UNIVERSIDAD NACIONAL DE INGENIERIA
FACULTAD DE CIENCIAS

Escuela Profesional de Matemática

TESIS PARA OPTAR EL TITULO PROFESIONAL DE LICENCIADO EN MATEMATICA

TITULADA:

ALGORITMO DE KARMARKAR Y ALGORITMO PRIMAL-DUAL DE PUNTOS INTERIORES PARA PROGRAMACION LINEAL.

PRESENTADO POR:
VARGAS TRUJILLO, Carlos Enrique

LIMA - PERU
1998

Índice

Resumen

Este trabajo de tesis estudia dos algoritmos de punto interior para la programación lineal; el algoritmo de Karmarkar y el primal–dual en una de sus versiones iniciales. Se desarrolla en detalle ambos algoritmos y se prueba que son polinomiales en el peor caso, en contraposición al algoritmo simplex que es un algoritmo exponencial en el peor caso. Estos algoritmos, así como algunos de sus variantes mas modernas, han demostrado tener un mejor desempeño (número de iteraciones y tiempo de CPU) que el algoritmo simplex para problemas de programación lineal de gran tamaño.

Introducción

Este trabajo de tesis trata acerca del algoritmo de Karmarkar y del algoritmo primal–dual de puntos interiores. El algoritmo de Karmarkar desarrollado en el año 1984, fue el primer algoritmo de puntos interiores eficiente para la programación lineal, basado en ideas de programación no lineal y geometría proyectiva. Resuelve los problemas de programación lineal en tiempo polinómial, recorriendo la región factible en busca del óptimo por su interior.

El algoritmo de Karmarkar ha revolucionado la literatura especializada en programación lineal, y porque no decirlo, la de la optimización en general, dando lugar a algoritmos nuevos, asi como a software, ya comercialmente disponible, que reducen mucho los tiempos obtenidos con el método simplex para problemas de programación lineal de gran tamaño. Inicialmente intentamos presentar en este trabajo solo el algoritmo de Karmarkar, pero debido al rápido desarrollo en ésta area, nos parece oportuno desarrollar también un algoritmo más eficiente que es el algoritmo primal–dual de puntos interiores. Sin embargo solo presentamos una introducción a los algoritmos primal–dual, un tratamiento mas extenso hubiera hecho este trabajo demasiado denso. El contenido de esta tesis está estructurado como sigue:

En el **Capítulo 1** elaboramos los elementos necesarios de la teoría de complejidad computacional.

En el **Capítulo 2** introducimos a las ideas y conceptos escenciales relacionados con el algoritmo de Karmarkar, para de este modo lograr que su estudio detallado sea mas facil de entender.

En el **Capítulo 3** desarrollamos el algoritmo de Karmarkar en detalle y estudiamos los principales aspectos relacionados con él.

En el **Capítulo 4** mostramos como convertir un problema de programación lineal en forma standar a la forma requerida por el algoritmo de Karmarkar. Si bien esto pudo hacerse en el *capitulo 3* preferimos hacerlo separadamente para no hacer la lectura de dicho capítulo demasiado difícil, finalmente

En el **Capítulo 5** presentamos una introducción a los algoritmos primal–dual de puntos interiores. El algoritmo que será presentado trabaja con una longitud de paso contante y converge linealmente. Para el estudio detallado de algoritmos primal–dual de convergencia superlineal recien desarrollados en los últimos años, lamentablemente no ha quedado espacio suficiente, asi que damos solamente referencias e indicaciones a la bibliografía respectiva.

En el **Capítulo 6** presentamos los resultados numéricos obtenidos con estos algoritmos y su comparación con el algoritmo simplex en cuanto al número de iteraciones y al tiempo de CPU, sobre un conjunto de problemas reales y sobre problemas generados aleatoriamente.

Capítulo 1

Complejidad computacional de la programación lineal

1. Complejidad computacional de la programación lineal

En este capítulo, se establecen las bases y resultados esenciales de la teoría de complejidad computacional; definiremos conceptos que serán necesarios para ser usados posteriormente a lo largo de este trabajo de tesis. Ponemos enfasis en el caso del problema de la programación lineal y se presentan las principales conclusiones que se han obtenido en relación al algoritmo simplex para la programación lineal.

Cabe mencionar que la presentación de los conceptos estudiados en este capítulo está basada en las ideas presentadas en el artículo escrito por B. Wah y C. Ramamoorthy [18] y en la monografía escrita por S. Nash y A. Sofer [14].

1.1 Noción de complejidad

La teoría de complejidad tiene dos objetivos principales

1. Analizar la eficiencia de diferentes algoritmos para un determinado problema y compararlos.

2. Clasificar los problemas utilizando como criterio su grado de dificultad.

Hay muchos criterios para caracterizar la eficiencia de un algoritmo, de los cuales los dos criterios más importantes son el tiempo necesario para ejecutar el algoritmo, el cual se da como una función del tamaño de la entrada, y el espacio de memoria, medido en bits, que el computador necesita, para la ejecución del algoritmo.

La complejidad de un algoritmo mide el tiempo ó espacio de memoria ó una combinación de ambos, requeridos para ejecutar el algoritmo. En este trabajo de tesis nos restringiremos en esencia al criterio del tiempo de ejecución ya que es una medida más fácil de analizar. Es decir, expresaremos la complejidad de un algoritmo como una función

del tamaño de la entrada, sobre la cual se ejecuta el algoritmo, calculando el tiempo de ejecución requerido para resolver un problema específico.

Sin embargo, para que nuestro análisis tenga validez general y, en particular, no dependa del tipo de computador usado, requerimos de una medida apropiada para el tiempo de ejecución. Por esta razón asumiremos que cada operación elemental (adición, multiplicación, división, comparación, etc) requiere de una unidad de tiempo. Esto motiva la siguiente definición (S. Nash y A. Sofer [14]).

Definición 1.1 *El tiempo de ejecución de un algoritmo es el número de operaciones computacionales requeridas, para resolver un problema particular usando un algoritmo específico.* □

1.2 Entrada y tamaño de la entrada de un problema de programación lineal

Definiremos ahora los conceptos de entrada y tamaño de la entrada aplicados al caso de la programación lineal. Según B. Wah [18] se define:

Definición 1.2 *Consideremos el problema de programación lineal:*

$$\begin{aligned} \min \quad & c^T x \\ \textit{sujeto a} \quad & Ax = b \\ & x \geq 0, \end{aligned}$$

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

Como entrada de un problema de programación lineal se entenderá el conjunto de datos que permite su caracterización. Una entrada o concretización para este problema es caracterizada por un conjunto específico (m, n, A, b, c) *de datos.* □

Definición 1.3 *El tamaño de la entrada de un problema de programación lineal ó simplemente el tamaño del problema, es la cantidad de bits requeridos para almacenar todos los datos de entrada del problema de programación lineal.*

De acuerdo con esta definición, el tamaño de un problema de programación lineal concreto, será el número de bits requeridos para representar el conjunto de datos (m, n, A, b, c). Si denotamos con L el número de bits requeridos para representar el conjunto de datos (m, n, A, b, c); tenemos según S. Nash y A. Sofer [14] que:

$$L = \left[\left[\sum_{j=1}^{n}\sum_{i=1}^{m}\log_2(|a_{ij}|+1)\right]\right] + \left[\left[\sum_{i=1}^{m}\log_2(|b_i|+1)\right]\right] + \left[\left[\sum_{j=1}^{n}\log_2(|c_j|+1)\right]\right] +$$
$$[\![\log_2(n+1)]\!] + [\![\log_2(m+1)]\!] + (mn+n+m+1)$$

(La notación $[\![x]\!]$, denota el mayor entero menor o igual a x).

El término final $(mn + n + m + 1)$ representa el espacio necesario para almacenar los signos de todos los números, más un bit adicional para indicar cuando el problema de programación lineal es un problema de minimización ó uno de maximización. Se ha agregado un 1 en cada término logarítmico con la finalidad de que ningún término dentro del logaritmo se reduzca a cero a causa de que algún a_{ij}, b_i o c_j sea nulo.

Desarrollamos a continuación un ejemplo propuesto por S. Nash y A. Sofer [14].

Ejemplo 1.1 *Calcular L, el tamaño del problema de programación lineal:*

$$\begin{aligned} \max \quad & z = 5x_1 + 7x_2 + 9x_3 + 12x_4 \\ \textit{sujeto a}: \quad & 3x_1 - 9x_2 + 6x_3 - 4x_4 = 12 \\ & 2x_1 + 3x_2 - 2x_3 + 7x_4 = 21 \\ & x_1,\ x_2,\ x_3,\ x_4 \geq 0 \end{aligned}$$

En este caso tenemos:

$$A = \begin{pmatrix} 3 & -9 & 6 & -4 \\ 2 & 3 & -2 & 1 \end{pmatrix},$$

$$c = (5 \quad 7 \quad 9 \quad 12)^T,$$

$$b = (12 \quad 21)^T,$$

$$n = 4,$$

$$m = 2.$$

Luego:

$$\begin{aligned}
\sum_{j=1}^{4}\sum_{i=1}^{2}\log_2(|a_{ij}|+1) &= \sum_{j=1}^{4}\Big\{\log_2(|a_{1j}|+1)+\log_2(|a_{2j}|+1)\Big\} \\
&= \log_2(|a_{11}|+1)+\log_2(|a_{21}|+1)+\log_2(|a_{12}|+1)+\log_2(|a_{22}|+1)+ \\
&\quad \log_2(|a_{13}|+1)+\log_2(|a_{23}|+1)+\log_2(|a_{14}|+1)+\log_2(|a_{24}|+1) \\
&= \log_2(|3|+1)+\log_2(|2|+1)+\log_2(|-9|+1)+\log_2(|3|+1)+ \\
&\quad \log_2(|6|+1)+\log_2(|-2|+1)+\log_2(|-4|+1)+\log_2(|1|+1) \\
&= \log_2 4+\log_2 3+\log_2 10+\log_2 4+\log_2 7+\log_2 3+\log_2 5+\log_2 2 \\
&= 16.621,
\end{aligned}$$

entonces:

$$\left[\left[\sum_{j=1}^{4}\sum_{i=1}^{2}\log_2(|a_{ij}|+1)\right]\right] = 16.$$

Además:

$$\begin{aligned}
\sum_{i=1}^{2}\log_2(|b_i|+1) &= \log_2(|b_1|+1)+\log_2(|b_2|+1) \\
&= \log_2(|12|+1)+\log_2(|21|+1) \\
&= \log_2(13)+\log_2(22) \\
&= 8.159,
\end{aligned}$$

$$\left[\left[\sum_{i=1}^{2}\log_2(|b_i|+1)\right]\right] = 8,$$

$$\sum_{j=1}^{4}\log_2(|c_j|+1) = \log_2(|c_1|+1)+\log_2(|c_2|+1)+\log_2(|c_3|+1)+\log_2(|c_4|+1)$$

$$= \log_2(|5|+1)+\log_2(|7|+1)+\log_2(|9|+1)+\log_2(|12|+1)$$

$$= \log_2 6+\log_2 8+\log_2 10+\log_2 13$$

$$= 12.607,$$

$$\left[\left[\sum_{j=1}^{4}\log_2(|c_j|+1)\right]\right] = 12,$$

$$[\![\log_2(n+1)]\!] = [\![\log_2(4+1)]\!] = [\![\log_2 5]\!] = 2,$$

$$[\![\log_2(m+1)]\!] = [\![\log_2(2+1)]\!] = [\![\log_2 3]\!] = 1.$$

Finalmente resulta:

$$L = 16+8+12+2+1+(2\times 4+4+2+1) \quad \textit{bits.}$$

$$L = 54 \quad \text{bits.}$$

Definición 1.4 (S. Nash y Sofer [14]) *Sean*

$$f:\mathbb{N}\mapsto\mathbb{R}^+ \qquad y \qquad g:\mathbb{N}\mapsto\mathbb{R}^+.$$

Decimos que $f(n)=\mathcal{O}(g(n))$, si existe una constante $c>0$ y un $n_0\in\mathbb{N}$ tal que

$$f(n)\leq cg(n), \quad \forall n\geq n_0 \qquad \square$$

Observación: $g(n)$ es un límite superior para $f(n)$ a partir de n_0.

Ejemplo 1.2 *Un polinomio $p(n)=\sum_{i=0}^{k}c_in^i$ es $\mathcal{O}(n^k)$. En otras palabras, ignoramos todos los términos de grado inferior a k y todas las constantes. Eso significa que sólo se está considerando el comportamiento asintótico de la función cuando $n\to\infty$.*

1.3 Algoritmo polinomial

En esta sección definimos lo que es un algoritmo polinomial. Un algoritmo polinomial podría (informalmente) definirse como aquel que puede resolver alguna entrada de un problema usando un número de operaciones elementales, el cual está acotado superiormente por algún polinomio cuya variable es el tamaño del problema. Según S. Nash y A. Sofer [14] se define:

Definición 1.5 *Sea* n *el tamaño de un problema, sea además* $T(n)$ *el tiempo de ejecución de un algoritmo para resolver el problema. El algoritmo es llamado* **polinomial**, *si existe un polinomio* $P(n)$ *tal que:*

$$T(n) = \mathcal{O}(P(n)). \qquad \square$$

Todos los algoritmos cuya complejidad no está acotada por algún polinomio, serán llamados exponenciales, porque 2^n es el paradigma para una tasa de crecimiento no polinomial.

En la siguiente tabla (S. Nash y A. Sofer [14]) se compara el crecimiento de algunas funciones polinomiales y otras no polinomiales. Sólo damos valores aproximados, ya que estos son suficientes para ver el orden de crecimiento de funciones polinomiales y no polinomiales.

n	n^2	n^3	n^{100}	2^n	$n!$
2	4	8	1×10^{30}	4	2
5	25	1×10^2	8×10^{69}	32	1×10^2
10	1×10^2	1×10^3	1×10^{100}	1×10^3	4×10^6
50	3×10^3	1×10^5	8×10^{169}	1×10^{15}	3×10^{64}
100	1×10^4	1×10^6	1×10^{200}	1×10^{30}	9×10^{157}

Tabla 1.1: **Comparación del orden de crecimiento**

La ventaja de los algoritmos polinomiales, podrá ser apreciada comparando el crecimiento de n^2 versus 2^n si n representa el tamaño de algún problema y si A_1 y A_2 son dos algoritmos para estos problemas y si además suponemos que la complejidad del algoritmo A_1 es $\mathcal{O}(n^2)$ y la del algoritmo A_2 es $\mathcal{O}(2^n)$. Entonces esto significa que:

Tiempo de ejecución del algoritmo A_1 $<$ $c_1 n^2$ y
Tiempo de ejecución del algoritmo A_2 $<$ $c_2 2^n$.

Luego, para un tamaño del problema de $n = 50$, asumiendo $c_1 = c_2 = 1$, el algoritmo A_1 requerirá no más de 2500 operaciones y el algoritmo A_2 podría requerir 1×10^{15}.
Existen dos anomalías con este sistema de evaluación de los algoritmos.
Primero, debemos tratar de obtener cotas polinomiales de grado bajo. Un algoritmo polinomial con complejidad $\mathcal{O}(n^{100})$ podría no ser de valor más práctico que el algoritmo A_2.
Segundo, las magnitudes de los coeficientes de las cotas polinomiales deben ser relativamente pequeñas. Si $c_1 = 2^{50}$ y $c_2 = 1$ entonces, aún cuando el tiempo de ejecución del algoritmo A_2 se incrementa con n mucho más rápido que en el algoritmo A_1 desde un punto de vista práctico A_1 no es manejable para algún valor de $n \geq 1$, no obstante el algoritmo A_2 es manejable para valores pequeños de n. Afortunadamente en la práctica esto no suele ocurrir, es decir cuando se encuentran algoritmos polinomiales, éstos son de grado bajo y con cotas polinomiales relativamente pequeñas.

1.4 Análisis del peor caso y del caso promedio

Como mencionamos en 1.1, la teoría de complejidad, caracteriza la eficiencia de un algoritmo, midiendo el tiempo necesario para ejecutar el algoritmo como una función del tamaño de la entrada, que el computador necesita, para la ejecución del algoritmo. Para el estudio de la complejidad, según B. Wah y C. Ramamoorthy [18], se pueden distinguir dos enfoques distintos: Un análisis en el peor caso y un análisis en el caso promedio. En esta sección estudiaremos brevemente ambos enfoques.

1.4.1 Análisis del peor caso

Sea X un problema determinado, I y J conjunto de índices y sea $X_i, i \in I$, la familia de sus concretizaciones. Sea $X_j(n), j \in J$, la familia de las concretizaciones de X, cuyo tamaño de la entrada es n, y sea $T_j(n)$ el tiempo de ejecución del algoritmo para la concretización de $X_j(n)$. Tenemos la siguiente definición:

Definición 1.6 (Comportamiento en el peor caso) *El comportamiento en el peor caso es definido como el máximo tiempo requerido por cualquier entrada posible, es decir:*

$$T(n) = \max\{T_j(n) | \quad j \in J\}. \qquad \square$$

$T(n)$ se llamará el tiempo de ejecución del algoritmo para resolver X. Se debe reconocer que, en esta definición del tiempo de ejecución, reside la mayor debilidad de la teoría de complejidad, porque $T(n)$ es la medida más pesimista posible para el tiempo de ejecución (para las concretizaciones cuyo tamaño de entrada es n).

1.4.2 Análisis del caso promedio

Como en 1.4.1, sea X un problema determinado y $X_i, i \in I$, la familia de sus concretizaciones. Sea $X_j(n), j \in J$, la familia de las concretizaciones de X, cuyo tamaño de la entrada es n. Sea $P(X_j(n))$ la probabilidad de que $X_j(n)$ ocurra, y sea $T_j(n)$ el tiempo de ejecución del algoritmo para la concretización $X_j(n)$. Según B.Wah y C. Ramamoorthy [18] se define:

Definición 1.7 (Comportamiento en el caso promedio $A(n)$) *El comportamiento en el caso promedio $A(n)$ es definido como el tiempo esperado, requerido por todas las concretizaciones de X, es decir:*

$$A(n) = \sum_{j \in J} P(X_j(n)) T_j(n). \qquad \square$$

1.4.3 Análisis en el peor caso vs Análisis en el caso promedio

En el análisis en el peor caso, se debe estudiar la concretización más desfavorable para su desempeño, de modo de asegurar que, para cualquier concretización del problema, el tiempo de ejecución esté acotado por alguna función de su tamaño. Sin embargo, estas concretizaciones son de rara ocurrencia y no son representativas de una concretización típica del problema. Es decir con este tipo de análisis no es posible dar una imagen real de cuando un gran porcentaje de concretizaciones de un tamaño dado puede ser rápidamente resuelto y sólo un porcentaje muy pequeño requiere de considerablemente más tiempo. En estas situaciones, serían preferibles medidas tales como el análisis en el caso promedio. Pero medidas que requieren una distribución de probabilidad de las concretizaciones, parecen ser más difíciles de analizar, ya que elegir la distribución de probabilidad y medir la probabilidad de ocurrencia de las diversas concretizaciones no es fácil de hacer. La función de distribución de probabilidad, está determinada por la experiencia y/o información especial acerca del problema concreto que se está estudiando.

Además el análisis del caso promedio casualmente es complicado y son necesarias simplificaciones y suposiciones para facilitar el análisis y el tratamiento matemático del comportamiento del algoritmo en promedio, o en relación a sus concretizaciones más típicas.

1.5 Comportamiento del algoritmo simplex en el peor caso

En esta secciόm discutiremos brevemente la performance del algoritmo simplex en el peor caso, y mostramos la principal conclusión a que se ha llegado en este caso: El algoritmo simplex no es polinomial.

Como mencionan S. Nash y A. Sofer [14], se han observado que el número de iteraciones requeridas por el algoritmo simplex para hallar la solución óptima a un problema de programación lineal, es a menudo un múltiplo del número de restricciones. Esto es

importante desde que un problema de n variables y m restricciones podría tener $\binom{n}{m}$ soluciones básicas (por supuesto, muchas de estas probablemente sean no factibles).

Siempre existió la posibilidad de que el algoritmo simplex pudiese examinar todas estas bases antes de hallar la base óptima, pero hasta 1970 nadie había sido capaz de exhibir una sería de problemas de programación lineal (con un número arbitrario de variables) donde el algoritmo simplex tenga que hacer un número exponencial de iteraciones para hallar la solución óptima.

Hay una considerable diferencia entre m y $\binom{n}{m}$ iteraciones. Si ponemos $n = 2m$ entonces los valores de estas dos cantidades son: Aún para valores pequeños de m el

m	$\binom{2m}{m}$
1	2
5	252
10	184756
20	1×10^{11}
50	1×10^{29}
100	9×10^{58}
200	1×10^{119}
300	1×10^{179}
400	2×10^{239}
500	3×10^{299}

Tabla 1.2: Comparación entre m y $\binom{2m}{n}$ iteraciones

número de bases posible es inmenso. Si tuvieramos una computadora capaz de realizar un billón de iteraciones simplex por segundo, entonces examinar $\binom{100}{50}$ bases (aquí $m = 50$) tomaría 3 199 243 548 302, 2 años ([14]).

Si el número de iteraciones simplex fuera proporcional a m, según S. Nash y A. Sofer [14], el método simplex sería polinomial. Además los mismos autores hacen ver que todas las operaciones de una iteración simplex, son cálculos vectoriales y matriciales, con un total de operaciones aritméticas del orden de $\mathcal{O}(mn)$.

La refactorización periódica de la matriz básica tiene una cantidad de operaciones aritméticas del orden de $\mathcal{O}(m^3)$. Luego el número de operaciones aritméticas de una iteración simplex es del orden de $\mathcal{O}(m^3 + mn)$ y si el número de iteraciones fuera siempre del orden de $\mathcal{O}(m)$, entonces la cantidad total de operaciones aritméticas del algoritmo simplex sería del orden de $\mathcal{O}(m^4 + nm^2)$. Este número es polinomial en m y n. Por otro lado, si $\binom{n}{m}$ iteraciones fueran requeridas, entonces el algoritmo simplex sería un algoritmo exponencial, ya que:

$$\binom{n}{m} = \frac{n!}{m!(n-m)!} = \left(\frac{n}{m}\right)\left(\frac{n-1}{m-1}\right)\left(\frac{n-2}{m-2}\right)\cdots\left(\frac{n-(m-1)}{m-(m-1)}\right),$$

cada factor es de la forma

$$\frac{n-i}{m-i}, \quad 0 \leq i \leq m-1.$$

Como

$$\frac{n-i}{m-i} > \frac{n}{m} \quad i = 0, \ldots, m-1, \quad \text{tenemos}$$

$$\binom{n}{m} = \frac{n!}{m!(n-m)!} = \prod_{i=0}^{m-1} \frac{n-i}{m-i} \geq \left(\frac{n}{m}\right)^m.$$

Si $n \geq 2m$, entonces:

$$\binom{n}{m} \geq \left(\frac{n}{m}\right)^m \geq 2^m.$$

Luego el algoritmo simplex sería un **algoritmo polinomial** (si el número de iteraciones fuera siempre del orden de $\mathcal{O}(m)$) ó un algoritmo exponencial (si el número de iteraciones fuera algunas veces $\binom{n}{m}$ y $n \geq 2m$).

En un artículo escrito en 1992, Klee y Minty [7] mostraron que existen problemas de tamaño arbitrario, para los cuales el algoritmo simplex requiere un número exponencial de iteraciones y de este modo probaron que el algoritmo simplex, es un algoritmo exponencial en el peor caso.

Presentamos aquí una variante del problema original de Klee y Minty [7], tomado del libro de S. Nash y A. Sofer [14]:

$$\begin{cases} \max\ z = \displaystyle\sum_{j=1}^{m} 10^{m-j} x_j \\ \text{sujeto a} : \ 2\displaystyle\sum_{j=1}^{i-1} 10^{i-j} x_j + x_i \leq 100^{i-j}, \quad i = 1, \ldots, m \\ x \geq 0 \end{cases}$$

Cuando se añaden variables de holgura hay 2m variables y m restricciones. Puede probarse (Klee-Minty [7]) que estos problemas tienen 2^m bases factibles y todas ellas son examinadas por el algoritmo simplex.

Ejemplo 1.3 *Si $m = 3$ el problema de Klee-Minty, tiene la forma:*

$$\begin{array}{rrcrcrcl} \max\ z & = & 100x_1 & + & 10x_2 & + & x_3 & \\ \textit{sujeto a} & : & x_1 & & & & & \leq\ 1 \\ & & 20x_1 & + & x_2 & & & \leq\ 100 \\ & & 200x_1 & + & 20x_2 & + & x_3 & \leq\ 10000 \\ & & x_1 & , & x_2 & , & x_3 & \geq\ 0. \end{array}$$

Si añadimos variables de holgura: $s_1, s_2, s_3 \geq 0$, el problema anterior se escribe como:

$$\begin{array}{rrcrcrcrcrcrcl} \max\ z & = & 100x_1 & + & 10x_2 & + & x_3 & & & & & & \\ \textit{sujeto a} & : & x_1 & & & & & + & s_1 & & & & = 1 \\ & & 20x_1 & + & x_2 & & & & & + & s_2 & & = 100 \\ & & 200x_1 & + & 20x_2 & + & x_3 & & & & & + & s_3 = 10000 \\ & & x_1 & , & x_2 & , & x_3 & , & s_1 & , & s_2 & , & s_3 \geq 0. \end{array}$$

La región factible es mostrada en la Figura 1.1.

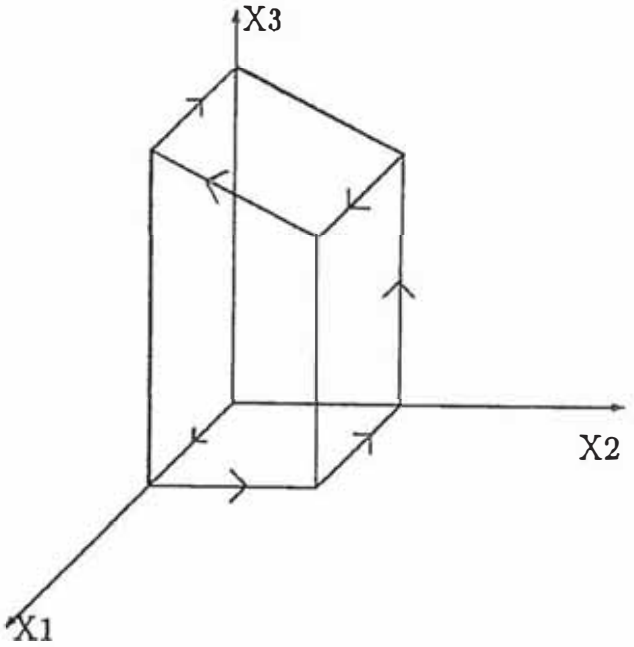


Figura 1.1: Problema de Klee-Minty

Cuando el algoritmo simplex es aplicado, la sucesión de soluciones básicas factibles es; como se muestra en la tabla:

Bases			z
$s_1 = 1$	$s_2 = 100$	$s_3 = 10000$	0
$x_1 = 1$	$s_2 = 80$	$s_3 = 9800$	100
$x_1 = 1$	$x_2 = 80$	$s_3 = 8200$	900
$s_1 = 1$	$x_2 = 100$	$s_3 = 8000$	1000
$s_1 = 1$	$x_2 = 100$	$x_3 = 8000$	9000
$x_1 = 1$	$x_2 = 80$	$x_3 = 8200$	9100
$x_1 = 1$	$s_2 = 80$	$x_3 = 9800$	9900
$s_1 = 1$	$s_2 = 100$	$x_3 = 10000$	10000

Tabla 1.3: Soluciones básicas factibles para el problema de Klee-Minty ($m = 3$)

Este problema tiene 2^3 soluciones básicas factibles y todos son examinadas por el algoritmo simplex.

A continuación, presentaremos las principales conclusiones a las que se llegó en relación al comportamiento del algoritmo simplex en el caso promedio.

1.6 Comportamiento del algoritmo simplex en el caso promedio

En esta sección presentaremos, brevemente, algunos resultados acerca del comportamiento del algoritmo simplex en el caso promedio y que, en cierto modo, validan su efectividad en la práctica. Nos referimos al trabajo de Nash y Sofer [14].

Estos autores mencionan que la evidencia empírica mostraba que un problema de programación lineal con n variables y m restricciones, requería entre m y 3m iteraciones para ser resuelto con el algoritmo simplex. Este número de iteraciones era considerado lo suficientemente bajo para hacer del algoritmo simplex, un algoritmo eficiente y efectivo.

Aún los ejemplos pesimistas de Klee y Minty, no lograron opacar el entusiasmo por el algoritmo simplex. Sin embargo, estudiar el comportamiento del algoritmo simplex en el caso promedio es difícil, y los resultados obtenidos son influenciados por la accesibilidad y tratabilidad de los instrumentos matemáticos apropiados.

En esta sección asumiremos que los problemas de programación lineal están en la forma:

$$\begin{aligned} \max\ z \quad & c^T x \\ \text{sujeto a} \quad : \quad & Ax \leq b \end{aligned}$$

donde $A \in \mathbb{R}^{m \times n}$. La fila i-ésima de A es denotada por a_i^T. Asumiremos también que un punto inicial factible x_0, es dado (para ciertos resultados, el vector b es escogido como $b = e \equiv (1, \ldots, 1)^T$, y así x_0 será automáticamente factible para estos problemas).

El primer teorema que demostró que, en promedio, el algoritmo simplex converge en un número de iteraciones que es polinomial en m y n fue descubierto por Borgwardt [1] en 1982. Su teorema es presentado a continuación sin prueba. Este teorema asume que

los coeficientes en el problema de programación lineal son escogidos aleatoriamente en $\mathbb{R}^n \setminus \{0\}$. El vector del lado derecho b, del problema de programación lineal considerado en el teorema no es aleatorio; es el vector $e = (1, \ldots, 1)^T$. Notese que no hay restricciones de no negatividad sobre las variables, es decir no se pide que $x \geq 0$.

Teorema 1.1 (Borgwardt[1], Nash y Sofer [14]) *Consideramos un problema de programación lineal de la forma*

$$\begin{aligned} \max \; z &= c^T x \\ \textit{sujeto a} \; : \; & Ax \leq e \end{aligned}$$

donde $c, a_1, \ldots, a_m$, las columnas de A, son escogidos aleatoriamente y están uniformemente distribuídos en $\mathbb{R}^n \setminus \{0\}$. Si el algoritmo simplex es aplicado a este problema, el número esperado de iteraciones está acotado, según Nash, Sofer y Borgwardt, por:

$$17n^3 m^{1/n-1} \qquad \square$$

Este resultado establece el comportamiento polinomial en el caso promedio del algoritmo simplex, pero la cota obtenida no correspondía a la observada en la práctica, esto es, entre m y 3m iteraciones.

