corte lógico y no afecta el valor de la solución óptima.

Una noción de dominación se agrega a esta situación. Si D es el conjunto factible para (7.5), se puede decir que un punto $y \in \{0,1\}^p$ es *factible* si $(x,y) \in D$ para alguna x. Entonces, un punto $y' \in \{0,1\}^p$ es *dominado* por y si $y \le y'$ y $(x,y) \in$ D siempre que $(x,y') \in$ D. En el ejemplo, $(y_1,y_2) = (0,0)$ domina $(0,1)$ porque $(0,0) \le (0,1)$ y $(x_1,x_2,0,1) \in D$

implica $(x_1,x_2) = (0,0)$, lo cual, finalmente implica que $(x_1,x_2,0,0) \in D$. Esto es evidente en la Fig. 7.5 porque cualquier punto que es una proyección de $(x_1,x_2,0,1)$, como es el punto $y = (0,1)$, cae bajo un punto que es una proyección de $(x_1,x_2,0,0)$ como es el punto $y = (0,0)$. Entonces todas las soluciones con $(y_1,y_2) = (0,1)$ pueden ser eliminadas.

De otra manera, $(y_1,y_2) = (0,0)$ no domina (1,0) a pesar del hecho de que $(0,0) \leq (1,0)$. Esto es porque $(x_1,x_2,1,0) \in$ D no implica $(x_1,x_2,0,0) \in$ D, ya que el anterior permite $x_1 > 0$ y el último no. Esto es evidente en la Fig. 7.5 porque los puntos que son proyecciones de $(x_1,x_2,1,0)$, como el segmento de linea $y = (1,0)$, no todos caen debajo del punto $y = (0,0)$. Así que no podemos borrar todas las soluciones con $(y_1,y_2) = (1,0)$.

7.3 Caracterización

El ejemplo de la sección previa ilustra el siguiente hecho.

Lemma 7.2 Si $S \subset \{0,1\}^p$,

$$y \in S \tag{7.8}$$

es un corte lógico para (7.5) si y sólo si cada $y' \notin S$ factible está dominada por alguna $y \in S$ factible.

Prueba. Suponga primero que (7.8) es un corte lógico y permita que $y' \notin S$ sea factible. Entonces, removiendo cualquier punto de la forma (x,y') de D no cambia E_p. Esto significa que la proyección $(x, cx - dy')$ de cualquier $(x, y, cx - dy') \in$ G cae debajo de la proyección $(x,cx - dy)$ de algún punto $(x, y, cx - dy) \in$ G para el cual $y \in S$. Entonces $cx - dy' \leq cx - dy$. Ya que $d \geq 0$, $y \leq y'$, y se sigue que y' está dominada por y.

Contrariamente, suponga que cada $y' \notin S$ factible está dominada por alguna $y \in S$ factible. Ya que $d \geq 0$, la proyección $(x, cx - dy')$ de cualquier $(x, y', cx - dy') \in$ G cae debajo de la proyección $(x, cx - dy)$ de $(x, y, cx - dy)$. Se sigue que (7.8) es un corte lógico.

En el ejemplo, el corte lógico $y_1 - y_2 \geq 0$ es equivalente a

$$y \in \{(0,0),(1,0),(1,1)\} \tag{7.9}$$

El Lemma 7.2 confirma que (7.9) es un corte lógico porque el único punto que no está incluido, el $y = (0,1)$, está dominado por el punto (0,0) en $\{(0,0),(1,0),(1,1)\}$.

Si S contiene todas los puntos factibles, entonces (7.8) es un corte válido. El corte (7.9) no es válido porque S no contiene al punto factible (0,1).

En programación entera pura (*i.e.* cuando las variables x no aparecen en (7.5)), y domina y' si y sólo si $y \leq y'$. El Lema 7.2. por lo tanto, se vuelve mucho más simple.

Corolario 7.1 (7.8) es un corte lógico para un problema puro de programación entera si y sólo si, para cada $y' \notin S$ factible, existe una $y \in S$ factible para la cual $y \leq y'$.

Ahora enfrentamos el problema de determinar cuando todos los cortes lógicos no válidos han sido identificados. En el ejemplo anterior, es claro de (7.6) y (7.7) que no se pueden cortar otros valores no factibles de y que no sean (0,1) sin cambiar el epígrafe. Por ejemplo, si cortamos (1,0), entonces se remueve el conjunto $\{(x_1,0,c_1x_1 - d_1) \mid 0 \leq x_1 \leq M\}$ de G_p (representado por una línea segmentada en la Fig. 7.5). Esto claramente altera el epígrafe proyectado E_p.

En general, podemos checar cuando hemos encontrado todos los cortes lógicos no válidos agregando los cortes lógicos conocidos al conjunto de restricciones y aplicando el siguiente corolario para determinar cuando existen más cortes lógicos no válidos.

Corolario 7.2 El problema (7.5) tiene un corte lógico no válido (7.8) si y sólo si algún $y' \in \{0,1\}^p$ factible es dominado por algún $y \neq y'$ factible.

Si agregamos el corte lógico (7.9) a (7.2), entonces ninguna y' factible es dominada por otro punto factible. Los únicos puntos factibles y' son (0,0),(1,0) y (1,1). Cuando $x_1 > 0$, $(x_1,0,1,0) \in D$ pero, $(x_1,0,0,0) \notin D$. Cuando $x_1,x_2 > 0$ $(x_1,x_2,1,1) \in D$ pero $(x_1,x_2,1,0)$, $(x_1,x_2,0,0) \notin D$. Así, ningún otro punto factible puede ser cortado.

8. Cortes lógicos de Benders

La idea detrás de la descomposición de Benders es generalizada a una base lógica por Hooker [33], aplicando la idea de la programación 0-1 al problema de satisfabilidad. Aquí la idea es aplicada a la programación lineal mixta-lógica.

Un corte de Benders para un problema de programación lineal mixta-lógica puede ser generado en cada nodo del árbol de búsqueda, de la siguiente manera. Se deja que las fórmulas lógicas contengan el problema maestro. El subproblema consiste de la relajación lineal en ese nodo; *i.e.*, el conjunto de restricciones lineales que es forzado por las proposiciones que son verdaderas en ese nodo. Entonces, si y_j se fija a verdadero para $j \in J_1$, la relajación lineal del problema es,

$$\begin{aligned} & \min \ cx - dy && && (7.10) \\ \text{s.a} \quad & Ax \geq a && (u) \\ & A^j x \geq d^j, \quad j \in J_1 && (u^j) \\ & x_j \geq 0. \end{aligned}$$

donde $Ax \geq a$ representa cortes agregados en el nodo raíz. Deje que u, u^j sean las variables duales comos se muestra. Entonces, el siguiente es un límite válido sobre el valor óptimo:

$$z \geq ua + \sum_{j \in J_1} u^j d^j.$$

Esto también puede ser escrito como

$$z \geq ua + \sum_{j \in J_1} u^j d^j y_j \qquad (7.11)$$

si cada y_j toma su valor actual, digamos 1 (para verdadero). De hecho (7.11) es un límite válido para cualquier y para la cual u permanece factible en el dual de (7.10), *i.e.*, cualquier y para la cual

$$ua + \sum_{j \in J_1} u^j A^j y_j \leq c \qquad (7.12)$$

Esto da el corte de Benders,

$$(7.12) \rightarrow (7.11)$$

Este corte puede ser generado en cualquier nodo no terminal y usado en cualquier nodo subsecuente para obtener un límite inferior sobre el valor óptimo, sin resolver la relajación lineal. Si el límite no es lo suficientemente grande para acotar el verdadero, se debe resolver el problema lineal.

En la práctica, es conveniente trabajar con costos reducidos. Si z^* es el valor óptimo de (7.10) y r es el vector de costos reducidos, entonces (7.11) y (7.12) respectivamente, se convierten en

$$z \geq z^* + \sum_{j \in J_1} u^j d^j (1 - y_j) \qquad (7.13)$$

y

$$- \sum_{j \in J_1} u^j A^j (1 - y_j) \leq r \qquad (7.14)$$

El corte de Bender es

$$(7.14) \rightarrow (7.13)$$

El corte generado en cada nodo es válido a través del árbol de búsqueda. No refuerza la relajación del problema lineal pero provee un límite sobre el valor óptimo que puede hacer obvia la solución de la relajación. Esto puede ser útil cuando el problema lineal es muy grande o muy difícil de resolver.

CAPITULO 5: APLICACIONES

En este capítulo se presentan ejemplos en cuatro áreas de aplicación para ilustrar los conceptos principales, ventajas y desventajas de la programación lineal mixta-lógica. Las dos primeras aplicaciones son importantes en Ingeniería Química. La primera comprende problemas en diseño de redes de procesos que es el punto fundamental de un nuevo campo en la Ingeniería Química, llamado Síntesis de Procesos y la segunda, la calendarización de procesos con transferencia cero, en plantas de proceso múltiple. La tercera aplicación, es el clásico problema de la localización de almacenes que fué escogido, porque se ajusta bastante bien al modelamiento lógico y para ejemplificar conceptos importantes mencionados en esta tesis; y finalmente, la cuarta aplicación es el problema de la *fiesta progresiva*, escogido para representar problemas en los cuales dominan los elementos discretos y donde puede verse la potencia de los métodos de inferencia lógica porque no ha podido ser resuelto con los métodos clásicos de programación entera. La idea fue la de seleccionar problemas que representaran aplicaciones reales con la complejidad que las caracteriza o que sirvieran de ilustración para muchos de los conceptos aquí expuestos.

1. Síntesis de Procesos

Los modelos de Síntesis de Procesos, frecuentemente involucran selecciones discretas (*i.e.* la decisión de que unidades integrar en un diagrama de flujo). Los enfoques de programación matemática suelen hacen uso de variables binarias 0-1 para modelar estas decisiones. Junto con las variables continuas, se pueden modelar los problemas de diseño y síntesis de procesos a través de una función objetivo y un conjunto de restricciones que representan los balances de masa y energía y las especificaciones. Sin embargo, el modelamiento es un paso crucial del éxito en el método de síntesis. En esta sección, se aplican las técnicas de programación lineal mixta-lógica, apegándose a los modelos descritos en el libro por publicarse de Grossmann, Biegler y Westerberg: **Systematic Methods for Chemical Process Design,** al modelamiento y la solución de problemas de síntesis de procesos.

1.1 Redes de Proceso

Las redes de proceso aparecen en aplicaciones en las cuales, un subconjunto de balances de masa son dados en proporciones fijas. Incluyen, por ejemplo complejos químicos industriales y secuencias de destilación. La Figura 5.1 presenta un ejemplo de una red química de proceso. El objetivo en estos problemas es el de encontrar una subred que

minimice los costos totales, fijos y variables en que incurren las unidades de proceso. Por simplicidad, se consideran las redes sin reciclaje, aunque su tratamiento es similar.

Para nuestros propósitos, una red de proceso es una red aciclica dirigida con dos clases de nodos: *nodos estructurales y nodos unitarios*. Un nodo estrucutral, representado como un círculo en la Fig. 5.1, es un nodo normal de red que conserva el flujo. Un nodo unitario, representado como un cuadrado, es un nodo de procesamiento que adicionalmente requiere un costo fijo de construcción si es que tiene que llevar flujo. Se asumirá que cada nodo fuente (nodo sin entradas) y cada nodo terminal (nodo sin salidas) es un nodo unitario fingido. Un nodo unitario i es un predecesor del nodo unitario j si existe una ruta directa de i a j. El nodo unitario j es un sucesor del nodo unitario i si es un predecesor de j. El nodo unitario k es un sucesor inmediato de la salida j de la unidad i (o un predecesor inmediato de la entrada j de i) si existe una ruta directa de la salida j de i a k (o de k a la entrada j de i) que no contiene ninguna otra unidad.

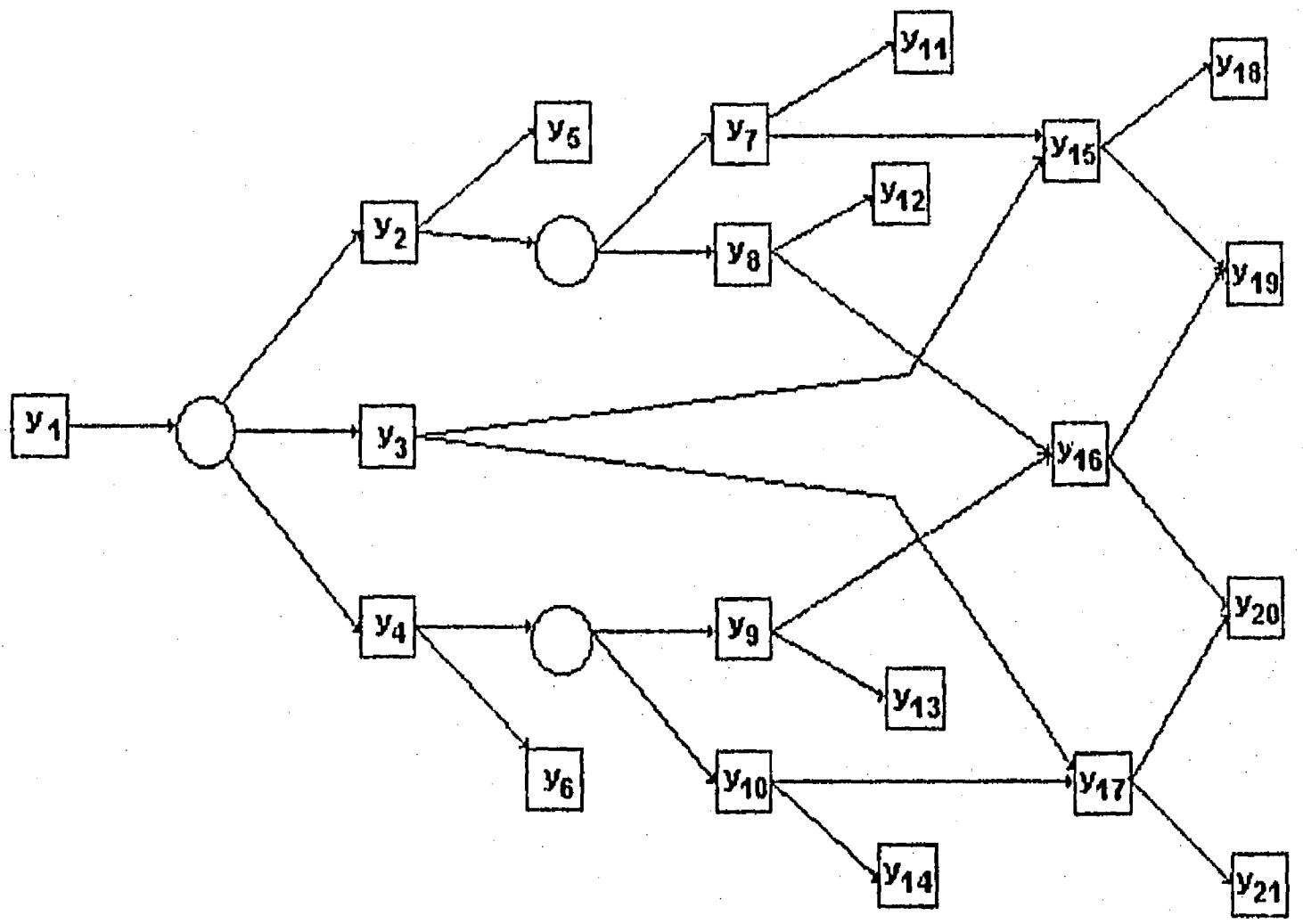


Fig. 5.1 *Ejemplo de una red de proceso*

Para cada nodo untiario i, existe una variable asociada de flujo Z_i. El flujo dentro de cada entrada j debe ser $\alpha_{ij}Z_i$, y el flujo de cada salida j debe ser $\beta_{ij}Z_i$, donde $\alpha_{ij}, \beta_{ij} > 0$. Se asocia cada nodo unitario i con una variable binaria y_i tal que $y_i = 1$ si la unidad i esta construida, y 0 en cualquier otro caso. $d_i \geq 0$ es el costo fijo de construcción de la unidad i. Si se deja que I sea el conjunto de índices de los nodos unitarios, J el de los nodos estructurales y A el conjunto de los arcos, el problema se puede formular con el siguiente modelo de programación lineal.

$$
\begin{aligned}
& \max \ cq - dy && && (1.1)\\
\text{s.a} \quad & \sum_{(i,j)\in A} q_{ij} = \sum_{(i,j)\in A} q_{jk}, && \forall j \in \mathrm{J} \\
& q_{ji} = \alpha_{ij} Z_i, && \forall (j,i)\in A,\ \forall i \in \mathrm{I} \\
& q_{ij} = \beta_{ij} Z_i, && \forall (j,i)\in A,\ \forall i \in \mathrm{I} \\
& Z_i \le M_i y_i, && \forall i \in \mathrm{I} \\
& q_{ij} \ge 0, && \forall (i,j)\in A \\
& Z_i \ge 0,\ y_i \in \{0,1\}, && \forall i \in \mathrm{I}
\end{aligned}
$$

Este modelo tiene la forma de un modelo clásico de programación lineal mixta-entera si hacemos $x = (q, Z)$.

1.2 Cortes lógicos para redes de proceso

No tiene caso construir una unidad que no acarree flujo. Por lo tanto, podemos ignorar soluciones con $y_i = 1$ y $Z_i = 0$, aún cuando puedan ser factibles. Como se explicó antes, no se pueden agregar restricciones algebraicas mixtas-enteras que dejen afuera todas esas soluciones, pero los cortes lógicos pueden excluir algunas de ellas.

Para encontrar todos los cortes lógicos para una red de proceso, se empieza con el siguiente lemma, presentado por Hooker, Yan, Grossmann y Raman [37]:

Lemma 1.1 La regla lógica

$$y_i \Rightarrow (y_{i_1} \vee \ldots \vee y_{i_m}) \qquad (1.2)$$

es un corte lógico para (1.1) si y sólo si $y_{i_1} = \ldots = y_{i_m} = 0$ implica $Z_i = 0$.

Prueba. Sea S el conjunto de puntos y que satisfacen (1.1), y $x = (q, Z)$.

Primero se asume que $y_{i_1} = \ldots = y_{i_m} = 0$ implica $Z_i = 0$. Se toma cualquier $y' \notin S$, dejando que y sea la misma que y', excepto que $y_i = 0$. Entonces $y \in S$ y $y \le y'$. También para cualquier $(x,y') \in D$, claramente $(x,y) \in D$, porque $Zi = 0$ implica que y_i puede ser cero. Así, la ecuación simbólica (1.2) es un corte lógico por el Lemma 1.1.

En contrapartida, se supone que (1.2) es un corte lógico. Se toma $(x,y') \in D$ para la cual $y_{i_1}' = \ldots = y_{i_m}' = 0$. Entonces ya sea que $y_i' = 0$, en cuyo caso debemos tener $Z_i = 0$, ó $y_i' = 1$, en cuyo caso $y' \notin S$. Pero en el último caso, existe una $y \in S$ con $y \le y'$ y $(x,y') \in D$. Pero $y \le y'$ y $y \in S$ implican $Z_i = 0$.

Obviamente no hay flujo a través de la unidad i ($Z_i = 0$) si cerramos todos los predecesores inmediatos de algunas entradas a i, o si cerramos todos los sucesores inmediatos de algunas salidas de i, porque cada $\alpha_{ij}, \beta_{ij} > 0$. Entonces se puede decir que (1.2) es un *corte inicial*, si $i_1, \ldots, i_m$ son todos los predecesores inmediatos de alguna

entrada a i, o son todos los sucesores inmediatos de alguna salida de i. Inmediatamente se tiene el siguiente Lemma.

Lemma 1.2 Cualquier corte inicial es un corte lógico.

Por ejemplo $y_{16} \Rightarrow (y_8 \vee y_9)$ es un corte inicial dela Fig. 5.1, y es también un corte lógico. No todos los cortes lógicos son iniciales, como por ejemplo el corte lógico $y_{16} \Rightarrow (y_2 \vee y_4)$. Se le llamará al corte lógico de la forma (1.2) mínimo si no es un corte lógico donde cualquier variable en la consecuencia es removida. Obviamente, un corte inicial es un corte mínimo. Tenemos,

Lemma 1.3. Si (1.2) es un corte lógico mínimo, entonces cualquiera de las unidades i_1, ..., i_m son sucesoras de la unidad i, o todas ellas son predecesoras de la unidad i.

Prueba. Suponga lo contrario. Sin pérdida de generalidad, permita que i_1, ..., i_r, sean predecesores de i y i_{r+1}, ..., i_m sucesores de i, con $r < m$. Ya que $y_i \Rightarrow (y_{i_1} \vee ... \vee y_{i_r})$ no es un corte lógico, dejando $y_{i_1} = ... = y_{i_r} = 0$, no (por el Lemma 5.1) se bloqueará el flujo dentro de i. Ya que $y_i \Rightarrow (y_{i_1} \vee ... \vee y_{i_m})$ no es un corte lógico, dejando $y_{i_{r+1}} = ... = y_{i_m} = 0$ no se bloqueará el flujo de i. Entonces por el Lemma 1.1 (1.2) no es un corte lógico, contrariamente a la hipótesis.

Lemma 1.4 Cualquier corte lógico de la forma (1.2) está implicado por una conjunción finita de cortes lógicos iniciales.

Prueba. Cualquier corte lógico de la forma (1.2) está implicado por un corte mínimo obtenido removiendo cero o más consecuentes. Se supone, por lo tanto que (1.2) es un corte mínimo. Por el Lemma 1.3, se puede asumir que todas las unidades que aparecen en el lado consecuente son predecesores de la unidad i. (El argumento es similar si todas ellas son sucesoras). El siguiente procedimiento genera un conjunto de cortes iniciales que implican la ecuación simbólica (1.2).

Dejando j_1, ..., j_k ser los predecesores inmediatos de cualquier entrada a la unidad i, y dejando que el conjunto F consista de (1.2). Entonces (1.2) está claramente implicada por:

$$\begin{aligned} &y_i \Rightarrow (y_{j_1} \vee ... \vee y_{j_k}), \\ &y_h \Rightarrow (y_{i_1} \vee ... \vee y_{i_m}), \quad \forall h \in \{j_1, ..., j_k \backslash i_1, ..., i_m\}, \end{aligned} \tag{1.3}$$

donde (1.3) es un corte inicial por el Lemma (1.2). Agregue las reglas (1.3) a F y borre (1.2) de F.

Para cualquier regla en F, repita el procedimiento de arriba. El procedimiento termina cuando F está vacía. Ya que la red tiene un número finito de unidades, el procedimiento generará un conjunto finito de cortes iniciales que implican (1.3).

Como un ejemplo, considere la Fig. 5.1 otra vez. $y_{16} \Rightarrow (y_2 \vee y_4)$ es un corte lógico, pero no inicial. Está implicado por los cortes iniciales $y_{16} \Rightarrow (y_8 \vee y_9)$, $y_8 \Rightarrow y_2$ y $y_{11} \Rightarrow y_4$.

Finalmente se mostrará que los cortes iniciales cortan todos los puntos enteros que pueden ser cortados por los cortes lógicos de cualquier forma.

Teorema 1.1 Si todos los cortes iniciales son agregados a (1.1), entonces no existen más cortes lógicos válidos.

Prueba. Por el Corolario 7.1 del capítulo anterior, es suficiente mostrar que ninguna y' no factible está dominada por una $y \neq y'$ factible. Considere cualquier y' factible. Deseamos mostrar que existe algún $(x,y') \in D$ para la cual, ningún otro punto $(x,y) \in D$ satisface $y \leq y'$. Para hacer esto, se supone, sin pérdida de generalidad que $y' = (\mathbf{0},e)$, donde $\mathbf{0}$ es un factor de k ($<p$) ceros y e un vector de $p - k$ unos.

Afirmación. Existen puntos $(x^{k+1}, 0, y^{-k+1})$, ..., $(x^p, 0, y^{-p}) \in$ D con $x^i = (q^i, Z^i)$ y $y^{-i} = (y^i_{k+1},...,y^i_p)$, tal que cada $Z^i_l > 0$.

Prueba de la Afirmación. Suponga que la afirmación es falsa para alguna $i \in \{k+1,..., p\}$. Entonces $y_1 = ... = y_k = 0$ implica que $Z_i = 0$, lo cual significa, por el Lemma 1.1 que $y_i \Rightarrow (y_1 \vee ... \vee y_k)$ es un corte lógico. Es por lo tanto, implicado por los cortes iniciales, lo cual contradice la suposición de que y' (que tiene $y'_i = 1$ y $y'_1 = ... = y'_k = 0$) satisface todos los cortes iniciales.

Ya que $y' = (\mathbf{0},e)$ satisface todos los cortes iniciales, la afirmación implica que $(x^{k+1},\mathbf{0},e)$, ..., $(x^p,\mathbf{0},e) \in D$. Por lo tanto se puede permitir que $(x,\mathbf{0},e)$ sea su combinación convexa usando pesos iguales, con $x = (q,Z)$, y note que $(x,\mathbf{0},e) \in D$. Ya que $Z_i > 0$ para $i = k+1$, ..., p no existe punto $(x,y) \in D$ con $y \leq y' = (\mathbf{0},e)$ y $y \neq y'$. Se sigue con el teorema.

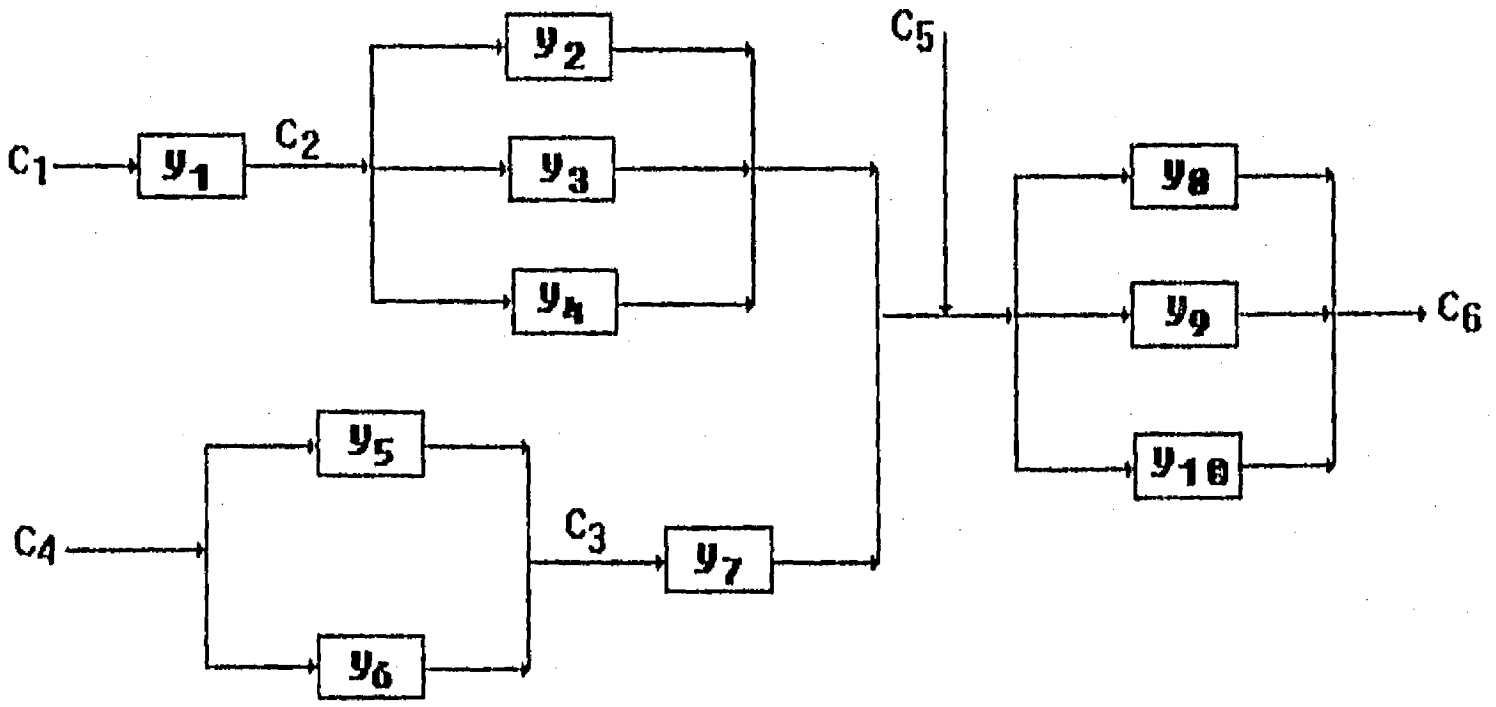


Fig. 5.2 *Red química de proceso*

Ejemplo. El siguiente ejemplo presentado en la Fig. 5.2 de una red de proceso químico tiene tres partes importantes de selección. La corriente C_2 puede escoger entre los reactores representados por y_2, y_3 y/o y_4, la corriente C_4, entre los reactores y_5 y y_6 y la corriente final que sale de estos reactores y que se mezcla con la corriente C_5. debe escoger entre los reactores y_8, y_9 y y_{10}. En todos los casos se contempla la posibilidad de usar uno sólo de estos reactores o de repartir el flujo en más de uno.

Para esta red química de proceso se pueden generar los siguientes cortes lógicos:

Expresión Algebraica	Expresión Simbólica
$y_2 + y_3 + y_4 - y_1 \geq 0$	$y_1 \Rightarrow y_2 \vee y_3 \vee y_4$
$y_1 - y_2 \geq 0$	$y_2 \Rightarrow y_1$
$y_1 - y_3 \geq 0$	$y_3 \Rightarrow y_1$
$y_1 - y_4 \geq 0$	$y_4 \Rightarrow y_1$
$y_5 + y_6 - y_7 \geq 0$	$y_7 \Rightarrow y_5 \vee y_6$
$y_7 - y_5 \geq 0$	$y_5 \Rightarrow y_7$
$y_7 - y_6 \geq 0$	$y_6 \Rightarrow y_7$
$y_8 + y_9 + y_{10} - y_1 \geq 0$	$y_1 \Rightarrow y_8 \vee y_9 \vee y_{10}$
$y_8 + y_9 + y_{10} - y_7 \geq 0$	$y_7 \Rightarrow y_8 \vee y_9 \vee y_{10}$

Otro conjunto muy importante de problemas que pueden resolverse exitosamente a través del uso de cortes lógicos es el de la síntesis óptima de redes de columnas de separación. En estos modelos se debe construir una red que seleccione las separaciones que produzcan los componentes esperados, utilizando las columnas de destilación que a su vez, representen un costo mínimo para la red. Dada su importancia y frecuente utilización, en la siguiente sección se presenta el modelamiento detallado de estos problemas.

1.3 Síntesis de secuencias de Columnas de Separación

Se considera el problema donde una alimentación multicomponente debe ser separada dentro de componentes esencialmente puros, a través del uso de columnas de separación instántaneas. Con el objeto de reducir el tamaño y la complejidad de los modelos lineales utilizados, se van a realizar una serie de suposiciones a lo largo de esta presentación, que simplifiquen dichos modelos y que eviten el incremento excesivo en su tamaño y la introducción de componentes no lineales.

Primero, como se presenta en la Fig. 5.3, se considerará una columna de destilación con una sola alimentación, en la cual se realizan separaciones entre componentes clave, pesados y ligeros, que son adyacentes entre ellos.

Si se consideran constantes la presión y el radio de reflujo, entonces, realizando cálculos rápidos con cualquier método, se pueden obtener relaciones de balance de masa, en términos de las corrientes de alimentación y de salida de la columna, como puede verse

en la Fig. 5.4, donde d_i y b_i representan los flujos de masa del componente i en las salidas del vapor destilado y del fondo de la columna, y γ_i son las fracciones de recuperación correspondientes obtenidas de los balances de masa de la columna. Si se asume que las fracciones γ_i son constantes, es claro que el conjunto de ecuaciones en (1.4) se pueden reducir a ecuaciones lineales.

