hw4

April 7, 2024

Name: Kefan Zheng

StudentId: 9086175008

Email: kzheng58@wisc.edu

1 Problem 1-1

Define variables:
$N = \{1, \ldots, 255\}$: 225 assets
x_i: the new holdings in the 524Fund for asset i
buy_i: how much of each asset to buy
$sell_i$: how much of each asset to sell

$$
\begin{aligned}
\min_{buy_i, sell_i} \quad & \sum_{i \in N} (buy_i + sell_i) && (1)\\
\text{s.t.} \quad & (x-b)^T Q (x-b) \leq 100 && (2)\\
& \sum_{i \in N} x_i = 1 && (3)\\
& buy_i \geq x_i - h_i, \quad \forall i \in N && (4)\\
& sell_i \geq h_i - x_i, \quad \forall i \in N && (5)\\
& x_i \geq 0, buy_i \geq 0, sell_i \geq 0, \quad \forall i \in N && (6)\\
& && (7)
\end{aligned}
$$

2 Problem 1-2

[1]:
```
using DataFrames, CSV, LinearAlgebra, NamedArrays

df = CSV.read("folio_mean.csv", DataFrame, header=false, delim=',')
(n,mmm) = size(df)

# Weekly numbers to annual (and flip returns to make more positive)
mu = -100/7*365*Vector(df[1:n,1])

df2 = CSV.read("folio_cov.csv",DataFrame,header=false,delim=',')
```

```julia
# Weekly numbers to annual, also reduce the risk a bit
Q = 0.5* (100/7*365)^2 * Matrix(df2)

df3 = CSV.read("folio_holding_benchmark.csv", DataFrame, header=false,␣
↪delim=',')

h = Vector(df3[1:n,1])
b = Vector(df3[1:n,2])

# Current tracking risk
benchmark_return = mu'*b

# Current holdings return
holdings_return = mu'*h

# Current tracking risk
active_risk = sqrt((h-b)'Q*(h-b))

println("Benchmark expected return: ", benchmark_return)
println("Holdings expected return: ", holdings_return)
println("Active Risk: ", active_risk)
```

```
Benchmark expected return: 9.48798378650461
Holdings expected return: 5.67073489826738
Active Risk: 32.90926641232401
```

[2]:

```julia
# model
using JuMP, HiGHS, Gurobi

N = 1:225

m = Model(Gurobi.Optimizer)

@variable(m, x[N] >= 0)
@variable(m, buy[N] >= 0)
@variable(m, sell[N] >= 0)

@objective(m, Min, sum(buy + sell))

@constraint(m, ([x[i] - b[i] for i in N])' * Q * ([x[i] - b[i] for i in N]) <=␣
↪100)
@constraint(m, sum(x) == 1)
@constraint(m, buy_constraint[i in N], buy[i] >= x[i] - h[i])
@constraint(m, sell_constraint[i in N], sell[i] >= h[i] - x[i])

optimize!(m)
```

```julia
x = [value(x[i]) for i in N]
buy = [value(buy[i]) for i in N]
sell = [value(sell[i]) for i in N]
trans = buy + sell

total_transacted = objective_value(m)
avg_10_trans_size = sum(sort(trans, rev=true)[1:10]) / 10
expected_return = mu' * x
active_risk = sqrt((x-b)'Q*(x-b))

println("The total number of dollars transacted(\$): ", total_transacted*1e9)
println("The average size of the largest 10 transactions(\$): ",␣
↪avg_10_trans_size*1e9)
println("The portfolio expected return(%): ", expected_return)
println("The portfolio active risk(%): ", active_risk)
```

```
Set parameter Username
Academic license - for non-commercial use only - expires 2025-04-05
Gurobi Optimizer version 11.0.1 build v11.0.1rc0 (mac64[arm] - Darwin 23.4.0
23E224)

CPU model: Apple M1
Thread count: 8 physical cores, 8 logical processors, using up to 8 threads

Optimize a model with 451 rows, 675 columns and 1125 nonzeros
Model fingerprint: 0xc0352a8c
Model has 1 quadratic constraint
Coefficient statistics:
  Matrix range     [1e+00, 1e+00]
  QMatrix range    [3e+03, 1e+05]
  QLMatrix range   [1e+04, 4e+04]
  Objective range  [1e+00, 1e+00]
  Bounds range     [0e+00, 0e+00]
  RHS range        [4e-04, 1e+00]
  QRHS range       [1e+04, 1e+04]
Presolve removed 410 rows and 410 columns
Presolve time: 0.01s
Presolved: 268 rows, 492 columns, 26183 nonzeros
Presolved model has 1 second-order cone constraint
Ordering time: 0.00s

Barrier statistics:
 AA' NZ     : 3.036e+04
 Factor NZ  : 3.124e+04 (roughly 1 MB of memory)
 Factor Ops : 4.948e+06 (less than 1 second per iteration)
 Threads    : 8
```

```
                  Objective               Residual
Iter       Primal          Dual         Primal    Dual     Compl     Time
   0   5.62075844e+03  0.00000000e+00  2.84e+04 1.00e-01  1.84e+01     0s
   1   1.26503761e+03 -5.88261413e+00  5.69e+03 2.29e-02  3.67e+00     0s
   2   2.90312377e+02 -5.93111407e+00  1.26e+03 5.65e-03  8.33e-01     0s
   3   4.13042595e+01 -4.19599904e+00  1.62e+02 2.78e-03  1.31e-01     0s
   4   4.09579167e+00 -4.18320458e+00  1.04e+01 5.77e-04  2.28e-02     0s
   5   2.55645936e+00  5.08076139e-02  4.09e+00 8.64e-05  5.79e-03     0s
   6   1.48828327e+00  2.25187206e-01  9.86e-01 1.38e-05  2.69e-03     0s
   7   1.19106380e+00  6.34924054e-01  1.30e-01 3.68e-06  1.11e-03     0s
   8   1.12089519e+00  9.89378969e-01  1.54e-03 1.81e-07  2.50e-04     0s
   9   1.09820079e+00  1.07670809e+00  7.73e-05 2.01e-07  4.09e-05     0s
  10   1.09372582e+00  1.09181693e+00  1.09e-06 2.00e-06  3.63e-06     0s
  11   1.09328068e+00  1.09291539e+00  3.53e-08 4.37e-07  6.97e-07     0s
  12   1.09310926e+00  1.09303233e+00  7.72e-09 2.30e-08  1.45e-07     0s
  13   1.09306437e+00  1.09305894e+00  5.44e-10 3.34e-09  1.02e-08     0s
  14   1.09306020e+00  1.09306005e+00  9.22e-09 1.22e-08  2.77e-10     0s

Barrier solved model in 14 iterations and 0.04 seconds (0.04 work units)
Optimal objective 1.09306020e+00


User-callback calls 88, time in user-callback 0.00 sec
The total number of dollars transacted($): 1.0930601964404004e9
The average size of the largest 10 transactions($): 4.973345999975171e7
The portfolio expected return(%): 8.975471256896084
The portfolio active risk(%): 9.999998860345132
```

3 Problem 1-3

Define variables:
$N = \{1, \ldots, 255\}$: 225 assets
x_i: the new holdings in the 524Fund for asset i
buy_i: how much of each asset to buy
$sell_i$: how much of each asset to sell
t_i: the size of the transaction for asset i

$$
\begin{aligned}
\min_{t_i} \; & \sum_{i \in N} t_i && (8)\\
\text{s.t. } & (x-b)^T Q (x-b) \leq 100 && (9)\\
& \sum_{i \in N} x_i = 1 && (10)\\
& t_i \geq (buy_i + sell_i)^{3/2}, \quad \forall i \in N && (11)\\
& buy_i \geq x_i - h_i, \quad \forall i \in N && (12)\\
& sell_i \geq h_i - x_i, \quad \forall i \in N && (13)\\
& t_i \geq 0, x_i \geq 0, buy_i \geq 0, sell_i \geq 0, \quad \forall i \in N && (14)\\
& && (15)
\end{aligned}
$$

4 Problem 1-4

[3]:
```julia
# model
using JuMP, HiGHS, Gurobi

N = 1:225

m = Model(Gurobi.Optimizer)

@variable(m, t[N] >= 0)
@variable(m, x[N] >= 0)
@variable(m, buy[N] >= 0)
@variable(m, sell[N] >= 0)

@variable(m, u >= 0)

@objective(m, Min, sum(t))

@constraint(m, ([x[i] - b[i] for i in N])' * Q * ([x[i] - b[i] for i in N]) <=
 ↪100)
@constraint(m, sum(x) == 1)
@constraint(m, buy_constraint[i in N], buy[i] >= x[i] - h[i])
@constraint(m, sell_constraint[i in N], sell[i] >= h[i] - x[i])

for i in N
    @constraint(m, [t[i], u, buy[i] + sell[i]] in RotatedSecondOrderCone())
    @constraint(m, [0.125, buy[i] + sell[i], u] in RotatedSecondOrderCone())
end

optimize!(m)

x = [value(x[i]) for i in N]
buy = [value(buy[i]) for i in N]
```

```julia
sell = [value(sell[i]) for i in N]
trans = buy + sell

total_market_impact = objective_value(m)
total_transacted = sum(trans)
avg_10_trans_size = sum(sort(trans, rev=true)[1:10]) / 10
expected_return = mu' * x
active_risk = sqrt((x-b)'Q*(x-b))

println("The total number of dollars transacted(\$): ", total_transacted*1e9)
println("The average size of the largest 10 transactions(\$): ",␣
↪avg_10_trans_size*1e9)
println("The portfolio expected return(%): ", expected_return)
println("The portfolio active risk(%): ", active_risk)
println("Total market impact: ", objective_value(m))
```

```
Set parameter Username
Academic license - for non-commercial use only - expires 2025-04-05
Gurobi Optimizer version 11.0.1 build v11.0.1rc0 (mac64[arm] - Darwin 23.4.0
23E224)

CPU model: Apple M1
Thread count: 8 physical cores, 8 logical processors, using up to 8 threads

Optimize a model with 1801 rows, 2251 columns and 4950 nonzeros
Model fingerprint: 0x74dd534b
Model has 451 quadratic constraints
Coefficient statistics:
  Matrix range     [7e-01, 1e+00]
  QMatrix range    [1e+00, 1e+05]
  QLMatrix range   [1e+04, 4e+04]
  Objective range  [1e+00, 1e+00]
  Bounds range     [0e+00, 0e+00]
  RHS range        [4e-04, 1e+00]
  QRHS range       [1e+04, 1e+04]
Presolve removed 205 rows and 0 columns
Presolve time: 0.01s
Presolved: 1823 rows, 2478 columns, 30418 nonzeros
Presolved model has 451 second-order cone constraints
Ordering time: 0.00s

Barrier statistics:
 Dense cols : 112
 AA' NZ     : 3.592e+04
 Factor NZ  : 5.060e+04 (roughly 2 MB of memory)
 Factor Ops : 5.623e+06 (less than 1 second per iteration)
 Threads    : 8
```

```
                 Objective              Residual
Iter       Primal          Dual         Primal    Dual     Compl     Time
   0   3.04779528e+04  0.00000000e+00  2.84e+04 1.00e-01  1.91e+01     0s
