UNIVERSIDAD CATÓLICA DE SANTA MARÍA
ESCUELA DE POSGRADO
MAESTRÍA EN MATEMÁTICA

Estudio Comparativo del Método Simplex y el Método de Puntos Interiores Primal-Dual para Programación Lineal

Arequipa 2007 - 2008

Tesis presentada por:
MANUEL ARENAS CÉSPEDES
para optar el grado de Magíster en Matemática.

Arequipa, Perú
2010

Estudio Comparativo del Método Simplex y el Método de Puntos Interiores Primal-Dual para Programación Lineal

2007 - 2008

Arequipa, Perú
2010

Resumen

En este trabajo hemos resaltado algunas de las desventajas computacionales más importantes del Método Simplex, las cuales justamente motivaron la creación de los métodos de puntos interiores para programación lineal. Analizamos la construcción del Método de Puntos Interiores Primal-Dual, donde presentamos los principales fundamentos matemáticos que soportan el método. También hemos incluido los detalles que hacen posible llevar este método al computador, los programas que ejecutan el algoritmo respectivo fueron realizados usando el MatLab, debido a que MatLab es adecuado para trabajar con programas relacionados al álgebra lineal y métodos numéricos en general. El método fue comparado con el clásico Método Simplex, donde se analizó su desempeño computacional tanto en teoría como en la práctica. Además, revisamos algunos aspectos relacionados al análisis de sensibilidad y su abordaje mediante estos métodos. Finalmente, los resultados computacionales fueron incluidos en este trabajo.

Abstract

In this work we have highlighted some of the most important computational drawbacks of the Simplex Method, which just led to the creation of interior point methods for linear programming. We analyze the construction of the Interior Point Primal-Dual Method, where we present the main mathematical foundations that support the method. We have also included the details that make it possible to bring this method to the computer, programs that are running the corresponding algorithm were performed using MatLab, because MatLab suitable for work with programs related to linear algebra and numerical methods in general. The method was compared with the classical simplex method, where computational performance is analyzed in theory and in practice. In addition, we review some aspects related to sensitivity analysis and its management by these methods. Finally, computational results were included in this work.

Índice general

Introducción

La Programación Lineal es una parte importante dentro de la Investigación de Operaciones. El Problema de Programación Lineal es uno de los modelos matemáticos de optimización que tiene las más favorables propiedades para su resolución, debido a que es un Problema de Programación Convexa, es decir, tanto la función objetivo como la región de factibilidad son convexas. Esta propiedad permite la construcción de métodos, ya sean exactos o iterativos, los cuales aseguran teóricamente su convergencia a una solución óptima.

Otro resultado importante dentro de la teoría de la Programación Lineal es el siguiente: Si existe solución óptima finita, entonces existe una solución óptima que es justamente un punto extremo. El Algoritmo Simplex se vale de esta propiedad y se desplaza a través de los extremos, en forma de soluciones básicas, en busca de la solución óptima, sin necesariamente analizar todas ellas. Esto lo torna un método inteligente, pero existe un inconveniente, una región de factibilidad en programación lineal puede tener una cantidad, aunque finita, inmensamente grande de extremos. La existencia de problemas particulares (V. Klee y G. Minty) donde el Simplex analiza todos los extremos, lo torna ineficiente desde el punto de vista teórico.

Por otro lado, los métodos modernos basados en puntos interiores abordan el problema de programación lineal desde otro punto de vista: se desplazan a través del interior de la región de factibilidad en busca de la solución óptima. Si bien el número de iteraciones puede ser aún mayor que las realizadas por el Simplex cuando el problema tiene pocas variables, el desempeño de estos métodos modernos se nota cuando el número de variables del problema se

incrementa. El soporte matemático para los métodos de puntos interiores lo conforman el clásico Método de Newton, Técnicas Lagrangianas y Estrategias de Barrera Logaritmo.

Para analizar la eficiencia del Algoritmo Simplex y los Algoritmos de Puntos Interiores, desde un punto de vista teórico y, por lo tanto, formal, es necesario introducir formalismos basados en términos del número de iteraciones requeridas para resolver diferentes ejemplares del problema de programación lineal. Por la literatura [1], se sabe que el Simplex puede realizar, en el peor de los casos, un número exponencial de iteraciones, aproximadamente $2^n \cdot L$, donde n es el número de variables y L una constante. Mientras que un método de puntos interiores moderno realiza aproximadamente $n \cdot L$ iteraciones [1][2]. Este constraste entre ambos algoritmos torna interesante investigar las virtudes y las desventajas de los métodos de puntos interiores cuando son comparados con el clásico Método Simplex.

Iniciaremos este trabajo haciendo una revisión de los principales conceptos relacionados con el Problema de Programación Lineal. Seguidamente, veremos los fundamentos teóricos que permitieron la construcción del Método de Puntos Interiores Primal-Dual. Finalmente, analizaremos las ventajas y desventajas que cada uno de estos métodos presentan.

El interés por este último tema radica en los siguiente: a pesar que el método Simplex es considerado ineficiente en teoría, pues el número de iteraciones es del orden exponencial, en la práctica suele tener un buen desempeño, más aún, tiene inmersa en su construcción toda una teoría de dualidad, la cual permite realizar el análisis de sensibilidad de un modo práctico.

El método de puntos interiore Primal-Dual, a pesar de ser considerado hoy en día el mejor algoritmo para la resolución del problema de programación lineal, pues el número de iteraciones requeridas es del orden polinomial, a primera vista parece no poseer buenas propiedades que favorezcan realizar análisis de sensibilidad, prefiriéndose muchas veces resolver nuevamente el problema cuando algunos datos fueron modificados. Resulta entonces interesante compararlos desde este punto de vista. Algo que sin duda resulta aún

más interesante, es la posibilidad de descubrir mecanismos para realizar este análisis de sensibilidad en el Método de Puntos Interiores Primal-Dual.

Capítulo 1

El Problema de Programación Lineal

El modelo denominado Problema de Programación Lineal, corresponde a la siguiente formulación matemática:

$$\begin{aligned} &\text{Minimizar } c^t x \\ &\text{Sujeto a:} \\ &\quad Ax = b \\ &\quad x \geq 0 \end{aligned} \tag{1.1}$$

donde A es una matriz real de m filas y n columnas, b es un vector columna de m componentes, c es un vector columna de n componentes y x representa el vector cuyas componentes son las incógnitas o variables de decisión.

Si un modelo está en la forma (1.1), se dice que el problema está en la forma estándar. Cualquier modelo puede ser convertido en esta forma de modo equivalente. En este proyecto trabajaremos específicamente con la forma estándar, esto se justifica, porque la mayoría de métodos operan sobre la forma (1.1).

Otro aspecto que debe ser aclarado, es que sólo trabajaremos con la forma de minimización, la razón se debe a que maximizar la función $c^t x$ es equivalente a minimizar la función $-c^t x$.

La expresión $c^t x$ se denomina Función Objetivo. Las expresiones $Ax = b$ y $x \geq 0$ se denominan Las Restricciones y Las Condiciones de no Negatividad, respectivamente. La Región de Factibilidad es denotada por el conjunto $\Omega = \{x \in \mathbb{R}^n : Ax = b,\ x \geq 0\}$. Si $x \in \Omega$, entonces x se llama Solución Factible. Observe que Ω puede ser Acotado, No Acotado o Vacío, en este último caso, se dice que el problema (1.1) es Infactible. Resolver el Problema de Programación Lineal consiste en encontrar $\bar{x} \in \Omega$ tal que $c^t \bar{x} \leq c^t x$, para todo $x \in \Omega$. En este caso, el vector $\bar{x}$ se llama Solución Óptima.

Seguidamente expondremos algunos conceptos ampliamente utilizados en programación lineal.

Conjunto Convexo.

Un conjunto C es convexo, si la combinación convexa de dos elementos cualesquiera del conjunto está en C. Es decir, si $x, y \in C$, para todo $\lambda \in [0, 1]$, se verifica que

$$\lambda x + (1 - \lambda) y \in C$$

En programación lineal, la región de factibilidad, Ω, es un conjunto convexo, más aún, es un poliedro convexo.

Función Convexa.

Una función $f : C \subset \mathbb{R}^n \to \mathbb{R}$, donde C es un conjunto convexo, se dice convexa, si para cualquier par de puntos $x, y \in C$, se cumple:

$$f\left(\lambda x + (1 - \lambda)\, y\right) \leq \lambda f\left(x\right) + (1 - \lambda)\, f\left(y\right), \qquad \lambda \in [0, 1]$$

Hiperplano.

Un hiperplano en $\mathbb{R}^n$ es un conjunto de la forma:

$$H = \left\{x \in \mathbb{R}^n :\ d^t x = k\right\}$$

donde $d \in \mathbb{R}^n,\ d \neq 0$, es un n-vector columna y k es un escalar. Un hiperplano es la extensión de una recta en $\mathbb{R}^2$ o un plano en $\mathbb{R}^3$. Debemos notar que el vector d es normal al hiperplano H.

Semiespacio.

Un hiperplano divide el espacio $\mathbb{R}^n$ en dos subregiones, éstas se denominan semiespacios. De este modo, un semiespacio es el conjunto

$$\left\{x \in \mathbb{R}^n : \ d^t x \geq k\right\}$$

donde d es un vector columna diferente de cero y k es un escalar. El otro semiespacio será el conjunto de puntos

$$\left\{x \in \mathbb{R}^n : \ d^t x \leq k\right\}$$

Observe que $\{x \in \mathbb{R}^n : \ d^t x \geq k\} \bigcup \{x \in \mathbb{R}^n : \ d^t x \leq k\} = \mathbb{R}^n$.

Restricción Activa.

Dada la desigualdad $d^t x \geq k$, donde $d \in \mathbb{R}^n$ y k es un escalar. Si $\bar{x} \in \mathbb{R}^n$ satisface $d^t \bar{x} = k$, entonces se dice que $\bar{x}$ hace *activa* la desigualdad $d^t x \geq k$. En programación lineal, estas desigualdades constituyen las restricciones. Para el caso de una restricción de igualdad $d^t x = k$, es claro que cualquier $\bar{x}$ que satisfaga dicha restricción la hará activa.

Punto Extremo.

Consideremos una región factible en programación lineal. Un punto extremo $\bar{x} \in \mathbb{R}^n$ es la intersección de n hiperplanos linealmente independientes. Es decir, $\bar{x}$ es punto extremo si hace activas n restricciones linealmente independientes. Si $\bar{x}$ fuera también factible, entonces se dice que es un punto extremo factible.

Punto Extremo No Degenerado.

Un punto extremo $\bar{x}$ se dice no degenerado si hace exactamente n restricciones activas. Cuando más de n restricciones son activas en $\bar{x}$, entonces el punto se denomina extremo degenerado.

Solución Básica.

Consideremos el problema de programación lineal en la forma estándar, con $m < n$ y rango de A completo (todas las filas de A linealmente indepen-

dientes):

$$\text{Minimizar } c^t x \tag{1.2}$$

$$Ax = b \tag{1.3}$$

$$x \geq 0 \tag{1.4}$$

Una solución básica se obtiene eligiendo desde A una submatriz B, cuadrada e inversible, de rango m, a esta submatriz se le llama matriz básica. Notemos que a cada componente x_j del vector x le corresponde una columna a_j de la matriz A. Reordenando las columnas de A, si fuera necesario, y también las componentes de x, particionamos la matriz A y el vector x de modo que se mantenga la compatibilidad:

$$A = \begin{bmatrix} B & N \end{bmatrix} \tag{1.5}$$

$$x = \begin{bmatrix} x_B \\ x_N \end{bmatrix} \tag{1.6}$$

Al vector x_N se le denomina vector no básico y a sus $n-m$ componentes se les llama variables no básicas. Por otro lado, al vector x_B se le llama vector básico y a sus m componentes se les denomina variables básicas. Reemplazando (1.5) y (1.6) en (1.3) y despejando x_B en función de x_N, tenemos:

$$\begin{bmatrix} B & N \end{bmatrix} \begin{bmatrix} x_B \\ x_N \end{bmatrix} = b$$

después de multiplicar en bloques, tenemos:

$$\begin{aligned} Bx_B + Nx_N &= b \\ B^{-1}Bx_B + B^{-1}Nx_N &= B^{-1}b \\ x_B + B^{-1}Nx_N &= B^{-1}b \\ x_B &= B^{-1}b - B^{-1}Nx_N \end{aligned} \tag{1.7}$$

Obsérvese que si damos valores arbitrarios a las variables del vector x_N, y

evaluamos x_B según (1.7), el vector $x = \begin{bmatrix} x_B \\ x_N \end{bmatrix}$ satisface automáticamente (1.3), pero aún no satisface (1.4). Si hacemos $x_N = 0$ y calculamos nuevamente x_B según (1.7), el vector

$$x = \begin{bmatrix} x_B \\ x_N \end{bmatrix} = \begin{bmatrix} x_B \\ 0 \end{bmatrix} = \begin{bmatrix} B^{-1}b \\ 0 \end{bmatrix}$$

además será un punto extremo, pues n restricciones linealmente independientes son activas en x. Esto es, $n - m$ restricciones activas a causa de $x_N = 0$ y m restricciones activas a causa de $Ax = b$. Este último vector x, calculado de ese modo, se denomina solución básica.

Por lo tanto, en programación lineal, una solución básica es un punto extremo. El teorema 1.1 muestra la validez de esta última afirmación.

Solución Básica Factible.

Haciendo $x_N = 0$ y calculando x_B según (1.7), si $x_B = B^{-1}b \geq 0$, entonces la solución básica

$$x = \begin{bmatrix} x_B \\ x_N \end{bmatrix} = \begin{bmatrix} B^{-1}b \\ 0 \end{bmatrix}$$

será además factible, pues satisface (1.3) pero ahora también (1.4). En este caso, la solución básica se denomina solución básica factible. En programación lineal, una solución básica factible es un extremo factible.

Solución Básica Factible No degenerada.

Es una solución básica factible donde $x_B > 0$. Observe que una solución básica factible no degenerada hace exactamente n restricciones activas, por lo que es un punto extremo no degenerado. Cuando una o más variables básicas toman el valor de cero, la solución básica factible se denomina degenerada. Note que una solución básica factible degenerada hace más de n restricciones activas, lo que significa que es un punto extremo degenerado.

Dirección de Decrecimiento.

Dado x factible y $d \in \mathbb{R}^n$, $d \neq 0$, d es una dirección de decrecimiento partiendo de x, si existe $\varepsilon > 0$ tal que, para todo $t \in \langle 0, \varepsilon]$

$$c^t(x + td) < c^t x$$

Para el caso de minimización, una dirección es de decrecimiento si, y sólo si, $c^t d < 0$.

1.1. Propiedades

En la sección anterior, cuando definimos una solución básica, vimos que al detectar una submatriz B cuadrada e inversible en la matriz A, automáticamente nos permitía calcular un punto extremo.

Consideremos el problema de programación lineal en la forma estándar, denotado ahora por

$$\begin{aligned} &\text{Minimizar } c^t x \\ &x \in X \end{aligned} \tag{1.8}$$

donde $X = \{x \in \mathbb{R}^n : \ Ax = b, \ x \geq 0\}$, $b \in \mathbb{R}^m$, $A \in \mathbb{R}^{m \times n}$ es de rango m, con $m < n$. A continuación, daremos algunos teoremas importantes en programación lineal.

El siguiente teorema relaciona las soluciones básicas con los puntos extremos.

Teorema 1.1 *Un punto extremo factible x de X es una solución básica factible para (1.8). Inversamente, si x es una solución básica factible de (1.8), entonces x es un punto extremo factible de X.*

Prueba. Vea la prueba en [1]. ■

El Algoritmo Simplex se fundamenta en el siguiente teorema.

Teorema 1.2 *Sea X la región de factibilidad del problema (1.8). Si existe una solución óptima en X, entonces existe un punto extremo óptimo en X.*

Prueba. Vea la prueba en [1]. ■

En base a los anteriores teoremas ahora podemos formular el siguiente:

Corolario 1.1 *Si existe una solución óptima para el problema de programación lineal en la forma estándar, entonces existe una solución básica factible óptima.*

Prueba. Consecuencia directa de los teoremas 1.1 y 1.2. ■

Por lo tanto, la solución óptima del Problema de Programación Lineal está justamente en un punto extremo factible, es decir, una Solución Básica Factible (un vértice del poliedro asociado a la región de factibilidad).

Debemos aclarar lo siguiente, los teoremas sólo dicen que si existe una solución óptima, existirá también una solución básica factible óptima. Pero no afirman que una solución óptima es una solución básica factible.

Consideremos el siguiente problema general de optimización

$$\begin{array}{c} \text{Minimizar } f(x) \\ x \in X \end{array} \tag{1.9}$$

donde $f : \mathbb{R}^n \mapsto \mathbb{R}$, $X \subset \mathbb{R}^n$, $X \neq \emptyset$. Un *minimizador local* x^* es una solución óptima con respecto al conjunto $V(x^*, \varepsilon) \bigcap X$, donde $V(x^*, \varepsilon)$ es una vecindad en torno de x^* y radio $\varepsilon > 0$. Por otro lado, un *minimizador global* será una solución óptima con respecto a todo X.

Cuando resolvemos un problema de optimización, casi siempre estamos interesados en obtener un minimizador global. Frecuentemente esta tarea es sumamente difícil y a veces imposible, y apenas es permitido obtener minimizadores locales. Afortunadamente, en programación lineal, tal situación es diferente, nosotros no tenemos que preocuparnos si la solución que obtenemos es local o global, pues gracias a la estructura convexa de este problema,

toda solución local es global. Esta última afirmación la formalizaremos en el teorema 1.3.

Un Problema de Programación Convexa es una formulación matemática del siguiente tipo:

$$\begin{aligned} &\text{Minimizar } f(x) \\ &x \in X \end{aligned} \tag{1.10}$$

donde $f : \mathbb{R}^n \mapsto \mathbb{R}$ es una función convexa, y $X \subset \mathbb{R}^n$ es un conjunto convexo. Claramente, el problema de programación lineal es un problema de programación convexa.

Teorema 1.3 *En un problema de programación convexa, todo minimizador local es minimizador global.*

La prueba de este teorema puede verse en algunos libros de optimización no lineal, por ejemplo [4].

Un aspecto importante que vale la pena resaltar es lo siguiente, un problema de programación lineal en el formato estándar puede tener un número extremadamente grande de soluciones básicas. Por ejemplo, si $A \in \mathbb{R}^{m \times n}$, entonces una cota superior para el número de soluciones básicas (extremos) es justamente

$$\binom{n}{m} = \frac{n!}{m!\,(n-m)!}$$

Así, cualquier algoritmo que utilice soluciones básicas para resolver el problema de programación lineal es un candidato a convertirse en ineficiente.