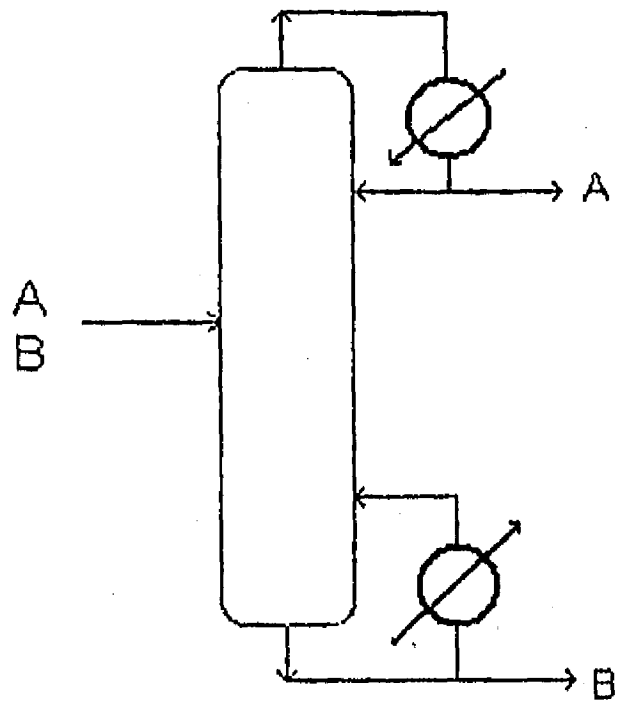
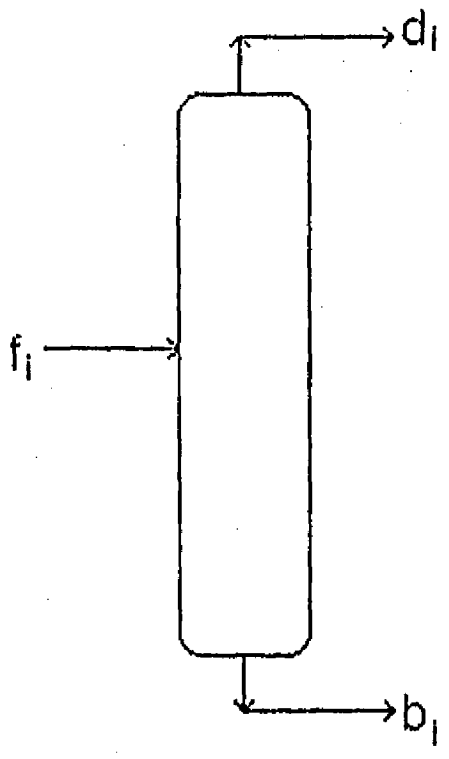


Fig. 5.3 *Columna de separación*

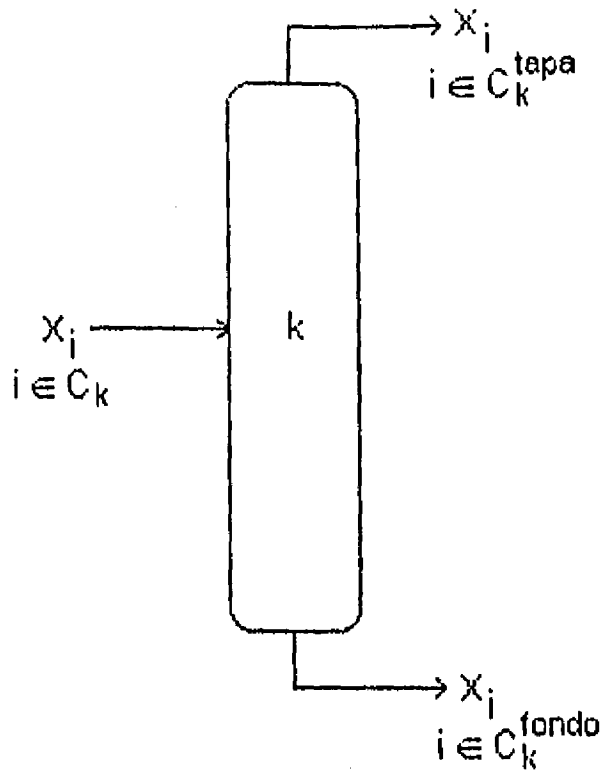


$$d_i = \gamma_i f_i$$
$$b_i = (1 - \gamma_i) f_i \qquad (1.4)$$

Fig. 5.4 *Balance de masa para una columna de destilación multicomponente*

Aunque se pueden usar las ecuaciones de balance de masa, tal como vienen dadas en (1.4), se considerará una simplificación adicional que permitirá escribir el modelo sólo en términos de las corrientes de alimentación total de cada columna de destilación. Si se asume una recuperación del 100%, entonces, para cada columna k, se pueden determinar

a-priori, la fracciones del total de la alimentación que van a ser recuperadas en la tapa y en el fondo, por las siguientes ecuaciones (ver Fig. 5.4).



$$\xi_k^{tapa} = \sum_{i \in C_k^{tapa}} x_i^F \Big/ \sum_{i \in C_k} x_i^F \qquad (1.5)$$

$$\xi_k^{fondo} = \sum_{i \in C_k^{fondo}} x_i^F \Big/ \sum_{i \in C_k} x_i^F$$

Fig. 5.5 *Módulo para flujo total y separación fuerte*

donde X^F_i es la fracción mol del componente i en la mezcla inicial, C_k, C_k^{tapa} y C_k^{fondo}, son los conjuntos de componentes involucrados en la salida del destilado y del fondo de la columna k.

Como ejemplo, se considerará la columna de la Fig. 5.6, la cual tiene como alimentación inicial una mezcla multicomponente. Si se aplican las Ecs. (1.5), es claro que $x^{tapa} = 0.2 + 0.4 = 0.6$ y $x^{fondo} = 0.3 + 0.1 = 0.4$. Para la columnna de la Fig. 5.5, que sólo tiene los componentes C y D en la alimentación, se sigue de la Ec. (1.4) que $x^{tapa} = 0.3/(0.1 + 0.3) = 0.75$, $x^{fondo} = 0.1/(0.1 + 0.3) = 0.25$ Con estas fracciones, se pueden expresar los flujos de las dos corrientes de producto en la columna, en términos del flujo total de alimentación F, dentro de la columna, como puede verse en las Figs. 5.6 y 5.7.

Ya que con las suposiciones anteriores se pueden modelar los balances de materia a través de los flujos de alimantación total para cada columna, es conveniente modelar las cargas de calor del re-hervidor y del condensador, y el costo de capital en términos de estas variables. Asumiendo las mismas cargas en el condensador y en el re-hervidor, las cargas de calor para la columna *k*, pueden ser expresadas como funciones lineales:

$$Q_k = K_k F_k \qquad (1.6)$$

donde K_k es una constante derivada de un cálculo rápido. Finalmente, el costo anualizado de la columna, que incluye el modelo de costo fijo para la inversión y los costos de las corrientes auxiliares de calor, viene dado por:

$$C_k = z_k + \beta_k F_k + (c_H + c_C)Q_k \qquad (1.7)$$

donde z_k es el costo fijo anual, β_k es el factor de tamaño para la columna, y c_H, c_C son los costos unitarios para el calentamiento y el enfriamiento del re-hervidor y del condensador, respectivamente.

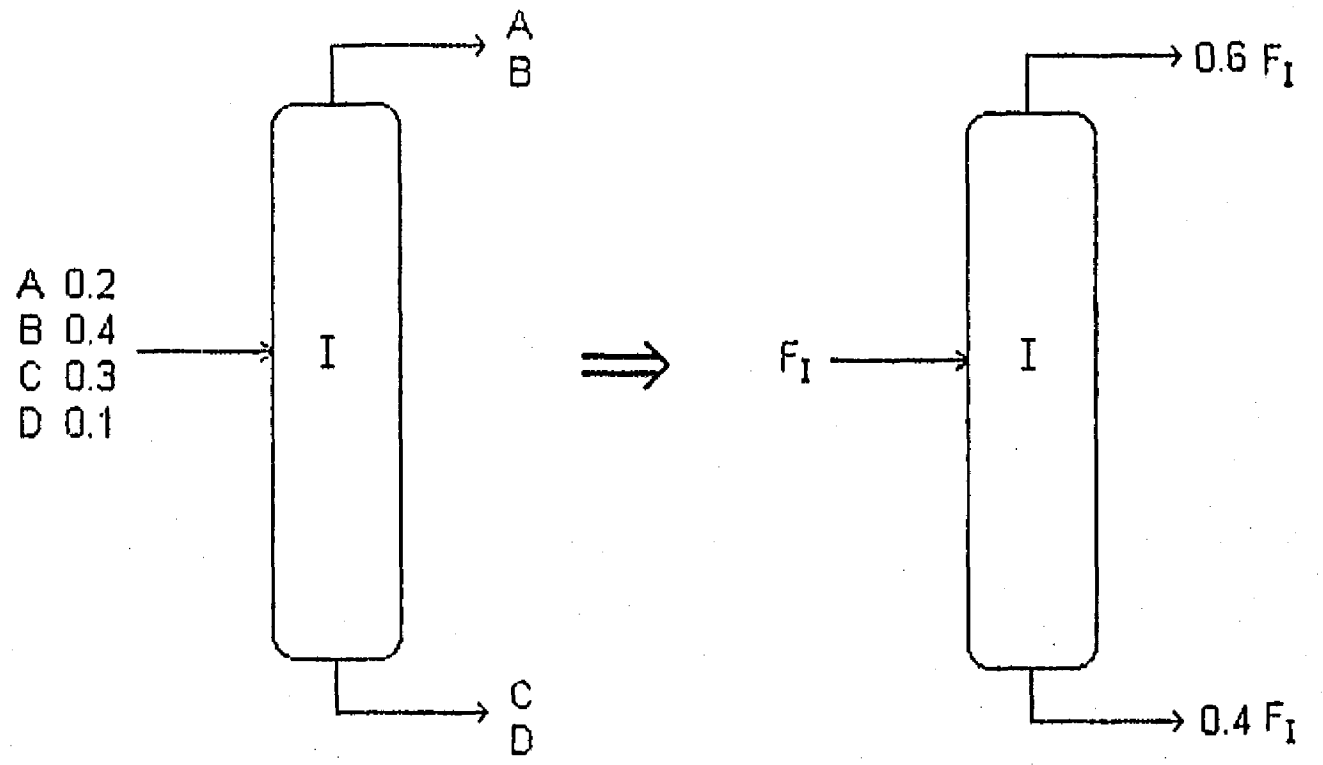


Fig. 5.6 *Ejemplo de la separación inicial de una mezcla de cuatro componentes*

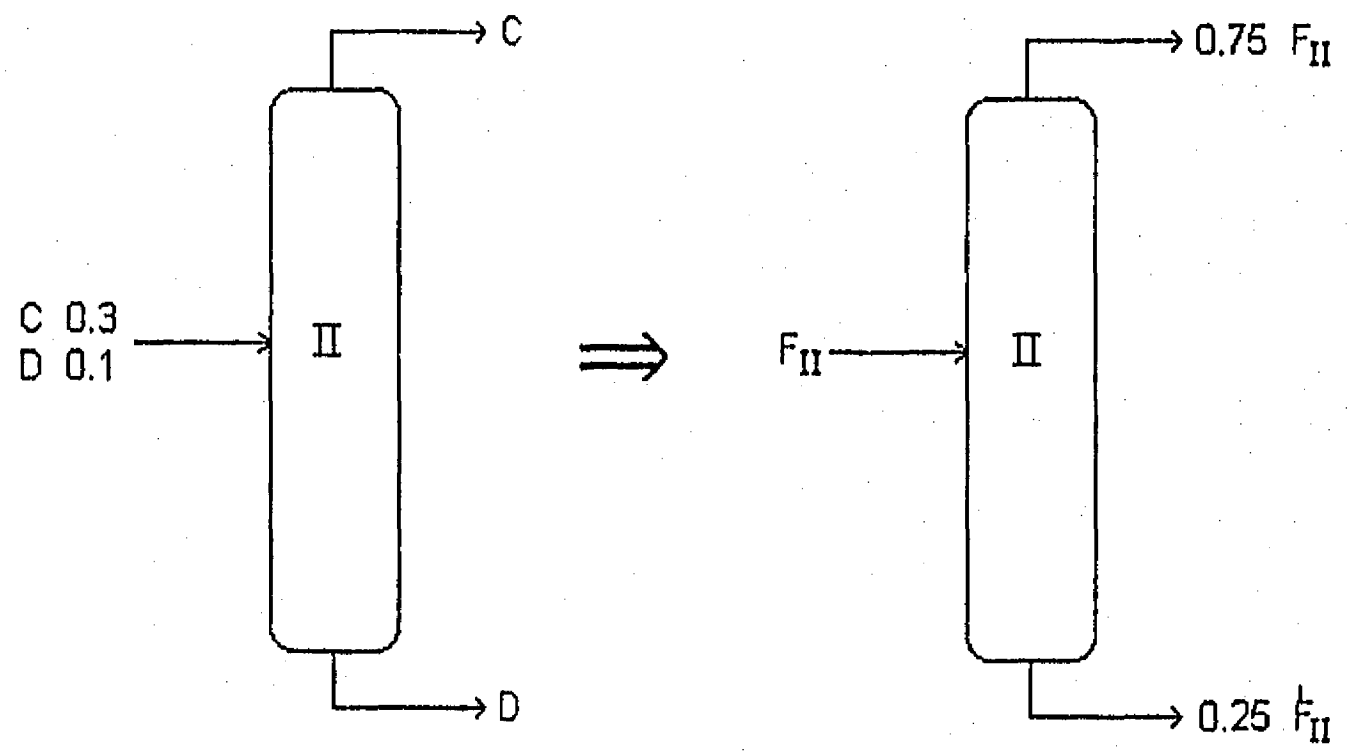


Fig. 5.7 *Ejemplo de separación intermedia para una mezcla de dos componentes*

Antes de presentar la forma general del modelo de programación lineal mixta-lógica, basado en las ecuaciones lineales de la seccion previa, se considerará como ejemplo, el caso donde se tiene una mezcla de 4 componentes A,B,C y D que queremos separar en productos esencialmente puros, de acuerdo al esquema presentado en la Fig. 5.8.

Los datos sobre la composición de esta mezcla, las constantes para el balance de calor, los datos de costo y las fracciones de recuperación, necesarias para las ecuaciones de balance de masa, basadas en los datos de alimentación y calculadas de acuerdo a la Ec. (1.5), se presentan en la Tabla 5.1

Primero, hay que generar la superestructura para este problema. La representación correspondiente en forma de red, presentada por Andrecovich y Westerberg (1985) se muestra en la Fig. 5.8. A cada una de las 10 columnas de destilación en esta red, se le pueden asignar variables Y_i que denoten su existencia potencial, y una variable F para su corriente de alimentación.

Los Balances de Masa correspondientes a cada nodo de la red son los siguientes:

Para el nodo inicial en la red, tenemos:

$$F_1 + F_2 + F_3 = 1000 \qquad (1.8)$$

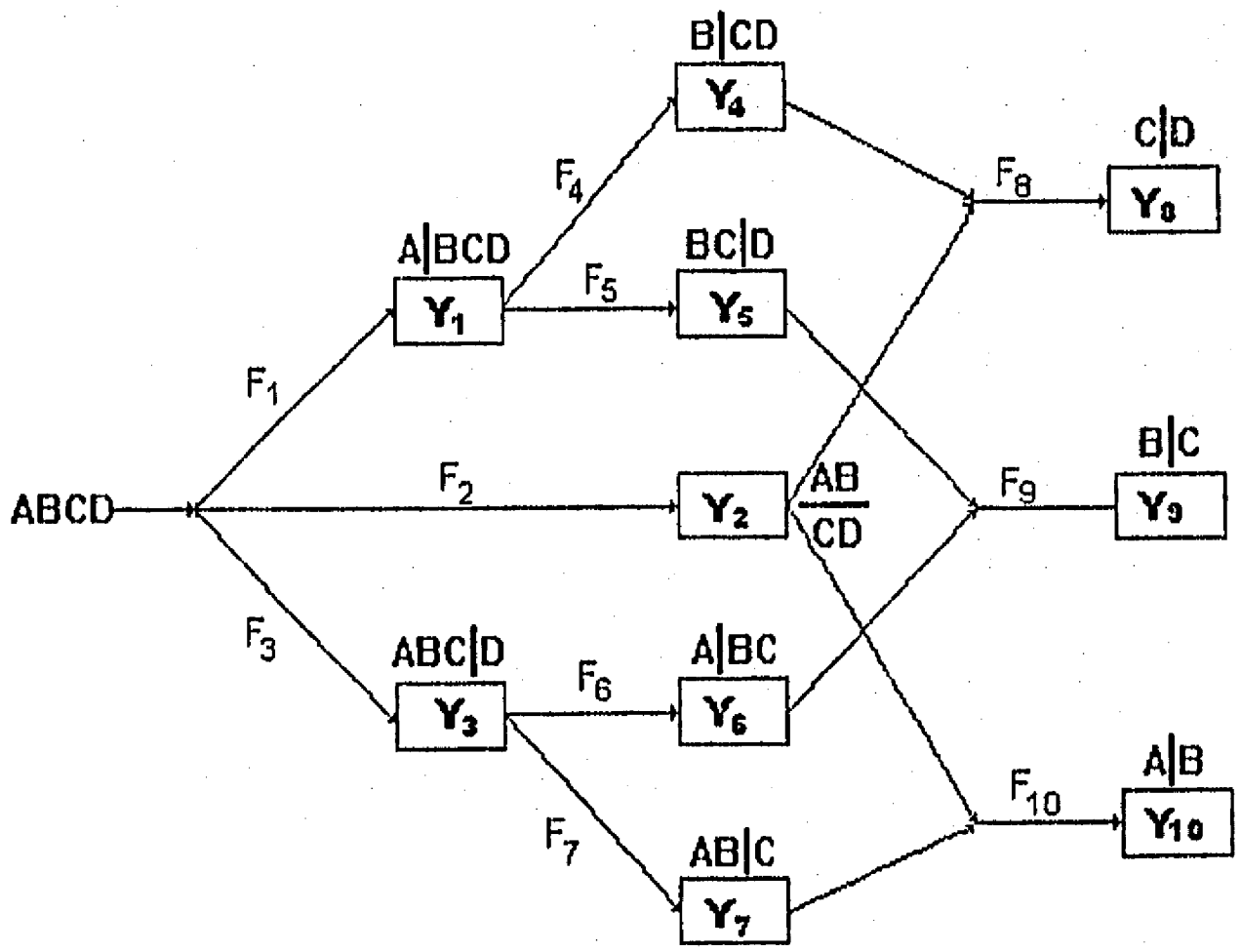


Fig. 5.8 *Superestructura para la separación del ejemplo de 4 componentes*

Para los nodos restandes en la red, en vez de considerar balances de masa alrededor de cada columna, se considerarán balances de masa para cada producto intermedio. La razón para esto es que en la superestructura de la Fig. 5.8 se asociaron flujos sólo a la alimentación de cada columna, de tal manera que las corrientes de producto, no necesariamente tienen un flujo asociado, como es el caso de las columnas 2,4,5,6 y 7.

Basados en las fracciones de recuperación de la Tabla 5.1, los balances de masa para cada corriente intermedia de producto se presentan en las ecuaciones (1.9) - (1.13).

Tabla 5.1: *Datos para el problema de ejemplo*

Alimentacion Inicial	Ftot = 1000 kgmol/hr			Composicion	A = 0.15
COSTO DE SERVICIOS:				(Fraccion Mol)	B = 0.3
Agua de Enfriamiento	Cc = 1.3(10^3$hr/10^6kJyr)				C = 0.35
Vapor	Ch = 34 (10^3$hr/10^6kJyr)				D = 0.2

		Costo de Inversion		Coeficientes	Fracciones de Recuperacion en la Superestructura	
k	Columna de Separacion	FIJO Zk	VARIABLE β_k	de Calor Kk		
1	A/BCD	145	0.42	0.028	ξ A = 0.15	ξ BCD = 0.85
2	AB/CD	52	0.12	0.042	ξ AB = 0.45	ξ CD = 0.55
3	ABC/D	76	0.25	0.054	ξ ABC = 0.8	ξ D = 0.2
4	B/CD	38	0.14	0.04	ξ B = 0.353	ξ CD = 0.647
5	BC/D	66	0.21	0.047	ξ BC = 0.765	ξ D = 0.235
6	A/BC	25	0.78	0.024	ξ A = 0.188	ξ BC = 0.812
7	AB/C	44	0.11	0.039	ξ AB = 0.5625	ξ C = 0.44
8	C/D	58	0.19	0.044	ξ C = 0.636	ξ D = 0.364
9	B/C	37	0.08	0.036	ξ B = 0.462	ξ C = 0.538
10	A/B	112	0.39	0.022	ξ A = 0.333	ξ B = 0.667

a) Corriente intermedia (BCD), producida en la columna 1, y dirigida a las columnas 4 y 5.

$$F_4 + F_5 - 0.85\, F_1 = 0 \qquad (1.9)$$

b) Corriente intermedia (ABC), producida en la columna 3, y dirigida a las columnas 6 y 7,

$$F_6 + F_7 - 0.8\, F_3 = 0 \qquad (1.10)$$

c) Corriente intermedia (AB), producida en las columnas 2 y 7, y dirigida a la columna 10,

$$F_{10} - 0.45\, F_2 - 0.563\, F_7 = 0 \qquad (1.11)$$

d) Corriente intermedia (BC), producida en las columnas 5 y 6, y dirigida a la columna 9,

$$F_9 - 0.765\, F_5 - 0.812\, F_6 = 0 \qquad (1.12)$$

e) Corriente intermedia (CD), producida en las columnas 2 y 4, y dirigida a la columna 8

$$F_8 - 0.55\, F_2 - 0.647\, F_4 = 0 \qquad (1.13)$$

Las cargas de calor del condensador y del rehervidor, se pueden representar por las variables continuas Q_k, k = 1,...,10, y de la ecuación (1.6), vienen dadas por las siguientes ecuaciones:

$$Q_k = K_k F_k, \qquad k = 1, \ldots, 10 \qquad (1.14)$$

donde los parámetros K_k vienen dados en la Tabla 5.1. Note que las ecuaciones anteriores aumen que las cargas en los condensadores y los rehervidores es la misma. En la práctica estos valores son frecuentemente cercanos.

Los 10 flujos en las ecuaciones (1.8) - (1.13) se encuentran acotados por el valor del flujo total, 1000:

$$0 \le F_k \le 1000, \qquad k = 1, \ldots, 10 \qquad (1.15)$$

Dado que cada unidad puede estar o no estar presente, si Y_i corresponde a la existencia de esa columna de destilación en la red seleccionada y $\neg Y_i$ indica su ausencia, podemos escribir las siguientes disyunciones:

$$Y_k \vee \neg Y_k, \qquad k = 1, \ldots, 10 \qquad (1.16)$$

La existencia de una unidad *k*, obliga a considerar su costo fijo, como parte de los costos de la red seleccionada, y si sólo seleccionamos equipo que vamos a usar, también sabemos que el flujo a través de ella, debe ser estrictamente mayor que cero. Por otro lado, la no existencia de la unidad *k*, nos indica un flujo de cero a través de ella y además un costo cero para la unidad *k*, ya que no está considerada dentro del esquema seleccionado.

Y_k: Z_k = Costo fijo de la unidad *k* y $F_k > 0$.
$\neg Y_k$: $F_k = 0$ (Flujo cero a través de esa unidad), y $Z_k = 0$.

La forma general del problema hace que al hacer un flujo cero, se haga automaticamente el costo cero (ya que el modelo pretende minimizar costos), por lo que las disyunciones sobre la existencia o no de la unidad *k*, pueden escribirse en la forma general de las disyunciones para los problemas de costo fijo, presentada por Beaumont (1985).

$$\{Z_k = \text{Costo Fijo}(k),\ F_k > 0\} \vee \{F_k = 0\}, \qquad k = 1, \ldots, 10 \quad (1.17)$$

La función objetivo vendrá dada por la minimización de la suma de los costos dados en la Ec. (1.7), para las 10 columnas. Esto es,

$$\min C = \sum_{k=1}^{10} (z_k + \beta_k F_k) + (34 + 1.3) \sum_{k=1}^{10} Q_k \qquad (1.18)$$

donde los coeficientes de costo z_k y β_k ,vienen dados en la Tabla 5.1.

La función objetivo en (15) está sujeta a las restricciones (1.8) a (1.15) correspondientes al modelo disyuntivo para determinar la secuencia de destilación óptima en la superestructura de la Fig. 5.8. Note que se tienen 20 variables continuas (F_k, Q_k, k = 1, ... ,10) y 16 ecuaciones: (1.8) a (1.13) y las diez ecuaciones en (1.14). Entonces, el problema tiene 4 grados de libertad. También se tienen 10 disyunciones en (1.16) y 10 límites superiores en (1.15), que acotan los valores posibles de los flujos.

Finalmente, se tiene información lógica, desprendida de la superestructura analizada, que nos fija y/o limita algunas posibilidades de combinación entre las unidades que pueden existir simultáneamente en la red de columnas de destilación, nos permite hacer inferencia lógica y fijar nuevas variables, una vez que se fija el valor de una de ellas, y que aún no se ha tomado en cuenta en el modelo.La información lógica contenida en la superestructura es la siguiente:

$$
\begin{aligned}
&Y_1 \Leftrightarrow Y_4 \vee Y_5 \\
&Y_2 \Leftrightarrow Y_8 \wedge Y_{10} \\
&Y_3 \Leftrightarrow Y_6 \vee Y_7 \\
&Y_4 \Rightarrow Y_8 \\
&Y_5 \Rightarrow Y_9 \\
&Y_6 \Rightarrow Y_9 \\
&Y_7 \Rightarrow Y_{10}
\end{aligned}
\qquad (1.19)
$$

Estas ecuaciones pueden ser convertidas a su forma conjuntiva normal (CNF), con el objeto de emplear el algoritmo de resolución unitaria para realizar la inferencia lógica correspondiente:

$$
\begin{aligned}
&\neg Y_1 \vee Y_4 \vee Y_5 \\
&\neg Y_4 \vee Y_1 \\
&\neg Y_5 \vee Y_1 \\
&\neg Y_2 \vee Y_8 \\
&\neg Y_2 \vee Y_{10} \\
&\neg Y_3 \vee Y_6 \vee Y_7 \\
&\neg Y_6 \vee Y_3 \\
&\neg Y_7 \vee Y_3 \\
&\neg Y_4 \vee Y_8 \\
&\neg Y_5 \vee Y_9 \\
&\neg Y_6 \vee Y_9 \\
&\neg Y_7 \vee Y_{10} \\
&\neg Y_8 \vee Y_4 \vee Y_2 \\
&\neg Y_9 \vee Y_5 \vee Y_6 \\
&\neg Y_{10} \vee Y_2 \vee Y_7
\end{aligned}
\qquad (1.20)
$$

Aparte de estas proposiciones expresadas en forma conjuntiva normal, tenemos más información lógica en la superestructura, que puede expresarse en la forma de cláusulas: "A lo más *m* de ...", que corresponderían a cláusulas extendidas y que pueden manejarse fácilmente en su forma original, dentro del código computacional, pero que traducidas a

su forma CNF originarían cientos de cláusulas proposicionales. Dichas ecuaciones son las siguientes:

$$\begin{gathered}\text{A lo más 1 de } \{Y_1, Y_2, Y_3\} \\ \text{A lo más 1 de } \{Y_4, Y_5\} \\ \text{A lo más 1 de } \{Y_6, Y_7\} \\ \text{A lo más 1 de } \{Y_2, Y_4\} \\ \text{A lo más 1 de } \{Y_5, Y_6\} \\ \text{A lo más 1 de } \{Y_2, Y_7\} \\ \text{A lo más 3 de } \{Y_1, Y_2, Y_3, Y_4, Y_5, Y_6, Y_7, Y_8, Y_9, Y_{10}\}\end{gathered} \tag{1.21}$$

Si se resuelve este problema disyuntivo lineal, se obtiene la secuencia óptima mostrada en la Fig. 5.9, la cual tiene un costo anualizado de \$3,308 x 10^3/año.

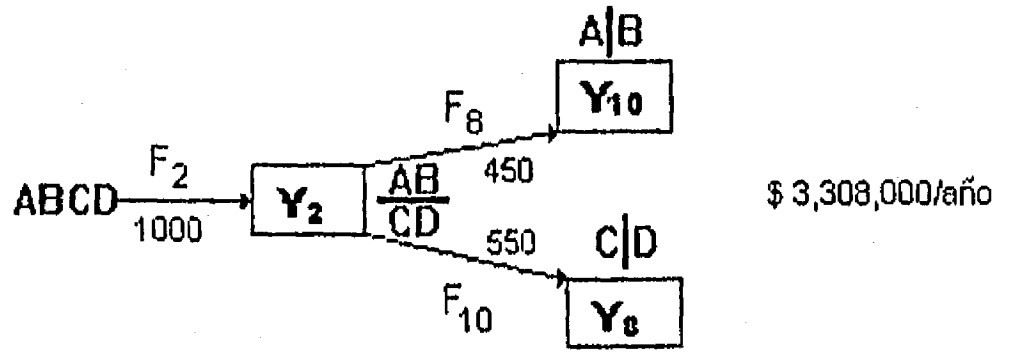


Fig. 5.9 *Secuencia óptima de columnas de destilación para el problema ejemplo*

Ya que la solución óptima en la Fig. 5.9 viene dada por la presencia de las unidades 2, 8 y 10, se puede anular esta selección de columnas, agregando la proposición lógica:

$$\text{A lo más 2 de } \{Y[2], Y[8], Y[10]\}$$

Resolviendo el problema disjunctivo lineal correspondiente, con esta proposición lógica, se obitene la segunda mejor solución, que se muestra en la Fig. 5.10 y que corresponde a una secuencia directa que tiene un costo anualizado de \$ 3,927 x 10^3/año.

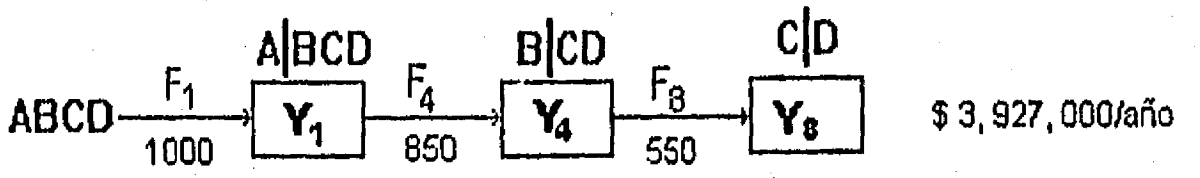


Fig. 5.10 *Segunda secuencia óptima*

Para obtener la tercera mejor solución, se hace esta selección infactible, agregando la siguiente proposición lógica:

$$\text{A lo más 2 de } \{Y[1], Y[4], Y[8]\}$$

Resolviendo el problema disjuntivo lineal correspondiente se obtiene la secuencia indirecta que se muestra enla Fig. 5.11, con un costo anualizado de \$ 4,102 x 10^3/año. Es interesante notar de las Figs. 5.9, 5.10 y 5.11, que la secuencia óptima de la Fig. 5.9 es la que tiene el flujo total de masa más bajo (2000 kgmol/hr), lo cual es consistente con la heurística de seleccionar la secuencia con el mínimo flujo total de masa. Sin embargo,

note que la tercera mejor solución tiene un flujo total de masa (2250 kgmol/hr) más bajo que la segunda mejor solución (2400 kgmol/hr).

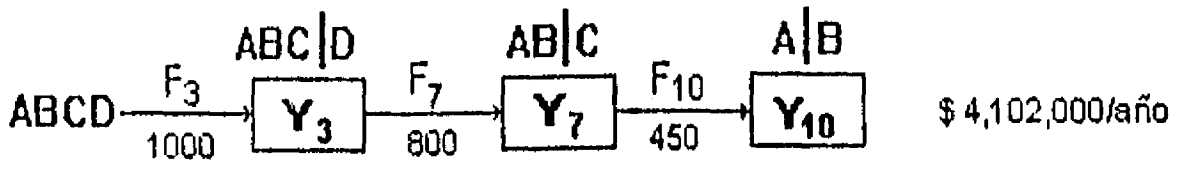


Fig. 5.11 *Tercera secuencia óptima*

1.3.1 Modelo lineal mixto-lógico

Basados en el ejemplo de la sección anterior, se puede generalizar fácilmente el modelo de programación lineal mixto-lógico para sintetizar secuencias óptimas de columnas de destilación para cualquier mezcla de *n* componentes que va a ser separada en componentes puros.

Primeramente, se definirán los siguientes conjuntos de índices, que se ilustrarán con los datos del ejemplo de la sección anterior.

a) IP = {m | m es un producto intermedio}
   Ej. IP = {(ABC),(BCD),(AB),(BC),(CD)}
b) COL = {k | k es una columna de destilación en la superestructura}
   Ej. COL = {1,2, … , 9,10}
c) FS_F = {Columnas k que tienen como alimentación la mezcla inicial}
   Ej. FS_F = {1,2,3}
d) FS_m = {Columnas k que producen producto intermedio m}
   Ej. Para m = (BCD), FS_m = {4,5}

A partir de estos conjuntos, la función objetivo de la Ec. (1.18) y las restricciones de las Ecs. (1.8) a la (1.16), pueden ser entonces escritas en el siguiente modelo lineal mixto-lógico:

$$\text{Min COSTO} = \sum_{k\in COL} [\,(Z_k + \beta_k F_k) + (c_H + c_C)Q_k\,]$$

$$\text{s.t.} \quad \sum_{k\in FS_F} F_k = F_{TOT}$$

$$\sum_{k\in FS_m} F_k - \sum_{k\in PS_m} \xi_k F_k = 0 \qquad m\in IP$$

$$Q_k - K_k F_k = 0 \qquad k\in COL \qquad (19)$$

$$\begin{bmatrix} Y_k \\ Z_k = \text{Costo Fijo } k \\ F_k > 0 \end{bmatrix} \quad \vee \quad \neg Y_k, \begin{bmatrix} F_k = 0 \end{bmatrix} \qquad k\in COL$$

$$0 \le F_k \le F_{TOT},\ Q_k \ge 0 \qquad k\in COL$$

donde F_{TOT} es el flujo de la mezcla inicial y ξ_k son las fracciones de recuperación del compuesto intermedio *m*, en la columna *k*.

Este problema, fué planteado inicialmente, en forma mixta-entera lineal por Andrecovich y Westerberg (1985), con la variante de que el conjunto de disyunciones $Y_k \vee \neg Y_k$, estaba representado por el siguiente conjunto de ecuaciones lineales:

$$F_k - U\, y_k = 0 \qquad k \in COL$$

donde $y_k = \{0,1\}$ es la variable binaria que es igual a cero si el equipo no está presente y es igual a uno si el equipo si se encuentra presente en la red seleccionada, y U es un límite superior para los flujos, que por simplicidad puede ser F_{TOT}.

Aunque la función objetivo de costo, es básicamente la misma, se multiplica el costo fijo de cada equipo por la variable binaria de existencia, esto es: $Z_k\, y_k$.

La información lógica asociada a la superestructura, fué originalmente incluída en el modelo lineal mixto-entero por Raman y Grossmann (1993) [51] en forma de ecuaciones lineales algebraicas, correspondientes a las proposiciones lógicas planteadas en (1.20), y utilizado alternativamente, dentro del modelo mixto-entero en forma algebraica y simbólica por los mismos autores en (1994) [52]. Finalmente, Hooker *et al.* (1994) [37] definieron como cortes lógicos la expresión algebraica entera, en términos de las variables binarias y_i, del conjunto de ecuaciones "*A lo más*..." planteadas en (1.21). Esta forma de representación minimiza el tamaño del problema a resolver en cada nodo del arbol de ramificación y acotamiento.

Note que el tamaño de ambos modelos de programación lineal, el mixto-lógico y el mixto-entero, es función del número de columnas de separación en la red representada en la superestructura, y no del número de secuencias utilizadas para separar la mezcla original en componentes puros.

Tanto en el modelo lineal mixto-lógico presentado en la sección previa como en el modelo lineal mixto-entero utilizado por Westerberg, Raman, Grossmann y Hooker, se asume que el enfriamiento en los condesadores y el calentamiento en los rehervidores sería realizado por corrientes de servicio (*i.e.* agua de enfriamiento y vapor, respectivamente). Sin embargo, a veces es deseable realizar integración de calor en la secuencia de columnas de destilación, ya que la energía, más que el costo de capital, tiende a ser el costo dominante del proceso.

Las dos principales alternativas que pueden ser consideradas para la integración de calor entre columnas de destilación, se muestran en las Figs. 5.12 y 5.13. En la Fig. 5.12, se tiene una secuencia indirecta para la separación de (ABC). Aquí, la primera columna opera a alta presión, de tal forma que su condensador puede ser usado como fuente para el rehervidor en la segunda columna que opera a baja presión. En la Fig. 5.13, la separación

de A y B se realiza con dos columnas: una a baja presión y otra a alta presión, de tal manera que que el condensador de la última puede ser usado como una fuente de calor del rehervidor de la primera. En otras palabras, la Fig.5.13 representa una alternativa para la integración de calor a través de multiefecto, para la misma tarea de separación, mientras que la Fig. 5.12 representa una alternativa de intercambio de calor entre conlumnas que realizan diferentes tareas de separación. En ambos casos, es claro que la selección de presión en las columnas de destilación, es de crítica importancia.

1.3.2 Modelamiento de los efectos de la presión

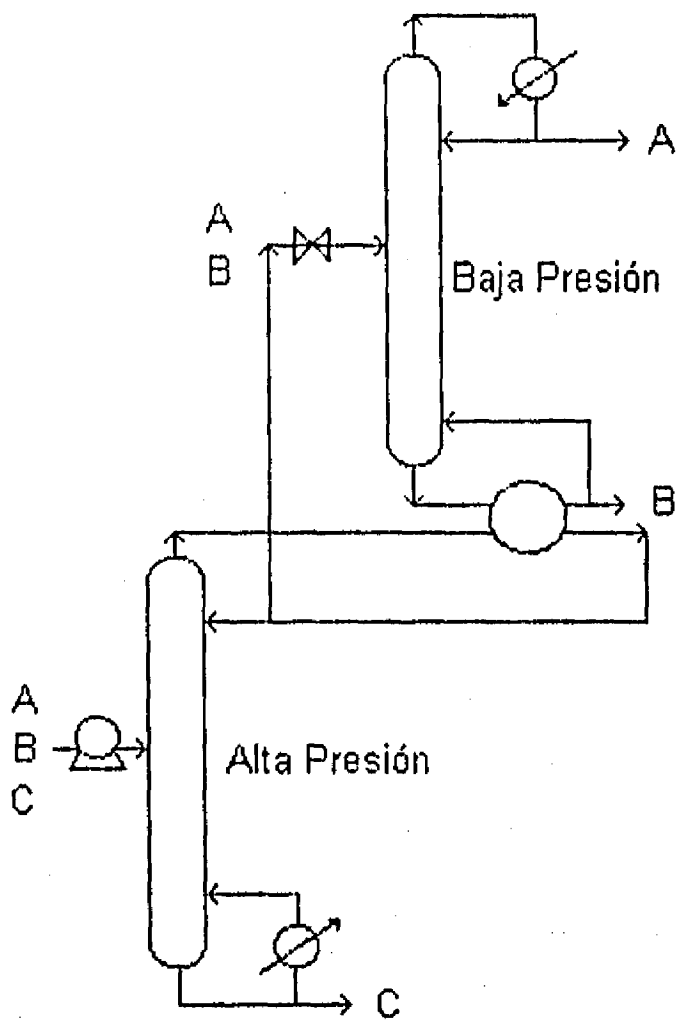


Fig.5.12*Integración de calor entre tareas*

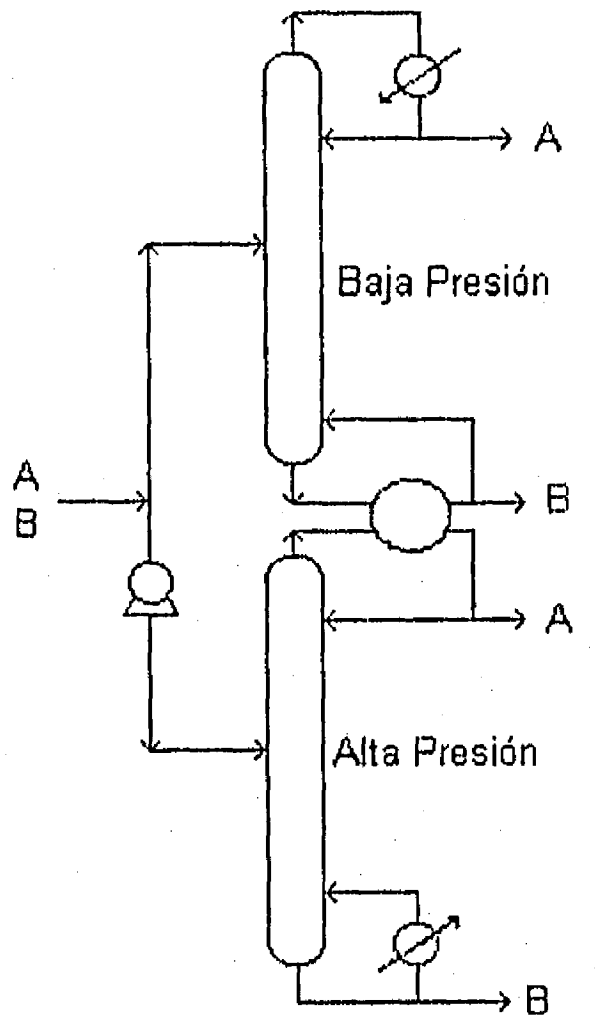


Fig.5.13*Multiefecto de la integración de calor*

Tratando explícitamente la presión de las columnas en los modelos que las describen, se introducen términos no-lineales porque, con el objeto de considerar los efectos en la temperatura, se necesitan calcular las temperaturas de burbuja y de rocío, como se muestra en la Fig. 5.14. Aunque es posible incluir explícitamente estas ecuaciones en un modelo de optimización no lineal, se hará la suposición de que ΔT_{RC}, la diferencia entre la temperatura del rehervidor (punto de rocío) y la temperatura del condensador (temperatura de burbuja), es una constante que es independiente de la presión de la columna. Esta constante, sería típicamente calculada por un método corto a una presión nominal. Además, se asumirá, que las columnas tendrán condensadores y rehervidores totales, como en la Fig. 5.14. Más allá, se asumirá que las temperaturas son constantes para las corrientes del destilado y del fondo.

En adición, se asumirá que los coeficientes de carga de calor K_k son constantes e independientes de las temperaturas. De esta manera es posible modelar el problema de síntesis de secuencias de columnas de destilación con integración de calor, aumentando en el modelo lineal mixto-lógico planteado en (1.22), restricciones adicionales que sólo dependerán de las temperatura de los condensadores y de los rehervidores, lo cual puede ser realizado a través de dos diferentes tipos de modelo: (*a*) Modelos con temperaturas continuas y (*b*) Modelos con temperaturas discretas.

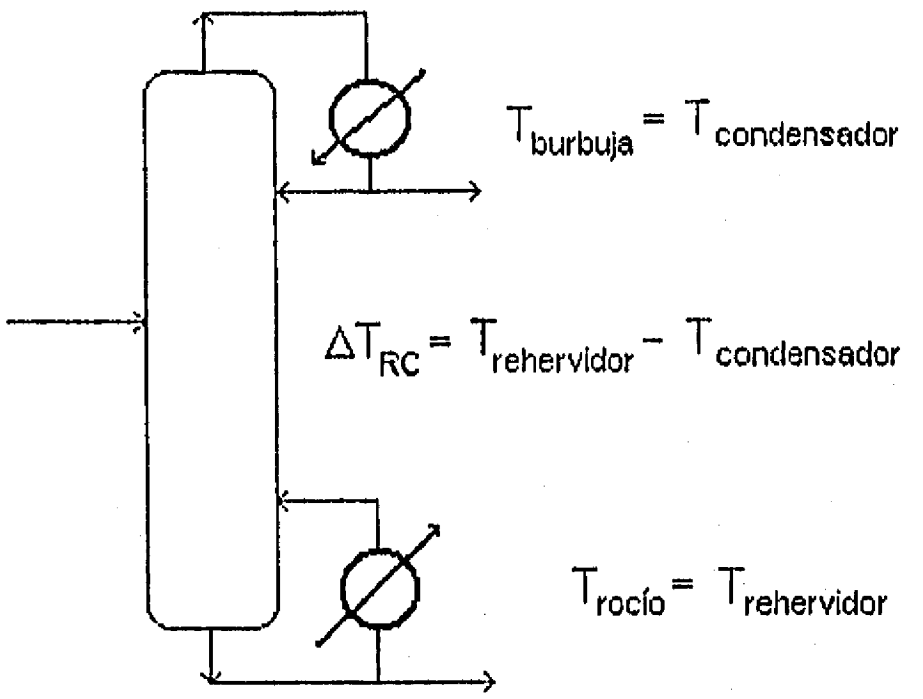


Fig. 5.14 *Modelando cambios de presión a través de* ΔT_{RC}

1.3.3 Modelo lineal mixto-lógico con Temperaturas Continuas

Se asumirá, para el modelo de esta sección, que la integración de calor será considerada entre diferentes tareas de serapración. Entonces, se excluirá la posibilidad de sintetizar columnas multiefecto, y por lo tanto, la superestructura, en términos de las columnas, será la misma que en las secciones anteriores. También se asumirá que sólo existe una corriente de servicio para el calentamiento y una para el enfriamiento (*i.e.* vapor y agua de enfriamiento). Finalmente, con el objeto de retener la linealidad del modelo, se asumirá que el costo fijo de la columna varía sólo con la presión, o equivalentemente, con la temperatura en el condensador (Raman y Grossmann, 1994) [52]. La fórmula general de costos que será considerada es la siguiente:

$$\text{Costo Fijo} = \alpha \left[1 + \gamma \left(\frac{T_C - T_{AE} - EMAT}{T_{AE} + EMAT} \right) \right] \qquad (1.23)$$

donde Tc es la temperatura del condensador, T_{AE} es la temperatura del agua de enfriamiento, EMAT es el acercamiento mínimo entre las temperaturas de los intercambiadores de calor y α y γ son coeficientes de costo. Note que $T_C \geq T_{AE} + EMAT$. También, si la columna no se selecciona, el costo fijo tiene que ser cero. Esto puede ser

realizado, introduciendo una nueva variable μ_k que representa los costos fijos que van a ser minimizados en la función objetivo

$$\begin{aligned} &\text{Si } \neg Y_k: \quad \mu_k = 0 \\ &\text{Si } \quad Y_k: \quad \mu_k = \alpha \left[1 + \gamma \left(\frac{T_C - T_{AE} - EMAT}{T_{AE} + EMAT} \right) \right] \end{aligned} \qquad (1.24)$$

En adición a las ecuaciones de balance de masa en (1.22), se necesitan restricciones que representen el intercambio de calor. Si T_R^k y T_C^k son las temperaturas del rehervidor y del condensador de la columna k, ΔT_{RC} es la diferencia de temperaturas en la columna, EMAT es el acercamiento mínimo de temperaturas en el intercambiador y T_V y T_A son las temperaturas del vapor y del agua de enfriamiento, se pueden aplicar las tres siguientes restricciones:

$$\begin{aligned} T_R^k &= T_C^k + \Delta T_{RC} \\ T_R^k &\le T_V - EMAT \\ T^k &\ge T + EMAT \qquad k \in COL \end{aligned} \qquad (1.25)$$

La primera, simplemente establece la relación entre las temperaturas del condensador y del rehervidor, mientras que las dos desigualdades proveen los límites de estas temperaturas en términos de la temperatura del vapor y del agua de enfriamiento.

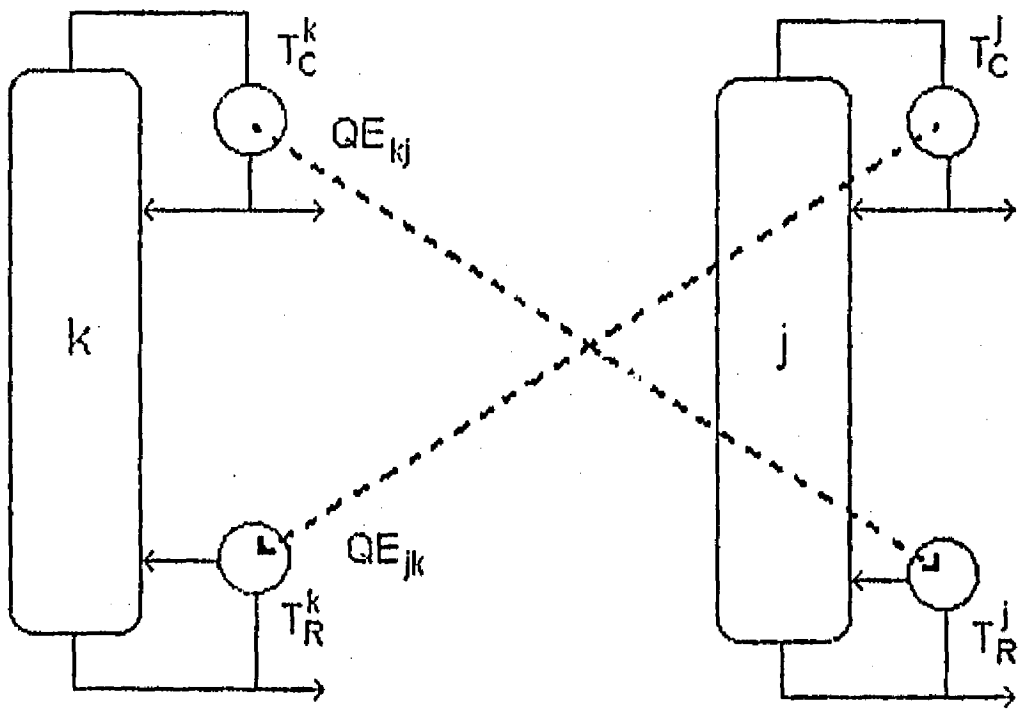


Fig. 5.15 *Definición de variables para la integración de calor entre diferentes tareas de separación*

Para considerar los intercambios potenciales de calor, se definirá la variable QEX_{kj} para denotar la cantidad de calor intercambiada entre las columnas k y j (ver Fig. 13), y también se definirá Z_{kj}, como variable de existencia, esto es:

Z_{kj}: El condensador de la columna *k* proporciona calor al rehervidor de la columna *j*
$\neg Z_{kj}$: No existe intercambio de calor entre las columnas.

Entonces, las dos siguientes restricciones condicionales aplican, donde Ω_k es un límite válido para QEX_{kj}:

$$\begin{array}{lll} \text{Si } Z_{kj}: & QEX_{kj} \le \Omega_k & \text{Si } \neg Z_{kj}: \quad QEX_{kj} = 0 \\ & T_C^k \ge T_R^k + EMAT & k, j \in COL,\ j \ne k \end{array} \qquad (1.26)$$

Finalmente, los balances de calor deben ajustarse para abarcar los intercambios de calor QEX_{kj} y las cargas de calor de enfriamientoy calentamiento QW_k y QS_k aplicadas para satisfacer el calor Q_k de cada columna. Entonces, las siguientes ecuaciones aplican:

$$\sum_{j \in COL \setminus k} QEX_{kj} + QW_k = Q_k \qquad k \in COL \qquad (1.27)$$

$$\sum_{j \in COL \setminus k} QEX_{kj} + QS_k = Q_k \qquad k \in COL$$

Definiendo la función objetivo en términos de los costos de inversión de las columnas de destilación, como en (1.22), y los costos de las corrientes de servicio, y considerando las restricciones en (1.22) y (1.24) a (1.27), el modelo lineal mixto-lógico que describe el problema viene dado por:

$$\text{Min COSTO} = \sum_{k \in COL} [\,(Z_k \mu_k + \beta_k F_k) + (c_H + c_C) Q_k\,]$$

$$\text{s.t.} \quad \sum_{k \in FS_F} F_k = F_{TOT}$$

$$\sum_{k \in FS_m} F_k - \sum_{k \in PS_m} \xi_k F_k = 0 \qquad m \in IP$$

$$Q_k - K_k F_k = 0$$

$$\sum_{j \in COL \setminus k} QEX_{kj} + QW_k = Q_k$$

$$\sum_{j \in COL \setminus k} QEX_{kj} + QS_k = Q_k \qquad (1.28)$$

$$T_R^k = T_C^k + \Delta T_{RC}$$

$$T_R^k \le T_V - EMAT$$

$$T^k \ge T + EMAT$$

$$\begin{bmatrix} Y_k \\ Z_k = \text{Costo Fijo } k \\ F_k > 0 \\ \mu_k = \alpha \left[1 + \gamma \left(\dfrac{T_C - T_{AE} - EMAT}{T_{AE} + EMAT} \right) \right] \end{bmatrix} \vee \begin{bmatrix} \neg Y_k \\ F_k = 0 \\ \mu_k = 0 \end{bmatrix}$$

$$\begin{bmatrix} Z_{kj} \\ QEX_{kj} \le \Omega_k \\ T_C^k \ge T_R^k + EMAT \end{bmatrix} \vee \begin{bmatrix} \neg Z_{kj} \\ QEX_{kj} = 0 \end{bmatrix}$$

$$0 \le F_k \le F_{TOT},\ Q_k \ge 0 \qquad k \in COL$$

Note que en esta formulación. el costo de los intercambiadores no está directamente contabilizado. La opción más simple será la de agregar un costo fijo que estaría sociado con cada variable de existencia Z_{kj}. La otra opción podría ser agregar la ecuación no lineal de área, con lo que el modelo en (1.28), se convertiría en no-lineal. Suponiendo que se resuelve el modelo tal cual está planteada en (1.28), se tiene que el costo computacional puede ser alto, debido principalmente al gran número de disyunciones que se tienen en el modelo (Y_k, Z_{jk}): sin embargo, se puede agilizar mucho el modelo si aparte de agregar las proposiciones lógicas derivadas del diagrama original, agregamos las siguientes proposiciones lógicas:

(a) El número de columnas en el modelo no puede exceder el número de componentes (m) menos uno:

$$\text{A lo más } m \text{ de } \{Y_1, Y_2, \ldots, Y_k\} \quad , \qquad k \in COL \qquad (1.29)$$

(b) Si una columna no es seleccionada, no puede intercambiar calor con las demás columnas:

$$Z_{jk} \Rightarrow Y_k \qquad \text{que en CNF, sería:} \quad \neg Z_{jk} \vee Y_k \qquad (1.30)$$
$$Z_{kj} \Rightarrow Y_k \qquad\qquad \neg Z_{kj} \vee Y_k \qquad k, j \in COL,\ j \neq k$$

(c) La columna j proporciona calor a la columjna k, o viceversa:

$$\neg Z_{jk} \vee \neg Z_{kj} \qquad (1.31)$$

Debe notarse que aunque los tres tipos de proposiciones lógicas planteadas arriba son redundantes, ayudan a limitar el número de conjunciones a ser exploradas, y por lo tanto limitan la enumeración necesaria en el árbol de ramificación y acotamiento.

Finalmente, es claro que una vez que la solución óptima ha sido encontrada con el modelo en (1.28), las presiones de operación de las columnas pueden ser simplemente calculadas de las temperaturas de los condensadores. Una versión no lineal mixta-entera (MINLP) de este modelo se desarrolló por Paules y Floudas en (1988).

1.3.4 Modelo lineal mixto-lógico con Temperaturas Discretas

En esta sección se presentará un modelo alternativo lógico lineal para sintetizar secuencias de columnas de destilación con integración de calor (la versión mixta-entera lineal de este modelo fué originalmente presentada por Andrecovich y Westerberg, 1985), basada en la discretización de las temperaturas. Aunque en un principio, esto puede parecer más restrictivo, se verá como la discretización de las temperaturas facilita la consideración de la integración multiefecto, así como la agregación de la integración de calor a través de ecuaciones de transbordo, que eliminan la necesitan de introducir disyunciones para los diferentes acoplamientos.

En principio, se puede resolver el problema de la integración de calor en secuencias de columnas de destilación como sigue. Primero, en la red de la Fig. 5.8, se puede postular un número diferente de columnas candidatas para cada tarea de separación. Este número

sería típicamente el número máximo de columnas que queremos que tengan separación multiefecto. Así, por ejemplo, para cada separación A/BCD se pueden postular dos diferentes columnas, cada una operando a una presión fija diferente. Los condensadores pueden ser tratados como corrientes calientes y los rehervidores como corrientes frías, ambos con temperaturas fijas. Sólo sus corrientes de flujos estarían desconocidas. Basado en este esquema de discretización para las temperaturas, simplemente, se puede agregar a nuestro modelo lógico lineal en (1.22), las ecuaciones del modelo de transbordo necesarias. Además, el hecho de que los flujos sean desconocidos, no representa ningún problema para conservar la linealidad del modelo.

Con el objeto de postular la superestrucutra correspondiente, se considera una versión simplificada del procedimiento sugerido por Andrecovich y Westerberg (1985). Habiendo determinado ΔT_{RC} para cada tarea de separación, el procedimiento para seleccionar las columnas candidatas operando a niveles discretos de temperatura, es el siguiente:

a) Se define el rango permisible de temperaturas para la integración de calor:

Temperatura más alta = Temperatura más caliente de las corrientes de servicio calientes - ΔT_{min}.
Temperatura más baja = Temperatura más baja de las corrientes de servicio frías + ΔT_{min}

b) Dentro del rango permisible de temperaturas, para cada tarea de separación, se crea una pila de columnas del fondo hasta el tope con cambios ΔT_{RC} de temperatura y con una diferencia ΔT_{min} entre columnas sucesivas; si la pila no alcanza el tope por más de ΔT_{min}, hay que crear una segunda pila de columnas del tope hasta el fondo.

Para ilustrar más claramente este procedimiento, se considera un ejemplo de tres componentes en la Tabla 5.2. Como se puede ver en la Fig. 5.16, el rango permisible de temperaturas, viene dado de 330°K a 540°K. Los 330°K son obtenidos agregando 10°K a la temperatura de salida del agua de enfriamiento, y los 540°K, substrayendo 10°K de la temperatura de la corriente de vapor a alta presión.

Tabla 5.2 Datos para la mezcla de 3 componentes

TAREA	ΔT_{RC}(°K)	CORRIENTES DE SERVICIO	
A/BC	100	Vapor de alta presión:	550°K
AB/C	80	Vapor de baja presión:	460°K
A/B	50	Agua de enfriamiento:	300-320°K
B/C	50		EMAT = 100°K

Considere la tarea de separación A/BC, la cual tiene un ΔT_{RC} de 1000°K como se puede ver en la Fig. 5.16. Empezando a 330°K, se considera primero una columna con una temperatura del condensador de 330°K y una temperatura del rehervidor de 430°K. Con un EMAT de 10°K; se apila, en el tope de esta columna, una cuya temperatura del condensador está a 440°K y del rehervidor a 540°K. En este caso, existen dos columnas que se pueden ajustar exactamente dentro del rango 330-540°K, como puede verse en la Fig. 5.16. En estas dos columnas, el condensador de la columna 1, a 440°K puede,

potencialmente, intercambiar calor con el rehervidor de la columna 2 a 430°K. Para la tarea de separación AB/C, se tiene un ΔT_{RC} de 80°K. En este caso, la primera columna tiene el condensador a 330°K y el rehervidor a 410°K. Con el EMAT de 10°K, se apila en el tope de la columna, una cuyo condensador está 420°K y cuyo rehervidor a 500°K. Ya que se perdió el tope del rango por 40°K, ahora se puede crear una segunda pila del tope hasta el fondo. La primera columna, en la segunda pila tiene el rehervidor a 540°K y el condensador a 460°K; la segunda columna tiene el rehervidor a 450°K y el condensador a 370°K. Entonces, se postularán cuatro columnas para esta tarea de separación, como puede verse en la Fig. 5.16.

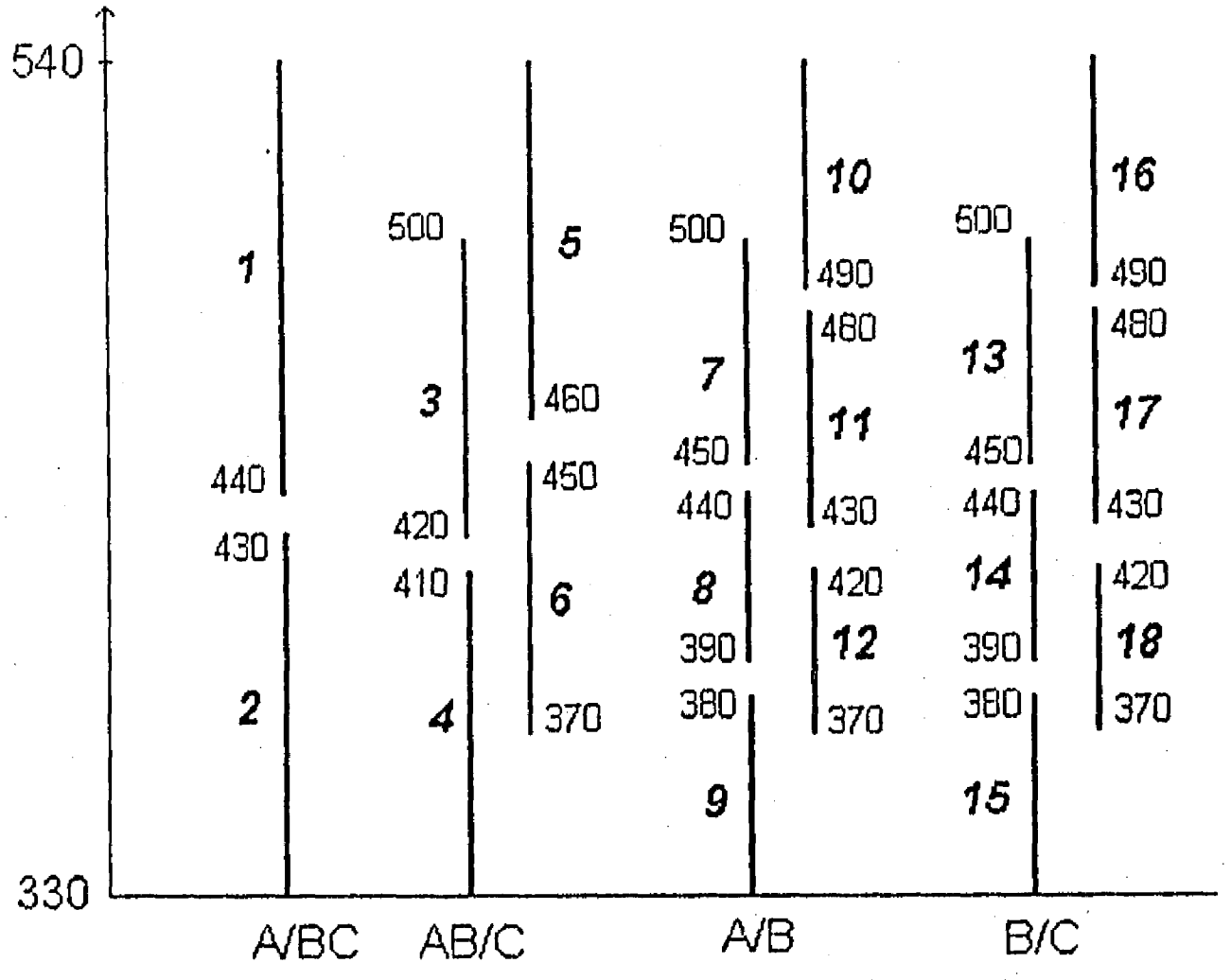


Fig. 5.16 *Columnas potenciales para la integración de calor multiefecto*

Para las tareas de separación A/B, B/C, ambas de las cuales tienen un ΔT_{RC} de 50°K, el procedimiento es completamente análogo al mencionado arriba. En cada caso, se obtienen dos pilas de 6 columnas, como se ve en la Fig. 5.16.

De la Fig. 5.16, se puede ver que a través del esquema de discretización se considerará un total de 18 columnas potenciales. La presión de operación de estas columnas puede ser obtenida haciendo cálculos de punto de burbuja para las temperaturas del condensador.

Asumiendo que se ha determinado el conjunto discreto potencial de columnas, como puede verse en la Fig. 5.16, lo que sigue es considerar la representación de la integración de calor entre estas columnas. En vez de considerar todos los intercambios de calor individuales posibles entre todos los condensadores y rehervidores, como se hizo para el modelo anterior, se analizará la integración de calor en el modelo lineal lógico a través de una cascada de calor. Esto es, tratando los condensadores como corrientes calientes y los

rehervidores como corrientes frías, se puede construir una cascada de calor basada en las temperaturas de estas corrientes. Ya que estas temperaturas pueden ser asumidas como constantes, es conveniente representar los intervalos de temperatura a temperaturas constantes. De esta forma, basados en las temperaturas de los rehervidores y de los condensadores de la Fig. 5.16, y de las diferentes corrientes de servicio, se puede construir una cascada de calor como se muestra en la Fig. 5.17. En el lado izquierdo, se tienen como entradas, el calor de los condensadores Q_k, y el calor de la corriente de vapor de baja presión, Q_{BP}. De el lado derecho, se tienen como salidas, los calores de los rehervidores, Q_k. En el tope de la cascada, se tiene como entrada el calor de la corriente de vapor a alta presión, Q_{AP}, y en la salida del fondo, el calor del agua de enfriamiento, Q_{AE}.

CALIENTE(Condensadores)		FRIO (Rehervidores)
	Q_{AP}	
	550	540	Q1,Q5,Q10,Q16
	510	500	Q3,Q7,Q13
Q10,Q16	490	480	Q11,Q17
QBP,Q5	460	450	Q6
Q7,Q13	450	440	Q8,Q14
Q1	440	430	Q2
Q11,Q17	430	420	Q12,Q18
Q3	420	410	Q4
Q8,Q14	390	380	Q9,Q15
Q6,Q12,Q18	380	360	
Q2,Q4,Q9,Q15	330	320	
	Q_{AE}	

Fig. 5.17 *Flujos de calor para columnas potenciales en la Fig. 5.16*

Note que todas las cargas de calor en la Fig. 5.17 son desconocidas. Sin embargo, esto no aumenta dificultad al modelo, ya que podemos realizar balances de calor alrededor de cada intervalo de temperatura, con una ecuación análoga al modelo de transbordo usado para la integración de calor. Esto es, para cada intervalo m = 1,2,..., M se puede escribir la ecuación,

$$R_m - R_{m-1} - \sum_{i \in CS_m} Q^i_{CS} + \sum_{j \in FS_m} Q^j_{FS} - \sum_{k \in CS_m} Q_k + \sum_{\in FS_m} Q_k = 0 \qquad (1.32)$$

donde R_m es el calor residual que sale del intervalo m; Q^i_{CS}, Q^j_{FS} son las cargas de calor de las corrientes de servicio calientes y frías en el intervalo m, y Q_k son las cargas de calor de las columnas. Son los conjuntos de columnas cuyas temperaturas del condensador y del rehervidor coinciden con las temperaturas de intervalo. De esta manera, incluyendo (1.32) en el modelo lógico lineal (1.22), se puede asegurar la máxima integración de calor en las columnas de destilación, ya que las cargas de calentamiento y de enfiriamiento serán incluídas como costos de operación en la función objetivo.

Basados en el esquema de discretización de la sección anterior, se puede desarrollar una superestructura de esta red, similar a la de la Fig. 5.8. Como ejemplo, si asumimos que

tenemos una mezcla ternaria (ABC), para la cual la discretización va a proporcionar dos columnas para cada tarea (A/BC), (AB/C), (A/B), (B/C). Podemos, simplemente duplicar las columnas en la superestructura como se ve en la Fig. 5.18. Note que aquí, todavía estamos asignando a cada columna un flujo de alimentación, y una variable de existencia Y_i. A dicha superestructura, se puede asignar un conjunto de índices similar al de la estructura de la Fig. 5.8.

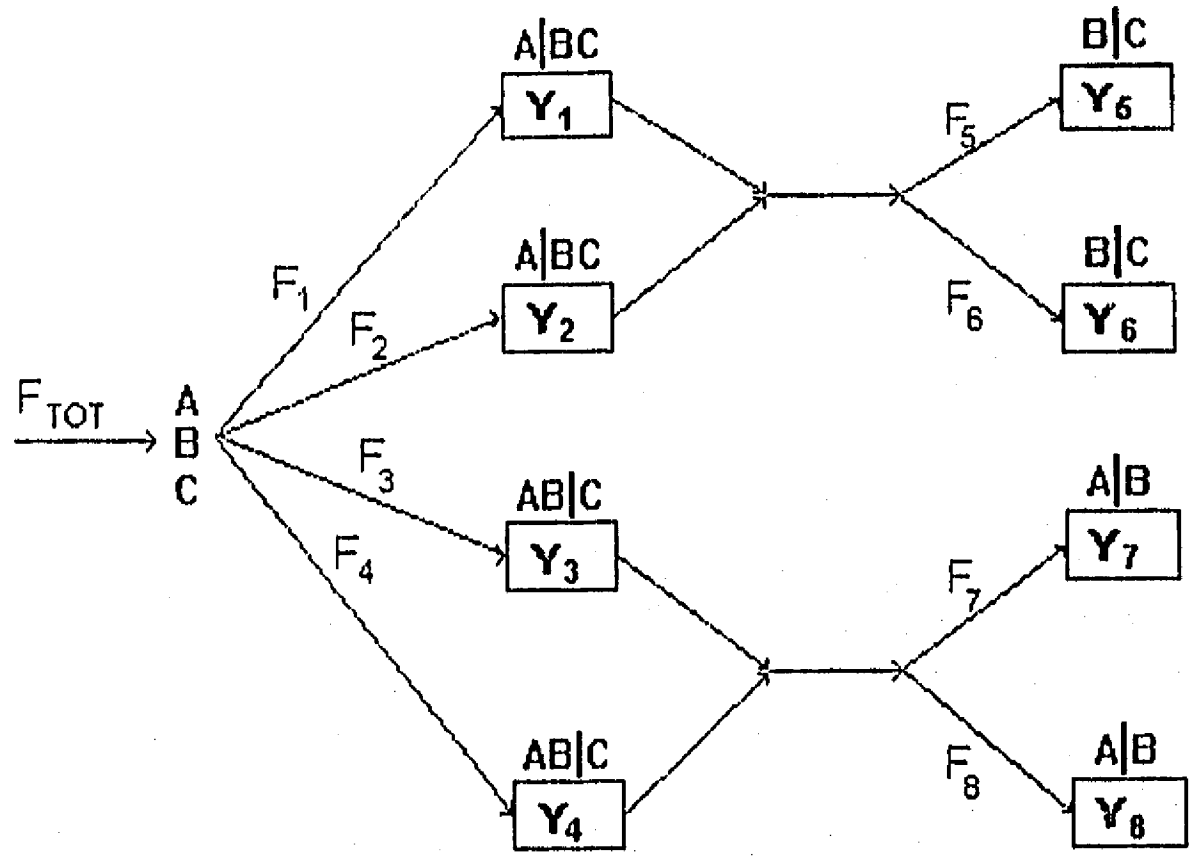


Fig. 5.8 *Superestructura y variables para una mezcla de tres componentes con dos columnas por cada tarea de separación*

Reemplazando las cargas de las corrientes de servicio en la función objetivo del modelo en (1.22), y agregando las ecuaciones de transbordo en (1.32) para la integración de calor, el modelo lógico lineal que describe esta superestructura será el siguiente:

$$\text{Min COSTO} = \sum_{k \in COL} [\,(Z_k + \beta_k F_k) + (c_H + c_C)Q_k\,]$$

$$\text{s.a} \quad \sum_{k \in FS_F} F_k = F_{TOT}$$

$$\sum_{k \in FS_m} F_k - \sum_{k \in PS_m} \xi_k F_k = 0 \qquad m \in IP$$

$$Q_k - K_k F_k = 0 \qquad k \in COL \qquad (1.33)$$

$$\begin{bmatrix} Y_k \\ Z_k = \text{Costo Fijo } k \\ F_k > 0 \end{bmatrix} \quad \vee \quad \begin{bmatrix} \neg Y_k, \\ F_k = 0 \end{bmatrix} \qquad k \in COL$$

$$0 \le F_k \le F_{TOT},\ Q_k \ge 0 \qquad k \in COL$$

1.4 Variables Semicontinuas

Los problema de síntesis de procesos pueden involucrar variables semicontinuas, como se describe en [52]. Estas son variables cuyos valores pueden caer dentro de ciertos intervalos. Por ejemplo, una entrada a una red de proceso puede consistir de alimentaciones obtenidas de un conjunto de proveedores, cada cual puede proveer o nada o una cantidad dentro de cierto rango. Tomando todas las posibles sumas de estos intervalos se llega a los intervalos dentro de los cuales debe estar el volumen de la entrada total. Lo mismo puede hacerse con las salidas, si suponemos que estas van a dar a determinados mercados que también pueden comprar o nada o una cantidad dentro de cierto rango.

Un flujo de variable semicontinua x_{ij} debe caer en uno de los intervalos $[a_t,b_t]$ para $t = 0, \ldots, T$. La representación disyuntiva más directa es,

$$\bigvee_{t=0}^{T} y_t$$
$$y_t \rightarrow (a_t \leq x_{ij} \leq b_t).$$

La proposición y_t es verdadera cuando x_{ij} cae en el intervalo t. Una formulación alternativa que demostró reducir el número de nodos en el árbol de búsqueda, es la siguiente

$$\begin{aligned} & a_0 \leq x_{ij} \leq b_T \\ & y_t \vee y'_t, \ t = 1, \ldots, T \\ & y_t \rightarrow (x_{ij} \geq a_t) \\ & y'_t \rightarrow (x_{ij} \leq b_{t-1}) \end{aligned} \qquad (1.34)$$

Aquí, y_t es verdadero cuando el volumen del flujo cae en el intervalo t o en otro mayor, y y'_t es verdadero cuando cae en el intervalo t - 1 o en uno menor.

Una representación lineal mixta-entera es

$$\begin{aligned} & x_{ij} \geq a_0 + \sum_{t=1}^{T} a_t y_t \\ & x_{ij} \leq b_0 + \sum_{t=1}^{T} b_t y_t \\ & \sum_{t=1}^{T} y_t \leq 1 \\ & y_t \in \{0,1\}, \ t = 1, \ldots, T \end{aligned} \qquad (1.35)$$

2. Calendarización de procesos con transferencia cero para plantas de proceso múltiple

Los procesos en etapas o por lotes (tipo *batch*) son usados en la manufactura de productos químicos y farmacéuticos, comida y cierto tipo de polímeros (Reeve, 1992). Ya que, comúnmente, los volúmenes de producción son bajos, las plantas usadas para estos

procesos, frecuentemente tienen instalaciones multiproducto, en las cuales, varios productos comparten las mismas piezas de equipo. Esto requiere que la producción en estas plantas sea calendarizada. Específicamente, uno tiene que decidir el orden de producción de estos productos y el tiempo disponible para cada uno de ellos. Esto implica que en la etapa de diseño, uno tiene que anticipar cómo se va a calendarizar la producción debido al impacto ecónomico que esta anticipación puede representar en el costo final de producción.

Cuando un proceso por lotes es usado para producir dos o más productos, se pueden tener dos tipos límites de plantas: (*a*) plantas de flujo, donde todos los productos requieren que todas las etapas sigan la misma secuencia de operaciones, y (*b*) plantas a destajo donde no todos los productos requieren de todas las etapas y/o de la misma secuencia de operaciones (ver Fig. 5.9). Mientras más similitud exista entre los productos que se fabrican, más cerca estará la planta real del modelo de planta de flujo; y viceversa, mientras más diferencia exista entre los productos, más se acercará el modelo al de planta a destajo. Debe notarse que las plantas de flujo, frecuentemente son llamadas "plantas producto múltiple", mientras que las plantas a destajo se conocen como "plantas de propósito múltiple".

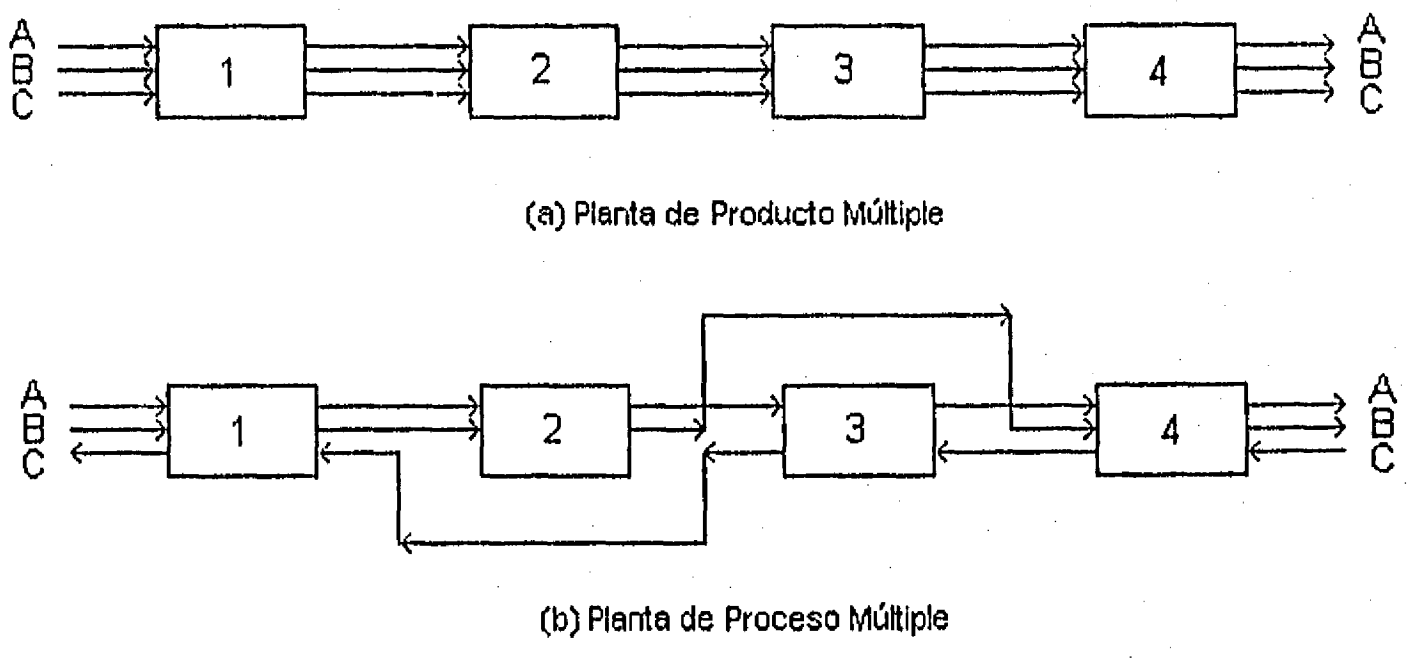


Fig. 5.9 *Plantas de Producto y de Propósito Múltiple*

Otro punto importante en las plantas de flujo es el tipo de campaña de producción que se usa para producir un número específico de etapas para los diferentes productos. Para ilustrar este punto, considere la producción de tres etapas para los productos A y B en una planta de dos etapas. Los tiempos de proceso vienen dados en la Tabla 5.3.

	Tiempo de Proceso (hrs).	
	Etapa 1	Etapa 2
A	5	2
B	2	4

Tabla 5.3 *Tiempos de proceso para una planta de dos productos*

Como puede verse en la Fig. 5.10 (*a*), una opción es la de usar campañas de producto en las cuales, todas las etapas de un producto dado, son producidas antes de cambiar a otro producto. La otra opción, mostrada en la Fig. 5.10 (*b*) es la de usar campañas de producto mezclado, en las cuales las diferentes etapas son producidas de acuerdo a alguna secuencia seleccionada (*i.e.* ABABAB). Note que la duración total de la campaña en la Fig. 5.10 (*a*) es de 29 horas, mientras que para la Fig. 5.10 (*b*) es de 25 horas. El tiempo de ciclo para la secuencia AAABBB en la Fig. 5.10 (*a*) es de 25 horas y para ABABAB en la Fig. 5.10 (*b*) es de 21 horas. Esto podría sugerir que las campañas de producto mezclado son más eficientes. Sin embargo, este no tiene que ser necesariamente el caso, si los tiempos de limpieza o de cambio de producción que se necesitan, son significativos, cuando uno cambia de un producto a otro. Por ejemplo, en el ejemplo presentado, los tiempos de limpieza son todos de una hora, y puede ser visto en la Fig. 5.11 que la duración total se incrementa de 25 a 30 horas y el tiempo de ciclo, de 21 a 27 horas.

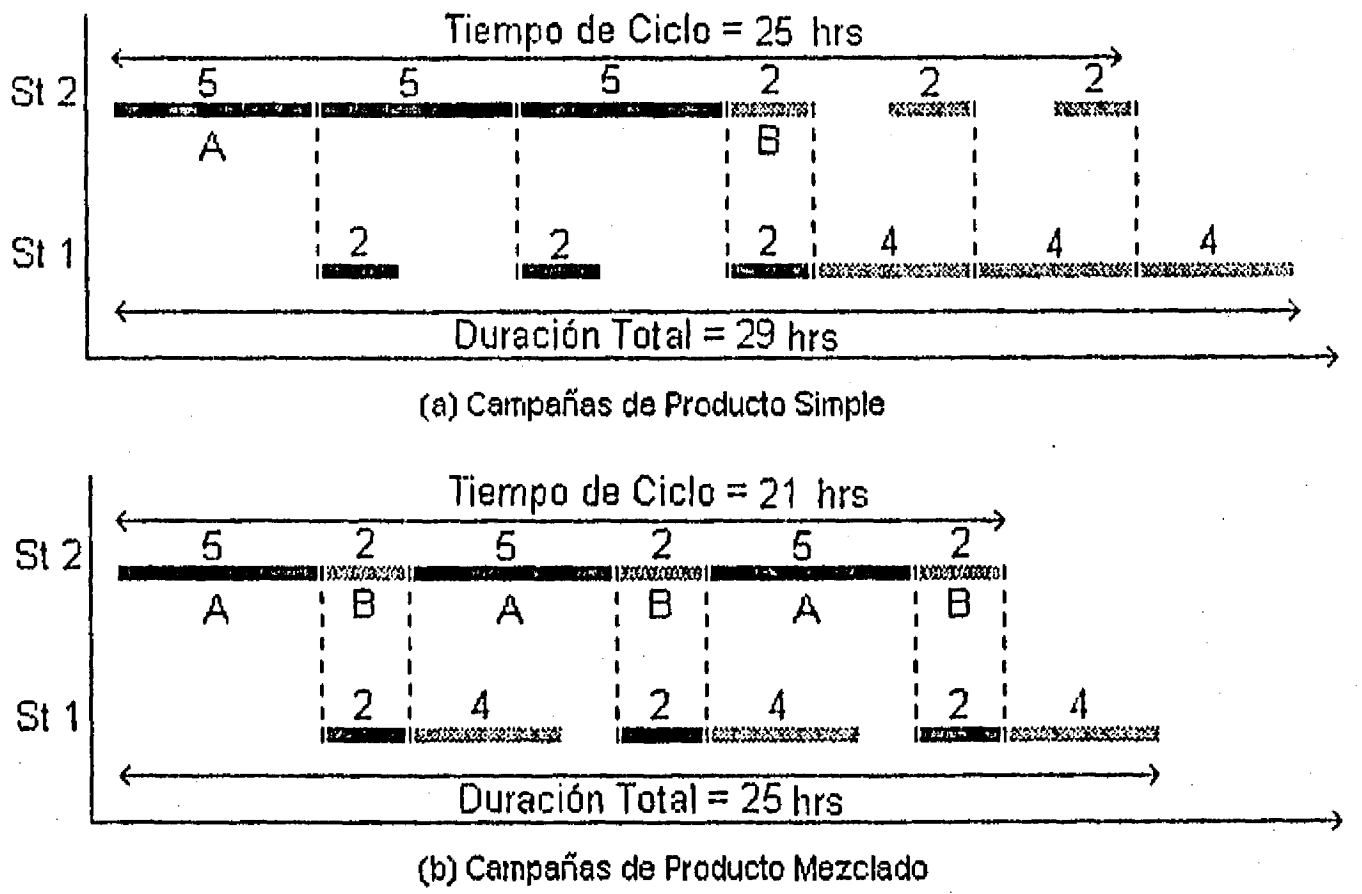


Fig. 5.10 *Calendarización para campañas de producto simple y de producto mezclado*

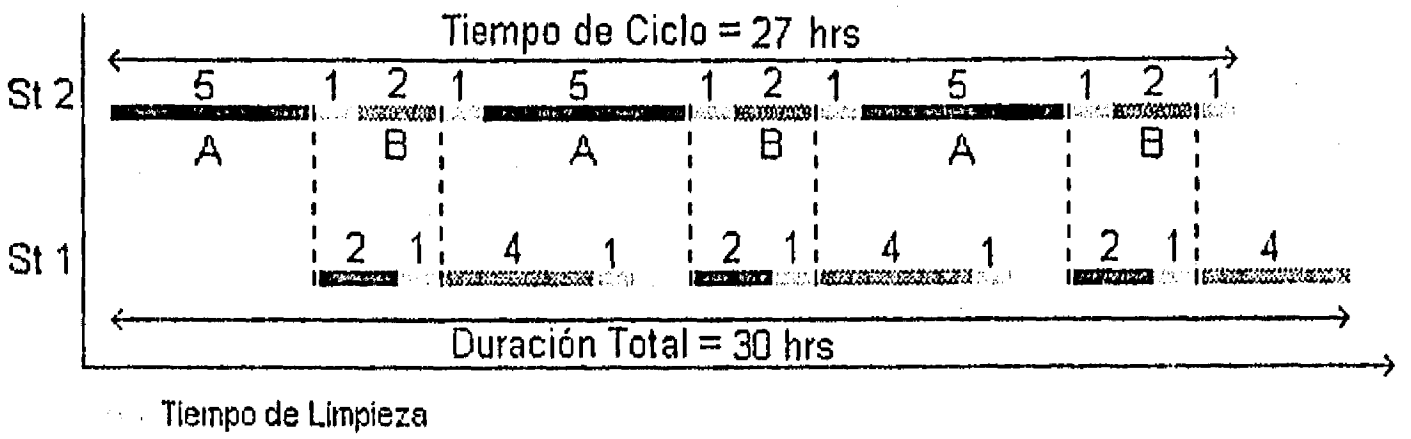


Fig. 5.11 *Efecto del tiempo de limpieza en el tiempo del ciclo*

Se puede notar que para el caso de los procesos en etapas con productos múltiples, generalmente, no es posible obtener expresiones en forma cerrada para los tiempos de los ciclos.

Durante toda esta descripción, hemos asumido que el proceso en cada etapa sería transmitido inmediatamente a la siguiente etapa. Es decir, que existe una transferencia de *espera-cero* o simplemente, *transferencia-cero*, y que es usada comúnmente cuando no existen contenedores de almacenamiento disponibles o cuando el proceso no puede permanecer más tiempo dentro del contenedor en curso, (*i.e.* debido a una reacción química). La *transferencia-cero* es la más restrictiva. En el otro extremo se tiene la opción de tiempo ilimitado de almacenamiento, en el cual se asume que el producto intermedio de cada etapa puede ser almacenado sin que exista un límite en la capacidad de almacenamiento de los contenedores que se usarán. Finalmente, una opción intermedia entre ambas posibilidades es la que considera la posibilidad de almacenar el material dentro del mismo contenedor de esa etapa.

En general, la *transferencia-cero*, requiere del tiempo de ciclo más largo, mientras que la de almacenamiento intermedio ilimitado, del tiempo de ciclo más corto. En la práctica, las plantas, normalmente operan dentro de una mezcla de las tres políticas de transferencia.

A su vez, la adición de tanques de almacenamiento de producto intermedio entre las diferentes etapas o de unidas paralelas de operación fuera del ciclo, pueden incrementar la eficiencia de la utilización del equipo.

Otra cuestión importante en el diseño y operación es la selección del ciclo de producción. La principal decisión involucrada es la fracción de transición entre los tiempos de limpieza y los inventarios. Cuanto más corto es el ciclo de producción, es menor el inventario que necesitamos arrastrar, pero más grande la fracción entre las transiciones.

Los tres niveles principales de decisión y sus correspondientes puntos, son los siguientes:

1. ***Nivel estructural***
   a) *Asignación de las tareas a cada equipo*
   b) *Número de unidades paralelas o de almacenamiento de producto intermedio*
2. ***Nivel de medición***
   a) *Definición del tamaño de los equipos*
3. ***Nivel de calendarización***
   a) *Naturaleza de las campañas de producción, políticas de transferencia*
   b) *Longitud de los ciclos de producción*
   c) *Secuenciación de los productos*

A nivel estructural, la asignación de tareas a los diferentes equipos, es una de las decisiones que puede tener el mayor impacto en la calendarización y en la economía. Sin embargo, la selección de la longitud del ciclo de producción, dada su importancia e impacto en el diseño y la economía del proceso, requiere de una evaluación detallada y de una optimización rigurosa. A su vez, para la mayoría de los problemas, el número de alternativas es muy grande y los costos económicos de la mayoría de las opciones, son, generalmente complejos, por lo que es necesario utilizar un enfoque sistemático de optimización.

Dada su importancia y la posibilidad de emplear modelos de optimización lineal y no lineal en el nivel de calendarización, que nos permitan seleccionar una alternativa óptima dentro de las posibilidades existentes, el primer problema abordado en el diseño de procesos por etapas que puede ser resuelto mediante un modelo de programación lineal, mixta-entera o mixta-lógica, es el de la selección de la longitud del ciclo de producción.

Ya que uno de los problemas de calendarización que ocurre con mucha frecuencia en los procesos químicos y que puede modelarse utilizando modelos lineales de optimización es el problema con transferencia-cero, a continuación se presenta el desarrollo matemático necesario para obtener un modelo de optimización que nos permita minimizar el tiempo de ciclo para este problema mediante un modelo lineal mixto-entero y compararlo contra el modelo lineal mixto-lógico que se propone en esta tesis.

2.1 Modelo lineal mixto-lógico para la calendarización de trabajos con transferencia-cero

Considere el caso de los problemas de calendarización de trabajos en el cual, un conjunto de trabajos $i \in I$ debe ser procesado secuencialmente en una serie de etapas, pero no todos los trabajos requieren pasar por todas las etapas. Esto es, cada trabajo es procesado en un subconjunto de etapas $j \in J(i)$. Además, se asume una transferencia-cero entre las etapas. Un ejemplo de un calendario factible para un problema así, se muestra en la Fig. 5.12.

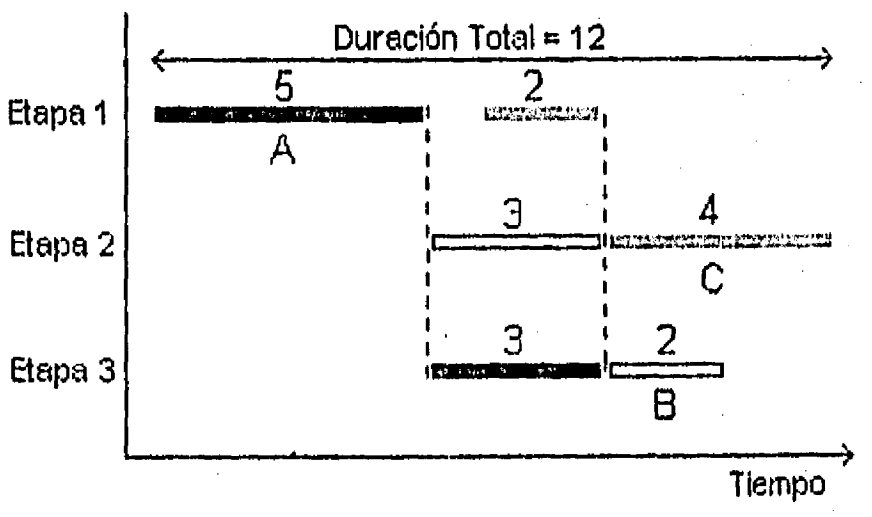


Fig. 5.12 *Ejemplo de calendarización factible*

Sin embargo, el objetivo es obtener una asignación de tareas que minimice el tiempo T de terminación. Con el objeto de obtener una solución factible, se necesita eliminar todo

antagonismo entre los tiempos de inicio de cada trabajo. Esto requiere que no se procesen dos trabajos simultáneamente en la misma etapa. Para cada par de trabajos i, k, las etapas con antagonismos potenciales son $C_{ik} = \{J(i) \cap J(k)\}$. Si dejamos que t_i sea el tiempo de comienzo del trabajo i y t_{ij} el tiempo de proceso del trabajo i en la etapa j, el tiempo final de cada trabajo i viene dado por:

$$t_i + \sum_{j \in J(i)} t_{ij}$$

mientras que la terminación del trabajo i en la etapa j viene dada por:

$$t_i + \sum_{m \in J(i),\, m \le j} t_{im}$$

y el tiempo inicial del trabajo i en la etapa j viene dado por:

$$t_i + \sum_{m \in J(i),\, m < j} t_{im}$$

La restricción que previene antagonismos entre el par de trabajos i y k en cualquier etapa crea la siguiente dicotomía:

$$\left[t_i + \sum_{\substack{m \in J(i) \\ m \le j}} t_{im} \le t_k + \sum_{\substack{m \in J(k) \\ m < j}} t_{km}\right] \vee \left[t_k + \sum_{\substack{m \in J(k) \\ m \le j}} t_{km} \le t_i + \sum_{\substack{m \in J(i) \\ m < j}} t_{im}\right] \qquad (2.1)$$

La disyunción asegura que cualquier trabajo i preceda al trabajo k o viceversa, en cualquier etapa j donde pueda ocurrir un antagonismo potencial. El problema de optimización lineal mixto-lógico puede expresarse a través de disyunciones como:

$$\begin{aligned} & \text{Min} \quad \text{Tiempo} \\ \text{s.a} \quad & \text{Tiempo} \ge t_i + \sum_{j \in J(i)} t_{ij} && \forall\, i \in I,\ \forall\, j \in C_{ik} \qquad (2.2) \\ & \left[t_i + \sum_{\substack{m \in J(i) \\ m \le j}} t_{im} \le t_k + \sum_{\substack{m \in J(k) \\ m < j}} t_{km}\right] \vee \left[t_k + \sum_{\substack{m \in J(k) \\ m \le j}} t_{km} \le t_i + \sum_{\substack{m \in J(i) \\ m < j}} t_{im}\right] \\ & \text{Tiempo} \ge 0,\ t_i \ge 0 && \forall\, i, k \in I,\ i < k \end{aligned}$$

que en su forma de representación mixta-entera, utilizando "M grandes", correspondería al modelo:

$$\begin{aligned} & \text{Min} \quad \text{Tiempo} \\ \text{s.a} \quad & \text{Tiempo} \ge t_i + \sum_{j \in J(i)} t_{ij} && \forall i \in I \qquad (2.3) \\ & t_i + \sum_{\substack{m \in J(i) \\ m \le j}} t_{im} \le t_k + \sum_{\substack{m \in J(k) \\ m < j}} t_{km} + M(1 - y_{ik}) && \forall j \in C_{ik},\ \forall i,k \in I,\ i<k \\ & t_k + \sum_{\substack{m \in J(k) \\ m \le j}} t_{km} \le t_i + \sum_{\substack{m \in J(i) \\ m < j}} t_{im} + M y_{ik} && \forall j \in C_{ik},\ \forall i,k \in I,\ i<k \\ & y_{ik} + y_{ki} = 1 && \forall i,k \in I,\ i<k \\ & \text{Time} \ge 0,\ t_i \ge 0,\ i \in I,\ y_{ik} = \{0,1\} && \forall i,k \in I,\ i<k \end{aligned}$$

donde y_{ik} y y_{ki} son variables binarias que ayudan al modelamiento mixto-entero de este problema. Las restricciones con "M-grandes" que están involucradas causan que la relajación lineal este problem sea muy pobre, y consecuentemente, la solución del problema lineal mixto-entero es a menudo muy costosa, ya que el hecho de que se satisfaga una disyunción, no necesariamente implica que la variable binaria correspondiente va a tomar valores enteros, y por lo tanto, se crean nodos innecesarios en el árbol de búsqueda. Aunque uno puede reforzar la relajación del problema lineal, usando el conjunto de ecuaciones que definen la envoltura convexa del problema, esto aumentaría notablemente el tamaño del mismo. Por otro lado, el modelo lineal mixto-lógico, usa menos variables y restricciones y permite la verificación directa del cumplimiento de las disyunciones.

Considere el problema de calendarización de trabajos mostrado en la Fig. 5.9, que involucra tres tabajos y tres etapas, con los tiempos de cumplimiento de cada trabajo para cada etapa mostrados en la Tabla 5.4:

Tabla 5.4. *Tiempos de proceso (t_{ij}) para cada trabajo en cada etapa*

Trabajos\Etapas	1	2	3
A	5	0	3
B	0	3	2
C	2	4	0

El problema lineal mixto entero de acuerdo al modelo (1.38) es el siguiente:

$$\begin{aligned} & \text{Min Tiempo} \\ \text{s.a} \quad & \text{Tiempo} \geq t_A + 8 \\ & \text{Tiempo} \geq t_B + 5 \\ & \text{Tiempo} \geq t_C + 6 \\ & t_A - t_C \leq -5 + M(1 - y_{AC}) \\ & t_C - t_A \leq -2 + M y_{CA} \\ & t_B - t_C \leq -1 + M(1 - y_{BC}) \\ & t_C - t_B \leq -6 + M y_{CB} \\ & t_A - t_B \leq -5 + M(1 - y_{AB}) \\ & t_B - t_A \leq M y_{AB} \\ & \text{Tiempo} \geq 0,\ t_A, t_B, t_C \geq 0 \qquad y_{AC}, y_{BC}, y_{AB} = \{0,1\} \end{aligned}$$

donde $M = 20$. Si usamos el modelo lineal mixto-lógico (1.37), tendríamos el siguiente problema:

$$\begin{aligned} & \text{Min Tiempo} \\ \text{s.a} \quad & \text{Tiempo} \geq t_A + 8 \\ & \text{Tiempo} \geq t_B + 5 \\ & \text{Tiempo} \geq t_C + 6 \\ & t_A - t_C \leq -5 \quad \vee \quad t_C - t_A \leq -2 \\ & t_B - t_C \leq -1 \quad \vee \quad t_C - t_B \leq -6 \\ & t_A - t_B \leq -5 \quad \vee \quad t_B - t_A \leq 0 \end{aligned}$$

Tiempo ≥ 0, $t_A, t_B, t_C \geq 0$

3. El problema de la localización de almacenes

A continuación se presenta el modelo de un simple problema de localización de almacenes, dada su utiliadad para ilustrar como se pueden generar cortes de la restricción tipo "maleta". El problema es escoger un conjunto de almacenes de capacidad limitada para que sirvan a un conjunto de puntos de demanda, mientras se minimizan los costos fijos y de transporte. Si dejamos

x_{ij} = flujo del almacén i al punto de demanda j.
f_i = costo fijo del almacén i.
k_i = capacidad del almacén i.
d_j = demanda en el punto j.
c_{ij} = costo unitario de transporte de i a j.

El costo fijo z_i que se paga si el almacén i se utiliza, es o cero o f_i, dependiendo de cuando el flujo que sale del almacén es positivo. Esto nos da la disyunciones

$$\left(z_i = f_i\right) \vee \left(\begin{matrix} z_i = 0 \\ \sum_j x_{ij} = 0 \end{matrix}\right), \quad \forall i$$

El modelo de programación lineal mixta-lógica puede por lo tanto, ser escrito como,

$$\min \sum_i z_i + \sum_{ij} c_{ij} x_{ij}$$

$$\begin{aligned} \text{s.a} \quad & \textstyle\sum_j x_{ij} \leq k_i, && \forall i \\ & \textstyle\sum_i x_{ij} \geq d_j, && \forall j \\ & x_{ij} \geq 0, && \forall i,j \\ & y_i \vee y'_i, \\ & y_i \rightarrow (z_i = f_i), && \forall i \\ & y'_i \rightarrow (\textstyle\sum_j x_{ij} = 0), && \forall i \end{aligned}$$

y el modelo tradicional lineal mixto-entero es

$$\min \sum_i k_i y_i + \sum_{ij} c_{ij} x_{ij}$$

$$\begin{aligned} \text{s.a} \quad & \textstyle\sum_j x_{ij} \leq k_i y_i, && \forall i \\ & \textstyle\sum_i x_{ij} \geq d_j, && \forall j \\ & x_{ij} \geq 0, && \forall i,j \\ & y_i \in \{0,1\}, && \forall i \end{aligned}$$

4. La fiesta progresiva

El problema final a ser considerado es un problema de calendarización de eventos llamado "la fiesta progresiva", surgido a raíz de la organización de un rally en Inglaterra el año pasado. El problema ganó alguna notoriedad cuando un grupo de gente de programación matemática y de programación por restricciones encontró que este problema no se podía resolver con los métodos clásicos de la programación lineal mixta-entera, pero sí con algunos artificios, intervención manual y técnicas de programación por restricciones.

Presenta dificultades insalvables para la programación matemática, principalmente porque parece que no existe ninguna formulación económica del problema dentro del entorno de la programación lineal mixta-entera. Sin embargo, este problema se puede formular mucho más fácilmente en el contexto más amplio de la programación lineal mixta-lógica y sirve para ilustrar las ventajas de las variables discretas multivalentes.

En una fiesta progresiva, el objetivo es hacer que las tripulaciones de un conjunto de yates visiten un subconjunto de yates en un número fijo de periodos, para encontrarse con los miembros de otras tripulaciones. Debido al costo de las provisiones que deben existir en los yates que servirán como anfitriones, el objetivo será el de minimizar el número de yates anfitriones, mientras que se pretende que los miembros de las tripulaciones se encuentren con el mayor número posible de tripulaciones de otros yates, dadas ciertas condiciones específicas.

