**산업시스템프로그래밍1 HW 02**

2018112462 유근태

1. 핵심 로직 설명

1-1. EDD 알고리즘 적용

Lots에 존재하는 마감 시간이 가까운 Job부터 처리하게끔 정렬을 진행하였다.

# Lots를 마감 기한이 빠른 순서로 정렬

lots = sorted(lots, key=lambda x: x[3])

1-2. Job Type별 총 시간을 확인하고, 시간의 양만큼 머신 수 배치하기

Job Type별 총 시간을 확인해보니 아래 표와 같았다.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Job Type | A | B | C | D | E | F | G | H | I | J | K | L | M |
| 총 시간 | 36 | 80 | 25 | 71 | 58 | 29 | 8 | 2 | 19 | 23 | 10 | 17 | 23 |

이 시간을 근거로 해서 머신 1개당 24시간으로 환산하여 작업을 처리할 머신 수를 지정해주었다. 예를 들어 A의 경우 36시간이기 때문에 머신 2대, B의 경우 80시간이기 때문에 4대, C의 경우 25시간에 해당하기 때문에 1대 이런 식으로 머신수를 지정해주고, 머신을 할당해주었다. 그리고 24시간에 전혀 미치지 않는 시간을 가진 Job Type의 경우에는 한 머신으로 묶어서 처리해주었다.

# Job Type에 따른 머신 배정 매핑

job\_type\_to\_machine = {

    'I': [0],  # 19시간

    'B': [1, 2, 3, 4],  # 80시간

    'E': [5, 6, 7],  # 58시간

    'M': [8],  # 23시간

    'D': [9, 10, 11],  # 71시간

    'G': [12],  # 8시간

    'A': [12, 13],  # 36시간

    'C': [14],  # 25시간

    'J': [15],  # 23시간

    'F': [16],  # 29시간

    'L': [17],  # 10시간

    'K': [17],  # 10시간

    'H': [17],  # 2시간

}

1-3. M16 뒤에 있는 작업 2개(J48, J59)를 M07에서 작업하도록 변경

텍스트, 스크린샷, 다채로움, 평행이(가) 표시된 사진

자동 생성된 설명

위에 있는 간트 차트는 [1-2]까지 진행한 스케줄링 결과값이다. 눈으로 확인할 수 있듯이, M07은 작업을 더 할 수 있는 여력이 있고, M16의 경우 너무 작업이 많은 것을 확인할 수 있다. 그러므로 우선적으로 M16 뒤에 있는 J48, J59 이 2개의 작업을 M7 뒤로 옮겨주었다. SETUP 타임은 당연히 고려해주었다.

# 기존 스케줄에서 머신 16의 작업을 확인

machine\_16\_jobs = [job for job in schedule if job[1] == 16]

last\_3\_jobs = machine\_16\_jobs[-2:]

# 머신 7의 마지막 완료 시간 확인

machine\_7\_last\_time = machines[7][1]

# 머신 7로 작업 이동 및 시간 재계산

for job in last\_3\_jobs:

    JobId, \_, JobType, \_, \_, Due, \_ = job

    Qty = job[4] - job[3]  # 작업의 양

    # Setup 시간이 필요한지 확인

    if machines[7][0] != JobType:

        schedule.append([-1, 7, 'S', machine\_7\_last\_time, machine\_7\_last\_time + 2, -1, -1])

        machine\_7\_last\_time += 2

    sTime = machine\_7\_last\_time

    cTime = sTime + Qty

    tar = max(0, cTime - Due)

    tar = min(tar, Qty)

    # 머신 7의 상태 업데이트

    machines[7][0] = JobType

    machines[7][1] = cTime

    # 스케줄에 반영

    schedule.remove(job)

    schedule.append([JobId, 7, JobType, sTime, cTime, Due, tar])

    # 머신 7의 마지막 완료 시간 업데이트

    machine\_7\_last\_time = cTime

# 전체 지연 시간 재계산

total\_tardi = sum(job[6] for job in schedule)

1-4. M09에 있는 마지막 작업(J44)을 M04에서 작업하도록 코드 변경

텍스트, 스크린샷, 다채로움, 라인이(가) 표시된 사진

자동 생성된 설명

위에 있는 간트 차트는 [1-3]까지 진행한 스케줄링 결과값이다. M09에 있는 J44를 SETUP 타임을 고려하더라도 시작 시간이 빠른 M04에 배치하면 시간이 감소될 것이라고 판단했고, 이를 반영해주었다.

machine\_9\_jobs = [job for job in schedule if job[1] == 9]

last\_job = machine\_9\_jobs[-1]

machine\_4\_last\_time = machines[4][1]

JobId, \_, JobType, \_, \_, Due, \_ = last\_job

Qty = last\_job[4] - last\_job[3]  # 작업의 양

if machines[4][0] != JobType:

    schedule.append([-1, 4, 'S', machine\_4\_last\_time, machine\_4\_last\_time + 2, -1, -1])

    machine\_4\_last\_time += 2

sTime = machine\_4\_last\_time

cTime = sTime + Qty

tar = max(0, cTime - Due)

tar = min(tar, Qty)

machines[4][0] = JobType

machines[4][1] = cTime

schedule.remove(last\_job)

schedule.append([JobId, 4, JobType, sTime, cTime, Due, tar])

total\_tardi = sum(job[6] for job in schedule)

2. 최종 tardiness 값

90

3. 최종 Gantt Chart

텍스트, 스크린샷, 다채로움, 평행이(가) 표시된 사진

자동 생성된 설명