1.2. Dualidad y Condiciones de Optimalidad

Sea el problema de programación lineal en su forma estándar, al cual lo denominaremos el *primal*

$$\begin{aligned} &\text{Minimizar } c^t x \\ &\text{Sujeto a:} \\ &\quad Ax = b \\ &\quad x \geq 0 \end{aligned} \tag{1.11}$$

donde $A \in \mathbb{R}^{m\times n}$, $b \in \mathbb{R}^m$, $c \in \mathbb{R}^n$.

Definición 1.1 *El problema de programación lineal*

$$\begin{aligned} &\textit{Maximizar } b^t y \\ &A^t y + z = c \\ &z \geq 0 \end{aligned} \tag{1.12}$$

donde $y \in \mathbb{R}^m$ y $z \in \mathbb{R}^n$, *se denomina* dual *del problema (1.11).*

Observación 1.1 *El número de variables duales es igual al número de restricciones primales. Podemos decir entonces que cada variable dual está asociada a una restricción del primal.*

Observación 1.2 *El número de restricciones duales es igual al número de variables primales. En este caso, podemos decir que cada variable del primal está asociada a una restricción dual.*

Muchos teoremas relacionan estos dos problemas, de ahí su importancia (véase por ejemplo [1]). Vamos a enumerar las propiedades más importantes debido a que serán de gran utilidad en la construcción del método de puntos interiores primal dual.

1. El dual del dual es el primal.

2. Sea x es una solución factible para el problema primal e y una solución factible para su correspondiente dual. Si $c^t x = b^t y$, entonces x e y son soluciones óptimas para sus problemas respectivos.

3. Si existe una solución óptima (finita) para uno de los problemas, entonces existe solución óptima (finita) para su correspondiente dual.

4. Si uno de los problemas es ilimitado, entonces su respectivo dual es infactible.

5. Finalmente, quizá uno de los principales resultados. Si uno de los problemas, digamos sin pérdida de generalidad, el primal, tiene solución óptima (finita) x^*, entonces existen vectores y^* y z^* tales que:

$a)$ $Ax^* = b$, $x^* \geq 0$

$b)$ $A^t y^* + z^* = c$, $z^* \geq 0$

$c)$ $x_i^* z_i^* = 0$, para $i = 1, 2, ..., n$

Esta última propiedad se conoce como el Teorema de Karush Kuhn Tucker y en realidad son condiciones necesarias y suficientes que caracterizan una solución óptima del problema de programación lineal en la forma estándar.

La anterior propiedad puede ser vista de un modo más práctico en forma de un sistema de ecuaciones e inecuaciones. Este sistema es conocido como las Condiciones de Optimalidad de Karush-Kuhn-Tucker (KKT) para el Problema de Programación Lineal en la Forma Estándar:

$$Ax = b, \quad x \geq 0 \tag{1.13}$$

$$A^t y + z = c, \quad z \geq 0 \tag{1.14}$$

$$z^t x = 0 \tag{1.15}$$

Lo que dicen estas condiciones es que, si $\bar{x}$ es una solución óptima para el problema primal, entonces existen vectores $\bar{y}$ y $\bar{z}$, tal que $(\bar{x}, \bar{y}, \bar{z})$ satisface (1.13), (1.14) y (1.15). Inversamente, si existe $(\hat{x}, \hat{y}, \hat{z})$ satisfaciendo (1.13), (1.14) y (1.15), entonces $\hat{x}$ es solución óptima de 1.11.

1.3. El Algoritmo Simplex y la Estrategia M-Grande

El clásico Algoritmo Simplex trabaja sobre un problema de programación lineal en la forma estándar, en el cual se conoce una base factible inicial B:

$$\text{Minimizar } c^t x$$

$$Ax = b \tag{1.16}$$

$$x \geq 0$$

donde $b \in \mathbb{R}^m$, $c \in \mathbb{R}^n$, $A \in \mathbb{R}^{m\times n}$ es de rango m, $m < n$. Denotemos por R al conjunto de índices no básicos y $\bar{b} = B^{-1}b$.

Algoritmo 1.1 (Simplex) *Elegir una matriz básica B tal que $x = \begin{bmatrix} B^{-1}b \\ 0 \end{bmatrix}$ sea una solución básica factible.*

Paso 1: *Hallar $x_B = B^{-1}b = \bar{b}$, hacer $x_N = 0$ y calcular $z = c_B^t x_B$.*

Paso 2: *Hallar $w = c_B^t B^{-1}$ y hallar $z_k - c_k = \max\limits_{j\in R}\{z_j - c_j\}$, donde*

$$z_j - c_j = wa_j - c_j$$

1. *Si $z_k - c_k \leq 0$, detenerse. La solución básica óptima es $x = \begin{bmatrix} B^{-1}b \\ 0 \end{bmatrix}$.*
2. *Caso contrario, ir al paso 3.*

Paso 3: *Hallar $y_k = B^{-1}a_k$*

1. *Si $y_k \leq 0$, detenerse. Existe solución óptima ilimitada.*
2. *Caso contrario, ir al paso 4.*

Paso 4: *Calcular*

$$x_k = \frac{\bar{b}_r}{y_{r,k}} = \min_{1\leq i\leq m}\left\{\frac{\bar{b}_i}{y_{i,k}} : \ y_{i,k} > 0\right\}$$

1. *Actualizar la base B, cambiando a_{B_r} por a_k.*
2. *Actualizar R, cambiando k por el índice B_r.*
3. *Volver al paso 1.*

En contraste, la estrategia M-Grande no es un algoritmo, sino que es un recurso del cual nos valemos para reparar una de las más grandes desventajas del algoritmo simplex, la base factible inicial B.

Consiste en resolver el siguiente problema:

$$\begin{aligned} &\text{Minimizar } c^t x + Mx_{a_1} + \cdots Mx_{a_m} \\ &Ax + Ix_a = b \\ &x, x_a \geq 0 \end{aligned} \tag{1.17}$$

donde la nueva matriz de coeficientes tecnológicos es $[\ A \quad I\ ]$, la cual es siempre de rango completo. Más aún, la matriz identidad $B = I$ representa una base inicial factible (si $b \geq 0$) y $M > 0$ es un número real tal que $M \to +\infty$ (en la práctica basta tomar M grande). Las nuevas variables $x_{a_1}, ..., x_{a_m}$ se denominan *variables artificiales*.

Una vez resuelto el problema (1.17) usándose el Algoritmo Simplex, resta apenas interpretar las soluciones óptimas del problema (1.16) a partir del las soluciones óptimas del problema (1.17). Así,

1. Si alguna variable óptima artificial es mayor que cero, cuando el Simplex se detuvo encontrando solución óptima finita o ilimitada sobre (1.17), entonces el problema (1.16) es infactible.

2. Si el Simplex se detuvo detectando valor objetivo ilimitado en (1.17) y las variables artificiales en ese momento valen cero, entonces el problema (1.16) debería tener el mismo veredicto.

3. Finalmente, si la solución óptima de (1.17) es finita y las variables artificiales en ese momento valen cero, entonces las variables originales deberían conformar una solución óptima finita del problema (1.16).

Capítulo 2

Eficiencia de Algoritmos

Un *algoritmo* es un conjunto de instrucciones para resolver un determinado problema, por lo general, los algoritmos resumen métodos. Un *método* es un procedimiento, con las justificativas matemáticas necesarias, por el cual se resuelve un determinado problema.

Definición 2.1 *Una* instancia *de un problema es un miembro particular de éste, donde se conocen los datos numéricos de modo explícito.*

Ejemplo 2.1 *Por ejemplo, considere el problema que consiste en sumar dos números enteros a y b, entonces una* instancia *asociada a este problema consiste de los datos numéricos (ingreso), digamos* $a = 3$ *y* $b = 9$.

Ejemplo 2.2 *El Problema de Programación Lineal en la forma estándar está dado por el siguiente modelo matemático:*

$$\begin{gathered} \textit{Minimizar } c^t x \\ \textit{Sujeto a:} \\ Ax = b \\ x \geq 0 \end{gathered}$$

Una instancia *para este porblema constituye por ejemplo el ingreso explícito de*

$$c = \begin{bmatrix} 5 \\ 7 \\ 9 \end{bmatrix} \qquad b = \begin{bmatrix} 10 \\ 60 \end{bmatrix} \qquad A = \begin{bmatrix} -1 & 2 & 3 \\ -3 & 5 & 4 \end{bmatrix}$$

Definición 2.2 *El* tamaño *o* medida *de una instancia ξ asociada a un problema, es el número de bits binarios necesarios para representar todos los datos numéricos de la instancia en la memoria de un computador, el tamaño de una instancia será representada por $|\xi|$.*

Sabemos que las computadoras trabajan usando el sistema binario para representar los datos, entonces, para almacenar un número entero Δ debe utilizar $\lceil 1 + \log_2 \Delta \rceil$ dígitos binarios. Si necesitamos especificar un signo a ese número, necesitamos de un bit adicional, es decir $1 + \lceil 1 + \log_2 \Delta \rceil$.

Por ejemplo, para representar cualquier entero positivo $\Delta \in [2^r, 2^{r+1}\rangle$, donde $r \geq 1$ es natural, requerimos $r + 1$ bits binarios.

Ejemplo 2.3 *Basados en [1]. Considere el Problema de Programación Lineal*

$$\begin{aligned} &\textit{Minimizar } c^t x \\ &\textit{Sujeto a:} \\ &Ax = b \\ &x \geq 0 \end{aligned}$$

donde $A \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}^m$ y $c \in \mathbb{R}^n$. Cuando todos los datos de una instancia de este problema son enteros, lo cual siempre podemos asumir, la medida de la instancia (longitud del ingreso) se puede expresar como un número L, donde:

$$\begin{aligned} L \;=\; & \lceil 1 + \log_2 m \rceil + \lceil 1 + \log_2 n \rceil + \sum_{j=1}^{n} (1 + \lceil \log_2(1 + |c_j|) \rceil) + \\ & + \sum_{i=1}^{m} \sum_{j=1}^{n} (1 + \lceil \log_2(1 + |a_{i,j}|) \rceil) + \sum_{i=1}^{m} (1 + \lceil \log_2(1 + |b_i|) \rceil) \end{aligned}$$

Este número aparece cuando se analiza, desde un punto de vista teórico, el desempeño que un algoritmo presenta cuando es sometido a una instancia de programación lineal de tamaño L.

En esta sección vamos a clasificar los algoritmos de acuerdo a su eficiencia, esto se conseguirá mediante la introducción algunos formalismos. Existen dos

criterios para medir la eficiencia de un algoritmo respecto a un determinado problema:

1. *El criterio empírico:* consiste en implementar el algoritmo en algún lenguaje de programación y con la ayuda de un computador, someterlo a diversos casos del problema para estimar su desempeño. Es de suponer, que con este criterio no tenemos garantía del comportamiento del algoritmo para cualquier caso del problema, puesto que el número de casos experimentados es finito.

2. *El criterio teórico:* consiste en determinar matemáticamente la cantidad de recursos necesarios, el tiempo de ejecución y el espacio necesario de memoria de computador para guardar los datos. Estos recursos deben ser expresados en función del tamaño del problema (intuitivamente, en el número de restricciones y variables). Una de las ventajas en este criterio es que no depende del computador que se esté utilizando, ni del lenguaje de programación, ni del estilo particular del programador, etc. Otra ventaja es que nos permite estudiar la eficiencia de un algoritmo con relación a problemas de cualquier tamaño.

Cuando se analiza formalmente el desempeño computacional de un algoritmo que resuelve un determinado problema, se toma en cuenta el *criterio teórico*. En este análisis, se busca, por lo general, determinar el número de iteraciones que realiza el algoritmo para resolver la peor instancia del problema (el peor caso para el algoritmo). Claramente, si conocemos el número de iteraciones que realiza el algoritmo para su peor caso de tamaño n, conseguiremos exponer una cota superior para el número de iteraciones que éste tomará para resolver cualquier instancia de tamaño n.

Al analizar la eficiencia de un algoritmo con el criterio teórico, el tiempo puede estar expresado en unidades de tiempo, pero puede estar expresado también en términos del número de operaciones elementales (asignaciones, comparaciones, sumas y multiplicaciones) y, algunas veces, en términos del número de iteraciones que realiza el algoritmo.

En este trabajo, se asumirá que una operación elemental demanda una unidad de tiempo para ser ejecutado.

2.1. El Principio de Invariabilidad

Un algoritmo está correctamente definido si éste resuelve el problema para todas sus *instancias* asociados. Para demostrar que un algoritmo es incorrecto, lo que tenemos que hacer es encontrar una *instancia* para la cual no es posible encontrar una respuesta correcta. Por otro lado, mostrar que un algoritmo es correcto es una tarea que por lo general es dificultosa.

Supongamos que tenemos tres computadoras C_1, C_2 y C_3, en las cuales ejecutamos una implementación del mismo algoritmo $\mathcal{A}$. Si el sistema de cómputo C_2 es dos veces más veloz que C_1, entonces probablemente C_2 ejecute el algoritmo en la mitad del tiempo que C_1. Lo mismo pasaría si usamos la computadora C_3 que es tres veces más rápida que C_1, ejecutaría el algoritmo en una tercera parte del tiempo usado por C_1 y dos terceras partes del tiempo usado por C_2. Como se puede observar, el tiempo de ejecución de estas tres implementaciones difiere sólo por una constante multiplicativa.

Algo similar sucedería si utilizáramos dos lenguajes de programación diferentes. Si por un lado, implementamos nuestro algoritmo con el lenguaje C++, y por el otro, implementamos en MatLab. Claramente el tiempo de ejecución de la implementación hecha en C++ será mucho menor que el tiempo usado al ejecutar nuestro algoritmo usando MatLab. Lo mismo ocurre si la implementación depende de la habilidad y técnica del programador.

Debido a que no existe un computador estándar dónde comparar los tiempos de ejecución, ni formas únicas de programar, es que se adoptó el principio de invariabilidad. *Mediante el principio de invariabilidad, dos implementaciones del mismo algoritmo no se diferencian en eficiencia por más de constante multiplicativa.* Es decir, si dos implementaciones del mismo algoritmo toman $t_1(n)$ y $t_2(n)$ segundos, respectivamente, para resolver una instancia

de un determinado problema de tamaño n, entonces debe existir una constante positiva c tal que $t_1(n) \leq c \cdot t_2(n)$ para $n \geq n_0$ suficientemente grande. Con esto, podemos medir la eficiencia del algoritmo usando $t_1(n)$ o $t_2(n)$, pues para nuestros fines teóricos, las dos implementaciones del algoritmo son prácticamente iguales, diferenciándose apenas por una constante multiplicativa.

En resumen, un cambio de máquina sólo nos permitirá resolver una instancia de un problema 10 ó 20 veces más rápido, pero sólo un cambio de algoritmo nos dará una mejora substancial a medida que el tamaño de la instancia aumente.

Decimos que un algoritmo toma un tiempo $O\left(t\left(n\right)\right)$, y leemos *orden de* $t\left(n\right)$, donde $t : \mathbb{N} \mapsto \mathbb{R}^+$ es una función y n representa la medida de la instancia, si existe una constante real $c > 0$ y una implementación del algoritmo capaz de resolver toda instancia de tamaño n en un tiempo limitado superiormente por $c \cdot t\left(n\right)$ segundos. Debemos notar que el uso segundos no es indispensable, pues pudimos haber usado horas como unidad de tiempo, bastando sólo cambiar la constante para limitar el tiempo por $c_1 \cdot t\left(n\right)$ horas, donde $c_1 = \frac{c}{3600}$. O pudimos usar minutos, variando el límite superior para $c_2 \cdot t\left(n\right)$ minutos.

Como vemos, el uso de segundos o cualquiera otra unidad de tiempo no hacen diferencia para nuestros fines teóricos, pues quedan absorbidos por el principio de invariabilidad.

En realidad, el uso de alguna unidad de tiempo no es indispensable para analizar formalmente el comportamiento de un algoritmo, pues basta conocer por ejemplo el número de operaciones elementales o el número de iteraciones. Digamos que $r\left(n\right)$ sea el número de operaciones elementales realizadas por el algoritmo, si conocemos la frecuencia con que opera un sistema de cómputo, digamos $c = 2 \times 10^9$ operaciones elementales por segundo, entonces es posible obtener el tiempo que tomará el algoritmo para resolver esta instancia: $\frac{r(n)}{c}$ segundos. Este hecho nos permite analizar algoritmos sin considerar la unidad de tiempo, lo cual torna el trabajo mucho más cómodo.

2.2. Notación Asintótica

Anteriormente mencionamos el *orden* de tiempo que un algoritmo toma para resolver una instancia de algún problema, ahora formalizaremos este concepto.

Definición 2.3 *Sea* $\mathbb{R}^+$ *el conjunto de números reales positivos, y* $f : \mathbb{N} \mapsto \mathbb{R}^+$ *una función arbitraria. Definimos el conjunto*

$$O\left(f\left(n\right)\right) = \left\{t : \mathbb{N} \mapsto \mathbb{R}^+ : \exists c \in \mathbb{R}^+, \exists n_0 \in \mathbb{N}, \forall n \geq n_0, t\left(n\right) \leq c \cdot f\left(n\right)\right\} \tag{2.1}$$

y lo llamamos "orden $f\left(n\right)$*". El conjunto* $O\left(f\left(n\right)\right)$ *representa todas las funciones reales* t *acotadas superiormente por un múltiplo real positivo de* f*, para suficientemente grande,* $n \geq n_0$*.*

Ejemplo 2.4 *Consideremos el problema de ordenamiento de un arreglo*

$$T\left[1 \quad \cdots \quad n\right]$$

en forma creciente. Existen varios algoritmos que resuelven este problema, analizaremos ellos con respecto al número de iteraciones realizadas:

1. *El algoritmo llamado* insertsort *requiere un número de iteraciones en el orden* n^2*, es decir, el número de iteraciones realizadas es* $t\left(n\right) \in O\left(n^2\right)$ *para resolver la peor instancia de este problema. Significa que requiere a lo más* $c \cdot n^2$ *iteraciones para resolver cualquier instancia del problema. En virtud de las observaciones últimas, por lo general decimos simplemente que "el algoritmo es* $O\left(n^2\right)$*".*

2. *Por otro lado, el algoritmo denominado* merguesort *requiere apenas un número de iteraciones en el orden* $n \cdot \log n$*, es decir, un número de iteraciones* $t\left(n\right) \in O\left(n \cdot \log n\right)$ *. Esto significa que a lo más requiere* $c \cdot n \cdot \log n$ *iteraciones para resolver la peor instancia del problema.*

En base a esto, podemos decir entonces que el algoritmo merguesort *es mejor que* insertsort*. La utilización del "orden de" nos permite clasificar los algoritmos de una manera formal y organizada.*

Ejemplo 2.5 *Si un algoritmo requiere un tiempo exacto de* $T(n)$ *segundos, donde* n *es el tamaño de la instancia asociado a un determinado problema, tal que* T *está definido por:*

$$T(n) = 2n^3 + 5n^2 - n + 2$$

Entonces, mediante la definición 2.3, el tiempo de ejecución de este algoritmo está en $O(n^3)$*. En este caso decimos que* $T(n)$ *es del orden* n^3*.*

Se dice que un algoritmo es *polinomial*, o para ser más precisos, el tiempo de ejecución tomado por una implementación computacional de este algoritmo está en el orden $p(n)$, si p es un polinomio en función del tamaño de la instancia n. Análogamente, un algoritmo se dice exponencial, si su implementación requiere un tiempo en $O(a^n)$ para resolver cualquier instancia de tamaño n, donde $a > 1$. Mediante esto, se puede clasificar a los algoritmos como eficientes o ineficientes. Claramente, un algoritmo es eficiente si este es polinomial. Por otro lado, un algoritmo será ineficiente si es exponencial.

2.3. Por qué Buscar Algoritmos Eficientes

Supongamos que tenemos una computadora A y una instancia de tamaño n asociada a un cierto problema. Si conocemos un algoritmo y sabemos que éste resuelve dicha instancia en un tiempo de $t(n) = 8^{-4} \times 2^n$ segundos. Para $n = 10$, el algoritmo dará una respuesta en 0,25 segundos, cuando el tamaño de la instancia aumenta para $n = 20$, la respuesta es dada en 4,26 minutos. Pero si el tamaño de la instancia es de $n = 30$, el algoritmo demora 3 días. Para $n = 40$ el algoritmo debería tardar un poco más de 8 años.

Supongamos que compramos otra computadora B, la cual es 64 veces más rápida que A. Entonces el algoritmo corriendo sobre esta nueva computadora debe resolver el problema en un tiempo $t(n) = 8^{-6} \times 2^n$. Así, vemos que para una instancia de tamaño $n = 40$, el algoritmo debe tardar ahora 48 días. Obviamente, es un gran logro, pero si el tamaño de la instancia aumenta

para $n = 46$, el algoritmo tardará nuevamente alrededor de 8 años para resolver el problema.

Supongamos ahora que conseguimos construir un nuevo algoritmo que resuelve el problema en $t(n) = 10^{-2} \times n^2$ segundos, para una instancia de tamaño n. Usando el mismo computador B, claramente este nuevo algoritmo demora 1 segundo para resolver una instancia de tamaño $n = 10$, demora 4 segundos para resolver una instancia de tamaño $n = 20$, demora 9 segundos para resolver una instancia de tamaño $n = 30$; y, para una instancia de tamaño 40, demorará tan sólo 16 segundos.

Por lo tanto, podemos concluir que: *una mejora en el tiempo de ejecución no se consigue sólo cambiando de computador, se tiene que cambiar de algoritmo.*

Observación 2.1 *El peor caso para el Algoritmo Simplex, lo constituye las instancia creadas por Victor Klee y George Minty [1]. En este caso, el algoritmo se desplaza por todos los vértices de la región factible. Para eso, ellos produjeron una instancia con $m = n$ restricciones de igualdad y $2n$ variables no negativas, para el cual el Algoritmo Simplex realiza $2^n - 1$ iteraciones para encontrar una solución óptima. Matemáticamente, el problema propuesto por Klee y Minty consiste en lo siguiente:*

$$\textit{Minimizar } x_n$$

$$0 \leq x_1 \leq 1$$

$$\varepsilon x_{j-1} \leq x_j \leq 1 - \varepsilon x_{j-1}, \quad j = 2, ..., n$$

$$x_j \geq 0, \quad j = 1, ..., n$$

donde ε es algún número racional en el intervalo $\langle 0, \frac{1}{2} \rangle$.

Por lo tanto, al requerir $t(n) = 2^n - 1$ iteraciones en lo que vendría a ser un peor caso, vea que $t(n) \in O(2^n)$, el Algoritmo Simplex es catalogado como un algoritmo de tiempo exponencial, desde el punto de vista teórico.

Esto a su vez, permite establecer formalmente que el Algoritmo Simplex es ineficiente.[1]

Observación 2.2 *Contrariamente a lo que pasa con el Algoritmo Simplex, los métodos de puntos interiores, motivo de esta tesis, son algoritmos de tiempo polinomial. Así, el algoritmo de Karmarkar requiere un tiempo de* $t(nL)$ *para resolver cualquier instancia de tamaño* L *y* n *variables, donde* $t(nL) \in O(n^{3,5}L)$*. Esto lo caracteriza como un algoritmo eficiente desde el punto de vista teórico [1].*

Observación 2.3 *Por otro lado, el Algoritmo de Puntos Interiores Primal-Dual, también es un algoritmo de tiempo polinomial. Este algoritmo requiere un tiempo* $T(nL)$ *para resolver cualquier instancia de tamaño* L *y* n *variables, donde* $T(nL) \in O(nL)$*. Esto lo torna, hoy en día, uno de los algoritmos más eficientes para la resolución del Problema de Programación Lineal [2][5].*

Queda pues plenamente justificado el estudio de los métodos de puntos interiores eficientes, sobre todo cuando se tratan con problemas de estructura complicada o de grandes dimensiones. No osbtante, en la práctica el Simplex aún continúa siendo útil para problemas de pequeña dimensión.

[1] Sin embargo, en la práctica, el Simplex tiene un desempeño razonable.

Capítulo 3

Método de Puntos Interiores Primal-Dual para Programación Lineal

La técnica para la construcción del método de puntos interiores primal dual consiste en convertir problemas de programación lineal en problemas sin restricciones (no necesariamente lineales), esto se consigue aplicando el método de Lagrange y el método de Fiacco & McCormick sobre las restricciones y las condiciones de no negatividad, respectivamente [5]. Las técnicas anteriores dan como resultado dos problemas de optimización sin restricciones los cuales serán abordados por el método de Newton.

3.1. El Método de Newton (1687)

Consideremos el sistema de ecuaciones no lineales dado por:

$$\begin{aligned} f_1(x) &= 0 \\ &\vdots \\ f_n(x) &= 0 \end{aligned} \tag{3.1}$$

donde $f_i : \mathbb{R}^n \longmapsto \mathbb{R}$ y $f_i \in C^1(\mathbb{R}^n)$, para $i = 1, ..., n$. Si definimos

$$F(x) = \begin{bmatrix} f_1(x) \\ \vdots \\ f_n(x) \end{bmatrix}$$

donde $F : \mathbb{R}^n \longmapsto \mathbb{R}^n$ y $F \in C^1(\mathbb{R}^n)$, entonces el sistema de ecuaciones (3.1) puede ser visto como

$$F(x) = 0 \tag{3.2}$$

La ecuación (3.2) es la forma en que nosotros identificaremos un sistema de ecuaciones no lineales. El jacobiano de F en el punto x, denotado por $F'(x)$, es definido por

$$J(x) = F'(x) = \begin{bmatrix} \nabla f_1(x)^t \\ \vdots \\ \nabla f_n(x)^t \end{bmatrix}$$

El método de Newton consiste en resolver un problema difícil mediante sucesivas resoluciones de un problema fácil, se espera que cada solución del problema fácil sea una mejor aproximación para el problema difícil. Así, sea $x^{(k)}$ una aproximación inicial a una solución del sistema $F(x) = 0$ (problema difícil), si tomamos una función L_k tal que $L_k(x) \approx F(x)$, para todo x en una vecindad de $x^{(k)}$, entonces esperamos que la solución del sistema $L_k(x) = 0$ (problema fácil) sea una mejor aproximación que $x^{(k)}$ para una solución de $F(x) = 0$.

El método de Newton es de carácter iterativo y está basado en la aproximación lineal de la función F en torno al punto actual $x^{(k)}$:

$$L_k(x) = F(x^{(k)}) + J(x^{(k)})(x - x^{(k)}) \tag{3.3}$$

El próximo punto, $x^{(k+1)}$, es definido como la solución de

$$L_k(x) = 0 \tag{3.4}$$

Si $J\left(x^{(k)}\right)$ es no singular, entonces (3.4) tiene solución única. En estas condi-

ciones, una iteración newton consiste en calcular $x^{(k+1)}$ previamente conocido $x^{(k)}$, mediante la resolución del sistema lineal:

$$\begin{aligned} J(x^{(k)})d^{(k)} &= -F(x^{(k)}) \\ x^{(k+1)} &= x^{(k)} + d^{(k)} \end{aligned} \tag{3.5}$$

Lo que dice (3.5) es que una vez conocido el *paso de newton* $d^{(k)}$, es posible calcular $x^{(k+1)}$.

Nota 3.1 *De un modo más directo, el método de Newton puede ser visto del siguiente modo: calcular* $x^{(k+1)}$ *una vez conocido* $x^{(k)}$*, mediante la siguiente regla:*

$$x^{(k+1)} = x^{(k)} - J^{-1}(x^{(k)})F(x^{(k)}) \tag{3.6}$$

El método de Newton es aplicado originalmente para resolver un sistema de ecuaciones $F(x) = 0$, donde $F : \mathbb{R}^n \longmapsto \mathbb{R}^n$ admite primera derivada continua. El siguiente algoritmo resume el método de newton.

Algoritmo 3.1 *(Algoritmo Básico de Newton)*

Sea $x^{(0)} \in \mathbb{R}^n$ *un punto inicial y* $\varepsilon > 0$ *el parámetro de precisión deseado:*

Paso 1: *Si* $\left\|F(x^{(k)})\right\| < \varepsilon$*, detenerse,* $x^{(k)}$ *es la aproximación buscada. Caso contrario, ir al paso 2.*

Paso 2: *Resolver el sistema* $J(x^{(k)})d^{(k)} = -F(x^{(k)})$ *y obtener el paso de newton* $d^{(k)}$*. Ir al paso 3.*

Paso 3: *Calcular* $x^{(k+1)} = x^{(k)} + d^{(k)}$*, hacer* $k \longleftarrow k + 1$*. Volver al paso 1.*

Naturalmente, el sistema $F(x) = 0$ puede no tener solución, entonces es posible modificar el algoritmo 3.1 para que finalice después de un número determinado de iteraciones indicando la posibilidad de infactibilidad.

El siguiente ejemplo muestra cómo puede ser aplicado el método de Newton para encontrar las intersecciones de dos curvas en $\mathbb{R}^2$. Como veremos, este problema se resume a la solución de un sistema de dos ecuaciones y dos variables.

Ejemplo 3.1 *Consideremos el sistema de ecuaciones dado por*

$$\begin{cases} (x_1-5)^2+(x_2-3)^2 &= 2 \\ x_1+x_2^2 &= 9 \end{cases}$$

El gráfico donde se aprecian las soluciones está representado en la figura 3.1. Este sistema lo podemos representar como

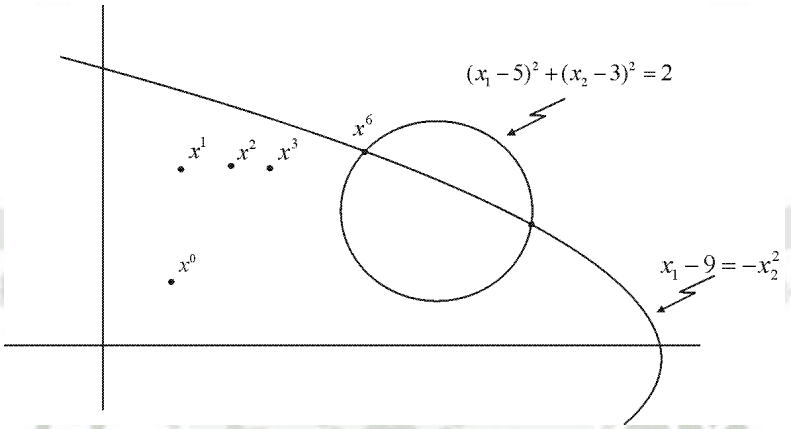

Figura 3.1: Intersecciones de la circunferencia $(x_1-5)^2+(x_2-3)^2=2$ y la parábola $x_1-9=-x_2^2$

$$F(x)=0$$

donde

$$x=\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}, \qquad F(x)=\begin{bmatrix} (x_1-5)^2+(x_2-3)^2-2 \\ x_1+x_2^2-9 \end{bmatrix}=\begin{bmatrix} F_1(x) \\ F_2(x) \end{bmatrix}$$

Observe que el jacobiano de F en un punto x está dado por:

$$J(x)=\begin{bmatrix} \frac{\partial F_1}{\partial x_1}(x) & \frac{\partial F_1}{\partial x_2}(x) \\ \frac{\partial F_2}{\partial x_1}(x) & \frac{\partial F_2}{\partial x_2}(x) \end{bmatrix}=\begin{bmatrix} 2(x_1-5) & 2(x_2-3) \\ 1 & 2x_2 \end{bmatrix}$$

Considerando como punto inicial $x^0 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$ *y* $\varepsilon = 0{,}000001$, *vemos que el método de Newton genera la siguiente sucesión de puntos en* $\mathbb{R}^2$*:*

$$x^0 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \quad x^1 = \begin{bmatrix} 1{,}666666 \\ 4{,}166666 \end{bmatrix}, \quad x^2 = \begin{bmatrix} 2{,}769993 \\ 2{,}830934 \end{bmatrix}, \quad x^3 = \begin{bmatrix} 3{,}476325 \\ 2{,}391059 \end{bmatrix}$$

$$x^4 = \begin{bmatrix} 3{,}741902 \\ 2{,}295062 \end{bmatrix}, \quad x^5 = \begin{bmatrix} 3{,}779283 \\ 2{,}284911 \end{bmatrix}, \quad x^6 = \begin{bmatrix} 3{,}780003 \\ 2{,}284731 \end{bmatrix}$$

En este punto, x^6*, notamos que* $\|F(x^6)\| = 5{,}519172 \times 10^{-7} < 0{,}000001$*, donde* $\|\cdot\|$ *representa la norma euclídea. Observe que el método de Newton otorga una aproximación casi exacta a la verdadera solución.*

Observe que. para encontrar la otra solución, debemos utilizar otro punto inicial. La elección del punto inicial es una desventaja del método de Newton, pues se requiere en teoría que dicho punto debería estar lo más cercano a la verdadera solución.

3.2. El Método de Lagrange (1788)

El método de Lagrange consiste en trasformar un problema de optimización con restricciones de igualdad en un problema sin restricciones. Consideremos el problema de programación no lineal

$$\begin{aligned} &\text{Minimizar } f(x) \\ &\text{Sujeto a:} \\ &g_i(x) = 0 \qquad i = 1, ..., m \end{aligned} \tag{3.7}$$

donde $f, g_i : \mathbb{R}^n \mapsto \mathbb{R}$. La técnica para resolver (3.7) consiste primeramente en definir la función Lagrangiana asociada a este problema

$$\mathcal{L}(x, y) = f(x) - \sum_{i=1}^{m} y_i \cdot g_i(x) \tag{3.8}$$

donde $y \in \mathbb{R}^m$ es el vector de variables lagrangianas. En segundo lugar minimizamos la función sin restricciones $\mathcal{L}(x, y)$, note que para minimizar (3.8) nosotros derivamos parcialmente $\mathcal{L}(x, y)$ e igualamos a cero, resultando el sistema de $n + m$ ecuaciones con $n + m$ variables:

$$\frac{\partial \mathcal{L}}{\partial x_j}(x, y) = \frac{\partial f}{\partial x_j}(x) - \sum_{i=1}^{m} y_i \cdot \frac{\partial g_i}{\partial x_j}(x) = 0 \qquad j = 1, ..., n \tag{3.9}$$

$$\frac{\partial \mathcal{L}}{\partial y_i}(x, y) = -g_i(x) = 0 \qquad i = 1, ..., m \tag{3.10}$$

La verdadera utilidad de este procedimiento está en que, bajo ciertas condiciones, tales como la convexidad de la función objetivo y las funciones que conforman las restricciones en (3.7), si encontramos un punto $(\bar{x}, \bar{y})$ que resuelve el sistema dado por (3.9) y (3.10), entonces $\bar{x}$ es un candidato potencial para ser solución de (3.7). Más aún, si las funciones en las restricciones son lineales, $\bar{x}$ será una solución global del problema.

El siguiente ejemplo muestra de manera detallada cómo puede ser aplicado el método de Lagrange en la optimización de problemas que encuadran en la forma requerida (3.7).

Ejemplo 3.2 *Consideremos el problema*

$$\begin{aligned} &\textit{Minimizar } 3x_1^2 + 5x_2^2 + 2x_1 - 3x_2 + 10 \\ &\textit{Sujeto a:} \\ &\qquad x_1^2 + x_2^2 = 8 \end{aligned} \tag{3.11}$$

Claramente, en este caso tenemos

$$f(x) = 3x_1^2 + 5x_2^2 + 2x_1 - 3x_2 + 10$$

y

$$g_1(x) = x_1^2 + x_2^2 - 8$$

La función Lagrangiana es de la forma

$$\mathcal{L}(x_1, x_2, y_1) = 3x_1^2 + 5x_2^2 + 2x_1 - 3x_2 + 10 - y_1\left(x_1^2 + x_2^2 - 8\right)$$

Observe que y_1 es la variable lagrangiana asociada a la única restricción del problema. Procedemos a minimizar $\mathcal{L}$, para lo cual hallamos las derivadas parciales e igualamos a cero, obteniendo el siguiente sistema no lineal de ecuaciones:

$$\begin{aligned} \frac{\partial \mathcal{L}}{\partial x_1}(x_1, x_2, y_1) &= 6x_1 + 2 - 2x_1 y_1 = 0 \\ \frac{\partial \mathcal{L}}{\partial x_2}(x_1, x_2, y_1) &= 10x_2 - 3 - 2x_2 y_1 = 0 \\ \frac{\partial \mathcal{L}}{\partial y_1}(x_1, x_2, y_1) &= -\left(x_1^2 + x_2^2 - 8\right) = 0 \end{aligned} \tag{3.12}$$

o mejor aún:

$$\begin{cases} 6x_1 + 2 - 2x_1 y_1 &= 0 \\ 10x_2 - 3 - 2x_2 y_1 &= 0 \\ -x_1^2 - x_2^2 + 8 &= 0 \end{cases}$$

Observe que en este caso, para aplicar el Método de Newton, hacemos:

$$F(x) = \begin{bmatrix} 6x_1 + 2 - 2x_1 y_1 \\ 10x_2 - 3 - 2x_2 y_1 \\ -x_1^2 - x_2^2 + 8 \end{bmatrix}, \; J(x) = \begin{bmatrix} 6 - 2y_1 & 0 & -2x_1 \\ 0 & 10 - 2y_1 & -2x_2 \\ -2x_1 & -2x_2 & 0 \end{bmatrix}$$

y consideramos el punto inicial

$$x^0 = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}, \; \textit{donde } \varepsilon = 0{,}000001$$

obteniendo después de 7 iteraciones:

$$\begin{aligned} x_1 &= 0{,}6881 \\ x_2 &= 2{,}7434 \\ y_1 &= 4{,}4532 \end{aligned}$$

Así, $\begin{bmatrix} 0{,}6881 \\ 2{,}7434 \end{bmatrix}$ *es una buena aproximación a la solución óptima del problema*

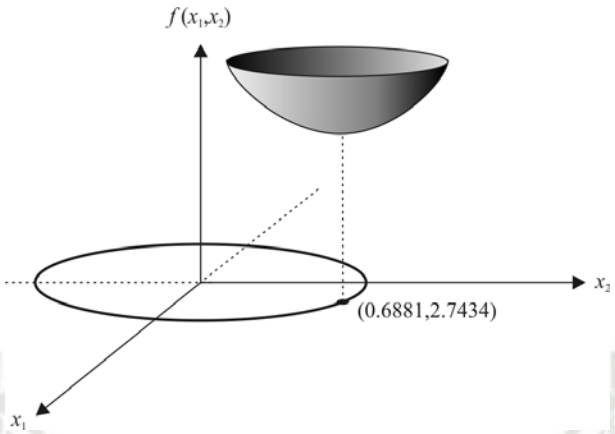

Figura 3.2:

(3.11).

3.3. El Método de Barrera Logaritmo (Fiacco & McCormick, 1968)

La idea del método es convertir un problema de minimización con condiciones de no negatividad en una familia monoparamétrica de problemas de minimización sin restricciones. De este modo, el problema

$$\begin{aligned} &\text{Minimizar } f(x) \\ &\text{Sujeto a:} \\ &x \geq 0 \end{aligned} \tag{3.13}$$

es reemplazado por la familia (debido a μ) de problemas sin restricciones dada por

$$\text{Minimizar } B_\mu(x) = f(x) - \mu \sum_{j=1}^{n} \ln(x_j) \tag{3.14}$$

El problema en (3.14) es parametrizado por el parámetro de barrera positivo[1] μ. Sea $x(\mu)$ el minimizador de $B_\mu(x)$ y $\bar{x}$ una solución de (3.13), Fiacco &

[1] La expresión $-\mu \sum_{j=1}^{n} \ln(x_j)$ en (3.14) no debe dejar que x se aproxime a la frontera de la región factible. Esto se consigue si $\mu > 0$. De ahí el nombre de función barrera.

McCormick [3] demostraron que cuando μ se aproxima a cero, entonces $x(\mu)$ se aproxima a $\bar{x}$. Es decir:

$$\mu \to 0 \Longrightarrow x(\mu) \to \bar{x}$$

Ejemplo 3.3 *Consideremos el problema no lineal dado por*

$$\begin{array}{l} \textit{Minimizar } (x_1-4)^2+(x_2-6)^2+9 \\ \textit{Sujeto a:} \\ \qquad x_1, x_2 \geq 0 \end{array} \tag{3.15}$$

Observe que

$$f(x) = (x_1-4)^2+(x_2-6)^2+9$$

Transformando este problema a la forma dada en (3.14), tenemos

$$\textit{Minimizar } B_\mu(x) = (x_1-4)^2+(x_2-6)^2+9-\mu\{\ln(x_1)+\ln(x_2)\}$$

Para minimizar este problema aparentemente irrestricto, pues debemos mantener $x_1 > 0$ y $x_2 > 0$, procedemos a calcular las derivadas parciales e igualarlas a cero, obteniéndose el sistema no lineal:

$$\begin{aligned} \frac{\partial B_u}{\partial x_1}(x_1,x_2) &= 2(x_1-4)-\frac{\mu}{x_1}=0 \\ \frac{\partial B_u}{\partial x_2}(x_1,x_2) &= 2(x_2-6)-\frac{\mu}{x_2}=0 \end{aligned}$$

o sino:

$$\left\{\begin{array}{rcl} 2(x_1-4)-\frac{\mu}{x_1} &=& 0 \\ 2(x_2-6)-\frac{\mu}{x_2} &=& 0 \end{array}\right. \tag{3.16}$$

Resolviendo, para un valor de μ fijo, obtenemos:

$$\begin{aligned} x_1(\mu) &= 2 \pm \sqrt{4+\frac{\mu}{2}} \\ x_2(\mu) &= 3 \pm \sqrt{9+\frac{\mu}{2}} \end{aligned}$$

Luego, haciendo tender μ hacia cero, debemos obtener la solución del problema original (3.15). En efecto,

$$\mu \to 0 \Longrightarrow \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \to \begin{bmatrix} 4 \\ 6 \end{bmatrix}$$

pero también

$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \to \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

Luego, tenemos dos candidatos para la solcuión óptima de (3.15), después de analizarlos, vemos que $\begin{bmatrix} 4 \\ 6 \end{bmatrix}$ *es el minimizador global procurado (vea la figura 3.3).*

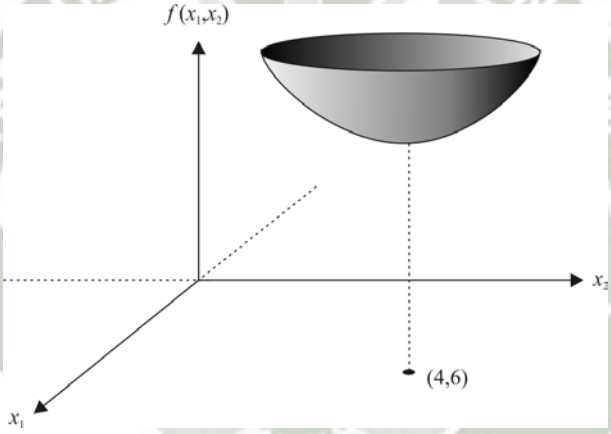

Figura 3.3: Solución óptima para el ejemplo 3.3 cuando $\mu \to 0$

Cuando el problema es de mayor dimensión, el sistema (3.16) puede ser impracticable, tornándose necesario utilizar un méttodo numérico para resolverlo, tal como el Método de Newton.

3.4. Construcción del Método de Puntos Interiores Primal-Dual

En esta sección resumiremos los hechos más importantes de la construcción del método de puntos interiores primal-dual. Como fue mencionado anteriormente, este método usará tanto el primal como su respectivo dual para resolver el problema de programación lineal en la forma estándar. Como complemento, el método estará fundamentado en resultados ya conocidos dados por el método de Newton, el método de Lagrange y el Método de Barrera. logaritmo.

El Método de Puntos Interiores Primal-Dual trabaja directamente, al igual que el Simplex, sobre un Problema de Programación Lineal Estándar general, al cual llamaremos en esta sección el *primal*:

$$\begin{aligned} &\text{Minimizar } c^t x \\ &\text{Sujeto a:} \\ &\quad Ax = b \\ &\quad x \geq 0 \end{aligned} \qquad \text{(Primal)}$$

donde $A \in \mathbb{R}^{m\times n}$ es de rango completo, $b \in \mathbb{R}^m$ y $c \in \mathbb{R}^n$. Como se sabe, la región de factibilidad para (Primal) es un poliedro convexo, el cual será representado por

$$P = \{x \in \mathbb{R}^n : \ Ax = b, \ x \geq 0\}$$

Definición 3.1 *Se dice que el punto $\bar{x} \in \mathbb{R}^n$ está en el interior de P, si $A\bar{x} = b$ y $\bar{x} > 0$.*

Ejemplo 3.4 *Considere el problema*

$$\begin{aligned} &\textit{Minimizar } 2x_1 + 3x_2 \\ &\textit{Sujeto a:} \\ &x_1 + x_2 + x_3 \ = \ 5 \\ &x_1, x_2, x_3 \ \geq \ 0 \end{aligned}$$

El punto $\begin{bmatrix}1 & 1 & 3\end{bmatrix}^T$ *está en el interior de*

$$P = \left\{x \in \mathbb{R}^3 : \ x_1 + x_2 + x_3 = 5, \ x_1, x_2, x_3 \geq 0\right\}$$

pero el punto $\begin{bmatrix}3 & 2 & 0\end{bmatrix}^T$*, a pesar de pertenecer a la región de factibilidad, no es un punto interior, vea figura 3.4.*

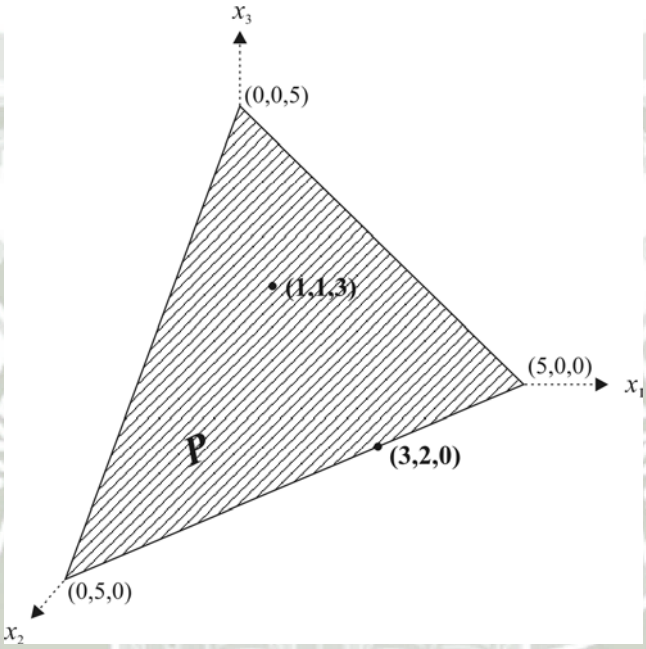

Figura 3.4: $\begin{bmatrix}1 & 1 & 3\end{bmatrix}^T$ está en el interior de P, mientras que $\begin{bmatrix}3 & 2 & 0\end{bmatrix}^T$ no, pues está en la frontera de P. Observe que la región factible no es todo el tetraedro, sino que sólo el triángulo bidimensional.

Definición 3.2 *El siguiente Problema de Programación Lineal, al cual llamaremos* Dual *asociado al problema (Primal), o simplemente* Dual*, está definido por:*

$$\begin{aligned} &\textit{Maximizar } b^t y \\ &\textit{Sujeto a:} \\ &A^t y + z = c \\ &z \geq 0 \end{aligned} \qquad \text{(Dual)}$$

El Método de Puntos Interiores Primal-Dual consiste en resolver iterativa y simultáneamente ambos problemas, Primal y Dual, utilizando para eso el Método de Barrera Logaritmo, el Método de Lagrange y el Método de Newton. Estas técnicas fueron introducidas al inicio de este capítulo.

Para tal objetivo, primeramente usamos el Método de Barrera Logaritmo y *eliminamos* las variables no negativas del problema (Primal), obteniendo así el siguiente problema:

$$\begin{aligned} &\text{Minimizar } c^t x - \mu \sum_{i=1}^{n} \ln(x_i) \\ &\text{Sujeto a:} \\ &\qquad b - Ax = 0 \end{aligned} \tag{3.17}$$

Observe que en el problema (3.17), la función objetivo es no lineal.

De modo similar, utilizamos el Método de Barrera Logaritmo y *eliminamos* las variables no negativas del problema (Dual), y obtenemos el siguiente problema:

$$\begin{aligned} &\text{Maximizar } b^t y + \mu \sum_{i=1}^{n} \ln(z_i) \\ &\text{Sujeto a:} \\ &\qquad c - A^t y - z = 0 \end{aligned} \tag{3.18}$$

Claramente, nada impide utilizar el mismo parámetro μ en (3.17) y (3.18). Observe que el signo "+.$^{\text{en}}$ la función objetivo de (3.18) se debe a que el problema es de maximización.

Algo que debemos tener en cuenta en el problema (3.17) es que las variables x_i, $i = 1, ..., n$, deben ser positivas, debido a la presencia de logaritmo. Por la misma razón, en el problema (3.18), las variables z_i, $i = 1, ..., n$, deben ser también positivas.

Seguidamente, utilizando el Método de Lagrange, *eliminamos* las restric-

ciones de igualdad en (3.17) y (3.18), obteniendo:

$$\text{Minimizar } \mathcal{L}_P\left(x, \lambda_P, \mu\right) = c^t x - \mu \sum_{i=1}^{n} \ln(x_i) + (b - Ax)^T \lambda_P \tag{3.19}$$

y

$$\text{Maximizar } \mathcal{L}_D\left(y, z, \lambda_D, \mu\right) = b^t y + \mu \sum_{i=1}^{n} \ln(z_i) + \left(c - A^t y - z\right)^T \lambda_D \tag{3.20}$$

Observe que $\mathcal{L}_P\left(x, \lambda_P, \mu\right)$ y $\mathcal{L}_D\left(y, z, \lambda_D, \mu\right)$, en (3.19) y (3.20), pueden ser tratados como funciones no lineales sin restrcciones. En realidad, tenemos que recordar que las componentes de x y z deben ser mantenidas estrictamente mayores que cero, debido al logaritmo, eso se conseguirá con un mecanismo adicional, como veremos más adelante.

Fijando $\mu > 0$ y utilizando las condiciones de optimalidad para minimizar $\mathcal{L}_P\left(x, \lambda_P, \mu\right)$ y maximizar $\mathcal{L}_D\left(y, z, \lambda_D, \mu\right)$, es decir, derivando parcialmente $\mathcal{L}_P$ con respecto a x y λ_P, y derivando parcialmente $\mathcal{L}_D$ con respecto a y, z y λ_D, e igualando a cero:

$$\begin{aligned} \frac{\partial \mathcal{L}_P}{\partial x_i}\left(x, \lambda_P, \mu\right) &= 0 \qquad i = 1, ..., n \\ \frac{\partial \mathcal{L}_P}{\partial \lambda_{Pi}}\left(x, \lambda_P, \mu\right) &= 0 \qquad i = 1, ..., m \end{aligned}$$

y

$$\begin{aligned} \frac{\partial \mathcal{L}_D}{\partial y_i}\left(y, z, \lambda_D, \mu\right) &= 0 \qquad i = 1, ..., m \\ \frac{\partial \mathcal{L}_D}{\partial z_i}\left(y, z, \lambda_D, \mu\right) &= 0 \qquad i = 1, ..., n \\ \frac{\partial \mathcal{L}_D}{\partial \lambda_{Di}}\left(y, z, \lambda_D, \mu\right) &= 0 \qquad i = 1, ..., n \end{aligned}$$

obtenemos respectivamente:

$$\begin{aligned}
b_i - a_i^T x &= 0 \quad i = 1, ..., m \quad (a_i \text{ es la } i\text{-fila de } A) \\
c_j - \frac{\mu}{x_j} - a_j^T \lambda_P &= 0 \quad j = 1, ..., n \quad (a_j \text{ es la } j\text{-columna de } A) \\
b_i - a_i^T \lambda_D &= 0 \quad i = 1, ..., m \quad (a_i \text{ es la } i\text{-fila de } A) \qquad (3.21) \\
\frac{\mu}{z_j} - \lambda_{Dj} &= 0 \quad j = 1, ..., n \\
c_j - a_j^T y - z_j &= 0 \quad j = 1, ..., n \quad (a_j \text{ es la } j\text{-columna de } A)
\end{aligned}$$

Sea $e = \begin{bmatrix} 1 & 1 & \cdots & 1 \end{bmatrix}^t \in \mathbb{R}^n$, dados $x, z \in \mathbb{R}^n$, con $x > 0$ y $z > 0$, denotemos:

$$X = \begin{bmatrix} x_1 & 0 & 0 & 0 \\ 0 & x_2 & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & x_n \end{bmatrix} \quad \text{y} \quad Z = \begin{bmatrix} z_1 & 0 & 0 & 0 \\ 0 & z_2 & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & z_n \end{bmatrix}$$

Claramente:

$$X^{-1} = \begin{bmatrix} \frac{1}{x_1} & 0 & 0 & 0 \\ 0 & \frac{1}{x_2} & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \frac{1}{x_n} \end{bmatrix} \quad \text{y} \quad Z^{-1} = \begin{bmatrix} \frac{1}{z_1} & 0 & 0 & 0 \\ 0 & \frac{1}{z_2} & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \frac{1}{z_n} \end{bmatrix}$$

Por lo tanto, las condiciones de optimalidad (3.21) pueden ser vistas como:

$$\begin{aligned}
b - Ax &= 0 \\
c - \mu X^{-1} e - A^t \lambda_P &= 0 \\
b - A\lambda_D &= 0 \qquad (3.22) \\
\mu Z^{-1} e - \lambda_D &= 0 \\
c - A^t y - z &= 0
\end{aligned}$$

Es posible mostrar que el vector λ_P es en realidad el vector de variables duales del problema (Primal), es decir: $\lambda_P = y$. Así mismo, λ_D es en realidad el vector de variables duales del problema (Dual), es decir: $\lambda_D = x$, tal como se mostrará en el lema que viene a continuación.

Antes de enunciar el lema, vamos a requerir la definición de dualidad para problemas generales de optimización.

Definición 3.3 *Denominamos problema dual asociado al problema*

$$\begin{gathered} \textit{Minimizar } f(x) \\ h(x) = 0 \\ g(x) \leq 0 \end{gathered}$$

donde $f : \mathbb{R}^n \to \mathbb{R}$, $h : \mathbb{R}^n \to \mathbb{R}^m$, $g : \mathbb{R}^n \to \mathbb{R}^p$ *y* f, h, g *con derivadas parciales continuas en* $\mathbb{R}^n$, *al problema de optimización dado por:*

$$\begin{gathered} \textit{Maximizar } f(x) + \sum_{i=1}^{m} \lambda_i h_i(x) + \sum_{i=1}^{p} u_i g_i(x) \\ \nabla f(x) + \sum_{i=1}^{m} \lambda_i \nabla h_i(x) + \sum_{i=1}^{p} u_i \nabla g_i(x) = 0 \\ u_i \geq 0, \qquad i = 1, 2, ..., p \end{gathered}$$

Lema 3.1 *En el sistema (3.22):*

1. λ_P *coincide con* y.
2. λ_D *coincide* x.