El problema puede ser definido más precisamente como sigue. Se da un conjunto I de botes. Cada bote i es ocupado por una tripulación de c_i personas y tiene espacio para K_i personas a bordo. El problema es el de minimizar el número de botes anfitriones. Cada tripulación i visita un bote anfitrión diferente h_{it} en cada periodo t, a menos que sea él mismo un bote anfitrión, lo cual se indica con el valor verdadero de la proposición δ_i, y en este caso $h_{it} = i$ para todo intervalo de tiempo t. A su vez se requiere que la tripulación de un bote anfitrión permanezca en su propio barco para recibir a las demás tripulaciones que lo visitan. Con el objeto de hacer que el mayor número posible de gente se conozca, no se permite que ningún par de tripulaciones visitantes se encuentre más de una vez en la fiesta. La proposición m_{ijt} será verdadera cuando las tripulaciones i y j de barcos que no son anfitriones, visitan el mismo bote en el periodo t.

Para checar las restricciones de capacidad, es conveniente definir una proposición v_{jt} que sea verdadera cuando $h_{it} = j$. Las únicas proposiciones que fueran restricciones lineales de desigualdad son las δ_i que fuerzan $z_i = 1$ cuando son verdaderas. Las demás proposicioens corresponden a conjuntos vacíos de restricciones.

El problema puede establecerse como sigue. La función objetivo cuenta el número de barcos anfitriones. El predicado *todos-diferentes* significa que todos los argumentos tienen diferentes valores.

$$\min \sum_{i \in I} z_i \qquad (4.1)$$

s.a

$z_i \geq 0$,	$i \in$ I	(4.2)
$v_{ijt} \equiv (h_{it} = j)$,	$i,j \in$ I, $t \in$ T	(4.3)
$\delta_i \vee$ *todos-diferentes* $(h_{it}, \ldots, h_{i\|I\|})$,	$i \in$ I	(4.4)
$\delta_i \equiv (h_{it} = i)$,	$i \in$ I, $t \in$ T	(4.5)
$\sum_{\substack{i \in I \\ i \neq j}} c_i v_{ijt} \leq K_j - c_j$,	$j \in$ I, $t \in$ T	(4.6)
$\delta_i \vee \delta_j \vee m_{ijt} \vee (h_{it} \neq h_{jt})$,	$i,j \in$ I, $i < j$, $t \in$ T	(4.7)
$\sum_{t \in T} m_{ijt} \leq 1$,	$i,j \in$ I, $i < j$	(4.8)
$\delta_i \rightarrow (z_i \geq 1)$,	$i \in$ I	(4.9)
$h_{it} \in \{1, \ldots, \|I\|\}$,	$i \in$ I, $t \in$ T.	

La fórmula (4.3) define v_{ijt}. La fórmula (4.4) dice que la tripulación i debería visitar un bote diferente en cada periodo a menos que sea un bote anfitrión. La fórmula (4.5) causa que una tripulación permanezca en su propio bote si y sólo si es un bote anfitrión.

La fórmula (4.6) es la restricción de la capacidad del bote. Debería ser interpretada como una fórmula lógica en lugar de interpretarla como una desigualdad en variables 0-1. La sumatoria significa que c_i se cuenta en la suma para cada v_{ijt} verdadero. La desigualdad como un todo significa que las suficientes v_{ijt} deben ser falsas de tal manera que la suma resultante sea a lo más $K_j - c_j$. La interpretación de las desigualdades como proposiciones lógicas se discutió en el capítulo anterior.

La fórmula (4.7) dice que si los yates i y j no son anfitriones (*i.e.*, δ_i y δ_j son falsos), entonces una de dos, o m_{ijt} es verdadero (ambas se encuentran en algún otro bote en ese periodo de tiempo) o $h_{it} \neq h_{jt}$ (no están ambas en el mismo bote). La siguiente fórmula (4.8) dice que un par de tripulaciones visitantes no debería encontrarse más de una vez. Aquí nuevamente, esta desigualdad debería interpretarse como proposición lógica.

El modelo entero tiene $O(|I|^2|T|)$ variables y restricciones. Note que el modelo lineal es trivial, ya que consiste sólo de la función objetivo y de restricciones de la forma $z_i = 1$. EL problema lineal se vuelve más interesante cuando se pueden agregar cortes algebraicos dentro del modelo y cortes lógicos que se procesen simbólicamente, para reforzar la relajación.

La formulación de un modelo de programación lineal es mucho más difícil. La restricción más desafiante es la que dice que las tripulaciones que no pertenecen a barcos anfitriones no se encuentren más de una vez.. Los autores de [60] remarcaron que si esto se formula usando las variables v_{ijt}, se generan $O(|I|^4|T|^2)$ restricciones. Ya que esto es totalmente impráctico, introdujeton $O(|I|^3|T|)$ variables y_{ijkt}, las cuales toman el valor de 1 cuando las tripulaciones j y k se encuentran en el bote i en el periodo t. Pero ya que había más de 40 botes en el problema, este modelo resultó en un enorme número de variables.

Un modelo diferente de programación lineal mixto-entero se sugiere aquí. Retiene las variables multivalentes h_{it} y fuerza las restricciones *todos-diferentes* en una manera poco ortodoxa, dadas las características de los modelos de programación lineal mixta-entera. Las variables h_{it} usadas no son explícitamente restringidas a tomar valores enteros, ya que las trestricciones restantes fuerzan su integralidad. El modelo tiene $O(|I|^2|T|)$ variables enteras y restricciones, muchos menos del modelo que usaron en Inglaterra [60], y que mostró que no puede ser resuelto.

El modelo de programación lineal resultante, puede ser formulado así

$$\min \sum_{i \in I} \delta_i \qquad (4.10)$$

s.a

$$\delta_j + (1 - v_{ijt}) \geq 1, \qquad i,j \in I, t \in T \qquad (4.11)$$

$$(1 - \delta_i) + v_{iit} \geq 1, \qquad i \in I, t \in T \qquad (4.12)$$

$$v_{ijt} + \alpha_{ijt} + \beta_{ijt} \geq 1, \qquad i,j \in I, t \in T \qquad (4.13)$$

$$- h_{it} + j \geq 1 - |I|(1 - \alpha_{ijt}), \qquad i,j \in I, t \in T \qquad (4.14)$$

$$h_{it} - j \geq 1 - |I|(1 - \beta_{ijt}), \qquad i,j \in I, t \in T \qquad (4.15)$$

$$\sum_{\substack{i \in I \\ i \neq j}} c_i v_{ijt} \leq K_j - c_j, \qquad j \in I, \ t \in T \qquad (4.16)$$

$$\sum_{t \in T} v_{ijt} \leq 1, \qquad i,j \in I, i \neq j \qquad (4.17)$$

$$\delta_i + \delta_j + m_{ijt} + \phi_{ijt} + \psi_{ijt} \geq 1, \qquad i,j \in I, i < j, t \in T \qquad (4.18)$$

$$- h_{it} + h_{jt} \geq 1 - |I|(1 - \phi_{ijt}), \qquad i,j \in I, i < j, \ t \in T \qquad (4.19)$$

$$h_{it} - h_{jt} \geq 1 - |I|(1 - \psi_{ijt}), \qquad i,j \in I, i < j, t \in T \qquad (4.20)$$

$$\sum_{t \in T} m_{ijt} \leq 1, \qquad i,j \in I, i < j \qquad (4.21)$$

La función objetivo (4.10) nuevamente cuenta el número de botes anfitriones. Las restricciones (4.11) y (4.12) requieren que las tripulaciones de los barcos permanezcan en sus propios botes si y sólo si se trata de un bote anfitrión. Las restricciones (4.13) y (4.14) usan un mecanismo disyuntivo para relacionar v_{ijt} a h_{it}. Ellas dicen que si $h_{it} = j$ (*i.e.*, $\neg\alpha_{ijt}$ y $\neg\beta_{ijt}$, lo que dice que h_{it} no es ni menor ni mayor que j sino igual), entonces $v_{ijt} = 1$. La restricción (4.15) es la misma restricción de capacidad que antes, interpretada en esta ocasión como una desigualdad 0-1. La restricción (4.16) juega el rol de la restricción *todos-diferentes*. Las restricciones (4.17)-(4.20) nuevamente hacen uso de un mecanismo disyuntivo para decir que si i y j no son botes anfitriones y $h_{it} = h_{jt}$, entonces $m_{ijt} = 1$. Como antes, (4.21) dice que un par de tripulaciones puede encontrarse a lo sumo una vez.

CAPITULO 6:RESULTADOS Y CONCLUSIONES

La finalidad de estos experimentos computacionales no fué la de comparar el mejor algoritmo posible de programación lineal mixta-lógica para un problema dado con el mejor algoritmo posible en competencia, sino la de aislar el efecto del desempeño de las diferentes herramientas de programación mixta-lógica que son ilustradas en cada problema. Al final, se hace una comparación del algoritmo más simple de programación lineal mixta-lógica contra el algoritmo más simple de programación lineal mixta-entera.

El algoritmo de programación lineal mixta-lógica presentado en la Fig. 3.1 del capítulo 3, se aplica como sigue. La regla de ramificación consiste en seleccionar la primera variable proposicional en la primera fórmula lógica no satisfecha. El algoritmo de procesamiento lógico es la resolución unitaria. La relajación de las fórmulas utilizada varía de caso a caso, como se describe en cada sección de este capítulo. El código se escribió en C y se compiló en una estación de trabajo SPARC 330 corriendo SUN OS versión 4.1.1. Las relajaciones lineales de los problemas en cada nodo del árbol de búsqueda se resolvieron utilizando las rutinas de la librería de CPLEX versión 3.0.

El algoritmo de programación lineal mixta-entera utilizado para efectuar comparaciones es un procedimiento clásico de ramificación y acotamiento que también utiliza la versión 3.0 de la librería de CPLEX para resolver la relajación lineal en cada nodo del árbol.. En este caso, la regla de ramificación utilizada consistió en escoger la variable cuyo valor en la relajación estuviera más cercano a 0.5.

También se reportan los tiempos de corrida y el número de nodos para la versión 3.0 del código cerrado de programación lineal mixta-entera de CPLEX. Se enfatiza, sin embargo, que la comparación con un código comercial no proporciona la suficiente información, como se reporta en [34]; ya que los detalles de la implementación de los códigos comerciales no son del conocimiento público, y aún cuando así lo fueran, es bastante difícil aislar los factores que pueden explicar las diferencias en el desempeño.

La programación lineal mixta-lógica ha mostrado muchas ventajas en los problemas de síntesis de procesos en Ingeniería Química, y los resultados computacionales para estos problemas, reportados en esta tesis, confirman los trabajos publicados previamente. Sin embargo, se escogió reportarlos nuevamente aquí porque la confirmación de los resultados experimentales es una clave elemental de la ciencia empírica que en ocasiones no ha recibido la suficiente importancia en la literatura algorítmica. Con respecto a los demás casos explorados, algunos de los problemas fueron de creación propia (como algunos problemas de localización de almacenes), o se tomaron de alguna base de datos

(como los problemas de localización de almacenes con capacidades grandes) o se seleccionaron de la literatura en el área (como los de calendarización de trabajos con transferencia-cero y los de síntesis de procesos). En otros casos, se cambiaron algunas de las especificaciones (como es el caso de utilizar variables semicontinuas en las redes de proceso), por lo que los resultados obtenidos no estaban reportados previamente. En todos los casos, dichos problemas fueron resueltos, para comparación, con nuestros códigos de programación lineal mixta-lógica y de ramificación y acotamiento y con CPLEX.

1. Problemas en Síntesis de Procesos

Los problemas de síntesis de redes de procesos químicos ilustran el uso de los cortes lógicos (no válidos) así como la ventaja del enfoque de programación lineal mixta-lógica en el modelamiento de las variables semicontinuas.

Si recordamos que el elemento discreto de estos problemas es la decisión de cuando instalar o no instalar una unidad i y sus arcos incidentes. Una de dos, o ocurre un costo fijo f_i o no existe costo fijo y ningún flujo sale de la unidad. Si z_i es el costo fijo pagado por la unidad i, tenemos la siguiente disyunción en el modelo utilizado

$$\left(z_i = f_i\right) \vee \begin{pmatrix} z_i = 0 \\ \sum_{(i,j)\in E} x_{ij} = 0 \end{pmatrix} \tag{1.1}$$

Esta disyunción puede ser expandida en dos disyunciones que pueden ser relajadas

$$(z_i \geq f_i) \vee (-z_i \geq 0) \tag{1.2}$$

$$\left(z_i \geq f_i\right) \vee \left(-\sum_{(i,j)\in E} x_{ij} \geq 0\right) \tag{1.3}$$

Ya que f_i es un límite superior en z_i, el corte elemental representado por la ecuación (5.6) del capítulo 3, aplicado a (1.2) es simplemente $0 \geq 0$, lo cual no nos sirve de nada, pero este mismo corte aplicado a la disyunción (1.3) nos proporciona la siguiente ecuación

$$\frac{z_i}{f_i} \geq \frac{1}{M_i} \sum_{(i,j)\in E} x_{ij}, \tag{1.4}$$

donde M_i es un límite superior en el flujo de salida de la unidad i. Se puede ver fácilmente que este corte define una faceta del envolvente convexo de la disyunción en (1.1), que también puede expresarse como $y_i \vee y'_i$ si hacemos y_i igual al primer elemento de la disyunción y y'_i al segundo.

Más aún, el Teorema 5.4 del capítulo 3 implica que la formulación 0-1 de la disyunción, $y_i \vee y'_i$ es integral. Se puede checar fácilmente que si las variables 0-1 y_1, y_2 corresponden a los dos elementos de la disyunción y_i, y'_i, entonces $y^*_1(y_2) = y^*_2(y_1) = 0$. Esto sugiere

que la formulación utilizada en la programación lineal mixta-lógica con estos cortes elementales puede no proporcionar ninguna ventaja sobre la relajación tradicional de la programación lineal mixta-entera.

Aunque una examinación minuciosa del problema nos puede dar cortes lógicos útiles, si examinamos una red de proceso como la descrita en la Fig. 5.8 del capítulo 5, podemos deducir cortes lógicos utilizando la conectividad de las unidades en la red. Por ejemplo, es claro que no se debería instalar una columna de destilación hasta que estemos seguros de que su salida va a estar conectada al menos a otra columna adyacente instalada, y de que al menos una de las columnas adyacentes que le proporcionan corrientes de entrada esté también instalada. Por ejemplo, la unidad 6 no debería ser instalada hasta que la unidad 3 y la 9 también lo estén. Esto produce los cortes lógicos

$$y_6 \rightarrow y_3, \quad y_6 \rightarrow y_9$$

lo cual puede ser escrito como las siguientes cláusulas,

$$y_3 \vee \neg y_6, \neg y_6 \vee y_9 \tag{1.5}$$

Estos cortes no son válidos porque no existe ninguna condición de infactibilidad que impida instalar una unidad que no acarre flujo, y bloquee estas soluciones. Aunque se puede sospechar que una búsqueda de ramificación y acotamiento no consideraría estas soluciones espurias, de tal manera que los cortes (1.5) no tendrían ningún efecto, la experiencia reportada en [37,50], nos dice que estos cortes son muy efectivos reduciendo el número de nodos del árbol de búsqueda, un hecho que se confirma aquí.

Es posible fabricar problemas de diseño de redes para los cuales la resolución unitaria sea incompleta para las fórmulas lógicas que haya en el modelo, cuando se agreguen los cortes lógicos, pero ninguno de los problemas resueltos tiene esta propiedad, y por esta razón, sólo se utilizó resolución unitaria para el proceso lógico.

La relajación de los cortes lógicos ilustra dos puntos. Uno es que el corte $y_3 \vee \neg y_6$, por ejemplo no tiene relajación porque y_6 está negada. Sin embargo, ya que $y_6 \vee y'_6$ está dada en el modelo, el corte implica $y_3 \vee y'_6$, lo cual puede ser escrito como las dos disyunciones siguientes, $(z_1 \geq f_1) \vee (-z_6 \geq 0)$

$$\left(z_1 \geq f_1\right) \vee \left(-\sum_{(i,j)\in E} x_{6j} \geq 0\right)$$

que respectivamente pueden generar los siguientes cortes elementales

$$\frac{z_1}{f_1} \geq \frac{z_6}{f_6} \tag{1.6}$$

$$\frac{z_1}{f_1} \geq \frac{1}{M_6} \sum_{(i,j)\in E} x_{6j} \tag{1.7}$$

FALTA PAGINA

No 6.41

METODOLOGIA	Mixta-Lógica	Mixta-Lógica	Mixta-Lógica	Mixta-Entera	CPLEX	Mixta-Lógica	Mixta-Lógica	Mixta-Entera	CPLEX
	* Sin cortes	*Cortes duales	*Cortes elementales			*Cortes elementales *Cortes lógicos *Resolución unitaria	*Cortes elementales *Cortes lógicos *Relajaciones lógicas		
No. de Nodos									
5 componentes	61	21	15	17	11	9	3	3	7
*Rest.a 4 unid.			15	49	29	13	3	3	4
6 componentes	1659	105	97	191	94	63	97	33	40
*Rest.a 5 unid.			9	163	56	5	3	3	15
Sogundos CPU									
5 componentes	0.91	3.39	0.41	0.31	0.33	0.35	0.4	0.18	0.4
*Rest.a 4 unid.			0.45	1.01	0.82	0.52	0.42	0.23	0.28
6 componentes	33.3	26.5	2.3	5.6	3.5	2.6	8.1	3.3	3.5
*Rest.a 5 unid.			0.8	5.9	2	0.6	0.9	0.4	1.4

Tabla 6.1 *Número de nodos y tiempo de CPU en segundos para redes de separación*

El segundo punto es que (1.7) puede ser deshechada ya que es implicada por (1.4) y por (1.6).

Los problemas de síntesis pueden ser modificados fijando el número de unidades a ser instaladas. Esto puede ser hecho con la fórmula,

$$\sum_i y_i = k$$

Para generar cortes elementales, esta fórmula se puede escribir como dos desigualdades.

$$\sum_i y_i \geq k \quad \text{y} \quad \sum_i y'_i \geq k.$$

Los cortes elementales de la forma (5.7) (capítulo 3) para estas desigualdades son, respectivamente,

$$\sum_i \frac{z_i}{f_i} \geq k \qquad y \qquad \sum_{ij} \frac{x_{ij}}{M_i} \leq n-k$$

donde n es el número de unidades potenciales.

Los resultados experimentales para los dos problemas de 5 componentes y los dos de 6 componentes estudiados por Raman y Grossmann [50] se presentan en la Tabla 6.1. El segundo problema de separación de 5 componentes fija el número total de unidades a 4, y el segundo problema de separación de 6 componentes fija el número total de unidades a 5. Los métodos de solución se agrupan por la fuerza de la formulación. Los problemas se resuelven primero por programación lineal mixta-lógica usando sólo la regla disyuntiva en la ramificación, sin ninguna relajación de las restricciones disyuntivas. Los resultados tan pobres en la primera columna de la tabla indican la importancia de usar las

relajaciones. La siguiente columna ilustra el costo de generar cortes duales, como se discutió en la sección 5.2 del capítulo 3.

Las siguientes tres columnas de la tabla comparan el algoritmo de programación lineal mixta-lógica con el algoritmo de programación lineal mixta-entera y con CPLEX, usando las relajaciones que tienen la fuerza de las relajaciones continuas tradicionales del problema original; en el caso de la mixta-lógica, esto requiere de los cortes elementales (1.4). La siguiente columna adiciona los cortes lógicos descritos anteriormente, y los procesa en forma simbólica por medio de resolución unitaria, sin agregar sus relajaciones en el modelo. Las últimas tres columnas agregan los cortes lógicos utilizando variables binarias en el modelo mixto-entero y en CPLEX, y sus relajaciones en el modelo mixto-lógico.

Los resultados sugieren que la adición de cortes lógicos no válidos puede ocasionar una mejora sustancial en el contexto de la programación lineal mixta-entera. También reducen el número de nodos generados por la rutina CPLEX MILP, lo cual indica que su empleo no sólo duplica la acción del preprocesador de CPLEX. Los experimentos reportados por Raman y Grossmann [50] proporcionan una indicación similar para el preprocesador de OSL. Los cortes lógicos reducen el número de nodos para el programa lineal mixto-lógico, pero esto no se refleja en los tiempos de cálculo. La comparación de los métodos agrupados sugiere, como se predijo antes, que la formulación tradicional 0-1 es al menos tan efectiva como la formulación lineal mixta-lógica. De hecho, la adición de las relajaciones para los cortes lógicos aumenta el tiempo de computación necesario para el enfoque de programación lineal mixta-lógica con respecto a la mixta-entera.

Este es un caso en el que el punto de vista lógico proporciona cortes útiles, pero el modelamiento basado en la lógica no proporciona ventajas computacionales.

Sin embargo, el uso de variables proposicionales es particularmente ventajoso cuando se agregan variables semicontinuas al modelo. Es ineficiente representar semicontinuidad con variables enteras por dos razones, ambas mencionadas anteriormente: la relajación continua, como cualquier relajación lineal de semicontinuidad es muy ineficiente, y la representación 0-1 no es integral.

La síntesis de una red de 10 y otra de 38 procesos, descritas en [55] fueron resueltos. Se utilizó la representación para variables semicontinuas descrita en la ecuación 1.4 de la sección 1.4 del capítulo 5. No se usó ninguna relajación para las disyunciones resultantes porque, como se especificó antes, ninguna relajación es útil. Se generaron cortes elementales para las disyunciones $y_i \vee y'_i$. Los cortes lógicos no válidos se usaron en los modelos de programación lineal mixta-lógica, mixta-entera y CPLEX, pero no se generaron relajaciones para ellas en el modelo mixto-lógico.

Los resultados aparecen en la Tabla 6.2. El problema de 10 procesos tiene 3 variables semicontinuas, y el de 38 procesos tiene 7. Diferentes versiones del problema se obtuvieron variando el tiempo en el horizonte y los límites de los intervalos.

PROBLEMA	Número de nodos			Segundos de CPU		
	Mixta-Lógica	Mixta-Entera	CPLEX	Mixta-Lógica	Mixta-Entera	CPLEX
10 procesos, version 1	5	29	24	0.24	0.82	0.65
10 procesos, version 2	13	35	52	0.41	0.88	1.47
38 procesos, version 1	729	1083	677	199	376	178
38 procesos, version 2	1907	3237	868	559	1173	271
38 procesos, version 3	1161	1999	345	306	836	104
38 procesos, version 4	1901	2861	747	514	1093	229
38 procesos, version 5	1081	1561	296	287	551	89

Tabla 6.2 *Número de nodos y segundos de CPU para la síntesis de redes de reactores*

Los resultados muestran que una representación lógica general de la semicontinuidad reduce ampliamente el tiempo de cálculo, aún cuando el número de variables expresadas en forma semicontinua no es muy grande comparado con el número de variables discretas del problema. Un enfoque razonable para estos problemas puede, por lo tanto, usar una representación 0-1 tradicional de las variables discretas que representan los procesos involucrados, o representar las variables de decisión por medio de los cortes relajados elementales de las disyunciones como en (1.4) y una representación basada en la lógica de la semicontinuidad. La programación lineal mixta-lógica proporciona esta clase de flexibilidad.

El preprocesador de CPLEX elimina la mayoría de los renglones y columnas del problema de 38 procesos (pero no del de 10 procesos) y por lo tanto obtiene un desempeño superior en estos problemas. Es imposible analizar este resultado sin un conocimiento detallado del preprocesador, porque la operación que probó ser tan efectiva en este caso, puede ser agregada al código de programación lineal mixta-lógica. En cualquier caso, el objetivo aquí es el de aislar elefecto de usar un procedimiento basado en la lógica contra la representación 0-1 de la semicontinuidad y eso puede observarse por la mejora en los resultados de la columna mixta-lógica con la mixta-entera, para todos los casos.

2. Calendarización de trabajos con transferencia cero

Los problemas de la calendarización de trabajos con *transferencia-cero* ilustran dos ventajas de la programación lineal mixta-lógica: *a*) puede resolver problemas con árboles de búsqueda mucho más pequeños que la programación lineal mixta-entera, y *b*) el tiempo de procesamiento en cada nodo es menor, ya que la eliminación de variables enteras hace las relajaciones lineales a resolver en cada nodo más pequeñas.

Como se discutió en la sección 5.1 del capítulo 3, no existe ninguna razón para introducir las relajaciones lineales de las restricciones disyuntivas típicas de los problemas de calendarización, y por lo tanto, se omiten. Si existen m trabajos y n máquinas, esto reduce el número de variables en la relajación lineal de $2m + mn$ a sólo $2m$.

Número de		Lineal Mixta-Lógica			Lineal Mixta-Entera			CPLEX		
Trabajos	Maquinas	nodos	tiempo	tiempo/ nodo	nodos	tiempo	tiempo/ nodo	nodos	tiempo	tiempo/ nodo
6	5	407	2.7	0.0066	689	10.1	0.0147	527	8.1	0.0154
7	5	1951	15.7	0.0080	3171	52.2	0.0165	2647	51.0	0.0193
8	5	14573	129.0	0.0089	24181	546.4	0.0226	16591	413.9	0.0249

Tabla 6.3 *Número de nodos, segundos de CPU y segundos/nodo para problemas de calendarización de trabajos con transferencia-cero*

Más aún, el modelo de programación lineal mixta-entera crea siempre un árbol más grande ya que su relajación continua no es integral. Esto puede seguirse del Corolario 5.1 del capítulo 3 que implica que la representación lineal mixta-entera de la disyunción

$$(t_k - t_i \geq r_{ik}) \vee (t_i - t_k \geq r_{ki})$$

es integral si y sólo si

$$\max \{t_k - t_i \mid t_i - t_k \geq r_{ki},\ (0,0) \leq (t_i, t_k) \leq (m_i, m_k)\} = r_{ki} - M_{ki} \qquad (2.1)$$
$$\max \{t_k - t_i \mid t_k - t_i \geq r_{ik},\ (0,0) \leq (t_i, t_k) \leq (m_i, m_k)\} = r_{ik} - M_{ik}$$

Definiendo M_{ki} y M_{ik} de acuerdo a la ecuación (5.11) de la sección 5.3 del capítulo 3, tenemos que $(M_{ki}, M_{ik}) = (r_{ki} + m_k, r_{ik} + m_i)$. También es fácil ver que los dos máximos en (2.1) son respectivamente iguales a $-r_{ki}$ y $-r_{ik}$. Así (2.1) implica que la representación lineal mixta-entera es integral si y sólo si $(r_{ki}, r_{ik}) = (m_k, m_i)$, lo cual no ocurre en la práctica.

Se presentan resultados para tres ejemplos de calendarización de trabajos con transferencia-cero en una planta química [52] , y los resultados aparecen en la Tabla 6.4. La programación lineal mixta-lógica genera cerca de un 60% de los nodos de la programación lineal mixta-entera y usa menos de la mitad del tiempo por nodo. Por lo tanto, corre de 3 a 4 veces más rápido que la programación mixta-entera para estos problemas.

3. Problemas de localización de almacenes

Los problemas de localización de almacenes ilustran la generación de cortes lógicos de una restricción tipo "maleta". También es un caso donde la formulación basada en la lógica resulta ser menos eficiente que la formulación tradicional, por lo que la mejora real la constituye la utilización de los cortes lógicos y no el uso de los cortes elementales para las restricciones disyuntivas.

La formulación de los cortes elementales para las restricciones disyuntivas $y_i \vee y'_i$ es la misma que la utilizada en los problemas de síntesis de procesos (1.4). Estos cortes, nuevamente definen una faceta del envolvente convexo del problema, y la representación 0-1 es también integral. El modelo resultante de programación lineal mixta-lógica es un

poco más grande que el de programación lineal mixta-entera, porque el modelo tradicional incluye la representación de la "*M grande*" en las restricciones de capacidad de los almacenes y en el modelo basado en la lógica, los cortes elementales practicamente duplicaron estas restricciones, por lo que puede esperarse que la programación mixta-entera presente cierta ventaja en tiempo de solución sobre la formulación basada en la lógica.

Problema	No. de al-	Radio	Nodos			Segundos de CPU			Segundos de CPU por nodo		
	macenes	Cap/Dem	Mixta-Lóg	Mixta-Ent	CPLEX	Mixta-Lóg	Mixta-Ent	CPLEX	Mixta-Lóg	Mixta-Ent	CPLEX
CAP41	16	1.37	57	81	62	8.6	8.8	5.5	0.15	0.11	0.09
CAP42	16	1.29	59	81	57	8.9	8.6	5.5	0.15	0.11	0.10
CAP43	16	1.29	61	83	42	9.1	8.9	4.4	0.15	0.11	0.10
CAP44	16	1.37	43	61	40	7.1	6.8	4.3	0.17	0.11	0.11
CAP51	16	2.75	1239	1429	1134	172	135	92	0.14	0.09	0.08
CAP61	16	3.86	2147	2631	3017	266	237	235	0.12	0.09	0.08
CAP71	16	16	3481	4495	8830	409	398	658	0.12	0.09	0.07
1	10	6.12	61	147	31	1.21	0.70	0.57	0.020	0.015	0.018
2	10	5.1	63	45	25	1.34	0.67	0.52	0.021	0.015	0.021
3	10	4.08	71	73	59	1.54	1.14	1.17	0.022	0.016	0.020
4	10	3.06	49	173	138	1.11	2.61	2.50	0.023	0.015	0.018
5	10	2.04	19	31	27	0.45	0.45	0.55	0.024	0.015	0.020
6	10	1.02	3	21	20	0.16	0.30	0.32	0.053	0.014	0.016

Tabla 6.4 *Número de nodos, tiempo de CPU en segundos y tiempo de CPU por nodo*

El hecho de que la capacidad total instalada en los almacenes debe acomodar el total de la demanda, nos da origen a una restricción válida tipo "maleta".

$$\sum k_i y_i \geq \sum d_j \qquad (3.1)$$

Que se puede ver como una fórmula lógica cuya relajación elemental puede ser agregada al modelo de programación lineal resuelto en cada nodo:

$$\sum k_i (z_i/f_i) \geq \sum d_j$$

Se pueden derivar cortes contiguos de (3.1) como se describe en la sección 5 del capítulo 4. Estas cláusulas tienen la forma $\vee_{i \in I}$ y tienen relajaciones elementales de la forma,

$$\sum k_i (z_i/f_i) \geq 1$$

Se resolvieron siete problemas de localización de almacenes con capacidades grandes de [7], y los resultados aparecen en la Tabla 6.4. Cada problema tiene 50 puntos de demanda con una demanda total de 58,268. Cada almacén tiene la misma capacidad, y el radio de la capacidad total de los almacenes sobre la capacidad total de la demanda también se muestra.

Los cortes contiguos fueron utilizados en el modelo lineal mixto-lógico, pero no en el modelo lineal mixto-entero. El resultado fué una reducción del 20-30% en el número de nodos pero un aumento del 30-50% en el tiempo de solución por nodo, ya que junto con los cortes elementales de las restricciones disyuntivas, las relajaciones de estos cortes agrandan el modelo. El resultado neto es que el modelo de programación mixta-lógica es un poco más lento que el de programación mixta-entera, lo que nos lleva a concluir que la mejor opción será la de utilizar el modelo lineal mixto-entero, agregando los cortes lógicos generados en cada nodo y expresándolos en forma de variables binarias.

Los problemas 1-6 de la Tabla 6.4 fueron resueltos para probar la hipótesis de que los cortes contiguos tienen un mayor efecto en problemas que están altamente restringidos, como se puede indicar por el radio de la capacidad total de almacenamiento entre la demanda total. Los problemas son idénticos, excepto por la capacidad de los almacenes. Existen 7 puntos de demandas con valores de 4,5,6,7,8,9 y 10. Los resultados obtenidos tendieron favorablemente a confirmar esta hipótesis.

4. El problema de la fiesta progresiva

La programación lineal mixta-lógica tiene ventajas sustanciales en el modelamiento y en la obtención de resultados computacionales para este problema. Su notación lógica permite una declaración más simple de las restricciones, como ya se vió en la Sección 4 del capítulo 5. La ventaja computacional más importante está en que del gran número de variables discretas necesarias en el problema, sólo una pocas de ellas corresponden a restricciones lineales. El programa lineal mixto-lógico puede procesarlas simbólicamente sin agrandar la relajación lineal en cada nodo del árbol de búsqueda, la cual contiene sólo unas cuantas restricciones por nodo.

A la formulación lineal mixta-lógica se le aumentó un simple corte lógico que restringe el número de botes anfitriones a no ser menor que el número de periodos:

$$\sum_{i \in I} \delta_i \geq |T|$$

Los resultados computacionales fueron los siguientes:

No. de yates	Periodos de tiempo	Nodos		Segundos de CPU		Seg. de CPU por nodo	
		Mixta-Lógica	CPLEX	Mixta-Lógica	CPLEX	Mixta-Lógica	CPLEX
5	2	171	1136	0.98	197	0.006	0.173
6	2	239	5433	1.41	1708	0.006	0.314
6	3	37	366	0.25	162.8	0.007	0.445
7	3	71	44	0.52	44.6	0.007	1.014
8	3	209	2307	1.63	3113	0.008	1.349
8	4	167	582	1.35	887.5	0.008	1.525
10	3	24142	*20000	12259	*54150	0.011	2.708
10	4	28923	*20000	319.4	*43223	0.011	2.161

* CPLEX terminó después de 20,000 nodos, sin encontrar una solución entera factible

Tabla 6.5 *Número de nodos, tiempo de CPU en segundos y tiempo de CPU por nodo para el problema de la fiesta progresiva*

Como puede obsrvarse en la Tabla 6.5, el tiempo de CPU por nodo permanece casi constante para la programación lineal mixta lógica, mientra que crece considerablemente para el caso de la programación lineal mixta-entera. Esto se explica claramente con los tamaños de los modelos utilizados en cada caso. Para la columna de programación mixta-lógica se utilizó el modelo (4.1)-(4.10) de la sección 4 del capítulo 5 y para la programación lineal mixta-entera, el modelo (4.10)-(4.21). También es interesante notar que para los casos de 10 botes y 3 y 4 periodos de tiempo, CPLEX ni siquiera pudo encontrar una solución entera factible para el problema. Este problema muestra muy claramente las ventajas del entorno de programación lineal mixta-lógica sobre los métodos tradicionales, para resolver este tipo de problemas.

