UNIVERSIDAD ANAHUAC

ESCUELA DE ACTUARIA

CON ESTUDIOS INCORPORADOS A LA UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

ESTUDIO DE UN ALGORITMO DE PROGRAMACION LINEAL CON TIEMPO POLINOMIAL

T E S I S

QUE PARA OBTENER EL TITULO DE:

A C T U A R I O

P R E S E N T A N

ANNE JOSEPHINA BURDER BOTELLO

CECILIA ELENA PRIETO PIZARRO SUAREZ

MEXICO, D. F. 1987

INDICE

INTRODUCCION

Introducción

Cualquier actividad medianamente compleja requiere de un plan o programa que conduzca al logro de su objetivo. Este programa se puede elaborar en forma intuitiva; sin embargo, al elaborarse de esta manera, lo más probable es que diferentes personas propongan diferentes programas para lograr el mismo objetivo. Lo anterior hace que sea necesario usar un criterio para seleccionar el más apropiado de los distintos programas propuestos; generalmente se utiliza el criterio de la máxima eficiencia, ya sea utilizando el mínimo de recursos, el menor tiempo posible o el logro del máximo beneficio.

Para aplicar el criterio de la máxima eficiencia en métodos intuitivos es necesario evaluar numerosas alternativas, lo cual no siempre puede llevarse a cabo con la oportunidad requerida, haciendo que la aplicación de este criterio sea practicamente imposible.

Tomando el ejemplo de una empresa siderúrgica que tiene la posibilidad de vender más de mil productos; la producción de cada uno de ellos consume diferentes cantidades de materia prima, requiere distintos tiempos de proceso y su utilidad marginal es distinta para cada producto. En este caso, la complejidad de la elaboración de un programa de producción resulta evidente; si a esto le agregamos las restricciones tecnológicas propias de los diferentes procesos de producción, podemos concluir que no es posible definir intuitivamente un programa "óptimo" en el tiempo requerido.

La solución de este tipo de problemas se ha hecho posible gracias a la aplicación de métodos matemáticos para crear modelos que representen en forma aproximada el comportamiento del sistema como un todo. Aún así, es necesaria la utilización de una computadora, ya que estos problemas pueden tener una enorme cantidad de variables relacionadas.

Uno de los métodos matemáticos que mayor éxito ha tenido en este campo es la Programación Lineal, quizá la explicación del éxito que tiene se debe a la relativa facilidad para crear el modelo matemático que representa al sistema completo. Esta facilidad proviene de la suposición de "linealidad" que se introduce en dichos modelos, debido a que estamos acostumbrados a pensar en términos "lineales".

El concepto de linealidad consiste, básicamente, en esperar que en cualquier actividad que se emprenda, si se duplican los esfuerzos se duplicaran los resultados. Esto, como se sabe, no es estrictamente cierto y se tienen numerosos ejemplos de lo contrario; pero aún así, estos modelos han resultado sumamente fructíferos y de aplicación muy general.

El uso de la Programación Lineal ha producido beneficios de gran magnitud en las áreas donde ha sido aplicada adecuadamente. Se ha utilizado con gran éxito en industrias complejas como son la siderúrgica y la del petróleo, así como también en la industria de la alimentación, los transportes, la planeación económica y muchas otras áreas (que sería imposible enumerar exhaustivamente).

Dada la enorme importancia que tiene la solución de problemas de programación lineal en la actualidad, resulta de interés el dar a conocer de una manera accesible los métodos de que se dispone en el presente para resolver este tipo de problemas. Dicha importancia ha sido propiciada principalmente por los grandes avances en computación, los cuales han permitido utilizar modelos cada vez mayores y más complejos.

El algoritmo utilizado generalmente para la solución de un problema de programación lineal es el método simplex, que fué propuesto por G. B. Dantzig en 1947 y desde entonces ha tenido algunas mejoras y variantes. Aunque en la práctica el método simplex ha resultado muy eficiente, teóricamente se ha demostrado que, en casos desfavorables, el número de operaciones que requiere para llegar a una solución es exponencial con respecto al tamaño

del problema.

Recientemente han aparecido nuevos métodos que culminaron en 1984 en el "Algoritmo de Karmarkar". Este algoritmo teóricamente requiere de un número de operaciones que se puede expresar como polinomio con respecto al tamaño del problema; lo anterior implica que se podría observar una diferencia significativa al resolver problemas que involucren una gran cantidad de variables. El algoritmo de Karmarkar es un descubrimiento reciente en la programación lineal que incorpora ideas matemáticas ajenas a las tradicionalmente empleadas en este campo. Esta situación hace relativamente difícil la interpretación y el acceso al artículo original de Karmarkar; es por esto, que consideramos importante y oportuno dedicar nuestros esfuerzos a facilitar su difusión.

El objetivo de la presente tesis es el hacer una exposición accesible del algoritmo de Karmarkar y compararlo con el algoritmo del método simplex. Para ello, se inicia el primer capítulo, con una descripción de la programación lineal y su desarrollo a través del tiempo. En el segundo capítulo se procede a definir el concepto de algoritmo, presentando también una clasificación de los mismos. En el tercer capítulo se expone brevemente el algoritmo del método simplex. Así como, en el cuarto capítulo se hace una exposición detallada del algoritmo de Karmarkar para finalmente presentar una evaluación comparativa de ambos algoritmos en el último capítulo, incluyendo además sus respectivos programas de computación.

Queremos agradecer al Act. Jorge Lozano su amable dirección en la realización de esta tesis, así como sus valiosos consejos durante la revisión de la misma. Asimismo, expresamos nuestra gratitud al Ing. Carlos Meyer L., al Lic. Carlos Prieto M. y al M. en C. Manuel Román E. por su desinteresada ayuda.

Capítulo 1. Programación Lineal

1.1 Definición

1.2 Breve Bosquejo Histórico

1.3 Dualidad

1.1 Definición de la Programación Lineal

La Programación Lineal es una de las técnicas de análisis cuantitativo más diversificada y de mayor aplicación en el campo de la Investigación de Operaciones; forma parte de una rama de las matemáticas llamada Programación Matemática, y por medio de ella se pueden encontrar soluciones óptimas a problemas prácticos de gran complejidad.

A través de los problemas de programación lineal se obtiene la asignación óptima de un número limitado de recursos, m, que pueden ser hombres, máquinas y tierras entre otros; estos se asignan a un número n de actividades distintas cumpliendo con ciertas restricciones dadas. Dependiendo de los objetivos, pueden existir varias asignaciones factibles de los recursos. La programación lineal se encarga de encontrar la asignación que maximice o minimice alguna cantidad numérica tal como la utilidad o el costo; esto es, busca un resultado que cumpla mejor con el objetivo específico de entre todas las posibles alternativas factibles.

Los problemas de asignación de recursos, así como los relacionados con fenómenos económicos, son algunos de los problemas en los que resulta útil el uso de la programación lineal. La continua expansión de las aplicaciones de la programación lineal demuestra la importancia de ésta como marco para el planteamiento de problemas.

La programación lineal debe su auge en gran medida al desarrollo de las computadoras electrónicas, ya que estas permiten acelerar el proceso para obtener la solución del problema. Además, las computadoras se emplean actualmente en el gobierno, las empresas y los hogares, lo cual ha contribuido a que el número de usuarios aumente en gran medida, permitiendo así que la programación lineal se difunda en un campo más amplio, encontrándose nuevas aplicaciones de la misma.

Se usa el adjetivo lineal, para describir una relación entre dos o más variables que son directamente proporcionales. El término programación se refiere a ciertas técnicas matemáticas para llegar a la mejor solución posible, empleando recursos limitados.

Un problema general de programación lineal se puede describir como sigue:

Dado un conjunto de m desigualdades o ecuaciones lineales con n variables (restricciones), se desea encontrar los valores no negativos de estas variables que satisfagan dichas restricciones y, a la vez, que maximicen o minimicen alguna función lineal de estas variables.

La formulación del modelo matemático para el problema general de programación lineal, es la siguiente:

$$
\begin{aligned}
\text{Minimizar} \quad z = {} & c_1 x_1 + c_2 x_2 + \ldots + c_n x_n \\
\text{Sujeto a} \quad & a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n \geq b_1 \\
& a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n \geq b_2 \\
& \vdots \qquad\qquad\qquad\qquad \vdots \\
& a_{m1}x_1 + a_{m2}x_2 + \ldots + a_{mn}x_n \geq b_m \\
& x_1 , \quad x_2 , \ldots , \quad x_n \geq 0
\end{aligned}
$$

Donde $c_1 x_1 + c_2 x_2 + \ldots + c_n x_n$ es la función objetivo que debe minimizarse y se denota por z. Los escalares c_1 , c_2 ,..., c_n son los coeficientes de costo, y x_1 , x_2 ,..., x_n son las variables de decisión que deben determinarse. La desigualdad $a_{i1}x_1 + a_{i2}x_2 + \ldots + a_{in}x_n \geq b_i$ denota la i-ésima restricción. El vector columna cuya i-ésima compónente es b_i, se llama el vector del lado derecho y representa los requerimientos mínimos que deben satisfacerse. Las restricciones x_1 , x_2 ,..., $x_n \geq 0$ son llamadas restricciones de no negatividad. Las constantes a_{ij} , b_i y c_j, son parámetros conocidos del problema. Un conjunto de variables x_1 , x_2 ,..., x_n que satisfaga todas las restricciones se llama un punto factible o solución factible, y al conjunto de

todos estos puntos se le llama región o espacio factible. Aquel punto que pertenezca al espacio factible y minimice el valor de la función objetivo se le conoce como valor óptimo [Bazaara, 1981].

Utilizando notación más compacta, el problema se convierte en:

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

Donde x es un vector columna n-dimensional,
c^T es un vector renglón n-dimensional,
b es un vector columna m-dimensional y
A es una matriz m x n .

El problema de programación lineal se puede enunciar como sigue: entre todos los vectores factibles, encuéntrese aquel que minimiza (o maximiza) la función objetivo.

Para poder representar un problema de optimización como un problema de programación lineal, se requieren varios supuestos que están implícitos en la formulación del modelo. A continuación se explican brevemente estos supuestos para poder así evaluar que tan bien se aplica la programación lineal a la solución de un problema dado [Dantzig, 1963].

1. Supuesto de Proporcionalidad.

El supuesto de proporcionalidad es aquel en el que las relaciones entre las variables de la función objetivo y cada una de las restricciones deben ser lineales. Para que esto suceda, es necesario que el uso de los recursos sea directamente proporcional al nivel de cada actividad. Dada una variable x_j, su contribución al costo total es de $c_j x_j$ y su contribución a la i-ésima restricción es $a_{ij} x_i$. Esto significa que si se duplica el valor x_i entonces se duplica su contribución al costo total y a cada una de las restricciones.

2. Supuesto de Aditividad.

En el supuesto de aditividad se requiere que la cantidad total de recursos definida por el sistema sea igual a la suma de las cantidades que fluyen hacia adentro del sistema menos la suma de las que fluyen hacia afuera. Esta suposición garantiza que el costo total sea la suma de los costos individuales de cada actividad.

3. Supuesto de Divisibilidad.

El supuesto de divisibilidad asegura que las variables de decisión se pueden dividir en cualquier nivel fraccional de modo que se permitan valores no enteros para las variables de decisión; en otras palabras las variables tienen que ser continuas. Lo anterior representa una limitación de la programación lineal.

4. Supuesto Determinístico.

El supuesto determinístico es aquel en el que todos los parámetros del modelo (a_{ij} , b_i y c_j) son constantes conocidas. Los modelos de programación lineal se formulan para poder seleccionar alguna acción futura; por lo tanto, si los coeficientes se basan en predicciones, la información tenderá a ser inadecuada.

Es muy poco usual que un problema práctico satisfaga completamente todos los supuestos de la programación lineal. Sin embargo, el modelo de programación lineal es con frecuencia una de las representaciónes más prácticas del problema.

1.2 Breve Bosquejo Histórico de la Programación Lineal

Los problemas de optimización fueron conocidos desde los siglos XVII y XVIII, cuando grandes matemáticos como Newton, Leibnitz, Lagrange y Bernoulli propusieron soluciones a ellos, desarrollando el cálculo infinitesimal y el cálculo de variaciones [Simonnard, 1966].

En 1936, Wassily Leontieff propuso una estructura matricial simple a la cual llamó Inter-Industry Input-Output Model. Esta estructura conceptualmente muy sencilla puede ser implementada a detalle para su utilización en planeación práctica, ya que existe una correspondencia uno a uno entre los procesos de producción y los bienes producidos por los mismos, con lo que se logra una relación entre un modelo lineal y un sistema interindustrial [Dantzig, 1981].

Más tarde, surgió la necesidad de crear un modelo con muchas actividades alternas, el cual debería ser a gran escala y soluble. La principal dificultad al resolver este problema utilizando el análisis de actividades, fué la de compararar todas las posibles soluciones y seleccionar aquella que fuera "la mejor" por medio de algún criterio. Esto provocó que en 1946 G. B. Dantzig formulara un modelo para representar las relaciones tecnológicas que usualmente se dan en la práctica y, en lugar de una función que se maximizara o minimizara (función objetivo), existía un gran número de reglas empíricas dadas por la autoridad para elegir así un conjunto de soluciones factibles [Dantzig, 1981].

Para 1947, Dantzig decidió que el objetivo debería ser explícito, formulando el problema de planeación como un conjunto de axiomas, los cuales se referirían a las relaciones entre dos tipos de conjuntos: el primero era el conjunto de bienes a ser producidos o consumidos, y el segundo era el de actividades o procesos de producción en donde los bienes iban a ser utilizados en proporciones fijas, siempre y cuando estas proporciones fueran múltiplos no negativos de cada una de ellas. El sistema matemático resultante era el de minimizar una función lineal sujeta a restricciones (ecuaciones y desigualdades) lineales. Dantzig resolvió este problema por medio de un algoritmo que posteriormente tituló Método Simplex [Dantzig, 1981].

Desde 1939, el matemático y economista soviético L. V. Kantorovich formuló y resolvió un problema específico de programación lineal, siendo éste la organización y planeación de

la producción de una empresa. Hitchcock, en 1941, formula y resuelve el problema de transporte. En 1945, G. Stigler publica la formulación del problema de la dieta óptima [Ignizio, 1982].

El término "programación lineal" apareció por primera vez como título de una publicación en 1949. Ese mismo año, el método simplex fué presentado durante una conferencia de programación lineal organizada por la Cowles Comission for Research in Economics, y el procedimiento del método fué publicado posteriormente bajo la dirección de T. C. Koopmans..Los primeros resultados teóricos acerca de la dualidad, basados en notas de J. von Neumann, aparecen en la misma colección de escritos (Cowles Comission Monograph No. 13).

Después de 1948, paralelamente a la actividad de la Cowles Comission, un gran número de matemáticos y economistas contribuyeron individualmente o en grupos al desarrollo de diferentes aspectos de la programación lineal. En particular, la RAND Corporation con G. B. Dantzig y W. Orchard-Hays, así como también L. R. Ford, D. R. Fulkerson y D. Gale de la Universidad de Princeton, trabajaron principalmente en la teoría matemática de la programación lineal y en el desarrollo de redes de transporte auxiliándose con computadoras electrónicas. En 1951 aparece la programación dinámica con Wolfowitz, Arrow, Harris y Johnson. La teoría de redes y flujos comienza en 1954 cuando Ford, Fulkerson y Hoffman muestran las conexiones de ésta con la teoría de gráficas [Simonnard, 1966].

