2024학년도 여 름 학기 Logistics Lab

연구참여 보고서

(원료하역 스케줄링 과제)

작성일자 : 2024.07.18

지도 교수: 김병인

지도 사수: 김한솔

참여 학생 : 고현민

목차

1. 연구 내용

1.1 문제 설명

기존 선박 원료하역 스케줄 편성과정은 전부 수작업으로 진행되었기 때문에, 매주 이것을 위한 회의가 필요해 많은 시간이 소요되었으며 사람의 작업으로 인한 오차가 발생한다는 문제점이 있다. 따라서 이전 연구에서는 포항항을 기준으로 해당 과정을 알고리즘 개발을 통해 자동화 시켰으며, 이번 연구에서는 광양항을 기준으로 해서 기존 연구에 야드적치계획 또한 만족시키도록 심화시킨 연구를 진행하였다.

원료가 운반되는 과정은 다음과 같이 말할 수 있다. 우선 배가 싣고 오면 언로더(하역기)를 통해 이를 퍼내어 컨베이어벨트에 올린다. 이후 해당 원료는 야드까지 가게 되고 야드로부터 시설까지 가게 되는 과정을 거친다. 본 연구에서는 원료를 어떤 방식으로 퍼내며, 어떤 과정으로 야드까지 보낼지에 대해 알고리즘을 구현하는 것을 목표로 한다.

1.2 연구 참여

본인은 졸업 후에 포스코에 취업하고 싶은 생각이 있다. 동시에 방학동안 쉬기보다는 이것저것 더 공부하고 싶은 마음이 있었다. 그러던 중에 김병인 교수님께서 포스코 과제를 많이 해오시고 계신다는 것을 알게 되었고 감사하게도 이번 여름학기동안 연구참여를 할 수 있도록 허가해주셔서 이렇게 6주 간의 좋은 기회를 얻게 되었다.

1.2.1 연구 상황 및 맡은 업무

본 연구 과제는 24년 3월부터 시작하였으며 6월 초 기준으로 막 데이터들을 다 받은 상황이었다. 따라서 내가 연구참여를 시작한 시점의 연구는 아직 초기단계라고 할 수 있으며 이제 수리모델을 세우고 알고리즘을 개발하여 초기해를 만들어야 하였다. 아직 2학년 1학기밖에 마치지 않았으며 연구경험, 코딩경험이 많지 않았던 나의 당시 상황으로는 할 수 있는 것이 많이 제한되었다. 따라서 내가 맡게 된 업무는 알고리즘으로부터 도출 된 output 결과값들을 간트차트를 통해 시각화 할 수 있도록 코드를 짜는 것이었다. 당시 맡은 업무는 상당히 간단하다고 말할 수 있으나 앞으로의 기간동안 기존에 궁금했던 것들에 대해서 공부하고 직접 봐보는 경험을 가져야겠다는 생각을 하였다.

1.2.2 선행연구

간트차트를 통해 결과값을 도출하기에 앞서 나는 공부해야겠다고 생각했던 것이 있었다. 이는 다음과 같다.

1. 파이썬 공부

2. 해당 연구 프로젝트 공부

1번 같은 경우에는 내가 이전에 c와 c++의 언어만 공부하였기 때문에 간트차트를 구현할 언어인 파이썬을 공부해야겠다고 생각하였다. 2번의 경우는 전에 말했듯이 단순히 시각화 일만 하고 끝나고 싶지 않았기 때문에 원료하역 스케줄링 과제에 대해서 배경지식을 이해하고 어떤 식으로 과제를 해결해나가는지에 대한 방법론 또한 알고 싶어서 공부해야겠다고 생각했다.

파이썬 공부는 마침 김병인 교수님의 강의 중에 '정보시스템기술'이라는 강의에서 산업공학에서 쓰이는 것들을 파이썬 언어로 다루어 놓으셨기 때문에 이를 통해 공부하고자 하였다. 처음에는 그냥 간트차트 코드를 작성하는 데 필요한 개념들만 듣고자 하였는데 막상 시작해보니 그냥 처음부터 끝까지 다 수강하자는 마음이 들어서 약 40개 정도의 강의를 모두 수강하였다.

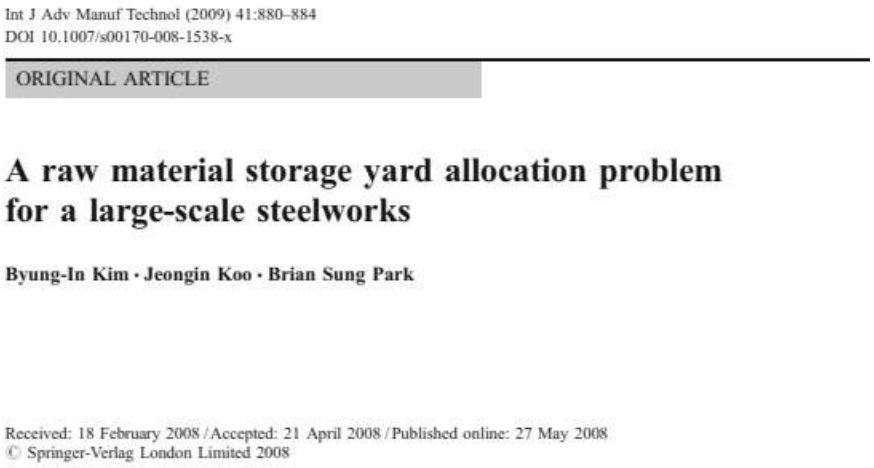
YouTube links for Information System Technology Lecture (Python, C++)

Lecture (Python)	Address
(POSTECH IST: Python 1) Basic	https://youtu.be/aQiYldsBfiE
(POSTECH IST: Python 2) String	https://youtu.be/aDTgt53qtWg
(POSTECH IST: Python 3) List & Tuple	https://youtu.be/k_Kb3vP-zSU
(POSTECH IST: Python 4) Function	https://youtu.be/AxnIaEuLAtc
(POSTECH IST: Extra) Knapsack Problem	https://youtu.be/lOcafMGbEG8
(POSTECH IST: Python 5) Generator	https://youtu.be/ofQV0ruHAX4
(POSTECH IST: Python 6) Object-Oriented Programming	https://youtu.be/9ACeaMzcoiI
(POSTECH IST: Python 7) Class, Module, Speed	https://youtu.be/iQbC1ZPjqNw
(POSTECH IST: Python 8) Exception Handling	https://youtu.be/4g6xTuZzh3M
(POSTECH IST: Python 9) File I/O	https://youtu.be/D5Jd78uYo9w
(POSTECH IST: Python 10) Data Structure	https://youtu.be/Qi4gw-xSEN8

(POSTECH IST: Extra) Shortest Path Problems	https://youtu.be/LE6LdlzKLHA
(POSTECH IST: Python 24) Ch7. Linked Lists	https://youtu.be/0_WO0sedpYY
(POSTECH IST: Python 25-1) Ch8. Trees (General Trees)	https://youtu.be/ySUpTxIEwI8
(POSTECH IST: Python 25-2) Ch8. Trees (Binary Trees)	https://youtu.be/WGJ2dHw-ft8
(POSTECH IST: Python 25-3) Ch8. Trees (Implementing)	https://youtu.be/O2UTLh2gm04
(POSTECH IST: Python 25-4) Ch8. Trees (Traversal)	https://youtu.be/5l6HwDvp3iI
(POSTECH IST: Python 26-1) Ch9. Priority Queues	https://youtu.be/8dpX5vCstTM
(POSTECH IST: Python 26-2) Ch9. Adaptable Priority Queues	https://youtu.be/ruUtdHr0Cus
(POSTECH IST: Python 27-1) Ch14. Graph Data Structure	https://youtu.be/U7AqCrzmis0
(POSTECH IST: Python 27-2) Ch14. Graph Algorithms	https://youtu.be/zmvcNpODKKY
(POSTECH IST: Python 28) Ch10. Maps & Hash Tables	https://youtu.be/y-bM1Y91ER0
(POSTECH IST: Python 29) Ch11. Binary Search Trees	https://youtu.be/w4At60divGY
(POSTECH IST: Python 30) Ch11. Balanced Search Trees	https://youtu.be/bgoAxEnxLWA
(POSTECH IST: Python 31) Web Crawling	https://youtu.be/4LXPJ1AJDlQ

