Proyecto 1 – Documentación

Juan Carlos Sigler Priego
174215

Carlos Mier Figueroa
174209

Óscar Arévalo
170507

Verano 2020

1 Introducción

En este proyecto se implementó la metodología especificada para el Proyecto I para explorar problemas de programación lineal y el método simplex. Se crearon en Matlab las funciones mSimplexFaseII_eq y mSimplex_leq para resolver la fase II y I del método Simpex, respectivamente. Después, se utilizaron estas funciones para explorar la complejidad del problema con problemas tipo Klee-Minty, así como la complejidad de manera empírica de un problema promedio de programación lineal con un set de cincuenta problemas aleatorios.

2 Resultados

2.1 Un problema con complejidad exponencial

Analizamos el ejemplo de Klee–Minty (simplificado, tomado de Kitahara, Mizuno):

minimizar

$$-\sum_{i=1}^{m} x_i$$

sujeto a

$$x_1 \leq 1$$

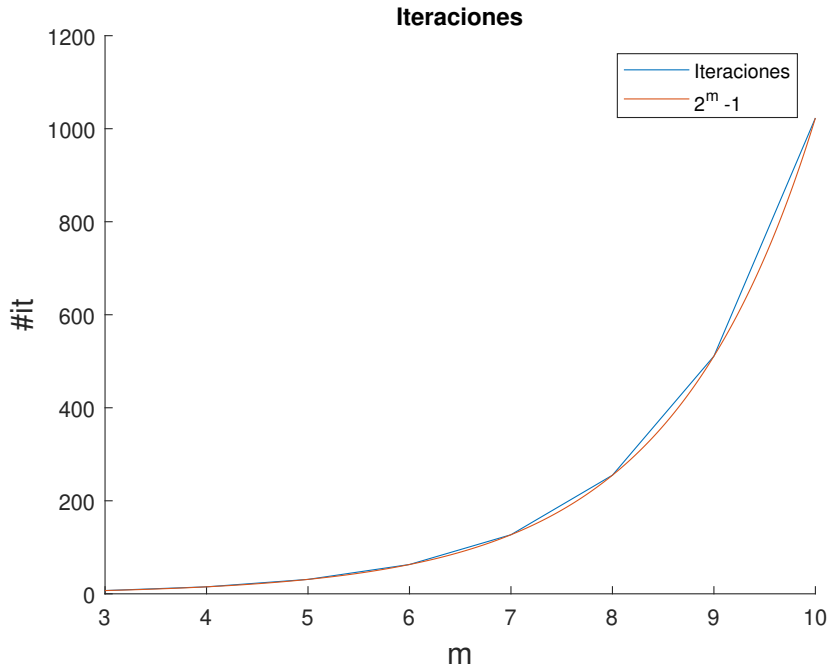
$$2\sum_{j=1}^{i-1} x_j + x_i \leq 2^i - 1, i = 2, ..., m$$

$$x_1, ..., x_m \geq 0$$

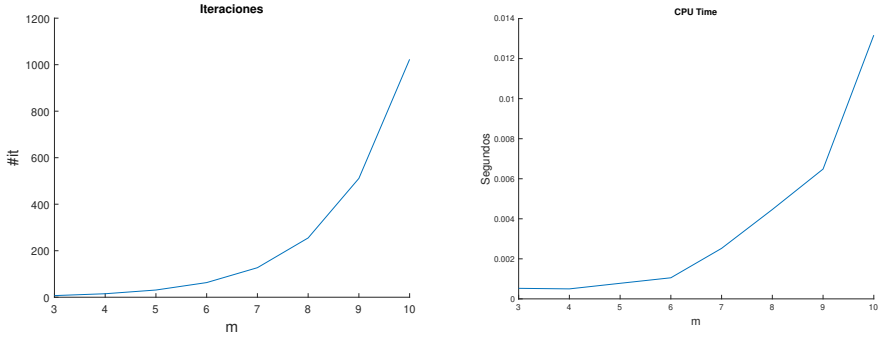
Utilizando el script "scriptKleeMinty.m" resolvemos el problema de Klee-Minty para las dimensiones $m = 3, 4, 5, 6, 7, 8, 9, 10$. El script nos arroja el número de iteraciones para cada dimensión junto con el tiempo de máquina. Reportamos los resultados en la siguiente tabla:

m	Iteraciones	CPU_Time(s)
3	7	0.0005261
4	15	0.0004988
5	31	0.0007774
6	63	0.0010533
7	127	0.0025228
8	255	0.0044635
9	511	0.0064833
10	1023	0.0131729

Vemos que, la relación que existe entre la m y la cantidad de iteraciones es $Iteraciones = 2^m - 1$. Es decir, la complejidad del algoritmo en un orden exponencial. Esto se muestra en las siguientes gráficas:



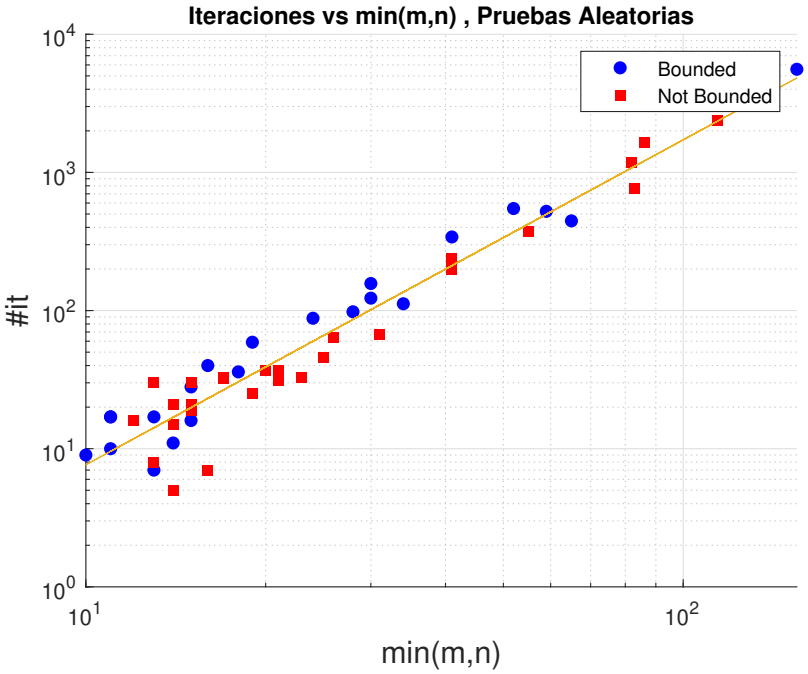
Además, vemos como existe una relación lineal entre la cantidad de operaciones y el tiempo de procesamiento de la CPU.



2.2 Estudio empírico de complejidad con problemas aleatorios.

Ahora, con el objetivo de analizar la complejidad de los problemas utilizando el método simplex analizamos cincuenta problemas aleatorios e intentamos deducir una expresión aproximada del error.

En primera instancia, se compara el minímo entre las dimensiones m y n contra el número de iteraciones que requiere cada problema, esto en escala logarítmica en los ejes X y Y. Con azul se ven los problemas acotados y con rojo los no acotados.



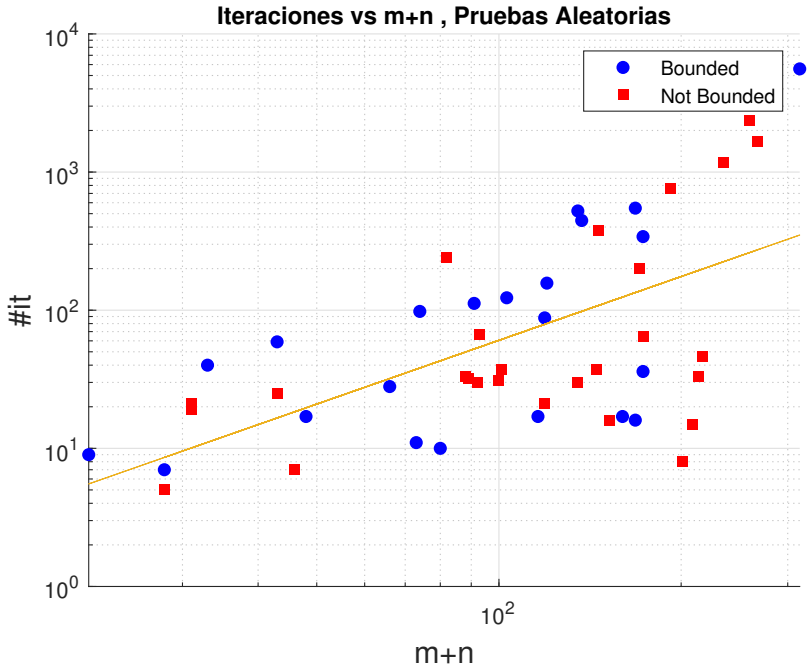
Con el fin de deducir una aproximación de la complejidad del método simplex del problema "promedio", realizamos una regresión lineal de los datos en escala

logarítmica. Deducimos así los valores estadísticos de p y C de la expresión:

$$\log(\#it) = p\log(\min(n,m)) + C \tag{1}$$

obtenemos entonces $p = 2.3523$ y $C = -3.3816$.

Utilizando otra idea para analizar la complejidad del método simplex para el problema promedio, comparamos el número de iteraciones que tomó cada problema con el valor de $m + n$ de cada problema. Los graficamos en escala log-log. Observamos entonces que los datos se ven más dispersos que cuando comparamos las iteraciones con $\min(m, n)$.



Con el fin de deducir otra aproximación de la complejidad del método simplex de problema "promedio", realizamos una regresión lineal respecto a estos datos en escala logarítmica. Así obtenemos una expresión de la complejidad de la siguiente forma:

$$\log(\#it) = p\log(m+n) + C \tag{2}$$

Con valores estadísticos de $p = 1.5336$ y $C = -2.9595$.
A nuestro parecer y basado en la experimentación llevada a cabo, la complejidad del método simplex del problema "promedio" se puede explicar de mejor manera con la expresión (1), ya que los datos se ajustan mejor a una regresión lineal basada en esta expresión. Proponemos entonces la siguiente expresión como aproximación de la complejidad del método simplex de problema "promedio":

$$\#it = e^{2.3523\log(\min(m,n))-3.3816}$$

$$\#it = \min(m,n)^{2.3523} e^{-3.3816}$$

Esto significa que el método simplex tiene una complejidad $\mathcal{O}(\min(m,n)^p)$, es decir, de complejidad polinomial con $p \approx 2$ (más cercano a 2 que a 3).

3 Detalles de implementación

A continuación listamos los archivos que generamos para la funcionalidad del proyecto:

- mSimplexFaseII_eq.m
  - Este archivo resuelve el problema min $c^T x$ sujeto a $Ax = b, x \geq 0$ dado una SBF. Una nota a recalcar es que utilizamos la regla de Bland para asegurar que el método de Simplex termine.
- mSimplex_leq.m
  - Este archivo resuelve el problema min $c^T x$ sujeto a $Ax \leq b, x \geq 0$.
- generaKleeMinty.m
  - Este archivo nos permite generar A, b, c del problema de Klee-Minty.
- scriptKleeMinty.m
  - En este archivo se hace el análisis de la sección 2.1
- generaProblemaAleatorio.m
  - Este archivo nos permite generar A, b, c de un problema aleatorio de programación lineal.
- scriptEmpirico.m
  - En este archivo se hace el análisis de la sección 2.2

Además, implementamos los siguientes tres archivos que nos ayudaron a verificar que la implementación fuera correcta.

- main.m
  - Aquí agregábamos un problema de programación lineal para probarlo independiente del proyecto.
- tablau.m
  - Esta función imprime de manera estética el tablau y nos ayudó mucho para checar cómo funcionaba nuestro código. Para hacer una prueba, puede descomentar la función tableau que se encuentra en mSimplexFaseIIeq.m (hay un comentario dentro del código que va a ver) y después correr main.m. Esto imprimirá el tableau en cada iteración del método.

- tests.m
  - Este archivo nos sirvió como una forma de comprobar el correcto funcionamiento del método de Simplex implementado. Aquí verificamos para muchos ejercicios vistos en clase que el código obtenga los resultados esperados. Así, cada que modificábamos el código, corríamos este archivo para asegurar que siguiera obteniendo las respuestas esperadas. Si todas las pruebas pasaban obteníamos el mensaje "Tests succeded", en caso contrario se muestra "Assertion failed" y podíamos ver qué caso está fallando para correrlo por separado.