En 1958, R. E. Gomory desarrolla un método para el cálculo de la programación entera, en el cual se muestra como generar los hiperplanos de corte. Las técnicas de "branch and bound" de la programación entera fueron desarrolladas en 1960 por A. H. Land y A. G. Doig de la London School of Economics. A. W. Tucker, H. W. Kuhn y D. Gale del Instituto Carnegie de Tecnología, se dedicaron a estudiar ciertos aspectos teóricos como son: la degeneración, los errores de redondeo, el método simplex revisado, las variables acotadas y la función objetivo convexa separable. Egon Balas, en

1965, construye un algoritmo para resolver los problemas lineales donde las variables únicamente pueden tomar los valores de 0 ó 1 (binarios) [Ignizio, 1982].

A partir de 1960 muchos investigadores han propuesto métodos de solución a los modelos lineales con variables enteras y binarias, pero estos métodos han sido poco eficientes en la práctica, cuando el número de variables de este tipo es grande. También, se comenzó a estudiar la complejidad y la elegancia matemática de los algoritmos sin que esto significara un incremento en la capacidad práctica para resolver problemas [Ignizio, 1982].

Después del desarrollo del método simplex algunos analistas se propusieron encontrar un procedimiento computacional más eficiente. Esto motivó a Klee y Minty, en 1972, a construir un ejemplo en el que mostraban que el algoritmo del método simplex tiene crecimiento exponencial; esto solo provocó un interés académico, ya que en la práctica el método simplex muestra un tiempo polinomial [Garey,1979].

En 1978, el soviético L. G. Khachian desarrolló el primer algoritmo de tiempo polinomial para la programación lineal. Este resultado teórico es muy importante, pero no puede ser utilizado en la práctica ya que su convergencia es extremadamente lenta [Khachian, 1978].

Finalmente, en 1984, Narendra Karmarkar propone un algoritmo para el problema de programación lineal también con tiempo polinomial [Karmarkar, 1984a]

1.3 Dualidad

Uno de los conceptos más importantes asociados a la programación lineal es el de dualidad. Etimológicamente, dual

significa doble, por lo que a cada problema de programación lineal se le asocia un problema dual correspondiente.

En programación lineal la dualidad se define como sigue: Dado cualquier problema de programación lineal

$$\begin{aligned} \max \quad & z = c^T x \\ \text{sujeto a} \quad & A x \leq b \\ & x \geq 0 \end{aligned} \qquad (1)$$

Existe siempre otro problema de programación lineal asociado:

$$\begin{aligned} \min \quad & z = b^T w \\ \text{sujeto a} \quad & A^T w \geq c \\ & w \geq 0 \end{aligned} \qquad (2)$$

Si al problema (1) se le denomina forma canónica del primal (primal), entonces, el problema (2) se denominará forma canónica del dual (dual). Ambos se relacionan de manera que la solución óptima de un problema proporciona toda la información necesaria para determinar la solución óptima del otro; esto es, si ya se resolvió uno de los problemas, ambos están resueltos.

El problema dual es importante por varias razones, de las cuales se mencionarán tres: las variables del problema dual pueden proporcionar información importante acerca de la solución óptima del problema de programación lineal original; en algunos casos, el problema dual se puede resolver en un número menor de iteraciones reduciendo así el tiempo y costo de computadora; y, finalmente, la dualidad es muy útil cuando se investigan los cambios en los coeficientes o en la formulación de un problema, lo que se llama análisis de sensibilidad [Ignizio, 1982].

Las relaciones entre el primal y el dual en su forma canónica se resumen como sigue [Ignizio, 1982]:

1. El dual del dual es el primal.
2. El dual de un problema de maximización es un problema de

minimización.

3. Si el primal tiene una solución óptima, entonces el dual también; si el primal es no acotado, entonces el dual es no factible; si el primal es no factible, el dual podrá ser no acotado o no factible.
4. El valor óptimo de la función objetivo del primal es el mismo que el del dual.
5. La traspuesta de la matriz de restricciones en el primal es la matriz de restricciones del dual.
6. Los coeficientes de la función objetivo del primal aparecen como las constantes del lado derecho del dual, y las constantes de las restricciones del primal son los coeficientes de la función objetivo del dual.
7. El sentido de las desigualdades del dual es contrario al sentido de las desigualdades en el primal.
8. A cada restricción del primal le corresponde una variable del dual y viceversa.

Para convertir el primal a su forma dual se siguen las siguientes reglas:

Primero, convertir el primal a su forma canónica. A la matriz que resulta de esta transformación se le llama A. En segundo lugar, esta matriz se transpone de lo que resulta A^T, la cual se multiplica por el vector w de variables duales. Finalmente, los coeficientes de la función objetivo en el primal pasan a ser las constantes del dual y viceversa. Quedando así:

$$\begin{aligned} \min \quad z &= b^T w \\ \text{sujeto a} \quad A^T w &\geq c \\ w &\geq 0 \end{aligned}$$

Capitulo 2. Algoritmos

2.1 Definición

2.2 Clasificación

2.1 Definición de Algoritmos

La palabra algoritmo proviene del nombre de un autor persa llamado Abu Ja'far Mohammed ibn Musa al Khowarizmi nacido en 825 a.d. [Horowitz,1984]. En general, un algoritmo describe el método por medio del cual debe realizarse una tarea; este consta de una secuencia de pasos, que si se efectúan con fidelidad, darán como resultado que la tarea o proceso se realice.

Alguna de las cualidades más importantes de los algoritmos son las siguientes: representan información mediante estructuras que permiten seguir una u otra secuencia de operaciones; terminan siempre después de un número finito de pasos; tratan principalmente con la manipulación de símbolos, que no son necesariamente representaciones numéricas y, finalmente, son universales [Goldschlager, 1986].

Un procesador puede ser una persona, o algún otro dispositivo mecánico electrónico, que efectúe un proceso obedeciendo o ejecutando el algoritmo. Un procesador debe ser capaz de interpretar el algoritmo, lo que significa que debe:

a) Comprender cada paso;
b) Realizar la operación correspondiente.

A partir del análisis anterior resulta evidente que una computadora es un tipo especial de procesador, y por ello ha tenido un impacto tan rápido e importante en tantos ámbitos de la vida.

Cuando el procesador es una computadora, el algoritmo ha de expresarse en una forma que reciba el nombre de programa. Cada paso del algoritmo está expresado por medio de una instrucción, o proposición, en el programa. Por lo anterior, el papel del algoritmo es fundamental, debido a que un programa en sí es un algoritmo. Además, el algoritmo es independiente tanto del

lenguaje en que se expresa como de la computadora que lo ejecuta.

En aquellos problemas para los que existen algoritmos, resulta de interés saber cuántos recursos de computadora se necesitan para su ejecución. Los principales recursos de computadora que interesan son el tiempo, la memoria y el hardware. El tiempo es tan solo el período transcurrido desde el inicio hasta el final de la ejecución de un algoritmo. La memoria es la cantidad de almacenamiento que necesita el algoritmo; y el hardware es la cantidad de mecanismo físico necesario para ejecutar el algoritmo. Un problema puede resolverse empleando algoritmos muy diferentes, los cuales tal vez utilicen distintas cantidades de recursos. Resulta interesante determinar el mejor algoritmo, que es aquel que utiliza menos recursos.

La teoría de algoritmos se puede dividir en dos aspectos [Horowitz,1984]: el primero trata acerca del diseño del algoritmo, y el segundo analiza el algoritmo en sí.

El diseñar un algoritmo es una actividad intelectual difícil, la cual requiere creatividad y conocimiento profundo. Existen varias técnicas para el diseño de algoritmos que han demostrado ser útiles debido a que generalmente dan lugar a buenos algoritmos; sin embargo, el diseñar un algoritmo es un arte para el cual no es posible plantear reglas generales.

Una razón por la cual es importante hacer el análisis de los algoritmos es el encontrar nuevos métodos para resolver el mismo problema más rápidamente [Horowitz, 1984].

Antes de hacer el análisis de un algoritmo es importante determinar el tipo de computadora en la que el algoritmo se ejecutará, esto es, por que el tiempo de ejecución y la cantidad de memoria depende del tipo de máquina que se utilice.

Lo primero que se debe de hacer al analizar un algoritmo es determinar el número de operaciones que utiliza. En segundo lugar,

se debe determinar un número suficiente de ejemplos que logren que el algoritmo exhiba todas sus variaciones.

Para hacer el análisis completo del tiempo de computadora de un algoritmo se requiere de dos fases [Horowitz, 1984]: un análisis a priori y un análisis a posteriori. En el análisis a priori se ignora tanto el tipo de computadora como el lenguaje de programación, y se concentra únicamente en determinar el orden de magnitud de la frecuencia de ejecución de las instrucciones. En el análisis a posteriori se obtiene el consumo real de tiempo y de memoria en computadora mientras se ejecuta el algoritmo.

Para el análisis a priori se usará la siguiente notación: Se dice que:

$$f(n) = O(g(n))$$

si existe una constante c tal que

$$|f(n)| \leq c|g(n)|$$

donde f(n) es el tiempo de computadora, y $n \geq 0$ el número de variables [Garey, 1980].

La cantidad de un recurso utilizada depende del tamaño de los datos de entrada. Si hay n caracteres de datos de entrada, es posible expresar la cantidad de recursos utilizados como función de n; esta función O(g(n)) estudia la forma en que se incrementa el tiempo de ejecución respecto al tamaño de los problemas que resuelve.

La notación O(g(n)) se refiere a que el tiempo de computadora empleado al ejecutar un mismo algoritmo con diferentes ejemplos, va a ser siempre menor igual que una constante por g(n); esto es, O(g(n)) es una cota superior del tiempo en computadora.

2.2 Clasificación de Algoritmos

Es de interés poder comparar y clasificar los algoritmos, y para esto existen diversos criterios, en el presente trabajo se expondrá únicamente uno de ellos.

Los algoritmos se pueden clasificar en base al tiempo que tardan en encontrar una solución, siendo esta la clasificación más usual. Aquellos algoritmos cuyo O(g(n)) se comporte como c^n para alguna constante c, reciben el nombre de algoritmos exponenciales. Los algoritmos exponenciales son no factibles para todos los tamaños de datos de entrada (número de variables), excepto para los más pequeños. Para aquellos algoritmos para los cuales O(g(n)) se comporta como n^c para alguna constante c, reciben el nombre de algoritmos polinomiales. Muchos algoritmos polinomiales son factibles para casi todos los tamaños de entrada, aunque por desgracia algunos no lo son [Goldschlager, 1986].

La complejidad de un problema es tan solo el tiempo en computadora O(g(n)) del mejor algoritmo que resuelve este problema. Aunque no se conoce la complejidad exacta de muchos problemas, lo que sí se conoce en muchos casos es O(g(n)) del mejor algoritmo encontrado hasta el momento para el problema. A esto se le llama cota superior de la complejidad del problema. Si no se conoce O(g(n)) del mejor algoritmo, algunas veces es posible probar que ningún algoritmo puede resolver el problema sin utilizar al menos cierta cantidad de recursos. A esta cantidad de recursos se le denomina cota inferior de la complejidad del problema [Goldschlager, 1986].

Hay un gran número de problemas para los que la existencia de un algoritmo veloz (de tiempo polinomial) sigue siendo una pregunta abierta. Aún no se han descubierto algoritmos rápidos para tales problemas, pero tampoco ha habido alguien capaz de probar que no existe tal algoritmo. Existen algunos algoritmos de tiempo exponencial que han sido muy útiles en la práctica. Esto se

debe a que la mayoría de los problemas pueden necesitar un tiempo menor que O(g(n)). Esta situación aparece en varios algoritmos conocidos como son el método simplex de G. Dantzig y el "branch and bound" de A. H. Land y A. G. Doig [Garey, 1980].

La mayor parte de los algoritmos creados para resolver problemas de optimización son de naturaleza iterativa. En ellos se busca un vector que resuelva el problema de programación, para lo cual es necesario seleccionar un vector inicial x_0, que después del uso del algoritmo, genera un vector mejorado x_1. Este proceso se repite para obtener una secuencia de puntos cada vez mejores $x_0, x_1, \ldots, x_j, \ldots$ hasta encontrar un punto solución.

Capitulo 3. El Algoritmo del Método Simplex

3.1 Generalidades

3.2 Desarrollo

3.2.1 Variables de Holgura y Artificiales

3.2.2 Notación y Definiciones

3.2.3 Mejoramiento de la Solución Básica Factible

3.3 Degeneración

3.1 Generalidades

El algoritmo más conocido y utilizado para resolver los problemas de programación lineal (PPL) es el llamado método simplex. Este es fundamentalmente un proceso iterativo para resolver los PPL en un número finito de pasos. Fué creado por G. B. Dantzig en 1947 y actualmente tiene mucha aceptación debido a su habilidad para modelar importantes problemas de decisión y su capacidad para producir soluciones en un tiempo razonable.

Para un PPL existe un espacio de soluciones factibles (formado por la intersección de las restricciones del PPL), que forma un conjunto convexo, es decir, que cualquier línea que une dos puntos de esta región estará también incluida en la región [Ignizio, 1982]. La frontera de esta región está delimitada por líneas o planos, y a la intersección de estas líneas o planos se los conoce como esquinas o puntos extremos.

La función objetivo (ver capítulo 1) toma distintos valores dentro del espacio solución, y si el valor mínimo es finito (es decir, solución óptima), este se encontrará en un punto extremo dentro del espacio de soluciones factibles; a este punto se le llamará óptimo. En el caso de que la función objetivo se pueda decrementar arbitrariamente, no habrá esquinas óptimas, a lo que se conoce como solución no acotada [Hadley, 1963].

Si el punto óptimo no es único, es decir, que existen otros puntos cuyo valor en la función objetivo es también solución óptima, se tiene lo que se conoce como solución óptima alternativa.

Si para un PPL existe una solución óptima, entonces al menos algún punto extremo será óptimo, y el número de puntos extremos de la región convexa será finito [Hadley,1963]. Lo anterior garantiza que el método simplex llega eventualmente a la solución óptima.

El método simplex [Hadley, 1963] es un procedimiento para moverse desde un punto extremo dado hasta un punto extremo óptimo. En cada iteración, solo es posible moverse hacia lo que intuitivamente son puntos extremos adyacentes. De todos los posibles puntos adyacentes, se escoge aquel que dá el mayor decremento a la función objetivo. En cada punto extremo, el método simplex informa si este punto es óptimo, y si no lo es, determina el siguiente punto extremo. En caso de que no exista un punto extremo adyacente y la función objetivo pueda ser decrementada arbitrariamente el método simplex informa que existe solución no acotada.

Sea el problema general de la programación lineal:

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

Una solución factible es un vector x que satisface todas las restricciones y el conjunto de estos vectores forma el espacio factible.

La búsqueda de la solución óptima al PPL por medio del algoritmo simplex se condensa a continuación [Ignizio, 1982]:

1. Iniciar la búsqueda en un punto extremo del espacio factible, es decir, en una solución básica factible.
2. Determinar si el moverse hacia un punto extremo adyacente provoca una mejoría en la función objetivo; si esto no es así, la solución actual es óptima; en caso de que sí haya una mejoría se procede al siguiente paso.
3. Seleccionar el punto extremo adyacente que ofrezca el mayor decremento a la función objetivo.
4. Continuar con los pasos 2 y 3 hasta encontrar la solución óptima o hasta que se pueda demostrar que el problema es no acotado ó no factible.

