IDO- Tarea 3

Marcelino Sánchez Rodríguez 191654

30/1/24

Problema de la ONU

Enunciado

ALGORITMO: Ramificación y Acotamiento 1. Inicializar la lista de nodos pendientes con el nodo raíz. 2. Mientras la lista de nodos pendientes no esté vacía, hacer: a. Seleccionar y eliminar un nodo de la lista de nodos pendientes. b. Si el nodo seleccionado es prometedor, entonces: i. Si el nodo seleccionado es una solución completa, comparar con la mejor solución actual. ii. De lo contrario, ramificar el nodo para generar nuevos nodos y agregarlos a la lista. 3. Devolver la mejor solución encontrada.

$$\begin{aligned} max \quad & z = 120x_1 + 80x_2 \\ & 2x_1 + x_2 + x_3 = 6 \\ & 7x_1 + 8x_2 + x_4 \leq 28 \\ & x_3 - .5x_4 = 2.5 \\ & x_i \in Z_+ \quad \text{para todo } i \end{aligned}$$

Modelo en julia

Por método de ramificación y acotamiento se tiene el siguiente modelo en julia

Primero realizamos un simplex normal:

```julia
using JuMP, HiGHS
model = Model(HiGHS.Optimizer)
@variable(model, x[1:4] >= 0)

@objective(model, Max, 120x[1] + 80x[2])

@constraint(model, 2*x[1] + x[2] + x[3] == 6)
@constraint(model, 7*x[1] + 8*x[2] + x[4] <= 28)
@constraint(model, x[3] - 0.5*x[4] == 2.5)

optimization_result = optimize!(model)
```

```
Running HiGHS 1.6.0: Copyright (c) 2023 HiGHS under MIT licence terms
Presolving model
2 rows, 3 cols, 6 nonzeros
2 rows, 3 cols, 6 nonzeros
```

```
Presolve : Reductions: rows 2(-1); columns 3(-1); elements 6(-2)
Solving the presolved LP
Using EKK dual simplex solver - serial
  Iteration        Objective     Infeasibilities num(sum)
          0    -4.9998787442e+01 Ph1: 2(3.375); Du: 2(49.9988) 0s
          2    -2.8000000000e+02 Pr: 0(0) 0s
Solving the original LP from the solution after postsolve
Model   status      : Optimal
Simplex   iterations: 2
Objective value     :  2.8000000000e+02
HiGHS run time      :          0.00
```

El resultado es:

```
value.(x)
```

```
4-element Vector{Float64}:
 -0.0
  3.5
  2.5
  0.0
```

Ramificación 1

Con lo cual decidimos ramificar en la variable x_3, dado que tuvimos soluciones fraccionadas:

Nodo 1

```
model = Model(HiGHS.Optimizer)
@variable(model, x[1:4] >= 0)

@objective(model, Max, 120x[1] + 80x[2])

@constraint(model, 2*x[1] + x[2] + x[3] == 6)
@constraint(model, 7*x[1] + 8*x[2] + x[4] <= 28)
@constraint(model, x[3] - 0.5*x[4] == 2.5)
@constraint(model, x[3]  >= 3)

optimization_result = optimize!(model)
```

```
Running HiGHS 1.6.0: Copyright (c) 2023 HiGHS under MIT licence terms
Presolving model
2 rows, 3 cols, 6 nonzeros
2 rows, 3 cols, 6 nonzeros
Presolve : Reductions: rows 2(-2); columns 3(-1); elements 6(-3)
Solving the presolved LP
Using EKK dual simplex solver - serial
  Iteration        Objective     Infeasibilities num(sum)
          0    -4.9998787442e+01 Ph1: 2(3.375); Du: 2(49.9988) 0s
          3    -2.4000000000e+02 Pr: 0(0) 0s
Solving the original LP from the solution after postsolve
Model   status      : Optimal
Simplex   iterations: 3
Objective value     :  2.4000000000e+02
HiGHS run time      :          0.00
```

```
value.(x)
```

```
4-element Vector{Float64}:
 0.0
 3.0
 3.0
 1.0
```

Nodo 2

```
model = Model(HiGHS.Optimizer)
@variable(model, x[1:4] >= 0)

@objective(model, Max, 120x[1] + 80x[2])

@constraint(model, 2*x[1] + x[2] + x[3] == 6)
@constraint(model, 7*x[1] + 8*x[2] + x[4] <= 28)
@constraint(model, x[3] - 0.5*x[4] == 2.5)
@constraint(model, x[3]  <= 2)

optimization_result = optimize!(model)
```

```
Running HiGHS 1.6.0: Copyright (c) 2023 HiGHS under MIT licence terms
Presolving model
```

```
Problem status detected on presolve: Infeasible
Model   status      : Infeasible
Objective value     :  0.0000000000e+00
HiGHS run time      :          0.00
ERROR:   No LP invertible representation for getDualRay
```

Solución

Por lo tanto la solución es la del nodo 1, porque es la única que tiene solución entera de los dos espacios posibles, y sabiendo que en uno es infactible.

Verdadera solución

De hecho la solución final del problema es:

```julia
using JuMP, HiGHS
model = Model(HiGHS.Optimizer)
@variable(model, x[1:4] >= 0, Int)

@objective(model, Max, 120x[1] + 80x[2])

@constraint(model, 2*x[1] + x[2] + x[3] == 6)
@constraint(model, 7*x[1] + 8*x[2] + x[4] <= 28)
@constraint(model, x[3] - 0.5*x[4] == 2.5)

optimization_result = optimize!(model)
value.(x)
```

```
Running HiGHS 1.6.0: Copyright (c) 2023 HiGHS under MIT licence terms
Presolving model
2 rows, 3 cols, 6 nonzeros
0 rows, 1 cols, 0 nonzeros
0 rows, 0 cols, 0 nonzeros
Presolve: Optimal

Solving report
  Status            Optimal
  Primal bound      240
  Dual bound        240
  Gap               0% (tolerance: 0.01%)
  Solution status   feasible
                    240 (objective)
                    0 (bound viol.)
                    0 (int. viol.)
                    0 (row viol.)
  Timing            0.00 (total)
                    0.00 (presolve)
                    0.00 (postsolve)
  Nodes             0
  LP iterations     0 (total)
                    0 (strong br.)
                    0 (separation)
                    0 (heuristics)

4-element Vector{Float64}:
 0.0
 3.0
 3.0
 1.0
```