Universidad Tecnológica de la Mixteca

Optimización del Problema de la Dieta Alimenticia : **D I E T E X**

Tesis Profesional

Que para obtener el Título de

Ingeniero en Computación

Presenta

URANIA RUIZ CRUZ

Acatlima, Huajuapan de León, Oaxaca Diciembre'98

Tesis Profesional presentada
el 10 de Diciembre de 1998
ante los sinodales:

Ing. Hugo Suárez Onofre
Dra. Virginia Berrón Lara
Lic. Tirso Miguel Angel Ramírez Solano
M.C. Luis René Marcial Castillo

Asesor:
M.C. Luis René Marcial Castillo

Coasesor:
Dra. Virginia Berrón Lara

A mi familia por su impulso constante
y porque no hay otra igual.

A mis padres porque siempre confiaron en mí
y me apoyaron moral y económicamente.

A Nere, Eucli y Riemann porque saben que
el esfuerzo lo realizamos todos juntos y porque
ahora les toca a ellos llegar a la meta.

Agradecimientos

A mi asesor por creer en mí y apoyarme para la realización de esta tesis.

A la Dra. Berrón por su ayuda.

A mis padres por haberme dado la herencia más valiosa : mi carrera.

A mis profesores por compartirme sus conocimientos.

A todas las personas que me auxiliaron y vieron por mí cuando los necesite para seguir adelante.

A mi comunidad porque me enseñaron el camino... Gracias.

CONTENIDO PÁGINA

Capítulo 1

Introducción

El objetivo del problema de la dieta es encontrar la combinación más barata de alimentos que satifagan todos los requerimientos nutricionales diarios de una persona. Este problema se plantea como un problema de *Programación Lineal*, para minimizar el costo del menú de tal forma que se satisfagan las restricciones que regulan el número de calorías y cantidades de vitaminas, minerales, grasas, sodio y colesterol en la dieta.

El problema de la dieta alimenticia puede resolverse desde dos enfoques principales; el primero corresponde a dar una solución mediante la modificación del **método simplex** (George Dantzig, 1947) [FO91], y el segundo mediante los denominados **métodos de punto interior** (Karmarkar, 1984) [AA93].

El presente trabajo consiste en hacer la **Optimización del Problema de la Dieta Alimenticia**, partiendo del modelo matemático y haciendo modificaciones al método simplex, de tal forma que nos permita resolver problemas de este tipo.

El **Problema de la Dieta Alimenticia** es modelado como un problema de programación lineal que tiene una función objetivo representando el costo de la dieta, restricciones acotadas representando las cantidades de cada producto que debe contener, y variables acotadas que representan la cantidad de nutrientes necesarios para una persona en su dieta. Algebraicamente el modelo matemático [1] es el siguiente:

$$
\begin{array}{ll}
min & cost = \sum_{i=1}^{n} (cost[i].x[i]) \\
s.a. & \\
 & Min_{nutr}[j] \leq \sum_{i=1}^{n} nutr[j][i].x[i] \leq Max_{nutr}[j], \quad 1 \leq j \leq m \\
 & Min_{alimento}[i] \leq x[i] \leq Max_{alimento}[i], \qquad\quad 1 \leq i \leq n
\end{array}
$$

[1] *min* significa minimizar y *s.a.* significa sujeto a.

Donde:

$cost[i]$	= costo de cada uno de los alimentos.
$x[i]$	= cantidad desconocida de cada alimento a comer.
$nutr[j][i]$	= cantidad de nutriente j en el alimento i.
$Min_{nutr}[j]$	= mínima cantidad de nutriente j requerido para una persona.
$Max_{nutr}[j]$	= máxima cantidad de nutriente j requerido para una persona.
$Min_{alimento}[i]$	= mínima cantidad de alimento i deseado por día.
$Max_{alimento}[i]$	= máxima cantidad de alimento i deseado por día.

El **Problema de la Dieta Alimenticia** es uno de los problemas de optimización que se ha estudiado desde la década de los 30's y 40's. Éste fue motivado por el deseo de la armada de encontrar la dieta que cumpliera con los requerimientos nutricionales necesarios en su alimentación y que además su costo fuera mínimo.

Uno de los primeros investigadores que estudió este problema fue George Stigler, pero utilizando un método heurístico. Posteriormente en 1947 Jack Laderman tomó el modelo de Stingler y usó el reciente método simplex de esa época, pero además Laderman también se apoyo en algunas operaciones manuales para obtener una solución [HIST/1].

El sistema **DIETEX** se desarrolló para contribuir en la resolución de uno de los problemas reales más importante en la vida diaria, tanto de empresas como de ciudadanos.

Entre algunas de las aplicaciones reales que se le pueden dar al sistema **DIETEX**, podemos mencionar su utilización en los restaurantes, las cárceles, los hospitales, los centros de beneficiencia pública, las familias a través de programas de orientación al consumidor y muchas otras, donde se observe que sólo unas pocas personas deciden la dieta de las personas que serán alimentadas.

Este trabajo consta de 9 capítulos. En este primer capítulo, se trata de dar un enfoque general acerca del problema, como resolverlo y algunas otras generalidades como resumen del trabajo realizado.

En el capítulo 2 se presentan las definiciones básicas de programación lineal en las que se apoya el algoritmo, así como también los modelos de los tipos de problemas lineales.

En el capítulo 3 se habla acerca de cómo es que se obtiene la forma explícita para resolver un problema lineal clásico, es decir, con una función objetivo, restricciones con desigualdades y variables positivas. También se analizan los diferentes casos que se pueden presentar para obtener un resultado final. Esta forma explícita es fundamental para el método simplex.

En el capítulo 4 se hace la obtención y el análisis de la forma explícita para problemas lineales con variables acotadas y variables positivas. Entonces se hacen modificaciones al método simplex clásico para obtener un método de solución a este tipo de problemas.

En el capítulo 5 se presenta el algoritmo que resuelve problemas que pueden ser modelados como el **Problema de la Dieta Alimenticia**; esto es que se definen con restricciones acotadas, variables acotadas y variables positivas.

En el capítulo 6 se muestran las rutinas que conforman al sistema **DIETEX**. Para cada rutina se describe brevemente su función, sus parámetros de entrada y salida (si es que los necesita), la forma en que se usa dentro del programa y el pseudocódigo. Así también se muestran las variables principales que usa el algoritmo.

En el capítulo 7 se presentan las pruebas aplicadas al sistema **DIETEX** y los resultados obtenidos. También se presentan las razones por las cuales se generaron cada uno de los problemas prueba aplicados.

En el capítulo 8 se concluye el trabajo de tesis. En esta parte se describen las ventajas y limitaciones del sistema **DIETEX**, las razones por las cuales se eligieron determinadas herramientas para su desarrollo, los problemas que se presentaron a lo largo del mismo, las conclusiones obtenidas después de aplicar las pruebas y las conclusiones del trabajo de tesis.

Finalmente el capítulo 9 es el manual de usuario del sistema **DIETEX**, dirigido a todas aquellas personas que deseen utilizarlo. En este manual se explica su funcionamiento y utilización, el formato de los archivos de entrada y salida, y como pueden interpretarse los resultados arrojados por el sistema **DIETEX**.

En la Bibliografía, se listan todas las fuentes de información que sirvieron de apoyo en la realización del presente trabajo de tesis y lo fundamentan.

Capítulo 2

Generalidades

Antes de comenzar a hablar del **Problema de la Dieta Alimenticia**, mencionaremos algunos conceptos básicos sobre los cuales se rige la programación lineal, y que nos servirán como base para entender los capítulos posteriores.

En el presente capítulo se presentan algunas definiciones y teoremas, así como los modelos del problema lineal con variables positivas y el problema lineal con variables acotadas y positivas. Estos modelos son la base para resolver el **Problema de la Dieta Alimenticia**, que se caracteriza por tener restricciones acotadas, variables acotadas y variables positivas.

2.1 Definiciones Básicas.

Definición 1.

Dados $x', x'' \in \Re^n$, llamamos **segmento** que une x' y x'', al conjunto

$$[x', x''] = \{\lambda x' + (1-\lambda)x'' \mid \lambda \in [0,1]\}. \tag{2.1}$$

Definición 2.

Un **conjunto** $C \subseteq \Re^n$ se dice **convexo** si para cada par de puntos $x', x'' \in C$, se tiene que $[x', x''] \subset C$. Es decir, si y sólo si contiene con cada par de puntos, todas las combinaciones $x(\lambda)$ que cumplen con la expresión (2.1) (combinaciones convexas), para $0 \leq \lambda \leq 1$.

Notación.

A lo largo de este trabajo $\Re^{m\times n}$ denotará al conjunto de las matrices $m \times n$ con entradas en los números reales.

Ejemplo 1:

Sean A y B matrices $m \times n$, b_1 y b_2 vectores de $\Re^m$ y el conjunto $C = \{x \in \Re^n | Ax = b_1, Bx \geq b_2\}$ [1]. Afirmamos que C es un conjunto convexo. En efecto, para cualquier $x_1, x_2 \in C$ se tiene $Ax_1 = b_1$, $Ax_2 = b_1$, $Bx_1 \geq b_2$ y $Bx_2 \geq b_2$, demostraremos que $(\lambda x_1 + (1-\lambda)x_2) \in C$. Pues

$$A(\lambda x_1 + (1-\lambda)x_2) = (\lambda Ax_1 + (1-\lambda)Ax_2) = \underbrace{\lambda b_1 + (1-\lambda)b_1}_{\text{por hipótesis}} = b_1$$

y

$$B(\lambda x_1 + (1-\lambda)x_2) = (\lambda Bx_1 + (1-\lambda)Bx_2) = \lambda b_2 + (1-\lambda)b_2 = b_2$$

Definición 3.

Sea $a = (a_1, a_2, \ldots, a_n) \in \Re^n$ y $t \in \Re$. Un semiespacio en $\Re^n$ es un conjunto de la forma:

$$E = \{x = (x_1, x_2, \ldots, x_n) \in \Re^n \mid a_1x_1 + a_2x_2 + \ldots + a_nx_n \geq t\}$$

Al hiperplano (n-1)-dimensional $a_1x_1 + a_2x_2 + \ldots + a_nx_n = t$ se le llama **frontera del espacio**. Un hiperplano es el conjunto de soluciones de una ecuación lineal.

Definición 4.

Llamaremos **conjunto factible** a una intersección finita de semiespacios:

$$F = \{x = (x_1, x_2, \ldots, x_n) \in \Re^n \mid a_{i1}x_1 + a_{i2}x_2 + \ldots + a_{in}x_n \geq b_i;$$
$$i = 1, 2, \ldots, k\}$$

Las propiedades de los conjuntos factibles son:

- Todo conjunto factible es convexo.
- Un conjunto factible puede ser acotado, no acotado e incluso vacío.
- En programación lineal todo conjunto factible debe satisfacer las condiciones $x_1 \geq 0, x_2 \geq 0, \ldots, x_n \geq 0$.

[1] De ahora en adelante $x = (x_1, x_2, \ldots, x_n)$ y $x \geq 0$ es equivalente a decir que $x_j \geq 0$ $\forall j = 1 \ldots n$. En general si tenemos las desigualdades $l \leq x \leq u$ diremos que los vectores $u = (u_1, u_2, \ldots, u_n)$ y $l = (l_1, l_2, \ldots, l_n)$.
Análogamente permitimos incluso $u = \infty$ si y solo si $u_j = \infty$ $\forall j = 1 \ldots n$.

Definición 5.

Un punto x' de un conjunto convexo C es un **punto extremo** o **vértice** de C si y sólo si, no es interior a un segmento de recta contenido en C; dicho de otra forma, si y sólo si

$$x' = (1 - \beta)y' + \beta z'$$

con $0 \leq \beta \leq 1$ y $y', z' \in C \Rightarrow x' = y' = z'$.

Definición 6.

La solución de un programa lineal, puede presentarse en alguna de las siguientes formas:

1. **Solución Óptima Única.**
   Ocurre siempre en un punto extremo de la región factible. Para este tipo de solución existen 2 posibilidades:

   a)Región factible acotada.

   b)Región factible no acotada.

2. **Soluciones Óptimas Alternativas.**
   No existe un punto único para la solución, sino que un conjunto de puntos forman la solución del problema. En este caso también se presentan las dos posibilidades de la solución anterior.

3. **Solución Óptima No Acotada.**
   Esta se presenta cuando la configuración de la región factible que tiene la función objetivo, es posible que se desplace tanto como se desee en la dirección del óptimo, sin encontrar un punto extremo que impida dicho desplazamiento. En este caso se dice que la solución se dezplaza a través de un **rayo**.

4. **Región Factible Vacía.**
   Esta solución existe siempre que no exista algún valor para el vector de incognitas del problema que satisfaga todas las restricciones. En este caso el problema se dice que es **no factible** o **inconsistente**.

Definición 7.

Un **politopo** es un conjunto formado por la intersección de un número finito de semiespacios cerrados.

Definición 8.

Un **politopo cónico** es un conjunto formado por la intersección de un número finito de semiespacios cerrados que pasan por un punto determinado.