3.2 Desarrollo

3.2.1 Variables de Holgura y Artificiales

Para poder utilizar el método simplex es necesario partir de una solución básica factible inicial. En muchos casos, es difícil conseguir tal solución y se requiere de algún trabajo previo antes de utilizar el método simplex. En primer lugar se transforman las restricciones de manera que el vector de requerimientos quede con signo mayor o igual a cero. Después, debido a que es más sencillo trabajar con igualdades que con desigualdades, estas últimas se convierten en igualdades, introduciendo variables x_{n+i} conocidas como variables de holgura. x_{n+i} representa a la variable de holgura de la i-ésima restricción, y su signo depende del signo de la desigualdad. Si la desigualdad es de tipo menor o menor igual, la variable de holgura será positiva, y en caso contrario será negativa. Estas variables tienen, en la mayoría de los casos, un costo asociado de cero en la función objetivo; al introducirlas en el conjunto original de restricciones, este se transforma en un conjunto de ecuaciones lineales simultáneas:

$$\sum a_{ij} x_j + x_{n+i} = b_i$$

El problema resultante tiene las mismas soluciones óptimas que el problema original, quedando el problema en su forma "standard" [Ignizio, 1982]:

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

Debido a que el método simplex comienza con una solución básica factible, en ocasiones es necesario incluir otras variables (llamadas variables artificiales) que completen una solución básica factible inicial para un conjunto modificado de

restricciones [Bazaara, 1981]. Como las variables artificiales se introducen únicamente por conveniencia matemática para dar un punto de partida, es necesario eliminarlas lo más pronto posible. Para esto existen dos procedimientos: el método de las dos fases y el método de la M grande de Charnes [Hadley,1963]. Estos métodos proporcionan ya sea un punto extremo del problema original o bien, indican que no existe espacio de soluciónes para el problema original.

3.2.2 Notación y Definiciones

Para el mejor entendimiento del método simplex, se introduce la siguiente notación [Ignizio, 1982]: x es el vector columna de orden nx1 que representa a todas las variables del modelo convertido (el modelo con todas las variables artificiales y de holgura apropiadas). Así, el modelo de programación lineal se escribe como:

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

Donde c^T = es el vector de los coeficientes de la función objetivo de dimensión 1xn.

A = es la matriz de los coeficientes de las restricciones convertidas de dimensión mxn y rango m.

b = es el vector de requerimientos de dimensión mx1.

Se denotará como A_j a la j-ésima columna de A. La matriz básica formada por m columnas linealmente independientes se denota como B, siendo esta una matriz no singular mxm. Las columnas de B se denotan como $b_1, b_2, \ldots, b_m$.

La matriz B determina una solución básica para Ax = b . Sin embargo, esta solución básica puede ser factible (si todas las componentes x_j de x son mayores o iguales a cero) o no factible (si una o más x_j <0). La solución básica determinada por B es:

$$x_B = B^{-1} b = \begin{matrix} x_{B1} \\ x_{B2} \\ \\ x_{Bm} \end{matrix}$$

Esta solución básica x_B se asocia también con un vector fila denotado c_B, el cual contiene los coeficientes de las variables básicas en la función objetivo.

Dada una solución básica factible, el valor de z se puede obtener mediante:

$$z = c_B x_B$$

Donde $c_B = (c_{B1}, c_{B2}, \ldots, c_{Bm})$.

Cualquier a_j de la matriz A se puede escribir como combinación lineal de las columnas de B, esto es,

$$\begin{aligned} a_j &= y_{1j} b_1 + \ldots + y_{mj} b_m \\ &= \Sigma\, y_{ij} b_i \\ &= B\, y_j \end{aligned}$$

Generalmente, lo anterior se escribe como:

$$y_j = B^{-1} a_j ,$$

Donde $$y_j = \begin{matrix} y_{1j} \\ y_{ij} \\ y_{mj} \end{matrix}$$

Cada y_{ij} es un escalar en el que el subíndice i se refiere a la columna asociada de B y el subíndice j se refiere al vector a_j.

La última definición de esta sección concierne al parámetro escalar z_j que está asociado a cada columna a_j de la matriz A, y esta definido como:

$$z_j = y_{1j} c_{B1} + \ldots + y_{mj} c_{Bm}$$

$$= \sum y_{ij}\, c_{Bi}$$
$$= c_B\, y_j$$

Como la matriz básica determinada por B cambia en cada iteración, el escalar z_j también.

3.2.3 Mejoramiento de la Solución Básica Factible

Dada cualquier base B, que determina un punto extremo, éste se puede mover a un punto extremo adyacente (es decir, hacia una solución básica factible mejorada) si se intercambia una de las columnas de la base (b_i) por una columna de la matriz A (a_j) que no esté en la base. Sin embargo, al hacer este intercambio, es necesario mantener siempre una solución básica factible y lograr un decremento en la función objetivo.

Esencialmente, existen dos reglas que deben de seguirse para el intercambio de variables. En primer lugar, debe determinarse el vector a_j que se introducirá a la base de manera que se mejore la solución; en segundo lugar, como alguno de los vectores b_i debe salir de la base, para que ésta siga siendo una base se selecciona aquel vector que permita que la nueva solución básica continúe siendo factible (Hadley, 1963).

Para determinar la variable de entrada se escogerá aquella cuyo valor $z_j - c_j$ sea el más positivo. El criterio $z_j - c_j$ mide el ahorro que resulta de la modificación de las variables básicas que se obtiene al incrementar la variable x_j en una unidad, menos el costo de aumentarla en una unidad. Por cada unidad x_j , el costo se verá reducido por una cantidad $z_j - c_j$, y se obtendrá una mejoría al aumentar x_j tanto como sea posible. Si $z_j - c_j < 0$, entonces al aumentar x_j , el ahorro neto es negativo y cada acción resultará en un mayor costo. Si $z_j - c_j = 0$ el incremento en x_j conducirá a una solución diferente con el mismo costo, independientemente de que x_j crezca o se mantenga a

nivel cero [Bazaara, 1981].

Para determinar la variable de salida x_{Br} que será reemplazada por a_j, se escogerá aquella que satisfaga:

$$x_{Br}/y_{rj} = \min \{ x_{Bi}/y_{ij} ; y_{ij} > 0 \}$$

Con lo que se garantiza que todas las x_{Bi} de la nueva base son positivas.

Finalmente, la optimalidad de la solución básica factible $x_B = B^{-1} b$ con $z = c_B x_B$ se alcanza cuando los escalares $z_j - c_j$ sean menores o iguales a cero para todas las columnas a_j de A.

En la figura 1 se presenta un diagrama de flujo para el método simplex.

Hasta ahora se ha descrito como resolver un PPL por medio del método simplex; sin embargo, es mucho más conveniente utilizar alguna de las formas tabulares existentes que resolver dichos problemas a mano. La forma tabular que se describe a continuación se conoce con el nombre de tableau.

El tableau es simplemente un lugar conveniente para guardar la información concerniente al PPL. Esta información consiste básicamente en:

1. $z = c_B x_B$: el valor de la función objetivo.
2. $x_B = B^{-1} b$: la solución básica factible.
3. $B = (b_1, \ldots, b_m)$: la matriz básica.
4. $y_j = B^{-1} a_j$ y $z_j - c_j$ que son los parámetros que indican ciertas condiciones como son la optimalidad, no acotamiento y mejoramiento de la solución.

Toda esta información se condensa en el tableau en cada iteración del método simplex. En la tabla 1 se presenta dicho tableau.

FIGURA 1

DIGRAMA DE FLUJO PARA EL METODO SIMPLEX

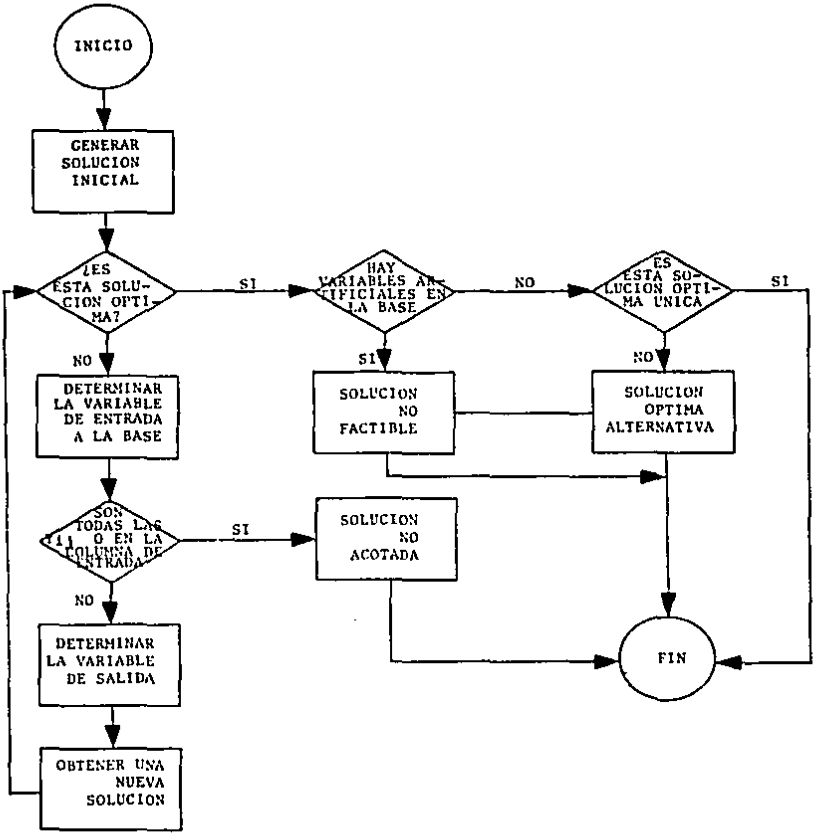


TABLEAU PARA EL METODO SIMPLEX

				C1	C2	Cj	0	0	M	M
CB	VECTORES EN LA BASE	B		a1	a2	aj	aj+1	an	q1	qs
CB1	B1	IB1		Y11	Y12	Y1j	Y1 j+1	Y1n	Y1 n+1	Y1 n+s
CB2	B2	IB2		Y21	Y22	Y2j	Y2 j+1	Y2n	Y2 n+1	Y2 n+s
CBm	Bm	IBm		Ym1	Ym2	Ymj	Ym j+1	Ymn	Ym n+1	Ym n+s
	Zj-Cj	Z=	Ym+1 0	Ym+1 1	Ym+1 2	Ym+1 j	Ym+1 j+1	Ym+1 n	Ym+1 n+1	Ym+1 n+s

ai , i=1,...,j Variables Originales

ai , i=j+1,...,n Variables de Holgura

qi , i=1,...,s Variables Artificiales

La primera columna lista los coeficientes de las variables básicas en la función objetivo. La tercera columna dá los valores de las variables básicas. Las columnas interiores son los vectores y_j asociados a cada variable x_j. El valor z de la función objetivo se encuentra del lado izquierdo del último renglón; este renglón también lista los valores $z_j - c_j$ para cada variable. La matriz básica no se observa explícitamente en el tableau pero puede obtenerse a partir de las variables básicas.

El método simplex detecta la existencia de una solución óptima alternativa cuando para algún vector x_j no incluido en la base se tiene que $z_j - c_j = 0$ y al menos un $y_{ij} > 0$. En el caso en que el problema sea no factible, el método simplex lo detecta cuando existen una o más variables artificiales en la base a nivel positivo. Por último, para el caso de que el problema tenga solución no acotada el método simplex lo detecta cuando no se cumplen las condiciones de optimalidad y existe alguna columna para la cual $y_{ij} \leq 0$, para toda j [Hadley, 1963].

3.3 Degeneración

El número de soluciones básicas de un PPL es finito, y el método simplex es un procedimiento iterativo que permite encontrar una nueva solución básica en cada iteración que al menos será igual a la anterior. Si cada solución básica fuera mejor a la anterior se podría garantizar que cada base aparecerá cuanto más una vez en los cálculos y que la base óptima se obtendrá después de un número finito de iteraciones.

La degeneración se dá cuando una variable no básica reemplace a una variable básica con valor cero o cuando la variable determinada por el criterio de salida no es única [Ignizio, 1982].

En la práctica la degeneración no es problema; el método simplex determina el vector a entrar y salir de la base sin

ninguna dificultad resolviendo así los problemas reales, aunque un gran número de variables básicas desaparezcan. Algunas veces cuando existe degeneración, la función objetivo puede tomar el mismo valor al moverse de una solución básica factible a otra. Si esto ocurre durante varias iteraciones sucesivas no se puede garantizar que la base no se repite, es decir, se llega a una situación en donde se cicla para siempre, repitiéndose siempre la misma secuencia de bases sin llegar a la solución óptima [Hadley, 1963].

Para evitar el ciclaje, un método sencillo es escoger aleatoriamente la variable a salir de la base de entre todas las posibles. Otra manera es alterar los valores numéricos del problema para que la degeneración no sea posible; lo anterior garantiza una variación de la función objetivo y hace imposible una repetición de bases. Los dos procedimientos más conocidos para la resolución del problema degenerado son: el método de perturbación de Charnes y el de los vectores ordenados lexicográficamente [Hadley, 1963].

Los casos en los que se ha llegado a un ciclo son ejemplos construídos artificialmente, y únicamente de importancia académica; no se conoce ningún problema práctico que haya ciclado, es por esto que la degeneración nunca ha impedido llegar a la solución óptima por medio del método simplex [Hadley, 1963].

Capítulo 4. El Algoritmo de Karmarkar

4.1 Generalidades

4.2 Conceptos Fundamentales

4.3 Descripción

4.4 Desarrollo

4.5 Transformación de un Problema de Programación Lineal a la Forma Canónica del Algoritmo de Karmarkar

4.1 Generalidades

Este algoritmo fué diseñado por un joven indú en el verano de 1984; este joven de nombre Narendra Karmarkar trabaja en los laboratorios Bell de AT&T en Nueva Jersey, donde originalmente dió a conocer su algoritmo. Según la revista Fortune [Gannes,1986], Karmarkar está reconocido como uno de los 10 investigadores que más han aportado a la ciencia.

El algoritmo de Karmarkar resuelve los problemas de programación lineal en tiempo polinomial al igual que el algoritmo elipsoidal de Khachian [Karmarkar,1984a]; lo cual, parece ser ofrece la ventaja de reducir el tiempo que utiliza la computadora para llegar a la solución.

La comunidad científica está experimentando con el algoritmo de Karmarkar para conocer sus alcances reales y las ventajas o desventajas que ofrece sobre otros métodos de programación lineal.

El algoritmo de Karmarkar es un proceso iterativo que dá una aproximación a la solución del problema de programación lineal, mas no la solución exacta. Este algoritmo, a diferencia del método simplex, recorre el interior del politopo formado por las restricciones. Para realizar esto, se elige un punto interior del espacio de soluciones, y se transforma dicho espacio de manera que el punto se localize en el centro del mismo; una vez realizado lo anterior, se buscará aquella dirección que favorezca la función objetivo. Posteriormente se elige un segundo punto sobre esta misma dirección, y se vuelve a transformar el espacio de manera que este punto nuevamente sea el centro del politopo. Este proceso se repite hasta llegar a la mejor aproximación de la solución óptima.

Karmarkar afirma [Hooker,1986] que su algoritmo se basa en dos ideas fundamentales:

1. Si la solución actual se encuentra cerca del centro del politopo, resulta aceptable moverse en aquella dirección que reduzca al máximo la función objetivo.

2. El espacio solución se puede transformar de manera que la solución actual quede cercana al centro del politopo, sin modificar la esencia del problema.

Lo primero es evidente en la figura 1. Como x_0 está cerca del centro del politopo, la solución actual puede mejorarse de manera substancial al moverse en la dirección que favorezca a la función objetivo. Sin embargo, si se hace el mismo proceso con x_1, se llega a la frontera de la región factible antes de lograr el mínimo valor de la función objetivo.

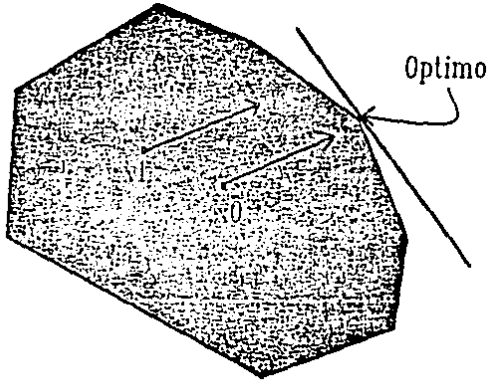
Para la segunda idea, Karmarkar notó que al observar desde un ángulo oblicuo la gráfica de un politopo, este se transforma. La proyección de la gráfica en la retina del ojo es una distorsión del problema original, un caso especial de una transformación proyectiva. En una transformación proyectiva las líneas rectas siguen siendo líneas rectas, mientras que los ángulos y las distancias sí cambian, aunque esta distorsión apenas modifica la esencia del problema.

Para que cualquier problema de programación lineal pueda ser resuelto por medio del algoritmo de Karmarkar, deberá ser transformado de manera que satisfaga las siguientes condiciones: el valor mínimo que tome la función objetivo deberá ser cero y el sistema formado por las restricciones deberá estar igualado a cero.

4.2 Conceptos Fundamentales

Al ser este algoritmo un procedimiento reciente y poco difundido, es importante dar a conocer los conceptos intuitivos para entender por qué funciona. Esto es de gran ayuda

FIGURA 1



para quienes estén interesados en conocer y entender el algoritmo, por que dá una visualización geométrica de cómo se llega eventualmente al resultado.

Los conceptos generales que se mencionarán son los siguientes:

a) Cota para la Función Objetivo
b) Optimización sobre una Esfera
c) Transformaciones Proyectivas
d) Funciones Potenciales Invariantes

a) Cota para la Función Objetivo

Sean: F el politopo definido por las restricciones $Ax=b$ y $x \geq 0$ y $a \in F$ un punto estrictamente interior de F.

Definiremos las siguientes esferas:

B_r = $\beta(a,r)$ la máxima esfera estrictamente contenida en el politopo F, con centro en a y radio r.

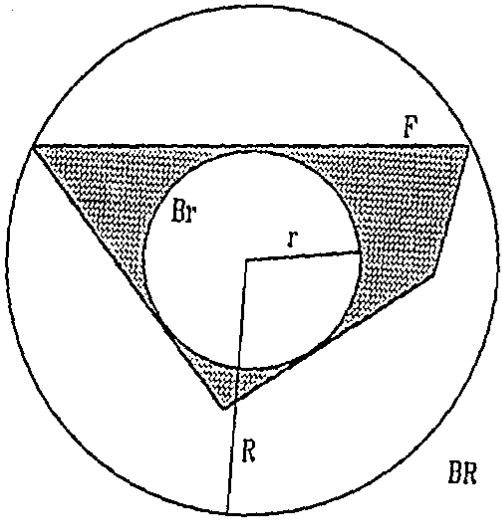
B_R = $\beta(a,R)$ la mínima esfera que contiene estrictamente al politopo F, con centro en a y radio R.

de lo que resulta: $B_r \subseteq F \subseteq B_R$ y $B_R = \tau B_r$, donde τ es un factor de proporcionalidad definido por: $1 < \tau = R/r$. Gráficamente, lo anterior se ilustra en la figura 2. Ahora, sea L una función lineal que dá el valor mínimo de la función objetivo en una región, entonces se tendrá que L_{B_r}, L_F y L_{B_R} son los valores mínimos que toma la función objetivo respectivamente para B_r, F y B_R.

Si $a_i \in B_r$, entonces la siguiente desigualdad se cumple:

$$L(a_i) - L_F \leq k(L(a) - L_F) \qquad k<1,$$

FIGURA 2



siempre y cuando A_1 se encuentre en aquella dirección que favorezca la función objetivo, es decir que el valor que tome esta función sea cada vez menor (ver figura 3).

Se sabe que el punto óptimo (solución óptima) se encuentra en un punto extremo del politopo, por lo que es necesario hacer la esfera B_r lo más grande posible y B_R lo más chica posible para así aproximarse lo más que se pueda a la solución, haciendo que el factor de proporcionalidad $\tau = R/r$ se acerque a uno (tienda a uno) para garantizar la aproximación a la solución. Entre menor sea el valor de τ, más rápida será la convergencia del algoritmo, esto es, la tasa de convergencia del algoritmo depende de τ.

Lo anterior se expresa matemáticamente de la siguiente forma:

$$0 \leq \frac{L_{B_R} - L_F}{L(a) - L_F} \leq 1 - \frac{1}{\tau} \qquad (1)$$

donde el cociente $(L_{B_R} - L_F) / (L(a) - L_F)$ indica cuánto se está aproximando el algoritmo a la solución óptima.

La expresión (1) se demostrará a continuación:
$\forall\ x \in B_R$, se cumple que existe una $y \in B_r$ tal que:

$$(x-a) = \tau\,(y-a), \qquad a \in F$$

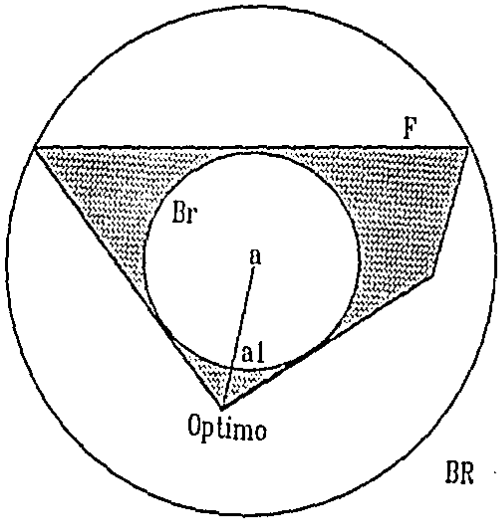
Por la linealidad de L, se tiene que:

$$L(x-a) = \tau\, L(y-a)$$
$$L(x) - L(a) = \tau\,(L(y) - L(a))$$

y sustituyendo $L(x)$ y $L(y)$ por el mínimo en B_R y B_r respectivamente resulta:

$$LB_R - L(a) = \tau\,(LB_r - L(a))$$
$$L(a) - LB_R = \tau\,(L(a) - LB_r)$$

FIGURA 3



que al despejar dá:

$$\frac{L(a) - LB_r}{L(a) - LB_R} = \frac{1}{\tau}$$

Como $B_r \supseteq F$, $L_{BR} \leq L_F$

$$\frac{L(a) - LB_r}{L(a) - L_F} \geq \frac{1}{\tau} \qquad \frac{LB_r - L(a)}{L(a) - L_F} \leq -\frac{1}{\tau}$$

Por lo anterior resulta:

$$\frac{L_{BR} - L_F}{L(a) - L_F} = \frac{L_{BR} - L(a)}{L(a) - L_F} + \frac{L(a) - L_F}{L(a) - L_F} \leq 1 - \frac{1}{\tau}$$

El factor $1 - 1/\tau$, indica, como ya se había mencionado anteriormente, cuánto se acerca al valor mínimo de la función objetivo al cambiar del punto a a un punto a'. a' es el mínimo de la esfera $\beta(a,r)$, que posteriormente pasará a ser el centro de la nueva esfera $\beta(a',r')$. Este proceso se repetirá hasta llegar a una aproximación adecuada de la solución óptima.

Debido a que cada nueva esfera debe estar estrictamente contenida dentro del politopo, el radio r va a ser cada vez más chico, razón por la cual el factor de proporcionalidad $\tau = R/r$ va a ser cada vez mayor (R se mantiene constante), con lo cual se llegará a:

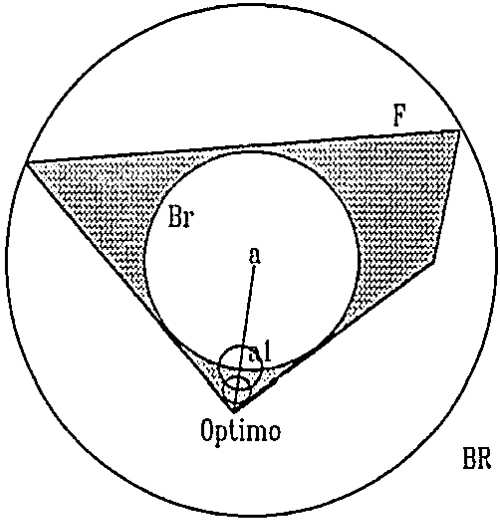
$$(1 - 1/\tau) \dashrightarrow 1 \qquad (2)$$

y lo que se pretendía era que $\tau \dashrightarrow 1$, es decir que la fórmula anterior tendiera a cero; ya que cuando esto se logra, se llega a la solución óptima. Ver figura 4.

Para evitar el problema definido en (2), la τ se acota aplicándole una transformación proyectiva al problema original.

b) Optimización sobre una Esfera

FIGURA 4



Para utilizar el algoritmo de Karmarkar, el problema general de programación lineal se reducirá al siguiente caso particular, conocido como forma canónica del problema:

$$\begin{aligned} \min\ & c^T x \\ \text{sujeto a}\ & A x = 0 \\ & e^T x = \Sigma x_i = 1 \\ & x \geq 0 \end{aligned} \qquad (5.1)$$

donde $x = (x_1, x_2, \ldots, x_n)^T$
$e^T = (1,1,\ldots,1)$
c es un vector n-dimensional
A es una matriz mxn

Sea $\Omega = \{ x \mid A x = 0 \}$ el subespacio vectorial formado por las restricciones del problema de programación lineal,
$\Delta = \{ x \mid x \geq 0, \Sigma x_i = 1 \}$ un simplex y
$\pi = \Omega \cap \Delta$ el politopo formado por el espacio solución del problema.

Si sustituimos el simplex Δ por una esfera S con centro en a, entonces, el problema a resolver queda como sigue:

$$\begin{aligned} \min\ & c^T x \\ \text{sujeto a}\ & A x = 0 \\ & x \in S \end{aligned} \qquad (5.2)$$

Como $a \in \pi$, entonces la intersección de π con la esfera S será otra esfera S' de menor dimension con centro y radio iguales a los de S. Así (5.2) puede reescribirse como :

$$\begin{aligned} \min\ & c^T x \\ \text{sujeto a}\ & x \in S' \end{aligned}$$

Para resolver este problema basta con proyectar ortogonalmente el vector c sobre Ω, para obtener así una

dirección d y moverse en el negativo de esta una distancia igual al radio de S. Ver figura 5.

c) Transformaciones Proyectivas

Se está interesado en aquellas transformaciones proyectivas del hiperplano $\Sigma = \{ x : \Sigma x_i = 1 \}$, descritas por la siguiente fórmula:

$$x' \dashrightarrow \frac{Dx}{e^T Dx}$$

donde $e^T = (1, 1, \ldots, 1)$ y
D es una matriz no singular.

Para cada punto interior del simplex Δ, existe una transformación única $T(a, a_0)$ del hiperplano Σ que fija todos los vértices del simplex, y mueve el punto $a = (a_1, a_2, \ldots, a_n)$ al centro del simplex $a_0 = (1/n)e$. Esta transformación está dada por :

$$xi' = \frac{x_i / a_i}{\Sigma x_j / a_j} \qquad i = 1,2,\ldots,n$$

Obsérvese que la matriz D es una matriz diagonal :

$$D = \begin{bmatrix} 1/a_1 & & & 0 \\ 0 & 1/a_2 & & 0 \\ 0 & & \ddots & 0 \\ 0 & & & 0 \\ 0 & & & 1/a_n \end{bmatrix}$$

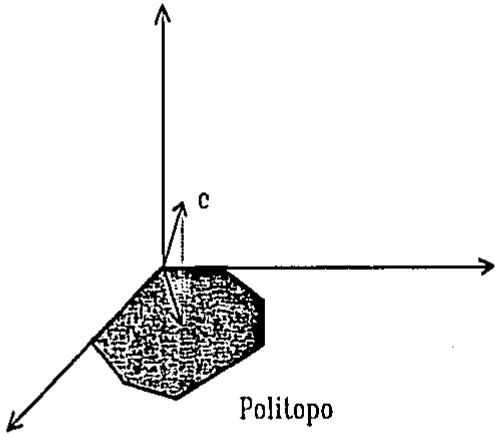
La transformación proyectiva $T(a, a_0)$ tiene las siguientes propiedades:

1) La transformación T es inyectiva y mapea el simplex a otro simplex, es decir, es una transformación biyectiva, y su inversa está dada por:

$$xi = \frac{a_i x_i'}{\Sigma a_j x_j'} \qquad i = 1,\ldots,n$$

2) Cada cara del simplex dada por $x_i = 0$ se mapea a la cara correspondiente $x_i' = 0$.

FIGURA 5



3) La imagen del punto $x = a_i$ es el centro del simplex dado por $x_i' = (1/n)$ e:

$$T(a) = \frac{a_i/a_i}{\sum a_j/a_j} e = \frac{1}{n} e = a_o$$

Ver figura 6.

4) Sea A_i la i-ésima columna de A. Entonces, al aplicar la transformación al siguiente sistema de ecuaciones:

$$\sum A_i x_i = 0$$

resulta :

$$\sum A_i \frac{a_i x_i'}{\sum a_i x_i'} = 0$$

o lo que es lo mismo:

$$\sum A_i' x_i' = 0$$

donde:

$$A_i' = A_i a_i$$

Se denotará al subespacio afín $\{ x' \mid A'x' = 0 \}$ por Ω'.

5) Si $a \in \Omega$, entonces el centro del simplex $a_o \in \Omega'$. Lo anterior se demostrará a continuación:

Sean: a_o el centro del simplex,
$\beta(a_o,r)$ la esfera más grande contenida en el simplex con centro a_o y radio r.
$\beta(a_o,R)$ la esfera más pequeña que contiene al simplex con centro a_o y radio R.

Entonces, el cociente formado por el radio de las dos esferas, $\tau = R/r$ es claramente n-1 debido a que $R = \sqrt{(n-1)/n}$ y $r = 1/\sqrt{n(n-1)}$, donde n es el número de variables.

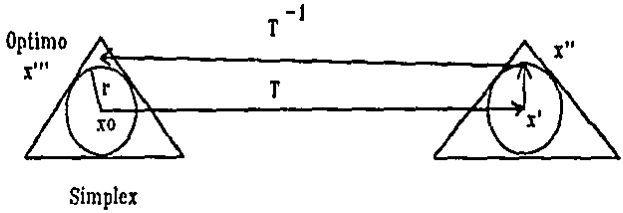
Es claro que:

$$\beta(a_o,r) \subseteq \Delta \subseteq \beta(a_o,R)$$

y de aquí que:

$$\beta(a_o,r) \cap \Omega' \subseteq \Delta \cap \Omega' \subseteq \beta(a_o,R) \cap \Omega'$$

FIGURA 6



lo que conduce a:

$$\beta^*(a_o,r) \subseteq \pi^* \subseteq \beta^*(a_o,R)$$

y además, como $R/r = n-1$, se demuestra que τ es igual a la dimensión del espacio.

d) Funciones Potenciales Invariantes

La transformación T definida en el subinciso anterior no preserva la linealidad de la función objetivo, y es por esto que Karmarkar introduce una función potencial cuya forma es invariante bajo la transformación y que se usa para medir el progreso del algoritmo y decidir cuando termina.