Prueba. Consideremos el problema (3.19),

$$\text{Minimizar } c^t x - \mu \sum_{i=1}^{n} \ln(x_i) + (b - Ax)^T \lambda_P$$

El dual de este problema, mediante la definición 3.3, es

$$\begin{aligned} &\text{Maximizar } c^t x - \mu \sum_{i=1}^{n} \ln(x_i) + (b - Ax)^T \lambda_P \\ &\text{Sujeto a:} \\ &c - \mu \begin{bmatrix} \frac{1}{x_1} \\ \vdots \\ \frac{1}{x_n} \end{bmatrix} - A^t \lambda_P = 0 \end{aligned} \tag{3.23}$$

Por otro lado, hallando la transpuesta en la ecuación de (3.23) y multiplicando por x, tenemos

$$\begin{aligned} c^t x - \mu \begin{bmatrix} \frac{1}{x_1} & \cdots & \frac{1}{x_n} \end{bmatrix} x - \lambda_P^t A x &= 0 \\ c^t x - \mu n - \lambda_P^t A x &= 0 \\ c^t x &= \mu n + \lambda_P^t A x \end{aligned}$$

Reemplazando en (3.23) tenemos

$$\begin{aligned} &\text{Maximizar } b^t \lambda_P - \mu \sum_{i=1}^{n} \ln(x_i) + \mu n \\ &\text{Sujeto a:} \\ &c - \begin{bmatrix} \frac{\mu}{x_1} \\ \vdots \\ \frac{\mu}{x_n} \end{bmatrix} - A^t \lambda_P = 0 \end{aligned}$$

Definiendo $w_i = \frac{\mu}{x_i}$ (observe que $w_i > 0$ si , y sólo si, $x_i > 0$) y reemplazando en el problema anterior, tenemos

$$\begin{aligned} &\text{Maximizar } b^t \lambda_P + \mu \sum_{i=1}^{n} \ln w_i + \mu n - \mu \sum_{i=1}^{n} (\ln \mu) \\ &\text{Sujeto a:} \\ &c - w - A^t \lambda_P = 0 \end{aligned}$$

Aplicando la estrategia de Lagrange al anterior problema, tenemos

$$\text{Maximizar } b^t\lambda_P + \mu\sum_{i=1}^{n}\ln w_i + \left(c - w - A^t\lambda_P\right)^t\gamma + \mu n - \mu\sum_{i=1}^{n}(\ln\mu) \quad (3.24)$$

donde γ es el vector de variables lagrangianas para (3.24). Finalmente, como $\mu n - \mu\sum_{i=1}^{n}(\ln\mu)$ no depende de las variables del problema, el anterior problema es equivalente a:

$$\text{Maximizar } b^t\lambda_P + \mu\sum_{i=1}^{n}\ln w_i + \left(c - w - A^t\lambda_P\right)^t\gamma \quad (3.25)$$

Observe que el problema (3.25) es el mismo que el problema (3.20), excepto por los nombres de los vectores λ_P, w y γ. Por lo tanto, renombrando γ por λ_D, w por z y λ_P por y, tenemos que (3.25) y (3.20) son el mismo problema. Así, λ_P coincide con y, tal como se quería probar.

Intercambiando papeles, ahora comenzando con el problema (3.20) y mediante un razonamiento análogo, se puede conseguir probar que λ_D coincide con x. ■

Usando el Lema 3.1, el sistema (3.22) es equivalente al sistema:

$$\begin{aligned} c - \mu X^{-1}e - A^t y &= 0 \\ b - Ax &= 0 \\ \mu Z^{-1}e - Ix &= 0 \\ c - A^t y - z &= 0 \end{aligned} \quad (3.26)$$

Como $c - A^t y = z$, entonces $z - \mu X^{-1}e = 0$. Multiplicando ambos lados de $z - \mu X^{-1}e = 0$ por X y ambos lados de $\mu Z^{-1}e - Ix = 0$ por Z, el sistema

(3.26) puede ser visto como:

$$\begin{aligned} b - Ax &= 0 \\ c - A^t y - z &= 0 \\ XZe - \mu e &= 0 \end{aligned}$$

En resumen, las condiciones de optimalidad para los problemas (3.19) y (3.20), están dadas por:

$$\begin{aligned} Ax &= b \\ A^t y + z &= c \\ XZe &= \mu e \end{aligned} \tag{3.27}$$

o sino, por:

$$\begin{aligned} Ax &= b \\ A^t y + z &= c \\ x_i z_i &= \mu, \quad i = 1, ..., n \end{aligned} \tag{3.28}$$

Recordando que asumimos $x > 0$ y $z > 0$, el sistema (3.27) prácticamente coincide con las condiciones de optimalidad (1.13), (1.14) y (1.15) del Problema de Programación Lineal Estándar (Primal):

$$\begin{aligned} Ax &= b, \quad x \geq 0 \\ A^t y + z &= c, \quad z \geq 0 \\ x_i z_i &= 0, \quad i = 1, ..., n \end{aligned}$$

Esto significa que, al resolver (3.27) o (3.28), manteniendo las componentes de los vectores x y z estrictamente positivas (caso contrario no es-

tarían definidos los logaritmos en (3.19) y (3.20)), y tomando el valor de $\mu > 0$ suficientemente pequeño (por la naturaleza del Método de Barrera), deberíamos aproximarnos arbitrariamente a la solución óptima del Problema de Programación Lineal Estándar (Primal).

Ahora, para resolver numéricamente el sistema (3.27) usaremos el algoritmo de Newton. Sean los puntos iniciales $x^{(0)}, z^{(0)} \in \mathbb{R}^n$, $y^{(0)} \in \mathbb{R}^m$, donde $x^{(0)} > 0$ y $z^{(0)} > 0$. Fijemos además un valor inicial[2] para el parámetro de penalización, $\mu = \mu_0$.

Como es familiar, el sistema (3.27) lo podemos representar por

$$F(x, y, z, \mu) = 0$$

donde

$$F(x, y, z, \mu) = \begin{bmatrix} Ax - b \\ A^t y + z - c \\ XZe - \mu e \end{bmatrix}$$

El jacobiano de F con repecto a (x, y, z) en el punto (x, y, z, μ), está dado por:

$$J(x, y, z, \mu) = \begin{bmatrix} A & 0 & 0 \\ 0 & A^t & I \\ Z & 0 & X \end{bmatrix}$$

Así, el paso de Newton, d, lo obtenemos resolviendo el sistema lineal:

$$J(x^{(0)}, y^{(0)}, z^{(0)}, \mu_0)d = -F(x^{(0)}, y^{(0)}, z^{(0)}, \mu_0) \tag{3.29}$$

Observe que en estas condiciones, el vector (paso de Newton) d es de la forma

$$d = \begin{bmatrix} dx \\ dy \\ dz \end{bmatrix}$$

[2] μ_0 no pude ser demasiado pequeño, debido a la inestabilidad numérica que ocasiona el Método de Barrera Logaritmo.

donde dx es una dirección newton en el x-espacio, dy en el y-espacio y dz en el z-espacio. Definamos

$$\begin{aligned} d_P &= b - Ax^{(0)} \\ d_D &= A^T y^{(0)} + z^{(0)} - c \end{aligned}$$

En estas condiciones, el sistema (3.29) puede ser visto como:

$$\begin{bmatrix} A & 0 & 0 \\ 0 & A^t & I \\ Z & 0 & X \end{bmatrix} \begin{bmatrix} dx \\ dy \\ dz \end{bmatrix} = \begin{bmatrix} d_P \\ -d_D \\ \mu e - XZe \end{bmatrix} \tag{3.30}$$

Podemos intentar aún despejar manualmente dx, dy y dz desde el sistema (3.30), de donde obtenemos:

$$Adx = d_P \tag{3.31}$$

$$A^t dy + dz = -d_D \tag{3.32}$$

$$Zdx + Xdz = \mu e - XZe \tag{3.33}$$

Multiplicando ambos lados de la ecuación (3.32) por $AZ^{-1}X$, tenemos

$$\begin{aligned} AZ^{-1}XA^t dy + AZ^{-1}Xdz &= -AZ^{-1}Xd_D \\ (AZ^{-1}XA^t)dy &= -AZ^{-1}Xd_D - AZ^{-1}Xdz \end{aligned} \tag{3.34}$$

Despejando Xdz de la ecuación (3.33) y reemplazando en la ecuación (3.34), tenemos

$$\begin{aligned} (AZ^{-1}XA^t)dy &= -AZ^{-1}Xd_D - AZ^{-1}(\mu e - XZe - Zdx) \\ &= -AZ^{-1}Xd_D - AZ^{-1}\mu e + AZ^{-1}XZe + AZ^{-1}Zdx \\ &= -AZ^{-1}Xd_D - \mu AZ^{-1}e + AXe + Adx \\ &= -AZ^{-1}Xd_D - \mu AZ^{-1}e + b + Adx \end{aligned} \tag{3.35}$$

Ahora, usando (3.35) y (3.31) tenemos:

$$(AZ^{-1}XA^t)dy = b - \mu AZ^{-1}e - AZ^{-1}Xd_D \tag{3.36}$$

Además, usando (3.32) y (3.33) obtenemos:

$$dz = -d_D - A^t dy \tag{3.37}$$

$$dx = Z^{-1}\left(\mu e - XZe - Xdz\right) \tag{3.38}$$

En resumen, el cálculo del paso de Newton en (3.29) puede ser hecho por partes, resolviendo separadamente los sistemas lineales consecutivos de menor dimensión (3.36), (3.37) y (3.38):

1. $(AZ^{-1}XA^t)dy = b - \mu AZ^{-1}e - AZ^{-1}Xd_D$
2. $dz = -d_D - A^t dy$
3. $dx = Z^{-1}\left(\mu e - XZe - Xdz\right)$

Puesto que debemos controlar que los vectores x y z sean estrictamente positivos, podemos utilizar el *criterio de la razón*, el cual consiste en calcular:

$$\alpha_P = \min_{1\leq i\leq n}\left\{-\frac{x_i}{dx_i} : \ dx_i < 0\right\}$$

y

$$\alpha_D = \min_{1\leq i\leq n}\left\{-\frac{z_i}{dz_i} : \ dz_i < 0\right\}$$

La lógica del criterio de la razón puede ser vista claramente en la Figura 3.5, su objetivo es mantener las variables x e z estrictamente positivas. No obstante, las variables y no deberían ser controladas debido a que son irrestrictas en signo, por la definición de dualidad en (1.12). Un razonamiento análogo vale para controlar la permanencia en el interior del primer cuadrante en el z-espacio.

Mediante esto, en cada iteración mantenemos $x > 0$ y $z > 0$. Así, dado $x^k > 0$, el Criterio de la Razón nos permite desplazarnos hacia $x^{k+1} > 0$, eso

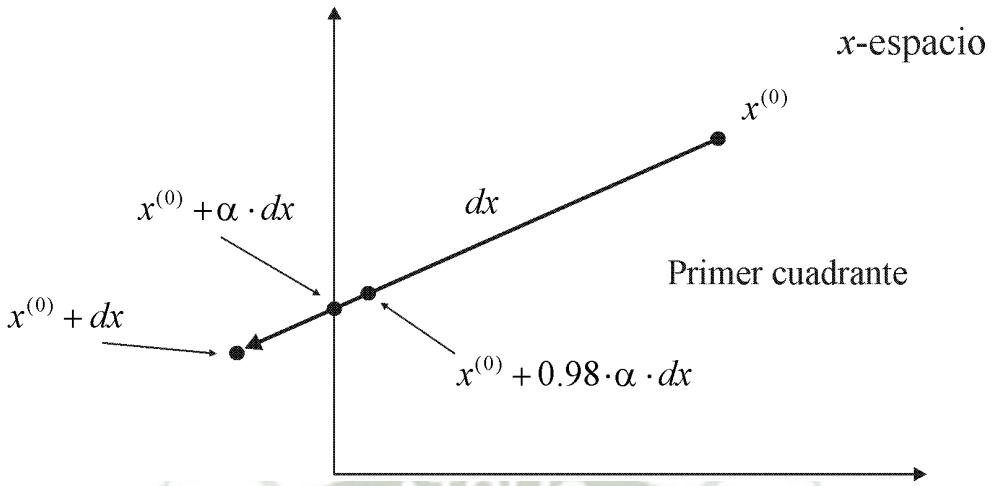

Figura 3.5: Observe que $x^{(0)}$ está en el primer cuadrante del x-espacio, mientras que dx representa la dirección de Newton relativa a dicho espacio. Observe también que $x^{(0)} + dx$ es infactible. Además, $x^{(0)} + \alpha \cdot dx$ es factible pero está en la frontera. Sin embargo, $x^{(0)} + 0{,}98\alpha \cdot dx$ está en el interior del primer cuadrante.

es conseguido haciendo:

$$\begin{aligned} x^{(1)} &= x^{(0)} + (0{,}98)(\alpha_P)dx \\ y^{(1)} &= y^{(0)} + (0{,}98)(\alpha_D)dy \\ z^{(1)} &= z^{(0)} + (0{,}98)(\alpha_D)dz \end{aligned}$$

Esto completa una iteración newton, pero a su vez constituye una iteración del Método de Puntos Interiores Primal-Dual. Para la siguiente iteración consideramos $x^{(1)}$, $y^{(1)}$ y $z^{(1)}$ como nuevos puntos iniciales.

La estrategia que acabamos de exponer sólo resuelve el sistema (3.28) con respecto a μ_0, para obtener una aproximación a una solución óptima para (Primal) deberíamos reducir μ, por ejemplo $\mu = \frac{\mu_0}{10}$, y repetir completamente el procedimiento de newton. Observe que nada impide reducir μ en cada iteración newton. En la práctica, según [5], dado μ_k, el siguiente parámetro

penalizador es elegido mediante:

$$\mu_{k+1} = \frac{|c^t x - b^t y|}{\Theta(n)}, \quad k = 0, 1, 2...$$

donde n es el número de variables y

$$\Theta(n) = \begin{cases} n^2 & \text{si } n \leq 5000 \\ n\sqrt{n} & \text{si } n > 5000 \end{cases} \tag{3.39}$$

La definición de Θ en (3.39) es por cuestiones prácticas y obedece a la recomendación hecha por el autor del método en [5]. Significa que en problemas de mediano porte es aceptable una reducción más rápida del parámetro penalizador μ_k, mientras que para problemas inmensos la reducción del parámetro penalizador debería ser más lenta, esto por cuestiones numéricas, pero debería ser posible estimar quizá otras estrategias para reducir μ_k con un fundamento teórico más apropiado.

Todo esto en su conjunto produce una reducción substancial en μ a cada paso newton, las iteraciones continúan hasta que la diferencia

$$\left|c^t x^{(k)} - b^t y^{(k)}\right|$$

sea suficientemente pequeña, esto se consigue controlando el error relativo:

$$\frac{\left|c^t x^{(k)} - b^t y^{(k)}\right|}{1 + |b^t y^{(k)}|} < \varepsilon \tag{3.40}$$

donde $\varepsilon > 0$ es la presición deseada. Se prefiere usar generalmente el error relativo como criterio de detención en problemas de optimización, debido a que con él medimos la proximidad de los valores objetivos teniendo en cuenta sus magnitudes, lo que no sería considerado si usáramos el error absoluto. Por ejemplo, si $c^t x^{(k)} = 1000000$ y $b^t y^{(k)} = 999990$, quizá debería se razonable deterner el algoritmo, pues debido a su magnitud de los valores objetivos ellos ya están muy próximos, lo cual no sería visualizado si usáramos el error absoluto como criterio de detención, pues éste vale 10, y probablemente el

algoritmo esté realizando más iteraciones de las necesarias.

Conclusión 3.1 *Cada iteración del Método de Puntos Interiores Primal-Dual puede ser interpretada como un procedimiento en el cual, el Problema de Programación Lineal Estándar y su Dual, son convertidos en problemas no lineales sin restricciones, mediante las técnicas de Lagrange y Barrera Logaritmo. El sistema resultante, dados los puntos iniciales, es solucionado mediante el Método de Newton disminuyéndose iterativamente el valor de μ y manteniéndose los variables x y z siempre positivas.*

3.4.1. El Algoritmo de Puntos Interiores Primal-Dual

El Método de Puntos Interiores Primal-Dual trabaja sobre un Problema de Programación Lineal Estándar con rango de A completo. Vale recalcar que cuando el rango de A es incompleto, es posible introducir variables artificiales al problema original, de un modo similar al que se realiza en la técnica M-Grande, la cual fue expuesta al final del primer capítulo.

El algoritmo necesita para iniciar tres vectores iniciales, $x, z \in \mathbb{R}^n$ donde $x, z > 0$ y $y \in \mathbb{R}^m$. El vector y es irrestricto, por lo que puede ser en escencia cualquier vector arbitrario. Por cuestiones prácticas, decidimos utilizar en la parte experimental $x = z = \begin{bmatrix} 1 & \cdots & 1 \end{bmatrix}^t$ y $y = \begin{bmatrix} 0 & \cdots & 0 \end{bmatrix}^t$, sin embargo, otras elecciones con una fundamentación más formal debería ser posible. El criterio de optimalidad con el cual se detendrá el algoritmo está soportado por la expresión introducida en (3.40). Finalmente, el cálculo del paso de Newton está soportado por los sistemas (3.36), (3.37) y (3.38).

Algoritmo 3.2 *(Algoritmo de Puntos Interiores Primal-Dual para Programación Lineal - PIPD)*

Inicialización: *Dados $y \in \mathbb{R}^m$ y $x, z \in \mathbb{R}^n$, donde $x, z > 0$. Definir la precisión requerida $\varepsilon > 0$, el n-vector columna $e = \begin{bmatrix} 1 & 1 & \cdots & 1 \end{bmatrix}^t$ y el valor inicial de $\mu > 0$.*

Paso 1: *Si $\frac{|c^t x - b^t y|}{1+|b^t y|} < \varepsilon$, pare el algoritmo, la aproximación a la solución óptima es x. Caso contrario, ejecutar los siguientes pasos.*

Paso 2: *Definir las matrices diagonales* $X = \text{diag}(x)$ *y* $Z = \text{diag}(z)$

Paso 3: *Calcular* $d_D = A^t y + z - c$

Paso 4: *Hallar* dy *resolviendo el sistema*

$$(AZ^{-1}XA^t)dy = b - \mu AZ^{-1}e - AZ^{-1}Xd_D$$

Paso 5: *Calcular* $dz = -d_D - A^t dy$

Paso 6: *Hallar* dx *resolviendo el sistema* $Zdx = \mu e - XZe - Xdz$

Paso 7: *Calcular* $\alpha_P = \min\limits_{1\leq i\leq n}\left\{-\frac{x_i}{dx_i} : dx_i < 0\right\}$. *Si* $\nexists i$, $i = 1, 2, ..., n$, *tal que* $dx_i < 0$, $\alpha_P = 1$.

Paso 8: *Calcular* $\alpha_D = \min\limits_{1\leq i\leq n}\left\{-\frac{z_i}{dz_i} : dz_i < 0\right\}$. $\nexists i$, $i = 1, 2, ..., n$, *tal que* $dz_i < 0$, $\alpha_D = 1$, $\alpha_D = 1$.

Paso 9: *Calcular* $\alpha = \min\{\alpha_P, \alpha_D\}$

Paso 9: *Hacer*

$$\begin{aligned} x &= x + (0{,}98)(\alpha)dx \\ y &= y + (0{,}98)(\alpha)dy \\ z &= z + (0{,}98)(\alpha)dz \end{aligned}$$

Paso 10: *Hacer* $\mu_{k+1} = \frac{|c^t x - b^t y|}{\Theta(n)}$ *y volver al paso 1.*

En el siguiente ejemplo mostraremos una iteración del algoritmo con respecto a un problema de dos variables.

