Operations Research, Spring 2024 (112-2)

Final Project Report

Adaptive Task Scheduling – Maximizing Productivity with Variable Efficiency

Team S:
Yuan Yong-Chen (B11705007)
Hsieh You-Yi (B11705024)
Cheng Chao-En (B11705037)
Wang Pei-Ling (40940323S)

June 1, 2024

1 Introduction

In our daily lives, there are always many tasks that require our attention. Unlike machines, however, humans don't operate at a consistent level of efficiency. We possess the remarkable ability to adapt and push ourselves when needed, yet we also experience fluctuations in productivity due to factors like fatigue, distractions, or external circumstances. Moreover, our lives aren't solely defined by work and chores; we also require time for leisure, relaxation, and entertainment to maintain a healthy balance.

In this project, we've taken these human complexities into account by integrating the variable of efficiency. By acknowledging that our productivity levels fluctuate and that we have diverse needs beyond work-related tasks, our aim is to develop solutions that are more adaptable and responsive to the dynamic nature of human life. The goal is to maximize productivity while ensuring that individuals can effectively manage their time and allocate it to various aspects of their lives, including work, leisure, and personal well-being.

2 Problem Description

In our model, we try our best to finish every released job on time. However, for a person, there are limited time (days/hours) and dynamic energy (efficiency) to finish the jobs. Thus, we are going to give an order or priority to the jobs. We assign every job a value and a penalty to illustrate how important a certain job is.

Every job has its release day and due day. If a person completes the job on time, he/she will gain a value, if completes the job but not on time, he/she will gain nothing, if doesn't complete the job, he/she will get a penalty. The value and the penalty are different among every job due to its importance. Our goal is to maximize the total value minus the total penalty. That is, to maximize our gain from the job scheduling.

Given the job and its release day, one cannot do the job before its release day. Once it is released, it can be assigned to the schedule. Moreover, completing one job consumes energy. It is hard for one to keep the same efficiency when doing the jobs. Therefore, the efficiency will drop after a job is completed. But if the job is an entertaining one, the efficiency may increase.

A person's mood also affects the efficiency. Mood may change day by day because it is the weekday or weekend, or depends on how many classes one needs to attend on that particular day. What's more, our group members all agree that there will be a boost of efficiency on the due day. Thus, if a job is assigned on the due day, the efficiency will increase.

For example, the default efficiency on a new day is 1. If a person's mood is very good on that day, the efficiency will +0.2. Thus, the initial efficiency of a day will be $1 + 0.2 = 1.2$. After finishing job i, the efficiency drops by 0.7. And job j is assigned to be done next on the same day. Originally, it takes 3 hours for one to complete job j when the efficiency is 1. Since his/her efficiency has dropped to 0.5 ($1.2 - 0.7 = 0.5$), he/she needs to spend $\frac{3}{0.5} = 6$ hours to finish job j. After finishing job j, the efficiency drops by 0.3. And now job k is assigned to be done next on the same day. Moreover, the day is the due day of job k. The efficiency boost is 0.6. Now the efficiency when doing job k is 0.8 ($0.5 - 0.3 + 0.6 = 0.8$) Originally, it takes 4 hours for one to complete job k when the efficiency is 1. Since his/her efficiency has become 0.8, he/she needs to spend $\frac{4}{0.8} = 5$ hours to finish job k.

Last but not least, as a person, a normal person, we can't work 24/7. Thus, there is a constraint to confirm that a person can't work more than a certain hour.

According to these rational assumptions, we know that a person can wisely choose a schedule to fulfill the jobs. Now, let's see how to explain these constraints and objective value by a mathematical model.

3 Model Formulation

Constants and Variables

Let $I = \{1, \cdots, n\}$ be the set of days, and $J = \{1, \cdots, m\}$ be the set of jobs.

Here are some given values:

B is the efficiency boost when a job is assigned on the due day.

H is the maximum time for doing jobs in a day.

$Release_j$ is the release day of job j, $\forall j \in J$.

Due_j is the due day of job j, $\forall j \in J$.

$Time_j$ is the time required to complete job j when the efficiency is 1, $\forall j \in J$.

$Value_j$ is the value of completing job j on time, $\forall j \in J$.

$Penalty_j$ is the penalty of not completing job j on time, $\forall j \in J$.

$EChange_j$ is the change of efficiency after completing job j, $\forall j \in J$.

$Mood_i$ is the efficiency increase or decrease on day i, $\forall i \in I$.

In the model, we introduce some additional variables to describe task processing:

$e_0 = 1$ (initial status on a day).

e_j is the efficiency when doing job j, $\forall j \in J$.

$\frac{Time_j}{e_j}$ is the time used to complete job j, $\forall j \in J$.

Let

$$x_{ij} = \begin{cases} 1 & \text{if job } j \in J \text{ is assigned on day } i \in I \\ 0 & \text{otherwise} \end{cases},$$

$$y_{ijk} = \begin{cases} 1 & \text{if job } j \in J \text{ is assigned before job } k \in J \text{ on day } i \in I, \\ 0 & \text{otherwise} \end{cases},$$

$$b_j = \begin{cases} 1 & \text{if job } j \in J \text{ is assigned on the due day} \\ 0 & \text{otherwise} \end{cases}, \text{ and}$$

$$c_j = \begin{cases} 1 & \text{if job } j \in J \text{ is completed on time,} \\ 0 & \text{otherwise} \end{cases}.$$

$$d_j = \begin{cases} 1 & \text{if job } j \in J \text{ is completed but not on time,} \\ 0 & \text{otherwise} \end{cases}.$$

Objective Function and Constraints

$$\max \quad \sum_{j\in J} Value_j c_j - Penalty_j(1-c_j-d_j)$$

$$\text{s.t.} \quad \sum_{i=1}^{n} x_{ij} \le 1,\ \forall j \in J$$

It means all the job could only be assigned once.

$$\sum_{i=1}^{Release_j-1} x_{ij} = 0,\ \forall j \in J$$

It means we can't assign jobs before the release day.

$$e_0 + \sum_{i\in I, j\in J, j\neq k} EChange_j y_{ijk} + \sum_{i\in I} Mood_i x_{ik} + Bb_k = e_k\ ,\ \forall k \in J$$

$$e_j \ge 0, \forall j \in J$$

It is used to count the efficiency while doing the job K

$$\sum_{j\in J} \frac{Time_j}{e_j} x_{ij} \le H\ ,\ \forall i \in I$$

It means we can't violate the time constraint.

$$x_{ij} \ge y_{ijk}\ ,\ \forall i \in I, j \in J, k \in J$$

$$x_{ik} \ge y_{ijk}\ ,\ \forall i \in I, j \in J, k \in J$$

Only when the job is assigned on that day, that the job could be before or after another job.

$$x_{ij} + x_{ik} \le y_{ijk} + y_{ikj} + 1\ ,\ \forall i \in I, j \in J, k \in J$$

If both of the jobs are assigned on the day then one must be before the other.

$$y_{ijk} + y_{ikj} \le \frac{x_{ik} + x_{ij}}{2}\ ,\ \forall i \in I, j \in J, k \in J, j \neq k$$

When both of the jobs are assigned on the day, exactly one of the job should be before another.

$$x_{Due_j j} \ge b_j\ ,\ \forall j \in J$$

Only if the job is assigned on the due day, then the due day boost could be count.

$$\sum_{i=Release_j}^{Due_j} x_{ij} \ge c_j\ ,\ \forall j \in J$$

$$\sum_{i=Due_j+1}^{n} x_{ij} \ge d_j\ ,\ \forall j \in J$$

If a job wants to be on time, it must be assigned between the release day and the due day.

$$x_{ij} = \{0,1\}\ ,\ \forall i \in I, j \in J$$

$$y_{ijk} = \{0,1\}\ ,\ \forall i \in I, j \in J, k \in J$$

$$b_j = \{0,1\}\ ,\ \forall J \in J$$

$$c_j = \{0,1\}\ ,\ \forall J \in J$$

$$d_j = \{0,1\}\ ,\ \forall J \in J$$

4 Method

To tackle this challenge, we digitalize subjective values such as B, H, $Time_j$, $Value_j$, $Penalty_j$, $EChange_j$, $Mood_j$ and recognize their variability among individuals. Our strategy involves employing programming techniques to generate diverse datasets, each embodying unique attributes to simulate different perspectives.

This digitalization process not only streamlines analysis but also enables the creation of personalized solutions tailored to individual requirements. Our objective is to develop a Python program that constructs a short-term planning model, leveraging the digitized data to provide comprehensive insights and adaptable strategies.

Heuristic Algorithm:

```python
import csv

class Job:
    def __init__(self, job_id, release, due, time, value, penalty, echange):
        self.id = job_id
        self.release = release
        self.due = due
        self.time = time
        self.value = value
        self.penalty = penalty
        self.echange = echange

def Compute_Available_Jobs_in_Order(n, Release_j, Due_j, Time_j, Value_j, Penalty_j,
    jobs_not_assigned, day):
    available_jobs = [job for job in jobs_not_assigned if job.release <= day <= job.
        due]
    available_jobs.sort(key=lambda job: (-job.value, job.time))
    return available_jobs

def process_file(filename):
    with open(filename, 'r') as file:
        reader = csv.reader(file)
        data = list(reader)

    # Extract parameters from CSV data
    n = int(data[0][1])  # Number of days
    m = int(data[1][1])  # Number of jobs
    B = float(data[2][1])  # Efficiency boost when a job is assigned on the due day
    H = float(data[3][1])  # Maximum time for doing jobs in a day
    Release_j = [int(x) for x in data[4][1:]]  # Release day of each job
    Due_j = [int(x) for x in data[5][1:]]  # Due day of each job
    Time_j = [int(x) for x in data[6][1:]]  # Time required to complete each job (hours)
    Value_j = [int(x) for x in data[7][1:]]  # Value of completing each job on time
    Penalty_j = [int(x) for x in data[8][1:]]  # Penalty of not completing each job on time
    EChange_j = [float(x) for x in data[9][1:]]  # Change of efficiency after completing each job
    Mood_i = [float(x) for x in data[10][1:]]  # Efficiency increase or decrease on each day

    # Initialize jobs list
    jobs = []
    for i in range(m):
        jobs.append(Job(i, Release_j[i], Due_j[i], Time_j[i], Value_j[i], Penalty_j[i], EChange_j[i]))

    # Initialize status
    jobs_not_assigned = jobs[:]
    jobs_assigned = set()

    total_value = 0
    total_penalty = 0

    # Simulate job assignment for each day
    for day in range(1, n + 1):
        jobs_assigned_today = []

        while True:
            # Step 1: Get the priority of currently unassigned jobs (CP array)
            CP = Compute_Available_Jobs_in_Order(n, Release_j, Due_j, Time_j, Value_j, Penalty_j, jobs_not_assigned, day)

            added_job = False

            # Step 2: Assign jobs
            for job in CP:
                flag_efficiency = False
                # Find the position to insert in jobs_assigned_today to maintain EChange order
                position = next((i for i, assigned_job in enumerate(jobs_assigned_today) if assigned_job.echange < job.echange), len(jobs_assigned_today))
                jobs_assigned_today.insert(position, job)

                # Calculate total work time after inserting job
                total_time = 0
                efficiency = 1 + Mood_i[day-1]
                for assigned_job in jobs_assigned_today:
                    if efficiency <= 0:
                        flag_efficiency = True
                        break
                    total_time += assigned_job.time / efficiency
                    efficiency += assigned_job.echange
                    if efficiency < 0:
                        flag_efficiency = True
                        break
                # Check if adding the current job exceeds time limit
                if total_time <= H and not flag_efficiency:
                    jobs_not_assigned.remove(job)
                    jobs_assigned.add(job)
                    added_job = True
                    break  # Once a job is found to be assignable, recompute CP
                else:
                    # If exceeds time limit, remove the inserted job
                    jobs_assigned_today.pop(position)
            if not added_job:
                break  # No more jobs can be assigned, end today's assignment
        for job in jobs_assigned_today:
            if day <= job.due:
                total_value += job.value
        # Print the jobs assigned for the day
        print(f"File: {filename}, Day {day}: Assigned jobs {[job.id for job in
            jobs_assigned_today]}")

    for job in jobs_not_assigned:
        total_penalty += job.penalty

    score = total_value - total_penalty
    return score

# List to store results for each file
results = []

# Loop through each file
for i in range(1, 11):
    filename = f'EC1.5/output_{i}.csv'
    score = process_file(filename)
    results.append((filename, score))

# Print the results for each file
for result in results:
    print(f"{result[0]}: Score = {result[1]}")
```

Listing 1: Job Scheduling Simulation Code

Each day, we evaluate all unassigned jobs that are eligible for assignment on that day by calculating their index value using a specific function. We then sort these jobs based on their index values in descending order. Starting with the job that has the highest index value, we attempt to assign it to the current day. If the job cannot be assigned, we proceed to the next job on the list. If none of the jobs can be assigned for that day, we move on to the next day. This process continues until all days have been considered.

Our job index calculation is defined as the following:
$days\_to\_due = job.due - today$
$days\_to\_end = n - today$

$$index = \begin{cases} \dfrac{job.value}{(days\_to\_due + 1) \times job.time} + \dfrac{|job.penalty|}{days\_to\_end + 1} & \text{if } today \leq job.due \\ \\ \dfrac{|job.penalty|}{days\_to\_end + 1} & \text{otherwise} \end{cases}$$

Case $today \leq job.due$**:**
If today is not later than the due day of the job, it means that we have the chance to finish the job and get the value, and we prefer jobs near to the due day and with high value/job.time value (First Term). We also don't want the job with high penalty not be done. (Second Term).

Case $today > job.due$**:**
If today is later than the due day of the job, it means that we don't have the chance to finish the job and get the value, so we just need to consider the penalty of not doing the job.

5 Data Collection and Generation

We are going to generate some data to test how well our heuristic algorithm perform. Noted that the symbol
$U(a, b)$ is the uniform distribution from a to b, and
$U\{a, b\}$ is the discrete uniform distribution from an integer a to an integer b.
The basic idea of data generating scenario is as follows:
$n = 10, m = 20$
$B \sim U(0, 1)$
$H = 8$
$Release_j \sim U\{0, m\}, \quad \forall j \in J$
$Due_j \sim U\{Release_j, m\}, \quad \forall j \in J$
$Time_j \sim U\{1, 8\}, \quad \forall j \in J$

$Value_j \sim U\{1, 50\}, \quad \forall j \in J$

$Penalty_j \sim U\{1, 10\}, \quad \forall j \in J$

$Echange_j \sim U(-0.5, 0), \quad \forall j \in J$

$$Mood_i \sim \begin{cases} U(-0.3, 0.7) & \text{if } i = 1, 2, ..., \lceil \frac{n}{2} \rceil \\ U(-0.7, 0.3) & \text{otherwise} \end{cases} .$$

Now, we are going to change some paraneters to see how much it will affect the job scheduling and the objective value:

Case 1: Other parameters remain the same, change H from 8 to 10. We want to determine how much increasing a person's working time can increase his/her value.

Case 2: Other parameters remain the same, double the original B, we want to determine whether a person may be given incentive to do his/her job on the due day if efficient boost raise.

Case 3: Other parameters remain the same, multiply by 1.2 and 1.5 with the original $Echange_j$ for all j in J.

6 Results

We generated 10 sets data for each scenario and find the gap between gurobi and heuristic for the change parameters and the original ones. The objective value and the gap between gurobi and heuristic is as follows:

Case 1:

	$H = 8$	$H = 10$
avg gurobi objective value	477.6	494.6
avg heuristic objective value	403.8	458.7
gap between gurobi and heuristic(%)	15.45	7.26

The objective value of $H = 8$ is less than $H = 10$, this is because one have more time to task with the job.

The gap between gurobi and heuristic is smaller when $H = 10$ because the heuristic may lose some time in everyday last few hours. The average time loss for a day should be equal. Therefore, when everyday hour is higher, the average time loss' proportion may be smaller, thus more likely close to the gurobi's objective value.

Case 2:

	original B	$B \cdot 2$
avg gurobi objective value	453.6	480.1
avg heuristic objective value	383.8	383.8
gap between gurobi and heuristic(%)	15.39	20.06

We observed that the heuristic objective value are the same in the two scenario. This is because we do not consider the efficiency boost in our heuristic algorithm. However, in intuition, it is more likely to schedule a job on the due day if efficiency boost is higher. And because of the efficiency boost, it will take less time to complete the job and more likely to assign additional job on that day. Therefore, the total value will rise. We do find this phenomenon in gurobi's objective value.

Due to no considering efficiency boost in heuristic algorithm, the gap between gurobi and heuristic is higher when efficiency boost is higher.

Case 3:

	original $Echange$	$Echange \cdot 1.2$
avg gurobi objective value	405.1	405.1
avg heuristic objective value	343.5	339.9
gap between gurobi and heuristic(%)	15.21	16.09

Another data set:

	original $Echange$	$Echange \cdot 1.5$
avg gurobi objective value	482.8	475.2
avg heuristic objective value	392.8	375.1
gap between gurobi and heuristic(%)	18.64	21.06

The objective value of the original $Echange$ is more than $Echange \cdot 1.2$ and $Echange \cdot 1.5$, this is because one meeds to spend more time to deal with the job.

The gap between gurobi and heuristic is smaller when it is original $Echange$ because the more time a job need, the more difficult for heuristic to assigned the job.

7 Conclusions

In conclusion, our project aimed to develop a flexible scheduling model that accounts for human variability in productivity and preferences. Through the implementation of a heuristic algorithm and analysis of generated data, we observed several important trends.

Firstly, we found that increasing the maximum daily working hours (H) resulted in higher objective values, indicating the potential benefits of allowing individuals more time to allocate to tasks. However, we also noted that the optimality gap between our heuristic algorithm and the Gurobi solver decreased with higher H values, suggesting that our heuristic performs relatively better in scenarios with longer workdays.

Secondly, we investigated the impact of efficiency boost (B) on job scheduling. While our heuristic algorithm did not explicitly consider efficiency boost, we observed higher objective values when B was doubled, indicating that individuals may be incentivized to complete tasks on their due dates due to the efficiency boost. However, the gap between Gurobi solution also increased in scenarios with higher B values, highlighting the importance of refining our heuristic to account for efficiency considerations.

Lastly, we examined the effects of changes in job efficiency ($EChange$) on scheduling outcomes. Our results indicated that higher efficiency requirements led to lower objective values, suggesting that increased job complexity may pose challenges for efficient task allocation. Moreover, the gap between Gurobi widened with higher efficiency requirements, underscoring the need for further enhancements to our scheduling model.

In future iterations of our project, we aim to address these limitations by incorporating continuous efficiency modeling and refining our heuristic algorithm to better capture dynamic productivity factors. By continuing to refine our model and validate its performance against real-world data, we hope to develop a robust scheduling tool that can adapt to diverse individual preferences and optimize task allocation for maximum efficiency and satisfaction.

8 Future Prospects

While our model provides valuable insights into task scheduling dynamics, there are opportunities for improvement to better align with real-world scenarios. One aspect that merits attention is the ability for individuals to split a single job and work on it at different times. Currently, our model assumes binary assignment variables (x_i) indicating whether a job is assigned on a particular day. However, in reality, individuals may choose to allocate partial time to a task across multiple days.

To address this limitation, we propose transitioning from binary to continuous assignment variables, allowing for finer granularity in task allocation. By representing assignments as continuous values between 0 and 1, we can capture the degree of completion for each job on each day. This approach would require defining a mechanism to track the progress of tasks over time, potentially through the use of integral calculus to compute the cumulative workload completed for each job.

Additionally, we recognize the challenge of modeling dynamic efficiency fluctuations over time.

While our model accounts for efficiency changes associated with completing each job, continuous efficiency modeling remains an area of ongoing exploration. One possible approach is to integrate mood and energy levels as dynamic variables that influence efficiency throughout the scheduling horizon. By incorporating mood fluctuations and energy dynamics into the model, we can better simulate real-world productivity variations and optimize task allocation accordingly.

Furthermore, an analysis of job assignments across different scenarios can provide valuable insights into scheduling effectiveness. By examining the allocation of each job and identifying patterns or discrepancies among various scenarios, we can offer tailored recommendations to enhance scheduling efficiency. This analysis may involve identifying critical tasks that contribute significantly to overall objective value, as well as evaluating the impact of different scheduling strategies on productivity and well-being.

In future iterations of our model, we aim to address these challenges and refine our approach to better reflect the complexities of real-world task scheduling. By incorporating continuous assignment variables, dynamic efficiency modeling, and detailed job assignment analysis, we can develop a more accurate and effective scheduling tool that meets the diverse needs of individuals and organizations.