ISyE524: Introduction to Optimization
Problem Set #1
Due: February 25 at 11:59PM

Notes and Deliverables:

- Submit solutions to all problems in the form of a single PDF file of the IJulia notebook to the course dropbox.

1 Snitches'R'Us

I am sure that you all know that Quidditch is the most popular game in the wizarding world. In order to play quidditch, you need a snitch. We are going to write a linear program to help Angelina Johnson, former chaser for Gryffindor, and now production manager for Snitches'R'Us. From past data, Angelina knows that if the production rate of snitches changes, additional costs are incurred. She estimates that the cost of increasing production is \$1.50 per snitch increased from one month to the next. Similarly, the cost of decreasing production is \$1.00 per unit decreased from one month to the next. It costs only \$0.10/snitch in production costs. Sales forecasts (in thousands) for the next six months are given in Table 1.

Table 1: Snitch Sales Forecasts

Month	**Demand**
July	4
August	8
September	20
October	12
November	6
December	2

For the purpose of this problem, assume the current month is June, the production level in June was 4000 units, and the June 30 inventory level is projected to be 2000 units. Storage of the first 8000 units costs \$0.20/unit/month. Additional storage (for units in excess of 8000) can be rented for \$0.50/unit/month. Demand in each month *must* be met.

1-1 Problem Formulate a linear program to determine the optimal production schedule.
: Hint. I had decision variables for the following quantities for each period:

- Number of snitches to produce during period t
- The increase in production rates between periods $t-1$ and t
- The decrease in production rates between periods $t-1$ and t
- Total inventory level at end of period t
- Number of snitches (up to 8000) at the end of period t
- Number of snitches (over 8000) at the end of period t

Answer:
Let T be the set of periods, with demand $d_t, t \in T$ as given in Table 1. We further assume the set T is ordered, and for any element $t \in T$, we use the operator $t-1$ to refer to the previous element in the ordered set. Define the following decision variables:

- x_t: The number of snitches to produce during period t
- p_t: The increase in production rates between periods $t-1$ and t
- n_t: The decrease in production rates between periods $t-1$ and t
- I_t: The total inventory level at end of period t
- L_t: The number of snitches (up to 8000) at the end of period t
- H_t: The number of snitches (over 8000) at the end of period t

$$\min \sum_{t \in T} (0.1x_t + 0.2L_t + 0.5H_t + 1.0n_t + 1.5p_t)$$

$$\begin{aligned}
I_t &= L_t + H_t && \forall t \in T \\
x_t &= x_{t-1} + p_t - n_t && \forall t \in T \setminus \{\text{July}\} \\
x_{July} &= 4000 + p_{July} - n_{July} && \\
I_t &= I_{t-1} + x_t - d_t && \forall t \in T \setminus \{\text{July}\} \\
I_{July} &= 2000 + x_{July} - d_{July} && \\
L_t &\le 8000 && \forall t \in T \\
x_t, p_t n_t, I_t, L_t, y_t, r_t &\ge 0 && \forall t \in T
\end{aligned}$$

There are multiple correct models for this one. Note that you could have also defined your p_t and n_t variables using constraints

$$\begin{aligned}
p_t &\ge x_t - x_{t-1} \\
n_t &\ge x_{t-1} - x_t
\end{aligned}$$

instead of just splitting the difference in production rate into two variables.

1-2 Problem Create your model from Problem 1-1 in the Julia and JuMP modeling language and solve it to determine the monthly production levels. Be sure to print the optimal solution value and the monthly production levels.

Answer:

Julia Code

```
# For multi-period problems, it is often easiest to have the sets
# (of decision-making periods) be a list of integers, which is then naturally ordered

T = 6
d = [4000,8000,20000,12000,6000,2000]
```

```

ProdCost = 0.1
LowInventoryCost = 0.2
HighInventoryCost = 0.5
IncreaseProductionCost = 1.5
DecreaseProductionCost = 1.0
HighInventoryLevel = 8000
InitialInventory = 2000
InitialProduction = 4000

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

@variable(m, x[1:T] >= 0) # Production during t
@variable(m, p[1:T] >= 0) # Production increase at beginning of t
@variable(m, n[1:T] >= 0) # Production decrease at beginning of t
@variable(m, inv[1:T] >= 0) # Inventory at the end of t
@variable(m, lo_inv[1:T] >= 0) # Inventory (up to 8000) at the end of t
@variable(m, high_inv[1:T] >= 0) # Inventory over 8000 at the end of t

# Minimize costs, 5 components: production, low inventory, high inventory,
# increase prodution, decrease production in each month
@objective(m, Min, sum(ProdCost*x[t] + LowInventoryCost*lo_inv[t] + HighInventoryCost*high_inv[t] +
IncreaseProductionCost*p[t] + DecreaseProductionCost*n[t] for t in 1:T))


# New production = old production + increase - decrease (all execpt first month)
@constraint(m, [t in 2:T], x[t] == x[t-1] + p[t] - n[t])
#Do it for first month
@constraint(m, x[1] == InitialProduction + p[1] - n[1])

# Inventory dynamics:  new inventory = old_inventory + production - demand (all except first)
@constraint(m, [t in 2:T], inv[t] == inv[t-1] + x[t] - d[t])
#Inventory dynamics, first period
@constraint(m, inv[1] == InitialInventory + x[1] - d[1])

# Split inventory into low and high
@constraint(m, [t in 1:T], inv[t] == lo_inv[t] + high_inv[t])

# Make sure that low inventory level does not exceed maximum
@constraint(m, [t in 1:T], lo_inv[t] <= HighInventoryLevel)

optimize!(m)

# Here is some fancy things for printing. Make a dict mapping integer index to month name.
using Dates
month_name = Dict(x-6 => Dates.monthname(x) for x in (7:12) )

using Formatting


printfmtln("{1:12s}{2:12s}{3:12s}{4:12s}{5:12s}{6:12s}{7:12s}", "Month", "Production",
```

```
    "Increase", "Decrease", "Inventory", "Low", "High")
    for t in 1:T
        printfmtln("{1:12s}{2:12.2f}{3:12.2f}{4:12.2f}{5:12.2f}{6:12.2f}{7:12.2f}",
        month_name[t], value(x[t]), value(p[t]), value(n[t]), value(inv[t]),
        value(lo_inv[t]), value(high_inv[t]))
    end
    printfmtln("Min cost schedule: {1:.2f}", objective_value(m))
```

Gives the following answer:

Julia Answer

```
Month       Production  Increase    Decrease    Inventory   Low         High
July          10000.00     6000.00        0.00     8000.00     8000.00        0.00
August        10666.67      666.67        0.00    10666.67     8000.00     2666.67
September     10666.67        0.00        0.00     1333.33     1333.33        0.00
October       10666.67        0.00        0.00       -0.00        0.00        0.00
November       8000.00        0.00     2666.67     2000.00     2000.00        0.00
December       8000.00        0.00        0.00     8000.00     8000.00        0.00
Min cost schedule: 25266.67
```

1-3 Problem Modify your answer from Problem 1-1 so that demand may be backlogged at a cost of \$.25 per unit per month. Be sure to enforce that demand for all snitches must eventually be met.

Answer:
Let T be the set of periods, with demand $d_t, t \in T$ as given in Table 1. We further assume the set T is ordered, and for any element $t \in T$, we use the operator $t-1$ to refer to the previous element in the ordered set. Define the following decision variables:

- x_t: The number of snitches to produce during period t
- p_t: The increase in production rates between periods $t-1$ and t
- n_t: The decrease in production rates between periods $t-1$ and t
- I_t: The total inventory level at end of period t
- L_t: The number of snitches (up to 8000) at the end of period t
- H_t: The number of snitches (over 8000) at the end of period t
- B_t: The number of snitches backlogged from period $t+1$ to meet demand at period t (or earlier)

$$\min \sum_{t \in T} (0.1x_t + 0.2L_t + 0.5H_t + 1.5p_t + 1.0n_t + 0.25B_t)$$

$$
\begin{aligned}
I_t &= L_t + H_t && \forall t \in T \\
x_t - x_{t-1} &= p_t - n_t && \forall t \in T \setminus \{\text{July}\} \\
x_{July} - 4000 &= p_{July} - n_{July} && \\
I_{t-1} + x_t + B_t &= d_t + I_t + B_{t-1} && \forall t \in T \setminus \{\text{July}\} \\
2000 + x_{July} + B_{July} &= 4000 + I_{July} && \\
L_t &\leq 8000 && \forall t \in T \\
x_t, p_t n_t, I_t, L_t, B_t &\geq 0 && \forall t \in T
\end{aligned}
$$

1-4 Problem Implement your model from Problem 1-3 in JuMP to determine an optimal production schedule. Print the objective value, the number of snitches in inventory, and the backlogged amounts in each period.

Answer:

Not so many changes from the previous model:

Julia Code

```julia

T = 6
d = [4000,8000,20000,12000,6000,2000]

ProdCost = 0.1
LowInventoryCost = 0.2
HighInventoryCost = 0.5
IncreaseProductionCost = 1.5
DecreaseProductionCost = 1.0
HighInventoryLevel = 8000
InitialInventory = 2000
InitialProduction = 4000
BacklogCost = 0.25

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

@variable(m, x[1:T] >= 0) # Production during t
@variable(m, p[1:T] >= 0) # Production increase at beginning of t
@variable(m, n[1:T] >= 0) # Production decrease at beginning of t
@variable(m, inv[1:T] >= 0) # Inventory at the end of t
@variable(m, lo_inv[1:T] >= 0) # Inventory (up to 8000) at the end of t
@variable(m, high_inv[1:T] >= 0) # Inventory over 8000 at the end of t
@variable(m, b[1:T] >= 0) # Snitches backlogged from period t+1 to meet demand in period t (or earlier)

# Minimize costs, : production, low inventory, high inventory, increase prodution,
decrease production, backlog cost in each month
@objective(m, Min, sum(ProdCost*x[t] + LowInventoryCost*lo_inv[t] +
HighInventoryCost*high_inv[t] + IncreaseProductionCost*p[t] +
DecreaseProductionCost*n[t] + BacklogCost*b[t] for t in 1:T))

```

```

# New production = old production + increase - decrease (all execpt first month)
@constraint(m, [t in 2:T], x[t] == x[t-1] + p[t] - n[t])
#Do it for first month
@constraint(m, x[1] == InitialProduction + p[1] - n[1])

# Inventory dynamics:  backlog in + production + old_inventory =
# backlog out + demand + new inventory (all except first)
@constraint(m, [t in 2:T], b[t] + x[t] + inv[t-1] == b[t-1] + d[t] + inv[t])
#Inventory dynamics, first period
@constraint(m, b[1] + x[1] + InitialInventory == d[1] + inv[1])

# Split inventory into low and high
@constraint(m, [t in 1:T], inv[t] == lo_inv[t] + high_inv[t])

# Make sure that low inventory level does not exceed maximum
@constraint(m, [t in 1:T], lo_inv[t] <= HighInventoryLevel)

optimize!(m)

# Here is some fancy things for printing. Make a dict mapping integer index to month name.
using Dates
month_name = Dict(x-6 => Dates.monthname(x) for x in (7:12) )

using Formatting


printfmtln("{1:12s}{2:12s}{3:12s}{4:12s}{5:12s}{6:12s}{7:12s}{8:12s}",
"Month", "Production", "Increase", "Decrease", "Backlog", "Inventory", "Low", "High")
for t in 1:T
    printfmtln("{1:12s}{2:12.2f}{3:12.2f}{4:12.2f}{5:12.2f}{6:12.2f}{7:12.2f}",
    month_name[t], value(x[t]), value(p[t]), value(n[t]), value(b[t]),
    value(inv[t]), value(lo_inv[t]), value(high_inv[t]))
end
printfmtln("Min cost schedule: {1:.2f}", objective_value(m))
```

Gives

Julia Answer

```
Month       Production  Increase    Decrease    Backlog     Inventory   Low         High
July           8333.33     4333.33        0.00        0.00     6333.33     6333.33        0.00
August         8333.33        0.00        0.00        0.00     6666.67     6666.67        0.00
September      8333.33        0.00        0.00     5000.00       -0.00        0.00        0.00
October        8333.33        0.00        0.00     8666.67       -0.00        0.00        0.00
November       8333.33        0.00        0.00     6333.33       -0.00        0.00        0.00
December       8333.33        0.00        0.00        0.00       -0.00        0.00        0.00
Min cost schedule: 19100.00
```

2 Least "Squares."

The set of six equations in four variables (1)—(6) does not have a unique solution.[1]

$$
\begin{aligned}
8x_1 - 2x_2 + 4x_3 - 9x_4 &= 17 & (1)\\
x_1 + 6x_2 - x_3 - 5x_4 &= 16 & (2)\\
x_1 - x_2 + x_3 &= 7 & (3)\\
x_1 + 2x_2 - 7x_3 + 4x_4 &= 15 & (4)\\
x_3 - x_4 &= 6 & (5)\\
x_1 + x_3 - x_4 &= 0 & (6)
\end{aligned}
$$

For each equation i, and values of variables $x = (x_1, x_2, x_3, x_4)$, let r_i be the ***absolute*** difference (error), or residual, between the left hand side and the right hand side. For example, for $i = 2$ and $x = (-5, 3, 1, 4)$, the error is

$$r_2 = |(1)(-5) + 6(3) - (1)(1) - 5(4) - 16| = | - 24| = 24.$$

2-1 Problem Write a linear programming instance that will minimize the total absolute error (sum of residuals)

$$r_1 + r_2 + r_3 + r_4 + r_5 + r_6$$

Answer:

This is called least-squares with the ℓ_1 norm: $\|\cdot\|_1$. Normal least squares is with the ℓ_2, or Euclidean norm $\|\cdot\|_2$.

Here is a model that splits the residual into positive and negative parts, and then minimizes the sum of the positive and negative parts:

$$\min p_1 + n_1 + p_2 + n_2 + p_3 + n_3 + p_4 + n_4 + p_5 + n_5 + p_6 + n_6$$

$$
\begin{aligned}
\text{s.t.} \quad 8x_1 - 2x_2 + 4x_3 - 9x_4 - 17 &= p_1 - n_1\\
x_1 + 6x_2 - x_3 - 5x_4 - 16 &= p_2 - n_2\\
x_1 - x_2 + x_3 - 7 &= p_3 - n_3\\
x_1 + 2x_2 - 7x_3 + 4x_4 - 15 &= p_4 - n_4\\
x_3 - x_4 - 6 &= p_5 - n_5\\
x_1 + x_3 - x_4 &= p_6 - n_6\\
p_1, n_1, p_2, n_2, p_3, n_3, p_4, n_4, p_5, n_5, p_6, n_6 &\ge 0\\
x_1, x_2, x_3, x_4 & \quad \text{FREE}
\end{aligned}
$$

You can also instead introduce one new variable r_i for each constraint and model $|\text{error}| \le r_i$.

[1] Most six equations with four variables don't.

$$
\begin{aligned}
\min\ & r_1 + r_2 + r_3 + r_4 + r_5 + r_6 \\
\text{s.t.}\quad & 8x_1 - 2x_2 + 4x_3 - 9x_4 - 17 \le r_1 \\
& -(8x_1 - 2x_2 + 4x_3 - 9x_4 - 17) \le r_1 \\
& x_1 + 6x_2 - x_3 - 5x_4 - 16 \le r_2 \\
& -(x_1 + 6x_2 - x_3 - 5x_4 - 16) \le r_2 \\
& x_1 - x_2 + x_3 - 7 \le r_3 \\
& -(x_1 - x_2 + x_3 - 7) \le r_3 \\
& x_1 + 2x_2 - 7x_3 + 4x_4 - 15 \le r_4 \\
& -(x_1 + 2x_2 - 7x_3 + 4x_4 - 15) \le r_4 \\
& x_3 - x_4 - 6 \le r_5 \\
& -(x_3 - x_4 - 6) \le r_5 \\
& x_1 + x_3 - x_4 \le r_6 \\
& -(x_1 + x_3 - x_4) \le r_6 \\
& r_1, r_2, r_3, r_4, r_5, r_6 \ge 0 \\
& x_1, x_2, x_3, x_4 \quad \text{FREE}
\end{aligned}
$$

2-2 Problem Create the instance you wrote down for Problem 2-1 in JuMP and solve it, printing the minimum total residual and the values of x that achieve it.

Answer:

I implemented both models

Julia Code

```
  # Let's just input the data here as a Matrix
M = 6
N = 4
A = [
  8 -2 4 -9
  1 6 -1 -5
  1 -1 1 0
  1 2 -7 4
  0 0 1 -1
  1 0 1 -1
]

b = [17 16 7 15 6 0] ;
using JuMP, HiGHS

m1 = Model(HiGHS.Optimizer)
#set_silent(m1)
@variable(m1, x[1:N]) #x variables
@variable(m1, p[1:M] >= 0) # variable representing positive part of error
```

```
@variable(m1, n[1:M] >= 0) # variable representing positive part of error

@objective(m1, Min, sum(p) + sum(n)) # Shorthand for sum(p[i] for i in 1:m)
@constraint(m1, split_error[i in 1:M], sum(A[i,j]*x[j] for j in 1:N) - b[i] == p[i] - n[i])

println(m1)
optimize!(m1)

println("total residual: ", objective_value(m1))
println("x variables: ", value.(x))
```

yields

Julia Answer

```
Running HiGHS 1.6.0: Copyright (c) 2023 HiGHS under MIT licence terms
Presolving model
6 rows, 16 cols, 32 nonzeros
6 rows, 16 cols, 32 nonzeros
...
Objective value     :  1.5419161677e+01
HiGHS run time      :          0.00
total residual: 15.41916167664671
x variables: [9.419161676646711, 8.023952095808388, 5.6047904191616835, 7.191616766467074]
```

The implementation

Julia Code

```
m2 = Model(HiGHS.Optimizer)
@variable(m2, x[1:N]) #x variables
@variable(m2, r[1:M] >= 0) #x variables
@expression(m2, e[i in 1:M], sum(A[i,j]*x[j] for j in 1:N) - b[i])
@objective(m2, Min, sum(r))
@constraint(m2, pos_error[i in 1:M], r[i] >= e[i])
@constraint(m2, neg_error[i in 1:M], r[i] >= -e[i])
println(m2)

println(m2)
optimize!(m2)

println("total residual: ", objective_value(m2))
println("x variables: ", value.(x))
```

Also gives the same answer

2-3 Problem I asked GPTv3.5 to answer this problem for me, and here is what is returned:

$$\begin{aligned}
\text{Minimize } & r_1 + r_2 + r_3 + r_4 + r_5 + r_6 \\
\text{subject to } & r_1 \geq 8x_1 - 2x_2 + 4x_3 - 9x_4 - 17, \\
& r_1 \geq -(8x_1 - 2x_2 + 4x_3 - 9x_4 - 17), \\
& r_2 \geq x_1 + 6x_2 - x_3 - 5x_4 - 16, \\
& r_2 \geq -(x_1 + 6x_2 - x_3 - 5x_4 - 16), \\
& r_3 \geq x_1 - x_2 + x_3 - 7, \\
& r_3 \geq -(x_1 - x_2 + x_3 - 7), \\
& r_4 \geq x_1 + 2x_2 - 7x_3 + 4x_4 - 15, \\
& r_4 \geq -(x_1 + 2x_2 - 7x_3 + 4x_4 - 15), \\
& r_5 \geq x_3 - x_4 - 6, \\
& r_5 \geq -(x_3 - x_4 - 6), \\
& r_6 \geq x_1 + x_3 - x_4, \\
& r_6 \geq -(x_1 + x_3 - x_4), \\
& r_1, r_2, r_3, r_4, r_5, r_6 \geq 0.
\end{aligned}$$

Is this a correct model for the problem? If so, state how (if at all) it is different from your answer. If not, please state how you would correct the model.

Answer:
The model is correct. It is equivalent to the second model in my answer. (GPTv3.5 decided not to do the 'split variable' formulation).

2-4 Problem For the same instance, write a *linear* program that will minimize the *maximum* error in any one equation. Namely find values of x that will

$$\min \max\{r_1, r_2, r_3, r_4, r_5, r_6\}$$

for the definition of error (residual) r_i I gave above.

Answer:

This is called least-squares with the ℓ_∞ norm: $\|\cdot\|_\infty$. Normal least squares is with the ℓ_2, or Euclidean norm $\|\cdot\|_2$. Again, we coudl use a split variable formulation to implement the minimax:

$$z$$

$$
\begin{aligned}
\text{s.t.} \quad 8x_1 - 2x_2 + 4x_3 - 9x_4 - 17 &= p_1 - n_1 \\
x_1 + 6x_2 - x_3 - 5x_4 - 16 &= p_2 - n_2 \\
x_1 - x_2 + x_3 - 7 &= p_3 - n_3 \\
x_1 + 2x_2 - 7x_3 + 4x_4 - 15 &= p_4 - n_4 \\
x_3 - x_4 - 6 &= p_5 - n_5 \\
x_1 + x_3 - x_4 &= p_6 - n_6 \\
z &\geq p_1 + n_1 \\
z &\geq p_2 + n_2 \\
z &\geq p_3 + n_3 \\
z &\geq p_4 + n_4 \\
z &\geq p_5 + n_5 \\
z &\geq p_6 + n_6 \\
p_1, n_1, p_2, n_2, p_3, n_3, p_4, n_4, p_5, n_5, p_6, n_6 &\geq 0 \\
x_1, x_2, x_3, x_4 FREE
\end{aligned}
$$

This one is easier using a ***single*** max residual variable r. No need to split the error into positive and negative parts.

$$\min r$$

$$
\begin{aligned}
\text{s.t.} \quad 8x_1 - 2x_2 + 4x_3 - 9x_4 - 17 &\leq r \\
-(8x_1 - 2x_2 + 4x_3 - 9x_4 - 17) &\leq r \\
x_1 + 6x_2 - x_3 - 5x_4 - 16 &\leq r \\
-(x_1 + 6x_2 - x_3 - 5x_4 - 16) &\leq r \\
x_1 - x_2 + x_3 - 7 &\leq r \\
-(x_1 - x_2 + x_3 - 7) &\leq r \\
x_1 + 2x_2 - 7x_3 + 4x_4 - 15 &\leq r \\
-(x_1 + 2x_2 - 7x_3 + 4x_4 - 15) &\leq r \\
x_3 - x_4 - 6 &\leq r \\
-(x_3 - x_4 - 6) &\leq r \\
x_1 + x_3 - x_4 &\leq r \\
-(x_1 + x_3 - x_4) &\leq r \\
r &\geq 0 \\
x_1, x_2, x_3, x_4 &\quad \text{FREE}
\end{aligned}
$$

2-5 Problem I asked GPTv3.5 to solve the problem, and here is what it returned

$$
\begin{aligned}
\text{Minimize } & \text{max\_residual} \\
\text{subject to } & r_1 = 8x_1 - 2x_2 + 4x_3 - 9x_4 - 17 \\
& r_2 = x_1 + 6x_2 - x_3 - 5x_4 - 16 \\
& r_3 = x_1 - x_2 + x_3 - 7 \\
& r_4 = x_1 + 2x_2 - 7x_3 + 4x_4 - 15 \\
& r_5 = x_3 - x_4 - 6 \\
& r_6 = x_1 + x_3 - x_4 \\
& \text{max\_residual} \geq r_1 \\
& \text{max\_residual} \geq r_2 \\
& \text{max\_residual} \geq r_3 \\
& \text{max\_residual} \geq r_4 \\
& \text{max\_residual} \geq r_5 \\
& \text{max\_residual} \geq r_6 \\
& x_1, x_2, x_3, x_4, r_1, r_2, r_3, r_4, r_5, r_6, \text{max\_residual} \geq 0.
\end{aligned}
$$

Is this a correct model for the problem? If so, state how (if at all) it is different from your answer. If not, please state how you would correct the model.

Answer:
This is not a correct model for the problem. GPTv3.5 mistakenly set all of the x variables to be non-negative. To correct the model, we just need to make th x_1, x_2, x_3, x_4 free variables.

2-6 Problem Implement your model from Problem 2-4 in JuMP and solve it. What is the minimum max-residual that can be achieved?

Answer:

Here is my model:

Julia Code

```julia
    m3 = Model(HiGHS.Optimizer)
set_silent(m3)

@variable(m3, x[1:N]) #x variables
@variable(m3, r >= 0) #r variable representing maximum residual
@expression(m3, e[i in 1:M], sum(A[i,j]*x[j] for j in 1:N) - b[i])

@constraint(m3, pos_error[i in 1:M], r >= e[i])
@constraint(m3, neg_error[i in 1:M], r >= -e[i])

@objective(m3, Min, r)
optimize!(m3)

```

```
println("maximum residual: ", objective_value(m3))
println("x variables: ", value.(x))
println("Residuals: ", value.(e))
```

This gives:

Julia Answer

```
maximum residual: 5.934426229508198
x variables: [3.9999999999999947, 3.475409836065566, 0.540983606557373, 0.4754098360655644]
Residuals: [5.9344262295082375, 5.934426229508196, -5.934426229508198, -5.934426229508226, -5.9344262295081
```

Note that many of the residuals are exactly the same.

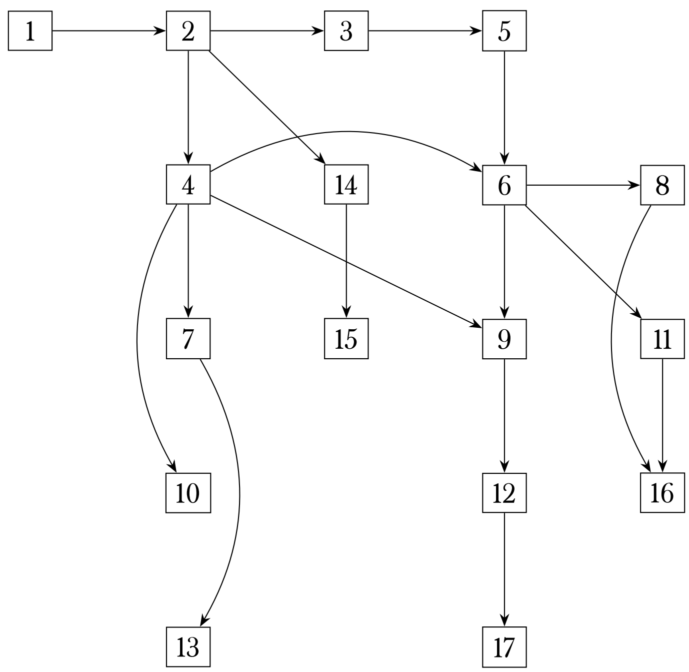
3 Stadium Building

A town council wishes to construct a small stadium in order to improve the services provided to the people living in the district. After the invitation to tender, a local construction company is awarded the contract and wishes to complete the task within the shortest possible time. All the major tasks are listed in the following table. Some tasks can only start after the completion of certain other tasks, as indicated by the "Predecessors" column.

Task	Description	Duration (in weeks)	Predecessors	Maximum reduction (in weeks)	Cost of reduction ($1k/wk)
1	Installing the construction site	2	none	0	–
2	Terracing	16	1	3	30
3	Constructing the foundations	9	2	1	26
4	Access roads and other networks	8	2	2	12
5	Erecting the basement	10	3	2	17
6	Main floor	6	4,5	1	15
7	Dividing up the changing rooms	2	4	1	8
8	Electrifying the terraces	2	6	0	–
9	Constructing the roof	9	4,6	2	42
10	Lighting of the stadium	5	4	1	21
11	Installing the terraces	3	6	1	18
12	Sealing the roof	2	9	0	–
13	Finishing the changing rooms	1	7	0	–
14	Constructing the ticket office	7	2	2	22
15	Secondary access roads	4	4,14	2	12
16	Means of signalling	3	8,11,14	1	6
17	Lawn and sport accessories	9	12	3	16

3-1 Problem Draw the directed-acyclic graph corresponding to the precedences between tasks

Answer:



3-2 Problem Make the following definitions:

- $T := \{1, 2, \ldots, 17\}$ the set of tasks
- d_t: The duration of task $t \in T$
- $P := \{(j, i) : j \in T \text{ directly follows } i \in T\}$

Write a general LP model to determine the minimum project completion time. (To make the rest of the problem a bit easier, I suggest you write the constraints all in $\geq$ form.)

Answer:

Let x_j be the starting time for task $j \in T$. then a linear program to minimize the makespan (the longest completion time of any project) is the following:

$$\min \quad t \tag{7}$$
$$t - x_j \geq d_j \quad \forall j \in T \tag{8}$$
$$x_j - x_i \geq d_i \quad \forall (j, i) \in P \tag{9}$$
$$x_i \geq 0 \quad \forall i \in T \tag{10}$$

3-3 Problem What is the earliest possible date of completion for the construction? Note that the last two columns of the table are not relevant for this part of the problem. Build a linear programming model in Julia and JuMP model to solve the instance. Print the minimum number of weeks required to complete construction. Also print the value dual variables associated with each constraint (if the dual variable is not equal to zero).

Recall that you can print the optimal dual value of a constraint named ILove524SoMuch after solving the instance as `dual(ILove524SoMuch)`

Answer:

Julia Code

```
using JuMP, HiGHS

tasks = 1:17
dur = [2 16 9 8 10 6 2 2 9 5 3 2 1 7 4 3 9]
pre = ( [], [1], [2], [2], [3], [4,5], [4], [6], [4,6], [4], [6], [9], [7], [2], [4,14], [8,11,14], [12

pred = Dict(zip(tasks,pre));

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

@variable(m, x[tasks]>=0)
@variable(m, makespan)

@constraint(m, makespan_con[j in tasks], makespan >= x[j] + dur[j])
@constraint(m, precedence_con[j in tasks, i in pred[j]], x[j] - x[i] >= dur[i])

@objective(m, Min, makespan)

optimize!(m)
status=termination_status(m)

println("The minimum #weeks required to complete the construction: ", objective_value(m))

using Printf
dual_prec= Dict((j,i) => dual(precedence_con[j,i]) for j in tasks for i in pred[j] if abs(dual(preceden
for (key,value) in dual_prec
    @printf("Precedence constraint %s has dual value: %.2f\n", key, value)
end

dual_ms = Dict(j => dual(makespan_con[j]) for j in tasks if abs(dual(makespan_con[j])) > 1.0e-6)
for (key,value) in dual_ms
    @printf("Makespan constraint %s has dual value: %.2f\n", key, value)
end
```

Gives the following output:

Julia Answer

```
The minimum #weeks required to complete the construction: 63.0
Precedence constraint (3, 2) has dual value: 1.00
Precedence constraint (6, 5) has dual value: 1.00
Precedence constraint (9, 6) has dual value: 1.00
Precedence constraint (12, 9) has dual value: 1.00
Precedence constraint (17, 12) has dual value: 1.00
Precedence constraint (5, 3) has dual value: 1.00
Precedence constraint (2, 1) has dual value: 1.00
```

```
        Makespan constraint 17 has dual value: 1.00
```

3-4 Problem What are the activities that are on the *critical path* for project completion? (You cannot reduce the project completion time unless you reduce the time of an activity on the critical path)

Answer:
Note that from the *dual* solution to the linear program, we can see which precedence constraints are "tight", and therefore on the critical path. The activities

$$\{1, 2, 3, 5, 6, 9, 12, 17\}$$

are on the critical path.

4 Broom Rental

Oliver Wood retired from his career as keeper at Puddlemere United to start the Wood Broom Rental Company wbc, [2] and he needs your help. There is a fleet of brooms that are distributed among a set of locations I. For each location $i \in I$, there is a current number of brooms c_i and a required number of brooms r_i. The distance between any pair of locations $i \in I$, $j \in I$ is d_{ij} (in km—Wizards use the metric system). The cost of transporting a broom from one location to another is κ galleon/km.

4-1 Problem Write a min-cost network flow problem that will determine the movement of all brooms to establish the required number of brooms at all agencies in a minimum cost manner.

Answer:

This is a simple min cost network flow problem, where x_{ij} is the net number of brooms to transfer from node i to node j.

- The network is $N = (V, A)$, where V is the set of places, and $A = (i, j) = \{(i, j) \mid i \in V, j \in V\}$ is the set of all pairs of cities.
- The cost on each arc is $c_{ij} := \kappa d_{ij} \quad \forall (i, j) \in A$, where

$$d_{ij} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}.$$

- The lower bound on each arc is $L_{ij} = 0 \, \forall (i, j) \in A$, and the upper bound is $U_{ij} = \infty \, \forall (i, j) \in A$.
- The net supply for each node/location is $b_i = \text{CurrentBrooms}_i - \text{RequiredBrooms}_i$

With those definitions, we have a standard min cost network flow problem.

$$\min \sum_{(i,j) \in A} c_{ij} x_{ij}$$

[2] In the Wizarding World, brooms may be rented and used, much like cars, in our normal Muggle world.

$$\sum_{j \mid (i,j) \in A} x_{ij} - \sum_{j \mid (j,i) \in A} x_{ji} = b_i \quad \forall i \in N$$
$$L_{ij} \le x_{ij} \le U_{ij} \quad \forall (i,j) \in A$$

4-2 Problem For the 10 locations in the table below, the (x, y) coordinates of each broom rental agency (in a grid based on kilometers) is given in below, as are the number of brooms currently in each location c_i and the number of brooms required for tomorrow's rentals r_i. Naturally brooms travel "as the crow flies," so distances between agencies are Euclidean distances. Use $\kappa = 0.5$\$/km for this instance. Implement your model from Problem 4-1 in Julia and JuMP to find the number of brooms to transport as well as the minimum cost.

Place	x **Coor.**	y **Coor.**	**Required Brooms**	**Current Brooms**
Hogwarts	0	0	10	8
Godric's Hollow	20	20	6	13
Little Whinging	18	10	8	4
Shell Cottage	30	12	11	8
The Leaky Cauldron	35	0	9	12
Ollivander's	33	25	7	2
Zonko's Joke Shop	5	27	15	14
Dervish and Banges	5	10	7	11
Little Hangleton	11	0	9	15
Weasley's Wizard Wheezes	2	15	12	7

Answer:

Julia Code

```
  I = [:Hogwarts, :GodricsHollow, :LittleWhinging, :ShellCottage, :TheLeakyCauldron,
  :Ollivanders, :Zonkos, :DervishAndBanges, :LittleHangleton, :WeasleysWizardWheezes ]

required = Dict(zip(I,[10, 6, 8, 11, 9, 7, 15, 7, 9, 12]))
current = Dict(zip(I, [8,13,4,8,12,2,14,11,15,7]))

xc = Dict(zip(I, [0, 20, 18, 30, 35, 33, 5, 5, 11, 2]))
yc = Dict(zip(I, [0, 20, 10, 12, 0, 25, 27, 10, 0, 15]))
Galleons_per_km = 0.5

# Compute the distances, Fancy using dict comprehension
c = Dict((i,j) => Galleons_per_km*sqrt((xc[i]-xc[j])^2 + (yc[i] - yc[j])^2) for i in I for j in I)

# Compute the net supply
b = Dict(i => current[i] - required[i] for i in I)


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

```

```
@variable(m, x[I,I] >= 0)

@objective(m, Min, sum(c[i,j]*x[i,j] for i in I for j in I if i != j))

@constraint(m, flowbalance[i in I], sum(x[i,j] for j in I) - sum(x[j,i] for j in I) == b[i])

optimize!(m)
println("Minimum Cost: ", round(objective_value(m),digits=1))
for i in I
    for j in I
        if value(x[i,j]) > 0.01
            println("Ship ", round(value(x[i,j]), digits=1), " brooms from ", i, " to ", j)
        end
    end
end
```

Julia Answer

```
Minimum Cost: 117.4
Ship 1.0 brooms from GodricsHollow to LittleWhinging
Ship 5.0 brooms from GodricsHollow to Ollivanders
Ship 1.0 brooms from GodricsHollow to Zonkos
Ship 3.0 brooms from TheLeakyCauldron to ShellCottage
Ship 5.0 brooms from DervishAndBanges to WeasleysWizardWheezes
Ship 2.0 brooms from LittleHangleton to Hogwarts
Ship 3.0 brooms from LittleHangleton to LittleWhinging
Ship 1.0 brooms from LittleHangleton to DervishAndBanges
```

5 Chess Sets and Duality

ViditChess is a customer maker of boxwood chess sets, and two different sizes of chess sets are available for order. The small set requires 3 hours of machining on a lathe, and the large set requires 2 hours. There are four lathes with skilled operators who each work a 40 hour week, so ViditChess production has 160 lathe-hours per week. The small chess set requires 1 kg of boxwood, and the large set requires 4 kg. Unfortunately, boxwood is scarce and only 200 kg per week can be obtained. When sold, each of the large chess sets yields a profit of \$8, and one of the small chess set has a profit of \$5. The problem is to decide how many sets of each kind should be made each week so as to maximize profit.

5-1 Problem Write out both the primal and dual linear programs.

Answer:

Define the following decision variables:

- x_1: the number of small chess sets produced by ViditChess per week
- x_2: the number of large chess sets produced by ViditChess per week

The the primal dual dual linear programs to maximize profit are the following:

$$\begin{aligned} \max \quad & 5x_1 + 8x_2 && (11) \\ \text{s.t.} \quad & 3x_1 + 2x_2 \le 160 && (12) \\ & x_1 + 4x_2 \le 200 && (13) \\ & x_1, x_2 \ge 0 && (14) \end{aligned}$$

$$\begin{aligned} \min \quad & 160y_1 + 200y_2 && (15) \\ \text{s.t.} \quad & 3y_1 + y_2 \ge 5 && (16) \\ & 2y_1 + 4y_2 \ge 8 && (17) \\ & y_1, y_2 \ge 0 && (18) \end{aligned}$$

5-2 Problem Chess guru MagzyBogues claims that the optimal solution to this problem is $(0, 50)$. Is he correct? Prove your answer using complementary slackness and linear programming duality.

Answer:

No. MagzyBogues is not correct.

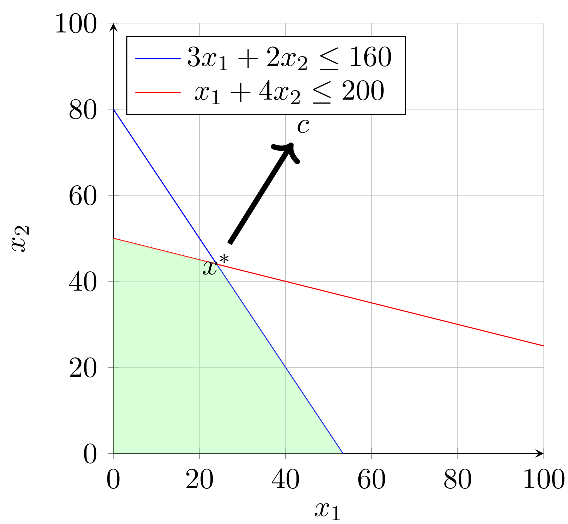
For $(0, 50)$ to be an optimal solution, its "complementary" solution (that satisfies the complementary slackness conditions) must be a feasible solution. Since the solution $(0, 50)$ is *slack* for the first primal (lathe) constraint (13), we know that the dual solution must have $y_1 = 0$. Since $x_2 > 0$ in MagzyBogues' solution, the dual constraint (17) must be tight, so $y_1 + 4y_2 = 8$. Since $y_1 = 0$, we know that the complementary solution to the primal solution $x = (0, 50)$ is then $y = (0, 2)$.

The solution $y = (0, 2)$ is ***not*** feasible to the dual. It is violated by the first constraint

$$3y_1 + y_2 = 3(0) + 2 \not\ge 5.$$

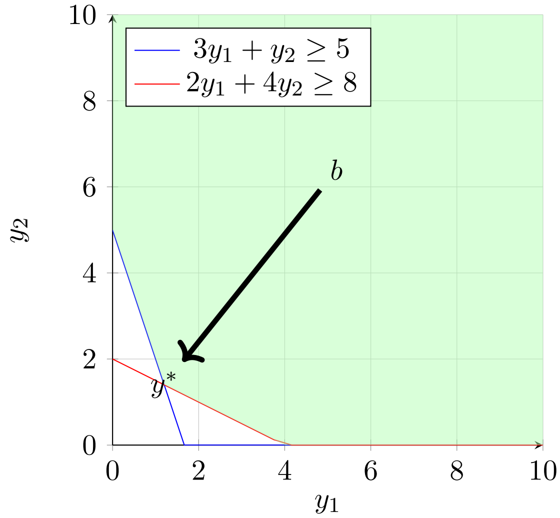
5-3 Problem Plot the feasible set and solve the primal LP graphically. Label the optimal point and find the optimal objective.

Answer:



5-4 Problem Plot the feasible set and solve the dual LP graphically. Label the optimal point and find the optimal objective.

Answer:



5-5 Problem ViditChess could increase its profit if there were more boxwood available for production. What is the maximum amount that ViditChess's owner AnishGiri should be willing to pay for an extra kg of boxwood?

Answer:

Since the optimal solution to the dual is

$$(y_1, y_2) = \left(\frac{6}{5}, \frac{7}{5}\right)$$

the marginal value of boxwood is \$1.40/kg.

5-6 Problem There are currently 200 units of boxwood available. Find the *range* of the available amount of boxwood available for which the current value of boxwood to AnishGiri remains the same.

Hint: changing this value changes the objective function of the dual. Determine for which objective *directions* your current dual solution remains optimal.

Answer:

The current dual objective function is

$$b = \begin{pmatrix} 160 \\ 200 \end{pmatrix}.$$

This objective function direction will remain optimal as long as by changing the coefficient 200, the vector does not rotate so much so as to make a different extreme point optimal. The inequality $3y_1 + y_2 \geq 5$ has normal vector $(3, 1)^\mathsf{T}$, so we know that if we decrease $b_2 = 200$ until $(160, 200 - t)^\mathsf{T}$

is in the same direction as $(3,1)^\mathsf{T}$, the current solution will remain optimal. In this case, we can compute the t so that

$$\frac{160}{3} = \frac{200-t}{1},$$

or $t = \frac{440}{3}$, so we can reduce the number of units of boxwood to 200-440/3 = 160/3, and the current solution remains optimal.

Doing a similar calculation, The inequality $2y_1 + 4_2 \geq 8$ has normal vector $(2,4)^\mathsf{T}$, we we know that if we increase $b_2 = 200$ until $(160, 200+t)^\mathsf{T}$ is in the same direction as $(2,4)^\mathsf{T}$, the current solution will remain optimal. So here $t = 120$, and we can increase the kg of boxwood to 320 and the current solution remains optimal.

Then as long as

$$b_2 \in \left[\frac{160}{3}, 320\right],$$

the current solution remains optimal. I tried to show this geomtrically in the figure below.