   1   9.38425889e+03 -7.73187505e+00  7.06e+03 1.84e-02  5.13e+00     0s
   2   8.14798868e+02 -1.45673560e+01  6.76e+02 1.21e-03  4.49e-01     0s
   3   1.73978838e+01 -1.65634618e+01  6.45e+00 1.35e-09  1.37e-02     0s
   4   1.12068240e+01 -5.48690626e-01  4.36e+00 5.55e-12  4.60e-03     0s
   5   3.47189625e+00 -3.28053565e-01  1.21e+00 1.26e-08  1.46e-03     0s
   6   1.44999618e+00  3.97789963e-02  3.06e-01 2.89e-09  5.33e-04     0s
   7   1.07525162e+00  2.54498383e-01  3.65e-02 4.44e-10  3.03e-04     0s
   8   5.37434885e-01  3.50637559e-01  6.56e-03 9.70e-11  6.90e-05     0s
   9   5.12421589e-01  4.34285280e-01  6.06e-04 6.73e-10  2.87e-05     0s
  10   4.74335887e-01  4.62300498e-01  9.07e-05 1.26e-09  4.42e-06     0s
  11   4.68421053e-01  4.67579405e-01  5.37e-06 3.35e-09  3.09e-07     1s
  12   4.68077217e-01  4.68015123e-01  2.77e-07 1.49e-07  2.28e-08     1s
  13   4.68054538e-01  4.68053154e-01  1.71e-05 4.66e-05  5.08e-10     1s
  14   4.68054538e-01  4.68053160e-01  1.43e-04 9.44e-05  5.06e-10     1s
  15   4.68054538e-01  4.68053161e-01  1.62e-04 2.92e-04  5.06e-10     1s
  16   4.68054538e-01  4.68053161e-01  2.44e-04 6.68e-05  5.06e-10     1s
  17   4.68054538e-01  4.68053161e-01  7.73e-04 1.30e-04  5.06e-10     1s
  18   4.68054539e-01  4.68053161e-01  3.08e-04 2.67e-04  5.06e-10     1s
  19   4.68054539e-01  4.68053178e-01  1.40e-04 7.89e-05  5.00e-10     1s
  20   4.68054539e-01  4.68053178e-01  1.54e-04 5.58e-05  5.00e-10     1s
  21   4.68054539e-01  4.68053180e-01  2.31e-04 9.88e-05  4.99e-10     1s
  22   4.68054539e-01  4.68053180e-01  9.31e-05 1.70e-04  4.99e-10     1s
  23   4.68054539e-01  4.68053181e-01  1.31e-04 2.90e-04  4.98e-10     2s
  24   4.68054539e-01  4.68053183e-01  1.71e-04 8.34e-04  4.98e-10     2s
  25   4.68054539e-01  4.68053182e-01  3.43e-04 2.22e-03  4.98e-10     2s
  26   4.68054538e-01  4.68053184e-01  2.32e-04 5.07e-03  4.97e-10     2s
  27   4.68054538e-01  4.68053188e-01  6.18e-04 7.05e-03  4.96e-10     2s
  28   4.68054538e-01  4.68053368e-01  5.14e-04 3.76e-03  4.30e-10     2s
  29   4.68054538e-01  4.68053369e-01  9.00e-04 3.00e-04  4.30e-10     2s
  30   4.68054538e-01  4.68053369e-01  4.31e-04 1.88e-05  4.30e-10     2s
  31   4.68054538e-01  4.68053369e-01  4.34e-04 6.00e-05  4.30e-10     2s
  32   4.68054538e-01  4.68053368e-01  8.72e-04 1.25e-04  4.30e-10     2s
  33   4.68054538e-01  4.68053368e-01  1.88e-03 9.81e-06  4.30e-10     2s
  34   4.68054538e-01  4.68053368e-01  3.99e-03 5.23e-05  4.30e-10     2s
  35   4.68054538e-01  4.68053368e-01  8.29e-03 1.09e-04  4.30e-10     2s
  36   4.68054537e-01  4.68053368e-01  6.30e-04 8.26e-06  4.30e-10     2s
  37   4.68054537e-01  4.68053368e-01  1.27e-04 4.37e-05  4.30e-10     2s
  38   4.68054537e-01  4.68053368e-01  1.42e-04 9.10e-05  4.30e-10     2s

Barrier solved model in 38 iterations and 2.37 seconds (1.17 work units)
Optimal objective 4.68077217e-01


User-callback calls 136, time in user-callback 0.00 sec
The total number of dollars transacted($): 2.353808633812046e9
```

```
The average size of the largest 10 transactions($): 4.4669704729051076e7
The portfolio expected return(%): 8.86417349335492
The portfolio active risk(%): 9.999961277117253
Total market impact: 0.46807721672222
```

5 Problem 2-1

[4]:
```julia
using LinearAlgebra

Q = [0 0 -2 -4 0 1; 0 1 -1 -1 3 -4; -2 -1 -1 -5 7 -4; -4 -1 -5 -3 7 -2; 0 3 7 7␣
↪-1 -2; 1 -4 -4 -2 -2 0]

eigenvalues = eigen(Q).values
eigenvectors = eigen(Q).vectors

for i in 1:length(eigenvalues)
    println("Eigenvalue and eigenvector ", i, ": ")
    println(eigenvalues[i])
    println(eigenvectors[:, i])
    println()
end
```

```
Eigenvalue and eigenvector 1:
-16.11909446064489
[-0.1987242977523469, -0.197556001253088, -0.5224071070766683,
-0.5828949996148007, 0.528041265986293, -0.1731384855171353]

Eigenvalue and eigenvector 2:
-3.7566481293641356
[0.34454207606212667, -0.48321415505502313, -0.12955757672414325,
0.20288580391709257, -0.19859106643536195, -0.7418952833059451]

Eigenvalue and eigenvector 3:
-0.5922928569671946
[0.7381140909502316, -0.08982267386571899, -0.05559540037785556,
0.21880615104455992, 0.5378837171867339, 0.3268540997258541]

Eigenvalue and eigenvector 4:
2.2331144580905438
[0.2365002066186307, 0.758157947138062, -0.5375521373305713,
0.07945936608368989, -0.14657878781811945, -0.22913477958491651]

Eigenvalue and eigenvector 5:
3.845741541835812
[0.43251242371642257, 0.20495876850846464, 0.48302190937691003,
-0.7018253449914204, -0.11575000585619687, -0.17792656596553236]
```

```
Eigenvalue and eigenvector 6:
10.389179447049866
[0.23235243843489226, -0.3203061416611552, -0.4300492238793125,
-0.26892755999951556, -0.5979395551718429, 0.4781424901030135]
```

6 Problem 2-2

Define variables:
D_{ij}: the element of row i and column j of matrix D

$$
\begin{aligned}
\min_{D_{ii}} \ & \sum_{i=1}^{6} D_{ii} && (16)\\
\text{s.t. } & Q + D \succeq 0 && (17)\\
& D_{ij} = 0, \quad \forall i \in \{1, \dots, 6\}, j \in \{1, \dots, 6\}, i \neq j && (18)\\
& D_{ii} \geq 0, \quad \forall i \in \{1, \dots, 6\} && (19)\\
& && (20)
\end{aligned}
$$

7 Problem 2-3

[5]:
```
using SCS

N = 1:6

m = Model(SCS.Optimizer)

@variable(m, D[N, N])

@objective(m, Min, sum(D[i, i] for i in N))

@constraint(m, (Q + D) in PSDCone())
for i in N
    for j in N
        if i != j
            @constraint(m, D[i, j] == 0)
        end
    end
end

optimize!(m)

D_value = zeros(6, 6)
for i in N
    for j in N
        D_value[i, j] = abs(value(D[i, j])) < 1e-6 ? 0 : value(D[i, j])
```

```
    end
end
println("Matrix D: ")
for i in N
    println(D_value[i, :])
end
println()
println("Sum of diagonal entries of matrix D: ", objective_value(m))
```

```
------------------------------------------------------------------
               SCS v3.2.4 - Splitting Conic Solver
        (c) Brendan O'Donoghue, Stanford University, 2012
------------------------------------------------------------------
problem:  variables n: 36, constraints m: 66
cones:    z: primal zero / dual free vars: 45
          s: psd vars: 21, ssize: 1
settings: eps_abs: 1.0e-04, eps_rel: 1.0e-04, eps_infeas: 1.0e-07
          alpha: 1.50, scale: 1.00e-01, adaptive_scale: 1
          max_iters: 100000, normalize: 1, rho_x: 1.00e-06
          acceleration_lookback: 10, acceleration_interval: 10
lin-sys:  sparse-direct-amd-qdldl
          nnz(A): 81, nnz(P): 0
------------------------------------------------------------------
 iter | pri res | dua res |   gap   |   obj   |  scale  | time (s)
------------------------------------------------------------------
     0| 2.93e+01  1.00e+00  2.25e+02 -4.93e+01  1.00e-01  2.19e-03
    75| 1.29e-04  2.84e-05  1.47e-04  7.80e+01  1.00e-01  2.71e-03
------------------------------------------------------------------
status:  solved
timings: total: 2.71e-03s = setup: 1.50e-03s + solve: 1.21e-03s
         lin-sys: 6.10e-05s, cones: 1.10e-03s, accel: 3.16e-06s
------------------------------------------------------------------
objective = 78.001581
------------------------------------------------------------------
Matrix D:
[5.000409211348522, 0.0, 0.0, 0.0, 0.0, 0.0]
[0.0, 8.000305232257379, 0.0, 0.0, 0.0, 0.0]
[0.0, 0.0, 20.000180196888756, 0.0, 0.0, 0.0]
[0.0, 0.0, 0.0, 22.000188945397788, 0.0, 0.0]
[0.0, 0.0, 0.0, 0.0, 16.000083594851954, 0.0]
[0.0, 0.0, 0.0, 0.0, 0.0, 7.000341028999374]

Sum of diagonal entries of matrix D: 78.00150820974379
```

8 Problem 2-4

$$
\begin{align}
\min_{D_{ij}} \; & \sum_{i=1}^{6}\sum_{j=1}^{6} |D_{ij}| \tag{21}\\
\text{s.t. } & Q + D \succeq 0 \tag{22}\\
& D_{ij} = D_{ji}, \quad \forall i \in \{1,\dots,6\}, j \in \{1,\dots,6\} \tag{23}\\
& D_{ij} \;\; free, \quad \forall i \in \{1,\dots,6\}, j \in \{1,\dots,6\} \tag{24}\\
& \tag{25}
\end{align}
$$

9 Problem 2-5

[6]:

```julia
using SCS

N = 6

m = Model(SCS.Optimizer)

@variable(m, D[1:N, 1:N])
@variable(m, absD[1:N, 1:N])

@objective(m, Min, sum(absD[i, j] for i in 1:N for j in 1:N))

@constraint(m, (Q + D) in PSDCone())
for i in 1:N
    for j in 1:N
        @constraint(m, absD[i, j] >= D[i, j])
        @constraint(m, absD[i, j] >= -D[i, j])
    end
end

for i in 1:N
    for j in i+1:N
        @constraint(m, D[i, j] == D[j, i])
    end
end

optimize!(m)

D_value = zeros(6, 6)
for i in 1:N
    for j in 1:N
        D_value[i, j] = abs(value(D[i, j])) < 1e-6 ? 0 : value(D[i, j])
    end
end
println("Matrix D: ")
for i in 1:N
```

```julia
    println(D_value[i, :])
end
println()
println("Sum of the absolute values of matrix D: ", objective_value(m))
```

```
------------------------------------------------------------------
               SCS v3.2.4 - Splitting Conic Solver
        (c) Brendan O'Donoghue, Stanford University, 2012
------------------------------------------------------------------
problem:  variables n: 72, constraints m: 123
cones:    z: primal zero / dual free vars: 30
          l: linear vars: 72
          s: psd vars: 21, ssize: 1
settings: eps_abs: 1.0e-04, eps_rel: 1.0e-04, eps_infeas: 1.0e-07
          alpha: 1.50, scale: 1.00e-01, adaptive_scale: 1
          max_iters: 100000, normalize: 1, rho_x: 1.00e-06
          acceleration_lookback: 10, acceleration_interval: 10
lin-sys:  sparse-direct-amd-qdldl
          nnz(A): 225, nnz(P): 0
------------------------------------------------------------------
 iter | pri res | dua res |   gap   |   obj   |  scale  | time (s)
------------------------------------------------------------------
     0| 1.88e+01  1.00e+00  5.20e+02 -2.33e+02  1.00e-01  1.09e-04
   100| 5.36e-04  1.10e-05  1.84e-04  7.80e+01  1.00e-01  5.80e-04
------------------------------------------------------------------
status:  solved
timings: total: 5.81e-04s = setup: 8.31e-05s + solve: 4.98e-04s
         lin-sys: 1.72e-04s, cones: 2.45e-04s, accel: 6.90e-06s
------------------------------------------------------------------
objective = 78.000272
------------------------------------------------------------------
Matrix D:
[3.5914704636205377, 0.26254997869462765, 0.393653682574359, 0.1432434214424251,
-0.3599868402402367, 0.24883663755294785]
[0.2625499600888529, 4.687945452950622, 0.9718958437761157, 0.18906334502197444,
-0.28409692620305677, 1.6044998977460372]
[0.3936536676628516, 0.9718958514719253, 9.920291508355623, 4.871318293646044,
-3.561585104687813, 0.28135885171067976]
[0.1432434053258234, 0.18906335157466517, 4.87131830383001, 10.919047353653003,
-5.667207187208757, 0.2102004480472532]
[-0.35998671401675186, -0.28409682246524154, -3.5615850045812336,
-5.667207085958163, 5.8783072341875195, -0.24747394093270542]
[0.24883674156932234, 1.6045000241815546, 0.281358981900423,
0.21020057715730925, -0.24747395994058455, 4.409278181141707]

Sum of the absolute values of matrix D: 78.00036438976116
```

10 Problem 3-1

Define variables:
$N = \{1, \dots, 10\}$: set of souvenirs
$W = 30$: weight limit
$V = 15$: volumn limit
w_i: weight of the i-th souvenir in the weight vector $w = [5, 6, 7, 6, 4, 6, 7, 3, 8, 5]$
v_i: volume of the i-th souvenir in the volume vector $v = [2, 4, 5, 3, 3, 2, 3, 1, 2, 4]$
z_i: whether to pick the i-th souvenir, 1 means pick, 0 means not pick

General model:

$$\begin{aligned}
\max \ & \sum_{i \in N} z_i && (26)\\
\text{s.t. } & \sum_{i \in N} w_i z_i \leq W && (27)\\
& \sum_{i \in N} v_i z_i \leq V && (28)\\
& z_i \in \{0, 1\}, \quad \forall i \in N && (29)\\
& && (30)
\end{aligned}$$

Specific model:

$$\begin{aligned}
\max \ & \sum_{i \in \{1, \dots, 10\}} z_i && (31)\\
\text{s.t. } & 5z_1 + 6z_2 + 7z_3 + 6z_4 + 4z_5 + 6z_6 + 7z_7 + 3z_8 + 8z_9 + 5z_{10} \leq 30 && (32)\\
& 2z_1 + 4z_2 + 5z_3 + 3z_4 + 3z_5 + 2z_6 + 3z_7 + z_8 + 2z_9 + 4z_{10} \leq 15 && (33)\\
& z_i \in \{0, 1\}, \quad \forall i \in \{1, \dots, 10\} && (34)\\
& && (35)
\end{aligned}$$

11 Problem 3-2

[7]:
```
using JuMP, HiGHS

function solveKnapsack(W, V)
    N = 1:10
    w = [5, 6, 7, 6, 4, 6, 7, 3, 8, 5]
    v = [2, 4, 5, 3, 3, 2, 3, 1, 2, 4]

    m = Model(HiGHS.Optimizer)
    set_silent(m)

    @variable(m, z[N], Bin)

    @objective(m, Max, sum(z))
```

```julia
    @constraint(m, sum(w[i]*z[i] for i in N) <= W)
    @constraint(m, sum(v[i]*z[i] for i in N) <= V)


    optimize!(m)
    return m, z
end

W = 30
V = 15
m, z = solveKnapsack(W, V)
println(termination_status(m))
println()
pick_res = [value(z[i]) for i in 1:length(z)]
selected_item = [i for i in 1:length(z) if value(z[i]) != 0]
println("Vector of whether to pick: ", pick_res)
println("Selected item number(z) = ", selected_item)
println("Selected item number = ", objective_value(m))
```

