Proyecto I: Método Simplex Fase II

Andrés Ángeles C.U. 131749

Mauricio Trejo C.U. 138886

20 de Octubre 2018

1. Implementación de la Fase II Revisada

La Fase II revisada o modificada es una implementación matricial de la Fase II del Método Simplex con dos fases. Sus principales objetivos son eliminar el uso de matrices inversas para calcular el vector de costos relativos y disminuir el uso de memoria sustituyendo los diccionarios por dos conjuntos de índices que se actualizan en cada paso y que indican qué variables entran y salen de la base.

Consideremos el siguiente problema[1] y verifiquemos que nuestro programa lo resuelve correctamente:

$$\begin{aligned} \text{maximizar} \quad & 3x_1 + x_2 + 3x_3 \\ \text{sujeto a} \quad & 2x_1 + x_2 + x_3 \leq 2 \\ & x_1 + 2x_2 + 3x_3 \leq 5 \\ & 2x_1 + 2x_2 + x_3 \leq 6 \\ & x_1, x_2, x_3 \geq 0 \end{aligned} \tag{1}$$

Si multiplicamos la función objetivo por -1 y agregamos variables de holgura x_4, x_5, x_6, entonces obtenemos el siguiente problema de minimización en forma estándar

$$\begin{aligned} \text{minimizar} \quad & -3x_1 - x_2 - 3x_3 \\ \text{sujeto a} \quad & 2x_1 + x_2 + x_3 + x_4 = 2 \\ & x_1 + 2x_2 + 3x_3 + x_5 = 5 \\ & 2x_1 + 2x_2 + x_3 + x_6 = 6 \\ & x \geq \mathbf{0} \end{aligned} \tag{2}$$

El tableau asociado al problema (2) es

	x_1	x_2	x_3	x_4	x_5	x_6	LD
x_4	2	1	1	1			2
x_5	1	2	3		1		5
x_6	2	2	1			1	6
	3	1	3				

Cuadro 1: Tableau en estado 0 asociado al problema (2)

Usamos la Regla de Bland para escoger las variables de entrada y de salida.

	x_1	x_2	x_3	x_4	x_5	x_6	LD
x_1	1	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$			1
x_5		$\frac{3}{2}$	$\frac{5}{2}$	$-\frac{1}{2}$	1		4
x_6		1		-1		1	4
		$-\frac{1}{2}$	$\frac{3}{2}$	$-\frac{3}{2}$			-3

Cuadro 2: Tableau en la 1$^{\text{ra}}$ iteración

[1]Este problema lo utiliza David Luenberger en su libro *Linear and Nonlinear Programming* para ejemplificar el uso del tableau en el capítulo tres, *El Método Simplex*, p.48.

	x_1	x_2	x_3	x_4	x_5	x_6	LD
x_1	1	$\frac{1}{5}$		$\frac{3}{5}$	$-\frac{1}{5}$		$\frac{1}{5}$
x_3		$\frac{3}{5}$	1	$-\frac{1}{5}$	$\frac{2}{5}$		$\frac{8}{5}$
x_6		1		-1		1	4
		$-\frac{7}{5}$		$-\frac{6}{5}$	$-\frac{3}{5}$		$-\frac{27}{5}$

Cuadro 3: Tableau final del problema (2)

La solución del problema (2) es $x^* = (x_1, x_2, x_3, x_4, x_5, x_6)^T = (\frac{1}{5}, 0, \frac{8}{5}, 0, 0, 4)^T$ y el valor óptimo es $z^* = -\frac{27}{5}$, por lo que la solución y el valor óptimo del problema original son

$$x^* = (x_1,\ x_2,\ x_3)^T = \left(\frac{1}{5},\ 0,\ \frac{8}{5}\right)^T, \quad z^* = \frac{27}{5}$$

Aquí incluimos el resultado obtenido en la consola de MATLAB al ejecutar nuestro programa[2]:

```
>> A = [2 1 1; 1 2 3; 2 2 1]; b = [2; 5; 6]; c = [3; 1; 3]; c = -c;
>> [x0, z0, ban, iter] = mSimplexFaseII(A, b, c, false)

x0 =

       1/5
       0
       8/5


z0 =

     -27/5


ban =

       0


iter =

       2
```

[2]El código de nuestra implementación de la Fase II del Método Simplex se encuentra al final del documento en el anexo junto con todos los programas que generan las gráficas y los demás resultados que se presentan a continuación.

2. El problema de Klee-Minty

Tomonari Kitahara y Shinji Mizuno presentaron una versión simplificada del problema de Klee-Minty de dimensión m, el cual queda dado por

$$\begin{array}{ll} \text{minimizar} & -\sum_{i=1}^{m} x_i \\ \text{sujeto a} & x_1 \leq 1 \\ & 2\sum_{j=1}^{i-1} x_j + x_i \leq 2^i - 1 \quad \text{para} \quad i = 2, \ldots, m \\ & x_i \geq 0 \quad \text{para} \quad i = 1, \ldots, m \end{array} \tag{3}$$

Este problema tiene algunas propiedades importantes que discutiremos brevemente.

Sea $h^T = (h_1, \ldots, h_m) \in \mathbb{R}^m$ un vector de variables de holgura, entonces el problema (3) en forma estándar es

$$\begin{array}{ll} \text{minimizar} & -\sum_{i=1}^{m} x_i \\ \text{sujeto a} & x_1 + h_1 = 1 \\ & 2\sum_{j=1}^{i-1} x_j + x_i + h_i = 2^i - 1 \quad \text{para} \quad i = 2, \ldots, m \\ & x_i, h_i \geq 0 \quad \text{para} \quad i = 1, \ldots, m \end{array} \tag{4}$$

El problema en forma estándar tiene m restricciones de igualdad y $n = 2m$ variables.

Lema 1 *El problema (4) tiene las siguientes propiedades*

1. *En cada SBF, existe $k \in \{1, 2, \ldots, m\}$, tal que x_k y h_k son variables básicas.*

2. *Existen 2^m soluciones básicas factibles.*

3. *La Fase II revisada realiza $(2^m - 1)$ iteraciones para resolver el problema.*

El tiempo de máquina y el número de iteraciones que realizó nuestra implementación de la Fase II para resolver el problema de Klee-Minty para $m = 3, \ldots, 10$ se muestran en el cuadro que sigue

Dimensión (m)	Iteraciones	Tiempo de máquina
3	7	0.0071
4	15	0.0246
5	31	0.0093
6	63	0.0019
7	127	0.0068
8	255	0.0056
9	511	0.0152
10	1023	0.0227

Cuadro 4: Tabla de resultados de resolver el problema (4) para distintos valores de m por primera vez

Notemos que el número de iteraciones en la segunda columna de la tabla que se muestra en el cuadro 4 cumple el tercer enunciado del lema (1), pues, para cada $m \in \{3, 4, \ldots, 10\}$, el número de iteraciones es $2^m - 1$.

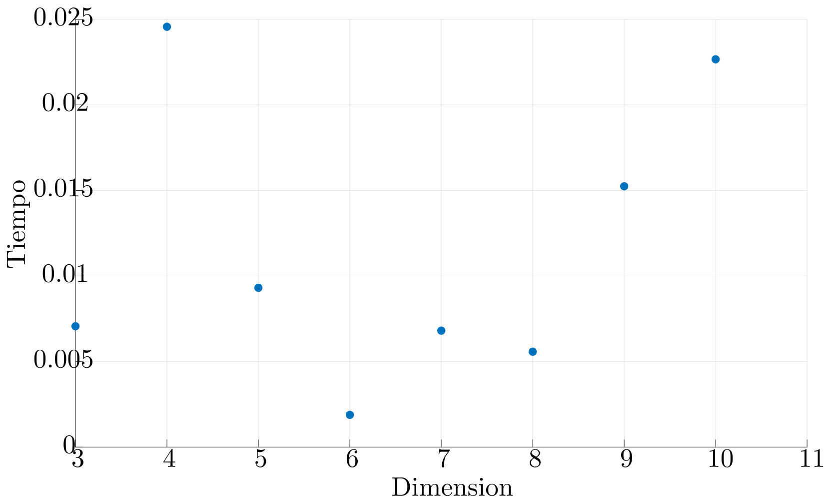

Figura 1: Gráfica de la dimensión (m) vs el número de iteraciones realizadas para resolver el problema (4) para distintos valores de m por primera vez

La gráfica anterior (Figura 1) nos muestra un resultado contradictorio: resolver el problema de Klee-Minty de dimensión $m = 4$ ocupa más tiempo de máquina que resolver los problemas para $m = 5, 6, \ldots, 10$ a pesar de que la Fase II realiza más iteraciones para llegar a la solución. Sin embargo, los resultados mejoran si ejecutamos el código más de una vez.

En el siguiente recuadro presentamos la tabla seguida de la gráfica de los tiempos de máquina para resolver cada problema que obtuvimos después de ejecutar el script `SimplexKleeMinty` un total de quince veces.

Dimensión (m)	Iteraciones	Tiempo de máquina
3	7	0.0003
4	15	0.0004
5	31	0.0006
6	63	0.0013
7	127	0.0025
8	255	0.0062
9	511	0.0100
10	1023	0.0214

Cuadro 5: Tabla de resultados de resolver el problema (4) para distintos valores de m después de 15 repeticiones

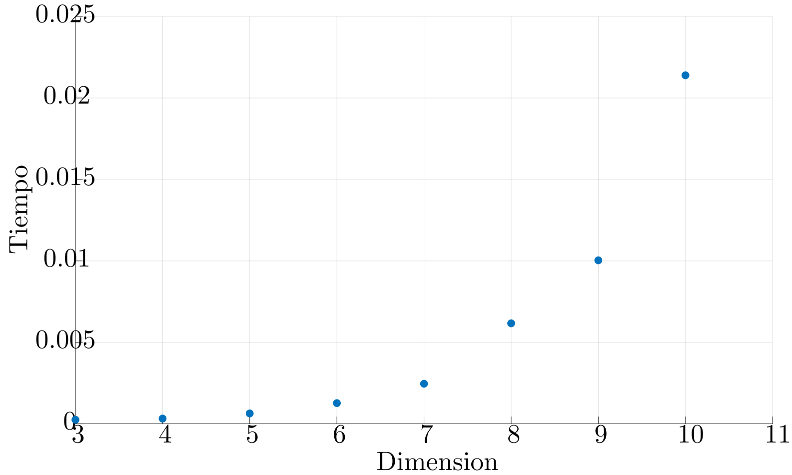

Figura 2: Gráfica de la dimensión (m) vs el número de iteraciones realizadas para resolver el problema (4) para distintos valores de m después de 15 repeticiones

En la gráfica correspondiente a la Figura 2 se aprecia claramente que el tiempo de máquina aumenta de forma exponencial conforme aumenta la dimensión del problema de Klee-Minty.

3. Estudio de la complejidad computacional de la Fase II revisada

Definición 1 (Notación de Bachmann–Landau) *Sean $\Omega \subset \mathbb{R}^+$ un conjunto no acotado y $f, g : \Omega \longrightarrow \mathbb{R}$ dos funciones tales que g es estrictamente positiva a partir de un valor $x_0 \in \Omega$, entonces escribimos que*

$$f(x) = \mathcal{O}(g(x)) \quad \textit{conforme} \quad x \longrightarrow \infty$$

si y sólo si $\exists\ M \in \mathbb{R}^+\ \exists\ x_0 \in \Omega\ \Big(|f(x)| \leq Mg(x)\Big)$

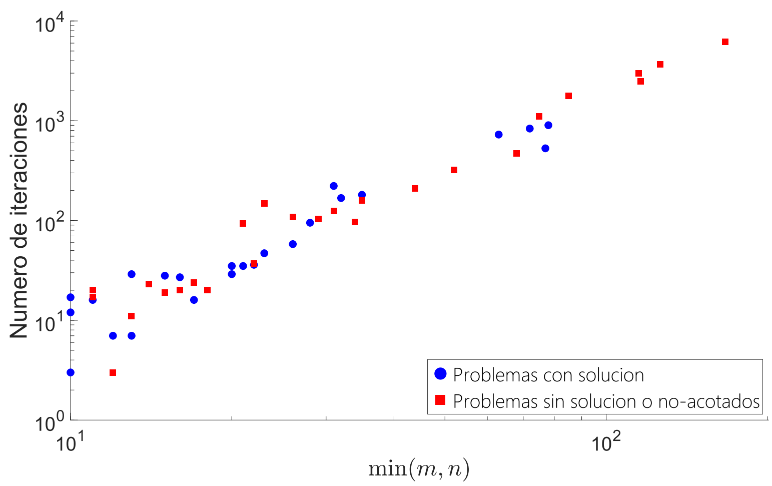

Figura 3: Gráfica de la mínima dimensión ($\text{mín}(m, n)$) vs el número de iteraciones realizadas para resolver cada problema para 50 problemas de Programación Lineal

La gráfica de la Figura 3 nos muestra que la relación entre $\log(\text{mín}(m,n))$ y el número de iteraciones en escala logarítmica, $\log(\gamma)$ es lineal, es decir,

$$\log(\gamma) = p \log\big(\text{mín}(m,n)\big) + k \quad \text{donde} \quad p \in \mathbb{R}^+,\ k \in \mathbb{R}$$

entonces

$$\begin{aligned}
& 10^{\log(\gamma)} = 10^{p\log\left(\text{mín}(m,n)\right)+k} \\
\implies \quad & \gamma = 10^k \,\text{mín}(m,n)^p \\
\implies \quad & \gamma = k_1\big(\text{mín}(m,n)\big)^p \quad \text{donde} \quad k_1 \in \mathbb{R}^+
\end{aligned}$$

Además, por definición, sabemos que el mínimo de m y n es menor o igual que ambas y que $m+n$ es estrictamente mayor que m y n, pues no existen problemas de Programación Lineal con dimensiones menores o iguales a cero, entonces

$$\begin{aligned}
& \big(\text{mín}(m,n)\big)^p \leq m^p, n^p \quad \text{y} \quad m^p, n^p \leq (m+n)^p \\
\implies \quad & \big(\text{mín}(m,n)\big)^p \leq (m+n)^p \qquad \forall\, m,n \geq 1 \\
\implies \quad & \gamma = k_1\big(\text{mín}(m,n)\big)^p \leq k_1(m+n)^p \qquad \forall\, m,n \geq 1
\end{aligned}$$

por lo que $\gamma = \mathcal{O}\big((m+n)^p\big)$ y podemos concluimos que el número de iteraciones, γ, incrementa de forma polinomial con grado $p \in \mathbb{N}$ conforme aumenta el mínimo entre m y n.

Anexos

A. Implementación de la Fase II Revisada

A.1. Código con problemas de prueba

MATLAB/probar_mSimplexFaseII.m

```
% Problemas de prueba para la funcion mSimplexFaseII

% 1 Ejemplos de clase

    fprintf("\n1 Ejemplos de clase\n");

    % 1.1 Problema acotado con conjunto factible no-vacio

    % Solucion: z* = -36, B = {1,2,3}, x* = (2,6,2)
    fprintf("\n1.1 Problema acotado con conjunto factible no-vacio\n
        ");
    A = [1 0; 0 2; 3 2]; b = [4; 12; 18]; c = [-3; -5];
    [x0, z0, ban, iter] = mSimplexFaseII(A, b, c, false)

    % 1.2 Problema con SBF degenerada

    % Solucion: z* = -2, B = {1,2}, x* = (2,2)
    % SBF degenerada en el estado 0
    fprintf("\n1.2 Problema con SBF degenerada en el estado 0\n");
    A = [-1 1; 1 0]; b = [0; 2]; c = [0; -1];
    [x0, z0, ban, iter] = mSimplexFaseII(A, b, c, false)

    % 1.3 Problema no acotado

    % No acotado en la 1ra iteracion
    fprintf("\n1.3 Problema no acotado en la 1ra iteracion\n");
    A = [1 -1; -1 1]; b = [1; 2]; c = [-1; 0];
    [x0, z0, ban, iter] = mSimplexFaseII(A, b, c, false)

% 2 Vanderbei

    fprintf("\n2 Vanderbei: ejercicios con soluciones\n");

    % 2.1 Problema acotado de maximizacion con conjunto
    % factible no vacio

    % Solucion: z* = 17, B = {1,3}, x* = (2,0,1,0)
    fprintf(['\n2.1 Problema de maximizacion acotado con conjunto'...
```

```
        ' factible no-vacio\n']);
    A = [2 1 1 3; 1 3 1 2]; b = [5; 3]; c = [6; 8; 5; 9]; c = -c;
    [x0, z0, ban, iter] = mSimplexFaseII(A, b, c, false)

    % 2.2 Problema de maximizacion con conjunto factible vacio

    fprintf(['\n2.2 Problema de maximizacion con conjunto '...
        ' factible vacio\n']);
    A = [-1 -1; -1 1; 1 2]; b = [-3; -1; 2]; c = [1; 3]; c = -c;
    [x0, z0, ban, iter] = mSimplexFaseII(A, b, c, false)

% 3 Luenberger

    fprintf("\n3 Ejemplos Luenberger\n");

    % 3.1 Problema de maximizacion acotado con conjunto factible no-
        vacio

    % Solucion: z* = 27/5, B = {1,3,6}, x* = (1/5, 8/5, 4)
    fprintf(['\n3.1 Problema de maximizacion acotado con conjunto'...
        'factible no-vacio\n']);
    A = [2 1 1; 1 2 3; 2 2 1]; b = [2; 5; 6]; c = [3; 1; 3]; c = -c;
    [x0, z0, ban, iter] = mSimplexFaseII(A, b, c, true)
```

A.2. Código de la Fase II Revisada

MATLAB/mSimplexFaseII.m

```
function [x0, z0, ban, iter] = mSimplexFaseII(A, b, c, imprimirPasos)

% Esta funcion realiza la Fase II del Metodo Simplex para problemas
% que tienen la siguiente forma
%
%                 minimizar    c'x
%                 sujeto a     Ax <= b  , x >= 0  , b >= 0
%
% In :  A ... mxn matrix
%       b ... column vector with as many rows as A
%       c ... column vector with as many entries as one row of A
%       imprimirPasos ... boolean variable which indicates whether
%       or not to print the step-by-step solution of the given
    problem
%
% Out:  xo ..... SBF optima del problema
%       zo ..... valor optimo del problema
%       ban .... indica casos:
%           -1 ... si el conjunto factible es vacio
%           0 .... si se encontro una solucion optima
```

```
%           1 .... si la funcion objectivo no es acotada.
%       iter ... numero de iteraciones (cambios de variables basicas)
%       que hizo el metodo

    format rat; %MATLAB imprime fracciones en vez de decimales

    iter = 0;

    %1 Definicion de las variables en estado 0

    [m, n] = size(A); %m variables basicas
    ban = 0;
    [N, B, c, A] = deal( 1:n, (n+1):(m+n), [c' zeros(1,m)], [A eye(m)
        ] );
    [lambda, h] = deal( c(B), b );
    rN = lambda*A(:, N) - c(N);

    %Si el conjunto factible es vacio, el Metodo Simplex no
    %tendra opcion mas que escoger un punto que no cumpla
    %la restriccion de no-negatividad.
    if any(b < 0)

        if imprimirPasos
            fprintf("\n\nConjunto factible vacio\n");
        end

        ban = -1;

    end

    if imprimirPasos
       imprimirTableau(A(:, N), c(B), rN, h, B, N, iter);
    end

    %2 Probamos la condicion de optimalidad
    while any(rN > 0) && ban == 0

        %2.1 Seleccionamos la variable de entrada mediante
        %la Regla de Bland
        e = find(rN > 0, 1);

        %3 Probamos si el problema es acotado
        if any(A(:, N(e)) > 0)

            %3.1 Seleccion de la variable de salida mediante
            %la Regla de Bland
```

```
        %Buscamos los indices de los denominadores no-positivos
        noPositivos = A(:, N(e)) <= 0;

        %Calculamos los cocientes y asignamos infinito a los
        %cocientes que tienen denominador menor o igual a cero
        cocientes = h./A(:, N(e));
        cocientes(noPositivos) = inf;

        %s es el indice que corresponde al minimo de cocientes
        [~, s] = min(cocientes);

        if imprimirPasos
            fprintf(['La variable de entrada es X%d y la '...
                'variable de salida es X%d\n'],[N(e), B(s)]);
        end

        %4 Redefinimos los conjuntos B y N
        [B(s), N(e)] = deal( N(e), B(s) );
        [A(:, B), A(:, N), h] = deal( eye(m),...
            A(:, B)\A(:, N), A(:, B)\h );
        iter = iter + 1;

        %4.1 Calculamos los nuevos costos relativos
        lambda = A(:, B)'\c(B)';
        lambda = lambda';
        rN = lambda*A(:, N) - c(N);

        if imprimirPasos
            imprimirTableau(A(:, N), c(B), rN, h, B, N, iter);
        end

    else

        %El problema es no-acotado.
        ban = 1;
        if imprimirPasos
            fprintf("\n\nProblema no-acotado\n");
        end

    end

end

if ban == 0
    x0 = zeros(m+n, 1);
    x0(B) = A(:,B)\h;
    z0 = c*x0;
```

```
        % Solo devolvemos los valores de las variables originales
        x0 = x0(1:n);
    else
        [x0, z0] = deal( [], [] );
    end

    return;

end

function imprimirTableau(AN, cB, rN, h, B, N, iter)

% Este metodo imprime el tableau asociado a la base B
% usando las matrices AB, AN y los vectores rN y h

    m = length(B);
    n = length(N);

    T = zeros(m+1,m+n+1);

    T(1:m, B) = eye(m);
    T(1:m, N) = AN;
    T(1:m, m+n+1) = h;
    T(m+1, N) = rN;
    T(m+1, m+n+1) = cB*h;

    fprintf("\nVariables basicas:      ");
    disp(B);
    fprintf("Variables no-basicas: ");
    disp(N);
    fprintf("\nIteracion %d\n\n", iter);
    disp(T);

    return;

end
```

B. El problema de Klee-Minty

B.1. Código para generar el problema de tamaño m

MATLAB/generaKleeMinty.m

```
function [A,b,c] = generaKleeMinty(m)

c = -ones(m, 1);
b = 2.^(1:m)' - 1;
```

```
A = 2*tril(ones(m, m), -1) + eye(m);

end
```

B.2. Código que mide el tiempo de máquina de la solución

MATLAB/SimplexKleeMinty.m

```
% Este script genera y resuelve el problema de Klee-Minty de
% dimension m = 3, 4,..., 10 usando la funcion generaKleeMinty y
% mSimplexFaseII para medir el tiempo y el numero de iteraciones
% que realiza el metodo en cada caso

M = 10;

iter = 3:M;
t = 3:M;

for m = 3:M
    [A, b, c] = generaKleeMinty(m);
    tic;
    [~, ~, ~, iter(m-2)] = mSimplexFaseII(A, b, c, false);
    t(m-2) = toc;
end

format; % Restablece el formato de impresion de MATLAB

fprintf("\n          m:");
disp(3:M);
fprintf("\nIteraciones:");
disp(iter);
fprintf("\n     Tiempo:");
disp(t);

scatter(3:M, t, 120, 'o', 'filled');
set(gca, 'xlim', [3, 11], 'fontsize', 20);
ylabel('Tiempo', 'fontname', 'Segoe UI Light', 'fontsize', 30);
xlabel('Dimension', 'fontname', 'Segoe UI Light', 'fontsize', 30);
grid on
```

C. Estudio de la complejidad computacional de la Fase II revisada

MATLAB/simplexEmpirico.m

```
% Este script usa la funcion mSimplexFaseII para resolver N problemas
```

```
% generados de manera aleatoria. En cada vuelta, se guardan el minimo
% entre las dimensiones del problema min(m,n), el numero de
    iteraciones
% y una variable booleana que indica si el problema tenia solucion o
    no

N = 5; % Numero de problemas que se quieren generar

res = zeros(N, 3);
res(:, 3) = false;

for i = 1:N
    % Generar dimensiones del problema
    m = round(10*exp(log(20)*rand()));
    n = round(10*exp(log(20)*rand()));

    % Generar A, b, c
    sigma = 100;
    A = round(sigma*randn(m,n));
    b = round(sigma*abs(randn(m,1)));
    c = round(sigma*randn(n,1));

    % Llamamos a la funcion mSimplexFaseII y resolvemos el problema
    [~, ~, ban, iter] = mSimplexFaseII(A, b, c, false);

    % Guardamos los resultados que nos interesan
    if ban == 0
        res(i, :) = [min(m,n), iter, true];
    else
        res(i, 1:2) = [min(m,n), iter];
    end
end
```