Ejemplo 3.5 *Consideremos el siguiente problema en la forma estándar*

$$\begin{aligned} &\textit{Minimizar } 2x_1 + 3x_2 \\ &\textit{Sujeto a} \\ &\quad 4x_1 + 2x_2 = 10 \\ &\quad x_1, x_2 \geq 0 \end{aligned}$$

Observe que $m = 1$, $n = 2$,

$$A = \begin{bmatrix} 4 & 2 \end{bmatrix}, \qquad b = [10] \qquad y \qquad c = \begin{bmatrix} 2 \\ 3 \end{bmatrix}$$

Además, usaremos como vectores iniciales

$$x^0 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \; z^0 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \; y^0 = \begin{bmatrix} 0 \end{bmatrix} \; y \; \mu^0 = 10$$

la precisión requerida $\varepsilon = 0{,}000001$ *y el vector auxiliar* $e = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$.

Paso 1: *Calculamos*

$$\frac{|c^t x^0 - b^t y^0|}{1 + |b^t y^0|} = \frac{|5 - 0|}{1 + |0|} = 5 > \varepsilon$$

Así, aún no estamos en condiciones óptimas.

Paso 2: *Creamos*

$$X = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \; y \; Z = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

Paso 3: *Calculamos*

$$d_D = d_D = A^t y^0 + z^0 - c = \begin{bmatrix} 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \end{bmatrix} - \begin{bmatrix} 2 \\ 3 \end{bmatrix} = \begin{bmatrix} -1 \\ -2 \end{bmatrix}$$

Paso 4: *Hallamos* d_y *resolviendo el sistema lineal*

$$\begin{aligned} (AZ^{-1}XA^t)dy &= b - \mu AZ^{-1}e - AZ^{-1}Xd_D \\ dy &= -2{,}1 \end{aligned}$$

Paso 5: *Calcular*

$$\begin{aligned} dz &= -d_D - A^t dy \\ &= -\begin{bmatrix} -1 \\ -2 \end{bmatrix} - \begin{bmatrix} 4 \\ 2 \end{bmatrix} (-2{,}1) \\ &= \begin{bmatrix} 9{,}4 \\ 6{,}2 \end{bmatrix} \end{aligned}$$

Paso 6: *Hallar dx resolviendo el sistema lineal*

$$\begin{aligned} Zdx &= \mu^0 e - XZe - Xdz \\ dx &= \begin{bmatrix} -0{,}4 \\ 2{,}8 \end{bmatrix} \end{aligned}$$

Paso 7: *Calcular*

$$\alpha_P = \min_{1 \le i \le n} \left\{ -\frac{x_i^0}{dx_i} : \ dx_i < 0 \right\} = 2{,}4$$

Paso 8: *Calcular*

$$\alpha_D = \min_{1 \le i \le n} \left\{ -\frac{z_i^0}{dz_i} : \ dz_i < 0 \right\} = 1$$

Paso 9: *Calcular* $\alpha = \min\{\alpha_P, \alpha_D\} = \{2{,}4, 1\} = 1$

Paso 9: *Hacer*

$$\begin{aligned} x^1 &= \begin{bmatrix} 1 \\ 1 \end{bmatrix} + (0{,}98)(1) \begin{bmatrix} -0{,}4 \\ 2{,}8 \end{bmatrix} = \begin{bmatrix} 0{,}608 \\ 3{,}144 \end{bmatrix} \\ y^1 &= [0] + (0{,}98)\,(1)\,(-2{,}1) = -2{,}058 \\ z^1 &= \begin{bmatrix} 1 \\ 1 \end{bmatrix} + (0{,}98)\,(1) \begin{bmatrix} 9{,}4 \\ 6{,}2 \end{bmatrix} = \begin{bmatrix} 10{,}212 \\ 7{,}076 \end{bmatrix} \end{aligned}$$

Paso 10: *Hacer*

$$\mu^1 = \frac{|c^t x^1 - b^t y^1|}{n^2} = 8{,}257$$

Volver al paso 1*, ahora comenzando con los puntos iniciales*

$$x^1 = \begin{bmatrix} 0{,}608 \\ 3{,}144 \end{bmatrix}, \qquad z^1 = \begin{bmatrix} 10{,}212 \\ 7{,}076 \end{bmatrix} \qquad y \qquad y^1 = -2{,}058$$

Observe que, después de la primera iteración:

$$\frac{|c^t x^1 - b^t y^1|}{1 + |b^t y^1|} = 0{,}3392$$

Después de 25 *iteraciones, la solución óptima aproximada del problema original está dada por el vector*

$$\begin{bmatrix} 2{,}4999 \\ 0{,}0000 \end{bmatrix}$$

3.4.2. Eficiencia Teórica del Algoritmo de Puntos Interiores Primal-Dual

Dentro de la gran variedad de algoritmos de puntos interiores, el Algoritmo Primal-Dual es el que presenta más ventajas, debido a su eficiencia y robustez, pero sobretodo a la facilidad de su implementación computacional.

Con una implementación numérica adecuada, el Algoritmo de Puntos Interiores Primal-Dual requiere hasta $O(nL)$ iteraciones para otorgar una buena aproximación a la solución óptima. Esta propiedad, al igual que en el Método de Karmarkar, le otorga la característica de algoritmo polinomial.

Existen muchas versiones del Método de Puntos Interiores Primal-Dual. Se eligió aquella expuesta en [5], por cuestiones didácticas.

3.4.3. Implementación del Algoritmo de Puntos Interiores Primal-Dual

En esta sección vamos a implementar computacionalmente el algoritmo de puntos interiores Primal-Dual. Específicamente, vamos a crear el código en MatLab y lo probaremos con algunas instancias de programación lineal. Para

tal objetivo, buscaremos algunos modelos clásicos de Programación Lineal y generaremos, mediante software, instancias de tamaños inmensos, entre 100 y 1000 variables y un número similar de restricciones, con las cuales verificaremos su desempeño.

```
function [v,x,iter]=PRIMALDUAL(A,b,c)
[m,n]=size(A);
e=ones(n,1);
mu=10;
iter=0;
x=ones(n,1);
y=zeros(m,1);
z=ones(n,1);
while abs(c'*x-b'*y) / (1+abs(b'*y)) > 0.000001
    iter=iter+1;
    X=diag(x);
    Z=diag(z);
    dD=A'*y+z-c;
    dy=inv(A*inv(Z)*X*A')*(b-mu*A*inv(Z)*e-A*inv(Z)*X*dD);
    dz=-dD-A'*dy;
    dx=inv(Z)*(mu*e-X*Z*e-X*dz);
    aP=1; aD=1;
    for i=1:n
        if dx(i)<0
            minimo=-x(i)/dx(i);
                if minimo<aP
                    aP=minimo;
                end
        end
        if dz(i)<0
            minimo=-z(i)/dz(i);
            if minimo<aD
                aD=minimo;
            end
```

```
        end
    end
    x=x+0.98*aP*dx;
    y=y+0.98*aD*dy;
    z=z+0.98*aD*dz;
    mu=abs(c'*x-y'*b)/n^2;
end
v=c'*x;
```

Para ejecutar el programa en la ventana de comando de MatLab, basta llamar a la función del siguiente modo

[v,x,iter]=PRIMALDUAL(A,b,c)

Observe que previamente se tiene que ingresar en la ventana de comandos del MatLab la matríz A, y los vectores b y c.

Capítulo 4

Comparación del Método Simplex y el Método de Puntos Interiores Primal-Dual para Programación Lineal

En este capítulo veremos las ventajas y desventajas que el Algoritmo de Puntos Interiores Primal-Dual (PI) tiene con respecto al Algoritmo Simplex. El capítulo fue dividido en tres secciones, en la primera vamos a comprobar en la práctica el desempeño de ambos frente a instancias del problema de programación lineal de diferentes tamaños. En la segunda parte, haremos un estudio de aspectos más subjetivos que estos métodos poseen, específicamente, nos referimos al análisis de sensibilidad. Finalmente, en la tercera sección, daremos un panorama sobre la implementación computacional, tipo de soluciones óptimas que estos métodos generan, comportamiento frente a ciclos (degeneración) y mal condicionamiento numérico.

Desde el punto de vista teórico, es claro que el algoritmo de Puntos Interiores tiene un mejor desempeño a medida que el tamaño del ejemplar de programación lineal aumente, pues el número de iteraciones es aproximadamente del orden $O(nL)$ para cualquier instancia del problema de programación lineal, esto incluye obviamente al peor caso. Como ya se comentó,

esto lo torna aparentemente un método bastante atractivo en la resolución del problema de programación lineal.

Por otro lado, el desempeño del algoritmo Simplex no es bueno en teoría, debido a que el número de operaciones elementales es del orden $O\left(2^n L\right)$, en el peor de los casos.[1] Una de las peores instancias con que puede encontrarse el Simplex en la resolución del problema de programación lineal, es justamente con instancias del tipo Klee y Minty (1972), las cuales obedecen al siguiente modelo matemático:

$$\begin{aligned} &\text{Minimizar } x_n \\ &0 \leq x_1 \leq 1 \\ &\varepsilon x_{j-1} \leq x_j \leq 1 - \varepsilon x_{j-1}, \quad j = 2, ..., n \\ &x_j \geq 0, \quad j = 1, ..., n \end{aligned} \tag{4.1}$$

donde ε es algún número racional en el intervalo $\left\langle 0, \frac{1}{2} \right\rangle$. O en su versión alternativa:

$$\begin{aligned} &\text{Maximizar } \sum_{j=1}^{n} y_j \\ &y_1 \leq 1 \\ &y_j + 2\sum_{k=1}^{j-1} y_k \leq \theta^{j-1}, \quad j = 2, 3, ..., n \\ &y_1, y_2, ..., y_n \geq 0 \end{aligned}$$

donde $\theta = \frac{1}{2}$, ε es algún número racional en el intervalo $\left[0, \frac{1}{2}\right]$. Estas instancias tienen $m = n$ restricciones de igualdad y $2n$ variables no negativas, para el cual el Algoritmo Simplex puede realizar $2^n - 1$ iteraciones. Es decir, puede visitar todos los vértices antes de llegar a la solución óptima.

Desde el punto de vista teórico, esta situación es mala para un algoritmo,

[1] El término $O\left(2^n L\right)$ fue visto en la parte de eficiencia de algoritmos, en el segundo capítulo.

debido a que un crecimiento exponencial en el número de iteraciones es fatal sin importar cuán bueno sea el sistema de cómputo utilizado. Para tener una idea clara, la figura 4.1 ilustra el comportamiento de una curva del tipo lineal y una exponencial.

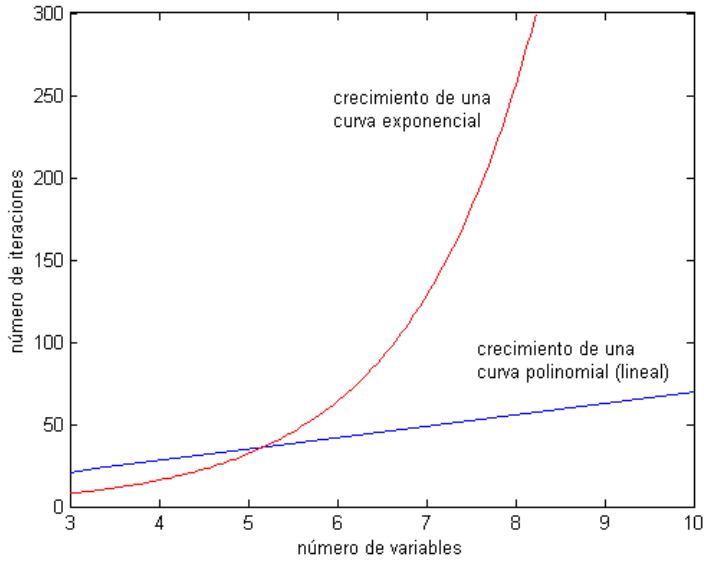

Figura 4.1:

Afortunadamente, el algoritmo Simplex no siempre está relacionado con el peor caso, y en la práctica es bastante efectivo para resolver una gran cantidad de problemas (instancias) reales. Más aún, una de las principales virtudes del Método Simplex es, sin lugar a dudas, el análisis de sensibilidad que se puede realizar una vez resuelto un determinado problema.

Todo esto torna interesante estudiar y desplegar las ventajas y desventajas de estos dos métodos.

4.1. Comparación desde el Punto de Vista Práctico entre el Algoritmo de Puntos Interiores Primal-Dual y el Algoritmo Simplex

En esta etapa vamos a generar instancias de problemas de diferentes dimensiones mediante un programa, donde la región factible es no vacía. Esto conseguiremos mediante la siguiente estrategia.

1. Generamos un vector arbitrario $\hat{x} \in \mathbb{R}^n$ cuyas componentes son todas no negativas
2. Generamos una matriz $A \in \mathbb{R}^{m \times n}$ de rango completo
3. Calculamos el vector $b \in \mathbb{R}^m$, del siguiente modo:

$$b = A\hat{x}$$

4. Finalmente, generamos un vector arbitrario $c \in \mathbb{R}^n$.

Con esto, construimos la siguiente instancia de programación lineal:

$$\begin{aligned} &\text{Minimizar } c^t x \\ &\text{Sujeto a:} \\ &\quad Ax = b \\ &\quad x \geq 0 \end{aligned} \tag{4.2}$$

Observe que el problema (4.2) siempre tiene región factible no vacía, pues

$$A\hat{x} = b \qquad \text{y} \qquad \hat{x} \geq 0$$

El programa en MatLab que realiza esta tarea es el siguiente:

```
function [A,b,c]=generador(m,n)
x=round(100*rand(n,1));
```

```
A=round( 100*rand(m,n)-100*rand(m,n) );
b=A*x;
c=round(100*rand(n,1)-100*rand(n,1));
```

Hay que aclarar que esta función (programa) puede no generar una matriz A de rango completo, pero debido a que las $m \times n$ componentes son generadas de forma aleatoria, todas entre -100 y $+100$, la probabilidad que esto ocurra es remota. No obstante, para ser más rigurosos, una instrucción que verifique si A es realmente de rango completo podría ser insertada (`rank(A)`).

Ejemplo 4.1 *Usando el anterior programa, vamos a generar una instancia de $m = 5$ restricciones y $n = 7$ variables.*

Después de ejecutar el programa, obtuvimos:

$$A = \begin{bmatrix} -35 & 28 & 36 & 30 & 23 & -2 & 42 \\ -23 & 37 & -23 & -36 & -22 & -5 & 29 \\ 30 & 94 & -25 & -27 & 13 & 51 & 48 \\ -41 & 50 & -10 & 53 & -24 & 63 & -27 \\ -12 & 7 & 8 & -31 & -4 & -6 & 48 \end{bmatrix}$$

$$b = \begin{bmatrix} 2024 \\ -7171 \\ 3130 \\ 4292 \\ -3535 \end{bmatrix} \qquad c = \begin{bmatrix} 76 \\ -12 \\ 21 \\ -9 \\ 45 \\ -26 \\ 58 \end{bmatrix}$$

Seguidamente, aplicamos el programa PRIMALDUAL sobre esta instancia y

obtenemos la aproximación a una solución óptima:

$$x^*_{PIPD} = \begin{bmatrix} 85{,}6834 \\ 0{,}0000 \\ 62{,}2414 \\ 78{,}8418 \\ 24{,}8158 \\ 76{,}8952 \\ 0{,}0000 \end{bmatrix}$$

cuyo valor óptimo de la función objetivo fue 6226,8699*, con* 12 *iteraciones. Por otro lado, al aplicar el algoritmo Simplex (usando la estrategia M-grande), en un programa implementado también en MatLab, obtuvimos:*

$$x^*_{Simplex} = \begin{bmatrix} 85{,}6834 \\ 0{,}0000 \\ 62{,}2414 \\ 78{,}8418 \\ 24{,}8158 \\ 76{,}8952 \\ 0{,}0000 \end{bmatrix}$$

con valor objetivo 6226,8692*, con* 9 *iteraciones.*

En el ejemplo 4.1 vimos cómo es posible utilizar el anterior programa y generar instancias de programación lineal de cualquier dimensión, de un modo análogo generaremos problema de grandes dimensiones para comparar ambos métodos con respecto al número de iteraciones y el tiempo de ejecución.

4.1.1. Caso 1

Aquí vamos a generar un problema aleatorio, pero con región no vacía, de 25 restricciones y 40 variables.

Después de ejecutar el programa PRIMALDUAL, obtuvimos como valor objetivo −160084,8648 en 27 iteraciones y en 0,063seg.

Por otro lado, al aplicar el programa SIMPLEXM, obtuvimos como valor objetivo −160084,8782 en 51 iteraciones y en 0,156seg.

Observamos que en ambos casos, los valores objetivos son casi iguales. Esto nos lleva a concluir que en este caso, ambos métodos trabajaron con éxito sobre el problema. Ambos tardaron poco tiempo para resolver la instancia, pero el primal-dual lo hizo más rápido y en menos iteraciones.

4.1.2. Caso 2

Aquí vamos a generar un problema aleatorio de 100 restricciones y 130 variables.

Después de ejecutar el programa PRIMALDUAL, obtuvimos como valor objetivo −61764,0177 en 24 iteraciones y en 0,8599seg.

Por otro lado, al aplicar el programa SIMPLEXM, obtuvimos como valor objetivo −61764,0323 en 210 iteraciones y en 5,39seg.

Observe que en ambos casos, los valores objetivos son similares. Esto nos lleva a concluir que ambos métodos trabajaron con éxito sobre esta instancia. Sin embargo, notamos una diferencia marcada en el tiempo de ejecución y el número de iteraciones, el primal-dual lo hizo seis veces más rápido con respecto al tiempo. Esto se esperaba debido a la naturaleza de los mismos.

4.1.3. Caso 3

Aquí generamos un problema aleatorio de 170 restricciones y 225 variables.

Después de ejecutar el programa PRIMALDUAL, obtuvimos como valor objetivo −237362,6643 en 37 iteraciones y en 6,234seg.

Por otro lado, al aplicar el programa SIMPLEXM, obtuvimos como valor objetivo $-237362{,}7607$ en 467 iteraciones y en 63,3910seg.

Nuevamente, los valores objetivos son también similares. Esto nos lleva a concluir que ambos métodos trabajaron con éxito sobre el problema.

4.1.4. Caso 4

Aquí vamos a generar un problema aleatorio de 250 restricciones y 300 variables.

Después de ejecutar el programa PRIMALDUAL, obtuvimos como valor objetivo $-130425{,}6605$ en 29 iteraciones y en 12,172seg.

Por otro lado, al aplicar el programa SIMPLEXM, obtuvimos como valor objetivo $-130425{,}6676$ en 704 iteraciones y en 470,437seg.

Observemos que en ambos casos, los valores objetivos son también muy similares. Esto nos lleva a concluir que ambos métodos trabajaron con éxito sobre el problema. Nuevamente, notamos una diferencia cada vez mayor en el desempeño de los mismos, pues el SIMPLEX tomó más de 7 minutos, en cambio el primal-dual tomó tan solo un poco más de 12 segundos.

4.1.5. Caso 5

Aquí vamos a generar un problema aleatorio de 500 restricciones y 678 variables.