[그림 1, 2 정보시스템기술 강의 목록]

또한 원료하역 스케줄링 과제 공부 같은 경우는 해당 부분을 다룬 논문을 리뷰해보고자 하였다. 논문은 2009년에 작성된 'A raw material storage yard allocation problem for a large scale steelworks'이라는 논문으로 저자는 B. Kim* , J.Koo , and B.S. Park으로 김병인 교수님께서 이전에 진행한 관련 연구를 담은 것이다. 논문의 내용을 요약하면 야드에 원료들을 적치하는 것을 최대한 효율적으로 할 수 있도록 이를 Mixed Integer Programming을 통해 수리모델로 해결한 것이다.



Int J Adv Manuf Technol (2009) 41:880–884
DOI 10.1007/s00170-008-1538-x

ORIGINAL ARTICLE

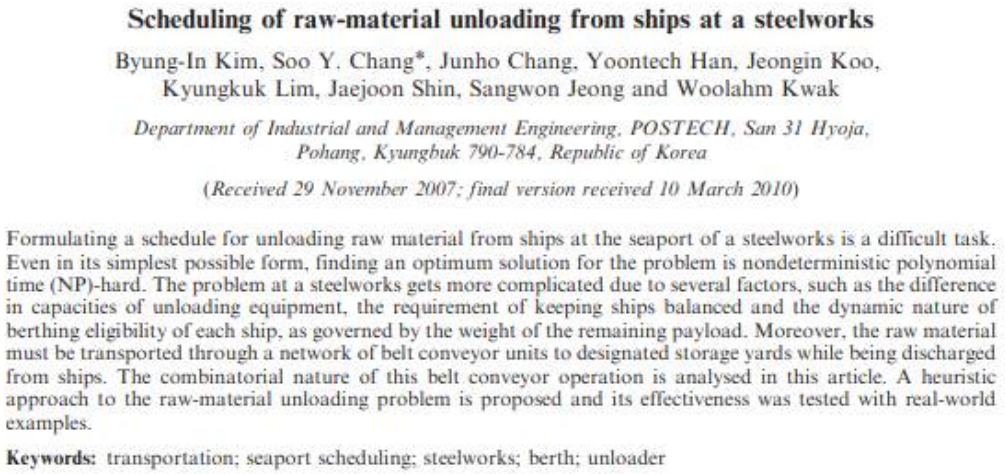
A raw material storage yard allocation problem for a large-scale steelworks

Byung-In Kim · Jeongin Koo · Brian Sung Park

Received: 18 February 2008 / Accepted: 21 April 2008 / Published online: 27 May 2008
© Springer-Verlag London Limited 2008

[그림 3 잘못 읽은 논문 사진]

논문리뷰가 처음이라서 약간의 겁을 먹긴 했지만 생각보다 많이 어렵지는 않았다. 약간 그냥 책이나 교과서를 읽는다는 느낌이라고 볼 수 있을 것 같다. 하지만 막상 다 읽고나니 내가 논문을 잘못 골랐다는 것을 추후에 알게 되었다. 본 연구와 관련된 논문은 2011년에 작성된 제목 'Scheduling of raw-material unloading from ships at a steelworks', 저자 B. Kim , S. Chang* , J. Chang , Y. Han , J. Koo , K. Lim , J. Shin , S. Jeong , W. Kwak으로 마찬가지로 김병인 교수님이 작성하신 논문이다.



Scheduling of raw-material unloading from ships at a steelworks

Byung-In Kim, Soo Y. Chang*, Junho Chang, Yoontech Han, Jeongin Koo, Kyungkuk Lim, Jaejoon Shin, Sangwon Jeong and Woolahm Kwak

Department of Industrial and Management Engineering, POSTECH, San 31 Hyoja, Pohang, Kyungbuk 790-784, Republic of Korea

(*Received 29 November 2007; final version received 10 March 2010*)

Formulating a schedule for unloading raw material from ships at the seaport of a steelworks is a difficult task. Even in its simplest possible form, finding an optimum solution for the problem is nondeterministic polynomial time (NP)-hard. The problem at a steelworks gets more complicated due to several factors, such as the difference in capacities of unloading equipment, the requirement of keeping ships balanced and the dynamic nature of berthing eligibility of each ship, as governed by the weight of the remaining payload. Moreover, the raw material must be transported through a network of belt conveyor units to designated storage yards while being discharged from ships. The combinatorial nature of this belt conveyor operation is analysed in this article. A heuristic approach to the raw-material unloading problem is proposed and its effectiveness was tested with real-world examples.

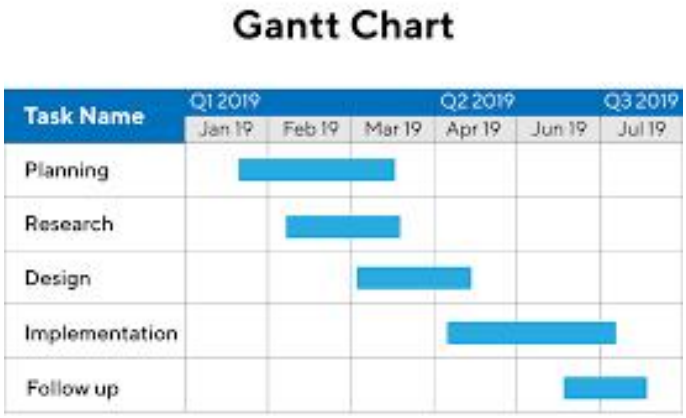
Keywords: transportation; seaport scheduling; steelworks; berth; unloader

[그림 4 원래 읽어야 했던 논문 사진]

잘못된 논문을 리뷰하였고 시간이 부족하여 원래 읽어야 했던 논문을 리뷰하지 못했다는 것이 아쉽긴 했지만 해당 논문은 연구참여가 끝나고도 천천히 리뷰할 수 있고 그래도 처음으로 논문을 읽고 공부했다는 사실을 의미있게 두기로 하였다.

1.2.3 output 자료 시각화

본 연구에서 output 자료 시각화는 파이썬 언어를 통해 간트차트로 구현해보고자 하였다. 간트 차트(Gantt Chart)란 프로젝트 관리에서 사용되는 시각적 도구로, 프로젝트의 일정과 진행 상황을 한눈에 파악할 수 있도록 도와준다. 이 차트는 시간 축을 따라 작업들의 시작과 종료 일정을 막대 형태로 나타내어, 프로젝트의 진행 상태를 시각적으로 표현한다.



[그림 4 간트차트 예시 그림]

내가 이번에 구현한 간트차트 코드는 다음과 같다.

```
# https://www.w3schools.com/colors/colors_picker.asp [색상표 참고]
import sys
import pandas as pd
import csv
import plotly
import plotly.express as px
import plotly.graph_objects as go
from plotly.subplots import make_subplots
import random

# python gantt_chart.py ship
# python gantt_chart.py unloader

def getCsv1(folder):
    numlist1 = []
    f =open(folder +'solution_ship.csv','r', encoding='utf-8')    # open file
csv at current folder as read mode
    csvReader =csv.reader(f)    # read file with reader
    c =-1
    for i in csvReader:
        if not any(i):  # Check if the row is empty
                continue  # Skip empty rows
        c +=1
        if c ==0:
```

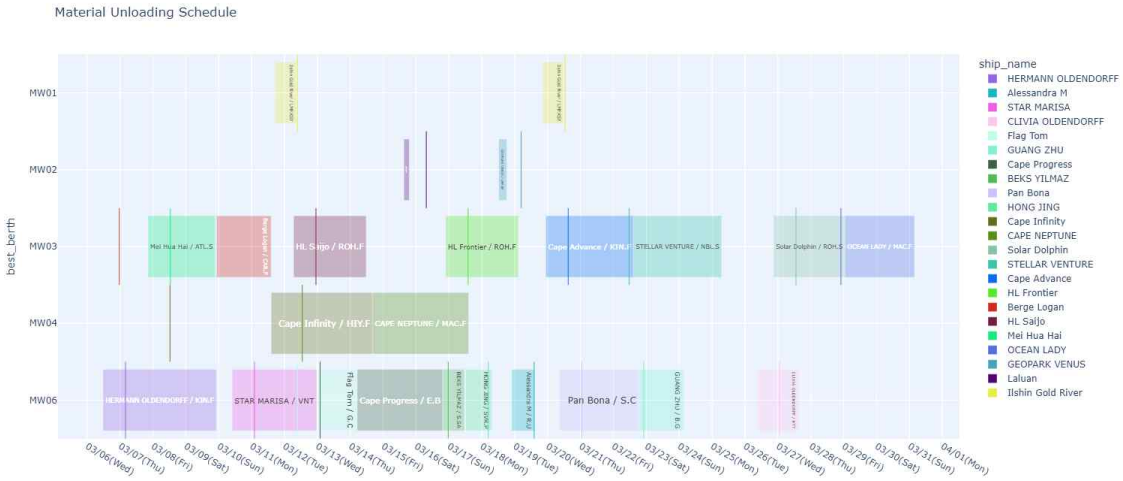
```
            continue # do not read the first line of csv
        x1 =dict(ship_name=i[0], best_berth=i[1], ETA=i[2], berthing_time=i[3],
start_Laytime=i[4],
workload_start_time=i[5],finish_Laytime=i[6],workload_finish_time=i[7],

departure_time=i[8],demurrage_flag=i[9],dispatch_money=i[10],demurrage_cost=i[11]
,brand_name=i[12],workload_amount=i[13],yard_index=i[14],
                yard_plan_flag=i[15],shift_flag=i[16],text=i[0]+' / '+i[12])
        numlist1.append(x1)
    f.close()
    return numlist1

def getCsv2(folder):
    numlist2 = []
    f =open(folder +'solution_unloader.csv','r', encoding='utf-8')    # open
file csv at current folder as read mode
    csvReader =csv.reader(f)    # read file with reader
    c =-1
    for i in csvReader:
        if not any(i):  # Check if the row is empty
                continue  # Skip empty rows
        c +=1
        if c ==0:
            continue # do not read the first line of csv
        x2 =dict(ship_name=i[0], best_berth=i[1], unloader_name=i[2], ETA=i[3],
berthing_time=i[4], start_Laytime=i[5], finish_Laytime=i[6],
                unloader_start_time=i[7],
unloader_finish_time=i[8],workload_start_time=i[9], workload_finish_time=i[10],
departure_time=i[11],demurrage_flag=i[12],

 dispatch_money=i[13],demurrage_cost=i[14],brand_name=i[15],workload_amount=i[16]
,workload_amount_unloader=i[17],yard_index=i[18],yard_plan_flag=i[19],
                shift_flag=i[20],text=i[0]+' / '+i[15],
yaxis_name=i[1]+'.'+i[2] )
        numlist2.append(x2)
    f.close()
    return numlist2

argv =sys.argv
name =argv[1]
if name =="ship": # solution_ship.csv
     # read csv input file
    #---------------------main--------------------
    folder ="./"

    df1 =getCsv1(folder)
    df =pd.DataFrame(df1)
    df =df.sort_values(by='best_berth', ascending=False)
    unique_ship_names =df['ship_name'].unique()
    colors_for_machine = {ship: "#{:06x}".format(random.randint(0, 0xFFFFFF))
for ship in unique_ship_names}

    fig1 =px.timeline(df, x_start="workload_start_time",
```

```
x_end="workload_finish_time", y="best_berth",
                      color="ship_name",
                      color_discrete_map=colors_for_machine,
                      title='Material Unloading Schedule',
                      text="text",
                      hover_data=df)
                      # height=5000, width=1000)
    fig1.update_yaxes(showgrid=True, minor_showgrid=True)
    fig1.update_xaxes(
        tickformat="%m/%d(%a)",
        dtick="D1"
    )
    fig1.update_traces(textposition='inside')
    fig1.update_traces(marker=dict(opacity=0.3))

    for index, row in df.iterrows():
        # best_berth 값을 기반으로 y0와 y1 위치 계산
        y_value =df['best_berth'].unique().tolist().index(row['best_berth'])
        fig1.add_shape(
            type='line',
            x0=row['finish_Laytime'], x1=row['finish_Laytime'],
            y0=y_value -0.5, y1=y_value +0.5,
            line=dict(color=colors_for_machine[row['ship_name']], width=1),  # 배
이름에 따른 색상 설정
            xref='x', yref='y'
        )

    random_number =random.randint(1000, 9999)

    # Create the filename with the random number
    filename =folder +f"ganttChart_material_{random_number}.html"
    # filename = folder + "ganttChart_material.html"

    # Save the plot to the file
    plotly.offline.plot(fig1, filename=filename)

elif name =="unloader": # solution_unloader.csv
     # read csv input file
    #----------------------main--------------------
    folder ="./"

    df1 =getCsv2(folder)
    df =pd.DataFrame(df1)
    df =df.sort_values(by='best_berth', ascending=False)
    unique_ship_names =df['ship_name'].unique()
    colors_for_machine = {ship: "#{:06x}".format(random.randint(0, 0xFFFFFF))
for ship in unique_ship_names}

    fig1 =px.timeline(df, x_start="workload_start_time",
x_end="workload_finish_time", y="yaxis_name",
                      color="ship_name",
                      color_discrete_map=colors_for_machine,
                      title='Material Unloading Schedule',
```

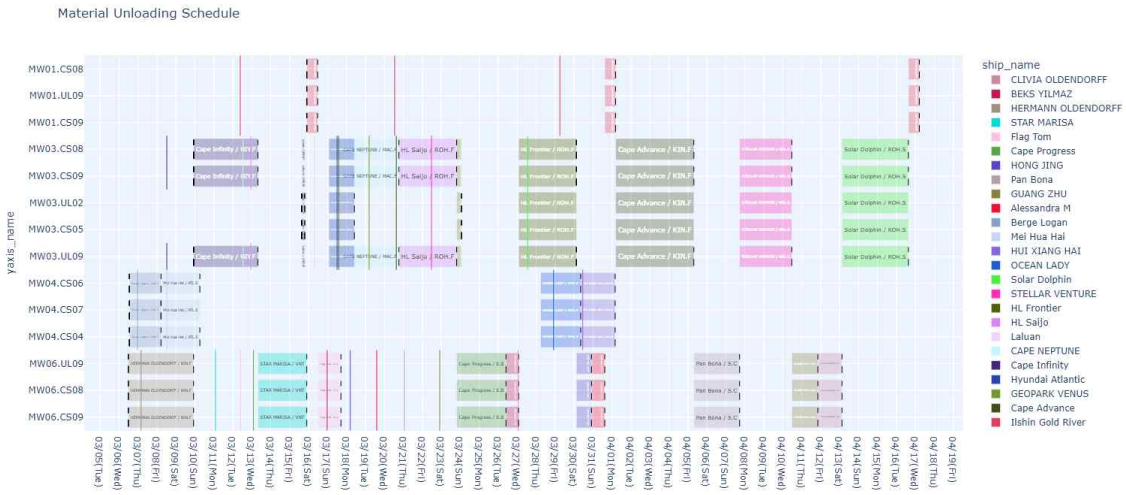
```
                      text="text",
                      hover_data=df)
                      # height=5000, width=1000)
    fig1.update_yaxes(showgrid=True, minor_showgrid=True)
    fig1.update_xaxes(
        tickformat="%m/%d(%a)",
        dtick="D1"
    )
    fig1.update_traces(textposition='inside')
    fig1.update_traces(marker=dict(opacity=0.3))

    for index, row in df.iterrows():
        # best_berth 값을 기반으로 y0와 y1 위치 계산
        y_value =df['yaxis_name'].unique().tolist().index(row['yaxis_name'])
        fig1.add_shape(
            type='line',
            x0=row['finish_Laytime'], x1=row['finish_Laytime'],
            y0=y_value -0.5, y1=y_value +0.5,
            line=dict(color=colors_for_machine[row['ship_name']], width=1),  # 배
이름에 따른 색상 설정
            xref='x', yref='y'
        )
        fig1.add_shape(
            type='line',
            x0=row['unloader_start_time'], x1=row['unloader_start_time'],
            y0=y_value -0.35, y1=y_value +0.35,
            line=dict(color='Black', width=1, dash='dash'),  # 배 이름에 따른 색상
설정
            xref='x', yref='y'
        )
        fig1.add_shape(
            type='line',
            x0=row['unloader_finish_time'], x1=row['unloader_finish_time'],
            y0=y_value -0.35, y1=y_value +0.35,
            line=dict(color='Black', width=1, dash='dash'),  # 배 이름에 따른 색상
설정
            xref='x', yref='y'
        )


    random_number =random.randint(1000, 9999)

    # Create the filename with the random number
    filename =folder +f"ganttChart_material_{random_number}.html"
    # filename = folder + "ganttChart_material.html"

    # Save the plot to the file
    plotly.offline.plot(fig1, filename=filename)
```

[그림 5 작성한 간트차트 코드]

코드설명을 하자면 일단 두가지 output 파일을 각각의 버전으로 결과값을 출력할 수 있도록 argv 함수를 통해 나누었다. 첫 번째는 solution_ship.csv 파일을 입력 받아 출력하는 것인데 x축은 시간, y축은 선석으로 나누어서 간트차트를 그리도록 하였다. 두 번째는 solution_unloader.csv 파일을 입력 받아 출력하는 것인데 x축은 동일하게 시간, y축은 선석.언로더명으로 나누어서 차트를 그리도록 하였다. 막대들의 색은 선박의 이름별로 나누도록 하였으며 추가로 넣은 기능 같은 경우에는 채선료 기준 시간 표시하기, 언로더별로 작업한 시간 점선으로 표시하기 등이 있다. 해당 코드를 정상적으로 컴파일 하여 볼 수 있는 간트차트들은 다음과 같다.



[그림 6 간트차트 ver2(solution_ship.csv)]



[그림 7 간트차트 ver2(solution_unloader.csv)]

2. 후기

사실 산경과를 선택하고 나서 한학기 전공과목들을 배우기는 했지만 이런 개념들이 어떠한 방식으로 연구에 사용되는지에 대해서는 잘 몰랐습니다. 그래서 이번 연구참여를 통해 이에 관해 알고 싶은 마음이 컸습니다.

사실 제가 연구 경험이 있는 것도 아니고 코딩실력이 뛰어난 것도 아니라 모든 게 '맨땅에 헤딩하기'였습니다. 교수님이나 대학원생분들이 쓰시는 용어들의 대부분을 몰라서 처음에는 어떤 걸 어디서부터 시작하면 좋을지 막막했습니다. 하지만 작지만 간트차트로 시각화하는 부분을 담당하고 매주 여러 미팅에 들어가서 듣고 여쭤보면서 하나씩 배우니 확실히 짧은 시간동안에도 많은 부분을 배울 수 있었습니다. 제가 연구참여 기간이 6주밖에 되지 않아서 어떤 걸 깊게 파고들기에는 부족했지만 전체적으로 어떤 느낌인지에 대해 알 수 있어서 좋았습니다. 특히 실제 연구실에서 연구들은 어떻게 이루어지는지, 기업과제는 어떻게 진행되는지, 문제접근은 어떻게 하고 어떤 걸 사용해 해결해 나가는지에 대해 알 수 있었습니다. 개인적으로는 이런 걸 하나하나 알아간다는 것이 상당히 즐거웠습니다. 다음에 또 한다면 그때는 직접 알고리즘을 개발하는 작업을 해보고 싶습니다.