S. Nash y A. Sofer [14], mencionan que un resultado con una conclusión más satisfactoria fue obtenido independientemente por Haimovich y Adler en 1983. Ellos usan una forma del problema de programación lineal diferente a Borwardt (el lado derecho b es ahora también aleatorio).

Teorema 1.2 (Haimovich-Adler [5], Nash y Sofer [14]) *Consideremos un problema de programación lineal en la forma:*

$$\begin{aligned} \max \; z &= c^T x \\ \textit{sujeto a} \; : \; & Ax \leq b \end{aligned}$$

donde c, b y los coeficientes de A son escogidos aleatoriamente. Si el algoritmo simplex es aplicado a este problema, entonces el número esperado de iteraciones está acotado por:

$$n\left(\frac{m-n+2}{n+1}\right) \qquad \square$$

Conclusión: En este capítulo, hemos dado los fundamentos de la teoría de complejidad computacional y nos hemos limitado al caso de la programación lineal. Hemos comparado los algoritmos exponenciales y polinomiales basado en la programación en el peor caso. De acuerdo a esto, se ha probado que para el algoritmo simplex existe una clase de problemas donde efectúa un número exponencial de iteraciones. Sin embargo, también vimos que existen algunos resultados en el análisis del caso promedio que ayudan a resolver la discrepancia entre el análisis en el peor caso y la eficiencia práctica observada en el algoritmo simplex. Los problemas para los cuales existen algoritmos polinomiales que las resuelven son llamados problemas de clase P. El algoritmo simplex no permitía responder a la pregunta de que el problema de programación lineal era o no un problema de clase P. Esta pregunta quedó respondida afirmativamete cuando L. Kachian en 1979 propuso un algoritmo polinomial para el problema de la programación lineal pero, desafortunadamente, las experiencias computacionales con el algoritmo de L. Kachian no fueron satisfactorias (lo cual significa que su performance en el caso promedio era cercana a su performance en el peor caso).

Un algoritmo polinomial eficiente para el problema de programación lineal fue propuesto por N. Karmarkar, el cual será tratado en detalle en los capítulos siguientes. Curiosamente, este y otros algoritmos que presentamos, son aplicaciones de la programación no-lineal a problemas de programación lineal.

Capítulo 2

Conceptos básicos relacionados con el algoritmo de Karmarkar

2. Conceptos básicos relacionados con el algoritmo de Karmarkar

En este capítulo presentaremos los fundamentos del algoritmo de Karmarkar, es decir los aspectos básicos en los que se fundamenta. Explicaremos el rol que desempeñan en el algoritmo conceptos como la transformación proyectiva, el rol jugado por un poliedro especial llamado simplejo standard, y la optimización de la función objetivo en una esfera inscrita en este simplejo. Dejaremos para el capítulo siguiente el estudio de la función potencial, ya que será allí donde se aprecie mejor el rol jugado por esta función dentro del algoritmo. Los conceptos presentados aquí tienen como base el artículo de Karmarkar [6].

2.1 Una noción intuitiva del algoritmo

Empezamos esta sección con una presentación intuitiva de las ideas básicas del algoritmo de Karmarkar, para que de este modo, la formalización de los conceptos presentados de manera intuitiva, sea mejor comprendida.

Supongamos que el problema de programación lineal que nos interesa considerar tiene la forma:

$$\begin{aligned} \min \quad & c^T x \\ \text{sujeto a}: \quad & Ax = b \\ & x \geq 0 \end{aligned}$$

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

Sea x^0 un punto interior del conjunto de soluciones factibles. A partir de este punto, el algoritmo de Karmarkar generará una sucesión $x^0, x^1, \ldots, x^k \in \mathbb{R}^n$ de puntos interiores que converge a la solución óptima del problema. En esta construcción, un punto de la sucesión se obtiene a partir del punto anterior. Para entender cómo se construye esta sucesión, consideremos la siguiente representación gráfica del problema (Figura 2.1), en la que las líneas punteadas corresponden a curvas de nivel de la función objetivo y x^p es

la solución óptima del problema.

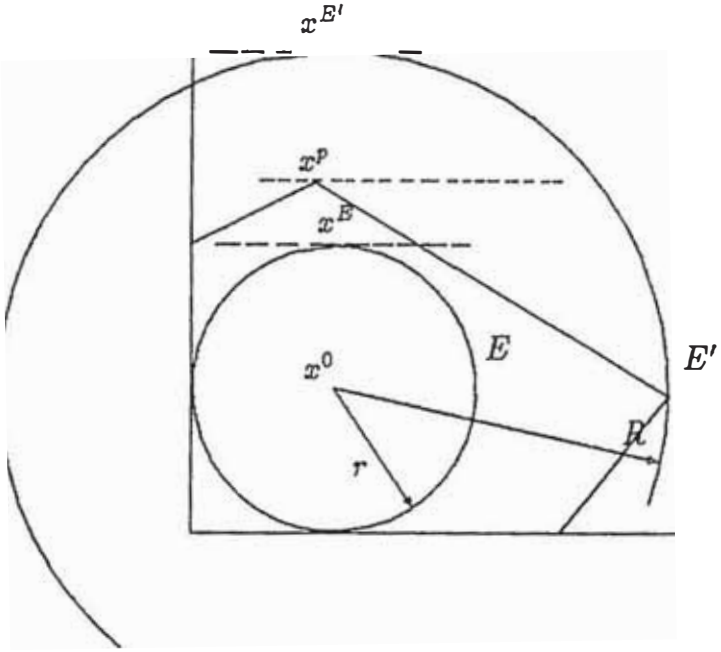


Figura 2.1: Gráfica del Problema

Supongamos que E y E' corresponden respectivamente, a la mayor esfera inscrita y a la menor esfera circunscrita al conjunto de soluciones factibles con centro en x^0 (sean r y R los radios de estas esferas). Consideremos ahora un nuevo problema de optimización, que corresponde a minimizar $c^T x$ sobre los puntos contenidos en E. La solución óptima de este problema corresponde al punto x^E de la figura, y se obtiene desplazando la función objetivo en la dirección apropiada hasta que sea tangente al borde de la esfera E.

Es evidente que al movernos de x^0 a x^E (y en este sentido este podría corresponder al siguiente punto de la sucesión, esto es, $x' = x^E$) nos hemos acercado a la solución óptima del problema. ¿Cómo se puede medir este acercamiento?. Para poder hacerlo en términos de la modificación que se ha producido en el valor de la función objetivo, consideremos el siguiente problema auxiliar:

Minimizar $c^T x$ sobre los puntos contenidos en E', la solución corresponde al punto $x^{E'}$ de la Figura 2.1.

Luego se tiene los siguientes problemas de optimización:

$$\begin{array}{lll} \min c^T x & \min c^T x & \min c^T x \\ Ax = b & \|x - x^0\| \leq r & \|x - x^0\| < R \\ x > 0 & x \geq 0 & x > 0 \end{array}$$

Debido a las relaciones de inclusión que existen entre los tres problemas de optimización que hemos definido, se cumple que

$$c^T x^{B'} \leq c^T x^P \leq c^T x^B, \tag{2.1}$$

y por lo tanto

$$c^T x^0 - c^T x^B \leq c^T x^0 - c^T x^P \leq c^T x^0 - c^T x^{B'}. \tag{2.2}$$

Ahora bien, si llamamos γ a la proporción entre el radio de la esfera circunscrita y el de la inscrita, es decir

$$\gamma = \frac{R}{r}.$$

Ahora, de la Figura 2.1, observamos que:

$$x^0 \vec{x^{B'}} = \gamma x^0 \vec{x^B},$$

entonces:

$$x^{B'} - x^0 = \gamma(x^B - x^0) \quad \text{ó} \quad x^0 - x^{B'} = \gamma(x^0 - x^B).$$

Entonces por la linealidad de la función objetivo se tiene que:

$$c^T x^0 - c^T x^{B'} = \gamma(c^T x^0 - c^T x^B). \tag{2.3}$$

A partir de (2.2) y (2.3) se obtiene

$$c^T x^0 - c^T x^P \leq \gamma(c^T x^0 - c^T x^B),$$

y luego

$$c^T x^0 - c^T x^B \geq \frac{1}{\gamma}(c^T x^0 - c^T x^P),$$

o bien;

$$c^T x^B - c^T x^0 \leq -\frac{1}{\gamma}(c^T x^0 - c^T x^P).$$

Sumando $c^T x^0 - c^T x^p$ a ambos lados de esta desigualdad se obtiene

$$c^T x^E - c^T x^p \leq (1 - \frac{1}{\gamma})(c^T x^0 - c^T x^p). \tag{2.4}$$

La desigualdad (2.4) representa uno de los elementos básicos en el que se fundamenta el algoritmo. Indica que si nos movemos del punto x^0 al punto x^E, logramos reducir la distancia inicial al valor óptimo (esto es $c^T x^0 - c^T x^p$) en el factor $1 - \frac{1}{\gamma} < 1$.

Recordemos que γ es una relación entre los radios de la esfera circunscrita y la inscrita y por lo tanto siempre será mayor que 1, pero mientras más cercano a 1 sea su valor, mayor será el acercamiento obtenido al valor óptimo.

Así por ejemplo, si $\gamma = 1.1$, $1 - \frac{1}{\gamma} \approx 0.09$ y por lo tanto, si la distancia inicial (del centro) al valor óptimo es 1000 en el punto x^E esta distancia se reduce a 90 (por lo menos).

El valor de γ depende fundamentalmente de dos factores: del punto inicial x^0 y de la frontera del conjunto de soluciones factibles. Esta dependencia resulta difícil de cuantificar en general. Un caso en que ambos factores juegan en contra del procedimiento mencionado anteriormente aparece representado en la Figura 2.2.

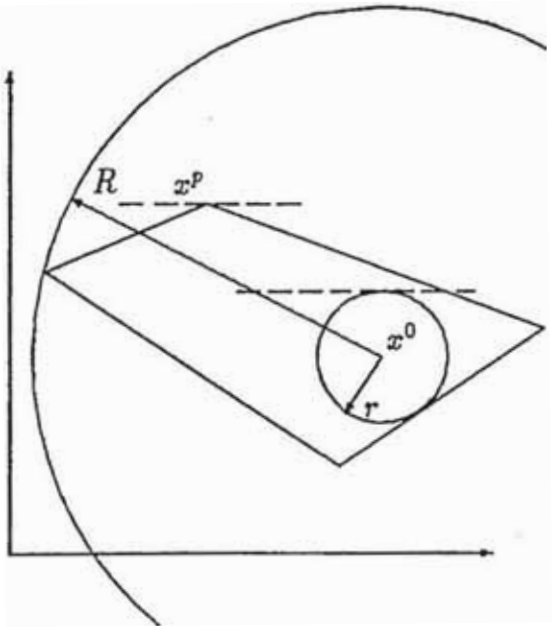


Figura 2.2:

En todo caso, en base a lo que ya ha sido expuesto, es posible diseñar un algoritmo secuencial que operaria del siguiente modo: A partir de un punto inicial x^0, optimizar la función objetivo sobre la mayor esfera inscrita en el conjunto de soluciones factibles con centro en x^0. Denominamos x^1 este punto x^1, y optimizamos nuevamente la función objetivo sobre la mayor esfera inscrita en el conjunto de soluciones factibles con centro en x^1, y así sucesivamente.

Ahora nótese, que si uno quiere asegurar que en un cierto número de iteraciones, el algoritmo se acercará suficientemente al óptimo, es necesario estimar a priori los valores de γ en las iteraciones sucesivas. Este es nuestro primer problema. En la medida que la geometría del conjunto de soluciones factibles es simple, es posible realizar tal estimación. Por ejemplo, consideremos un conjunto correspondiente a un triángulo equilátero como el que aparece en la Figura 2.3 y en la que se escoge como punto x^0, al centro geométrico de este triángulo.

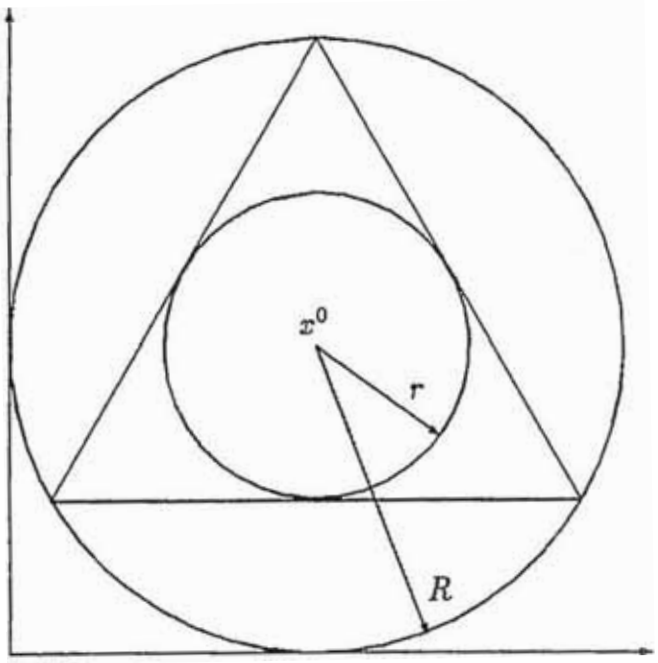


Figura 2.3:

Por Geometría elemental es conocido, que en este caso $\gamma = \frac{R}{r} = 2$. En general, en un espacio n-dimensional, el equivalente a esta figura corresponderá a los puntos $x \in \mathbb{R}^n$ que satisfacen:

$$x > 0 \qquad \text{y} \qquad \sum_{i=1}^{n} x_i = 1. \tag{2.5}$$

Probaremos que en este caso es posible hacer una estimación de γ.

Para esto, el algoritmo de Karmarkar, llevará a cabo una transformación del conjunto de soluciones factibles, de modo que la geometría de este conjunto se transforme en una como la que aparece en la Figura 2.3.

En este conjunto se llevarán a cabo las optimizaciones sucesivas de la función objetivo sobre la esfera inscrita. Antes de describir esta transformación que será llamada **Transformación Proyectiva**, damos en la siguiente sección, la formalización del concepto de puntos $x \in \mathbb{R}^n$ que cumplan (2.5).

2.2 El simplejo $\mathcal{S}^n$

En esta sección, definiremos el concepto de simplejo $\mathcal{S}^n$, el cual en base a la motivación hecha en **2.1** juega un papel importante en el algoritmo de Karmarkar. Antes recordemos la definición de cápsula convexa de un número finito de puntos.

Definición 2.1 *Sea $X = \{x^1, x^2, \ldots, x^m\}$ un conjunto de puntos en $\mathbb{R}^n$. La cápsula convexa de X, denotada por $\mathcal{C}(X)$, es:*

$$\mathcal{C}(X) := \{\sum_{i=1}^{m} \lambda_i x^i \ / \ \ x^i \in X, \lambda_i \geq 0, \sum_{i=1}^{m} \lambda_i = 1\}$$

Ahora, según Nikaido [15], se define:

Definición 2.2 *Un m-simplejo $\mathcal{S}$ $(0 \leq m < n)$, es la cápsula convexa de $m+1$ puntos en $\mathbb{R}^n$, llamados los vértices de $\mathcal{S}$.*

Ejemplo 2.1 *Se muestran a continuación, algunos m-simplejos*

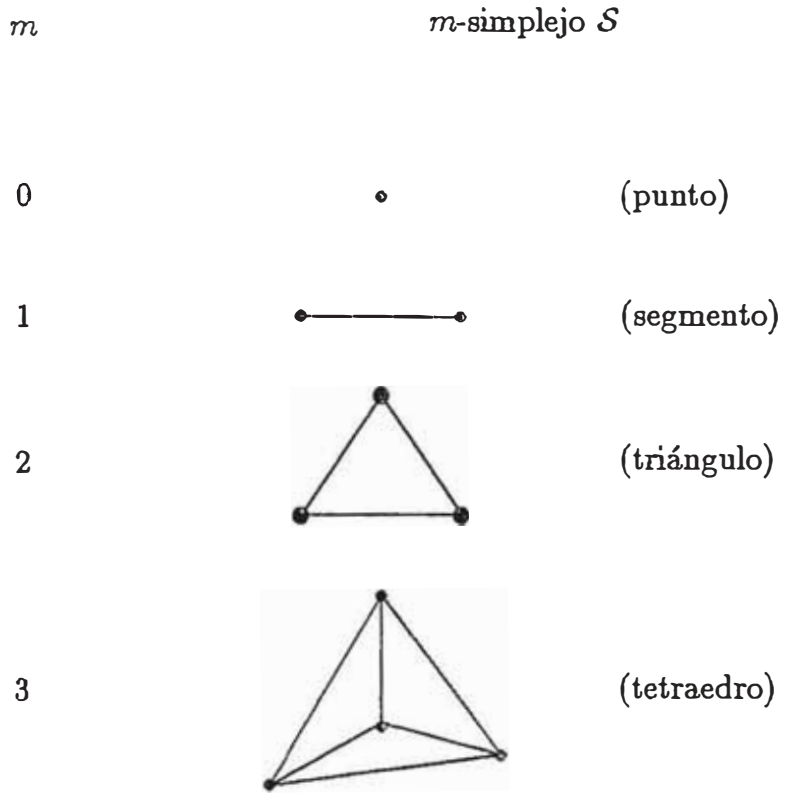


Figura 2.4:

Sea $X = \{e_1, e_2, \ldots, e_n\}$ donde e_i es el i-ésimo vector unitario en $\mathbb{R}^n$ (e_i es el vector cuyas componentes son todas nulas excepto el i-ésimo que es la unidad). Según Nikaido [15] se define:

Definición 2.3 *El simplejo $\mathcal{S}^n$ es la cápsula convexa de $X = \{e_1, e_2, \ldots, e_n\}$ es decir* $\mathcal{S}^n := \{x \in \mathbb{R}^n : \quad x \geq 0, \quad \sum_{i=1} x_i = 1\}$.

Ejemplo 2.2 *La Figura 2.5 ilustra el caso de $n = 3$*

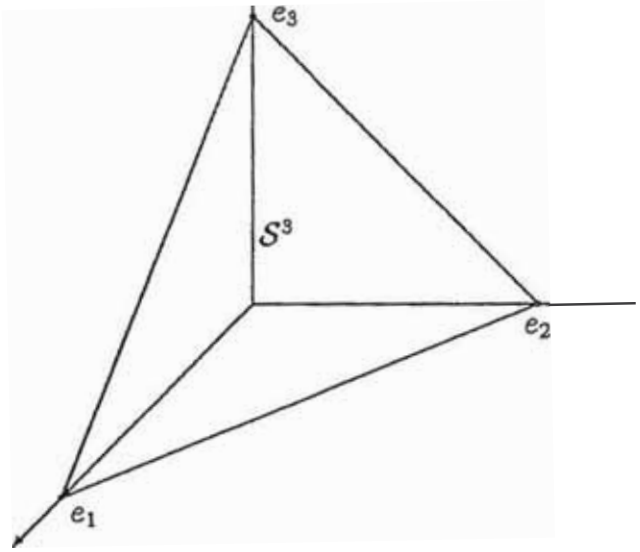


Figura 2.5: Caso $n = 3$

Definición 2.4 *El vector*

$$a^0 := \frac{1}{n}\begin{pmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{pmatrix}$$

perteneciente a $\mathcal{S}^n$ será llamado el centro de $\mathcal{S}^n$.

Lema 2.1 (Karmarkar [6]) *Sean*

$$\begin{aligned} \mathcal{S}^n &:= \{x \in \mathbb{R}^n : \quad x \geq 0, \quad e^T x = 1\}, \\ B(a^0; r) &:= \{x \in \mathbb{R}^n : \quad e^T x = 1, \quad \|x - a^0\|_2 \leq r\}, \\ B(a^0; R) &:= \{x \in \mathbb{R}^n : \quad e^T x = 1, \quad \|x - a^0\|_2 \leq R\}, \end{aligned}$$

donde $e^T = (1, 1, \ldots, 1)$.

Entonces para

$$r := \frac{1}{\sqrt{n(n-1)}}, \quad R = \sqrt{\frac{n-1}{n}},$$

se cumple:

$$B(a^0; r) \subset \mathcal{S}^n \subset B(a^0; R).$$

Prueba: Los vértices de $\mathcal{S}^n$ son los vértices unitarios e_i, y uno puede darse cuenta fácilmente que R es igual a la distancia del centro $a^0 := \frac{1}{n}e$ de $\mathcal{S}^n$ a cada vértice e_i, es decir:

$$R = \|\frac{1}{n}e - e_i\|_2 = \sqrt{\frac{n-1}{n}}.$$

Similarmente r es igual a la distancia del centro $a^0 := \frac{1}{n}e$ a el punto en el borde $(\frac{1}{n-1}, \frac{1}{n-1}, \cdots, \underbrace{0}_{i}, \cdots, \frac{1}{n-1})$

la cual es:

$$r = \sqrt{(\frac{1}{n} - \frac{1}{n-1})^2(n-1) + \frac{1}{n^2}} = \frac{1}{\sqrt{n(n-1)}} \quad \square$$

Geométricamente esto significa que el simplejo $\mathcal{S}^n$ tiene una bola inscrita y una bola circunscrita en el hiperplano:

$$H := \{x \in \mathbb{R}^n : \quad e^T x = 1\}.$$

Gráficamente:

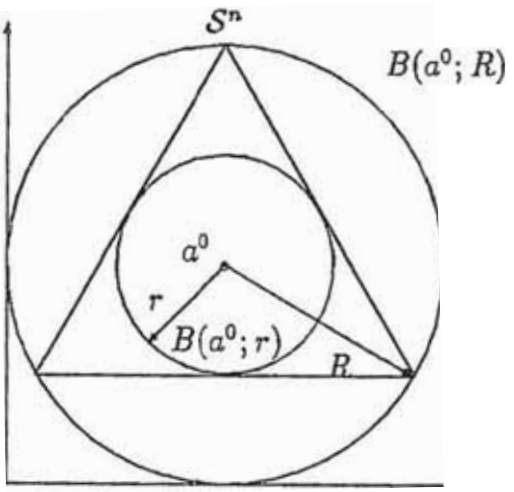


Figura 2.6: Esfera inscrita y circunscrita a $\mathcal{S}^n$

2.3 Aplicación de la transformación proyectiva

En esta sección describimos la transformación proyectiva. Esta transformación es uno de los procedimientos analíticos de mayor relevancia que el algoritmo de Karmarkar emplea y como veremos posteriormente en el Capítulo 3, será empleada para transformar el conjunto factible del problema de programación lineal sobre la intersección de un simplejo y un espacio afín.

Definición 2.5 (Karmárkar [6]) *Sea* $\mathcal{S}^n := \{x : \quad x > 0, \quad \sum_{i=1}^{n} x_i = 1\}$. *Sea, además,* $a = (a_1, a_2, \ldots, a_n) \in \mathcal{S}^n, \quad a_i > 0, \quad i = 1, \ldots, n$. *Consideremos la matriz diagonal* $D := diag(a_1, a_2, \ldots, a_n)$.

$$Por \qquad T: \begin{array}{ccc} \mathcal{S}^n & \mapsto & \mathcal{S}^n \\ x & \mapsto & y, \end{array}$$

$y = T(x;a) = \dfrac{D^{-1}x}{e^T D^{-1}x}$, *se define una transformación proyectiva, donde* $e^T = (1,\ldots,1)$. *Su relación inversa es dada por:*

$$T^{-1}: \mathcal{S}^n \longmapsto \mathcal{S}^n$$

$$y \longmapsto x$$

$$x = T^{-1}(y;a) = \frac{Dy}{e^T Dy}. \qquad \square$$

En forma vectorial la transformación proyectiva puede escribirse como:

$$T(x_1, x_2, \ldots, x_n)^T = \frac{1}{\sum_{i=1}^{n} \frac{x_i}{a_i}} \left(\frac{x_1}{a_1}, \frac{x_2}{a_2}, \ldots, \frac{x_n}{a_n}\right)^T$$

y la transformación inversa como:

$$T^{-1}(y_1, y_2, \ldots, y_n)^T = \frac{1}{\sum_{i=1}^{n} a_i y_i} \left(a_1 y_1, a_2 y_2, \ldots, a_n y_n\right)^T.$$

El siguiente lema resume las propiedades más importantes de esta transformación; y entre otras propiedades garantiza la existencia de la transformación inversa. El lema fue establecido por Karmarkar [6].

Lema 2.2 *La transformación proyectiva tiene las siguientes propiedades:*

(1) $T(a;a) = \dfrac{1}{n}e.$

(2) *Para* $x \neq x'$ *es* $T(x;a) \neq T(x';a)$, *es decir* T *es inyectiva.*

(3) $T(x;a) \in \mathcal{S}^n \quad \forall x \in \mathcal{S}^n.$

(4) T *es suryectiva, es decir para cualquier punto* $y \in \mathcal{S}^n$, *existe un único punto* $x \in \mathcal{S}^n$, *que satisface:*

$$y = T(x;a).$$

El punto x *se da mediante*

$$x = \frac{Dy}{e^T Dy}.$$

Si x e y satisfacen (4) escribiremos:

$$x = T^{-1}(y, a).$$

Prueba:

1. $T(x; a) = \dfrac{D^{-1}x}{e^T D^{-1}x}$, entonces

$$T(a; a) = \frac{D^{-1}a}{e^T D^{-1}a} = \frac{1}{n}e.$$

2. Supongamos que $T(x; a) = T(x'; a)$, es decir

$$\frac{D^{-1}x}{e^T D^{-1}x} = \frac{D^{-1}x'}{e^T D^{-1}x},$$

$$\text{entonces} \quad \frac{x}{e^T D^{-1}x} = \frac{x'}{e^T D^{-1}x'} \implies x = x'\left(\frac{e^T D^{-1}x}{e^T D^{-1}x'}\right),$$

$$\implies x = x'\left(\frac{\sum_{i=1}^{n}\left(\frac{x_i}{a_i}\right)}{\sum_{i=1}^{n}\left(\frac{x'_i}{a_i}\right)}\right),$$

$$\implies x = x'.$$

3. Sea $y = T(x; a)$.

$$\implies y_j = \frac{\frac{x_j}{a_j}}{\sum_{i=1}^{n}\frac{x_i}{a_i}}.$$

Es inmediato que $y_j > 0$, $\forall j = 1, \ldots, n$ ya que $x, a \in \mathcal{S}^n$.

Además:

$$\sum_{j=1}^{n} y_j = \sum_{j=1}^{n}\left(\frac{\frac{x_j}{a_j}}{\sum_{i=1}^{n}\frac{x_i}{a_i}}\right) = \frac{\sum_{j=1}^{n}\frac{x_j}{a_j}}{\sum_{i=1}^{n}\frac{x_i}{a_i}} = 1.$$

$$\implies y = T(x; a) \in \mathcal{S}^n$$

4. Sea $y \in \mathcal{S}^n$.

Definiendo $x := \dfrac{Dy}{e^T Dy}$,

$$x_j = \frac{a_j y_j}{\sum_{i=1}^{n} a_i y_i},$$

resulta $x_j \geq 0;\ \sum_{j=1}^{n} x_j = 1 \quad \Longrightarrow \quad x \in \mathcal{S}^n,\ y = \dfrac{D^{-1}x}{e^T D^{-1}x}$. $\square$

Observación: En la definición de transformación proyectiva se ha escrito $y = T(x; a)$ en lugar de sólo $y = T(x)$, para indicar la dependencia de la transformación respecto de a. El ejemplo siguiente pone esta dependencia en evidencia.

Ejemplo 2.3 *Sea* $\mathcal{S}^3 = \{(x_1, x_2, x_3) \in \mathbb{R}^3 \ / \ \ x_1 + x_2 + x_3 = 1, \quad x_1, x_2, x_3 \geq 0\}$. *El punto* $a = (\frac{1}{4}, \frac{3}{8}, \frac{3}{8})^T \in \mathcal{S}^3$ *dá la siguiente transformación proyectiva:*

$$T\Big((x_1, x_2, x_3); (\frac{1}{4}, \frac{3}{8}, \frac{3}{8})\Big) = \frac{1}{4x_1 + \frac{8}{3}x_2 + \frac{8}{3}x_3}\Big(4x_1, \frac{8}{3}x_2, \frac{8}{3}x_3\Big)$$

Por ejemplo para $(x_1, x_2, x_3) = (\frac{1}{3}, \frac{1}{3}, \frac{1}{3}) \in \mathcal{S}^3$ *se tiene:*

$$T\Big((\frac{1}{3}, \frac{1}{3}, \frac{1}{3}); (\frac{1}{4}, \frac{3}{8}, \frac{3}{8})\Big) = \frac{1}{28}(12, 8, 8)$$

Capítulo 3

Algoritmo de Karmarkar

3. Algoritmo de Karmarkar

En este capítulo presentamos el algoritmo de Karmarkar, pero antes (en las secciones 3.1, 3.2, 3.3) establecemos los resultados previos necesarios a fin de poder presentar el algoritmo de un modo más simple. Damos también la prueba de que el algoritmo es polinomial y presentamos los teoremas en los cuales se fundamenta el algoritmo.

3.1 Definición del problema

Para resolver un problema de programación lineal, Karmarkar considera que el problema está escrito en la forma siguiente:

$$\left.\begin{array}{rl} \min z = & c^T x \\ \text{sujeto a}: & Ax = 0, \\ & e^T x = 1, \\ & x \geq 0. \end{array}\right\} \quad (\mathbf{P})$$

donde: $A \in \mathbb{R}^{m \times n}$, $\text{ran}(A) = m$, $c, x \in \mathbb{R}^n, e = (1, 1, \ldots, 1)^T$.
Dos suposiciones fundamentales se asumen para el problema (P):

(A1) El punto $x^0 = (\frac{1}{n}, \cdots, \frac{1}{n})$ es factible para el problema (P).

(A2) El valor óptimo de la función objetivo del problema (P) es cero.

A primera vista la forma del problema de programación lineal (P) y las suposiciones (A1) y (A2) pueden parecer muy restrictivas. Sin embargo, como veremos posteriormente (Capítulo 4), cualquier problema de programación lineal puede ser llevado a la forma (P). Cuando un problema de programación lineal está escrito en la forma (P), se dice que está en la **forma normal de Karmarkar**.

3.2 Transformación proyectiva del problema lineal

En esta sección veremos cómo cambia el problema lineal (P), bajo la transformación proyectiva. Para ver esto, sea:

$$\begin{aligned} \mathcal{S}^n := & \quad \{x \in \mathbb{R}^n : \quad e^T x = 1, \quad x \geq 0\}, \\ a = & \quad (a_1, a_2, \ldots, a_n) \in \mathcal{S}^n, \quad a_i > 0 \quad i = 1, \ldots, n, \\ D := & \quad \text{diag}(a_1, a_2, \ldots, a_n). \end{aligned}$$

Consideremos la transformación proyectiva

$$\begin{aligned} T(x; a): \quad & \mathcal{S}^n \mapsto \mathcal{S}^n \\ & x \mapsto y = \frac{D^{-1}x}{e^T D^{-1} x}, \end{aligned}$$

la cual, por el **Lema 2.2** es biyectiva, y su relación inversa es dada por:

$$\begin{aligned} T^{-1}(y; a): \quad & \mathcal{S}^n \mapsto \mathcal{S}^n \\ & y \mapsto x = \frac{Dy}{e^T Dy}. \end{aligned}$$

El problema lineal (P) puede escribirse también como:

$$\begin{aligned} \min z = & \quad c^T x \\ \text{sujeto a}: & \quad x \in \Pi \end{aligned}$$

donde

$$\begin{cases} \Pi = \Omega \cap \mathcal{S}^n, \\ \Omega = \{x \in \mathbb{R}^n : \quad Ax = 0\}, \\ \mathcal{S}^n = \{x \in \mathbb{R}^n : \quad x \geq 0, \quad e^T x = 1\}. \end{cases}$$

Veamos ahora cómo cambia la región factible Π, bajo la transformación proyectiva. Si usamos la relación

$$x = \frac{Dy}{e^T Dy},$$

obtenemos:

$$Ax = A(\frac{Dy}{e^T Dy}) = 0 \qquad \Longleftrightarrow \qquad (AD)y = 0.$$

Luego Ω es transformado en

$$\Omega' = \{y \in \mathbb{R}^n : \quad (AD)y = 0\},$$

pero, observamos algo más:

$$y = \frac{D^{-1}x}{e^T D^{-1}x},$$

implica

$$y \geq 0 \qquad \text{y} \qquad e^T y = \frac{e^T D^{-1}x}{e^T D^{-1}x} = 1.$$

Entonces: $\mathcal{S}^n$ es transformado bajo la transformación proyectiva en

$$\mathcal{S}^n = \{y \in \mathbb{R}^n : \quad y \geq 0 \quad e^T y = 1\}.$$

Luego Π es transformado en:

$$T(\Pi) = \Pi' = \Omega' \cap \mathcal{S}^n.$$

Ahora veamos cómo cambia la función objetivo $c^T x$. Ponemos:

$$x = \frac{Dy}{e^T Dy},$$

entonces:

$$c^T x = \frac{c^T Dy}{e^T Dy} = \frac{(Dc)^T y}{e^T Dy}.$$

Luego, cuando aplicamos la transformación proyectiva $T(x;a)$ a el problema lineal **(P)** este es transformado en:

$$\min w(y) = \frac{(Dc)^T y}{e^T Dy},$$

$$\left.\begin{array}{ll} \text{sujeto a}: & (AD)y = 0, \\ & e^T y = 1, \\ & y \geq 0. \end{array}\right\} \qquad \textbf{(PLF)}$$

Presentamos a continuación dos lemas que establecen relaciones entre las soluciones de los problemas lineales **(P)** y **(PLF)**.

Lema 3.1 (S. Mehrotra[11]) *$y = T(x;a)$ es factible para* **(PLF)** *si y sólo si x es factible para* **(P)**.

Prueba:

$\Longrightarrow$) Si $y = T(x;a) = \dfrac{D^{-1}x}{e^T D^{-1}x}$ es factible para (**PLF**), entonces

$$\begin{cases} (AD)y &= 0, \\ e^T y &= 1, \\ y &\geq 0. \end{cases}$$

De $(AD)y = 0$ tenemos

$$(AD)(\frac{D^{-1}x}{e^T D^{-1}x}) = 0 \qquad \Longleftrightarrow \qquad Ax = 0,$$

y por la transformación inversa $x = \dfrac{Dy}{e^T Dy}$, se tiene,

$$x \geq 0 \qquad \text{y} \qquad e^T x = \frac{e^T Dy}{e^T Dy} = 1.$$

Entonces x es factible para (**P**).

$\Longleftarrow$) Esto se probó cuando se vió la transformación del problema (**P**) en el problema (**PLF**). □

Lema 3.2 (S. Mehrotra[11]) *$y^* = T(x^*;a)$ es una solución óptima para* (**PLF**) *si y sólo si x^* es una solución óptima para* (**P**). *Además $w^* = w(y^*) = 0$.*

Prueba: Consecuencia inmediata del **Lema 3.1**. □

El siguiente lema permite **linealizar** la función objetivo del problema lineal (**PLF**).

Lema 3.3 (S. Mehrotra[11]) *y^* es solución óptima para* (**PLF**) *si y sólo si y^* es solución óptima del problema lineal*

$$\begin{aligned} \min v(y) \;&=\; (Dc)^T y, \\ \textit{sujeto a:} \quad & (AD)y = 0, \\ & e^T y = 1, \\ & y \geq 0. \end{aligned}$$

Además $v^* = v(y^*) = 0$.

Prueba: Por el **Lema 3.2**, si y^* es óptimo para el problema lineal **(PLF)** entonces $w(y^*) = 0$, pero

$$w(y^*) = \frac{(Dc)^T y^*}{e^T D y^*} = 0 \iff (Dc)^T y^* = 0,$$

es decir $v^* = v(y^*) = 0$.

Además:

$$(AD)y^* = 0 \quad \wedge \quad e^T y^* = 1, \quad y^* \geq 0.$$

Esto implica que y^* es óptimo de

$$\begin{aligned} \min v(y) &= (Dc)^T y, \\ \text{sujeto a :} \quad & (AD)y = 0, \\ & e^T y = 1, \\ & y \geq 0. \end{aligned} \qquad \square$$

3.3 Optimización sobre una esfera

Esta sección está basada en el desarrollo hecho en el Capítulo 2 (**Sección 2.1**), en el cual se vió que era posible construir un algoritmo eficiente para el problema de programación lineal, al optimizar la función objetivo $c^T x$, sobre un conjunto más reducido el cual recordemos que era la esfera inscrita en el conjunto de soluciones factibles. Brevemente recordemos que en la **Sección 2.1** consideramos el problema de programación lineal:

$$\begin{aligned} &\min \quad c^T x. \\ &\text{sujeto a :} \quad \left.\begin{aligned} Ax &= b, \\ x &\geq 0, \end{aligned}\right\} \textbf{(PL)} \end{aligned}$$

donde $A \in \mathbb{R}^{m \times n}$, $c, x \in \mathbb{R}^n$, $b \in \mathbb{R}^m$. Sea $\mathcal{P} = \{x \in \mathbb{R}^n : \quad Ax = b, \quad x \geq 0\}$, el conjunto de soluciones factibles de este problema **(PL)**. Sea además E una esfera inscrita en $\mathcal{P}$ de centro x^0 y radio r y E' una esfera circunscrita a $\mathcal{P}$ de centro x^0 y radio R. Luego, en la **Sección 2.1**, resolvimos:

$$\begin{aligned} &\min \quad c^T x. \\ &\text{sujeto a} \quad x \in E. \end{aligned}$$

y obtuvimos que:

$$c^T x^B - c^T x^p \leq (1 - \frac{1}{\gamma})(c^T x^0 - c^T x^p),$$

donde $c^T x^p$ es el valor óptimo y $\gamma = \frac{R}{r}$.

Tal como mencionamos en su oportunidad, si uno quiere tener la seguridad que en un cierto número de iteraciones, el algoritmo se acercará suficientemente al óptimo, entonces necesitamos estimar a priori el valor de γ, el cual depende de los radios de la esfera circunscrita e inscrita al poliedro $\mathcal{P}$. Ahora, en el caso general es dificil trabajar directamente con el problema de programación lineal (**PL**), ya que γ cambiará con cada problema concreto que se considere. Aún cuando es teóricamente posible encontrar para un poliedro dado $\mathcal{P}$ el radio de una esfera inscrita y circunscrita a él, este es un problema relativamente complicado. Karmarkar probó que todo problema de programación lineal (**PL**) puede convertirse a la forma (**P**) con una ventaja: Cuando el problema está en la forma (**P**) es fácil definir una esfera inscrita y circunscrita al conjunto de soluciones factibles, cuyos radios sólo dependan del número de variables n y de ningún otro parámetro, lo cual no ocurre en el caso del problema (**PL**). Además al estar el problema en la forma (**P**) se podrá probar la convergencia polinomial del algoritmo y ciertos aspectos relacionados con él.

En la Sección 3.2, vimos que después de aplicar la transformación proyectiva al problema (**P**) este se transforma en el problema de programación lineal equivalente ($\tilde{P}$):

$$\begin{array}{rrcl} \min & (Dc)^T y. & & \\ & (AD)y & = & 0, \\ \text{sujeto a:} & e^T y & = & 1, \\ & y & \geq & 0. \end{array} \Bigg\} (\tilde{P})$$

Ahora apliquemos las ideas de la **Sección 2.1** a este problema de programación lineal.

Consideremos una restricción del problema de programación lineal ($\tilde{P}$). Para definir esta restricción haremos uso del **Lema 2.1**, en donde para el simplejo $\mathcal{S}^n$ encontramos $B(a^0, \alpha r)$, la esfera inscrita en $\mathcal{S}^n$ de centro a^0 y radio αr, $\alpha \in \langle 0, 1\rangle$, $r = \frac{1}{\sqrt{n(n-1)}}$.

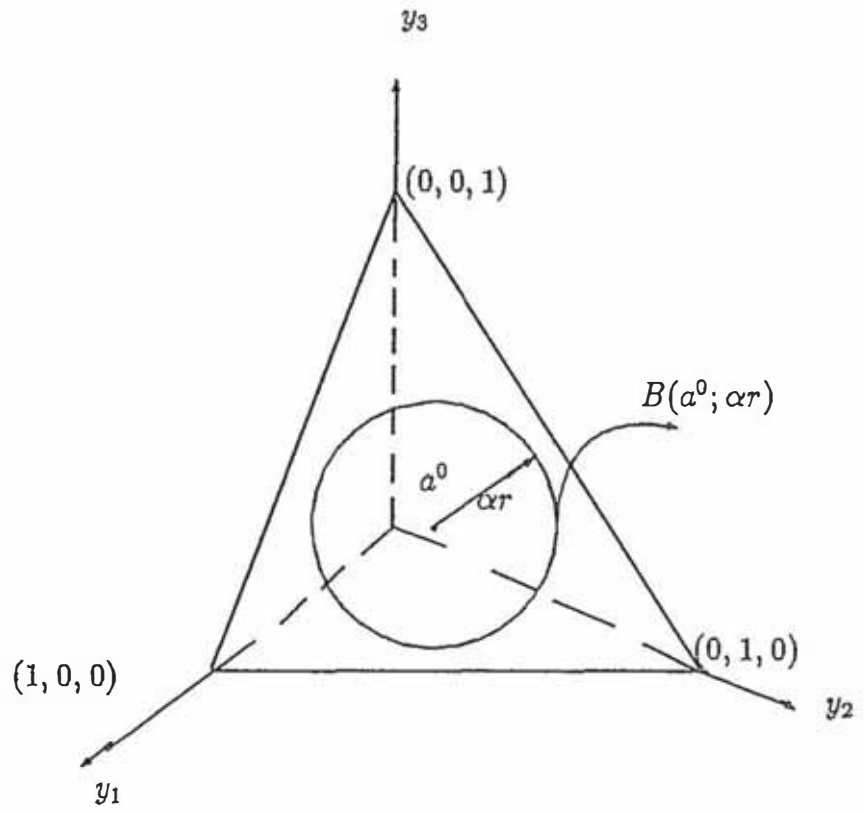


Figura 3.1: Esfera inscrita en $\mathcal{S}^2$

Ahora consideremos el problema restringido:

$$\begin{array}{rl} \min & (Dc)^T y. \\ & \left.\begin{array}{r} (AD)y = 0, \\ e^T y = 1, \\ y \geq 0, \\ y \in B(a^0; \alpha r). \end{array}\right\} \textbf{(PLB)} \\ \text{sujeto a :} & \end{array}$$

La especificación del valor del parámetro α será discutida posteriormente. Desde que $y \geq 0$ está implicado por la intersección de la esfera y la restricción $\{y \in \mathbb{R}^n : \quad e^T y = 1\}$

el problema **(PLB)** es equivalente al problema

$$\left.\begin{array}{rl} \min & (Dc)^T y. \\ & (AD)y = 0, \\ \text{sujeto a :} & e^T y = 1, \\ & y \in B(a^0; \alpha r). \end{array}\right\} (\overline{\mathbf{PLB}})$$

La siguiente Figura 3.2, ilustra la región factible sobre el simplejo 3-dimensional.

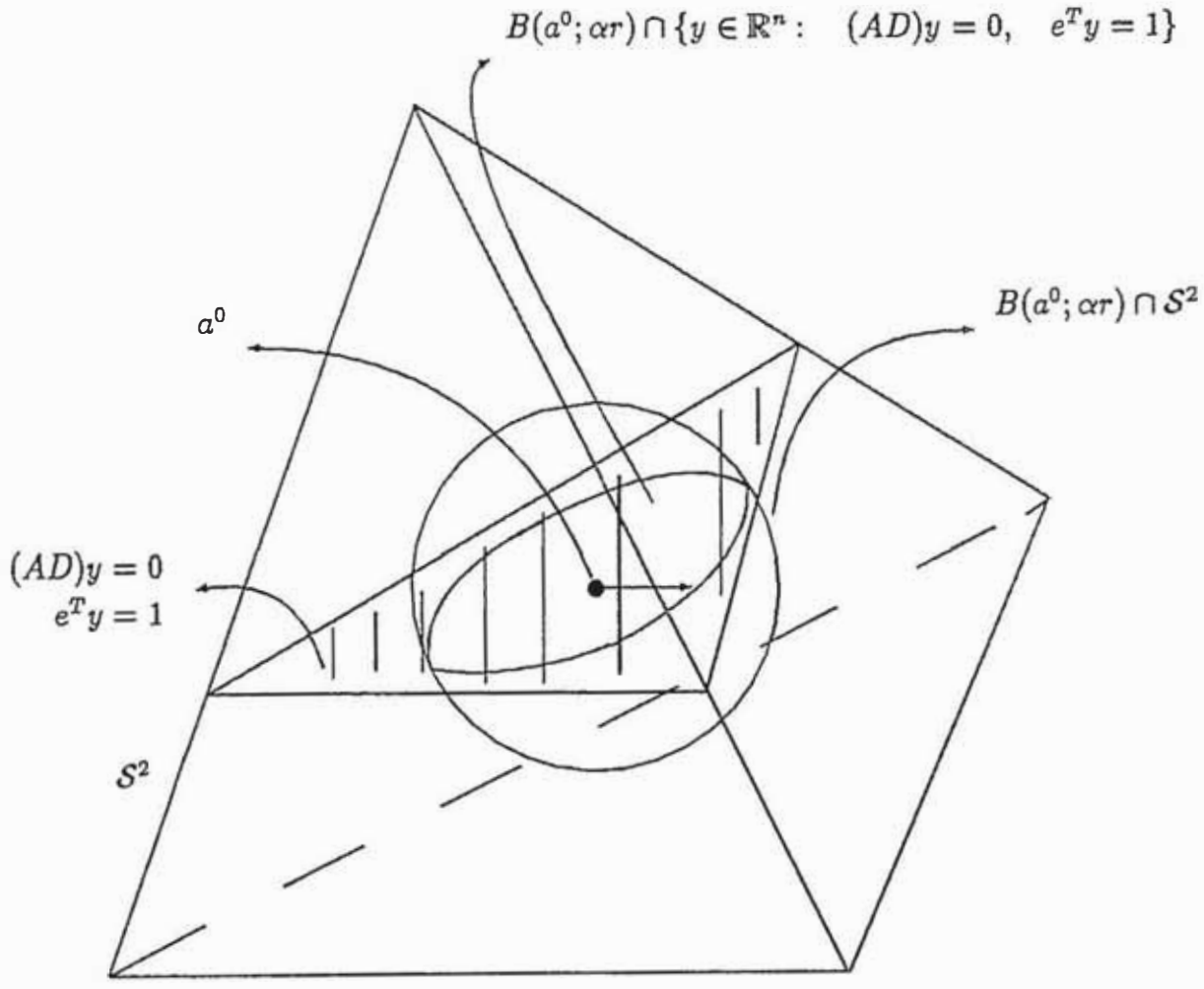


Figura 3.2: Región factible sobre $\mathcal{S}^2$

Ahora debemos resolver el problema $(\overline{\mathbf{PLB}})$. Pero para hacer esto, debemos establecer dos lemas previos. Definamos

$$B := \begin{pmatrix} AD \\ e^T \end{pmatrix},$$

sabemos que $\text{ran}(A) = m$, D es una matriz diagonal con n componentes positivas, y e^T es un vector fila de n unos. El siguiente lema, establece que BB^T es invertible.

Lema 3.4 (Karmarkar [6]) *Sea*

$$B := \begin{pmatrix} AD \\ e^T \end{pmatrix},$$

entonces BB^T es regular.

Prueba: Vamos a demostrar que $\text{Nuc}(B^T) = \{0\}$.

$$\begin{aligned} &\text{Sea} \quad 0 = B^T \begin{pmatrix} z \\ z_{m+1} \end{pmatrix} = DA^T z + z_{m+1} e, \\ &\Longrightarrow \quad 0 = ADA^T z + z_{m+1} Ae \end{aligned}$$

Por (A1) tenemos $Ae = 0$, entonces

$$\begin{aligned} & 0 = ADA^T z, \\ \Longrightarrow \quad & z = 0 \quad (\text{por } \text{ran}(A) = m) \quad \text{y} \quad z_{m+1} = 0 \\ \Longrightarrow \quad & \text{ran}(B) = m + 1 \\ \Longrightarrow \quad & BB^T \quad \text{es regular.} \qquad \square \end{aligned}$$

Consideremos el vector $c_p = (I - B^T(BB^T)^{-1}B)Dc$, el cual es la proyección ortogonal del vector Dc en el espacio nulo de la matriz B.
El siguiente lema establece una conclusión muy importante, acerca de la solución del problema de programación lineal $(\mathbf{P})$ si $c_p = 0$.

Lema 3.5 (Karmarkar [6]) *Si $c_p = 0$, entonces cualquier $x \in \Omega \cap \mathcal{S}^n$ es solución óptima del problema de programación lineal $(\mathbf{P})$.*

Prueba: Si $c_p = 0$, entonces $Dc \in \mathrm{Im}(B^T)$. Luego, existen $z \in \mathbb{R}^m$, $z_{m+1} \in \mathbb{R}$ con

$$\begin{aligned} Dc &= DA^T z + z_{m+1} e, \\ c &= A^T z + z_{m+1} D^{-1} e. \end{aligned}$$

Si $x^* \in \Omega \cap \mathcal{S}^n$ fuese la solución óptima del problema de programación $(\mathbf{P})$, entonces se cumpliría

$$0 = c^T x^* = (Ax^*)^T z + z_{m+1} e^T D^{-1} x^*$$

Por

$$Ax^* = 0 \quad \text{y} \quad e^T D^{-1} x^* > 0,$$

resultaría

$$0 = c^T x^* = z_{m+1} e^T D^{-1} x^*,$$

entonces

$$z_{m+1} = 0.$$

De $c = A^T z$ se obtiene que para cualquier $x \in \Omega \cap \mathcal{S}^n$ se tiene $c^T x = (Ax)^T z = 0$. □

Damos a continuación el algoritmo para resolver el problema $(\mathbf{PLB})$, podemos asumir por el lema anterior que $c_p \neq 0$.

Algoritmo A

1. Proyecte (Dc) ortogonalmente en el espacio nulo de B.

$$c_p = (I - B^T(BB^T)^{-1}B)Dc.$$

2. Normalice c_p:

$$\widehat{c_p} := \frac{c_p}{\|c_p\|_2}.$$

3. Tomar un paso de longitud αr en la dirección $-\widehat{c_p}$

$$b' = a^0 - \alpha r \widehat{c_p}.$$

Teorema 3.6 *El punto b' retornado por el* **algoritmo A** *maximiza $(Dc)^T y$ sobre $B(a^0; \alpha r) \cap \{y \in \mathbb{R}^n : \quad (AD)y = 0, \quad e^T y = 1\}$, $\alpha \in \langle 0, 1\rangle$.*

Prueba: Primeramente mostramos que b' es factible para $\overline{\mathbf{PLB}}$.

Obviamente $Bc_p = 0$ y por esto tenemos que: $\quad b' = a^0 - \alpha r \hat{c}_p$

$$\begin{aligned}(AD)(a^0 - \alpha r\hat{c}_p) &= ADa^0 - \alpha r AD\frac{c_p}{\|c_p\|_2} \\ &= 0 - 0 = 0\end{aligned}$$

Además:

$$\begin{aligned}e^T b' &= e^T(a^0 - \alpha r\hat{c}_p) \\ &= e^T\frac{e}{n} - \alpha r\frac{e^T c_p}{\|c_p\|_2} \\ &= 1 - 0 = 1.\end{aligned}$$

Luego b' es factible para $(\overline{\mathbf{PLB}})$.

Ahora para cualquier $z \in B(a^0; \alpha r) \cap \{y \in \mathbb{R}^n : \quad (AD)y = 0, \quad e^T y = 1\}$
se obtiene debido a: $B(b' - z) = 0$ lo siguiente:

$$\begin{aligned}(Dc)^T(b' - z) &= (c_p + B^T(BB^T)^{-1}BDc)^T(b' - z) \\ &= c_p^T(b' - z),\end{aligned}$$

$$\text{entonces} \quad (Dc)^T(b' - z) = c_p^T(b' - z) \tag{3.1}$$

$$\begin{aligned}\text{Por} \quad c_p(b' - z) &= \|c_p^T\|_2\hat{c}_p^T(a^0 - \alpha r\hat{c}_p - z) \\ &= \|c_p^T\|_2(\hat{c}_p^T(a^0 - z) - \alpha r),\end{aligned}$$

$\hat{c}_p^T(a^0 - z) \leq \|\hat{c}_p\|_2\|a^0 - z\|_2 \leq \alpha r$ y $z \in B(a^0; \alpha r)$ obtenemos en (3.1);

$$\begin{aligned}c_p^T(b' - z) \leq 0 &\implies (Dc)^T(b' - z) \leq 0 \\ &\implies (Dc)^T b' \leq (Dc)^T z \\ &\quad \forall z \in B(a^0; \alpha r) \cap \{y \in \mathbb{R}^n : \quad (AD)y = 0, \quad e^T y = 1\}, \quad \alpha \in \langle 0, 1\rangle.\end{aligned}$$

Hasta aquí hemos empleado sólo el argumento sobre la esfera inscrita al simplejo. Veremos el rol que desempeña la esfera circunscrita al simplejo, y finalmente veremos cómo el argumento presentado en la **Sección 2.1** funciona bien y permite resolver el problema $(\tilde{P})$ eficientemente. Primero consideremos una **relajación** del problema $(\tilde{P})$. La construcción

del problema de relajación requerido, considera la esfera circunscrita $B(a^0; R)$ al simplejo $\mathcal{S}^n$, con $R = \sqrt{\frac{n-1}{n}}$ (**Lema 2.1**). Ahora consideremos el problema obtenido añadiendo la restricción (redundante) $y \in B(a^0; R)$ en el problema $(\tilde{P})$, pero relajando las restricciones de no–negatividad:

$$\left.\begin{array}{rrl} \min & (Dc)^T y. & \\ & (AD)y = 0, & \\ \text{sujeto a :} & e^T y = 1, & \\ & y \in B(a^0; R). & \end{array}\right\} (\widehat{PLR})$$

Por el **algoritmo A**, la solución de este problema es dada por:

$$\bar{b} = a^0 - \frac{Rc_p}{\|c_p\|_2}. \tag{3.2}$$

(La prueba en el **teorema 3.6** no depende del radio de la esfera).

La siguiente figura ilustra la relajación sobre la esfera circunscrita.

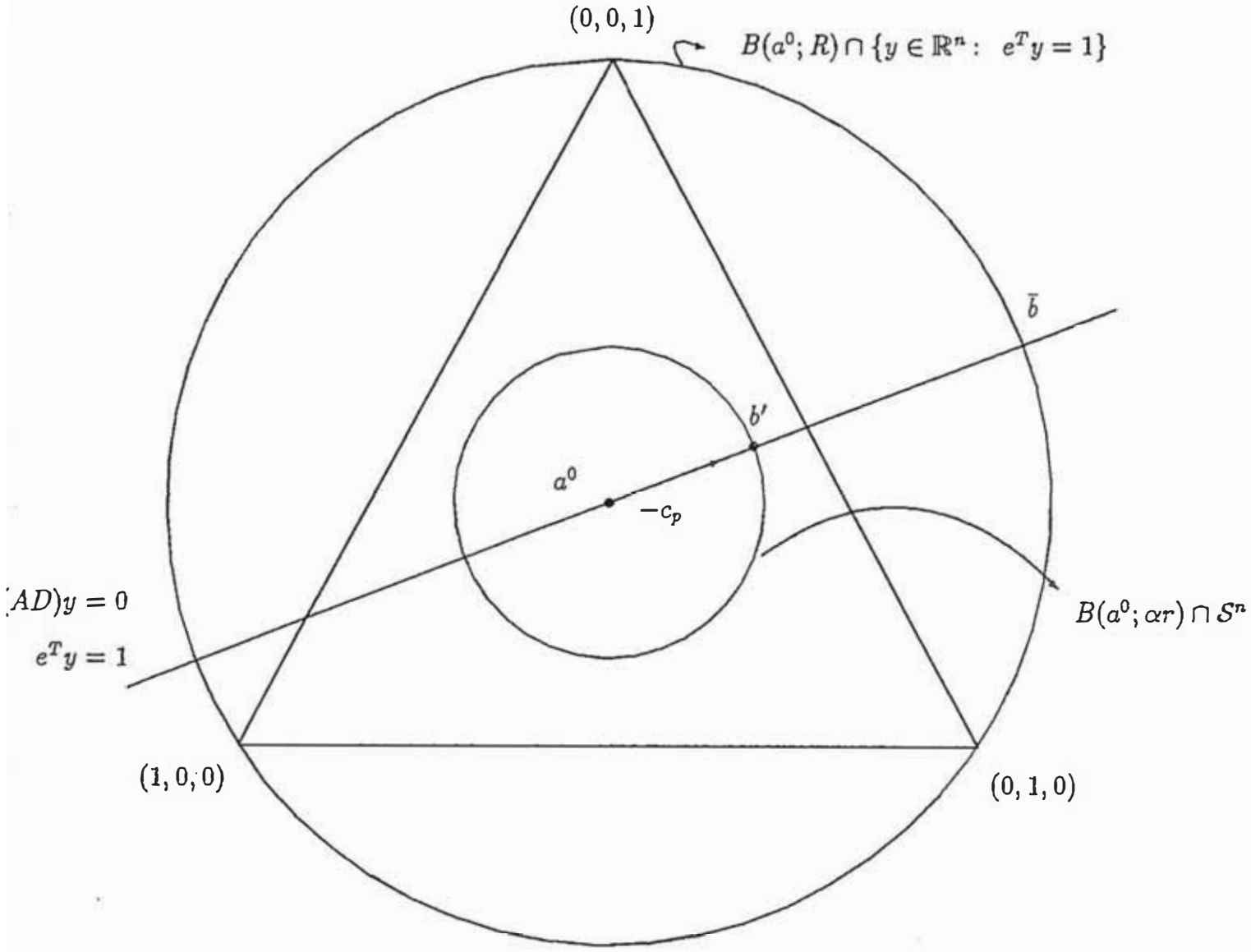


Figura 3.3: Relajación sobre la esfera circunscrita.

Uno puede ahora obtener una estimación, para medir el progreso con respecto a la función objetivo $(Dc)^T y$ como sigue: primero por el **Teorema 3.6**, nótese que:

$$(Dc)^T b' < (Dc)^T a^0.$$

Denotamos con y^* la solución óptima del problema $(\bar{P})$. Desde que el problema $(\overline{\mathbf{PLB}})$ (resuelto por b') es una restricción de este problema y el problema $\widehat{PLR}$ (cuya solución es

$\bar{b}$) es una relajación de este problema, tenemos:

$$(Dc)^T\bar{b} \leq (Dc)^T y^* \leq (Dc)^T b' < (Dc)^T a^0.$$

Por otra parte, usando (3.2) y por el hecho de que $c_p = (I - B^T(BB^T)^{-1}B)Dc$, obtenemos:

$$\begin{aligned} 0 < (Dc)^T(a^0 - b') \leq (Dc)^T(a^0 - y^*) &\leq (Dc)^T(a^0 - \bar{b}) \\ &= \frac{R(Dc)^T}{\|c_p\|_2} c_p \\ &= \frac{R}{\alpha r}(Dc)^T(a^0 - b'), \end{aligned}$$

luego:

$$\begin{aligned} (Dc)^T(a^0 - y^*) &\leq \frac{R}{\alpha r}(Dc)^T(a^0 - b') \\ &= \frac{R}{\alpha r}\left[(Dc)^T(a^0 - y^*) - (Dc)^T(b' - y^*)\right]. \end{aligned}$$

Resolviendo para $(Dc)^T(b' - y^*)$ en términos de $(Dc)^T(a^0 - y^*) > 0$, obtenemos:

$$\frac{(Dc)^T(b' - y^*)}{(Dc)^T(a^0 - y^*)} \leq 1 - \frac{\alpha r}{R} \equiv 1 - \frac{\alpha}{n-1}. \tag{3.3}$$

Note que bajo la suposición (A2) tenemos: $(Dc)^T y^* = 0$, así:

$$\frac{(Dc)^T b'}{(Dc)^T a^0} \leq 1 - \frac{\alpha}{n-1}. \tag{3.4}$$

De este modo en el punto b' el valor de la función objetivo, comparado con el valor de la función objetivo en el centro $a^0 := \frac{1}{n}e$, es reducido por un factor menor o igual a $1 - \frac{\alpha}{n-1}$. De acuerdo a lo desarrollado en la **Sección 2.1** tenemos que la distancia al valor óptimo, disminuye en el factor $1 - \frac{1}{\gamma} = 1 - \frac{\alpha}{n-1}$, nótese además cómo el valor de γ es sólo función de n (ya que α tomará un valor determinado), y de este modo hemos podido probar cómo las ideas de la **Sección 2.1** funcionan bien en el caso del problema $(\tilde{P})$. Nótese que tenemos a priori una estimación de γ al estar el problema en la forma $(\tilde{P})$.

Ahora, si aplicamos a

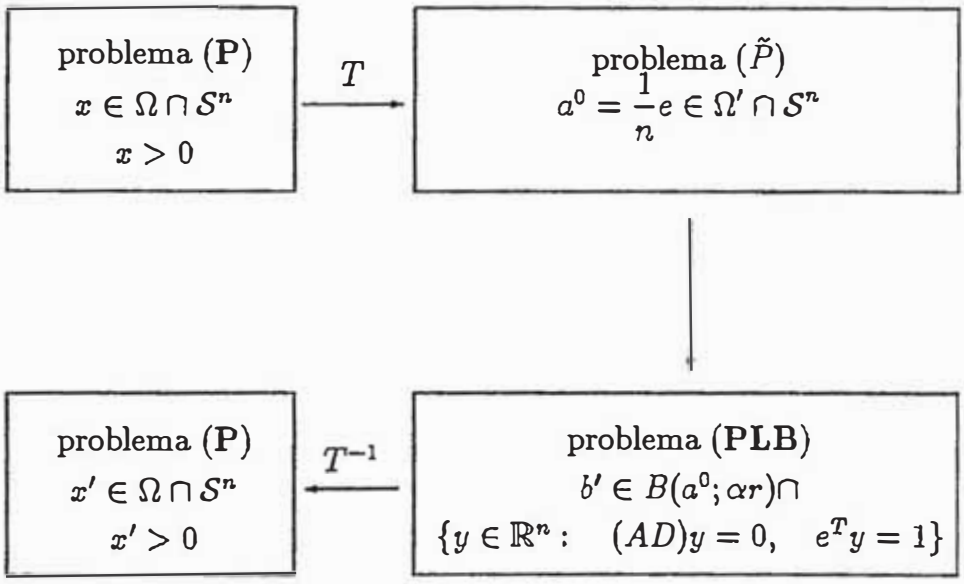
$$b' = a^0 - \alpha r \frac{c_p}{\|c_p\|_2},$$

la transformación proyectiva inversa $T^{-1}(b')$ obtenemos el punto $x' = T^{-1}(b')$ como:

$$x' = \frac{Db'}{e^T Db'}.$$

Observe que $b' > 0$, porque está en el interior de la esfera inscrita al simplejo $\mathcal{S}^n$, entonces es también $x' > 0$. x' corresponderá al segundo punto de la secuencia generada en el conjunto original. Ahora x' servirá como punto base para definir la siguiente transformación proyectiva. De este modo podemos definir un **proceso iterativo**, el cual crea en cada iteración un nuevo punto factible estrictamente positivo, con el valor de la función objetivo en este punto, menor que el valor objetivo en el punto anterior por un factor cercano a $1 - \frac{\alpha}{n-1}$.

El siguiente organigrama, muestra cómo pasar de una solución factible a la siguiente solución factible



3.4 Descripcion del algoritmo de Karmarkar

En esta sección, usaremos el trabajo hecho en las secciones anteriores a fin de presentar el algoritmo de Karmarkar de un modo compacto. Buscamos una solución del

problema (P) descrito en la Sección 3.1. El algoritmo de Karmarkar generará una secuencia $x^1, x^2, \ldots, x^k, \ldots$ de puntos positivos, estrictamente factibles x^k. Con cada punto x^k está asociada una matriz diagonal D_k definida mediante:

$$D_k := \text{diag}(x_1^k, x_2^k, \ldots, x_n^k).$$

Sea $\alpha \in \langle 0, 1\rangle$ (su elección será discutida posteriormente). Sean

$$L = \left[\!\left[\sum_{j=1}^{n}\sum_{i=1}^{m}\log_2(|a_{ij}|+1)\right]\!\right] + \left[\!\left[\sum_{j=1}^{n}\log_2(|c_j|+1)\right]\!\right] + [\![\log_2(n+1)]\!] + [\![\log_2(m+1)]\!]$$
$$+mn+n+m+1$$

y

$$r = \frac{1}{\sqrt{n(n-1)}}$$

Algoritmo de Karmarkar:

Paso 0: Dado $x^0 := a^0 := \frac{1}{n}e$, $k := 0$, $r := \frac{1}{\sqrt{n(n-1)}}$, L, $\alpha \in \langle 0, 1\rangle$,

$\beta := \sqrt{\frac{n}{n-1}}\alpha < 1.$

Paso 1: Calcule:

$$D_k := \text{diag}(x_1^k, x_2^k, \ldots, x_n^k),$$

$$B_k := \begin{pmatrix} AD_k \\ e^T \end{pmatrix}$$

$$c_p^k := (I - B_k^T(B_kB_k^{-1})B_k)D_kc$$

Si $c_p^k = 0$,

entonces $x^* = x^k$ es la solución óptima.

De lo contrario ir al paso 2.

Paso 2: Calcular

$$y^{k+1} := \frac{1}{n}e - \alpha r\frac{c_p^k}{\|c_p^k\|_2}$$
$$x^{k+1} := \frac{D_ky^{k+1}}{e^TD_ky^{k+1}}.$$

Paso 3: Si $c^T x^{k+1} > 2^{-L}$,

entonces $k := k + 1$. Ir al paso 1.

De lo contrario x^{k+1} es la solución.

Paso 4: $x^* = x^{k+1}$.

Observación: Para propósitos teóricos el algoritmo termina cuando

$$c^T x^{k+1} < 2^{-L},$$

pero para propósitos prácticos, el algoritmo finaliza cuando el valor de la función objetivo es suficientemente cercano a cero. El fundamento teórico para este criterio de terminación ($c^T x^{k+1} < 2^{-L}$) se encuentra en el lema 3.8.

Antes de presentar este lema probaremos que el conjunto factible del problema de programación lineal en la forma normal de Karmarkar está acotado.

Lema 3.7 *Sea* $M := \{x \in \mathbb{R}^n / Ax = b, \quad x \geq 0\} \neq \phi$. *Entonces* M *es acotado si y solo si no existe un vector* $d \in \mathbb{R}^n$, $d \neq 0$ *con* $Ad = 0$, $d \geq 0$

Prueba:

($\Longleftarrow$) $d \neq 0$, $d \geq 0$, $Ad = 0 \quad \Longrightarrow \quad Ax + td \in M$ para cada $x \in M$ y para todo $t \in \mathbb{R}$, luego M no está acotado.

($\Longrightarrow$) si M no es acotado entonces existe una sucesión $\{x_k\} \subset M$, $\|x_k\|_2 \to \infty$, definamos $d_k := \dfrac{x_k}{\|x_k\|_2}$, entonces existe una subsucesión $\{d_{k_j}\}$ con $d_{k_j} \to d$, $d \geq 0$, $Ad = 0$. $\square$

Este lema puede ser aplicado a un problema de programación lineal en la forma normal de Karmarkar para lo cual basta definir

$$\hat{A}x := \begin{pmatrix} A \\ e^T \end{pmatrix} x = \begin{pmatrix} 0 \\ 1 \end{pmatrix},$$

entonces por el lema anterior no existe $x \geq 0$ con $\hat{A}x = 0$.

Ahora tenemos el siguiente lema.

Lema 3.8 (ver **C. Gonzaga** [4]) *Consideremos el problema de programación lineal* (**PL**). *Si el conjunto factible está acotado, entonces para todo punto factible x, se cumple:*

$$c^T x \leq 2^L$$

.

Prueba: Sea x un vértice del conjunto factible. El sistema $Ax = b$ puede particionarse como

$$A_B x_B + A_N x_N = b \quad ,$$

donde x_B es un vector cuyas componentes son las componentes positivas de x y A_B es la matriz con las correspondientes columnas de A. Entonces $x_N = 0$ y el sistema puede escribirse como $A_B x_B = b$.

Si x es un vértice entonces las columnas de A_B son linealmente independientes. Asumiendo que A_B es cuadrada (caso contrario las ecuaciones redundantes pueden eliminarse). Entonces la solución del sistema es dada por la regla de Cramer,

$$x_i = \frac{\Delta_i}{\Delta} \quad , \quad \text{para} \quad i \in B,$$

donde $\Delta = det(A_B)$ y Δ_i es el determinante obtenido por sustitución de b por la columna A_i. Sean l_1 y l_c, el número total de bits usado por $\{A, b\}$ y por c respectivamente. Desde que $\lfloor \Delta_i \rfloor \leq 2^{l_1}$ y $\Delta \neq 0$, $x_i \leq 2^{l_1}$. De aqui sigue que $c_i x_i \leq 2^{l_1 + l_c} = 2^l$,
$c^T x \leq 2^l \leq 2^L$. Esto establece el teorema para todos los vértices del conjunto factible. Como el conjunto factible esta acotado, cualquier punto factible puede ser expresado como una combinación convexa $\sum_{j \in J} \lambda_j x^j$ de vertices, donde $\lambda_j \geq 0$ para $j \in J$ y $\sum_{j \in J} \lambda_j = 1$. Se sigue que $c^T x = \sum_{j \in J} \lambda_j c^T x^j \leq 2^L \sum_{j \in J} \lambda_j = 2^L$, completando la prueba. □

Usaremos este lema en la prueba del teorema **3.12**, donde estableceremos el comportamiento polinomial del algoritmo. La siguiente figura muestra cómo pasar de la iteración x^k a la siguiente x^{k+1}.

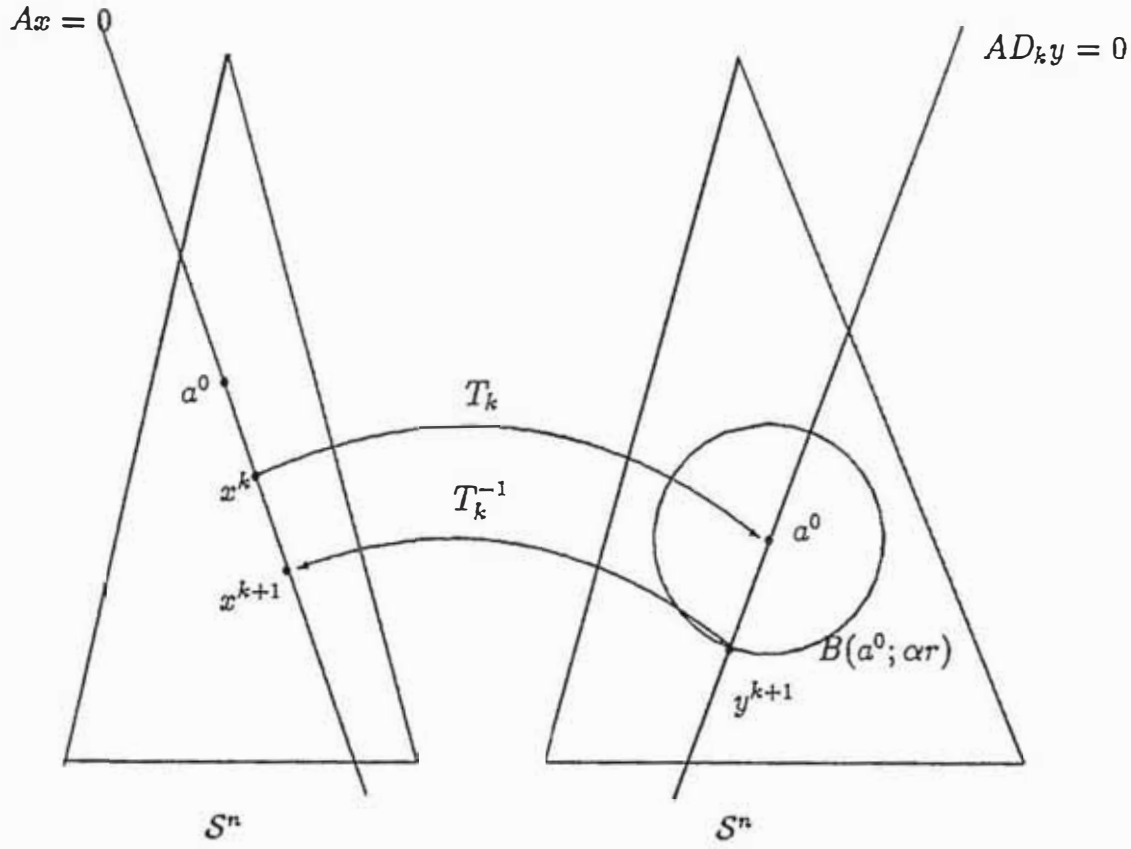


Figura 3.4: Construcción de x^{k+1} a partir de x^k.

Ejemplo 3.1 *Resolver*

$$\begin{aligned} &\min \quad x_2 \\ &\textit{sujeto a}: \\ &\qquad x_1 + x_2 - 2x_3 = 0, \\ &\qquad x_1 + x_2 + x_3 = 1, \\ &\qquad x_1 \;,\; x_2 \;,\; x_3 \geq 0. \end{aligned}$$

Solución: En este caso $n = 3$, $m = 1$, $A = [11 - 2]$, $c = (0, 1, 0)^T$
El punto $x^0 = (1/3, 1/3, 1/3)^T$ es factible. La solución óptima es $(2/3, 0, 1/3)^T$, el valor de la función objetivo es cero, las hipótesis del algoritmo de Karmarkar se cumplen.

Iteración 1: Partimos del punto $x^0 := (\frac{1}{3}, \frac{1}{3}, \frac{1}{3})^T$.

Obtenemos $D_1 = \text{diag}(\frac{1}{3}, \frac{1}{3}, \frac{1}{3})$. Como $e^T D_1^{-1} x = \sum_{i=1}^{3} 3x_i = 3$, la transformación proyectiva resulta en

$$T(x) = \frac{D_1^{-1} x}{e^T D^{-1} x} = x.$$

$$D_1 c \implies D_1 c = \begin{pmatrix} 0 \\ 1/3 \\ 0 \end{pmatrix}$$

$$AD_1 \implies AD_1 = (\tfrac{1}{3}, \tfrac{1}{3}, -\tfrac{2}{3}).$$

Calculamos $B_1 = \begin{pmatrix} AD_1 \\ e^T \end{pmatrix} \implies B_1 = \begin{pmatrix} \frac{1}{3} & \frac{1}{3} & -\frac{2}{3} \\ 1 & 1 & 1 \end{pmatrix}$,

$$c_p = (I - B_1^T (B_1 B_1^T)^{-1} B_1) D_1 c =$$

$$= \left(\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} - \begin{bmatrix} \frac{1}{3} & \frac{1}{3} & -\frac{2}{3} \\ 1 & 1 & 1 \end{bmatrix} \left(\begin{bmatrix} \frac{1}{3} & \frac{1}{3} & -\frac{2}{3} \\ 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} \frac{1}{3} & \frac{1}{3} & -\frac{2}{3} \\ 1 & 1 & 1 \end{bmatrix}^T \right)^{-1} \times \right.$$

$$\left. \times \begin{bmatrix} \frac{1}{3} & \frac{1}{3} & -\frac{2}{3} \\ 1 & 1 & 1 \end{bmatrix} \right) \begin{pmatrix} 0 \\ \frac{1}{3} \\ 0 \end{pmatrix} = \begin{pmatrix} \frac{1}{6} \\ -\frac{1}{6} \\ 0 \end{pmatrix},$$

luego

$$\|c_p\| = \sqrt{\frac{1}{6^2} + \frac{1}{6^2} + 0^2} = \frac{\sqrt{2}}{6}.$$

Obsérvese que en este caso la dirección c_p nos lleva desde $(\frac{1}{3}, \frac{1}{3}, \frac{1}{3})^T$ hacia la solución óptima $(\frac{2}{3}, 0, \frac{1}{3})^T$.

El nuevo punto es:

$$y = \frac{e}{n} - \alpha\gamma\frac{c_p}{\|c_p\|}, \qquad \alpha = 0.25 = \frac{1}{4}, \qquad r = \frac{1}{\sqrt{3(3-1)}} = \frac{1}{\sqrt{6}},$$

$$y = \begin{pmatrix} 1/3 \\ 1/3 \\ 1/3 \end{pmatrix} - (\frac{1}{4})(\frac{\sqrt{6}}{\sqrt{2}}) \begin{pmatrix} 1/6 \\ -1/6 \\ 0 \end{pmatrix},$$

$$y = \begin{pmatrix} 0.4119 \\ 0.2548 \\ 0.3333 \end{pmatrix}.$$

Aplicando la transformación inversa, el nuevo punto del proceso iterativo es:

$$x' = \frac{D_1 y}{e^T D_1 y} = \begin{pmatrix} 0.4119 \\ 0.2548 \\ 0.3333 \end{pmatrix}.$$

El valor de la función objetivo en este nuevo punto es $c^T x' = 0.2548$.

Iteración 2: $D_2 = \text{diag}(0.4119, 0.2548, 0.3333)$.

Calculamos $D_2 c$

$$D_2 c = \begin{pmatrix} 0 \\ 0.2548 \\ 0 \end{pmatrix},$$

$$AD_2 = (0.4119, 0.2548, -0.6666),$$

$$B_2 = \begin{pmatrix} AD_2 \\ e^T \end{pmatrix} = \begin{pmatrix} 0.4119 & 0.2548 & -0.6666 \\ 1 & 1 & 1 \end{pmatrix},$$

$$c_p = (I - B^T(BB^T)^{-1}B)D_2c,$$

$$c_p = \begin{pmatrix} +0.1243 \\ -0.1455 \\ 0.0212 \end{pmatrix},$$

$$\|c_p\| = 0.1925.$$

El nuevo punto en el conjunto transformado es:

$$y = \frac{e}{n} - \alpha r \frac{c_p}{\|c_p\|} = \begin{pmatrix} 0.4051 \\ 0.2494 \\ 0.3456 \end{pmatrix}.$$

Aplicando la transformación inversa, el nuevo punto del proceso iterativo en el conjunto original es:

$$x^2 = \frac{D_2 y}{e^T D_2 y} = \begin{pmatrix} 0.4828 \\ 0.1838 \\ 0.3333 \end{pmatrix}.$$

El valor de la función objetivo en este nuevo punto es $c^T x^2 = 0.1838$.

3.5 Función potencial y convergencia del algoritmo

En esta sección definimos la llamada **función potencial**. Esta función es usada por Karmarkar para medir el progreso del algoritmo y para demostrar que el proceso iterativo generado por la aplicación de la transformación proyectiva y la optimización de la función objetivo sobre la esfera inscrita en el simplejo $\mathcal{S}^n$, es polinomial. A continuación definimos la función potencial.

Definición 3.1 (Karmarkar [6]) *Sea $c^T x$ la función objetivo del problema lineal* (P). *Consideremos*

$$\mathcal{X} = \{x \in \mathbb{R}^n : \quad Ax = 0, \quad e^T x = 1, \quad c^T x > 0, \quad x > 0\}$$

La función potencial

$$f : \mathcal{X} \mapsto \mathbb{R}$$

es definida por

$$f(x) = \sum_{j=1}^{n} \ln\left(\frac{c^T x}{x_j}\right) \qquad \square \tag{3.5}$$

Nótese que si $c^T x \to 0 \implies f(x) \to -\infty$.

Veamos, a continuación, cómo cambia la función potencial bajo la transformación proyectiva. Para esto, sea $a := (a_1, \ldots, a_n) \in \mathcal{S}^n$, $a_i > 0$, $i = 1, \ldots, n$,
$D := \text{diag}(a_1, \ldots, a_n)$. Por la transformación proyectiva tenemos

$$x = \frac{Dy}{e^T Dy}, \quad x_j = \frac{a_j y_j}{e^T Dy}, \quad j = 1, \ldots, n,$$

entonces

$$\frac{c^T x}{x_j} = \frac{\frac{(Dc)^T y}{e^T Dy}}{\frac{a_j y_j}{e^T Dy}} = \frac{(Dc)^T y}{a_j y_j}.$$

Ahora la función potencial (3.5) se transforma en:

Sea F la función definida por:

$$F(y) = F(T(x; a)) := \sum_{j=1}^{n} \ln\left(\frac{(Dc)^T y}{a_j y_j}\right),$$

entonces

$$\begin{aligned} F(y) &= \sum_{j=1}^{n} \ln\left(\frac{(Dc)^T y}{a_j y_j}\right) = \sum_{j=1}^{n} \ln\left(\frac{(Dc)^T y}{y_j}\right) - \sum_{j=1}^{n} \ln(a_j) \qquad (3.6) \\ &= \sum_{j=1}^{n} \ln\left(\frac{(Dc)^T y}{y_j}\right) - \ln\left(\det(D)\right). \end{aligned}$$

El siguiente teorema da una relación importante entre $f(x)$ y $F(y)$.

Teorema 3.9 (Karmarkar [6]) *Se cumple* $F(x') - F(y') = f(x) - f(y)$, $\forall x, y \in \mathcal{S}^n$ *y* $x' = \dfrac{D^{-1}x}{e^T D^{-1}x}$, $y' = \dfrac{D^{-1}y}{e^T D^{-1}y}$.

Prueba:

$$\begin{aligned} F(x') - F(y') &= \sum_{j=1}^{n} \ln\left(\frac{(Dc)^T x'}{a_j x'_j}\right) - \sum_{j=1}^{n} \ln\left(\frac{(Dc)^T y'}{a_j y'_j}\right) \\ &= f(x) - f(y). \qquad \square \end{aligned}$$

Ahora mediremos el decremento en el valor de la función potencial (3.5) ó equivalentemente en la función potencial (3.6) en el y-conjunto en la iteración k. Usando la función potencial (3.6), obtenemos con las notaciones en el algoritmo de Karmarkar:

$$F(b') - F(a^0) = n \ln\left(\frac{(Dc)^T b'}{(Dc)^T a^0}\right) - \sum_{j=1}^{n} \ln(n b'_j),$$

desde que $a_j^0 = \dfrac{1}{n}$ para $j = 1, \ldots, n$. Recordando que en (3.4) obtuvimos que

$$\frac{(Dc)^T b'}{(Dc)^T a^0} \leq 1 - \frac{\alpha}{n-1},$$

se cumple

$$\ln\left[\frac{(Dc)^T b'}{(Dc)^T a^0}\right] \leq \ln\left(1 - \frac{\alpha}{n-1}\right) \leq -\frac{\alpha}{n-1}.$$

Usando esta desigualdad, obtenemos

$$F(b') - F(a^0) \leq -\frac{n\alpha}{n-1} - \sum_{j=1}^{n} \ln(n b'_j). \tag{3.7}$$

Luego, si $-\sum_{j=1}^{n} \ln(n b'_j)$ no se incrementa demasiado en relación a $\dfrac{n\alpha}{n-1}$ podremos obtener una cota en la disminución en la función potencial. Para ver que esto es cierto necesitamos el siguiente lema.

Lema 3.10 (Karmarkar [6]) *Si* $|x| \leq \beta < 1$, *entonces* $|\ln(1+x) - x| \leq \dfrac{x^2}{2(1-\beta)}$.

Prueba: Probaremos que:

$$x \geq \ln(1+x) \geq x - \frac{x^2}{2(1-|x|)}.$$

La primera desigualdad $x \geq \ln(1+x)$ es consecuencia directa de la función logaritmo.
La segunda desigualdad es probada por Karmarkar mediante el desarrollo del logaritmo

en serie de Taylor:

$$\begin{aligned}\ln(1+x) &= x - \frac{x^2}{2} + \frac{x^3}{3} - \frac{x^4}{4} + \cdots \\ &= x - \frac{x^2}{2}\left(1 - \frac{2}{3}x + \frac{2}{4}x^2 + \cdots\right) \\ &\geq x - \frac{x^2}{2}\left(1 + |x| + |x|^2 + \cdots\right) \\ &= x - \frac{x^2}{2(1-|x|)},\end{aligned}$$

entonces

$$\ln(x+1) \geq x - \frac{x^2}{2(1-|x|)} \geq x - \frac{x^2}{2(1-\beta)}.$$

De $x \geq \ln(x+1) \geq x - \dfrac{x^2}{2(1-\beta)}$ obtenemos:

$$-\frac{x^2}{2(1-\beta)} \leq \ln(1+x) - x \leq \frac{x^2}{2(1-\beta)},$$

luego

$$|\ln(1+x) - x| \leq \frac{x^2}{2(1-\beta)}. \quad \square$$

Si seleccionamos $\alpha \in \langle 0, 1\rangle$, suficientemente pequeño tal que

$$\sqrt{\frac{n}{n-1}}\alpha < 1,$$

podemos entonces establecer el siguiente lema.

Lema 3.11 (Karmarkar [6]) *Sea* $\beta := \sqrt{\dfrac{n}{n-1}}\alpha < 1 \quad , \qquad 0 < \alpha < 1$, *entonces*

$$\left|\sum_{j=1}^{n} \ln(nb'_j)\right| \leq \frac{\beta^2}{2(1-\beta)}.$$

Prueba: La sumatoria $\displaystyle\sum_{j=1}^{n} \ln(nb'_j)$ puede escribirse como:

$$\begin{aligned}\sum_{j=1}^{n} \ln(nb'_j) &= \sum_{j=1}^{n} \ln(n) + \sum_{j=1}^{n} \ln(b'_j) = -\sum_{j=1}^{n} \ln\left(\frac{1}{n}\right) + \sum_{j=1}^{n} \ln(b'_j), \\ &= \sum_{j=1}^{n} \left(\ln(b'_j) - \ln\left(\frac{1}{n}\right)\right) = \sum_{j=1}^{n} \ln\left(\frac{b'_j}{\frac{1}{n}}\right) = \sum_{j=1}^{n} \ln\left(\frac{b'_j}{a_j^0}\right),\end{aligned}$$

ya que $a_j^0 = \dfrac{1}{n}, \quad j = 1, \ldots, n.$

$b' \in B(a^0; \alpha r)$ implica:

$$\begin{aligned}
\|b' - a^0\|^2 = \sum_{j=1}^{n} \left(b'_j - \frac{1}{n}\right)^2 &\leq \alpha^2 r^2 \\
\Longrightarrow \sum_{j=1}^{n} n^2 \left(b'_j - \frac{1}{n}\right)^2 &\leq n^2\alpha^2 r^2 \\
\Longrightarrow \sum_{j=1}^{n} \left(\frac{b'_j - \frac{1}{n}}{\frac{1}{n}}\right)^2 &\leq n^2\alpha^2 \frac{1}{n(n-1)} = \beta^2 \\
\Longrightarrow \left|\frac{b'_j - \frac{1}{n}}{\frac{1}{n}}\right| &\leq \beta, \qquad j = 1, \ldots, n.
\end{aligned}$$

Usando el **Lema 3.10**, tenemos:

$$\left|\ln\left(1 + \frac{b'_j - \frac{1}{n}}{\frac{1}{n}}\right) - \frac{b'_j - \frac{1}{n}}{\frac{1}{n}}\right| \leq \frac{1}{2(1-\beta)}\left(\frac{b'_j - \frac{1}{n}}{\frac{1}{n}}\right)^2$$

$$\Longrightarrow -\frac{1}{2(1-\beta)}\left(\frac{b'_j - \frac{1}{n}}{\frac{1}{n}}\right)^2 \leq \ln\left(1 + \frac{b'_j - \frac{1}{n}}{\frac{1}{n}}\right) - \frac{b'_j - \frac{1}{n}}{\frac{1}{n}} \leq \frac{1}{2(1-\beta)}\left(\frac{b'_j - \frac{1}{n}}{\frac{1}{n}}\right)^2$$

$$\Longrightarrow -\frac{1}{2(1-\beta)}\sum_{j=1}^{n}\left(\frac{b'_j - \frac{1}{n}}{\frac{1}{n}}\right)^2 \leq \sum_{j=1}^{n}\ln\left(1 + \frac{b'_j - \frac{1}{n}}{\frac{1}{n}}\right) - \sum_{j=1}^{n}\frac{b'_j - \frac{1}{n}}{\frac{1}{n}} \leq \frac{1}{2(1-\beta)}\sum_{j=1}^{n}\left(\frac{b'_j - \frac{1}{n}}{\frac{1}{n}}\right)^2$$

$$\begin{aligned}
\text{Por} \quad \sum_{j=1}^{n}\left(\frac{b'_j - \frac{1}{n}}{\frac{1}{n}}\right) &= n\sum_{j=1}^{n}\left(b'_j - \frac{1}{n}\right) \\
&= n\left(\sum_{j=1}^{n} b'_j - \sum_{j=1}^{n}\frac{1}{n}\right), \\
&= n(1-1) = 0,
\end{aligned}$$

resulta finalmente

$$-\frac{1}{2(1-\beta)}\sum_{j=1}^{n}\left(\frac{b'_j-\frac{1}{n}}{\frac{1}{n}}\right)^2 \leq \sum_{j=1}^{n}\ln\left(1+\frac{b'_j-\frac{1}{n}}{\frac{1}{n}}\right) \leq \frac{1}{2(1-\beta)}\sum_{j=1}^{n}\left(\frac{b'_j-\frac{1}{n}}{\frac{1}{n}}\right)^2$$

$$\Longrightarrow \left|\sum_{j=1}^{n}\ln\left(1+\frac{b'_j-\frac{1}{n}}{\frac{1}{n}}\right)\right| \leq \frac{1}{2(1-\beta)}\sum_{j=1}^{n}\left(\frac{b'_j-\frac{1}{n}}{\frac{1}{n}}\right)^2$$

$$\leq \frac{\beta^2}{2(1-\beta)}.$$

$$\Longrightarrow \left|\sum_{j=1}^{n}\ln(nb'_j)\right| \leq \frac{\beta^2}{2(1-\beta)}. \qquad \square$$

Ahora podemos mejorar la estimación (3.7), por el **Lema 3.11**:

$$F(b')-F(a^0) \leq -\frac{n\alpha}{n-1}-\sum_{j=1}^{n}\ln(nb'_j),$$

$$\Longrightarrow \quad F(b')-F(a^0) \leq -\frac{n\alpha}{n-1}+\frac{\beta^2}{2(1-\beta)}.$$

Como $\beta=\sqrt{\frac{n}{n-1}}\alpha$, tenemos:

$$F(b')-F(a^0) \leq -\frac{n\alpha}{n-1}+\frac{n\alpha^2}{2(n-1)\left(1-\alpha\sqrt{\frac{n}{n-1}}\right)},$$

$$F(b')-F(a^0) \leq -\left(\frac{n\alpha}{n-1}-\frac{n\alpha^2}{2(n-1)\left(1-\alpha\sqrt{\frac{n}{n-1}}\right)}\right).$$

Si definimos:

$$\delta := \frac{n\alpha}{n-1}-\frac{n\alpha^2}{2(n-1)\left(1-\alpha\sqrt{\frac{n}{n-1}}\right)},$$

entonces:

$$F(b')-F(a^0) \leq -\delta$$

Nótese que la anterior relación significa que la función $F(\cdot)$ puede ser reducida por una constante en cada iteración.

Observaciones:

1. Para valores grandes de n, $\beta \approx \alpha$.

2. $\lim_{n\to\infty} \delta = \alpha - \frac{\alpha^2}{2(1-\alpha)}$. Si ponemos $\alpha = \frac{1}{4}$, entonces $\lim_{n\to\infty} \delta = \frac{5}{4} \geq \frac{1}{8}$, en general

$$\alpha - \frac{\alpha^2}{2(1-\alpha)} > 0 \iff$$

$$\frac{3\alpha^2 - 2\alpha}{2(\alpha - 1)} > 0 \iff \alpha < 2/3.$$

Ahora si hacemos $x^k = T^{-1}(a^0)$ y $x^{k+1} = T^{-1}(b')$, se cumple

$$F(T(x^{k+1})) - F(T(x^k)) \leq -\delta,$$

y por el Teorema 3.9

$$f(x^{k+1}) - f(x^k) = F(T(x^{k+1})) - F(T(x^k)) \leq -\delta,$$

entonces:

$$f(x^{k+1}) - f(x^k) \leq -\delta.$$

Es decir, esta reducción de δ en el valor de $F(T(x^k))$ se traduce en la misma reducción en $f(x^k)$ y debido al siguiente teorema esta reducción en $f(x^k)$ se traduce también en una reducción en $c^T x^k$.

Teorema 3.12 (Karmarkar [6]) *Sea L definido como en el algoritmo de Karmarkar (vea seccion 3.4). En un número total de operaciones aritméticas del orden $\mathcal{O}(n^4 L)$ el algoritmo de Karmarkar encuentra un punto factible x^k tal que:*

$$c^T x^k \leq 2^{-L},$$

donde $x^0 := a^0 = \frac{1}{n} e$ es el centro del simplejo S^n.

Prueba: Sean $\{x^k\}$ los puntos generados por el algoritmo de Karmarkar. La función potencial cumple:

$$f(x^{k+1}) \leq f(x^k) - \delta;$$

y después de la k-ésima iteración tenemos:

$$\begin{aligned} & f(x^k) \leq f(x^0) - k\delta. \\ \Longrightarrow \quad & f(x^k) - f(x^0) \leq -k\delta. \end{aligned}$$

$x^0 = \left(\frac{1}{n}, \cdots, \frac{1}{n}\right)$, $f(x^k)$ es dado por:

$$f(x^k) = n\ln(c^T x^k) - \sum_{j=1}^{n} \ln(x_j^k),$$

$f(x^0) = n\ln(c^T x^0) - \sum_{j=1}^{n} \ln(x_j^0)$, entonces resulta:

$$\begin{aligned} f(x^k) - f(x^0) &= n\ln(c^T x^k) - \sum_{j=1}^{n} \ln(x_j^k) - n\ln(c^T x^0) + \sum_{j=1}^{n} \ln(x_j^0) \\ &= n\ln\left(\frac{c^T x^k}{c^T x^0}\right) - \left(\sum_{j=1}^{n} \ln(x_j^k) - \sum_{j=1}^{n} \ln(x_j^0)\right). \end{aligned}$$

Como $x_j^0 = \frac{1}{n}, \quad j = 1, \ldots, n$, se cumple:

$$\begin{aligned} f(x^k) - f(x^0) &= n\ln\left(\frac{c^T x^k}{c^T x^0}\right) - \left(\sum_{j=1}^{n} \ln(x_j^k) + \sum_{j=1}^{n} \ln(n)\right) \\ &= n\ln\left(\frac{c^T x^k}{c^T x^0}\right) - \sum_{j=1}^{n} \left(\ln(x_j^k) + \ln(n)\right) \\ &= n\ln\left(\frac{c^T x^k}{c^T x^0}\right) - \sum_{j=1}^{n} \ln(n x_j^k), \end{aligned}$$

entonces

$$n\ln\left(\frac{c^T x^k}{c^T x^0}\right) - \sum_{j=1}^{n} \ln(n x_j^k) \leq -k\delta.$$

Usando el hecho de que la media geométrica es menor o igual a la media aritmética:

$$\left[\prod_{j=1}^{n} (x_j^k)\right]^{1/n} \leq \frac{\sum_{j=1}^{n} x_j^k}{n}$$

y aprovechando de que $\sum_{j=1}^{n} x_j^k = 1$,

tenemos

$$\sum_{j=1}^{n} \ln(n x_j^k) = n \ln \left[n \left(\prod_{j=1}^{n} x_j^k \right)^{1/n} \right] \leq n \ln \left(\sum_{j=1}^{n} x_j^k \right) = 0.$$

Entonces resulta

$$n \ln \left(\frac{c^T x^k}{c^T x^0} \right) \leq -k\delta,$$

lo que implica

$$c^T x^k \leq (c^T x^0) \exp\left(-\frac{k\delta}{n} \right).$$

Luego tenemos que para $k \geq 2\frac{n}{\delta} L$ y por empleo del **Lema 3.4**

$$\begin{aligned} c^T x^k \leq (c^T x^0) \exp\left(\frac{-k\delta}{n} \right) &\leq \left(\sum_{j=1}^{n} \frac{c_j}{n} \right) \exp(-2L) \\ &\leq 2^L 2^{-2L} = 2^{-L}, \end{aligned}$$

es decir $c^T x^k \leq 2^{-L}$.

Ahora en cada iteración el algoritmo calcula

$$c_p^k = \left(I - B_k^T (B_k B_k^T)^{-1} B_k \right) D_k c$$

lo cual implica calcular la inversa de $(B_k B_k^T)$ ó equivalentemente resolver un sistema lineal de ecuaciones de la forma $(B_k B_k^T)u = v$ el cual toma en el peor de los casos un número de operaciones aritmeticas del orden de n^3 (es decir $\mathcal{O}(n^3)$) y como el número de iteraciones $k = \mathcal{O}(nL)$ la complejidad total del algoritmo es $\mathcal{O}(n^3 \cdot nL) = \mathcal{O}(n^4 L)$ □

3.6 Comentarios adicionales sobre la complejidad del algoritmo

El mayor esfuerzo computacional del algoritmo de Karmarkar en cada iteración está dado por el cálculo de c_p^k:

$$c_p^k = \left(I - B_k^T (B_k B_k^T)^{-1} B_k \right) D_k c.$$

Tenemos que encontrar la inversa de $B_k B_k^T$ ó equivalentemente, hallar la solución de la ecuación normal:

$$B_k B_k^T \lambda = B_k D_k c.$$

Estos cálculos dominan la complejidad teórica del algoritmo. Sabemos que:

$$B_k = \begin{bmatrix} AD_k \\ e^T \end{bmatrix}.$$

Entonces

$$B_k B_k^T = \begin{bmatrix} AD_k^2 A^T & AD_k e \\ (AD_k e)^T & e^T e \end{bmatrix} = \begin{bmatrix} AD_k^2 A^T & 0 \\ 0 & n \end{bmatrix}.$$

D_k es una matriz diagonal que contiene la solución de la k-ésima iteración

$$D_k = \text{diag}(x^k).$$

Entonces

$$c_p^k = \left[I - (D_k A^T, e) \begin{bmatrix} AD_k^2 A^T & AD_k e \\ e^T D_k A^T & n \end{bmatrix}^{-1} \begin{pmatrix} AD_k \\ e^T \end{pmatrix} \right] D_k c,$$

$$= \left[I - (D_k A^T, e) \begin{bmatrix} AD_k^2 A^T & 0 \\ 0^T & n \end{bmatrix}^{-1} \begin{pmatrix} AD_k \\ e^T \end{pmatrix} \right] D_k c,$$

$$c_p^k = \left[I - (AD_k)^T (AD_k^2 A^T)^{-1} AD_k - \frac{1}{n} ee^T \right] D_k c.$$

Poniendo

$$\begin{aligned} d &:= (AD_k^2 A^T)^{-1} AD_k D_k c \\ &= [(AD_k)(AD_k)^T]^{-1} AD_k D_k c, \end{aligned}$$

d satisface las ecuaciones normales

$$(AD_k)(AD_k)^T d = AD_k D_k c,$$

es decir, d es solución de:

$$\min \|(AD_k)^T d - D_k c\|_2 \quad , \qquad d \in \mathbb{R}^n.$$

Otra posibilidad de solucionar las ecuaciones normales es una factorización QR por los métodos Householder ó Givens. En las implementaciones del algoritmo de Karmarkar se aplica la descomposición de Cholesky a la matriz $B_k B_k^T$ (en realidad a la expresión $AD_k^2 A^T$) es decir:

$$B_k B_k^T = LU,$$

donde L es una matriz triangular inferior y U es una matriz triangular superior.
El procedimiento de Cholesky escoge

$$l_{ii} = u_{ii} \qquad i = 1, \ldots, m+1$$

con lo cual resulta que $U = L^T$ y $B_k B_k^T = LL^T$, entonces sólo es necesario evaluar y almacenar la matriz L.

De esta forma, la inversión de la matriz $B_k B_k^T$ queda sujeta a las rápidas técnicas de inversión de matrices triangulares.

3.7 Estabilidad numérica

En esta sección presentamos un ejemplo que prueba que el algoritmo de Karmarkar es más estable numéricamente que el algoritmo simplex para una determinada familia de problemas de programación lineal. Este ejemplo le fue proporcionado al autor de la tesis por C. Roos [16].

Considere el siguiente problema de programación lineal

$$\begin{aligned} \max \quad & z = c^T x \\ \text{sujeto a} \quad & Ax \leq b \end{aligned}$$

donde

$$A = \begin{bmatrix} \frac{1}{2} & \frac{1}{3} & \cdots & \frac{1}{n+1} \\ \frac{1}{3} & \frac{1}{4} & \cdots & \frac{1}{n+2} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{1}{n+1} & \frac{1}{n+2} & \cdots & \frac{1}{2n} \end{bmatrix},$$

$$b_i = \sum_{j=1}^{n} \frac{1}{i+j} \qquad i = 1, \ldots, n,$$

$$c_i = \frac{2}{i+1} + \sum_{j=2}^{n} \frac{1}{i+j} \qquad i = 1, \ldots, n.$$

Este problema tiene una única solución óptima:

$$x = (1, 1, \ldots, 1)^T,$$

y el problema dual tiene también una solución óptima única:

$$y = (2, 1, 1, \ldots)^T.$$

Cuando este problema es resuelto por el algoritmo simplex los resultados son malos, se alejan de la solución verdadera $x = (1, 1, \ldots, 1)^T$. La razón es la mala estructura de la matriz A. Al aplicar el algoritmo de Karmarkar a este problema los resultados son mucho mejores. La Tabla 3.1 muestra las soluciones producidas por el algoritmo de Karmarkar para varios valores de n.

Capítulo 4

Forma normal de Karmarkar

4. Forma normal de Karmarkar

El algoritmo de Karmarkar sólo es aplicable a problemas de programación lineal en la llamada forma normal de Karmarkar, para los cuales el valor óptimo es conocido y vale cero, y además está dado un punto inicial interior a partir del cual se empieza a iterar.

En este capítulo veremos que todo problema de programación lineal, puede transformarse en un problema de programación lineal que satisface estos requerimientos; es decir que puede llevarse a la forma normal de Karmarkar, la cual recordemos que es:

$$\left.\begin{array}{rl} \min & c^T x \\ \text{sujeto a}: & Ax = 0 \\ & e^T x = 1 \\ & x \geq 0 \end{array}\right\}, \qquad \text{(P)}$$

donde: $A \in \mathbb{R}^{m\times n}$, $c, x \in \mathbb{R}^n$, $e := (1, \ldots, 1)^T \in \mathbb{R}^n$, $b \in \mathbb{R}^m$ y $c^T x^* = 0$, donde x^* es la solución de (P) y además $x_0 := \frac{1}{n}e$ es un punto inicial interior factible, $\text{ran}(A) = m$.

El desarrollo de este capítulo está basado en el artículo de Karmarkar [6].

4.1 Transformación de un problema lineal a la forma normal de Karmarkar

Consideremos el problema de programación lineal de la forma:

$$\begin{array}{rl} & \max d^T v \\ \text{sujeto a}: & \left.\begin{array}{rl} A_0 v & \leq \hat{b} \\ v & \geq 0 \end{array}\right\} \qquad \text{(PL)} \end{array} \qquad (4.1)$$

donde: $A_0 \in \mathbb{R}^{k\times l}$, $v, d \in \mathbb{R}^l$, $\hat{b} \in \mathbb{R}^k$, $\text{ran}(A_0) = k$.

El problema dual para (4.1) se escribe como:

$$\min \hat{b}^T u$$

$$\text{sujeto a :} \quad \left.\begin{array}{rcl} A_0^T u & \geq & d \\ u & \geq & 0 \end{array}\right\} \quad \textbf{(PD)} \tag{4.2}$$

Supongamos que ambos problemas de programación lineal (4.1) y (4.2) son factibles y sea v_0 un punto factible para (4.1) y u_0 un punto factible para (4.2). De esta forma tenemos que:

$$\hat{b}^T u_0 \geq (A_0 v_0)^T u_0 \geq d^T v_0, \tag{4.3}$$

es decir:

$$\hat{b}^T u_0 \geq d^T v_0. \tag{4.4}$$

De la teoría de la dualidad (Teorema de Kuhn-Tucker) para programación lineal se infiere que hay igualdad en (4.4) si y sólo si v_0 y u_0 son soluciones óptimas para (4.1) y (4.2) respectivamente. De ahí, se deduce que (4.1) tiene una solución óptima finita si y sólo si la combinación de los problemas primal y dual:

$$\left.\begin{array}{rcl} A_0 v & \leq & \hat{b} \\ A_0^T u & \geq & d \\ \hat{b}^T u - d^T v & = & 0 \\ v \geq 0 \quad , & & u \geq 0 \end{array}\right\} \tag{4.5}$$

es factible.

Al introducir variables de holgura no negativas, (4.5) puede reescribirse en la siguiente forma:

$$\left.\begin{array}{rcl} A_0 v + s & = & \hat{b} \\ A_0^T u - r & = & d \\ \hat{b}^T u - d^T v & = & 0 \end{array}\right\} \tag{4.6}$$

donde: $v \geq 0, \quad u \geq 0, \quad s \geq 0, \quad r \geq 0$

Si $v_0, \quad u_0, \quad , s_0, \quad r_0$ son todos vectores positivos entonces (4.6) será factible, si y sólo si, el valor mínimo de λ es **cero** en el problema de programación lineal (4.7).

Introducción de una variable auxiliar:

$$\begin{aligned} &\min \quad \lambda \\ &\left.\begin{aligned} A_0 v + s + \lambda(\hat{b} - A_0 v_0 - s_0) &= \hat{b} \\ A_0^T u - r + \lambda(d - A_0^T u_0 - r_0) &= d \\ \hat{b}^T u - d^T v - \lambda(\hat{b}^T u_0 - d^T v_0 &= 0 \end{aligned}\right\} \end{aligned} \tag{4.7}$$

donde: $v \geq 0,\ u \geq 0,\ s \geq 0,\ r \geq 0.$

Si (4.1) tiene una solución óptima, entonces el valor óptimo en (4.7) será cero. Debe notarse que:

$$v = v_0, \quad u = u_0, \quad s = s_0, \quad r = r_0, \quad \lambda = 1 \tag{4.8}$$

es un punto factible positivo, que puede usarse como punto de partida. De este modo se ha demostrado como (4.1) puede reformularse para cumplir la exigencia de que el valor óptimo sea cero; y también se da un punto inicial factible. El problema reformulado adquiere la forma siguiente:

$$\min \lambda$$

$$\text{sujeto a} \begin{bmatrix} A_0 & 0 & I & 0 & \hat{b} - A_0 v_0 - s_0 \\ 0 & A_0^T & 0 & -I & d - A_0^T u_0 - r_0 \\ -d^T & b^T & 0 & 0 & -\hat{b}^T u_0 + d^T v_0 \end{bmatrix} \begin{bmatrix} v \\ u \\ s \\ r \\ \lambda \end{bmatrix} = \begin{bmatrix} \hat{b} \\ d \\ 0 \end{bmatrix} \tag{4.9}$$

donde: $v = v_0,\ u = u_0,\ s = s_0,\ r = r_0,\ \lambda = 1$ es un punto inicial interior factible.

Existen diferentes formas mediante las cuales se puede manejar el problema con el lado derecho que es diferente a cero. (Es decir, el problema se "homogenizará"). A continuación, se presentará el método utilizado por Karmarkar [6]. Este método ha funcionado bien en la práctica y no han existido problemas con su implementación. En lo sucesivo se empleará la notación:

$$c := \underbrace{(0, \ldots, 0, 1)}_{2k+2l}^T, \tag{4.10}$$

$$b := \begin{bmatrix} \hat{b} \\ d \\ 0 \end{bmatrix}, \tag{4.11}$$

$$x := \begin{bmatrix} v \\ u \\ s \\ r \\ \lambda \end{bmatrix}, \tag{4.12}$$

$$m := k + l + 1 \quad , \quad n := 2k + 2l + 1 \tag{4.13}$$

$$A := \begin{bmatrix} A_0 & \vdots & 0 & \vdots & I & \vdots & 0 & \vdots & \hat{b} - A_0 v_0 - s_0 \\ 0 & \vdots & A_0^T & \vdots & 0 & \vdots & -I & \vdots & d - A_0^T u_0 + r_0 \\ -d^T & \vdots & b^T & \vdots & 0 & \vdots & 0 & \vdots & -\hat{b}^T u_0 + d^T v_0 \end{bmatrix}, \tag{4.14}$$

$\text{ran}(A) = m$ porque las filas de A son linealmente independientes.

El problema (4.9) será entonces reemplazado por:

$$\left.\begin{array}{rrcl} & \min c^T x, & & \\ \text{sujeto a :} & Ax & = & b, \\ & x & \geq & 0, \\ A \in \mathbb{R}^{m \times n}, \quad c, x \in \mathbb{R}^n & , & b \in \mathbb{R}^m & \end{array}\right\} \tag{4.15}$$

Además

$$a := (v_0, u_0, s_0, r_0, 1)^T, \tag{4.16}$$

es un punto inicial factible, y el valor óptimo es cero. Con ayuda de una transformación proyectiva, este problema puede transformarse a la **forma normal de Karmarkar**; después de lo cual puede resolverse por medio del algoritmo de Karmarkar. Ahora la transformación proyectiva que se usará difiere, ligeramente, de la que se usó anteriormente. Esto es debido a que considerabamos que teníamos el problema de programación lineal en la forma (**P**), y en esta forma la ecuación del simplex aparecía incorporada en las restricciones del problema (**P**), lo cual no ocurre en (4.15).

Sea $P_+ := \{x \in \mathbb{R}^n : \quad x \geq 0\}$

$$\mathcal{S}^{n+1} := \{x \in \mathbb{R}^{n+1} : \quad \sum_{i=1}^{n+1} x_i = 1, \quad x \geq 0\}$$

se definirá una transformación proyectiva que asocie a cada punto en P_+ un punto en $\mathcal{S}^{n+1}$. En la Figura 4.1 se representa esta transformación para el caso bidimensional.

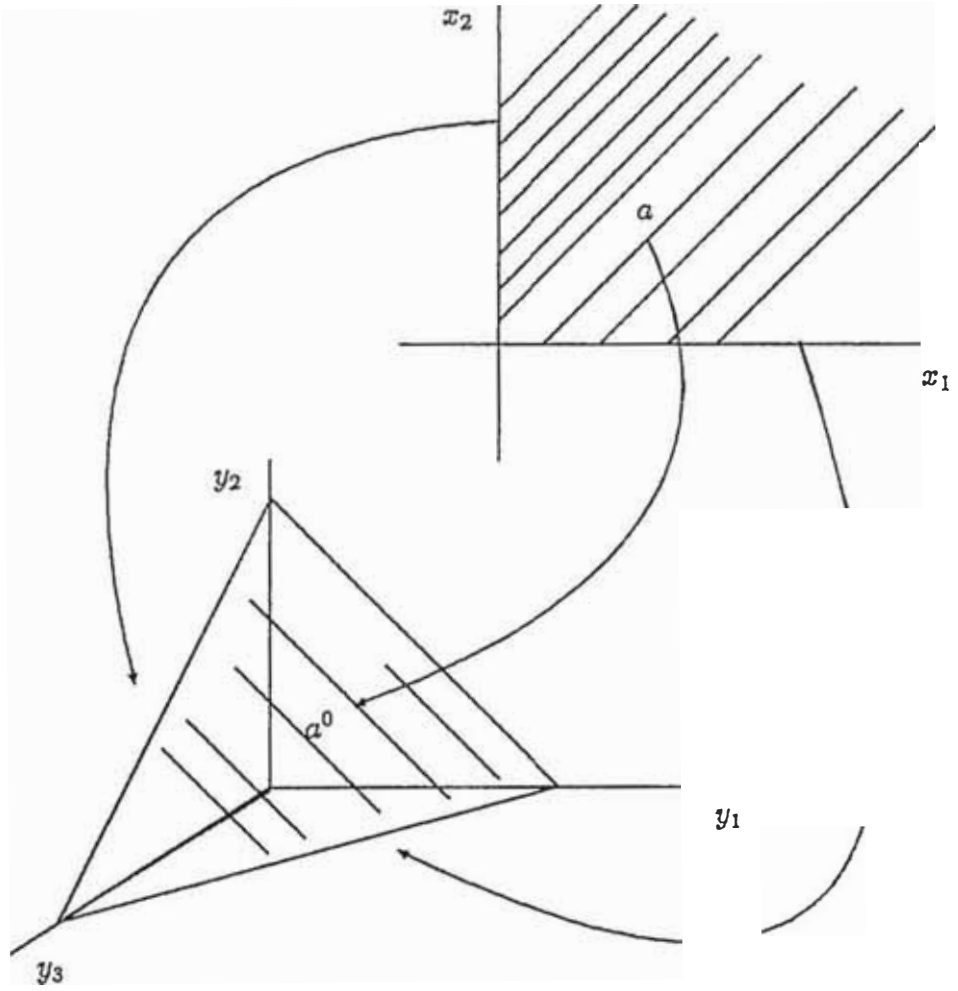


Figura 4.1: Transformación proyectiva

Consideremos la transformación proyectiva

$$T : P_+ \mapsto \mathcal{S}^{n+1}$$

definida por:

$$T(x_1, x_2, \ldots, x_n)^T = \frac{1}{1 + \sum_{j=1}^{n} \frac{x_j}{a_j}} (\frac{x_1}{a_1}, \frac{x_2}{a_2}, \cdots, \frac{x_n}{a_n}, 1)^T,$$

$$\text{es decir:} \qquad y_i = \frac{\frac{x_i}{a_i}}{1 + \sum_{j=1}^{n} (\frac{x_j}{a_j})}, \qquad i = 1, \ldots, n$$

$$y_{n+1} = \frac{1}{1 + \sum_{j=1}^{n} (\frac{x_j}{a_j})},$$

donde $a = (a_1, a_2, \ldots, a_n)$ denota el punto dado por (4.16).

Esta transformación es biyectiva y su inversa es dada por:

$$T^{-1} : \mathcal{S}^{n+1} \mapsto P_+$$

$$T^{-1}(y_1, y_2, \ldots, y_n, y_{n+1})^T = \frac{1}{y_{n+1}} (a_1 y_1, \ldots, a_n y_n)^T,$$

$$\text{es decir:} \qquad x_i = \frac{a_i y_i}{y_{n+1}}, \qquad i = 1, \ldots, n.$$

Ahora es claro que la imagen del punto $a = (a_1, a_2, \ldots, a_n)^T$ es el centro del simplejo $\mathcal{S}^{n+1}$, desde que

$$\begin{aligned} T(a_1, a_2, \ldots, a_n)^T &= \frac{1}{1 + \sum_{j=1}^{n} \frac{a_j}{a_j}} (\frac{a_1}{a_1}, \frac{a_2}{a_2}, \cdots, \frac{a_n}{a_n}, 1)^T \\ &= \frac{1}{n+1} (1, 1, \ldots, 1, 1)^T \\ &= \frac{1}{n+1} e^T. \end{aligned}$$

Si A_i denota la i-ésima columna de A en (4.15), entonces $Ax = b$ puede escribirse como

$$\sum_{i=1}^{n} A_i x_i = b.$$

Escribiendo $x_i = \dfrac{a_i y_i}{y_{n+1}}$, obtenemos

$$\frac{\sum_{i=1}^{n} A_i a_i y_i}{y_{n+1}} = b \qquad \Longrightarrow \qquad \sum_{i=1}^{n} A_i a_i y_i - b y_{n+1} = 0.$$

Si definimos las columnas de una matriz $\hat{A}$ por medio de

$$\hat{A}_i := a_i A_i \qquad i = 1, \ldots, n. \tag{4.17}$$

$$\hat{A}_{n+1} := -b, \tag{4.18}$$

obtenemos un sistema de ecuaciones homogéneo:

$$\sum_{i=1}^{n+1} \hat{A}_i y_i = 0,$$

el cual era el objetivo de la transformación.

Ahora deseamos que el valor óptimo en el problema transformado sea también cero. Sea

$$Z = \{x \in \mathbb{R}^n : \quad c^T x = 0\}$$

Sustituyendo

$$x_i = \frac{a_i y_i}{y_{n+1}}, \qquad i = 1, \ldots, n,$$

obtenemos:

$$\sum_{i=1}^{n+1} \frac{c_i a_i y_i}{y_{n+1}} = 0.$$

Definamos $\hat{c}$ mediante:

$$\hat{c}_i = a_i c_i \qquad i = 1, \ldots, n, \qquad \hat{c}_{n+1} = 0,$$

entonces $c^T x = 0$ implica $\hat{c}^T y = 0$.

Luego Z se transforma en

$$Z' = \{y \in \mathbb{R}^{n+1} : \quad \hat{c}^T y = 0\}$$

que es lo que deseabamos.

El problema transformado es:

$$\begin{aligned} \min \quad & \hat{c}^T y \\ \text{sujeto a} \quad & \hat{A}y = 0, \\ & e^T y = 1, \\ & y \geq 0. \end{aligned} \tag{4.19}$$

El punto $a^0 = \dfrac{1}{n+1}e$ es un punto de partida factible. Para ver esto, escribamos el sistema homogéneo $\hat{A}y = 0$ en la forma

$$\sum_{i=1}^{n+1} \hat{A}_i y_i = 0,$$

entonces

$$\sum_{i=1}^{n+1} \hat{A}_i y_i = \sum_{i=1}^{n} \hat{A}_i y_i + \hat{A}_{n+1} y_{n+1}.$$

Por (4.17) y (4.18) obtenemos

$$\sum_{i=1}^{n+1} \hat{A}_i y_i = \sum_{i=1}^{n} a_i A_i y_i - b y_{n+1},$$

pero ahora en estas ecuaciones reemplazamos las coordenadas del punto a^0

$$a_i^0 = y_i = \frac{1}{n+1}, \quad i = 1, \ldots, n+1.$$

$$\begin{aligned} \Longrightarrow \quad \sum_{i=1}^{n+1} \hat{A}y_i = & \sum_{i=1}^{n} a_i A_i \left(\frac{1}{n+1}\right) - b\left(\frac{1}{n+1}\right) = \\ = & \frac{1}{n+1}\left(\sum_{i=1}^{n} a_i A_i - b\right), \end{aligned}$$

pero $\displaystyle\sum_{i=1}^{n} a_i A_i - b = 0$ ya que el punto a dado por (4.16) es un punto factible para el problema (4.15). Luego $a^0 = \dfrac{1}{n+1}e$ es un punto inicial factible de (4.19).

Además si $y = y^*$ es óptimo del problema (4.19) entonces $T^{-1}(y^*)$ será una solución óptima del problema 4.15 (**lemas 3.2 y 3.3**).

Ejemplo 4.1 *Transformar el problema de programación lineal*

$$\begin{aligned} \max: \quad & 2v_1 + v_2, \\ \textit{sujeto a} \quad & v_1 - v_2 \leq 2, \\ & v_1 + 2v_2 \leq 4, \\ & v_1 \ , \ v_2 \geq 0, \end{aligned}$$

a la forma normal de Karmarkar.

Solución: En este caso tenemos

$$A_0 = \begin{bmatrix} 1 & -1 \\ 1 & 2 \end{bmatrix},$$

$$\begin{aligned} \hat{b} &= (2,4)^T, \\ d &= (2,1)^T, \\ v &= (v_1, v_2). \end{aligned}$$

El problema dual asociado es

$$\begin{aligned} \min: \quad & 2u_1 + u_2, \\ \text{sujeto a} \quad & u_1 + u_2 \geq 2, \\ & -u_1 + 2u_2 \geq 1, \\ & u_1 \ , \ u_2 \geq 0. \end{aligned}$$

Por (4.5) la combinación de los problemas primal y dual es

$$\begin{aligned} v_1 - v_2 &\leq 2, \\ v_1 + 2v_2 &\leq 4, \\ u_1 + u_2 &\geq 2, \\ -u_1 + 2u_2 &\geq 1, \\ 2u_1 + u_2 - 2v_1 - v_2 &= 0, \\ v_1 \geq 0, \quad v_2 \geq 0, \quad u_1 \geq 0, \quad u_2 &\geq 0. \end{aligned}$$

Al introducir variables de holgura no negativas, podemos reescribir el problema anterior en la forma (4.6)

$$\begin{aligned} v_1 - v_2 + s_1 &= 2, \\ v_1 + 2v_2 + s_2 &= 4, \\ u_1 + u_2 - r_1 &= 2, \\ -u_1 + 2u_2 - r_2 &= 1, \\ 2u_1 + u_2 - 2v_1 - v_2 &= 0, \end{aligned}$$

$$v_1 \geq 0, \quad v_2 \geq 0, \quad u_1 \geq 0, \quad u_2 \geq 0, s_1 \geq 0, \quad s_2 \geq 0, \quad r_1 \geq 0, \quad r_2 \geq 0.$$

Escogemos $v_0, u_0, s_0, r_0 \in \mathbb{R}^2$ vectores positivos, una posible elección es:

$$\begin{aligned} v_0 &:= (1,1)^T, \\ u_0 &:= (1,1)^T, \\ s_0 &:= (1,1)^T, \\ r_0 &:= (1,1)^T. \end{aligned}$$

Por (4.7) y (4.9) el problema reformulado adquiere la forma siguiente:

$$\min \lambda$$

$$\text{sujeto a} \quad \begin{bmatrix} 1 & -1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 1 & 2 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & -1 & 0 & 1 \\ 0 & 0 & -1 & 2 & 0 & 0 & 0 & -1 & 1 \\ -2 & -1 & 2 & 4 & 0 & 0 & 0 & 0 & -3 \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \\ u_1 \\ u_2 \\ s_1 \\ s_2 \\ r_1 \\ r_2 \\ \lambda \end{bmatrix} = \begin{bmatrix} 2 \\ 4 \\ 2 \\ 1 \\ 0 \end{bmatrix}$$

Por (4.10), (4.11), (4.12), (4.13) y (4.14) tenemos

$$
\begin{aligned}
c &:= (0,0,0,0,0,0,0,0,1)^T,\\
b &:= (2,4,2,1,0)^T,\\
x &= (x_1,x_2,x_3,x_4,x_5),x_6,x_7,x_8,x_9)^T := (v_1,v_2,u_1,u_2,s_1,s_2,r_1,r_2,\lambda)^T,\\
m &:= 5,\\
n &:= 9,
\end{aligned}
$$

$$
A := \begin{bmatrix}
1 & -1 & 0 & 0 & 1 & 0 & 0 & 0 & 1\\
1 & 2 & 0 & 0 & 0 & 1 & 0 & 0 & 0\\
0 & 0 & 1 & 1 & 0 & 0 & -1 & 0 & 1\\
0 & 0 & -1 & 2 & 0 & 0 & 0 & -1 & 1\\
-2 & -1 & 2 & 4 & 0 & 0 & 0 & 0 & -3
\end{bmatrix}.
$$

Por (4.15) el problema de programación lineal anterior es reemplazado por:

$$\min x_9$$

$$
\begin{array}{rrrrrrrrrrl}
 & x_1 & -x_2 & & & & +x_5 & & & +x_9 & = 2\\
 & x_1 & +2x_2 & & & & & +x_6 & & & = 4\\
\text{sujeto a} & & & +x_3 & +x_4 & & & & -x_7 & +x_9 & = 2\\
 & & & -x_3 & +2x_4 & & & & & -x_8 \quad +x_9 & = 1\\
 & -2x_1 & -x_2 & +2x_3 & +4x_4 & & & & & -3x_9 & = 0
\end{array}
$$

Además $a := (1,1,1,1,1,1,1,1,1)^T$ es un punto inicial factible (como puede verificarse fácilmente).

Por (4.17) y (4.18) definimos las columnas de una nueva matriz $\hat{A}$ por medio de

$$
\begin{aligned}
\hat{A}_i &= a_i A_i \quad i = 1,\ldots,9,\\
\hat{A}_{10} &= -(2,4,2,1,0)^T.
\end{aligned}
$$

Como $a_i = 1, \quad i = 1,\ldots,9$ obtenemos, debido a esto

$$
\begin{aligned}
\hat{A}_i &= A_i \quad i = 1,\ldots,9,\\
\hat{A}_{10} &= -(2,4,2,1,0)^T.
\end{aligned}
$$

Luego $\hat{A}$ es dado por:

$$A := \begin{bmatrix} 1 & -1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & -2 \\ 1 & 2 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & -4 \\ 0 & 0 & 1 & 1 & 0 & 0 & -1 & 0 & 1 & -2 \\ 0 & 0 & -1 & 2 & 0 & 0 & 0 & -1 & 1 & -1 \\ -2 & -1 & 2 & 4 & 0 & 0 & 0 & 0 & -3 & 0 \end{bmatrix}.$$

Definamos $\hat{c}$ mediante

$$\begin{aligned} \hat{c}_i &= a_i c_i, \quad i = 1, \ldots, 9, \\ \hat{c}_{10} &= 0. \end{aligned}$$

Nuevamente, como $a_i = 1 \quad i = 1, \ldots, 9$ tenemos

$$\begin{aligned} \hat{c}_i &= c_i, \quad i = 1, \ldots, 9, \\ \hat{c}_{10} &= 0. \end{aligned}$$

Finalmente por (4.19) el problema de programación lineal, en la forma normal de Karmarkar es:

$$\min\ y_9$$

$$\text{sujeto a} \quad \begin{aligned} y_1 - y_2 + y_5 + y_9 - 2y_{10} &= 2, \\ y_1 + 2y_2 + y_6 - 4y_{10} &= 4, \\ +y_3 + y_4 - y_7 + y_9 - 2y_{10} &= 2, \\ -y_3 + 2y_4 - y_8 + y_9 - y_{10} &= 1, \\ -2y_1 - y_2 + 2y_3 + 4y_4 - 3y_9 &= 0, \\ y_1 + y_2 + y_3 + y_4 + y_5 + y_6 + y_7 + y_8 + y_9 + y_{10} &= 1, \end{aligned}$$

$$y_i \geq 0 \quad i = 1, \ldots, 10 \ ,$$

y donde además el punto $a^0 = \frac{1}{10}(1,1,1,1,1,1,1,1,1,1)^T$ es un punto inicial factible.

Finalmente resumimos en forma de un algoritmo los pasos necesarios para llevar un problema de programación lineal, a la forma normal de Karmarkar.

Algoritmo B

Paso 0: Definir los datos de entrada del problema de programación lineal (4.1): $A_0 \in \mathbb{R}^{k\times l}, d \in \mathbb{R}^l, v \in \mathbb{R}^l, \hat{b} \in \mathbb{R}^k$, y escogemos arbitrariamente $v_0, r_0 \in \mathbb{R}^l$ con $v_0 > 0, r_0 > 0$, $u_0, s_0 \in \mathbb{R}^k$ con $u_0 > 0, s_0 > 0$ y $\lambda \in \mathbb{R}^n$ con $\lambda = 1$.

Paso 1: Definir los datos de entrada del problema de programación lineal (4.15):

$$A := \begin{bmatrix} A_0 & 0 & I & 0 & \hat{b} - A_0 v_0 - s_0 \\ 0 & A_0^T & 0 & -I & d - A_0^T u_0 + r_0 \\ -d^T & b^T & 0 & 0 & -\hat{b}^T u_0 + d^T v_0 \end{bmatrix},$$

$$\begin{aligned} c &:= (\underbrace{0, \ldots, 0}_{2k+2l}, 1)^T, \\ b &:= (\hat{b}, d, 0)^T, \\ x &:= (v, u, s, r, \lambda)^T, \\ m &:= k + l + 1, \\ n &:= 2k + 2k + 1, \\ a &:= (v_0, u_0, s_0, r_0, 1)^T \end{aligned}$$

Paso 2: Definir los datos de entrada del problema lineal (4.19):

$$\begin{aligned} \hat{A}_i &:= a_i A_i, \qquad i = 1, \ldots, n, \\ A_{n+1} &:= -b, \\ c_i &:= a_i c_i, \qquad i = 1, \ldots, n, \\ \hat{c}_{n+1} &:= 0, \\ y &:= (y_1, y_2, \ldots, y_n). \end{aligned}$$

Paso 3: Calcular la solución del problema lineal (4.15):

Si y^* es solución óptima del problema lineal (4.19), entonces el punto x^* con coordenadas

$$x_i^* = \frac{a_i y_i^*}{y_{n+1}} \qquad i = 1, \ldots, n$$

es la solución óptima del problema lineal (4.15).

Paso 4: Recuperar la solución del problema lineal (4.1):

Del punto x^* escoger las 'l' primeras componentes, es decir

$$x_i^* = \frac{a_i y_i^*}{y_{n+1}} \qquad i = 1, \ldots, l$$

el vector cuyas componentes son estos números resuelve (4.1).

Capítulo 5

Algoritmo primal–dual de puntos interiores

5. Algoritmo primal–dual de puntos interiores

En este capítulo presentamos otra aproximación de punto interior para la programación lineal, que es conocido como el algoritmo primal–dual de puntos interiores. Este algoritmo trabaja directamente con el problema de programación lineal en forma standard y no requiere que el problema sea transformado a un formato especial como si ocurre con el algoritmo de Karmarkar y por lo tanto no se requiere que el valor óptimo (i.e. $c^T x^*$, x^* solución óptima) sea conocido. Este algoritmo trabaja simultáneamente con el problema primal y dual y resuelve el problema de la programación lineal en $\mathcal{O}(\sqrt{n}\ln(1/\varepsilon))$ iteraciones donde ε es la precisión deseada para la solución del problema de programación lineal. El desarrollo de este capítulo está basado en los artículos escritos por: R. Monteiro [13], M. Kojima [8], C. Gonzaga [4], C. Roos [16] y S. Wright [19].

5.1 Las condiciones de optimalidad de primer orden

En esta sección revisaremos las condiciones de optimalidad para problemas de optimización con restricciones de igualdad, las cuales serán formuladas usando la llamada **Función de Lagrange.**

Un problema de optimización no-lineal con restricciones de igualdad tiene la forma:

$$\begin{aligned} \min \quad & c(x) \\ \text{sujeto a}: \quad & g(x) = 0 \end{aligned} \tag{5.1}$$

donde $c : \mathbb{R}^n \mapsto \mathbb{R}$, $g : \mathbb{R}^n \mapsto \mathbb{R}^m$ son en general funciones de clase C^1.

Las condiciones de óptimo para (5.1) pueden ser formuladas minimizando la función de Lagrange que se define como:

$$L(x, \lambda) := c(x) - \lambda^T g(x) \tag{5.2}$$

donde $\lambda \in \mathbb{R}^m$ es conocido como el **vector de multiplicadores de Lagrange.**

Las condiciones de óptimo de primer orden para (5.1) son:

$$\begin{aligned} \nabla_x L(x,y) &= \nabla c(x) - (J(g(x)))^T \lambda &= 0 \\ \nabla_\lambda L(x,y) &= -g(x) &= 0 \end{aligned} \tag{5.3}$$

donde ∇_x y ∇_λ denotan las derivadas parciales de L respecto de x e y y donde $J(g(x))$ denota el jacobiano de $g(x)$.

Las condiciones de óptimo (5.3) son en general sólo condiciones necesarias para el óptimo. Una solución óptima de (5.1) satisface (5.3), pero una solución de (5.3) no es necesariamente una solución óptima para (5.1). Si el problema es convexo (es decir, $c(x)$ y $g(x)$ son funciones convexas) entonces es conocido que estas condiciones son necesarias y suficientes para el óptimo. En resumen hemos mostrado que un problema de programación no-lineal puede reducirse a un conjunto de ecuaciones no-lineales. Por esta razón, un método de solución para el problema de programación no-lineal (5.1) necesita un procedimiento para calcular una solución de las condiciones de óptimo (5.3). Las ecuaciones (5.3) son ecuaciones no-lineales, el método que usualmente se usa para resolver ecuaciones no-lineales es el **método de Newton** el cual es recordado en la siguiente sección.

5.2 El método de Newton

En esta sección, repasamos el método de Newton para resolver ecuaciones no-lineales, el material presentado aquí es bien conocido, por lo que esta sección será breve.

Si $f : \mathbb{R} \mapsto \mathbb{R}$ y debemos resolver

$$f(x) = 0,$$

dado un punto inicial x^0, calculamos una sucesión de soluciones aproximadas mediante

$$x^{k+1} := x^k - \frac{f(x^k)}{f'(x^k)}, \qquad k = 0, 1, 2, \ldots.$$

Paramos cuando $|f(x^k)| < \varepsilon$, donde ε es la precisión deseada para la solución.

Sea $f : \mathbb{R}^n \mapsto \mathbb{R}^n$ y consideremos el problema de resolver $f(x) = 0$.

El jacobiano de f en el punto x, $J(x)$ es definido como la matriz con componentes (i, j)

$$\left(\frac{\partial f_i}{\partial x_j}(x)\right)_{i,j}$$

y el método de Newton es:

$$x^{k+1} := x^k - [J(x^k)]^{-1} f(x^k)$$

Si hacemos $\Delta_k x := x^{k+1} - x^k$ entonces la ecuación anterior puede escribirse como:

$$J(x^k)\Delta_k x = -f(x^k) \tag{5.4}$$

Si en (5.3) definimos:

$$f(x, y) = \begin{bmatrix} \nabla c(x) - J(g(x))y \\ -g(x) \end{bmatrix}$$

entonces la dirección de búsqueda $\overline{\Delta x} = (\Delta x, \Delta y)$ en la k-ésima iteración del método de Newton para $f(x) = 0$ son dadas debido a (5.4) por:

$$\begin{bmatrix} \nabla^2 c(x^k) - \sum_{i=1}^{m} y_i \nabla^2 g_i(x^k) & -\nabla g(x^k) \\ -\nabla g(x^k)^T & 0 \end{bmatrix} \begin{bmatrix} \Delta_k x \\ \Delta_k y \end{bmatrix} = - \begin{bmatrix} \nabla c(x^k) - \nabla g(x^k) y^k \\ -g(x^k) \end{bmatrix} \tag{5.5}$$

5.3 La función barrera

Recordemos que el problema de programación lineal que estamos considerando es:

$$(P) \qquad \begin{aligned} \min \quad & c^T x \\ \text{sujeto a} \quad & Ax = b, \\ & x \geq 0, \end{aligned}$$

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

Con $\mathcal{F}_p^0$ denotamos el interior de la región factible del problema lineal (P), es decir

$$\mathcal{F}_p^0 = \{x \in \mathbb{R}^n : \quad Ax = b, \quad x > 0\}.$$

Tenemos la siguiente definición.

Definición 5.1 (ver C. Roos[16]) *Dado $\mu > 0$ se define $f_\mu : \mathcal{F}_p^0 \mapsto \mathbb{R}$ por*

$$f_\mu(x) := c^T x - \mu \sum_{j=1}^{n} \ln x_j,$$

f_μ es llamada la función barrera logarítmica para (P) con parámetro de barrera μ.

El siguiente teorema da una propiedad importante de la función barrera.

Teorema 5.1 (ver C. Roos[16]) *La función f_μ es estrictamente convexa.*

Prueba: Sabemos que:

$$\begin{aligned} f_\mu(x) &:= c^T x - \mu \sum_{j=1}^{n} \ln x_j \\ &= \sum_{j=1}^{n} (c_j x_j - \mu \ln x_j). \end{aligned}$$

La función $\ln(x_j)$ es cóncava, lo cual implica que $-\mu \ln(x_j)$ es convexa. Por esto, la función barrera es una suma de funciones convexas, lo cual implica que la función es convexa. Ahora veamos que f_μ es estrictamente convexa: Si definimos

$$X := \operatorname{diag}(x) = \begin{bmatrix} x_1 & 0 & \dots & 0 \\ 0 & x_2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & x_n \end{bmatrix},$$

$$e := \begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix},$$

entonces:

$$X^{-1}e = \begin{bmatrix} x_1^{-1} & 0 & \dots & 0 \\ 0 & x_2^{-1} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & x_n^{-1} \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix} = \begin{bmatrix} x_1^{-1} \\ x_2^{-1} \\ \vdots \\ x_n^{-1} \end{bmatrix}.$$

Entonces el resultado anterior, también lo podemos verificar del hecho que:

$$\nabla f_\mu = c - \mu X^{-1} e \qquad \text{y} \qquad \nabla^2 f_\mu = \mu X^{-2}.$$

Ahora sea $v \in \mathbb{R}^n \setminus \{0\}$ entonces:

$$v^T \nabla^2 f_\mu v = \mu v^T X^{-2} v = \mu \sum_{j=1} (v_j x_j^{-1})^2 > 0$$

para $x > 0$ y $\mu > 0$. Esto prueba que el Hessiano de f_μ es definido positivo lo cual implica que f_μ es estrictamente convexa. □

5.4 El problema primal–dual

En esta sección consideraremos el problema primal–dual y establecemos las condiciones de óptimo para este problema, además definimos conceptos importantes para uso posterior en el desarrollo del algoritmo.

Consideremos el problema primal en forma standard:

$$(P) \qquad \begin{aligned} \min \quad & c^T x, \\ \text{sujeto a} \quad & Ax = b, \\ & x \geq 0, \end{aligned}$$

donde c, $x \in \mathbb{R}^n$, $b \in \mathbb{R}^m$, $A \in \mathbb{R}^{m \times n}$, $\text{ran}(A) = m$.

El problema lineal (P) tiene asociado el problema dual:

$$(D) \qquad \begin{aligned} \max \quad & b^T y, \\ \text{sujeto a} \quad & A^T y + z = c, \\ & z > 0. \end{aligned}$$

El siguiente teorema, establece una relación entre las soluciones factibles de (P) y (D).

Teorema 5.2 (ver C. Roos[16]) *Sean x y z soluciones factibles para (P) y (D) respectivamente. Entonces: $c^T x - b^T y = x^T z \geq 0$. Además, si $x^T z = 0$, entonces x es una solución óptima de (P) y (y, z) es una solución óptima de (D).*

Prueba: La prueba es directa, se tiene que:

$$0 \leq x^T z = x^T (c - A^T y) = c^T x - (Ax)^T y = c^T x - b^T y.$$

De aquí: $b^T y \leq c^T x$. Esto implica que $c^T x$ es una cota superior para el valor objetivo óptimo de (D) y $b^T y$ es una cota inferior para el valor objetivo óptimo de (P) y además si $b^T y = c^T x$, entonces el par (x, y) es optimal $\square$

Nota: El número $c^T x - b^T y$ se llama brecha dual y al número $x^T z$ se le conoce como condición de complementaridad.

Condiciones de optimalidad: Debido al Teorema 5.2 podemos decir que una solución primal–dual factible (x, y, z) es óptima si y sólo si la brecha dual es cero. Nótese que desde que $x, z \geq 0$ $x^T z = 0$ si y sólo si $x_i z_i = 0$, $i = 1, \ldots, n$.

Esto da las siguientes condiciones de optimalidad (las cuales coinciden con las condiciones de Karush-Kuhn-Tucker) para el problema primal–dual, el cual es un sistema de ecuaciones que caracteriza el óptimo

$$(P-D) \quad \begin{cases} A^T y + z &= c \\ Ax &= b \\ x^T z &= 0 \\ x, z &\geq 0 \end{cases} \tag{5.6}$$

Damos a continuación, algunas notaciones que serán usadas posteriormente.

Notación 1: Los conjuntos factibles para los problemas (P), (D) y (P–D) serán denotados como $\mathcal{F}_p$, $\mathcal{F}_D$ y $\mathcal{F}$ respectivamente. Así se llama

$\mathcal{F}_p := \{x \in \mathbb{R}^n : \quad Ax = b, \quad x \geq 0\}$ el conjunto primal admisible y
$\mathcal{F}_D := \{(y, z) \in \mathbb{R}^m \times \mathbb{R}^n : \quad A^T y + z = c, \quad z \geq 0\}$ el conjunto dual admisible,
$\mathcal{F} := \{(x, y, z) \in \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}^n : \quad A^T y + z = c, \quad Ax = b, \quad x, z \geq 0\}$ el conjunto primal–dual admisible.

Los puntos interiores de $\mathcal{F}_p$, $\mathcal{F}_D$ y $\mathcal{F}$ serán denotados como: $\mathcal{F}_p^o$, $\mathcal{F}_D^o$, $\mathcal{F}^o$ respectivamente. Así:

$$\mathcal{F}_p^o := \{x \in \mathbb{R}^n : \quad Ax = b, \quad x > 0\},$$

$$\mathcal{F}_D^\circ := \{(y,z) \in \mathbb{R}^m \times \mathbb{R}^n : \quad A^T y + z = c, \quad z > 0\},$$
$$\mathcal{F}^\circ := \{(x,y,z) \in \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}^n : \quad A^T y + z = c, \quad Ax = b, \quad x, z > 0\}.$$

Estos conjuntos se considerarán no vacíos.

Notación 2: Dados los vectores x, z con X, Z. Denotaremos la matriz diagonal definida por estos vectores. El símbolo e denotará el vector de unos, con dimensión dada por el contexto.

5.4.1 Lagrangeano y condiciones de optimalidad de primer orden

Apliquemos el método de la función barrera logarítmica a el problema (P). La función barrera logarítmica para el problema (P) es:

$$f_\mu := c^T x - \mu \sum_{j=1}^{n} \ln(x_j)$$

y esto convierte al problema (P) en:

$$P_\mu \quad \begin{cases} \min : \quad c^T x - \mu \sum_{j=1}^{n} \ln(x_j), \\ \text{sujeto a :} \qquad Ax = b. \end{cases}$$

El problema lagrangeano para este problema es:

$$\min L(x,\lambda) = c^T x - \mu \sum_{j=1}^{n} \ln(x_j) - \lambda^T (Ax - b),$$

las condiciones de óptimo de primer orden, se obtienen derivando $L(x,\lambda)$ respecto de x y λ:

$$\begin{aligned} \nabla_x L(x,\lambda) = 0 \implies c - \mu X^{-1} e - A^T \lambda &= 0 \\ \nabla_\lambda L(x,\lambda) = 0 \implies Ax - b &= 0 \end{aligned}$$

En el problema (D) despejamos z de:

$$A^T y + z = c \implies z = c - A^T y.$$

Como la variable y no está restringida en signo y λ tampoco lo está, entonces podemos hacer $\lambda = y$, es decir $z = c - A^T y = c - A^T \lambda$. Luego tenemos las siguientes condiciones necesarias para (P_μ) en combinación con (D):

$$\begin{cases} z - \mu X^{-1} e &= 0, \\ Ax - b &= 0, \\ c - A^T y - z &= 0. \end{cases} \tag{5.7}$$

(5.7) puede escribirse también como:

$$\begin{cases} Xz &= \mu e, \\ Ax &= b, \\ A^T y + z &= c, \end{cases} \tag{5.8}$$

pero desde que:

$$Xz = \begin{bmatrix} x_1 & 0 & \dots & 0 \\ 0 & x_2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & x_n \end{bmatrix} \begin{bmatrix} z_1 \\ z_2 \\ \vdots \\ z_n \end{bmatrix} = \begin{bmatrix} x_1 z_1 \\ x_2 z_2 \\ \vdots \\ x_n z_n \end{bmatrix},$$

(5.8) se deja escribir como:

$$S_\mu(x, y, z) \quad \begin{cases} x_i z_i &= \mu \quad i = 1, \dots, n \\ Ax &= b \\ A^T y + z &= c \\ x, z &\geq 0. \end{cases}$$

$S_\mu(x, y, z)$ es llamado también sistema perturbado de Karush-Kuhn-Tucker (K-K-T), porque son idénticas a las condiciones (K-K-T) para (P) las cuales por (5.6) son:

$$\begin{array}{l} \text{Condiciones (K-K-T)} \\ \text{para el} \\ \text{problema (P)} \end{array} \quad \begin{cases} x_i z_i &= 0 \quad i = 1, \dots, n \\ Ax &= b \\ A^T y + z &= c \\ x, z &\geq 0, \end{cases}$$

excepto que las condiciones de complementaridad $x_i z_i = 0$ han sido perturbadas por μ (si $\mu = 0$ son idénticas a las condiciones de complementaridad). De este modo una solución de $S_\mu(x, y, z)$ para μ suficientemente pequeño es una buena aproximación a la solución óptima para el problema (P). Como fue probado en el **Teorema 5.1** que la función barrera logarítmica

$$f_\mu(x) := c^T x - \mu \sum_{j=1}^{n} \ln(x_j)$$

es estrictamente convexa. Además el conjunto $\{x \in \mathbb{R}^n : \ Ax = b\}$ es convexo, es decir, el problema

$$\begin{aligned} \min \quad & c^T x - \mu \sum_{j=1}^{n} \ln(x_j) \\ \text{sujeto a:} \quad & Ax = b, \\ & c, x \in \mathbb{R}^n, \ b \in \mathbb{R}^m, \ A \in \mathbb{R}^{m \times n}, \end{aligned}$$

es un problema convexo, entonces las condiciones de primer orden (i.e, el sistema $S_\mu(x, y, z)$) son condiciones necesarias y suficientes para optimalidad.

5.4.2 Curva central

En esta sección, definimos el importante concepto de **curva central.**

Por lo argumentado en la sección previa, las condiciones de primer orden son condiciones necesarias y suficientes para la optimalidad del problema (P_μ) y (D). Luego para cada $\mu > 0$ el sistema $S_\mu(x, y, z)$ tiene una única solución $(x(\mu), y(\mu), z(\mu))$ y define una curva llamada **curva central**, esto es precisamente el contenido de la siguiente definición.

Definición 5.2 (ver C. Gonzaga[4]) *La curva central se define como el conjunto de puntos* $(x(\mu), y(\mu), z(\mu))$ *tal que:*

$$\mathcal{C}: \quad \begin{cases} A^T y(\mu) + z(\mu) &= c \\ Ax(\mu) &= b \\ x_i(\mu) z_i(\mu) &= \mu \qquad i = 1, 2, \ldots, n \\ x, z &> 0, \end{cases} \tag{5.9}$$

es decir $(x(\mu), y(\mu), z(\mu)) \in \mathcal{C} \Longleftrightarrow (x(\mu), y(\mu), z(\mu))$ *satisface (5.9) con* $\mu > 0$.

Desde que: $x_i(\mu)z_i(\mu) = \mu \qquad i = 1, 2, \ldots, n$, tenemos

$$\sum_{i=1}^{n} x_i(\mu)z_i(\mu) = \sum_{i=1}^{n} \mu = n\mu,$$

entonces

$$x^T(\mu)z(\mu) = n\mu,$$

tomando $x^T(\mu)z(\mu)$ como un objetivo a minimizar queremos encontrar puntos sobre la curva central con $\mu \to 0$. En el algoritmo primal–dual los puntos iterados siguen a la curva $\mathcal{C}$ en la dirección de la disminución de μ.

El algoritmo inicia con un valor $\mu_0 > 0$, un punto inicial (x^0, y^0, z^0) y genera:

a) una sucesión $\{\mu_k\}_{k\geq 1}$ con $\mu_k \to 0$,

b) una sucesión $\{(x^k, y^k, z^k)\}$ en $\mathcal{F}^\circ$, la cual (debido a que $\mu_k \to 0$) converge a la solución óptima del problema primal–dual (5.6).

Ahora el punto $(x^k, y^k, z^k) \in \mathcal{F}^\circ$ está en la curva $\mathcal{C}$ si y solo si $x_i^k z_i^k = \mu$ para todo $i = 1, \ldots, n$ pero en general los productos $x_i^k z_i^k$ de los puntos iterados serán distintos para cada i, y por lo tanto la condición $x_i^k z_i^k = \mu$ no es satisfecha exactamente, y es debido a que la curva central $\mathcal{C}$ es básicamente una trayectoria teórica, $\mathcal{C}$ está formada por todas las soluciones del sistema (5.9) esto es,

$$\mathcal{C} = \{(x(\mu), y(\mu), z(\mu)) : \quad \mu > 0\}.$$

En general no es posible obtener una representación paramétrica de $\mathcal{C}$; lo que el algoritmo genera son puntos próximos a esta trayectoria.

Por lo tanto es necesario trabajar con ciertas vecindades que limiten dichos puntos iterados. Necesitamos medir la desviación de los productos $x_i^k z_i^k$, del k-ésimo punto iterado, de la condición $x_i^k z_i^k = \mu$, la desviación es medida comparando los productos $x_i^k z_i^k$ con

$$\mu = \frac{(x^k)^T z^k}{n} = \frac{\sum_{i=1}^{n} x_i^k z_i^k}{n}$$

y para esto damos la siguiente definición.

Definición 5.3 (ver C. Gonzaga[4]) *Sea* $(x, y, z) \in \mathcal{F}^\circ$, $\mu = (x^k)^T z^k$. *Definimos la medida de* <u>proximidad</u> *por:*

$$\delta(x^k, y^k, z^k, \mu) := \frac{1}{\mu}\|X_k Z_k e - \mu e\|_2 = \frac{1}{\mu}\left\| \begin{pmatrix} x_1^k z_1^k \\ \vdots \\ x_n^k z_n^k \end{pmatrix} - \begin{pmatrix} \mu \\ \vdots \\ \mu \end{pmatrix} \right\|_2 \quad \square$$

Lema 5.3 (ver C. Gonzaga[4]) *Si* $\frac{1}{\mu}\|X_k Z_k e - \mu e\|_2 < 1$ *entonces* $x^k, z^k > 0$.

Prueba: Supongamos que existe i tal que $x_i^k = 0$ ó $z_i^k = 0$

$$\Longrightarrow \quad \frac{1}{\mu}\|X^k Z_k e - \mu e\|_2 \geq \frac{1}{\mu}|x_i^k z_i^k - \mu| = \frac{1}{\mu}|0 - \mu| = 1$$

lo cual es una contradicción. $\square$

Si en la definición 5.3 hacemos:

$$X_k Z_k e - \mu e = \mu p, \qquad \text{donde}$$

$$\|p\|_2 = \delta(x^k, y^k, z^k, \mu),$$

obtenemos, multiplicando por e^T la expresión $X_k Z_k e - \mu e = \mu p$, y aplicando la desigualdad de Cauchy–Schwarz:

$$(x^k)^T z^k \leq (n + \delta(x^k, y^k, z^k, \mu)\sqrt{n})\mu. \tag{5.10}$$

Damos a continuación la definición de vecindad $\mathcal{N}_2(\theta)$.

Definición 5.4 (ver C. Gonzaga[4]) *Sea* $\theta \in \langle 0, 1\rangle$, $\mu = \dfrac{(x^k)^T z^k}{n}$, *entonces* $\mathcal{N}_2(\theta)$ *es definido por:*

$$\begin{aligned} \mathcal{N}_2(\theta) &:= \{(x^k, y^k, z^k) \in \mathcal{F}^\circ : \quad \|X_k Z_k e - \mu e\|_2 \leq \theta\mu\} \\ &= \{(x^k, y^k, z^k) \in \mathcal{F}^\circ : \quad \delta(x^k, y^k, z^k, \mu) \leq \theta\}. \quad \square \end{aligned}$$

El algoritmo primal–dual generará puntos en $\mathcal{N}_2(\theta)$, **(teorema 5.9)**

Ilustramos el caso $n = 2$, donde tenemos:

$$\mathcal{N}_2(\theta) = \{(x^k, y^k, z^k) \in \mathcal{F}^\circ : \quad \|X_k Z_k e - \mu e\|_2 \leq \theta\mu, \ \ \theta \in \langle 0, 1\rangle\}$$

entonces:

$$\begin{aligned}
(x_1^k z_1^k - \mu)^2 + (x_2^k z_2^k - \mu)^2 &\leq \theta^2\mu^2 \\
\Longrightarrow \quad \left(\frac{2x_1^k z_1^k - x_1^k z_1^k - x_2^k z_2^k}{2}\right)^2 + \left(\frac{2x_2^k z_2^k - x_1^k z_1^k - x_2^k z_2^k}{2}\right)^2 &\leq \theta^2\mu^2 \\
\Longrightarrow \quad \left(\frac{x_1^k z_1^k - x_2^k z_2^k}{2\mu}\right)^2 + \left(\frac{x_2^k z_2^k - x_1^k z_1^k}{2\mu}\right)^2 &\leq \theta^2 \\
\Longrightarrow \quad \left(\frac{x_1^k z_1^k - x_2^k z_2^k}{x_1^k z_1^k + x_2^k z_2^k}\right)^2 &\leq \theta^2 \\
\Longrightarrow \quad -\sqrt{\theta} \leq \frac{x_1^k z_1^k - x_2^k z_2^k}{x_1^k z_1^k + x_2^k z_2^k} &\leq \sqrt{\theta}.
\end{aligned}$$

De $-\sqrt{\theta} \leq \dfrac{x_1^k z_1^k - x_2^k z_2^k}{x_1^k z_1^k + x_2^k z_2^k}$ podemos deducir:

$$\begin{aligned}
-\sqrt{\theta}x_1^k z_1^k - \sqrt{\theta}x_2^k z_2^k + x_2^k z_2^k &\leq x_1^k z_1^k \\
\Longrightarrow \quad (1-\sqrt{\theta})x_2^k z_2^k &\leq (1+\sqrt{\theta})x_1^k z_1^k \\
\Longrightarrow \quad x_2^k z_2^k &\leq \frac{1+\sqrt{\theta}}{1-\sqrt{\theta}} x_1^k z_1^k \qquad (\alpha)
\end{aligned}$$

De $\dfrac{x_1^k z_1^k - x_2^k z_2^k}{x_1^k z_1^k + x_2^k z_2^k} \leq \sqrt{\theta}$ deducimos:

$$\begin{aligned}
x_1^k z_1^k - x_2^k z_2^k &\leq \sqrt{\theta}x_1^k z_1^k + \sqrt{\theta}x_2^k z_2^k \\
\Longrightarrow \quad (1-\sqrt{\theta})x_1^k z_1^k &\leq (1+\sqrt{\theta})x_2^k z_2^k \\
\Longrightarrow \quad \left(\frac{1-\sqrt{\theta}}{1+\sqrt{\theta}}\right) x_1^k z_1^k &\leq x_2^k z_2^k \qquad (\beta)
\end{aligned}$$

de (α) y (β)

$$\left(\frac{1-\sqrt{\theta}}{1+\sqrt{\theta}}\right) x_1^k z_1^k \leq x_2^k z_2^k \leq \left(\frac{1+\sqrt{\theta}}{1-\sqrt{\theta}}\right) x_1^k z_1^k$$

Los límites superior e inferior son rectas y como

$$\lim_{\theta \to 0^+} \frac{1-\sqrt{\theta}}{1+\sqrt{\theta}} = 1 = \lim_{\theta \to 0^+} \frac{1+\sqrt{\theta}}{1-\sqrt{\theta}},$$

decimos que los bordes de esta vecindad $\mathcal{N}_2(\theta)$ convergen a la curva central por arriba y por abajo. Gráficamente podemos ver éste proceso en la figura (5.1)

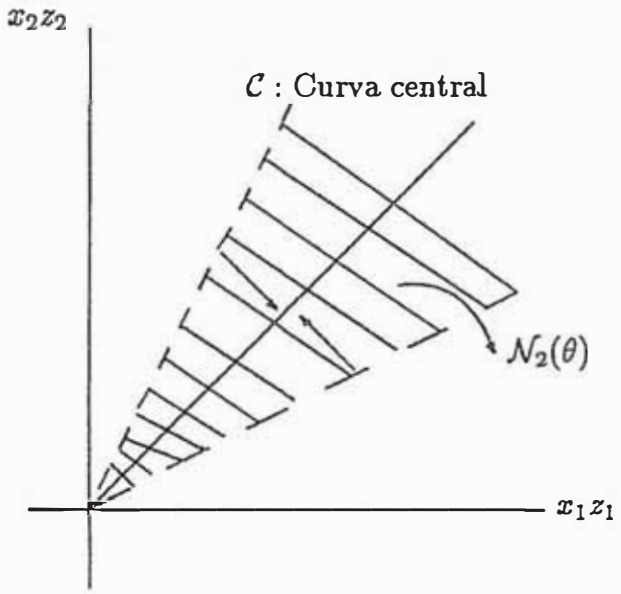


Figura 5.1: Vecindad $\mathcal{N}_2(\theta)$

5.4.3 Desarrollo del algoritmo primal–dual

En esta sección desarrollamos el algoritmo primal–dual y este desarrollo lo haremos teniendo como base lo presentado en las secciones anteriores.

Recordemos que el sistema $S_\mu(x, y, z)$

$$\begin{cases} Xz &= \mu e \\ Ax - b &= 0 \\ c - A^T y - z &= 0 \\ x, z &\geq 0, \end{cases}$$

daba condiciones necesarias y suficientes para optimalidad del problema P_μ. Ahora fijemos el parámetro real $\sigma \in \langle 0, 1 \rangle$, el cual como veremos posteriormente tiene relación con el

tamaño de paso α y μ y consideremos ahora el sistema:

$$\begin{cases} Xz = \sigma\mu e, \\ Ax - b = 0, \\ c - A^T y - z = 0, \\ x, z \geq 0. \end{cases}$$

Para resolver este sistema por el metodo de Newton definimos la función no–lineal $F_\sigma : \mathbb{R}^{2n+m} \longmapsto \mathbb{R}^{2n+m}$

$$F_\sigma(x, y, z) := \begin{bmatrix} Xz - \sigma\mu e \\ Ax - b \\ c - A^T y - z \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

Asumamos que (x^k, y^k, z^k) es dado con $x^k > 0$, $z^k > 0$, entonces una iteración del método de Newton aplicado al sistema:

$$F_\sigma(x, y, z) = 0$$

es igual a

$$\nabla F_\sigma(x^k, y^k, z^k) \begin{bmatrix} \Delta_k x \\ \Delta_k y \\ \Delta_k z \end{bmatrix} = -F_\sigma(x^k, y^k, z^k),$$

donde:

$$\nabla F_\sigma(x^k, y^k, z^k) = \begin{bmatrix} Z_k & 0 & X_k \\ A & 0 & 0 \\ 0 & -A^T & -\mathrm{I} \end{bmatrix}.$$

Entonces las direcciones de Newton obtenidas a partir de los puntos x^k, y^k, z^k se obtienen del siguiente sistema lineal:

$$\begin{bmatrix} Z_k & 0 & X_k \\ A & 0 & 0 \\ 0 & -A^T & -\mathrm{I} \end{bmatrix} \begin{bmatrix} \Delta_k x \\ \Delta_k y \\ \Delta_k z \end{bmatrix} = - \begin{bmatrix} X_k z^k - \sigma\mu e \\ Ax^k - b \\ c - A^T y^k - z^k \end{bmatrix}.$$

Pero como x^k , y^k , z^k es factible para (P) y (D) entonces:

$$\begin{cases} Ax^k - b = 0, \\ c - A^T y^k - z^k = 0, \\ x \geq 0 \quad , \quad z \geq 0. \end{cases}$$

Asi el sistema anterior se reduce a:

$$\begin{bmatrix} Z_k & 0 & X_k \\ A & 0 & 0 \\ 0 & -A^T & -\mathrm{I} \end{bmatrix} \begin{bmatrix} \Delta_k x \\ \Delta_k y \\ \Delta_k z \end{bmatrix} = - \begin{bmatrix} X_k z^k - \sigma\mu e \\ 0 \\ 0 \end{bmatrix}.$$

De aquí obtenemos:

$$Z_k \Delta_k x + X_k \Delta_k z = -X_k z^k + \sigma\mu e \quad , \tag{5.11}$$

$$A\Delta_k x = 0 \quad , \tag{5.12}$$

$$-A^T \Delta_k y - \Delta_k z = 0 \quad . \tag{5.13}$$

Recordemos que:

$$X_k = diag(x_1^k, \cdots, x_n^k),$$

$$Z_k = diag(z_1^k, \cdots, z_n^k),$$

y denotando por $v(\mu) = -X_k z^k + \sigma\mu e$, podemos escribir el sistema anterior como:

$$Z_k \Delta_k x + X_k \Delta_k z = v(\mu) \tag{5.14}$$

$$A\Delta_k x = 0 \tag{5.15}$$

$$-A^T \Delta_k y - \Delta_k z = 0 \tag{5.16}$$

Multiplicamos (5.14) por Z_k^{-1}, obteniendo

$$\Delta_k x = Z_k^{-1} v(\mu) - Z_k^{-1} X_k \Delta_k z. \tag{5.17}$$

De (5.16) resulta:

$$\Delta_k z = -A^T \Delta_k y.$$

Multiplicando (5.17) por A y considerando (5.15) obtenemos:

$$A\Delta_k x = AZ_k^{-1} v(\mu) - AZ_k^{-1} X_k \Delta_k z \ ,$$

$$\Longrightarrow \quad 0 = AZ_k^{-1} v(\mu) - AZ_k^{-1} X_k \Delta_k z \ ,$$

reemplazando $\Delta_k z = A^T \Delta_k y$ resulta

$$0 = AZ_k^{-1} v(\mu) + AZ_k^{-1} X_k A^T \Delta_k y \ ,$$

entonces tenemos

$$\Delta_k y = -(AZ_k^{-1} X_k A^T)^{-1} AZ_k^{-1} v(\mu),$$

$$\text{(por 5.16)} \quad \Delta_k z = A^T (AZ_k^{-1} X_k A^T)^{-1} AZ_k^{-1} v(\mu),$$

$$\text{(por 5.14)} \quad \Delta_k x = \{Z_k^{-1} - Z_k^{-1} X_k A^T (AZ_k^{-1} X_k A^T)^{-1} AZ_k^{-1}\} v(\mu).$$

Ahora sea:

$$x^{k+1} := x(\alpha) = x^k + \alpha \Delta_k x \ ,$$
$$y^{k+1} := y(\alpha) = y^k + \alpha \Delta_k y \ ,$$
$$z^{k+1} := z(\alpha) = z^k + \alpha \Delta_k z \ , \ \alpha \in \langle 0, 1],$$

el nuevo punto iterado.

Debemos tener en cuenta lo siguiente:

a) Denotemos la relación entre el valor del parámetro de barrera μ correspondiente al punto $(x(\alpha), y(\alpha), z(\alpha))$ con $\mu(\alpha)$.

b) Tenemos que construir una sucesión $\{\mu_k\}_{k\geq 1}$ con $\lim_{k\to\infty} \mu_k = 0$, pero que además satisfaga una relación adicional crucial para la convergencia del algoritmo.

c) Tenemos que elegir el tamaño de paso α.

d) Tenemos que garantizar que $(x(\alpha), y(\alpha), z(\alpha)) \in \mathcal{N}_2(\theta)$ y por lo tanto que este punto es factible.

e) Tenemos que garantizar la convergencia del algoritmo en tiempo polinomial.

Estos puntos serán estudiados en los lemas y teoremas que siguen.

Lema 5.4 (S. Wright [19]) *Sea $(\Delta_k x, \Delta_k y, \Delta_k z)$ la dirección generada por el método de Newton. Entonces se cumple:*

$$(\Delta_k x)^T \Delta_k z = 0,$$

$$\mu(\alpha) = (1 - \alpha(1 - \sigma))\mu.$$

Prueba:

De (5.13) resulta $\Delta_k z = -A^T \Delta_k y$

$$\Longrightarrow (\Delta_k x)^T \Delta_k z = -(\Delta_k x)^T A^T \Delta_k y$$

$$\Longrightarrow (\Delta_k x)^T \Delta_k z = \underbrace{-(A\Delta_k x)^T}_{=0 \text{ (por 5.12)}} \Delta_k y$$

$$\Longrightarrow (\Delta_k x)^T \Delta_k z = 0.$$

De (5.11) se tiene

$$\begin{bmatrix} z_1^k & & O \\ & z_2^k & \\ & \ddots & \\ O & & z_n^k \end{bmatrix} \begin{bmatrix} \Delta_k x_1 \\ \Delta_k x_2 \\ \vdots \\ \Delta_k x_n \end{bmatrix} + \begin{bmatrix} x_1^k & & O \\ & x_2^k & \\ & \ddots & \\ O & & x_n^k \end{bmatrix} \begin{bmatrix} \Delta_k z_1 \\ \Delta_k z_2 \\ \vdots \\ \Delta_k z_n \end{bmatrix}$$

$$= \begin{bmatrix} -x_1^k z_1^k + \sigma\mu \\ -x_2^k z_2^k + \sigma\mu \\ \cdots\cdots \\ -x_n^k z_n^k + \sigma\mu \end{bmatrix}$$

$$\Longleftrightarrow \quad \begin{bmatrix} z_1^k \Delta_k x_1 \\ z_2^k \Delta_k x_2 \\ \vdots \\ z_n^k \Delta_k x_n \end{bmatrix} + \begin{bmatrix} x_1^k \Delta_k z_1 \\ x_2^k \Delta_k z_2 \\ \vdots \\ x_n^k \Delta_k z_n \end{bmatrix} = \begin{bmatrix} -x_1^k z_1^k + \sigma\mu \\ -x_2^k z_2^k + \sigma\mu \\ \cdots\cdots \\ -x_n^k z_n^k + \sigma\mu \end{bmatrix},$$

entonces

$$\begin{array}{lcl} z_1^k \Delta_k x_1 + x_1^k \Delta_k z_1 & = & -x_1^k z_1^k + \sigma\mu \\ z_2^k \Delta_k x_2 + x_2^k \Delta_k z_2 & = & -x_2^k z_2^k + \sigma\mu \\ \cdots\cdots\cdots\cdots\cdots & \cdots\cdots & \cdots\cdots\cdots \\ z_n^k \Delta_k x_n + x_n^k \Delta_k z_n & = & -x_n^k z_n^k + \sigma\mu \end{array}$$

sumando miembro a miembro se cumple

$$\begin{array}{rcl} (z_1^k \Delta_k x_1 + z_2^k \Delta_k x_2 + \cdots + z_n^k \Delta_k x_n) & + & (x_1^k \Delta_k z_1 + x_2^k \Delta_k z_2 + \cdots + x_n^k \Delta_k z_n) \\ & = & -(x_1^k z_1^k + x_2^k z_2^k + \cdots + x_n^k z_n^k) + n\sigma\mu, \end{array}$$

lo cual puede escribirse como:

$$\begin{aligned}
(z^k)^T\Delta_k x + (x^k)^T\Delta_k z &= -(x^k)^T z^k + n\sigma\mu \\
&= -(x^k)^T z^k + n\sigma\left(\frac{(x^k)^T z^k}{n}\right) \\
\Longrightarrow \quad (z^k)^T\Delta_k x + (x^k)^T\Delta_k z &= -(x^k)^T z^k + \sigma(x^k)^T z^k \\
\Longrightarrow \quad (z^k)^T\Delta_k x + (x^k)^T\Delta_k z &= -(1-\sigma)(x^k)^T z^k. \qquad (5.18)
\end{aligned}$$

Ahora tenemos

$$x(\alpha) = x^k + \alpha\Delta_k x$$

$$z(\alpha) = z^k + \alpha\Delta_k z$$

$$\begin{aligned}
\Longrightarrow \quad (x(\alpha))^T z(\alpha) &= (x^k + \alpha\Delta_k x)^T(z^k + \alpha\Delta_k z) \\
&= (x^k)^T z^k + \alpha((x^k)^T\Delta_k z + (\Delta_k x)^T z^k) + \alpha^2 \underbrace{(\Delta_k x)^T\Delta_k}_{=0} z.
\end{aligned}$$

Por (5.18) queda:

$$\begin{aligned}
(x(\alpha))^T z(\alpha) &= (x^k)^T z^k - \alpha(1-\sigma)(x^k)^T z^k \\
&= (1-\alpha(1-\sigma))(x^k)^T z^k \\
\Longrightarrow \quad \mu(\alpha) &= (1-\alpha(1-\sigma))\mu \quad \square
\end{aligned}$$

Este lema implica que el valor del parámetro de barrera μ decrece con un incremento en α. Ahora cuando se calculan las direcciones empleando el metodo de Newton, se toma usualmente una longitud de paso unitaria ($\alpha = 1$), en este caso tendríamos

$$\mu_{k+1} = \sigma\mu_k \qquad\qquad k = 0, 1, \cdots$$

Ahora tenemos que $\sigma \in \langle 0, 1\rangle$ y una forma de escoger σ es $\sigma = 1 - \dfrac{\theta}{\sqrt{n}}$.
En el algoritmo primal-dual se toman longitudes de paso unitarias, y se toma $\theta = 0.4$, de este modo

$$\mu_{k+1} = \left(1 - \frac{0.4}{\sqrt{n}}\right)\mu_k.$$

Esta última relación es también necesaria para probar la convergencia del algoritmo (teorema 5.10). Despues de $k-$iteraciones tenemos $\mu_k = \left(1 - \frac{\theta}{\sqrt{n}}\right)^k \mu_0$, luego: $\lim_{k\to\infty} \mu_k = 0$. Esto implica que la sucesión $\{\mu_k\}_{k\geq 1}$ converge para cero que es lo que deseabamos. Finalmente en particular, si $0 < \alpha \leq 1$ y $\sigma = 1 - \frac{\theta}{\sqrt{n}}$ resulta

$$\mu_{k+1} \leq \left(1 - \frac{\theta}{\sqrt{n}}\right) \mu_k$$

Algoritmo Primal – Dual

Paso 0: Dado $\theta = 0.4$, $\sigma = 1 - \frac{0.4}{\sqrt{n}}$. Sea $(x^0, y^0, z^0) \in \mathcal{N}_2(\theta)$ un punto inicial, $\varepsilon \in \langle 0, 1\rangle$, $k := 0$.

Paso 1: Sean X_k, Z_k las matrices diagonales de x^k y z^k respectivamente.

Paso 2: Cálculo del valor de μ

$$\mu_{k+1} = \left(1 - \frac{0.4}{\sqrt{n}}\right) \mu_k$$

Paso 3: Cálculo de las direcciones de Newton

Calcular:

$$\begin{aligned}
\Delta_k x &= \{Z_k^{-1} - Z_k^{-1} X_k A^T (A Z_k^{-1} X_k A^T)^{-1} A Z_k^{-1}\}(-X_k z^k + \sigma \mu_k e) \\
\Delta_k y &= -(A Z_k^{-1} X_k A^T) A Z_k^{-1} (-X_k z^k + \sigma \mu_k e) \\
\Delta_k z &= A^T (A Z_k^{-1} X_k A^T)^{-1} A Z_k^{-1} (-X_k z^k + \sigma \mu_k e)
\end{aligned}$$

Paso 4: Cálculo de los nuevos puntos iterados

$$\begin{cases}
x^{k+1} := x^k + \Delta_k x \\
y^{k+1} := y^k + \Delta_k y \\
z^{k+1} := z^k + \Delta_k z
\end{cases}$$

Paso 5: Si $c^T x^k - b^T y^k < \varepsilon$

ir al **Paso 6**, sino $k := k + 1$,

retornar al **Paso 1**.

Paso 6:

$$\begin{cases} x^* := x^{k+1} \\ y^* := y^{k+1} \\ z^* := z^{k+1} \end{cases}$$

Fin del algoritmo.

Observación: A pesar de que el parámetro $\alpha \in \langle 0, 1]$ puede tener longitud variable, nosotros solo tomaremos $\alpha = 1$, existe una variante del algoritmo presentado aqui que escoge longitudes de paso variables y que tiene convergencia superlineal, este algoritmo es llamado el algoritmo predictor–corrector, una buena presentación de este algoritmo se encuentra en S. Wright [19]

5.4.4 Cálculo del punto inicial

Estudiaremos la manera de escoger un punto interior $(x^0, y^0, z^0) \in \mathcal{N}_2(0.4)$ de modo de poder inicializar el algoritmo.

Sea $(x^0, y^0, z^0) \in \mathbb{R}^{n+m+n}$ un punto arbitrario satisfaciendo $x^0 > 0$ $z^0 > 0$.

Sean k y λ dos números positivos suficientemente grandes, su magnitud sera especificada posteriormente. Junto con el par de problemas lineales (P) y (D), que deseamos resolver,

consideremos el par de problemas artificiales primal y dual:

$$(P') \quad \left\{ \begin{array}{lrcl} & \min \quad c^T x + k x_{n+1} & & \\ \text{sujeto a} & A x + (b - A x^0) x_{n+1} & = & b, \\ & (A^T y^0 + z^0 - c)^T x + x_{n+2} & = & \lambda, \\ & (x, x_{n+1}, x_{n+2}) & \geq & 0, \end{array} \right.$$

donde x_{n+1} y x_{n+2} son variables artificiales,

$$(D') \quad \left\{ \begin{array}{lrcl} & \max \quad b^T y + \lambda y_{m+1} & & \\ \text{sujeto a} & A^T y + (A^T y^0 + z^0 - c) y_{m+1} + z & = & c, \\ & (b - A x^0)^T y + z_{n+1} & = & x, \\ & y_{m+1} + z_{n+2}) & = & 0, \\ & (z, z_{n+1}, z_{n+2}) & \geq & 0, \end{array} \right.$$

donde y_{m+1} , z_{n+1} y z_{n+2} son variables artificiales.

Necesitamos tomar k y λ satisfaciendo

$$\begin{array}{rrcl} & \lambda & > & (A^T y^0 + z^0 - c)^T x^0, \\ \text{y} & k & > & (b - A x^0)^T y^0, \end{array}$$

para que $(x^0, x^0_{n+1}, x^0_{n+2})$ y $(y^0, y^0_{m+1}, z^0, z^0_{n+1}, z^0_{n+2})$ sean soluciones factibles de (P') y (D') respectivamente, donde

$$\begin{aligned} x^0_{n+1} &= 1 \quad , \\ x^0_{n+2} &= \lambda - (A^T y^0 + z^0 - c)^T x^0 \quad , \\ y^0_{m+1} &= -1 \quad , \\ z^0_{n+1} &= k - (b - A x^0)^T y^0 \quad , \\ z^0_{n+2} &= 1. \end{aligned}$$

De este modo podemos aplicar el algoritmo primal–dual a los problemas artificiales (P') y (D') con estas soluciones factibles iniciales.

Sean x^* y (y^*, z^*) soluciones óptimas de los problemas originales (P) y (D) respectivamente. El siguiente **teorema** , establece condiciones suficientes sobre las constantes k y λ para que el algoritmo aplicado a los problemas artificiales (P') y (D') tenga éxito en generar soluciones aproximadas de los problemas originales (P) y (D)

Teorema 5.5 (ver M. Kojima [8]) *Si además de*

$$\begin{aligned} & \lambda > (A^T y^0 + z^0 - c)^T x^0, \\ y \quad & k > (b - Ax^0)^T y^0, \end{aligned}$$

suponemos que

$$\begin{aligned} & \lambda > (A^T y^0 + z^0 - c)^T x^*, \\ y \quad & k > (b - Ax^0)^T y^*, \end{aligned}$$

entonces se cumple:

(a) Una solución factible $(\hat{x}, \hat{x}_{n+1}, \hat{x}_{n+2})$ de (P') es óptima si y solo si $\hat{x}$ es una solución óptima de (P) y $\hat{x}_{n+1} = 0$.

(b) Una solución factible $(\hat{y}, \hat{y}_{m+1}, \hat{z}, \hat{z}_{n+1}, \hat{z}_{n+2})$ de (D') es óptima si y solo si $(\hat{y}, \hat{z})$ es una solución óptima de (D) y $\hat{y}_{m+1} = 0$

Prueba: Definamos

$$\begin{aligned} x^\circ_{n+1} &= 0 \ , \\ x^\circ_{n+2} &= \lambda - (A^T y^0 + z^0 - c)^T x^* \ . \end{aligned}$$

Entonces $(x^*, x^\circ_{n+1}, x^\circ_{n+2})$ es una solución factible de (P'). Sea (x, x_{n+1}, x_{n+2}) una solucion factible arbitraria de (P') con $x_{n+1} > 0$. Entonces obtenemos

$$c^T x^* + k x^\circ_{n+1} = (y^*)^T b$$

$$
\begin{aligned}
&= (y^*)^T\{Ax + (b - Ax^0)x_{n+1}\} \\
&< (c - z^*)^T x + kx_{n+1} \quad \text{(debido a } \quad Ay^* + z^* = c \quad , \quad x_{n+1} > 0, \\
&\qquad\qquad\qquad\qquad\qquad\qquad k > (b - Ax^0)^T y^*) \\
&\leq c^T x + kx_{n+1} \qquad \text{(debido a } \quad (z^*)^T x \geq 0)
\end{aligned}
$$

Esto implica que (x, x_{n+1}, x_{n+2}) con $x_{n+1} > 0$ no puede ser una solución óptima de (P'). En otras palabras, toda solución minimal de (P') debe satisfacer $x_{n+1} = 0$. Por la continuidad, vemos también que $(x^*, x^*_{n+1}, x^*_{n+2})$ es una solución óptima de (P'). De este modo si una solución factible $(\hat{x}, \hat{x}_{n+1}, \hat{x}_{n+2})$ de (P') es óptima, entonces $\hat{x}_{n+1} = 0$ y $c^T\hat{x} = c^T x^*$.

Como $\hat{x}$ satisface todas las restricciones de (P), este punto debe ser una solución óptima de (P).

Reciprocamente, si $(\hat{x}, 0, \hat{x}_{n+2})$ es una solución factible de (P'), y si $\hat{x}$ es solución óptima de (P), entonces el valor objetivo $c^T\hat{x} + k\hat{x}_{n+1}$ coincide con el valor objetivo óptimo $c^T x^* + kx^*_{n+1}$, luego este es solución óptima de (P'). De este modo hemos probado (a). La prueba de la parte (b) es similar $\square$

Si tomamos $x^0 = e \quad , \quad y^0 = 0 \quad \text{y} \quad z^0 = e,$
entonces las condiciones

$$
\begin{aligned}
\lambda &> (A^T y^0 + z^0 - c)^T x^0 \quad , \\
k &> (b - Ax^0)^T y^0 \quad , \\
\lambda &> (A^T y^0 + z^0 - c)^T x^* \quad , \\
\text{y} \quad k &> (b - Ax^0)^T y^* \quad ,
\end{aligned}
$$

impuestas sobre las constantes k y λ darían

$$
\begin{aligned}
\lambda &> (e - c)^T e = n - c^T e \quad , \\
k &> 0 \quad , \\
\lambda &> (e - c)^T x^* = e^T x^* - c^T x^* \quad ,
\end{aligned}
$$

$$\text{y} \quad k \; > \; (b - Ae)^T y^* = b^T y^* - (y^*)^T Ae \quad .$$

Como $\mu_k = \dfrac{(x^k)^T z^k}{n}$ entonces $\mu_0 = 1$,

y $\|X_0 Z_0 e - e\|_2 = 0 \leq 0.4$,

entonces $(x^0, y^0, z^0) \in \mathcal{N}_2(\theta)$.

5.5 Convergencia del algoritmo Primal–Dual

En esta sección desarrollamos el teorema que establece la convergencia polinomial del algoritmo primal–dual, pero antes demostraremos que los puntos iterados (x^k, y^k, z^k) pertenecen a $\mathcal{N}_2(\theta)$, para $k = 1, 2, \ldots$. Ahora necesitamos probar el siguiente lema que será usado como lema auxiliar para probar un resultado que establece una cota para el producto $\Delta_k x_i \Delta_k z_i$

Lema 5.6 (ver Gonzaga [4]) *Sean u y v vectores arbitrarios en $\mathbb{R}^n$ con $u^T v \geq 0$. Entonces*

$$\|UVe\|_2 \leq 2^{-3/2} \|u + v\|_2,$$

donde

$$U = diag(u_1, \ldots, u_n), \qquad V = diag(v_1, \ldots, v_n).$$

Prueba: Vamos a usar el siguiente resultado

Sean $\alpha, \beta \in \mathbb{R}$:

$$\begin{aligned} \Longrightarrow \quad (\alpha - \beta)^2 = \alpha^2 + \beta^2 - 2\alpha\beta \; &\geq \; 0 \quad , \\ \Longrightarrow \quad (\alpha + \beta)^2 \; &\geq \; 4\alpha\beta \quad , \\ \text{entonces si} \quad \alpha\beta \geq 0 \Longrightarrow \quad |\alpha + \beta|^2 \; &\geq \; 4|\alpha\beta|. \end{aligned} \tag{5.19}$$

Despues tenemos:

$$\begin{aligned} 0 \leq u^T v \; &= \; \sum_{u_i v_i \geq 0} u_i v_i + \sum_{u_i v_i < 0} u_i vi \\ &= \; \sum_{i \in \mathcal{P}} |u_i v_i| - \sum_{i \in \mathcal{M}} |u_i v_i| \quad , \end{aligned} \tag{5.20}$$

donde $\mathcal{P} = \{i: \quad u_i v_i \geq 0\}, \quad \mathcal{M} = \{i: \quad u_i v_i < 0\}$.

$$\begin{aligned} \Longrightarrow \qquad \|UVe\|_2 &= ((u_1v_1)^2 + (u_2v_2)^2 + \cdots + (u_nv_n)^2)^{1/2} \\ &= \left[\sum_{i\in\mathcal{P}}(u_iv_i)^2 + \sum_{i\in\mathcal{M}}(u_iv_i)^2\right]^{1/2} \end{aligned}$$

Como $\|\cdot\|_2 \leq \|\cdot\|_1$, entonces

$$\begin{aligned} \|UVe\|_2 &= [\|(u_iv_i)_{i\in\mathcal{P}}\|_2^2 + \|(u_iv_i)_{i\in\mathcal{M}}\|_2^2]^{1/2} \\ &\leq [\|(u_iv_i)_{i\in\mathcal{P}}\|_1^2 + \|(u_iv_i)_{i\in\mathcal{M}}\|_1^2]^{1/2} \\ &\leq (2\|(u_iv_i)_{i\in\mathcal{P}}\|_1^2)^{1/2} \qquad \text{(por 5.20)} \\ &= [2(|u_1v_1| + \cdots + |u_lv_l|)^2]^{1/2} \\ &= \sqrt{2}\left(\sum_{i\in\mathcal{P}}|u_iv_i|\right) \\ &\leq \sqrt{2}\left(\frac{1}{4}\right)\sum_{i\in\mathcal{P}}(u_i+v_i)^2 \quad \text{(por 5.19)} \\ &\leq 2^{-3/2}\sum_{i=1}^{n}(u_i+v_i)^2 \\ &= 2^{-3/2}\|u+v\|_2^2. \quad \square \end{aligned}$$

En el **Lema 5.4** se vio que $(\Delta_k x)^T \Delta_k z = 0$, pero los productos individuales no son necesariamente iguales a cero. El lema siguiente da una cota para $\Delta_k x_i \Delta_k z_i$.