Como se mencionó anteriormente, el conjunto de las funciones lineales no permanece invariante bajo las transformaciones proyectivas, pero los cocientes de las funciones lineales sí, esto es, que al aplicarseles una transformación proyectiva, el resultado continúa siendo un cociente de funciones lineales. Así, Karmarkar crea su función potencial invariante g(x), la cual es asociada con cada función objetivo lineal l(x), y que depende de los logaritmos de los cocientes de las funciones lineales:

$$g(x) = \Sigma \ln \frac{l(x)}{x_j} + k \qquad k = \text{constante}$$

La función potencial invariante tiene las siguientes propiedades:

1) Cualquier reducción que se desee obtener del valor l(x) puede lograrse reduciendo el valor g(x).

2) Mediante la transformación proyectiva $T(a,a_o)$, g(x) se mapea a una función de la misma forma.

3) La optimización de g(x) puede lograrse de manera aproximada minimizando una función lineal l(x).

4.3 Descripción

Para facilitar la comprensión del algoritmo, se darán una serie de supuestos que posteriormente se eliminarán, para mostrar cómo se transforma un problema de programación lineal a la forma canónica del algoritmo de Karmarkar. Esta forma se presenta a continuación:

Forma canónica del problema:

$$\begin{aligned} \min \quad & c^T x \\ \text{sujeto a} \quad & A x = 0 \\ & \Sigma x_i = 1 \\ & x \geq 0 \end{aligned}$$

Donde c es un vector n-dimensional,
A es una matriz mxn

Sean: $\Omega = \{ x \mid A x = 0 \}$ el subespacio vectorial,
$\Delta = \{ x \mid x \geq 0, \Sigma x_i = 1 \}$ el simplex,
$\pi = \Omega \cap \Delta$, el polítopo y
$a_0 = e/n$, el centro del simplex que se toma como solución factible inicial.

Supuestos del Algoritmo

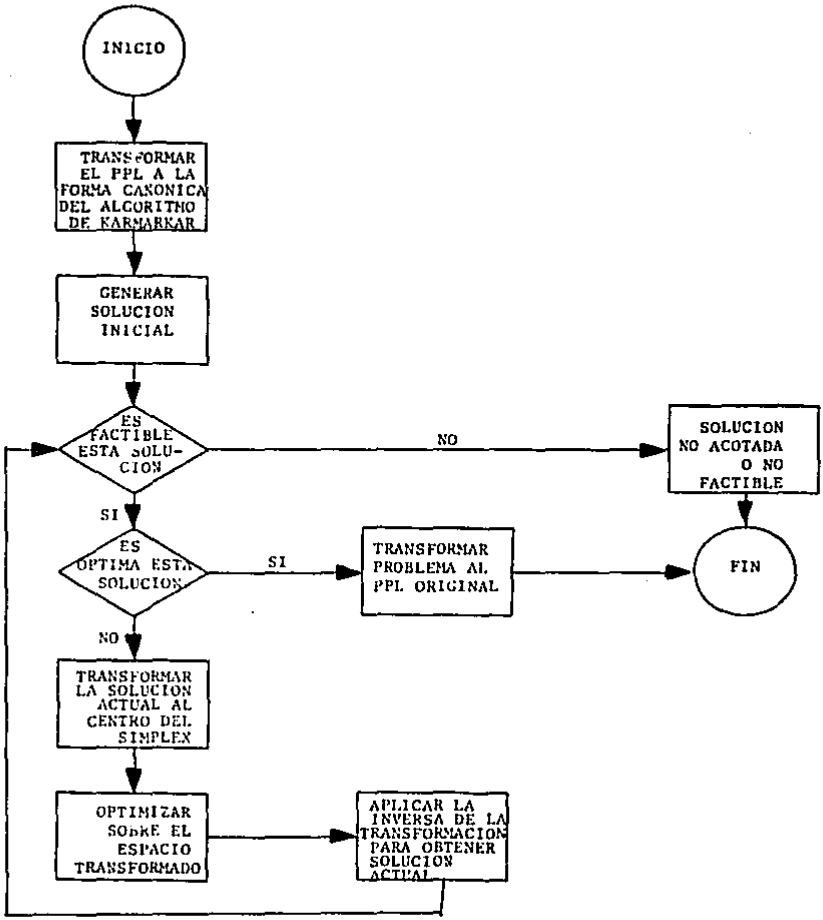
1. $c^T x \geq 0, \quad x \in \pi$
2. a_0 es una solución factible inicial.

Una vez que se aplica el algoritmo, se obtendrán los siguientes resultados:

ya sea que $x_o \in \pi$ tal que $c^T x_o = 0$ ó
una prueba de que el $\min \{ c^T x \mid x \in \pi \} > 0$.

FIGURA 7

DIAGRAMA DE FLUJO DEL ALGORITMO DE KARMARKAR



4.4 Desarrollo

Para el desarrollo, en primer lugar, se enunciarán a grosso modo los pasos que describen el algoritmo, y posteriormente serán descritos a detalle.

El algoritmo crea la secuencia de puntos

$$x^{(0)}, x^{(1)}, \ldots, x^{(k)}, x^{(k+1)}, \ldots$$

mediante los siguientes pasos:

Paso 0 Inicio: $x^{(0)} = a_0$ el centro del simplex.

Paso 1 Cálculo del siguiente punto en la secuencia

$$x^{(k+1)} = \sigma(x^{(k)})$$

Paso 2 Prueba de Factibilidad.

Paso 3 Prueba de Optimalidad.

Regreso al paso 1.

En la figura 7 se presenta el diagrama de flujo del algoritmo de Karmarkar.

Ahora se procederá a la descripción en detalle de los pasos anteriores:

Paso 1 Cálculo del Siguiente Punto de la Secuencia

Para facilitar la comprensión, se cambiará a la siguiente notación:

$$a = x^{(k)} \quad y \quad b = x^{(k+1)}$$

Este paso consiste, a su vez, de una secuencia de tres pasos:

1. Aplicar la transformación proyectiva $T(a,a_o)$ al simplex que mapea el punto a al centro a_o.
2. Optimizar (de manera aproximada) la función objetivo transformada sobre una esfera inscrita para encontrar b' (el óptimo en Ω').
3. Aplicar la transformación inversa a b' para obtener b (el óptimo en Ω).

1. Sean : $a = (a_1,\ldots, a_n)$ la solución actual y
$D = \text{diag } a_1,\ldots,a_n$ la matriz diagonal cuyo i-ésimo elemento en la diagonal es a_i,

entonces $T(a,a_o)$ está dada por:

$$x' = \frac{D^{-1} x}{e^T D^{-1} x} \qquad \text{donde } e^T = (1,1,\ldots,1)$$

y su inversa está dada por:

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

Sea f' la transformada de la función potencial invariante definida por:

$$f(x) = f'(T(x))$$

Sea Ω' la transformación del espacio afín Ω, entonces:

$$A x = 0 \qquad A Dx' = 0$$

y llamaremos Ω' al espacio nulo de AD.

Para este paso, se calcula también una matriz B definida por:

$$B = \begin{bmatrix} AD \\ e^T \end{bmatrix}$$

es decir, la matriz AD aumentada por una fila de unos.

2. Sea r el radio de la esfera más grande inscrito en π

con centro a_o. Entonces:

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

Se optimizará sobre una esfera más chica $\beta(a_o, \alpha r)$ con $0 < \alpha < 1$ por dos razones:

a) Permite que la optimización de $f'(y)$ sea aproximada por medio de la optimización de una función lineal.

b) Si lo que se desea es resolver aproximadamente las operaciones aritméticas, entonces lo anterior proporciona un margen para absorber los errores de redondeo sin salirse del simplex.

En particular, un valor para α que funciona es: $\alpha = 1/4$ y corresponde a $\delta > 1/8$. Esto último se demuestra en el teorema que se enunciará a continuación:

Teorema 1. El valor que toma la función potencial $f(x)$ está acotado de la siguiente manera:

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

(2) el valor mínimo que toma la función objetivo es estrictamente positivo.

Nota. Para la demostración del teorema anterior y todos los subsecuentes, ver [Karmarkar, 1984b].

Restringiremos al espacio afín $\Omega' = \{ y \mid ADy = 0 \}$ con la ecuación $\Sigma y_i = 1$, y se llamará Ω'' al espacio afín que resulte.

Lo que interesa es optimizar $f'(y)$ sobre

$$\beta(a_o, \alpha r) \cap \Omega''$$

y para ello, se prueba primero la existencia de un punto a que logra una reducción constante en la función potencial. Ver Teorema 2.

Teorema 2. Existe un punto $b' \in \beta(a_0, \alpha r) \cap \Omega''$ tal que $f'(b') \leq f'(a_0) - \delta$, donde δ es una constante.

En el Teorema 3 se demuestra que la minimización de $f'(x)$ puede hacerse aproximadamente minimizando la función lineal $c'^T x$.

Teorema 3. Sea b' un punto que minimiza $c'^T x$ dentro de $\beta(a_0, \alpha r) \cap \Omega''$. Entonces:

$$f'(b') \leq f'(a_0) - \delta$$

donde δ es una constante positiva que depende de α. Para $\alpha = 1/4$ se toma $\delta = 1/8$.

Finalmente se describe un algoritmo para minimizar $c'^T x$ sobre $\beta(a_0, \alpha r)$.

Algoritmo

i). Proyectar c' ortogonalmente al espacio nulo de B:

$$c_p = [I - B^T(BB^T)^{-1}B]c'$$

donde $c' = Dc$

ii). Normalizar c_p:

$$cp = \frac{c_p}{\|c_p\|}$$

iii). Acercarse una distancia αr en la dirección $-c_p$:

$$b' = a_0 - \alpha r\, c_p$$

Teorema 4. El punto b' minimiza $c'^T x$ sobre $\beta(a_0, \alpha r) \cap \Omega''$.

3. La transformación inversa está dada por:

$$b = \frac{D\, b'}{e^T D\, b'}$$

Paso 2 Prueba de Factibilidad

En cada iteración, se espera que la función potencial $f(x)$ definida por:

$$f(x) = \sum \ln \frac{c^T x}{x_i}$$

tenga una mejoría de por lo menos δ. Si esta mejoría no se logra, es decir, que

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

entonces se detiene el proceso, y se concluye que el valor mínimo de la función objetivo es estrictamente positivo.

Cuando la forma canónica del problema se obtuvo a partir de la transformación de un problema de programación lineal standard, entonces la situación mencionada anteriormente corresponde a que el problema de programación lineal original no tenga un óptimo finito, es decir, que este problema sea no factible o no acotado.

Paso 3 Prueba de Optimalidad

Esta prueba se lleva a cabo constantemente y consiste en moverse de la solución actual a un punto extremo de la esfera y probar si este punto es óptimo ó no.

Se dá un parámetro de terminación del algoritmo q (ver Teorema 5), y el objetivo es encontrar una solución factible x tal que:

$$\frac{c^T x}{c^T a_o} \leq 2^{-q}$$

La función objetivo c^Tx se asocia con la función potencial invariante:

$$f(x) = \sum \ln \frac{c^T x}{x_j}$$

y alguna de las condiciones del Teorema 1 se cumple.

Teorema 5. En $O(n(q + \log n))$ iteraciones el algoritmo

encuentra un punto factible x tal que:

$$\frac{c^T x}{c^T x_0} \leq 2^{-n}$$

4.5 Transformación de un Problema de Programación Lineal a la Forma Canónica

Para esta transformación, Karmarkar propone los siguientes pasos:

1. Sea el siguiente problema de programación lineal standard:

$$\begin{aligned} \min \quad & c^T x \\ \text{sujeto a} \quad & A x \leq b \\ & x \geq 0 \end{aligned} \qquad (5.3)$$

(el sentido de las desigualdades no afecta la transformación) se añaden las variables de holgura para convertir las restricciones en desigualdad en igualdad, resultando el siguiente sistema:

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

el signo del vector de requerimientos es irrestricto.

2. Suponer que las variables de decisión están acotadas por alguna cantidad suficientemente grande M. Esto añade una restricción más al sistema, de lo cual resulta:

$$\begin{aligned} \min \quad & c^T x \\ \text{sujeto a} \quad & A x = b \\ & e^T x \leq M \qquad \text{donde } e^T = (1,1,\ldots,1) \\ & x \geq 0 \end{aligned}$$

3. Convertir esta última desigualdad en igualdad con una variable

de holgura $s \geq 0$. Para facilitar la notación, se llamará x' al vector formado por las variables de decisión originales y la nueva variable de holgura, es decir $x' = (x,s)$, quedando el problema:

$$\begin{aligned} \min \quad & c^T x \\ \text{sujeto a} \quad & A\,x' = b \\ & e^T x' = M \\ & x \geq 0 \end{aligned}$$

4. Crear una nueva variable $x'' = (1/M)x'$, y substituirla en el problema del paso anterior:

$$\begin{aligned} \min \quad & c^T x'' \\ \text{sujeto a} \quad & A\,x'' = (1/M)b \\ & e^T x'' = (1/M)M = 1 \\ & x'' \geq 0 \end{aligned}$$

5. Como $e^T x''$ es igual a la unidad, entonces el escalar $(1/M)b$ y el vector c_m se reescriben como:

$$(1/M)b = (1/M)b\,(e^T x'') = ((1/M)b\,e^T)x''$$
$$c_m = (c_m\,e^T)x''$$

donde c_m es el valor mínimo conocido que toma la función objetivo $c^T x$. Entonces, substituyendo en el paso anterior:

$$\begin{aligned} \min \quad & c^T x'' = (c_m\,e^T)x'' \\ \text{sujeto a} \quad & A\,x'' = ((1/M)b\,e^T)x'' \\ & e^T x'' = 1 \\ & x'' \geq 0 \end{aligned}$$

Restando el lado derecho de las dos primeras ecuaciones resulta:

$$\begin{aligned} \min \quad & (c^T - c_m\,e^T)x'' = 0 \\ \text{sujeto a} \quad & (A - (1/M)b\,e^T)x'' = 0 \\ & e^T x'' = 1 \\ & x'' \geq 0 \end{aligned}$$

Finalmente, utilizando notación compacta:

$$\begin{aligned} \min \quad & c^T x = 0 \\ \text{sujeto a} \quad & A x = 0 \\ & e^T x = 1 \\ & x \geq 0 \end{aligned}$$

que es la forma canónica necesaria para utilizar el algoritmo de Karmarkar.

Esta transformación se aplica únicamente para aquellos problemas en los que se conoce el valor mínimo de la función objetivo; ya sea que este valor sea cero o una constante.

En el caso de que el mínimo del problema sea desconocido, Karmarkar propuso originalmente el uso de una "función objetivo deslizante" [Karmarkar, 1984a]. Este método se basa en acotar los valores que toma la función objetivo; pero tiene como propósito principal el demostrar la complejidad polinomial del algoritmo. Posteriormente, sugirió [Karmarkar,1984b] que el problema se resolviera por medio de un problema de factibilidad que combinara el primal con el dual, pero este proceso cuadruplica el tamaño del problema por lo que tampoco es de uso práctico. Los científicos Todd y Burrell [Todd y Burrell,1985] proponen un método que utiliza variables duales que no aumenta el tamaño del problema.

Para este trabajo, se utilizará la siguiente transformación, en caso de que el mínimo sea desconocido (Calvillo, 1987):

Cualquier PPL puede transformarse a la forma:

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

Utilizando el teorema débil de la dualidad, y agregando variables de holgura, el problema anterior se transforma en:

$$
\begin{aligned}
c^T x - b^T y &= 0 \\
A x + h &= b \\
A^T y - s &= c \\
x, y, s, h &\geq 0
\end{aligned}
$$

el cual es equivalente al siguiente problema, si el mínimo de z es cero:

$$
\begin{aligned}
&\min z \\
\text{sujeto a}\quad & A x + h + (b - A x_0 - h_0)z = b \\
& A^T y - s + (c - A^T y_0 + s_0)z = c \\
& c^T x - b^T y + (b^T y_0 - c^T x_0) = 0 \\
& x, y, s, h \geq 0
\end{aligned}
$$

que ya es de la forma (5.3), entonces solamente es necesario aplicar la transformación definida en 4.5 para llegar a la forma canónica del algoritmo de Karmarkar.

Capítulo 5. Evaluación de los Algoritmos Simplex y Karmarkar

5.1 Comparación Teórica y Práctica

5.2 Ejemplo

5.3 Programas

5.1 Comparaciones Teóricas y Prácticas

En este capítulo se harán las comparaciones entre el algoritmo del método simplex (MS) y el algoritmo de Karmarkar (AK). A continuación se presenta un cuadro sinóptico de ambos algoritmos:

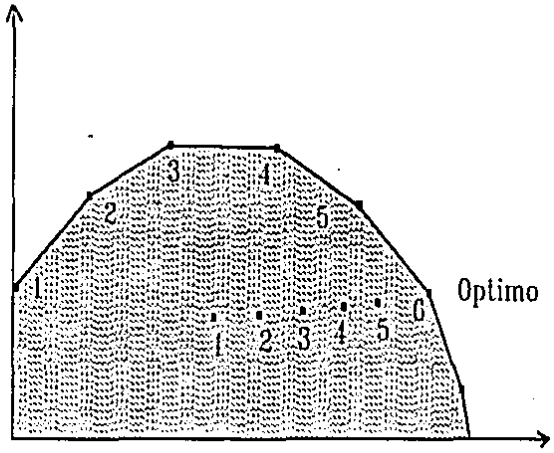
Nombre:	Método Simplex
Fecha de Nacimiento:	1947
Lugar de Nacimiento:	Corporación RAND
Padre:	George B. Dantzig
Modus Operandi:	Ir de un punto extremo hacia un punto extremo adyacente de la región factible, mejorando siempre el valor que toma la función objetivo.

Nombre:	Algoritmo de Karmarkar
Fecha de Nacimiento:	1984
Lugar de Nacimiento:	AT&T Bell Lab.
Padre:	Narendra K. Karmarkar
Modus Operandi:	Moverse a través del politopo en aquella dirección que favorezca a la proyección ortogonal de la función objetivo, transformando siempre el punto actual al centro del politopo.

Los algoritmos Simplex y Karmarkar son radicalmente distintos. El primero resuelve un problema de programación lineal examinando los puntos extremos de la frontera de la región factible; el segundo, es un método interior, esto es se mueve dentro del politopo hasta llegar a la mejor aproximación del óptimo en la frontera.

La figura 1 ilustra cómo se aproximan los dos métodos a la solución óptima. En este problema pequeño, el AK requiere no menos iteraciones (círculos) que el MS (puntos). Pero, para un problema

FIGURA 1



grande (8500 variables o más) el AK requerirá únicamente una fracción de las iteraciones utilizadas por el MS [Hooker,1986].

A medida que aumenta el número de vértices del politopo, el AK será más eficiente, esto es debido a que la superficie del mismo tiende a "suavizarse". Mientras que el MS por ser de tiempo exponencial resulta menos eficiente (más lento) cuanto mayor sea el número de vértices que tiene que recorrer para encontrar la solución.

Karmarkar, [Karmarkar, 1984a] demuestra que su algoritmo converge en tiempo polinomial, esto es, que el tiempo de ejecución es una función polinomial que depende del número de variables (n) y de bits (L) necesarios para cargar la matriz del problema. El resultado al cual llegó es:

$$O(n^{3.5} L^2) \quad \text{de tiempo de ejecución}$$

Dado que el AK es un método interior, requiere un tiempo de ejecución de $O(n^{3.5} L)$, que esencialmente es el tiempo que se necesita para resolver el problema de mínimos cuadrados [Hooker, 1986]:

$$\min \; c^{*T} x \quad \text{sobre} \quad \beta(a_0, \alpha r) \qquad (1)$$

En adición a lo anterior, se debe de tomar en cuenta que el algoritmo requiere en el caso más desfavorable de nL iteraciones y es por esto que Karmarkar llega al resultado (1).

El MS en cambio, se mueve a lo largo de los puntos extremos de la región factible, que como es sabido son todas las combinaciones de un sistema de m ecuaciones con n variables:

$$C_m = \binom{n}{m} = \frac{n!}{m!(n-m)!} \qquad (2)$$

El comportamiento de (2) es exponencial. Por lo anterior se deduce que en el peor de los casos, es decir, que el algoritmo

tenga que visitar todas las esquinas de la región factible, el tiempo de ejecución del MS es exponencial.

Por la estructura del MS, en cada iteración se cambia una variable básica por una no básica. Suponiendo que la base inicial no contenga ninguna de las variables óptimas, entonces el número de iteraciones requeridas por el algoritmo será de $O(n)$, donde n es el número de variables. Además, para cada variable que se introduce a la base se requieren $O(n^2)$ operaciones.

Por todo lo anterior se concluye que en la práctica el MS se comporta como un algoritmo polinomial cuyo tiempo de ejecución es $O(n^3)$.

La transformación necesaria para convertir un problema de programación lineal a la forma canónica que requiere el AK es considerablemente más elaborada que la que requiere el MS.

5.2 Ejemplos

A continuación se procederá a resolver un problema de programación lineal utilizando primero el algoritmo del Método Simplex y posteriormente el algoritmo de Karmarkar.

Se escogió un problema sencillo para ejemplificar todos los pasos necesarios para la solución de un problema de programación lineal por medio del algoritmo de Karmarkar. Posteriormente se resuelven problemas no triviales.

Sea el siguiente problema de programación lineal en R^2:

Ejemplo 1.

$$\begin{aligned} \min\ z &= x_1 + x_2 \\ \text{sujeto a}\quad x_1 + x_2 &\leq 1/2 \\ -x_1 + x_2 &\geq -1/4 \\ x_1,\ x_2 &\geq 0 \end{aligned} \qquad (P)$$

Ver figura 2.

Algoritmo del Método Simplex

Transformando (P) a la forma "standard" del Método Simplex resulta:

$$\begin{array}{lrcl} \min z = & x_1 + x_2 + 0x_3 + 0x_4 & & \\ \text{sujeto a} & x_1 + x_2 + x_3 & = & 1/2 \\ & x_1 - x_2 + x_4 & = & 1/4 \\ & x_1, \ x_2, \ x_3, \ x_4 & \geq & 0 \end{array}$$

y acomodando el problema en el tableau queda:

	c_j		1	1	0	0
c_B	vectores	b	a_1	a_2	a_3	a_4
0	x_3	1/2	1	1	1	0
0	x_4	1/4	1	-1	0	1
	$z_j - c_j$	0	-1	-1	0	0

Como se cumplen las condiciones de optimalidad, la solución es la siguiente:

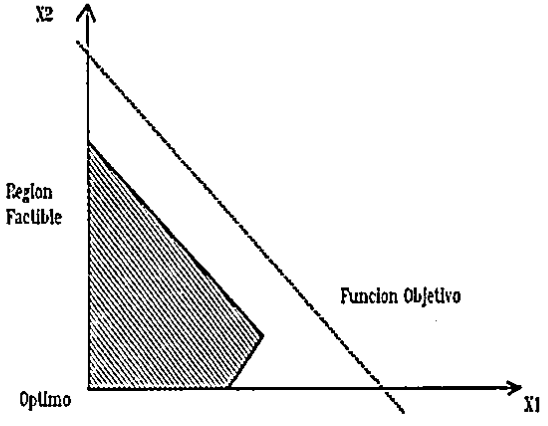
$$\begin{aligned} x_1 &= 0 \\ x_2 &= 0 \\ x_3 &= 1/2 \\ x_4 &= 1/4 \end{aligned}$$

Y el valor óptimo que toma la función objetivo es $z = 0$.

Algoritmo de Karmarkar

Para transformar (P) a la forma canónica del algoritmo de Karmarkar se le añaden variables de holgura, y se supone además que las variables están acotadas por una M suficiente grande. A partir de la figura 2, es aceptable utilizar un valor de M = 1.

FIGURA 2



Notar que el óptimo de este problema es $z = 0$. De lo anterior resulta:

$$\begin{aligned} \min\ z &= x_1 + x_2 \\ \text{sujeto a}\quad & x_1 + x_2 + x_3 = 1/2 \\ & -x_1 + x_2 - x_4 = -1/4 \\ & x_1 + x_2 + x_3 + x_4 + x_5 = 1 \end{aligned} \qquad (1)$$

Nótese:

$$A = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ -1 & 1 & 0 & -1 & 0 \end{bmatrix} \qquad b = \begin{bmatrix} 1/2 \\ -1/4 \end{bmatrix}.$$

$$c^T = (1, 1, 0, 0, 0)$$

Es claro que $\sum x_i = e^T x = 1$, por lo que (1) se puede reescribir como:

$$\begin{aligned} \min\ & c^T x \\ \text{sujeto a}\quad & (A - b\,e^T)\,x = 0 \\ & e^T x = 1 \end{aligned}$$

donde

$$A - b\,e^T = \begin{bmatrix} 1/2 & 1/2 & 1/2 & -1/2 & -1/2 \\ -3/4 & 5/4 & 1/4 & -3/4 & 1/4 \end{bmatrix} = A^o$$

Pero $A^o x = 0$ si y solo si $A' x = 0$, donde:

$$A' = \begin{bmatrix} 1 & 1 & 1 & -1 & -1 \\ -3 & 5 & 1 & -3 & 1 \end{bmatrix}$$

Entonces se resolverá el siguiente problema:

$$\begin{aligned} \min\ & c^T x \\ \text{sujeto a}\quad & A' x = 0 \\ & e^T x = 1 \end{aligned}$$

que es la forma canónica del algoritmo de Karamrkar.

Como solución factible inicial $x^{(0)}$, se elige la siguiente:

$$x^{(0)} = \begin{bmatrix} 1/8 \\ 1/8 \\ 1/4 \\ 1/4 \\ 1/4 \end{bmatrix}$$

Nótese que

$$e^T x^{(0)} = 1$$
$$A' x^{(0)} = 0 \quad \text{y}$$
$$c^T x^{(0)} = 1/4$$

$$b' = x^{(0)} - \alpha r\, c_B = \begin{bmatrix} .099 \\ .0985 \\ .29 \\ .255 \\ .255 \end{bmatrix}$$

Que es la solución en el espacio transformado.

Para calcular la solución en el espacio original, se aplica la transformación inversa a b':

$$x^{(1)} = \frac{Db'}{c^T Db'} = \frac{\begin{bmatrix} .0124 \\ .0124 \\ .0727 \\ .0639 \\ .0639 \end{bmatrix}}{.2254} = \begin{bmatrix} .0549 \\ .0546 \\ .3228 \\ .28376 \\ .28376 \end{bmatrix}$$

Con lo que $c^T x^{(1)} = .1096$.

Haciendo la prueba de factibilidad:

$$f(x) = \Sigma \ln \frac{c^T x}{x_i}$$

$$f(x^{(1)}) = \Sigma \ln \frac{c^T x^{(1)}}{x_i} = -1.58$$

$$f(x^{(0)}) = 1.386$$

$$f(x^{(1)}) \leq f(x^{(0)}) - \delta$$

Cumpliéndose las condiciones de factibilidad de la solución actual del problema.

Haciendo la prueba de optimalidad:

$$\frac{c^T x^{(1)}}{c^T a^{(0)}} = \frac{.1096}{.250} = .4384 > 2^{-q} = .00097666$$

donde $q = 10$.

Como el resultado obtenido hasta ahora es mayor que 2^{-q}, y lo que se quiere es que sea menor o igual a este valor, el proceso se repite. Los resultados que se obtienen son los siguientes:

$$B = \begin{bmatrix} AD \\ e^T \end{bmatrix} = \begin{bmatrix} .0549 & .0546 & .3228 & -.2837 & -.2837 \\ -.1649 & .2733 & .3228 & -.8513 & .2837 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

$$B^T B = \begin{bmatrix} AD^2\, AT & 0 \\ 0 & 5 \end{bmatrix} \qquad AD^2\, A^T = \begin{bmatrix} .26525 & .26525 \\ .26525 & .98925 \end{bmatrix}$$

$$(B^TB)^{-1} = \begin{bmatrix} (AD^2\,A^T)^{-1} & 0 \\ 0 & 1/5 \end{bmatrix} \qquad (AD^2\,A^T)^{-1} = \begin{bmatrix} 5.15 & -1.38 \\ -1.38 & 1.38 \end{bmatrix}$$

$$B^T(B^TB)^{-1}\,B = \begin{bmatrix} .2783 & .1461 & .2663 & -.3152 & -.2823 \\ & .2753 & .2664 & -.0273 & .3082 \\ & & .5861 & -.1378 & -.1378 \\ & & & .9443 & .0547 \\ & & & & .9287 \end{bmatrix}$$

$$I-B^T(B^TB)^{-1}\,B = \begin{bmatrix} .7248 & -.1475 & -.2660 & -.3115 & .0241 \\ & .7248 & -.2660 & .0271 & -.3085 \\ & & .4158 & .1362 & .1362 \\ & & & .0571 & .0533 \\ & & & & .0727 \end{bmatrix}$$

Con lo que:

$$c_B = \begin{bmatrix} .0316 \\ .0316 \\ -.0293 \\ -.0156 \\ -.0157 \end{bmatrix} \qquad \text{y} \qquad \|c_B\| = .05785$$

$$c_B = \begin{bmatrix} .5467 \\ .5467 \\ -.5058 \\ -.2691 \\ -.2719 \end{bmatrix}$$

Así se llega al siguiente resultado b':

$$b' = x^{(1)} - \alpha r\, c_B = \begin{bmatrix} .0244 \\ .0244 \\ .3483 \\ .2950 \\ .2952 \end{bmatrix}$$

Y aplicando la transformación inversa al resultado anterior se llega a:

$$x^{(2)} = \frac{Db'}{e^T\,Db'} = \frac{\begin{bmatrix} .0013 \\ .0013 \\ .1115 \\ .0826 \\ .0827 \end{bmatrix}}{.2794} = \begin{bmatrix} .0048 \\ .0048 \\ .3989 \\ .2957 \\ .2958 \end{bmatrix}$$

Con lo que $c^T x^{(2)} = .0096$.