5. Conclusiones

Después de haber aplicado el conjunto de herramientas de la programación lineal mixta-lógica a los tipos de problemas ejemplificados en esta tesis, se puede concluir, en base a lo expuesto, la importancia y el impacto de las ventajas de la programación lineal mixta-lógica sobre la programación lineal mixta-entera, mismas que como potenciales, se apuntaban en la introducción, y que se han ejemplificado a lo largo de esta tesis.

Al permitir el uso de cualquier esquema de ramificación en combinación con cualquier relajación que no necesariamente involucrara las variables discretas, la programación lineal mixta-lógica agregó la flexibilidad necesaria para mejorar los tiempos de respuesta de una manera contundente, como fué el caso de los ejemplos de calendarización de trabajos y de modelos para variables semicontinuas en la síntesis de procesos.

Ya que el esquema de programación lineal mixta-lógica incluyó sólo las variables continuas en los problemas lineales, los tamaños de los problemas lineales a resolver en cada nodo del árbol de búsqueda fueron considerablemente más pequeños que los que se necesitaron en los árboles de programación lineal mixta-entera. Esta ventaja quedó dramáticamente comprobada en el problema de la fiesta, donde la diferencia en los tiempos de CPU entre las dos metodologías fué tan grande, aunque, en otros casos como los de la localización de almacenes, la ventaja de remover las variables enteras se perdió por la necesidad de agregar cortes para tener una buena relajación.

Se pudieron obtener valores de variables lógicas, fijadas por procedimientos de inferencia simbólica en cada nodo, y aunque en muchos casos, estas mismas variables podrían haber sido fijadas resolviendo la relajación convencional continua, el procesamiento lógico fué mucho más rápido, ya que pudo ser realizado por medio de un procedimiento de resolución unitaria, lineal en tiempo. Como pudo observarse en los resultados, en el problema de la fiesta, el procesamiento lógico fué mucho más poderoso que la programación lineal mixta-entera. También, en los ejemplos de la calendarización de trabajos y en el uso de variables semicontinuas en los problemas de síntesis de procesos, la resolución unitaria fijó variables con valores de verdadero o falso, mientras que estas

mismas variables recibieron valores fraccionales en la relajación continua de la programación mixta-entera, lo cual favoreció enormemente los tiempo de cómputo, y el número de nodos de los árboles de búsqueda.

La programación lineal mixta-lógica pudo proporcionar relajaciones más fuertes de varias maneras: algunas veces, los cortes válidos para las restricciones disyuntivas pudieron ser más fuertes que la relajación continua de cualquier formulación obvia 0-1 y, el procesamiento lógico pudo generar restricciones lógicas (cortes lógicos) cuyas relajaciones lineales pudieron ser agregadas al modelo lineal y se utilizaron como planos en un algoritmo convencional de ramificación y corte, pero el procesamiento lógico proporcionó una forma sistemática de generarlos. Esto quedó ilustrado en los problemas de localización de almacenes y de la fiesta progresiva.

El punto de vista lógico pudo sugerir cortes lógicos que pudieron derivarse fácilmente de un entendimiento intuitivo del problema pero no fueron fácilmente revelados por el análisis poliédrico convencional. También pudo sugerir cortes lógicos no-válidos, los cuales no cambiaron el valor óptimo pero fueron más fuertes porque excluyeron algunas soluciones factibles. Los problemas de redes de proceso químico fueron un excelente ejemplo de la aplicación de ambos tipos de cortes.

Finalmente, es un hecho el que, en muchos casos, las variables enteras, particularmente las variables 0-1 intentan expresar lo que originalmente fué concebido como restricciones lógicas; es por eso que, en dichos casos, el enfoque de la programación lineal mixta-lógica permitió una formulación mucho más natural. En otros casos, los problemas pudieron ser expresados mejor con variables discretas multivaluadas. A su vez, las restricciones en aquellas variables que no pudieron ser expresadas en forma de desigualdad pudieron ser formuladas con predicados que eran más naturales para los algoritmos de procesamiento lógico. Esto quedó más que comprobado con los ejemplos de la "fiesta progresiva" y de síntesis de procesos.

En conclusión, se puede considerar a la programación lineal mixta-lógica como un entorno de herramientas poderosas y diversas, para la solución de problemas continuos con elementos discretos, que representa un enfoque importante, útil y en cierto sentido novedoso en el campo de la optimización mixta-entera.

APENDICE: Presentación detallada de los ejemplos analizados

En este apéndice se presentan los datos completos de los ejemplos utilizados para probar los diferentes modelos y los valores óptimos de la función objetivo y de las variables discretas y continuas más importantes de los problemas analizados en el capítulo 6.

1. Problemas en Síntesis de Procesos

A continuación se muestran los datos, las cláusulas lógicas utilizadas, el diagrama de separación y los resultados obtenidos para el problema de separación de 5 componentes:

Alimentacion Inicial Ftot = 1000 kgmol/hr Composicion A = 0.2
COSTO DE SERVICIOS: (Fraccion Mol) B = 0.4
Agua de Enfriamiento Cc = 1.3(10^3$hr/10^6kJyr) C = 0.2
Vapor Ch = 34 (10^3$hr/10^6kJyr) E = 0.05 D = 0.15

		Costo de Inversion		Coeficientes	Fracciones de Recuperacion en la Superestructura							
	Columna de	FIJO	VARIABLE	de Calor								
k	Separacion	Z k	β κ	K k								
1	A/BCDE	120	0.041	0.041	ξ	A	= 0.2000	ξ	BCDE	= 0.8000		
2	AB/CDE	107	0.049	0.049	ξ	AB	= 0.6000	ξ	CDE	= 0.4000		
3	ABC/DE	92	0.038	0.038	ξ	ABC	= 0.8000	ξ	DE	= 0.2000		
4	ABCD/E	101	0.035	0.035	ξ	ABCD	= 0.9500	ξ	E	= 0.0500		
5	B/CDE	86	0.051	0.051	ξ	B	= 0.5000	ξ	CDE	= 0.5000		
6	BC/DE	75	0.036	0.036	ξ	BC	= 0.7500	ξ	DE	= 0.2500		
7	BCD/E	67	0.030	0.030	ξ	BCD	= 0.9375	ξ	E	= 0.0625		
8	A/BCD	75	0.028	0.028	ξ	A	= 0.2105	ξ	BCD	= 0.7895		
9	AB/CD	156	0.042	0.042	ξ	AB	= 0.6316	ξ	CD	= 0.2684		
10	ABC/D	76	0.054	0.054	ξ	ABC	= 0.8421	ξ	D	= 0.1579		
11	A/BC	38	0.040	0.040	ξ	A	= 0.7500	ξ	BC	= 0.2500		
12	AB/C	66	0.047	0.047	ξ	AB	= 0.7500	ξ	C	= 0.2500		
13	B/CD	125	0.024	0.024	ξ	B	= 0.5333	ξ	CD	= 0.4667		
14	BC/D	44	0.039	0.039	ξ	BC	= 0.8000	ξ	D	= 0.2000		
15	C/DE	86	0.042	0.042	ξ	C	= 0.5000	ξ	DE	= 0.5000		
16	CD/E	75	0.040	0.040	ξ	C	= 0.8750	ξ	E	= 0.1250		
17	A/B	112	0.022	0.022	ξ	A	= 0.3333	ξ	B	= 0.6667		
18	B/C	37	0.036	0.036	ξ	B	= 0.6667	ξ	C	= 0.3333		
19	C/D	58	0.440	0.044	ξ	C	= 0.5714	ξ	D	= 0.4286		
20	D/E	102	0.430	0.043	ξ	D	= 0.7500	ξ	E	= 0.2500		

Tabla 1.1 *Valores de los parámetros para el modelo de separación de 5 componentes*

Cláusulas en Forma Conjuntiva Normal

$\neg Y_1 \vee Y_5 \vee Y_6 \vee Y_7$	$\neg Y_5 \vee Y_1$
$\neg Y_6 \vee Y_1$	$\neg Y_7 \vee Y_1$
$\neg Y_2 \vee Y_{17}$	$\neg Y_2 \vee Y_{15} \vee Y_{16}$
$\neg Y_3 \vee Y_{11} \vee Y_{12}$	$\neg Y_3 \vee Y_{20}$
$\neg Y_4 \vee Y_8 \vee Y_9 \vee Y_{10}$	$\neg Y_8 \vee Y_4$
$\neg Y_9 \vee Y_4$	$\neg Y_{10} \vee Y_4$
$\neg Y_5 \vee Y_{15} \vee Y_{16}$	$\neg Y_6 \vee Y_{18}$
$\neg Y_6 \vee Y_{20}$	$\neg Y_7 \vee Y_{13} \vee Y_{14}$
$\neg Y_8 \vee Y_{13} \vee Y_{14}$	$\neg Y_9 \vee Y_{17}$
$\neg Y_9 \vee Y_{19}$	$\neg Y_{10} \vee Y_{11} \vee Y_{12}$
$\neg Y_{11} \vee Y_{18}$	$\neg Y_{12} \vee Y_{17}$
$\neg Y_{13} \vee Y_{19}$	$\neg Y_{14} \vee Y_{18}$
$\neg Y_{15} \vee Y_{20}$	$\neg Y_{16} \vee Y_{19}$
$\neg Y_{11} \vee Y_3 \vee Y_{10}$	$\neg Y_{12} \vee Y_3 \vee Y_{10}$
$\neg Y_{13} \vee Y_7 \vee Y_8$	$\neg Y_{14} \vee Y_7 \vee Y_8$
$\neg Y_{15} \vee Y_2 \vee Y_5$	$\neg Y_{16} \vee Y_2 \vee Y_5$
$\neg Y_{17} \vee Y_2 \vee Y_9 \vee Y_{12}$	$\neg Y_{18} \vee Y_6 \vee Y_{11} \vee Y_{14}$
$\neg Y_{19} \vee Y_9 \vee Y_{13} \vee Y_{16}$	$\neg Y_{20} \vee Y_3 \vee Y_6 \vee Y_{15}$

Cláusulas Extendidas

$Y_5 + Y_6 + Y_7 \geq 1$
$Y_{15} + Y_{16} \geq 1$
$Y_{11} + Y_{12} \geq 1$
$Y_8 + Y_9 + Y_{10} \geq 1$
$Y_{13} + Y_{14} \geq 1$
$Y_3 + Y_{10} \geq 1$
$Y_7 + Y_8 \geq 1$
$Y_2 + Y_5 \geq 1$
$Y_2 + Y_9 + Y_{12} \geq 1$
$Y_6 + Y_{11} + Y_{14} \geq$
$Y_3 + Y_9 + Y_{16} \geq 1$
$Y_3 + Y_6 + Y_{15} \geq 1$

$Y_1 + Y_2 + Y_3 + Y_4 \geq 1$	$Y_1 + Y_2 + Y_3 + Y_4 \leq 1$
$\sum_{i=1}^{20} Y_i \geq 4$	$\sum_{i=1}^{20} Y_i \leq 4$

Tabla 1.2 *Cláusulas lógicas para el modelo de separación de 5 componentes*

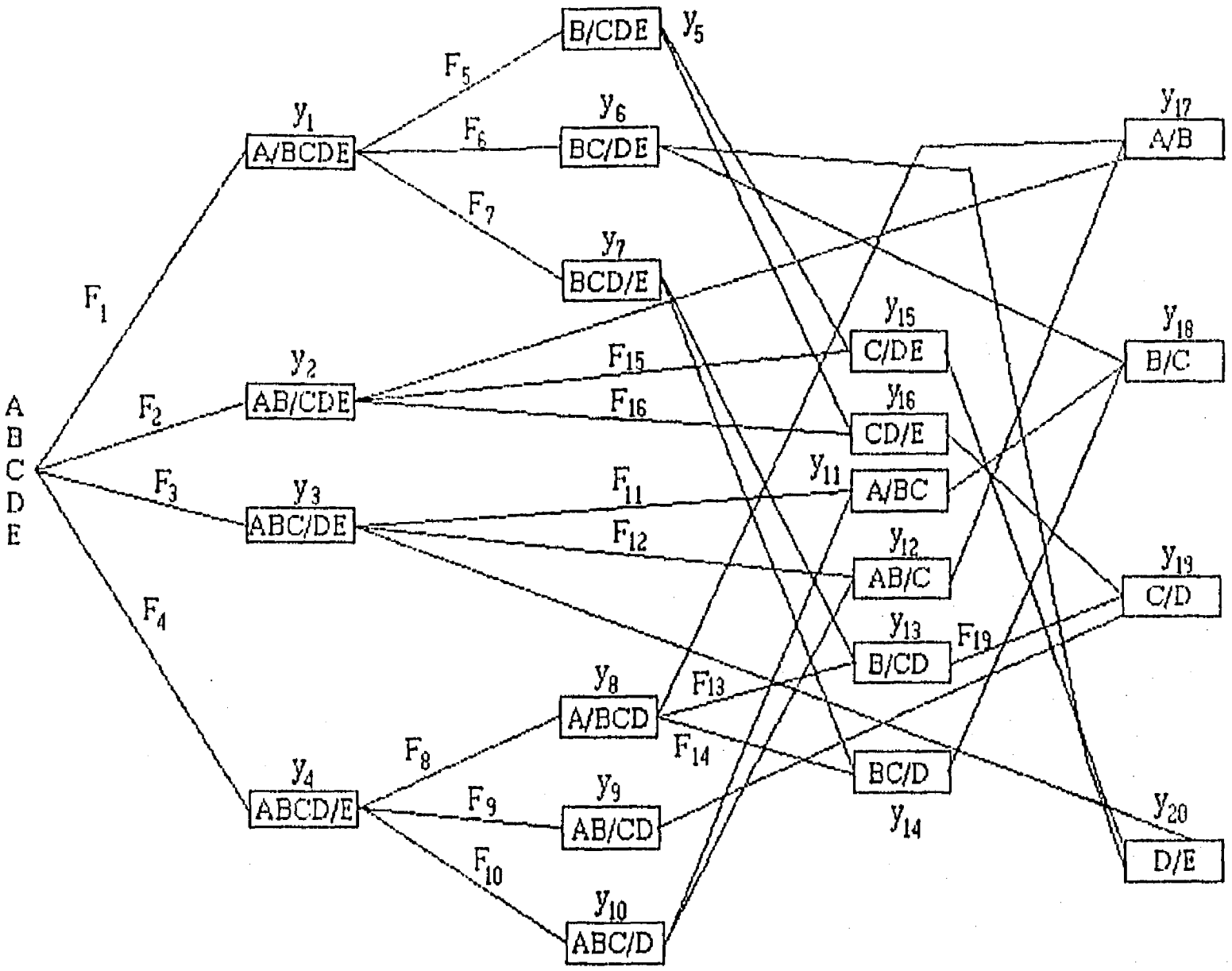


Fig. 1.1 *Superestructura de la red de separación de 5 componentes*

Como puede observarse en la Tabla 1.3, las unidades necesarias para obtener el costo mínimo de la red de separación de 5 componentes, corresponden a las columnas de destilación indicadas por y_4, y_8, y_{13} y y_{19}. El costo mínimo obtenido para este modelo es de $\$ 4304.09 \times 10^5$ y los resultados óptimos de los flujos de materia y energía se dan en la siguiente tabla.

FLUJOS	kgmol/hr	CALOR	Btu
F Tot =	1000.00	QTot =	119.43
F[4] =	1000.00	Q[4] =	35.00
F[8] =	950.00	Q[8] =	26.60
F[13] =	750.02	Q[13] =	18.00
F[19] =	350.04	Q[19] =	15.40

Tabla 1.3 *Valores óptimos de los flujos de materia y calor para el modelo de separación de 5 componentes (todos los demás valores son cero)*

Para el problema de 6 componentes, los datos, las cláusulas lógicas, el diagrama y los resultados obtenidos fueron:

Alimentacion Inicial Ftot = 1000 kgmol/hr Composicion A = 0.0714
COSTO DE SERVICIOS: (Fraccion Mol) B = 0.12084
Agua de Enfriamiento Cc = 1.3(10^3$hr/10^6kJyr) E =0.2857 C = 0.14284
Vapor Ch = 34 (10^3$hr/10^6kJyr) F =0.0714 D = 0.2857

		Costo de Inversion		Coeficientes	Fracciones de Recuperacion			
	Columna de	FIJO	VARIABLE	de Calor	de la superestructura			
k	Separacion	Z k	β κ	K k				
1	A/BCDEF	2102	0.13	0.0370	ξ A = 0.0714		ξ BCDEF = 0.9286	
2	AB/CDEF	2134	0.16	0.0510	ξ AB = 0.2143		ξ CDEF = 0.7857	
3	ABC/DEF	2120	0.4	0.0540	ξ ABC = 0.3571		ξ DEF = 0.6429	
4	ABCD/EF	2134	0.41	0.0520	ξ ABCD = 0.6429		ξ EF = 0.3571	
5	ABCDE/F	2117	0.42	0.0320	ξ ABCDE = 0.9286		ξ F = 0.0714	
6	B/CDEF	2078	0.37	0.0450	ξ B = 0.1538		ξ CDEF = 0.8462	
7	BC/DEF	2091	0.35	0.0470	ξ BC = 0.3077		ξ DEF = 0.6923	
8	BCD/EF	2100	0.4	0.0360	ξ BCD = 0.6154		ξ EF = 0.3846	
9	BCDE/F	2065	0.28	0.0440	ξ BCDE = 0.9231		ξ F = 0.0769	
10	A/BCDE	2120	0.32	0.0410	ξ A = 0.0769		ξ BCDE = 0.9231	
11	AB/CDE	2107	0.41	0.0490	ξ AB = 0.2308		ξ CDE = 0.7692	
12	ABC/DE	2092	0.39	0.0380	ξ ABC = 0.3846		ξ DE = 0.6154	
13	ABCD/E	2101	0.26	0.0350	ξ ABCD = 0.6923		ξ E = 0.3077	
14	C/DEF	2054	0.29	0.0530	ξ C = 0.1818		ξ DEF = 0.8182	
15	CD/EF	2064	0.22	0.0370	ξ CD = 0.5455		ξ EF = 0.4545	
16	CDE/F	2077	0.18	0.0330	ξ CDE = 0.9091		ξ F = 0.0909	
17	B/CDE	2086	0.37	0.0510	ξ B = 0.1667		ξ CDE = 0.8333	
18	BC/DE	2075	0.34	0.0360	ξ BC = 0.3333		ξ DE = 0.6667	
19	BCD/E	2067	0.15	0.0300	ξ BCD = 0.6667		ξ E = 0.3333	
20	A/BCD	2075	0.2	0.0280	ξ A = 0.0909		ξ BCD = 0.9091	
21	AB/CD	2156	0.36	0.0420	ξ AB = 0.3333		ξ CD = 0.6667	
22	ABC/D	2076	0.25	0.0540	ξ ABC = 0.5556		ξ D = 0.4444	
23	D/EF	2060	0.33	0.0380	ξ D = 0.4444		ξ EF = 0.5556	
24	DE/F	2082	0.19	0.0360	ξ DE = 0.8889		ξ F = 0.1111	
25	C/DE	2086	0.4	0.0420	ξ C = 0.2000		ξ DE = 0.8000	
26	CD/E	2075	0.38	0.0400	ξ CD = 0.6000		ξ E = 0.4000	
27	B/CD	2025	0.1	0.0240	ξ B = 0.2500		ξ CD = 0.7500	
28	BC/D	2044	0.11	0.0390	ξ BC = 0.5000		ξ C = 0.5000	
29	A/BC	2038	0.14	0.0400	ξ A = 0.2000		ξ BC = 0.8000	
30	AB/C	2066	0.21	0.0470	ξ AB = 0.6000		ξ C = 0.4000	
31	E/F	2039	0.2	0.0460	ξ E = 0.8000		ξ F = 0.2000	
32	D/E	2102	0.35	0.0430	ξ D = 0.5000		ξ E = 0.5000	
33	C/D	2058	0.19	0.0440	ξ C = 0.3333		ξ D = 0.6667	
34	B/C	2037	0.08	0.0360	ξ B = 0.4583		ξ C = 0.5417	
35	A/B	2112	0.39	0.0220	ξ A = 0.3714		ξ B = 0.6286	

Tabla 1.4 *Valores de los parámetros para el modelo de separación de 6 componentes*

Cláusulas en Forma Conjuntiva Normal

$\neg Y_1 \vee Y_6 \vee Y_7 \vee Y_8 \vee Y_9$	$\neg Y_6 \vee Y_1$
$\neg Y_7 \vee Y_1$	$\neg Y_8 \vee Y_1$
$\neg Y_9 \vee Y_1$	$\neg Y_2 \vee Y_{14} \vee Y_{15} \vee Y_{16}$
$\neg Y_2 \vee Y_{35}$	$\neg Y_3 \vee Y_{23} \vee Y_{24}$
$\neg Y_3 \vee Y_{29} \vee Y_{30}$	$\neg Y_4 \vee Y_{20} \vee Y_{21} \vee Y_{22}$
$\neg Y_4 \vee Y_{31}$	$\neg Y_5 \vee Y_{10} \vee Y_{11} \vee Y_{12} \vee Y_{13}$
$\neg Y_{10} \vee Y_5$	$\neg Y_{11} \vee Y_5$
$\neg Y_{12} \vee Y_5$	$\neg Y_{13} \vee Y_5$
$\neg Y_6 \vee Y_{14} \vee Y_{15} \vee Y_{16}$	$\neg Y_7 \vee Y_{23} \vee Y_{24}$
$\neg Y_7 \vee Y_{34}$	$\neg Y_8 \vee Y_{27} \vee Y_{28}$
$\neg Y_8 \vee Y_{31}$	$\neg Y_9 \vee Y_{17} \vee Y_{18} \vee Y_{19}$
$\neg Y_{10} \vee Y_{17} \vee Y_{18} \vee Y_{19}$	$\neg Y_{11} \vee Y_{25} \vee Y_{26}$
$\neg Y_{11} \vee Y_{35}$	$\neg Y_{12} \vee Y_{29} \vee Y_{30}$
$\neg Y_{12} \vee Y_{32}$	$\neg Y_{13} \vee Y_{20} \vee Y_{21} \vee Y_{22}$
$\neg Y_{14} \vee Y_{23} \vee Y_{24}$	$\neg Y_{15} \vee Y_{33}$
$\neg Y_{15} \vee Y_{31}$	$\neg Y_{16} \vee Y_{25} \vee Y_{26}$
$\neg Y_{17} \vee Y_{25} \vee Y_{26}$	$\neg Y_{18} \vee Y_{32}$
$\neg Y_{18} \vee Y_{34}$	$\neg Y_{19} \vee Y_{27} \vee Y_{28}$
$\neg Y_{20} \vee Y_{27} \vee Y_{28}$	$\neg Y_{21} \vee Y_{33}$
$\neg Y_{21} \vee Y_{35}$	$\neg Y_{22} \vee Y_{29} \vee Y_{30}$
$\neg Y_{23} \vee Y_3 \vee Y_7 \vee Y_{14}$	$\neg Y_{23} \vee Y_{31}$
$\neg Y_{24} \vee Y_3 \vee Y_7 \vee Y_{14}$	$\neg Y_{24} \vee Y_{32}$
$\neg Y_{25} \vee Y_{11} \vee Y_{16} \vee Y_{17}$	$\neg Y_{25} \vee Y_{32}$
$\neg Y_{26} \vee Y_{11} \vee Y_{16} \vee Y_{17}$	$\neg Y_{26} \vee Y_{33}$
$\neg Y_{27} \vee Y_8 \vee Y_{19} \vee Y_{20}$	$\neg Y_{27} \vee Y_{33}$
$\neg Y_{28} \vee Y_8 \vee Y_{19} \vee Y_{20}$	$\neg Y_{28} \vee Y_{34}$
$\neg Y_{29} \vee Y_3 \vee Y_{12} \vee Y_{22}$	$\neg Y_{29} \vee Y_{34}$
$\neg Y_{30} \vee Y_3 \vee Y_{12} \vee Y_{22}$	$\neg Y_{30} \vee Y_{35}$
$\neg Y_{31} \vee Y_4 \vee Y_8 \vee Y_{15} \vee Y_{23}$	$\neg Y_{32} \vee Y_{12} \vee Y_{18} \vee Y_{24} \vee Y_{25}$
$\neg Y_{33} \vee Y_{15} \vee Y_{21} \vee Y_{26} \vee Y_{27}$	$\neg Y_{34} \vee Y_7 \vee Y_{18} \vee Y_{28} \vee Y_{29}$
$\neg Y_{35} \vee Y_2 \vee Y_{11} \vee Y_{21} \vee Y_{30}$	

Cláusulas Extendidas

$Y_6 + Y_7 + Y_8 + Y_9 \geq 1$	$Y_{10} + Y_{11} + Y_{12} + Y_{13} \geq 1$
$Y_{14} + Y_{15} + Y_{16} \geq 1$	$Y_{17} + Y_{18} + Y_{19} \geq 1$
$Y_{20} + Y_{21} + Y_{22} \geq 1$	$Y_{23} + Y_{24} \geq 1$
$Y_{25} + Y_{26} \geq 1$	$Y_{27} + Y_{28} \geq 1$
$Y_{29} + Y_{30} \geq 1$	
$Y_1 + Y_2 + Y_3 + Y_4 + Y_5 \geq 1$	$Y_1 + Y_2 + Y_3 + Y_4 + Y_5 \leq 1$
$\sum_{i=1}^{35} Y_i \geq 5$	$\sum_{i=1}^{35} Y_i \leq 5$

Tabla 1.5 *Cláusulas lógicas para el modelo de separación de 6 componentes*

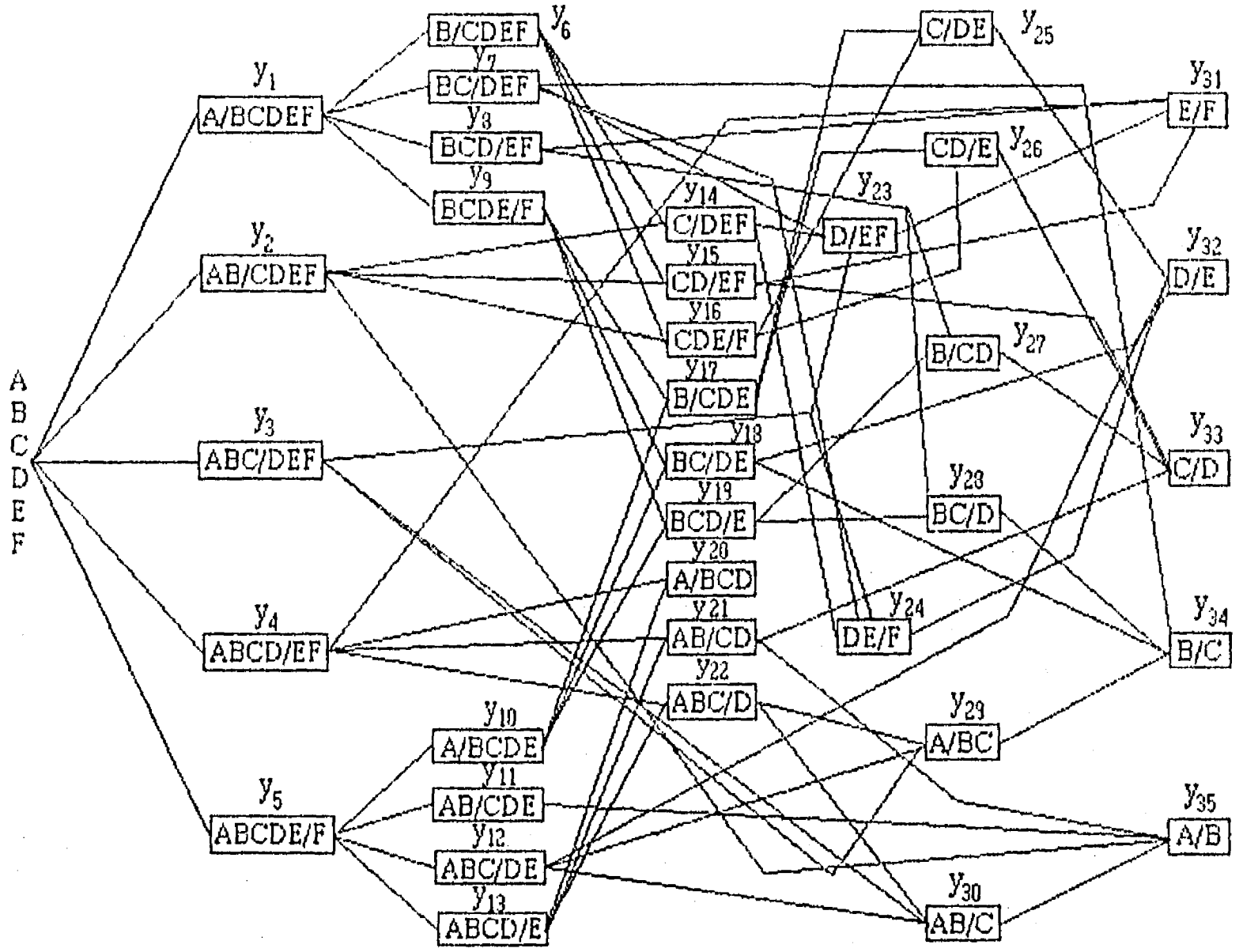


Fig. 1.2 *Superestructura de la red de separación de 6 componentes*

Como puede observarse en la Tabla 1.6, las unidades necesarias para obtener el costo mínimo de la red de separación de 6 componentes, corresponden a las columnas de destilación indicadas por y_1, y_8, y_{28}, y_{31} y y_{34}. El costo mínimo obtenido para este modelo es de $ 18,170.35 x 105 y los resultados óptimos de los flujos de materia y energía se dan en la siguiente tabla.

FLUJOS	kgmol/hr	CALOR	Btu
F Tot =	1000.00	QTot =	119.43
F[1] =	1000.00	Q[1] =	37.00
F[8] =	928.60	Q[8] =	33.43
F[28] =	571.46	Q[28] =	22.29
F[31] =	357.14	Q[31] =	16.43
F[34] =	285.73	Q[34] =	10.29

Tabla 1.6 *Valores de los flujos óptimos de materia y calor para el modelo de separación de 6 componentes (todos los demás valores son cero)*

El primer problema utilizado para mostrar la efectividad de la programación lineal mixta-lógica en el modelado y tratamiento de las variables semicontinuas fué una red con 10

procesos químicos. En la Fig. 1.3 se muestra la superestructura de la red, y en las Tablas 1.7-1.11 los datos de entrada utilizados en el modelo de síntesis de esta red de procesos.

Número de procesos = 10
Número de periodos = 1
Número de productos químicos = 6
Número de mercados = 1
Tasa de interés = 0.10
Tasa de impuesto = 0.45
Tiempo de vida proyectada = 100
Límite en el número de expansiones / proceso=2
Factor de capital de trabajo/proceso = 0.15
Capacidades existentes (kton/año) = 0
Valor de rescate/proceso = 0.10
Límite inferior sobre la expansión/proceso = 0 kton/año
Límite superior sobre la expansión/proceso = 100 kton/aúo
Límite de la Inversión = $ 1E5/AÑ0

Tabla 1.7 *Datos generales de la superestructura para la síntesis de 10 procesos*

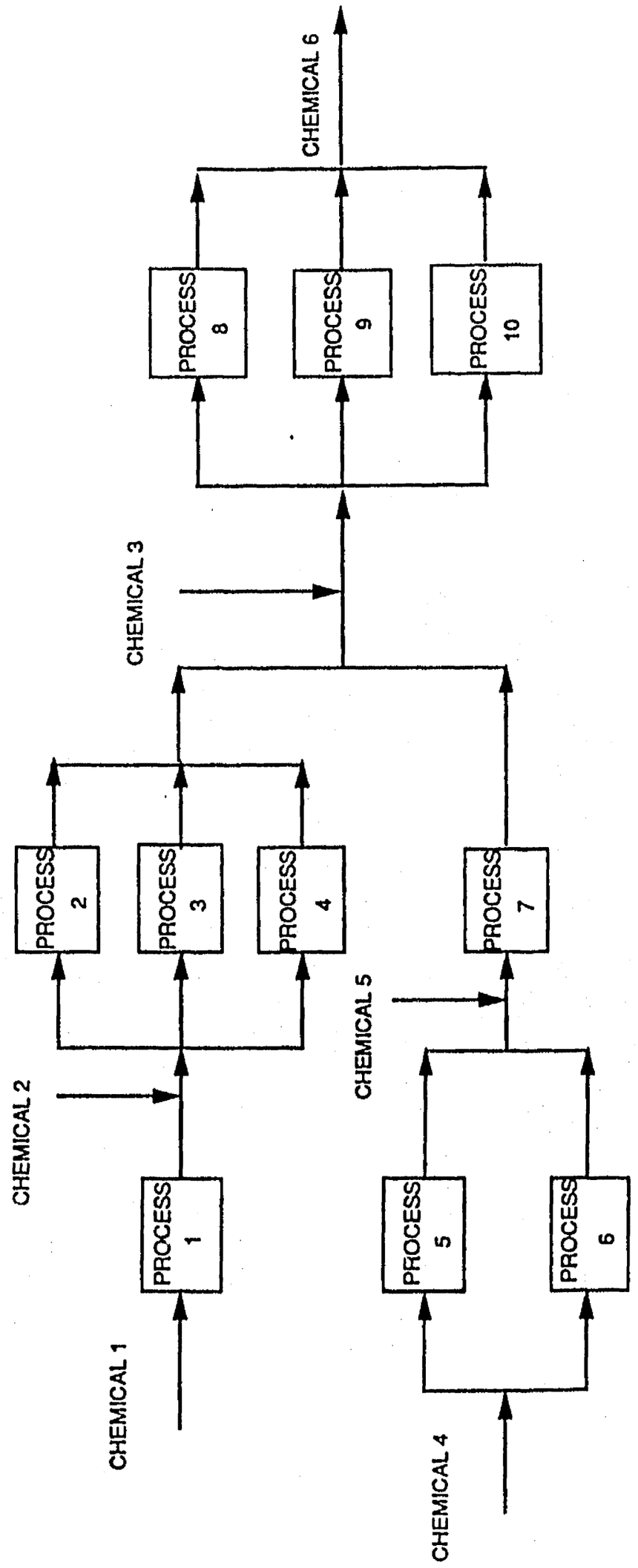
Proceso	Producto principal	Coeficiente de Inversion Variable $1E2	Fijo $1E5	Costo de operacion/uni producto $1E2/ton
1	2	2.1	85	0.30
2	3	3.4	30	0.55
3	3	2.5	25	0.70
4	3	2.2	10	0.60
5	5	4.7	170	0.30
6	5	5.0	180	0.80
7	3	2.1	11	1.00
8	6	2.6	110	0.70
9	6	3.7	130	0.90
10	6	4.9	200	1.00

Tabla 1.8 *Número del producto principal, coeficientes de inversión y costo de operación por unidad de producto para cada proceso (síntesis de 10 procesos)*

Producto Químico	Costo de compra/ unidad de producto (kton/año)	Lím.sup. Compra (kton/año)	VENTAS Lím.sup. (kton/año)	VENTAS Precios ($1E2/ton)	Costo de Inventario
1	23.39	200			23.39
2					44.74
3					60.00
4	24.15	200			24.15
5					36.84
6			145	120	120.00

Tabla 1.8 *Costos y precios de venta de cada producto químico (síntesis de 10 procesos)*

Fig. 1.3 *Superestructura para la síntesis de una red de 10 procesos*



Proceso	PRODUCTO QUIMICO 1	2	3	4	5	6
1	1.11	-1.00				
2		1.22	-1.00			
3		1.05	-1.00			
4		1.05	-1.00			
5				1.18	-1.00	
6				1.11	-1.00	
7			-1.00		1.05	
8			1.06			-1.00
9			1.11			-1.00
10			1.18			-1.00

Tabla 1.9 *Coeficientes del balance de materia: positivo para entrada, negativo para salida y cero para producto no presente en ese proceso (síntesis de 10 procesos)*

En la Tabla 1.10 se presentan las cláusulas lógicas que se utilizaron para resolver este problema,

Cláusulas en Forma Conjuntiva Normal

$\neg Y_5 \vee Y_7$ $\quad\quad$ $\neg Y_6 \vee Y_7$

$\neg Y_2 \vee Y_1$ $\quad\quad$ $\neg Y_3 \vee Y_1$

$\neg Y_4 \vee Y_1$ $\quad\quad$ $\neg Y_7 \vee Y_5 \vee Y_6$

$\neg Y_1 \vee Y_2 \vee Y_3 \vee Y_4$ $\quad\quad$ $\neg Y_1 \vee Y_8 \vee Y_9 \vee Y_{10}$

$\neg Y_7 \vee Y_8 \vee Y_9 \vee Y_{10}$ $\quad\quad$ $\neg Y_8 \vee Y_1 \vee Y_7$

$\neg Y_9 \vee Y_1 \vee Y_7$ $\quad\quad$ $\neg Y_{10} \vee Y_1 \vee Y_7$

Cláusulas Extendidas

$Y_5 + Y_6 \geq 1$ $\quad\quad$ $Y_2 + Y_3 + Y_4 \geq 1$

$Y_8 + Y_9 + Y_{10} \geq 1$

Tabla 1.10 *Cláusulas lógicas para el modelo de síntesis de la red de 10 procesos*

En este modelo de síntesis de una red con 10 procesos, se consideró el precio de venta del producto 6 y el de compra de los productos 1 y 4 que fueron las entradas activas de la red, en la superestructura de la Fig. 1.3, para introducir semicontinuidad en dichas variables. Sin utilizar intervalos, el valor óptimo de la variable correspondiente al flujo de la demanda del producto 6 fué de 100 kton/año y el de las variables de compra de los productos 1 y 2, fueron de 111 kton/año y 12.543 kton/año, respectivamente. En el primer caso que corresponde a la versión 1 de la Tabla 6.2 del capítulo 6, los intervalos considerados fueron:

Tipo de corriente	Producto	Mercado	Variable	INTERVALOS			
DEMANDA	6	1	S61	S61 = 0	20<S61<50	100<S61<120	130<S61<145
COMPRA	1	1	P11	P11 = 0	20<P11<80	100<P11<120	150<P11<200
COMPRA	2	1	P41	P41 = 0	10<P41<50	100<P41<120	150<P11<200

Tabla 1.11(a) *Intervalos de la versión 1 de la síntesis de 10 procesos químicos*

Para este caso, la utilidad óptima obtenida en la función objetivo fué de \$7,834.75 E2 y las unidades presentes en el modelo fueron: y_1, y_4, y_6, y_7 y y_8. El segundo caso, correspondiente a la versión 2 de la misma Tabla, se utilizaron los siguientes intervalos,

Tipo de corriente	Producto	Mercado	Variable	INTERVALOS
DEMANDA	6	1	S61	S61 = 0 50<S61<90 110<S61<120 130<S61<145
COMPRA	1	1	P11	P11 = 0 20<P11<80 100<P11<120 150<P11<200
COMPRA	2	1	P41	P41 = 0 20<P41<80 100<P41<120 150<P11<200

Tabla 1.11(b) *Intervalos de la versión 2 de la síntesis de 10 procesos químicos*

En este segundo caso, la utilidad óptima obtenida en la función objetivo fué de \$7,162.14 E2 y las unidades activadas para la superestructura sólo fueron y_1, y_4 y y_8.

El segundo caso considerado en las redes de procesos químicos fué la síntesis de la superestructura con 38 procesos químicos mostrada en la Fig. 1.4. Los datos de entrada y las cláusulas lógicas para este ejemplo se muestran en las tablas 1.12- 1.16

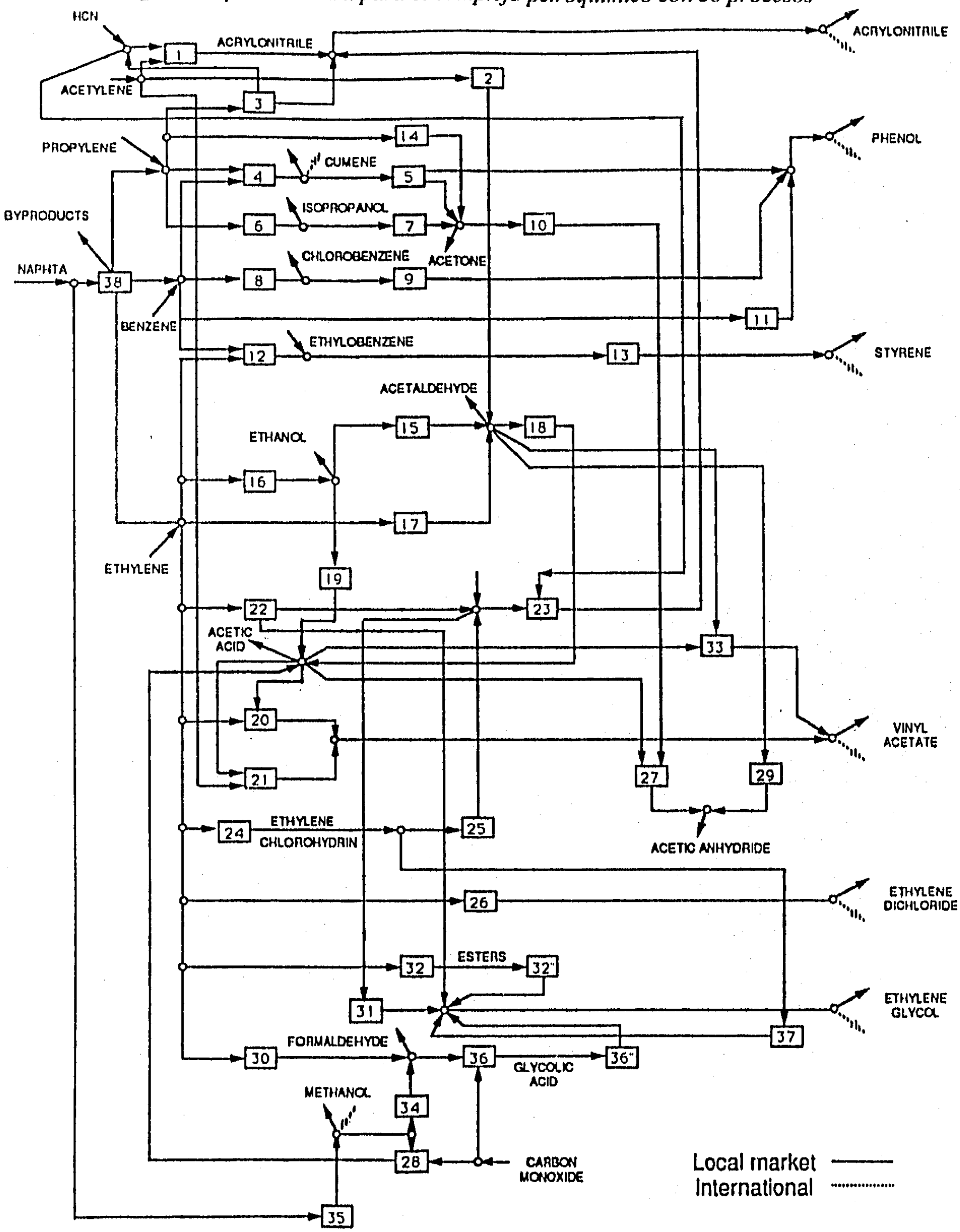
Número de procesos = 38	Límites sobre la expansión
Número de periodos = 1	para cada proceso
Número de productos químicos = 28	kton/año
Número de mercados = 2	Inferiores = 0
Tasa de interés = 0.10	Superiores = 500
Tasa de impuesto = 0.45	
Tiempo de vida proyectada = 100	

Tabla 1.12 *Datos generales de la superestructura para la síntesis de 38 procesos*

Cláusulas en Forma conjuntiva Normal	
$Y_4 \vee \neg Y_5$	$Y_6 \vee \neg Y_7$
$Y_8 \vee \neg Y_9$	$Y_5 \vee Y_7 \vee \neg Y_{10} \vee Y_{14}$
$\neg Y_{10} \vee Y_{27}$	$\neg Y_{12} \vee Y_{13}$
$\neg Y_{15} \vee Y_{16}$	$Y_2 \vee Y_{15} \vee Y_{17} \vee \neg Y_{18}$
$Y_{16} \vee Y_{19}$	$Y_{18} \vee Y_{19} \neg Y_{20} \ Y_{28}$
$Y_{18} \vee Y_{19} \ \neg Y_{21} \ Y_{28}$	$\neg Y_{22} \vee Y_{23} \vee Y_{31}$
$\neg Y_{24} \vee Y_{25} \vee Y_{37}$	$Y_{24} \vee \neg Y_{25}$
$Y_{23} \vee \neg Y_{25} \vee Y_{31}$	$Y_{18} \vee Y_{19} \vee \neg Y_{27} \vee Y_{28}$
$Y_{10} \vee \neg Y_{27}$	$Y_2 \vee Y_{15} \vee Y_{17} \vee \neg Y_{29}$
$Y_{18} \vee Y_{19} \vee Y_{28} \vee \neg Y_{33}$	$Y_2 \vee Y_{15} \vee Y_{17} \vee \neg Y_{33}$
$Y_{30} \vee Y_{34} \vee \neg Y_{36}$	$Y_{24} \vee \neg Y_{37}$
$Y_3 \vee Y_4 \vee Y_6 \vee Y_{14} \vee \neg Y_{38}$	$Y_4 \vee Y_8 \vee Y_{11} \vee Y_{12} \vee \neg Y_{38}$
$Y_{12} \vee Y_{16} \vee Y_{17} \vee Y_{20} \vee Y_{22} \vee Y_{24} \vee Y_{26} \vee Y_{30} \vee Y_{32} \vee \neg Y_{38}$	

Tabla 1.13 *Cláusulas lógicas para el modelo de síntesis de la red de 38 procesos*

Fig. 1.4 *Superestructura para el complejo petroquímico con 38 procesos*



Proceso	Producto principal	Capacidades Existentes	Coeficiente de Inversion Variable $1E2	Fijo $1E5	Costo de operacion/uni producto $1E2/ton
1	11		3.7	337.1	0.8
2	12		2.5	125	1.5
3	11		3.1	321	1
4	14		1	80.8	1.1
5	13		2.1	96.6	1.1
6	15		0.8	62	0.7
7	13		1.3	105	1
8	16		1.5	102	2.5
9	17		1.5	115	1.5
10	27		0.9	17.5	2
11	17		1.7	118	2
12	8	39.9	0.6	50	0.7
13	18	25	3	237	0.7
14	13		1.2	52.9	2
15	12		2	120	1.6
16	19	300	0.8	37	1
17	12		1.8	110	1.3
18	20		1.9	40	0.8
19	20		2.5	42	1.5
20	21		2.4	75	2.5
21	21		2.4	75	3
22	5		5.8	101	1.5
23	11		4	353	1.3
24	28		3	150	2
25	5		2.5	130	2.3
26	23		1.5	85	1
27	22		0.8	25	1.5
28	20		2	41	1
29	22		0.9	27	2
30	25		3	120	2
31	24		3.9	87	3
32	24		3.5	85	2
33	21		0.9	80	3
34	25		2.6	111	0.4
35	10		0.7	54	1.3
36	24		4.3	89	3
37	24		6.8	131	3.5
38	4	200	0.9	12.8	1

Tabla 1.14 *Número del producto principal, capacidades existentes, coeficientes de inversión y costo de operación por unidad de producto para cada proceso (síntesis de 38 procesos)*

Producto Químico	Nombre	Costo de compra/uni. de producto $1E2/ton		Límites superiores Compra		VENTAS Lim.sup.kton/año		Precios $1E2/ton	
		Mercado	Costo	Mercado	kton/año	Merc.1	Merc.2	Merc.1	Merc.2
1	Acido Nitrico	1	8.2	1	200				
2	Propileno	1	4.0	1	125				
3	Benzeno	1	4.4	1	200				
4	Etileno	1	3.5	1	100				
5	Oxido de etileno	1	8.6	1	40				
6	Acetileno	1	5.5	1	250				
7	Dióxido de carbón	1	0.4	1	600				
8	Etilbenzeno	1	6.6	1	40				
9	Nafta	1	4.1	1	1000				
10	Metanol	2	2.6	2	20	40		2.4	
11	Acrilonitrilo					150	25	8.4	8.6
12	Acetaldehido					30		6.3	
13	Acetona					75		6.4	
14	Cumeno					60	25	5.5	5.7
15	Isopropanol					30		5.4	
16	Clorobenzeno					30		12.8	
17	Fenol					50	7.5	7.3	7.5
18	Estireno					150	50	7.5	7.7
19	Etanol					70		6.1	
20	Acido acético					30		5.4	
21	Vinil acetato					75	20	7.4	7.6
22	Anhidrido acético					100		7.9	
23	Dicloro-etileno					75	5	3.1	3.3
24	Etilenglicol					250	20	7.5	7.7
25	Formaldehído					50		1.6	
26	Productos secundarios					500		3.3	
27	Keteno								
28	Clorohidrinetileno								

Tabla 1.15 *Costos y límites de compra y venta en cada mercado, para cada producto químico (síntesis de 38 procesos)*

	PRODUCTO QUIMICO													
Proceso	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	0.58					0.63					-1.00			
2						0.64						-1.00		
3	-0.06	1.25									-1.00			
4		0.40	0.69											-1.00
5													-1.00	2.30
6		0.74												
7													-1.00	
8			1.00											
9														
10													1.57	
11			1.01											
12			0.76	0.28				-1.00						
13								1.14						
14		0.78											-1.00	
15												-1.00		
16				0.60										
17				0.67								-1.00		
18												1.10		
19														
20				0.35										
21						0.32								
22				0.88	-1.00									
23	0.56				0.92						-1.00			
24				0.39										
25					-1.00									
26				0.30										
27														
28							0.56			0.56				
29												1.20		
30				1.17										
31					0.75									
32				0.53										
33												0.60		
34										0.42				
35									0.50	-1.00				
36							0.53							
37														
38		-0.38	-0.22	-1.00					3.08					

Tabla 1.16(a) *Coeficientes del balance de materia: positivo para entrada, negativo para salida y cero para producto no presente en ese proceso (síntesis de38 procesos)*

	PRODUCTO QUIMICO													
Proceso	15	16	17	18	19	20	21	22	23	24	25	26	27	28
1														
2														
3														
4														
5			-1.7											
6	-1													
7	1.1													
8		-1												
9		1.26	-1											
10													-1	
11			-1											
12														
13				-1										
14														
15					1.34									
16					-1									
17														
18						-1								
19					0.98	-1								
20						0.71	-1							
21						0.72	-1							
22								-0.03						
23														
24														-1
25														1.97
26									-1					
27						0.65		-1					0.46	
28						-1								
29								-1						
30											-1			
31										-1				
32										-1				
33						0.82	-1				-1			
34														
35														
36										-1	0.57			
37										-1				1.44
38												-1.81		

Tabla 1.16(b) *Coeficientes del balance de materia: positivo para entrada, negativo para salida y cero para producto no presente en ese proceso (síntesis de 38 procesos)*

En lo que respecta a este problema se hicieron 5 corridas con diferentes valores en los intervalos y con una variación en la longitud de periodo. Para la primera versión se consideró una longitud de periodo de 10 años. Los valores óptimos sin intervalos de las corrientes que se vendieron de producto fueron: para el producto 11, 102.89 kton/año, para el producto 24, 250 kton/año y para el producto 26, 248.30 kton/año. Con respecto a

las corrientes de compra, para el producto 2 fué de 125 kton/año, para el producto 3, 75.16 kton/año, para el producto 4, 100 kton/año y para el producto 9, 422.51 kton/año.

Para las otras cuatro corridas, se consideró una longitud de periodo de 4, y los valores óptimos sin intervalos de las corrientes que se vendieron de producto, ahora fueron: para el producto 11, 0 kton/año, para el producto 24, 250 kton/año y para el producto 26, 168.48 kton/año. Con respecto a las corrientes de compra, para el producto 2, ahora fué de 50.07 kton/año, para el producto 3, 84.86 kton/año, para el producto 4, 100 kton/año y para el producto 9, 286.69 kton/año. En las Tablas 1.17 (a)-(e) se muestran los diferentes intervalos utilizados.

TIPO	Producto	Variable	INTERVALOS				
Venta	11	S11	S11 = 0	30<S11<45	55<S11<75	90<S11<120	135<S11<155
Venta	24	S24	S24 = 0	60<S24<70	120<S24<130	180<S24<190	240<S24<260
Venta	26	S26	S26 = 0	125<S26<150	200<S26<275	375<S26<400	475<S26<525
Compra	2	P02	P02 = 0	30<P02<35	60<P02<65	90<P02<95	120<P02<130
Compra	3	P03	P03 = 0	20<P03<30	45<P03<55	70<P03<80	95<P03<105
Compra	4	P04	P04 = 0	25<P04<30	50<P04<55	75<P04<80	95<P04<105
Compra	9	P09	P09 = 0	250<P09<300	400<P09<550	750<P09<800	950<P09<1050

Tabla 1.17(a) *Intervalos de la versión 1 de la síntesis de 38 procesos químicos*

Para esta versión la utilidad neta óptima fué de $4,012.79 E2 y las unidades presentes en las superestructura óptima: y_1, y_3, y_4, y_5, y_8, y_{12}, y_{13}, y_{16}, y_{17}, y_{26}, y_{28}, y_{32} y y_{38}.

TIPO	Producto	Variable	INTERVALOS				
Venta	11	S11	S11 = 0	30<S11<45	55<S11<75	90<S11<120	135<S11<155
Venta	24	S24	S24 = 0	60<S24<70	120<S24<130	180<S24<190	240<S24<260
Venta	26	S26	S26 = 0	125<S26<150	200<S26<275	375<S26<400	475<S26<525
Compra	2	P02	P02 = 0	30<P02<35	60<P02<65	90<P02<95	120<P02<130
Compra	3	P03	P03 = 0	20<P03<30	45<P03<55	70<P03<80	95<P03<105
Compra	4	P04	P04 = 0	25<P04<30	50<P04<55	75<P04<80	95<P04<105
Compra	9	P09	P09 = 0	250<P09<300	400<P09<550	750<P09<800	950<P09<1050

Tabla 1.17(b) *Intervalos de la versión 2 de la síntesis de 38 procesos químicos*

En este caso, la utilidad neta óptima fué de $1,542.81 E2 y las unidades presentes en las superestructura óptima: y_4, y_5, y_6, y_8, y_{12}, y_{13}, y_{14}, y_{16}, y_{28}, y_{32} y y_{38}.

TIPO	Producto	Variable	INTERVALOS				
Venta	11	S11	S11 = 0	30<S11<45	55<S11<75	90<S11<120	135<S11<155
Venta	24	S24	S24 = 0	60<S24<70	120<S24<130	180<S24<190	240<S24<260
Venta	26	S26	S26 = 0	125<S26<170	200<S26<275	375<S26<400	475<S26<525
Compra	2	P02	P02 = 0	30<P02<35	50<P02<65	90<P02<95	120<P02<130
Compra	3	P03	P03 = 0	20<P03<30	45<P03<55	80<P03<90	95<P03<105
Compra	4	P04	P04 = 0	25<P04<30	50<P04<55	75<P04<80	95<P04<105
Compra	9	P09	P09 = 0	250<P09<300	400<P09<550	750<P09<800	950<P09<1050

Tabla 1.17(c) *Intervalos de la versión 3 de la síntesis de 38 procesos químicos*

Aquí, la utilidad neta óptima fué de \$1,596.55 E2 y las unidades presentes en las superestructura óptima, al igual que en el caso anterior, y_4, y_5, y_6, y_8, y_{12}, y_{13}, y_{14}, y_{16}, y_{28}, y_{32} y y_{38}.

TIPO	Producto	Variable	INTERVALOS
Venta	11	S11	S11 = 0 30<S11<45 55<S11<75 90<S11<120 135<S11<155
Venta	24	S24	S24 = 0 60<S24<70 120<S24<130 180<S24<190 240<S24<260
Venta	26	S26	S26 = 0 125<S26<150 200<S26<275 375<S26<400 475<S26<525
Compra	2	P02	P02 = 0 30<P02<35 60<P02<65 90<P02<95 120<P02<130
Compra	4	P04	P04 = 0 25<P04<30 50<P04<55 75<P04<80 95<P04<105
Compra	9	P09	P09 = 0 250<P09<300 400<P09<550 750<P09<800 950<P09<1050

Tabla 1.17(d) *Intervalos de la versión 4 de la síntesis de 38 procesos químicos*

Considerando sólo 6 variables semicontinuas en vez de 7 como en los ejemplos anteriores, la utilidad neta óptima fué de \$1,542.81 E2 y las unidades presentes en las superestructura óptima, al igual que en las dos tablas anteriores, y_4, y_5, y_6, y_8, y_{12}, y_{13}, y_{14}, y_{16}, y_{28}, y_{32} y y_{38}.

TIPO	Producto	Variable	INTERVALOS
Venta	11	S11	S11 = 0 30<S11<45 55<S11<75 90<S11<120 135<S11<155
Venta	24	S24	S24 = 0 60<S24<70 120<S24<130 180<S24<190 240<S24<260
Venta	26	S26	S26 = 0 125<S26<170 200<S26<275 375<S26<400 475<S26<525
Compra	2	P02	P02 = 0 30<P02<35 50<P02<65 90<P02<95 120<P02<130
Compra	4	P04	P04 = 0 25<P04<30 50<P04<55 75<P04<80 95<P04<105
Compra	9	P09	P09 = 0 250<P09<300 400<P09<550 750<P09<800 950<P09<1050

Tabla 1.17(e) *Intervalos de la versión 5 de la síntesis de 38 procesos químicos*

Para este último caso, también con 6 variables semicontinuas, la utilidad neta óptima fué de \$1,596.55 E2 y al igual que en las tres tablas anteriores, las unidades presentes en las superestructura óptima fueron: y_4, y_5, y_6, y_8, y_{12}, y_{13}, y_{14}, y_{16}, y_{28}, y_{32} y y_{38}.

2. Calendarización de Trabajos con transferencia cero

Para el caso de la calendarización de trabajos con espera cero, se probaron 3 ejemplos con 6, 7 y 8 trabajos en 5 etapas. En este caso, como se mencionó en el capítulo anterior, no se utilizó ninguna información lógica ni como cláusula de procesamiento simbólico ni como corte relajado elemental. Los datos de los ejemplos utilizados se muestran en las Tablas 2.1, 2.2 y 2.3. En la Tabla 2.4 se muestran las colisiones entre las diferentes etapas necesarias para completar los trabajos de cada proceso.

Multiproceso, 6 trabajos, 5 etapas, espera cero					
Trabajo\Etapa	1	2	3	4	5
A	3		5	6	2
B		3	4	5	3
C	2	3		6	5
D	4	2	5	4	
E	3	4	6		2
F	2	6	5	7	3

Tabla 2.1 *Datos para el multiproceso de 6 trabajos, 5 etapas, transferencia cero*

Multiproceso, 7 trabajos, 5 etapas, espera cero					
Trabajo\Etapa	1	2	3	4	5
A	3		5	6	2
B		3	4	5	3
C	2	3		6	5
D	4	2	5	4	
E	3	4	6		2
F	2	6	5	7	3
G	5	2	4	5	4

Tabla 2.2 *Duración por etapas del multiproceso de 7 trabajos, 5 etapas, transferencia cero*

Multiproceso, 8 trabajos, 5 etapas, espera cero					
Trabajo\Etapa	1	2	3	4	5
A	3		5	6	2
B		3	4	5	3
C	2	3		6	5
D	4	2	5	4	
E	3	4	6		2
F	2	6	5	7	3
G	5	2	4	5	4
H	4	8	2	8	3

Tabla 2.3 *Duración por etapa del multiproceso de 8 trabajos, 5 etapas, transferencia cero*

6 trabajos, 5 etapas, espera cero					
Colisiones	1	2	3	4	5
A.B			1	1	1
A.C	1			1	1
A.D	1		1	1	
A.E	1		1		1
A.F	1		1	1	1
B.C		1		1	1
B.D		1	1	1	
B.E		1	1		1
B.F		1	1	1	1
C.D	1	1		1	
C.E	1	1			1
C.F	1	1		1	1
D.E	1	1	1		
D.F	1	1	1	1	
E.F	1	1	1		1

7 trabajos, 5 etapas, espera cero					
Colisiones	1	2	3	4	5
A.B			1	1	1
A.C	1			1	1
A.D	1		1	1	
A.E	1		1		1
A.F	1		1	1	1
A.G	1		1	1	1
B.C		1		1	1
B.D		1	1	1	
B.E		1	1		1
B.F		1	1	1	1
B.G		1	1	1	1
C.D	1	1		1	
C.E	1	1			1
C.F	1	1		1	1
C.G	1	1		1	1
D.E	1	1	1		
D.F	1	1	1	1	
D.G	1	1	1	1	
E.F	1	1	1		1
E.G	1	1	1		1
F.G	1	1	1	1	1

8 trabajos, 5 etapas, espera cero					
Colisiones	1	2	3	4	5
A.B			1	1	1
A.C	1			1	1
A.D	1		1	1	
A.E	1		1		1
A.F	1		1	1	1
A.G	1		1	1	1
A.H	1		1	1	1
B.C		1		1	1
B.D		1	1	1	
B.E		1	1		1
B.F		1	1	1	1
B.G		1	1	1	1
B.H		1	1	1	1
C.D	1	1		1	
C.E	1	1			1
C.F	1	1		1	1
C.G	1	1		1	1
C.H	1	1		1	1
D.E	1	1	1		
D.F	1	1	1	1	
D.G	1	1	1	1	
D.H	1	1	1	1	
E.F	1	1	1		1
E.G	1	1	1		1
E.H	1	1	1		1
F.G	1	1	1	1	1
F.H	1	1	1	1	1
G.H	1	1	1	1	1

Tabla 2.4 *Colisiones de los diferentes procesos en cada etapa*

En la Tabla 2.5 se muestran los resultados obtenidos para estos problemas. Estos abarcan, el tiempo mínimo de duración de cada trabajo y el tiempo de inicio de cada trabajo, en los los problemas de multiproceso de 6, 7 y 8 trabajos.

No.de Trabajos	6	7	8
Duración Total	39	44	52
Trabajo	Tiempo de Inicio		
A	4	9	9
B	0	0	0
C	22	27	35
D	24	29	37
E	17	22	30
F	7	12	12
G		1	1
H			18

Tabla 2.5 *Duración total del proceso y tiempo de inicio de cada trabajo para los problemas de multiproceso*

3. Problemas de localización de almacenes

Para los problemas de localización de almacenes tenemos dos casos. El primero que consta de 6 ejemplos de problemas con 10 almacenes cada uno y que aparecen numerados como problema 1,2,3,4,5,6 en Tabla 6.4 del capítulo 6. En la Tabla 3.1 se listan los costos fijos de cada almacen y el costo variable (debe multiplicarse por el flujo que sale del almacén *i* al punto de demanda *j*). La función objetivo es la minimización de los costos fijos + los costos variables de estos almacenes. En el último renglón se lista la cantidad demandada por cada punto de demanda del problema.

Almacén	Costo Fijo	Punto de Demanda 1	2	3	4	5	6	7
1	400	50	55	60	65	70	75	80
2	350	55	50	55	60	65	70	75
3	320	60	55	50	55	60	65	70
4	280	65	60	55	50	55	60	65
5	250	70	65	60	55	50	55	60
6	180	75	70	65	60	55	50	55
7	170	80	75	70	65	60	55	50
8	160	85	80	75	70	65	60	55
9	140	90	85	80	75	70	65	60
10	120	95	90	85	80	75	70	65
SumDemanda=49	Demanda	4	5	6	7	8	9	10

Tabla 3.1 *Costos fijos y variables de los almacénes y cantidad demandada por consumidor*

En la Tabla 3.2 se listan las capacidades por almacén, la suma de las capacidades para cada problema y el radio que se obtiene de dividir la suma de las capacidades entre la suma de las demandas de los consumidores. También se listan los costos mínimos para cada problema y los almacenes que deben estar presentes en la solución óptima de cada uno.

Problema	Capacidad/ almacen	Suma de las Capacidades	RADIO:Sum cap/ Sum Demanda	Solución Optima COSTO	Almacenes
1	30	300	6.12	3125	4,7
2	25	250	5.10	3140	4,6
3	20	200	4.08	3270	3,6,7
4	15	150	3.06	3470	3,5,6,7
5	10	100	2.04	3740	2,4,5,6,7
6	5	50	1.02	5045	1,2,3,4,5,6,7,8,9,10

Tabla 3.2 *Capacidades, radio y solución óptima de los problemas de localización de almacenes*

Los problemas referenciados como CAP41, CAP42, CAP43, CAP44, CAP51, CAP61 y CAP71 en la Tabla 6.4 del capítulo 6 fueron tomados de la base de datos en Investigación de Operaciones del Imperial College y están basados en los problemas estándares de prueba de almacenes de gran capacidad utilizados por Beasley [7] para estandarizar las pruebas de algoritmos para este tipo de problemas. Todos tienen 16 almacenes y 50 consumidores. La suma de la demanda de los 50 consumidores es de 58268 en todos los casos y se tienen los mismos valores de demanda de los 50 consumidores para todos los ejemplos, y estos son:

Consumidor	Demanda	Consumidor	Demanda	Consumidor	Demanda	Consumidor	Demanda	Consumidor	Demanda
1	146	11	5495	21	38	31	231	41	1681
2	87	12	904	22	807	32	322	42	1117
3	672	13	1466	23	551	33	685	43	275
4	1337	14	143	24	304	34	12912	44	500
5	31	15	615	25	814	35	325	45	2241
6	559	16	564	26	337	36	366	46	733
7	2370	17	226	27	4368	37	3671	47	222
8	1089	18	3016	28	577	38	2213	48	49
9	33	19	253	29	482	39	705	49	1464
10	32	20	195	30	495	40	328	50	222

Tabla 3.3 *Demanda de los consumidores en los problemas de localización de almacenes de gran capacidad*

Los costos variables, que son los mismos para todos los problemas, se reproducen en la Tabla 3.4. Nuevamente, la función objetivo viene dada por los costos fijos + los costos variables que deben de multiplicarse por el flujo que sale del almacén *i* a al consumidor *j*.