Lema 5.7 (ver **S. Wright [19]**) *Si* $(x^k, y^k, z^k) \in \mathcal{N}_2(\theta)$, *entonces se cumple:*

$$\|\Delta_k X \Delta_k Z e\|_2 \leq \frac{\theta^2 + n(1-\sigma)^2}{2^{3/2}(1-\theta)}\mu.$$

Prueba: Sea $D = X_k^{1/2} Z_k^{-1/2}$ una matríz diagonal. Por (5.11) sabemos que:

$$Z_k\Delta_k x + X_k \Delta_k z = -X_k z^k + \sigma\mu e.$$

Multiplicamos por $(X_kZ_k)^{-1/2}$ y obtenemos:

$$\begin{aligned} \underbrace{X_k^{-1/2}Z_k^{1/2}}\Delta_k x \;+\; \underbrace{X_k^{1/2}Z_k^{-1/2}}\Delta_k z &= (X_kZ_k)^{-1/2}(-X_kZ_ke+\sigma\mu e) \\ D^{-1}\Delta_k x \;+\; D\Delta_k z &= (X_kZ_k)^{-1/2}(-X_kZ_ke+\sigma\mu e) \end{aligned}$$

Como $\Delta_kX\Delta_kZ = \Delta_kXD^{-1}D\Delta_kZ = (D^{-1}\Delta_kX)(D\Delta_kZ)$

se cumple

$$\|\Delta_kX\Delta_kZe\|_2 = \|(D^{-1}\Delta_kX)(D\Delta_kZ)e\|_2. \tag{5.21}$$

Ahora podemos aplicar el lema anterior a esta ecuación para lo cual se debe hacer:

$u = D^{-1}\Delta_kx, \qquad v = D\Delta_kz,$ obteniendo

$$\begin{aligned} \|\Delta_kX\Delta_kZe\|_2 &\le 2^{-3/2}\|D^{-1}\Delta_kx + D\Delta_kz\|_2^2 \\ &= 2^{-3/2}\|(X_kZ_k)^{-1/2}(-X_kZ_ke+\sigma\mu e)\|_2^2 \\ &= 2^{-3/2}\sum_{i=1}^{n}\frac{(-x_i^kz_i^k+\sigma\mu)^2}{x_i^kz_i^k} \end{aligned}$$

Sea $m_0 = \min_i x_i^kz_i^k$

$$\begin{aligned} \Longrightarrow \quad 2^{-3/2}\sum_{i=1}^{n}\frac{(-x_i^kz_i^k+\sigma\mu)^2}{x_i^kz_i^k} &\le \frac{2^{-3/2}}{m_0}\sum_{i=1}^{n}(-x_i^kz_i^k+\sigma\mu)^2 \\ &= \frac{2^{-3/2}}{m_0}\|X_kZ_ke-\sigma\mu e\|_2^2. \end{aligned} \tag{5.22}$$

Ya que $(x^k, y^k, z^k) \in \mathcal{N}_2(\theta)$ se cumple

$$\mu - \min x_i^kz_i^k \le |\mu - \min x_i^kz_i^k| \le \|\mu e - X_kZ_ke\|_2 \le \mu\theta$$

se obtiene:

$$\min x_i^kz_i^k \ge (1-\theta)\mu \tag{5.23}$$

También:

$$\begin{aligned} e^T(X_kZ_ke-\mu e) &= (x^k)^Tz^k - \mu e^Te = n\mu - n\mu = 0 \\ \Longrightarrow \quad \|X_kZ_ke-\sigma\mu e\|_2^2 &= \|(X_kZ_ke-\mu e)+(1-\sigma)\mu e\|_2^2 \end{aligned}$$

$$
\begin{aligned}
&= \|X_kZ_ke - \mu e\|_2^2 + 2(1-\sigma)\mu \underbrace{e^T(X_kZ_ke - \mu e)}_{=0} \\
&\quad +(1-\sigma)^2\mu^2 e^T e \\
&\leq \theta^2\mu^2 + (1-\sigma)^2\mu^2 n. \qquad (5.24)
\end{aligned}
$$

Sustituyendo (5.23) y (5.24) en (5.22) resulta finalmente

$$\|\Delta_k X \Delta_k Z e\| \leq \frac{\theta^2 + n(1-\sigma)^2}{2^{3/2}(1-\theta)}\mu. \qquad \square$$

El siguiente lema establece cuánto se aleja el punto $(x(\alpha), y(\alpha), z(\alpha))$ de la curva central según la medida en norma 2.

Lema 5.8 (ver S. Wright [19]) *Si* $(x^k, y^k, z^k) \in \mathcal{N}_2(\theta)$, *se cumple*

$$
\begin{aligned}
\|X_k(\alpha)Z_k(\alpha)e - \mu(\alpha)e\|_2 &\leq |1-\alpha|\|X_kZ_ke - \mu e\|_2 + \alpha^2\|\Delta_k X \Delta_k Z e\|_2 \\
&\leq |1-\alpha|\theta\mu + \alpha^2\left[\frac{\theta^2 + n(1-\sigma^2)}{2^{3/2}(1-\theta)}\right]\mu.
\end{aligned}
$$

Prueba: Por (5.18)

$$Z\Delta x + X\Delta z = -XZe + \sigma\mu e,$$

en coordenadas

$$z_i^k\Delta_k x_i + x_i^k\Delta_k z_i = -x_i^k z_i^k + \sigma\mu, \quad i = 1, \ldots, n \qquad (5.25)$$

también

$$X_k(\alpha)Z_k(\alpha)e - \mu(\alpha)e = [x_i^k(\alpha)z_i^k(\alpha) - \mu(\alpha)], \quad i = 1, \ldots, n. \qquad (5.26)$$

$$\text{Del Lema 5.4} \qquad \text{sabemos que} \qquad \mu(\alpha) = (1 - \alpha(1-\sigma))\mu \qquad (5.27)$$

luego resulta de (5.26) y del **Lema 5.4**, para todo $i = 1, \ldots, n$:

$$
\begin{aligned}
x_i^k(\alpha)z_i^k(\alpha) - \mu(\alpha) &= x_i^k z_i^k + \alpha(z_i^k\Delta_k x_i + x_i^k\Delta_k z_i) + \alpha^2\Delta_k x_i\Delta_k z_i - (1-\alpha(1-\sigma))\mu \\
&= x_i^k z_i^k + \alpha(-x_i^k z_i^k + \sigma\mu) + \alpha^2\Delta_k x_i\Delta_k z_i - (1-\alpha(1-\sigma))\mu \\
&= x_i^k z_i^k(1-\alpha) + \alpha\sigma\mu + \alpha^2\Delta_k x_i\Delta_k z_i - (1-\alpha+\alpha\sigma)\mu \\
&= x_i^k z_i^k(1-\alpha) + \alpha^2\Delta_k x_i\Delta_k z_i - (1-\alpha)\mu,
\end{aligned}
$$

entonces

$$\begin{aligned}\|X_k(\alpha)Z_k(\alpha)e-\mu(\alpha)e\|_2 &= \left\|\begin{pmatrix} x_1^k z_1^k(1-\alpha)-(1-\alpha)\mu+\alpha^2\Delta_k x_1\Delta_k z_1 \\ \vdots \\ x_n^k z_n^k(1-\alpha)-(1-\alpha)\mu+\alpha^2\Delta_k x_n\Delta_k z_n\end{pmatrix}\right\|_2 \\ &\leq |1-\alpha|\|X_kZ_ke-\mu e\|_2+\alpha^2\|\Delta_kX\Delta_kZe\|_2 \\ &\leq |1-\alpha|\theta\mu+\alpha^2\left[\frac{\theta^2+n(1-\sigma)^2}{2^{3/2}(1-\theta)}\right]\mu \qquad \square\end{aligned}$$

El siguiente teorema define una relación entre θ y σ y prueba que aun tomando la longitud de paso máxima $\alpha=1$, a lo largo de la dirección generada por el método de Newton, los puntos iterados están en $\mathcal{N}_2(\theta)$.

Teorema 5.9 (ver S. Wright [19]) *Sea $\theta\in\langle 0,1\rangle$, y $\sigma\in\langle 0,1\rangle$ tales que satisfacen la siguiente relación*

$$\frac{\theta^2+n(1-\sigma)^2}{2^{3/2}(1-\sigma)}\leq\sigma\theta \tag{5.28}$$

Entonces si $(x^k,y^k,z^k)\in\mathcal{N}_2(\theta)$ tenemos que:

$$(x(\alpha),y(\alpha),z(\alpha))\in\mathcal{N}_2(\theta),\quad \forall\alpha\in\langle 0,1].$$

Prueba: Del lema 5.4

$$\begin{aligned}\|X_k(\alpha)Z_k(\alpha)e-\mu(\alpha)e\|_2 &= |1-\alpha|\theta\mu+\alpha^2\left[\frac{\theta^2+n(1-\sigma)^2}{2^{3/2}(1-\theta)}\right]\mu \\ &\leq (1-\alpha)\theta\mu+\alpha^2\sigma\theta\mu \\ &\leq (1-\alpha+\sigma\alpha) \\ &= (1-\alpha(1-\sigma))\theta\mu \\ &= \theta\mu(\alpha) \qquad (Lema\quad 5.4) \qquad (5.29)\\ \Longrightarrow\quad \|X_k(\alpha)Z_k(\alpha)e-\mu(\alpha)e\| &\leq \theta\mu(\alpha)\end{aligned}$$

esto establece que $(x(\alpha),y(\alpha),z(\alpha))$ satisface la condición de pertenecer a $\mathcal{N}_2(\theta)$.

Recordemos que:

$$\mathcal{N}_2(\theta)=\left\{(x^k,y^k,z^k)\in\mathcal{F}^\circ:\quad \|X_kZ_ke-\mu e\|_2\leq\theta\mu,\quad \theta\in\langle 0,1\rangle\right\}.$$

Para que $(x(\alpha), y(\alpha), z(\alpha)) \in \mathcal{N}_2(\theta)$,falta mostrar que $(x(\alpha), y(\alpha), z(\alpha)) \in \mathcal{F}^o$.

$$x(\alpha) = x^k + \alpha\Delta_k x, \quad y(\alpha) = y^k + \alpha\Delta_k y, \quad z(\alpha) = z^k + \alpha\Delta_k z,$$

$$\begin{aligned} \textit{entonces} \quad Ax(\alpha) &= Ax^k + \alpha \underbrace{A\Delta_k x}_{=0} = b \quad (\textit{por} \quad 5.12) \\ A^T y(\alpha) + z(\alpha) &= A^T(y^k + \alpha\Delta_k y) + z^k + \alpha\Delta_k z \\ &= \underbrace{A^T y^k + z^k}_{=c} + \alpha\underbrace{(A^T\Delta_k y + \Delta_k z)}_{=0 \ \textit{por}\ 5.13} \\ \textit{entonces} \quad A^T y(\alpha) + z(\alpha) &= c. \end{aligned}$$

Solo falta probar que: $x(\alpha) > 0, \quad z(\alpha) > 0 \quad \forall\alpha \in \langle 0, 1]$.

De

$$\begin{aligned} -x_i(\alpha)z_i(\alpha) + \mu(\alpha) \leq |x_i(\alpha)z_i(\alpha) - \mu(\alpha)| &\leq \|X(\alpha)Z(\alpha)e - \mu(\alpha)e\| \\ &\leq \theta\mu(\alpha) \end{aligned}$$

tenemos

$$x_i(\alpha)z_i(\alpha) \geq (1-\theta)\mu(\alpha) = (1-\theta)(1-\alpha(1-\sigma))\mu > 0$$

donde la desigualdad estricta es una consecuencia de $\theta \in \langle 0, 1\rangle$, $\alpha \in \langle 0, 1]$ y $\sigma \in \langle 0, 1\rangle$, luego $(x(\alpha), y(\alpha), z(\alpha)) \in \mathcal{F}^o$.

Nota: Para nuestro caso basta tomar $\alpha = 1$

Hasta este punto, la prueba de validez del algoritmo primal-dual, es casi completa, solo falta probar que, nuestra forma de escoger los parámetros θ y σ,

$$\theta = 0.4, \qquad\qquad \sigma = 1 - \frac{0.4}{\sqrt{n}},$$

satisface la condicion (5.28) para todo $n \geq 1$. Pero esto no es dificil de establecer, tenemos

$$\frac{0.4}{\sqrt{n}} \leq 1 - \frac{0.4}{\sqrt{2}(0.6)} \ \forall n \geq 1, \quad \Longrightarrow \quad \frac{0.4}{\sqrt{2}(0.6)} \leq 1 - \frac{0.4}{\sqrt{n}}, \quad \Longrightarrow \quad \frac{(0.4)^2}{\sqrt{2}(0.6)} \leq 0.4\left(1 - \frac{0.4}{\sqrt{n}}\right).$$

$$\text{Por} \quad \frac{(0.4)^2}{\sqrt{2}(0.6)} = \frac{2(0.4)^2}{2^{3/2}(0.6)} \qquad \text{resulta:} \qquad \frac{(0.4)^2 + n\left(\frac{0.4}{\sqrt{n}}\right)^2}{2^{3/2}(1-0.4)} \leq 0.4\left(1 - \frac{0.4}{\sqrt{2}}\right),$$

finalmente $$\frac{(0.4)^2 + n\left(1 - \left(1 - \frac{0.4}{\sqrt{n}}\right)\right)^2}{2^{3/2}(1 - 0.4)} \leq 0.4\left(1 - \frac{0.4}{\sqrt{2}}\right)$$

Ahora, la validez del algoritmo primal–dual es completa.

El siguiente **teorema**, prueba que el algoritmo primal–dual converge en tiempo polinomial.

Teorema 5.10 (ver S. Wright [19]) *Dado $\varepsilon \in \langle 0, 1\rangle$, supongamos que el punto inicial $(x^0, y^0, z^0) \in \mathcal{N}_2(0.4)$ en el algoritmo primal–dual satisface*

$$\mu_0 \leq \frac{1}{\varepsilon^\tau}$$

para alguna constante positiva τ. Entonces existe un índice K con $K = \mathcal{O}(\sqrt{n}\ln(1/\varepsilon))$ tal que:

$$\mu_k \leq \varepsilon \qquad \text{para todo} \qquad k > K$$

Prueba: Tenemos

$$\mu_{k+1} \leq \left(1 - \frac{0.4}{\sqrt{n}}\right)\mu_k.$$

Tomando logaritmos resulta:

$$\ln(\mu_{k+1}) \leq \ln\left(1 - \frac{0.4}{\sqrt{n}}\right) + \ln(\mu_k).$$

Por aplicación repetida de esta fórmula y usando $\mu_0 \leq \frac{1}{\varepsilon^\tau}$, tenemos

$$\ln(\mu_k) \leq k\ln\left(1 - \frac{0.4}{\sqrt{n}}\right) + \ln(\mu_0) \leq k\ln\left(1 - \frac{0.4}{\sqrt{n}}\right) + \tau\ln\left(\frac{1}{\varepsilon}\right).$$

La estimacion para la funcion logaritmo

$$\ln(1 + \beta) \leq \beta \qquad \forall \beta > -1,$$

implica que

$$\ln(\mu_k) \leq k\left(-\frac{0.4}{\sqrt{n}}\right) + \tau\ln\left(\frac{1}{\varepsilon}\right)$$

De este modo, el criterio de convergencia $\mu_k \leq \varepsilon$ es satisfecha si tenemos

$$k\left(-\frac{0.4}{\sqrt{n}}\right) + \tau\ln\left(\frac{1}{\varepsilon}\right) \leq \ln(\varepsilon)$$

Esta desigualdad se mantiene para todo k que satisface

$$k \geq K = \frac{1}{0.4}(\tau + 1)\sqrt{n} \ln\left(\frac{1}{\varepsilon}\right),$$

esto completa la prueba $\square$

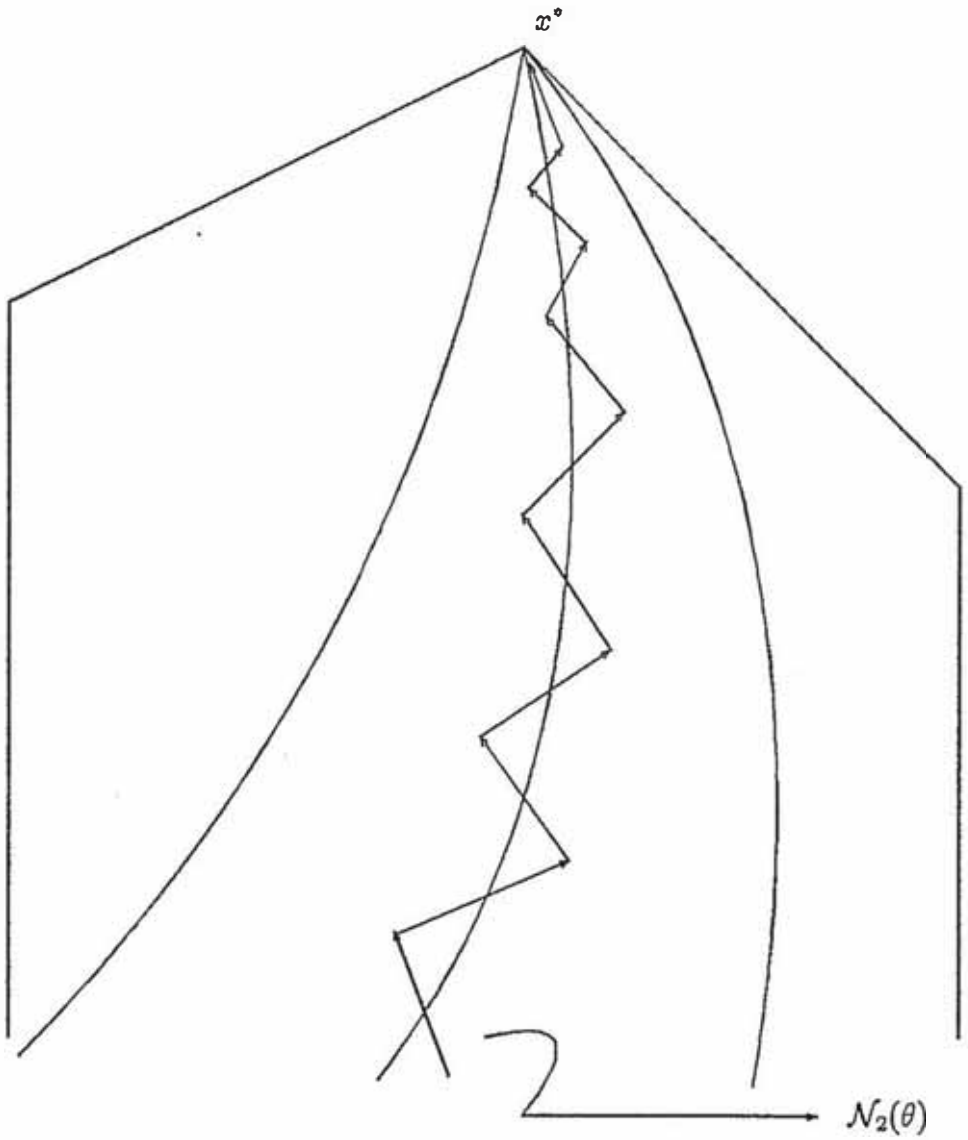


Figura 5.2: Convergencia de las iteraciones

Observación: Existe una variante del algoritmo primal–dual presentado aqui, llamado el **algoritmo predictor–corrector** el cual es actualmente el mas eficiente computacionalmente, y se encuentra implementado en diferentes software de uso comercial (HOPDM, CPLEX, OB1, etc). El algoritmo predictor–corrector tiene (entre otra cualidades) convergencia superlineal, un tratamiento de este y otros algoritmos primales–duales esta fuera del objetivo de esta tesis, el lector interesado puede consultar el excelente libro de S. Wright [19]

Capítulo 6

Resultados numéricos

6. Resultados numéricos

En este capítulo presentamos resultados numéricos que muestran la performance (menor número de iteraciones y tiempo de CPU) del algoritmo de Karmarkar y del algoritmo primal–dual en comparación con el algoritmo simplex. Los resultados numéricos que presentamos se encuentran en los artículos de I. Lustig, C. Monma, D. Shanho [12] y R. Marsten, M. Saltzman [10], estos artículos contienen resultados computacionales que comparan, la performance de estos algoritmos, utilizando para este fin software ya comercialmente disponible en el cual estos algoritmos se encuentran implementados.

6.1 Performance numérica

R. Marsten, M. Saltzman [10] mencionan en su articulo: "Una de las más sorprendentes e importantes características de los algoritmos de punto interior es que el número de iteraciones requeridas es insensible al tamaño del problema. Ellos requieren entre 20 y 80 iteraciones para ser resueltos. Existe evidencia experimental que indica que el número de iteraciones se incrementa con el logaritmo del número de variables. Dado este comportamiento, la superioridad de un algoritmo de punto interior comparado al algoritmo simplex depende de qué tan eficiente pueden ser realizados los pasos individuales del algoritmo".

Queremos mencionar aquí, que existe un artículo de I. Lusting, R. Marsten [9] que contiene resultados computacionales que comparan una implementación del algoritmo primal–dual predictor–corrector en un software denominado OB1 con el algoritmo simplex implementado en un software denominado MINOS 5.0 esta comparación es hecha sobre un conjunto de modelos de pequeño y mediano tamaño, y sus resultados prueban que cuando $m + n$ (número de ecuaciones+número de variables) $\geq$ 2500 los algoritmos de punto interior tienen mejor performance que el algoritmo simplex.

Antes de presentar los resultados numéricos damos algunos detalles de índole numérica, sobre la implementación de estos algoritmos.

6.1.1 Implementación

El trabajo en cada paso de los algoritmos de punto interior, es dominado por el requerimiento de resolver cierto sistema de ecuaciones.

En el algoritmo de Karmarkar buscamos d que satisfaga,

$$(AD^2A^T)d = AD^2c, \qquad \text{donde} \quad D := diag(x_1, \cdots, x_n),$$

en el algoritmo primal–dual buscamos d que satisfaga

$$(AD^2A^T)d = -AZ^{-1}(-Xz + \sigma\mu e),$$

donde $D = Z^{-1/2}X^{1/2}$.

Usaremos la abreviatura $(AD^2A^T)d = r$ para referirnos a cualquiera de estos sistemas, r es el vector del lado derecho que representa a AD^2c o a $-AZ^{-1}(-Xz + \sigma\mu e)$ segun sea el caso.

La matríz AD^2A^T es simétrica y definida positiva, cabe mencionar que las diferentes variantes de un algoritmo de punto interior tienen el sistema $(AD^2A^T) = r$ en común, solo la definición de D y del vector del lado derecho es diferente.

Ahora la solución del sistema

$$(AD^2A^T)d = r,$$

es obtenido usando la factorización de Cholesky de AD^2A^T,

$$AD^2A^T = LL^T,$$

donde L es triangular inferior, reduce el sistema $(AD^2A^T)d = r$ a $(LL^T)d = r$ lo cual puede resolverse con dos sustituciones triangulares

$$Ly = r \qquad \text{para hallar} \quad y \tag{6.1}$$

y luego por

$$L^Td = y \qquad \text{para hallar} \quad d. \tag{6.2}$$

Ahora como mencionan R. Marsten, M. Saltzman [10] la factorización de AD^2A^T toma cerca del 75% del total del tiempo de computo, mientras que la solución de los sistemas

(6.1), (6.2) toma cerca del 10% del tiempo. Estos mismos autores afirman que la aproximación de Cholesky para resolver el sistema $(AD^2A^T)d = r$ está basada en dos factores claves:

1. La cantidad de elementos no nulos en la matríz de Cholesky L no depende de la matríz D y es por esta razón la misma en cada iteración.

2. El número de elementos no nulos de L depende solo del ordenamiento de las filas de A

Asimismo mencionamos que el trabajo requerido en la factorización y la solución de los sistemas triangulares resultantes (6.1), (6.2), es directamente proporcional al número de elementos no nulos en L. Por lo tanto es crucial ordenar las filas de A de modo de obtener un factor L esparcido.

La factorización de Cholesky de una matriz usualmente produce lo que usualmente se conoce con el nombre de fill–in; que consiste en lo siguiente, algun elemento en la matriz triangular inferior L es no–nulo aun cuando la misma localización en la matriz original AD^2A^T es cero.

Existen algoritmos heuristicos para minimizar el fill–in y de este modo obtener un factor L esparcido, el mas empleado es el llamado **minimum degree** desarrollado por A. George, J. Liu [3], aqui el término degree refiere al número de elementos no–nulos en una columna de la matriz, excluyendo la diagonal.

Para ilustrar la importancia del ordénamiento de filas, consideremos el problema SHIPO4S, tabla **6.2**, con las filas de A en el orden dado, L tiene 39674 elementos no–nulos (61% densa); despues de aplicarle el algoritmo , L tiene 4400 elementos no–nulos (51% densa).

Además los autores mencionan que el tiempo usado por el algoritmo es usualmente modesto, esto es algo muy favorable para problemas de programación lineal grandes.

Aun cuando existen otros aspectos importantes relacionados con la implementación de estos algoritmos no profundizamos en ellos, en el libro de S. Wright [19] se estudia con regular detalle estos aspectos.

Presentamos a continuación los resultados numéricos.

6.1.2 Pruebas numéricas

A continuación presentamos los resultados numéricos obtenidos tanto por los algoritmos de punto interior como también por el algoritmo simplex, primeramente en la tabla 6.1 se compara el desempeño del algoritmo de Karmarkar (implementado en un software denominado KORBX) y el algoritmo primal–dual (implementado en el software denominado OB1). Debido a la mejor performance del algoritmo primal–dual es que en adelante solo compararemos el algoritmo primal–dual con el algoritmo simplex. La **Tabla 6.2** presenta un conjunto de problemas lineales que son generados aleatoriamente y forman parte de un conjunto de problemas de prueba que se hallan en la NETLIB. La primera columna de la tabla da el nombre del problema. Las otras tres columnas dan el número de filas, columnas y elementos no nulos respectivamente. La **Tabla 6.3** reporta los resultados del algoritmo simplex sobre estos problemas, en la implementación simplex, se usa el MINOS 5.0 que es la mejor implementación del simplex actualmente. La **Tabla 6.4** muestra los correspondientes resultados sobre estos mismos problemas obtenidos por el algoritmo primal–dual. En la **Tabla 6.5** se reportan los resultados computacionales de la implementación primal–dual comparada con el simplex.

Vemos de estas pruebas numéricas, la mejor performance del algoritmo primal–dual sobre el algoritmo simplex tanto en número de iteraciones como en tiempo de CPU, a medida que se incrementa el tamaño del problema. Quiza sea también necesario mencionar que los algoritmos de punto interior son poco afectados por degeneración lo cual no ocurre con el algoritmo simplex.

6.2 Conclusiones

El algoritmo de puntos interiores de Karmarkar marca un nuevo desarrollo en programación lineal. Este algoritmo y otros desarrollados a partir de la idea original de Karmarkar, como los algoritmos primales–duales, marcan un nuevo desarrollo en programación lineal, pero no podemos decir que un algoritmo ha venido para reemplazar al otro. Ambos juegan papeles complementarios dentro de la programación lineal, el algoritmo simplex continua siendo un algoritmo estandard para uso rutinario de la programación lineal. Sin embargo, los algoritmos de puntos interiores, se usarán más gradualmente en problemas grandes de programación lineal, los conceptos matemáticos para entender un algoritmo de punto interior son más elaborados que los necesarios para entender como funciona el algoritmo simplex.

Ademas es necesario mencionar que estos métodos ya han sido extendidos a otras areas como la programación convexa y programación entera con buenos resultados tanto teoricos como prácticos.

Bibliografía

[1] K. Borgwardt, The Simplex method, Springer Verlag, New York, 1987.

[2] P. Gazmuri, La complejidad computacional en problemas de optimización combinatoria: Conceptos básicos y nuevos enfoques. Apuntes de Ingeniería 9, 1982 pp. 61-86. Universidad de Chile.

[3] A. George, J. Liu, The evolution of the minimum degree ordering, SIAM Review, 31, 1989, pp. 1-19.

[4] C. Gonzaga, On the complexity of linear programming. Resenhas IME-USP 1996, Vol 2, N° 2, 197-207.

[5] M. Haimovich, I. Adler, The simplex method is very good! - On the expected number of pivot steps and related propiertes of random linear programs. Mathematical programming 69, 1993 pp. 215-228.

[6] N. Karmarkar, A new polynomial–time algorithm for linear programming, Combinatorica 4, 1984, pp. 373-395.

[7] V. Klee, G. Minty, How good is the simplex algorithm?, O. Shisha, ed. in Inequalities, Academic Press, New York, 1972, pp. 159-175.

[8] M. Kojima, A primal–dual interior point algorithm for linear programming, Progress in mathematical programming. Interior point and related methods. N. Meggido, ed. Springer Verlag. New York. 1989, pp. 29-47.

[9] I. Lustig, R. Marsten, Computational experience with a primal–dual interior point method for linear programming. Linear Algebra and its applications, 152, pp. 191-222, 1991.

[10] R. Marsten, M. Saltzman, Interior points methods for linear programming: Just call newton, lagrange and Fiacco and McCormick!. Interfaces 20: 4 July–August 1990 pp. 105-106.

[11] S. Mehrotra, A relaxed versión of Karmarkar's method. Mathematical programming 40, pp. 289-315, 1986.

[12] I. Lusting, C.Monma, D.Shanno, An implementation of a primal–dual interior point method for linear programming. ORSA journal on computing Vol 1, $N°$ 2, 1989, pp. 70-83.

[13] R. Monteiro, Interior path–following primal-dual algorithms. Part I: Linear programming. Mathematical programming, 44, 1989, pp. 27-41.

[14] S. Nash y A. Sofer, Linear and nonlinear programming, the McGraw-Hill Companies. Inc, 1996.

[15] H. Nikaido, Métodos matemáticos del análisis economico moderno. Editorial Vinces-Vives, 1978.

[16] C. Roos, Primal–dual algorithms for linear programming based on the logarithmic barrier method. Jota, Vol 83, $N°$ 1, pp. 1-26, 1994.

[17] J. Vial, A polinomial method of approximate centers for linear programming. Mathematical programming 54, 1992, pp. 295-306.

[18] B. Wah, Theory of algorithms and computation complexity with aplications to software design. University of California, Berkeley, 1998.

[19] S. Wright, Primal-dual interior points methods. SIAM, 1997.