CS/ECE/ISyE 524 - Spr 2021 - HW 9

Skylar Hou

1. Making Change

How should you make change for 99 cents if the goal is to minimize the total weight of the coins used? You may use any number of each type of coin. Here are the weights of each coin:

Type of Coin	penny	nickel	dime	quarter
Weight	2.500	5.000	2.268	5.670

Print the number of each type of coin used, and the total weight.

In [3]:
```
using PyPlot, JuMP, HiGHS, Ipopt, Gurobi
```

In [8]:
```
# problem data
weights = [ 2.500, 5.000, 2.268, 5.670]
values = [ 1, 5, 10, 25 ]
n = length(weights) # total number of coin types

m = Model(HiGHS.Optimizer)

@variable(m, num[1:n] >= 0, Int)

@constraint(m, sum(num[i]*values[i] for i in 1:n) == 99)
@expression(m, totalweight, sum(num[i]*weights[i] for i in 1:n))

@objective(m, Min, totalweight)

optimize!(m)
println("the number of each type of coin used: ", value.(num))
println("the total weight: ", objective_value(m))
```

```
Running HiGHS 1.4.0 [date: 1970-01-01, git hash: bcf6c0b22]
Copyright (c) 2022 ERGO-Code under MIT licence terms
Presolving model
1 rows, 4 cols, 4 nonzeros
1 rows, 4 cols, 4 nonzeros
Objective function is integral with scale 500

Solving MIP model with:
   1 rows
   4 cols (0 binary, 4 integer, 0 implied int., 0 continuous)
   4 nonzeros

         Nodes      |    B&B Tree     |            Objective Bounds
|  Dynamic Constraints |       Work
     Proc. InQueue |  Leaves   Expl. | BestBound       BestSol              Gap
|   Cuts   InLp Confl. | LpIters     Time

         0       0         0   0.00%   10              inf                  inf
0      0      0         0     0.0s
 T       0       0         0   0.00%   10              31.546            68.30%
0      0      0         0     0.0s

Solving report
  Status            Optimal
  Primal bound      31.546
  Dual bound        31.546
  Gap               0% (tolerance: 0.01%)
  Solution status   feasible
                    31.546 (objective)
                    0 (bound viol.)
                    0 (int. viol.)
                    0 (row viol.)
  Timing            0.00 (total)
                    0.00 (presolve)
                    0.00 (postsolve)
  Nodes             1
  LP iterations     0 (total)
                    0 (strong br.)
                    0 (separation)
                    0 (heuristics)
the number of each type of coin used: [4.0, 0.0, 2.0, 3.0]
the total weight: 31.546
```

2. Comquat Computers

Comquat owns four production plants at which personal computers are produced. Comquat can sell up to 20,000 computers per year at a price of $3,500 per computer. For each plant the production capacity, cost per computer, and fixed cost of operating the plant for a year are given below. Determine how Comquat can maximize its yearly profit from computer production.

Plant	Production capacity	Plant fixed cost ($ Million)	Cost per computer ($)
1	10,000	9	1,000
2	8,000	5	1,700

Plant	Production capacity	Plant fixed cost ($ Million)	Cost per computer ($)
3	9,000	3	2,300
4	6,000	1	2,900

In [15]:
```
using JuMP, HiGHS
m = Model(HiGHS.Optimizer)
set_silent(m)

price     = 3.5e3                       # price at which we can sell a single co
                                        # (regardless of where it is produced)
cap       = 20000                       # maximum computers that can be sold per

capacity  = [10000, 8000, 9000, 6000]   # production capacity for each plant
fixedcost = [9e6, 5e6, 3e6, 1e6]        # fixed cost for each plant
itemcost  = [1e3, 1.7e3, 2.3e3, 2.9e3]  # cost per computer for each factory


# insert your code here, with integer variable vector z and real variable vector
@variable(m, z[1:4], Bin)
@variable(m, x[1:4] >= 0, Int)
@constraint(m, c[i in 1:4], x[i] <= z[i] * capacity[i])
@constraint(m, sum(x[i] for i in 1:4) <= cap)

@expression(m, cost, sum(fixedcost[i]*z[i] + itemcost[i]*x[i] for i in 1:4))
@expression(m, profit, sum(x[i] for i in 1:4) * price)

@objective(m, Max, profit - cost)

optimize!(m)
println("The maximal profit is: ", objective_value(m))
println("Here is a tally of factory number, whether it's open or not, and how ma
value.([1:4 z x])
```

```
The maximal profit is: 2.56e7
Here is a tally of factory number, whether it's open or not, and how many compu
ters it should produce.
```

Out[15]:
```
4×3 Matrix{Float64}:
 1.0  1.0  10000.0
 2.0  1.0   8000.0
 3.0  0.0      0.0
 4.0  1.0   2000.0
```

3. ABC Investments

ABC Inc. is considering several investment options. Each option has a minimum investment required as well as a maximum investment allowed. These restrictions, along with the expected return are summarized in the following table (figures are in millions of dollars):

Option	Minimum investment	Maximum investment	Expected return (%)
1	3	27	13
2	2	12	9
3	9	35	17

Option	Minimum investment	Maximum investment	Expected return (%)
4	5	15	10
5	12	46	22
6	4	18	12

Because of the high-risk nature of Option 5, company policy requires that the total amount invested in Option 5 be no more that the combined amount invested in Options 2, 4 and 6. In addition, if an investment is made in Option 3, it is required that at least a minimum investment be made in Option 6. ABC has $80 million to invest and obviously wants to maximize its total expected return on investment. Which options should ABC invest in, and how much should be invested?

In [21]:
```
using JuMP, HiGHS
m = Model(HiGHS.Optimizer)
set_silent(m)

lows  = [  3,  2,  9,  5, 12,  4 ]  # minimum investment
highs = [ 27, 12, 35, 15, 46, 18 ]  # maximum investment
ret   = [ 13,  9, 17, 10, 22, 12 ]  # expected return

# your code here
n = length(lows)

@variable(m, z[1:n], Bin)
@variable(m, x[1:n] >= 0, Int)
@constraint(m, c1[i in 1:n], z[i] * lows[i] <= x[i])
@constraint(m, c2[i in 1:n], x[i] <= z[i] * highs[i])
@constraint(m, sum(x[i] for i in 1:n) <= 80)
@constraint(m, x[5] <= sum(x[i] for i in [2, 4, 6]))
@constraint(m, x[6] >= z[3] * lows[6])

@expression(m, profit, sum(x[i]*ret[i]/100 for i in 1:n))

@objective(m, Max, profit)

optimize!(m)
println("The maximum return on investment (net profit) is: \$", objective_value(
println("This is an average return of: ", 100*objective_value(m)/value(sum(x)),
println("Here is a tally of each investment, whether we invest or not, and how m
value.([1:6 z x])
```

```
The maximum return on investment (net profit) is: $13.5 million.
This is an average return of: 16.875 %
Here is a tally of each investment, whether we invest or not, and how much we i
nvest (in millions of $).
```

Out[21]:
```
6×3 Matrix{Float64}:
 1.0  0.0   0.0
 2.0  0.0   0.0
 3.0  1.0  34.0
 4.0  1.0   5.0
 5.0  1.0  23.0
 6.0  1.0  18.0
```

4. Paint production

As part of its weekly production, a paint company produces five batches of paints, always the same, for some big clients who have a stable demand. Every paint batch is produced in a single production process, all in the same blender that needs to be cleaned between each batch. The durations of blending paint batches 1 to 5 are 40, 35, 45, 32 and 50 minutes respectively. The cleaning times depend of the colors and the paint types. For example, a long cleaning period is required if an oil-based paint is produced after a water-based paint, or to produce white paint after a dark color. The times are given in minutes in the following matrix A where A_{ij} denotes the cleaning time after batch i if it is followed by batch j.

$$A = \begin{bmatrix} 0 & 11 & 7 & 13 & 11 \\ 5 & 0 & 13 & 15 & 15 \\ 13 & 15 & 0 & 23 & 11 \\ 9 & 13 & 5 & 0 & 3 \\ 3 & 7 & 7 & 7 & 0 \end{bmatrix}$$

Since the company has other activities, it wishes to deal with this weekly production in the shortest possible time (blending and cleaning). What is the corresponding order of paint batches? The order will be applied every week, so the cleaning time between the last batch of one week and the first of the following week needs to be accounted for in the total duration of cleaning.

In [27]:
```
# A[i,j] is the time it takes to clean after batch i if followed by batch j
A = [  0  11   7  13  11
       5   0  13  15  15
      13  15   0  23  11
       9  13   5   0   3
       3   7   7   7   0 ]

# time it takes to blend a particular batch.
b = [ 40, 35, 45, 32, 50 ]

using JuMP, HiGHS
m = Model(HiGHS.Optimizer)
set_silent(m)

n = length(b)  # number of paints

# YOUR CODE HERE
paints = 1:n
@variable(m, x[paints, paints], Bin)
@variable(m, u[paints] >= 1)

@constraint(m, c1[j in paints], sum(x[i,j] for i in paints) == 1)
@constraint(m, c2[i in paints], sum(x[i,j] for j in paints) == 1)
@constraint(m, c3[i in paints], x[i,i] == 0)

@constraint(m, c4[i in paints, j in paints[2:end]], u[i] - u[j] + n * x[i,j] <=
@objective(m, Min, sum(x[i,j]*A[i,j] for i in paints, j in paints))

optimize!(m)

X = value.(x);
```

```
for i = 1:n
    for j=1:n
        if (X[i,j] >= .99)
            println(" blend ",i," is followed by blend ",j," requiring cleanup t
        end
    end
end

println("and the minimum cleanup time is: ", objective_value(m), " min, (plus ",
```

```
 blend 1 is followed by blend 4 requiring cleanup time 13
 blend 2 is followed by blend 1 requiring cleanup time 5
 blend 3 is followed by blend 5 requiring cleanup time 11
 blend 4 is followed by blend 3 requiring cleanup time 5
 blend 5 is followed by blend 2 requiring cleanup time 7
and the minimum cleanup time is: 41.0 min, (plus 202 min of blending)
0; Iter: Time           0; average =           0; Bound =           0
```

In [ ]: