ISyE524: Introduction to Optimization
Problem Set #1
Due: February 11, 2024 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.
- If you need to write answers on paper, you can take a picture of you work and add that to your IJulia notebook

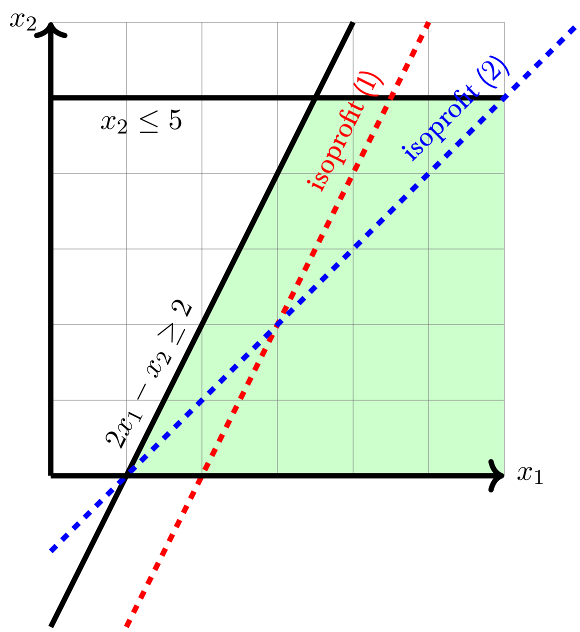
1 Graphical Method Extravaganza

Consider the following linear programming problem:

$$\begin{aligned} \max\ & -4x_1 + 2x_2 \\ \text{subject to: } & x_2 \leq 5 \\ & 2x_1 - x_2 \geq 2 \\ & x_1, \quad x_2 \geq 0 \end{aligned}$$

1-1 Problem Find an optimal solution to this linear program using the graphical solution method. (Show the feasible region as part of your answer.)

Answer:



I drew the feasible region and isoprofit line in red. One optimal solution is $(x_1, x_2) = (1, 0)$ with optimal objective value -4.

1-2 Problem How many optimal solutions does this linear program have? If more than one, provide another optimal solution.

Answer:
There are infinitely many optimal solutions to this problem. Another optimal solution is $(x_1, x_2) = (3.5, 5)$, again with optimal objective value -4. In fact, any solution on the line segment between these two points is an optimal solution.

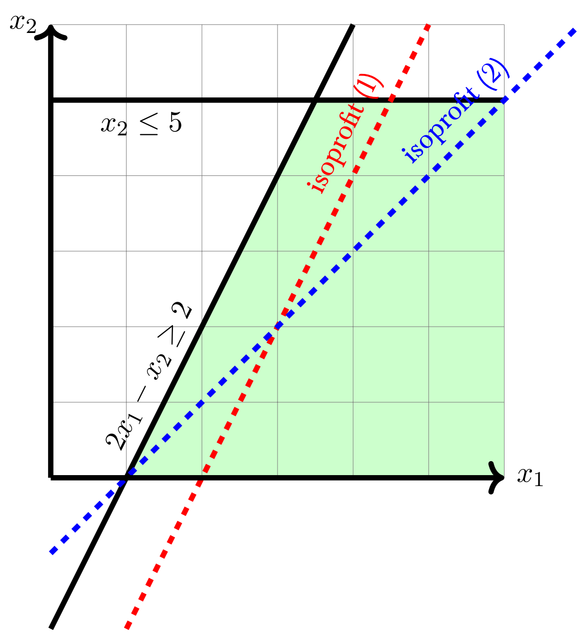
1-3 Problem Now suppose the objective is changed to maximize $-x_1 + x_2$. Answer questions 1-1 and 1-2 again.

Answer:
I drew the isoprofit line for this problem in blue. Now there is a unique optimal solution at $(x_1, x_2) = (3.5, 5)$ with optimal objective value 1.5.

1-4 Problem Now suppose the objective is changed to minimize $-x_1 + x_2$. Does the LP now have an optimal solution? If so, provide one. If not, what do we call this LP?

Answer:

The feasible region is plotted below, and an isoprofit line is in blue.



The isoprofit line is the same slope as in Problem 1-3, however in this case, the value gets smaller as we move the line to the right. We can keep moving the line to the right infinitely far and continue intersecting the feasible region. The LP does not have an optimal solution. It is *unbounded*.

2 Standard Form

Consider the following LP:

$$\begin{aligned} \min\ & 2x_1 - x_2 + 3x_3 - 2x_4 \\ \text{s.t. } & 3x_2 - x_3 + 4x_4 \geq 10 \\ & 4x_1 - 7x_2 + x_3 - x_4 = 5 \\ & x_1 - 3x_3 + 8x_4 \leq 3 \\ & x_1, x_4 \geq 0 \\ & x_2, x_3 \text{ Unrestricted in Sign (Free)} \end{aligned}$$

2-1 Problem Implement the instance and Julia/JuMP and solve it, printing the optimal solution and optimal solution value.

Answer:

Julia Code

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

# I made an array of variables, and index them as necessary
@variable(m, x[1:4])

@objective(m, Min, 2x[1] - x[2] + 3x[3] - 2x[4])
@constraint(m, x[1] >= 0)
@constraint(m, x[4] >= 0)
@constraint(m, 3x[2] - x[3] + 4x[4] >= 10)
@constraint(m, 4x[1] -7x[2] + x[3] - x[4] == 5)
@constraint(m, x[1] - 3x[3]+ 8x[4] <=3)
optimize!(m)
println("objective: ", objective_value(m))
println("solution: ", value.(x))  # dot operator on value() returns value of each entry in the vector
```

Gives

Julia Answer

```
objective: 15.937500000000004
solution:
[7.593750000000001, 3.84375, 1.5312500000000007, -0.0]
```

2-2 Problem Convert the problem to the standard form:

$$\begin{aligned} \max_{x \in \mathbb{R}^n} \quad & c^\top x \\ \text{subject to:} \quad & Ax \leq b \\ & x \geq 0 \end{aligned}$$

What are A, b, c,and x in this problem? Clearly indicate how the decision variables of your transformed LP relate to those of the original LP.

Answer:

First, identify the issues. This is a min problem, and be want it to be max. We have two free variables (x_2, x_3). We also have one nonpositive variable (x_4). There is an equality constraint and a $\geq$ constraint.

We'll start with the variables: Let $x_2 = v_2 - w_2$. Let $x_3 = v_3 - w_3$. Then x_1, v_2, w_2, v_3, w_3, and $x_4 \geq 0$. Replace everywhere x_2 and x_3 appear:

$$\begin{aligned}
\min 2x_1 - (v_2 - w_2) + 3(v_3 - w_3) - 2x_4 \\
\text{s.t. } 3(v_2 - w_2) - (v_3 - w_3) + 4x_4 \geq 10 \\
4x_1 - 7(v_2 - w_2) + (v_3 - w_3) - x_4 = 5 \\
x_1 - 3(v_3 - w_3) + 8x_4 \leq 3 \\
x_1, v_2, w_2, v_3, w_3, x_4 \geq 0
\end{aligned}$$

Now let's simplify and fix the objective:

$$\begin{aligned}
-\max -2x_1 + v_2 - w_2 - 3v_3 + 3w_3 + 2x_4 \\
\text{s.t.} \qquad 3v_2 - 3w_2 - v_3 + w_3 + 4x_4 \geq 10 \\
4x_1 - 7v_2 + 7w_2 + v_3 - w_3 - x_4 = 5 \\
x_1 \qquad - 3v_3 + 3w_3 + 8x_4 \leq 3 \\
x_1, v_2, w_2, v_3, w_3, x_4 \geq 0
\end{aligned}$$

Finally, fix the constraints:

$$\begin{aligned}
-\max -2x_1 + v_2 - w_2 - 3v_3 + 3w_3 + 2x_4 \\
\text{s.t.} \qquad -3v_2 + 3w_2 + v_3 - w_3 - 4x_4 \leq -10 \\
4x_1 - 7v_2 + 7w_2 + v_3 - w_3 - x_4 \leq 5 \\
-4x_1 + 7v_2 - 7w_2 - v_3 + w_3 + x_4 \leq -5 \\
x_1 \qquad - 3v_3 + 3w_3 + 8x_4 \leq 3 \\
x_1, v_2, w_2, v_3, w_3, x_4 \geq 0
\end{aligned}$$

And we're done! We have

$$x = \begin{bmatrix} x_1 \\ v_2 \\ w_2 \\ v_3 \\ w_3 \\ x_4 \end{bmatrix} \qquad A = \begin{bmatrix} 0 & -3 & 3 & 1 & -1 & -4 \\ 4 & -7 & 7 & 1 & -1 & -1 \\ -4 & 7 & -7 & -1 & 1 & 1 \\ 1 & 0 & 0 & -3 & 3 & 8 \end{bmatrix}$$

$$c = \begin{bmatrix} -2 & 1 & -1 & -3 & 3 & 2 \end{bmatrix}, \quad b = \begin{bmatrix} -10 \\ 5 \\ -5 \\ 3 \end{bmatrix}.$$

Note we also have $x_2 = v_2 - w_2$ and $x_3 = v_3 - w_3$.

2-3 Problem Solve the standard-form LP in Julia and report the objective value and the value of each decision variable in an optimal solution to the original LP.

Answer:

Julia Code

```
# Here I will do a "matrix form" of the LP,
# showing off some more of Julia and Jump's operators,
# especially for those who like matlab

#standard form
A = [0 -3 3 1 -1 -4; 4 -7 7 1 -1 -1 ; -4 7 -7 -1 1 1; 1 0 0 -3 3 8]
c = [-2; 1; -1; -3; 3; 2]
b= [-10; 5; -5; 3]

using JuMP, HiGHS
m = Model(HiGHS.Optimizer)
@variable(m, x[1:6] >= 0)
@objective(m, Max, c'*x)
@constraint(m, A*x .<= b)  # Note the '.' makes entrywise <=
optimize!(m)

# Here I report the translated values!  So we should get exactly the same answer
println("x1 = ", value(x[1]))
println("x2 = ", value(x[2])-value(x[3]))
println("x3 = ", value(x[4])-value(x[5]))
println("x4 = ", value(x[6]))
println("objective: ", -(objective_value(m)))
```

Gives matching solution:

Julia Answer

```
x1 = 7.593750000000002
x2 = 3.84375
x3 = 1.5312500000000007
x4 = 0.0
objective: 15.937500000000005
```

3 Mmmmmmmmmmmmmmmm. Pork.

Jacob's Meat Packing Plant produces 480 hams, 400 pork bellies, and 230 picnic hams every day. Each of these products can be sold either fresh or smoked. The total number of hams, bellies, and picnics that can be smoked during a normal working day is 420. In addition, up to 250 products can be smoked on overtime, at a higher cost (and therefore a lower net profit). The net profits (in $ per product) are the following:

	Fresh	Smoked (Regular)	Smoked (Overtime)
Hams	8	14	11
Bellies	4	12	7
Picnics	4	13	9

3-1 Problem Write a linear program that will maximize the total net profit.

Answer:
Define the following decision variables

- h_f: # of hams to sell fresh each day
- h_r: # of hams to smoke in regular time each day
- h_o: # of hams to smoke in overtime each day
- b_f: # of bellies to sell fresh each day
- b_r: # of bellies to smoke in regular time each day
- b_o: # of bellies to smoke in overtime each day
- p_f: # of picnics to sell fresh each day
- p_r: # of picnics to smoke in regular time each day
- p_o: # of picnics to smoke in overtime each day

The the following linear programming instance maximize the daily profit.

$$\max 8h_f + 14h_r + 11h_o + 4b_f + 12b_r + 7b_o + 4p_f + 13p_r + 9p_o.$$

$$\begin{aligned}
\text{s.t.} \qquad h_f + h_r + h_o &\leq 480 \\
b_f + b_r + b_o &\leq 400 \\
p_f + p_r + p_o &\leq 230 \\
h_r + b_r + p_r &\leq 420 \\
h_o + b_o + p_o &\leq 250 \\
h_f, h_r, h_o, b_f, b_r, b_o, p_f, p_r, p_o &\geq 0
\end{aligned}$$

3-2 Problem Create your instance from Problem 3-1 in the JuMP modeling language and solve it, printing the profit and number of each type of meat product to produce.

Answer:

Julia Code

```
using JuMP, Clp

m = Model(Clp.Optimizer)
@variable(m, hf >= 0)
@variable(m, hr >= 0)
@variable(m, ho >= 0)
@variable(m, bf >= 0)
@variable(m, br >= 0)
@variable(m, bo >= 0)
@variable(m, pf >= 0)
@variable(m, pr >= 0)
@variable(m, po >= 0)

@objective(m, Max, 8*hf + 14*hr + 11*ho + 4*bf + 12*br + 7*bo + 4*pf + 13*pr + 9*po)

@constraint(m, hf + hr + ho <= 480)
@constraint(m, bf + br + bo <= 400)
@constraint(m, pf + pr + po <= 230)
@constraint(m, hr + br + pr <= 420)
@constraint(m, ho + bo + po <= 250)

optimize!(m)

println("total profit: ", objective_value(m))
println("Fresh Hams: ", value(hf))
println("Hams Smoked (Regular) ", value(hr))
println("Hams Smoked (Overtime) ", value(ho))
println("Fresh Bellies: ", value(bf))
println("Bellies Smoked (Regular) ", value(br))
println("Bellies Smoked (Overtime) ", value(bo))
println("Fresh Picnics: ", value(pf))
println("Picnics Smoked (Regular) ", value(pr))
println("Picnics Smoked (Overtime) ", value(po))
```

Julia Answer

```
total profit: 10910.0
Fresh Hams: 440.0
Hams Smoked (Regular) 0.0
Hams Smoked (Overtime) 40.0
Fresh Bellies: 0.0
Bellies Smoked (Regular) 400.0
Bellies Smoked (Overtime) 0.0
Fresh Picnics: 0.0
Picnics Smoked (Regular) 20.0
Picnics Smoked (Overtime) 210.0
```

4 Mission Impossible

```
Lisa:  ''I'm going to become a vegetarian''
Homer:  ''Does that mean you're not going to eat any pork?''
Lisa:  ''Yes''
Homer:  ''Bacon?''
Lisa:  ''Yes Dad''
Homer:  ''Ham?''
Lisa:  ''Dad all those meats come from the same animal!''
Homer:  ''Right Lisa, some wonderful, magical animal!''
```

Jacob's Meat Alternative Plant (`jmap`) produces a set of plant-based pork products P using Impossible Meat. (For example, it may be that $P = \{\text{Ham, Bacon, Pork Belly}\}$.) A product $p \in P$ can be prepared (and then sold) using a cooking method $c \in C$. (For example, it may be that $C = \{\text{Fresh, Smoked, Fried}\}$.) `jmap` produces b_p products of type $p \in P$ per day. The total number of products that can be prepared using cooking method $c \in C$ each day is given by the parameter u_c. The net revenue to `jmap` of product $p \in P$ when prepared by cooking method $c \in C$ is h_{pc}. Each cooking method $c \in C$ has an associated cost of q_c for any product sold using that method.

4-1 Problem Write a linear program that will maximize the total net profit for `jmap`. Just write the mathematical model using general notation.

Answer:
Let x_{pc} be the number of pork product $p \in P$ to cook with cooking method $c \in C$ on each day. Then a linear program that will maximize the profit is the following:

$$\max \sum_{p \in P} \sum_{c \in C} (h_{pc} - q_c) x_{pc}$$

$$\begin{aligned} \text{s.t.} \quad & \sum_{c \in C} x_{pc} \leq b_p \quad && \forall p \in P \\ & \sum_{p \in P} x_{pc} \leq u_c \quad && \forall c \in C \\ & x_{pc} \geq 0 \quad && \forall p \in P, c \in C \end{aligned}$$

4-2 Problem Implement your instance from Problem 4-1 in JuMP on the data in the data file `pork-general.csv`. You can either re-type the data, or use Julia DataFrames and CSV to build.

If you wish, you can use this code to help read the data from the CSV file:

Julia Code

```julia
# You may need to Pkg.add() these
using DataFrames, CSV, NamedArrays
df = CSV.read("pork-general.csv", DataFrame, delim =',')

cook_type = propertynames(df)[3:end]
product = convert(Array, df[3:end,1])
```

```

max_daily_product = Dict(zip(product,df[3:end,2]))
max_daily_cook = Dict(zip(cook_type,df[1,3:end]))
proc_cost = Dict(zip(cook_type,df[2,3:end]))

h = Matrix(df[3:end,3:end])
h_NA = NamedArray(h, (product, cook_type), ("Product", "Cook Type")) ;
```

Answer:

Julia Code

```
using DataFrames, CSV, NamedArrays
df = CSV.read("pork-general.csv", DataFrame, delim =',')

cook_type = propertynames(df)[3:end]
product = convert(Array, df[3:end,1])

max_daily_product = Dict(zip(product,df[3:end,2]))
max_daily_cook = Dict(zip(cook_type,df[1,3:end]))
proc_cost = Dict(zip(cook_type,df[2,3:end]))

h = Matrix(df[3:end,3:end])
h_NA = NamedArray(h, (product, cook_type), ("Product", "Cook Type")) ;


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

# Number of product p to process using cooking method c
@variable(m, x[product, cook_type] >= 0)

# Objective
@expression(m, profit, sum( (h_NA[p,c]-proc_cost[c])*x[p,c] for p in product, c in cook_type))
@objective(m, Max, profit)

# Do not exceed the max daily product for p
@constraint(m, [p in product], sum(x[p,c] for c in cook_type) <= max_daily_product[p])

# Do not exceed max daily cook
@constraint(m, [c in cook_type], sum(x[p,c] for p in product) <= max_daily_cook[c])

optimize!(m)
for p in product
    for c in cook_type
        if value(x[p,c]) > 0.01
            println("Process ", value(x[p,c]), " of product ", p, " using cooking method ", c)
        end
    end
end
```

```
println("Max total profit: ", objective_value(m))
```

Julia Answer

```
Process 380.0 of product Ham using cooking method Fresh
Process 100.0 of product Ham using cooking method Cured
Process 230.0 of product PorkBelly using cooking method SmokedRegular
Process 170.0 of product PorkBelly using cooking method Pickled
Process 230.0 of product Picnic using cooking method Pickled
Process 310.0 of product Bacon using cooking method Fresh
Process 190.0 of product Bacon using cooking method SmokedRegular
Process 245.0 of product Tenderloin using cooking method SmokedOvertime
Max total profit: 14725.0
```

5 Optiyours and Optimine

A mining company ("OptiMine") is planning mine operations for the next week. There is a set I of mine locations available for ore extraction. Each mine location $i \in I$ has u_i tons of ore available. Mining at each location $i \in I$ has an associated cost per ton c_i. Finally, there is a set of elements E, and each mine location $i \in I$ has a different percentage of each type of element. The value α_{ei} is the percentage of element $e \in E$ in the ore mined from location $i \in I$. The Optimine company must mine at least K tons of ore total, and each element $e \in E$ must have at least a percentage β_e.

5-1 Problem Formulate a linear program to help OptiMine plan mining operations to meet requirements at a minimal cost. Give a general form (parameters only – no numbers) of the math model.

Answer:
Define the decision variables x_i as the tons of ore mined from location $i \in I$. Then we can write a linear program to help OptiMine plan mining operations to meet requriments at a minimal cost as the following:

$$
\begin{aligned}
\min \quad & \sum_{i \in I} c_i x_i \\
\text{s.t.} \quad & \sum_{i \in I} x_i \geq K \\
& \sum_{i \in I} \alpha_{ei} x_i \leq \beta_e \sum_{i \in I} x_e \quad \forall e \in E \\
& 0 \leq x_i \leq u_i \qquad \forall i \in I
\end{aligned}
$$

5-2 Problem Here is ChaptGPT (v4.0's) answer for this problem, when I asked to write it in Julia and JuMP:

Julia Code

```
    using JuMP
    using GLPK # Or any other LP solver

```

```
    # Define the data structures
    I = # Set of mine locations
    E = # Set of elements
    u = # Dictionary mapping each location i ∈ I to the available tons of ore u_i
    c = # Dictionary mapping each location i ∈ I to the cost per ton c_i
    α = # Dictionary mapping each (element, location) pair (e, i) to the percentage α_{ei}
    β = # Dictionary mapping each element e ∈ E to the required minimum percentage β_e
    K = # Minimum total tons of ore to be mined

    # Create a new model
    model = Model(GLPK.Optimizer)

    # Decision variables
    @variable(model, x[i in I] >= 0) # Tons of ore to mine from each location

    # Objective: Minimize total cost
    @objective(model, Min, sum(c[i] * x[i] for i in I))

    # Constraint: Mine at least K tons of ore in total
    @constraint(model, sum(x[i] for i in I) >= K)

    # Constraints: Ensure the percentage of each element
    for e in E
        @constraint(model, sum(α[(e, i)] * x[i] for i in I) / sum(x[i] for i in I) >= β[e])
    end

    # Solve the model
    optimize!(model)

    # Check results
    if termination_status(model) == MOI.OPTIMAL
        println("Optimal solution found")
        for i in I
            println("Mine ", i, " - Tons of ore: ", value(x[i]))
        end
    else
        println("No optimal solution found")
    end
```

Is this a correct linear programming model for the problem? If it is not correct, please explain why and fix the model. If it is a correct model, please comment on any differences between your mathematical model and ChatGPT (v4.0)'s Julia/JuMP implementation.

Answer:

This is *not* a linear programming model. The "Ensure the percentage of each element" constraint is *not written in a linear fashion.* Using the fact that we know that $\sum_{i \in I} x_i \geq K > 0$, we can multiply both slides of ChaptGPT's inequality by $\sum_{i \in I} x_i \geq K > 0$ and get an equivalent linear inequality. Its implementation in Jump should look like this:

Julia Code

```
for e in E
    @constraint(model, sum(α[(e, i)] * x[i] for i in I) >= β[e] * sum(x[i] for i in I))
end
```

5-3 Problem Implement and solve this instance of the model in Julia/JuMP. Display the optimal objective value and the optimal variable values.

The cost, ore available, tons of each element required, and percentage of elements in the ore at each mining location are given in the .csv file "mine.csv." A sample of the data is given in the table below. There is a minimum percentage of each element required across the total ore extracted, also given in the .csv file.

The data is provided in the "optimine.csv" on Canvas. You can use the code snippet provided below to read the .csv files and load them into Julia.

Julia Code

```
#You might need to run "Pkg.add(...)" before using these packages
using DataFrames, CSV, NamedArrays

#Load the CSV data file (should be in same directory as notebook)
df = CSV.read("optimine.csv",DataFrame,delim=',');

# create a list of mines
mines = convert(Array,df[2:end,1])

# create a list of elements
#  here we take from the DataFrame header (into Julia Symbol)
elements = propertynames(df)[2:6]

# create a dictionary of the total cost of mining at each location
c = Dict(zip(mines,df[2:end,7]))

# create a dictionary of the max tons available at each location
u = Dict(zip(mines,df[2:end,8]))

# create a dictionary of the amount required of each element
β = Dict(zip(elements,df[1,2:6]))

# create a matrix of the % of each element at each loation
mine_element_matrix = Matrix(df[2:end,2:6])

# rows are mines, columns are elements
α = NamedArray(mine_element_matrix, (mines, elements), ("mines","elements"))

K = 3000 ;
```

Answer:

Here is my Julia Code

Julia Code

```
#You might need to run "Pkg.add(...)" before using these packages
using DataFrames, CSV, NamedArrays

#Load the data file
df = CSV.read("optimine.csv",DataFrame,delim=',');

# create a list of mines
mines = convert(Array,df[2:end,1])

# create a list of elements,
#elements = 1:5
elements = propertynames(df)[2:6]

# create a dictionary of the total cost of mining at each location
c = Dict(zip(mines,df[2:end,7]))

# create a dictionary of the max tons available at each location
u = Dict(zip(mines,df[2:end,8]))

# create a dictionary of the amount required of each element
β = Dict(zip(elements,df[1,2:6]))

# create a matrix of the % of each element at each loation
mine_element_matrix = Matrix(df[2:end,2:6])

# rows are mines, columns are elements, here the names/indices are just integers, but they don't have to be
α = NamedArray(mine_element_matrix, (mines, elements), ("mines","elements"))

# We are required to mine K = 3000 tons
K = 3000
;
using JuMP, HiGHS

m = Model(HiGHS.Optimizer)

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

@objective(m, Min, sum(c[i]*x[i] for i in mines)) # minimize costs

# Data was written intranspose format, so get subscripts right here.
@constraint(m, meet_req[e in elements], sum(α[i,e]*x[i] for i in mines) >=  β[e] * sum(x[i] for i in mines)
@constraint(m, max_con[i in mines], x[i] <= u[i]) # don't exceed tons available
@constraint(m, dem_con, sum(x[i] for i in mines) >= K) # meet total demand

set_silent(m)  # You can turn off HiGHS output if you want
optimize!(m)

println("Solution: mine at each location as follows:")
for i in mines
    @printf("%s: %.4f\n", i, value(x[i]))
```

```
    end
    @printf("This has a cost of: %.2f\$", objective_value(m))
```

Which gives the following output:

Julia Answer

```
Solution: mine at each location as follows:
1: 254.6667
2: 628.0000
3: 117.0000
4: 547.0000
5: 0.0000
6: 0.0000
7: 0.0000
8: 737.0000
9: 111.0000
10: 605.3333
This has a cost of: 62556.33$
```