

UNIVERSIDAD DE SEVILLA.  
FACULTAD DE MATEMATICAS.

ESQUEMAS DE ENUMERACION IMPLICITA ORDENADA:  
PROGRAMACION ENTERA.

Visado en Sevilla

El Director

A handwritten signature in black ink, consisting of stylized initials 'JPA' enclosed within a circular flourish, with a long vertical line extending downwards from the bottom of the circle.

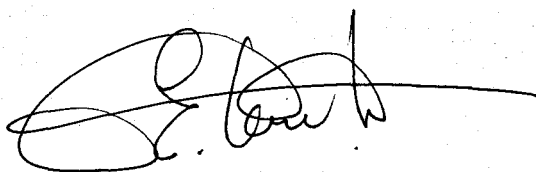
Fdo. Justo Puerto Albandoz.

Memoria que presenta

Eduardo Conde Sánchez

para optar al grado de

Doctor en Matemáticas.

A handwritten signature in black ink, featuring a large, stylized initial 'E' followed by a series of loops and a long horizontal line extending to the right.

Fdo. Eduardo Conde Sánchez.

291  
77

*Dedicada a mi familia y a*

*M<sup>a</sup>. Luisa.*

*Eduardo Conde Sánchez.*

## INDICE.

|   |    |
|---|----|
| PROLOGO.  | 1  |
| CAPITULO 1: DIRECCIONES ENTERAS DE BUSQUEDA.  |    |
| 1.1 INTRODUCCION.   | 8  |
| 1.2 OPERACION DE SELECCION.   | 10 |
| Dirección Entera de Búsqueda. Relaciones.   |    |
| Dirección Entera de Búsqueda Ordenada. Aplicaciones.                                    |    |
| 1.3 APLICACION AL PROBLEMA ENTERO GENERAL (PEG).  | 21 |
| Optimalidad de la regla de mínima etiqueta.   |    |
| Ejemplos de Direcciones Enteras de Búsqueda Ordenada.                                   |    |
| Algoritmo Aditivo de Balas.   |    |
| Algoritmo de Glover para el problema de Knapsack modular.                               |    |
| 1.4 ESQUEMA GENERAL DE ENUMERACION IMPLICITA.   | 32 |
| CAPITULO 2: ESTRUCTURACION DE $Z_+^n$ SEGUN UN GREDOIDE LOCAL<br>PARCIALMENTE ORDENADO. |    |
| 2.1 INTRODUCCION.   | 35 |
| 2.2 ORDEN C-LEXICO EN $Z_+^2$ .   | 37 |
| 2.3 EXTENSION DEL ORDEN A $Z_+^n$ .   | 40 |
| 2.4 EXTENSION DEL ORDEN C-LEXICO A CONOS.   | 50 |

## CAPITULO 3: PROGRAMACION LINEAL ENTERA: RELAJACION MODULAR.

|  |    |
|--|----|
| 3.1 INTRODUCCION.                                      | 53 |
| 3.2 CONSTRUCCION DE LA RELAJACION MODULAR.             | 56 |
| 3.3 RESOLUCION DEL PROBLEMA MODULAR ENTERO.            | 60 |
| Operación de Selección.                                |    |
| Supresión de ramas de búsqueda.                        |    |
| Esquema general del Algoritmo C-LEXMOD.                |    |
| 3.4 EXACTITUD DEL ALGORITMO C-LEXMOD.                  | 68 |
| 3.5 COMPLEJIDAD DEL ALGORITMO C-LEXMOD.                | 73 |
| 3.6 CONDICION SUFICIENTE DE OPTIMALIDAD.               | 75 |
| 3.7 ESTUDIO COMPUTACIONAL DEL ALGORITMO C-LEXMOD.      | 82 |
| 3.8 UN ALGORITMO DE RAMIFICACION Y ACOTACION PARA PEP. | 85 |

## CAPITULO 4: PROBLEMAS ESPECIALES.

|  |     |
|--|-----|
| 4.1 INTRODUCCION.  | 93  |
| 4.2 PROBLEMAS CON UN UNICO OBJETIVO.                               | 94  |
| Funciones explícitamente cuasiconvexas:                            |     |
| Objetivos cuadráticos.   |     |
| 4.3 PROBLEMA MULTIOBJETIVO ENTERO.                                 | 105 |
| Algoritmo de generación del conjunto de puntos eficientes de PELM. |     |

## CAPITULO 5: ALGORITMOS NO EXACTOS.

|  |     |
|--|-----|
| 5.1 INTRODUCCION.                                      | 111 |
| 5.2 ALGUNOS RESULTADOS DE OPTIMIZACION COMBINATORIA.   | 113 |
| Algoritmo greedy para problemas CIMP y CIMMP.          |     |
| 5.3 FORMULACION DE PEG COMO UN CIMMP.                  | 117 |
| 5.4 APLICACION A PROBLEMAS PEG CON OBJETIVOS LINEALES. | 121 |
| Construcción del intervalo error.                      |     |
| Construcción del test de salida del algoritmo.         |     |
| Esquema del algoritmo greedy para objetivos lineales.  |     |
| Problema de Knapsack restringido.                      |     |
| Problema lineal entero con restricciones lineales.     |     |
| REFERENCIAS.   | 135 |

## PROLOGO.

El concepto de problema de optimización es intuitivamente fácil de entender, consiste en determinar una alternativa óptima frente al criterio seguido de entre un conjunto de posibilidades. Sin embargo la descripción formal de este problema es algo más compleja. Schrijver (1986) define un problema como un subconjunto  $\Pi \subseteq \Sigma^* \times \Sigma^*$ , donde  $\Sigma$  es un conjunto finito llamado código y  $\Sigma^*$  es el conjunto de secuencias ordenadas de símbolos (elementos del código). El problema de optimización consiste en determinar un elemento  $x^* \in \Sigma^*$  fijado  $z \in \Sigma^*$ , o bien decidir que no existe este elemento. A la cadena  $z$  se le denomina parámetros del problema o parámetros de entrada y a  $x^*$  solución.

Por Programación Entera se entiende el conjunto de técnicas destinadas a la resolución de problemas de optimización en los que la cadena solución  $x^*$  representa un vector con componentes enteras. Los problemas de optimización más estudiados dentro de la Programación Entera son los lineales, en ellos los parámetros de entrada se determinan a partir de la cadena  $(A, b, c)$  siendo  $A$  una matriz de dimensiones  $m \times n$ ,  $b$  un vector columna  $m$  dimensional y  $c$  un vector fila de dimensión  $n$ . Para representarlos usaremos la siguiente formulación

ESQUEMAS DE ENUMERACION IMPLICITA ORDENADA:  
PROGRAMACION ENTERA.

minimizar  $cx$

sujeto a:  $Ax \leq b$

$$x \in \mathbb{Z}_+^n$$

Existe una gran variedad de problemas reales que admiten la formulación anterior, por indicar algunos ejemplos, podemos citar los problemas de secuenciación de tareas, problemas de planificación como el de localización de servicios, problemas de diseño como el de determinación de recorridos en grafos, problemas estadísticos en el análisis de datos y fiabilidad, o incluso problemas en biología molecular o física de alta energía.

Este amplio campo de aplicación ha hecho que la Programación Entera haya sido profundamente estudiada en las últimas décadas. Es prácticamente imposible citar a todos aquellos autores que han contribuido al conocimiento actual del problema, sin embargo algunos avances significativos fueron llevados a cabo por Ford y Fulkerson en problemas sobre grafos, Gomory aportó su conocido método de planos de corte, Edmonds afrontó con éxito el problema del emparejamiento (matching) óptimo en grafos... Mas recientemente, podríamos citar a Cook y posteriormente Karp por sus novedosos resultados sobre teoría de complejidad o a Zions y Wallenius por sus trabajos en optimización entera multiobjetivo.

En contra de lo que podría suponerse a la vista del gran esfuerzo investigador desarrollado, en la actualidad no se conoce ningún algoritmo de complejidad polinomial en el tamaño de los parámetros de entrada para resolver los problemas enteros más



simples, los lineales.

¿Por qué son difíciles de resolver los problemas de Programación Entera?

En primer lugar se debe indicar que la dificultad de resolución depende fuertemente de las dimensiones del vector de variables de decisión (Lenstra (1983) demostró que fijada el número de variables, el problema lineal entero es resoluble en tiempo polinomial). Cuando la dimensión crece, el número de puntos factibles crece, en general, exponencialmente. Sin embargo, el número de soluciones factibles no ha impedido el desarrollo de técnicas satisfactorias de resolución en otras áreas de la Optimización. Para problemas con variables continuas existen algoritmos efectivos desde la época de Newton y Leibniz. La dificultad estriba en que las limitaciones de la Matemática actual impiden la determinación de caracterizaciones eficaces del conjunto de soluciones óptimas de un problema entero general, con lo cual es inevitable enumerar todas las soluciones o al menos una fracción de las mismas. Esto hace que en la actualidad exista la creencia generalizada de la imposibilidad de construir un algoritmo polinomial para este tipo de problemas de optimización.

A pesar de la intratabilidad de los problemas considerados, o gracias a ella, en la actualidad existe una amplia gama de algoritmos. Muchos de ellos desarrollan esquemas específicos que aprovechan la estructura especial del conjunto factible de los problemas que resuelven. La mayoría de estos se enmarcan en

ESQUEMAS DE ENUMERACION IMPLICITA ORDENADA:  
PROGRAMACION ENTERA.

Optimización Combinatoria. No obstante, en líneas generales se puede identificar dos grandes grupos de esquemas algorítmicos para el problema entero, aquellos basados en técnicas de Descripción Poliédrica y los que se basan en técnicas de Enumeración Parcial.

Dentro de cada grupo, es difícil distinguir las diferentes técnicas de resolución, ya que muchas veces interactúan y se nutren unas de otras. Podríamos identificar como técnicas especialmente relevantes en Descripción Poliédrica las de caracterización de poliedros con vértices enteros y las de planos de corte. En el segundo grupo son significativas las técnicas de ramificación y acotación, enumeración implícita y las basadas en la resolución de problemas relajados.

El objetivo de esta memoria es concerniente a este segundo grupo de técnicas de resolución.

En primer lugar, se propone un esquema general de enumeración implícita que denominamos ordenada y que engloba como casos particulares algoritmos clásicos como el de Balas (1965) o el de Glover (1969).

En segundo lugar, usando este marco general se construyen algoritmos para diferentes problemas justificándose su exactitud. En estos algoritmos se conjuga la enumeración implícita con técnicas de relajación y con ramificación y acotación.

Por último se trata el tema de la construcción de algoritmos aproximados. Para ello se utilizan técnicas propias de la Optimización Combinatoria, poniéndose una vez más de manifiesto la

interconexión entre la teoría de la Optimización Combinatoria y la de Programación Entera.

Antes de comenzar con el desglose de los capítulos indicaremos algunas cuestiones de notación usada en la memoria. En primer lugar se hace uso de superíndices para identificar vectores y subíndices para sus componentes.  $e^i$  representa el vector con valor 1 en su componente  $i$  y el resto de componentes nulas. El símbolo  $\leq$  entre vectores indica la desigualdad componente a componente, con  $\leq_1$  designamos el orden lexicográfico.  $[a]$  es el mayor entero, menor que el valor real  $a$ .  $|A|$  representa el cardinal del conjunto  $A$ . Por último, en el producto matricial no se utiliza la identificación de vectores traspuestos a menos que sea imprescindible para la comprensión de la expresión, en los casos en los que pueda existir ambigüedad se detallarán los vectores fila o columnas usados.

Descritos estos preliminares desglosamos brevemente el contenido de cada capítulo.

En el primer capítulo se desarrolla y formaliza el esquema de enumeración implícita ordenada. Este esquema tiene como piezas claves las operaciones de supresión y de selección de elementos. La primera de ellas aprovecha las características del problema considerado por lo que su descripción debe ser hecha localmente para cada tipo de problema considerado. La selección de elementos sin embargo se puede describir globalmente. En este capítulo se formaliza esta operación utilizando para ello las denominadas

ESQUEMAS DE ENUMERACION IMPLICITA ORDENADA:  
PROGRAMACION ENTERA.

estructuras gredoides. Como resultado se obtiene la definición de una dirección entera de búsqueda ordenada que es la herramienta usada en los esquemas de enumeración y que posee la propiedad de no duplicidad de elementos generados.

En el segundo capítulo se construye una dirección de búsqueda particular denominada c-léxico. A través de ella se pueden definir búsquedas sin duplicidad en la restricción de conos poliédricos al conjunto de números enteros, lo que será de gran ayuda a la hora de tratar problemas enteros con objetivos no lineales.

En el tercer capítulo, se desarrolla un esquema de enumeración implícita, basado en la dirección entera asociada al orden c-léxico. El algoritmo, denominado C-LEXMOD, es empleado para la resolución de una relajación del problema lineal entero original. Esta relajación fue introducida por Gomory y se denomina problema lineal entero con restricciones modulares. Posteriormente se obtiene una condición suficiente bajo la cual ambos problemas son equivalentes. Se compara esta condición con la formulada por Gomory encontrándose relaciones bajo las cuales esta última es mejorada. En la parte final del capítulo se detalla un estudio computacional en el que se compara el algoritmo C-LEXMOD con otros algoritmos del mismo tipo observándose un comportamiento favorable del primero. A continuación se propone un esquema de ramificación y acotación que utiliza estos problemas relajados en la resolución del problema entero lineal.

En el cuarto capítulo, se muestran algunas aplicaciones del

ESQUEMAS DE ENUMERACION IMPLICITA ORDENADA:  
PROGRAMACION ENTERA.

esquema general en la construcción de algoritmos para problemas especiales. En concreto se considera el problema de minimización general con objetivos cuadráticos y el problema lineal con objetivos múltiples.

Por último, en el capítulo cinco se propone un algoritmo aproximado para la Programación Entera. Se obtiene un intervalo que determina el error cometido por la solución propuesta por el algoritmo. Finalmente se particulariza este intervalo para problemas especiales como el de mochila con cotas en las variables o el problema lineal entero general.

## CAPITULO 1

### DIRECCIONES ENTERAS DE BUSQUEDA

#### 1.1 INTRODUCCION.

En las últimas décadas han aparecido en la literatura gran cantidad de técnicas de resolución de problemas de optimización discreta, la mayor parte de ellas se engloba en dos grupos principales: Técnicas de Enumeración Parcial y Técnicas de Descripción Poliédrica. En este capítulo se estudia una metodología que pertenece al primer grupo y que denominamos Enumeración Implícita Ordenada.

Haciendo una simplificación de un método de enumeración implícita, se puede decir que en primera instancia se determina una familia de problemas de fácil resolución asociada al problema de optimización original. La resolución de cada uno de los problemas de la familia concluye con una solución del problema original. No obstante esto no es necesario en la mayoría de los casos. Existen subfamilias de problemas cuya resolución ofrece el mismo resultado. El objetivo de la Enumeración Implícita consiste en determinar una de estas subfamilias utilizando como herramientas las propiedades especiales que verifica el problema

ESQUEMAS DE ENUMERACION IMPLICITA ORDENADA:  
PROGRAMACION ENTERA.

bajo consideración.

En el proceso se usan dos operaciones elementales, la de selección del elemento o elementos de la familia que serán estudiados en cada iteración del algoritmo y la operación de supresión de aquellos elementos cuyo estudio no es necesario para resolver el problema inicial.

El esquema anteriormente descrito será tanto más exitoso cuanto más reducida sea la subfamilia de problemas simples que deban ser estudiados. Existen numerosas aplicaciones en la literatura que utilizan con éxito esta idea, como ejemplo podemos citar las técnicas de Programación Dinámica o los Métodos de Ramificación y Acotación (MRA) aplicados a la Programación Matemática, o técnicas aplicadas en Algorítmica para la construcción de esquemas de búsqueda en profundidad y esquemas de búsquedas en paralelo. En esta memoria se aplicarán estas técnicas en la resolución de Problemas Enteros Generales (PEG).

Como hemos indicado antes, dos son las operaciones básicas que deben determinarse, la operación de selección y la de supresión de elementos, esta última depende fuertemente de las características del problema estudiado por lo que por el momento nos centraremos en la operación de selección. En la siguiente sección se define una serie de nociones que nos ayudarán a representar esta operación en el contexto de los problemas de optimización en variables enteras.

## 1.2 OPERACION DE SELECCION.

En la introducción del capítulo hemos hablado de la definición de problemas de fácil resolución como la base de cualquier algoritmo de Enumeración Implícita. En la mayoría de los algoritmos recogidos en esta memoria, cada problema simple consiste en la evaluación de la función objetivo en un único punto. En estos casos la familia de problemas coincidirá con un conjunto  $E$  de puntos que contiene a las soluciones factibles. La regla de selección es la que determina los puntos que serán evaluados en cada iteración.

### 1.2.1 Definición.

Una regla de selección es un operador  $\phi$  que en la iteración  $k$  de un algoritmo elige un elemento del conjunto  $E$  en función del conjunto  $X(k)$  de elementos estudiados en iteraciones previas :

$$\phi(X(k)) = x(k+1) \in E \setminus X(k)$$

Para identificar el operador  $\phi$  usaremos una familia  $\mathfrak{F} \subseteq 2^E$  y exigiremos que  $X(k) \cup \{x(k+1)\} \in \mathfrak{F}$ . No obstante, en la mayoría de los casos esto no determina unívocamente a  $x(k+1)$  por lo que especificaremos de antemano una propiedad  $\mathfrak{P}$  en la elección del elemento.



A partir de la condición anterior, dada una iteración  $k$ ,  $X(k)$  puede representarse explícitamente como

$$X(k) = \{x(1), x(2), \dots, x(k)\}$$

con la propiedad:

$$\{x(1), x(2), \dots, x(i)\} \in \mathfrak{F} \quad \forall i=1 \dots k.$$

Este concepto está muy relacionado con el que Korte y Lovasz introducen en 1983 bajo el nombre de *gredoide*.

### 1.2.2 Definición. (Korte y Lovász (1983))

Dado un conjunto  $E$  finito se define un *gredoide* como el par  $(E, \mathfrak{F})$  donde  $\mathfrak{F} \subseteq 2^E$  es una familia de conjuntos que contiene al vacío y tal que dados  $X, Y \in \mathfrak{F}$  con  $|X| < |Y|$  entonces  $\exists y \in Y \setminus X$  tal que  $X \cup \{y\} \in \mathfrak{F}$ .

El concepto surgió como una generalización de la noción de *matroide* (Whitney (1935)) y su aplicación ha girado fundamentalmente en torno al campo de la Optimización Combinatoria, concretamente se han usado como estructura base de algoritmos greedy para problemas de optimización en grafos (Parker y Rardin (1988)).

Por otro lado muchas de las caracterizaciones del concepto de *matroide* se hacen a través de la noción de *gredoide* (Dietrich (1989)). Por último, a través de lo que se denomina función de rango de un *gredoide* se define las funciones localmente submodulares que son de gran importancia en optimización puesto

que existen algoritmos polinomiales para la minimización de funciones submodulares (Lovász (1983), Lawler (1985)).

A continuación veremos que el concepto de gredoide también puede ser aplicado a la Programación Entera lográndose con ello dos ventajas significativas, en primer lugar se engloba en un esquema general gran parte de los algoritmos de enumeración implícita existentes actualmente, por otro lado se obtienen propiedades globales útiles en el desarrollo de nuevos algoritmos.

### 1.2.3 Dirección Entera de Búsqueda. Relaciones.

En este apartado definimos el concepto de dirección entera de búsqueda en términos de la cual se construirá la regla de selección de cualquier algoritmo de enumeración implícita. Nuestro propósito es la resolución de problemas de Programación Entera, por eso definimos  $E$  como un conjunto finito verificando  $0 \in E \subseteq \mathbb{Z}_+^n$  y de forma que contenga a toda solución factible del problema de optimización. Puesto que en los problemas tratados siempre se supondrá que el conjunto factible es acotado la elección de  $E$  no supone ninguna restricción.

#### 1.2.3.1 Definición.

Llamaremos intervalo entero con vértice en el origen de coordenadas y definido por los valores enteros  $k_1 \dots k_n$  al conjunto:

$$I(k_1, k_2, \dots, k_n) = \{0, 1, \dots, k_1\} \times \{0, 1, \dots, k_2\} \times \dots \times \{0, 1, \dots, k_n\}$$

### 1.2.3.2 Definición.

A cualquier familia de conjuntos  $\mathfrak{F} \subseteq 2^E$  con  $E$  un conjunto finito,  $0 \in E \subseteq I(k_1, k_2, \dots, k_n)$  y tal que  $(E, \mathfrak{F})$  sea un gredoide le denominaremos dirección entera de búsqueda.

La relación que existe entre el concepto de gredoide y el de regla de selección viene dado a partir de la siguiente propiedad:

### 1.2.3.3 Propiedad. (Korte y Lovász (1983))

Dado  $X \in \mathfrak{F}$ ,  $(E, \mathfrak{F})$  un gredoide, entonces existe una ordenación de los elementos de  $X$  (llamada ordenación factible)  $\{x(1) \dots x(k)\}$  con  $k = |X|$  tal que  $\{x(1) \dots x(i)\} \in \mathfrak{F} \forall i = 1 \dots k$ .

Por tanto en un problema de programación entera, fijada la dirección entera de búsqueda se tendrá restringida la selección de elementos a través de  $\mathfrak{F}$  y la propiedad  $\mathfrak{P}$  usada por la regla. Simbólicamente, podríamos expresar la selección como la aplicación de la propiedad  $\mathfrak{P}$  sobre el rango definido por el gredoide:

$$\phi(X(k)) = \mathfrak{P}(\{x \in E \setminus X(k) / X(k) \cup \{x\} \in \mathfrak{F}\}).$$

Para garantizar propiedades útiles en el desarrollo de esquemas de enumeración implícita es necesario trabajar con estructuras algo más restrictivas, en la próxima sección introducimos el concepto propuesto por Goecke y Schrader en 1990 denominado gredoide local parcialmente ordenado.

#### 1.2.4 Dirección Entera de Búsqueda Ordenada. Aplicaciones.

Comenzamos con la definición de un tipo especial de gredoide.

##### 1.2.4.1 Definición. (Goecke y Schrader (1990))

Se denomina gredoide local parcialmente ordenado a aquél par  $(E, \mathfrak{F})$  con  $E$  un conjunto finito y  $\mathfrak{F} \subseteq 2^E$  verificando:

- (a)  $\forall A, B \subseteq C, A, B \text{ y } C \in \mathfrak{F}$  entonces  $A \cup B \in \mathfrak{F}$ .
- (b)  $\forall A, B \subseteq C, A, B \text{ y } C \in \mathfrak{F}$  entonces  $A \cap B \in \mathfrak{F}$ .

En todos los algoritmos que se desarrollan en esta memoria se usan reglas de selección basadas en gredoides locales parcialmente ordenados. Esto se debe a la existencia de unos conjuntos en  $\mathfrak{F}$  denominados **camino**s o **cadena**s que son únicos para cada uno de los puntos de  $E$ . Por tanto una regla de selección que genere caminos selecciona puntos evitando duplicaciones como puede deducirse del siguiente resultado.

##### 1.2.4.2 Propiedad. (Goecke y Schrader (1990))

Dado el par  $(E, \mathfrak{F})$  son equivalentes (a) y (b):

- (a)  $(E, \mathfrak{F})$  es un gredoide local parcialmente ordenado.
- (b)  $(E, \mathfrak{F})$  verifica:

$$(b-1) \forall A, B, C \in \mathfrak{F} / A, B \subseteq C \Rightarrow A \cup B \in \mathfrak{F}.$$

$$(b-2) \forall A \in \mathfrak{F}, a \in A, \exists (\text{único}) Z_a \subseteq A / a \in Z_a \text{ de forma que } a \text{ es el único elemento de } Z_a \text{ que verifica que } Z_a \setminus \{a\} \in \mathfrak{F}.$$

A  $Z_a$  lo denominamos camino o cadena con elemento terminal a.

Con esta propiedad, dado  $A \in \mathfrak{J}$ ,  $A = \bigcup_{a \in A} Z_a = \bigcup_{a \in A_T} Z_a$ ,

donde  $A_T$  representa el conjunto de elementos  $a \in A / \neg \exists b \in A$  cumpliendo  $Z_a \subset Z_b$  (estrictamente).  $A_T$  se denomina el conjunto de elementos terminales de A.

Los caminos son de gran importancia en el esquema de enumeración implícita por esto definimos un tipo especial de gredoide local para el cual, este tipo de conjuntos se puede caracterizar en términos de una relación binaria  $\mathfrak{K}$  definida sobre E que es un orden parcial, esto es reflexiva, antisimétrica y transitiva (Zimmermann (1981)).

#### 1.2.4.3 Definición. (debo)

Se denomina dirección entera de búsqueda ordenada (debo) al par  $(E, \mathfrak{J})$ , donde E es un conjunto finito  $0 \in E \subseteq \mathbb{Z}_+^n$ ,  $\mathfrak{K}$  es una relación de orden parcial en E y  $\mathfrak{J}$  es una familia de subconjuntos de E tal que:

- (a)  $\emptyset \in \mathfrak{J}$ .
- (b)  $X \in \mathfrak{J} \Leftrightarrow 0 \in X \wedge \forall x \in X, \forall t \in E / 0 \mathfrak{K} t \wedge t \mathfrak{K} x \Rightarrow t \in X$ .
- (c)  $\forall x \in E, x \neq 0 \exists (\text{único}) x^- \in E \setminus \{x\} / x^- \mathfrak{K} x \wedge \{t \in E / x^- \mathfrak{K} t \wedge t \mathfrak{K} x\} = \{x, x^-\}$ .

Dado cualquier punto  $x \in E \setminus \{0\}$  a  $x^- \in E$  que verifica el apartado (c) de la definición anterior le denominamos antecesor

directo de  $x$  y como veremos (Teorema 1.2.4.7) este tipo de puntos juega un papel importante en la determinación del rango de aplicación de la regla de selección.

#### 1.2.4.4. Teorema.

Un par  $(E, \mathfrak{J})$  verificando las propiedades de la definición 1.2.4.3 es un gredoide local parcialmente ordenado.

#### Demostración.

Sean  $A, B, C \in \mathfrak{J}$ ,  $A, B \subseteq C$ .

En primer lugar, por ser  $A$  y  $B$  elementos de la familia  $\mathfrak{J}$  queda claro que  $0 \in A \cup B$  y  $0 \in A \cap B$ .

Sea  $x \in A \cup B$ ,  $x \neq 0 \Rightarrow x \in A$  ó  $x \in B$ , en cualquier caso  $\forall t \in E / 0 \mathfrak{R} t \wedge t \mathfrak{R} x \Rightarrow t \in A$  ó  $t \in B \Rightarrow t \in A \cup B$ .

Sea  $x \in A \cap B$ ,  $x \neq 0 \Rightarrow x \in A$  y  $x \in B$ , con lo cual  $\forall t \in E / 0 \mathfrak{R} t \wedge t \mathfrak{R} x \Rightarrow t \in A$  y  $t \in B \Rightarrow t \in A \cap B$ . ■

Hasta el momento no se ha hecho uso de la propiedad de unicidad de antecesor directo. Esta propiedad caracteriza los caminos en una estructura de este tipo como podremos ver en el próximo resultado:

1.2.4.5 Teorema.

En una dirección entera de búsqueda ordenada el único camino con punto terminal  $a \in E$  está totalmente ordenado según el orden parcial  $\mathfrak{K}$  verificándose que el 0 es el primer elemento de la ordenación y que, dos vectores son consecutivos en la ordenación si el primero es antecesor directo del segundo.

**Demostración.**

La demostración se deduce a partir del resultado que indica que en un gredoide local parcialmente ordenado existe un único camino con punto terminal  $a \in E$ . Determinemos este camino mediante un procedimiento constructivo:

Sea  $x_m = a$ , por la propiedad de unicidad de antecesor directo  $\exists$  (único)  $x_m^-$ , llamémoslo  $x_{m-1}$ . Sucesivamente  $x_i = x_{i+1}^-$ . Puesto que  $E$  es finito y  $\mathfrak{K}$  es un orden parcial obtendremos  $x_1 \in E / x_1^- = 0 = x_0$ . Consideremos el conjunto  $Z_a \equiv \{x_0, x_1, \dots, x_n\}$  anteriormente construido, probaremos que es un camino con punto terminal  $a$ .

En primer lugar  $Z_a \in \mathfrak{J}$  puesto que  $\{x_0, x_1, \dots, x_i\} \in \mathfrak{J} \forall i, i=0, 1, \dots, n$ , por inducción.

En segundo lugar  $a$  es el único elemento de  $Z_a$  tal que  $Z_a \setminus \{a\} \in \mathfrak{J}$  como consecuencia de la inducción anterior y debido al hecho de que al considerar  $b \in Z_a \setminus \{a\} \Rightarrow b = x_i^- \ i < n$  se verifica:

$$\{t \in E / 0 \mathfrak{K} t \wedge t \mathfrak{K} x_i\} \not\subseteq Z_a \setminus \{b\}$$

y por tanto  $Z_a \setminus \{b\} \notin \mathfrak{J}$  por definición. ■

ESQUEMAS DE ENUMERACION IMPLICITA ORDENADA:  
PROGRAMACION ENTERA.

Veamos como afecta la propiedad que acabamos de probar al conjunto de elementos terminales de  $A \in \mathfrak{J}$ .

1.2.4.6 Teorema.

Dados los conjuntos  $A, B \in \mathfrak{J}$ ,  $A \subset B$ ,  $b \in B \setminus A \Rightarrow \neg \exists a_1, a_2 \in A_T$   
/  $Z_{a_1} \subseteq Z_b \wedge Z_{a_2} \subseteq Z_b$ .

Demostración.

Sea  $a_1, a_2 \in A_T$ ,  $a_1 \neq a_2$  /  $Z_{a_1} \subseteq Z_b \wedge Z_{a_2} \subseteq Z_b$ . Puesto que  $Z_b$  está completamente ordenado, según se probó anteriormente:

$$a_1 \mathfrak{K} a_2 \text{ ó } a_2 \mathfrak{K} a_1.$$

Supongamos que  $a_1 \mathfrak{K} a_2$ . Sabemos que  $Z_{a_1}$  está completamente ordenado y que  $\forall t \in Z_{a_1} \setminus \{0, a_1\}$ ,  $0 \mathfrak{K} t \wedge t \mathfrak{K} a_1$ , por tanto  $0 \mathfrak{K} a_1$  con lo cual  $a_1 \in Z_{a_2}$ . Por lo tanto, por definición del conjunto  $Z_{a_2} \forall t \in E / 0 \mathfrak{K} t \wedge t \mathfrak{K} a_1 \Rightarrow t \in Z_{a_2}$ , es decir  $Z_{a_1} \subseteq Z_{a_2}$ . Por ser  $a_1$  un elemento de  $A_T$  se tiene:

$$Z_{a_1} \equiv Z_{a_2}.$$

Por último, puesto que  $Z_{a_2} \setminus \{a_1\} \in \mathfrak{J} \Rightarrow a_1 = a_2$ , por la definición de camino, con lo cual se llega a una contradicción de la hipótesis de partida. ■

La propiedad asegura que ninguno de los elementos de  $A_T$  es redundante, es decir, en términos de un esquema de enumeración



ESQUEMAS DE ENUMERACION IMPLICITA ORDENADA:  
PROGRAMACION ENTERA.

implícita diríamos que los puntos de  $A_T$  representan ramas de búsqueda a partir de las cuales se generan nuevos elementos de  $E$  evitando duplicaciones de los mismos.

Esta propiedad nos será de utilidad a la hora de eliminar puntos que nunca serán examinados en un algoritmo de enumeración implícita, debido a que no se puede acceder a ellos desde el 0 mediante un camino completamente ordenado.

Anteriormente hemos definido la regla de selección en función de la propiedad  $\mathcal{P}$  y de la familia  $\mathcal{F}$ , así

$$\phi(X(k)) = \mathcal{P}(\{x \in E \setminus X(k) / X(k) \cup \{x\} \in \mathcal{F}\}).$$

El par  $(E, \mathcal{F})$  se construye a partir de una relación binaria  $\mathcal{R}$  de orden parcial sobre  $E$ , sin embargo para determinar  $\phi$  no es necesario conocer explícitamente la familia de conjuntos  $\mathcal{F}$ . En el siguiente resultado se probará que es suficiente conocer el antecesor directo de cada elemento de  $E$  para construir los conjuntos sobre los cuales se aplica la propiedad  $\mathcal{P}$  en la definición de  $\phi$ .

1.2.4.7. Teorema.

Una regla de selección  $\phi$  definida en base a una de  $(E, \mathcal{F})$  queda determinada por la definición de antecesor directo de los puntos de  $E \setminus \{0\}$  y por la propiedad  $\mathcal{P}$  de la siguiente forma:

$$\phi(X(k)) = \mathcal{P}(\{x \in E \setminus X(k) / x^- \in X(k)\}) \forall k \in \mathbb{N}.$$

**Demostración.**

Sea  $\mathfrak{K}$  la relación binaria de orden parcial definida en  $E$  y sea  $(E, \mathfrak{J})$  el par dado por la definición 1.2.4.3. Supongamos conocido  $x^-$  para cada  $x \in E \setminus \{0\}$ , en esta situación se probará por doble inclusión para  $X \in \mathfrak{J}$  la igualdad de conjuntos siguiente:

$$\{x \in E \setminus X / X \cup \{x\} \in \mathfrak{J}\} \equiv \{x \in E \setminus X / x^- \in X\}$$

En primer lugar si  $X \cup \{x\} \in \mathfrak{J}$ , según el teorema 1.2.4.4  $(E, \mathfrak{J})$  es un gredoide local parcialmente ordenado con lo cual la propiedad 1.2.4.2 garantiza la existencia del camino  $Z_x \subseteq X \cup \{x\}$ . Por último, la caracterización del teorema 1.2.4.5 asegura que  $x^- \in Z_x$  y por tanto  $x^- \in X$ , de donde

$$\{x \in E \setminus X / X \cup \{x\} \in \mathfrak{J}\} \subseteq \{x \in E \setminus X / x^- \in X\}$$

Por otro lado, sea  $x \in E / x^- \in X$ , como  $X \in \mathfrak{J}$  se tiene garantizada la existencia de  $Z_{x^-} \subseteq X$  y por el teorema 1.2.4.5,  $Z_{x^-} \cup \{x\} \in \mathfrak{J}$ .

Usando la propiedad (a) de la definición 1.2.4.1 con  $C \equiv E$  se tiene

$$\{Z_{x^-} \cup \{x\}\} \cup X \in \mathfrak{J}$$

por tanto

$$X \cup \{x\} \in \mathfrak{J}$$

con lo que se obtiene la inclusión en sentido contrario que concluye la demostración. ■

En el siguiente apartado se estudia la aplicación que tienen los conceptos anteriores en la optimización de un Problema Entero.

### 1.3 APLICACION AL PROBLEMA ENTERO GENERAL (PEG).

Consideremos el problema entero general:

$$(PEG) \min f(x)$$

$$sa: x \in \mathcal{D}, \mathcal{D} \subset \mathbb{Z}_+^n \text{ acotado.}$$

Dado  $x \in \mathcal{D} \Rightarrow x = (x_1 \dots x_n) \in \mathbb{Z}_+^n$ , además por estar acotado  $0 \leq x_i \leq k_i \forall i=1 \dots n$ . Sea el intervalo entero  $I(k_1 \dots k_n)$ .

Tomemos  $E \equiv I(k_1, k_2 \dots k_n)$  y una relación binaria  $\mathfrak{K}$  sobre  $E$  verificando las propiedades que aparecen en la definición de una *debo*.

Goecke y Schrader (1990) definen el concepto de *restricción de un gredoide* según  $A \subseteq E$  como el par  $(E \setminus A, \mathfrak{K} \setminus A)$  donde

$$\mathfrak{K} \setminus A \equiv \{ X \in \mathfrak{K} / X \subseteq E \setminus A \}.$$

La restricción de un gredoide local parcialmente ordenado sigue siéndolo.

En la aplicación de los conceptos referidos a direcciones enteras a problemas de optimización, generalmente consideraremos como gredoide la siguiente restricción:

$$\text{Sea } \mathcal{D}' \equiv \{ x \in \mathcal{D} / \exists y \in \mathcal{D}, y \mathfrak{K} x \} \text{ tomemos } (I(k_1, k_2 \dots k_n) \setminus \mathcal{D}', \mathfrak{K} \setminus \mathcal{D}').$$

La definición de este gredoide local implica que una vez generado un vector  $x(k) \in \mathcal{D}$ , ninguno de sus vectores sucesores directos será considerado por el algoritmo. Es decir, estamos eliminando como objeto de nuestro estudio todos aquellos puntos  $a \in E$  tal que  $Z_a \cap \mathcal{D}' \neq \emptyset$ . Puesto que en el caso que nos ocupa tenemos caracte-

rizados los caminos  $Z_a$  a través de secuencias de vectores totalmente ordenados según la relación de orden, está claro que estos puntos no son de interés para el problema de minimización, siempre que la función objetivo  $f$  sea creciente respecto a  $\mathfrak{K}$ .

El siguiente resultado prueba la exactitud de esquema de enumeración implícita cuando la propiedad  $\mathfrak{D}$  selecciona el vector con menor valor objetivo. En esta situación diremos que estamos usando la regla de selección de mínima etiqueta. Existen otras reglas de selección algunas de las cuales serán estudiadas en capítulos posteriores. Una de ellas que se denominará "regla de mínimo incremento", selecciona aquel vector  $x$  tal que  $\Delta(x)=f(x)-f(x^-)$  sea mínimo y da lugar a los algoritmos *greedy* que se estudian en el capítulo 5 en el que se justifica que, en general, estos algoritmos sólo determinan la solución óptima en el caso en el que la familia de conjuntos completamente ordenados según  $\mathfrak{K}$  sea un matroide.

### 1.3.1 Teorema.

Si  $x^*$  es el primer vector perteneciente a  $\mathfrak{D}$  generado por un algoritmo de enumeración implícita basado en una *debo*, dada por la relación de orden  $\mathfrak{K}$  y la regla de mínima etiqueta entonces,  $x^*$  es óptima si  $f$  es creciente respecto de  $\mathfrak{K}$ .

ESQUEMAS DE ENUMERACION IMPLICITA ORDENADA:  
PROGRAMACION ENTERA.

Demostración.

Basta probar que  $\forall k \neg \exists x \in E \setminus X(k) / f(x) < f(y) \wedge y \in X(k)$ .  
Por inducción,  $X(1) \equiv \{x(1)\} \equiv \{0\}$  lo verifica.

Supongámoslo cierto para  $k$ , ie.  $\neg \exists x \in E \setminus X(k) / f(x) < f(y) \wedge y \in X(k)$   
si en la iteración  $k+1$  no se verificase la hipótesis entonces  
necesariamente  $y = x(k+1) \wedge \exists x \in E \setminus X(k+1) / f(x) < f(x(k+1))$ .

Puesto que  $E \in \mathfrak{J}$ ,  $\exists Z_x / Z_x \setminus \{x\} \in \mathfrak{J}$ , según el teorema 1.2.4.5 se  
tiene  $0 \in Z_x$  y sus elementos están totalmente ordenados. Sea  $\omega$  el  
primer elemento de la secuencia ordenada que no pertenece a  
 $X(k+1)$ . Según se ha tomado  $\omega$  se tiene  $\omega \in X(k+1)$ .

Por otro lado, por ser  $f$  una función creciente respecto al  
orden  $\mathfrak{K}$ , se verifica  $f(\omega) \leq f(x) < f(x(k+1))$  por lo que

$$x(k+1) \neq \operatorname{argmin}_x \{f(x) / x \in X(k+1)\}$$

en contra de lo supuesto inicialmente. ■

En general, no es posible determinar condiciones más débiles  
para la función objetivo  $f()$  de forma que el resultado de  
optimalidad del teorema pueda garantizarse ya que, a la vista de  
la demostración, es equivalente el hecho de ser  $f()$  creciente en  $\mathfrak{K}$   
y el de ser  $X(k)$  un conjunto de nivel para cada iteración  $k$ , y  
esto último es lo que garantiza la optimalidad del primer punto  
factible generado.

En la próxima sección se estudian dos algoritmos clásicos  
como son el Aditivo de Balas (1965) y el empleado por Glover  
(1969) en la resolución de problemas Lineales Enteros con una  
Restricción Modular (LERM), demostrándose que utilizan reglas de

selección basadas en direcciones enteras de búsqueda, por tanto estos algoritmos son casos particulares del esquema general que se desarrolla en este capítulo.

### 1.3.2 Ejemplos de Direcciones Enteras de Búsqueda Ordenada.

#### 1.3.2.1 Algoritmo Aditivo de Balas.

El algoritmo Aditivo de Balas ha sido aplicado de muy diversas formas a problemas de optimización en variables enteras. Aunque inicialmente surgió para resolver el problema Entero 0-1 con objetivo lineal, posteriormente se aplicó a problemas con variables enteras no binarias e incluso a problemas con objetivos múltiples (Klein y Hannan (1982)). En esta sección se usará una aplicación de este algoritmo que servirá como ejemplo de los conceptos definidos con anterioridad.

Consideremos un problema de Programación Entera Lineal:

$$\begin{aligned} \min c'x, \quad & 0 \leq c_1 \leq c_2 \leq \dots \leq c_n \\ \text{sa: } Ax = b, \quad & x \in \mathbb{Z}_+^n \end{aligned}$$

Supongamos que el politopo  $\{Ax = b, x \geq 0\}$  es acotado, en este caso  $x_i \in \{0, 1, \dots, k_i\}$  en los puntos factibles. Tomemos  $E = I(k_1, \dots, k_n)$  y  $\bar{y}$  la dirección entera de búsqueda ordenada generada por el orden parcial que a continuación describimos:

En primer lugar dado  $x$  un punto factible, podemos hacer el siguiente cambio de variable:

$$x_i = x_{i0} + 2x_{i1} + \dots + 2^{p_i} x_{ip_i}, \quad x_{ij} \in \{0, 1\} \quad \forall i=1, \dots, n, \quad j=1, \dots, p_i.$$

Tomemos estas nuevas  $p = \prod_{i=1}^n (p_i + 1)$  variables y las ordenamos de

ESQUEMAS DE ENUMERACION IMPLICITA ORDENADA:  
PROGRAMACION ENTERA.

menor a mayor costo asociado, donde el costo asociado a la variable  $x_{ir}$  es  $2^r c_i$ . Sean estas nuevas variables ya ordenadas las que denotamos por  $y_1 \dots y_p$ .

Por construcción, dado  $x \in E = I(k_1 \dots k_n)$  existe una única representación en términos de las variables  $y$ 's. Definimos la relación binaria

$$x^1 \leq_B x^2 \Leftrightarrow y_i^1 = y_i^2 \quad i=1 \dots r, \quad y_{r+1}^1 = 0 \wedge y_{r+1}^2 = 1 \text{ ó } 0, \quad y_{r+j}^1 = 0 \quad j=1 \dots (p-r)$$

$$r \in \{1 \dots p\}.$$

La relación binaria  $\leq_B$  es un orden parcial a partir del cual se construyen los conjuntos de la familia  $\tilde{\mathcal{X}}$  del gredoide. Además es una dirección entera de búsqueda ordenada puesto que por definición todo  $x \in I(k_1 \dots k_n) \setminus \{0\}$  tiene un único antecesor directo  $\bar{x}$ .

Dicho esto, sólo queda especificar la operación de selección del esquema de enumeración implícita. Siguiendo con la notación introducida en secciones anteriores podemos describir la selección de nuevos elementos como se indica a continuación:

$$\phi(X(k)) = \mathcal{P}(\{x \in E \setminus X(k) / X(k) \cup \{x\} \in \tilde{\mathcal{X}}\}) = \operatorname{argmin}_x \{cx / x \notin X(k) \wedge \bar{x} \in X(k)\}$$

Esta regla de selección es de mínima etiqueta, por otro lado por ser todos los coeficientes de costo positivos, la función objetivo es creciente con respecto al orden  $\leq_B$  con lo cual el teorema 1.3.1

ESQUEMAS DE ENUMERACION IMPLICITA ORDENADA:  
PROGRAMACION ENTERA.

garantiza la optimalidad del primer punto factible generado. El algoritmo de Balas se completaría con el test de acotación de nodos que aquí no se describe por formar parte de la operación de supresión del esquema.

Veamos una representación gráfica de una parte del gredoi de Balas aplicado al siguiente ejemplo:

$$\begin{aligned} \min & x_1 + 3x_2 \\ \text{sa: } & x_1 + x_2 \geq 8, \quad x_1, x_2 \in \mathbb{Z}_+^1 \\ & x_1 \leq 4, \quad x_2 \leq 5. \end{aligned}$$

Hacemos el cambio de variable para  $k_1 = 4$  y  $k_2 = 5$ :

$$\begin{aligned} x_1 &= x_{10} + 2x_{11} + 4x_{12} \\ x_2 &= x_{20} + 2x_{21} + 4x_{22} + 8x_{23} \end{aligned}$$

Ordenamos las nuevas variables según el costo asociado:

$$(x_{10}, x_{11}, x_{20}, x_{12}, x_{21}, x_{22}, x_{23}) = (y_1, y_2, y_3, y_4, y_5, y_6, y_7)$$

Veamos una posible secuencia de sucesores directos:

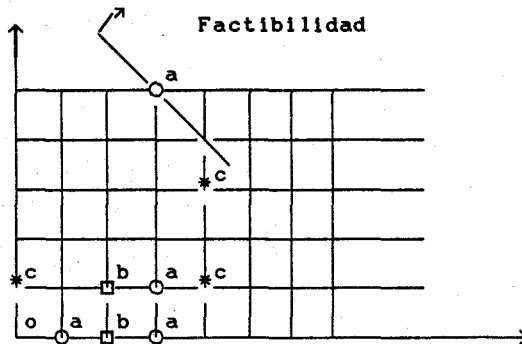
| $y_1$ | $y_2$ | $y_3$ | $y_4$ | $y_5$ | $y_6$ | $y_7$ | — Cambio de variable — | $x_1$ | $x_2$ |
|-------|-------|-------|-------|-------|-------|-------|------------------------|-------|-------|
| 0     | 0     | 0     | 0     | 0     | 0     | 0     |                        | 0     | 0     |
| 1     | 0     | 0     | 0     | 0     | 0     | 0     |                        | 1     | 0     |
| 1     | 1     | 0     | 0     | 0     | 0     | 0     |                        | 3     | 0     |
| 1     | 1     | 1     | 0     | 0     | 0     | 0     |                        | 3     | 1     |
| 1     | 1     | 1     | 0     | 0     | 1     | 0     |                        | 3     | 5     |

Gráficamente, podemos representar la evolución de este y otros caminos hasta conseguir factibilidad de su punto terminal respecto al problema anteriormente propuesto o bien llegar a un punto



ESQUEMAS DE ENUMERACION IMPLICITA ORDENADA:  
PROGRAMACION ENTERA.

terminal sin sucesores. En la figura se representan los elementos de la misma cadena con una misma letra (salvo el origen que está en todas las cadenas):



El problema que presenta el algoritmo Aditivo de Balas es que el gredoide que construye hace uso de la linealidad de la función objetivo con lo cual no se puede extender a la resolución de problemas no lineales. La idea de definir un orden parcial en el conjunto  $\{0,1\}^n$  fue utilizada posteriormente por Bowman y Starr (1979). Ellos generalizan el algoritmo de Balas mediante la construcción de un algoritmo de enumeración implícita que selecciona los puntos estudiados en cada iteración de acuerdo con un orden parcial, no obstante la definición del orden también hace uso de la linealidad del objetivo.

### 1.3.2.2 Algoritmo de Glover para el problema de Knapsack modular.

Glover desarrolló un algoritmo de enumeración implícita en 1969 para resolver un problema de Knapsack entero sobre un grupo aditivo, concretamente su formulación es la siguiente

$$\begin{aligned} \min \quad & cx \\ \text{sa:} \quad & ax=b \pmod{d} \\ & x \in \mathbb{Z}_+^n \end{aligned} \quad (\text{PKM})$$

donde  $c, a \in \mathbb{Z}_+^n$  y  $b, d$  son enteros positivos y el conjunto factible está acotado,  $\{x \in \mathbb{Z}_+^n / ax=b\} \subseteq I(k_1, \dots, k_n)$ .

Este tipo de problemas, que será estudiado con mayor profundidad en un capítulo posterior, fue introducido por Gomory(1965) y se usa como relajación del problema entero obtenido al considerar la restricción del conjunto factible en aritmética común.

El algoritmo base desarrollado por Glover genera un vector entero en cada iteración  $i = 1, 2, 3, \dots$  obteniéndose una secuencia  $x(1), x(2), \dots, x(i), \dots$  en la que cada  $x(i)$  se puede expresar en función de un vector  $x(p)$  con  $p < i$  al cual se incrementa una de sus componentes en una unidad, es decir

$$x(i) = x(p) + e_r$$

Glover demuestra la exactitud del algoritmo haciendo uso de las cuatro propiedades siguientes:

- (1) Si  $p \neq q$  entonces  $x(p) \neq x(q)$ .
- (2) Si  $p < q$  entonces  $cx(p) \leq cx(q)$ .
- (3)  $x(i)$  es óptima si  $i$  es el primer índice para el cual se verifica  $ax(i) = b \pmod{d}$ .
- (4) La secuencia de vectores generados por el algoritmo es finita

ESQUEMAS DE ENUMERACION IMPLICITA ORDENADA:  
PROGRAMACION ENTERA.

y concluye con la solución óptima del problema, siempre que éste sea factible.

Para garantizar las cuatro propiedades anteriores Glover diseñó el siguiente procedimiento de generación del vector  $x(k)$  en base al conjunto  $X(k) \equiv \{x(1) \dots x(k)\}$ :

$$x(k+1) = \operatorname{argmin}_x \{ cx / x = x(t_j) + e^j \quad j \in \{1 \dots n\} \}$$

donde  $t_j \in \{1, 2 \dots k-1\} \forall j = 1 \dots n$ . Inicialmente  $x(1)=0$  y  $t_j=1 \forall j$  después, si en la iteración  $k$ , el mínimo anterior se alcanza en  $j$ , el valor  $t_j$  se actualiza como se indica a continuación:

$$t_j := \min\{i / i > t_j \wedge x(i) = x(1) + e^r, \quad 1 < i \text{ con } r > j\}$$

Estos valores  $t_j, j=1 \dots n$  son los que Glover denomina índices de transición.

El algoritmo propuesto por este autor contempla la posibilidad de suprimir vectores dominados, no obstante, para los propósitos de esta sección no es necesario describir esta operación. A continuación se prueba que la operación de selección de este algoritmo está basada en una dirección entera de búsqueda ordenada, para ello se determinará una relación de orden parcial en  $\mathbb{Z}_+^n$  inducida por el proceso de actualización de los índices de transición. Además esta relación binaria verifica la unicidad de antecesor directo.

1.3.2.2.1 Definición.

Sean  $x, y \in \mathbb{Z}_+^n$ , se dice que  $x \leq_c y$ , si y sólo si  $x = y$  ó bien existen vectores enteros  $z^1 \dots z^m$  verificando  $z^1 = x$ ,  $z^m = y$ ,  $z^{i-1} = z^i - e_j$  para  $j = \min\{p / z_p^i \neq 0\}$   $i=2 \dots m$ .

La relación anterior es un orden parcial verificando que todo vector de  $\mathbb{Z}_+^n \setminus \{0\}$  tiene un único antecesor directo  $x^-$  dado por la siguiente expresión:

$$x^- = x - e^j \text{ con } j = \min\{p / x_p \neq 0\}.$$

1.3.2.2.2 Teorema.

La operación de selección del algoritmo de Glover está basada en la debo determinada por el orden parcial  $\leq_c$  sobre  $I(k_1 \dots k_n)$  y por la regla de mínima etiqueta.

Demostración.

Tomemos  $X(k) = \{x(1) \dots x(k)\}$  cuyos vectores fueron generados por el algoritmo en las  $k$  primeras iteraciones. Sea  $E \subseteq \mathbb{Z}_+^n$  finito,  $0 \in E$  y tal que contiene a todo punto factible. Sea  $(E, \mathfrak{J}_c)$  el glpo basado en  $\leq_c$ . Según el teorema 1.2.4.7 la operación de selección  $\phi$  basada en  $(E, \mathfrak{J}_c)$  puede expresarse:

$$\phi(X(k)) = \mathfrak{P}\{x \in E \setminus X(k) / x^- \in X(k)\}.$$

Veamos que coincide con la operación de selección dada por Glover.

Tomemos un vector  $x(p) = x(1) + e^r \in X(k)$ ,  $1 < p$ . Teniendo en cuenta que el índice de transición  $t_j$  recorre todos los índices de las iteraciones, existirá una iteración  $i(j)$  para cada  $j < r$  en la

ESQUEMAS DE ENUMERACION IMPLICITA ORDENADA:  
PROGRAMACION ENTERA.

cual  $t_j = p$ , por tanto los vectores que se pueden generar a lo largo del algoritmo a partir de  $x(p)$  son aquellos  $x(q_j) = x(p) + e^j$ ,  $j < r$ . A partir de este resultado y teniendo en cuenta que  $x(1) = 0$ , por inducción podemos garantizar que  $x(p) = x(1) + e^r$  tiene sus primeras  $r-1$  componentes nulas, con lo cual  $x(q_j) = x(p) \forall j < r$ . Esto prueba que en la iteración  $k$  del algoritmo:

$$\{ x(t_j) + e^j / j \in \{1 \dots n\} \} \subseteq \{ x \in E / x \in \{x(1) \dots x(k)\} \}$$

Probaremos que el vector con menor valor objetivo de este último conjunto está contenido en el primero, con lo cual se tendrá probada la equivalencia de las operaciones de selección.

Supongamos que  $x \in E$ ,  $x \in X(k)$  es el vector donde se alcanza el mínimo valor objetivo. Supongamos que  $x = x(i) + e^j$ ,  $i \leq k$ , por construcción de  $\leq_c$ ,  $x = x(i) + e^j$  para algún valor de  $j$ . Puesto que  $x \notin X(k)$  si no perteneciera al primer conjunto debería verificarse  $i > t_j$ , lo cual implica (propiedad (2))

$$cx(i) \geq cx(t_j)$$

de donde

$$c(x(i) + e^j) \geq c(x(t_j) + e^j)$$

por tanto el mínimo valor objetivo se alcanza en el primer conjunto. ■

### 1.3.2.2.3 Corolario.

Si el problema PKM es factible, el algoritmo de Glover determina su solución óptima en un número finito de iteraciones.

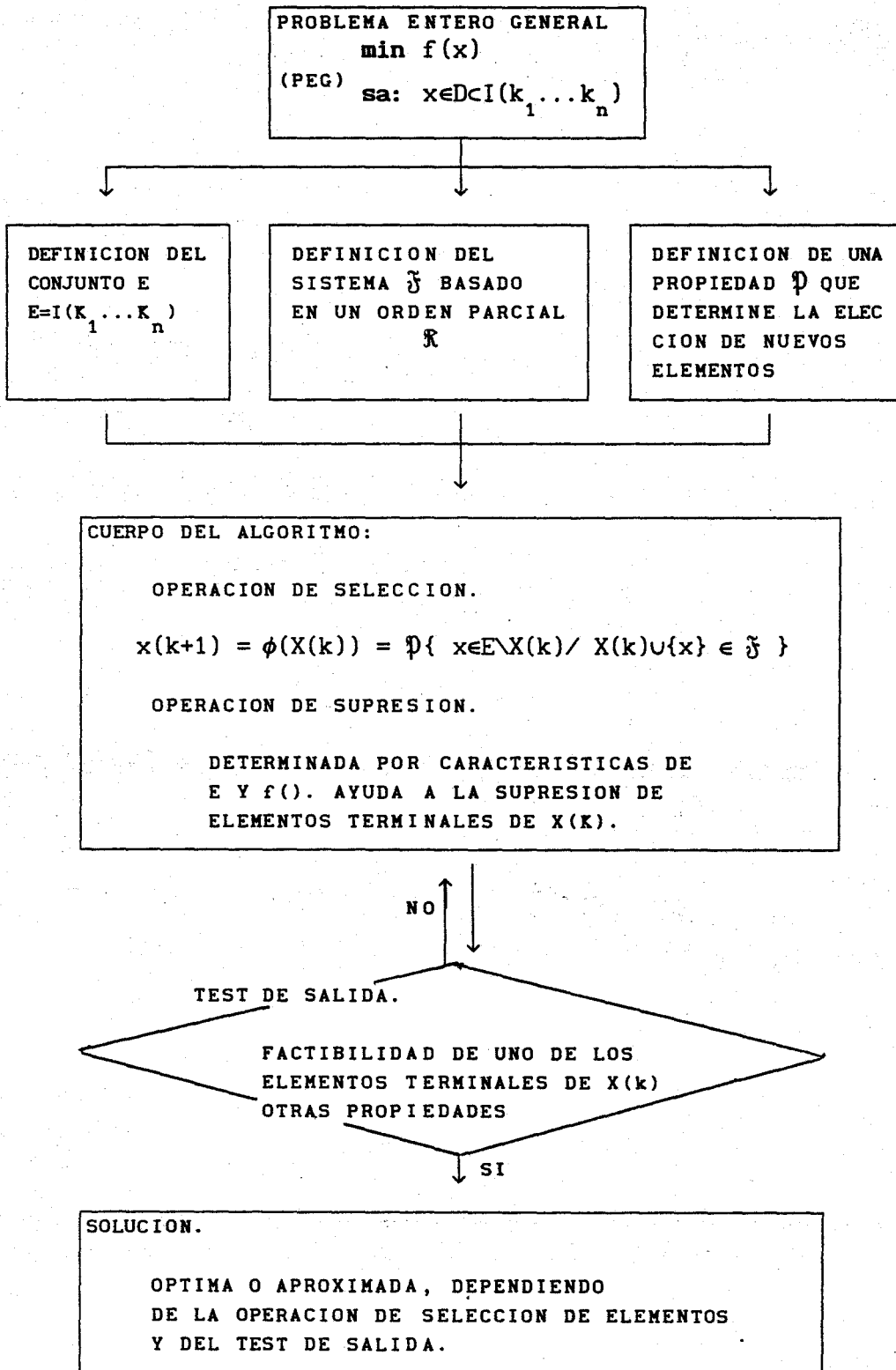
#### Demostración.

Si PKM es factible, puesto que E es finito, en alguna iteración se genera un vector factible. Por otro lado la función objetivo de PKM es creciente en  $\leq_c$  y el teorema anterior garantiza que la regla de selección es de mínima etiqueta lo cual implica la optimalidad del primer punto factible generado por el algoritmo por aplicación del teorema 1.3.1. ■

### 1.4 ESQUEMA GENERAL DE ENUMERACION IMPLICITA.

En el esquema aparecen los elementos que hemos descrito con anterioridad y que determinarán el procedimiento de construcción de algoritmos que aplicaremos en la memoria:

ESQUEMAS DE ENUMERACION IMPLICITA ORDENADA:  
PROGRAMACION ENTERA.



ESQUEMAS DE ENUMERACION IMPLICITA ORDENADA:  
PROGRAMACION ENTERA.

La propiedad  $\mathcal{P}$  que aparece en el esquema y que fija la forma en que se seleccionan los elementos, determina en la mayoría de los casos la exactitud o no del algoritmo. Por otro lado es la que influirá más decisivamente en la complejidad del mismo.

En el siguiente capítulo se construye un orden parcial que se utilizará posteriormente en el desarrollo de un algoritmo de enumeración implícita para los problemas con restricciones modulares y en un esquema aproximado para la programación entera.



## CAPITULO 2

### ESTRUCTURACION DE $Z_+$ SEGUN UN GREDOIDE LOCAL PARCIALMENTE ORDENADO.

#### 2.1 INTRODUCCION.

Las estructuras ordenadas han sido empleadas en problemas de optimización por un gran número de autores en los últimos años (ver Zimmermann (1981)). En la mayoría de los problemas la relación de orden se define sobre el conjunto de soluciones factibles y responde a consideraciones sobre el modelado del problema real. Estos problemas fueron resueltos inicialmente de forma individual con lo cual, como Zimmermann (1981) apunta en su libro, muchos resultados conocidos en Optimización Clásica fueron reinventados. Zimmermann recoge gran cantidad de técnicas de resolución y las muestra como generalizaciones de técnicas básicas de Programación Matemática. No obstante en la mayoría de los casos necesita estructurar algebraicamente el conjunto de soluciones factibles.

En esta memoria se hace uso sólo de la información debida a la relación de orden definida sobre el conjunto factible. Esta información se incorpora en el desarrollo de un esquema de

ESQUEMAS DE ENUMERACION IMPLICITA ORDENADA:  
PROGRAMACION ENTERA.

enumeración implícita a través de lo que en el capítulo anterior se denominaba dirección entera de búsqueda ordenada. Este concepto definía la operación de selección de los subproblemas estudiados en cada iteración del algoritmo.

Para la construcción de la dirección entera de búsqueda es necesario ordenar parcialmente el conjunto de soluciones factibles, pero debemos exigir además la unicidad de antecesor directo (Definición 1.2.4.3). Esta propiedad adicional es precisamente la que permite la búsqueda evitando duplicaciones en el árbol de cadenas definidas por el orden parcial.

En este capítulo se construye un orden parcial sobre  $\mathbb{Z}_+^n$  que verifica esta propiedad especial, al cual denominamos c-léxico. La idea que motiva su construcción es la de definir la secuencia de sucesores de un punto usando alternativamente las direcciones elementales  $e^i$   $i=1\dots n$ , respetando con ello, la simetría en el sentido de no destacar ninguna dirección especial. Además, como se verá en general los vectores tienen más de un sucesor directo (en dimensión superior a 2) propiedad que será utilizada en el desarrollo del algoritmo para problemas con restricciones modulares del capítulo 3 y en el paso atrás del algoritmo aproximado para problemas enteros descrito en el capítulo 5.

Por último, se extiende la definición del orden c-léxico a la intersección de conos poliédricos con  $\mathbb{Z}_+^n$ . Para ello utilizamos el concepto de base de Hilbert dado por Giles y Pulleyblank (1979).

## 2.2 ORDEN C-LEXICO EN $\mathbb{Z}_+^2$ .

### 2.2.1 Definición.

El orden c-léxico en  $\mathbb{Z}_+^2$  es la relación binaria dada por:  
 $(x,y), (u,v) \in \mathbb{Z}_+^2$ ,  $(x,y) \preceq_c (u,v) \Leftrightarrow$  se verifican las condiciones  
 (a) y (b) siguientes:

(a)  $(x,y) \preceq_1 (u,v)$  para el orden lexicográfico.

(b) Se verifica uno de los tres casos siguientes:

(b-1) Si  $x \neq 0, y \neq 0 \longrightarrow \Phi(x,y) = \Phi(u,v)$ .

(b-2) Si  $x=0 \longrightarrow \Phi(x,y) \geq \Phi(u,v)$ .

(b-3) Si  $y=0 \longrightarrow \Phi(x,y) \leq \Phi(u,v)$ .

Donde,

$$\Phi(x,y) = \left[ \frac{x-y}{2} \right]$$

con  $[\ ]$  representando la parte entera de un número.  $\square$

**Nota.** De la definición anterior se deduce que el punto  $(0,0)$ , se relaciona con todos los restantes pares enteros positivos puesto que se verifica (b-2) o (b-3),  $\forall (u,v) \in \mathbb{Z}_+^2$ . A este punto lo denominamos raíz.

Al orden estricto lo notaremos por  $<_c$ .

2.2.2 Teorema.

El c-léxico es un orden parcial en  $\mathbb{Z}_+^2$ .

**Demostración.**

Consideremos tres pares enteros  $p=(x,y)$ ,  $q=(u,v)$  y  $r=(t,s)$  se verifican las siguientes propiedades:

-Reflexiva:

$$p \leq_c p, \text{ puesto que } \Phi(x,y)=\Phi(x,y) !$$

-Antisimétrica:

$$\left. \begin{array}{l} p \leq_c q, \\ q \leq_c p \end{array} \right\} \Rightarrow p = q, \text{ puesto que en particular, } p \text{ y } q$$

están relacionados según el orden lexicográfico.

-Transitiva:

Supongamos  $p \leq_c q \leq_c r$ , entonces  $p \leq_c r$  para el orden lexicográfico. Vamos a probar que se verifica la condición (b) de la definición del orden c-léxico. Para probarlo distinguimos varios casos, según la desigualdad del apartado (b) que se cumple en las relaciones entre  $p$ ,  $q$  y  $r$ , por tanto tenemos que considerar nueve casos.

$$p \leq_c q \leq_c r$$

$$\begin{array}{cc} (b-n) & (b-m) \end{array}$$

Los casos  $(b-1)(b-1)$ ,  $(b-2)(b-2)$ ,  $(b-2)(b-1)$ ,  $(b-3)(b-1)$  y  $(b-3)(b-3)$  se obtienen como consecuencia de la definición.

ESQUEMAS DE ENUMERACION IMPLICITA ORDENADA:  
PROGRAMACION ENTERA.

Caso (b-1)(b-2):

No puede darse pues  $u=0$  y  $p \leq_c q \Rightarrow x=0$ , por tanto va en contra de (b-1).

Caso (b-1)(b-3):

$$(b-3) \Rightarrow v = 0 \Rightarrow x < u,$$

$$(b-1) \Rightarrow \left[ \frac{x-y}{2} \right] = \left[ \frac{u}{2} \right] = k \Rightarrow$$

$$(x-y=2k \text{ ó } x-y=2k+1) \text{ y } (u=2k \text{ ó } u=2k+1) \Rightarrow x \geq u !!!.$$

Caso (b-2)(b-3)

$$(b-2) \Rightarrow \Phi(x,y) \geq \Phi(u,v) \wedge x=0$$

$$(b-3) \Rightarrow \Phi(u,v) \leq \Phi(s,t) \wedge v=0$$

pero si  $x=0 \Rightarrow \Phi(x,y) \leq 0$   $\wedge$  si  $v=0 \Rightarrow \Phi(u,v) \geq 0$  por tanto

$$\Phi(x,y) = \Phi(u,v) = 0$$

$[-y/2]=0 \Rightarrow y=0$  y por lo anterior  $\Phi(x,y) \leq \Phi(s,t)$ , con lo cual la condición (b-3) asegura  $p \leq_c r$ .

Caso (b-3)(b-2)

$$(b-3) \Rightarrow \Phi(x,y) \leq \Phi(u,v) \wedge y=0$$

$$(b-2) \Rightarrow \Phi(u,v) \geq \Phi(s,t) \wedge u=0$$

por otro lado  $(x,y) \leq_c (u,v) \wedge u=0 \Rightarrow x=0$ , además por ser  $u=0 \Rightarrow \Phi(u,v) \leq 0$  luego la única forma de que se verifiquen estas dos condiciones es la siguiente:

$$0 = \Phi(x,y) = \Phi(u,v) \geq \Phi(s,t) \wedge x=0 \text{ que es la condición}$$

(b-2) con lo cual  $p \leq_c r$ . ■

## 2.3 EXTENSION DEL ORDEN A $\mathbb{Z}_+^n$ .

### 2.3.1 Definición.

Dados  $x, y \in \mathbb{Z}_+^n$ , con  $n \geq 3$  diremos que  $x \leq_c y$  según la relación c-léxico sii  $x = y$  o bien se verifica:

(1) Si  $\forall i \ y_i > 0$  :

$$(a) (x_1, x_{i+1}) \leq_c (y_1, y_{i+1}) \ \forall i=1 \dots (n-1).$$

(b) Si  $x \neq y$  sea  $k = \max\{0, 1 / y_1 - y_{i+1} \text{ es par } \}$

entonces:

Si  $k \neq 0$  debe cumplirse

$$(b-1) (x_k, x_{k+1}) <_c (y_k, y_{k+1})$$

$$(b-2) (x_1 \dots x_k) \leq_c (y_1 \dots y_k)$$

$$(b-3) (x_{k+2} \dots x_n) \leq_c (y_{k+2} \dots y_n).$$

Donde, si (b-2) es estricta entonces (b-3) también.

Si  $k = 0$  debe cumplirse

$$(b'-1) (x_1, x_2) <_c (y_1, y_2)$$

$$(b'-2) x_1 \leq y_1 - 1$$

$$(b'-3) (x_2 \dots x_n) \leq_c (y_2 \dots y_n)$$

Donde, si (b'-2) es estricta entonces (b'-3) también.

(2) Si  $\exists i \ / \ y_i = 0$  :

Se cumple (1) para los vectores  $x', y'$  obtenidos al suprimir aquellas componentes que son nulas en  $y$ .

**Nota.**

La definición es recurrente y tiene sentido puesto que ya se ha definido el orden c-léxico en  $\mathbb{Z}_+^2$ .

### 2.3.2 Teorema.

La relación c-léxico es un orden parcial en  $\mathbb{Z}_+^n$ .

#### Demostración.

La propiedad reflexiva se verifica a partir de la definición. La propiedad antisimétrica es consecuencia de esta misma propiedad en  $\mathbb{Z}_+^2$ .

Por otro lado la propiedad transitiva es cierta en  $\mathbb{Z}_+^3$ , como consecuencia de la misma propiedad en  $\mathbb{Z}_+^2$ .

Veremos que la propiedad se verifica en  $\mathbb{Z}_+^n$ . Supongamos  $x \leq_c z$ ,  $x, y, z \in \mathbb{Z}_+^n$ , entonces se verifica  $(x_i, x_{i+1}) \leq_c (y_i, y_{i+1}) \leq_c (z_i, z_{i+1})$   $i=1 \dots n-1$ , además  $y \leq_c z$  implica, según las condiciones (b-i) de la definición de c-léxico en  $\mathbb{Z}_+^n$ , que la última desigualdad se da estrictamente para un conjunto de índices I que depende únicamente de z. Por otro lado puesto que la relación c-léxico es transitiva y antisimétrica en  $\mathbb{Z}_+^2$  se verifica:

$$(x_i, x_{i+1}) \leq_c (z_i, z_{i+1}) \quad i=1 \dots n-1 \quad (\text{estrictas para } i \in I),$$

y por lo tanto queda probado que  $x \leq_c z$ . Con esto se concluye que esta relación es un orden parcial en  $\mathbb{Z}_+^n$ . ■

A continuación demostraremos la existencia y unicidad de antecesor directo de todo  $x \in \mathbb{Z}_+^n \setminus \{0\}$ . Para hacerlo necesitamos un resultado previo enunciado como lema.

ESQUEMAS DE ENUMERACION IMPLICITA ORDENADA:  
PROGRAMACION ENTERA.

2.3.3 Lema. (Condición suficiente de ordenación)

Consideremos  $(y_1 \dots y_{n-1}), (x_1 \dots x_{n-1}) \in \mathbb{Z}_+^{n-1}$  siendo las componentes del vector  $x$  estrictamente positivas. Sea  $(y_{n-1}, y_n) \leq_c (x_{n-1}, x_n)$  siendo  $y_n, x_n > 0 \wedge x_n - x_{n-1}$  impar, entonces:

$$(y_1 \dots y_n) \leq_c (x_1 \dots x_n).$$

**Demostración.**

Supongamos  $(y_1 \dots y_{n-1}) \neq (x_1 \dots x_{n-1})$  en otro caso la prueba es inmediata. Sea  $k = \max \{0, i / x_i - x_{i-1} \text{ es par}, i=1 \dots (n-2)\}$  por definición debe verificarse:

Supuesto  $k > 0$  (el caso  $k=0$  se deduce de la demostración),

$$(y_1 \dots y_k) \leq_c (x_1 \dots x_k)$$

$$(y_k, y_{k+1}) <_c (x_k, x_{k+1}) \quad [1]$$

$$(y_{k+2} \dots y_{n-1}) \leq_c (x_{k+2} \dots x_{n-1}).$$

Si probamos que  $(y_{k+2} \dots y_n) \leq_c (x_{k+2} \dots x_n)$  habremos concluido, puesto que  $x_n - x_{n-1}$  es impar luego  $k_1 = \max \{0, i / x_i - x_{i+1} \text{ es par}, i=1 \dots, (n-1)\} = k$ . Además si  $(y_1 \dots y_k) <_c (x_1 \dots x_k) \Rightarrow (y_{k+2} \dots y_{n-1}) <_c (x_{k+2} \dots x_{n-1})$  por tanto  $(y_{k+2} \dots y_n) <_c (x_{k+2} \dots x_n)$ .

Para probarlo, sabemos

$$(y_{k+2} \dots y_{n-1}) \leq_c (x_{k+2} \dots x_{n-1})$$



ESQUEMAS DE ENUMERACION IMPLICITA ORDENADA:  
PROGRAMACION ENTERA.

$$(y_{n-1}, y_n) \leq_c (x_{n-1}, x_n) \wedge x_n - x_{n-1} \text{ es impar.}$$

Puesto que  $x_i - x_{i-1}$  es impar para  $i=(k+2)\dots(n-2)$ ; usando la definición obtendremos (supuesto que  $(y_{k+2}\dots y_n) \neq (x_{k+2}\dots x_n)$ ):

$$(y_{k+2}, y_{k+3}) <_c (x_{k+2}, x_{k+3}),$$

$$(y_{k+3}\dots y_n) \leq_c (x_{k+3}\dots x_n) \text{ siendo estricta si } y_{k+2} < x_{k+2} - 1$$

Tomemos  $r = \max \{ i / y_i \neq x_i, i=(k+2)\dots n \}$ , entonces:

$$\left. \begin{array}{l} (y_{r-1}, y_r) <_c (x_{r-1}, x_r) \\ x_{r-1} - x_r \text{ es impar} \Rightarrow \text{el antecesor del par es } (x_{r-1} - 1, x_r) \end{array} \right\} \Rightarrow y_{r-1} < x_{r-1}$$

operando de la misma forma obtendremos después de un número finito de pasos que  $y_{k+2} < x_{k+2}$  luego  $(y_{k+2}, y_{k+3}) <_c (x_{k+2}, x_{k+3})$  puesto que ya sabíamos que se daba la desigualdad " $\leq$ " entre ambos pares.

Por otro lado como  $y_r \neq x_r$  se tiene:

$(y_r, y_{r+1}) <_c (x_r, x_{r+1})$  pero por ser  $y_{r+1} = x_{r+1} \Rightarrow y_r = x_r - 1$  además  $(y_{r+1}\dots y_n) = (x_{r+1}\dots x_n)$  por tanto  $(y_r\dots y_n) <_c (x_r\dots x_n)$  que junto con  $(y_{r-1}, y_r) <_c (x_{r-1}, x_r)$  determina  $(y_{r-1}\dots y_n) <_c (x_{r-1}\dots x_n)$  por definición. Si se opera de la misma forma durante un número finito de pasos obtenemos  $(y_{k+2}\dots y_n) <_c (x_{k+2}\dots x_n)$ .

Esto se ha obtenido bajo el supuesto  $(y_{k+2}\dots y_n) \neq (x_{k+2}\dots x_n)$ . En caso contrario, de [1] obtenemos por ser  $(y_{k+2}\dots y_n) = (x_{k+2}\dots x_n)$  y según la definición del orden, que  $(y_1\dots y_k) = (x_1\dots x_k)$  y por último, de ser  $(y_k, y_{k+1}) <_c (x_k, x_{k+1})$  con  $y_k = x_k \wedge$

$x_{k+1} - x_k$  par, se obtiene que  $y_{k+1} = x_{k+1} - 1$  y por tanto:

$$(y_1 \dots y_n) = (x_1 \dots x_k, x_{k+1} - 1, x_{k+2} \dots x_n) = x^-.$$

En consecuencia queda probado el lema.  $\square$

**Nota.**

Supuesto que alguna de las componentes de  $x$  es nula en el Lema 1, la misma componente en  $y$  debe ser nula, (en  $\mathbb{Z}_+^2$  es cierto a partir de la definición y en  $\mathbb{Z}_+^n$  por el apartado (a) de la definición del orden), en este caso, suprimiendo las componentes nulas comunes obtenemos puntos en un espacio de dimensión inferior a  $n$  para los que sigue siendo válido el resultado, con lo cual se tiene la propiedad para los puntos originales ya que en la definición del orden no se tienen en cuenta las componentes que hemos suprimido.

**2.3.4 Teorema.**

Sea  $(\mathbb{Z}_+^n, \leq_c)$  el conjunto de vectores enteros positivos ordenados según el  $c$ -léxico entonces:

Todo vector  $x \in \mathbb{Z}_+^n \setminus \{0\}$  tiene un único antecesor directo.

ESQUEMAS DE ENUMERACION IMPLICITA ORDENADA:  
PROGRAMACION ENTERA.

**Demostración.**

Sea  $x \in \mathbb{Z}_+^n$  verificando  $x_i > 0 \forall i$ , probaremos que tiene un único antecesor directo, posteriormente se probará lo análogo cuando alguna de sus componentes es nula (excepto para el vector nulo). Es de interés en la demostración el conocimiento del siguiente hecho:

En  $\mathbb{Z}_+^2$  dado  $x=(x_1, x_2)$  con  $x_1, x_2 \neq 0$ :

- Si  $x_1 - x_2$  es impar  $x^- = (x_1 - 1, x_2)$

- Si  $x_1 - x_2$  es par  $x^- = (x_1, x_2 - 1)$

En otro caso  $x_1 = 0$  ó  $x_2 = 0$ , y el c-léxico induce el orden natural sobre los ejes, por tanto existe único antecesor en estos casos salvo para el (0,0).

Para demostrar el resultado consideramos tres casos que dependen del valor de  $k = \max \{0, i / x_i - x_{i+1} \text{ es par}, i=1 \dots (n-1) \}$ .

**Caso 1.  $k=n-1$ .**

Sea  $x^-=(x_1 \dots x_{n-1}, x_n - 1)$ , se verifica  $x^- \leq_c x$  por definición, además  $[x^-, x] = \{x^-, x\}$ . Veamos que es único, para ello supongamos  $y <_c x$ , probaremos que  $y \leq_c x^-$ .

Por definición:

$$(y_1 \dots y_{n-1}) \leq_c (x_1 \dots x_{n-1})$$

$$(y_{n-1}, y_n) <_c (x_{n-1}, x_n).$$

Puesto que el único antecesor directo de  $(x_{n-1}, x_n)$  es  $(x_{n-1}, x_n - 1)$  debe cumplirse  $(y_{n-1}, y_n) \leq_c (x_{n-1}, x_n - 1)$ . Este último vector tiene diferencia impar entre sus componentes y puesto que  $(y_1 \dots y_{n-1}) \leq_c$

ESQUEMAS DE ENUMERACION IMPLICITA ORDENADA:  
PROGRAMACION ENTERA.

$(x_1 \dots x_{n-1})$  aplicando el lema se obtiene el resultado  $y \leq_c x^-$ .

Caso 2.  $0 < k < n-1$ .

Sea  $x^- = (x_1 \dots x_k, x_{k+1} - 1, x_{k+2} \dots x_n)$  que es un antecesor directo de  $x$ . Probaremos la unicidad del mismo modo que antes.

Sea  $y <_c x$ , por definición:

$$\left. \begin{array}{l} (y_1 \dots y_k) \leq_c (x_1 \dots x_k) \\ (y_k, y_{k+1}) <_c (x_k, x_{k+1}) \\ (y_{k+2} \dots y_n) \leq_c (x_{k+2} \dots x_n) \end{array} \right\} \begin{array}{l} \text{(caso 1)} \\ \Rightarrow (y_1 \dots y_{k+1}) \leq_c (x_1 \dots x_{k+1} - 1) \end{array}$$

Por otro lado, como  $(y_k, y_{k+1}) <_c (x_k, x_{k+1}) \wedge x_k - x_{k+1}$  es par  $\Rightarrow y_{k+1} < x_{k+1}$  de donde  $(y_{k+1}, y_{k+2}) <_c (x_{k+1}, x_{k+2})$ .

Además  $(y_{k+2} \dots y_n) \leq_c (x_{k+2} \dots x_n)$ .

Si existiese  $r \geq r+2$  tal que  $y_r \neq x_r$  implicaría  $(y_{k+2} \dots y_n) <_c (x_{k+2} \dots x_n)$  este último vector tiene todas sus diferencias impares, luego  $y_{k+2} < x_{k+2} \wedge (y_{k+3} \dots y_n) \leq_c (x_{k+3} \dots x_n)$ . Como  $y_{k+2} < x_{k+2} \Rightarrow (y_{k+1}, y_{k+2}) <_c (x_{k+1} - 1, x_{k+2})$  y por tanto:

$$\left. \begin{array}{l} (y_1 \dots y_{k+1}) \leq_c (x_1 \dots x_{k+1} - 1) \\ (y_{k+1}, y_{k+2}) <_c (x_{k+1} - 1, x_{k+2}) \\ (y_{k+3} \dots y_n) <_c (x_{k+3} \dots x_n) \end{array} \right\} \Rightarrow y <_c x^-$$

Esto se ha probado en el supuesto  $(y_{k+2} \dots y_n) \neq (x_{k+2} \dots x_n)$  en caso contrario, por definición de  $y \leq_c x$  tenemos:

ESQUEMAS DE ENUMERACION IMPLICITA ORDENADA:  
PROGRAMACION ENTERA.

$$\left. \begin{aligned} (y_1 \dots y_k) &= (x_1 \dots x_k) \\ (y_k, y_{k+1}) &<_c (x_k, x_{k+1}) \text{ con } x_{k+1} - x_k \text{ par} \\ y_k &= x_k \end{aligned} \right\} \Rightarrow y_{k+1} = x_{k+1} - 1$$

y por tanto

$$y = (x_1 \dots x_k, x_{k+1}, x_{k+2} \dots x_n) = x^-.$$

Caso 3.  $k = 0$ .

Sea  $x^- = (x_1 - 1, x_2 \dots x_n)$  que es antecesor directo de  $x$  por definición. Sea  $y <_c x \Rightarrow (y_1, y_2) <_c (x_1, x_2) \wedge (y_2 \dots y_n) \leq_c (x_2 \dots x_n)$  siendo esta última desigualdad estricta si  $y_1 < x_1 - 1$ .

Como  $(y_1, y_2) <_c (x_1, x_2)$  y este último vector tiene como único antecesor directo a  $(x_1 - 1, x_2)$  se tiene:

$$\left. \begin{aligned} (y_1, y_2) &\leq_c (x_1 - 1, x_2) \\ (y_2, y_3) &\leq_c (x_2, x_3) \end{aligned} \right\} \begin{array}{l} \text{(lema)} \\ \Rightarrow (y_1, y_2, y_3) \leq_c (x_1 - 1, x_2, x_3) \end{array}$$

Operando de la misma forma con el resto de las componentes (todas las diferencias  $x_i - x_{i+1}$  son impares) se obtiene finalmente  $y \leq_c x^-$  con lo cual queda probado el resultado.

El resultado anterior sigue siendo válido si alguna de las componentes de  $x$  es nula, basta para ello suprimir las componentes nulas de  $x$  y después aplicar el resultado al vector resultante. Una vez obtenido el antecesor directo, al añadir las componentes nulas obtenemos la misma propiedad debido a las características

ESQUEMAS DE ENUMERACION IMPLICITA ORDENADA:  
PROGRAMACION ENTERA.

del orden en  $\mathbb{Z}_+^2$  (si  $(y_1, y_2) \leq_c (x_1, x_2)$  y  $x_1$  ó  $x_2 = 0 \Rightarrow y_1$  ó  $y_2 = 0$ ).

Una vez probada la propiedad anterior es de utilidad el caracterizar los antecesores y los sucesores directos de un vector  $x \in \mathbb{Z}_+^n$ . Mientras que la unicidad de antecesores directos asegura que no se efectuarán cálculos inútiles en un algoritmo que seleccione puntos según una *debo*, los sucesores directos serán útiles en la evolución del algoritmo. No obstante un punto no tiene siempre un único sucesor, incluso existen puntos sin sucesores. Estos puntos tienen la propiedad de poseer sucesores directos en una dimensión superior. A continuación caracterizamos los dos tipos de puntos:

**Antecesores directos.**

Sea  $x \in \mathbb{Z}_+^n \setminus \{0\}$ , según el resultado anterior, para encontrar el antecesor directo debemos operar de la siguiente forma:

(1) Suprimir las componentes nulas de  $x$ , sea  $x'$  el vector resultante.

(2) Sea  $k = \max \{ 0, i / x'_i - x'_{i+1} \text{ es par} \}$ .

Disminuir una unidad la componente con valor  $x'_{k+1}$ .

(3) Añadir al vector resultante aquellas componentes nulas suprimidas en el paso 1.

De lo anterior se deduce que un mismo vector puede tener varios sucesores directos, de hecho el vector nulo tiene  $n$

ESQUEMAS DE ENUMERACION IMPLICITA ORDENADA:  
PROGRAMACION ENTERA.

sucesores.

**Sucesores directos.**

Sea  $x \in \mathbb{Z}_+^n / x_i \neq 0 \forall i$ . El conjunto de sucesores directos se obtiene aplicando una de las reglas siguientes:

(1) Aumentar en una unidad la componente  $x_i$ , cuando las componentes  $i-1$  e  $i$  del nuevo vector determinen la diferencia par de mayor índice.

(2) Aumentar en una unidad  $x_1$  cuando  $x_1 - x_2$  sea la única diferencia par de  $x$ .

Si  $x$  tiene componentes nulas, son también sucesores directos aquellos puntos obtenidos al aplicar una de las reglas anteriores a cualquier vector obtenido de  $x$  suprimiendo todas las componentes nulas salvo una de ellas que es la incrementada por la regla usada.

Ejemplo:

Son sucesores del  $(0,2,3,4)$  los vectores  $(0,2,3,5)$  y  $(1,2,3,4)$ .

## 2.4 EXTENSION DEL ORDEN C-LEXICO A CONOS.

En las secciones anteriores hemos definido y estudiado una relación de orden parcial en el cuadrante entero positivo  $Z_+^n$ , en ésta se extenderá la definición a una familia de subconjuntos de  $Z^n$  a la cual pertenece el cuadrante  $Z_+^n$ . A tales subconjuntos los denominamos restricción entera de conos.

### 2.4.1 Definición.

Sean  $v^1 \dots v^p$ ,  $p$  vectores del espacio vectorial  $R^n$ , denominamos restricción entera del cono generado por estos vectores al conjunto

$$\Lambda_Z(v^1 \dots v^p) = Z^n \cap \{ x / \exists \lambda_1 \dots \lambda_p \geq 0, x = \sum_1^p \lambda_i v^i \}$$

A partir de ahora supondremos que los vectores  $v^i$   $\forall i$  tienen componentes racionales y los suponemos escalados de tal forma que todas sus componentes son enteras con m.c.d. igual a 1.

### 2.4.2 Definición.

Denominamos orígenes múltiples relativos al cono  $\Lambda_Z(v^1 \dots v^p)$  al conjunto  $O_\Lambda$  dado por la siguiente expresión:

$$O_\Lambda = \{ x \in Z^n / \exists \lambda_1 \dots \lambda_p \in [0, 1) \wedge x = \sum_1^p \lambda_i v^i \}.$$

La definición del orden c-léxico sobre restricciones enteras de conos se efectúa en base a la representación de sus puntos enunciada en el siguiente resultado.



