HW8

April 15, 2018

1 Homework 8: Integer programs

Name: Zihao Qiu
Email: zqiu34@wisc.edu
Campus ID: 9079810942

1.1 1. Thrift store

In [2]:
```
using JuMP, Cbc

m = Model(solver=CbcSolver())

@variable(m, xp>=0, Int)
@variable(m, xn>=0, Int)
@variable(m, xd>=0, Int)
@variable(m, xq>=0, Int)

@constraint(m, xp + 5*xn + 10*xd + 25*xq == 99)

@objective(m, Min, 2.5*xp + 5*xn + 2.268*xd + 5.670*xq)

status = solve(m)

println("xp = ", getvalue(xp))
println("xn = ", getvalue(xn))
println("xd = ", getvalue(xd))
println("xq = ", getvalue(xq))

println("objective = ", getobjectivevalue(m))
```

```
xp = 4.0
xn = 0.0
xd = 7.000000000000001
xq = 1.0
objective = 31.546
```

To minimize the total weight, we should use 4 pennies, 0 nickel, 7 dimes and 1 quarter.

1.2 2. Checked luggage

```
In [1]: using JuMP, Cbc

        weight = [5 6 7 6 4 6 7 3 8 5]
        volume = [2 4 5 3 3 2 3 1 2 4]

        m = Model(solver=CbcSolver())

        @variable(m, x[1:10], Bin)

        @constraint(m, sum{weight[i]*x[i], i=1:10} <= 30)
        @constraint(m, sum{volume[i]*x[i], i=1:10} <= 15)

        @objective(m, Max, sum(x))

        status = solve(m)

        for i in 1:10
            println(i, " : ", getvalue(x[i]))
        end

        println("objective = ", getobjectivevalue(m))
```

```
1 : 1.0
2 : 0.0
3 : 0.0
4 : 1.0
5 : 1.0
6 : 1.0
7 : 0.0
8 : 1.0
9 : 0.0
10 : 1.0
objective = 6.0
```

So we should choose souvenirs of numbers: 1, 4, 5, 6, 8, 10.

1.3 3. Comquat Computers

```
In [29]: using JuMP, Cbc

         total_pc = 20000
         price_per_pc = 3500

         prod_cap = [10000 8000 9000 6000]
         plant_fixcost = [9000000 5000000 3000000 1000000]
         cost_per_pc = [1000 1700 2300 2900]
```

```
n = length(prod_cap)

m = Model(solver=CbcSolver())

@variable(m, z[1:4], Bin)
@variable(m, x[1:4]>=0, Int)

@constraint(m, x .<= 20000*z) # if x>0 then z=1

for i=1:n
    @constraint(m, x[i]<=prod_cap[i])
end

@constraint(m, sum{x[i], i=1:n} <= total_pc)

@objective(m,Max,sum{x[i]*(price_per_pc-cost_per_pc[i])-z[i]*plant_fixcost[i],i=1:n})

status = solve(m)

println(status)
println("z = ", getvalue(z))
println("x = ", getvalue(x))
println("objective = ", getobjectivevalue(m))
```

```
Optimal
z = [1.0,1.0,0.0,1.0]
x = [10000.0,8000.0,0.0,2000.0]
objective = 2.56e7
```

To maximize the profit, plant1 should produce 10,000 computers, plant2 should produce 8,000 computers, plant4 should produce 2,000 computers.

1.4 4. ABC Investments

In [2]:
```
using JuMP, Gurobi

min_invest = [3 2 9 5 12 4]
max_invest = [27 12 35 15 46 18]
exp_ret = [0.13 0.09 0.17 0.1 0.22 0.12]

total = 80

n = length(exp_ret)

m = Model(solver=GurobiSolver(OutputFlag=false))
```

```julia
    @variable(m, z[1:n], Bin)
    @variable(m, x[1:n]>=0)

    @constraint(m, x .<= total*z) # if x>0 then z=1
    @constraint(m, sum{x[i], i=1:n} <= total)

    for i=1:n
        @constraint(m, min_invest[i]*z[i] <= x[i])
        @constraint(m, x[i]*z[i] <= max_invest[i])
    end

    @constraint(m, x[5] <= x[2]+x[4]+x[6])
    @constraint(m, z[3] <= z[6]) # if z[3]=1 then z[6]=1 <=> z[3] <= z[6]

    @objective(m, Max, sum{x[i]*exp_ret[i], i=1:n})

    status = solve(m)
    println(status)
    println("z = ", getvalue(z))
    println("x = ", getvalue(x))
    println("objective = ", getobjectivevalue(m))
```

```
Academic license - for non-commercial use only
Optimal
z = [0.0,0.0,1.0,1.0,1.0,1.0]
x = [0.0,0.0,35.0,5.0,22.5,17.5]
objective = 13.5
```

To maximize the profit, we should invest option3 with 35 million, option4 with 5 million, option5 with 22.5 million, option6 with 17.5 million.