Definición 9.

Un **poliedro** es un politopo acotado y no vacío.

Definición 10.

Sea B una submatriz no singular $m \times m$ resultado de agrupar m columnas linealmente independientes de A de dimensión $m \times n$.

Si todos los $n - m$ elementos del vector x denominados variables no básicas, se hacen cero y se resuelve la ecuación $Ax = b$ en las variables básicas, entonces la solución resultante recibe el nombre de **solución básica asociada a la base** B.

Las $n-m$ columnas de A que no forman parte de B se agrupan en una matriz $n \times (n - m)$ denominada R (asociada a las variables no básicas). Como consecuencia las variables no básicas forman x^R.

Definición 11.

Si una o más de las variables básicas de una solución es cero, la solución se denomina **solución básica degenerada.**

Definición 12.

Una solución básica en la que todos sus componentes son positivos recibe el nombre de **solución básica factible.**

Definición 13.

Si algún componente de la solución básica es cero, entonces se dice que se trata de una **solución básica factible degenerada.**

Definición 14.

Una **dirección** del politopo $P = \{x \in \Re^n : Ax = b, x \geq 0\}$ es un vector no nulo $d \in \Re^n$ tal que para todo $x_0 \in P$, el **rayo** $\{x \in \Re^n : x = x_0 + \lambda b, \lambda \geq 0\}$ pertenece a P.

Definición 15.

Dado un problema en la forma:

$$\begin{aligned} min \quad & z = c^T x \\ s.a. \quad & \\ & Ax = b \\ & l \leq x \leq u \end{aligned}$$

se dice que un vector x es una **solución básica factible** de este sistema, si x^B es resultado de

$$Bx^B = b, x^{R_l} = l^{R_l}, x^{R_u} = u^{R_u}.$$

Además si $l^B < x^B < u^B$, entonces x es una solución básica factible no degenerada. En otro caso, si algún valor de x^B es igual a l_j o u_j, la solución se denomina básica factible degenerada.

2.2 Modelo de los Problemas Lineales

Un problema de programación lineal es un problema de minimizar o maximizar una función lineal dentro de un región formada por restricciones lineales del tipo de desigualdad, igualdad o ambas. El siguiente modelo nos muestra un ejemplo:

$$min\ z = c^T x \tag{2.2}$$

$$s.a.$$
$$Ax \geq b$$
$$x \geq 0$$

Mediante manipulaciones adecuadas, el problema se puede transformar a un problema equivalente para facilitarnos la resolución de este tipo de problemas.

También puede ayudarnos el convertir un problema de maximización en un problema de minimización y viceversa, multiplicando los coeficientes de la función objetivo por -1, siendo la solución del problema original igual a -1 por el óptimo del nuevo problema. La ecuación (2.3) representa esto de forma algebraica.

$$max \sum_{j=1}^{n} c_j x_j = -min \sum_{j=1}^{n} -c_j x_j \tag{2.3}$$

Es posible transformar el problema (2.2) en otro problema lineal en la forma estándar (todas las restricciones son de igualdad y todas las variables son no negativas), sólo sustrayendo el vector y de **variables de holgura**, como sigue:

$$min\ z = c^T x \tag{2.4}$$

$$s.a.$$
$$Ax - y = b$$
$$x, y \geq 0$$

El método simplex está diseñado para aplicarse a problemas en la forma estándar, por eso es importante que los problemas que deseamos resolver, primeramente los cambiemos a su forma estándar y después apliquemos el método.

Los politopos y los poliedros son conjuntos convexos por definición, entonces el conjunto de soluciones de nuestro problema lineal $K = \{x \in \Re^n : Ax = b, x \geq 0\}$, es un politopo. Por ésta razón decimos que la solución de un problema lineal es un punto extremo, en donde se maximiza o minimiza una función denominada **función objetivo**, y que está sujeta a restricciones.

Capítulo 3

Forma Explícita del Problema Estándar de Programación Lineal

Si el número de restricciones m y el número de variables n es relativamente grande, es necesario representar este tipo de problema en una computadora.

Entonces se obtiene una forma algebraica general denominada **forma explícita**, a partir de la cual se origina el denominado **algoritmo simplex.**

En este capítulo se muestra la obtención y el análisis de la forma explícita con el objeto de comprender el algoritmo simplex clásico, y en los capítulos posteriores se tratarán las variaciones del problema y del algoritmo necesarias para resolver **El Problema de la Dieta Alimenticia.**

3.1 Obtención de la Forma Explícita

Para obtener la forma explícita, entonces partimos de la suposición de que nuestro problema se encuentra en la forma estándar[1] como se muestra a continuación:

$$min \ z = c^T x \tag{3.1}$$

$$s.a. \ Ax = b \tag{3.2}$$

$$x \geq 0$$

[1] Denótese por c el vector fila $(c_1, c_2, \ldots, c_n)$ y considérense los vectores columna x y b, y la matriz A de $m \times n$. Escribiendo A como $[a_1, a_2, \ldots, a_n]$ en donde a_j es la $j-ésima$ columna de A.

donde $x \in \Re^n$, $A \in \Re^{m \times n}$, $m \leq n$, $b \in \Re^m$.

Si B es una base factible, entonces es posible expresar que:

$$A = [B, R], B \in \Re^{m \times m}, R \in \Re^{m \times (n-m)}$$

$$x = [x^B, x^R]^T, x^B \in \Re^m, x^R \in \Re^{n-m}$$

Para comprobar que el valor de la función objetivo corresponde a una solución básica factible no degenerada, dicha solución deberá encontrarse en la intersección en $\Re^n$ de los m hiperplanos correspondientes a (3.2) y los $n - m$ hiperplanos correspondientes a x^R =0.

Por esta razón, se hace la siguiente sustitución en (3.2):

$$Ax = [B, R] \begin{pmatrix} x^B \\ x^R \end{pmatrix} = b$$

$$Bx^B + Rx^R = b$$

y si despejamos x^B

$$B^{-1}Bx^B + B^{-1}Rx^R = B^{-1}b$$

$$x^B + B^{-1}Rx^R = B^{-1}b$$

$$x^B = -B^{-1}Rx^R + B^{-1}b \tag{3.3}$$

obtenemos la solución en forma explícita: $\tilde{x}^B = B^{-1}b$, la cual corresponde al vector de solución con respecto a la base factible B.

La función objetivo que se obtiene sustituyendo (3.3) en (3.1) es:

$$z = (c^B, c^R) \begin{pmatrix} x^B \\ x^R \end{pmatrix} = c^B x^B + c^R x^R$$

$$= c^B(B^{-1}b - B^{-1}Rx^R) + c^R x^R$$

$$= \underbrace{c^B B^{-1} b}_{\tilde{z}} - (c^B B^{-1} R - c^R)x^R$$

$$= \tilde{z} - \sum_{j \in J} (c^B B^{-1} a^j - c_j)x_j$$

$$= \tilde{z} - \sum_{j \in J} (\xi_j - c_j)x_j$$

donde $\xi_j = c^B B^{-1} a^j$ y a^j es una columna de R.

Retomando (3.3) tenemos que:

$$x^B + B^{-1}Rx^R = B^{-1}b$$
$$x^B + \sum B^{-1}a^j x_j = B^{-1}b$$
$$x^B + \sum_{j\in J} y^j x_j = y^0 = B^{-1}b$$

Finalmente las ecuaciones (3.1) y (3.2) se tranforman en:

$$\begin{array}{ll} min & z = \tilde{z} - \sum_{j\in J}(\xi_j - c_j)x_j \\ s.a. & \\ & x^B + \sum_{j\in J} y^j x_j = y^0 \\ & x_j \geq 0 \end{array}$$

Este problema recibe el nombre de **problema en forma explícita respecto a la base** B, y de su análisis que se explica en la siguiente sección se obtiene el **método simplex.**

3.2 Análisis de la Forma Explícita

Caso 1.

Si $\xi_j - c_j \leq 0$, $\forall j \in J$ entonces $\begin{pmatrix} x^B \\ x^R \end{pmatrix}$ es el vector de valores de las incógnitas que corresponden a la solución óptima, pues $z = \tilde{z} - \sum_{j\in J}(\xi_j - c_j)x_j \geq \tilde{z}$

Caso 2.

$\exists k \in J$ tal que $\xi_k - c_k > 0$, entonces conviene buscar una nueva solución factible básica con $x_k > 0$. Porque el valor de la función aún puede disminuir más sin que la solución se salga de la región factible.

Para analizar si esto es posible, debe construirse una familia de soluciones $\{x(\theta)\}$ como se muestra a continuación:

$$[x(\theta)]_j = \begin{cases} \theta & \text{si } j = k, \quad \theta \geq 0 \\ 0 & \text{si } j \in J \setminus \{k\} \\ x_j^B - y_j^k\theta \geq 0 & \text{si } j \in I \end{cases} \quad \text{;donde I=índices de variables básicas} \tag{3.4}$$

con el objeto de determinar cual de todas las variables no básicas puede entrar a la base para formar parte de la solución.

Todas las variables no básicas contribuyen al valor de la función objetivo con un valor de cero, mientras que las variables básicas son las que influyen totalmente sobre el valor actual de la función objetivo. Así es que si se cambia una variable básica por una variable no básica, entonces el valor de la función objetivo podrá aumentar o disminuir.

Como lo que estamos buscando es minimizar el costo, debemos encontrar una variable no básica que haga que esto suceda. Entonces después de haber generado todos los valores $x(\theta) \geq 0$, con la función (3.4), se analizan con base en los siguientes puntos:

• Si $y_j^k \leq 0$, entonces θ puede ser arbitrario siempre que sea positivo ($\theta \geq 0$).

• Si $y_j^k > 0$, entonces debe cumplirse que $x_j^B - y_j^k\theta \geq 0$. Despejando θ deducimos que:

$$\begin{aligned} x_j^B &\geq y_j^k\theta \\ x_j^B/y_j^k &\geq \theta \end{aligned} \tag{3.5}$$

Ahora sean $y_{\hat{1}}^k, y_{\hat{2}}^k, \ldots, y_{\hat{i}}^k \geq 0$, entonces $\theta_{\hat{1}} \leq x_{\hat{1}}^B/y_{\hat{1}}^k, \theta_{\hat{2}} \leq x_{\hat{2}}^B/y_{\hat{2}}^k, \ldots, \theta_{\hat{i}} \leq x_{\hat{i}}^B/y_{\hat{i}}^k$.

La conclusión que se deriva de este análisis es:

$$\theta = min\{x_j^B/y_j^k : y_j^k > 0, 1 \leq j \leq m\},$$

porque se busca el θ mínimo de todos, que satisfaga la ecuación (3.5) $\forall y_j^k > 0$.

El valor de la variable no básica x_k se incrementa de 0 a θ_{min} convirtiendose en una variable básica, y la variable x_j sale de la base con valor 0.

Después de este proceso, sólo se necesita reconfigurar la base por medio del pivoteo. Las variables básicas se modifican de la siguiente forma: $\check{x}_j^B = x_j^B - \theta y_j^k$, $j = 1, \ldots, m$.

3.3 Algoritmo Simplex

A continuación basándonos en el análisis de la forma explícita anterior, tenemos el algoritmo simplex siguiente:

Dada una base inicial B^0 factible de P:

$$\begin{array}{ll} min & z = c^T x \\ s.a. & \\ & Ax \geq b \\ & x \geq 0 \end{array}$$

Paso 0. Hacer $v = 0$. I^v.

Paso 1. Llevar el problema a la forma explícita respecto a la baseB^v.

Paso 2. "Criterio de Entrada"
Calcular $\xi_k - c_k = max_{j \in J}\{\xi_j - c_j\}$
Si $\xi_k - c_k \leq 0$ Entonces
FIN: $(x^{B^v}, 0)$ es la solución óptima.

Paso 3. "Criterio de Salida"
Si $y_i^k \leq 0 \ \forall \ i \in I$ Entonces
FIN: P no tiene óptimo finito.
En otro caso
Calcular $\theta_v := \frac{x_l}{y_l^k} := min_{i|y_i^k>0}\{\frac{x_i}{y_i^k}\} \quad i \in I$

Paso 4. "*Actualizar*"
Hacer $I^{v+1} = I^v \setminus \{l\} \bigcup \{k\}; v \leftarrow v + 1$
ir al Paso 2.

Capítulo 4

Forma Explícita del Problema de Programación Lineal con Variables Acotadas y Variables Positivas

Hasta el capítulo anterior hemos tratado con problemas de programación lineal con variables positivas. El método de solución de este tipo de problema es llamado **método simplex clásico**.

A partir de este capítulo, el tipo de problemas lineales que vamos a tratar tienen variables positivas y variables acotadas. Su forma estándar es como se muestra:

(4.1)

$$\begin{aligned} min \quad & z = c^T x \\ s.a. \quad & \\ & Ax = b \\ & l \leq x \leq u \end{aligned}$$

Para las variables positivas $l = 0$ y $u = \infty$, mientras que para las variables acotadas, l y u tienen cualquier valor positivo o negativo.