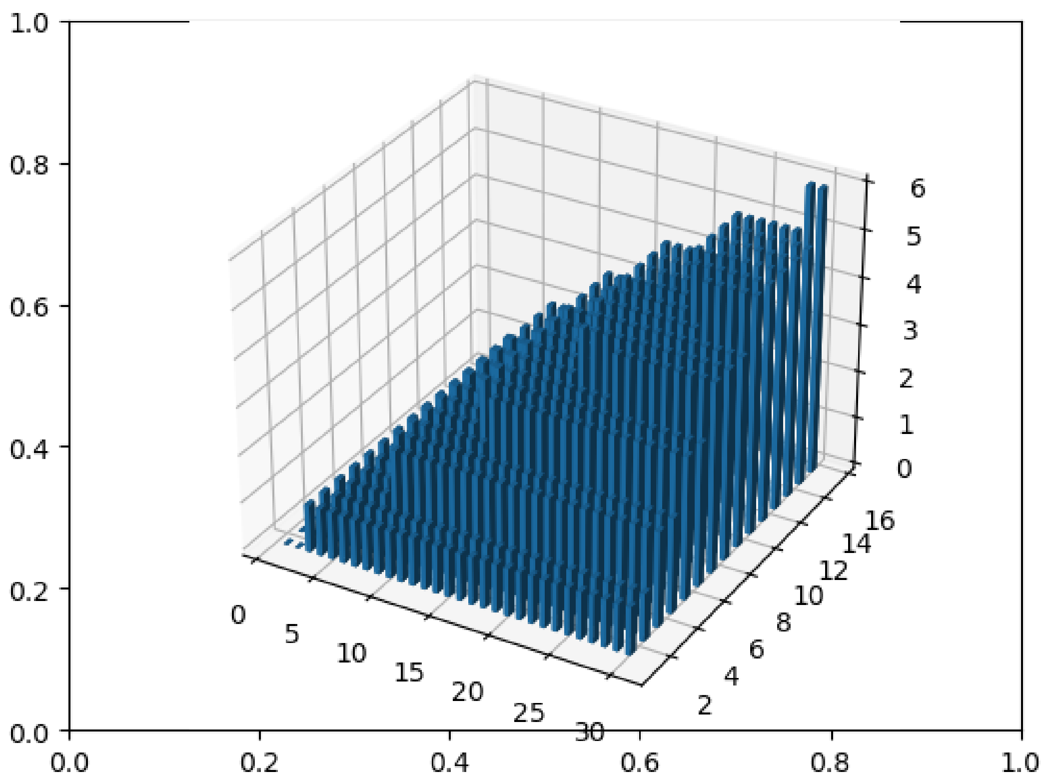
```
OPTIMAL

Vector of whether to pick: [1.0, 1.0, 0.0, 1.0, 1.0, 1.0, 0.0, 1.0, 0.0, 0.0]
Selected item number(z) = [1, 2, 4, 5, 6, 8]
Selected item number = 6.0
```

12 Problem 3-3

[8]:

```julia
using PyPlot

function plot3d(vals)
    nrows = size(vals,1)
    ncols = size(vals,2)
    _x = 1:nrows
    _y = 1:ncols
    # Make a meshgrid
    x = _x' .* ones(ncols)
    y = ones(nrows)' .* _y
    # Unravel
    x = vec(x)
    y = vec(y)
    # Heights
    dz = vec(vals')
    z = zeros(length(dz))

    dx = 0.4
    dy = 0.4
```

```
    bar3D(x,y,z,dx,dy,dz)
    # Only needed in vscode
    # display(gcf())
end;

W = 30
V = 15
vals = zeros(W, V)
for i in 1:W
    for j in 1:V
        m, z = solveKnapsack(i, j)
        vals[i, j] = objective_value(m)
    end
end

plot3d(vals)
```




```
[8]: PyObject <mpl_toolkits.mplot3d.art3d.Poly3DCollection object at 0x324929990>
```

13 Problem 4-1

Define variables:
o_1: whether boiler B1 is operated
o_2: whether boiler B2 is operated
o_3: whether boiler B3 is operated
b_1: steam produced by boiler B1
b_2: steam produced by boiler B2
b_3: steam produced by boiler B3
p_1: whether turbines T1 is operated
p_2: whether turbines T2 is operated
p_3: whether turbines T3 is operated
t_1: steam used by turbines T1
t_2: steam used by turbines T2
t_3: steam used by turbines T3

$$
\begin{align}
\min_{b_1,b_2,b_3,t_1,t_2,t_3} \quad & 10b_1 + 8b_2 + 7b_3 + 2t_1 + 3t_2 + 4t_3 \tag{36}\\
\text{s.t.} \quad & b_1 + b_2 + b_3 \geq t_1 + t_2 + t_3 \tag{37}\\
& 4t_1 + 5t_2 + 6t_3 \geq 8000 \tag{38}\\
& 400o_1 \leq b_1 \leq 1000o_1 \tag{39}\\
& 200o_2 \leq b_2 \leq 900o_2 \tag{40}\\
& 300o_3 \leq b_3 \leq 800o_3 \tag{41}\\
& 300p_1 \leq t_1 \leq 600p_1 \tag{42}\\
& 500p_2 \leq t_2 \leq 800p_2 \tag{43}\\
& 600p_3 \leq t_3 \leq 900p_3 \tag{44}\\
& o_1, o_2, o_3, p_1, p_2, p_3 \in \{0, 1\} \tag{45}\\
& b_1, b_2, b_3, t_1, t_2, t_3 \geq 0 \tag{46}\\
& \tag{47}
\end{align}
$$