Despues de ejecutar el programa PRIMALDUAL, obtuvimos como valor objetivo $-516995{,}1245$ en 66 iteraciones y en 266,922seg.

Por otro lado, al aplicar el programa SIMPLEXM, éste no se detuvo inclusive después de 3600seg.

Observación 4.1 *De acuerdo a los casos analizados, cabe señalar que el Algoritmo Simplex es relativamente tan rápido como el Algoritmos de Puntos Interiores Primal-Dual, para instancias de aproximadamente* 100 *variables. Solamente en problemas realmente grandes, alrededor de* 300 *variables o más, el Algoritmo de Puntos Interiores Primal-Dual es significativamente más rápido. Debemos recalcar que no estamos interesados en este trabajo por el tipo de implementación, sino, simplemente estamos comparando el desempeño de ambos algoritmos usando un mismo recurso, en este caso, el MatLab. Estamos seguros que si los algoritmos que intervienen en estos experimentos son implementados en otro lenguaje de programación, por ejemplo C++, los tiempos de ejecución disminuirían significativamente, pero permanecerían proporcionales.*

A continuación, vamos a resumir en una tabla los resultados de los cinco casos tratados, esto nos ayudará a visualizar mejor la comparación desde un punto de vista práctico, en lo que refiere al número de variables y restricciones, el tiempo y número de iteraciones. Observe que:

- m denotará el número de restricciones
- n denotará el número de variables
- v_p denotará el valor objetivo obtenido por el algoritmo de puntos interiores primal-dual
- v_s denotará el valor objetivo obtenido por el algoritmo Simplex
- k denota el número de iteraciones
- t el tiempo transcurrido en segundos

Tabla comparativa del desempeño de los Algoritmos Simplex y de Puntos Interiores Primal-Dual

	Dimensión		Puntos Interiores			Simplex		
Caso	m	n	v_p	k_p	t_p	v_s	k_s	t_s
1	25	40	−160084,86	27	0,0630	−160084,88	51	0,156
2	100	130	−61764,02	24	0,8599	−61764,03	210	5,390
3	170	225	−237362,66	36	6,2340	−237362,76	467	63,391
4	250	300	−130425,66	29	12,172	−130425,67	704	470,43
5	500	678	−516995,12	66	266,92	–	–	Más de 3600

En la tabla anterior podemos ver que en todos los casos resueltos por ambos algoritmos, las soluciones óptimas obtenidas prácticamente coinciden, esto indica que ellos llegaron a las soluciones óptimas. Por otro lado, el número de iteraciones del Método de Puntos Interiores Primal Dual no crece mucho cuando el número de restricciones y variables aumentan. En contraste, el Simplex no tiene esa propiedad y vemos que el número de iteraciones aumentan de un modo preocupante. Finalmente, es indiscutible que el tiempo de ejecución del Algoritmo de Puntos Interiores Primal Dual es muy inferior que el tomado por el Simplex. Más aún, en el caso 5, el Simplex no llegó a la solución en los primeros 3600 segundos, por lo que tuvimos que detener el programa.

4.2. Análisis de las Ventajas y Desventajas Cualitativas del Método de Puntos Interiores Primal-Dual y el Método Simplex: Análisis de Sensibilidad

Entendemos por análisis de sensibilidad en programación lineal, al estudio de la reacción del modelo matemático cuando éste es sometido a modifica-

ciones tanto en el vector de disponibilidades b, el vector de costos c o la matríz de coeficientes tecnológicos A. Además, en este rubro puede ser considerado también la interpretación que hacemos a las variables duales, o comunmente denominado en economía *precio sombra*. Este análisis es realizado frecuentemente una vez que el problema original ya fue resuelto.

Así, a partir de una solución de un Problema de Programación Lineal es posible extraer información importante sobre sus sensibilidades. Esto por lo general evita algunas veces volver a resolver el problema completamente, teniéndose apenas que realizar pequeñas modificaciones a partir de la solución óptima. En esta sección vamos a tocar los puntos más importantes del Análisis de Sensibilidad en programación lineal y veremos su interpretación a través de los métodos que hemos analizado, Simplex y de Puntos Interiores. Debemos aclarar que ya existe una teoría desarrollada sobre sensibilidad basada en una solución básica factible óptima, la cual es otorgada por el método Simplex. La intención es ahora analizar si algo análogo es posible realizar cuando usamos el método de puntos interiores primal-dual.

En este trabajo realizaremos el análisis de sensibilidad para el precio sombra, cuando existen cambios en el vector de costos c y cambios en el vector de disponibilidades b, el análisis de la repercución de los cambios en la matriz A no serán incluidos en este trabajo, debido a que por lo general, esto requiere una nueva resolución del problema.

4.2.1. Precio Sombra

Un abordaje clásico nos dice que, una vez resuelto el problema de programación lineal mediante el algoritmo Simplex, es posible saber si alteraciones en los coeficientes de disponibilidad (vector b) influyen de manera importantes en en valor objetivo óptimo, y de qué manera influyen. Esto tiene importancia en ciencias, economía e ingeniería, debido a que permite elegir una política alternativa que puede prevenir ganancias o pérdidas.

Inicialmente, veremos el papel que desempeña el Simplex en la determinación de los precios sombra. Luego, veremos cómo podemos utilizar el

Algoritmo de Puntos Interiores para este mismo fin.

Precio Sombra en el Método Simplex

Sea B^* la base óptima, sabemos que el vector básico óptimo y el valor objetivo óptimo pueden ser expresados por:

$$\begin{aligned} x_B^* &= (B^*)^{-1} b \\ z^* &= c_B^t x_B^* \end{aligned}$$

Considere un cambio marginal[2] Δb en el vector de disponibilidades b:

$$b^* \longleftarrow b^* + \Delta b$$

Este cambio ocasiona cambios en la solución óptima y en el valor objetivo óptimo:

$$\begin{aligned} \Delta x_B &= (B^*)^{-1} \Delta b \\ \Delta z &= c_B^t \Delta x_B \end{aligned}$$

Combinando las ecuaciones anteriores, tenemos:

$$\Delta z = c_B^t \Delta x_B = c_B^t (B^*)^{-1} \Delta b$$

Definiendo el vector fila w^* mediante la siguiente regla de correspondencia:

$$w^* = c_B^t (B^*)^{-1}$$

la ecuación anterior se convierte en:

$$\Delta z = w^* \Delta b$$

[2]Es decir, un cambio que no modifica la base.

Observe que, si consideramos una sola i-componente del vector fila w^*, tendríamos la siguiente ecuación:

$$w_i^* = \frac{\Delta z}{\Delta b_i}$$

lo cual indica que w_i^* proporciona el cambio en el valor óptimo de la función objetivo como resultado de un cambio marginal en la i-componente del vector de términos independientes b. Estos parámetros de sensibilidad juegan un papel fundamental en aplicaciones de ingeniería y ciencias. Como se verá en las secciones siguientes, los parámetros de sensibilidad son de hecho variables duales.

Otro enfoque basado en derivadas es el siguiente. Como

$$z = c_B^t \left(B^*\right)^{-1} b$$

entonces

$$\frac{\partial z}{\partial b} = c_B^t \left(B^*\right)^{-1} = w^* \tag{4.3}$$

Esdecir,

$$\frac{\partial z}{\partial b} = w^* \qquad \Longrightarrow \qquad \frac{\partial z}{\partial b_i} = w_i^*$$

Así por ejemplo, si $\frac{\partial z}{\partial b_i} = 3$, entonces por cada unidad que se incremente b_i, la función objetivo de incrementará en 3 unidades. Por otro lado, $\frac{\partial z}{\partial b_i} = -3$, entonces por cada unidad que se incremente b_i, la función objetivo disminuirá en 3 unidades.

Por tal motivo, las variables w_i son conocidas por *precios sombra* y son de gran utilidad en ciencias e ingeniería. Sin lugar a dudas, los precios sombra otorgados por el Método Simplex tienen importancia y, en adición a la solución óptima, conforman un aspecto positivo presentado por este método.

Por lo tanto, resulta inmediato obtener mediante el Simplex estos valores, basta imprimir $c_B^t \left(B^*\right)^{-1}$ al final del procedimiento de la resolución del problema. Esto nos permitirá conocer los valores del precio sombra.

Precio Sombra en el Método de Puntos Interiores Primal Dual

Inicialmente, por el razonamiento anterior y debido a que el algoritmo de puntos interiores primal-dual no trabaja con soluciones básicas factibles, parece complicado obtener los precios sombra en este caso, pues no disponemos de una base en cada iteración.

Pero, si observamos la ecuación (4.3), veremos que en verdad se tratan de las variables duales. Debido a que el método de puntos interiores primal-dual resuelve ambos problemas simultáneamente, primal y dual, no es difícil modificar el algoritmo para que en la última iteración, cuando está en condiciones de optimalidad, imprima estos valores. Si observamos el Algoritmo 3.2, en el paso 9 disponemos de las variables duales, basta entonces imprimir el vector y para disponer de los precios sombra.

Ejemplo 4.2 *Considere el siguiente problema*

$$\begin{aligned} &\textit{Minimizar } -x_1 - 3x_2 \\ &\textit{Sujeto a:} \\ &\quad x_1 + 2x_2 + x_3 = 10 \\ &\quad 2x_1 + x_2 + x_4 = 20 \\ &\quad x_1, x_2, x_3, x_4 \geq 0 \end{aligned}$$

Despues de aplicar el algoritmo Simplex, obtuvimos como solución óptima

$$x^*_{Simplex} = \begin{bmatrix} 0 \\ 5 \\ 0 \\ 15 \end{bmatrix}, \qquad z^*_{simplex} = -15 \qquad y \qquad w^*_{Simplex} = \begin{bmatrix} -1{,}5 \\ 0 \end{bmatrix}$$

Así, $w^*_1 = -1{,}5$ *y* $w^*_2 = 0$. *Esto significa que, por cada unidad que se aumente sobre* $b_1 = 10$, *entonces la función objetivo debería decrecer en una proporción a* $1{,}5$. *Por otro lado, incrementos o decrecimientos sobre* $b_2 = 20$ *no deberían modificar el valor objetivo óptimo, debido a que* $w^*_2 = 0$.

Después de aplicar el algorimo de puntos interiores primal-dual, obtuvi-

mos también los mismos resultados:

$$x^*_{Puntos\ interiores} = \begin{bmatrix} 0 \\ 5 \\ 0 \\ 15 \end{bmatrix}, \quad z^*_{Puntos\ interiores} = -15 \quad y \quad w^*_{Puntos\ interiores} = \begin{bmatrix} -1{,}5 \\ 0 \end{bmatrix}$$

Una vez que se decidió por alguna política después de realizar el análisis en el precio sombra, ya sea de aumentar algunos valores en el vector de disponibilidades, el siguiente paso consiste en realizar un análisis de sensibilidad en el vector b, lo cual se verá más adelante.

4.2.2. Cambios en el vector de costos c

Supongamos que el vector c sufre una modificación en una o varias de sus componentes, así, el nuevo vector sería:

$$c \leftarrow c + \Delta c$$

donde Δc es justamente el vector de variación. Analicemos cómo afecta este cambio a la solución óptima actual.

Cambios en el vector de costos c utilizando el Simplex, un abordaje clásico

Sea B^* la base actual óptima generada por el algoritmo Simplex. Consideremos las condiciones de optimalidad:

$$z_j - c_j = c_B^t \left(B^*\right)^{-1} a_j - c_j$$

Supongamos que modificamos la i componente del vector c. Al reemplazar el nuevo vector c, obtendremos dos casos:

Si c_i es básico: En este caso, el vector c_B debería ser alterado:

$$\begin{aligned} z_j - c_j &= c_B^t \left(B^*\right)^{-1} a_j - c_j \\ &= \left[\begin{array}{ccccccc} c_{B_1} & c_{B_2} & \cdots & c_i + \Delta c_i & \cdots & c_{B_m} \end{array} \right] \left(B^*\right)^{-1} a_j - c_j \\ &= \left(\left[\begin{array}{ccccccc} c_{B_1} & c_{B_2} & \cdots & c_i & \cdots & c_{B_m} \end{array} \right] + \right. \\ &\quad \left. + \left[\begin{array}{ccccccc} 0 & 0 & \cdots & \Delta c_i & \cdots & 0) \end{array} \right] \right) \left(B^*\right)^{-1} a_j - c_j \\ &= c_B^t \left(B^*\right)^{-1} a_j + \Delta c_i \left(\left(B^*\right)^{-1} a_j\right)_i - c_j \end{aligned}$$

Luego, si

$$\Delta c_i \left(\left(B^*\right)^{-1} a_j\right)_i \leq c_j - c_B^t \left(B^*\right)^{-1} a_j$$

la base B^* aún es óptima. Caso contrario, las condiciones de optimalidad son violadas y se procede a reiniciar el algoritmo simplex hasta que el criterio de optimalidad sea satisfecho. Si se usa el tablero simplex, esto se resume a continuar iterando hasta que los costos reducidos sean no positivos.

Si c_i es no básico: En este caso, vemos que:

$$\begin{aligned} z_i - c_i &= c_B^t \left(B^*\right)^{-1} a_i - (c_i + \Delta c_i) \\ &= c_B^t \left(B^*\right)^{-1} a_i - c_i - \Delta c_i \end{aligned}$$

Luego, si

$$\Delta c_i \geq c_B^t \left(B^*\right)^{-1} a_i - c_i$$

entonces la base B^* aún es óptima. Caso contrario, la factibilidad es alterada y se debería continuar realizando operaciones simplex hasta satisfacer las condiciones óptimas, es decir, que los nuevos costos reducidos sean no positivos.

Cambios en el vector de costos c utilizando el Algoritmo de Puntos Interiores Primal-Dual

En contraste con el Simplex, el algoritmo de puntos interiores primal-dual no utiliza un criterio de optimalidad basado en una base factible, sino que reconoce una solución óptima cuando:

$$\frac{|c^t x - b^t y|}{1 + |b^t y|} < \varepsilon$$

Por lo tanto, un cambio en el vector c probablemente altere esta desigualdad. Cuando esto ocurra, podemos simplemente utilizar las soluciones óptimas actuales y considerarlas como puntos iniciales para las siguientes iteraciones, hasta alcanzar la optimalidad deseada.

4.2.3. Cambios en el Vector de Disponibilidades b

Una vez que un análisis en los precios sombra fue realizado y una nueva decisión de disponibilidades fue tomada, resta reemplazar los nuevos datos en el vector b. Esto, como es de esperarse, trae consigo efectos que pueden o no afectar la optimalidad.

Cambios en el vector b, un abordaje mediante el Método Simplex

Supongamos ahora que el vector b es reemplazado por el vector b', entonces $B^{-1}b$ será reemplazado por $B^{-1}b'$. Desde que los costos reducidos $z_j - c_j \leq 0$ para todas las variables no básicas, la única posibilidad de violar la optimalidad es mediante la infactibilidad. En efecto, esto se debe a que las variables básicas son calculadas mediante:

$$x_B = B^{-1}b$$

y los cambios en el vector b podrían provocar que x_B tenga algunas componentes negativas. En estos casos, se debería aplicar el Método Dual-Simplex

para restaurar la factibilidad, una vez hecho esto, se debería proceder a continuar aplicando el Simplex hasta que la optimalidad sea alcanzada.

Cambios en el vector b, un abordaje desde el Método de Puntos Interiores Primal-Dual

La única expresión que mide la optimalidad en el Método de Puntos Interiores Prima-Dual es la expresión:

$$\frac{|c^t x - b^t y|}{1 + |b^t y|} < \varepsilon$$

donde x y y son los vectores factibles del primal y dual, recpectivamente. Por lo tanto, dados los vectores óptimos x^* y y^* del primal y dual, respectivamente. Entonces, si un cambio en el vector b altera la expresión $\frac{|c^t x - b^t y|}{1+|b^t y|}$ tal que ésta es mayor o igual a ε, entonces deberíamos tomar x^* y y^* como los nuevos puntos iniciales y resolver el nuevo problema hasta la optimalidad sea alcanzada. Caso contrario, si las alteraciones en b no afectan la optimalidad, ningún procedimiento adicional debería ser realizado y los vectores x^* y y^* continuarían siendo óptimos.

4.3. Panorama sobre la Implementación Computacional, Tipos de Soluciones Óptimas, Degeneración y Mal Condicionamiento Numérico

En esta sección complementamos el trabajo haciendo una breve descripción de otros aspectos relacionados a los métodos tratados aquí, ellos son importantes, debido a que también influyen en cierta magnitud el desempeño de los mismos.

4.3.1. Implementación Computacional

Es claro que existen varios programas comerciales que ejecutan el Simplex, debido a su popularidad durante muchos años. Sin embargo, es posible, sin mucho esfuerzo, programar a nivel experimental el Simplex usando por ejemplo MatLab, tal como lo hicimos en este tranajo. Algo que no hay que desmerecer, es que el Simplex inclusive puede ejecutarse usando lo que se denomina el Tableado Simplex, este recurso tiene mucha importancia didáctica cuando se imparte en estudiantes de pregrado.

Por otro lado, el Método de Puntos Interiores Primal-Dual actualmente es casi desconocido en nuestro medio. Sin embargo, llevar a computador este algoritmo resulta inclusive más práctico que el Simplex, debido que requiere menos código.

La gran diferencia la determina el punto inicial. Mientras que en el Simplex se requiere una base factible para iniciar el algoritmo, en el método de puntos interiores es posible inicializarlo con vectores casi del todo arbitrarios.

4.3.2. Tipo de Soluciones Óptimas

Sabemos que el Simplex, cuando existe solución óptima, éste otorga una solución óptima que es justamente un punto extremo (SBF). Esta propiedad resulta conveniente en muchos procesos, pero puede ser considerada en otros casos muy radical. Esto se aprecia en algunos trabajos recientes relacionados a la medicina[3], donde no es recomendable tomar los extremos y sí alguna solución óptima alternativa.

En contraste, la naturaleza de los métodos de puntos interiores permite construir sucesiones de puntos que se aproximan a la solución óptima, la cual no necesariamente es un punto extremo, pero sí óptima. Esta flexibilidad los torna, en algunos casos, más apropiados que el Simplex para ciertos procesos recientes. Cuando no hay soluciones alternativas, la sucesión necesariamente converge a un punto extremo óptimo, otorgando apenas una aproximación

[3]Vea por ejemplo: http://pages.cs.wisc.edu/~arinbjor/papers/lppaper.pdf

a ésta. En este sentido, podemos decir que el Simplex tiene una pequeña ventaja debido a que otorga, salvo por el error del computador, una solución exacta.

4.3.3. Degeneración

Una solución básica en un problema de programación lineal con n variables, es un punto que se encuentra sobre al menos n hiperplanos linealmente independientes. Cuando algún hiperplano adicional a esos n pasa por dicho punto, estamos frente a una solución básica degenerada. Cuando el Simplex es ejecutado sobre un problema que contenga alguna solución básica factible degenerada, y en alguna iteración este punto es analizado por el algoritmo, puede ocurrir que se quede estacionado, $x^{k+1} = x^k$, debido a que el algoritmo original no puede reconocer este fenómeno. Decimos entonces que ocurrió ciclaje en el algoritmo.

Sabemos que el ciclaje es una debilidad en el Método Simplex clásico, a pesar que existen estrategias para tratar con este problema, tales como la regla de Bland y la regla Lexicográfica, los cuales constas en la literatura. Por lo general, estas estrategias que previenen ciclaje resultan muy caras computacionalmente y algunos softwares comerciales inclusive no lo aplican.

Por otro lado, los métodos de puntos interiores no presentan este defecto, debido a que la sucesión de puntos creada por las iteraciones de estos métodos, se desplazan por el interior relativo de la región factible. Este último hecho lo torna muy ventajoso con respecto al simplex.

4.3.4. Inestabilidad Numérica

Es claro que en ambos algoritmos existe la posibilidad de mal condicionamiento sobre todo en la matriz A, es decir, que algunos términos pueden ser extremadamente grandes y otros demasiados pequeños en valor absoluto. Este fenómeno puede ocasionar que en las iteraciones sucesivas se acumule un error que perjudique el resultado final, con considerables consecuencias. En

este trabajo, no se consideraron correcciones para este fin. Sin embargo, es posible analizar los errores que podrían cometer los programas y en algunos casos, deberían emplearse técnicas de escalamiento previos a la resolución del problema.

Conclusiones Finales

1. En teoría y también en la práctica, el Método de Puntos Interiores Primal-Dual es más eficiente que el Método Simplex, cuando el tamaño de la instancia de Programación Lineal es grande.

2. No resulta práctico ejecutar el Algoritmo de Puntos Interiores Primal-Dual manualmente, debido a que inclusive para pequeños problemas, el número de iteraciones puede ser considerable. Esto no contradice su eficiencia computacional, debido a que ella se mide cuando es el algoritmo es sometido a instancias de gran tamaño.

3. Para instancias de programación lineal, cuyo tamaño es mediano, alrededor de 100 variables, el Simplex tiene un desempeño razonable.

4. El número de iteraciones que el Simplex realiza, crece en un orden exponencial con respecto al número de variables.

5. El número de iteraciones requeridas por el Algotimo de Puntos Interiores Primal-Dual crece polinomialmente con respecto al número de variables. Esto lo torna más adecuado que el Simplex para problemas inmensos.

6. Es posible realizar de un modo bastante fácil el análisis de sensibilidad elemental usando el Método de Puntos Interiores Primal-Dual. El proceso de recuperación de la optimalidad se resume a la reutilización de los puntos óptimos anteriores, previos a la alteración de los datos. Esto definitivamente resultará práctico para aplicar este método en diferentes áreas de estudio.

7. El algoritmo de Puntos Interiores Primal-Dual no necesita una base inicial como el Simplex, por lo que se puede prescindir de utilizar variables artificiales.

Trabajos Futuros y Recomendaciones

Los programas realizados en este trabajo, los cuales ejecutan el Método de Puntos Interiores Primal Dual, tienen un carácter experimental. No obstante, ellos pueden ser implementados con vías a crear programas comerciales eficientes y que realicen análisis de sensibilidad según las propuestas hechas en este trabajo.

El estudio de Métodos de Puntos Interiores del tipo Primal-Dual no es un asunto cerrado, debido a que existen hoy en día muchos trabajos que se apoyan en esta estrategia, pero que difieren de uno u otro modo en ciertos puntos, todo con la finalidad de mejorar aún más la eficiencia de algoritmos de esta clase para la resolución del Problema de Programación Lineal. Es de interés del autor, sugerir la investigación futura en esta área.

Bibliografía

[1] Bazaraa M. S., Jarvis J. J., Sherali H. D. *Programación Lineal y Flujo en Redes*. Second Edition. John Wiley & Sons. 1990.

[2] Enrique Castillo, Antonio J. Conejo, Pablo Pedregal, Ricardo García y Natalia Alguacil, *Formulación y Resolución de Modelos de Programación Matemática en Ingeniería y Ciencia*. 2002. (http://www.sectormatematica.cl/libros.htm)

[3] AV Fiacco and CP McCormick, *Nonlinear Programming: Sequential Unconstrained Minimization Techniques* (New York, Wiley, 1968).

[4] Luenberger, David E., *Programación Lineal y No Lineal*. Addison-Wesley Iberoamericano S.A., E.U.A, 1989.

[5] Marsten Roy, Subramanian Rahhika y Saltzman Mattew, Reports, Technological institute of Georgia, 1992.

[6] Prawda Witemberg, Juan. *Métodos y Modelos de Investigación de Operaciones*. Ed. Limusa, México, 1986.

[7] Salazar Gonzales, J., *Programación Matemática*, Ed. Diáz de Santos S.A., Madrid, España, 2001.

[8] Taha H. A. *Investigación de Operaciones, una introducción*. Sexta Edición. Prentice Hall. 1997.

Proyecto de Tesis – Maestría en Matemática

Manuel Arenas Céspedes

1. Preámbulo

Muchos problemas de la vida real, en diversas áreas de la ciencia, pueden ser representados mediante una formulación matemática, esta formulación es conocida como un modelo matemático. La Investigación de Operaciones trata sobre el estudio de estos modelos, sus propiedades y métodos de resolución.

Dentro de la Investigación de Operaciones, el modelo matemático denominado Problema de Programación Lineal, es uno de los más representativos. La importancia se debe no sólo a la gran variedad de problemas que encuadran dentro de este tipo de formulación, sino porque el modelo en sí presenta propiedades favorables que permiten una eficiente resolución y porque puede ser utilizado como un subproblema de métodos sofisticados y más generales.

Desde 1947, fecha en que G. Dantzig publicó formalmente el Algoritmo Simplex, el Problema de Programación Lineal fue aplicado en distintos campos, ya sea el comercial, científico, etc. El Algoritmo Simplex es uno de los más sofisticados e interesantes en la historia de la Investigación de Operaciones y representó uno de los más notorios avances dentro de la comunidad científica. Su éxito sería completo sino fuera opacado por su ineficiencia computacional teórica.

A pesar que el Algoritmo Simplex resuelve eficientemente la mayoría de casos del Problema de Programación Lineal, él puede tomar un tiempo extremadamente grande cuando es sometido a problemas particulares. En efecto, Victor Klee y G. Minty mostraron que existen casos de Programación Lineal donde el Simplex necesita realizar un número exponencial[1] de iteraciones.

En 1978 surgió un método que requería apenas un número polinomial[2] de iteraciones para resolver el Problema de Programación Lineal, lamentablemente, su éxito teórico no acompañó al práctico. Sin embargo, el avance no se detuvo, en los últimos veinte años aparecieron diversos métodos eficientes[3]

[1] Intuitivamente: alrededor de 2^n iteraciones, donde n es el número de variables..

[2] Intuitivamente: alrededor de $p(n)$, donde p es un polinomio y n el número de variables.

[3] Que el número de iteraciones es polinomial, y el polinomio es de grado bajo

y con una estructura totalmente diferente a la del Simplex. Los más representativos son, sin lugar a dudas, el Método de Karmarkar y el Método de Puntos Interiores Primal-Dual.

En este proyecto se analizará y se evaluará, tanto en teoría como en la práctica, dos de los métodos más representativos para la resolución del Problema de Programación Lineal en la actualidad. Específicamente, nos referimos al tradicional Método Simplex y al moderno Método de Puntos Interiores Primal Dual.

2. Planteamiento Teórico

2.1. Problema de Investigación

2.1.1. Enunciado del Problema

Un Estudio Comparativo del Método Simplex y el Método de Puntos Interiores Primal-Dual para Programación Lineal.

2.1.2. Descripción del Problema

Campo de Investigación El campo de investigación está comprendido dentro del área de Matemática, específicamente, dentro de La Investigación de Operaciones. En cuanto al nivel, es un estudio descriptivo-comparativo. En cuanto al tipo, es una investigación documental y experimental.

Análisis de Variables

Independientes: Método Simplex, Método de Puntos Interiores Primal-Dual, indicadores tales como el número de operaciones elementales, número de iteraciones y el tiempo.

Dependientes: Eficiencia Computacional.

Justificación Por muchos años en las universidades locales, dentro de un curso de Investigación de Operaciones, ya sea en una carrera de Matemática, Ingeniería, Economía, entre otras, solamente se ha venido enseñando como método estándar para resolver el Problema de Programación Lineal: el Método Simplex. Pero generalmente no se habla del lado negativo que presenta este método, ni de los métodos alternativos que existen en la actualidad.

Otro aspecto es la implementación computacional, resulta sumamente difícil llevar al computador, en fase experimental, el Algoritmo Simplex, tarea que debería ser obligatoria en la carrera de Ingeniería de Sistemas, por ejemplo.

Sin embargo, en el caso del Método de Puntos Interiores Primal Dual, la situación es totalmente diferente. Pues, además de tener una eficiencia comprobada, en teoría y en la práctica, resulta fácil de implementar y llevar al computador en forma de programa. E inclusive su ejecución manual no requiere mucho esfuerzo.

Este trabajo se justifica, porque contribuye a la difusión del Método de Puntos Interiores Primal-Dual en nuestro medio, con esto, se busca tener disponibles herramientas alternativas para resolver problemas de Programación Lineal, sobre todo cuando éstos tienen dimensiones inmensas.

2.2. Marco Conceptual

El Problema de Programación Lineal es un modelo matemático dentro de la Investigación de Operaciones. Es uno de los problemas que tiene propiedades favorables para su resolución, debido a que es un Problema de Programación Convexa: tanto la función objetivo como la región de factibilidad son convexas. Esta propiedad permite la construcción de métodos, ya sean exactos o iterativos, los cuales aseguran teóricamente su convergencia a una solución óptima.

Otro resultado importante dentro de la teoría de la Programación Lineal es el siguiente: Si existe solución óptima finita, entonces existe una solución óptima que es justamente un punto extremo. El Algoritmo Simplex se vale de esta propiedad y se desplaza a través de los extremos, en forma de solución básica, en busca de la solución óptima, sin necesariamente analizar todos ellas. Esto lo torna un método inteligente, pero existe un inconveniente, una región de factibilidad en programación lineal puede tener una cantidad, aunque finita, inmensamente grande de extremos. La existencia de problemas particulares (V. Klee y G. Minty) donde el Simplex analiza todos los extremos lo torna ineficiente desde el punto de vista teórico.

Los métodos modernos de puntos interiores abordan el problema desde otro punto de vista: se desplazan a través del interior de la región de factibilidad en busca de la solución óptima, si bien el número de iteraciones puede ser aún mayor que las realizadas por el Simplex cuando el problema tiene pocas variables, el desempeño de estos métodos nuevos se nota a medida que

el número de variables del problema se incrementa. El soporte matemático de estos métodos lo conforman el Método de Newton, técnicas Lagrangianas y estrategias de barrera.

La ineficiencia y eficiencia del Simplex y de los métodos modernos, respectivamente, se mide en términos del número de iteraciones requeridas para resolver el problema. El Simplex realiza, en el peor de los casos, un número exponencial de iteraciones, aproximadamente $2^n \cdot L$ iteraciones, donde n es el número de variables y L una constante. Mientras que el mejor método moderno realiza aproximadamente $n \cdot L$ iteraciones, este hecho torna su uso indispensable para problemas con un número grande de variables.

A continuación haremos una breve reseña de los temas que serán incluidos en este trabajo:

1. Haremos un repaso a los fundamentos teóricos asociados al Problema de Programación Lineal. Esto consiste en la revisión de los conceptos y propiedades más importantes. Aún en esta sección, revisaremos algunos conceptos relacionados a la Dualidad.

2. Mediante la introducción de formalismos matemáticos, veremos cómo es que se clasifican los algoritmos. Estos criterios nos permitirán catalogar un algoritmo como eficiente o ineficiente. El objetivo de este sección será analizar los aspectos computacionales del Método Simplex y la existencia de métodos más eficientes.

3. Introduciremos algunas estrategias sobre las cuales está soportada la construcción del Método de Puntos Interiores Primal-Dual. Estas son: El Método de Newton, El Método de Lagrange y el Método de Barrera Logaritmo.

4. Finalmente, de una manera natural, utilizando todas las herramientas vistas en las secciones previas, haremos una descripción preliminar de la construcción del Método de Puntos Interiores Primal-Dual. Más aún, realizaremos una comparación entre este método y el Método Simplex, resaltando las ventajas y desventajas, tanto teóricas como computacionales, que estos métodos presentan.

La estructura capitular de la tesis es como sigue:

Introducción

Capítulo 1: El Problema de Programación Lineal

1. Propiedades
2. Dualidad y Condiciones de Optimalidad

Capítulo 2: Eficiencia de Algoritmos

1. Principio de Invariabilidad
2. Notación Asintótica
3. Por qué buscar algoritmos eficientes

Capítulo 3: Método de Puntos Interiores Primal-Dual para Programación Lineal

1. Método de Newton
2. Método de Lagrange
3. Método de Barrera Logaritmo
4. Construcción del Método de Puntos Interiores Primal-Dual

Capítulo 4: Un Estudio Comparativo del Método de Puntos Interiores Primal-Dual y el Método Simplex

1. Comparación desde el punto de vista práctico del Algoritmo de Puntos Interiores Primal-Dual y el Algoritmo Simplex
2. Análisis de las ventajas y desventajas cualitativas del Método de Puntos Interiores Primal-Dual y el Método Simplex: análisis de sensibilidad
3. Un panorama sobre la implementación computacional, tipos de soluciones óptimas, degeneración y mal condicionamiento numérico

Conclusiones Finales

Bibliografía

2.3. Antecedentes

La existencia de un algoritmo de tiempo polinomial para resolver el Problema de Programación Lineal, fue mostrada en 1978 por L. G. Khachian. Este hecho colocó al Problema de Programación Lineal en la clase $\mathcal{P}$, la clase de problemas para los cuales existen algoritmos eficientes.

En 1984, N. Karmarkar propuso un algoritmo de tiempo polinomial basado en transformaciones proyectivas, el Algoritmo de Karmarkar. Este algoritmo fue un serio competidor para el clásico Algoritmo Simplex (1947).

El más importante Método de Puntos Interiores es el Primal-Dual, propuesto por B. Jansen, C. Roos, C. Terlaky y J. P. Vial: *Primal-Dual Algorithms for Linear Programming Based on the Logarithmic Barrier Method*, Octubre de 1994.

A pesar que en los años noventa, cuando los métodos de puntos interiores estaban en su auge, creemos que muchos trabajos de natulareza comparativa fueron realizados a nivel internacional. Sin embargo, en la búsqueda realizada por nosotros a nivel regional, no se ha encontrado un trabajo similar.

2.4. Objetivos

1. Analizar los hechos más importantes de la construcción del Método Simplex, poniéndo énfasis en su eficiencia.

2. Analizar la construcción del Método de Puntos Interiores Primal–Dual para Programación Lineal e implementar en computador, de un modo particular, el algoritmo respectivo.

3. Comparar, en teoría y en la práctica, el desempeño de ambos métodos, cuando éstos son sometidos a problemas de gran dimensión.

4. Estudiar la viabilidad para realizar análisis de sensibilidad cuando se usa el Algoritmo de Puntos Interiores Primal-Dual en la resolución de problemas de programación lineal.

2.5. Hipótesis

Es posible utilizar con éxito el método de puntos interiores Primal-Dual, alternativo al Método Simplex, para resolver el Problema de Programación Lineal

3. Planteamiento Operacional

La realización de este trabajo se enmarca dentro de los aspectos exploratorios (investigación científica en artículos de publicación internacional), descriptivos (el proponer métodos modernos de resolver un problema que históricamente era resuelto por un método clásico) y experimentales (la comparación del desempeño en computador).

3.1. Técnicas, Instrumentos y Materiales de Verificación

3.1.1. Técnicas

Se utilizará la técnica de Obervación Documental.

3.1.2. Instrumentos

Los instrumentos que se utilizarán son:

1. Fichas documentales
2. Registro documental
3. Matrices de registro

3.1.3. Materiales

1. Bienes informáticos.
2. Se realizarán gráficos que ayuden a visualizar las ventajas y desventajas del método propuesto frente al tradicional.
3. Se utilizará una computadora para ejecutar los programas resultantes y realizar la fase de experimentación y comparación.

3.2. Campo de Verificación

3.2.1. Ubicación Espacial

El ámbito de estudio tiene un contexto global, debido principalmente a que el método de puntos interiores, relativamente nuevo, es de dominio científico y universal.

3.2.2. Ubicación Temporal

Dentro del campo científico, los factores y circunstancias que determinan el descubrimiento de nuevas estrategias para resolver diferentes problemas, suelen ser de momento, ya que nuevos avances buscan mejorar el desempeño de los métodos. Por lo tanto, este trabajo es de naturaleza coyuntural.

3.2.3. Fuentes

Artículos de publicación internacional

1. B. Jansen, C. Roos, C. Terlaky, J. P. Vial, *Primal-Dual Algorithms for Linear Programming Based on the Logarithmic Barrier Method.* Journal of Optimization and Applications. Vol. 83, No. 1, pp. 1-26, October 1994

Libros Especializados

1. E. Castillo, A. J. Conejo, P. Pedregal, R. García y N. Alguacil, *Formulación y Resolución de Modelos de Programación Matemática en Ingeniería y Ciencia*, 20 de febrero de 2002.

2. Taha H. A. *Investigación de Operaciones, una Introducción.* Sexta Edición. Prentice Hall. 1997.

3. Mokhtar S. Bazaraa, John J. Jarvis, Hanif D. Sherali, *Linear Programming and Network Flows*, Second Edition. John Wiley & Sons. 1990.

Internet Páginas WEB de universidades, nacionales y extranjeras, que proporcionen artículos libres sobre temas relacionados.

3.3. Estrategia de Recolección de Datos y Desarrollo de la Tesis

Puesto que el trabajo tiene, en parte, un soporte teórico matemático, primero se analizará la construcción de los métodos, paso a paso, con las debidas justificaciones.

Posteriormente, se realizará un estudio comparativo del método propuesto con una técnica clásica, el Método Simplex.

Finalmente, la parte práctica constituye la experimental, mediante esto, se ratificará la efectividad del método propuesto cuando sean sometido a

diferentes problemas de Programación Lineal, los resultados serán resumidos en tablas comparativas donde se aprecie, de un modo claro, la eficiencia tanto en tiempo como en espacio requerido.

4. Cronograma de Trabajo

El desarrollo de este proyecto obedecerá el siguiente cronograma:

Tiempo	Noviembre 2006				Diciembre 2006				Enero 2007				Febrero 2007			
Actividades	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4
1. Recolección de datos			X	X	X	X	X	X								
2. Estructuración de resultados									X	X	X	X	X			
3. Informe final														X	X	X

Arequipa, 31 de octubre de 2006

Manuel Arenas Céspedes