### 2.4.3 Teorema.

Todo  $x \in \Lambda_{\mathbb{Z}}(v^1 \dots v^p)$  admite de forma única una representación en las condiciones siguientes:

$$x = o(x) + \sum_1^p \mu(x)_i v^i$$

donde  $o(x) \in O_{\Lambda}$  y  $\mu(x) \in \mathbb{Z}_+^n$ .

#### Demostración.

Dado  $x \in \Lambda_{\mathbb{Z}}(v^1 \dots v^p)$  se tiene por definición  $x = \sum_1^p \lambda_i v^i$  además esta expresión es única debido al Teorema de Representación de Conjuntos Poliédricos.

Sea  $\lambda(x)_i = \lambda_i - [\lambda_i]$  y  $\mu(x)_i = [\lambda_i]$ ,  $o(x) = \sum_1^p \lambda(x)_i v^i$ . Esta descomposición de los escalares  $\lambda_i$  es la única que verifica todas las condiciones del teorema, en otro caso supongamos que  $\lambda_i = \bar{\lambda}(x)_i + \bar{\mu}(x)_i$  en las condiciones del teorema. Restando ambas expresiones de  $\lambda_i$  obtendríamos  $\lambda(x)_i - \bar{\lambda}(x)_i \in \mathbb{Z} \cap (-1, 1) \Rightarrow \lambda(x)_i = \bar{\lambda}(x)_i \Rightarrow \mu(x)_i - \bar{\mu}(x)_i$  con lo cual se garantiza el resultado. ■

A partir de la demostración anterior se deduce que todo vector de  $\Lambda_{\mathbb{Z}}(v^1 \dots v^p)$  se puede poner como una combinación entera positiva de los vectores del conjunto

$$O_{\Lambda} \cup \{v^1 \dots v^p\}$$

por tanto son una base de Hilbert según la definición de Giles y Pulleyblank (1979).

**2.4.4 Definición.** (Orden c-léxico en  $\Lambda_z(v^1 \dots v^p)$ )

Sean  $v^1 \dots v^p$  vectores n-dimensionales en las condiciones impuestas anteriormente, sean  $x, y \in \Lambda_z(v^1 \dots v^p)$  decimos que x está relacionado con y según el orden c-léxico definido en  $\Lambda_z(v^1 \dots v^p)$  y lo representamos por  $x \leq_{\Lambda_c} y$  y si se verifica una de las dos condiciones siguientes:

(a)  $x = 0$ .

(b)  $\lambda(x) \equiv \lambda(y) \wedge \mu(x) \leq_c \mu(y)$ .

□

A partir de la definición anterior y haciendo uso de la unicidad de la representación de los puntos de  $\Lambda_z(v^1 \dots v^p)$  es inmediato probar que la nueva relación binaria hereda todas las propiedades del orden parcial c-léxico.

**2.4.5 Propiedad.**

La relación binaria c-léxico definida en  $\Lambda_z(v^1 \dots v^p)$  es un orden parcial verificando que todo punto del conjunto  $\Lambda_z(v^1 \dots v^p)$  excepto el vector nulo tiene un único antecesor directo.

Esta propiedad asegura la construcción de una dirección entera de búsqueda ordenada verificándose que todos los sucesores de un punto x se encuentran contenidos en el cono de origen x y aristas  $v^1 \dots v^p$ . Este hecho será utilizado en la construcción de algoritmos de enumeración implícita para la optimización de funciones cuadráticas en un conjunto de vectores enteros. Posteriormente se aplicará en optimización multiobjetivo.

## CAPITULO 3.

### PROGRAMACION LINEAL ENTERA: RELAJACION MODULAR.

#### 3.1 INTRODUCCION.

Un problema lineal entero es aquél en el que, tanto la función a optimizar como las restricciones están dadas en términos de funciones lineales en la variables de decisión que se suponen enteras positivas. Este problema de optimización tiene carácter  $\mathcal{NP}$ -completo (Karp (1972), Parker y Rardin (1988)) lo que hace que, aunque ha sido ampliamente estudiado en la literatura, actualmente se sigue investigando en la obtención de nuevos resultados que mejoren el tiempo de cómputo de los algoritmos existentes. En este capítulo se estudia una aplicación del método de enumeración implícita basado en direcciones de búsqueda ordenada. Usaremos este esquema para resolver un problema relajado del problema entero original conocido con el nombre de problema **Lineal Entero con Restricciones Modulares (LERM)**.

El problema relajado LERM fue usado inicialmente por Gomory (1965). Gomory probó que suprimiendo las restricciones de no negatividad de algunas variables el problema entero puede transformarse en uno cuyas columnas de coeficientes de la matriz

ESQUEMAS DE ENUMERACION IMPLICITA ORDENADA:  
PROGRAMACION ENTERA.

de restricciones pertenecen a un grupo abeliano. Si al resolver este problema, las variables no restringidas en signo toman valores positivos, el problema original está resuelto.

Este primer trabajo de Gomory dio lugar a una metodología de resolución de la Programación Entera que se muestra como una alternativa a los Métodos basados en Hiperplanos de Corte (MHC) o a los Métodos de Ramificación y Acotación (MRA) basados en la resolución de relajaciones continuas del problema.

Para resolver un problema lineal modular se usan principalmente dos tipos de algoritmos. El primer tipo se basa en esquemas de determinación de caminos más cortos en grafos. Esto es posible gracias a la identificación del conjunto factible del problema con el conjunto de caminos que unen dos vértices de un grafo. Concretamente el grafo construido tiene un nodo asociado a cada elemento del grupo al cual pertenecen las columnas de la matriz de restricciones del problema modular. Los dos vértices entre los cuales debemos determinar el camino de longitud mínima son aquellos asociados al vector nulo y al vector que constituye el lado derecho de las restricciones del problema modular. Los autores que a continuación se mencionan construyeron algoritmos de este tipo en los cuales aprovechan la estructura especial que tiene el grafo utilizado : Hu(1970), Shapiro(1968), Chen y Zions (1970).

Wolsey (1973) observó que bajo esta metodología tenía una interpretación natural el hecho de buscar una solución del

ESQUEMAS DE ENUMERACION IMPLICITA ORDENADA:  
PROGRAMACION ENTERA.

problema relajado que fuese solución óptima del problema original. En su trabajo demuestra la existencia de  $k \in \mathbb{N}$  tal que el  $k$ -ésimo camino más corto entre los vértices mencionados anteriormente se identifica con la solución del problema original. Esto motivó la construcción de un algoritmo de enumeración de los caminos más cortos de un grafo basado en el trabajo de Lawler (1972) y Wolsey (1973). Una versión más reciente de este algoritmo puede encontrarse en Nemhauser y Wolsey (1988).

Una desventaja de esta metodología consiste en la poca efectividad de los algoritmos cuando el tamaño de los grafos se incrementa, por esta razón se han investigado posibles representaciones del problema relajado que tengan en cuenta esta limitación. En esta dirección están los trabajos de Gorry et al. (1972) y (1973).

El segundo tipo de algoritmos usados a la hora de resolver el problema lineal modular se basa en esquemas de enumeración implícita. Son importantes los trabajos de Floyd(1962), Dantzig(1963), Glover (1966), Farbey et al.(1967), Glover (1969), Glover y Litzler(1969), Gorry y Shapiro(1971), Chen y Zions (1976). El esquema que se propone en este capítulo se enmarca dentro de este bloque de algoritmos. Para el desarrollo del proceso de enumeración se usará la dirección entera de búsqueda ordenada basada en la relación c-léxico. Posteriormente se probarán algunas propiedades que haciendo uso de la naturaleza del conjunto factible permitirán la construcción de un test de

ESQUEMAS DE ENUMERACION IMPLICITA ORDENADA:  
PROGRAMACION ENTERA.

supresión de ramas de búsqueda.

En la primera sección del capítulo se describe la construcción del problema relajado, en la segunda se describe el algoritmo de resolución cuya exactitud se demuestra posteriormente. A continuación se obtiene una condición suficiente, distinta de la propuesta por Gomory (1969), para garantizar la optimalidad de la solución del problema modular respecto del problema original. Las últimas secciones se dedican al estudio teórico de la complejidad del algoritmo así como a la construcción de un esquema para la resolución del problema lineal entero original.

### 3.2 CONSTRUCCION DE LA RELAJACION MODULAR.

En esta sección se construye una relajación del problema entero original, de la cual se obtiene el problema modular clásico introducido por Gomory (1965). Para su desarrollo se parte de la siguiente formulación:

$$\begin{array}{ll} \max & \bar{c} \bar{x} \\ \text{sa:} & \bar{A} \bar{x} \leq \bar{b} \quad \bar{x} \in \mathbb{Z}_+^n \end{array} \quad (\text{PEP-1})$$

donde  $\bar{A}$  es una matriz  $m \times n$  de números enteros y  $\bar{b}$  es un vector entero de forma que el conjunto factible que determinan está acotado.

Introduciendo  $m$  variables de holgura podemos obtener una formulación equivalente del problema anterior:

ESQUEMAS DE ENUMERACION IMPLICITA ORDENADA:  
PROGRAMACION ENTERA.

$$\begin{aligned} \max \quad & \bar{c}' \bar{x}' \\ \text{sa: } & \bar{A}' \bar{x}' = \bar{b}' \quad \bar{x}' \in \mathbb{Z}_+^{n+m} \end{aligned} \quad (\text{PEP-2})$$

Para construir el problema modular usaremos una matriz básica asociada al sistema de ecuaciones no redundantes:  $\bar{A}' \bar{x}' = \bar{b}'$ .

La elección de esta matriz básica es arbitraria, sin embargo a continuación se elige una matriz concreta cuyas propiedades serán tratadas más adelante. Tomaremos la submatriz B de  $\bar{A}'$  cuyas columnas corresponden con las variables que son básicas en la solución óptima del problema lineal obtenido al suprimir la restricción de integridad sobre las variables.

Sea N la matriz cuyas columnas no son básicas y supongamos que las coordenadas de  $\bar{x}'$  están ordenadas de tal forma que

$$\bar{x}' = \begin{pmatrix} \bar{x}'_B \\ \bar{x}'_N \end{pmatrix}$$

donde el subíndice representa el conjunto de índices asociados a las componentes básicas y no básicas respectivamente.

En esta situación podemos formular PEP-2 de forma equivalente haciendo uso de estas nuevas matrices:

$$\begin{aligned} \max \quad & (\bar{c}'_B B^{-1} \bar{b}') - (\bar{c}'_B B^{-1} N - \bar{c}'_N) \bar{x}'_N \\ \text{sa: } & \bar{x}'_B = B^{-1} \bar{b}' - B^{-1} N \bar{x}'_N \\ & \bar{x}'_B \geq 0, \quad \bar{x}'_N \geq 0 \text{ enteras.} \end{aligned} \quad (\text{PEP-3})$$

Según se ha seleccionado la base B, la condición de optimalidad del problema lineal asegura las desigualdades:

ESQUEMAS DE ENUMERACION IMPLICITA ORDENADA:  
PROGRAMACION ENTERA.

$$(\bar{c}'_B B^{-1} N - \bar{c}'_N) \geq 0.$$

En el problema relajado de PEP-3 que se construye posteriormente, las variables  $\bar{x}'_N$  son positivas lo que hace, teniendo en cuenta las desigualdades anteriores, que una vez resuelto este problema se obtenga una cota superior del problema PEP-1 que mejora a la obtenida resolviendo su relajación continua. Esta es una de las razones por las cuales se elige la matriz básica óptima para la formulación de PEP-3.

Para obtener la relajación modular de PEP-3 se suprime la restricción de no negatividad impuesta sobre las variables básicas, obteniéndose con ello un conjunto factible que contiene al del problema original:

$$B^{-1} \bar{b}' - B^{-1} N \bar{x}'_N \in \mathbb{Z}^m$$

$$\bar{x}'_N \geq 0 \text{ enteras.}$$

Este conjunto también puede representarse en término de un sistema de congruencias:

$$B^{-1} N \bar{x}'_N = B^{-1} \bar{b}' \pmod{1}$$

$$\bar{x}'_N \in \mathbb{Z}_+^n.$$

Según propiedades elementales de Algebra Lineal, sabemos que los elementos de  $B^{-1}$  son racionales con denominador  $D = |\det(B)|$ , por tanto se pueden reescribir las ecuaciones anteriores en módulo  $D$ :

$$Ax = b \pmod{\Delta}$$

$$x \in \mathbb{Z}_+^n.$$

donde

$$A = D B^{-1} N \in \mathbb{Z}^{m \times n}, \quad b = D B^{-1} \bar{b}' \in \mathbb{Z}_+^m, \quad \Delta = D \cdot 1 \in \mathbb{Z}_+^m, \quad x = \bar{x}'_N.$$



ESQUEMAS DE ENUMERACION IMPLICITA ORDENADA:  
PROGRAMACION ENTERA.

Por último eliminando la constante  $(\bar{c}'_B B^{-1} \bar{b}')$  de la función objetivo de PEP-3 y haciendo  $c = (\bar{c}'_B B^{-1} N - \bar{c}'_N)$  obtenemos la formulación equivalente del problema Lineal Entero con Restricciones Modulares

$$\begin{aligned} \min \quad & cx \\ \text{sa:} \quad & Ax = b \pmod{\Delta} \\ & x \in \mathbb{Z}_+^n \end{aligned} \quad (\text{LERM-1})$$

En la construcción de este problema relajado hemos obtenido el vector  $\Delta$  con todas las componentes iguales. Esto no ocurre en otras formulaciones existentes. La más conocida en la literatura es la que hace uso de la Forma Normal de Smith de una matriz. Se propuso esta formulación con la intención de reducir las dimensiones de un grafo asociado al conjunto factible del problema modular. La construcción de este grafo permite resolver el problema LERM-1 con técnicas de Programación Dinámica (Nemhauser-Wolsey(1988)). Para obtener la formulación del problema modular bajo esta perspectiva se parte de la siguiente relajación del problema PEP-2:

$$\begin{aligned} \max \quad & \bar{c}' \bar{x}' - \beta \omega \\ \text{sa:} \quad & \bar{A}' \bar{x}' - K\omega = \bar{b}' \\ & \bar{x}' \in \mathbb{Z}_+^n, \omega \in \mathbb{Z}^p. \end{aligned} \quad (\text{PEP-4})$$

donde  $K$  es una  $m \times p$  matriz entera,  $p \leq m$  y  $\beta$  es un  $p$ -vector. En esta situación se sabe (Salkin y Mathur (1989)) que tomando  $K = B$ , matriz básica óptima de la relajación continua de PEP-2, y  $\beta = u_0 B$  con  $u_0$  solución factible del problema dual de la relajación

ESQUEMAS DE ENUMERACION IMPLICITA ORDENADA:  
PROGRAMACION ENTERA.

continua de PEP-2 entonces el problema PEP-4 es equivalente a

$$\begin{aligned} \min & (\bar{c}'_B B^{-1} N - \bar{c}'_N) \bar{x}'_N \\ \text{sa: } & R B \bar{x}'_N - \Delta \omega = R \bar{b}' \quad (\text{PEP-5}) \\ & \bar{x}'_N \in \mathbb{Z}_+^n, \quad \omega \in \mathbb{Z}^p. \end{aligned}$$

donde  $R$  es una  $m \times m$  matriz entera unimodular tal que  $\exists C$ ,  $p \times p$  matriz entera unimodular, verificando  $RKC = \Delta$ . Donde  $\Delta$  es una matriz diagonal con elementos enteros positivos. La existencia de estas matrices está garantizada por la Forma Normal de Smith.

Por último describiendo el conjunto factible por medio de congruencias y tomando  $c = (\bar{c}'_B B^{-1} N - \bar{c}'_N)$ , obtendremos un nuevo problema lineal modular que es una relajación del problema PEP-1:

$$\begin{aligned} \min & cx \\ \text{sa: } & R B \bar{x}'_N = R \bar{b}' \pmod{\Delta} \quad (\text{LERM-2}) \\ & x \in \mathbb{Z}_+^n. \end{aligned}$$

donde  $\Delta = (\delta_1 \dots \delta_n)'$ .

### 3.3 RESOLUCION DEL PROBLEMA MODULAR ENTERO.

En esta sección se dan las bases de un algoritmo para el problema LERM que denominamos C-LEXMOD. Este, es un esquema de enumeración implícita que aprovecha las propiedades del sistema de congruencias en la supresión de ramas de búsqueda.

ESQUEMAS DE ENUMERACION IMPLICITA ORDENADA:  
PROGRAMACION ENTERA.

En principio se partirá de una formulación del problema en la que los índices de las componentes están ordenados según coeficientes de costos crecientes:

$$\begin{aligned} & \min cx \\ & \text{sa: } Ax = b \pmod{\Delta} \qquad \qquad \qquad (\text{LERM-3}) \\ & \quad x \in \mathbb{Z}_+^n. \end{aligned}$$

donde  $c \in \mathbb{R}^n$ ,  $0 \leq c_1 \leq c_2 \leq \dots \leq c_n$ ,  $(A, b) \in \mathbb{M}_{m \times (n+1)}$  con coeficientes enteros,  $\Delta = (\delta_1 \dots \delta_n)'$ ,  $\delta_i \in \mathbb{Z}_+$ .

El algoritmo C-LEXMOD tiene algunas similitudes con el de Glover (1969), estudiado en el Capítulo 1 y que se emplea para el problema de Knapsack modular. Sin embargo la operación de selección de puntos se basa en la ordenación dada por la relación c-léxico. De esta forma se consiguen obtener buenas propiedades en relación con el almacenamiento de información y se evitarán algunos de los problemas que presenta el algoritmo de Glover relativos a lo que él denomina índices de transición. Concretamente los problemas que surgen al suprimir ramas de búsqueda del algoritmo.

En el capítulo 1 se dio un esquema general de un algoritmo de enumeración implícita basado en direcciones enteras de búsqueda ordenada, según este esquema debemos elegir un superconjunto E del conjunto factible, asimismo debemos especificar el orden parcial que genera el gredoide local y por último debemos fijar la

ESQUEMAS DE ENUMERACION IMPLICITA ORDENADA:  
PROGRAMACION ENTERA.

propiedad  $\mathcal{P}$  que define la operación de selección de nuevos elementos.

En primer lugar para construir E podemos usar cotas sobre las variables del problema PEP-1. Estas cotas pueden venir dadas explícitamente en la formulación inicial o bien pueden determinarse en función de los elementos de  $\bar{A}$  y  $\bar{b}$  (Giles y Pulleyblank (1979), Papadimitriou (1982)). No obstante, en este problema especial podemos usar un resultado de Gomory (1965) que indica que la solución de LERM-2(3) verifica  $\sum_{j=1}^n x_j \leq D-1$ , con  $D = |\det(B)|$ , así podemos tomar  $E = I(D-1 \dots D-1)$ .

Continuando con el desarrollo especificado en la sección 1.3 debemos determinar el gredoiide local parcialmente ordenado  $(E \setminus \mathcal{D}', \mathfrak{J}_c \setminus \mathcal{D}')$  sobre la cual se aplica el esquema de enumeración implícita.

En este caso la familia  $\mathfrak{J}_c$  se determina haciendo uso del orden c-léxico y de la definición 1.2.4.3, por otro lado

$$\mathcal{D}' = \{x \in \mathcal{D} / \exists y \in \mathcal{D}, y \leq_c x\} \text{ y } \mathcal{D} = \{x \in \mathbb{Z}_+^n / Ax = b \pmod{\Delta}\}.$$

Por último la propiedad  $\mathcal{P}$  empleada en la operación de selección será la regla de mínima etiqueta, es decir en la iteración k:

$$\phi(X(k)) = \operatorname{argmin}\{cx / x \in E \setminus X(k) \wedge X(k) \cup \{x\} \in \mathfrak{J}\}$$

en caso de objetivos mínimos empatados elegimos el máximo lexicográfico.

ESQUEMAS DE ENUMERACION IMPLICITA ORDENADA:  
PROGRAMACION ENTERA.

Para continuar con la aplicación del esquema general del algoritmo al caso particular del problema LERM-2(3) profundizaremos en la descripción de la Operación de Selección y posteriormente se indicará como se realiza la Operación de Supresión.

### 3.3.1 Operación de Selección.

En esta sección se describe el procedimiento mediante el cual se generan vectores a los que se aplicará un test de factibilidad en cada una de las iteraciones del algoritmo. Según se ha dicho con anterioridad, la operación de selección de estos nuevos elementos se efectúa según la regla de mínima etiqueta aplicada al conjunto de aquellos sucesores directos, en el orden c-léxico, de los vectores estudiados en alguna iteración anterior.

Con esto, la operación de selección queda descrita funcionalmente aunque no algorítmicamente, para hacerlo se detallará a continuación la estructura utilizada para el almacenamiento de información. Esta información se refiere fundamentalmente al conjunto de sucesores directos de los vectores estudiados y a su valor objetivo. Una vez descritas estas estructuras de información se indicará como se actualizan en cada iteración, posteriormente se probarán ciertos resultados que garantizaran que la actualización descrita corresponde con la aplicación de la regla de mínima etiqueta en la generación de soluciones.

ESQUEMAS DE ENUMERACION IMPLICITA ORDENADA:  
PROGRAMACION ENTERA.

Para el almacenamiento de los sucesores, según el orden c-léxico, se utilizará un sistema de  $n$  listas enlazadas  $S_i$   $i=1\dots n$  donde  $n$  es la dimensión del vector  $x$  de variables de decisión. En principio  $S_i$  consta únicamente del elemento  $e^i$  que tiene todas sus componentes nulas salvo el valor 1 en la  $i$ -ésima componente. Haremos uso, asimismo de la lista  $L$  que en cualquier iteración consiste del primer vector almacenado en cada lista  $S_i$   $i=1\dots n$ . Este conjunto contendrá las posibles elecciones  $y$  y en cada iteración se tomará aquel vector de  $L$  con menor objetivo que es máximo para el orden lexicográfico.

Supongamos que en la iteración  $k$  es seleccionado de  $L$  el elemento  $x(k)$ , sean  $y^1\dots y^r$  sus sucesores directos. Para almacenar  $y^1$  supongamos que  $y^1 = x(k) + e^1$ . En este caso añadimos el vector  $y^1$  al final de la lista  $S_1$ . Una vez realizada esta operación con todos los sucesores podemos suprimir a  $x(k)$  de  $L$ . Su lugar en  $L$  es ocupado por el elemento que sucede a  $x(k)$  en la lista del tipo  $S$  donde estaba almacenado. Una vez descrita la estructura de información necesaria para implementar la operación de selección y determinada la forma en que se actualizan sus componentes pasaremos a desarrollar la operación de supresión de ramas de búsqueda del esquema de enumeración implícita.

### 3.3.2 Supresión de ramas de búsqueda.

En cada iteración, una vez determinado  $x(k)$  seleccionaremos de entre sus sucesores, aquellos que serán almacenados. Esta selección se hace en base a una propiedad que será probada en la próxima sección. En ella se establece que si  $y$  es uno de los sucesores de  $x(k)$  de forma que el lado derecho que obtiene  $Ay \pmod{\Delta}$ , ya ha sido obtenido con anterioridad por alguno de los puntos generados por el algoritmo, entonces la solución óptima que es máxima para el orden lexicográfico no es sucesor de  $y$  en el orden c-lexico. Por tanto se puede suprimir el punto  $y$  de posteriores consideraciones.

Por esta razón, en la iteración  $k$  sólo se almacenarán en alguna de las listas  $S_i$  aquellos sucesores que generen nuevos elementos del grupo aditivo. Para chequear esta condición debemos implementar una estructura que almacene los elementos del grupo generados desde que comenzó el algoritmo y sobre la cual sea eficiente la búsqueda e inserción de elementos. Una buena elección es la que en Sedgewick (1983) se denomina "top-down 2-3-4 tree". Éstas son árboles de búsqueda, en los que una operación de búsqueda-inserción se realiza con complejidad  $O(\log N)$  siendo  $N$  el número de elementos del árbol y supuesto que cada uno de ellos tiene una etiqueta unidimensional. En nuestro caso podemos utilizar estas estructuras ordenando los nodos del árbol según el orden lexicográfico (que es completo). Puesto que la comparación de dos vectores necesita en el peor de los casos  $m$  operaciones, la

ESQUEMAS DE ENUMERACION IMPLICITA ORDENADA:  
PROGRAMACION ENTERA.

complejidad de una operación de búsqueda o inserción es  $O(\log(mN))$ .

A continuación presentamos el algoritmo de enumeración implícita que se basa en las operaciones anteriormente descritas.

### 3.3.3 Esquema general del Algoritmo C-LEXMOD.

El algoritmo se compone de una etapa de inicialización y de las iteraciones, estas a su vez se construyen a través de tres operaciones básicas:

- Selección de elementos de la lista L.
- Determinación de los sucesores del elemento generado.
- Supresión de ramas de búsqueda.

Describimos a continuación estas operaciones.

#### *Inicialización.*

Sea L la lista ordenada  $L = \{e^1, \dots, e^n\}$ .

$S_i = \{e^i\} \forall i=1 \dots n$ .

$V(i)=1 \forall i=1 \dots n$ . (Indica que existe un vector en la i-ésima posición de L, posteriormente puede tomar el valor cero indicando lo contrario)

Almacenar en el árbol de búsqueda T los elementos del grupo generados por los elementos de L, esto es  $\{a^1 \dots a^n\}$  donde  $a^i \in Z^m$  es la i-ésima columna de la matriz de restricciones A.

Almacenar en una lista de etiquetas  $E(i)$  los elementos  $(i, c_i)$   $i \in \{1 \dots n\}$ . Esta lista contiene el valor objetivo de los elementos de L (inicialmente  $c_i$ ) y el índice de la lista a la que



ESQUEMAS DE ENUMERACION IMPLICITA ORDENADA:  
PROGRAMACION ENTERA.

pertenece el vector  $(S_1)$ .

Además están colocados en orden creciente de costos.

Hacer  $k=1$ .

Iteración  $k=1, 2, \dots$

**Selección.** Sea  $x(k)$  el elemento de  $L$  con menor valor objetivo, en caso de empate de valores objetivos elegir aquel que sea máximo para el orden lexicográfico. Supongamos que ocupa la posición  $i$  de  $L$ . Suprimir  $x(k)$  de  $L$ .

**Determinación de sucesores directos.** Calcular los sucesores directos de  $x(k)$  en el orden c-léxico., sean  $y^1 \dots y^r$ .

**Supresión-almacenamiento de sucesores.** Para  $j=1 \dots r$ :

Calcular el elemento del grupo que se obtiene mediante  $y^j$ , sea  $b^j = Ay^j$ .

Si  $b^j \in T$ , suprimir  $y^j$  de posteriores consideraciones.

Si  $b^j \notin T$ :

Si  $b^j = b$ . FIN:  $x^* = y^j$  es solución óptima.

Si  $b^j \neq b$ , puesto que  $y^j$  es sucesor directo de  $x(k)$  se tiene  $y^j = x(k) + e^l$  para algún valor de  $l \in \{1 \dots n\}$ .

Introducir, en este caso, el vector  $y^j$  en la última posición de la lista  $S_1$ . Si  $V(l) = 0$  introducir  $y^j$  en la  $l$ -ésima posición de la lista  $L$  e introducir el par  $(l, cy^j)$  en la lista de etiquetas haciendo que se mantenga ordenada por costos crecientes.

**Adaptación.** Introducir en la posición  $i$ -ésima de  $L$  el primer vector  $x \in S_1$  e introducir el par  $(i, cx)$  en la lista  $E()$  del

ESQUEMAS DE ENUMERACION IMPLICITA ORDENADA:  
PROGRAMACION ENTERA.

mismo modo que se ha descrito anteriormente. Caso de no existir este elemento hacer  $V(i)=1$ .

Hacer  $k=k+1$ .

Tanto las demostraciones de las propiedades que verifica la secuencia de puntos generada por el algoritmo como la justificación de la operación de supresión de nodos serán tratadas en el próximo apartado.

### 3.4 EXACTITUD DEL ALGORITMO C-LEXMOD.

Probaremos algunas propiedades que garantizarán la finalización del algoritmo con la solución óptima siempre que el problema tenga soluciones factibles.

#### 3.4.1. Teorema.

Supongamos que en el esquema del algoritmo C-LEXMOD eliminamos la operación de supresión, entonces la operación de selección implementada es la regla de mínima etiqueta, es decir

$$\phi(X(k)) = \operatorname{argmin}_x \{ cx / x \in EX(k) \wedge X(k) \cup \{x\} \in \tilde{X} \}$$

#### Demostración.

Según el esquema del algoritmo y supuesto que no se ha suprimido ningún vector se verifica que las listas  $S_i$ ,  $i=1\dots n$  recogen en la iteración  $k$  todos los sucesores directos de los vectores generados hasta el momento con lo cual a partir del

ESQUEMAS DE ENUMERACION IMPLICITA ORDENADA:  
PROGRAMACION ENTERA.

Teorema 1.2.4.7 en la iteración  $k+1$  se tiene

$$\{ x \in E \setminus X(k) / X(k) \cup \{x\} \in \mathfrak{F} \} = \bigcup_{i=1}^n S_i.$$

Para probar el resultado basta observar que los vectores almacenados en la lista  $S_i$  ocupan posiciones en orden creciente de costo. Puesto que  $L$  está compuesta del primer elemento de cada lista  $S_i$   $i=1 \dots n$ , se tendrá la propiedad.

Usaremos una inducción sobre el índice de la iteración.

Sea  $p=1$ . En este caso es cierto puesto que cada lista tiene sólo un elemento.

Sea  $p=2$ . Puesto que en la iteración anterior se eligió  $e^1$  sus sucesores directos habrán sido almacenados en las listas. Así si  $S_i$  es actualizada  $i \neq 1$  tendrá como primer elemento  $e^1$  y en segundo lugar  $e^1 + e^i$  que verifican la propiedad pues todos los coeficientes de costo son positivos.  $S_1$  tiene un único elemento:  $2e^1$ .

Supongamos cierto la propiedad para  $p-1$ . En este caso, las soluciones generadas en las  $p$  primeras iteraciones del algoritmo verifican  $cx(1) \leq cx(2) \leq \dots \leq cx(p-1) \leq cx(p)$ .

Sea  $y = e^1 + x(p)$  sucesor directo de  $x(p)$  que será almacenado en la última posición de  $S_1$ . Sea  $y^0$  un vector de  $S_1$ , por tanto  $y^0 = e^1 + x(q)$  con  $q < p$  puesto que  $y^0$  no puede ser sucesor directo de  $x(p)$  en el orden  $c$ -léxico. Por las desigualdades anteriores se tiene:

ESQUEMAS DE ENUMERACION IMPLICITA ORDENADA:  
PROGRAMACION ENTERA.

$$cx(q) \leq cx(p) \Rightarrow cx(q)+c_1 \leq cx(p)+c_1 \Rightarrow cy^0 \leq cy.$$

Por tanto se ha probado la propiedad. ■

### 3.4.2. Corolario.

La regla de selección descrita en el esquema del algoritmo C-LEXMOD implica una ordenación lexicográfica decreciente en la lista  $S_1$ , de aquellos vectores con el mismo valor objetivo, para cualquier índice  $i$ .

#### Demostración.

Con la misma notación de la demostración anterior, si suponemos que  $cy^0 = cy$  entonces  $cx(q) = cx(p)$  y según la regla de selección, por ser  $q < p$ ,  $x(q) >_1 x(p)$  con lo que  $y^0 >_1 y$ . ■

La propiedad recogida en el corolario anterior asegura que para la aplicación de la regla de selección empleada en C-LEXMOD basta conocer los elementos de la lista  $L$ , esto influirá favorablemente en la complejidad teórica del esquema.

Las propiedades anteriores aseguran la exactitud del algoritmo en la resolución de LERM-2(3) en virtud del teorema 1.3.1. A continuación se estudia como afecta la operación de supresión de elementos al esquema anterior. Se probará que, si un vector es suprimido, implícitamente se suprime el conjunto de vectores enteros con componentes mayores o iguales que éste. Esta propiedad es muy interesante puesto que acelera la enumeración implícita.

### 3.4.3. Teorema .

Sea  $x^0$  un punto suprimido por el algoritmo, sea  $x^*$  la solución óptima máxima para el orden lexicográfico, entonces  $x^*$  posee al menos una componente estrictamente menor que  $x^0$ .

#### Demostración.

Supongamos que el problema LERM-3 es factible, entonces el conjunto de soluciones óptimas verifican  $\sum_{i=1}^n x_i \leq (D-1)$  según se indicó en la sección 3.2, por tanto existirá el máximo lexicográfico  $x^*$  por ser este conjunto acotado. Supongamos que  $x^*$  tiene todas sus componentes mayores que las de  $x^0$ , sea

$$x^* = x^0 + x^1, \quad x_j^1 \geq 0 \quad \forall j = 1 \dots n.$$

Según el algoritmo si  $x^0$  es suprimido entonces

$$\exists i / cx(i) \leq cx^0 \wedge Ax^0 = Ax(i).$$

Tomemos  $x(i)+x^1$  que tiene componentes positivas y veamos que posee un valor objetivo óptimo

$$c(x(i)+x^1) \leq c(x^0+x^1) = cx^* \wedge A(x(i)+x^1) = A(x^0+x^1) = Ax^*.$$

Por tanto

$$cx(i) = cx^0.$$

Esta igualdad junto con la operación de selección del algoritmo implica que  $x(i)$  es lexicográficamente mayor que  $x^0$  con lo que

$$x(i) + x^1 >_1 x^0 + x^1 = x^*$$

obteniéndose un resultado que contradice la elección de  $x^*$ . ■

#### 3.4.4. Teorema.

Si  $x^0$  ha sido suprimido en la iteración  $k$ , no será generado en ninguna iteración posterior ningún vector  $x$  verificando  $x_j \geq x_j^0$   $\forall j=1\dots n$ .

#### Demostración.

Supongamos que  $\exists i / cx(i) \leq cx^0$ ,  $Ax(i) = Ax^0$ , entonces  $x^0$  es suprimido en una iteración  $k > i$  del algoritmo. Supongamos que en la iteración  $j > k$  se genera el vector  $x(j) / x(j)_1 \geq x^0_1 \forall l=1\dots n$  (alguna de estas desigualdades estricta). Tomemos  $x(i)+x(j)-x^0 \geq 0$   $A(x(i)+x(j)-x^0) = Ax(j)$ . Si  $c(x(i)+x(j)-x^0) > cx(j) \Rightarrow cx(i) > cx^0$  lo cual es imposible. Por tanto necesariamente  $c(x(i)+x(j)-x^0) = cx(j)$  pues en otro caso  $x(j)$  no hubiese sido generado.

Por tanto  $cx(i) = cx^0 \Rightarrow$  según la regla de selección  $\phi()$ ,  $x(i)$  es mayor lexicográfico que  $x^0 \Rightarrow x(i)-x^0$  tiene la primera componente no nula, positiva  $\Rightarrow x(i)+x(j)-x^0$  es mayor lexicográfico que  $x(j)$ , lo cual es contradictorio pues en tal caso  $x(i)+x(j)-x^0$  hubiese sido generado por el algoritmo en lugar de  $x(j)$ . ■

### 3.5 COMPLEJIDAD DEL ALGORITMO C-LEXMOD.

En este apartado se estudia el número de operaciones aritméticas realizadas en el algoritmo C-LEXMOD en el peor de los casos posibles. Suponemos que el problema grupo viene dado con los coeficientes de la función objetivo en orden creciente. La constante  $\delta$  representa el número de elementos del grupo aditivo,  $n$  el número de variables y  $m$  el número de restricciones.

#### *Modulo de inicialización.*

Inicialización de la lista L:  $n^2$  operaciones de asignación.

Inicialización de  $V(i)$   $i=1\dots n$ :  $n$  operaciones.

Inicialización del árbol de búsqueda T: Según se indicó en la sección 3.3.2, la operación de búsqueda-inserción en una estructura *top-down* 2-3-4 el número de operaciones aritméticas realizadas depende logarítmicamente respecto al número de nodos en el árbol. Por tanto para la inicialización de T se obtiene complejidad de orden  $m+\log(m)+\log(2m)+\dots+\log((n-1)m)$ , que está acotada por  $m+(n-1)\log(m)+n\log(n)$  donde se ha supuesto, sin pérdida de generalidad, que las  $n$  columnas de la matriz de restricciones son distintas.

Inicialización de las listas  $S_i$   $i=1\dots n$ :  $n$  operaciones.

Inicialización de la lista de etiquetas  $E(i)$ ,  $i=1\dots n$ :  $O(n)$  operaciones.

La suma de estas operaciones hace que la parte de inicialización tenga complejidad  $O(n^2+n\log(mn)+4n+m)$ .

*Iteraciones.*

Selección: Para elegir el vector con objetivo mínimo que es máximo lexicográfico se puede usar una estructura *top-down* 2-3-4 de forma que la selección se realiza con un numero de operaciones aritméticas  $O(\log(n))$ .

Determinación de sucesores directos:  $n^2$  operaciones.

Supresión-Almacenamiento de sucesores: Para cada uno de los sucesores (a lo sumo  $n$ ) del punto seleccionado se realizan las siguientes operaciones:

Cálculo del lado derecho:  $(m+1)n$  operaciones.

Supresión-Almacenamiento en T: Puesto que  $k$  es el índice de la iteración actual, el número de vectores almacenados en el árbol T está acotado superiormente por  $n+kn$  ( $n$  iniciales y a lo sumo  $n$  por cada uno de los vectores generados). Por tanto, el número de operaciones aritméticas en la supresión-almacenamiento de un vector es  $O(\log(mn(k+1)))$ .

Comparación del lado derecho con  $b$ :  $m$  operaciones.

Almacenamiento del sucesor: 1 operación.

Almacenamiento de su etiqueta en  $E()$ : Consiste en el cálculo del valor objetivo del vector almacenado y en la posible reordenación de la lista  $E()$ , por lo que la complejidad es  $O(n)$ .

Adaptación: Introducir en la  $i$ -ésima posición de  $L$  el primer elemento de  $S_i$  requiere 1 operación y la reordenación de etiquetas en  $E()$  es de orden  $O(n)$ .



ESQUEMAS DE ENUMERACION IMPLICITA ORDENADA:  
PROGRAMACION ENTERA.

Teniendo en cuenta que el test de finalización del algoritmo se acepta antes de  $\delta$  iteraciones, puesto que no se generarán elementos del grupo aditivo duplicados, la complejidad de esta segunda parte del algoritmo es

$$O(\delta n^2 m + n \log((mn)^{\delta+1} \prod_{k=1}^{\delta+1} k)).$$

En definitiva, para  $m$  y  $n$  fijos el algoritmo C-LEXMOD tiene una complejidad inferior al presentado por Hu(1970) ( $O(\delta^2)$ ) y a las versiones de los algoritmos de Gomory, Hu y Shapiro presentadas por Chen(1976), ya que

$$\delta n^2 m + n \log((mn)^{\delta+1} \prod_{k=1}^{\delta+1} k) \leq \delta n^2 m + n(\delta+1) \log(mn(\delta+1))$$

con lo que queda complejidad inferior a  $O(\delta \log(\delta))$ .

### 3.6 CONDICION SUFICIENTE DE OPTIMALIDAD.

En este apartado se obtiene una condición suficiente bajo la cual la solución óptima del problema LERM-1 lo es del problema PEP-1.

#### 3.6.1. Teorema.

Sea  $B$  la matriz básica usada para formular PEP-1 como LERM-1 y  $D = |\det(B)|$  entonces si

$$D^{-1} \leq \min_{i,j} \left\{ \frac{(B^{-1}b)_i}{(B^{-1}N)_{ij}} / (B^{-1}N)_{ij} > 0 \right\}.$$

ambos problemas son equivalentes.

ESQUEMAS DE ENUMERACION IMPLICITA ORDENADA:  
PROGRAMACION ENTERA.

**Demostración.**

Según la notación usada en la construcción de la relajación modular del problema PEP-1 tenemos

$$\begin{pmatrix} \bar{x}'_B \\ \bar{x}'_N \end{pmatrix} = \begin{pmatrix} B^{-1}\bar{b}' \\ 0 \end{pmatrix}$$

solución de la relajación continua de PEP-2. A partir de la matriz básica B se deducen las ecuaciones:

$$\bar{x}'_B = B^{-1}\bar{b}' - B^{-1}N\bar{x}'_N$$

en PEP-3. Por otro lado, sabemos que la solución de LERM-1 verifica

$$1 \bar{x}'_N \leq D-1.$$

Así se tiene la siguiente cadena de desigualdades

$$\bar{x}' \geq 0 \Leftrightarrow B^{-1}\bar{b}' \geq B^{-1}N\bar{x}'_N \Leftrightarrow \sum_{j=1}^n (\bar{x}'_N)_j (B^{-1}N)_{ij} \leq (B^{-1}\bar{b}')_i$$

como

$$\sum_{j=1}^n (\bar{x}'_N)_j (B^{-1}N)_{ij} \leq (D-1) \max_j \{ (B^{-1}N)_{ij} > 0 \}$$

basta imponer

$$D-1 \leq \frac{(B^{-1}\bar{b}')_i}{\max_j \{ (B^{-1}N)_{ij} > 0 \}} \quad \forall i = 1 \dots m$$

de donde se deduce el resultado. ■

Salkin y Mathur(1989) recogen una condición suficiente dada por Gomory(1969) bajo la cual se obtiene la equivalencia de los

problemas LERM-1 y PEP-1. Esta condición se basa en la definición del siguiente conjunto:

$$K_B(d) = \{ t \in \mathbb{R}^n / d(t, \partial K_B) \geq d \wedge B^{-1}t \geq 0 \}$$

donde  $K_B$  es el cono agudo generado por las columnas de  $B$ ,  $\partial K_B$  es su frontera y  $d(t, H)$  es la mínima distancia euclídea entre el punto  $t$  y un punto de  $H$ .

### 3.6.2 Condición suficiente de Optimalidad. (Gomory(1969))

Si  $b$  pertenece a  $K_B(1_m(D-1))$  donde  $D = |\det(B)|$  y  $1_m$  es el máximo de la norma euclídea en el conjunto de columnas no básicas de la matriz  $\bar{A}'$  del problema PEP-2, entonces toda solución óptima de LERM-1 es solución óptima de PEP-1.

A continuación se obtienen restricciones sobre los parámetros del problema PEP-1 de forma que la condición suficiente del teorema 3.6.1 mejore a la condición dada por Gomory, es decir bajo tales restricciones si el vector  $b$  verifica la condición de Gomory también verificará la del teorema 3.6.1.

### 3.6.3 Teorema.

La condición suficiente 3.6.1 mejora a la condición suficiente de Gomory siempre que se verifique:

$$1_m \begin{pmatrix} \|\beta_1\| \\ \vdots \\ \|\beta_m\| \end{pmatrix} \geq D \begin{pmatrix} \omega_1 \\ \vdots \\ \omega_m \end{pmatrix}$$

donde  $\beta_1^0 = (\beta_{11}^0 \dots \beta_{1m}^0)'$ ,  $\beta_1 = \text{sig}(\beta_1^0 b_1) \beta_1^0$ ,  $\text{sig}()$  es la función signo,  $\beta_{ij}^0$  el adjunto  $(i,j)$  de la matriz  $B$ ,  $D = |\det(B)|$  y  $\omega_1 = \max_j \{(B^{-1}N)_{ij} > 0\}$ .

#### Demostración.

En primer lugar se sabe que  $\beta_1^0 = (\beta_{11}^0 \dots \beta_{1m}^0)$  es el vector ortogonal a las columnas de  $B$   $\{ b^1 \dots b^{i-1}, b^{i+1} \dots b^m \}$  por construcción. Esto implica que

$$\beta_1 t \geq 0$$

representa el semiespacio cuyo hiperplano frontera pasa por el origen de coordenadas y es generado por los  $(m-1)$  vectores independientes  $\{ b^1 \dots b^{i-1}, b^{i+1} \dots b^m \}$ . Además según la definición de  $\beta_1$ , se verifica que  $b^i$  está en el semiespacio positivo, i.e.  $\beta_1 b^i > 0$  (el mayor estricto se da por ser  $\det B \neq 0$ ). Por tanto

$$\{ t \in \mathbb{R}^m / \beta_1 t \geq \|\beta_1\| 1_m (D-1) \wedge B^{-1} t \geq 0 \}$$

representa el conjunto de vectores del cono agudo generado por las columnas de  $B$  que distan más de  $1_m (D-1)$  de la cara del cono  $K_B$

ESQUEMAS DE ENUMERACION IMPLICITA ORDENADA:  
PROGRAMACION ENTERA.

generada por las columnas  $\{ b^1 \dots b^{i-1}, b^{i+1} \dots b^m \}$ ,  $\forall i=1 \dots m$ .

Según la definición de  $K_B(1_m(D-1))$  se verifica la siguiente igualdad de conjuntos

$$K_B(1_m(D-1)) \equiv \{ t \in \mathbb{R}^m / \beta_i t \geq \|\beta_i\| 1_m(D-1) \ i=1 \dots m \wedge B^{-1}t \geq 0 \}.$$

Claramente,  $K_B(1_m(D-1))$  es un cono agudo y tiene como único vértice el vector  $t^0$  que es solución del siguiente sistema de ecuaciones

$$\beta_i t^0 = \|\beta_i\| 1_m(D-1) \ i=1 \dots m.$$

En forma matricial

$$\beta t^0 = 1_m(D-1) \begin{pmatrix} \|\beta_1\| \\ \vdots \\ \|\beta_m\| \end{pmatrix}$$

con  $\beta$  la matriz  $m \times m$  cuyas filas son los vectores  $\beta_i'$ ,  $i=1 \dots m$ .

En el sistema de ecuaciones anterior las filas de  $\beta$  son linealmente independientes, en caso contrario

$$\exists \lambda_k \ k=1 \dots m \text{ con } \lambda_j \neq 0 / \sum_{k=1}^m \lambda_k \beta_k = 0$$

lo cual implica

$$\left( \sum_{k=1}^m \lambda_k \beta_k \right) b^j = 0$$

Por definición de los vectores  $\beta_k$ , el miembro izquierdo de la ecuación anterior es

ESQUEMAS DE ENUMERACION IMPLICITA ORDENADA:  
PROGRAMACION ENTERA.

$$\lambda_j \beta_j b^j$$

con lo que el vector  $\beta_j$  es ortogonal a toda una base de  $\mathbb{R}^m$  como es  $\{b^1 \dots b^m\}$ , lo cual implica que  $\beta_j = 0$ , pero esto es contradictorio por definición de  $\beta_{ij}^0$ , ya que implicaría que  $\det B = 0$ .

En conclusión,  $\beta \in \mathbb{M}_{m \times m}$  es invertible y podemos determinar la única solución del sistema anterior

$$t^0 = \frac{1}{m} (D-1) \beta^{-1} \begin{pmatrix} \|\beta_1\| \\ \vdots \\ \|\beta_m\| \end{pmatrix}.$$

Por la forma en que se han construido los vectores  $\beta_1$ , cualquier dirección del cono generado por las columnas de B es dirección de  $k_B(1 (D-1))$  con lo cual, en virtud del Teorema de Representación de Conjuntos Poliédricos.

$$t \in K_B(1 (D-1)) \Leftrightarrow t = t^0 + \sum_{k=1}^m \lambda_k b^k, \quad \lambda_k \geq 0 \quad \forall k=1 \dots m.$$

Supongamos ahora que se verifica la condición suficiente de Gomory para el vector b, entonces, según lo anterior

$$b = t^0 + \sum_{k=1}^m \lambda_k b^k, \quad \lambda_k \geq 0 \quad \forall k=1 \dots m$$

con lo cual, para que se verifique la condición del teorema 3.6.1 es necesario y suficiente que

$$B^{-1} t^0 \geq (D-1) \omega.$$

A partir de estas últimas desigualdades, sustituyendo  $t^0$  por su valor anteriormente determinado obtenemos el resultado indicado en el teorema:

ESQUEMAS DE ENUMERACION IMPLICITA ORDENADA:  
PROGRAMACION ENTERA.

$$I_m \begin{pmatrix} \|\beta_1\| \\ \vdots \\ \|\beta_m\| \end{pmatrix} \cong D \begin{pmatrix} \omega_1 \\ \vdots \\ \omega_m \end{pmatrix} \quad \blacksquare$$

A continuación se aplica la condición suficiente del teorema 3.6.3 a un problema numérico.

**Ejemplo.**

El problema que consideramos fue expuesto por Gomory (1969) y posteriormente recogido por Salkin y Mathur (1989) (pag. 386).

$$\begin{aligned} \max \quad & 2x_1 + x_2 + x_3 + 3x_4 + x_5 \\ \text{sa:} \quad & 2x_2 + x_3 + 4x_4 + 2x_5 \leq 41 \\ & 3x_1 - 4x_2 + 4x_3 - x_4 - x_5 \leq 47 \\ & x_1, x_2, x_3, x_4, x_5 \in \mathbb{Z}_+ \end{aligned}$$

Después de resolver su relajación continua obtenemos la solución óptima (43, 20.5, 0, 0, 0) asociada a la base B:

$$B = \begin{pmatrix} 0 & 2 \\ 3 & -4 \end{pmatrix} \quad \text{con } |\det(B)| = D = 6, \quad N = \begin{pmatrix} 1 & 4 & 2 & 1 & 0 \\ 4 & 1 & -1 & 0 & 1 \end{pmatrix}.$$

En esta situación:

$$B^{-1}N = \begin{pmatrix} 2 & 3 & 1 & 2/3 & 1/3 \\ 1/2 & 2 & 1 & 1/2 & 0 \end{pmatrix} \quad \beta = \begin{pmatrix} 4 & 2 \\ 3 & 0 \end{pmatrix} \quad \beta^{-1} = \begin{pmatrix} 0 & 1/3 \\ 1/2 & -2/3 \end{pmatrix},$$

además  $\|\beta_1\| = 20^{1/2}$ ,  $\|\beta_2\| = 3$ ,  $\omega_1 = 3$ ,  $\omega_2 = 2$  y la condición del teorema queda:

ESQUEMAS DE ENUMERACION IMPLICITA ORDENADA:  
PROGRAMACION ENTERA.

$$17^{1/2} \begin{pmatrix} 20^{1/2} \\ 3 \end{pmatrix} \geq 6 \begin{pmatrix} 3 \\ 2 \end{pmatrix}$$

por tanto, la condición suficiente del teorema 3.6.1 mejora a la condición de Gomory.

### 3.7 ESTUDIO COMPUTACIONAL DEL ALGORITMO C-LEXMOD.

En esta sección se presentan los resultados computacionales realizados al algoritmo C-LEXMOD, o más correctamente a su implementación, ya que en una primera aproximación se puede decir que un algoritmo es un procedimiento general con el que se obtiene la respuesta a todo problema apropiado mediante un simple cálculo de acuerdo con un método dado. No obstante, en este método se dejan sin especificar detalles prácticos que corresponden a su implementación, que les da forma a través de una secuencia de instrucciones. Por lo tanto es inevitable que los resultados computacionales dependan del código mediante el que se implementó el algoritmo.

El estudio que se ha realizado puede enmarcarse dentro del segundo tipo propuesto por Jackson et al (1991) que corresponde al caso en el que se selecciona el rango de problemas al que se aplica, en nuestro caso particular la selección se hace a través del número de variables (n) y el número de elementos del grupo aditivo (D) de problemas de knapsack modulares. Por otro lado se ha utilizado para su implementación un código de investigación sin



ESQUEMAS DE ENUMERACION IMPLICITA ORDENADA:  
PROGRAMACION ENTERA.

hacer un estudio sobre su portabilidad, estabilidad o flexibilidad de uso sobre diferentes máquinas.

Los tiempos de ejecución corresponden a máquinas con arquitecturas similares (CDC 6400, RCA 70/6, SUN 330) y en ningún caso se usó procesamiento en paralelo.

La mayoría de las implementaciones comparadas corresponden a los algoritmos que aparecían en la introducción como esquemas de enumeración implícita, si bien se incluyen además los algoritmos de Gomory (1965) y Shapiro (1968) con el fin de comparar computacionalmente con estos esquemas que se basan en la utilización de técnicas de Programación Dinámica.

La tabla que aparece a continuación es la que recogen Salkin y Mathur (1989) (pg. 504) modificada con la inclusión de los tiempos de cómputo correspondientes al algoritmo C-LEXMOD.

| Algoritmo | D = 20            |      |      | D = 60            |      |      | D = 100 |      |      |
|-----------|-------------------|------|------|-------------------|------|------|---------|------|------|
|           | n=2               | n=10 | n=19 | n=6               | n=30 | n=59 | n=10    | n=50 | n=99 |
| Gomory    | .004              | .017 | .033 | .034              | .155 | .306 | .083    | .391 | .791 |
| Hu        | .008              | .008 | .008 | .059              | .059 | .059 | .163    | .163 | .163 |
| Shapiro   | .006              | .010 | .015 | .029              | .070 | .139 | .064    | .147 | .196 |
| Floyd     | .014              | .014 | .014 | .124              | .124 | .124 | .344    | .342 | .244 |
| Farbey    | .007              | .007 | .007 | .053              | .053 | .053 | .146    | .146 | .146 |
| Dantzig   | .005              | .006 | .011 | .055              | .068 | .098 | .169    | .175 | .184 |
| C-LEXMOD  | .000 <sup>+</sup> | .003 | .007 | .000 <sup>+</sup> | .015 | .053 | .002    | .038 | .114 |

ESQUEMAS DE ENUMERACION IMPLICITA ORDENADA:  
PROGRAMACION ENTERA.

La tabla resume los resultados obtenidos sobre muestras de problemas-grupo cíclicos (con una sola restricción) generados como se indica a continuación. Se seleccionó inicialmente el número  $D$  de elementos del grupo y el número de variables  $n$ . Cada uno de los costos  $c_j$  asociados a las variables fue obtenido según una distribución uniforme en el intervalo  $[20,90]$ . Los coeficientes de la restricción asociados a cada una de las variables  $x_j$ ,  $a_j$  se seleccionaron eligiendo  $a_1$  del conjunto  $G_D = \{0,1,2,\dots,D-1\}$  según una distribución uniforme. A continuación  $a_2$  del conjunto  $G_D \setminus \{a_1\}$  posteriormente  $a_3$  en  $G \setminus \{a_1, a_2\}$  y así sucesivamente hasta que  $a_n$  fue elegido. Se determinó para cada uno de estos problemas la solución para cada uno de los posibles elementos del lado derecho de la restricción, en total  $D$ . Posteriormente se tomó el tiempo promedio de resolución eliminando de esta forma el efecto de la selección del lado derecho de la restricción.

Como se puede comprobar en la tabla, los tiempos de cómputo del algoritmo C-LEXMOD mejoran al resto de algoritmos. Coincide con el de Farbey en dos columnas sin embargo difiere de éste en que el algoritmo C-LEXMOD es dependiente de la razón  $\frac{n}{D}$ , de forma que para valores pequeños de este cociente mejora al algoritmo de Shapiro que es el más eficiente del resto de algoritmos según el estudio realizado.

### 3.8 UN ALGORITMO DE RAMIFICACION Y ACOTACION PARA PEP.

Los autores Parker y Rardin (1988) (pag. 159) definen un algoritmo de ramificación y acotación en optimización discreta como un procedimiento de enumeración parcial que emplea tests de factibilidad y comparación con respecto a soluciones factibles, en la operación de supresión de problemas *candidatos*. Por problema candidato se entiende aquél que optimiza la función objetivo del problema original en un subconjunto del conjunto de puntos factible. Asimismo, estos autores ponen de manifiesto (capítulo 5) que una de las técnicas más prometedoras para la resolución de problemas discretos NP-completos, consiste en aquella que combina esquemas de ramificación y acotación, con la resolución de problemas relajados en la construcción de algoritmos.

En esta sección se construye un algoritmo de ramificación y acotación que usa problemas relajados modulares para resolver el problema entero puro en el que aparecen cotas superiores de las variables de decisión. En términos teóricos esto no es restrictivo, al menos cuando el poliedro está acotado, ya que en este caso se pueden obtener cotas de las variables en función de los parámetros del problema (Giles y Pulleyblank(1979) Papadimitriou(1982)).

En el algoritmo de Ramificación y Acotación, se usan como problemas candidatos, relajaciones modulares de diferentes problemas lineales obtenidos en el proceso. Estos problemas poseen

ESQUEMAS DE ENUMERACION IMPLICITA ORDENADA:  
PROGRAMACION ENTERA.

cotas superiores de las variables, no obstante se pueden resolver mediante el esquema C-LEXMOD modificando la operación de determinación de sucesores directos de forma que, si uno de éstos vectores viola alguna de las cotas es suprimido de posteriores consideraciones. Al resolver un problema candidato se determinará un punto factible del problema entero inicial, o bien una cota inferior del problema obtenido al suprimir los modulos del problema candidato. Además como se puso de manifiesto en la formulación de PEP-3, en este último supuesto la cota obtenida siempre mejora a la que se obtendría al resolver la relajación continua.

Para demostrar la exactitud del método se necesita un resultado, que además motiva la forma en que se realizará la operación de ramificación y que pasamos a describir a continuación:

Definimos el problema PEP(b,u) dado por

$$\min cx$$

$$\text{sa: } D\bar{x}'_B + Ax = b \quad \text{PEP}(b, u)$$

$$x \leq u, \bar{x}'_B \leq u_B, x \in \mathbb{Z}_+^n, x_B \in \mathbb{Z}_+^m$$

donde  $A = DB^{-1}N$ ,  $b \in DB^{-1}\bar{b}' \in \mathbb{Z}_+^m$  y  $c = \bar{c}'_B B^{-1}N - \bar{c}'_N$  y suponemos  $c_j > 0 \forall j$ . En esta situación PEP(b,u) es equivalente a PEP-3 cuando se consideran cotas superiores sobre las variables.

ESQUEMAS DE ENUMERACION IMPLICITA ORDENADA:  
PROGRAMACION ENTERA.

La relajación modular del problema anterior es

min  $Cx$

sa:  $Ax = b \pmod{\Delta}$  LERM(b, u, D)

$x \leq u, x \in \mathbb{Z}_+^n$

donde  $\Delta = D \mathbf{1}$ .

Según la formulación de PEP(b, u) se tiene que  $\bar{x}'_B = 1/D(b - Ax)$  por tanto podemos representar su solución óptima como

$$(\bar{x}'_B, x) = (1/D(b - Ax^*(b, u)), x^*(b, u)),$$

y diremos que el vector  $x^*(b, u)$  define la solución óptima del problema PEP(b, u).

A continuación se demostrará un resultado que justifica la construcción de los problemas candidatos usados en el algoritmo.

Estos problemas se definirán en virtud de los valores  $t_j^k, u_j^k, j, k \in \{1 \dots n\}$ , que dependen de la solución óptima  $x^*(b, u, D)$  del problema LERM(b, u, D) según la siguiente expresión:

$$t_j^k = 0 \text{ si } j \neq k \wedge t_k^k = x^*(b, u, D)_k + 1$$

$$u_j^k = x^*(b, u, D)_j, j=1 \dots (k-1), u_k^k = u_k - x^*(b, u, D)_k - 1, u_j^k = u_j, j=k+1 \dots n.$$

### 3.8.1 Proposición.

Sea  $x^*(b,u)$  el vector que define la solución óptima del problema PEP(b,u), entonces si  $x^*(b,u)$  no coincide con la solución óptima  $x^*(b,u,D)$  del problema LERM(b,u,D) se verifica:

$$x^*(b,u) = t^k + x$$

con  $x$  una solución factible de uno de los problemas candidatos LERM(b-At<sup>k</sup>, u<sup>k</sup>, D).

#### Demostración.

En primer lugar  $x^*(b,u)$  no puede tener sus componentes menores o iguales que las componentes de  $x^*(b,u,D)$  pues en tal caso, por ser  $c_j > 0 \forall j$ , y  $x^*(b,u)$  punto factible de LERM(b,u,D) se tendría que  $x^*(b,u,D)$  no es solución óptima, en contra de lo supuesto. Por lo tanto se deduce que  $x^*(b,u)$  debe pertenecer a uno de los conjuntos

$$G_k \equiv \{t \in Z^n / t_j^k \leq t_j \leq \omega_j^k\}$$

donde  $\omega_j^k = x^*(b,u,D)_j$ ,  $j=1 \dots k-1$ ,  $\omega_j^k = u_j$ ,  $j = k \dots n$ , para algún  $k \in \{1 \dots n\}$ .

Supuesto  $x^*(b,u) \in G_k$  podemos hacer el cambio de variables

$$x^*(b,u) = t^k + x, \quad x \in Z_+^n$$

como

$$t_j^k \leq t_j^k + x_j \leq \omega_j^k \quad \forall j=1 \dots n$$

sustrayendo  $t_j^k$  de ambas desigualdades se obtiene que  $x \leq u^k, x \in \mathbb{Z}_+^n$ .

Por otro lado, sustituyendo el cambio de variables anterior en el problema LERM(b,u,D) queda como sigue:

$$\begin{aligned} & ct^k + \min cx \\ & \text{sa:} \\ & Ax = b - At^k \\ & 0 \leq x \leq u^k \end{aligned}$$

que es equivalente a LERM(b-At<sup>k</sup>,u<sup>k</sup>,D). ■

### 3.8.2 Corolario.

Supuesto  $c_j > 0, u_j < +\infty \forall j=1\dots n$ , entonces para cualquier problema factible PEP(b,u) se puede determinar  $t^0, u^0 \in \mathbb{Z}_+^n / x^*(b,u)$  es solución de LERM(b-At<sup>0</sup>,u<sup>0</sup>,D).

#### Demostración.

Basta aplicar la proposición anterior un número finito de veces, teniendo en cuenta que si  $x^*(b,u) \neq x^*(b,u,D)$  entonces  $\exists k \in \{1\dots n\}$  tal que  $x^*(b,u)$  es solución factible de LERM(b-At<sup>k</sup>,u<sup>k</sup>,D) y en este problema la cota  $u_k^k$  decrece en una unidad respecto a  $u_k$  (además del posible decrecimiento del resto de cotas). Por esta razón, cada vez que se aplica la proposición anterior se reduce el intervalo entero que contiene al conjunto factible del problema modular con lo cual en un número finito de pasos se debe obtener los vectores  $t^0$  y  $u^0$  en las condiciones deseadas. ■

ESQUEMAS DE ENUMERACION IMPLICITA ORDENADA:  
PROGRAMACION ENTERA.

El corolario anterior conduce directamente a la construcción de un esquema de ramificación y acotación para el problema PEP(b,u) que tras un número finito de iteraciones finalizará con la solución óptima.

### 3.8.3 Algoritmo MRA(PEP)

*Inicialización.*

Hacer  $L = \{\text{LERM}(b, u, D)\}$

$z(\text{LERM}(b, u, D)) = 0$

$\bar{z} = +\infty.$

*Iteraciones.*  $k=1, 2, \dots$

1. Si  $L \neq \emptyset$  determinar el problema almacenado en L que verifica:

$$z(\text{LERM}(b^l, u^l, D)) = \min \{ z(\text{LERM}(b^r, u^r, D)) / \text{LERM}(b^r, u^r, D) \in L \}.$$

Suprimir este problema de la lista L.

Resolver  $\text{LERM}(b^l, u^l, D)$  usando el algoritmo C-LEXMOD. Sea

$x^*(b^l, u^l, D)$  su solución.

1.1 Si  $0 \leq 1/D(b - Ax^*(b^l, u^l, D)) \leq u_B$ :

1.1.1 Actualizar  $\bar{z} = \min\{\bar{z}, z(\text{LERM}(b^l, u^l, D)) + cx^*(b^l, u^l, D)\}$

Si se ha modificado  $\bar{z}$  hacer  $\bar{x} = x^*(b^l, u^l, D)$  y

suprimir aquellos problemas de L que

verifiquen  $\bar{z} \leq z(\text{LERM}(b^t, u^t, D))$ .

1.1.2 Hacer  $l=l+1$  e ir al paso 1.

1.2 En caso contrario, para cada  $k=1 \dots n$  hacer:

$$t_j^k = 0 \text{ si } j \neq k \wedge t_k^k = x^*(b^l, u^l, D)_k + 1$$



ESQUEMAS DE ENUMERACION IMPLICITA ORDENADA:  
PROGRAMACION ENTERA.

$$u_j^1 = x^*(b^1, u^1, D)_j \quad j=1 \dots (k-1)$$

$$u_k^1 = u_k^1 - x^*(b^1, u^1, D)_k - 1, \quad u_j^1 = u_j^1 \quad j=k+1 \dots n.$$

Almacenar LERM( $b^1 - At^1_k, u^1_k, D$ ) en L y hacer

$$z(\text{LERM}(b^1 - At^1_k, u^1_k, D)) = z(\text{LERM}(b^1, u^1, D)) + ct^1_k.$$

Sea  $l=l+1$ , volver al paso 1.

2 Si  $L = \emptyset$  :

2.1 Si  $\bar{z} = +\infty$ , PEP es infactible, FIN.

2.2 Si  $\bar{z} < +\infty$ , PEP tiene solución óptima  $\bar{x}$ .

Nemhauser y Wolsey(1988) desarrollaron un algoritmo de ramificación y acotación para el problema lineal entero puro con cotas en las variables. Su esquema se basa, al igual que el presentado en esta sección, en la resolución de relajaciones modulares. Sin embargo, el método usado para resolver estas relajaciones aplica un algoritmo de determinación de caminos más cortos a un grafo identificado con el conjunto factible, esquema que ya ha sido mencionado en la introducción del capítulo. En este grafo existen arcos de  $n$  tipos diferentes, uno por cada variable, de tal forma que un subcamino con vértice inicial 0 se corresponde con el vector entero que tiene por componente  $i$ -ésima el número de arcos del tipo  $i$  contenidos en el subcamino. Esto es una desventaja a la hora de trabajar con problemas con variables acotadas ya que para saber si en la aplicación del algoritmo de

ESQUEMAS DE ENUMERACION IMPLICITA ORDENADA:  
PROGRAMACION ENTERA.

caminos más cortos se genera un vector que rebasa alguna de las cotas superiores, sería necesario llevar el recuento del número de arcos asociados a cada variable en cada uno de los subcaminos generados. Obviamente esto supone un esfuerzo computacional bastante considerable por lo que los autores optaron por eliminar las cotas superiores de los problemas candidatos generados por el algoritmo, produciéndose, en general, intersecciones no vacías entre sus conjuntos factibles. En este punto el algoritmo propuesto en esta sección es una mejora que puede llegar a ser considerable cuando la dimensión del vector de variables crece.

## CAPITULO 4.

### PROBLEMAS ESPECIALES

#### 4.1 INTRODUCCION.

El objetivo de este capítulo es el de mostrar algunas aplicaciones del orden c-léxico generalizado sobre conos, en la construcción de esquemas de enumeración implícita para algunos problemas especiales de optimización entera. En concreto se considera el problema de minimización general con objetivos cuadráticos y el problema lineal con objetivos múltiples. En ambos casos se construyen algoritmos de resolución que son justificados teóricamente.

Los dos algoritmos se ajustan básicamente a las ideas expuestas en el primer capítulo, esto es, la operación de selección se construye en base al gredoides local generado por el orden c-léxico en conos, sin embargo difieren en que mientras el primero busca la factibilidad del punto generado de acuerdo a una regla de mínima etiqueta, el segundo enumera el conjunto de puntos eficientes seleccionándolos de acuerdo al mínimo lexicográfico. Esta diferencia responde a la distinta naturaleza de ambos problemas. En los dos casos la finitud del algoritmo es garan-

tizada, siempre que el conjunto factible sea acotado, en virtud de la no duplicación de puntos generados por una dirección entera de búsqueda.

#### 4.2 PROBLEMAS CON UN UNICO OBJETIVO.

En esta sección se propone un esquema de enumeración implícita para resolver el problema de Programación Entera en el que, tanto la función objetivo como las restricciones se determinan en base a funciones reales generales. Podemos partir de la siguiente formulación del problema:

$$\begin{aligned} & \min f(x) \\ & \text{sa: } g_i(x) \leq b_i \quad i=1\dots m \quad (\text{PG}) \\ & \quad x \in I(k_1 \dots k_n) \end{aligned}$$

donde  $f(), g_i()$  son funciones reales definidas en  $\Omega \subseteq \mathbb{R}^n$  con  $I \equiv [0, k_1] \times \dots \times [0, k_n] \subseteq \Omega$ .

La idea que motiva la construcción del algoritmo para este problema de optimización, se fundamenta en la construcción de una relación de orden parcial respecto de la cual la función objetivo sea creciente. Una vez construida, se impondrán algunas condiciones bajo las cuales es posible aplicar el teorema 1.3.1 para garantizar la optimalidad del algoritmo basado en la *debo* determinada por esta relación.

El orden parcial utilizado es el c-léxico generalizado sobre un cono poliédrico contenido en lo que llamamos cono de

crecimiento relativo a la función objetivo, definido a continuación.

#### 4.2.1 Definición.

Se denomina cono de crecimiento de la función  $f$  en un conjunto  $D \subseteq \Omega$ , al conjunto:

$$\Lambda(f, D) \equiv \bigcap_{x \in D} \Lambda(f, x)$$

donde

$\Lambda(f, x) \equiv \{ u \in \mathbb{R}^n / f(x+\lambda u) \text{ es una función creciente en el conjunto } \{\lambda \geq 0 / x+\lambda u \in \Omega\} \}$ .

Como se apunta en la definición, el conjunto  $\Lambda(f, D)$  es efectivamente un cono ya que, si  $u \in \Lambda(f, x) \Rightarrow \mu u \in \Lambda(f, x) \forall \mu \geq 0 \wedge \forall x \in D$ . No obstante este cono no tiene porqué ser poliédrico, es más puede que no sea convexo con lo cual, en principio no se puede definir la extensión del orden  $c$ -léxico sobre la restricción entera del cono.

Supongamos que  $\Lambda(f, D)$  es convexo y conocemos  $d^1 \dots d^n \in \Lambda(f, I)$  vectores linealmente independientes, entonces el cono agudo que generarán también está incluido en el cono de crecimiento:

$$\Lambda_c \equiv \left\{ \sum_{j=1}^n \lambda_j d^j / \lambda_j \geq 0 \right\} \subseteq \Lambda(f, I).$$

Por tanto se puede definir el orden  $c$ -léxico en este cono poliédrico, verificándose que  $\forall x, y \in I(k_1 \dots k_n) / x \preceq_{\Lambda_c} y \Rightarrow f(x) \leq f(y)$  por definición de  $\Lambda(f, I)$  y del orden  $\preceq_{\Lambda_c}$  (sección 2.4).

ESQUEMAS DE ENUMERACION IMPLICITA ORDENADA:  
PROGRAMACION ENTERA.

Puesto que  $f()$  es creciente respecto a la relación  $\leq_{\Lambda_c}$  en  $I(k_1 \dots k_n)$ , se puede deducir que una vez determinado un punto factible  $x$ , no es necesario considerar los vectores  $y$ , factibles, tales que  $x \leq_{\Lambda_c} y$ . De esta forma, la operación de supresión del algoritmo queda descrita y motiva el siguiente esquema para el problema de optimización PG.

#### 4.2.2 Algoritmo PG.

*Inicialización.*

Hacer  $X(0) = O_{\Lambda}$ , donde  $O_{\Lambda}$  es el conjunto de orígenes múltiples asociados a los vectores  $d^1 \dots d^n$  en la definición de  $\leq_{\Lambda_c}$  (sección 2.4).

Si alguno de los vectores anteriores es factible, elegir el de menor valor y finalizar. En otro caso hacer  $k=1$ , ir a la iteración  $k$ .

*Iteración  $k=1, 2, 3 \dots$*

Seleccionar  $x_k$  dado por la expresión:

$$x_k = \operatorname{argmin}_x \{ f(x) / x \in X(k-1) \wedge x \in I(k_1 \dots k_n) \setminus X(k-1) \}$$

Si  $x_k$  es factible FIN.

En otro caso hacer  $X(k) = X(k-1) \cup \{x_k\}$ ,  $k=k+1$ , ir a la iteración  $k$ .

#### 4.2.3 Teorema.

Sean  $d^1 \dots d^n \in \Lambda(f, D)$  los vectores que generan  $\Lambda_c$ , donde  $\Lambda(f, D)$  es el cono de crecimiento relativo a la función objetivo de PG que suponemos convexo. Entonces, si

$$\{ x \in I(k_1 \dots k_n) / g_i(x) \leq b_i \quad i=1 \dots m \} \subseteq 0 + \Lambda_c$$

y el problema PG es factible, el algoritmo 4.2.2 determina la solución óptima en un número finito de iteraciones.

#### Demostración.

El algoritmo desarrolla una enumeración implícita en

$$0 + \Lambda_c \supseteq \{ x \in I(k_1 \dots k_n) / g_i(x) \leq b_i \quad i=1 \dots m \}$$

utilizando en la selección una regla de mínima etiqueta, además la función  $f()$  es creciente en  $\Lambda_c$  con lo cual el teorema 1.3.1 garantiza que al generar el primer vector factible se obtiene también la optimalidad del problema.

Por otro lado la finitud del número de iteraciones sigue del hecho de que el orden utilizado es una dirección entera de búsqueda ordenada con lo cual no se producen duplicaciones en el conjunto de vectores generados de acuerdo con el teorema 1.2.4.6, como todos estos vectores pertenecen a  $I(k_1 \dots k_n)$ , el número de iteraciones es finito. ■

A continuación se aplica el desarrollo anterior a un tipo especial de funciones explícitamente cuasiconvexas, determinándose las direcciones  $d^1 \dots d^n$  y formulando el problema de tal forma que

se verifiquen las condiciones bajo las cuales el algoritmo 4.2.2 determina la solución óptima.

#### 4.2.4 Funciones explícitamente cuasiconvexas:

Objetivos cuadráticos.

##### 4.2.4.1 Proposición.

Sea la función  $f: \Omega \subseteq \mathbb{R}^n \rightarrow \mathbb{R}$ ,  $I \subseteq \Omega$ , una función explícitamente cuasiconvexa y  $\Omega$  un conjunto convexo, entonces  $\Lambda(f, I)$  es un cono convexo.

##### Demostración.

Sean  $u, v \in \Lambda(f, I)$  probaremos que  $u+v \in \Lambda(f, I)$ . Tomemos  $x$  en el interior del cubo  $I$  y  $\lambda_0 > 0$  de forma que  $\forall \lambda \in [0, \lambda_0]$ ,  $x+\lambda u$ ,  $x+\lambda v$  y  $x+\lambda(u+v) \in I$ , en tal caso

$$f(x) \leq f(x+\lambda u) \leq f(x+\lambda u+\lambda v) \quad \forall \lambda \in [0, \lambda_0]$$

por ser  $u, v \in \Lambda(f, I)$ .

Por otro lado,  $f$  es explícitamente cuasiconvexa, por lo que  $f(x) \leq f(x+\lambda u+\lambda v) \quad \forall \lambda \geq 0 / x+\lambda(u+v) \in \Omega$ , pues en otro caso se llegaría a contradecir las desigualdades válidas en  $\lambda \in [0, \lambda_0]$ . Con esto se tiene probado  $u+v \in \Lambda(f, I)$  ya que en caso contrario  $\exists \lambda_1, \lambda_2 \geq 0, \lambda_1 < \lambda_2 \wedge f(x+\lambda_1(u+v)) > f(x+\lambda_2(u+v)) \geq f(x)$ , pero se verifica  $x+\lambda_1(u+v) \in (x, x+\lambda_2(u+v)) \subseteq \Omega$ , por lo que  $f$  no sería cuasiconvexa. ■



Bajo las hipótesis de la proposición anterior,  $\Lambda(f,I)$  es un cono convexo, por tanto cumple una de las condiciones del teorema 4.2.3 que justifica la exactitud del algoritmo 4.2.2.

Para determinar  $d^1 \dots d^n$  es necesario considerar funciones objetivo particulares, en el próximo apartado se determinan estos vectores para un tipo de funciones cuadráticas.

#### 4.2.4.2 Funciones Cuadráticas.

Consideremos el problema

$$\begin{aligned} \min \quad & cx + \frac{1}{2} xQx \\ \text{sa: } & g_i(x) \leq b_i \quad i=1 \dots m \\ & x \in I(k_1 \dots k_n) \end{aligned}$$

donde  $Q$  es una matriz semidefinida positiva.

La función objetivo del problema anterior es explícitamente cuasiconvexa en  $\mathbb{R}^n$ , puesto que, al ser  $Q$  semidefinida positiva  $f(x) = cx + \frac{1}{2} xQx$  es convexa (Béla Martos (1975), capítulo 9). Por tanto el cono de crecimiento  $\Lambda(f,I)$  es convexo según la proposición 4.2.4.1, en este caso, se probará además que se trata de un cono poliédrico.

##### 4.2.4.2.1 Teorema.

Sea  $f(x) = cx + \frac{1}{2} xQx$ , entonces se verifica la siguiente igualdad de conjuntos:

$$\Lambda(f,I) = \{ u \in \mathbb{R}^n / k_i e^i Q u \geq -cu, \quad i=1 \dots n \}.$$

ESQUEMAS DE ENUMERACION IMPLICITA ORDENADA:  
PROGRAMACION ENTERA.

**Demostración.**

Dado un vector  $u \in \mathbb{R}^n$ ,  $u \in \Lambda(f, x^0)$ ,  $x^0 \in I \Leftrightarrow \frac{d}{dt} f(x^0 + tu) \geq 0$   
 $\forall t \geq 0 \Leftrightarrow cu + x^0 Q u + t u Q u \geq 0 \forall t \geq 0 \Leftrightarrow cu + x^0 Q u \geq 0$ .

Como  $x^0 \in I$ ,  $x^0 = \sum_{i=1}^n \lambda_i k_i e^i$ ,  $\sum_{j=1}^n \lambda_j \leq 1$ ,  $\lambda_i \geq 0$ , por lo tanto

$$u \in \Lambda(f, I) \Leftrightarrow cu + \sum_{i=1}^n \lambda_i k_i e^i Q u \geq 0, \quad \sum_{j=1}^n \lambda_j \leq 1, \quad \lambda_i \geq 0 \quad i=1 \dots n \Leftrightarrow$$

$$\Leftrightarrow k_i e^i Q u \geq -cu \quad \forall i=1 \dots n. \quad \blacksquare$$

Si denotamos por  $\text{diag}(k_1 \dots k_n)$  la matriz diagonal con elementos  $k_1 \dots k_n$ , el cono de crecimiento de  $f$  en  $I$  viene dado por

$$Eu \geq 0$$

donde  $E = \text{diag}(k_1 \dots k_n) Q + C$ , siendo  $c_{ij} = c_i$ ,  $i, j=1 \dots n$ . Si  $E$  es invertible es fácil calcular sus direcciones extremas que coinciden con las columnas de la matriz inversa, ya que

$$u \in \Lambda(f, I) \Leftrightarrow Eu = \lambda, \quad \lambda \in \mathbb{R}_+^n \Leftrightarrow u = E^{-1} \lambda, \quad \lambda \in \mathbb{R}_+^n.$$

Denotemos  $\Lambda_c = \Lambda(f, I)$  si  $0 + \Lambda_c \supseteq \{ x \in I(k_1 \dots k_n) / g_i(x) \leq b_i \quad \forall i \}$  entonces podemos aplicar directamente el algoritmo PG, en otro caso realizaremos un cambio de coordenadas de forma que se satisfaga la condición anterior. En el cambio de coordenadas se traslada el origen a un punto  $x^0$ . Para que se verifique la condición de inclusión, es suficiente que  $E(k_i e^i - x^0) \geq 0 \quad \forall i$ , es decir  $k_i E e^i \geq E x^0 \quad \forall i$ . Tomando  $x^0 = -k E^{-1} \mathbf{1}$  donde  $k$  es el menor escalar positivo que verifica  $x^0 \in Z^n \wedge k_i E e^i \geq -k \mathbf{1} \quad \forall i$  se verifica la condición de inclusión.

ESQUEMAS DE ENUMERACION IMPLICITA ORDENADA:  
PROGRAMACION ENTERA.

Haciendo el cambio de orígenes de coordenadas se pueden reorientar los ejes del sistema de referencia de forma que obtengamos un problema equivalente a PG con el mismo formato que éste

$$\begin{aligned} \min \bar{f}(\bar{x}) &= \bar{c}\bar{x} + \frac{1}{2} \bar{x} \bar{Q} \bar{x} \\ \text{sa: } g_i(\bar{x}) &\leq \bar{b}_i \quad i=1 \dots m \\ \bar{x} &\in \bar{I}(k_1 \dots k_n) \end{aligned}$$

de forma que  $\{\bar{x} \in \bar{I}(k_1 \dots k_n), g_i(\bar{x}) \leq \bar{b}_i, i=1 \dots m\} \subseteq 0 + \bar{\Lambda}_c$ , con  $\bar{\Lambda}_c$  y  $\bar{I}(k_1 \dots k_n)$ , los transformados del cono de crecimiento y del intervalo entero  $I(k_1 \dots k_n)$  según el cambio de coordenadas.

En esta situación la función objetivo no tiene porqué ser creciente respecto a  $\preceq_{\bar{\Lambda}_c}$ , en concreto, por construcción, sólo se puede asegurar  $\bar{x} \preceq_{\bar{\Lambda}_c} \bar{y} \Rightarrow \bar{f}(\bar{x}) \leq \bar{f}(\bar{y})$  cuando  $\bar{x}, \bar{y}$  están en el conjunto  $\bar{I}(k_1 \dots k_n)$ . A continuación se da una condición suficiente bajo la cual se puede asegurar que  $\bar{f}$  es creciente respecto a  $\preceq_{\bar{\Lambda}_c}$ .

**4.2.4.2.2 Proposición.**

Si se verifica  $cd^i + x^0 Q d^i \geq 0 \wedge d^i Q d^j \geq 0 \forall i, j=1 \dots$ , entonces  $\forall \bar{x}, \bar{y} \in 0 + \bar{\Lambda}_c, \bar{x} \preceq_{\bar{\Lambda}_c} \bar{y} \Rightarrow \bar{f}(\bar{x}) \leq \bar{f}(\bar{y})$ .

ESQUEMAS DE ENUMERACION IMPLICITA ORDENADA:  
PROGRAMACION ENTERA.

**Demostración.**

Sean  $x$ ,  $y$  las coordenadas de  $\bar{x}$ ,  $\bar{y} \in 0+\bar{\Lambda}_c$  respecto del sistema de referencia original, entonces  $x = x^0+d$  con  $d \in \Lambda_c$  es decir,  $d = \sum_1^n \lambda_j d^j$ ,  $\lambda_j \geq 0$  por otro lado según la definición del orden parcial  $\leq_{\Lambda_c}$  (sección 2.3), si  $\bar{x} \leq_{\Lambda_c} \bar{y} \Rightarrow y = x+d^*$ ,  $d^* = \sum_1^n \mu_j d^j$ ,  $\mu_j \geq 0$ . Con lo cual para que se verifique la desigualdad

$$\bar{f}(\bar{x}) = f(x) \leq f(y) = \bar{f}(\bar{y})$$

es suficiente que  $d^* \in \Lambda(f, x) \Leftrightarrow cd+xQd \geq 0 \Leftrightarrow \sum_1^n \mu_j (cd^j+xQd^j) \geq 0$   
 $\Leftrightarrow cd^j+xQd^j \geq 0 \forall j \Leftrightarrow cd^j+x^0Qd^j+dQd^j \geq 0 \forall j \forall d \in \Lambda_c \Leftrightarrow cd^j+x^0Qd^j \geq 0$   
 $\wedge d^1Qd^j \geq 0 \forall i, j. \blacksquare$

Supuesto que se verifican las condiciones de la proposición anterior podremos aplicar el algoritmo PG, en otro caso es necesario modificar las actualizaciones en las iteraciones como se describe a continuación:

*Iteración k=1, 2, 3...*

Seleccionar

$$x_k = \operatorname{argmin}_x \{ f(x) / x \in X(k-1) \wedge x \in I(k_1 \dots k_n) \setminus X(k-1) \}$$

Si  $x_k$  es factible suprimir de  $X(k-1)$  el conjunto

$$\{ x \in I(k_1 \dots k_n) / f(x) \geq f(x_k) \}.$$

En otro caso hacer  $X(k) = X(k-1) \cup \{x_k\}$ ,  $k=k+1$ , ir a la

iteración  $k$  hasta que no puedan ser seleccionados más vectores.

4.2.4.2.3 Ejemplo.

Consideramos el problema de localizar un centro en un punto con coordenadas enteras, que pertenece al conjunto factible

$$\{ x \in I(k_1 \dots k_n) / g_i(x) \leq b_i \quad \forall i \}$$

Si buscamos la localización que diste menos en norma euclídea del punto de coordenadas  $a \in \mathbb{R}^n \setminus I$ , podemos formular el problema de optimización como sigue:

$$\begin{aligned} \min \quad & \|x-a\|_2^2 \\ \text{sa:} \quad & g_i(x) \leq b_i \quad i=1 \dots m \\ & x \in I(k_1 \dots k_n) \end{aligned}$$

o equivalentemente

$$\begin{aligned} \min \quad & f(x) = -ax + \frac{1}{2} xIx \\ \text{sa:} \quad & g_i(x) \leq b_i \quad i=1 \dots m \\ & x \in I(k_1 \dots k_n). \end{aligned}$$

En esta situación el cono de crecimiento  $\Lambda(f, I)$  viene dado por la siguiente expresión:

$$\begin{bmatrix} k_1 - a_1 & -a_2 & \dots & -a_n \\ -a_1 & k_2 - a_2 & \dots & -a_n \\ & & \dots & \\ -a_1 & -a_2 & \dots & k_n - a_n \end{bmatrix} u \geq 0$$

El determinante de esta matriz es

$$-\prod_1^n k_i \left( \sum_1^n \frac{a_j}{k_j} - 1 \right)$$

por tanto es invertible siempre que  $a$  no pertenezca al hiperplano determinado por los puntos  $\{k_i e^i, i=1 \dots n\}$

En este supuesto las direcciones extremas del cono de crecimiento son

ESQUEMAS DE ENUMERACION IMPLICITA ORDENADA:  
PROGRAMACION ENTERA.

$$d_i^j = r \frac{a_j}{k_i k_j} \quad i \neq j, \quad d_j^j = \frac{1}{k_j} \left( 1 + r \frac{a_j}{k_j} \right) \text{ con } r = -1 / \left( \sum_1^n \frac{a_1}{k_1} - 1 \right).$$

En primer lugar, supongamos que  $a_j \leq 0 \forall j$ , esta situación se puede conseguir siempre que  $a_j \leq 0$  ó  $a_j \geq k_j \forall j$  mediante el cambio de variables  $y_j = k_j - x_j \forall j / a_j \geq k_j$ . En estas condiciones se observa que  $d_i^j \leq 0 \forall j \neq i$  ya que al ser  $a_j \leq 0 \Rightarrow r \geq 1$ . Por otro lado

$$g_i = \sum_1^n d_i^j = r/k_i \quad \forall i$$

con lo cual  $g \in \Lambda(f, I)$  y además por ser un cono convexo

$$d^j - \frac{a_j}{k_j} g = \frac{1}{k_j} e^j \in \Lambda(f, I).$$

Por tanto en esta situación podemos tomar  $\Lambda_c = \langle\langle e^1 \dots e^n \rangle\rangle$  que está en las condiciones bajo las cuales el teorema 4.2.3 asegura la exactitud del algoritmo 4.2.2.

Supongamos ahora que  $\exists j / a_j \in (0, k_j)$  entonces tomamos como nuevo origen de coordenadas

$$x^0 = -k \begin{bmatrix} t \\ r/k_1 \\ \vdots \\ r/k_n \end{bmatrix}$$

tal que  $k = -\min\{ -k_i a_i, k_i(k_i - a_i) \quad i=1 \dots n \}$  y  $t$  es el menor entero positivo que hace que  $x^0 \in \mathbb{Z}^n$ . Con esta elección de  $x^0$  operamos como se ha indicado en la sección anterior.

#### 4.3 PROBLEMA MULTIOBJETIVO ENTERO.

En esta sección se describe un procedimiento de resolución del problema lineal entero múltiple, esto es, aquél cuyo objetivo se compone de un conjunto finito de funciones lineales y cuyo conjunto factible está determinado por los vectores con componentes enteras contenidos en un politopo.

Gran parte de los problemas reales admiten una formulación como la que se acaba de describir, lo que hace que hayan surgido gran cantidad de técnicas multiobjetivo en la literatura (Roy (1971), Ecker y Gal y Nedoma (1972), Evans y Steuer (1973), Konada (1975), Yu y Zeleny (1975), Zionts (1977), Klein y Hannan (1982), Korhonen et al. (1984), Ramesh et al. (1989)).

El problema ha sido tratado fundamentalmente desde dos puntos de vista; la obtención de soluciones que maximizan la función de utilidad del decisor y la optimización vectorial.

La primera de estas interpretaciones asume implícitamente el conocimiento de la función de utilidad del decisor. Los métodos originados interactúan con el decisor que debe indicar su preferencia entre pares de alternativas presentadas. A través de esta información y, suponiendo que la función de utilidad verifica propiedades especiales, se construye la mayoría de los algoritmos en esta metodología (Ramesh et al. (1986), (1989)).

La segunda interpretación del problema conduce a la construcción de algoritmos de determinación del conjunto de puntos eficientes. Para este problema no son válidas en general, las

ESQUEMAS DE ENUMERACION IMPLICITA ORDENADA:  
PROGRAMACION ENTERA.

técnicas existentes en Programación Múltiple con variables continuas (Zionts (1977)) por lo cual, como proponen Klein y Hannan (1982), se requiere la construcción de algoritmos que desarrollen una enumeración implícita en el conjunto de soluciones factibles. En esta sección se estudiará un algoritmo de este tipo que usa la *debo* definida por el orden c-léxico generalizado en la resolución del problema que se formula a continuación:

$$\begin{aligned} \min Cx \\ \text{sa: } Ax \leq b \\ x \in I(k_1 \dots k_n) \end{aligned} \quad (\text{PLEM})$$

donde  $C \in \mathbb{M}_{p \times n}$ ,  $\text{rango}(C) = p$ ,  $A \in \mathbb{M}_{m \times n}$ ,  $b \in \mathbb{Z}^m$ ,  $a_{ij}, c_{ij} \in \mathbb{Z} \forall ij$ .

Se dice que un punto  $x$  es eficiente para el problema PLEM si es factible y no existe  $y \in I(k_1 \dots k_n)$  factible tal que

$$Cy \leq Cx$$

componente a componente, con alguna desigualdad estricta, en caso contrario se dice que  $x$  es dominado.

En primer lugar de la condición de dominación se puede deducir que  $\forall y$  factible, pueden ser suprimidos como objeto del estudio aquellos vectores  $x$  del conjunto

$$y + \Lambda \cap \{ x \in I(k_1 \dots k_n) / Ax \leq b \}$$

donde  $\Lambda = \{ t \in \mathbb{R}^n / Ct \geq 0 \wedge Ct \neq 0 \}$ . Esto es evidente, ya que para cada  $x$  en el conjunto anterior se tiene  $x = y + t \Rightarrow t = x - y \wedge C(x - y) \geq 0$ ,  $C(x - y) \neq 0$  con lo que se verifica la condición de dominación del punto  $x$ .



ESQUEMAS DE ENUMERACION IMPLICITA ORDENADA:  
PROGRAMACION ENTERA.

Al conjunto  $\Lambda$  lo denominamos como de dominación y es interesante hacer notar que no depende de  $y$ . El esquema de enumeración implícita hace uso de esta idea, que se refleja en el próximo teorema.

4.3.1 Teorema.

Supongamos que las columnas de  $C$  están ordenadas de tal forma que  $C = [C_B | C_N]$  con  $\det(C_B) \neq 0$ , entonces dados  $x^1, x^2 \in \mathbb{Z}^n$  tal que

$$x^2 = x^1 + \left[ \begin{array}{c|c} C_B^{-1} & -C_B^{-1}C_N \\ \hline 0 & I_{n-p} \end{array} \right] \lambda$$

con  $\lambda \in \mathbb{R}_+^n / \exists i \in \{1 \dots p\} / \lambda_i > 0$ , entonces  $x^2$  es un punto dominado.

Demostración.

Sea  $x = x^2 - x^1 = \begin{bmatrix} x_B \\ x_N \end{bmatrix}$ , según la igualdad anterior

$$x_B = C_B^{-1} \lambda_B - C_B^{-1} C_N \lambda_N$$

$$x_N = \lambda_N$$

por tanto

$$x_B = C_B^{-1} \lambda_B - C_B^{-1} C_N x_N$$

$$Cx = C_B x_B + C_N x_N = \lambda \geq 0, \lambda_B \neq 0$$

con lo cual  $Cx^2 \geq Cx^1$  componente a componente con al menos una desigualdad estricta. ■

ESQUEMAS DE ENUMERACION IMPLICITA ORDENADA:  
PROGRAMACION ENTERA.

Sean  $d^1 \dots d^n$  las  $n$  columnas de la matriz  $\left[ \begin{array}{c|c} C_B^{-1} & -C_B^{-1} C_N \\ \hline 0 & I_{n-p} \end{array} \right]$ ,

notemos en primer lugar que son linealmente independientes por ser el determinante de la matriz igual a  $\det(C_B^{-1}) \neq 0$  además según el teorema anterior el cono agudo generado por estas columnas está contenido en el cono de dominación, de hecho se  $p=n$  ambos coinciden. Sea  $\Lambda_c = \langle\langle d^1 \dots d^n \rangle\rangle$  este cono, según se indicó anteriormente dado  $y$ , punto factible, podemos suprimir aquellos puntos  $y + \{t \in \Lambda_c, t_i > 0 \ i \in \{1 \dots p\}\}$ . Por tanto si definimos el orden  $c$ -léxico generalizado asociado a  $\Lambda_c$  podremos suprimir aquellos sucesores directos  $x$  de un vector  $x^-$  factible tales que  $x - x^- = \sum_1^n (\mu(x)_i - \mu(x^-)_i) d^i = d^j$  para algún  $j \in \{1 \dots p\}$ , donde  $\mu(x)^-$  es el antecesor directo de  $\mu(x)$  para el orden  $c$ -léxico. Con esta supresión, no serán generados ninguno de los sucesores de  $x$  en el orden  $c$ -léxico generalizado. Así queda determinada la operación de supresión del algoritmo de enumeración implícita que describiremos. Antes vamos a suponer que se verifica

$$0 + \Lambda_c \supseteq \{x \in I(k_1 \dots k_n) / Ax \leq b\}.$$

Esta hipótesis es necesaria ya que el algoritmo desarrolla la búsqueda en los sucesores del 0 según el orden  $\leq_{\Lambda_c}$ , esto es  $0 + \Lambda_c \cap \mathbb{Z}^n$ , no obstante, no supone ninguna restricción ya que  $\Lambda_c$  está generado por  $n$  vectores independientes con lo cual siempre se puede realizar un cambio de variables análogo al indicado en la

sección 4.2 de forma que se verifique la condición. Además en esta situación el cono de dominación no depende más que de la matriz C, por lo que una vez hecho el cambio de variables, todo el razonamiento sigue siendo válido aplicado al cono transformado de  $\Lambda_c$ .

#### 4.3.2 Algoritmo de generación del conjunto de puntos

eficientes de PLEM.

*Inicialización.*

Hacer  $X(0) = O_{\Lambda}$  donde  $O_{\Lambda}$  es el conjunto de orígenes múltiples del orden  $\leq \Lambda_c$  con  $\Lambda_c$  el cono agudo generado por las columnas de la

matriz 
$$\left[ \begin{array}{c|c} C_B^{-1} & -C_B^{-1}C_N \\ \hline 0 & I_{n-p} \end{array} \right].$$

Verificar si alguno de los puntos de  $X(0)$  es factible, en tal caso seleccionar  $x \in X(0)$  que alcanza en  $Cx$  el mínimo lexicográfico, almacenarlo en el conjunto  $E_f$ , suprimir aquellos elementos de  $X(0)$  dominados, si queda algún otro vector factible operar del mismo modo, en otro caso hacer  $k=1$ , ir a la iteración  $k$ .

*Iteración  $k=1, 2, 3, \dots$*

Elegir  $x(k)$  el vector de  $I(k_1 \dots k_n) \setminus X(k-1)$  tal que  $x(k)^- \in X(k-1)$  con  $x(k) - x(k)^- \neq d^j$   $j \in \{1 \dots p\}$  cuando  $x(k)^-$  sea factible y que además obtiene el mínimo lexicográfico del vector  $Cx(k)$ , de entre todos aquellos que están en las mismas condiciones.

Si  $x(k)$  es factible y no es dominado por un punto de  $E_f$  hacer  $E_f = E_f \cup \{x_k\}$ . Suprimir de  $X(k-1)$  aquellos puntos dominados por  $x(k)$ .

En cualquier caso hacer  $X(k) = X(k-1) \cup \{x(k)\}$ ,  $k=k+1$  y volver a iterar hasta que no sea posible ninguna selección.

#### 4.3.3 Teorema.

El algoritmo anterior finaliza en un número finito de iteraciones con el conjunto  $E_f$  de puntos eficientes de PLEM.

#### Demostración.

En primer lugar, si se elimina la operación de supresión el esquema anterior enumera el conjunto  $0 + \Lambda_c \cap I(k_1 \dots k_n)$ , con lo cual está garantizada la finitud ya que no se producen duplicaciones de los puntos generados. Por otro lado la operación de supresión está garantizada por el teorema 2.3.1, por tanto como

$$\{x \in I(k_1 \dots k_n) / Ax \leq b\} \subseteq 0 + \Lambda_c$$

se tiene garantizado que el conjunto de puntos eficientes está contenido en  $E_f$ . Por último, supongamos que existe  $x(k) \in E_f$  dominado por el punto eficiente  $x(1) \in E_f$ . Entonces  $Cx(1) \leq Cx(k)$  (distintos) con lo cual, todo antecesor  $x$  de  $x(1)$  en el orden  $c$ -léxico generalizado verifica  $Cx \leq Cx(k)$  (distintos). Por tanto, por la operación de selección del algoritmo 4.3.2 se tiene  $1 < k$ , por lo que  $x(k)$  es suprimido en la iteración  $k$  pues  $x(1) \in E_f$  en esta iteración. ■

## CAPITULO 5.

### ALGORITMOS NO EXACTOS.

#### 5.1 INTRODUCCION.

En este capítulo se desarrollan algunos algoritmos heurísticos para problemas de Programación Lineal Entera. Esto es algo plenamente justificable puesto que estos problemas son, como ya se ha comentado en capítulos anteriores, de la clase  $NP$ -completos. Aunque existen casos muy concretos con complejidad polinomial como aquellos en los que el conjunto factible tiene vértices enteros (por ejemplo los determinados por matrices totalmente unimodulares o matrices equilibradas (Schrijver (1986))), lo cierto es que, la necesidad de algoritmos aproximados, ha hecho que existan en la literatura gran cantidad de esquemas heurísticos que se han desarrollado paralelamente en el tiempo con los algoritmos exactos. Como ejemplo podemos citar a los siguientes autores Senju y Toyoda (1968), Faaland y Hillier (1979), Dobson (1982), Echols et al. (1986), Fox y Scudder (1986), Vassilev y Genova (1991).

La operación de selección de puntos realizada en los algoritmos que se describirán se basa en la dirección de búsqueda

ESQUEMAS DE ENUMERACION IMPLICITA ORDENADA:  
PROGRAMACION ENTERA.

generada por el orden c-léxico, no obstante el esquema seguido difiere del propuesto en el capítulo 1 ya que en esta situación, la operación de supresión de elementos no tiene sentido pues en general, se carece de propiedades que garanticen la no optimalidad de puntos.

Por otro lado la regla de selección utilizada es la que denominamos de "mínimo incremento", esto es, de entre los posibles puntos seleccionables elegimos aquél que produzca un mínimo incremento respecto del valor objetivo de su antecesor directo. Esta regla da lugar a lo que se conoce en la literatura con el nombre de selección greedy que podríamos traducir como, hambrienta y que hace referencia al hecho de elegir la mejor posibilidad local, sin tener en cuenta que puede conducir a empeoramientos posteriores. Precisamente este debilitamiento en la regla de selección es el que conduce a métodos más rápidos y con menos necesidades de almacenamiento de información.

Al modificar la regla de selección  $\phi(X(k))$  se pierde la optimalidad garantizada por el teorema 1.3.1, por tanto es interesante dar una medida de la "proximidad" a la solución óptima. Esto se lleva a cabo por medio del intervalo error, que contiene el valor óptimo del objetivo y que depende funcionalmente del valor de la solución heurística aportada por el algoritmo. Para determinarlo impondremos al test de salida algunas condiciones adicionales al requerimiento de factibilidad de los algoritmos exactos, con ello garantizamos ciertas propiedades

ESQUEMAS DE ENUMERACION IMPLICITA ORDENADA:  
PROGRAMACION ENTERA.

sobre la cadena  $Z_x^h$  determinada por la solución heurística  $x^h$ , que permiten construir una cota inferior del intervalo error alternativa al objetivo óptimo del problema continuo.

Para aprovechar estas propiedades se hace uso de algunas herramientas propias de la Optimización Combinatoria que aparecen en la siguiente sección a modo de resumen.

## 5.2 ALGUNOS RESULTADOS DE OPTIMIZACION COMBINATORIA.

En la introducción se ha comentado la forma de proceder de los algoritmos greedy. Estos esquemas, generalmente, producen soluciones aproximadas no obstante, bajo ciertas hipótesis estructurales del conjunto factible del problema de optimización, se garantiza la optimalidad de la solución propuesta. La estructura a la que nos referimos se denomina matroide y son ampliamente conocidas y estudiadas en Teoría de Grafos desde que Witney las introdujera en 1935.

### 5.2.1 Definición.

Un matroide sobre un conjunto  $G$  finito, es un par  $(G, \mathfrak{F})$  que verifica:

(1)  $\emptyset \in \mathfrak{F}$ .

(2)  $H_1 \subseteq H_2 \in \mathfrak{F} \Rightarrow H_1 \in \mathfrak{F}$ .

(3)  $\forall H \subseteq G$ , si  $H_1$  y  $H_2 \subseteq H$  y son maximales en  $\mathfrak{F}$  respecto de la inclusión de conjuntos, entonces tienen el mismo cardinal.  $\square$

Consideremos el siguiente problema:

$$\begin{aligned} \max \sum_{g \in C} f(g) \\ \text{sa: } C \in \mathfrak{F}. \end{aligned} \quad (\text{CIMP})$$

donde  $f: G \rightarrow \mathbb{R}_+$ . A  $f()$  la denominamos función de peso y el problema de optimización se conoce con el nombre de Conjunto Independiente de Maximo peso (CIMP). El próximo resultado garantiza la optimalidad de la solución obtenida por el siguiente esquema greedy:

### 5.2.2 Algoritmo greedy para un problema CIMP.

*Inicialización.*

Tomar  $C_g = \{ g^0 \}$ , siendo  $g^0$  el elemento con mayor peso de  $G$ .

Hacer  $k=1$ , ir a la iteración  $k$ .

*Iteración  $k=1, 2, 3, \dots$*

Seleccionar  $g^k = \operatorname{argmax} \{ f(g) / g \in G \setminus C_g \wedge C_g \cup \{g\} \in \mathfrak{F} \}$ .

Hacer  $C_g := C_g \cup \{g^k\}$ ,  $k := k+1$ , ir a la iteración  $k$  hasta que no puedan ser seleccionados más elementos de  $G$ .

### 5.2.3 Teorema. (Parker y Rardin (1988))

Sea  $f: G \rightarrow \mathbb{R}_+$  una función de peso, entonces se verifica que el algoritmo greedy 5.2.2 determina la solución óptima de CIMP sii  $(G, \mathfrak{F})$  es un matroide.

Aunque la estructura del problema CIMP se observa con frecuencia en problemas de Optimización Combinatoria, para un



ESQUEMAS DE ENUMERACION IMPLICITA ORDENADA:  
PROGRAMACION ENTERA.

problema de Programación Entera general, es difícil verificar las condiciones exigidas por el teorema 5.2.3. Supongamos que deseamos resolver el siguiente problema

$$\begin{aligned} & \max h(x) \\ \text{sa: } & x \in \mathcal{D} \subset \mathbb{Z}_+^n. \end{aligned}$$

Podríamos formularlo como un problema CIMP sin más que tomar la familia  $\mathfrak{J} \subset \mathcal{P}(\mathcal{D})$  formada por los conjuntos unitarios junto con el vacío. En este caso se puede verificar que  $(\mathcal{D}, \mathfrak{J})$  es un matroide pero el algoritmo Greedy (CIMP) nos lleva a una enumeración total del conjunto factible. Para evitar esto debemos introducir en  $\mathfrak{J}$  conjuntos no unitarios que ayuden a resolver el problema sin necesidad de enumerar totalmente el conjunto.

Desgraciadamente no es fácil introducir conjuntos no unitarios en  $\mathfrak{J}$  sin violar alguna de las condiciones que definen una estructura matroide. Por esta razón existen en la literatura algunos resultados que dan una medida del error cometido por el algoritmo greedy 5.2.2 cuando trabaja sobre estructuras más pobres. Una de estas estructuras es la que se denomina Sistema Independiente que es una familia  $\mathfrak{J}_I$  de subconjuntos de  $G$  verificando las dos primera propiedades de la definición 5.2.1.

Dado un sistema independiente  $(G, \mathfrak{J}_I)$  y una función de peso  $f: G \rightarrow \mathbb{R}_+$  se conoce con el nombre de Conjunto Independiente

ESQUEMAS DE ENUMERACION IMPLICITA ORDENADA:  
PROGRAMACION ENTERA.

Maximal de Mínimo Peso a la solución del siguiente problema de optimización:

$$\min_{g \in C} \sum f(g) \quad (\text{CIMMP})$$

sa:  $C \in \mathfrak{J}_I$ ,  $C$  maximal en  $\mathfrak{J}_I$  respecto a la inclusión.

En el próximo resultado se garantizan cotas para el valor objetivo óptimo del problema anterior cuando se emplea para su resolución el siguiente esquema greedy:

#### 5.2.4 Algoritmo greedy para un problema CIMMP.

*Inicialización.*

Tomar  $C_g = \{g^0\}$ , siendo  $g^0$  el elemento con menor peso de  $G$ .

Hacer  $k=1$ , ir a la iteración  $k$ .

*Iteración  $k=1, 2, 3, \dots$*

Seleccionar  $g^k = \operatorname{argmin} \{ f(g) / g \in G \setminus C_g \wedge C_g \cup \{g\} \in \mathfrak{J} \}$ .

Hacer  $C_g := C_g \cup \{g^k\}$ ,  $k := k+1$ , ir a la iteración  $k$  hasta que

no puedan ser seleccionados más elementos de  $G$ .

### 5.2.5 Teorema. (Parker y Rardin (1988))

Sea  $(G, \mathfrak{I}_I)$  un Sistema Independiente, y supongamos que todo miembro maximal de  $\mathfrak{I}_I$  tiene cardinalidad  $a$ . Sea  $C^*$  el óptimo de CIMMP,  $C_g$  el conjunto propuesto por el algoritmo greedy 5.2.4 entonces se verifica la siguiente desigualdad:

$$\sum_{g \in C_g} f(g) \leq \frac{1}{d} \sum_{g \in C^*} f(g) + \left(1 - \frac{1}{d}\right) a f_{\max},$$

donde  $f_{\max}$  es el valor máximo alcanzado por  $f()$  en un punto contenido en un conjunto independiente y donde  $d$  viene dado por la siguiente expresión:

$$d = \max\{ur(S)/lr(S), S \in \mathfrak{I}_I\}.$$

$$lr(S) = \min\{|C| : C \text{ es un subconjunto maximal de } S\}$$

$$ur(S) = \max\{|C| : C \text{ es un subconjunto maximal de } S\}$$

Observemos que para estructuras matroides se tiene  $d=1$  y por tanto el resultado garantiza la obtención del óptimo mediante el algoritmo greedy 5.2.4.

### 5.3 FORMULACION DE PEG COMO UN CIMMP.

En este apartado se determina un problema equivalente a

$$\min h(x) \quad (\text{PEG})$$

$$\text{sa: } x \in \mathcal{D} \subset \mathbb{Z}_+^n.$$

que tiene la estructura del problema del conjunto independiente maximal de mínimo peso. El sistema independiente obtenido en este

ESQUEMAS DE ENUMERACION IMPLICITA ORDENADA:  
PROGRAMACION ENTERA.

problema no verifica la condición de igual cardinal de cadenas maximales, por tanto la cota del teorema 5.2.5 no es válida. No obstante, como se verá en la sección 5.4.1, es posible extender el resultado, obteniéndose así una cota más general, aplicable al problema CIMMP construido.

Supongamos, en primer lugar, que la función objetivo del problema PEG es creciente en las componentes de su argumento, es decir si  $x \leq y$  componente a componente, entonces  $h(x) \leq h(y)$ . Además trataremos el caso en que  $\mathcal{D} \subseteq I(k_1 \dots k_n)$  y el vector nulo no es factible, en otro caso éste sería la solución óptima.

Consideramos la restricción del gredoide local dado por el c-léxico definido en la sección 1.3

$$(I(k_1, k_2 \dots k_n) \setminus \mathcal{D}', \mathfrak{J} \setminus \mathcal{D}')$$

donde  $\mathcal{D}' \equiv \{x \in \mathcal{D} \wedge \exists y \in \mathcal{D}, y \leq_c x\}$ , y  $\mathfrak{J}$  el gredoide local parcialmente ordenado asociado a  $\leq_c$  en  $I(k_1 \dots k_n)$ . En esta situación  $I(k_1, k_2 \dots k_n) \setminus \mathcal{D}' \neq \emptyset$  pues contiene al vector nulo. A continuación se determina el sistema independiente que se usará en el problema CIMMP equivalente a PEG.

### 5.3.1 Definición.

Denominamos  $\mathfrak{J}_I$  a la familia de subconjuntos de  $I(k_1, k_2 \dots k_n) \setminus \mathcal{D}'$  dada por la siguiente expresión:

$$\mathfrak{J}_I = \{ A \subseteq I(k_1, k_2 \dots k_n) \setminus \mathcal{D}' \mid \exists x \in \mathcal{D}, A \subseteq Z_x \}$$

donde por  $Z_x$  se representa la cadena con elemento terminal  $x$  del gredoide  $(I(k_1, k_2 \dots k_n) \setminus \mathcal{D}', \mathfrak{J} \setminus \mathcal{D}')$ .

ESQUEMAS DE ENUMERACION IMPLICITA ORDENADA:  
PROGRAMACION ENTERA.

Como se deduce de la definición anterior,  $\mathfrak{J}_I$  determina un sistema independiente.

Por último construimos la función de peso del problema CIMMP que llamaremos función incremento de  $h()$ .

5.3.2 Definición.

Sea  $h: I(k_1 \dots k_n) \rightarrow \mathbb{R}$ , creciente en componentes, se denomina función incremento de  $h()$  respecto a la familia  $\mathfrak{J}_I$  a la función  $f()$  determinada como sigue:

$$f(x) = \begin{cases} h(x) - h(x^-) & \text{si } x \neq 0 \wedge \exists A \in \mathfrak{J}_I / x \in A. \\ h(0) & \text{si } x = 0. \\ +\infty & \text{en otro caso.} \end{cases}$$

5.3.3 Proposición.

El problema PEG con  $h()$  creciente en componentes y  $\mathfrak{D} \subseteq I(k_1, k_2 \dots k_n)$  es equivalente al problema CIMMP

$$\min \sum_{g \in C} f(g)$$

sa:  $C \in \mathfrak{J}_I$ ,  $C$  maximal en  $\mathfrak{J}_I$  respecto a la inclusión.

donde  $\mathfrak{J}_I$  y  $f()$  son los definidos en 5.3.1 y 5.3.2.

Demostración.

En primer lugar estudiaremos los conjuntos maximales del sistema independiente determinado por  $\mathfrak{J}_I$ :

Sea  $C \in \mathfrak{J}_I$  maximal, por definición de  $\mathfrak{J}_I$ ,  $\exists x \in C$  máximo para  $\leq_c$ , por ser  $C$  maximal  $Z_x \subseteq C$ , donde  $Z_x$  es el camino con punto

ESQUEMAS DE ENUMERACION IMPLICITA ORDENADA:  
PROGRAMACION ENTERA.

terminal que no puede tener sucesores directos en  $I(k_1, k_2 \dots k_n) \setminus \mathcal{D}'$ .  
Según el teorema de caracterización de cadenas (1.2.4.5)  $Z_x$   
contiene todos los vectores de  $I(k_1, k_2 \dots k_n)$  menores que  $x$  para  $\leq_c$ ,  
luego  $Z_x \equiv C$ . Además sabemos que el primer elemento de  $Z_x$  es el 0  
y que para dos elementos consecutivos en la secuencia ordenada  $Z_x$ ,  
el primero es antecesor directo del segundo, por tanto

$$\sum_{y \in C} f(y) = \sum_{x \in Z_x \setminus \{0\}} (h(y) - h(y^-)) + h(0) = h(x).$$

Para finalizar, basta probar que la solución óptima de PEG se  
alcanza en aquellos vectores  $x$  factibles tales que  $Z_x \in \mathcal{F} \setminus \mathcal{D}'$ , en  
otro caso

$$Z_x \cap \mathcal{D}' \neq \emptyset \Rightarrow \exists y \leq_c x, x \neq y / y \in Z_x \Rightarrow h(y) \leq h(x).$$

Por tanto los problemas PEG y CIMMP son equivalentes. ■

Según el resultado anterior, para cualquier función objetivo  
 $h()$  creciente en componentes y cualquier conjunto factible acotado  
 $\mathcal{D}$ , se puede obtener el problema equivalente CIMMP. No obstante,  
como ya se ha dicho, el sistema independiente  $(I(k_1 \dots k_n), \mathcal{F}_1)$  no  
verifica, en general, las condiciones bajo las cuales el teorema  
5.2.5 garantiza una cota del error cometido por la solución  
heurística. A continuación se deduce una nueva cota. En su  
construcción se utilizan funciones objetivo lineales, además es  
necesario imponer ciertas propiedades adicionales a la solución  
generada por el algoritmo greedy 5.2.4, con lo cual, posteriormente  
debemos modificar el test de salida de este algoritmo para  
garantizar la cota del error.

#### 5.4 APLICACION A PROBLEMAS PEG CON OBJETIVOS LINEALES.

Consideramos el siguiente problema de optimización:

$$\begin{aligned} \min h(x) \\ \text{sa: } x \in \mathcal{D} \subset \mathbb{Z}_+^n. \quad (\text{PEG}) \\ 0 \leq x_i \leq k_i \quad \forall i=1 \dots n. \end{aligned}$$

donde  $h(x) = cx$  y suponemos que el problema es factible y que el vector de costos es  $c = (c_1 \dots c_n)$  con  $0 \leq c_1 \leq \dots \leq c_n$ .

Según la definición de  $f$ ,  $\forall x \in I(k_1, k_2 \dots k_n) \setminus (\mathcal{D}' \cup \{0\})$  con incremento finito  $f(x) \in \{c_1, c_2 \dots c_n\}$ , esta es la propiedad fundamental que se utilizará tanto en la construcción del intervalo error como en la deducción del test de salida del algoritmo.

##### 5.4.1 Construcción del intervalo error.

Suponemos en este apartado que el algoritmo greedy 5.2.4 al aplicarlo al problema CIMMP propuesto en 5.3.3, obtiene como solución la cadena maximal  $Z_x^h$  y sea  $Z_*$  la cadena maximal óptima.