En este capítulo mostramos como se obtiene la forma explícita para poder resolver este tipo de problemas. Y a partir de esta, hacer modificaciones al método simplex clásico para obtener él método de solución que buscamos. La obtención de esta forma explícita es la que nos servirá de base posteriormente para resolver el **Problema de la Dieta Alimenticia**.

4.1 Cambio de Variable

Lo que debemos hacer primero para resolver este tipo de problema es transformarlo de un problema con variables acotadas a un problema con variables positivas o **variables libres**. Esto con la finalidad de utilizar posteriormente lo que ya conocemos del método simplex clásico y obtener la solución.

La forma más rápida de trasformar este problema es añadir variables de holgura como se muestra a continuación:

$$min\ z = c^T x$$

$$s.a.\ Ax = b$$

$$x + x_1^h = u$$

$$x - x_2^h = l$$

$$x, x_1^h, x_2^h \geq 0$$

y resolver el problema como se explicó en el capítulo anterior.

Posiblemente obtengamos buenos resultados cuando debamos resolver problemas que tienen pocas incógnitas, pero no es así para problemas donde n y m son muy grandes, porque el número de variables se triplica y el número de restricciones sería $m + 2n$.

Ahora nuestro problema de optimización de costos se convierte en un problema de optimización de espacio de almacenamiento y eficiencia en tiempo de solución, por el manejo de tantas variables.

La forma más adecuada para resolver el problema (4.1) consiste en hacer un cambio de variable de la siguiente manera:

$$l_j \leq x_j \leq u_j \Rightarrow l_j - l_j \leq x_j - l_j \leq u_j - l_j \Rightarrow 0 \leq x_j - l_j \leq u_j - l_j$$

$$0 \leq x'_j \leq u_j - l_j$$

donde:

$$x'_j = x_j - l_j \Rightarrow x_j = x'_j - l_j$$

De esta manera, el número de variables no sufre ningún cambio, unicamente se trabaja con variables positivas que son equivalentes a las variables acotadas del problema original.

4.2 Obtención de la Forma Explícita

Sea el problema (4.1), que tiene una matriz $A \in \Re^{m\times n}$ de rango m, y $B \in \Re^{m\times m}$ es una matriz base, de tal forma que:

$$A = [B, R_l, R_u] \tag{4.2}$$

y

$$x = [x^B, x^{R_l}, x^{R_u}]^T \tag{4.3}$$

Para este problema vamos a hacer una pequeña variación, con respecto a la notación utilizada. Denotemos I al conjunto de variables básicas, J al conjunto de variables no básicas, J_l al conjunto de variables no básicas que se encuentren en su cota inferior y J_u al conjunto de variables no básicas que se encuentren en su cota superior.

La idea principal en que se basará el método simplex para variables acotadas es que a partir de una solución básica factible, se realiza el análisis de los costos de las variables no básicas para determinar si es posible mejorar la solución.

Recordando la definición (15) del capítulo 2, para encontrar la solución básica factible no degenerada sustituimos (4.2) en la ecuación (4.1):

$$Bx^B + R_l x^{R_l} + R_u x^{R_u} = b$$

obteniendo:

$$x^B = B^{-1}b - B^{-1}R_l x^{R_l} - B^{-1}R_u x^{R_u} \tag{4.4}$$

A partir de lo cual la función objetivo es:

$$z = (c^B, c^{R_l}, c^{R_u}) \begin{pmatrix} x^B \\ x^{R_l} \\ x^{R_u} \end{pmatrix} = c^B x^B + c^{R_l} x^{R_l} + c^{R_u} x^{R_u}$$

$$= c^B(B^{-1}b - B^{-1}R_l x^{R_l} - B^{-1}R_u x^{R_u}) + c^{R_l} x^{R_l} + c^{R_u} x^{R_u}$$

$$= \underbrace{c^B B^{-1} b}_{\tilde{z}} + (c^{R_l} - c^B B^{-1} R_l) x^{R_l} + (c^{R_u} - c^B B^{-1} R_u) x^{R_u}$$

$$= \tilde{z} + \sum_{j\in J_l} (c_j - c^B B^{-1} a^j) x_j + \sum_{j\in J_u} (c_j - c^B B^{-1} a^j) x_j$$

$$= \tilde{z} + \sum_{j\in J_l} (c_j - \xi_j) x_j + \sum_{j\in J_u} (c_j - \xi_j) x_j$$

- 3^{er} *Problema Prueba*

  Para un problema con 32 variables con cotas inferiores iguales a cero y cotas superiores positivas diferentes de cero y 11 restricciones con cotas superiores e inferiores positivas y diferentes de cero, se evalua:

  1. El tipo de solución encontrada.
  2. El número de iteraciones.
  3. El número de evaluaciones de función.
  4. El tiempo de respuesta.
  5. El valor del óptimo (cantidades y costo total).

7.1 1^{er} Problema Prueba

Para realizar la primera prueba del algoritmo, se generó un problema con las siguientes características:

- Variables con cotas inferiores positivas o iguales a cero y cotas superiores positivas.
- Restricciones con cotas superiores e inferiores positivas y diferentes de cero.
- Dimensión n=64.
- Restricciones m=11.

Esta prueba consistió en ejecutar el algoritmo SIMPLEX, cambiando 2 veces la función objetivo como sigue:

1. Primero se ejecutó el algoritmo tomando una función objetivo con costos positivos.
2. Después se cambiaron de signo los costos de la función objetivo de 1, y se inició con esto la búsqueda del óptimo.

En los dos casos, se evaluaron los siguientes aspectos:

1. El número de iteraciones.
2. El número de evaluaciones de función.
3. El tiempo de respuesta.
4. El tipo de solución encontrada al problema.
5. El valor del óptimo (cantidades y costo total).

Aclaración:
Cuando DIETEX tiene un tiempo de respuesta menor que 1 segundo, el tiempo es redondeado a 0 segundos.

Los resultados obtenidos de las evaluaciones son los siguientes:

Evaluación de desempeño de la ejecución 1

El problema original tiene óptimo:20022.02

Tipo Sol.: SOLUCION BASICA FACTIBLE OPTIMA FINITA

Iteraciones de fase 1: 1
Iteraciones de fase 2: 0
Iteraciones de fase 3: 1
Evaluaciones de función en fase 1: 0
Evaluaciones de función en fase 2: 0
Evaluaciones de función en fase 3: 1
Tiempo de Respuesta: 0.0000000 segundos

Solución óptima de la evaluación 1

X[1]=1.0000
X[2]=1.0000
X[3]=1.0000
X[4]=1.0000
X[5]=2.0000
X[6]=2.0000
X[7]=2.0000
X[8]=2.0000
X[9]=0.0000
X[10]=0.0000
X[11]=0.0000
X[12]=0.0000
X[13]=1.0000
X[14]=1.0000
X[15]=1.0000
X[16]=1.0000
X[17]=2.0000
X[18]=2.0000
X[19]=2.0000
X[20]=2.0000
X[21]=0.0000
X[22]=0.0000
X[23]=0.0000
X[24]=0.0000
X[25]=1.0000
X[26]=1.0000
X[27]=1.0000
X[28]=1.0000

X[29]=2.0000
X[30]=2.0000
X[31]=2.0000
X[32]=2.0000
X[33]=0.0000
X[34]=0.0000
X[35]=0.0000
X[36]=0.0000
X[37]=1.0000
X[38]=1.0000
X[39]=1.0000
X[40]=1.0000
X[41]=2.0000
X[42]=2.0000
X[43]=2.0000
X[44]=2.0000
X[45]=0.0000
X[46]=0.0000
X[47]=0.0000
X[48]=0.0000
X[49]=1.0000
X[50]=1.0000
X[51]=1.0000
X[52]=1.0000
X[53]=2.0000
X[54]=2.0000
X[55]=2.0000
X[56]=2.0000
X[57]=0.0000
X[58]=0.0000
X[59]=0.0000
X[60]=0.0000
X[61]=1.0000
X[62]=1.0000
X[63]=1.0000
X[64]=1.0000

Evaluación de función objetivo=20022.02

Evaluación de desempeño de la ejecución 2

Evaluación de función objetivo = 19977.98047
Evaluación de función objetivo = 19977.98047
Evaluación de función objetivo = 19977.98047
Evaluación de función objetivo = 19977.98047
Evaluación de función objetivo = 19977.98047
Evaluación de función objetivo = 19977.98047

El problema original tiene óptimo:19936.63

Tipo Sol.: SOLUCION BASICA FACTIBLE OPTIMA FINITA

Iteraciones de fase 1: 1
Iteraciones de fase 2: 6
Iteraciones de fase 3: 1
Evaluaciones de función en fase 1: 0
Evaluaciones de función en fase 2: 6
Evaluaciones de función en fase 3: 1
Tiempo de Respuesta: 0.0000000 segundos

Solución óptima de la evaluación 2

X[1]=1.0000
X[2]=1.0000
X[3]=1.0000
X[4]=1.0000
X[5]=10.00000
X[6]=2.0000
X[7]=2.0000
X[8]=2.0000
X[9]=0.0000
X[10]=0.0000
X[11]=0.0000
X[12]=0.0000
X[13]=1.0000
X[14]=1.0000
X[15]=1.0000
X[16]=1.0000
X[17]=2.0000
X[18]=2.0000
X[19]=2.0000
X[20]=2.0000
X[21]=0.0000
X[22]=0.0000
X[23]=0.0000

X[24]=0.0000
X[25]=1.0000
X[26]=1.0000
X[27]=1.0000
X[28]=1.0000
X[29]=2.0000
X[30]=2.0000
X[31]=2.0000
X[32]=2.0000
X[33]=0.0000
X[34]=0.0000
X[35]=0.0000
X[36]=6.6895
X[37]=1.0000
X[38]=1.0000
X[39]=1.0000
X[40]=1.0000
X[41]=2.0000
X[42]=2.0000
X[43]=2.0000
X[44]=2.0000
X[45]=0.0000
X[46]=0.0000
X[47]=0.0000
X[48]=0.0000
X[49]=1.0000
X[50]=1.0000
X[51]=10.00000
X[52]=1.0000
X[53]=10.00000
X[54]=2.0000
X[55]=2.0000
X[56]=10.00000
X[57]=0.0000
X[58]=0.0000
X[59]=0.0000
X[60]=0.0000
X[61]=1.0000
X[62]=10.00000
X[63]=1.0000
X[64]=1.0000

Evaluación de función objetivo=19936.63

7.2 2^o Problema Prueba

El segundo problema prueba que se generó cumple con las siguientes características:

- Variables con cotas inferiores y superiores positivas.
- Restricciones con cotas superiores e inferiores positivas y diferentes de cero.
- Dimensión n=24.
- Restricciones m=11.

Para esta segunda prueba se trata de comprobar que el problema no tiene solución y evaluar el desempeño del algoritmo, ya que cuando no se encuentra una solución inicial, el algoritmo busca una solución extendida por medio de un método de combinaciones de todos los valores permitidos para cada una de las variables.

Los aspectos evaluados en este caso son:

1. El tipo de solución encontrada.
2. En caso de no tener solución inicial, el tiempo empleado para localizar una solución inicial básica extendida.
3. El número de iteraciones.
4. El número de evaluaciones de función de cada una de las fases.
5. El tiempo de respuesta total.

Observaciones:

1. Para los problemas se observó que se ejecutan más iteraciones:
   (a) En la fase 1 cuando el problema no tiene una tiene una solución inicial básica factible. Esto es porque busca una solución extendida haciendo todas las combinaciones de las variables hasta que la encuentra.
   (b) En la fase 2 cuando la solución inicial básica factible no es el óptimo.
2. Cuando un problema tiene variables acotadas con cotas inferiores iguales a cero, el óptimo permanece en cero.
3. Como el problema de la Dieta Alimenticia es un problema de minimización con una función objetivo de costos positivos, en los resultados obtenidos se observó que el 80% de las veces la solución inicial básica factible es el óptimo.
4. Se generaron problemas prueba con una función objetivo de costos negativos sólo para evaluar el desempeño del algoritmo, y comprobar que tanto puede mejorar la solución inicial.
5. Una característica importante es que la dimensión de los problemas que resuelve DIETEX es mucho mayor que la de los problemas que resuelve TORA y NEW (el programa que se encuentra en internet).
6. A nivel de programación, las funciones para medir el tiempo de ejecución que proporciona el lenguaje utilizado, sólo pueden medir segundos como mínimo. Por tal razón el algoritmo registra un tiempo de cero cuando ocupa menos de 1 segundo en el proceso.
7. El tiempo que mide DIETEX, es sólo el tiempo que ocupa el algoritmo para pasar por sus fases; sin incluir el tiempo que ocupe para accesar los archivos o en presentar los datos en la pantalla.
8. Los problemas 2 y 3 se resolvieron en Tora después de haber realizado manualmente la fase 1 del algoritmo.
9. El método utilizado para encontrar una solución extendida hace que se incremente el tiempo de búsqueda exponencialmente con respecto a la cantidad de variables que tiene un problema. Siendo esta una desventaja que presenta este algoritmo.

$$= \tilde{z} - \sum_{j \in J_l} (\xi_j - c_j) x_j - \sum_{j \in J_u} (\xi_j - c_j) x_j$$

donde $\xi_j = c^B B^{-1} a^j$ y a^j es una columna de R.

Así de (4.4) llegamos a:

$$x^B + B^{-1} R_l x^{R_l} + B^{-1} R_u x^{R_u} = B^{-1} b$$
$$x^B + \sum_{j \in J_l} B^{-1} a^j x_j + \sum_{j \in J_u} B^{-1} a^j x_j = B^{-1} b$$
$$x^B + \sum_{j \in J_l} y^j x_j + \sum_{j \in J_u} y^j x_j = B^{-1} b$$

Finalmente el problema (4.1) se tranforma en

$$\begin{array}{ll} min & z = \tilde{z} - \sum_{j \in J_l} (\xi_j - c_j) x_j - \sum_{j \in J_u} (\xi_j - c_j) x_j \\ s.a. & \\ & x^B + \sum_{j \in J_l} y^j x_j + \sum_{j \in J_u} y^j x_j = B^{-1} b \\ & 0 \leq x_j \leq u_j - l_j \end{array}$$

4.3 Análisis de la Forma Explícita

Siempre que la función objetivo pueda mejorar debe cuidarse de mantener la factibilidad de la solución. Por lo cual de cumplirse que:

- Para las variables no básicas que se encuentren en su cota inferior, los costos deberán ser negativos. Es decir, $(c^B B^{-1} a^j - c_j) < 0$ para $x_j = l_j$

- Para las variables no básicas que se encuentren en su cota superior, los costos deberán ser positivos. Es decir, $(c^B B^{-1} a^j - c_j) > 0$ para $x_j = u_j$

De la misma forma que en el capítulo anterior, analizamos la forma explícita de la siguiente forma:

Después de haber calculado

$$\alpha = max\{max_{j\in J_l}\{\xi_j - c_j\}, max_{j\in J_u}\{-(\xi_j - c_j)\}\},$$

podemos tener los siguientes casos:

Caso 1.

Si $\alpha < 0$, entonces $\begin{pmatrix} x^B \\ x^{R_l} \\ x^{R_u} \end{pmatrix}$ es el vector de valores de las incógnitas que corresponden a la solución óptima, pues

$$z = \tilde{z} - \sum_{j\in J_l}(\xi_j - c_j)x_j - \sum_{j\in J_u}(\xi_j - c_j)x_j \geq \tilde{z}$$

Caso 2.

Si $\alpha \geq 0$, entonces existe $k \in J$ tal que la variable x_k al entrar a la base puede mejorar la solución. Se dice que la variable x_k es candidata a entrar a la base.

Para determinar si esto es posible, construimos una familia de soluciones $\{x(\theta)\}$, con el objeto de determinar cual de todas las variables básicas será reemplazada por x_k para formar parte de la solución. Entonces

$$[x(\theta)]_j = \begin{cases} x_i/\delta y_i^k & \text{si } i \in I \text{ y } \delta y_i^k > 0 \\ (u_i - x_i)/-\delta y_i^k & \text{si } i \in I \text{ y } \delta y_i^k < 0 \\ u_k & \text{si } k \in u \end{cases} \qquad (4.5)$$

donde I=índices de variables básicas.

El valor de δ se determina de la siguiente manera:

- Si $\xi_k - c_k > 0$ entonces $\delta = 1$
- Si $\xi_k - c_k < 0$ entonces $\delta = -1$

Ahora tomando nuestra familia de soluciones generada con base en la función (4.5), tenemos que:

• Si $x_k \in R_u$, entonces el valor θ elegido deberá ser no negativo para mantener la factibilidad del problema ($\theta \geq 0$).

• Si $x_k \in R_l$, entonces el valor θ elegido deberá ser no positivo para mantener la factibilidad del problema ($\theta \leq 0$).

Entonces para determinar cual es la variable básica que debe salir se busca el máximo valor positivo para las variables que se encuentran en su cota superior y el mínimo valor negativo para las variables que se encuentran en su cota inferior. Esto lo expresamos como

$$\theta = min\{min_{\delta y_i^k > 0}\{x_i/\delta y_i^k\}, min_{\delta y_i^k < 0}\{(u_i - \tilde{x}_i)/ - \delta y_i^k\}, u_k\}.$$

Finalmente

Si $\theta = u_k$ entonces no hay cambio en la base y x_k cambia a su cota contraria.

Si $\theta = \tilde{x}_q/\delta y_q^k$ entonces x_q sale de la base a nivel de su cota inferior y x_k entra con valor

$$\begin{cases} \theta & \text{si } \delta > 0 \\ u_k - \theta & \text{si } \delta < 0 \end{cases}$$

Si $\theta = (u_q - \tilde{x}_q)/ - \delta y_q^k$ entonces x_q sale de la base a nivel de su cota superior y x_k entra con valor

$$\begin{cases} \theta & \text{si } \delta > 0 \\ u_k - \theta & \text{si } \delta < 0 \end{cases}$$

Después de este proceso, sólo se necesita reconfigurar la base por medio del pivoteo, y partiendo de la nueva solución obtenida, se inicia todo el proceso otra vez.

Capítulo 5

Algoritmo para la Solución del Problema de la Dieta Alimenticia

En este capítulo se hacen modificaciones al método simplex para resolver el **Problema de la Dieta Alimenticia**, que es un problema lineal con restricciones acotadas, variables acotadas y variables positivas; las modificaciones están basadas en la explicación del capítulo anterior.

En el capítulo anterior se mostró el algoritmo simplex por el método de las dos fases, y ahora se presenta el algoritmo simplex para la solución del **Problema de la Dieta Alimenticia**, que consta de tres fases:

- La fase 1 es para preparar el problema antes de resolverlo, es decir, primero se obtiene un problema equivalente al problema original en forma estándar, luego se hacen los cambios de variable necesarios y por último se busca una solución factible inicial al problema. Si el problema no tiene una base factible inicial, entonces no tiene una solución factible.

- En la fase 2 se resuelve el problema, es decir, partiendo de la solución factible inicial encontrada en la fase 1, se busca la solución óptima. En esta fase es donde se usa el método simplex. Cuando se encuentra una solución óptima entonces se pasa a la siguiente fase.

- Finalmente en la fase 3 se hacen los cambios de variable necesarios nuevamente, para recuperar la solución óptima del problema original.

El **Problema de la Dieta Alimenticia** tiene la forma siguiente:

$$
\begin{aligned}
&min. \quad c^T x \\
&s.a. \\
&\qquad L_1 \leq f_1(x) \leq U_1 \\
&\qquad L_2 \leq f_2(x) \leq U_2 \\
&\qquad\qquad \vdots \\
&\qquad L_m \leq f_m(x) \leq U_m \\
&\qquad l_1 \leq x_1 \leq u_1 \\
&\qquad l_2 \leq x_2 \leq u_2 \\
&\qquad\qquad \vdots \\
&\qquad l_n \leq x_n \leq u_n
\end{aligned}
$$

Sea A la matriz de restricciones que se forma de este problema de dimensión $m \times n$, b_L y b_U los vectores de cotas inferiores y superiores de las restricciones de dimensión m y x el vector de incógnitas de dimensión n, tal que $m \leq n$.

A continuación se muestran los pasos del algoritmo para resolver el **Problema de la Dieta Alimenticia** basado en tres fases. El problema original sólo tiene variables acotadas, entonces lo primero que hace el algoritmo es pasar el problema original a su forma estándar, añadiendo variables positivas.

5.1 Primera Fase

Paso 1.- Obtención del problema equivalente en forma estándar.

Si denotamos a $A = (a_1, a_2, \ldots, a_m)^T$, donde A es la matriz de restricciones del problema original, a_r es un renglón de la matriz A y $r = 1, \ldots, m$.

Entonces sea $A' = (-a_1, a_1, -a_2, a_2, \ldots, -a_m, a_m)^T \cup I$, la matriz de restricciones del problema estándar de dimensión $2m \times (n + 2m)$, donde I es una matriz identidad de dimensión $2m \times 2m$, generada al introducir las variables de holgura al problema original, y el vector $b' = b_L \cup b_U = (b_{L_1}, b_{U_1}, b_{L_2}, b_{U_2}, \ldots, b_{L_m}, b_{U_m})^T$ es de dimensión $2m$.

Entonces el problema estándar que se genera es de la forma:

$$
\begin{aligned}
&min. \quad c^T x \\
&s.a. \\
&\qquad -f_1(x) + x_1^h = L_1 \\
&\qquad f_1(x) + x_2^h = U_1 \\
&\qquad -f_2(x) + x_3^h = L_2 \\
&\qquad f_2(x) + x_4^h = U_2 \\
&\qquad \vdots \\
&\qquad -f_m(x) + x_{2m-1}^h = L_m \\
&\qquad f_m(x) + x_{2m}^h = U_m \\
&\qquad\qquad l_1 \leq x_1 \leq u_1 \\
&\qquad\qquad l_2 \leq x_2 \leq u_2 \\
&\qquad\qquad \vdots \\
&\qquad\qquad l_n \leq x_n \leq u_n \\
&x_1^h, x_2^h, \ldots, x_{2m-1}^h, x_{2m}^h \geq 0
\end{aligned}
$$

Paso 2.- Cambio de variables.

Se busca que todas las variables del problema tengan su límite inferior igual a cero. Entonces para cada tipo de variable se hace un cambio de variable, presentándose dos casos:

A) Variables Acotadas con sus límites superior e inferior finitos.

Se hace un cambio de variable para obtener variables acotadas con su límite inferior igual a cero. Entonces para cada variable x_j, donde $j = 1, \ldots, n$ que cumpla $l_j \leq x_j \leq u_j$, hacer el cambio de variable $x_j = x'_j + l_j$, donde x'_j es una variable acotada por $0 \leq x'_j \leq \mu_j$ y $\mu_j = u_j - l_j$.

B) Variables Acotadas con uno de sus límites infinito. [1]

Se hace un cambio de variable para obtener variables positivas con límite inferior igual a cero y límite superior infinito. Entonces para cada variable x_j, donde $j = 1, \ldots, n$ que cumpla:

1.- $l_j \leq x_j \leq \infty$ y $l_j \neq 0$, hacer el cambio de variable $x_j = x'_j + l_j$, donde x'_j es una variable positiva con límites $0 \leq x'_j \leq \infty$.

2.- $l_j \leq x_j \leq u_i$ y $l_j = -\infty$, hacer el cambio de variable $x_j = u_j - x'_j$, donde x'_j es una variable positiva con límites $0 \leq x'_j \leq \infty$.

[1] Para resolver el problema de la dieta alimenticia, este caso no es necesario considerarlo por las mismas características del problema.

Paso 3.- Encontrar una solución factible inicial.

Necesitamos encontrar una solución básica factible x_B para este problema lineal, que corresponda a una base $B = I$.

A) Tabla Inicial.
Sean:

$$\begin{array}{ll} \hat{I} & = \{i \in \{1,\ldots,2m\}\colon x_i \in x_B\} \quad = \{n+1, n+2, \ldots, n+2m\} \\ J_0 & = \{j \in \{1,\ldots,n\}\colon x_j = 0\} \quad = \{1,2,\ldots,n\} \\ J_u & = \{j \in \{1,\ldots,n\}\colon x_j = \mu_j\} \quad = \emptyset \\ L_u & = \{l \in \{1,\ldots,n\}\colon \mu_j \neq \infty\} \end{array}$$

Donde:

μ_j es un conjunto de límites superiores de las variables x_j, si $j \in L_u$
$B = a^{n+1}, a^{n+2}, \ldots, a^{n+2m}\} = I$ es la base inicial
a^j es un vector columna de la matriz A y $j \in \hat{I}$ para calcular el vector x_B
$x_B = B^{-1}b'$
$x_N = 0$
$Y = A' \setminus I = \{y^1, y^2, \ldots, y^n\}$ es la matriz de multiplicadores
$C = c^1, c^2, \ldots, c^n$ es el vector de costos
$C_B = 0$

B) Verificación de la solución.

SI x_B es una solución básica factible no degenerada, ENTONCES IR A LA FASE II
EN OTRO CASO, continuar.

C) Cálculo de la solución básica extendida.

Como inicialmente $J_0 = \{1,\ldots,n\}$ y $J_u = \emptyset$, entonces se pasan a su cota superior todas las variables que se encuentren en su cota inferior una por una, hasta encontrar una solución básica inicial. Es decir, para todo $j \in J_0$ hacer $j \in J_u$ y $j \notin J_0$.

SI x_B es una solución básica factible no degenerada, ENTONCES IR A LA FASE II
EN OTRO CASO, continuar.

SI no se encuentra una solución básica factible, ENTONCES TERMINAR porque el problema no tiene una solución básica factible.

5.2 Segunda Fase

Paso 1.- Asignación de precios.

Se realiza el cálculo de costos reducidos. Para cada una de las variables no básicas x_j hacer $\varsigma^j = -c^j$, donde $j \in J = J_0 \cup J_u$, $\varsigma = \{\varsigma^1, \varsigma^2, \ldots, \varsigma^n\}$ el vector de costos reducidos.

Paso 2.- Comprobación del óptimo.

Hacer $\alpha = max\{max_{j \in J_0}\{\varsigma^j\}, max_{j \in J_u}\{-\varsigma^j\}\}$
SI $\alpha < 0$, ENTONCES IR A LA FASE III porque esta es la solución óptima para el problema.
SI $\alpha > 0$, ENTONCES continuar.

Paso 3.- Determinación de la columna de pivoteo.

La variable no básica x^k puede entrar a la base, pues encuentra una dirección de descenso. Entonces si α del paso anterior es mayor que cero, entonces la variable x^k es candidata para entrar a la base.

Paso 4.- Determinación de la fila de pivoteo.

A) Valor de δ.

SI $\varsigma^k > 0$, ENTONCES $\delta = 1$
SI $\varsigma^k < 0$, ENTONCES $\delta = -1$

B) Determinar la variable básica x_i^k que sale de la base haciendo:

$\theta_0 = min\{\ min\{x_i/\delta y_i^k\}\ para\ \delta y_i^k > 0,\ min\{(\mu_i - x_i)/ - \delta y_i^k\}\ si\ i \in L_u\ y\ para\ \delta y_i^k < 0,\ \mu_k\ sik \in L_u\}$

C) Evaluación de θ_0.

SI $\theta_0 = \mu_k$ ENTONCES no hay cambio en la base y x^k pasa a su cota contraria. IR AL PASO 6.

SI $\theta_0 = x_i/\delta y_i^k$ ENTONCES x_i sale de la base a nivel cero y entra x^k con valor:

$$\begin{cases} \theta_0 & \text{si } \delta > 0 \\ \mu_k - \theta_0 & \text{si } \delta < 0 \end{cases}$$

SI $\theta_0 = (\mu_i - x_i)/ - \delta y_i^k$ ENTONCES x_i sale de la base a su nivel

μ_i y entra x^k con valor:

$$\begin{cases} \theta_0 & \text{si } \delta > 0 \\ \mu_k - \theta_0 & \text{si } \delta < 0 \end{cases}$$

Paso 5.- Pivoteo.

Ahora el pivote es y_i^k y entonces se hace el pivoteo como sigue:

$y_r^l = y_r^l - (y_i^l * y_r^k)/y_i^k$ para todo $l \in J$ y $r \in \hat{I}$.
$\varsigma^l = \varsigma^l - (y_i^l * \varsigma^k)/y_i^k$
$x_r = x_r - (y_r^k * x_i)/y_i^k$

Paso 6.- Adaptar solución óptima.

$x'_B = x_B - \mu_k y_i^k$ para todos $i \in J_u$
IR AL PASO 2.

5.3 Tercera Fase

Paso 1.- Recuperar solución óptima del problema original.

Ahora que se ha encontrado la solución óptima del problema equivalente, se calcula la solución óptima del problema original.

A) Cambio de variable para variables acotadas.

Para cada x'_j donde $j \in J$ y $0 \leq x'_j \leq \mu_j$, hacer el cambio de variable $x'_j = x_j - l_j$, dado que $l_j \leq x_j \leq u_j$, $u_j = \mu_j + l_j$, la solución es $x = x_{\hat{I}} \cup x_J$, $x_{\hat{I}} = x'_B$ y $x_J = 0$.

B) Cambio de variable para variables positivas. [2]

Si $\mu_j = \infty$ y $l_j \neq 0$ entonces hacer $x'_j = x_j + l_j$, donde $l_j \leq x_j \leq \infty$.

Si $l_j = -\infty$ entonces hacer $x'_j = u_j - x_j$, donde $-\infty \leq x_j \leq u_j$.

[2] Para resolver el problema de la dieta alimenticia, este caso no es necesario considerarlo por las mismas características del problema.

Capítulo 6

Pseudocódigo

En este capítulo, se muestra el pseudocódigo de las rutinas que se implementaron para el algoritmo que da solución al **Problema de la Dieta Alimenticia.**

El pseudocódigo recibe un problema de la forma:

$$
\begin{aligned}
&min. \quad c^T x \\
&s.a. \\
&\qquad L_1 \leq f_1(x) \leq U_1 \\
&\qquad L_2 \leq f_2(x) \leq U_2 \\
&\qquad\qquad \vdots \\
&\qquad L_m \leq f_m(x) \leq U_m \\
&\qquad l_1 \leq x_1 \leq u_1 \\
&\qquad l_2 \leq x_2 \leq u_2 \\
&\qquad\qquad \vdots \\
&\qquad l_n \leq x_n \leq u_n
\end{aligned}
$$

El resultado que se obtiene es x, el vector de soluciones que corresponde a cada una de las incógnitas del problema.

Para cada una de las rutinas que se presentan, se explica su funcionamiento, cuales son sus parámetros de entrada y salida (si es que los necesita), su utilización dentro de un programa y su pseudocódigo.

Estas rutinas sólo requieren ser transportadas a algún lenguaje de programación y probar el algoritmo.

6.1 Variables

Las variables que utiliza el algoritmo son las siguientes:

m	número de restricciones.
n	número de incognitas.
costos[1..n]	coeficientes de costos de la función objetivo.
LRest[1..m]	límite inferior de las restricciones.
URest[1..m]	límite superior de las restricciones.
MatrizA[0..2m][0..n]	matriz de coeficientes de las restricciones. La posición $i = 0$ se utiliza para guardar el número de variable básica a la que pertenece la fila. La posición $j = 0$ se utiliza para guardar el número de variable no básica a la que pertenece la columna.
LVar[1..n]	límite inferior de las variables.
UVar[1..n]	límite superior de las variables.
ConstanteFuncion	constante que forma parte de la función objetivo al realizar el cambio de variable.
IndicesUVar[1..n]	vector de índices de las variables con cota superior finita. Cada casilla representa una variable. Si la casilla vale 1, entonces el límite superior es finito. Si la casilla vale -1, entonces el límite superior es ∞.
UVarEstándar[1..n]	vector de cotas superiores de las variables al realizar el cambio de variable. Está directamente relacionado con IndicesUVar.
Vectorb[1..2m]	vector b del problema estándar.
Status[1..n+2m]	vector de estados de todas las variables. Si la casilla vale 0, entonces la variable se encuentra en su cota inferior. Si la casilla vale 1, entonces la variable se encuentra en su cota superior. Si la casilla vale -1, entonces es básica o es ∞.

6.2 Rutinas

SIMPLEX : Rutina principal que sigue paso a paso el algoritmo del capítulo anterior.

Forma de llamado:
SIMPLEX()

Pseudocódigo:

```
PedirDatos()
EquivalenteEstandar()
CambioVar()
Si NO SolucionInicial() Entonces
        Imprimir El problema no tiene una solucion basica factible
En otro caso
        AsignarPrecios()
        Mientras (ColumnaPivote ←ComprobarOptimo()) ≠ -1 Entonces
                FilaPivoteo(ColumnaPivote)
                Si NO ValidaSolucion() Entonces
                        error y terminar
        Repetir
        RecuperarSolucion()
```

EquivalenteEstandar : Esta rutina se encarga de transformar el problema original a un problema equivalente en la forma estándar.

Las restricciones pasan de la forma $L \leq f(x) \leq U$ a la forma $f(x) + x^h = b$.

Forma de llamado:
EquivalenteEstandar()

Pseudocódigo:

```
Para i de 1 hasta n
    Para j de 1 hasta m
        MatrizA[j+m][i] ← MatrizA[j][i]*(-1)
Para i de 1 hasta m
    Vectorb[i] ← URes[i]
    Vectorb[i+m] ← LRes[i]
```

CambioVar : Esta rutina realiza el cambio de variable para cada una de las variables acotadas. Las variables pasan de estar acotadas de la forma $l \leq x \leq u$ a $0 \leq x' \leq \mu$.

Forma de llamado:

CambioVar()

Pseudocódigo:

```
ConstanteFuncion ← 0
Para i de 1 hasta n
    Si IndicesUVar[i] ≠ -1 Entonces
        UVarEstandar[i] ← UVar[i]-LVar[i]
        ConstanteFuncion ← ConstanteFuncion + costos[i]*LVar[i]
Para i de 1 hasta 2m
    Para j de 1 hasta n
        Vectorb[i] ← Vectorb[i] - costos[i]*LVar[i]
```

SolucionInicial : Esta rutina busca una solución factible inicial x_B para el problema lineal que corresponda a la base $B = I$.

Parámetros de salida:

Inicio	Es un valor de verdad que indica si existe o no una solución inicial para el problema que se quiere resolver.

Forma de llamado:

valor ← SolucionInicial()

Pseudocódigo:

```
Para i de 1 hasta 2m
    Matriz[i][0] ← i+n
    Status[n+i] ← -1
Para i de 1 hasta n
    Matriz[0][i] ← i
    Status[i] ← 0
Si ValidaSolucion() Entonces
    regresa VERDADERO
En otro caso
    Si SolucionExtendida() Entonces
        regresa VERDADERO
    En otro caso
        regresa FALSO
```

AsignarPrecios : Esta rutina es para inicializar los costos reducidos en el tableau. El vector de costos reducidos es calculado a partir de los multiplicadores $\pi^T = C_B^T B^{-1}$, con base en la ecuación $\varsigma^j = c^j - \pi^T a_j$, para $j \in J = J_0 \cup J_u$.

Forma de llamado:

AsignarPrecios()

Pseudocódigo:

```
Para i de 1 hasta n
    costos[i] ← costos[i]*-1
```

ComprobarOptimo : La función de ésta rutina es comprobar que se ha llegado al óptimo. Si ς^q es negativo hemos llegado al óptimo, en otro caso es necesario seguir pivoteando.

Parámetros de salida:

ColumnaPivoteo	Vale -1 si el punto donde su ubica la solución actual del problema es la solución óptima, en otro caso tiene un valor positivo que es el número de la variable que va a entrar a la base.

Forma de llamado:

valor ← ComprobarOptimo()

Pseudocódigo:

```
varaux ← 1
Si Status[varaux]=0 Entonces
    varaux2← costos[varaux]
Si Status[varaux]=1 Entonces
    varaux2← costos[varaux]*-1
Para i de 2 hasta n+2m
    Si Status[i]=0 y varaux2<costos[i] Entonces
        varaux2 ← costos[i]
        varaux ← i
    Si Status[i]=1 y varaux2<costos[i]*-1 Entonces
        varaux2 ← costos[i]*-1
        varaux ← i
Si costos[varaux]>0 Entonces
    regresa varaux
En otro caso
    regresa -1
```

FilaPivoteo : Determina el número de la variable que sale de la base. Si existe dicha variable, entonces realiza el pivoteo.

$$\theta = min\{min_{\delta y_i^k > 0}\{\tilde{x}_i / \delta y_i^k\}, min_{\delta y_i^k < 0}\{(u_i - \tilde{x}_i)/ - \delta y_i^k\}, u_k\}$$

Parámetros de entrada:

VarIn	Es el número de la variable que entra a la base.

Forma de llamado:
FilaPivoteo(ColumnaPivoteo)

Pseudocódigo:

```
IndVarIn ← casilla de la variable de entrada
Si costos[IndVarIn]>0 Entonces
    sigma ← 1
Si costos[IndVarIn]<0 Entonces
    sigma ← -1
IndVarOut ← 1
NuevoMultiplicador ← sigma*MatrizA[IndVarOut][IndVarIn]
Si NuevoMultiplicador>0 Entonces
      TetaCero ← Vectrob[IndVarOut]/NuevoMultiplicador
      NumCaso ← 1
Si (VarOut ← variable basica) ≠ -1 Entonces
      Si NuevoMultiplicador<0 Entonces
            TetaCero ← (UVarEstandar[VarOut] - Vectorb[i])/
                                  NuevoMultiplicador*-1
            NumCaso ← 2
Si (VarOut ← variable que entra) ≠ -1 Entonces
      TetaCero ← UVarEstandar[VarOut]
      NumCaso ← 3
Para i de 2 hasta n+2m
      NuevoMultiplicador ← sigma*MatrizA[i][IndVarIn]
      Si NuevoMultiplicador>0 Entonces
            NuevoTetaCero ← Vectorb[i]/NuevoMultiplicador
            Si TetaCero > NuevoTetaCero Entonces
                  TetaCero ← NuevoTetaCero
                  NumCaso ← 1
                  IndVarOut ← i
      Si (VarOut ← variable basica) ≠ -1 Entonces
            Si NuevoMultiplicador<0 Entonces
                  NuevoTetaCero ← (UVarEstandar[VarOut] - Vectorb[i])/
                                        NuevoMultiplicador*-1
                  Si TetaCero > NuevoTetaCero Entonces
                        TetaCero ← NuevoTetaCero
```

```
                    NumCaso ← 2
                    IndVarOut ← i
        Si (VarOut ← variable que entra) ≠ -1 Entonces
            NuevoTetaCero ← UVarEstandar[VarOut]
            Si TetaCero>NuevoTetaCero Entonces
                TetaCero ← NuevoTetaCero
                NumCaso ← 3
                IndVarOut ← i
VarOut ← variable que sale
Si NumCaso=1 Entonces
    HacerPivoteo(VarIn,VarOut)
    Status[VarOut] ← 0
    Status[VarIn] ← -1
Si NumCaso=2 Entonces
    HacerPivoteo(VarIn,VarOut)
    Status[VarOut] ← 1
    Status[VarIn] ← -1
Si NumCaso=1 Entonces
    Si Status[VarIn]=0 Entonces
        Status[VarIn] ← 1
    En otro caso
        Si Status[VarIn]=1 Entonces
            Status[VarIn] ← 0
```

HacerPivoteo : Esta rutina se encarga de hacer el pivoteo en el tableau, teniendo el pivote, la variable que entra y la variable que sale de la base.

$$\theta = x_r^B / y_r^k$$
$$\check{x}_j^B = x_j^B - \theta y_j^k, \quad j = 1 \ldots m$$
$$\check{c}_j^B = c_j^B - \theta c_j, \quad j = 1 \ldots n$$

Parámetros de entrada:

VarIn	Es el número de la variable que entra a la base.
VarOut	Es el número de la variable que sale de la base.

Forma de llamado:
HacerPivoteo(VarIn,VarOut)

Pseudocódigo:

```
IndVarOut ← ubicacion en la base de VarOut
IndVarIn ← ubicacion de VarIn
Pivote ← MatrizA[IndVarOut][IndVarIn]
Para i de 1 hasta n
    Para j de 1 hasta2m
        Si i ≠ IndVarIn y j ≠ IndVarOut Entonces
                MatrizA[j][i] ← MatrizA[j][i]-(MatrizA[IndVarOut][i]
                    *MatrizA[j][IndVarIn]/Pivote)
Para i de 1 hasta 2m
    Si i ≠ IndVarOut Entonces
            MatrizA[i][IndVarIn] ← MatrizA[i][IndVarIn]/
                Pivote*-1
Para i de 1 hasta n
    Vectorb[i] ← Vectorb[i]-MatrizA[i][IndVarIn]*
                Vectorb[IndVarOut]/Pivote
MatrizA[0][IndVarIn] ← VarOut
MatrizA[IndVarOut][0] ← VarIn
```

ValidaSolucion : En esta rutina, la solución x_B es evaluada para verificar su validez, es decir, para determinar si es una solución básica factible.

Si ς^q es negativo y el vector $y = B^{-1}a_q$ es no positivo entonces el problema lineal no es acotado. $x(\theta)$ es factible para todo $\theta \geq 0$ y la función objetivo tiende a $-\infty$ cuando $\theta \to \infty$. En este caso la solución es una dirección d con $c^T d = \varsigma^q < 0$.

Parámetros de salida:

Valida	Es VERDADERO si la solución obtenida es válida. Es FALSO si no es una solución factible.

Forma de llamado:

valor ← ValidaSolucion()

Pseudocódigo:

```
Para i de 1 hasta 2m
    varaux ← 0
    Para j de 1 hasta n
        Si Status[j]=1 Entonces
            varaux ← varaux+Vectorb[i]-UVarEstandar[j]*
                MatrizA[i][j]
    Si varaux≥0 Entonces
        varaux2 ← cota superior de la variable
        Si varaux2≠-1 Entonces
            Si varaux>varaux2 Entonces
                regresa FALSO
    En otro caso
        regresa FALSO
regresa VERDADERO
```

RecuperaSolucion : Esta rutina realiza nuevamente el cambio de variable para recuperar la solución del problema original. Todas las variables pasan de ser acotadas por $0 \leq x'_j \leq \mu_j$ a $l_j \leq x_j \leq u_j$, aplicando el cambio de variable $x'_j = x_j - l_j$.

Forma de llamado:
RecuperaSolucion()

Pseudocódigo:

```
Para i de 1 hasta n
    Si Status[i]=-1 Entonces
        IndiceAux ← numero de variable
        costos[i] ← Vectorb[IndiceAux]
    Si Status[i]=0 Entonces
        IndiceAux ← numero de variable
        costos[i] ← LVar[IndiceAux]
    Si Status[i]=1 Entonces
        Si tiene cota superior Entonces
            Imprimir El problema no tiene solucion finita
        En otro caso
            IndiceAux ← numero de variable
            costos[i] ← UVarEstandar[IndiceAux]+LVar[IndiceAux]
```

Capítulo 7

Pruebas

En el presente capítulo se presentan los resultados obtenidos al aplicar las diferentes pruebas al algoritmo para la Solución del Problema de la Dieta Alimenticia: **DIETEX**.

Se generaron distintos problemas prueba y el algoritmo DIETEX los resolvió ejecutandose en una computadora 486DX a 100MHz con 8Mb de memoria RAM y disco duro de 500Mb. En cada ejecución se mide:

1. **Eficiencia**
   (a) Número total de iteraciones.
   (b) Número de evaluaciones de la función objetivo.
   (c) Tiempo de respuesta.

2. **Solución**
   (a) Cantidad óptima de cada alimento de la dieta alimenticia.
   (b) Costo total de la dieta alimenticia óptima.

Al generar cada uno de los problemas prueba, se consideraron los siguientes aspectos para cada uno de ellos:

- 1^{er} *Problema Prueba*

  Para un problema con 64 variables con cotas inferiores positivas o iguales a cero y cotas superiores positivas, y 11 restricciones con cotas superiores e inferiores positivas y diferentes de cero se evalua:

  1. El número de iteraciones.
  2. El número evaluaciones de función.
  3. El tiempo de respuesta.
  4. El tipo de solución encontrada.
  5. El valor del óptimo (cantidades y costo total).

- 2^{o} *Problema Prueba*

  Para un problema con 24 variables con cotas inferiores y superiores positivas, 11 restricciones con cotas superiores e inferiores positivas y diferentes de cero, y sin solución óptima se evalua :

  1. El tipo de solución encontrada.
  2. En caso de no tener solución inicial, el tiempo empleado para localizar una solución inicial básica extendida.
  3. El número de iteraciones.
  4. El número de evaluaciones de función.
  5. El tiempo de respuesta total.

Los resultados obtenidos al aplicar esta prueba son los siguientes:

Evaluación de desempeño

Tipo Sol.: NO EXISTE SOLUCION INICIAL BASICA FACTIBLE
Iteraciones de fase 1: 8388608
Iteraciones de fase 2: 0
Iteraciones de fase 3: 0
Evaluaciones de función en fase 1: 0
Evaluaciones de función en fase 2: 0
Evaluaciones de función en fase 3: 0
Tiempo de Respuesta: 249.00000000 segundos

Evaluación de solución

NO EXISTE SOLUCION INICIAL BASICA FACTIBLE

7.3 3^{er} Problema Prueba

Para realizar la tercera prueba del algoritmo, se generó un problema con las siguientes características:

- Variables con cotas inferiores positivas o iguales a cero y cotas superiores positivas.
- Restricciones con cotas superiores e inferiores positivas y diferentes de cero.
- Dimensión n=32.
- Restricciones m=11.

El objetivo de esta tercera prueba es comprobar el desempeño del algoritmo al encontrarse con un óptimo igual a cero, para lo cual se toman en cuenta los siguientes aspectos al realizar la evaluación:

1. El número de iteraciones.
2. El número de evaluaciones de función.
3. El tiempo de respuesta.
4. El tipo de solución encontrada al problema.
5. El valor del óptimo (cantidades y costo total).

Los resultados obtenidos de esta prueba son los siguientes:

Evaluación de desempeño

El problema original tiene óptimo:0.0

Tipo Sol.: SOLUCION BASICA FACTIBLE OPTIMA FINITA

Iteraciones de fase 1: 1
Iteraciones de fase 2: 0
Iteraciones de fase 3: 1
Evaluaciones de función en fase 1: 0
Evaluaciones de función en fase 2: 0
Evaluaciones de función en fase 3: 1
Tiempo de Respuesta: 0.0000000 segundos

Evaluación de solución

X[1]=0.0000
X[2]=0.0000
X[3]=0.0000
X[4]=0.0000
X[5]=0.0000
X[6]=0.0000
X[7]=0.0000
X[8]=0.0000
X[9]=0.0000
X[10]=0.0000
X[11]=0.0000
X[12]=0.0000
X[13]=0.0000
X[14]=0.0000
X[15]=0.0000
X[16]=0.0000
X[17]=0.0000
X[18]=0.0000
X[19]=0.0000
X[20]=0.0000
X[21]=0.0000
X[22]=0.0000
X[23]=0.0000
X[24]=0.0000
X[25]=0.0000
X[26]=0.0000
X[27]=0.0000
X[28]=0.0000
X[29]=0.0000
X[30]=0.0000
X[31]=0.0000
X[32]=0.0000

Evaluación de función objetivo=0.0

Capítulo 8

Conclusiones

Este capítulo se ha destinado para escribir las conclusiones a las que se ha llegado, después de realizar el algoritmo DIETEX basado en el método SIMPLEX para la Optimización del Problema de la Dieta Alimenticia, así como realizar pruebas de su implementación, siendo este el objetivo del trabajo de tesis.

A continuación se describen los puntos relevantes con que se puede concluir este trabajo:

1. Una característica de DIETEX es que inicia buscando una solución inicial al problema dentro de su región factible.
2. A partir de la solución inicial encontrada el algoritmo se mueve hacia cualquier otro punto de solución que signifique mejorarla, minimizando el costo total de la dieta, resultado de evaluar la función objetivo.
3. En caso de no encontrar una solución inicial, el algoritmo busca una solución inicial básica extendida evaluando todas las posibles combinaciones de los valores de las variables en la solución.
4. El tiempo que mide DIETEX, es sólo el tiempo que ocupa el algoritmo para pasar por sus fases, sin incluir el tiempo que le tome accesar los archivos o en presentar los datos en la pantalla.
5. La unidad mínima de tiempo que manejan las funciones para medir el tiempo del lenguaje de programación usado, es el segundo. Por tal razón el algoritmo registra un tiempo de cero cuando se llevó menos de 1 segundo en el proceso.
6. El algoritmo realiza más iteraciones:
   (a) En la fase 1 cuando el problema no tiene una solución inicial básica factible. Esto es porque busca una solución extendida haciendo todas las combinaciones de las variables hasta que la encuentra.

(b) En la fase 2 cuando la solución inicial básica factible no es el óptimo.

7. El número de iteraciones que realiza DIETEX en 1 segundo es aproximadamente 33689, por esta razón DIETEX ocupa milesimas de segundo en el proceso. Si para encontrar la solución extendida de un problema con 28 variables se necesitó un tiempo de 249 segundos, entonces el tiempo se dispara exponencialmente para un problema con 64 o más variables. Siendo esta una desventaja que presenta este algoritmo. Para solucionar este problema sólo debe cambiarse el método de búqueda.

8. Cuando un problema tiene variables acotadas con cotas inferiores iguales a cero, la solución inicial es cero y es la solución óptima del problema.

9. Los problemas reales que cumplen con el modelo del Problema de la Dieta Alimenticia tienen costos positivos, pero con el objeto de probar el desempeño del algoritmo DIETEX, se generaron problemas prueba con costos negativos.

10. En los resultados obtenidos de resolver problemas con un función objetivo positiva se observó que el 80% de las veces la solución inicial básica factible es el óptimo.

11. Cuando el problema se ejecuta en el sistema Windows el proceso de ejecución ocupa más tiempo que cuando se ejecuta sobre el sistema MS-DOS.

12. El algoritmo tiene la capacidad de resolver problemas de hasta 1000 variables (alimentos) y 1000 restricciones (nutrientes), sin presentarse ninguna falla.

13. Los problemas prueba resueltos por el sistema DIETEX, también se resolvieron manualmente auxiliándose del programa TORA [TH5E], que es un programa educativo para resolver problemas lineales. Esto para comparar los resultados arrojados por el sistema DIETEX.
    La limitante del programa de apoyo, TORA, es que sólo tiene capacidad de resolver problemas pequeños: 45 variables, 44 restricciones y sus restricciones con una sola cota.

14. En internet se encontró un programa para resolver el Problema de la Dieta Alimenticia, que al igual que el sistema DIETEX sólo ha resuelto problemas prueba ficticios. Este programa nos permite resolver problemas de hasta 64 variables, 11 restricciones y tanto variables como restricciones con cota superior e inferior.
    La dirección electrónica en internet de este programa es:
    http://www.mcs.anl.gov/home/otc/Guide/CaseStudies/diet /table.htm

15. Se eligió el lenguaje de programación C para desarrollar el sistema DIETEX, porque es el que mayores alcances nos permite tener con respecto al manejo de memoria, de archivos, de variables y menores requerimientos de equipo tienen los programas ejecutables generados.

16. Inicialmente se pensó en el sistema DIETEX para red, pero el compilador de C en UNIX de la UTM nos presentó problemas de librerías. Por eso se optó por un sistema para PC.

17. El sistema DIETEX se programó en lenguaje C del Silicon Valley Software (SVS) [SVS91] porque:

    (a) Permite compilar programas con arreglos estáticos de tamaño mayor a 80 y con punto flotante, no siendo este el caso de cualquier otro compilador de C.