La Tabla 3.5 contiene los datos generales como la capacidad por almacén, el costo fijo por almacén la suma de las capacidades de los almacenes, el radio entre la suma de las capacidades de los almacenes y la suma de las demandas de los consumidores, y la solución óptima de los problemas de almacenes de gran capacidad.

	Almacén															
Consumidor	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1	46.2	70.9	52.4	35.8	39.6	45.5	30	26.4	44	37	35.8	28.7	50.6	34.5	70.9	41.5
2	36.8	62.7	44.2	27.6	30.2	37	21.1	26.1	35.8	29.7	26.4	20.5	58.8	25.2	62.1	32.6
3	7.31	39.3	29.2	20.7	13.6	22.3	32.5	52.6	22.5	35.2	14.6	28.7	85.5	16.6	34.2	23.1
4	24.2	22.4	15.7	22.2	15.9	15	48.1	60	19.4	51.8	17.3	36.4	101	30.3	45.3	39.6
5	55.3	69.4	50.9	34.3	40.3	44	49.2	30.8	42.5	57.7	36.6	32.8	64.7	44.5	81.1	58.8
6	11.5	42.4	29	18.6	13.4	22.1	28.3	48.8	22.3	31.8	12.6	24.9	81.4	12.5	36.4	19.6
7	34.6	12	18.2	27.8	24.8	20.5	58.5	65.5	22.4	62.2	24.1	43.2	110	40.7	55.7	50
8	30.7	24.4	5.85	15.4	12.5	8.14	47.3	53.2	10.1	52.7	11.7	30.9	97.2	29.6	55.2	42.7
9	61.2	75.2	56.7	40.2	46.2	49.9	55.1	36.7	48.3	63.6	42.3	38.7	58.8	50.4	87	64.7
10	45.6	62.4	43.8	27.2	32.8	36.9	35.4	17.1	35.5	44	29	24	61	35.8	72.3	46.9
11	25.7	37.5	19	2.3	8.39	12	36.1	40.1	10.6	44	4.6	17.8	84.1	19.3	52.5	33.8
12	19.6	35.5	17	9.32	1.36	10	36.3	45.7	10.2	41.6	2.72	21.9	87.7	18.5	46.4	31.6
13	26.1	29	10.5	10.8	7.86	3.54	42.7	48.6	7.89	48.1	7.08	26.3	92.3	25	52.9	38.1
14	13.7	35.3	28.6	24	16	24.7	38.8	58.1	24.9	41.6	17	34.3	91.9	23	32.4	29.4
15	27.9	58.6	41.3	26.5	25.7	34.4	11.9	35.3	33.8	20.4	23.8	19.4	64.9	18.2	52.8	23.4
16	45.5	63.1	44.6	28	32.7	37.7	32.8	14.4	36.2	41.3	28.9	22.4	62.2	34.1	70.7	44.3
17	31.1	46.4	27.9	11.3	17.3	21	30.3	31.5	19.5	38.9	13.6	12	77.3	22.3	57.9	36.7
18	26	30.7	12.2	9.1	7.81	7.64	41.9	46.9	3.79	48	7.57	24.6	90.9	24.9	52.9	38.1
19	37.4	49.2	30.7	14	20.1	23.7	40.6	40.4	22.3	49.2	16.3	22.3	86.1	31.9	64.2	46.4
20	44.1	72.4	53.9	37.2	40.4	46.9	28	32.7	45.5	34.5	37.5	30.1	48.8	34.3	69.5	39.5
21	41.6	53.4	34.9	18.3	24.3	28	42.9	42.6	26.5	51.4	20.6	24.5	80.7	34.2	68.4	48.6
22	28.7	60.4	44.7	32.4	29.1	37.8	18.5	41.9	38	13.8	27.7	26	69.4	21.5	49.8	19.8
23	21.9	36.1	17.6	6.9	4.09	10.6	35.7	43.3	11.4	41.5	0	19.4	85.2	18.4	48.8	32.2
24	16.1	42.3	35.6	29.4	22.4	31	39.3	59.7	31.2	38.2	23.4	35.9	92.9	23.4	25.4	26
25	29.6	48.8	30.2	13.6	16.8	23.3	24.8	30.3	21.9	33.4	13	6.47	72.8	18.2	54.8	32.6
26	12.2	36.1	24.3	16.7	8.76	17.4	34.5	50.8	17.6	38.1	9.7	26.9	87.5	16.7	39.1	26
27	64.4	93.1	74.6	58	60.6	67.4	48.4	48.3	66.3	54.9	56.8	50.9	28.4	54.7	89.9	59.9
28	19.2	38.3	19.8	9.68	4.21	12.9	33.4	44.1	13.1	38.7	2.78	20.3	86.1	15.6	46	29.4
29	17.8	46.6	29.3	15.5	13.7	22.4	23.9	42.4	22.6	29.2	12.3	18.6	77	6.14	42.7	20.6
30	25.2	51.4	44.8	38.5	31.5	40.2	48.4	68.9	40.4	47.4	32.5	45	102	32.6	16.3	35.2
31	16.1	48.1	35.6	25.2	20	28.7	28	51.5	28.9	26.4	19.2	29.6	81.1	16	37.2	14.2
32	14.5	41.5	24.5	13.5	8.89	17.6	29.9	45.4	17.8	35.2	7.45	21.5	82.9	12.1	41.3	24.7
33	19.6	51.3	37.9	25.3	22.3	30.9	24.5	48	31.1	22.9	21.5	28.3	77.6	16	40.7	10.7
34	28.9	17.8	15.8	25	20.2	17.8	52.8	62.8	20	56.4	20.3	39.7	105	35	49.9	44.6
35	30	55.6	37.1	22.2	23.4	30.2	17.8	32.9	29.6	26.3	19.6	15.1	65.7	18.3	54.9	29.3
36	32.9	49.7	31.2	14.5	20.2	24.2	29.7	27.9	22.8	38.2	16.4	11.4	73.6	23.1	58.7	37.4
37	26.6	20.1	16.2	22.7	18	15.5	50.5	60.5	19.9	54.1	18	37.4	103	32.7	47.7	42.3
38	27.5	28.7	12.4	14	9.26	6.81	44.2	51.8	11.2	49.5	9.27	28.7	94.5	26.4	51.7	39.5
39	77.3	92.5	73.9	57.3	63.1	67	67.6	48.7	66.4	76.1	60.7	55.7	58.1	68.5	104	79
40	21.8	26.3	19.6	23.9	15.9	17.1	46.5	60.3	21.5	49.7	17.3	36.4	99.5	28.7	41.4	37.5
41	22.6	42.1	23.6	9.4	9.81	16.6	29.7	37.3	16	36.9	6	13.4	79.2	13.8	48.3	28.2
42	35.6	47.4	28.9	12.2	18.3	21.9	36.8	36.6	20.5	45.4	14.5	18.5	82.3	28.1	62.4	42.6
43	58.6	75.3	56.8	40.2	45.8	49.9	45.5	28.7	48.4	54	42	36.6	48	47.5	84	56.9
44	34	65.2	46.6	32.5	32.5	39.7	18.7	41.5	39.1	19.1	29.1	26.2	64.2	25	55	25.1
45	75.3	94.1	75.5	58.9	64.5	68.6	60.1	48.8	67.2	66.6	60.7	55.7	40.2	63.6	100	71.6
46	77.9	91	72.5	55.8	61.9	65.6	71.7	53.3	64.1	80.2	58.1	55.3	62.7	67.1	104	83.2
47	70.2	83.3	64.7	48.1	54.2	57.8	64	45.7	56.4	72.5	50.4	47.6	67.3	59.3	95.9	75.5
48	51.9	80.2	61.7	45	48.2	54.7	35.8	40.5	53.3	42.3	44.4	37.9	41	42.1	77.3	47.3
49	23.3	23.4	16.7	21.4	15	14.2	47.1	59.2	18.6	50.8	16.3	35.5	101	29.4	44.3	38.6
50	32	54.1	35.5	18.9	21.8	28.6	22.1	28.9	27.2	30.6	18	11.8	67.5	20.3	56.8	33.6

Tabla 3.4 *Costos variables de los problemas de almacenes de gran capacidadad*

Problema	Capacidad/ almacen	Costo Fijo	Suma de las Capacidades	RADIO:Sum cap/ Sum Demanda	Solución Optima COSTO	Solución Optima Almacenes
CAP41	5000	7500	80000	1.37	1040417.6	1,2,3,4,5,6,7,8,9,11,12,13,14
CAP42	5000	12500	80000	1.29	1097968.3	1,2,3,4,5,6,8,9,11,12,13,14
CAP43	5000	17500	80000	1.29	1152968.3	1,2,3,4,5,6,8,9,11,12,13,14
CAP44	5000	25000	80000	1.37	1235468.3	1,2,3,4,5,6,8,9,11,12,13,14
CAP51	10000	17500	160000	2.75	1025159.3	2,3,4,6,7,8,11,13
CAP61	15000	7500	240000	3.86	932584.5	1,2,3,4,6,7,8,9,11,12,13
CAP71	58268	7500	932288	16.00	932584.51	1,2,3,4,6,7,8,9,11,12,13

Tabla 3.5 *Datos generales y solución óptima de los problemas de almacenes de gran capacidad*

En las Fig. 3.1 y 3.2 se presentan gráficas que permiten apreciar el comportamiento de la programación lineal mixta-lógica contra el de la programación lineal mixta-entera en lo que respecta al número de nodos y al tiempo de cómputo. Como puede observarse, el número de nodos de la programación mixta-lógica siempre es menor que el de la mixta-entera; sin embargo, conforme crece el radio de la suma de las capacidades de los almacenes entre la suma de las demandas de los consumidores, el tiempo de solución empieza a ser mucho mayor para la programación lineal mixta-lógica.

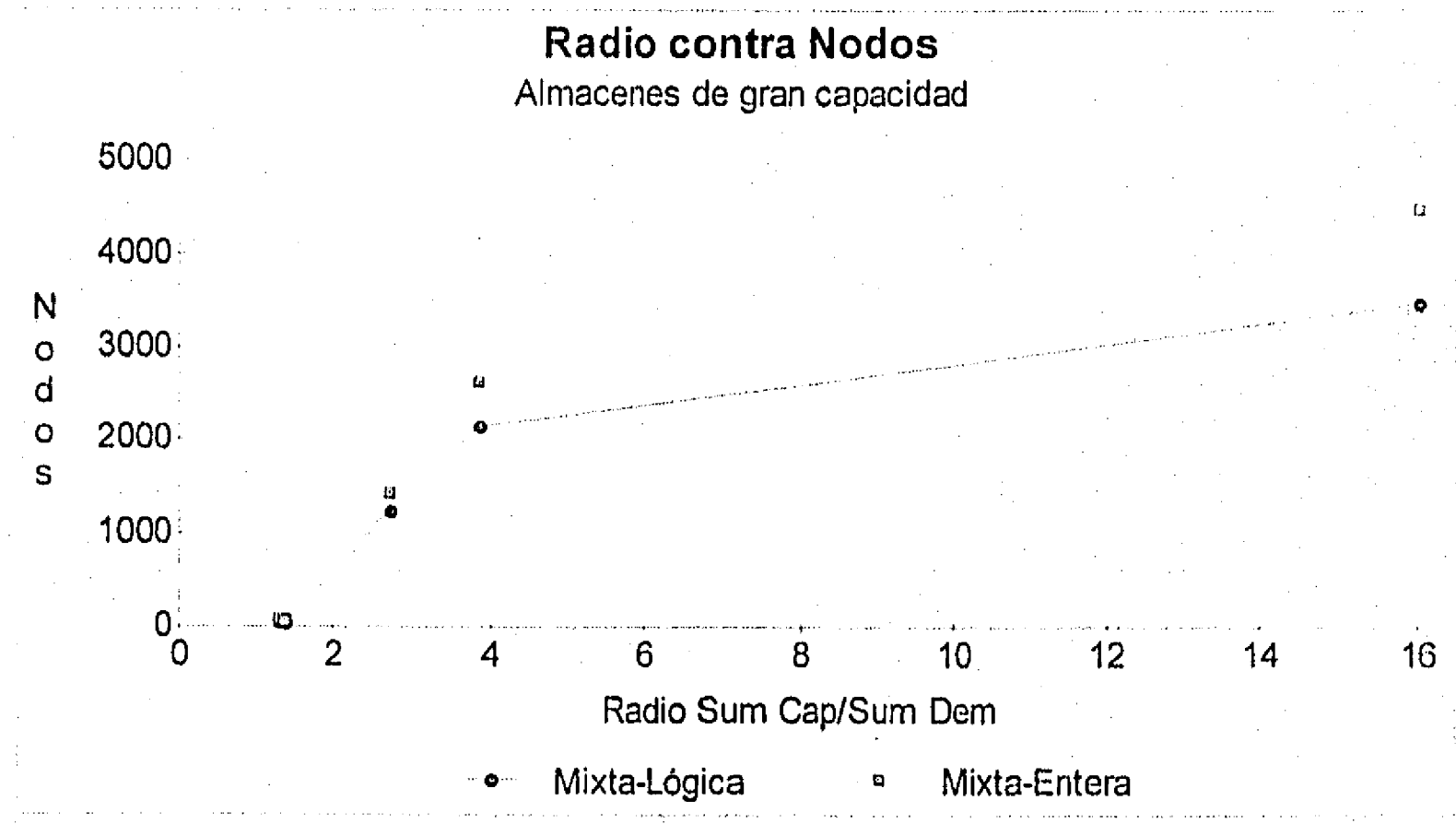


Fig. 3.1 *Número de nodos contra el radio de la suma de las capacidades entre la suma de las demandas para los problemas de almacenes de gran capacidad*

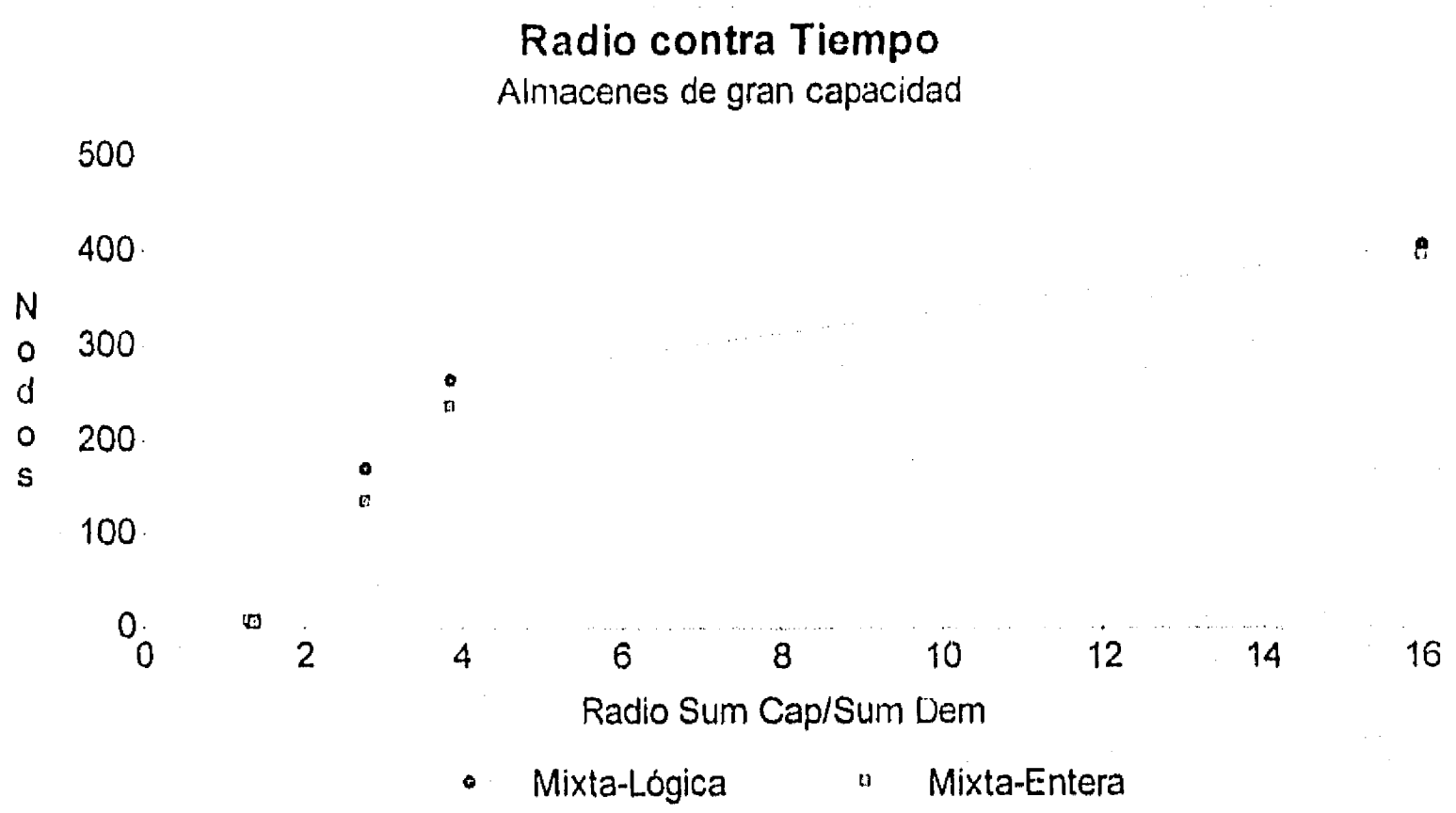


Fig. 3.2 *Tiempo de cómputo contra el radio de la suma de las capacidades entre la suma de las demandas para los problemas de almacenes de gran capacidad*

4. El problema de la fiesta progresiva

Los datos originales del problema de la fiesta progresiva se muestran en la Tabla 4.1. Para hacer las diferentes pruebas se tomaron secuencialmente los primeros 5, 6, 7, 8 y 10 yates.

No. de Yate	Capacidad	Tamaño de Tripulación	No. de Yate	Capacidad	Tamaño de Tripulación	No. de Yate	Capacidad	Tamaño de Tripulación
1	12	2	15	8	2	29	6	2
2	12	2	16	8	2	30	8	4
3	12	4	17	8	2	31	7	4
4	12	4	18	7	2	32	8	5
5	12	4	19	7	2	33	7	4
6	10	2	20	8	3	34	7	4
7	10	2	21	8	4	35	6	4
8	10	2	22	6	2	36	6	4
9	8	1	23	6	2	37	9	7
10	10	3	24	6	2	38	7	5
11	10	4	25	6	2	39	6	5
12	8	2	26	6	2	40	0	2
13	12	6	27	6	2	41	0	3
14	8	2	28	6	2	42	0	4

Tabla 4.1 *Datos originales del problema de la fiesta progresiva*

En la Tabla 4.2 se presenta el valor óptimo del número de yates, así como los yates seleccionados para cada uno de los ejemplos probados. En la Tabla 4.3 (a) - (e) se presentan los recorridos de la tripulación de cada yate (en qué yate se van a encontrar en ese momento) en cada uno de los periodos de tiempo.

Datos		Solución Optima	
No. de yates	Periodos de tiempo	No.de yates anfitriones	Yates Anfitriones
5	2	3	1,2,5
6	2	3	1,2,5
6	3	3	1,2,3
7	3	3	1,2,3
8	3	3	1,2,3
8	4	4	1,2,3,4
10	3	4	1,3,4,10
10	4	4	1,2,3,5

Tabla 4.2 *Soluciones óptimas para los problemas de la fiesta progresiva*

Yates	Periodo de tiempo 1	2	Yates
1	1	1	1
2	2	2	2
3	1	2	3
4	2	1	4
5	5	5	5

Tabla 4.3 (*a*) *Recorrido de las tripulaciones en el problema de la fiesta progresiva para 5 yates y 2 periodos de tiempo*

	Periodo tiempo			Periodo de tiempo		
Yates	1	2	Yates	1	2	3
1	1	1	1	1	1	1
2	2	2	2	2	2	2
3	1	2	3	3	3	3
4	2	1	4	1	2	3
5	5	5	5	1	3	2
6	5	1	6	2	1	3

Tabla 4.3 (b) *Recorrido de las tripulaciones en el problema de la fiesta progresiva para 6 yates y 2 y 3 periodos de tiempo*

Yates	Periodo de tiempo 1	2	3
1	1	1	1
2	2	2	2
3	3	3	3
4	1	3	2
5	3	1	2
6	2	1	3
7	3	1	2

Tabla 4.3 (c) *Recorrido de las tripulaciones en el problema de la fiesta progresiva para 7 yates y 3 periodos de tiempo*

Yates	Periodos de Tiempo 1	2	3	Yates	Periodos de Tiempo 1	2	3	4
1	1	1	1	1	1	1	1	1
2	2	2	2	2	2	2	2	2
3	3	3	3	3	3	3	3	3
4	1	2	3	4	4	4	4	4
5	1	3	2	5	1	2	3	4
6	2	1	3	6	1	3	4	2
7	2	3	1	7	1	4	2	3
8	3	1	2	8	2	3	1	4

Tabla 4.3 (d) *Recorrido de las tripulaciones en el problema de la fiesta progresiva para 8 yates y 3 y 4 periodos de tiempo*

Yates	Periodo de Tiempo 1	2	3	Yates	Periodo de Tiempo 1	2	3	4
1	1	1	1	1	1	1	1	1
2	1	2	3	2	2	2	2	2
3	3	3	3	3	3	3	3	3
4	4	4	4	4	1	2	3	5
5	1	4	3	5	5	5	5	5
6	3	1	4	6	1	3	5	2
7	3	4	1	7	1	5	2	3
8	1	2	10	8	2	1	5	3
9	4	3	1	9	2	3	1	5
10	10	10	10	10	3	1	2	5

Tabla 4.3 (e) *Recorrido de las tripulaciones en el problema de la fiesta progresiva para 10 yates y 3 y 4 periodos de tiempo*

BIBLIOGRAFIA

[1] Aiba, A., K. Sakai, Y. Sato, D.J. Hawley and R. Hasegawa, **Constraint logic programming language CAL**, Fifth Generation Computer Systems, Springer, (Toky0, 1988).

[2] Balas, E., Disjunctive programming: *Cutting planes from logical conditions*, in O.L. Mangasarian, R. R. Meyer and S. M. Robinson, eds., **Nonlinear Programming 2**, Academic Press (New York, 1975) 279-312.

[3] Balas, E., *A note on duality in disjunctive programming*, Journal of Optimization Theory and Applications **21** (1977) 523-527.

[4] Balas, E., *Disjunctive programming*, Annals Discrete Mathematics **5** (1979) 3-51.

[5] Barth, P., **Logic-Based 0-1 Constraint Programming**, Kluwer Academic Publishers (Boston, 1995).

[6] Beaumont, N., *An algorithm for disjunctive programs*, European Journal of Operational Research **48** (1990) 362-371.

[7] Beasley, J. E., *An algorithm for solving large capacitated warehouse location problems*, European Journal of Operational Research **3** (1988) 314-325.

[8] Blair, C., *Two rules for deducing valid inequalities for* 0-1 *problems*, SIAM Journal of Applied Mathematics **31** (1976) 614-617.

[9] Blair, C., R. G. Jeroslow, and J. K. Lowe, *Some results and experiments in programming techniques for propositional logic*, Computers and Operations Research **13** (1988) 633-645.

[10] Bollapragada, S., O. Ghattas and J. N. Hooker, *Optimal Design of truss structures by mixed logical and linear programming*, manuscript, Graduate School of Industrial Administration, Carnegie Mellon University, Pittsburgh, PA 15213 USA (1995).

[11] Bratko, I., **PROLOG Programming for Artificial Intelligence**, International computer Science, Addison-Wesley (1986).

[12] BULL Corporation, **CARME VI User's Guide and Reference Manual**, Artificial Intelligence Development Centre, BULL S.A. (France 1990).

[13] Chandru, V., C. R. Coullard, P.L. Hammer, M. Montañez, and X. Sun, *On renamable Horn and generalized Horn functions*, Annals of Mathematics and AI 1 (1990) 33-48.

[14] Chandru, V., and J. N. Hooker, *Extended Horn sets in propositional logic*, Journal of the ACM **38** 205-221.

[15] Chandru, V., and J. N. Hooker, *Detecting embedded Horn structure in propositional logic*, Information Processing Letters **42** (1992) 109-111.

[16] Comerauer, A., H. Kanonia, R. Pasero and P. Roussel, *Un système de communication homme-machine en français*, technical report, Université d'Aix-Marseilles II, Groupe intelligence artificielle (1973).

[17] Colmerauer, A., *An introduction to Prolog III*, Communicatins of the ACM **33** (1990) 52-68.

[18] Cooper, M.C., *An optimal k-consistency algorithm*, Artificial Intelligence 41 (1989) 89-95.

[19] Dincbas, M., P. Van Hentenryck, H. Simonis, A. Aggoun, T. Graf, F. Bertier, *The constraint programming language CHIP*, Proceedings on the International Conference on Fifth Generation Computer Systems FGCS-88, Tokyo, December 1988.

[20] Drexl, A., and C. Jordan, *A comparison of logic and mixed-integer programming solvers for batch sequencing with sequence-dependent setups*, to appear in INFORMS Journal on Computing.

[21] Freuder, E. C., *Exploiting structure in constraint satisfaction problems*, in B. Mayoh, E. Tyugu and J. Penjam, eds., **Constraint Programming**, Springer (1993) 50-74.

[22] Glover, F., *Surrogate constraint duality in mathematical programming*, Operations Research **23** 434-451.

[23] Granot, F., and P. L. Hammer, *On the use of boolean functions in* 0-1 *linear programming*, Methods of Operations Research (1971) 154-184.

[24] Haken, A., *The intractability of resolution*, Theoretical computer Science **39** (1985) 297-308.

[25] Hammer, P.L., and S. Rudeanu, **Boolean Methods in Operations Research and Related Areas**, Springer Verlag (Berlin, New York, 1968).

[26] Hooker, J. N., *Resolution vs. Cutting plane solution of inference problems: Some computational experience*, Operations Research Letters 7 (1988) 1-7.

[27] Hooker, J. N., *Generalized resolution and cutting planes*, Annals of Operations Research **12** (1988) 217-239.

[28] Hooker, J. N., *A quantitative approach to logical inference*, Decision Support Systems **4** (1988) 45-69.

[29] Hooker, J. N., *Input proofs and rank one cutting planes*, ORSA Journal on Computing **1** (1989) 137-145.

[30] Hooker, J. N., *Generalized resolution for* 0-1 *linear inequalities*, Annals of Mathematics and AI **6** (1992) 271-286.

[31] Hooker, J. N., *Logical inference and polyhedral projection*, Proceedings, Computer Science Logic Workshop (CSL'91), Lecture Notes in Computer Science **626** (1992) 184-200.

[32] Hooker, J.N., *Logic-based methods for optimization*, in A. Borning, ed., **Principles and Practice of Constraint Programming**, Lecture Notes in Computer Science **874** (1994) 336-349.

[33] Hooker, J.N., *Logic-based Benders decomposition*, manuscript, Graduate School of Industrial Administration, Carnegie Mellon University, Pittsburgh, PA 15213 USA (1996).

[34] Hooker, J. N., *Testing heuristics: We have it all wrong*, Journal of Heuristics **1** (1995) 33-42.

[35] Hooker, J.N., and N.R. Natraj, *Solving* 0-1 *optimization problems with k-tree relaxation*, in preparation.

[36] Hooker, J.N., and G. Rago, *Partial instantiation methods for logic programming*, Graduate School of Industrial Administration, Carnegie Mellon University, Pittsburgh, PA 15213 USA (1995).

[37] Hooker, J.N., H. Yan, I. Grossmann, and R. Raman, *Logic cuts for processing networks with fixed charges*, Computers and Operations Research **21** (1994) 265-279.

[38] Howard, R. A., and J. E. Matheson, *Influence diagrams*, in R. A. Howard and J. E. Matheson, eds., **The Principles and Applications of Decision Analysys**, v.2, Strategic Decision Group, Menlo Park, CA (1981).

[39] Jaffar, J., and J.-L. Lassez, *Constraint logic programming*, Proceedings of the 14th ACM Symposium on Principles of Programming Languages, München, ACM (1987) 111-119.

[40] Jaffar, J., andJ.-L. Lassez, *From unification to constraints*, Logic programming **87**, Proceedings of the 6th Conference, Springer (1987) 1-18.

[41] Jeroslow, R. E., *Representability in mixed integer programming, I: Characterization resuts*, Discrete Applied Mathematics **17** (1987) 223-243.

[42] Jeroslow, R. E., and J. K. Lowe, *Modeling with integer variables*, Mathematical Programming Studies **22** (1984) 167-184.

[43] Kowalski, R. A., *Predicate logic as programming language*, Proceedings of the IFIP Congress, North-Holland (Amsterdam, 1974) 569-574.

[44] McAloon, K., and C. Tretkoff, *2LP:Linear programming and logic programming*, in P. van Hentenryck and V. Saraswat, eds., **Principles and Practice of Constraint Programming**, MIT Press (1995) 99-114.

[45] McAloon, K., and C. Tretkoff, *Optimization and Computational Logic*, to be published by Wiley.

[46] Puger, J.-F., *A C++ implementation of CLP*, Technical report 94-01, ILOG S.A., Gentilly, France (1994).

[47] Quine, W. V., *The problem of simplifying truth functions*, American Mathematical Monthly **59** (1952) 521-531.

[48] Quine, W. V., *A way to simplify truth functions,* American Mathematical Monthly **62** (1955) 627-631.

[49] Raman, R., and I. E. Grossmann, *Relation between MILP modeling and logical inference for chemical process synthesis*. Computer and Chemical Engineering **15** (1991) 73-84.

[50] Raman, R., and I. E. Grossmann, *Symbolic integration of logic in MILP branch and bound methods for the synthesis of process networks*, Annals of Operations Research **42** (1993) 169-191.

[51] Raman, R., and I. E. Grossmann, *Symbolic integration of logic in mixed-integer linear programming techniques for process synthesis*, Computers and Chemical Engineering **17** (1993) 909-927.

[52] Raman, R., and I. E. Grossmann, *Modeling and computational techniques for logic based integer programming*. Computer and Chemical Engineering **18** (1994) 563-578.

[53] Remy, C., *Programming by constraints*, Micro Systemes No. **104** (1990) 147-150.

[54] Robinson, J. A., *A machine-oriented logic based on the resolution principle*, Journal of the ACM **12** (1965) 23-41.

[55] Sahinidis, N. V. and I. E. Grossmann, *Multiperiod investment model for processing networks with dedicated and flexible plants*, Computers and Chemical Engineering **30** (1987) 1165-1171.

[56] Schlipf, J. S., F. S. Annexstein, J. V. Franco and R. P. Swaminathan, *On finding solutions for extended Horn formulas*, Information Processing Letters **54** (1995) 133-137.

[57] Sciamma, D., J. Gay, A. Guillard, *CHARME: A constraint oriented approach to scheduling and resource allocation*, Artificial Intelligence in the Pacific Rim. Proceedings of the Pacific Rim International Conference on Artificial Intelligence, Nagoya, Japan (1990) 71-76.

[58] Sheng, C.-L., **Threshold Logic**, Academic Press (New York, 1969).

[59] Simonis, H., and M. Dincbas, *Propositional calculus problems in CHIP*, in F. Benhamou and A. Colmerauer, eds., **Costraint Logic Programming: Selected Research**, MIT Press (Cambridge, MA, 1993) 269-285.

[60] Smith, B.M., S.C. Brailsford, P.M. Hubbard, H.P. Williams, *The progressive party problem: Integer linear programming and constraint programming compared*, in U.Montanari and F. Rossi, eds., Proceedings of Principles and Practice of Constraint Programming, Cassis, France, Springer (1995) 36-52.

[61] Sterling, L., and E. Shapiro, **The Art of Prolog: Advanced Programming Techniques**, MIT Press (Cambridge, MA, 1986).

[62] Swaminathan, R.P., and D. K. Wagner, *The arborescence-realization problem*, Discrete Applied Mathematics **59** (1995) 267-283.

[63] Tsang, E., **Foundations of Constraint Satisfaction** (London, Academic Press) 1993.

[64] Turkay, M., and I.E. Grossmann, *Logic-based MINLP algorithms for the optimal synthesis of process networks*, Computer and Chemical Engineering **20** (1996) 959-978.

[65] Van Hentenryck, P., **Constraint Satisfaction in Logic Programming**, MIT Press (Cambridge, MA, 1988).

[66] Williams, H.P., *Fourier-Motzkin elimination extension to integer programming problems*, Journal of Combinatorial Theory **21** (1976) 118-123.

[67] Williams, H.P., **Logical problems and integer programming**, Bulletin of the Institute of Mathematics and its Implications **13** (1977) 18-20.

[68] Williams, H.P., *Linear and integer programming applied to the propositional calculus*, International Journal of Systems Research and Information Science **2** (1987) 81-100.

[69] Williams, H.P., *An alternative explanation of disjunctive formulations*, European Journal of Operational Research **72** (1994) 200-203.

[70] Williams, H.P., *Logic applied to integer programming and integer programming applied to logic*, European Journal of Operational Research **81** (1995) 605-616.

[71] Wilson, J.M., *Compact normal forms in propositional logic and integer programming formulations*, Computers and Operations Research **90** (1990) 309-314.

[72] Wilson, J. M., *Generating cuts in integer programming with families of specially ordered sets*, European Journal of Operational Research **46** (1990) 101-108.

[73] Wilson, J.M., *A note on logic cuts and valid inequalities for certain standard* (0-1) *integer programs*, manuscript, Loughborough University Business School, Loughborough, Leicestershire LZE11 3TU, U.K. (1995).