CS/ECE/ISyE 524 — Introduction to Optimization — Spring 2021

Final Course Project: Due 5/2/21, 12:05pm

Course Schedular

Yasmine Abdennadher (abdennadher@wisc.edu), Hussain Alkhayat (halkhayat@wisc.edu), Omar Zawawi (ozawawi@wisc.edu)

Table of Contents

1. Introduction

With the semester coming to an end, many students find themselves scrambling to find the best schedule that will allow them to complete their schoolwork while still having some free time to enjoy the nice weather outside. Which is why we have decided to use optimization techniques to find the best work schedule for a student with multiple deadlines. To ensure our solution is as optimal as possible, we will be either maximizing the summed priority of the tasks or minimizing added time throughout the whole week.

To achieve this, we have created five main tasks with the following characteristics:

```
-End time: This represents the deadline of each task. This
parameter is represented using an integer
(Monday = 1...Sunday = 7)

-Duration: This is the estimated amount of minutes needed
to complete a task

-Difficulty: Each task will be rated on a difficulty from
1-3. This is
initialized by the user

-Priority: This will be a dynamic parameter that changes
from day to day based on the difficulty
rating of the task and the deadline.

-Room for error: Sometimes, certain tasks will take longer
than expected. Room for error means to
calculate extra hours that can be spent on a task each
day. This variable is calculated based on
difficulty and the total length of the task itself
```

We also created a list of miscellaneous tasks with smaller durations meant to pop up throughout the week. Such tasks can be: going to the gym, grocery shopping, returning a package, and so on. These tasks will have similar characteristics as above and will disappear from the task array at the end of the day.

The rest of the report will go as follows. Section 2 will introduce the different models we plan on presenting and describe various aspects within them. Section 3 will hold the primary code of both models and the optimal schedules, and section 4 will highlight the difference between the two models and draw a conclusion.

2. Mathematical Models

2. Mathematical Model

DAY BY DAY MODEL

Combining all the above descriptions together we aim to have the following model:

For this project our primary goal will be completing the following deadlines:

```
- CS 524 Final Project
- Reading Summary
- Calc 2 Homework
- Essay
- Research
- DepenedTask
    - This task depends on other tasks.
    (i.e. This task CANNOT be started before finishing the
tasks it depends on)
```

With one of the following as our pop up chores each day:

```
- Gym day 1
- Gym day 2
- Gym day 3
- Meal Prep
- File Taxes
- Groceries
- Cleaning
```

For day one, New Tasks is represented as all the major tasks plus one of the chores. Then for all following days New Tasks will be one of the chores while Rolling Tasks will be the unfinished tasks from the previous days.

As stated in the introduction each task will have a list of characteristics that are calculated either at the beginning of the week or are updated day by day. Room for error for example will be a static characteristic that is only initilized at the beginning; whereas completness and priority is updated day by day to determine the best tasks to complete each day. These calculations are as follow:

- $\text{RoomForError} = \text{Difficulty} * 0.1$
- $\text{Completness} = \frac{\text{Curation} - \text{Completed hours for the task}}{\text{Duration}}$
- $Priority = \frac{DeadlineFactor + (Duration - Completeness)}{(EndTime - CurrentDay + 1) * 0.1}$

- where Deadline factor is a decimal that varies for each task day by day based on how far away the deadline is to the current day

The main objective of this model is to maximize the priority each day; in other words, we want to ensure that we are completing the tasks with the highest priority first and then move on to the next.

$$\max \sum_{i=1}^{7} \sum_{j=10}^{22} P_i T_{t[i]j}$$

Where i is the task index and j is the time slot within the day (in the TIMES array)

To achieve this we implement the following constraints:

- $If\ deadline\ is\ tonight\ for\ a\ task$
  - $Time\ spent\ on\ specific\ task \leq 6hours$
- $Otherwise$
  - $Time\ spent\ on\ specific\ task \leq 3hours$
- $Amount\ of\ hours\ put\ in\ task\ today + Completed\ Hours \leq Duration + Room\ for\ Error$
- $Sum\ of\ task\ at\ each\ time \leq 1$
- $Sum\ of\ hours\ worked\ in\ a\ day = 10$
- $A\ dependent\ task\ can\ only\ start\ if\ the\ tasks\ that\ it\ depends\ on\ are\ completed$

To further explain the above constraints we have added a description for each one bellow

- We first introduce the Task variable, which will be a binary matrix that is of the size $[\text{i in Tasks,j in TIMES}]$ The TIMES array will look as follows:

$$T_{t[i]j} \in \{0,1\} i = 1, \ldots, 7 j = TIMESarray$$

T will look like this:

Tasks\TIMES	10:00-10:15	10:15-10:30	10:30-10:45	...	21:45-22:00
CS 524 Project	0	0	0	...	0
Reading Summary	0	0	1	...	0
Calc 2 HW	0	0	0	...	0
Essay	1	1	0	...	0
Research	0	0	0	...	1
DependTask	0	0	0	...	0

- Then we begin laying out our constraints: If a deadline is not due this current day then the time required to work of the specified task is maximum six hours if the task is due tomorrow or three hours if it is due any other day. Otherwise, if the task is due this day then there will be no constraint on the hours spent on the task

$$\sum_{i=\text{task 1}}^{\text{task 7}} \begin{cases} \begin{cases} \sum_{i=10}^{22} T_{i,j} \leq 6 & E_i - D = 1 \\ \sum_{i=10}^{22} T_{i,j} \leq 4 & \text{otherwise} \end{cases} & E_i - D \neq 0 \\ \sum_{i=10}^{22} T_{i,j} > 0 & \text{otherwise} \end{cases}$$

The amount of hours spent on a task each day plus the completed hours in the previous day must be less than or equal to the total duration of the task plus the initialized room for error.

$$\sum_{i=\text{task 1}}^{\text{task 7}} (\sum_{i=10}^{22} T_{i,j})0.25 + C_i \leq D_i + (D_i R)$$

$$\text{where D} = \text{day, C} = \text{completness of the task, and R} = \text{room for error}$$

During each time period there must at most one task within that slot

$$\sum_{i=\text{task 1}}^{\text{task 7}} \sum_{i=10}^{22} T_{ij} \leq 1$$

Each day there are ten total work hours

$$\sum_{i=10}^{22} T_{:j} \leq 10$$

A dependent task can only start if the tasks that it depends on are completed. An array called Dependecies will be initiliazed in the beginning that will hold a list of indicies that point out which tasks are dependent on (Example: Dependinces$[6]$ will print out an list of integers $[2, 3]$. This means that Task 6 is dependent on Task 2 and 3).

FULL WEEK MODEL

Using the same tasks as the previous model, minus the dependency task, this model solves the full week in one go. The primary opjective is to minimize the total time

spent on each task:

$$\min \sum_{j=1}^{7} \sum_{i=1}^{6} T_{ij}$$

where i represents the tasks and j represents the days in the week

Next we implement the following variable:

$$T_{ij} \geq 0 i = 1, \ldots, 6 j = 1, \ldots, 7$$

Where each slot in the array will hold amount of hours spent on each task thay

With the following constraints:

- $Time\ spent\ on\ a\ task\ is \geq Duration + Room\ for\ Error$
- $Total\ time\ spent\ on\ all\ tasks\ each\ day \leq 10$
- $If\ a\ task\ takes\ more\ than\ 3\ days$
  - $Time\ spent\ on\ the\ task\ each\ day \leq 6$
- $otherwise$
  - $Time\ spent\ on\ the\ task\ each\ day \leq 3$

The result of this optimization will provide the hours spent on each task each day. Which we will then push into a function to include breaks every three hours and print out a nice model.

3. Code

In [1]:
```
using Pkg
using PyCall
np = pyimport("numpy")
using NamedArrays, JuMP, Gurobi
```

DAY BY DAY MODEL

Initilizing Tasks

```
- We will begin my initializing our main tasks and
creating the different parameters.
```

In [23]:
```
Tasks = ["CS524Project", "ReadingSummary", "Calc2", "Essay", "Research",
#Tasks that depend on other, 0 means it is independent and any value repr
Dependecies = [[0],[0],[0],[0],[0],[2,3]]

#Each of the lists below take the same index ordering as in the Tasks lis
EndTime = [6,2, 5,7, 2,7] #Deadline by day
Duration = [10,3,3,16,2,2] #Duration in hours
difficulty = [3,1,2,2,1,3] #1-3 Higher is more difficult
Deadlinepriority = [0.7, 0.6, 0.5, 0.4, 0.3, 0.2, 0.1] #index 1 = initial
            #if its EndTime is 0 days away from deadline. Index 2 = 1 day
priority = []
RoomForError = []
completnessPerc = np.zeros(length(Tasks))
completnessHours = np.zeros(length(Tasks))
completnessHours_error = np.zeros(length(Tasks))

#initilizing the fields that depend on a formula
for i in 1:length(Tasks)
    append!(priority, Deadlinepriority[EndTime[i]-1+1] + Duration[i]/(End
    append!(RoomForError, difficulty[i]*0.1)
end
# Checking if the values are correctly implemented
for i in 1:length(Tasks)
   println("Tasks: ", Tasks[i], " | EndTime: ", EndTime[i], " | Duration:
end


TIMES = ["10:00-10:15", "10:15-10:30", "10:30-10:45", "10:45-11:00", "11:
println()
```

```
Tasks: CS524Project | EndTime: 6 | Duration: 10 | Difficulty: 3 | Priorit
y: 0.34285714285714286 | RoomForError: 0.30000000000000004
Tasks: ReadingSummary | EndTime: 2 | Duration: 3 | Difficulty: 1 | Priori
ty: 0.7 | RoomForError: 0.1
Tasks: Calc2 | EndTime: 5 | Duration: 3 | Difficulty: 2 | Priority: 0.35
| RoomForError: 0.2
Tasks: Essay | EndTime: 7 | Duration: 16 | Difficulty: 2 | Priority: 0.30
000000000000004 | RoomForError: 0.2
Tasks: Research | EndTime: 2 | Duration: 2 | Difficulty: 1 | Priority: 0.
6666666666666666 | RoomForError: 0.1
Tasks: DepenTask | EndTime: 7 | Duration: 2 | Difficulty: 3 | Priority: 0
.125 | RoomForError: 0.30000000000000004
```

Aditional Functions

This function prints out a nice output of the schedule in the following manner (time interval : Task/Break)

In [24]:
```
function printModel(TIMES, output)
    for i in 1:length(TIMES)
        T = 1
        for j in 1:length(Tasks)
            if output[j,i] == 1
                println( TIMES[i]," ", Tasks[j])
                T = 0
            end
        end
        if T == 1
            println(TIMES[i]," Break")
        end
    end

    println()
end
```

Out[24]:
```
printModel (generic function with 1 method)
```

This function takes in the result of the optimization problem, records how many 15-minute intervals put for each task, enforces an hour break for every 12 15-minute intervals of working (4 hours), and returns an organized result

In [25]:
```
# This function takes in the result of the optimization problem,
# records how many 15-minute intervals put for each task,
# enforces an hour break for every 12 15-minute intervals of working (4 h
#returns an organized result
function organize_model(output,TIMES,Tasks, weekModel)

    intervals_per_task = []
    result_matrix = np.zeros((length(Tasks), length(TIMES)))

    # retrieves the amount of 15 minute intervals put for a task
    if weekModel == false
        for i in Tasks
            append!(intervals_per_task, sum(output[i,:]))
        end
    else
        for i in 1:length(Tasks)
            append!(intervals_per_task, output[i])
        end
    end


    index = 1 #last index stopped at
    contIntervalsWorked = 0 #will track how many conitnous hours of worki
    for i in 1:length(Tasks)
        length = intervals_per_task[i] #length of intervals used for this
        if length == 0
            continue
        end
        for t in 1:length
            if contIntervalsWorked < 12 #can work more before break
                result_matrix[i, index] = 1
                index +=1
                contIntervalsWorked += 1
            else
                index += 4 # add an hour break
                if index > 48
                    return result_matrix
                end
                result_matrix[i, index] = 1
                contIntervalsWorked = 0
                index+=1
            end
        end
    end

    return result_matrix
end
```

Out[25]:
```
organize_model (generic function with 1 method)
```

Optimization Problem

- This main cell will hold a seven day for loop that solves the optmization problem each day.

In [26]:
```
# Extra chores that need to be done within the week. Each day, we'll rand
# Each of those chores take 2 hours.
chores = ["Gym1", "Meal Prep", "File Taxes", "Groceries", "Gym2", "Gym3",

for day in 1:7

    ## ADD Daily/chores tasks and update var "Task" "Duration" "compltnes
    chore = rand(1:length(chores))
    append!(Tasks, [chores[chore]])
    append!(EndTime, day)
    append!(Duration, 2)
    append!(priority, 1 ) #to enfore getting them done on the same day (W
    append!(RoomForError, 0.0)
    append!(completnessHours,0)
    append!(completnessHours_error,0)
    append!(completnessPerc,0)


    m = Model(Gurobi.Optimizer)
    set_optimizer_attribute(m, "OutputFlag", 0)
    println("DAY: ", day)


    #variables
    @variable(m, Task[i in Tasks,j in TIMES], Bin)

    #constraints
    for i in 1:length(Tasks)
        #if deadline is not today
        if EndTime[i] - day != 0
            #if deadline is tomorrow, allow to work on task at most 6hrs
            if EndTime[i] - day == 1
                @constraint(m, sum(Task[Tasks[i],j] for j in TIMES) <= 6*
            else
                #work on tasks maximum of 3 hours
                @constraint(m, sum(Task[Tasks[i],j] for j in TIMES) <= 3*
            end
        end

    end

    @constraint(m, sum(Task[Tasks[length(Tasks)],TIMES[j]] for j in 41:48


    # amount of hours put in task today + amount of hours put before <= t
    @constraint(m, cons2[i in 1:length(Tasks)], sum(Task[Tasks[i],j] for
        <= Duration[i] + RoomForError[i]*Duration[i])

    @constraint(m, cons3[j in TIMES], sum(Task[i,j] for i in Tasks) <= 1)

    @constraint(m, sum(Task) <= 10*4) #total amount of work hours in a da

    #Dependnecies
    for i in 1:length(Dependecies) # loops thorugh each task
        if Dependecies[i][1] != 0 #check if it is an independent task or
            for j in 1:length(Dependecies[i]) # loop throught the task th
```

```
                if completnessPerc[Dependecies[i][j]] < 1 # check if task
                    DependableTaskIndex = i
                    TaskToBeDependantOn = Dependecies[i][j]
                    @constraint(m, sum(Task[Tasks[DependableTaskIndex],k]
                end
            end
        end
    end

    #maximize priority by working the most on important tasks
    @objective(m, Max, sum(priority[i]*Task[Tasks[i], j] for i in 1:lengt

    optimize!(m)

    output = getvalue.(Task)

    organizedModel = organize_model(output,TIMES,Tasks, false)
    printModel(TIMES,organizedModel)


    println("Amount of hours to spend on each task today")
    #UPDATE THE VARIABLES THAT WE HAVE AS A FUNCTION
    for j in 1:length(Tasks)
        sum1 = sum(getvalue.(Task[Tasks[j], :]))
        println(Tasks[j], ": ", sum1*0.25)
        if completnessPerc[j] < 1
            completnessHours_error[j] = completnessHours_error[j] + sum1*
            completnessHours[j] = completnessHours[j] + sum1*0.25
            completnessPerc[j] = (completnessHours[j])/(Duration[j]+(Room
            priority[j] = Deadlinepriority[EndTime[j]-day+1] + (Duration[
        end
    end
    pop!(Tasks)
    pop!(EndTime)
    pop!(Duration)
    pop!(priority)
    pop!(RoomForError)
    pop!(completnessHours)
    pop!(completnessHours_error)
    splice!(chores,chore)
end
```

```
-------------------------------------------
Warning: your license will expire in 5 days
-------------------------------------------

Academic license - for non-commercial use only - expires 2021-05-06
DAY: 1
10:00-10:15 ReadingSummary
10:15-10:30 ReadingSummary
10:30-10:45 ReadingSummary
10:45-11:00 ReadingSummary
11:00-11:15 ReadingSummary
11:15-11:30 ReadingSummary
11:30-11:45 ReadingSummary
11:45-12:00 ReadingSummary
12:00-12:15 ReadingSummary
```

```
12:15-12:30 ReadingSummary
12:30-12:45 ReadingSummary
12:45-13:00 ReadingSummary
13:00-13:15 Break
13:15-13:30 Break
13:30-13:45 Break
13:45-14:00 Break
14:00-14:15 ReadingSummary
14:15-14:30 Calc2
14:30-14:45 Calc2
14:45-15:00 Calc2
15:00-15:15 Calc2
15:15-15:30 Calc2
15:30-15:45 Calc2
15:45-16:00 Calc2
16:00-16:15 Calc2
16:15-16:30 Calc2
16:30-16:45 Calc2
16:45-17:00 Calc2
17:00-17:15 Research
17:15-17:30 Break
17:30-17:45 Break
17:45-18:00 Break
18:00-18:15 Break
18:15-18:30 Research
18:30-18:45 Research
18:45-19:00 Research
19:00-19:15 Research
19:15-19:30 Research
19:30-19:45 Research
19:45-20:00 Research
20:00-20:15 File Taxes
20:15-20:30 File Taxes
20:30-20:45 File Taxes
20:45-21:00 File Taxes
21:00-21:15 File Taxes
21:15-21:30 File Taxes
21:30-21:45 Break
21:45-22:00 Break

Amount of hours to spend on each task today
CS524Project: -0.0
ReadingSummary: 3.25
Calc2: 2.75
Essay: -0.0
Research: 2.0
DepenTask: 0.0
File Taxes: 2.0

-------------------------------------------
Warning: your license will expire in 5 days
-------------------------------------------

Academic license - for non-commercial use only - expires 2021-05-06
DAY: 2
10:00-10:15 CS524Project
10:15-10:30 CS524Project
```

```
10:30-10:45 CS524Project
10:45-11:00 CS524Project
11:00-11:15 CS524Project
11:15-11:30 CS524Project
11:30-11:45 CS524Project
11:45-12:00 CS524Project
12:00-12:15 CS524Project
12:15-12:30 CS524Project
12:30-12:45 CS524Project
12:45-13:00 CS524Project
13:00-13:15 Break
13:15-13:30 Break
13:30-13:45 Break
13:45-14:00 Break
14:00-14:15 Calc2
14:15-14:30 Calc2
14:30-14:45 Calc2
14:45-15:00 Essay
15:00-15:15 Essay
15:15-15:30 Essay
15:30-15:45 Essay
15:45-16:00 Essay
16:00-16:15 Essay
16:15-16:30 Essay
16:30-16:45 Essay
16:45-17:00 Essay
17:00-17:15 Essay
17:15-17:30 Break
17:30-17:45 Break
17:45-18:00 Break
18:00-18:15 Break
18:15-18:30 Essay
18:30-18:45 Essay
18:45-19:00 Groceries
19:00-19:15 Groceries
19:15-19:30 Groceries
19:30-19:45 Groceries
19:45-20:00 Groceries
20:00-20:15 Groceries
20:15-20:30 Groceries
20:30-20:45 Groceries
20:45-21:00 Break
21:00-21:15 Break
21:15-21:30 Break
21:30-21:45 Break
21:45-22:00 Break

Amount of hours to spend on each task today
CS524Project: 3.0
ReadingSummary: -0.0
Calc2: 0.75
Essay: 3.0
Research: -0.0
DepenTask: -0.0
Groceries: 2.0

-------------------------------------------
```

```
Warning: your license will expire in 5 days
------------------------------------------

Academic license - for non-commercial use only - expires 2021-05-06
DAY: 3
```

```
Invalid coefficient -Inf on variable Task[ReadingSummary,10:00-10:15].

Stacktrace:
 [1] error(::String) at ./error.jl:33
 [2] _assert_isfinite(::GenericAffExpr{Float64,VariableRef}) at /Users/ya
bdennadher/.julia/packages/JuMP/YXK4e/src/aff_expr.jl:330
 [3] MathOptInterface.ScalarAffineFunction(::GenericAffExpr{Float64,Varia
bleRef}) at /Users/yabdennadher/.julia/packages/JuMP/YXK4e/src/aff_expr.j
l:358
 [4] moi_function at /Users/yabdennadher/.julia/packages/JuMP/YXK4e/src/a
ff_expr.jl:364 [inlined]
 [5] set_objective_function(::Model, ::GenericAffExpr{Float64,VariableRef
}) at /Users/yabdennadher/.julia/packages/JuMP/YXK4e/src/objective.jl:110
 [6] set_objective(::Model, ::MathOptInterface.OptimizationSense, ::Gener
icAffExpr{Float64,VariableRef}) at /Users/yabdennadher/.julia/packages/Ju
MP/YXK4e/src/objective.jl:128
 [7] top-level scope at /Users/yabdennadher/.julia/packages/JuMP/YXK4e/sr
c/macros.jl:801
 [8] top-level scope at In[26]:67
 [9] include_string(::Function, ::Module, ::String, ::String) at ./loadin
g.jl:1091
```

In [6]:

```
for i in 1:length(Tasks)
    println(Tasks[i], ": ")
    println("Total amount of hours spent: ", completnessHours_error[i])
    println("Initial Prediction: ", Duration[i])
    println()
end
```

```
CS524Project:
Total amount of hours spent: 15.0
Initial Prediction: 10

ReadingSummary:
Total amount of hours spent: 3.5
Initial Prediction: 3

Calc2:
Total amount of hours spent: 4.0
Initial Prediction: 3

Essay:
Total amount of hours spent: 22.5
Initial Prediction: 16

Research:
Total amount of hours spent: 2.25
Initial Prediction: 2

DepenTask:
Total amount of hours spent: 3.0
Initial Prediction: 2
```

WEEK MODEL

In [7]:
```
m = Model(Gurobi.Optimizer)
set_optimizer_attribute(m, "OutputFlag", 0)

pop!(Tasks) #removing dependTask
pop!(RoomForError)

m = Model(Gurobi.Optimizer)
set_optimizer_attribute(m, "OutputFlag", 0)


@variable(m, Task[i in 1:length(Tasks),j in 1:7] >= 0)

#Duration and deadline contraint
@constraint(m, cons1[i in 1:length(Tasks)], sum(Task[i,j] for j in 1:7) >
@constraint(m, cons2[i in 1:7], sum(Task[:,i] ) <= 10) # max hours per da

for i in 1:length(Tasks)
    if Duration[i]/7 >= 3
        @constraint(m,[j in 1:7] ,Task[i,j]  <= 6) #duration cons
    else
        @constraint(m,[j in 1:7], Task[i,j]  <= 3) #duration cons
    end
end


@objective(m, Min, sum(Task[i, j] for i in 1:length(Tasks), j in 1:7))
optimize!(m)

output = getvalue.(Task)

dayResults = []
for day in 1:7
    for i in 1:length(Tasks)
        append!(dayResults, output[i,day]*4)
    end

    results = organize_model(dayResults, TIMES, Tasks, true)
    println("Day: ", day)
    printModel(TIMES, results)

    println("Amount of hours to spend on each task on day ", day)

    for j in 1:length(Tasks)
        println(Tasks[j], ": ", output[j, day])
    end
    println()
    dayResults = []
end
```

```
------------------------------------------
Warning: your license will expire in 5 days
------------------------------------------

Academic license - for non-commercial use only - expires 2021-05-06

------------------------------------------
```

```
Warning: your license will expire in 5 days
-------------------------------------------

Academic license - for non-commercial use only - expires 2021-05-06
Day: 1
10:00-10:15 CS524Project
10:15-10:30 CS524Project
10:30-10:45 ReadingSummary
10:45-11:00 ReadingSummary
11:00-11:15 ReadingSummary
11:15-11:30 ReadingSummary
11:30-11:45 ReadingSummary
11:45-12:00 ReadingSummary
12:00-12:15 ReadingSummary
12:15-12:30 ReadingSummary
12:30-12:45 ReadingSummary
12:45-13:00 ReadingSummary
13:00-13:15 Break
13:15-13:30 Break
13:30-13:45 Break
13:45-14:00 Break
14:00-14:15 ReadingSummary
14:15-14:30 ReadingSummary
14:30-14:45 Calc2
14:45-15:00 Calc2
15:00-15:15 Calc2
15:15-15:30 Calc2
15:30-15:45 Calc2
15:45-16:00 Calc2
16:00-16:15 Calc2
16:15-16:30 Calc2
16:30-16:45 Calc2
16:45-17:00 Calc2
17:00-17:15 Calc2
17:15-17:30 Break
17:30-17:45 Break
17:45-18:00 Break
18:00-18:15 Break
18:15-18:30 Calc2
18:30-18:45 Essay
18:45-19:00 Essay
19:00-19:15 Essay
19:15-19:30 Essay
19:30-19:45 Research
19:45-20:00 Research
20:00-20:15 Research
20:15-20:30 Research
20:30-20:45 Research
20:45-21:00 Research
21:00-21:15 Research
21:15-21:30 Research
21:30-21:45 Break
21:45-22:00 Break

Amount of hours to spend on each task on day 1
CS524Project: 0.6000000000000005
ReadingSummary: 3.0
```

```
Calc2: 3.0
Essay: 1.1999999999999993
Research: 2.2

Day: 2
10:00-10:15 CS524Project
10:15-10:30 CS524Project
10:30-10:45 CS524Project
10:45-11:00 CS524Project
11:00-11:15 CS524Project
11:15-11:30 CS524Project
11:30-11:45 CS524Project
11:45-12:00 CS524Project
12:00-12:15 CS524Project
12:15-12:30 CS524Project
12:30-12:45 CS524Project
12:45-13:00 CS524Project
13:00-13:15 Break
13:15-13:30 Break
13:30-13:45 Break
13:45-14:00 Break
14:00-14:15 ReadingSummary
14:15-14:30 Calc2
14:30-14:45 Calc2
14:45-15:00 Essay
15:00-15:15 Essay
15:15-15:30 Essay
15:30-15:45 Essay
15:45-16:00 Essay
16:00-16:15 Essay
16:15-16:30 Essay
16:30-16:45 Essay
16:45-17:00 Essay
17:00-17:15 Essay
17:15-17:30 Break
17:30-17:45 Break
17:45-18:00 Break
18:00-18:15 Break
18:15-18:30 Essay
18:30-18:45 Essay
18:45-19:00 Break
19:00-19:15 Break
19:15-19:30 Break
19:30-19:45 Break
19:45-20:00 Break
20:00-20:15 Break
20:15-20:30 Break
20:30-20:45 Break
20:45-21:00 Break
21:00-21:15 Break
21:15-21:30 Break
21:30-21:45 Break
21:45-22:00 Break

Amount of hours to spend on each task on day 2
CS524Project: 3.0
ReadingSummary: 0.2999999999999998
```

```
Calc2: 0.6000000000000001
Essay: 3.0
Research: 0.0

Day: 3
10:00-10:15 CS524Project
10:15-10:30 CS524Project
10:30-10:45 CS524Project
10:45-11:00 CS524Project
11:00-11:15 CS524Project
11:15-11:30 CS524Project
11:30-11:45 CS524Project
11:45-12:00 CS524Project
12:00-12:15 CS524Project
12:15-12:30 CS524Project
12:30-12:45 CS524Project
12:45-13:00 CS524Project
13:00-13:15 Break
13:15-13:30 Break
13:30-13:45 Break
13:45-14:00 Break
14:00-14:15 Essay
14:15-14:30 Essay
14:30-14:45 Essay
14:45-15:00 Essay
15:00-15:15 Essay
15:15-15:30 Essay
15:30-15:45 Essay
15:45-16:00 Essay
16:00-16:15 Essay
16:15-16:30 Essay
16:30-16:45 Essay
16:45-17:00 Essay
17:00-17:15 Break
17:15-17:30 Break
17:30-17:45 Break
17:45-18:00 Break
18:00-18:15 Break
18:15-18:30 Break
18:30-18:45 Break
18:45-19:00 Break
19:00-19:15 Break
19:15-19:30 Break
19:30-19:45 Break
19:45-20:00 Break
20:00-20:15 Break
20:15-20:30 Break
20:30-20:45 Break
20:45-21:00 Break
21:00-21:15 Break
21:15-21:30 Break
21:30-21:45 Break
21:45-22:00 Break

Amount of hours to spend on each task on day 3
CS524Project: 3.0
ReadingSummary: 0.0
```

```
Calc2: 0.0
Essay: 3.0
Research: 0.0

Day: 4
10:00-10:15 CS524Project
10:15-10:30 CS524Project
10:30-10:45 CS524Project
10:45-11:00 CS524Project
11:00-11:15 CS524Project
11:15-11:30 CS524Project
11:30-11:45 CS524Project
11:45-12:00 CS524Project
12:00-12:15 CS524Project
12:15-12:30 CS524Project
12:30-12:45 CS524Project
12:45-13:00 CS524Project
13:00-13:15 Break
13:15-13:30 Break
13:30-13:45 Break
13:45-14:00 Break
14:00-14:15 Essay
14:15-14:30 Essay
14:30-14:45 Essay
14:45-15:00 Essay
15:00-15:15 Essay
15:15-15:30 Essay
15:30-15:45 Essay
15:45-16:00 Essay
16:00-16:15 Essay
16:15-16:30 Essay
16:30-16:45 Essay
16:45-17:00 Essay
17:00-17:15 Break
17:15-17:30 Break
17:30-17:45 Break
17:45-18:00 Break
18:00-18:15 Break
18:15-18:30 Break
18:30-18:45 Break
18:45-19:00 Break
19:00-19:15 Break
19:15-19:30 Break
19:30-19:45 Break
19:45-20:00 Break
20:00-20:15 Break
20:15-20:30 Break
20:30-20:45 Break
20:45-21:00 Break
21:00-21:15 Break
21:15-21:30 Break
21:30-21:45 Break
21:45-22:00 Break

Amount of hours to spend on each task on day 4
CS524Project: 3.0
ReadingSummary: 0.0
```

```
Calc2: 0.0
Essay: 3.0
Research: 0.0

Day: 5
10:00-10:15 CS524Project
10:15-10:30 CS524Project
10:30-10:45 CS524Project
10:45-11:00 CS524Project
11:00-11:15 CS524Project
11:15-11:30 CS524Project
11:30-11:45 CS524Project
11:45-12:00 CS524Project
12:00-12:15 CS524Project
12:15-12:30 CS524Project
12:30-12:45 CS524Project
12:45-13:00 CS524Project
13:00-13:15 Break
13:15-13:30 Break
13:30-13:45 Break
13:45-14:00 Break
14:00-14:15 Essay
14:15-14:30 Essay
14:30-14:45 Essay
14:45-15:00 Essay
15:00-15:15 Essay
15:15-15:30 Essay
15:30-15:45 Essay
15:45-16:00 Essay
16:00-16:15 Essay
16:15-16:30 Essay
16:30-16:45 Essay
16:45-17:00 Essay
17:00-17:15 Break
17:15-17:30 Break
17:30-17:45 Break
17:45-18:00 Break
18:00-18:15 Break
18:15-18:30 Break
18:30-18:45 Break
18:45-19:00 Break
19:00-19:15 Break
19:15-19:30 Break
19:30-19:45 Break
19:45-20:00 Break
20:00-20:15 Break
20:15-20:30 Break
20:30-20:45 Break
20:45-21:00 Break
21:00-21:15 Break
21:15-21:30 Break
21:30-21:45 Break
21:45-22:00 Break

Amount of hours to spend on each task on day 5
CS524Project: 3.0
ReadingSummary: 0.0
```

```
Calc2: 0.0
Essay: 3.0
Research: 0.0

Day: 6
10:00-10:15 CS524Project
10:15-10:30 Essay
10:30-10:45 Essay
10:45-11:00 Essay
11:00-11:15 Essay
11:15-11:30 Essay
11:30-11:45 Essay
11:45-12:00 Essay
12:00-12:15 Essay
12:15-12:30 Essay
12:30-12:45 Essay
12:45-13:00 Essay
13:00-13:15 Break
13:15-13:30 Break
13:30-13:45 Break
13:45-14:00 Break
14:00-14:15 Essay
14:15-14:30 Break
14:30-14:45 Break
14:45-15:00 Break
15:00-15:15 Break
15:15-15:30 Break
15:30-15:45 Break
15:45-16:00 Break
16:00-16:15 Break
16:15-16:30 Break
16:30-16:45 Break
16:45-17:00 Break
17:00-17:15 Break
17:15-17:30 Break
17:30-17:45 Break
17:45-18:00 Break
18:00-18:15 Break
18:15-18:30 Break
18:30-18:45 Break
18:45-19:00 Break
19:00-19:15 Break
19:15-19:30 Break
19:30-19:45 Break
19:45-20:00 Break
20:00-20:15 Break
20:15-20:30 Break
20:30-20:45 Break
20:45-21:00 Break
21:00-21:15 Break
21:15-21:30 Break
21:30-21:45 Break
21:45-22:00 Break

Amount of hours to spend on each task on day 6
CS524Project: 0.39999999999999947
ReadingSummary: 0.0
```

```
Calc2: 0.0
Essay: 3.0
Research: 0.0

Day: 7
10:00-10:15 Essay
10:15-10:30 Essay
10:30-10:45 Essay
10:45-11:00 Essay
11:00-11:15 Essay
11:15-11:30 Essay
11:30-11:45 Essay
11:45-12:00 Essay
12:00-12:15 Essay
12:15-12:30 Essay
12:30-12:45 Essay
12:45-13:00 Essay
13:00-13:15 Break
13:15-13:30 Break
13:30-13:45 Break
13:45-14:00 Break
14:00-14:15 Break
14:15-14:30 Break
14:30-14:45 Break
14:45-15:00 Break
15:00-15:15 Break
15:15-15:30 Break
15:30-15:45 Break
15:45-16:00 Break
16:00-16:15 Break
16:15-16:30 Break
16:30-16:45 Break
16:45-17:00 Break
17:00-17:15 Break
17:15-17:30 Break
17:30-17:45 Break
17:45-18:00 Break
18:00-18:15 Break
18:15-18:30 Break
18:30-18:45 Break
18:45-19:00 Break
19:00-19:15 Break
19:15-19:30 Break
19:30-19:45 Break
19:45-20:00 Break
20:00-20:15 Break
20:15-20:30 Break
20:30-20:45 Break
20:45-21:00 Break
21:00-21:15 Break
21:15-21:30 Break
21:30-21:45 Break
21:45-22:00 Break

Amount of hours to spend on each task on day 7
CS524Project: 0.0
ReadingSummary: 0.0
```

```
Calc2: 0.0
Essay: 3.0
Research: 0.0
```

In [8]:
```
for i in 1:length(Tasks)
    println(Tasks[i], ": ")
    println("Total amount of hours spent: ", sum(output[i, :]))
    println("Initial Prediction: ", Duration[i])
    println()
end
```

```
CS524Project:
Total amount of hours spent: 13.0
Initial Prediction: 10

ReadingSummary:
Total amount of hours spent: 3.3
Initial Prediction: 3

Calc2:
Total amount of hours spent: 3.6
Initial Prediction: 3

Essay:
Total amount of hours spent: 19.2
Initial Prediction: 16

Research:
Total amount of hours spent: 2.2
Initial Prediction: 2
```

4. Results and Discussion

If we look at the results of the two models, we'll notice that the week model will try to work on each task every day until they're all done, which explains why we might see very short instances of a task in a day. Contrarily, the day-by-day model prioritizes performing the more urgent tasks first. Although it has longer times spent on each task, the day-by-day model gives better results because it considers factors like the deadline, duration, and completeness to assign each task an appropriate priority. By maximizing the priority of the tasks performed each day, the day-by-day model becomes able to organize its time better, achieve more efficiently, and offer a more convenient schedule for humans.

Tasks	Predicted Hours	Day Result	Week Result
CS 524 Project	10	13.25	13
Reading Summary	3	3.5	3.3
Calc 2 HW	3	3.75	3.6
Essay	16	19.25	19.2
Research	2	2.25	2.2
DependTask	2	2.75	N/A

In addition to the above tasks, the day-by-day model includes a task that demonstrates how dependecy will work in this scenario. In other words, we created a task called "DepenTask" that could only start once reading summary and calc 2 homewok are complete. Analyzing the printed out results in the day-by-day model, we see that this dependcy task only starts in the day once the two tasks have been worked on that week.

5. Conclusion

Common scheduling algorithms/models focus mainly on finding a schedule that finishes all tasks before their deadlines. The results of such models are rigid and offer no relaxation of the constraints, which makes them unsuitable for human consumption. When humans perform tasks, there is often plenty of room for error where not all the time spent on a task is used productively, and each day builds on the results of the previous days. The difficulty, length, and deadlines also contribute a lot to how a task is approached. For that reason, we consider those factors in creating the notion of priority that we try to maximize. Simply put, our day-to-day model incorporates knowledge on how we humans usually work on our tasks to create a schedule that is more accurate to our behaviors, and suitable for our consumption.

Future direction:

Categories:

> We could possibly add categories to different tasks and modify the model so that that tasks in the same category are preferred to be performed next to each other.

Acceptance threshold:

> Sometimes there isn't enough time to finish all your tasks which in our case makes an infeasible model. We can modify the model so that it makes it acceptable to finish a certain percentage of a task if time was an issue. For example, submitting an 80% done homework is better than submitting nothing.

Multitasking:

> Some simple tasks can be perfromed simultaneously such as cooking and watching a show at the same time. We don't have to be attending the stove or oven the whole time a meal is getting prepared