    (b) Al ejecutarse nuestras aplicaciones es posible que requieran más memoria de la disponible (este es el caso de DIETEX, para trabajar con problemas muy grandes). Para evitar fallas por sobreflujo en la pila u otros conflictos, nos permite recompilar nuestras aplicaciones definiendo una región de memoria de cualquier tamaño. Esta región definida es memoria virtual que podrán accesar las aplicaciones cuando se estén ejecutando.
    Ahora la limitante sería la capacidad del disco.

    (c) Es posible que nuestras aplicaciones accesen hasta 16Mb de memoria extendida.

18. Se pensó que el sistema DIETEX podría tener una interface en ambiente gráfico, pero no fué posible porque limitaba al sistema DIETEX a correr en determinados tipos de equipo y su capacidad de trabajar con problemas muy grandes. Así es que mejor se realizó en el compilador de Borland C++ ver. 3.1 [BC91].
    Para versiones posteriores del sistema DIETEX se piensa mejorar la apariencia de la interface.

19. Como se mencionó en el capítulo 1, el objetivo de esta tesis, era hacer la optimización del Problema de la Dieta Alimenticia partiendo del modelo matemático y del método SIMPLEX, y hacer la implementación del algoritmo resultante para realizar las pruebas. Finalmente el objetivo se ha cumplido, quedando expuesto a la resolución de problemas reales. Pues la optimización se realizó desde el punto de vista matemático pero el hecho de que la dieta resultante sea apropiada para una persona no se asegura, pues depende de los datos que contega el problema.

20. Fué una tarea difícil encontrar problemas prueba reales muy grandes (mayor de 500 variables y restricciones). Por tal razón se deja a los usuarios resolver otros problemas con el algoritmo.

Capítulo 9

Manual de Usuario

El sistema **DIETEX**, consta de 2 partes:

- El algoritmo que da solución al **Problema de la Dieta Alimenticia**, que es un problema con restricciones acotadas, variables acotadas y variables positivas.

- La interface del sistema **DIETEX** trabaja bajo ambiente de MS-DOS.

En este Capítulo se explica la forma de ejecución del sistema **DIETEX**. Se describen la forma de ejecución y funcionamiento de cada una de las partes del sistema, los parámetros de entrada necesarios, el formato que deben tener los archivos de entrada y salida, y además se muestra un ejemplo.

9.1 Ejecución y Funcionamiento

El presente manual está dirigido a todos los usuarios que se interesen por utilizar el sistema **DIETEX** para resolver problemas semejantes al **Problema de la Dieta Alimenticia**, con restricciones acotadas, variables acotadas y variables positivas, y tienen la forma:

$$
\begin{aligned}
&min. \quad c^T x \\
&s.a. \\
&\qquad L_1 \leq f_1(x) \leq U_1 \\
&\qquad L_2 \leq f_2(x) \leq U_2 \\
&\qquad\qquad \vdots \\
&\qquad L_m \leq f_m(x) \leq U_m \\
&\qquad l_1 \leq x_1 \leq u_1 \\
&\qquad l_2 \leq x_2 \leq u_2 \\
&\qquad\qquad \vdots \\
&\qquad l_n \leq x_n \leq u_n
\end{aligned}
$$

El sistema **DIETEX** fué implementado en el lenguaje de programación C del Silicon Valley Software (SVS) [SVS91] y Borland C++ ver. 3.1 [BC91]. Los requerimentos mínimos de equipo son:

1. Una computadora 486 Dx o posteriores.
2. Coprocesador matemático.
3. Monitor VGA color.
4. 8 Mb de memoria RAM mínimo.
5. Disco duro de 120Mb mínimo.
6. Sistema Operativo MS-DOS o WINDOWS.

Para trabajar con el sistema **DIETEX** necesita:

1. 400 Kb de disco para copiar los archivos del sistema **DIETEX**.
2. 8Mb libres en disco duro para resolver problemas de hasta 1000 alimentos (variables) y 1000 nutrientes (restricciones).
3. Espacio disponible para guardar el problema y para que **DIETEX** guarde los archivos de desempeño y solución.

Además copiar los archivos siguientes en su disco duro:

- C:\DIETEX\DIETEX.EXE y
- C:\DIETEX\SPXPSVS.EXE

Ahora ya está listo para empezar a trabajar, para lo cual deberá ejecutar: *DIETEX < return >*

El menú principal de **DIETEX** es el siguiente:

FIG. 1

La opción NUEVO se utiliza para capturar un nuevo problema. **DIETEX** crea un archivo con extensión .spx en el directorio C:\DIETEX.

La opción ABRIR se usa para cargar en memoria un problema existente. El usuario no puede especificar una ruta para el archivo que desea abrir, así que el problema debe encontrarse en C:\DIETEX.

La opción SALIR se usa para salir de **DIETEX**.

Si elige la opción NUEVO, entonces aparecerá en su pantalla lo siguiente:

FIG. 2

Si presiona 'N' entonces regresará al menú principal mostrado en la FIG. 1.

Si presiona 'S' entonces aparecerá la siguiente ventana:

FIG. 3

Deberá teclear el nombre del archivo sin la extensión de máximo 8 caracteres. A continuación se teclean los datos del problema. Cada una de las ventanas siguientes muestran las instrucciones correspondientes.

Posteriormente aparecerá la siguiente ventana:

FIG. 4

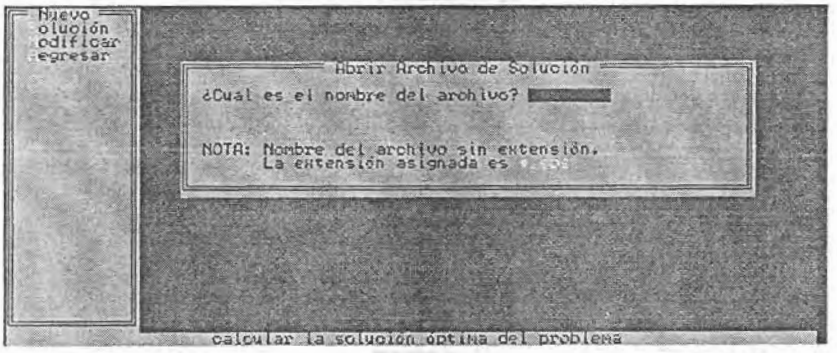
La opción SOLUCIONAR resuelve el problema.

La opción MODIFICAR permite cambiar todos los valores del problema, excepto el número de alimentos (n) y el número de nutrientes (m).

Siempre que elija la opción REGRESAR, le llevará a la pantalla anterior.

Antes de resolver el problema se le mostrará lo siguiente:

FIG. 5



Se teclea el nombre del archivo sin extensión. Este es el nombre del archivo de solución y de desempeño.

Cuando el problema ha sido resuelto se muestra la FIG. 6, donde la opción CANTIDADES muestra las porciones óptimas de cada alimento y la opción COSTO muestra el costo total de la dieta. En caso de que no se haya encontrado una solución, se muestra la FIG. 7.

FIG. 6

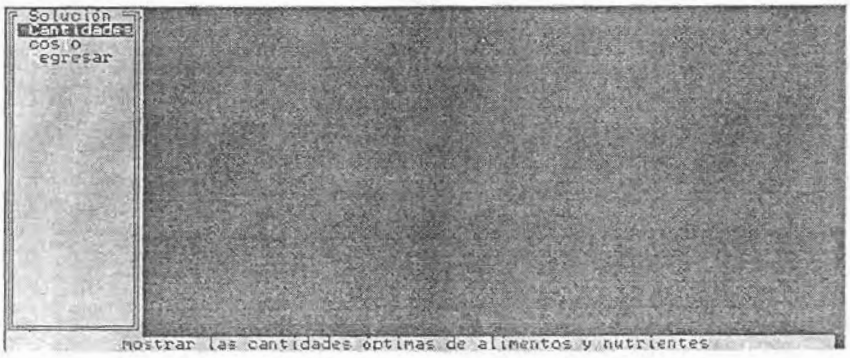
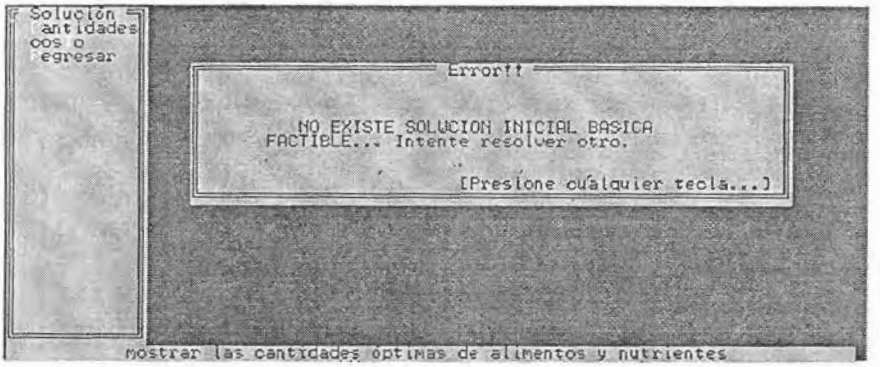


FIG. 7



Las figuras 8 y 9 muestran la solución de un problema.

FIG. 8

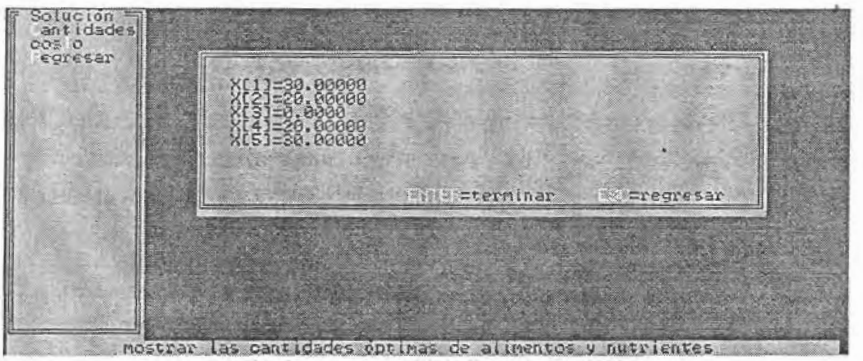
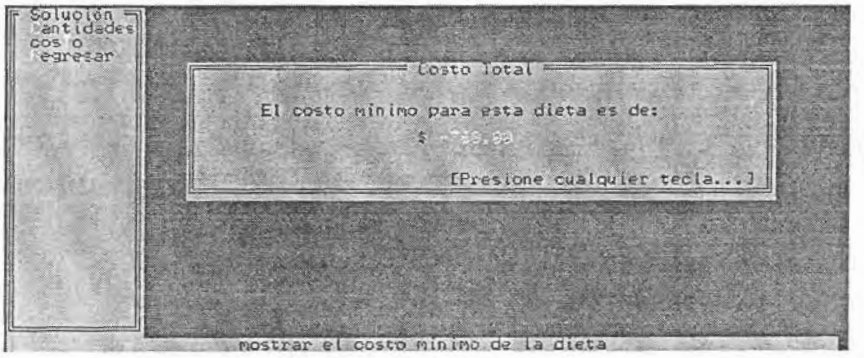
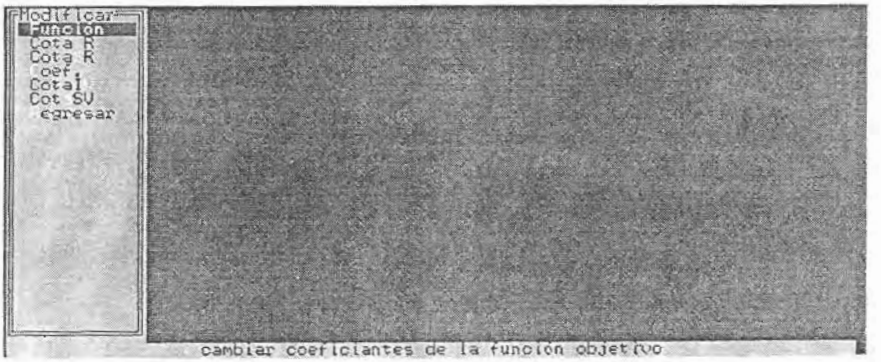


FIG. 9



La siguiente ventana muestra las opciones de modificar. Cada una de las opciones permite modificar solo los datos del problema que serán afectados.

FIG. 10



Algunas consideraciones importantes son las siguientes:

1. Cada opción lo lleva a diferentes submenues, y en la parte inferior de la pantalla podrá observar una breve descripción de la opción seleccionada.
2. Para moverse en las opciones de los menús, sólo deberá utilizar las teclas ↑ ó ↓.
3. Para elegir una opción, selecciónela y presione < *enter* > o presione la letra resaltada de la opción.
4. No podrá introducir valores negativos desde el teclado. Si por alguna razón esto es necesario, puede hacerlo desde el archivo del problema, pero sólo está permitido en los costos; de otro modo el sistema le reportará un error.
5. Cada vez que suceda algún tipo de error, el sistema realice una operación o el usuario deba introducir un dato al sistema, este mostrará un cuadro de mensajes con una breve descripción de lo que ocurre.
6. Cuando un problema no tiene solución o no tiene un óptimo finito, usted puede saberlo cuando intente ver las cantidades y el costo, abriendo el archivo de solución que tiene la extensión *.SPS o abriendo el archivo de desempeño con extensión *.SPM.

NOTA: para otras operaciones se recomienda ver la sección *"Resumen de instrucciones"*.

9.2 Parámetros de Entrada

Es posible que no quiera usar la interface del sistema **DIETEX**, únicamente se interese en resolver problemas contenidos en archivos. Esto es posible ejecutando lo siguiente:

Caso 1.

SPXPSVS *"problema" "solución"* $<$ *return* $>$.

ó

Caso 2.

SPXPSVS *"problema"* $<$ *return* $>$.

Donde:

- *"problema"* es el nombre del archivo con extensión *.SPX que contiene al problema que se quiere resolver.
- *"solución"* es el nombre del archivo con extensión *.SPS donde se guardará la solución del problema. El archivo de desempeño toma el mismo nombre pero con la extesión *.SPM

Si elige el caso 1, los archivos de solución y desempeño pueden tener un nombre diferente.

Pero si se elige el caso 2, todos los archivos tendrán el mismo nombre aunque diferente extensión.

9.3 Archivos de Entrada

DIETEX necesita que cualquier problema que quiera resolver se encuentre en un archivo, por esta razón cuando se elige introducir un nuevo problema, debe escribir también el nombre del archivo.

El formato que deben cumplir el archivo problema es muy importante, pues el sistema puede detectar errores y no podrá resolverlos.

El archivo de entrada es un archivo de texto que contiene una línea por dato, así que el tamaño del archivo depende del número de variables y de restricciones que contenga el problema.

Se recomienda al usuario que para facilitar su tarea de capturar el archivo problema, siempre y cuando contenga menos de 80 variables y 80 restricciones, usar el sistema **DIETEX**, pues el sistema le pedirá dato por dato.

En caso de que el problema sea muy grande y si así lo desea el usuario, puede crear el archivo problema desde cualquier editor de texto, pero recuerde que la extensión debe ser *.SPX y cumpliendo con el formato que se muestra a continuación:

ORDEN DE LOS DATOS POR LÍNEA DEL ARCHIVO	
Línea núm.	**Contenido**
1	valor de n o número de variables.
2	valor de m o número de restricciones.
3 a $(2+n)$	valor de los costos de cada variable.
$(3+n)$ a $(2+n+m)$	valor de los mínimos de las restricciones.
$(3+n+m)$ a $(2+n+m+n*m)$	valor de los coeficientes de las restricciones.
$(3+n+m+n*m)$ a $(2+n+2m+n*m)$	valor de los máximos de las restricciones.
$(3+n+2m+n*m)$ a $(2+2n+2m+n*m)$	valor de las cotas inferiores de las variables.
$(3+n+2m+n*m)$ a $(2+2n+2m+n*m)$	valor de las cotas superiores de las variables.

9.4 Archivos de Salida

Después de haber ejecutado el algoritmo **DIETEX**, se generan 2 archivos de salida. Estos son archivos de texto que **DIETEX** utiliza como sigue:

- La solución se guarda en un archivo con extensión *.SPS. En este archivo se encuentran las cantidades óptimas de cada alimento que constituyen la dieta alimenticia y que son las primeras n líneas del archivo; así como también, el costo total de la dieta alimenticia óptima que es la última línea del archivo.
- Para el usuario que le interese el desempeño del algoritmo, se crea un archivo con el mismo nombre del archivo solución, pero con extensión *.SPM, donde se registra la medida de desempeño del algoritmo durante su ejecución al resolver el problema. En este archivo se encuentra el desplazamiento de la solución entre iteraciones, el óptimo del problema original si es que lo encuentra, el tipo de solución encontrada, el número de iteraciones en cada fase, el número evaluaciones a la función objetivo en cada fase y el tiempo de proceso usado por algoritmo.

Cuando se obtiene una solución óptima finita, el archivo de solución presenta las características mencionadas anteriormente.

En caso de que el problema no tenga una solución básica factible, entonces el archivo solución contendrá el mensaje: "NO EXISTE SOLUCION INICIAL BASICA FACTIBLE".

Si el problema no tiene solución óptima finita, entonces el archivo tendrá el mensaje: "NO EXISTE SOLUCION BASICA FACTIBLE OPTIMA FINITA".

Si no se encuentra una solución válida al problema, es decir que la solución se salió de la región factible, entonces el archivo solución tendrá el siguiente mensaje: "NO EXISTE SOLUCION VALIDA".

9.5 Ejemplos

9.5.1 Ejemplo

Como ejemplo se muestra la ejecución de un problema prueba que se encuentra en el archivo prob1.spx.

Forma de ejecución 1:

DIETEX < *enter* >
A & prob1 < *enter* > & S & prob1 < *enter* >

Forma de ejecución 2:

SPXPSVS prob1.spx prob1.sps < *enter* >

Donde:

- El nombre del archivo problema prueba es *prob1.spx*. Se trata de un problema con 5 variables acotadas, 3 restricciones acotadas y costos negativos.
- El archivo solución de salida es *prob1.sps*.
- El archivo desempeño de salida es *prob1.spm*.

El archivo problema de entrada **Prob1.spx** es:

```
5
3
-10
-6
-3
-5
-8
30
20
70
1
1
1
1
1
1
```

1
0
0
0
0
0
1
1
1
100
50
80
0
0
0
0
0
30
25
25
30
30

El archivo solución de salida **Prob1.sps** es:

X[1]=30.00000
X[2]=20.00000
X[3]=0.0000
X[4]=20.00000
X[5]=30.00000

Evaluación de función objetivo=-760.00

El archivo desempeño de salida **Prob1.spm** es:

Evaluación de función objetivo = 0.0000
Evaluación de función objetivo = 0.0000
Evaluación de función objetivo = 0.0000
Evaluación de función objetivo = -300.00000

El problema original tiene óptimo:-760.00

Tipo Sol.: SOLUCION BASICA FACTIBLE OPTIMA FINITA

Iteraciones de fase 1: 1
Iteraciones de fase 2: 4
Iteraciones de fase 3: 1
Evaluaciones de función en fase 1: 0
Evaluaciones de función en fase 2: 4
Evaluaciones de función en fase 3: 1
Tiempo de Respuesta: 0.0000000 segundos

9.5.2 Ejemplo

Este es un ejemplo de la aplicación "real" de **DIETEX** en un menú. En la siguiente tabla se muestra el problema:

Nombre	Significado	Medida	Costo($)/ Porción	Límite Inferior	Límite Superior	Porción		Costo($) Total Alimento
x1	Zanahoria cruda rallada	1/8 Kg.	10	1	30	1	1/8 Kg.	10
x2	Lechuga cruda	1 hoja	6	5	25	5	5 hojas	30
x3	Pastel de manzana	50 g.	3	1	25	1	50 g.	3
x4	Pollo rostizado	50 g.	5	3	30	3	150 g.	15
x5	Espaguetis	7 g.	8	7	30	7	49 g.	56

Las columnas **Nombre** y **Significado** nos enlistan todos los alimentos que queremos que contenga nuestro menú. Para **DIETEX** cada alimento es una variable. La columna **Medida** se refiere a la cantidad de alimento por porción. La columna **Costo($)/Porción** muestra el costo de cada porción de alimento. Los datos de las columnas **Límite Superior** y **Límite Inferior** se refieren a el número de porciones mínimas y máximas permitidas de cada alimento. Las columnas **Porción** y **Costo($)/Porción** corresponden a la solución encontrada por **DIETEX**. Para interpretar los resultados es necesario multiplicar el número de porciones por el **Costo($)/Porción** para obtener el **Costo($) Total/Alimento**, y el número de porciones por la **Medida** para determinar la cantidad de cada alimento a utilizar en el menú. El costo total del menú es la suma de todos los costos totales por alimento, resultando en este caso $114.00.

En la siguiente tabla se muestran las restricciones sobre los nutrientes de cada alimento para este menú:

Nutrientes	Proteínas	Calcio	Grasas
Mínimo	30 g.	20 mg.	70 g.
x1	1	1	0
x2	1	1	0
x3	1	0	1
x4	1	0	1
x5	1	0	1
Máximo	100 g.	50 mg.	80 g.

Estas restricciones representan la cantidad de nutrientes por alimento y la cantidad total mínima y máxima permitida para la alimentación diaria de una persona. Estas deben ser definidas por un nutriólogo. Para nuestro ejemplo los datos de las restricciones fueron establecidos sin considerar la opinión de un nutriólogo.

Para resolver este problema con el sistema **DIETEX**, se capturarón los datos en el archivo Menu.spx y se obtuvieron los siguientes resultados:

El archivo problema de entrada **Menu.spx** es:

```
5
3
10
6
3
5
8
30
20
70
1
1
1
1
1
1
1
0
0
0
0
0
1
```

1
1
100
50
80
1
5
1
3
7
30
25
25
30
30

El archivo solución de salida **Menu.sps** es:
X[1]=1.0000
X[2]=5.0000
X[3]=1.0000
X[4]=3.0000
X[5]=7.0000

Evaluación de función objetivo=114.00

NOTA: compárelo con los datos de la columna Porción en la tabla anterior.

El archivo desempeño de salida **Menu.spm** es:

El problema original tiene óptimo:114.00

Tipo Sol.: SOLUCION BASICA FACTIBLE OPTIMA FINITA

Iteraciones de fase 1: 1
Iteraciones de fase 2: 0
Iteraciones de fase 3: 1
Evaluaciones de función en fase 1: 0
Evaluaciones de función en fase 2: 0
Evaluaciones de función en fase 3: 1
Tiempo de Respuesta: 0.0000000 segundos

9.6 Resumen de Instrucciones

En la tabla que se presenta a continuación, las opciones del menú se dan en el orden en que deben elegirse desde el menú principal del sistema.

La notación utilizada es la siguiente:

|| para indicar que puede elegir una u otra opción.

& para indicar que debe elegirse una opción después de otra.

Operación	Opción (es) del Menú	Teclas
Introducir una nueva dieta alimenticia	Nuevo	N
Abrir un archivo de dieta alimenticia existente	Abrir	A
Calcular las cantidades óptimas del menú	Abrir o Nuevo, Solución	A || N & S
Modificar la dieta alimenticia	Abrir o Nuevo, Modificar	A || N & M
Ver las cantidades óptimas de alimentos	Abrir o Nuevo, Solución, Cantidades	A || N & S & C
Ver el costo total de la dieta óptima	Abrir o Nuevo, Solución, Costo	A || N & S & T
Cambiar el costo de los alimentos	Abrir o Nuevo, Modificar, Función	A || N & M & F
Cambiar la cantidad de nutrientes de la dieta	Abrir o Nuevo, Modificar, Coef.	A || N & M & C
Cambiar las cantidades mínimas de nutrientes	Abrir o Nuevo, Modificar, CotaIR	A || N & M & I
Cambiar las cantidades máximas de nutrientes	Abrir o Nuevo, Modificar, CotaSR	A || N & M & S
Cambiar las cantidades mínimas de alimentos en la dieta alimenticia	Abrir o Nuevo, Modificar, CotaIV	A || N & M & V
Cambiar las cantidades máximas de alimentos en la dieta alimenticia	Abrir o Nuevo, Modificar, CotaSV	A || N & M & A
Terminar la ejecución de DIETEX	Salir	S

Bibliografía

[AA93] Arbel, Ami.- *"Exploring Interior - Point Linear Programming, Algoritms and Software"*.- The MIT Press.- 1993.

[SVS91] SVS C^3.- *"Systems Manual"*.- Silicon Valley Software.- San Mateo California.- 1991.

[BC91] Borland International, Inc..- *"Turbo C++ for WINDOWS, ver. 3.0"*.- Borland International.- USA.- 1991.

[FO91] Fuente O'connor, Jose Luis de la.- *"Solución de sistemas de ecuaciones, programación lineal y entera"*.- Escuela Técnica Superior de Ingenieros Industriales.- España.- 1991.- pp. 301-380.

[BM89] Bazaraa, Mokhtar S..- *"Programación Lineal y Flujo en Redes"*.- LIMUSA.- México.- 1989.- pp. 13-34, 89-211.

[TH93] Taha, Hamdy A..- Software del libro: *"Investigación de Operaciones"*.- Ed. Alfaomega.- 1993.

[TS87] Torregrosa Sanchez, Juan.- *"Algebra Lineal y sus Aplicaciones: Teoría y Problemas"*.- McGraw-Hill.- España.- 1987.- pp. 351-363.

[RI93] Rios Insua, Sixto.- *"Investigación Operativa: Optimización"*.- Editorial Centro de Estudios Ramón Areces.- España.- 1993.- pp. 13-97.

[BA88] Barsov, A. S..- *"Que es la Programación Lineal"*.- MIR.- URSS.- 1988.

[HIST/1] The Diet Problem.- *"http://www.mcs.anl.gov/home/otc/Guide/CaseStudies/diet/history.htm"*