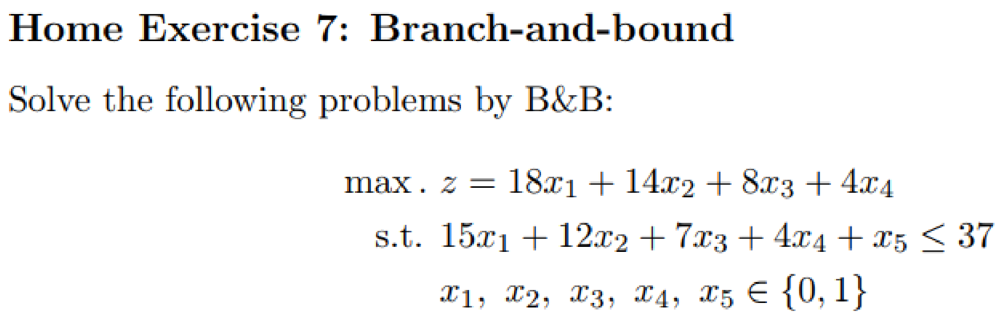


Home Exercise 7: Branch-and-bound

Solve the following problems by B&B:

$$\begin{aligned} \max.\ & z = 18x_1 + 14x_2 + 8x_3 + 4x_4 \\ \text{s.t.}\ & 15x_1 + 12x_2 + 7x_3 + 4x_4 + x_5 \leq 37 \\ & x_1,\ x_2,\ x_3,\ x_4,\ x_5 \in \{0, 1\} \end{aligned}$$

Instead of formulating the problem as binary, we can add the constraints x[1:5] >= 0 and x[1:5] <= 1 as LP-relaxation at the start of the problem.

The problem is now reformulated as below:

max . z = 18x1 + 14x2 + 8x3 + 4x4
 s.t. 15x1 + 12x2 + 7x3 + 4x4 + x5 ≤ 37
  x1, x2, x3, x4, x5 >= 0
  x1, x2, x3, x4, x5 <= 1

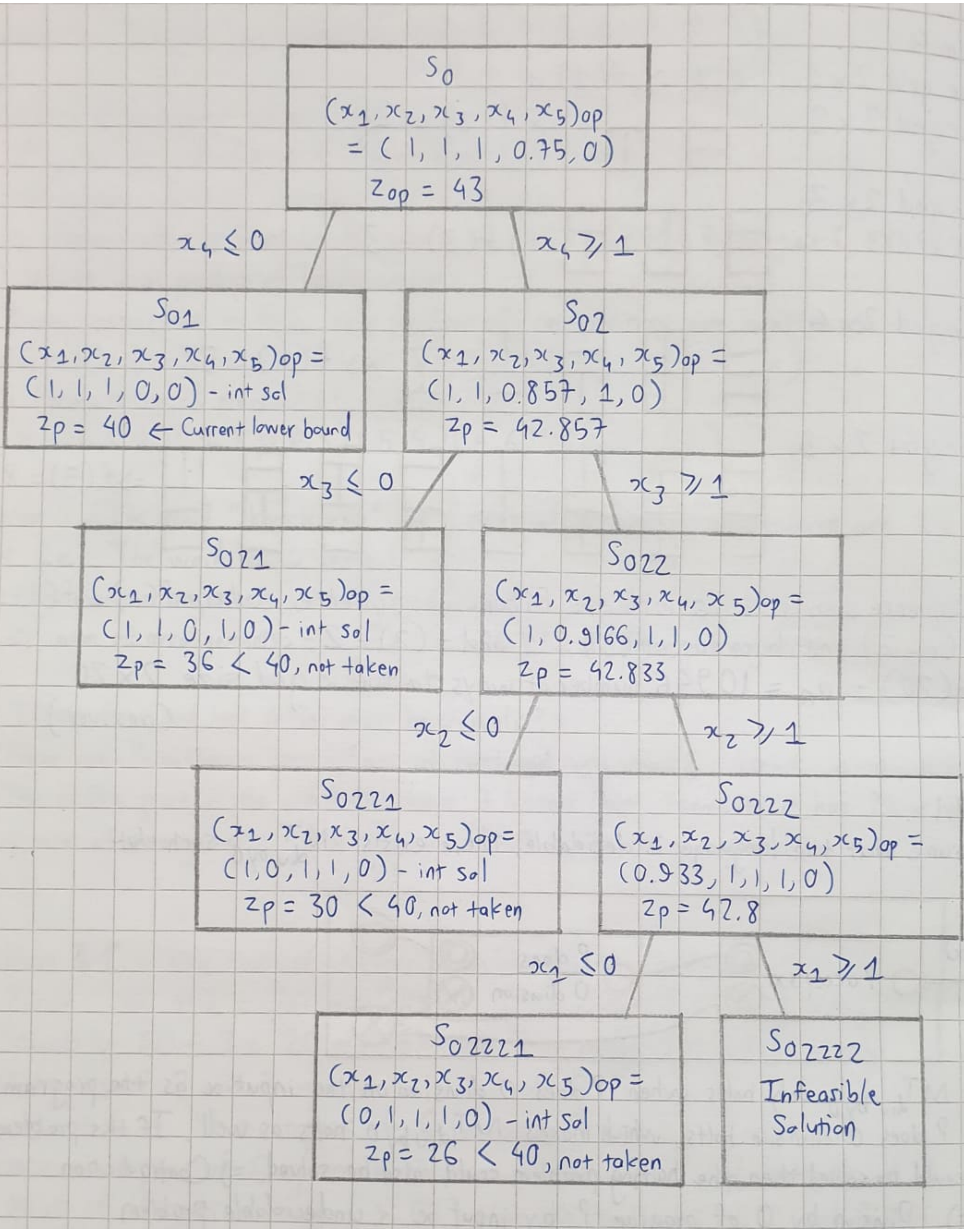
Now we will apply the Branch-and-Bound algorithm on this problem and regard x[1:5] as integers instead of binary. The Branch-and-Bound tree produces the final result as:

$[x1, x2, x3, x4, x5]_{optimal}$ = [1, 1, 1, 0, 0]

$z_{optimal}$ = 40

For calculation of optimization for each subproblem, I use Julia to solve and the code is attached at the end of this report

The step-by-step solution by the algorithm is illustrated next page



S_0

$(x_1, x_2, x_3, x_4, x_5)op = (1, 1, 1, 0.75, 0)$

$Z_{op} = 43$

$x_4 \leq 0$

$x_4 \geq 1$

S_{01}

$(x_1, x_2, x_3, x_4, x_5)op = (1, 1, 1, 0, 0)$ - int sol

$Zp = 40$ ← Current lower bound

S_{02}

$(x_1, x_2, x_3, x_4, x_5)op = (1, 1, 0.857, 1, 0)$

$Zp = 42.857$

$x_3 \leq 0$

$x_3 \geq 1$

S_{021}

$(x_1, x_2, x_3, x_4, x_5)op = (1, 1, 0, 1, 0)$ - int sol

$Zp = 36 < 40$, not taken

S_{022}

$(x_1, x_2, x_3, x_4, x_5)op = (1, 0.9166, 1, 1, 0)$

$Zp = 42.833$

$x_2 \leq 0$

$x_2 \geq 1$

S_{0221}

$(x_1, x_2, x_3, x_4, x_5)op = (1, 0, 1, 1, 0)$ - int sol

$Zp = 30 < 40$, not taken

S_{0222}

$(x_1, x_2, x_3, x_4, x_5)op = (0.933, 1, 1, 1, 0)$

$Zp = 42.8$

$x_1 \leq 0$

$x_1 \geq 1$

S_{02221}

$(x_1, x_2, x_3, x_4, x_5)op = (0, 1, 1, 1, 0)$ - int sol

$Zp = 26 < 40$, not taken

S_{02222}

Infeasible Solution

Home Exercise 7

March 26, 2022

[1]:

```
using JuMP, Cbc # modelling language and solver
d1 = Model(with_optimizer(Cbc.Optimizer,logLevel = 0)) # create the model, select the solver
@variable(d1, x[1:5], Bin) #creates the non-negative variables x1 and x2
@constraint(d1, 15*x[1] + 12*x[2] + 7*x[3] + 4*x[4] + x[5] <= 37) # constraint 1
@objective(d1, Max, 18*x[1] + 14*x[2] + 8*x[3] + 4*x[4]) # declare the objective function
optimize!(d1) # solve the optimization problem

# printing out the solution
x_value = value.(x)
print("Optimal values: $(x_value), \nOptimal objective: $(objective_value(d1))\n")
```

```
Optimal values: [1.0, 1.0, 1.0, 0.0, 0.0],
Optimal objective: 40.0
```

[4]:

```
using JuMP, Cbc # modelling language and solver
d1 = Model(with_optimizer(Cbc.Optimizer,logLevel = 0)) # create the model, select the solver
@variable(d1, x[1:5] >= 0)
@constraint(d1, x[1] <= 1)
@constraint(d1, x[2] <= 1)
@constraint(d1, x[3] <= 1)
@constraint(d1, x[4] <= 1)
@constraint(d1, x[5] <= 1)
@constraint(d1, 15*x[1] + 12*x[2] + 7*x[3] + 4*x[4] + x[5] <= 37) # constraint 1
@objective(d1, Max, 18*x[1] + 14*x[2] + 8*x[3] + 4*x[4]) # declare the objective function
optimize!(d1) # solve the optimization problem

# printing out the solution
x_value = value.(x)
print("Optimal values: $(x_value), \nOptimal objective: $(objective_value(d1))\n")
```

```
Optimal values: [1.0, 1.0, 1.0, 0.7500000000000007, 0.0],
Optimal objective: 43.0
```

```
Presolve 1 (-5) rows, 4 (-1) columns and 4 (-6) elements
0  Obj -0 Dual inf 67.642853 (4)
1  Obj 43
Optimal - objective value 43
After Postsolve, objective 43, infeasibilities - dual 0 (0), primal 0 (0)
Optimal objective 43 - 1 iterations time 0.002, Presolve 0.00
```

[5]:

```
using JuMP, Cbc # modelling language and solver
d1 = Model(with_optimizer(Cbc.Optimizer,logLevel = 0)) # create the model,␣
 ↪select the solver
@variable(d1, x[1:5] >= 0)
@constraint(d1, x[1] <= 1)
@constraint(d1, x[2] <= 1)
@constraint(d1, x[3] <= 1)
@constraint(d1, x[4] <= 1)
@constraint(d1, x[5] <= 1)
@constraint(d1, 15*x[1] + 12*x[2] + 7*x[3] + 4*x[4] + x[5] <= 37)
@constraint(d1, x[4] <= 0)
@objective(d1, Max, 18*x[1] + 14*x[2] + 8*x[3] + 4*x[4]) # declare the␣
 ↪objective function
optimize!(d1) # solve the optimization problem

# printing out the solution
x_value = value.(x)
print("Optimal values: $(x_value), \nOptimal objective:␣
 ↪$(objective_value(d1))\n")
```

```
Optimal values: [1.0, 1.0, 1.0, 0.0, 0.0],
Optimal objective: 40.0
Presolve 0 (-7) rows, 0 (-5) columns and 0 (-11) elements
Optimal - objective value 40
After Postsolve, objective 40, infeasibilities - dual 0 (0), primal 0 (0)
Optimal objective 40 - 0 iterations time 0.002, Presolve 0.00
```

[6]:

```
using JuMP, Cbc # modelling language and solver
d1 = Model(with_optimizer(Cbc.Optimizer,logLevel = 0)) # create the model,␣
 ↪select the solver
@variable(d1, x[1:5] >= 0)
@constraint(d1, x[1] <= 1)
@constraint(d1, x[2] <= 1)
@constraint(d1, x[3] <= 1)
@constraint(d1, x[4] <= 1)
@constraint(d1, x[5] <= 1)
@constraint(d1, 15*x[1] + 12*x[2] + 7*x[3] + 4*x[4] + x[5] <= 37)
@constraint(d1, x[4] >= 1)
@objective(d1, Max, 18*x[1] + 14*x[2] + 8*x[3] + 4*x[4]) # declare the␣
 ↪objective function
```

```
optimize!(d1) # solve the optimization problem

# printing out the solution
x_value = value.(x)
print("Optimal values: $(x_value), \nOptimal objective:␣
 ↪$(objective_value(d1))\n")
```

```
Optimal values: [1.0, 1.0, 0.8571428571428574, 1.0, 0.0],
Optimal objective: 42.85714285714286
Presolve 1 (-6) rows, 3 (-2) columns and 3 (-8) elements
0  Obj 4 Dual inf 52.642854 (3)
1  Obj 42.857143
Optimal - objective value 42.857143
After Postsolve, objective 42.857143, infeasibilities - dual 0 (0), primal 0 (0)
Optimal objective 42.85714286 - 1 iterations time 0.002, Presolve 0.00
```

[7]:

```
using JuMP, Cbc # modelling language and solver
d1 = Model(with_optimizer(Cbc.Optimizer,logLevel = 0)) # create the model,␣
 ↪select the solver
@variable(d1, x[1:5] >= 0)
@constraint(d1, x[1] <= 1)
@constraint(d1, x[2] <= 1)
@constraint(d1, x[3] <= 1)
@constraint(d1, x[4] <= 1)
@constraint(d1, x[5] <= 1)
@constraint(d1, 15*x[1] + 12*x[2] + 7*x[3] + 4*x[4] + x[5] <= 37)
@constraint(d1, x[4] >= 1)
@constraint(d1, x[3] <= 0)
@objective(d1, Max, 18*x[1] + 14*x[2] + 8*x[3] + 4*x[4]) # declare the␣
 ↪objective function
optimize!(d1) # solve the optimization problem

# printing out the solution
x_value = value.(x)
print("Optimal values: $(x_value), \nOptimal objective:␣
 ↪$(objective_value(d1))\n")
```

```
Optimal values: [1.0, 1.0, 0.0, 1.0, 0.0],
Optimal objective: 36.0
Presolve 0 (-8) rows, 0 (-5) columns and 0 (-12) elements
Optimal - objective value 36
After Postsolve, objective 36, infeasibilities - dual 0 (0), primal 0 (0)
Optimal objective 36 - 0 iterations time 0.002, Presolve 0.00
```

[8]:

```
using JuMP, Cbc # modelling language and solver
d1 = Model(with_optimizer(Cbc.Optimizer,logLevel = 0)) # create the model,␣
 ↪select the solver
```

```
@variable(d1, x[1:5] >= 0)
@constraint(d1, x[1] <= 1)
@constraint(d1, x[2] <= 1)
@constraint(d1, x[3] <= 1)
@constraint(d1, x[4] <= 1)
@constraint(d1, x[5] <= 1)
@constraint(d1, 15*x[1] + 12*x[2] + 7*x[3] + 4*x[4] + x[5] <= 37)
@constraint(d1, x[4] >= 1)
@constraint(d1, x[3] >= 1)
@objective(d1, Max, 18*x[1] + 14*x[2] + 8*x[3] + 4*x[4]) # declare the␣
 ↪objective function
optimize!(d1) # solve the optimization problem

# printing out the solution
x_value = value.(x)
print("Optimal values: $(x_value), \nOptimal objective:␣
 ↪$(objective_value(d1))\n")
```

```
Optimal values: [1.0, 0.9166666666666667, 1.0, 1.0, 0.0],
Optimal objective: 42.833333333333336
Presolve 1 (-7) rows, 2 (-3) columns and 2 (-10) elements
0  Obj 12 Dual inf 35.499998 (2)
1  Obj 42.833333
Optimal - objective value 42.833333
After Postsolve, objective 42.833333, infeasibilities - dual 0 (0), primal 0 (0)
Optimal objective 42.83333333 - 1 iterations time 0.002, Presolve 0.00
```

[9]:

```
using JuMP, Cbc # modelling language and solver
d1 = Model(with_optimizer(Cbc.Optimizer,logLevel = 0)) # create the model,␣
 ↪select the solver
@variable(d1, x[1:5] >= 0)
@constraint(d1, x[1] <= 1)
@constraint(d1, x[2] <= 1)
@constraint(d1, x[3] <= 1)
@constraint(d1, x[4] <= 1)
@constraint(d1, x[5] <= 1)
@constraint(d1, 15*x[1] + 12*x[2] + 7*x[3] + 4*x[4] + x[5] <= 37)
@constraint(d1, x[4] >= 1)
@constraint(d1, x[3] >= 1)
@constraint(d1, x[2] <= 0)
@objective(d1, Max, 18*x[1] + 14*x[2] + 8*x[3] + 4*x[4]) # declare the␣
 ↪objective function
optimize!(d1) # solve the optimization problem

# printing out the solution
x_value = value.(x)
```

```
print("Optimal values: $(x_value), \nOptimal objective:␣
 ↪$(objective_value(d1))\n")
```

```
Optimal values: [1.0, 0.0, 1.0, 1.0, 0.0],
Optimal objective: 30.0
Presolve 0 (-9) rows, 0 (-5) columns and 0 (-13) elements
Optimal - objective value 30
After Postsolve, objective 30, infeasibilities - dual 0 (0), primal 0 (0)
Optimal objective 30 - 0 iterations time 0.002, Presolve 0.00
```

