Implementación de un Algoritmo de Puntos Interiores para Programación Lineal

Marielba Rojas
Depto. de Computación
y Tecnología de la Información

Marianela Lentini
Depto. de Matemáticas
Puras y Aplicadas

Universidad Simón Bolívar
Apartado 89000, Caracas 1080-A, Venezuela
sun!emsca!usb!marielba@sun.com
sun!emsca!usb!mlentini@sun.com

Resumen

Se presentarán consideraciones numéricas sobre la implementación de un Algoritmo Primal de Reducción de Potencial para programación lineal con el procedimiento para actualización de cotas inferiores original de Clovis C. Gonzaga.

Las consideraciones incluyen la resolución de sistemas de ecuaciones esparcidas, simétricas y positivas definidas de gran escala, mediante el método de Gradiente Conjugado Precondicionado, y un enfoque novedoso en el cálculo de un precondicionador para este método.

1 Introducción

Consideremos el problema de programación lineal en su forma estándar

$$P: \quad min\ c^T x \tag{1}$$

$$s.a.\ Ax = b \tag{2}$$

$$x \geq 0 \tag{3}$$

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

Las técnicas para resolver este problema incluyen:

- El Método Simplex (1946) de George Dantzig [3], con complejidad exponencial en el peor caso pero con comportamiento lineal en la práctica. El método avanza por puntos en la frontera del conjunto factible.
- El Método de Elipsoides (1978) de Khachiyan [12] con complejidad de $O(n^4 L)$, pero ineficiente en casos prácticos. Con este método se demostró que el problema de programación lineal estaba en P. La estrategia utilizada involucra la construcción de elipsoides que se van reduciendo hasta "encerrar" el óptimo.
- Los Métodos de Puntos Interiores, desarrollados desde la aparición del Algoritmo de Karmarkar en 1984 [10] con complejidad polinomial con órdenes entre $O(n^3 L)$ y $O(n^{3.5} L)$. Estos métodos han mostrado un excelente desempeño práctico, especialmente para problemas de gran escala. A diferencia del Simplex, estos métodos avanzan por puntos interiores del conjunto factible que convergen al óptimo.

En este trabajo describiremos los detalles relativos a la implementación numérica de un método de puntos interiores para programación lineal . Otros trabajos que enfocan los problemas de implementación subyacentes a estos algoritmos son los de Adler et al. [1], [2]; McShane et al. [15]; Monma y Morton [17]; Marsten et al. [14]; Mehrotra [16]; Gill et al. [6]; entre los más representativos.

El método seleccionado se clasifica como un algoritmo Primal de Reducción de Potencial, el mismo fue presentado por Gonzaga en [9]; en este método sólo se almacenan las variables primales, a diferencia de otros métodos que almacenan o bien las variables duales (métodos duales) o ambos grupos de variables (métodos primales–duales).

2 Algoritmo Numérico

El algoritmo que presentamos es el resultado de reformular el algoritmo de Gonzaga dado en [9], el cual reduce en cada paso la siguiente función llamada función potencial

$$x > 0 \longmapsto f_v(x) = q\log(c^T x - v) - \sum_{i=1}^{n} \log x_i$$

donde v es una cota inferior para el costo óptimo y $q \geq n$ es constante.

Se harán las siguientes hipótesis:

Si $S = \{x/Ax = b \ y \ x \geq 0\}$ (conjunto factible) y $S^\circ = \{x/Ax = b \ y \ x > 0\}$ (interior relativo de S), entonces

(i) A es de rango completo.

(ii) S es acotado.

(iii) S° es no vacío.

(iv) P tiene una solución óptima $\hat{x}$ con costo $\hat{v}$.

Algoritmo Práctico de Reducción de Potencial

Dados $q > n$, $v_\circ < \hat{v}$, $x^\circ \in S^\circ$

$k := 0$

REPEAT

Cambio de Escala: sea $X_k = diag(x_1^k, x_2^k, \ldots, x_n^k)$

$x^k = X_k e$

$\overline{f}_v(e) = q\log(\overline{c}^T e - v_k)$

$\overline{A} = AX_k$

$\overline{c} := X_k c$

Calculos Relativos a la Proyeccion:

$(AX_k^2 A^T)u = AX_k\overline{c}$

$c_p := \overline{c} - X_k A^T u$

$(AX_k^2 A^T)w = Ax^k$

$e_p := e - X_k A^T w$

$\alpha := \frac{e-e_p}{\|e-e_p\|^2}$

Cota Inferior:

$v := \bar{c}^T e - c_p^T + sup\{\mu / c_p - \mu a \geq 0\}$

Direccion: usar una de las formulas siguientes

(i) Cauchy ($q > n$): $h := -\frac{q}{c^T x^k - v} c_p + e_p$

(ii) Karmarkar ($q = n$): $h := -c_p + \frac{\bar{c}(e-e_p) - v_{k+1}}{\|e-e_p\|^2} e_p$

Busqueda Lineal:

$\overline{\lambda} := argmin\{\overline{f}_v(e + \lambda h) / e + \lambda h > 0,\ \lambda > 0\}$

Calculo del nuevo punto factible:

$y := e + \overline{\lambda} h$

Retorno al espacio original:

$x^{k+1} := X_k y$

UNTIL $\frac{|c^T x^k - v|}{|c^T x^k|} < \epsilon$

Donde e_p, c_p son las proyecciones de e y de $\bar{c}$ sobre el espacio nulo de $\overline{A} = AX_k$, respectivamente.

Como puede observarse hay dos direcciones posibles para reducir la función objetivo: la dirección de Cauchy, que da origen a los algoritmos llamados afines y la de Karmarkar, que origina el algoritmo del mismo nombre. Ambas direcciones se obtienen combinando una dirección de minimización del costo y una dirección de centralización o alejamiento de la frontera de S.

En cuanto al cambio de escala, este se realiza en forma implícita en todos los casos, salvo para el vector c para el cual se realiza el cálculo efectivo de $\bar{c}$.

Bajo las hipótesis establecidas, este algoritmo tiene una complejidad de $O(n^{3.5}L)$ operaciones en el peor caso, donde L es el número total de bits necesarios para representar los datos del problema.

De las operaciones que realiza el algoritmo, la más costosa es la asociada a la proyección, pues requiere de la resolución de dos sistemas de ecuaciones, cuya matriz es

$$D_k = AX_k^2 A^T.$$

Además, la matriz de estos sistemas se obtiene como un producto de matrices y esta

operación también es costosa; en las implementaciones debe evitarse el cálculo de este producto. Las implementaciones eficientes [1], [17] conocidas hasta el presente, almacenan la parte invariante del producto (asociada a los elementos de A), para reducir los cálculos; sin embargo, esta estrategia es costosa en almacenamiento.

En razón de lo anterior en la implementación que se describe en este trabajo, se le asignó mayor prioridad a la búsqueda de una alternativa eficiente tanto para resolver los sistemas como para evitar el costo en almacenamiento.

En las siguientes subsecciones describiremos las técnicas y algoritmos utilizados para implementar los siguientes aspectos del algoritmo:

- Proyección.
- Búsqueda lineal.
- Cálculo del punto inicial.
- Cálculo de una cota inferior inicial para $\hat{v}$.
- Escogencia del parámetro q.
- Criterio de parada.

2.1 Proyección

La matriz de los sistemas de ecuaciones involucrados en el cálculo de la proyección, tiene las siguientes características:

- Es simétrica.
- Es positiva definida.
- Es esparcidacon un grado de esparcidad en muchos casos similar al de A, que puede llegar a un 99% para la mayoría de las aplicaciones.

Un método que ha dado excelentes resultados en problemas esparcidade gran tamaño es el Método de Gradiente Conjugado Precondicionado (ver [7] para un excelente estudio

del método), por lo cual se decidió utilizarlo en esta aplicación. No obstante, el método presenta el inconveniente de que requiere del cálculo de un precondicionador para ser eficiente.

Un precondicionador ampliamente utilizado es el factor de Cholesky "incompleto", $\tilde{L}$, que se obtiene al aplicar el Método de Cholesky sin calcular los elementos en posiciones donde hay ceros en la matriz original del sistema para evitar el *fill-in* es decir, la aparición de elementos no nulos en posiciones donde había ceros.

En esta aplicación se utiliza un factor de Cholesky "incompleto" como precondicionador, pero el mismo no se calcula con procedimientos convencionales pues eso implicaría calcular y almacenar D_k o ciertos productos externos (ver [1], [14], [16], [6]).

En lugar de usar el Método de Cholesky directamente sobre D_k, aplicamos transformaciones de Householder para econtrar la descomposición QR de la matriz $B = (AX_k)^T$, es decir calculamos Q ortogonal y R triangular superior tales que $B = QR$. Es fácil observar que R^T es el factor de Cholesky de B^TB. De esta manera *no es necesario* almacenar D_k, sólo se guarda A en formato *sparse*.

Al aplicar la factorización QR (con pivoteo por razones de estabilidad numérica) podría generarse *fill-in* por lo cual en la implementación se sigue una estrategia análoga a la empleada en la factorización incompleta de Cholesky. El factor $\tilde{L}$ así obtenido se utiliza para resolver sistemas triangulares en las iteraciones de Gradiente Conjugado, en razón de lo cual la estabilidad de esos sistemas es de suma importancia. Manteuffel [13] identificó un conjunto de matrices para los cuales el factor incompleto de Cholesky conduce a sistemas estables; es necesario realizar un estudio similar para nuestra estrategia.

Es bien conocido que el *fill-in* en los procedimientos de eliminación depende del orden en que aparecen las filas de la matriz de coeficientes. En nuestro caso cabe esperar que un reordenamiento de las filas y columnas mejoren el comportamiento del precondicionador, tal como lo anuncian Duff y Meurant en [5]. La mayoria de las implementaciones mencionadas utilizan técnicas de reordenamiento con excelentes resultados.

2.2 Búsqueda Lineal

Una vez obtenida la dirección de minimización h es necesario calcular la longitud del paso a tomar sobre esa dirección para reducir la función objetivo, al menos en una cantidad que garantice los resultados teóricos. Nuestra decisión en este sentido fue utilizar el mismo criterio en el que se basan dichos resultados teóricos: se escoge como paso sobre h la distancia del punto factible actual a la frontera del conjunto factible.

Este cálculo produce un punto en la frontera de S y el algoritmo presentado trabaja sobre puntos en el interior de S; por esta razón, una vez calculada la distancia a la frontera, se toma un porcentaje (0.9995 como sugieren McShane et al. [15]) de esa distancia para mantener el iterado en S°. Si bien esta estrategia no es muy elegante, ha dado excelentes resultados en la práctica.

2.3 Cálculo del punto inicial

El algoritmo supone que se conoce un punto interior inicial. Las implementaciones deben proveer algún mecanismo para calcular este punto interior. En nuestro caso se utiliza un método clásico fase I- fase II.

En la fase I se aplica el algoritmo a un problema modificado P', para el cual se conoce un punto interior inicial, y tal que una solución óptima del mismo es un punto interior inicial para el problema original. En base a ese punto, la fase II aplica el algoritmo al problema original.

El problema modificado es el siguiente

$$
\begin{aligned}
P' : \quad & \\
& min\ c^T x + c_a x_a \\
& s.a.\ Ax + (b - A\tilde{x}^\circ)x_a = b \\
& x, x_a \geq 0
\end{aligned}
$$

Para cualquier $\tilde{x}^\circ > 0$ y $x_a = 1$, el vector $< \tilde{x}^\circ, x_a >$ es un punto interior para P'. La fase I del algoritmo debe minimizar x_a, por lo que el costo asociado, c_a, debe ser mayor

que los otros costos. En nuestra implementación tomamos

$$c_a = n^2 * \max_{1 \le i \le n} |c_i|$$

Para determinar la terminación de la fase I se emplea una tolerancia de factibilidad $\gamma \approx 10^{-3}$ que se interpreta de la siguiente forma:

(i) Si $|x_a| < \gamma$, entonces terminó la fase I y el último iterado es un punto interior para P.

(ii) Si se cumple el criterio de parada y $x_a < -\gamma$, entonces el problema no tiene solución acotada.

(iii) Si se cumple el criterio de parada y $x_a \ge \gamma$, entonces el problema no tiene solución factible.

Una elección obvia para x^0 es el punto e, pero también puede usarse cualquier vector que satisfaga la restricción de positividad, en este sentido puede consultarse [15].

2.4 Cálculo de una cota inferior inicial v_0 para $\hat{v}$

Para resolver este problema hay al menos dos opciones, una es calcular la cota inicial usando el procedimiento de Todd y Burrel [18], que implica resolver un sistema de ecuaciones; la otra opción es tomar $v_0 = -\infty$ lo que es válido si el procedimiento de actualización es eficiente, como en el caso del procedimiento de Gonzaga [9]. La decisión en este sentido fue la de tomar v_0 como el menor valor representable en punto flotante de doble precisión.

2.5 Escogencia del parámetro q

Los resultados teóricos [8] que establecen polinomialidad para algoritmos afines requieren que $q \ge n + \sqrt{n}$, $q = O(n)$; el algoritmo de Karmarkar requiere que $q = n$.

Decidimos utilizar el valor de $q = K * (n + \sqrt{n})$, para la versión afín polinomial del algoritmo, y $q = n$ para el algoritmo de Karmarkar. La constante K se utiliza para dar

mayor peso a la componente de minimización del costo en la dirección de reducción. En la actualidad se utiliza $K = \max_{1 \leq i \leq n} |c_i|$.

2.6 Criterio de Parada

En lugar de usar una precisión de 2^{-L} como en los resultados teóricos se escoge una precisión de $\epsilon \approx 10^{-8}$, pero también puede usarse el ϵ de la máquina.

Puede ocurrir que por problemas de precisión el criterio anterior no llegue a satisfacerse, por lo que se require de un criterio adicional que en nuestro caso consiste en detectar un incremento en la función objetivo.

3 Resultados

En la actualidad se realizan experimentos sobre datos de prueba obtenidos a través de *netlib* [4] con una implementación desarrollada en lenguaje C [11] sobre un econsiste en SUN [1] Sparc 1+ bajo Unix [2]. En la exposición se presentarán resultados numéricos sobre dichos datos.

4 Conclusiones

De la experiencia obtenida en este trabajo, se ha determinado que es posible realizar implementaciones eficientes de los algoritmos de puntos interiores para programación lineal , siempre que se tome en cuenta a cabalidad las características del problema para poder lograr un equilibrio entre utilización de memoria y número de operaciones.

Lo anterior se aplica en particular a la resolución de sistemas de ecuaciones esparcida, simétricas y positivas definidas que los mencionados algoritmos contemplan. En este sentido, se considera muy prometedor el enfoque seguido en este trabajo, especialmente si se utilizan técnicas de reordenamiento y más aún si se logra incorporar el uso del paralelismo en estas operaciones.

[1] SUN es marca registrada de SUN Microsystems Inc.

[2] Unix es marca registrada de AT&T

Cabe destacar que existen otros aspectos de implementación que deben ser analizados en detalle pues pueden contribuir a mejorar sustancialmente la eficiencia de la misma. Entre estos tenemos: el cálculo de c_a en la fase I, el procedimiento para actualizar la cota inferior y el cálculo de q.

Referencias

[1] I. Adler, N. Karmarkar, M. Resende, y G. Veiga. Data structures and programming techniques for the implementation of Karmarkar's algorithm. *ORSA Journal on Computing*, 1(2), Abril 1989.

[2] I. Adler, N. Karmarkar, M. Resende, y G. Veiga. An implementation of Karmarkar's algorithm for linear programming. *Mathematical Programming*, 44, 1989.

[3] G. Dantzig. *Linear Programming and Extensions.* Princeton University Press, New Jersey, 1963.

[4] J. Dongarra y E. Grosse. Distribution of mathematical software via electronic mail. *Communications of the ACM*, 30(5), 1987.

[5] I. Duff y G. Meurant. The effect of ordering on Preconditioned Conjugate Gradients. *BIT*, 29, 1989.

[6] P. Gill, W. Murray, J. Tomlin, y M. Wright. On projected Newton barrier methods for linear programming and an equivalence to Karmarkar's projective method. *Mathematical Programming*, 36, 1986.

[7] G. Golub y Ch. Van Loan. *Matrix Computations.* The John Hopkins University Press, Baltimore, 2ª edición, 1989.

[8] C. Gonzaga. Polynomial affine algorithm for linear programming. Reporte Técnico, COPPE-UFRJ, Río de Janeiro, Enero 1988.

[9] C. Gonzaga. On lower bounds updates in primal potential reduction algorithm for linear programming. Reporte Técnico, COPPE-UFRJ, Río de Janeiro, Junio 1990.

[10] N. Karmarkar. A new polynomial time algorithm for linear programming. *Combinatorica*, 4, 1984.

[11] B. Kernighan y D. Ritchie. *The C Programming Language.* Prentice–Hall, Inc., New Jersey, 1978.

[12] L. Khachiyan. A polynomial algorithm for linear programming. *Soviet Mathematics Doklady*, 20, 1979.

[13] T. Manteuffel. Shifted incomplete cholesky factorization. In I. Duff y G. Stewart, editores, *Sparse Matrices Symposium, 1978.* Publicaciones de SIAM , 1979.

[14] R. Marsten, M. Saltzman, D. Shanno, G. Pierce, y J. Ballintijn. Implementation of a dual affine interior point algorithm for linear programming. *ORSA Journal on Computing*, 1(4), 1989.

[15] K. McShane, C. Monma, y D. Shanno. An implementation of a primal dual interior point method for linear programming. *ORSA Journal on Computing*, 1(2), 1989.

[16] S. Mehrotra. Implementations of affine scaling methods: Approximate solutions of systems of linear equations using Preconditioned Conjugate Gradient Methods. Reporte Técnico 89-04, Department of Industrial Engineering and Management Sciences, Northwestern University, Agosto 1989.

[17] C. Monma y A. Morton. Computational experience with a dual affine variant of Karmarkar's method for linear programming. *Operations Research Letters*, 6, 1987.

[18] M. Todd y B. Burrell. An extension of Karmarkar's algorithm for linear programming using dual variables. *Algorithmica*, 1, 1986.