Haciendo la prueba de factibilidad:

$$f(x^{(2)}) = -9,22$$

Con lo que: $f(x^{(2)}) \leq f(x^{(1)}) - \delta$

Haciendo la prueba de optimalidad:

$$\frac{c^T x^{(2)}}{c^T x^{(0)}} = \frac{.0096}{.250} = .0384 > 2^{-n} = .00976$$

Dado que el resultado es mayor que 2^{-n}, este proceso se repite hasta llegar al siguiente resultado:

$$x^{(3)} = \begin{bmatrix} -0.0003511 \\ -0.0003511 \\ 0.4731215 \\ 0.2636521 \\ 0.2639286 \end{bmatrix}$$

Con lo que $c^T x^{(3)} = -.0007022$

Lo anterior indica que la solución actual es no factible, por lo que no se hace prueba de optimalidad. Es por esto que se toma a x(2) como la mejor aproximación a la solución del problema.

A continuación se resolverá un PPL cuyo mínimo es distinto de cero. Este problema se resolvió con la ayuda del programa presentado en sección 5.3 de este trabajo.

Ejemplo 2.

$$\begin{aligned} \min \quad & x1 + x2 + 0x3 \\ \text{sujeto a} \quad & 2x1 - 3x2 + x3 = 0 \\ & x1 + x2 + x3 = 1 \end{aligned}$$

El resultado se encontró en la tercera iteración y es el siguiente:

$$x(3) = \begin{bmatrix} 0 \\ 1/4 \\ 3/4 \end{bmatrix}$$

que corresponde a $cTx(3) = 1/4$.

5.3 Programas

A continuación se presentan los programas para ambos algoritmos:

Programa del Algoritmo del Metodo Simplex

```
10  CLS:PRINT:PRINT "PROGRAMA DE PROGRAMACION LINEAL":PRINT:PRINT
20  INPUT "DE EL NUMERO DE VARIABLES INVOLUCRADAS EN EL SISTEMA:",
    NV
30  INPUT "DE EL NUMERO DE RESTRICCIONES DEL SISTEMA:",NR
40  PRINT:PRINT:DIM TAB1(NR,NV),S$(NR),S(NR),VB(NR): NV1 = NV
50  PRINT "DESEA 1.-MAXIMIZAR":PRINT "       2.-MINIMIZAR":INPUT "
```

```
    OPCION:",F:IF F=2 THEN F=-1
60  PRINT:PRINT: "DE EL COEFICIENTE DE CADA VARIABLE EN LA FUNCION
    OBJETIVO"
70  FOR H=1 TO NV:PRINT "DE EL COEFICIENTE DE X";MID$(STR$(H),2);
    :INPUT K:TAB1(0,H)=F*K
80  NEXT H:FOR H=1 TO NR:CLS:PRINT"DE LOS COEFICIENTES DE CADA VA
    RIABLE PARA LA RESTRICCION #";H:FOR J=1 TO NV:PRINT "COFICIEN
    TE DE X";MID$(STR$(J),2);:INPUT TAB1(H,J):NEXT J:INPUT "DE EL
    SIGNO DE LA RESTRICCION(<,>,= ,";S$
90  S(H)=1:IF S$="<" THEN S(H)=2 ELSE IF S$="=" THEN S(H)=1.1
100 NV1=NV1+INT(S(H)):INPUT "DE EL VALOR PARA LA RESTRICCION:",TAB
    1(H,0):NEXT J,H
110 DIM TABL(NR,NV1):FOR H=0 TO NR:FOR J=0 TO NV:TABL(H,J)=TAB1(H,
    J):NEXT J,H
120 FOR H=1 TO NR:IF S(H)=1 THEN VB(H)=J:TABL(0,J)=0:TABL(H,J)=1:J
    =J+1:GOTO 140
130 TABL(0,J)=-9E+20:TABL(H,J)=1:VB(H)=J:J=J+1:IF S(H)=2 THEN TABL
    (H,J)=-1:TABL(0,J)=0:J=J+1
140 NEXT H
147 ;
148 REM COMIENZA A ITERAR
149 ;
150 EN=0:MIN=0:FOR H=1 TO NV1:ZMC(H)=0:FOR J=1 TO NR:ZMC(H)=ZMC(H)
    +TABL(J,H)*TABL(0,VB(J)):NEXT J:ZMC(H)=ZMC(H)-TABL(0,H):IF ZMC
    (H)<MIN THEN EN=H:MIN=ZMC(H)
160 NEXT H:IF EN=0 THEN GOTO 230
170 VS=0:MIN=9E+20:FOR H=1 TO NR:IF TABL(H,EN)<=0 THEN 190
180 K=TABL(H,0)/TABL(H,EN):IF K<MIN THEN VS=H:MIN=K
190 NEXT H:IF VS=0 THEN PRINT "SOLUCION DEGENERADA":END
200 VB(VS)=EN:K=TABL(VS,EN):FOR H=0 TO NV1:TABL(VS,H)/K:NEXT H
210 FOR H=1 TO NR:IF H<>VS THEN K=-TABL(H,EN):FOR J=0 TO NV1:TABL(
    H,J)=TABL(H,J)+TABL(VS,J)*K:NEXT J
220 NEXT H:GOTO 150
230 CLS:PRINT "SOLUCI'ON DEL SISTEMA:"FOR H=1 TO NR:PRINT "VARIABL
    E X";MID$(STR$(VB(H)),2);"=";TABL(H,0):NEXT H
240 FOR H=1 TO NR:Z=Z+TABL(H,0)*TABL(0,VB(H))*F:NEXT H:
250 PRINT "VALOR DE LA FUNCION OBJETIVO:";Z
```

Programa del Algoritmo de Karmarkar

```
10  INPUT "DE EL NUMERO DE VARIABLES INVOLUCRADAS EN EL SISTEMA",
    NV
20  INPUT "DE EL NUMERO DE RESTRICCIONES DEL  SISTEMA",NR:NN=NV+1
30  NM=NR+1,NI=2*(NM)
40  PRINT:PRINT:DIM A(NR,NN),S$(NR),S(NR),BI(NR),C(NN),ID(NN,NN),
    D(NN,NN),AO(NR,NN),B(NM,NN),BTB(NM,NM),AD(NR,NN),CAD(NR,NN),C
    C(NN,NN),ADA(NR,NR),TBTB(NM,NI),BTBI(NM,NI),TBTI(NM,NN)
50  KOUNT=0
60  PRINT:PRINT "DE EL COEFICIENTE DE CADA VARIABLE EN LA FUNCION
    OBJETIVO"
70  FOR H=1 TO NN:PRINT "COEFICIENTE DE X";MID$(STR$(H),2);:INPUT
    C(H):NEXT H
80  FOR H=1 TO NR:CLS:PRINT "DE LOS COEFICIENTES DE CADA VARIABLE
    PARA LA RESTRICCION #";H
90  FOR J=1 TO NN:PRINT "COEFICIENTE DE X";MID$(STR$(J),2);:INPUT
    A(H,J):NEXT J:NEXT H
110 PRINT:FOR H=1 TO NR:PRINT "DE EL VALOR DE B(";H;INPUT BI(H):N
    EXT H
120 DC=0:ALFA=.25:D=10
130 FOR I=1 TO NN:E(I)=1:X(I)=1/NN:DC(I)=C(I)*X(I):DC=DC+DC(I):NE
    XT I
135 PRINT:PRINT "X=          DC=     "
140 FOR I=1 TO NN:PRINT X(I),DC(I):NEXT I
142 FOR I=1 TO NN:FOR J=1 TO NN:ID(I,J)=0:D(I,J)=0:NEXT J, NEXT I
144 FOR I=1 TO NN:FOR J=1 TO NN:IF (I=J) THEN ID(I,J)=1 AND D(I,J
    )=X(I):NEXT J:NEXT I
150 O=1/SQR(NV*(NV-1))
152 PRINT:KOUNT=KOUNT+1:PRINT "No. DE ITERACIONES =";KOUNT
160 IF KOUNT=1 THEN GOTO 228
170 FOR I=1 TO NN:X(I)=TBP(I):DC(I)=C(I)*X(I):NEXT I
224 FOR I=1 TO NN:FOR J=1 TO NN:IF I=J THEN D(I,J)=X(I):NEXT J:NE
    XT I
```

```
228 FOR I=1 TO NR:FOR J=1 TO NN:AO(I,J)=A(I,J)-BI(I)*E(J):NEXT J:
    NEXT I
260 PRINT:PRINT "COMIENZA SUBRUTINA DE MULTIPLICACION"
270 GOSUB 2000
271 FOR I=1 TO NN:FOR J=1 TO NN:B(I,J)=0:NEXT J:NEXT I
272 FOR J=1 TO NM:FOR J=1 TO NM:BTB(J,I)=0:NEXT I:NEXT J
273 FOR I=1 TO NR:FOR J=1 TO NN:B(I,J)=AO(I,J):NEXT J:NEXT I
277 FOR I=1 TO NR:FOR J=1 TO NR:BTB(I,J)=ADA(I,J):NEXT J:NEXT I
280 FOR I=1 TO NN:B(NM,I)=1:NEXT I
320 BTB(NM,NM)=NN
330 PRINT:PRINT "COMIENZA SUBRUTINA INVERSA"
340 GOSUB 3000
350 NCP=0
360 FOR I=1 TO NN:U=0:FOR J=1 TO NN:U=U+(ID(I,J)-CC(I,J))*X(J):NEXT J:
    CP(I)=U:NCP=NCP+CP(I)**2:CPN(I)=CP(I)/NCP:BP(I)=X(I)-ALFA*D*C
    PN(I):NEXT I
400 PRINT:PRINT "EL NUEVO VECTOR B ES:":FOR I=1 TO NN:PRINT "BP(I
    )=";BP(I):NEXT I
410 VFOP=0:EDBP=0
420 FOR I=1 TO NN:FOR J=1 TO NN:U=U+D(I,J)*BP(J):NEXT J :DBP(I)=U
430 EDBP=EDBP+DBP(I):TBP(I)=DBP(I)/EDBP:VFOP=VFOP+TBP(I)*C(I):NEX
    T I
450 PRINT:PRINT "EL VALOR DE LA FUNCION OBJETIVO ES:";VFOP
460 IF (VFOP/DC) > (1/2**Q) OR VFOP <> 0 THEN GOTO 152
470 PRINT:PRINT "EN LA ITERACION No.:";KOUNT;"SE LLEGA A QUE EL V
    ALOR OPTIMO DE LA FUNCION OBJETIVO ES:";VFOP
490 STOP
2000 REM SUBROUTINE MULTIPLICACION
2010 FOR I=1 TO NR:FOR K=1 TO NN:U=0:T=0:FOR J=1 TO NM
2020 U=U+AO(I,J)*D(J,K):T=T+AO(I,J)*D(J,K)**2:NEXT J
2030 AD(I,K)=U:CAD(I,K)=T:NEXT K:NEXT I
2040 FOR I=1 TO NR:FOR K=1 TO NR:U=0:FOR J=1 TO NN:U=U+CAD(I,J)*
    AO(K,J):NEXT J:ADA(I,K)=U:NEXT K:NEXT I
2060 RETURN
3000 REM SUBROUTINE INVERSA
3010 FOR I=1 TO NM:FOR J=1 TO NI:TBTB(I,J)=0:NEXT J:NEXT I
3015 FOR I=1 TO NM:FOR J=1 TO NM:TBTB(I,J)=BTB(I,J):NEXT J:NEXT I
3020 FOR I=1 TO NM:FOR J=NM+1 TO NI:IF (J=NM+1) THEN TBTB(I,J)=1
3030 NEXT J:NEXT I
3040 FOR K=1 TO NM:H=TBTB(K,K):FOR J=1 TO NI:TBTB(K,J)=TBTB(K,J)/H
3050 NEXT J:FOR I=1 TO NM:IF (I=H) THEN GOTO 3100:G=TBTB(I,H)
3060 FOR L=1 TO NI:TBTB(I,L)=TBTB(I,L)-G*TBTB(K,L):NEXT L
3100 NEXT I:NEXT K
3110 FOR I=1 TO NM:FOR J=NM+1 TO NI:BTBI(I,J-NM)=TBTB(I,J):NEXT J
3120 NEXT I
3122 GOSUB 3500
3130 RETURN
3500 REM SUBROUTINE MULTIPLICACION BIS
3510 FOR I=1 TO NM:FOR K=1 TO NN:U=0:FOR J=1 TO NM:U=U+B(J,K)*BTB
    I(J,I)
3520 NEXT J:TBTI(I,K)=U:NEXT K:NEXT I
3530 FOR I=1 TO NN:FOR K=1 TO NN:U=0:FOR J=1 TO NM:U=U+B(J,K)*TBT
    I(J,I)
3540 NEXT J:CC(I,K)=U:NEXT K:NEXT I
3550 RETURN
```

Apéndice. El Algoritmo Elipsoidal

Apéndice

El Algoritmo Elipsoidal

En enero de 1979, en la revista rusa Doklady, el matemático L. G. Khachian publicó un artículo llamado "Un Algoritmo Polinomial para Programación Lineal", que fué un resultado teórico importante ya que se sabe que el método simplex no es un algoritmo polinomial.

El algoritmo de Khachian, conocido como método elipsoidal, resuelve un sistema lineal de desigualdades de la forma: $Gx \leq h$, donde G es una matriz y x y h son vectores columna. Nótese que no existen restricciones de no negatividad en x ni una función objetivo explícita a ser minimizada ó maximizada [Khachian, 1978].

Para utilizar el algoritmo de Khachian es necesario convertir el problema de programación lineal en un sistema equivalente de desigualdades que tengan la misma solución. Esto se logra transformando el problema original de la siguiente forma:

Sea el siguiente problema original:

$$\begin{aligned} \max \quad & c^T y \\ \text{sujeto a} \quad & A y \leq b \\ & y \geq 0 \end{aligned} \qquad (1)$$

Por el teorema débil de la dualidad, cualquier solución al siguiente sistema de desigualdades resolverá (1), donde u denota el vector solución del dual:

$$\begin{aligned} A y &\leq b \\ - A^T u &\leq -c \\ - c^T y + b^T u &\leq 0 \end{aligned} \qquad (2)$$

$$-y \leq 0$$
$$-u \leq 0$$

Si se tienen n' variables y m' restricciones para el problema de programación lineal dado por (1), entonces habrá $n = n' + m'$ variables y $m = 2n' + 2m' + 1$ restricciones en el sistema $Gx \leq h$ dado por (2). Además, si el problema de programación lineal tiene una solución única, el espacio factible para el sistema de desigualdades dado por (2) consiste en un único punto [Ignizio, 1982].

El algoritmo de Khachian resuelve (2) comenzando en una esfera con centro en el origen $x_0 = (y,u)$ y radio r suficientemente grande para asegurar que la solución óptima x^* esté contenida en la esfera, es decir, $r \geq ||x^*||$. Posteriormente, el algoritmo procede a construir una serie de elipsoides sucesivamente más pequeñas cuyo centro converge a la solución de (2) y por tanto de (1); cada elipsoide sucesiva se construye a partir de la elipsoide anterior de tal manera que siempre contenga a la solución. La demostración de la convergencia del algoritmo muestra que el volumen de una elipsoide se puede reducir a cualquier valor deseado dentro de k iteraciones, donde k es un polinomio en términos de n [Khachian, 1978].

El procedimiento general [Ignizio, 1982] para construir una elipsoide en cada k-ésima iteración consiste de los siguientes pasos:

1. Probar cada desigualdad de (2) y seleccionar alguna restricción que no se cumpla. Los coeficientes de las variables de esta restricción se denotarán por medio de un vector columna g_k; si todas las desigualdades se satisfacen se termina el algoritmo, ya que x_k resuelve (2) y (1). De otra manera continuar al siguiente paso.

2. Cortar la elipsoide actual por la mitad pasando un plano por el centro actual x_k paralelo al hiperplano de la restricción

que no se cumple (g_i x $\leq$ h), quitando la región g_i (x - x_k) >0. Como x_k no cumple g_i x $\leq$ h, entonces toda x tal que g_i (x - x_k) > 0 tampoco cumple la restricción y por lo tanto debe descartarse esa región.

3. Construir una nueva elipsoide de menor volúmen con un nuevo centro x_{k+1} que contenga completamente a la media elipsoide del paso 2. Regresar al paso 1.

La ecuación de una elipsoide se puede escribir de la forma

$$E = \{ w \mid w = x + Q z , ||z|| \leq 1 \}$$

donde E es la elipsoide definida por (x,Q) donde x es un vector n-dimensional que define el centro de la elipsoide y Q es una matriz de transformación nxn. Esta elipsoide es la imagen de una esfera euclideana con $||z|| \leq 1$. La elipsoide se define completamente al conocer el centro x y la matriz de transformación Q.

Khachian dá el siguiente sistema de ecuaciones para actualizar x y Q (en la correspondiente elipsoide nueva) en cada iteración:

$$F_k = - Q_k g_i$$

$$x_{k+1} = x_k + (Q_k F_k) / (n+1)||F_k||$$

$$Q_{k+1} = 2^{(1/(n^2-1))} Q_k \, ORT(F_k) D$$

Donde $ORT(F_k)$ es una matriz ortogonal de nxn cuya primera columna está dada por:

$$F_k / ||F_k||$$

y D es la matriz diagonal dada por

$$D = diag\ (n/(n+1),\ n/\sqrt{n^2-1},\ \ldots,\ n/\sqrt{n^2-1}).$$

Los cálculos son directos, a excepción de cuando se quiere encontrar la matriz ortogonal cuya primera columna es $F_k / ||F_k||$.

Una matriz ortogonal es una matriz tal que su inversa es la transpuesta de sí misma; para encontrar esta matriz ortogonal se

puede utilizar el procedimiento de ortogonalización de Gram-Schmidt.

Como el algoritmo de Khachian requiere de un valor inicial para x_0 y Q_0, Khachian sugiere que se use $x_0 = 0$ y $Q_0 = rI$, donde I es una matriz identidad $n \times n$ y $r = 2^L$ es el radio de la esfera inicial que engloba a la solución.

A continuación se presentará un ejemplo del método descrito anteriormente:

Considérese el siguiente sistema de desigualdades:

$$\begin{aligned} -x_1 &\leq -10 \\ x_1 &\leq 12 \\ -x_2 &\leq 1 \\ x_2 &\leq 1 \end{aligned}$$

En seguida se harán los cálculos para la primera iteración. Por conveniencia al graficar, se utilizará un radio $r = 16$ que es suficientemente grande como para contener a todos los puntos factibles.

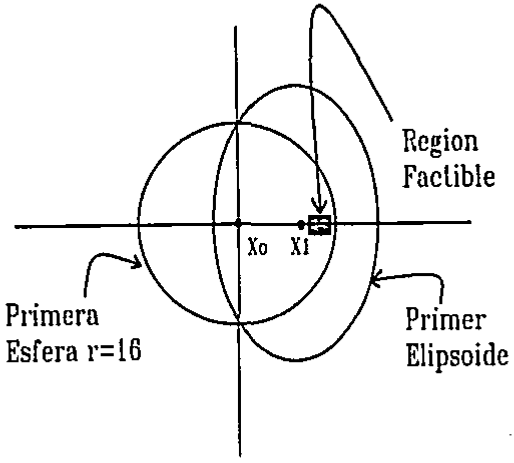
$$x_{(0)} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \qquad Q_{(0)} = \begin{bmatrix} 16 & 0 \\ 0 & 16 \end{bmatrix} \qquad g_1 = \begin{bmatrix} -1 \\ 0 \end{bmatrix} \qquad D = \begin{bmatrix} 2/3 & 0 \\ 0 & 2/\sqrt{3} \end{bmatrix}$$

$$x_{(1)} = \begin{bmatrix} 16/3 \\ 0 \end{bmatrix} \qquad Q_{(1)} = \begin{bmatrix} 10.91 & 0 \\ 0 & 18.90 \end{bmatrix}$$

$x_{(1)}$ y $Q_{(1)}$ definen una elipsoide con centro (16/3,0) y ejes a=10.91 y b = 18.90. En la figura 1 se muestran la esfera inicial, la región factible y el primer elipsoide generado por el ejemplo. Nótese que para la primera iteración se eligió la primera restricción, es decir $-x_1 \leq -10$.

Se hace pasar un hiperplano a través del centro actual (0,0) paralelo al plano de la primera restricción, para quitar así la

FIGURA 1



mitad izquierda de la esfera inicial. La nueva elipsoide es de menor volúmen y contiene completamente el lado derecho de la esfera inicial, y por tanto a la región factible. Las siguientes iteraciones del algoritmo darán como resultado elipsoides cada vez más pequeñas, este proceso se continúa hasta que los centros de las elipsoides converjan a la región factible. Ver figura 1.

CONCLUSIONES

Conclusiones

Antes de entrar de lleno a las conclusiones, quisiéramos hacer énfasis en que nuestro propósito no ha sido el hacer una apología del método de Karmarkar. Nuestra intención es el hacer accesible un nuevo desarrollo que ha abierto una amplia posibilidad para enfrentar el tipo de problemas que se mencionan en este trabajo. La efectividad del método de Karmarkar se establecerá una vez que se haya estudiado y quizás mejorado en el futuro.

A continuación exponemos las conclusiones que consideramos más sobresalientes y que pudieran desprenderse de la exposición anterior.

1. Aún cuando Karmarkar afirmó haber obtenido una rapidez de solución 50 veces mayor que el método simplex, a la fecha no ha sido posible reproducir este resultado, a pesar de que hay numerosas personas dedicadas a ello. Sin embargo, el mismo hecho de que se hayan realizado tantos esfuerzos por lograrlo indica la importancia que se le ha dado a este nuevo método.

2. La importancia de este método radica tanto en el orden teórico como en el práctico.

En el contexto teórico, este método expone posibilidades no exploradas anteriormente ya que es un método radicalmente distinto del simplex. Esto dá pie a que se realicen investigaciones que pudieran conducir, como siempre ha sucedido, ya sea a mejoras en el método o inclusive a nuevos descubrimientos importantes basados en él. Además, proporciona otro método de orden polinomial, lo cual constituye una ventaja en sí mismo.

Desde el punto de vista práctico, ofrece la posible ventaja de la rapidez, si es que llegara a realizarse consistentemente.

3. Dado que la solución de problemas de mínimos cuadrados ha alcanzado un nivel muy avanzado, una posible mejora, sobre la cual ya se está trabajando, podría consistir en la aplicación de alguno de dichos métodos en sustitución del método tradicional propuesto originalmente por Karmarkar.

4. Aún cuando teóricamente el método simplex tiene complejidad exponencial, en la mayoría de los problemas prácticos se comporta más bien como un método polinomial, es decir, ha resultado ser muy eficiente. Esto hace pensar que no habrá una sustitución inmediata del método simplex por el de Karmarkar, ya que la inversión existente en sistemas de computadora para la solución de PPL basados en el método simplex es de tal magnitud, que se requeriría de una demstración palapable de la superioridad del método de Karmarkar sobre el método simplex. Tal demostración no existe hasta el momento.

GLOSARIO

Glosario

[Hadley,1969.Hadley,1963.Simonnard,1966]

Casco Convexo (Convex Hull). Se llama casco convexo o envolvente convexa a la cerradura convexa de un conjunto finito de puntos $x_1, x_2, \ldots, x_n$.

Cerradura Convexa (Convex Closure). Se llama cerradura convexa de un conjunto E a la intersección de todos los conjuntos convexos que contienen a E.

Combinación Lineal Convexa. Se llama combinación lineal convexa de p puntos x_i, a cada x definido por:

$$x = \Sigma \mu_i x_i \ , \quad \mu_i \geq 0 \ , \quad \Sigma \mu_i = 1.$$

Conjunto Afinmente Independiente. Un conjunto finito $x_0, \ldots, x_n$ es afinmente independiente si el conjunto $x_1 - x_0, \ldots, x_n - x_0$ es linealmente independiente. La indexación de los elementos es arbitraria.

Conjunto Convexo. Un conjunto convexo C, es un conjunto de puntos tales que:

$$x_1, x_2 \in C \qquad \mu x_1 + (1 - \mu)x_2 \in C$$
$$0 \leq \mu \leq 1$$

es decir, un conjunto convexo es aquel que contiene al segmento de recta que une a dos puntos cualesquiera pertenecientes al conjunto.

Espacio Nulo. Se llama espacio nulo al conjunto de vectores x de R^n que tienen como imágen el origen de R^m. Este es el conjunto de soluciones de $Ax = 0$.

Hiperplano. Se llama hiperplano al conjunto de la forma:

$$H = \{ x \in R^n \mid A_i x = b_i \}$$

donde A_i representa el renglón i-ésimo de una matriz A mxn no nula y b_i es el escalar correspondiente al i-ésimo renglón.

Un hiperplano divide al espacio en dos regiones llamadas

semiespacios.

Poliedro. Se llama poliedro P al conjunto definido por la intersección de un número finito de semiespacios:

$$P = \{ x \mid Ax \leq b \}$$

donde A es una matriz mxn no nula cuyo i-ésimo renglón es A_i, y b es un vector columna de m componenetes con al menos una componente distinta de cero.

Politopo. Se llama politopo o poliedro convexo K, al casco convexo de un número finito de puntos $x_1,\ldots,x_p$:

$$K = \{ x \mid x = \Sigma \mu_i x_i \ , \ \Sigma \mu_i = 1 \ , \ \mu_i \geq 0 \}$$

Punto Extremo. Un punto x es un punto extremo de un conjunto convexo si y solo si no existen puntos x_1, x_2 distintos en el conjunto tales que:

$$x = \mu x_2 + (1 - \mu)x_1 \qquad 0 < \mu < 1$$

entonces se tiene que $x = x_1 = x_2$, es decir, x no se puede representar como una combinación convexa estricta de dos puntos distintos del conjunto.

Semiespacio. Se llama semiespacio cerrado a la colección de puntos de la forma:

$$\{ x \mid A_i x \geq b_i \} \quad ó$$
$$\{ x \mid A_i x \leq b_i \}$$

Se llama semiespacio abierto a la colección de puntos de la forma:

$$\{ x \mid A_i x > b_i \} \quad ó$$
$$\{ x \mid A_i x < b_i \}$$

donde A_i representa el renglón i-ésimo de una matriz A no nula y b_i es el escalar correspondiente al renglón i-ésimo.

Simplex. Se llama n - simplex (n - simplejo) al casco convexo de un conjunto afinmente independiente con vértices $x_0,\ldots,x_n$.

Solución no Acotada. Se dice que hay solución no acotada cuando el valor que toma la función objetivo puede incrementarse o decrementarse sin llegar a un óptimo finito.

Solución no Factible. Se dice que hay solución no factible cuando las restricciones son inconsistenetes o cuando no existe un punto que satisfaga las restricciones de no negatividad.

Subespacio Afín. El conjunto de las soluciones de $Ax = b$ se llama un subespacio afín de $\mathbb{R}^n$. Tiene la misma dimensión que la del subespacio generado por las soluciones de $Ax = 0$; la única diferencia consiste en que el subespacio afín se ha trasladado fuera del origen.

BIBLIOGRAFIA

Bibliografía

Libros

Bazaara, M. Jarvis, J. Programación Lineal y Flujo en Redes. Trad. O. Hernández. Ed. Limusa. México. 1981.

Calvillo G. Métodos de la Programación Lineal. Centro de Investigación y Estudios Avanzados del IPN. México. 1987.

Dantzig, G. Linear Programming and Extensions. Ed. Princeton University Press. New Jersey. 1963.

Dorfman, R. Samuelson, P. Solow, R. Linear Programming and Economic Analysis. Ed. Mc. Graw-Hill. U.S.A. 1958.

Garey, M. Jhonson, D. Computers and Intractability a Guide to the Theory of NP-Completeness. Ed. W.H. Freeman and Co. San Francisco. 1979

Goldschlager, L. Lister, A. Introducción Moderna a la Ciencia de la Computación con un Enfoque Algoritmico. Trad. L. Fournier. Ed. Prentice-Hall Hispanoamericana. México. 1986.

Hadley, G. Algebra Lineal Linear Algebra. Trad. J. Arias. Ed. Fondo Educativo Interamericano. Colombia. 1969.

Hadley, G. Linear Programming. Ed. Addison-Wesley Pub. Co. U.S.A. 1963.

Horowitz, E. Sahni, S. Fundamentals of Computer Algorithms. Ed.

Computer Science Press. U.S.A. 1984.

Ignizio, J. — Linear Programming in Single & Multiple Objective Systems. Ed. Prentice-Hall International. New Jersey. 1982.

Luenberger, D. — Linear and Non Linear Programming. Ed. Addison-Wesley Pub. Co. U.S.A. 1984.

Simonnard, M. — Linear Programing. Trad. W. Jewell. Ed. Prentice-Hall International. New Jersey. 1962.

Revistas

Angier Natalie. — Time. 'Folding the Perfect Corner'. Dic. 84. 1984.

Emmett Arielle. — Spectrum. 'Karmarkar's Algorithm: A Threat to the Simplex?'. Dic 3. 54-55. 1985.

Gannes Stuart. — Fortune. ' People at the Frontiers of Science'. Vol. 21. Oct 13. 39-49. 1986.

Gill Philip et al. — Mathematical Programming. ' On Projected Newton Barrier Methods for Linear Programming and an Equivalence to Karmarkar's Projective Method'. Vol. 36. 183-209. 1986.

Hooker J. — Interfaces. 'Karmarkar's Linear Programming Algorithm'. Vol 16. Jul-Ago 4. 75-90. 1986.

Karmarkar Narendra. — Combinatorica. 'A New Polynomial - Time Algorithm for Linear Programming'. Tomo 4. Vol. 4. 373-395. 1984 b.

Kolata Gina. Science. 'A Fast Way to Solve Hard Problems'. Sept 21. 1379-1380. 1984.

Kolata Gina. Science. ' Mathematician's Amazed by Russian's Discovery'. Vol. 206. Nov 2. 545-546. 1979.

Rocket Andrew. Byte. 'Karmarkar's algorithm'. Sep. 1987.

Todd Michael. Computing Reviews. 'A New Polynomial Time Algorithm for Linear Programming'. Feb. 95-96. 1986.

Articulos

Colmenarns, O. 'Karmarkar's Linear Programming Algorithm Better or Worse than the Classical Method'. Graduate School of Management, UCLA. Los Angeles, Ca. 90024. Abr 3. 1985

Dantzig, G. 'Reminiscences about the Origins of Linear Programming' . Systems Optimization Laboratory. Department of Operations Research. Stanford University. Stanford, Ca. 94035. 1981.

Karmarkar, N. 'A New Polynomial-Time Algorithm for Linear programming'. AT&T Bell Laboratories Murray Hill, New Jersey 07974. 1984 a.

Khachian, L. 'Polynomial Algorithm for Linear Programmming'. Trad. M. Viach. Computing Center Academy of Sciences. USSR, Moscow. 1978.

Todd, M. Burrell, B. 'An Extension of Karmarkar's Algorithm for

L. P. using Dual Variables'. School of O. R. and Industrial Engineering. Cornell University. Ithaca, New York 14850. Technical Report no. 648. January 1985.