[10]:

```
using JuMP, Cbc # modelling language and solver
d1 = Model(with_optimizer(Cbc.Optimizer,logLevel = 0)) # create the model,␣
 ↪select the solver
@variable(d1, x[1:5] >= 0)
@constraint(d1, x[1] <= 1)
@constraint(d1, x[2] <= 1)
@constraint(d1, x[3] <= 1)
@constraint(d1, x[4] <= 1)
@constraint(d1, x[5] <= 1)
@constraint(d1, 15*x[1] + 12*x[2] + 7*x[3] + 4*x[4] + x[5] <= 37)
@constraint(d1, x[4] >= 1)
@constraint(d1, x[3] >= 1)
@constraint(d1, x[2] >= 1)
@objective(d1, Max, 18*x[1] + 14*x[2] + 8*x[3] + 4*x[4]) # declare the␣
 ↪objective function
optimize!(d1) # solve the optimization problem

# printing out the solution
x_value = value.(x)
print("Optimal values: $(x_value), \nOptimal objective:␣
 ↪$(objective_value(d1))\n")
```

```
Optimal values: [0.9333333333333333, 1.0, 1.0, 1.0, 0.0],
Optimal objective: 42.8
Presolve 0 (-9) rows, 0 (-5) columns and 0 (-13) elements
Optimal - objective value 42.8
After Postsolve, objective 42.8, infeasibilities - dual 0 (0), primal 0 (0)
Optimal objective 42.8 - 0 iterations time 0.002, Presolve 0.00
```

[11]:

```
using JuMP, Cbc # modelling language and solver
d1 = Model(with_optimizer(Cbc.Optimizer,logLevel = 0)) # create the model,␣
 ↪select the solver
@variable(d1, x[1:5] >= 0)
@constraint(d1, x[1] <= 1)
@constraint(d1, x[2] <= 1)
@constraint(d1, x[3] <= 1)
@constraint(d1, x[4] <= 1)
```

```
@constraint(d1, x[5] <= 1)
@constraint(d1, 15*x[1] + 12*x[2] + 7*x[3] + 4*x[4] + x[5] <= 37)
@constraint(d1, x[4] >= 1)
@constraint(d1, x[3] >= 1)
@constraint(d1, x[2] >= 1)
@constraint(d1, x[1] <= 0)
@objective(d1, Max, 18*x[1] + 14*x[2] + 8*x[3] + 4*x[4]) # declare the␣
 ↪objective function
optimize!(d1) # solve the optimization problem

# printing out the solution
x_value = value.(x)
print("Optimal values: $(x_value), \nOptimal objective:␣
 ↪$(objective_value(d1))\n")
```

```
Optimal values: [0.0, 1.0, 1.0, 1.0, 0.0],
Optimal objective: 26.0
Presolve 0 (-10) rows, 0 (-5) columns and 0 (-14) elements
Optimal - objective value 26
After Postsolve, objective 26, infeasibilities - dual 0 (0), primal 0 (0)
Optimal objective 26 - 0 iterations time 0.002, Presolve 0.00
```

[12]:

```
using JuMP, Cbc # modelling language and solver
d1 = Model(with_optimizer(Cbc.Optimizer,logLevel = 0)) # create the model,␣
 ↪select the solver
@variable(d1, x[1:5] >= 0)
@constraint(d1, x[1] <= 1)
@constraint(d1, x[2] <= 1)
@constraint(d1, x[3] <= 1)
@constraint(d1, x[4] <= 1)
@constraint(d1, x[5] <= 1)
@constraint(d1, 15*x[1] + 12*x[2] + 7*x[3] + 4*x[4] + x[5] <= 37)
@constraint(d1, x[4] >= 1)
@constraint(d1, x[3] >= 1)
@constraint(d1, x[2] >= 1)
@constraint(d1, x[1] >= 1)
@objective(d1, Max, 18*x[1] + 14*x[2] + 8*x[3] + 4*x[4]) # declare the␣
 ↪objective function
optimize!(d1) # solve the optimization problem

# printing out the solution
x_value = value.(x)
print("Optimal values: $(x_value), \nOptimal objective:␣
 ↪$(objective_value(d1))\n")
```

```
Presolve determined that the problem was infeasible with tolerance of 1e-08
Analysis indicates model infeasible or unbounded
```

```
0  Obj -0 Primal inf 6.0888711 (4) Dual inf 17.46531 (4)
5  Obj 42.8 Primal inf 0.13119921 (1)
Primal infeasible - objective value 42.8
PrimalInfeasible objective 42.8 - 5 iterations time 0.002
```

```
Result index of attribute MathOptInterface.VariablePrimal(1) out of bounds.␣
 ↪There are currently 0 solution(s) in the model.

Stacktrace:
  [1] check_result_index_bounds
    @ /opt/julia/packages/MathOptInterface/YDdD3/src/attributes.jl:139 [inlined]
  [2] get(model::Cbc.Optimizer, attr::MathOptInterface.VariablePrimal, x::
 ↪MathOptInterface.VariableIndex)
    @ Cbc /opt/julia/packages/Cbc/vMMGG/src/MOI_wrapper/MOI_wrapper.jl:797
  [3] get(model::MathOptInterface.Utilities.CachingOptimizer{Cbc.Optimizer,␣
 ↪MathOptInterface.Utilities.UniversalFallback{MathOptInterface.Utilities.
 ↪GenericModel{Float64, MathOptInterface.Utilities.
 ↪ModelFunctionConstraints{Float64}}}}, attr::MathOptInterface.VariablePrimal,␣
 ↪index::MathOptInterface.VariableIndex)
    @ MathOptInterface.Utilities /opt/julia/packages/MathOptInterface/YDdD3/src/
 ↪Utilities/cachingoptimizer.jl:757
  [4] get(b::MathOptInterface.Bridges.LazyBridgeOptimizer{MathOptInterface.
 ↪Utilities.CachingOptimizer{Cbc.Optimizer, MathOptInterface.Utilities.
 ↪UniversalFallback{MathOptInterface.Utilities.GenericModel{Float64,␣
 ↪MathOptInterface.Utilities.ModelFunctionConstraints{Float64}}}}}, attr::
 ↪MathOptInterface.VariablePrimal, index::MathOptInterface.VariableIndex)
    @ MathOptInterface.Bridges /opt/julia/packages/MathOptInterface/YDdD3/src/
 ↪Bridges/bridge_optimizer.jl:1039
  [5] get(model::MathOptInterface.Utilities.CachingOptimizer{MathOptInterface.
 ↪AbstractOptimizer, MathOptInterface.Utilities.
 ↪UniversalFallback{MathOptInterface.Utilities.GenericModel{Float64,␣
 ↪MathOptInterface.Utilities.ModelFunctionConstraints{Float64}}}}, attr::
 ↪MathOptInterface.VariablePrimal, index::MathOptInterface.VariableIndex)
    @ MathOptInterface.Utilities /opt/julia/packages/MathOptInterface/YDdD3/src/
 ↪Utilities/cachingoptimizer.jl:757
  [6] _moi_get_result(::MathOptInterface.Utilities.
 ↪CachingOptimizer{MathOptInterface.AbstractOptimizer, MathOptInterface.
 ↪Utilities.UniversalFallback{MathOptInterface.Utilities.GenericModel{Float64,␣
 ↪MathOptInterface.Utilities.ModelFunctionConstraints{Float64}}}}, ::
 ↪MathOptInterface.VariablePrimal, ::Vararg{Any, N} where N)
    @ JuMP /opt/julia/packages/JuMP/b3hGi/src/JuMP.jl:1199
  [7] get(model::Model, attr::MathOptInterface.VariablePrimal, v::VariableRef)
    @ JuMP /opt/julia/packages/JuMP/b3hGi/src/JuMP.jl:1232
  [8] value(v::VariableRef; result::Int64)
    @ JuMP /opt/julia/packages/JuMP/b3hGi/src/variables.jl:943
  [9] value
    @ /opt/julia/packages/JuMP/b3hGi/src/variables.jl:943 [inlined]
 [10] _broadcast_getindex_evalf
    @ ./broadcast.jl:648 [inlined]
 [11] _broadcast_getindex
    @ ./broadcast.jl:621 [inlined]
```

```
[12] getindex
   @ ./broadcast.jl:575 [inlined]
[13] macro expansion
   @ ./broadcast.jl:984 [inlined]
[14] macro expansion
   @ ./simdloop.jl:77 [inlined]
[15] copyto!
   @ ./broadcast.jl:983 [inlined]
[16] copyto!
   @ ./broadcast.jl:936 [inlined]
[17] copy
   @ ./broadcast.jl:908 [inlined]
[18] materialize(bc::Base.Broadcast.Broadcasted{Base.Broadcast.
↪DefaultArrayStyle{1}, Nothing, typeof(value), Tuple{Vector{VariableRef}}})
   @ Base.Broadcast ./broadcast.jl:883
[19] top-level scope
   @ In[12]:18
[20] eval
   @ ./boot.jl:360 [inlined]
[21] include_string(mapexpr::typeof(REPL.softscope), mod::Module, code::String ⌟
↪filename::String)
   @ Base ./loading.jl:1116
```