In [18]:

```julia
using JuMP
```

In [139]:

```julia
using NLPModelsJuMP, NLPModels
```

In [19]:

```julia
using GLPK
```

In [28]:

```julia
using Distributions
```

In [30]:

```julia
using Random
```

In [153]:

```julia
using DataFrames
```

In [154]:

```julia
using CSV
```

In [20]:

```julia
H = 4
J = 3
Tᵈ = 1
Tˢ = 1
Ξ = 1
```

Out[20]:

```
1
```

ψ

In [24]:

```julia
ψ = [ 1 0 0 ; 0 1 0 ; 0 0 1 ; 1 0 0 ];
```

In [26]:

```julia
χ = zeros(H,J);
```

In [38]:

```
Random.seed!(1234)

μ = [1.2, 0.8, 0.6]

Σ = [0.5 0.0 0.0;
     0.0 0.5 0.0;
     0.0 0.0 0.5
    ]

dξ = reshape(rand(MvNormal(μ,Σ), Ξ ), (J,Tˢ,Ξ) )
dξ = round.(dξ , digits = 2)
```

Out[38]:

```
3×1×1 Array{Float64, 3}:
[:, :, 1] =
 1.81
 0.16
 0.25
```

In [264]:

```
zᴾ
```

Out[264]:

```
4-element Vector{Int64}:
  7000
  9000
 11000
  7000
```

In [275]:

```
zᵖ = [7; 9; 11;7 ] * 1000
zᶜ = zᵖ *2

zᵐ = fill(0, H, J)

for h=1
        zᵐ[h , 2] = (1/2) * zᵖ[h]
        zᵐ[h , 3] = (3/4) * zᵖ[h]
end

for h=2
        zᵐ[h , 3] = (1/2) * zᵖ[h]
        zᵐ[h , 1] = (3/4) * zᵖ[h]
end

for h=3
        zᵐ[h , 1] = (1/2) * zᵖ[h]
        zᵐ[h , 2] = (3/4) * zᵖ[h]
end

for h=4
        zᵐ[h , 2] = (1/2) * zᵖ[h]
        zᵐ[h , 3] = (3/4) * zᵖ[h]
end

zᵐ
```

Out[275]:

```
4×3 Matrix{Int64}:
    0  3500  5250
 6750     0  4500
 5500  8250     0
    0  3500  5250
```

###
###

main model

###
###

In [259]:

```
main = Model(GLPK.Optimizer)
@variable(main, ψ[1:H , 1:J], Bin)
@variable(main, 0 ≤ χ[1:H , 1:J])
@variable(main, α[1:H , 1:J , 1:Tᵈ], Bin)
@variable(main, -1000000 ≤ θ)

x = all_variables(main)[1 : (H * J * 2) ];
```

Stage 1 min $\sum_{h\in\Psi}\sum_{j\in\Phi} z_h^p \psi_{hj}\left\{T^d + T^s\right\} + \sum_{h\in\Psi}\sum_{j\in\Phi} z_{hj}^m \chi_{hj} + \mathbb{E}_\xi\left[Q\left(\psi_{hj}, \chi_{hj}, d_{jt}(\xi)\right)\right]$

In [261]:

```
@objective(main, Min, sum( zᵖ[h] * ψ[h,j] * (Tᵈ + Tˢ) + zᵐ[h,j] * χ[h,j]
                        for h in 1:H for j in 1:J))
```

```
UndefVarError: zᵐ not defined

Stacktrace:
 [1] macro expansion
   @ C:\Users\e29115\.julia\packages\MutableArithmetics\8xkW3\src\rewrite.j
l:279 [inlined]
 [2] macro expansion
   @ C:\Users\e29115\.julia\packages\JuMP\klrjG\src\macros.jl:1260 [inlined]
 [3] top-level scope
   @ .\In[261]:1
 [4] eval
   @ .\boot.jl:360 [inlined]
 [5] include_string(mapexpr::typeof(REPL.softscope), mod::Module, code::Stri
ng, filename::String)
   @ Base .\loading.jl:1116
```

sub model

γ

In [ ]:

```
function sub(ψ , χ)
    sub = Model(GLPK.Optimizer)
    @variable(sub, α[1:H , 1:J , 1:Tˢ , 1:Ξ], Bin )
    @variable(sub, γξ[1:J , 1:Tˢ , 1:Ξ]);
    @objective(sub, Min,  (1/Ξ) * sum(zᶜ[j] * γξ[j,t,ξ] for j in 1:J for t in 1:Tˢ for ξ in
    con1_S2 = @constraint(sub, demand_met[ j in 1:J , t in 1:Tˢ , ξ in 1:Ξ],
     sum(α[h,j,t,ξ] * (ψ[h,j] + χ[h,j]) for h in 1:H ) + γ[j,t,ξ] ≥ dξ[j,t,ξ]  );
    con2_s2 = @constraint(sub, permanent_allocability[h in 1:H , j in 1:J , t in 1:Tˢ , ξ i
            α[h,j,t,ξ] ≤ ψ[h,j] + 10 * χ[h,j] );
    con3_s2 = @constraint(sub, no_more_than_one_station[h in 1:H , t in 1:Tˢ , ξ in 1:Ξ],
          sum( α[h,j,t,ξ] for j in 1:J) ≤ 1 );
end
```

$$\underset{\gamma_{jt}(\xi)}{minimise} \quad \frac{1}{n}\sum_{t\in\text{\mT}^s}\sum_{j\in\Phi}\sum_{\xi\in\Xi} z_j^c \gamma_{jt}(\xi)$$

$$\sum_{j\in\Phi}\alpha_{hjt}(\xi) \le 1 \qquad \forall h\in\Psi, \quad \forall t\in \text{\mT}^s \quad \forall \xi\in\Xi=\{1,\ldots,n\}$$

$$\alpha_{hjt}(\xi) \le \psi_{hj} + M\chi_{hj} \qquad \forall h\in\Psi \quad \forall j\in\Phi \quad \forall t\in\text{\mT}^s \quad \forall\xi\in\Xi=\{1,\ldots,n\}$$

In [96]:

```
sub = Model(GLPK.Optimizer)
@variable(sub, α[1:H , 1:J , 1:Tˢ , 1:Ξ], Bin )
@variable(sub, γ[1:J , 1:Tˢ , 1:Ξ]);
@objective(sub, Min,  (1/Ξ) * sum(zᶜ[j] * γ[j,t,ξ] for j in 1:J for t in 1:Tˢ for ξ in 1:Ξ)
con1_S2 = @constraint(sub, demand_met[ j in 1:J , t in 1:Tˢ , ξ in 1:Ξ],
     dξ[j,t,ξ] ≤ sum(α[h,j,t,ξ] * (ψ[h,j] + χ[h,j]) for h in 1:H ) + γ[j,t,ξ]    )

con2_s2 = @constraint(sub, permanent_allocability[h in 1:H , j in 1:J , t in 1:Tˢ , ξ in 1:
            α[h,j,t,ξ] ≤ ψ[h,j] + 10 * χ[h,j] );
con3_s2 = @constraint(sub, no_more_than_one_station[h in 1:H , t in 1:Tˢ , ξ in 1:Ξ],
           sum( α[h,j,t,ξ] for j in 1:J) ≤ 1 );
optimize!(sub)
l = value.(α)
```

Out[96]:

```
4×3×1×1 Array{Float64, 4}:
[:, :, 1, 1] =
 1.0  0.0  0.0
 0.0  1.0  0.0
 0.0  0.0  1.0
 1.0  0.0  0.0
```

###
###

defining a function for second stage integer dual

##

when we want to get the dual the x (first stage variables are considered as fixed)

when we want to get the coefficents of x, x should be variable

therefore, we need to define two models one for dual and one for coefficient

##

###
###

In [234]:

```
#############################################
# when we want to get the dual the x (first stage variables are considered as fixed)
# when we want to get the coefficents of x, x should be variable
# therefore, we need to define two models one for dual and one for
#############################################

function sub_dual(ψ , χ)
    sub_for_dual = Model(GLPK.Optimizer)
    @variable(sub_for_dual,  α[1:H,1:J,1:Tˢ , 1:Ξ])
    @variable(sub_for_dual, 0 ≤ γ[1:J , 1:Tˢ , 1:Ξ])
    @objective(sub_for_dual, Min,  sum(zᶜ[j] * γ[j,t,ξ] for j in 1:J for t in 1:Tˢ for ξ in
    for h in 1:H
            for j in 1:J
                for t in 1:Tˢ
                    for ξ in 1:Ξ
                        if l[h,j,t,ξ] == 0
                            con = @constraint(sub_for_dual, α[h,j,t,ξ] == 0)
                        else
                            con = @constraint(sub_for_dual, α[h,j,t,ξ] == 1)
                        end
                    end
                end
            end
        end

    con1_S2 = @constraint(sub_for_dual, demand_met[ j in 1:J , t in 1:Tˢ , ξ in 1:Ξ],
         dξ[j,t,ξ] ≤ sum(α[h,j,t,ξ] * (ψ[h,j] + χ[h,j]) for h in 1:H ) + γ[j,t,ξ]   )
    con2_s2 = @constraint(sub_for_dual, permanent_allocability[h in 1:H , j in 1:J , t in 1
                α[h,j,t,ξ] ≤ ψ[h,j] + 10 * χ[h,j] );
    con3_s2 = @constraint(sub_for_dual, no_more_than_one_station[h in 1:H , t in 1:Tˢ , ξ i
               sum( α[h,j,t,ξ] for j in 1:J) ≤ 1 );
    optimize!(sub_for_dual)
   #print(sub_for_dual)

    con_equal = all_constraints(sub_for_dual, AffExpr, MOI.EqualTo{Float64})
    con_less = all_constraints(sub_for_dual, AffExpr, MOI.LessThan{Float64})
    λ1 = dual.(con_equal)
    λ2 = dual.(con_less)
    λ = append!(λ1 , λ2)

    no_con_equal = length(con_equal)
    no_con_less = length(con_less)
    no_all_con = no_con_equal + no_con_less;
    @show no_con_equal
    @show no_con_less
    @show no_all_con;

    return λ
end

ψ = [ 1 0 0 ; 0 1 0 ; 0 0 1 ; 1 0 0 ];
χ = zeros(H,J)

sub_dual(ψ , χ)
```

```
no_con_equal = 12
no_con_less = 19
no all con = 31
```

Out[234]:

```
31-element Vector{Float64}:
 0.0
 0.0
 0.0
 0.0
 0.0
 0.0
 0.0
 0.0
 0.0
 0.0
 0.0
 0.0
 0.0
 ⋮
 0.0
 0.0
 0.0
 0.0
 0.0
 0.0
 0.0
 0.0
 0.0
 0.0
 0.0
 0.0
```

```
########################################################################################################
########################################################################################################
```

function for coefficients of ψ and χ

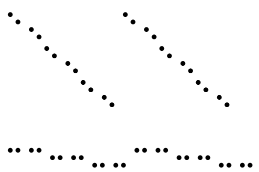
```
########################################################################################################
########################################################################################################
```

In [224]:

```
function sub_coeff()
    sub_for_coeff = Model(GLPK.Optimizer)
    @variable(sub_for_coeff,     ψ[1:H , 1:J], Bin )
    @variable(sub_for_coeff, 0 ≤ χ[1:H , 1:J])
    @variable(sub_for_coeff,  α[1:H, 1:J , 1:Tˢ, 1:Ξ])
    @variable(sub_for_coeff, 0 ≤ γ[1:J , 1:Tˢ , 1:Ξ])
    @objective(sub_for_coeff, Min,  sum(zᶜ[j] * γ[j,t,ξ] for j in 1:J for t in 1:Tˢ for ξ i
    for h in 1:H
            for j in 1:J
                for t in 1:Tˢ
                    for ξ in 1:Ξ
                        if l[h,j,t,ξ] == 0
                            con = @constraint(sub_for_coeff, α[h,j,t,ξ] == 0)
                        else
                            con = @constraint(sub_for_coeff, α[h,j,t,ξ] == 1)
                        end
                    end
                end
            end
        end

    con1_S2 = @NLconstraint(sub_for_coeff, demand_met[ j in 1:J , t in 1:Tˢ , ξ in 1:Ξ],
         dξ[j,t,ξ] ≤ sum(α[h,j,t,ξ] * (ψ[h,j] + χ[h,j]) for h in 1:H ) + γ[j,t,ξ]    )
    con2_s2 = @constraint(sub_for_coeff, permanent_allocability[h in 1:H , j in 1:J , t in
                α[h,j,t,ξ] ≤ ψ[h,j] + 10 * χ[h,j] );
    con3_s2 = @constraint(sub_for_coeff, no_more_than_one_station[h in 1:H , t in 1:Tˢ , ξ
               sum( α[h,j,t,ξ] for j in 1:J) ≤ 1 );

    vr = all_variables(sub_for_coeff)
    vr_index = [vr[i].index.value for i in 1:length(vr)]
    df = DataFrame(variable = vr , index = vr_index);
    #@show df

    nlp = MathOptNLPModel(sub_for_coeff)
    q = zeros(nlp.meta.nvar)
    jac(nlp, q)
    A1 = jac(nlp, q)[ : , 1:24]
    return A1
end
sub_coeff()
```

Out[224]:

```
31×24 SparseArrays.SparseMatrixCSC{Float64, Int64} with 48 stored entries:
```



In [246]:

```
ψ = [ 1 0 0 ; 0 1 0 ; 0 0 1 ; 1 0 0 ];
χ = zeros(H,J)
sub_dual(ψ , χ)
sub_coeff()

cut = @constraint(main, θ ≥ sub_dual(ψ , χ)' * sub_coeff()
```

```
no_con_equal = 12
no_con_less = 19
no_all_con = 31
no_con_equal = 12
no_con_less = 19
no_all_con = 31
```

Out[246]:

```
1×24 adjoint(::Vector{Float64}) with eltype Float64:
 0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  …  0.0  0.0  0.0  0.0  0.0  0.0  0.
0
```

In [ ]:

```

```