INSTITUTO TECNOLÓGICO AUTÓNOMO DE MÉXICO

Optimización Numérica | Proyecto 1 – Reporte

Juan Carlos Sigler   Alonso Martinez   Paulina Carretero

Índice general

1 Marco teórico

En el presente documento discutimos una implementación del algoritmo del conjunto activo para resolver problemas de programación cuadrática con restricciones tanto de igualdad como desigualdad, como en el problema (P) a continuación:

$$\begin{array}{ll} \min & \frac{1}{2}\vec{x}^\top G\vec{x} + \vec{c}^\top\vec{x} \\ \text{sujeto a} & Ax_i = b_i \quad i \in \mathcal{E} \\ & Ax_j \leq b_j \quad j \in \mathcal{I} \end{array} \tag{P}$$

Nos servimos de las libreías `JuMP` [DHL17], y `MAT` del ecosistema Julia para aplicar el método Simplex y para leer archivos en formato `.mat` respectivamente. Adicionalmente llevamos a cabo *benchmarks* con el paquete `BenchmarkTools` para evaluar el desempeño de nuestro algoritmo y pruebas unitarias para hacer el proceso de desarrollo más fácil.

2 Problemas

Probamos nuestra implementación con tres problemas y presentamos los resultados de acuerdo a lo especificado en la asignación del proyecto.

2.1 Problema 1: Problema chico

$$\min q(x) = (x-1)^2 + (y-2.5)^2$$

Sujeto a

$$\begin{aligned} -x + 2y - 2 &\leq 0 \\ x + 2y - 6 &\leq 0 \\ x - 2y - 2 &\leq 0 \\ -x &\leq 0 \\ -y &\leq 0 \end{aligned}$$

Con $x_0 = (2, 0)^\top$ & $W_0 = \{3\}$.

Para este problema, nuestro algoritmo imprime lo siguiente:

```
Rama 1. ||d_k|| = 1.8, q(x) = -4.05, α=1.667
Rama 2.

Rama 1. ||d_k|| = 1.6, q(x) = -6.45, α=0.5k = 1
Rama 2. j = 2, μ=0.0

Concluyó método del conjunto activo en 2 iteraciones
El punto de paro fue:
2-element Vector{Float64}:
 1.4
 1.7
```

Obtenemos que el punto Karush-Kuhn-Tucker (KKT) es:

$$(\vec{x}^\star, \vec{\mu}^\star) = \left(\begin{bmatrix} 1.4 \\ 1.7 \end{bmatrix}, \begin{bmatrix} 0.399 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \right),$$

y se alcanza en dos iteraciones. El valor de la función objetivo en el óptimo es -1.60.

2.2 Problema 2: Klee-Minty

$$\begin{array}{ll} \min & \frac{1}{2}\vec{x}^\top G\vec{x} - \sum_{i=1}^{n} x_i \\ \text{sujeto a} & x_1 \leq 1 \\ & 2\sum_{j=1}^{i-1} x_j + x_i \leq 2^i - 1 \quad , i = 2, \dots, n \\ x_1, \dots, x_n \geq 0 & \end{array}$$

Sea $n = 15$. Aplica el método empezando con W_0 un subconjunto aleatorio de 5 entradas de los últimos 10 restricciones de positividad.

Para encontrar W_0 que cumple las restricciones impuestas usamos el siguiente comando:

```julia
W_0 = [falses(length(b)-10); rand((false, true), 10)]
# Ahora hay que asegurar que son exactamente 5
Δ = sum(W_0) - 5
if Δ < 0
    # Faltan restricciones
    candidates = findall(W_0 .== false)
    # Eligiendo solo los que están entre las 10 restricciones de positividad
    candidates = candidates[candidates .>= length(b) - 10]
    selected = rand(candidates, abs(Δ))

    W_0[selected] .= trues(abs(Δ))
elseif Δ > 0
    # Hay que quitar
    candidates = findall(W_0 .== true)
    selected = rand(candidates, abs(Δ))

    W_0[selected] .= falses(abs(Δ))
end
```

La selección es aleatoria, entonces la W_0 que presentamos está sujeta a cambios, pero la presentamos con fines ilustrativos de cualquer manera.

```
julia>
Bool[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 1, 1]
```

Para este problema, nuestro algoritmo imprime lo siguiente:

```
julia> or = activeSetMethod(G, c, A, b, n_eq, W_0)
α = 0.0, j = 16

Rama 1. ||d_k|| = 10000.0, q(x) = 0.0, α=0.0k = 0
α = 0.0, j = 17

Rama 1. ||d_k|| = 10000.0, q(x) = 0.0, α=0.0k = 1
α = 0.0, j = 18

Rama 1. ||d_k|| = 10000.0, q(x) = 0.0, α=0.0k = 2
α = 0.0, j = 19

Rama 1. ||d_k|| = 10000.0, q(x) = 0.0, α=0.0k = 3
α = 0.0, j = 20

Rama 1. ||d_k|| = 10000.0, q(x) = 0.0, α=0.0k = 4
α = 0.0, j = 22

Rama 1. ||d_k|| = 10000.0, q(x) = 0.0, α=0.0k = 5
α = 0.0, j = 24

Rama 1. ||d_k|| = 10000.0, q(x) = 0.0, α=0.0k = 6
α = 0.0, j = 26

Rama 1. ||d_k|| = 10000.0, q(x) = 0.0, α=0.0k = 7
α = 0.0, j = 27

Rama 1. ||d_k|| = 10000.0, q(x) = 0.0, α=0.0k = 8
α = 0.0, j = 30

Rama 1. ||d_k|| = 10000.0, q(x) = 0.0, α=0.0k = 9
Rama 2. j = 1, μ=0.0

Concluyó método del conjunto activo en 10 iteraciones
El punto de paro fue:
```

```
15-element Vector{Float64}:
 0.0
 0.0
 0.0
 0.0
 0.0
 0.0
 0.0
 0.0
 0.0
 0.0
 0.0
 0.0
 0.0
 0.0
 0.0
```

El óptimo se obtiene en 10 iteraciones y el punto KKT es:

$$(\vec{x}^\star, \vec{\mu}^\star) = \left(\begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 1 \\ 1 \end{bmatrix} \right)$$

.

La representación completa del vector $\vec{\mu}$ es

```
julia> print(or.μ)
[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0]
```

.

El valor óptimo de la función objetivo en $\vec{x}_\star$ es 0.0.

Para este problema la rutina `quadprog` de Matlab imprime:

```
>> quadprog(G,c,A,b)

Minimum found that satisfies the constraints.

Optimization completed because the objective function is non-decreasing in
feasible directions, to within the value of the optimality tolerance,
and constraints are satisfied to within the value of the constraint tolerance.



ans =

   1.0e-10 *

    0.2220
    0.1175
    0.0685
    0.0744
    0.1014
```

Nuestra solución coincide con la de Matlab salvo error de redondeo.

2.3 Problema 3: Problema Woodinfe

$$\begin{array}{ll} \text{minimizar} & \frac{1}{2}\vec{x}^\top G\vec{x} + \vec{c}^\top\vec{x} \\ \text{sujeto a} & A\vec{x} = \vec{b} \\ & x_i \geq \ell_i \quad \text{si } \ell_i \text{ es finito,} \\ & x_i \leq u_i \quad \text{si } u_i \text{ es finito.} \end{array}$$

- Definimos $J \subset I$ como sigue:
  - Para $x_j \geq \ell_j$ definimos $\left|g_j\left(x_0\right)\right| \leq 8\varepsilon_m \max\left\{\left|\ell_j\right|, 1\right\} \Longrightarrow j \in J$
  - Para $x_j \leq u_j$ definimos $\left|g_j\left(x_0\right)\right| \leq 8\varepsilon_m \max\left\{\left|u_j\right|, 1\right\} \Longrightarrow j \in J$

Donde ε_m es el épsilon de la máquina `eps(Float64)`.

A continuación presentamos el código que se utilizó para construir el problema y encontrar las restricciones que pertenecen a J. El código se presenta recortado (excluímos el código para obtener los datos del archivo .mat y comentarios aclaratorios) en interés de la brevedad, pero se puede encontarar código completo en el script incluído `script3.ipynb`.

```
n_eq = length(b)
A_eq = problem["A"]
G = I(length(c))

A = [A_eq; -I(length(l))]
b = [b; -l]

mask = isfinite.(u)
A = [A; I(length(u))[mask, :]]
# Igual cambiando b para que dimensiones coincidan
b = [b; u[mask]]
b = b[:]

x_0 = linprog(A, b, n_eq)εₘ

  = eps(Float64)

J_l = abs.(l - x_0) .<= 8 * εₘ *  max.(abs.(l), ones(length(l)))

J_u = abs.(x_0 - u) .<= 8 * εₘ *  max.(abs.(u), ones(length(u)))
J_u = J_u[isfinite.(u)]

J = [J_l; J_u]

# Concatenando con un índice aleatorio
R_index = rand(findall(J), 1)
notJ = falses(size(A, 1) - n_eq)
notJ[R_index] .= true
W_0 = [trues(n_eq); notJ]
```

De el cómputo anterior obtenemos W_0 que documentamos a continuación a manera de ejemplo puesto que dada la naturaleza aleatoria del índice, puede cambiar.

```
julia> println(W_0)
Bool[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0,
      0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
     0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1,
      0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
```

Corriendo el algoritmo con las especificaciones dadas obtenemos el output que se imprime a continuación:

```
julia> or = activeSetMethod(G, c, A, b, n_eq, copy(W_0))
α = -0.0, j = 38

Rama 1. ||d_k|| = 158.4, q(x) = 90250.0, α=-0.0k = 0
α = -0.0, j = 39

Rama 1. ||d_k|| = 158.6, q(x) = 90250.0, α=-0.0k = 1
α = -0.0, j = 40

Rama 1. ||d_k|| = 159.3, q(x) = 90250.0, α=-0.0k = 2
α = -0.0, j = 42

Rama 1. ||d_k|| = 160.2, q(x) = 90250.0, α=-0.0k = 3
α = -0.0, j = 43

Rama 1. ||d_k|| = 160.9, q(x) = 90250.0, α=-0.0k = 4
α = -0.0, j = 77

Rama 1. ||d_k|| = 161.7, q(x) = 90250.0, α=-0.0k = 5
α = -0.0, j = 79

Rama 1. ||d_k|| = 161.7, q(x) = 90250.0, α=-0.0k = 6
α = -0.0, j = 85

Rama 1. ||d_k|| = 161.7, q(x) = 90250.0, α=-0.0k = 7
α = -0.0, j = 88

Rama 1. ||d_k|| = 162.0, q(x) = 90250.0, α=-0.0k = 8
α = -0.0, j = 89

Rama 1. ||d_k|| = 162.8, q(x) = 90250.0, α=-0.0k = 9
α = -0.0, j = 90

Rama 1. ||d_k|| = 162.8, q(x) = 90250.0, α=-0.0k = 10
α = -0.0, j = 92

Rama 1. ||d_k|| = 162.8, q(x) = 90250.0, α=-0.0k = 11
α = -0.0, j = 95

Rama 1. ||d_k|| = 162.8, q(x) = 90250.0, α=-0.0k = 12
α = -0.0, j = 97

Rama 1. ||d_k|| = 162.8, q(x) = 90250.0, α=-0.0k = 13
α = -0.0, j = 100

Rama 1. ||d_k|| = 162.8, q(x) = 90250.0, α=-0.0k = 14
α = -0.0, j = 102

Rama 1. ||d_k|| = 162.8, q(x) = 90250.0, α=-0.0k = 15
α = -0.0, j = 105

Rama 1. ||d_k|| = 162.8, q(x) = 90250.0, α=-0.0k = 16
α = -0.0, j = 103

Rama 1. ||d_k|| = 162.9, q(x) = 90250.0, α=-0.0k = 17
α = -0.0, j = 106

Rama 1. ||d_k|| = 162.8, q(x) = 90250.0, α=-0.0k = 18
α = -0.0, j = 108

Rama 1. ||d_k|| = 162.8, q(x) = 90250.0, α=-0.0k = 19
α = -0.0, j = 109
```

```

Rama 1. ||d_k|| = 162.8, q(x) = 90250.0, α=-0.0k = 20
α = -0.0, j = 113

Rama 1. ||d_k|| = 162.6, q(x) = 90250.0, α=-0.0k = 21
α = -0.0, j = 116

Rama 1. ||d_k|| = 162.6, q(x) = 90250.0, α=-0.0k = 22
α = -0.0, j = 118

Rama 1. ||d_k|| = 162.6, q(x) = 90250.0, α=-0.0k = 23
α = -0.0, j = 119

Rama 1. ||d_k|| = 162.6, q(x) = 90250.0, α=-0.0k = 24
α = -0.0, j = 121

Rama 1. ||d_k|| = 162.6, q(x) = 90250.0, α=-0.0k = 25
α = -0.0, j = 123

Rama 1. ||d_k|| = 162.6, q(x) = 90250.0, α=-0.0k = 26
α = 0.0, j = 54

Rama 1. ||d_k|| = 162.6, q(x) = 90250.0, α=0.0k = 27
α = 0.0, j = 55

Rama 1. ||d_k|| = 162.6, q(x) = 90250.0, α=0.0k = 28
α = 0.0, j = 56

Rama 1. ||d_k|| = 132.7, q(x) = 90250.0, α=0.0k = 29
α = 0.0, j = 57

Rama 1. ||d_k|| = 105.2, q(x) = 90250.0, α=0.0k = 30
α = 0.0, j = 58

Rama 1. ||d_k|| = 105.2, q(x) = 90250.0, α=0.0k = 31
α = 0.0, j = 59

Rama 1. ||d_k|| = 105.2, q(x) = 90250.0, α=0.0k = 32
α = 0.0, j = 60

Rama 1. ||d_k|| = 105.2, q(x) = 90250.0, α=0.0k = 33
α = 0.0, j = 61

Rama 1. ||d_k|| = 105.2, q(x) = 90250.0, α=0.0k = 34
α = 0.0, j = 62

Rama 1. ||d_k|| = 105.2, q(x) = 90250.0, α=0.0k = 35
α = 0.0, j = 64

Rama 1. ||d_k|| = 105.2, q(x) = 90250.0, α=0.0k = 36
α = 0.0, j = 65

Rama 1. ||d_k|| = 105.2, q(x) = 90250.0, α=0.0k = 37
α = 0.0, j = 66

Rama 1. ||d_k|| = 105.2, q(x) = 90250.0, α=0.0k = 38
α = 0.0, j = 67

Rama 1. ||d_k|| = 105.2, q(x) = 90250.0, α=0.0k = 39
α = 0.0, j = 68

Rama 1. ||d_k|| = 105.2, q(x) = 90250.0, α=0.0k = 40
```

```
α = 0.0, j = 69

Rama 1. ||d_k|| = 105.2, q(x) = 90250.0, α=0.0k = 41
α = 0.0, j = 70

Rama 1. ||d_k|| = 94.16, q(x) = 90250.0, α=0.0k = 42
α = 0.0, j = 71

Rama 1. ||d_k|| = 87.21, q(x) = 90250.0, α=0.0k = 43
α = 0.0, j = 72

Rama 1. ||d_k|| = 59.18, q(x) = 90250.0, α=0.0k = 44
α = 0.0, j = 125

Rama 1. ||d_k|| = 26.0, q(x) = 90250.0, α=0.0k = 45
α = 0.0, j = 132

Rama 1. ||d_k|| = 15.75, q(x) = 90250.0, α=0.0k = 46
α = 0.0, j = 133

Rama 1. ||d_k|| = 15.75, q(x) = 90250.0, α=0.0k = 47
α = 0.0, j = 136

Rama 1. ||d_k|| = 15.75, q(x) = 90250.0, α=0.0k = 48
α = 0.0, j = 137

Rama 1. ||d_k|| = 15.75, q(x) = 90250.0, α=0.0k = 49
α = 0.0, j = 138

Rama 1. ||d_k|| = 15.75, q(x) = 90250.0, α=0.0k = 50
α = 3.832, j = 75

Rama 1. ||d_k|| = 9.6, q(x) = 90080.0, α=3.832
Rama 2.
α = 0.8197, j = 128

Rama 1. ||d_k|| = 61.0, q(x) = 82880.0, α=0.8197k = 52
Rama 2.
α = 1.274, j = 78

Rama 1. ||d_k|| = 39.25, q(x) = 79790.0, α=1.274
Rama 2.
α = 1.598, j = 127

Rama 1. ||d_k|| = 42.33, q(x) = 76430.0, α=1.598
Rama 2.
α = 0.3265, j = 129

Rama 1. ||d_k|| = 30.62, q(x) = 75480.0, α=0.3265k = 55
Rama 2.
α = 1.869, j = 107

Rama 1. ||d_k|| = 26.75, q(x) = 74050.0, α=1.869
Rama 2.
α = 2.667, j = 38

Rama 1. ||d_k|| = 43.12, q(x) = 72560.0, α=2.667
Rama 2.
α = 2.667, j = 99

Rama 1. ||d_k|| = 31.16, q(x) = 71300.0, α=2.667
Rama 2.
```

```
α = 2.593, j = 36

Rama 1. ||d_k|| = 38.57, q(x) = 70130.0, α=2.593
Rama 2.
α = 3.558, j = 76

Rama 1. ||d_k|| = 19.4, q(x) = 69640.0, α=3.558
Rama 2.
α = 2.483, j = 131

Rama 1. ||d_k|| = 40.27, q(x) = 68390.0, α=2.483
Rama 2.
α = 1.5, j = 112

Rama 1. ||d_k|| = 10.0, q(x) = 68190.0, α=1.5
Rama 2.
α = 3.048, j = 107

Rama 1. ||d_k|| = 15.72, q(x) = 67900.0, α=3.048
Rama 2.
α = 1.875, j = 124

Rama 1. ||d_k|| = 10.67, q(x) = 67730.0, α=1.875
Rama 2.
α = 1.712, j = 134

Rama 1. ||d_k|| = 8.0, q(x) = 67620.0, α=1.712
Rama 2.
α = 3.352, j = 78

Rama 1. ||d_k|| = 12.98, q(x) = 67460.0, α=3.352
Rama 2.
α = 2.526, j = 93

Rama 1. ||d_k|| = 5.937, q(x) = 67410.0, α=2.526
Rama 2.
α = 2.353, j = 96

Rama 1. ||d_k|| = 4.25, q(x) = 67370.0, α=2.353
Rama 2.
α = 1.429, j = 110

Rama 1. ||d_k|| = 3.5, q(x) = 67350.0, α=1.429
Rama 2.
α = 2.27, j = 86

Rama 1. ||d_k|| = 5.727, q(x) = 67300.0, α=2.27
Rama 2.
α = 2.077, j = 117

Rama 1. ||d_k|| = 4.816, q(x) = 67270.0, α=2.077
Rama 2.
α = 3.327, j = 77

Rama 1. ||d_k|| = 5.524, q(x) = 67240.0, α=3.327
Rama 2.
α = 1.595, j = 101

Rama 1. ||d_k|| = 3.135, q(x) = 67220.0, α=1.595
Rama 2.
α = 4.201, j = 104

```

```
Rama 1. ||d_k|| = 1.19, q(x) = 67220.0, α=4.201
Rama 2.
α = 3.165, j = 101

Rama 1. ||d_k|| = 1.279, q(x) = 67220.0, α=3.165
Rama 2.
α = 9.982, j = 113

Rama 1. ||d_k|| = 0.9649, q(x) = 67220.0, α=9.982
Rama 2. j = 1, μ=0.0

Concluyó método del conjunto activo en 77 iteraciones
El punto de paro fue:
89-element Vector{Float64}:
 51.47
 47.0
 49.53
 50.0
  ⋮
  3.734e-13
  6.455
  9.264e-14
  9.333
```

El punto óptimo completo es:

```
julia> print(round.(x_star; sigdigits=4))
[51.47, 47.0, 49.53, 50.0, 10.0, 49.47, 43.0, 49.53, 25.0, 10.0, 35.0, 44.26, 34.46, 20.0, 5.0, 26.27, 30.0, 65.0, 100.0, 100.0, 90.0, 50.0, 10.0, 20.0, 25.0, 10.0, 15.0, 0.0, 50.0, 40.0, 20.0, 5.0, 15.0, 20.0, 25.0, 30.0, 20.0, 0.0, 42.69, 20.72, 33.06, 6.07, 15.19, 2.479, 10.0, 15.0, 43.8, 15.64, 25.32, 33.54, 3.636, 7.87, 12.67, 5.727, 1.279, 39.27, -1.482e-14, 11.28, 9.667, 37.64, 7.182, 7.479, 9.667, 22.14, 2.818, 1.242, 10.67, 15.96, 3.9, 15.33, 1.1, 18.7, 5.956e-14, 0.9649, 4.364, 9.506, 10.33, 4.333, 17.82, 10.71, 15.33, 4.702, 6.364, 6.061, 30.0, 3.734e-13, 6.455, 9.264e-14, 9.333]
```

con un valor óptimo de 108291.05, que se obtuvo en 77 iteraciones.

3 Conclusiones (?)

```
BenchmarkTools.Trial: 42 samples with 1 evaluation.
 Range (min … max):  102.039 ms … 196.765 ms  ┊ GC (min … max):  0.00% … 8.27%
 Time  (median):     120.942 ms               ┊ GC (median):    13.35%
 Time  (mean ± σ):   121.101 ms ±  18.822 ms  ┊ GC (mean ± σ):   8.36% ± 6.81%

 Memory estimate: 45.41 MiB, allocs estimate: 36556.
```

Figura chida

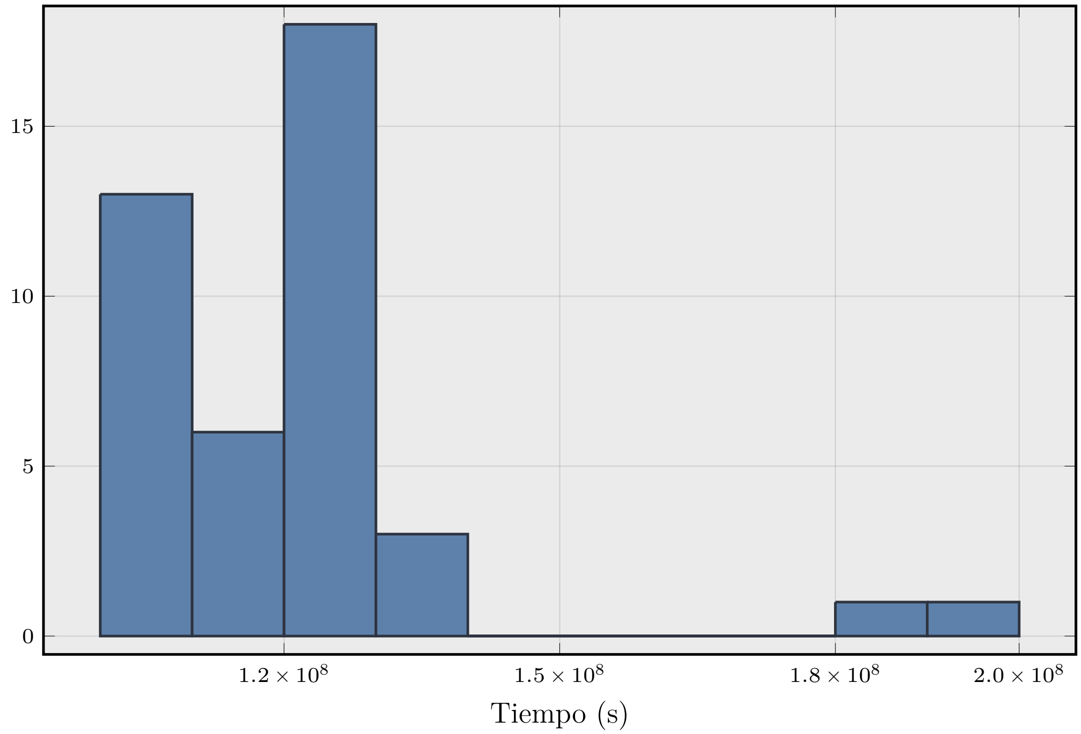

Figura 1: Histograma de frecuencia por tiempo

Referencias

[DHL17] Iain Dunning, Joey Huchette y Miles Lubin. “JuMP: A Modeling Language for Mathematical Optimization”. En: *SIAM Review* 59.2 (2017), págs. 295-320. DOI: 10.1137/15M1020575.