14 Problem 4-2

[9]:
```
N = 1:3
produce_cost = [10, 8, 7]
min_steam_produce = [400, 200, 300]
max_steam_produce = [1000, 900, 800]

process_cost = [2, 3, 4]
min_steam_process = [300, 500, 600]
max_steam_process = [600, 800, 900]

power_produced = [4, 5, 6]
mini_power = 8000
```

```julia
m = Model(HiGHS.Optimizer)
# set_silent(m)

@variable(m, b[i in N] >= 0)
@variable(m, t[i in N] >= 0)
@variable(m, b_indicator[i in N], Bin)
@variable(m, t_indicator[i in N], Bin)

@objective(m, Min, sum(b.*produce_cost) + sum(t.*process_cost))

@constraint(m, sum(b) >= sum(t))
@constraint(m, sum(t.*power_produced) >= mini_power)
for i in N
    @constraint(m, b[i] >= min_steam_produce[i] * b_indicator[i])
    @constraint(m, b[i] <= max_steam_produce[i] * b_indicator[i])
    @constraint(m, t[i] >= min_steam_process[i] * t_indicator[i])
    @constraint(m, t[i] <= max_steam_process[i] * t_indicator[i])
end

optimize!(m)

total_power = sum(value(t[i])*power_produced[i] for i in N)
println(termination_status(m))
println()
println("Minimum cost = ", objective_value(m))
println("Total power = ", total_power)
println()
println("Steam produced by each boiler: ", value.(b))
println()
println("Steam processed by each turbine: ", value.(t))
```

```
Running HiGHS 1.7.0 (git hash: 50670fd4c): Copyright (c) 2024 HiGHS under MIT
licence terms
Coefficient ranges:
  Matrix [1e+00, 1e+03]
  Cost   [2e+00, 1e+01]
  Bound  [1e+00, 1e+00]
  RHS    [8e+03, 8e+03]
Presolving model
14 rows, 12 cols, 33 nonzeros  0s
10 rows, 10 cols, 25 nonzeros  0s

Solving MIP model with:
   10 rows
   10 cols (4 binary, 0 integer, 0 implied int., 6 continuous)
   25 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%   3900            inf                  inf
0      0      0         0     0.0s
 R       0       0         0   0.00%   15720           15720              0.00%
0      0      0         4     0.0s

Solving report
  Status            Optimal
  Primal bound      15720
  Dual bound        15720
  Gap               0% (tolerance: 0.01%)
  Solution status   feasible
                    15720 (objective)
                    0 (bound viol.)
                    0 (int. viol.)
                    0 (row viol.)
  Timing            0.00 (total)
                    0.00 (presolve)
                    0.00 (postsolve)
  Nodes             1
  LP iterations     4 (total)
                    0 (strong br.)
                    0 (separation)
                    0 (heuristics)
OPTIMAL

Minimum cost = 15720.0
Total power = 8000.0

Steam produced by each boiler: 1-dimensional DenseAxisArray{Float64,1,…} with
index sets:
    Dimension 1, 1:3
And data, a 3-element Vector{Float64}:
   0.0
 620.0
 800.0

Steam processed by each turbine: 1-dimensional DenseAxisArray{Float64,1,…}
with index sets:
    Dimension 1, 1:3
And data, a 3-element Vector{Float64}:
   0.0
 520.0
 900.0
```

15 Problem 5-1

Define variables:
m_t: pounds of milk to buy each week
c_t: pounds of Colby to sell each week
z_t: pounds of Mozerrella to sell each week
ic_t: inventory of Colby at end of each week
iz_t: inventory of Mozerella at end of each week
d_t: indicator if whether or not milk was bought each week
μ_t: maximum amount of Colby could sell in week t
λ_t: maximum amount of Mozzarella could sell in week t
f_t: a fixed cost if delivery of milk occurs in week t
p_t: per unit price of milk in week t

$$
\begin{aligned}
\max_{c_t, z_t, d_t, m_t, ic_t, iz_t} \quad & \sum_{t=1}^{8} (2.5c_t + 3z_t - d_t f_t - m_t p_t - 0.2m_t - 0.25ic_t - 0.25iz_t) && (48)\\
\text{s.t. } & ic_{t-1} + 0.5m_t - c_t = ic_t && (49)\\
& iz_{t-1} + 0.4m_t - z_t = iz_t && (50)\\
& ic_t + iz_t <= 500 && (51)\\
& m_t <= 100000 d_t && (52)\\
& m_t \geq 0 && (53)\\
& \mu_t \geq c_t \geq 0 && (54)\\
& \lambda_t \geq z_t \geq 0 && (55)\\
& ic_t \geq 0 && (56)\\
& iz_t \geq 0 && (57)\\
& d_t \in \{0, 1\} && (58)\\
& \text{for all } \ t \in \{1, \dots, 8\} && (59)\\
& && (60)
\end{aligned}
$$

16 Problem 5-2

[10]:
```
# initial colby (pounds)
i_colby = 120

# initial mozerella (pounds)
i_moz = 80

# Demand for colby cheese (pounds)
d_colby = [150 200 225 50 400 50 300 200]
# Demand for mozerella cheese (pounds)
d_moz = [200 400 300 500 100 500 200 350]

# Fixed cost for a delivery of milk ($)
fc_milk = [1000 1400 800 1200 600 1000 400 800]
```

```julia
# Per-unit cost for milk ($/pound)
p_milk = [1 0.8 0.8 1.2 1.2 1.0 1.5 0.6]

# Processing cost of milk ($/pound)
milk_proc_cost = 0.2

# inventory cost ($/pound)
cheese_inventory_cost = 0.25

# max total inventory (pounds)
max_inventory = 500

# colby price ($/pound)
colby_price = 2.5

# mozzerella price ($/pound)
moz_price = 3.0

# colby/milk
colby_per_milk = 0.5

# moz/milk
moz_per_milk = 0.4

# Number of time periods
T = 8

# The maximum milk you would buy in any period is surely no more than the total␣
↪demand for cheese
max_milk = [sum(d_colby[1:t])+sum(d_moz[1:t]) for t in 1:T]
```

[10]:
```
8-element Vector{Int64}:
  350
  950
 1475
 2025
 2525
 3075
 3575
 4125
```

[11]:
```julia
using HiGHS

model = Model(HiGHS.Optimizer)
# set_silent(model)
```

```julia
@variable(model, m[t in 1:T] >= 0)
@variable(model, d_colby[t] >= c[t in 1:T] >= 0)
@variable(model, d_moz[t] >= z[t in 1:T] >= 0)
@variable(model, ic[t in 0:T] >= 0)
@variable(model, iz[t in 0:T] >= 0)
@variable(model, d[t in 1:T], Bin)

@objective(model, Max, sum(colby_price*c[t] + moz_price*z[t] - d[t]*fc_milk[t] - m[t]*p_milk[t] - milk_proc_cost*m[t] - cheese_inventory_cost*(ic[t] + iz[t]) for t in 1:T))

@constraint(model, ic[0] == 120)
@constraint(model, iz[0] == 80)
@constraint(model, colby_inventory_constraint[t in 1:T], ic[t-1] + colby_per_milk * m[t] - c[t] == ic[t])
@constraint(model, moz_inventory_constraint[t in 1:T], iz[t-1] + moz_per_milk * m[t] - z[t] == iz[t])
@constraint(model, max_inventory_constraint[t in 1:T], ic[t] + iz[t] <= max_inventory)
@constraint(model, m .<= 100000*d)

optimize!(model)

println(termination_status(model))
println("Total maximum profit: (\$)", objective_value(model))
week_milk_purchased = [t for t in 1:T if value(d[t]) > 1e-4]
println("Weeks milk purchased: ", week_milk_purchased)
total_inventory_each_week = [value(ic[t]) + value(iz[t]) for t in 1:T]
for t in 1:T
    if abs(total_inventory_each_week[t]) < 1e-4
        total_inventory_each_week[t] = 0.0
    end
end
println("Total cheese inventory at end of week: ", total_inventory_each_week)
```

```
Running HiGHS 1.7.0 (git hash: 50670fd4c): Copyright (c) 2024 HiGHS under MIT
licence terms
Coefficient ranges:
  Matrix [4e-01, 1e+05]
  Cost   [2e-01, 1e+03]
  Bound  [1e+00, 5e+02]
  RHS    [8e+01, 5e+02]
Presolving model
32 rows, 48 cols, 94 nonzeros  0s
32 rows, 48 cols, 94 nonzeros  0s

Solving MIP model with:
```

```
   32 rows
   48 cols (8 binary, 0 integer, 0 implied int., 40 continuous)
   94 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%   11587.5         -inf                 inf
0      0      0         0     0.0s
 S       0       0         0   0.00%   11587.5         -1002.5          1255.86%
0      0      0         0     0.0s
 R       0       0         0   0.00%   2468.531694     -788.8194444     412.94%
0      0      0        24     0.0s
 S       0       0         0   0.00%   2468.531694     -37.17378731    6740.52%
24      5      0        24     0.0s
 S       0       0         0   0.00%   1959.119745     553.6074627      253.88%
50      7      0        33     0.0s
 L       0       0         0   0.00%   1565.778828     1442.777778        8.53%
108     16      0        51     0.0s

12.5% inactive integer columns, restarting
Model after restart has 29 rows, 44 cols (7 bin., 0 int., 0 impl., 37 cont.),
and 88 nonzeros

         0       0         0   0.00%   1565.54822      1442.777778        8.51%
14      0      0        83     0.0s
         0       0         0   0.00%   1565.54822      1442.777778        8.51%
14     13      0       100     0.0s

28.6% inactive integer columns, restarting
Model after restart has 21 rows, 34 cols (5 bin., 0 int., 0 impl., 29 cont.),
and 66 nonzeros

         0       0         0   0.00%   1564.833333     1442.777778        8.46%
8      0      0       110     0.0s
         0       0         0   0.00%   1564.833333     1442.777778        8.46%
8      8      0       119     0.0s

Solving report
  Status            Optimal
  Primal bound      1442.77777778
  Dual bound        1442.77777778
  Gap               0% (tolerance: 0.01%)
  Solution status   feasible
                    1442.77777778 (objective)
                    0 (bound viol.)
```

```
                    2.22044604925e-16 (int. viol.)
                    0 (row viol.)
  Timing            0.03 (total)
                    0.00 (presolve)
                    0.00 (postsolve)
  Nodes             1
  LP iterations     139 (total)
                    0 (strong br.)
                    41 (separation)
                    48 (heuristics)
OPTIMAL
Total maximum profit: ($)1442.7777777777785
Weeks milk purchased: [3, 7]
Total cheese inventory at end of week: [0.0, 0.0, 500.00000000000006,
294.44444444444446, 0.0, 0.0, 400.0000000000001, 0.0]
```

[ ]: