L shaped method

This example comes from:

Birge, J. R., & Louveaux, F. (2011). Introduction to stochastic programming. Springer Science & Business Media.

The general form

$$\min c^T x$$
$$Ax \leq b$$
$$\min q^T y$$
$$Tx + Wy \leq h$$

The benders cut

$$\theta \geq e - Ex$$

$$e = \sum_{k \in K} p_k \lambda_k h_k$$

$$E = \sum_{j \in K} p_k \lambda_k T_k$$

- p is probability of realization
- λ simplex multiplier
- h coefficient of x in the second stage
- T the RHS of second stage

Stage 1

$$\begin{aligned} z = \min \quad & 100x_1 + 150x_2 \\ \text{s. t.} \quad & x_1 + x_2 \leq 120 \\ & -x_1 \leq -40 \\ & -x_2 \leq -20 \end{aligned}$$

Stage 2

(realization 1)

$$\begin{aligned} z = \min \quad & -24y_1 - 28y_2 \\ \text{s. t.} \quad & 6y_1 + 10y_2 - 60x_1 \leq 0 \\ & 8y_1 + 5y_2 - 80x_2 \leq 0 \\ & y_1 \leq 500 \\ & y_2 \leq 100 \\ & y \geq 0 \end{aligned}$$

(realization 2)

$$
\begin{aligned}
z = \min \quad & -28y_1 - 32y_2 \\
\text{s. t.} \quad & 6y_1 + 10y_2 - 60x_1 \leq 0 \\
& 8y_1 + 5y_2 - 80x_2 \leq 0 \\
& y_1 \leq 300 \\
& y_2 \leq 300 \\
& y \geq 0
\end{aligned}
$$

Type *Markdown* and LaTeX: α^2

In [11]:

```
using JuMP, GLPK
using NLPModels, NLPModelsJuMP
using Printf

function print_iteration(k, args...)
    f(x) = Printf.@sprintf("%12.4e", x)
    println(lpad(k, 9), " ", join(f.(args), " "))
    return
end

##########################
# parameters main
c1 = [100 ; 150]
A1 = [1 1 ; -1 0 ; 0 -1]
b1 = [120 ;  -40 ; -20]

##########################
# parameters sub
c2 = [-24 -28 ;
      -28 -32]
A2 = [6 10 ; 8 5 ; 1 0 ; 0 1] #y
A3 = [-60 0 ; 0 -80 ; 0 0 ; 0 0 ] # x
b2 = [0  0  500  100;
      0  0  300  300]

#############################
p = [0.4 0.6]


####################################
# defining the main model but don't solve it
####################################
main = Model(GLPK.Optimizer)
@variable(main, x[1:2])
@variable(main, -1000000 ≤ θ)
@objective(main, Min, c1' * x)
@constraint(main, A1 * x .≤ b1);

#################
# I define the sub problem inside functions

function sub_dual(x)
    λ = zeros(2 , 4) # 2 realization and 4 constraints
    for i in 1:2 # 2 realization
        sub = Model(GLPK.Optimizer)
        @variable(sub, 0 ≤ y[1:2])
        @objective(sub, Min, c2[i , :]' * y)
        @constraint(sub, A2 * y + A3 * x .≤ b2[i , :])
        optimize!(sub)
        all_con = all_constraints(sub, AffExpr, MOI.LessThan{Float64})
        λ[i , :] = dual.(all_con)
        λ = round.(λ , digits =2)
    end
    return λ
end

###############################
# I define a function to give me the h and T
# I don't need to solve this model
```

```julia
function sub_coef()
    T = [] # coefficient of x
    h = zeros(2,4)  # RHS
    for i in 1:2
        sub = Model(GLPK.Optimizer)
        @variable(sub, x[1:2])
        @variable(sub, 0 ≤ y[1:2])
        @objective(sub, Min, c2[i , :]' *y)
        @constraint(sub, A2 * y + A3 * x .≤ b2[i , :])
        nlp = MathOptNLPModel(sub)
        s = zeros(nlp.meta.nvar)
        T = jac(nlp , s)[ : , 1:2]
        h[ i , :] = nlp.meta.ucon
    end
    return (T , h)
end

sub_coef()

function initialize()
    for i in 1:10
        optimize!(main)
        xᵏ = value.(x)
        θᵏ = value(θ)
        λᵏ = sub_dual(xᵏ)
        Tᵏ = sub_coef()[1]
        hᵏ = sub_coef()[2]
        eᵏ = sum(p[i] * λᵏ[i,j]' * hᵏ[i,j] for i in 1:2 for j in 1:4)
        Eᵏ = sum(p[i] * λᵏ[i, :]' * Tᵏ  for i in 1:2)
        wᵛ = eᵏ - Eᵏ * xᵏ
        print_iteration(i,    θᵏ,    wᵛ)
        if θᵏ ≥ wᵛ -10
            println("***********************************")
            println("****   we are at optimality*****")
            break
        end
        cut = @constraint(main, θ ≥ eᵏ - Eᵏ * x )
        println("add the $(cut)" )
    end
end

initialize()
```

```
        1  -1.0000e+06  -7.4704e+03
add the 83.52 x[1] + 180.48000000000002 x[2] + θ >= -520.0
        2  -7.4704e+03  -7.4704e+03
**********************************
****   we are at optimality*****
```

In [ ]: