DESARROLLO DE UN ALGORITMO DE DESCOMPOSICION UTILIZADO EN PROGRAMAS LINEALES

FELIX EDUARDO VACA OBANDO

TESIS SOMETIDA AL CUERPO DOCENTE DE LA COORDENACION DE LOS PROGRAMAS DE POST-GRADUACION DE INGENIERIA DE LA UNIVERSIDAD FEDERAL DE RIO DE JANEIRO COMO PARTE DE LOS REQUISITOS NECESARIOS PARA LA OBTENCION DEL GRADO DE MASTER EN CIENCIA (M. Sc.)

Aprobada por:

Presidente

RIO DE JANEIRO
ESTADO DA GUANABARA-BRASIL
DICIEMBRE DE 1971

A

Maria Luisa,

Guadalupe y

Trotsky Eduardo

AGRADECIMIENTOS

Al *Profesor Rodrigo Alvaro Restrepo* por su valiosa orientación durante la elaboración de esta tesis;

Al *Profesor Nelson Maculan Filho* por su eficiente colaboración en el desarrollo de la misma;

Al *Profesor Alexandre Arduino* por las sugerencias;

A la *COPPE* por la ayuda financiera;

Al *Departamento de Cálculo Científico (DCC)* por la utilización de sus instalaciones;

A *Maria de Lourdes de Almeida* por su excelente trabajo de mecanografía.

SUMARIO

Este trabajo se refiere a la descomposición de programas lineales. Es presentado el desarrollo teórico del principio de descomposición de Dantzig-Wolfe y su algoritmo.

Luego es elaborado un programa Fortran para este algoritmo, y se resuelve un pequeño ejemplo numérico para ilustrar el método.

Antes del desarrollo del programa son presentadas todas las instrucciones necesarias, para quienes estén interesados en la aplicación directa del algoritmo.

ABSTRACT

This work is concerned with the decomposition of linear programs. There is presented a rigorous theoretical development of Dantzig-Wolfe's decomposition principles and the associated algorithm. Then, a Fortran program is constructed for this algorithm, and a simple numerical problem is solved, for ilustrating the technique. Before developing the computer program, all the instructions needed are presented, for the users that may be interested only on the direct application of the algorithm.

SUMÁRIO

O presente trabalho refere-se à decomposição de programas lineares. É apresentado o desenvolvimento teórico do princípio de decomposição de Dantzig-Wolfe e seu algorítmo. Depois é elaborado um programa Fortran para êste algorítmo, e resolve-se um exemplo numérico para ilustrar o método.

Antes do desenvolvimento do programa são apresentadas tôdas as instruções necessárias para quem esteja interessado na aplicação direta do algorítmo.

INDICE

INTRODUCCIÓN

Cuando la teoría de optimización es utilizada para la solución de problemas prácticos de economía o de la indústria, es frecuente que tengamos que resolver problemas con un gran número de restricciones y de variables. En este caso el algoritmo simplex ordinario no proporciona un metodo efectivo de computacción. *DANTZIG y WOLFE* [1] desarrollaron un algoritmo, el cual en ciertos casos permite una descomposición en diferentes problemas parciales y desde los cuales posteriormente se calcula la solución para el problema inicial.

La publicación del principio de descomposición de Dantzig-Wolfe en 1960 fué el inicio de un trabajo intensivo en programación matemática de grande escala. El procedimiento es mas eficiente cuando es aplicado a programas lineales cuyas matrices de coeficientes tienen una estructura angular, esto es, uno o mas bloques independientes relacionados por ecuaciones acopladas. Estas son manipuladas para formar un "master program" equivalente, con un número de filas un poco mayor que el de las ecuaciones acopladas del problema original, pero que contiene mucho más columnas. Este programa es resuelto sin necesidad de tabular todas esas columnas, ge-

nerándolas cuando el algoritmo simplex las necesita. El algoritmo resultante incluye iteraciones entre un conjunto de problemas independientes, cuyas funciones objetivas contienen un parámetro variable, y el master program. Los subproblemas reciben un conjunto de parámetros (dual o multiplicador simplex) desde el master program. Estos proporcionan sus soluciones al master program el cual combina estas con las soluciones anteriores y calcula el nuevo vector dual. Estos nuevamente entran en las funciones objetivas de los subproblemas y la iteración continua hasta que sea cumplido un test de optimización.

CAPITULO I

1. DESCOMPOSICION DE PROBLEMAS PARCIALMENTE DESACOPLADOS

1.1 PROBLEMAS PARCIALMENTE DESACOPLADOS

Dos restricciones aplicadas a un conjunto de variables se llaman desacopladas si el conjunto de variables que en ellas existen pueden ser divididas en dos conjuntos mutuamente exclusivos, de tal manera que la primera restricción explicitamente incluye solamente las variables del primer subconjunto y la segunda restricción explicitamente incluye solo las variables del segundo subconjunto , por ejemplo, $x_1 + x_2 = 3$ y $x_3 + x_4 = 4$.

En general, un problema de programación matemática de la forma:

$$P: \max \{f(x) \mid X \in S \subseteq R^n\} \qquad (1)$$

es parcialmente desacoplado si después de la partición de las variables, incluidas en el vector X, en subvectores $\overline{X}_1$, $\overline{X}_2, \ldots \overline{X}_p$ el conjunto de restricciones S puede ser escrito en la forma

$$S = \{X = (\overline{X} , \overline{X} , \ldots \overline{X}_p) \mid X \in S_0 \quad y$$

$$\overline{X}_j \in S_j \text{ para } j = 1, \ldots, p\} \qquad (2)$$

donde, para evitar casos triviales, se asume que algún S_j con $j \geqslant 1$ no es todo el espacio. Cuando, como es usual el caso en problemas de programación, el conjunto S es definido por un conjunto finito de restricciones funcionales de la forma $g^i(X) \geqslant 0$, un problema parcialmente desacoplado será uno en el que algunas de estas restricciones funcionales aplicadas solo a las componentes del vector X, que constituyen el vector $\overline{X}_j$, y el conjunto de todas las $\overline{X}_j$ que satisfacen estas restricciones constituyen el conjunto S_j.

En casos particulares donde f(x) tiene la forma:

$$f(x) = \sum_{j=1}^{p} f_j(\overline{X}_j) \qquad (3)$$

(esto ocurre automáticamente en todo problema de programación lineal) y donde el primer conjunto de restricciones $X \in S_0$ no está presente en (2), se dice que el problema es completamente desacoplado. En estos casos

$$\max \{f(x) \mid X_j \in S_j, \quad j=1,\ldots,p\} =$$

$$= \sum_{j=1}^{p} \max \{f_j(\overline{X}_j) \mid \overline{X}_j \in S_j\}$$

y por tanto el problema de programación original (1), puede

ser resuelto mediante soluciones separadas de cada uno de los p subproblemas.

$$P_j: \max \{f_j(X_j) | X_j \in S_j\}, \; j=1,..,p$$

El total de trabajo requerido para resolver estos p subproblemas es en general menor que la cantidad de trabajo requerida para resolver problemas acoplados del mismo tamaño del problema original.

Los problemas en los cuales la función objetiva f tiene la forma dada por (3) pero en los que algunas restricciones acopladas de la forma $X \in S_0$ están presentes, pueden ser llamados, problemas de descomposición, y estos serán los problemas que estudiaremos en este trabajo.

1.2 DESCOMPOSICION DE PROBLEMAS DE PROGRAMACION LINEAL

Cuando el problema que estamos considerando es un programa de programación lineal, cada una de las funciones f_j en (3) es lineal y puede ser de la forma

$$f_j(X_j) = C_j X_j \;, \; j=1,\ldots,p \qquad (4)$$

Por tanto, en los problemas lineales cada uno de los conjuntos S_0, $S_1, \ldots S_p$ puede ser descrito, después de añadir las respectivas variables de folga, por un conjunto finito de ecuaciones lineales, y estos conjuntos por tanto son de la forma:

$$S_0 = \{X = (\overline{X}_1, \ldots, \overline{X}_p) \mid \sum_{j=1}^{p} \overline{A}_j \overline{X}_j = b_0\} \qquad (5)$$

$$S_j = \{\overline{X}_j \mid A_j \overline{X}_j = b_j,\ \overline{X}_j \geqslant 0\},\ j=1,\ldots,p \qquad (6)$$

para matrices $\overline{A}_1, \ldots, \overline{A}_p$, $A_1, \ldots, A_p$ y vectores $b_0, b_1, \ldots, b_p$ dados. Por tanto, para estos problemas la matriz A de las restricciones y el vector b, para restricciones lineales, de la forma AX = b pueden ser representadas gráficamente en forma de bloques partidos, como indicamos a continuación:

$$
A = \begin{array}{c|cccccc}
 & n_1 & n_2 & n_3 & & & n_p \\
\hline
m_0 & \bar{A}_1 & \bar{A}_2 & \bar{A}_3 & & & \bar{A}_p \\
m_1 & A_1 & & & & & \\
m_2 & & A_2 & & & & \\
m_3 & & & A_3 & & & \\
 & & & & & & \\
m_p & & & & & & A_p \\
\hline
 & C_1 & C_2 & C_3 & & & C_p
\end{array}
\qquad
b = \begin{array}{c}
b_0 \\ b_1 \\ b_2 \\ b_3 \\ \\ b_p
\end{array}
$$

Donde la matriz $\bar{A}$ que contiene m_0 filas, deberá estar dividida verticalmente en p submatrices $\bar{A}_j$ $(j=1,\ldots,p)$ teniendo respectivamente los mismos índices de las matrices A_j de los p subproblemas independientes. El vector demanda b y el vector costo C están divididos horizontalmente y verticalmente respectivamente como la matriz A.

Estas matrices, generalmente son llamadas matrices angulares. Existen otras clases de descomposición de problemas de programación lineal algo mas generales que estos y que también tienen matrices angulares como restricciones, pero estos no serán considerados en este trabajo.

1.3 TRANSFORMACIONES DE PROBLEMAS DE PROGRAMACION LINEAL PARCIALMENTE DESACOPLADOS

Los problemas de programación lineal parcialmente desacoplados serán transformados de tal manera que disminuyen considerablemente la cantidad de cálculos requerida para el cómputo de los programas óptimos. Estas transformaciones que disminuyen el número de filas de la matriz de restricciones, a expensas del incremento del número de columnas, depende de la estructura del conjunto de restricciones S_j y por esta razón será conveniente considerar dos casos separadamente, dependiendo de si estos conjuntos de restricciones constituyen conjuntos limitados o conjuntos ilimitados.

CASO 1 - CADA CONJUNTO S_j ES LIMITADO Y NO VACIO

En este caso siendo cada S_j un conjunto convexo cerrado, no vacio, limitado, debe contener un conjunto E_j no

vacio, de puntos extremos, y todo punto de S_j será representable como una combinación convexa de estos puntos extremos. Además, en el caso lineal en consideración, cada uno de estos conjuntos S_j es también un conjunto poliédrico limitado por un número finito de hiperplanos y por tanto estos puntos extremos, corresponden a las intersecciones de algunos de estos hiperplanos, que también debe ser finito. Por consiguiente podemos escribir.

$$E_j = \{X_{j1}, X_{j2}, \ldots X_{js_j}\} \tag{8}$$

y para cada $\overline{X}_j$ en S_j

$$\overline{X}_j = \sum_{i=1}^{s_j} \lambda_{ji} X_{ji} \quad \text{con } \lambda_{ji} \geqslant 0 \text{ y } \sum_{i=1}^{s_j} \lambda_{ji} = 1 \tag{9}$$

Luego definimos para cada índice j y cada punto extremo X_{ji} un vector t_{ji} y un número c_{ji} mediante.

$$t_{ji} = \overline{A}_j X_{ji} \tag{10}$$

$$c_{ji} = C_j X_{ji} \tag{11}$$

Entonces la función objetiva para el problema de programación

$$f(x) = \sum_{j=1}^{p} C_j X_j = \sum_{j=1}^{p} \sum_{i=1}^{s_j} c_{ji} \lambda_{ji}$$

y la condición para que X esté en S_0 será

$$\sum_{j=1}^{p} \bar{A}_j X_j = \sum_{j=1}^{p} \sum_{i=1}^{s_j} t_{ji} \lambda_{ji} = b_0$$

Por tanto, si los vectores X_{ji} son todos conocidos, el problema de programación lineal en consideración puede ser escrito en la siguiente forma:

$$\max \left\{ \sum_{j=1}^{p} \sum_{i=1}^{s_j} c_{ji} \lambda_{ji} \middle| \sum_{j=1}^{p} \sum_{i=1}^{s_j} t_{ji} \lambda_{ji} = \right.$$

$$\left. = b_0, \sum_{i=1}^{s_j} \lambda_{ji} = 1 \ , \ \lambda_{ji} \geqslant 0 \ , \ \forall_{ji} \right\} \qquad (12)$$

el cual, como el problema original, es um problema de programación lineal con una matriz de restricciones partida que es de la siguiente forma:

$$
T = \begin{array}{c|c|c|c|c|c|}
 & \overbrace{\qquad\qquad}^{s_1} & \overbrace{\qquad\qquad}^{s_2} & \overbrace{\qquad\qquad}^{s_3} & & \overbrace{\qquad\qquad}^{s_p} \\
\hline
m_o & t_{11}\cdots t_{1s_1} & t_{21}\cdots t_{2s_2} & t_{31}\cdots t_{3s_3} & & t_{p1}\cdots t_{ps_p} \\
\hline
1 & 1\ 1\ \ldots\ 1 & 0\ 0\ \ldots\ 0 & 0\ldots 0 & \ldots\ldots & 0\ 0\ \ldots\ 0 \\
 & 0\ 0\ \ldots\ 0 & 1\ 1\ \ldots\ 1 & 0\ldots 0 & \ldots\ldots & 0\ 0\ \ldots\ 0 \\
 & 0\ 0\ \ldots\ 0 & 0\ 0\ \ldots\ 0 & 11\ldots 1 & \ldots\ldots & 0\ 0\ \ldots\ 0 \\
 & \ldots\ldots\ldots & \ldots\ldots\ldots & \ldots\ldots & \ldots\ldots & \ldots\ldots\ldots \\
 & 0\ 0\ \ldots\ 0 & 0\ 0\ \ldots\ 0 & 0\ldots 0 & \ldots\ldots & 1\ 1\ \ldots\ 1 \\
\hline
\end{array}
$$

Como condición previa, esta matriz T tiene un número de columnas mucho mayor que el de la matriz A originalmente considerada, pero puede tener un número muy pequeño de filas ya que cada conjunto de restricciones de la forma $\overline{X}_j \in S_j$ fué reemplazado por una simple fila. Si el método simplex Revisado se utiliza para la solución de este problema, cada base para T será en general mucho más pequeña que una base para A, pero se requiere más trabajo para obtener el vector $Z_{ji} - C_{ji}$ (en la notación usual del procedimiento simplex)y también para provar la optimización de cualquier solución del problema. Esta dificultad será solucionada en la próxima sección, mediante el fornecimiento de una prueba para optimización de las soluciones. Pero antes de entrar a discutir esta prueba, vamos a considerar el segundo caso de

transformación del problema.

CASO 2 - ALGUN CONJUNTO S_j ES ILIMITADO

Siendo S_j un conjunto ilimitado, se sabe [2] que este posee puntos extremos X_{ji} $(i=1,\ldots n_j)$ y un número finito de vectores extremos, que salen desde ciertos puntos extremos. Las direcciones R_{ji} de los vectores extremos del conjunto S_j serán definidos por vectores ℓ_{ji} $(i=1,\ldots \ell_j)$ multiplicados por un escalar. Algunos ℓ_{ji} pueden ser obtenidos durante el proceso del algoritmo simplex. Por tanto todo punto $\overline{X}_j$ de S_j puede ser representado como la suma de una combinación convexa de puntos extremos y de una combinación lineal positiva de ℓ_{ji} [2]

$$\overline{X}_j = \sum_{i=1}^{s_j} \lambda_{ji} X_{ji} + \sum_{i=1}^{\ell_j} \mu_{ji} \ell_{ji} \qquad (13)$$

con

$$\sum_{i=1}^{s_j} \lambda_{ji} = 1 \quad , \quad \lambda_{ji} \geqslant 0 \quad , \quad \mu_{ji} \geqslant 0$$

Definimos para cada indices i, j un vector m_{ji} y un número e_{ji} tales que:

$$m_{ji} = \overline{A}_j \ \ell_{ji} \tag{14}$$

$$e_{ji} = c_j \ \ell_{ji} \tag{15}$$

y la función objetiva del problema original quedará de la siguiente forma:

$$f(x) = \sum_{j=1}^{p} \sum_{i=1}^{s_j} c_{ji} \ \lambda_{ji} + \sum_{j=1}^{p} \sum_{i=1}^{\ell_j} e_{ji} \ \mu_{ji}$$

y la condición para que X esté en S_0 sera:

$$\sum_{j=1}^{p} \overline{A}_j \ X_j = \sum_{j=1}^{p} \sum_{i=1}^{s_j} \lambda_{ji} \ t_{ji} +$$

$$+ \sum_{j=1}^{p} \sum_{i=1}^{\ell_j} \mu_{ji} \ m_{ji} = b_0$$

y por tanto el problema de programación lineal puede ser escrito de la siguiente forma:

$$\max \left\{ \sum_{j=1}^{p} \sum_{i=1}^{s_j} c_{ji} \ \lambda_{ji} + \right.$$

$$+ \sum_{j=1}^{p} \sum_{i=1}^{\ell_j} e_{ji} \ \mu_{ji} \ \Big| \ \sum_{j=1}^{p} \sum_{i=1}^{s_j} \lambda_{ji} \ t_{ji} +$$

$$+ \sum_{j=1}^{p} \sum_{i=1}^{\ell_j} \mu_{ji} \ m_{ji} = b_0 \quad ,$$

$$\left.\sum_{i=1}^{s_j} \lambda_{ji} = 1 \quad , \quad \lambda_{ji} \geqslant 0 \ , \ \mu_{ji} \geqslant 0 \right\} \qquad (16)$$

el cual también es un problema de programación lineal cuya matriz de restricciones está partida como indicamos a continuación.

$$T_m = \begin{array}{c|c|c|c|c|}
 & s_1+\ell_1 & s_2+\ell_2 & & s_p+\ell_p \\ \hline
m_o & t_{11}\cdots t_{1s_1}\ ,\ m_{11}\cdots m_{1\ell_1} & t_{21}\cdots t_{2s_2}\ ,m_{21}\cdots m_{2\ell_2} & & t_{p1}\cdots t_{ps_p},m_{p1}\cdots m_{p\ell_p} \\ \hline
1 & 1\cdots\cdots 1\ ,\ 0\cdots\cdots 0 & & & \\ \hline
1 & & 1\cdots\cdots 1\ ,0\cdots\cdots 0 & & \\ \hline
 & & & & \\
1 & & & & 1\cdots\cdots 1,0\cdots\cdots 0 \\ \hline
\end{array}$$

1.4 METODO DE DESCOMPOSICION DE DANTZIG-WOLFE

Vamos a considerar un procedimiento para probar la optimización de una solución viable básica λ^0 del programa de programación lineal *L*. Este procedimiento que se debe a G. Dantzig y P. Wolfe y que es conocido como Método de Descomposición de Dantzig-Wolfe, evita la necesidad de realizar todos los cómputos, excepto un número relativamente pequeño de columnas de la matriz T de la sección anterior y esto reduce la prueba para optimizar la solución de los p problemas de programación lineal desacoplados, que generalmente no son grandes.

Para justificar el procedimiento utilizaremos el siguinte lema.

LEMA 1.4.1 - Siendo λ^0 una solución básica viable del problema L, y siendo (π, u) las variables duales correspondientes a esta solución λ^0, con π correspondiendo a las filas que contienen los elementos t_{ji} de T y u correspondiendo a las p últimas filas de T. λ^0 es óptima si y solamente si, para cada j,

$$\max \{ (\pi \overline{A}_j - C_j)\overline{X}_j + u_j \mid A_j\overline{X}_j = b_j , \overline{X}_j \geqslant 0\} \leqslant 0$$

Prueba:

Utilizando la notación y resultados usuales para el método simplex, podemos afirmar que la solución via - ble básica λ^0 es óptima, si y solo si, para cada i y j, $z_{ji} - C_{ji} \geqslant 0$; pero, por las reglas usuales del procedimiento simplex, denotamos por T_{ij} la columna correspondiente de T, tal que

$$z_{ji} - C_{ji} = (\pi, u)\ T_{ji} - C_{ji}$$

$$= \pi t_{ji} + u_j - c_{ji}$$

$$= (\pi\ \overline{A}_j - C_j)\ X_{ji} + u_j$$

y por consiguiente

$$\max_{ji} \{z_{ji} - C_{ji}\} = \max\{(\pi\overline{A}_j - C_j)\ X_{ji} + u_j \mid X_{ji} \in E_j\}$$

$$= \max\{(\pi\overline{A}_j - C_j)\overline{X}_j + u_j \mid \overline{X}_j \in S_j\}$$

la última igualdad proviene del hecho que la solución de un problema de programación lineal es alcanzada en uno de los

puntos extremos de estos conjuntos viables. De esta igualdad sigue inmediatamente la afirmación del lema.

Habiendo establecido este lema, la prueba deseada para optimización es inmediata; un simple cálculo del vector dual (π,u) y con este calcular las funciones objetivas de los p problemas de programación lineal

$$\max\{(\pi\bar{A}_j - C_j)\, X_j + u_j \mid A_jX_j = b_j\ ,\ X_j \geqslant 0\}$$

y luego resolver estos problemas. Si el mínimo valor de todos estos problemas es no positivo, el correspondiente λ^0 es un programa para T; además las soluciones de estos problemas indican cuales columnas de T entrarán en la base para la próxima iteración y por tanto el lema precedente no solo provee una prueba para optimización, así como también un proceso algorítmico para la solución del problema de programación lineal dado.

1.5 ALGORITMO DE DESCOMPOSICION

A continuación describimos una iteración para la solución del problema de programación lineal, donde consideraremos, solamente el Caso 1, esto es, los S_j son limi-

tados y no vacios. Siendo B la base que corresponde al programa encontrado en la iteración anterior y $C^B = C_{ji}$ el vector de costos correspondientes a las columnas que se encuentran en la base B, utilizamos el método <u>simplex revisado</u> y podemos conocer las (m + p) variables del dual mediante .... $\bar{\pi} = C^B B^{-1}$.

Calculamos, mediante el criterio simplex, los Z_{ji} que son iguales al producto de $\bar{\pi}$ con cada vector.

$$(t_{ji}, 0, \ldots 0, 1, 0 \ldots 0) = (t_{ji}, e_j)$$

t_{ji} tendrá m_o componentes y

e_j será el jesimo vector unitario y tendrá p componentes.

Este producto podemos descomponerlo en los subvectores π y u, donde π es el subvector formado por las m_o primeras componentes del vector $\bar{\pi}$ y

$$u = (u_1, u_2, \ldots u_p)$$

formado por las p últimas componentes de $\bar{\pi}$

Si $Z_{ji} - C_{ji} = \pi\, t_{ji} + u_j - C_{ji} \leqslant 0$ (para todo i y j), el programa asociado con B es mínimo.

Si $Z_{ji} - C_{ji} = \pi\, t_{ji} + u_j - C_{ji} > 0$ por lo me-

nos para un ji, el programa asociado con B no es mínimo y por tanto cambiamos la base, sustituyendo en B un nuevo vector t_{ji}. Este vector t_{kr} que entra en la base se determina mediante el criterio Simplex.

$$Z_{kr} - C_{kr} = \max_{ji} (Z_{ji} - C_{ji}) > 0$$

El criterio Simplex usual determina el vector que sale de B y es reemplazado por t_{kr}.

Por esta razón en cada iteración y utilizando siempre el algoritmo Simplex Revisado solo se necesita conocer el máximo valor de $Z_{ji} - C_{ji}$ en el conjunto de columnas ji de la matriz T, para saber si obtuvimos el programa mínimo, o si el programa actual puede ser mejorado mediante el cambio de la columna kr en esa base.

El <u>algoritmo de descomposición</u>, por tanto, esencialmente consiste en aplicar el método Simplex Revisado al problema (12) en el cual la mayoría de los coeficientes .... (t_{ji}, C_{ji}) permanecen desconocidos y solamente son calculados en cada iteración, los coeficientes que son necesarios para escoger el vector que entra en la base. El procedimiento es como sigue:

Para cada j resolvemos el problema de programación lineal

$$A_j X_j = b_j$$

$$X_j \geqslant 0$$

$$\max\ (\pi \overline{A}_j - C_j) X_j$$

La solución de cada uno de estos problemas es un programa máximo de punto extremo X_{jr_j} tal que ..... $r_j \in \{1, \ldots s_j\}$ y tal que

$$(\pi \overline{A}_j - C_j) X_{jr_j} = \max \left[(\pi \overline{A}_j - C_j) X_{ji} \right]$$

Luego determinamos el índice k tal que

$$(\pi \overline{A}_k - C_k) X_{kr_k} + u_k = \max_{j \in (1, \ldots p)} \left[(\pi \overline{A}_j - C_j) X_{jr_j} + u_j \right]$$

y haciendo $r_k = r$ tenemos

$$Z_{kr} - C_{kr} = \max\ (Z_{ji} - C_{ji})$$

Si $Z_{kr} - C_{kr} \leqslant 0$, el programa básico λ_{ji} es minimo y mediante (9) calculamos el programa mínimo $\overline{X}_j$.

Si $Z_{kr} - C_{kr} > 0$, el vector (t_{kr}, e_k) y el costo asociado C_{kr} son introducidos en la base, t_{kr} y C_{kr} son calculados mediante

$$t_{kr} = \bar{A}_k X_{kr_k}$$

$$C_{kr} = C_k X_{kr_k}$$

Para determinar el vector que sale de la base y para transformar la inversa solamente se requiere conocer el vector

$$Y_{kr} = B^{-1} \begin{bmatrix} t_{kr} \\ e_k \end{bmatrix}$$

El algoritmo de descomposición es válido para el caso 2 si se mantiene la siguiente consideración.

Suponemos que la maximización de uno de los p subproblemas parciales nos lleva a encontrar un programa máximo infinito. Si esto acontece existirá la columna a_{jk} de la matriz A_j y si designamos por B_j a la base correspondiente y con

$$Y_{jk} = (B_j^{-1})\ a_{jk}$$

tenemos

$$Y_{jk} \leqslant 0$$

Cualquiera que sea el valor positivo de la variable X_{jk}, las variables básicas se mantienen positivas, y el valor de la función objetiva tiende a ∞. De esta forma se puede obtener una dirección del vector extremo definido por el vector ℓ_{jk} y que puede ser expresado por

$$\ell_{ji} = (-Y_{jk}, 1, 0) \geqslant 0$$

La regla deducida será la siguiente:

Si la maximización de uno de los p subproblemas nos lleva a un programa máximo infinito, definiendo una dirección ℓ_{ji} de un vector extremo, introducimos en la base de T_m la columna correspondiente a m_{ji} y su costo e_{ji} correspondiente los cuales están definidos por (14) y (15).

1.6 INICIACION DEL ALGORITMO

Para empezar el algoritmo de descomposición tenemos que encontrar un programa básico inicial y para esto, podemos utilizar el método de las variables artificiales.

Serán determinados (p + 1) programas iniciales, esto es, un programa inicial para cada uno de los p subproblemas y 1 para el master program.

Si los problemas no tienen soluciones iniciales obvias, aplicamos el método de las variables artificiales primero a los p subproblemas y luego al master program. Si uno de los (p + 1) problemas no tiene solución inicial, podemos concluir que el problema original dado no tiene un programa viable.

CAPITULO II

1. SUBRUTINAS PARA EL ALGORITMO DE DESCOMPOSICION

A continuación describimos las subrutinas que serán utilizadas con el objeto de elaborar un programa para resolver el algoritmo de descomposición.

2.1 SUBRUTINA RESEX

La subrutina RESEX utiliza el método Simplex Revisado para optimizar problemas de programación lineal.

El almacenamiento del cuadro Simplex y de otras matrices será en un arreglo unidimensional. Debido a la limitación del lenguaje FORTRAN, el segundo miembro de las restricciones se almacenará en la columna cero y el vector costos en la fila cero. Todos los coeficientes de las restricciones estarán con el signo cambiado.

Debido al arreglo unidimensional se calculará los índices de cada término con dos parámetros que posteriormente describiremos, así a un elemento

$a(i,k)$ corresponderá un $a(I*ZSCHR+K*SSCHR+1)$

Tenemos el problema

$$\max f(x) = C_j X_j$$

tal que

$$\sum_j P_j X_j \leqslant b$$

Suponemos que una solución viable básica ha sido determinada por el método de las variables artificiales y que solamente tenemos restricciones de la forma

$$b + \sum_j P_j X_j \geqslant 0 \quad , \; b \geqslant 0$$

entonces tenemos que

A = matriz de los coeficientes de las restricciones incluyendo el vector b y los C_j. Su dimensión será $(M + 1)*(N + 1)$

N = número de variables independientes (variables no básicas)

M = número de variables dependientes (variables básicas)

ZSCHR = paso de fila, distancia en la memoria entre dos elementos consecutivos de una columna.

SSCHR = paso de columna, distancia en la memoria de dos elementos consecutivos de una fila.

PROT 1 contiene los índices de las variables no básicas

PROT 2 contiene los índices de las variables básicas

B = matriz identidad generada en el programa y donde posteriormente se almacenará la inversa y el vector dual. Esta matriz, no está incluida en el arreglo de los coeficientes y por tanto las variables no básicas iniciales toman los índices de 1 hasta N y las variables básicas iniciales toman los índices N + 1 hasta N + M. Al final del cálculo se intercambian y tanto PROT 1 como PROT 2 almacenan los índices de las variables no básicas y básicas en cada iteración, al final las variables χ obtendrán los siguientes resultados:

variable en PROT 1; actual variable no básica, valor 0

variable en PROT 2; actual variable básica y será igual al valor de b_i

C = elementos de la columna pivot

FALL este parámetro nos proporciona información al fi

nal del cálculo, dependiendo del problema, asi:

FALL = 0 el problema tiene una solución finita

FALL = 1 el problema no tiene una solución finita

LIST 5 = localización del elemento inicial de la k^a forma parcial en la matriz que contiene a todas las restricciones de los subproblemas.

L1 = lista de los índices de las columnas de A que en una dada iteración no pertenecen a la base

L2 = lista de los índices de las columnas de B que para una iteración dada no pertenecen a la base

L3 = lista de los índices de las columnas que en una iteración dada pertenecen a la base.

2.2 SUBRUTINA MP8

Esta subrutina es utilizada para encontrar el elemento máximo de una lista dada de elementos. Y contiene los siguientes parámetros.

A = matriz en la cual están incluidos los números entre los cuales se va a buscar el número mayor

ZNR = índice de la linea de A en la que están los elementos de la lista.

ZSCHR = paso de linea en la matriz A

SSCHR = paso de columna en la matriz A

KP = índice de la columna de A a la cual pertenece el elemento mayor de la lista

L1 = lista de los índices de las columnas de A entre los cuales se va a buscar el número mayor.

L10 = número de columnas de A

MAX = mayor elemento de la lista

LIST 5 = con el significado anterior

Para iniciar el programa se escoge el primer elemento de la lista como el máximo provisorio, si la lista contiene un solo elemento termina el programa.

Luego se compara sucesivamente el elemento escogido como el máximo provisorio con cada uno de los elementos siguientes de la lista y si el elemento comparado es mayor, se lo substituye como máximo.

2.3 SUBRUTINAS MP10 y MP11

La subrutina MP10 sirve para calcular en cada iteración la matriz inversa correspondiente, a partir de los datos iniciales de las columnas que pertenecen a la base.

La subrutina MP11 es utilizada para transformar el vector b, es decir, para en cada iteración calcular la solución viable básica.

Estas dos subrutinas contienen los mismos parámetros diferenciándose solamente en que la subrutina MP11 posee un parámetro mas que es LIST 5 y que tiene el significado anteriormente descrito.

Los parámetros y su significado son:

B = matriz cuyos elementos van a ser transformados

C = vector de la columna básica

M = número de filas de la matriz B

IP = índice de la fila pivot

S = número de columnas a ser transformadas

ZSCH1 = paso de fila de la matriz B

SSCH1 = paso de columna de la matriz B

2.4 PROGRAMA PARA EL ALGORITMO DE DESCOMPOSICION

El programa para el algoritmo de descomposición utilizará las subrutinas RESEX, PRVEC y en esta la subrutina MP7 cuyos parámetros son los mismos de MP8 a escepción de LIST 5. La subrutina PRVEC posee casí los mismos parámetros

de RESEX, ya que su objetivo es optimizar el master program y obtener el dual correspondiente.

El problema es

$$\text{Min } f(x) = \sum_{j=1}^{p} C_j X_j$$

tal que

$$\sum_{j=1}^{p} \bar{A}_j X_j \leqslant b_o$$

$$A_j X_j \leqslant b_j$$

$$X_j \geqslant 0$$

Estos datos serán almacenados en dos arreglos unidimensionales A y B de la siguiente manera:

A =

0	C_1	C_2	.....	C_p
b_o	$\bar{A}_1$	$\bar{A}_2$	.....	$\bar{A}_p$

B =

0	0.....0	0	0.....0	.....	0	0.....0
b_1	A_1	b_2	A_2	.....	b_p	A_p

En el arreglo B, los subproblemas son almacenados consecutivamente con la finalidad de ahorrar espacio.

Por el principio de descomposición el problema original se transforma en T en el que están ahora incluidos los vectores costo y el segundo miembro de las restricciones. En cada iteración escogemos un vector W, que proviene

$$W = \begin{bmatrix} c_{ji} \\ t_{ji} \\ 0 \\ 1 \\ \vdots \\ 0 \end{bmatrix}$$

de uno de los problemas parciales.

Con objeto de calcular la solución del problema original en cada iteración colocamos en C todos los vectores de los problemas parciales que han entrado en el master program y en LA las soluciones correspondientes al master program.

La lista de parámetros y su significado es como sigue

A	corresponde a la tabla A, la cual incluye el vector costos y el segundo miembro de las restricciones
ZSCHR =	paso de fila de A
SSCHR =	paso de columna de A
M =	número de restricciones de la tabla A
N =	número de formas parciales de B
I1 =	número total de filas del problema original
I2 =	número de columnas del problema original
B	corresponde a la tabla B consistente de N formas parciales
LIST 1 =	número de columnas de la kª forma parcial
LIST 2 =	número de filas de la kª forma parcial
LIST 3 =	paso de fila de la kª forma parcial
LIST 4 =	paso de columna de la kª forma parcial
LIST 5 =	localización del elemento inicial de la kª forma parcial de B
LIST 6 =	localización del elemento inicial de $\bar{A}_k$
I3 =	el número mayor de filas de una forma parcial = = max(LIST 2(K))
I4 =	el número mayor de columnas de una forma parcial

= max (LIST 1 (K))

PROTO = resultado que indica el número de columnas de la k^a forma parcial contenidas en la solución

X = vector solución

P = vector de las variables duales del master program

BI = donde se almacena la matriz identidad de los problemas parciales

D = vector que contiene la función objetiva para un problema parcial dado

CP = columna pivot para una iteración dada

T = corresponde al master program

BT = matriz identidad del master program

PROT 3 contiene los índices de las variables no básicas del master program

PROT 4 contiene los índices de las variables básicas del master program

ZSCH1 = paso de fila del master program

SSCH1 = paso de columna del master program

C = vectores de los problemas parciales que entraron en el master program

LA = soluciones del master program

KP = índice que indica de que forma parcial es el vector que entra en el master program

FALL = 0 el problema tiene solución finita

FALL = 1 el problema no tiene solución finita

LIST 7 = localización en C del elemento inicial del conjunto de vectores de la k^{a} forma parcial, que entraron en el master program

Y = contiene las soluciones de cada uno de los problemas parciales, en una iteración dada

V = contiene los b_j de los problemas parciales

2.5 EJEMPLO

Como aplicación del algoritmo de descomposición maximizamos el siguiente ejemplo: [5]

$$\max f(x) = 18 + X_1 + 8X_2 + \frac{1}{2} X_3 + X_4$$

sujeto a

$$\begin{aligned}
X_1 + 4X_2 + \tfrac{7}{2} X_3 + \tfrac{1}{2} X_4 &\leqslant 1 \\
2X_1 + 3X_2 &\leqslant 6 \\
5X_1 + X_2 &\leqslant 5 \\
3X_3 - X_4 &\leqslant 12 \\
-3X_3 + X_4 &\leqslant 0 \\
X_3 &\leqslant 4 \\
X_1, X_2, X_3, X_4 &\geqslant 0
\end{aligned}$$

El programa utilizado y los resultados obtenidos constan en el apéndice

A P E N D I C E

Programa Fortran para el

Algoritmo de Descomposición

```
// JOB T  00FF 10FF                                    A 63                170176823

LOG DRIVE   CART SPEC   CART AVAIL  PHY DRIVE
  0000        00FF        00FF        0000
  0001        10FF        10FF        0001
                          20FF        0002

V2 M05   ACTUAL 32K  CONFIG 32K

// FOR
* LIST SOURCE PROGRAM
* ONE WORD INTEGERS
C
C SUBRUTINA PARA DETERMINAR EL ELEMENTO MAXIMO DE UNA LISTA
C
      SUBROUTINE MP7(A,J1,ZNR,ZSCHR,SSCHR,KP,L1,L10,JL1,MAX)
      INTEGER ZNR, ZSCHR,SSCHR
      REAL MAX
      DIMENSION A(1),L1(1)
      KH=ZNR*ZSCHR+L1(1)*SSCHR+1
      MAX=A(KH)
      KP=L1(1)
      IF(L10-2)2,3,3
    3 DO 1 K=2,L10
      KH=ZNR*ZSCHR+L1(K)*SSCHR+1
      IF(A(KH)-MAX)1,1,4
    4 MAX=A(KH)
      KP=L1(K)
    1 CONTINUE
    2 RETURN
      END

FEATURES SUPPORTED
 ONE WORD INTEGERS

CORE REQUIREMENTS FOR MP7
 COMMON      0  VARIABLES      6  PROGRAM    144

 END OF COMPILATION

// DUP

*STORE      WS  UA  MP7
CART ID 00FF   DB ADDR  2B4F   DB CNT   000A

// FOR
* LIST SOURCE PROGRAM
* ONE WORD INTEGERS
C  SUBRUTINA PARA DETERMINAR EL ELEEMENTO MAXIMO DE UNA LISTA L1
      SUBROUTINE MP8(A,J1,ZNR,ZSCHR,SSCHR,KP,L1,L10,JL1,MAX,LIST5)
      INTEGER ZNR, ZSCHR,SSCHR
      REAL MAX
      DIMENSION A(1),L1(1)
      KH=ZNR*ZSCHR+L1(1)*SSCHR+1+LIST5
      MAX=A(KH)
      KP=L1(1)
      IF(L10-2)2,3,3
    3 DO 1 K=2,L10
      KH=ZNR*ZSCHR+L1(K)*SSCHR+1+LIST5
      IF(A(KH)-MAX)1,1,4
    4 MAX=A(KH)
```

```
      KP=L1(K)
    1 CONTINUE
    2 RETURN
      END

FEATURES SUPPORTED
 ONE WORD INTEGERS

CORE REQUIREMENTS FOR MP8
 COMMON      0  VARIABLES      6  PROGRAM    152

 END OF COMPILATION

// DUP

*STORE      WS  UA  MP8
CART ID 00FF   DB ADDR  2B59   DB CNT   000A

// FOR
* LIST SOURCE PROGRAM
* ONE WORD INTEGERS
C
C SUBRUTINA PARA DETERMINAR EL ELEMENTO MAXIMO DE UNA LISTA
C
C
C COMPUTO DE LA MATRIZ INVERSA EN EL METODO SIMPLEX REVISADO
C
      SUBROUTINE MP10(B,JB,C,JC,M,IP,KS,ZSCH1,SSCH1)
      INTEGER ZSCH1,SSCH1,KS
      DIMENSION B(1),C(1)
      DO 2 K=1,KS
      KH=IP*ZSCH1+K*SSCH1+1
      IF(B(KH))5,2,5
    5 IM=M+1
      DO 3 II=1,IM
      I=II-1
      IF(I-IP)6,3,6
    6 IF(C(II))7,3,7
    7 KH=K*SSCH1+1
      KHO=KH+I*ZSCH1
      KH1=KH+IP*ZSCH1
      B(KHO)=B(KHO)+B(KH1)*C(II)
   3  CONTINUE
   2  CONTINUE
      DO 4 K=1,KS
      KH=IP*ZSCH1+K*SSCH1+1
   4  B(KH)=B(KH)*C(IP+1)
      RETURN
      END

FEATURES SUPPORTED
 ONE WORD INTEGERS

CORE REQUIREMENTS FOR MP10
 COMMON      0  VARIABLES     12  PROGRAM    230

 END OF COMPILATION

// DUP

*STORE      WS  UA  MP10
```

```
CART ID 00FF   DB ADDR  2B63   DB CNT   000F

// FOR
* LIST SOURCE PROGRAM
* ONE WORD INTEGERS
C
C COMPUTO DE LA PRIMERA COLUMNA DE A
C
      SUBROUTINE MP11(B,JB,C,JC,M,IP,KS,ZSCH1,SSCH1,LIST5)
      INTEGER ZSCH1,SSCH1,KS
      DIMENSION B(1),C(1)
      DO 2 K=1,KS
      KH=IP*ZSCH1+K*SSCH1+1+LIST5
      IF(B(KH))5,2,5
    5 IM=M+1
      DO 3 II=1,IM
      I=II-1
      IF(I-IP)6,3,6
    6 IF(C(II))7,3,7
    7 KH=K*SSCH1+1+LIST5
      KHO=KH+I*ZSCH1
      KH1=KH+IP*ZSCH1
      B(KHO)=B(KHO)+B(KH1)*C(II)
   3  CONTINUE
   2  CONTINUE
      DO 4 K=1,KS
      KH=IP*ZSCH1+K*SSCH1+1+LIST5
   4  B(KH)=B(KH)*C(IP+1)
      RETURN
      END

FEATURES SUPPORTED
 ONE WORD INTEGERS

CORE REQUIREMENTS FOR MP11
 COMMON      0  VARIABLES     12  PROGRAM    240

 END OF COMPILATION

// DUP

*STORE      WS  UA  MP11
CART ID 00FF   DB ADDR  2B72   DB CNT   0010

// FOR
* LIST SOURCE PROGRAM
* ONE WORD INTEGERS
C
C  METODO SIMPLEX REVISADO PARTIENDO DE UNA SOLUCION VIABLE

      SUBROUTINE RESEX(A,J1,B,JB,C,JC,N,M,ZSCHR,SSCHR,PROT1,IPRO1,
     1 PROT2,IPRO2,FALL,L1,JL1,L2,JL2,L3,JL3,LIST5)
      INTEGER PROT1(1),PROT2(1),FALL,ZSCHR,SSCHR,KS,ZSCH1,SSCH1
      REAL MAX
      DIMENSION A(1),B(1),C(1),L1(1),L2(1),L3(1)

C  INDICES DE LAS VARIABLES NO BASICAS

      DO 1 K=1,N
      L1(K)=K
    1 PROT1(K)=K
```

```
C INDICES DE LAS VARIABLES BASICAS

      DO 2 I=1,M
    2 PROT2(I)=N+I
      L10=N
      L20=0
      ZSCH1=1
      SSCH1=M+1
      ZSCHR=1
      SSCHR=M+1

C GENERACION DE LA MATRIZ IDENTIDAD

      M1=M+1
      DO 3 I=1,M1
      DO 3 K=1,M
      KH=(I-1)*ZSCH1+K*SSCH1+1
      B(KH)=0.
      IF(K-(I-1))3,11,3
   11 B(KH)=-1.
    3 CONTINUE
C
C  DETERMINACION DE LA COLUMNA PIVOT
C
  100 MAX=0
      Q1=0
      IF(L10)120,200,120
  120 CALL MP8(A,J1,0,ZSCHR,SSCHR,KP,L1,L10,JL1,MAX,LIST5)
      WRITE(5,901)MAX,KP
  901 FORMAT(7X,'MAX=',F10.4/7X,'KP=',I4)
  200 IF(L20)210,300,210
  210 DO 221 I=1,L20
      KH=L2(I)*SSCH1+1
      IF(B(KH)-Q1)221,221,220
  220 KS=I
      Q1=B(KH)
  221 CONTINUE
  300 IF(MAX-0.39333E-02)301,301,303
  301 IF(Q1-0.39333E-02)302,302,303
  302 FALL=0
      WRITE(5,902)FALL
  902 FORMAT(7X,'SOLUCION FINITA FALL=',I2)
      WRITE(5,903)A(IN)
  903 FORMAT(7X,'SOLUCION OPTIMA=',F10.4)
      GO TO 900
  303 K=0
      IF(MAX-Q1)304,400,400
  304 K=1
      KP=L3(KS)
      DO 305 II=1,M1
      I=II-1
      KH=I*ZSCH1+KS*SSCH1+1
      C(II)=-B(KH)
  305 CONTINUE
      GO TO 500
  400 KH=KP*SSCHR+1+LIST5
      C(1)=-A(KH)
      DO 401 I=1,M
      Q1=0.
      DO 402 J=1,M
```

```
      KH1=KH+J*ZSCHR
      KH2=I*ZSCH1+J*SSCH1+1
      IF(A(KH1))403,402,403
  403 IF(B(KH2))404,402,404
  404 Q1=Q1+A(KH1)*B(KH2)
  402 CONTINUE
      C(I+1)=Q1
  401 CONTINUE
C
C  DETERMINACION DE LA FILA PIVOT
C
  500 IP=0
      DO 501 I=1,M
      IF(C(I+1))501,501,502
  501 CONTINUE
      FALL=1
      WRITE(5,904)FALL
  904 FORMAT(7X,'SOLUCION INFINITA FALL=',I2)
      GO TO 900
  502 KH=I*ZSCHR+1+LIST5
      Q1=A(KH)/C(I+1)
      IP=I
      J=IP
      DO 601 I=J,M
      IF(C(I+1))601,601,602
  602 KH=I*ZSCHR+1+LIST5
      IF(A(KH)/C(I+1)-Q1)603,601,601
  603 Q1=A(KH)/C(I+1)
      IP=I
  601 CONTINUE
      WRITE(5,905)IP
  905 FORMAT(7X,'IP=',I4)
C
C DIVISION DE LA COLUMNA PIVOT PARA EL ELEMENTO PIVOT
C
      C(IP+1)=1./C(IP+1)
      DO 604 II=1,M1
      IF((II-1)-IP)605,604,605
  605 C(II)=-C(II)*C(IP+1)
  604 CONTINUE
C
C  TRANSFORMACION DE LA PRIMERA COLUMNA DE A
C
      CALL MP11 (A,J1,C,JC,M,IP,1,ZSCH1,0,LIST5)
      IN=LIST5+1

C TRANSFORMACION DE LA PRIMERA FILA DE A
C
      DO 607 J=1,N
      IF(J-KP)606,607,606
  606 Q1=0.
      DO 608 I=1,M
      KH=IP*ZSCH1+I*SSCH1+1
      KH1=I*ZSCH1+J*SSCH1+1+LIST5
      IF(B(KH))609,608,609
  609 Q1=Q1+B(KH)*A(KH1)
  608 CONTINUE
      KH=J*SSCHR+1+LIST5
      A(KH)=A(KH)-Q1*C(1)
  607 CONTINUE
      KH=KP*SSCHR+1+LIST5
```

```
      A(KH)=-C(1)
C
C  TRANSFORMACION DE LA INVERSA
C
      CALL MP10(B,JB,C,JC,M,IP,M,ZSCH1,SSCH1)
      IF(K)708,610,708
  610 IF(L20-1)612,611,611
  611 DO 613 J=1,L20
      IF(IP-L2(J))613,614,613
  613 CONTINUE
      GO TO 612
  614 IF(L3(J)-KP)615,800,615
  615 IF(L10-1)617,616,616
  616 DO 618 I=1,L10
      IF(L1(I)-KP)618,619,618
  619 L1(I)=L3(J)
  618 CONTINUE
  617 L3(J)=KP
      GO TO 800
  612 L20=L20+1
      L2(L20)=IP
      L3(L20)=KP
      IF(L10-1)800,701,701
  701 DO 702 I=1,L10
      IF(L1(I)-KP)702,704,702
  702 CONTINUE
      GO TO 800
  704 L10=L10-1
      DO 703 J=I,L10
  703 L1(J)=L1(J+1)
      GO TO 800
  708 IF(L20-1)707,709,709
  709 DO 705 J=1,L20
      IF(IP-L2(J))705,800,705
  705 CONTINUE
  707 L10=L10+1
      L1(L10)=L3(KS)
      L20=L20-1
      DO 706 J=KS,L20
      L2(J)=L2(J+1)
  706 L3(J)=L3(J+1)
  800 K=PROT1(KP)
      PROT1(KP)=PROT2(IP)
      PROT2(IP)=K
      GO TO 100
  900 RETURN
      END

FEATURES SUPPORTED
 ONE WORD INTEGERS

CORE REQUIREMENTS FOR RESEX
 COMMON      0  VARIABLES     24  PROGRAM   1432

 END OF COMPILATION

// DUP

*STORE      WS  UA  RESEX
CART ID 00FF   DB ADDR  2B82   DB CNT   005A
```

```
// FOR
* LIST SOURCE PROGRAM
* ONE WORD INTEGERS
C
C  SUBRUTINA PARA OPTIMIZAR EL MASTER PROGRAM Y CALCULAR EL VECTOR DUAL
C
      SUBROUTINE PRVEC(A,J1,B,JB,C,JC,N,M,ZSCHR,SSCHR,FALL,PROT1,IPRO1,
     1PROT2,IPRO2,L1,JL1,L2,JL2,L3,JL3,P,NP,LA,NLA,IIP,KKP,
     1 LIST1,I3)
      INTEGER PROT1(1),PROT2(1),FALL,ZSCHR,SSCHR,KS,ZSCH1,SSCH1
      REAL MAX,LA(1)
      DIMENSION A(1),B(1),C(1),L1(1),L2(1),L3(1),P(1)
      L10=N
      L20=0
      ZSCH1=1
      SSCH1=M+1
      ZSCHR=1
      SSCHR=M+1
      M1=M+1
C
C  DETERMINACION DE LA COLUMNA PIVOT
C
  100 MAX=0
      Q1=0
      IF(L10)120,200,120
  120 CALL MP7(A,J1,0,ZSCHR,SSCHR,KP,L1,L10,JL1,MAX)
      WRITE(5,901)MAX,KP
  901 FORMAT(7X,'MAX=',F10.4/7X,'KP=',I4)
  200 IF(L20)210,300,210
  210 DO 221 I=1,L20
      KH=L2(I)*SSCH1+1
      IF(B(KH)-Q1)221,221,220
  220 KS=I
      Q1=B(KH)
  221 CONTINUE
  300 IF(MAX-0.39333E-02)301,301,303
  301 IF(Q1-0.39333E-02)302,302,303
  302 FALL=0
      WRITE(5,902)FALL
  902 FORMAT(7X,'SOLUCION FINITA FALL=',I2)
      WRITE(5,903)A(1)
  903 FORMAT(7X,'SOLUCION OPTIMA=',F10.4)
C
C ORDENACION DE LA SOLUCION DE MASTER PROGRAM
C
      DO 334 I=1,N
  334 LA(I+1)=0.
      DO 335 I=1,M
      KH=PROT2(I)
      KH1=I*ZSCH1+1
      IF(KH-N)336,336,335
  336 LA(KH+1)=A(KH1)
  335 CONTINUE
      WRITE(5,810)(LA(I),I=2,7)
  810 FORMAT(5X,'SOLUCION LA(I)'/(6F10.4))

C COLOCACION DEL VECTOR DUAL

      DO 337 I=1,M
      KH=I*SSCH1+1
  337 P(I)=B(KH)
```

```
      IIP=IIP+1
      WRITE(5,811)(P(I),I=1,3)
  811 FORMAT(10X,'P(I)'/(3F10.4))
      GO TO 900
  303 K=0
      IF(MAX-Q1)304,400,400
  304 K=1
      KP=L3(KS)
      DO 305 II=1,M1
      I=II-1
      KH=I*ZSCH1+KS*SSCH1+1
      C(II)=-B(KH)
  305 CONTINUE
      GO TO 500
  400 KH=KP*SSCHR+1
      C(1)=-A(KH)
      DO 401 I=1,M
      Q1=0.
      DO 402 J=1,M
      KH1=KH+J*ZSCHR
      KH2=I*ZSCH1+J*SSCH1+1
      IF(A(KH1))403,402,403
  403 IF(B(KH2))404,402,404
  404 Q1=Q1+A(KH1)*B(KH2)
  402 CONTINUE
      C(I+1)=Q1
  401 CONTINUE
C
C  DETERMINACION DE LA FILA PIVOT
C
  500 IP=0
      DO 501 I=1,M
      IF(C(I+1))501,501,502
  501 CONTINUE
      FALL=1
      WRITE(5,904)FALL
  904 FORMAT(7X,'SOLUCION INFINITA FALL=',I2)
      GO TO 900
  502 KH=I*ZSCHR+1
      Q1=A(KH)/C(I+1)
      IP=I
      J=IP
      DO 601 I=J,M
      IF(C(I+1))601,601,602
  602 KH=I*ZSCHR+1
      IF(A(KH)/C(I+1)-Q1)603,601,601
  603 Q1=A(KH)/C(I+1)
      IP=I
  601 CONTINUE
      WRITE(5,905)IP
  905 FORMAT(7X,'IP=',I4)
      C(IP+1)=1./C(IP+1)
      DO 604 II=1,M1
      IF((II-1)-IP)605,604,605
  605 C(II)=-C(II)*C(IP+1)
  604 CONTINUE
C
C  TRANSFORMACION DE LA PRIMERA COLUMNA DE A
C
      CALL MP10 (A,J1,C,JC,M,IP,1,ZSCH1,0)
C
```

```
C TRANSFORMACION DE LA PRIMERA FILA DE A
C
      DO 607 J=1,N
      IF(J-KP)606,607,606
  606 Q1=0.
      DO 608 I=1,M
      KH=IP*ZSCH1+I*SSCH1+1
      KH1=I*ZSCH1+J*SSCH1+1
      IF(B(KH))609,608,609
  609 Q1=Q1+B(KH)*A(KH1)
  608 CONTINUE
      KH=J*SSCHR+1
      A(KH)=A(KH)-Q1*C(1)
  607 CONTINUE
      KH=KP*SSCHR+1
      A(KH)=-C(1)
C
C  TRANSFORMACION DE LA INVERSA
C
      CALL MP10(B,JB,C,JC,M,IP,M,ZSCH1,SSCH1)
      IF(K)708,610,708
  610 IF(L20-1)612,611,611
  611 DO 613 J=1,L20
      IF(IP-L2(J))613,614,613
  613 CONTINUE
      GO TO 612
  614 IF(L3(J)-KP)615,800,615
  615 IF(L10-1)617,616,616
  616 DO 618 I=1,L10
      IF(L1(I)-KP)618,619,618
  619 L1(I)=L3(J)
  618 CONTINUE
  617 L3(J)=KP
      GO TO 800
  612 L20=L20+1
      L2(L20)=IP
      L3(L20)=KP
      IF(L10-1)800,701,701
  701 DO 702 I=1,L10
      IF(L1(I)-KP)702,704,702
  702 CONTINUE
      GO TO 800
  704 L10=L10-1
      DO 703 J=I,L10
  703 L1(J)=L1(J+1)
      GO TO 800
  708 IF(L20-1)707,709,709
  709 DO 705 J=1,L20
      IF(IP-L2(J))705,800,705
  705 CONTINUE
  707 L10=L10+1
      L1(L10)=L3(KS)
      L20=L20-1
      DO 706 J=KS,L20
      L2(J)=L2(J+1)
  706 L3(J)=L3(J+1)
  800 K=PROT1(KP)
      PROT1(KP)=PROT2(IP)
      PROT2(IP)=K
      GO TO 100
  900 RETURN
```

```
      END

FEATURES SUPPORTED
 ONE WORD INTEGERS

CORE REQUIREMENTS FOR PRVEC
 COMMON      0  VARIABLES     24  PROGRAM   1474

 END OF COMPILATION

// DUP

*STORE      WS  UA  PRVEC
CART ID 00FF   DB ADDR  2BDC   DB CNT   005D

// FOR
* LIST SOURCE PROGRAM
* ONE WORD INTEGERS
* IOCS(2501READER,1403PRINTER)
C
C METODO DE DESCOMPOSICION DE DANTIZIG-WOLFE PARA PROBLEMAS DE
C PROGRAMACION LINEAL
C FELIX EDUARDO VACA OBANDO
C
      INTEGER R,S,ZSCHR,SSCHR,ZSCH1,SSCH1,PROTO(3),PROT1(2),PROT2(3),
     1 FALL,PROT3(6),PROT4(3)
      REAL MAX,LA(7)
      DIMENSION LIST1(2),LIST2(2),LIST3(2),LIST4(2),LIST5(2),A(10),
     1 B(21),X(5),P(3),BI(16),D(4),CP(4),LI(3),L2(3),L3(3),L1(2),
     1 T(28),C(14),LIST6(2),BT(16),LL(6),V(7),Y(6),LIST7(2)
      READ(8,10)(LIST1(K),K=1,2),(LIST2(K),K=1,2),(LIST3(K),K=1,2),
     1 (LIST4(K),K=1,2),(LIST5(K),K=1,2),(L1(K),K=1,2),(LIST6(K),K=1,2),
     1 (LIST7(K),K=1,2)
   10 FORMAT(2I10)
      READ(8,11)(A(I),I=1,10)
   11 FORMAT(2F10.4)
      READ(8,12)(B(I),I=1,9)
   12 FORMAT(3F10.4)
      READ(8,13)(B(I),I=10,21)
   13 FORMAT(4F10.4)
      READ(8,14)(V(I),I=1,7)
   14 FORMAT(7F10.4)
      WRITE(5,20)(LIST1(K),K=1,2)
      WRITE(5,21)(LIST2(K),K=1,2)
      WRITE(5,22)(LIST3(K),K=1,2)
      WRITE(5,23)(LIST4(K),K=1,2)
      WRITE(5,24)(LIST5(K),K=1,2)
      WRITE(5,25)(LIST6(K),K=1,2)
      WRITE(5,26)(LIST7(K),K=1,2)
      WRITE(5,27)(L1(K),K=1,2)
   20 FORMAT(7X,'LIST1(K)=',2I3)
   21 FORMAT(7X,'LIST2(K)=',2I3)
   22 FORMAT(7X,'LIST3(K)=',2I3)
   23 FORMAT(7X,'LIST4(K)=',2I3)
   24 FORMAT(7X,'LIST5(K)=',2I3)
   25 FORMAT(7X,'LIST6(K)=',2I3)
   26 FORMAT(7X,'LIST7(K)=',2I3)
   27 FORMAT(7X,'L1(K)=',2I3)
      WRITE(5,28)(A(I),I=1,10)
   28 FORMAT(7X,'MATRIZ A'/(2F10.4))
```

```
      WRITE(5,29)(V(I),I=1,7)
   29 FORMAT(15X,'V(I)'/(3F10.4,4F10.4))
      WRITE(5,30)(B(I),I=1,9)
   30 FORMAT(5X,'PRIMER PROBLEMA PARCIAL'/(3F10.4))
      WRITE(5,31)(B(I),I=10,21)
   31 FORMAT(5X,'SEGUNDO PROBLEMA PARCIAL'/(4F10.4))
      I2=4
      I3=2
      M=1
      N=2
      ZSCHR=1
      SSCHR=2
      IP=0
      ZSCH1=1
      SSCH1=M+N+1
      MC=(M+N)*N
      M2=M+N
      M3=I3*N
      DO 50 I=1,M3
   50 Y(I)=0.
C
C INDICES DE LAS VARIABLES BASICAS Y NO BASICAS DEL MASTER PROGRAM
C
      DO 51 K=1,MC
   51 LL(K)=K
      IA=1
      IS=0
      DO 52 K=1,MC
      MT=1+IS
      IF(K-MT)54,53,54
   53 KH=M+IA
      PROT4(KH)=K
      IS=IS+M2
      IA=IA+1
      GO TO 52
   54 PROT3(K)=K
   52 CONTINUE
      IS=0
      DO 55 K=1,M2
      IF(K-M)57,57,56
   56 KH=1+IS
      PROT3(KH)=MC+K
      IS=IS+M2
      GO TO 55
   57 PROT4(K)=MC+K
   55 CONTINUE

C GENERACION DE LA MATRIZ IDENTIDAD DEL MASTER PROGRAM

      M1=M+N+1
      DO 77 I=1,M1
      DO 77 K=1,M2
      KH=(I-1)*ZSCH1+K*SSCH1+1
      BT(KH)=0.
      IF(K-(I-1))77,78,77
   78 BT(KH)=-1.
   77 CONTINUE
      I6=I3+1
      I7=((M+N)*N+1)*I3
      DO 80 I=I6,I7
   80 C(I)=0.
```

```
      DO 81 I=1,M2
   81 P(I)=0.
      DO 82 I=1,N
   82 PROTO(I)=1
      M3=(M+N+1)*(1+N*(M+N))
      DO 83 I=1,M3
   83 T(I)=0.
      DO 84 K=1,N
      II=K-1
      LR=II*M2
      KH=(LR+1)*SSCH1+M+K+1
   84 T(KH)=-1.

C VECTOR (B CERO,1,1..1) EN T
C
      M2=M+1
      DO 85 I=1,M2
   85 T(I)=A(I)
      LIS=M+N
      M3=M+2
      M4=LIS+1
      DO 86 I=M3,M4
   86 T(I)=1.
C
C CALCULO DE P*A Y DE LA NUEVA FUNCION OBJETIVA C-P*A
C
   90 MAX=0.
      R=0
      IS=0
      DO 91 K=1,N
      II=LIST2(K)+1
      DO 92 I=1,II
      KH=LIST5(K)+I
      KH1=I+IS
   92 B(KH)=V(KH1)
      IS=IS+II
   91 CONTINUE
      DO 93 K=1,N
      M1=LIST1(K)
      DO 94 I=1,M1
      Q=0.
      DO 95 S=1,M
      KH=S*ZSCHR+(R+I)*SSCHR+1
   95 Q=Q+P(S)*A(KH)
      KH=R+I
      KH1=LIST5(K)+I*LIST4(K)+1
      KH2=KH*SSCHR+1
      D(KH)=A(KH2)-Q
   94 B(KH1)=D(KH)
      WRITE(5,32)
   32 FORMAT(5X,'MATRIZ B CON NUEVA FUNCION OBJETIVA')
      WRITE(5,30)(B(I),I=1,9)
      WRITE(5,31)(B(I),I=10,21)
      CALL RESEX(B,21,BI,16,CP,4,LIST1(K),LIST2(K),LIST3(K),LIST4(K),
     1 PROT1,2,PROT2,3,FALL,LI,3,L2,3,L3,3,LIST5(K))
      IF(FALL)400,100,400
  100 DO 101 I=1,M1
  101 X(I+1)=0.
C
C SOLUCION OPTIMA DEL PROBLEMA K COLOCADA EN LUGAR APROPIADO
```

```
      M2=LIST2(K)
      DO 110 I=1,M2
      KH=PROT2(I)
      KH1=LIST5(K)+I*LIST3(K)+1
      IF(KH-M1)111,111,110
  111 X(KH+1)=B(KH1)
  110 CONTINUE
      M3=M1+1
      WRITE(5,33)(X(I),I=2,M3)
   33 FORMAT(7X,'SOLUCION DEL PROBLEMA PARCIAL'/(2F10.4))
      DO 120 I=1,M1
      KH=K*I3+I
  120 Y(KH)=X(I+1)
C
C CALCULO DE (II*A-C)*X+U
C
      Q=0.
      DO 130 I=1,M1
      KH=R+I
  130 Q=Q+X(I+1)*D(KH)
      WRITE(5,34)Q
   34 FORMAT(//7X,'Q=',F10.4)
      KH=M+K
      Q=Q-P(KH)
      R=R+L1(K)
      IF(Q-MAX)93,93,131
  131 MAX=Q
      KP=K
   93 CONTINUE
      WRITE(5,35)KP
   35 FORMAT( 7X,'KP=',I3)
      WRITE(5,36)MAX
   36 FORMAT( 7X,'MAX=',F10.4)
C
C  ORDENACION DE LOS VECTORES X EN  C
C
      IF(MAX)140,202,140
  140 IM=PROTO(KP)-1
      LOT=IM+(KP-1)*LIS
      KH1=(LOT+2)*SSCH1+1
      T(KH1)=MAX
      M1=LIST1(KP)
      DO 141 I=1,M1
      KH=KP*I3+I
  141 X(I+1)=Y(KH)
      DO 142 I=1,M1
      KH=PROTO(KP)*I3+I+ LIST7(KP)
  142 C(KH)=X(I+1)
      WRITE(5,37)(C(I),I=3,14)
   37 FORMAT(10X,'C(I)',/(2F10.4))
C
C CALCULO DE LOS TIJ DE LA COLUMNA QUE ENTRA

      DO 150 I=1,M
      Q4=0.
      DO 151 J=1,M1
      JJ=J-1
      KH=LIST6(KP)+JJ*SSCHR+1+I*ZSCHR
  151 Q4=Q4+A(KH)*X(J+1)
      KH1=(LOT+2)*SSCH1+I*ZSCH1+1
  150 T(KH1)=Q4
```

```
      KH=KH1+KP
      T(KH)=-1.
      WRITE(5,38)(T(I),I=1,28)
   38 FORMAT(10X,'MATRIZ T'/(4F10.4))
      PROTO(KP)=PROTO(KP)+1
      CALL PRVEC(T,28,BT,16,CP,4,6,3,1,4,FALL,PROT3,6,PROT4,3,LL,6,L2,6,
     1 L3,6,P,3,LA,7,IP,KP,LIST1(KP),I3)
      IF(IP)200,201,200
  200 GO TO 90
  201 FALL=1
      WRITE(5,40)FALL
   40 FORMAT(7X,'SOLUCION INFINITA FALL=',I3)
      GO TO 400
  202 R=0
      KP=0
      FALL=0
      WRITE(5,39)FALL
   39 FORMAT(7X,'SOLUCION FINITA FALL=',I3)
C
C GENERACION DEL VECTOR X
C
      M1=I2+1
      DO 203 I=1,M1
  203 X(I)=0.
      DO 204 K=1,N
      IF(PROTO(K))205,210,205
  205 M2=LIST1(K)
      DO 206 I=1,M2
      M3=PROTO(K)
      DO 206 S=1,M3
      KH=I+R+1
      KH1=S+KP+1
      KH2=I*ZSCH1+(KP+S)*I3
  206 X(KH)=X(KH)+C(KH2)*LA(KH1)
  210 KP=KP+M+N
  204 R=R+L1(K)
      DO 211 I=1,I2
      KH=I*SSCHR+1
  211 X(1)=X(1)+X(I+1)*A(KH)
      X(1)=X(1)+A(1)
      WRITE(5,41)(X(I),I=2,5)
   41 FORMAT(7X,'VECTOR SOLUCION ORDENADO'/(4F10.4))
      WRITE(5,42)X(1)
   42 FORMAT(7X,'SOLUCION OPTIMA=',F10.4)
  400 CALL EXIT
      END

FEATURES SUPPORTED
 ONE WORD INTEGERS
 IOCS

CORE REQUIREMENTS FOR
 COMMON      0  VARIABLES    380  PROGRAM   2724

 END OF COMPILATION

// XEQ

      LIST1(K)=  2  2
      LIST2(K)=  2  3
      LIST3(K)=  1  1
```

```
    LIST4(K)=  3  4
    LIST5(K)=  0  9
    LIST6(K)=  2  6
    LIST7(K)=  2  8
    L1(K)=  2  2
    MATRIZ A
18.0000     1.0000
 1.0000    -1.0000
 8.0000    -4.0000
 0.5000    -3.5000
 1.5000    -0.5000
            V(I)
 0.0000     6.0000     5.0000     0.0000    12.0000     0.0000     4.0000
  PRIMER PROBLEMA PARCIAL
 0.0000     6.0000     5.0000
 0.0000    -2.0000    -5.0000
 0.0000    -3.0000    -1.0000
  SEGUNDO PROBLEMA PARCIAL
 0.0000    12.0000     0.0000     4.0000
 0.0000    -3.0000     3.0000    -1.0000
 0.0000     1.0000    -1.0000     0.0000
  MATRIZ B CON NUEVA FUNCION OBJETIVA
  PRIMER PROBLEMA PARCIAL
 0.0000     6.0000     5.0000
 1.0000    -2.0000    -5.0000
 8.0000    -3.0000    -1.0000
  SEGUNDO PROBLEMA PARCIAL
 0.0000    12.0000     0.0000     4.0000
 0.0000    -3.0000     3.0000    -1.0000
 0.0000     1.0000    -1.0000     0.0000
    MAX=     8.0000
    KP=    2
    IP=    1
    MAX=    -4.3333
    KP=    1
    SOLUCION FINITA FALL= 0
    SOLUCION OPTIMA=    16.0000
    SOLUCION DEL PROBLEMA PARCIAL
 0.0000     2.0000


    Q=    16.0000
  MATRIZ B CON NUEVA FUNCION OBJETIVA
  PRIMER PROBLEMA PARCIAL
16.0000     2.0000     3.0000
-4.3333    -2.0000    -5.0000
-2.6666    -3.0000    -1.0000
  SEGUNDO PROBLEMA PARCIAL
 0.0000    12.0000     0.0000     4.0000
 0.5000    -3.0000     3.0000    -1.0000
 1.5000     1.0000    -1.0000     0.0000
    MAX=     1.5000
    KP=    2
    IP=    2
    MAX=     5.0000
    KP=    1
    IP=    3
    SOLUCION FINITA FALL= 0
    SOLUCION OPTIMA=    20.0000
    SOLUCION DEL PROBLEMA PARCIAL
 4.0000    12.0000
```

```
    Q=    20.0000
    KP=   2
    MAX=    20.0000
       C(I)
 0.0000     0.0000
 0.0000     0.0000
 0.0000     0.0000
 0.0000     0.0000
 4.0000    12.0000
 0.0000     0.0000
       MATRIZ T
18.0000     1.0000     1.0000     1.0000
 0.0000     0.0000    -1.0000     0.0000
 0.0000     0.0000     0.0000     0.0000
 0.0000     0.0000     0.0000     0.0000
 0.0000     0.0000     0.0000    -1.0000
20.0000   -20.0000     0.0000    -1.0000
 0.0000     0.0000     0.0000     0.0000
    MAX=    20.0000
    KP=    5
    IP=    1
    MAX=     0.0000
    KP=    1
    SOLUCION FINITA FALL= 0
    SOLUCION OPTIMA=    19.0000
  SOLUCION LA(I)
 1.0000     0.0000     0.0000     0.9500     0.0500     0.0000
       P(I)
-1.0000     0.0000     0.0000
  MATRIZ B CON NUEVA FUNCION OBJETIVA
  PRIMER PROBLEMA PARCIAL
 0.0000     6.0000     5.0000
 0.0000    -2.0000    -5.0000
 4.0000    -3.0000    -1.0000
  SEGUNDO PROBLEMA PARCIAL
 0.0000    12.0000     0.0000     4.0000
-5.0000    -3.0000     3.0000    -1.0000
-1.5000     1.0000    -1.0000     0.0000
    MAX=     4.0000
    KP=    2
    IP=    1
    MAX=    -2.6666
    KP=    1
    SOLUCION FINITA FALL= 0
    SOLUCION OPTIMA=     8.0000
    SOLUCION DEL PROBLEMA PARCIAL
 0.0000     2.0000


    Q=     8.0000
  MATRIZ B CON NUEVA FUNCION OBJETIVA
  PRIMER PROBLEMA PARCIAL
 8.0000     2.0000     3.0000
-2.6666    -2.0000    -5.0000
-1.3333    -3.0000    -1.0000
  SEGUNDO PROBLEMA PARCIAL
 0.0000    12.0000     0.0000     4.0000
-3.0000    -3.0000     3.0000    -1.0000
 1.0000     1.0000    -1.0000     0.0000
    MAX=     1.0000
    KP=    2
    IP=    2
    MAX=     0.0000
```

```
    KP=    1
    SOLUCION FINITA FALL= 0
    SOLUCION OPTIMA=     0.0000
    SOLUCION DEL PROBLEMA PARCIAL
 0.0000     0.0000


    Q=     0.0000
    KP=   1
    MAX=     8.0000
       C(I)
 0.0000     0.0000
 0.0000     2.0000
 0.0000     0.0000
 0.0000     0.0000
 4.0000    12.0000
 0.0000     0.0000
       MATRIZ T
19.0000     0.0500     1.0000     0.9500
 0.0000     0.0000    -1.0000     0.0000
 8.0000    -8.0000    -1.0000     0.0000
 0.0000     0.0000     0.0000     0.0000
 0.0000     0.0000     0.0000    -1.0000
-1.0000   -20.0000     0.0000    -1.0000
 0.0000     0.0000     0.0000     0.0000
    MAX=     8.0000
    KP=    2
    IP=    1
    MAX=     0.0000
    KP=    1
    SOLUCION FINITA FALL= 0
    SOLUCION OPTIMA=    19.9999
  SOLUCION LA(I)
 0.8750     0.1250     0.0000     1.0000     0.0000     0.0000
       P(I)
-2.0000     0.0000     0.0000
  MATRIZ B CON NUEVA FUNCION OBJETIVA
  PRIMER PROBLEMA PARCIAL
 0.0000     6.0000     5.0000
-0.9999    -2.0000    -5.0000
 0.0000    -3.0000    -1.0000
  SEGUNDO PROBLEMA PARCIAL
 0.0000    12.0000     0.0000     4.0000
 0.0000    -3.0000     3.0000    -1.0000
-1.0000     1.0000    -1.0000     0.0000
    MAX=     0.0000
    KP=    2
    SOLUCION FINITA FALL= 0
    SOLUCION OPTIMA=     0.0000
    SOLUCION DEL PROBLEMA PARCIAL
 0.0000     0.0000


    Q=     0.0000
  MATRIZ B CON NUEVA FUNCION OBJETIVA
  PRIMER PROBLEMA PARCIAL
 0.0000     6.0000     5.0000
-0.9999    -2.0000    -5.0000
 0.0000    -3.0000    -1.0000
  SEGUNDO PROBLEMA PARCIAL
 0.0000    12.0000     0.0000     4.0000
-6.5000    -3.0000     3.0000    -1.0000
 0.5000     1.0000    -1.0000     0.0000
```

```
      MAX=     0.5000
      KP=    2
      IP=    2
      MAX=    -4.9999
      KP=    1
      SOLUCION FINITA FALL= 0
      SOLUCION OPTIMA=     0.0000
      SOLUCION DEL PROBLEMA PARCIAL
   0.0000     0.0000


      Q=     0.0000
      KP=   1
      MAX=     0.0000
      SOLUCION FINITA FALL=  0
      VECTOR SOLUCION ORDENADO
   0.0000     0.2500     0.0000     0.0000
      SOLUCION OPTIMA=    20.0000
// XEQ

// * 00382
```

BIBLIOGRAFIA

[1] G. B. DANTZIG - P. WOLFE

"The Decomposition Algorithm for Linear Programming", Operations Res., 8, Jan., Feb., Oct., 1960

[2] M. SIMONNARD

"Linear Programming", Prentice-Hall Inc., Englewood, N.J., 1966 - Sec. 10

[3] G. B. DANTZIG

"Linear Programming and Extensions", Princeton Univ. Press, Princeton, N.J., 1963, Chap. 23

[4] G. HADLEY

"Algebra Lineal", Bilingua, 1969

[5] H.P. KUNZI - H.G.TZSCHACH - C.A. ZEHNDER

"Numerical Methods of Mathematical Optimization", Academic Press Inc., 1968

[6] L.S. LASDON

"Optimization Theory for Large Systems", MacMillan Series for Operations Res., 1970, Chap. 3

[7] A. R. RESTREPO

Notas de aula de Programación Lineal, 1970

[8] S. I. GASS

"Programación Lineal", CECSA, 1967

[9] T. PACITTI

Fortran - Monitor - Principios, Ao Livro Técnico S/A 1970