Definimos las siguientes expresiones:

$$\begin{aligned} n_j &= |\{x \in Z_x^h / f(x) = c_j\}|, \quad j=1 \dots n. \\ \hat{f}(x) &= c_n - f(x) \geq 0 \quad \forall x. \end{aligned}$$

Sea  $\forall i=1, 2, \dots$   $G^i = \{x^1, x^2, \dots, x^i\} / x^r \neq x^j \neq 0$  si  $r \neq j \wedge f(x^j) \leq$

ESQUEMAS DE ENUMERACION IMPLICITA ORDENADA:  
PROGRAMACION ENTERA.

$$f(x^{j+1}) \leq f(x) \quad \forall x \in I(k_1 \dots k_n) \setminus (\mathcal{D}' \cup G^1 \cup \{0\}), \quad \forall j = 1 \dots (i-1).$$

Según la definición de los conjuntos  $G^1$ , se tiene  $G^1 \subset G^{i+1}$ . Por tener el conjunto  $I(k_1 \dots k_n) \setminus \mathcal{D}'$  cardinal finito, existe el mayor índice  $m \in \mathbb{N}$  para el cual  $\forall x \in G^m, f(x) \in \{c_1 \dots c_n\}$ .

Según las definiciones anteriores, por ser  $Z_*$  la solución óptima de CIMMP,  $\exists i \leq m$  tal que  $Z_* \subseteq G^i \cup \{0\}$ , puesto que  $f(0) = 0$ , podemos escribir

$$(|Z_*| - 1)c_n - \sum_{x \in Z_*} f(x) = \sum_{x \in Z_* \setminus \{0\}} \hat{f}(x) = \sum_{i=1}^m |Z_* \cap G^i| (\hat{f}(x^i) - \hat{f}(x^{i+1})). \quad (1)$$

donde  $\hat{f}(x^{m+1}) = 0$ , por convenio.

Según la definición de los conjuntos  $G^1$ , existe para todo  $k = 1 \dots (n-1)$  el primer índice  $i_k /$

$$\forall x \in G^{i_k} \Rightarrow f(x) \in \{c_1 \dots c_k\} \wedge \exists x \in G^{i_k+1} / f(x) = c_{k+1}.$$

Con esta definición de los índices  $i_k, k=1 \dots (n-1)$ .

$$\sum_{k=1}^{n-1} |Z_* \cap G^{i_k}| (\hat{f}(x^{i_k}) - \hat{f}(x^{i_k+1})). \quad (2)$$

Supongamos que se verifica la siguiente condición:

$$|Z_* \cap G^{i_k}| \neq 0 \Rightarrow |Z_g \cap G^{i_k}| \neq 0 \quad (3)$$

entonces, la expresión (2) puede escribirse de la siguiente forma



ESQUEMAS DE ENUMERACION IMPLICITA ORDENADA:  
PROGRAMACION ENTERA.

$$\sum_{\substack{k=1 \\ |Z_* \cap G^{i_k}| \neq 0}}^{n-1} \frac{|Z_* \cap G^{i_k}|}{|Z_g \cap G^{i_k}|} |Z_g \cap G^{i_k}| (\hat{f}(x^{i_k}) - \hat{f}(x^{i_{k+1}})). \quad (4)$$

Por último, según la demostración de la proposición 5.3.3,  $Z_* = Z_{x^*}$  donde  $x^*$  es un punto factible, con lo cual  $x^* \in I(k_1 \dots k_n)$  por tanto por definición de los conjuntos  $G^{i_k}$ ,

$$|Z_* \cap G^{i_k}| = \sum_{j=1}^k x_j^* \leq \sum_{j=1}^k k_j$$

Puesto que  $(\hat{f}(x^{i_k}) - \hat{f}(x^{i_{k+1}})) = c_n - c_k - c_n + c_{k+1} = c_{k+1} - c_k \geq 0$  podemos

acotar la expresión (4) superiormente por

$$\begin{aligned} & \sum_{\substack{k=1 \\ |\sum_{j=1}^k n_j| \neq 0}}^{n-1} \frac{\sum_{j=1}^k k_j}{\sum_{j=1}^k n_j} |Z_g \cap G^{i_k}| (\hat{f}(x^{i_k}) - \hat{f}(x^{i_{k+1}})) \leq \\ & \leq \max_{\substack{k=1 \dots n \\ |\sum_{j=1}^k n_j| \neq 0}} \left\{ \frac{\sum_{j=1}^k k_j}{\sum_{j=1}^k n_j} \right\} \sum_{k=1}^{n-1} |Z_g \cap G^{i_k}| (\hat{f}(x^{i_k}) - \hat{f}(x^{i_{k+1}})) = \\ & = W \sum_{x \in Z_g \setminus \{0\}} \hat{f}(x) = W ((|Z_g| - 1)c_n - \sum_{x \in Z_g} f(x)). \end{aligned}$$

donde

ESQUEMAS DE ENUMERACION IMPLICITA ORDENADA:  
PROGRAMACION ENTERA.

$$W = \max_{k=1 \dots n} \left\{ \frac{\sum_{j=1}^k k_j}{\sum_{j=1}^k n_j} \right\}$$

$$|\sum_{j=1}^k n_j| \neq 0$$

Por tanto, a partir de las desigualdades anteriores, si definimos

$\theta_{\min}$  = mínima longitud de una cadena maximal,

se puede dar el siguiente intervalo error para la solución óptima del problema CIMMP propuesto en 5.3.3.

$$\left[ (\theta_{\min} - 1)c_n + W \left( \sum_{x \in Z_g} f(x) - (|Z_g| - 1)c_n \right), \sum_{x \in Z_g} f(x) \right] \quad (\text{IE})$$

Es importante resaltar que el intervalo error IE es válido para cualquier tipo de conjunto  $\mathcal{D}$  que verifique las condiciones impuestas, no obstante debemos calcular  $\theta_{\min}$  y para ello es conveniente particularizar para distintos tipos de regiones de puntos factibles.

#### 5.4.2 Construcción del test de salida del algoritmo.

En el apartado anterior se ha construido el intervalo error que contiene al objetivo óptimo del problema CIMMP propuesto en 5.3.3. Este intervalo depende de la solución  $Z_x^h$  generada por el algoritmo greedy 5.2.4, no obstante, en la construcción se hacía uso de la expresión (3). Para que la cadena  $Z_x^h$  verifique esta condición adicional es necesario modificar el test de salida

ESQUEMAS DE ENUMERACION IMPLICITA ORDENADA:  
PROGRAMACION ENTERA.

del algoritmo como indicamos a continuación.

Sea  $Z_x^h = \{x^1=0, x^2, x^3 \dots x^h\}$ , la secuencia de vectores enteros ordenados según  $\leq_c$ . Denominamos  $c_{\min} = \min \{f(x) / x \in Z_x^h \setminus \{0\}\}$  y sea  $k = \max \{i / f(x^i) = c_{\min}\}$ . En esta situación se verifica el resultado siguiente.

5.4.2.1 Proposición.

Sea  $Z_x^h$  una cadena maximal del gredoide  $(I(k_1 \dots k_n) \setminus \mathcal{D}', \mathfrak{F} \setminus \mathcal{D}')$  verificando:

- (1)  $x^h$  es factible.
- (2) No existe  $Z_y = \{x^1 \dots x^k, y^{k+1} \dots y^p\} / f(y^r) \leq c_{\min}$ ,  $y^p$  factible con menor valor objetivo que el de  $Z_x^h$ .
- (3) No existe una cadena maximal con punto terminal factible que contenga un elemento distinto de cero con peso menor estrictamente que  $c_{\min}$ .

Entonces es válido el intervalo error IE deducido a partir de  $Z_x^h$ .

**Demostración.**

En primer lugar se probará que  $Z_x^h$ , tiene un objetivo inferior a alguna de las cadenas obtenidas por el algoritmo greedy 5.2.4. Puesto que el intervalo IE es válido para cualquier solución propuesta por este algoritmo, también lo será para  $Z_x^h$ .

Según (1)  $x^h$  es el único punto factible de la cadena  $Z_x^h$  por ser cadena maximal del gredoide  $(I(k_1 \dots k_n) \setminus \mathcal{D}', \mathfrak{F} \setminus \mathcal{D}')$ . Además (3) indica que las cadenas maximales que contienen un elemento con

ESQUEMAS DE ENUMERACION IMPLICITA ORDENADA:  
PROGRAMACION ENTERA.

peso inferior a  $c_{\min}$  son infactibles, por tanto podemos inicializar  $C_g$  en el algoritmo 5.2.4 con el vector  $x^h$ . Esto implica que la cadena  $C_g$  generada por el esquema greedy contiene a  $\{x^1 \dots x^k\}$  por lo que la condición (2) implica que el objetivo de  $C_g$  es superior o igual al de  $Z_x^h$ .

Por último se prueba que se verifica la condición (3) de la sección 5.4.1, esto es

$$|Z_* \cap G^{i^k}| \neq 0 \Rightarrow |Z_g \cap G^{i^k}| \neq 0, \forall k.$$

En caso contrario, sea  $k_0$  el primer índice  $k$  que no lo verifica, por construcción de los conjuntos  $G^i$  se deduce

$$c_{k_0} = \min \{f(x) / x \in Z_* \setminus \{0\}\} < c_{\min}$$

que contradice la hipótesis (3) de la proposición. ■

El resultado anterior sugiere un algoritmo que implemente una búsqueda en profundidad en el árbol de cadenas maximales del gredoide  $(I(k_1 \dots k_n) \setminus \mathcal{D}', \mathfrak{J} \setminus \mathcal{D}')$ . Para garantizar el test de salida dado por las tres condiciones de la proposición 5.4.2.1., en el desarrollo del algoritmo se actualiza el valor de  $c_{\min}$  así como el de  $x^k$  que es almacenado en  $x_{\max}$  en el esquema que presentamos a continuación.

### 5.4.3 Esquema del algoritmo greedy para objetivos lineales.

#### Inicialización.

1 o Elegir  $x = e^i$  siendo  $i$  el menor índice para el que  $e^i$  es un punto factible o bien  $H \neq \emptyset$  con  $e^i$  el vector con todas las componentes nulas salvo la  $i$  que vale 1 y donde el conjunto  $H$  se construye de la forma que indicamos a continuación:

$$H = \{ y / y \text{ es sucesor directo de } x \} \cap I(k_1 \dots k_n)$$

2 o Hacer  $c_{\min} = c_i$ ,  $x_{\max} = x$ ,  $F = \emptyset$ ,  $k = 1$ .

3 o Ir a la iteración  $k$ .

#### Iteración $k=1,2,3,\dots$

4 o Hacer  $x = \operatorname{argmin} \{ c'y / y \in H \}$ . Pueden ocurrir uno de los dos casos siguientes:

4.1 o  $x \in I(k_1 \dots k_n) \setminus D$  en cuyo caso actualizamos los parámetros como se indica:

4.1.1 o Si  $c'(x-x^-) \leq c_{\min}$  hacer  $x_{\max} = x$  y  $c_{\min} = c'(x-x^-)$ .

4.1.2 o  $H = \{ y / y \text{ es sucesor directo de } x \} \cap I(\omega_1 \dots \omega_n)$ .

Ir al paso 5.1 con  $t = x$ .

4.2 o Si  $x \in D$ , hacer  $F = F \cup \{x\}$ .

5 o Actualizar  $H$  de la forma que se indica a continuación:

$$H = \{ y / y \text{ es sucesor directo de } x^- / y - x^- < x - x^- \text{ (lexicográfico)} \} \cap I(k_1 \dots k_n). \text{ Hacer } t = (x^-).$$

5.1 o Si  $H = \emptyset$  pueden darse tres posibilidades:

ESQUEMAS DE ENUMERACION IMPLICITA ORDENADA:  
PROGRAMACION ENTERA.

5.1.1  $t = x_{\max} \neq 0$ .

-Si  $F \neq \emptyset$  y  $c_{\min} = c_1$  SOLUCION: tomar como  $Z_x^h$  la cadena de elementos antecesores de aquél y  $\in F$  con menor valor objetivo.

-Si  $F = \emptyset$  ó  $c_{\min} \neq c_1$  tomar  $x = x^-$ ,  $x_{\max}$  el máximo en el orden c-léxico de los puntos y, tales que  $y < x \wedge c'(y - y^-) = c_{\min}$ . Si no existiese éste tomar  $x_{\max} = 0$ . Ir a 5.

5.1.2  $t = x_{\max} = 0$ .

-Si  $F = \emptyset$  el problema es INFECTIBLE.

-Si  $F \neq \emptyset$  SOLUCION: tomar como  $Z_x^h$  la cadena de elementos antecesores de aquél y  $\in F$  con menor valor objetivo.

5.1.3  $t \neq x_{\max}$ , volver al paso 5 con  $x = x^-$ .

6 Hacer  $k = k+1$ . Ir a la iteración k.

A continuación daremos algunas aplicaciones del algoritmo.

#### 5.4.4. Problema de Knapsack.

Consideramos un problema del tipo:

$$\max \{ \sum^n c_j y_j : \sum^n a_j y_j \leq b', x \in \mathbb{Z}_+^n \}$$

donde  $c_j, a_j \in \mathbb{Z}_+^1, j=1 \dots n, b' \in \mathbb{Z}_+^1, 0 \leq c_1 \leq \dots \leq c_n, 0 < a_j \leq b' \forall j$ .

En este caso  $k_j = \left\lceil \frac{b'}{a_j} \right\rceil \forall j$ , realizamos el cambio  $y_j = k_j - x_j$  y el

problema queda como sigue:

$$\min \sum_1^n c_j x_j$$

$$\text{sa: } \sum_1^n a_j x_j \geq b$$

$$0 \leq x_j \leq \omega_j, x_j \text{ entero } \forall j.$$

El intervalo error que contiene el objetivo óptimo del problema se obtiene de la formulación general de IE en la cual se sustituye:

$$\theta_{\min} = \min \{ \lceil b/a_j \rceil + 1, j=1 \dots n \}$$

donde  $\lceil t \rceil$  representa el menor entero mayor o igual que  $t$ .

ESQUEMAS DE ENUMERACION IMPLICITA ORDENADA:  
PROGRAMACION ENTERA.

Ejemplo.

Para el problema:

$$\begin{aligned} \min x_1 + 1.1 x_2 \\ \text{sa: } 6x_1 + 9x_2 \geq 48 \end{aligned}$$

Obtenemos;

$W = \max \{ 8/8, 14/8 \}$ ,  $\theta_{\min} = 7$  y la solución propuesta por el algoritmo (5.4.3) es la cadena maximal:

$$C_g = \{(0,0), (1,0), (2,0) \dots (7,0), (8,0)\}$$

correspondiente al punto terminal factible (8,0) con valor objetivo 8. El intervalo obtenido es [5.2,8] mientras que el valor óptimo es el 6.6 alcanzado en el punto (0,6).

En el ejemplo anterior se observa que la solución propuesta por Greedy (CIMMP) se extiende a lo largo de un eje. Esto siempre ocurre para este tipo de problemas puesto que sólo es necesario sondear una rama en el árbol de cadenas maximales ya que con todas ellas se obtiene factibilidad. A continuación se trata un modelo de Knapsack con cotas en las variables lo cual hace que no todas las ramas conduzcan a puntos factible y por ello el algoritmo deberá emplear *backtracking*.



#### 5.4.5. Problema de Knapsack restringido.

El problema que consideramos es:

$$\min \sum_{j=1}^n c_j x_j$$

$$\text{sa: } \sum_{j=1}^n a_j x_j \geq b$$

$$0 \leq x_j \leq k_j, x_j \text{ entero y } a_j > 0 \forall j.$$

donde las cotas  $k_j$  pueden hacer infactible alguno de los puntos que verifican la primera restricción.

En primer lugar debemos modificar el intervalo calculado anteriormente. Para ello basta actualizar el parámetro  $\theta_{\min}$ :

$$\theta_{\min} = \min \{ \lceil b/a_j \rceil + 1, k_j + 1, j=1 \dots n \}.$$

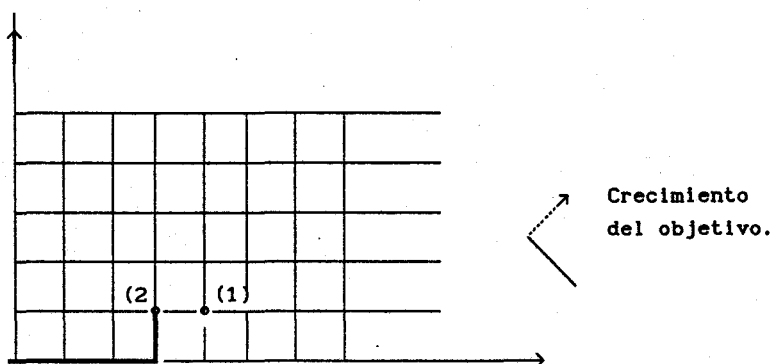
Como comentábamos antes, en este caso no todas las cadenas maximales conducen a puntos factibles, con lo cual la búsqueda en profundidad adquiere sentido. Asimismo habrá que hacer uso de la condición de salida del algoritmo para determinar una cadena maximal que esté en las condiciones de asegurar que el intervalo obtenido a partir de ella contiene el objetivo óptimo.

ESQUEMAS DE ENUMERACION IMPLICITA ORDENADA:  
PROGRAMACION ENTERA.

A continuación se comparan las soluciones obtenidas para el siguiente problema por medio del algoritmo greedy de Dobson (1982) y el algoritmo greedy 5.4.3.

$$\begin{aligned} \min & x_1 + 4x_2 \\ \text{sa: } & x_1 + 2x_2 \geq 5 \\ & x_1 \leq 4, x_1, x_2 \in \mathbb{Z}_+ . \end{aligned}$$

Dobson obtiene después de 5 iteraciones el punto (4,1) con objetivo 8. Sin embargo la solución obtenida por Greedy (CIMMP) es una cadena que tiene asociada el punto factible (3,1) que es la solución óptima. Gráficamente:



(1) Es la solución dada por Dobson.

(2) La solución dada por Greedy(CIMMP).

Podemos observar una aportación interesante como es el hecho de que mientras que el primer algoritmo incrementa valores de variables sin posibilidad de decrementarlas en lo sucesivo, en el segundo algoritmo se tiene en cuenta esta posibilidad.

#### 4.4.6 Problema lineal entero con restricciones lineales.

Consideremos el problema

$$\begin{aligned} \min \quad & cx \\ \text{sa: } \quad & Ax \leq b \\ & x \in \mathbb{Z}_+^n \end{aligned}$$

Donde suponemos que si  $a_{ij} \notin \mathbb{Z}$  disponemos de cotas sobre las variables de la forma  $0 \leq x_j \leq k_j$ ,  $j=1 \dots n$ . Si  $a_{ij} \in \mathbb{Z}$ , entonces podemos construir estas cotas explícitamente usando el conocido resultado de Papadimitrou (1982):

Dado  $P = \{x \in \mathbb{R}_+^n / Ax \leq b\}$  donde  $(A,b)$  es una matriz entera  $m \times (n+1)$  y sea  $\theta_A = \max_{ij} |a_{ij}|$ ,  $\theta_b = \max_i |b_i|$ ,  $\theta = \max\{\theta_A, \theta_b\}$ , entonces la envolvente convexa de  $S = P \cap \mathbb{Z}_+^n$  tiene sus puntos extremos acotados por  $\omega = ((m+n)n\theta)^n$ , ie. si  $x$  es punto extremo de  $\text{conv}(S) \Rightarrow x_j \leq \omega \forall j=1 \dots n$ .

Para obtener en este caso el intervalo que contiene el objetivo óptimo, una vez obtenida la cadena  $Z_x$  propuesta por el algoritmo greedy, se resuelve un problema lineal para determinar  $\theta_{\min}$ . En la construcción de este problema se tiene en cuenta que dado un conjunto maximal  $Z_x$ , con elemento terminal factible  $x$ , entonces el cardinal de  $Z_x$  viene dado por la suma de las componentes de  $x$  más uno. Esto se debe a la caracterización de los conjuntos  $Z_x$  hecha en el capítulo 1. Además por ser  $x$  factible, el cardinal mínimo de una cadena maximal está acotado inferiormente por el objetivo óptimo del problema siguiente:

ESQUEMAS DE ENUMERACION IMPLICITA ORDENADA:  
PROGRAMACION ENTERA.

$$\begin{aligned} & \min 1'x \\ \text{sa: } & Ax \geq b, x \in \mathbb{R}^n \\ & 0 \leq x_j \leq k_j \quad \forall j=1 \dots n \end{aligned}$$

con lo cual podemos tomar  $\theta_{\min} = \lceil 1'x^* \rceil + 1$ , con  $x^*$  la solución óptima del problema anterior.

Una vez obtenido este parámetro, basta sustituir en la expresión general del intervalo error (IE).

Del mismo modo se determinan las cotas del intervalo error cuando el conjunto factible del problema considerado viene dado por expresiones más generales. El único requisito que se necesita es que el objetivo sea lineal y que el problema en variables continuas que define el valor  $\theta_{\min}$  pueda ser resuelto sin dificultad.

## REFERENCIAS.

- Balas E. (1965). An Additive Algorithm for Solving Linear Programs with Zero-One Variables, *Operations Research* 13(4), 517-546.
- Béla Martos. (1975). Nonlinear Programming, North-Holland.
- Bowman V. J. and Starr J. H. (1979). Partial Orderings in Implicit Enumeration, *Annals of Discrete Mathematics* 99-116.
- Chen D. S. and Zionts S. (1970). An Exposition of the Group Theoretic Approach to Integer Linear Programming, *School of Management Working Paper NO. 78, State University of New York at Buffalo*.
- Chen D. S. and Zionts S. (1976). Comparison of Some Algorithms for Solving the Group Theoretic Integer Programming Problem, *Operations Research* 24, 1120-1128.
- Dantzig G. (1963). Linear Programming and Extensions, *Princeton University Press*.
- Dietrich B. L. (1989). Matroids and Antimatroids -A Survey, *Discrete Mathematics* 78, 223-237.
- Dobson G. (1982). Worst-case Analysis of Greedy Heuristics for Integer Programming with Nonnegative Data, *Mathematics of Operations Research* 7, 515-531.

Echols R.E. and Cooper L. (1968). Solution of Integer Linear Programming Problems by Direct Search, *Journal of the ACM* 15, 75-84.

Ecker J. and Kouada I. (1975). Finding Efficient Points for Linear Multiple Objective Programs, *Mathematical Programming*, 8 375-377.

Evans J. and Steuer R. (1973). A Revised Simplex Method for Linear Multiple Objective Programs, *Mathematical Programming* 8, 54-72.

Faaland B.H. and Hillier F.S. (1979). Interior Path Methods for Heuristic Integer Programming Procedures, *Operations Research* 27, 1069-1087.

Farbey B., Land A. and Murchalnd J. (1967). The Cascade Algorithm for Finding All Shortest Distances in Directed Graph, *Management Science* 14(1), 19-28.

Floyd R. (1962). Algorithm 97: Shortest Path, *Communications of the ACM* 5(6), 345.

Fox G.E. and Scudder G.D. (1986). A Simple Strategy for Solving a Class of 0-1 Integer Programming Models, *Computers and Operations Research* 13, 707-712.

Gal T. and Nedoma J. (1972). Multiparametric Linear Programming, *Management Science*, 18 406-421.

Giles R. and Pulleyblank W.R. (1979). Total Dual Integrality and Integral Polyhedra, *Linear Algebra and its Applications* 25, 191-196.

Glover F. (1966). An Algorithm for Solving the Linear Integer Problem Over a Finite Additive Group, with Extensions to Solving General and Certain Nonlinear Integer Programs, *Operations Research Center Report 66-29*, the University of California at Berkeley.

Glover F. (1969). Integer Programming over a Finite Additive Group, *SIAM J. Control* 7, 213-231.

Glover F., and Litzler L. (1969) Extension of an Asymptotic Integer Programming Algorithm to General Integer Programming Problem, *School of Business Report, the University of Texas at Austin*.

Goecke O. and Schrader R. (1990). Minor characterization of Undirected Branching Greedoids -A short proof, *Discrete Mathematics* 82, 93-99.

Gomory R. E. (1965). On the Relation between Integer and Non-integer Solutions to Linear Programs, *Proc. Nat. Acad. Sci. U.S.A* 53(2), 260-265.

Gomory R. E. (1969). Some Polyhedra Related to Combinatorial Problems, *Journal of Linear Algebra and Applications* 2(4), 451-558.

Gorry G. A., Shapiro J. F. (1971). An Adaptive Group Theoretic Algorithm for Integer Programming Problems, *Management Science* 17(5), 285-306.

Gorry G. A., Northup W. D. and Shapiro J. F. (1973). Computational Experience with a Group Theoretic Integer Programming Algorithm, *Mathematical Programming* 4, 171-192.

Gorry G. A., Shapiro J. F. and Wolsey L. A. (1972). Relaxation Methods for Pure and Mixed Integer Programming Problems, *Management Science* 18, 229-239.

- Hu T. C. (1970). On the Asymptotic Integer Algorithm, *Journal of Linear Algebra and its Applications* 3(2), 279-294.
- Jackson R., Boggs P., Nash S., Powell S. (1991). Guidelines for Reporting Results of Computational Experiments. Report of the AD HOC Committee, *Mathematical Programming*, 49, 413-425.
- Karp R.M. (1972). Reducibility Among Combinatorial Problems, *Complexity of Computer Computations* Miller R.E and Thatcher J.W, eds, Plenum Press, New York, 85-103.
- Klein D. and Hannan E. (1982). An Algorithm for the Multiple Objective Integer Linear Programming Problem, *E.J.O.R.* 9, 378-385.
- Korhonen P., Wallenius J. and Zionts S. (1984). Solving the Discrete Multiple Criteria Problem using Convex Cones, *Management Science*, 11 1336-1345.
- Korte B. and Lovász L. (1983). Structural Properties of Greedoids, *Combinatorica* 3, 359-374.
- Lawler E. L. (1972). A Procedure for Computing the K Best Solutions to Discrete Optimization Problems and its Application to the Shortest Path Problem, *Management Science* 18(7), 401-405.
- Lawler E. L. (1985). Submodular Functions and Polymatroid Optimization -Annotated Bibliographies. M. O'hEigeartaig, J. K. Lenstra and A. H. G. RinnoyKan (eds.), *John Wiley & Sons, New York*, 32-38.
- Lenstra J.H.W. (1983). Integer Programming with a Fixed Number of Variables, *Mathematics of Operations Research*, 8 538-548.
- Liqun Qi. (1988). Odd Submodular Functions, Dilworth Functions and Discrete Convex Functions, *Mathematics of Operations Research* 13, 435-446.



Lovász, L. (1983). Submodular Functions and Convexity. *Mathematical Programming: The State of the Art*. Springer-Verlag, 235-257.

Nemhauser G.L, Rinnooy Kan A.H.G (1989). Handbooks in Operations Research and Management Science: Optimization. 1, North-Holland.

Nemhauser G.L. and Wolsey, L.A. (1988). Integer and Combinatorial Optimization, *Wiley-Interscience*.

Papadimitriou C.H. and Steiglitz K. (1982). Combinatorial Optimization: Algorithms and Complexity. *Prentice-Hall*, Englewood Cliffs.

Parker R.G. and Rardin, R.L. (1988). Discrete Optimization. *Academic Press, Inc.*

Ramesh R., Karwan M. and Zionts S. (1989). Preference Structure Representation using Convex Cones in Multicriteria Integer Programming, *Management Science*, 9 1092-1105.

Ramesh R. and Zionts S. (1986). A Class of Practical Interactive Branch and Bound algorithms for Multicriteria Integer Programming, *E.J.O.R.*, 26 161-172.

Roy B. (1971). Problems and Methods with Multiple Objective Functions, *Mathematical Programming* 1 239-266.

Salkin H. M. and Mathur K. (1989). Foundations of Integer Programming, *North-Holland*.

Schrijver A. (1986). Theory of Linear and Integer Programming, *Wiley-Interscience*.

Senju S. and Toyoda Y. (1968). An Approach to Linear Programming with 0-1 Variables, *Management Science* 15, 196-207.

- Shapiro J. (1968). Dynamic Programming Algorithms for Integer Programming Problem-I: The integer Programming Problem Viewed as a Knapsack Type Problem, *Operations Research* 16(5), 928-947.
- Vassilev V. and Genova K. (1991). An Algorithm of Internal Feasible Directions for Linear Integer Programming, *E.J.O.R.* 52, 203-214.
- Whitney H. (1935). On the Abstract Properties of Linear Dependences, *American Journal of Mathematics* 57, 509-533.
- Wolsey L. A. (1973). Generalized Dynamic Programming Methods in Integer Programming, *Mathematical Programming* 4, 222-232.
- Yu P.L. and Zeleny M. (1975). The Set of all Nondominated Solutions in Linear Cases and in Multicriteria Simplex Method, *J. Math. Anal. Appl.* 49 430-468.
- Zimmermann U. (1981). Linear and Combinatorial Optimization in Ordered Algebraic Structures, *North-Holland Publishing Company*.
- Zionts S. (1977). Integer Linear Programming with Multiobjectives, *Annals of Discrete Mathematics* 1, 551,562.