hw1

September 26, 2024

1 HW Assignment 1, CMOR3 464 INDE 543 - MANUFACTURING PROCESSES AND SYSTEMS

2 Line Setup LP Problem Formulation

2.1 Problem Description

We are tasked with optimizing a production line to meet daily demand under given constraints. The problem includes calculating production with different numbers of workstations, overtime hours, and efficiency levels to determine the most cost-effective setup.

2.1.1 Data Provided:

Costs:

- Unit Material Costs: $30
- Hourly Wage: $60
- Overtime Hourly Wage: $90

Current Production:

- Current Production: 315 units/day
- Daily Demand: 420 units/day

Other Costs:

- One Time Training Costs: $10,000

2.1.2 Working Schedule:

- **Workday Start**: 8:00 AM
- **Workday End**: 4:00 PM
- **Overtime Max**: 7:00 PM
- **Normal Work Hours**: 8 hours
- **Maximum Overtime Hours**: 11 hours (including 1 hour break)
- **Break Time**: 1 hour

2.1.3 Line Setup Information:

# of Workstations	Overtime (hrs/day)	Capacity (u/day)	Efficiency
6	0		
6	3		
7	0		
7	3		
8	0		
8	3		
9	0		
9	3		

2.1.4 Activities Data:

Activities	Time (seconds)	Immediate Predecessors	Workers Allotted
1	30	n/a	1
2	50	n/a	2
3	40	1	1
4	50	1	3
5	20	2	2
6	10	3, 4	4
7	10	4, 5	5
8	20	2	3
9	10	6	4
10	40	9	5
11	20	7	5
12	30	7	7
13	50	9	4
14	50	10	6
15	10	11	6
16	40	8, 12	7

2.1.5 Working Hours:

Metric	Value
Beginning Work Time	8:00 AM
End Work Time	4:00 PM
Overtime Max	7:00 PM
Normal Hours	8 hours
Maximum Overtime Hours	11 hours
Break Time	1 hour
Normal Productive Time	7 hours
Maximal Overtime Productive Time	10 hours

2.2 Objective:

The objective is to write Linear Programs (LPs) that optimize the number of workstations. We can use Type 2 Model from the textbook:

2.3 Type 2 Assembly Line Balancing Model

2.3.1 Variables:

- $ t_i $: Time required for activity $ A_i $, where $ i = 1, 2, ..., N $
- − $ N = 16 $ for this example
- $ S $: The number of workstations, where $ S = 1, 2, ..., n $
- − n is variable, but it'll be $n \in \{6, 7, ..., 9\}$
- $ C $: Cycle time of the line (maximum time allowed per workstation)
- $ x_{is} $: A binary variable indicating whether activity $ A_i $ is assigned to station $ S $

2.3.2 Objective:

Minimize cycle time $ C $, i.e., the time required to complete all tasks assigned to a workstation.

$$\text{Min } C$$

2.3.3 Constraints:

1. **Cycle Time Constraint**:
   The total time assigned to any workstation must not exceed the cycle time $ C $:

$$\sum_i x_{is} \cdot t_i \leq C, \quad \forall s \in S$$

2. **Assignment Constraint**:
   Each activity must be assigned to exactly one workstation:

$$\sum_s x_{is} = 1, \quad \forall i \in N$$

3. **Precedence Constraint**:
   Precedence relationships between activities must be respected. If activity $ A_i $ must precede activity $ A_j $, then the assignment must reflect that:

$$\sum_{s=1}^{S} x_{is} \cdot s \leq \sum_{s=1}^{S} x_{js} \cdot s, \quad \forall (i, j) \text{ where } A_i \text{ precedes } A_j$$

2.3.4 Problem Summary:

- The goal is to balance the assembly line by minimizing the cycle time while respecting the constraints of workstation assignment and task precedence.
- Each workstation must process tasks in a way that satisfies the overall demand, ensuring efficiency and minimizing idle time.

3 Initialize the model

import gurobipy as gp from gurobipy import GRB

[224]:
```
# Dictionary to store time for each job
job_times = {
    1: 30,
    2: 50,
    3: 40,
    4: 50,
    5: 20,
    6: 10,
    7: 10,
    8: 20,
    9: 10,
    10: 40,
    11: 20,
    12: 30,
    13: 50,
    14: 50,
    15: 10,
    16: 40
}
```

[225]:
```
import numpy as np

# Define the precedence dictionary based on your table (activity -> list of␣
↪immediate predecessors)
precedence_dict = {
    1: [],
    2: [],
    3: [1],
    4: [1],
    5: [2],
    6: [3, 4],
    7: [4, 5],
    8: [2],
    9: [6],
    10: [9],
    11: [7],
    12: [7],
    13: [9],
    14: [10],
    15: [11],
    16: [8, 12]
}

# Initialize an empty precedence matrix with zeros (16x16)
```

```python
precedence_matrix = np.zeros((num_jobs, num_jobs), dtype=int)

# Fill the precedence matrix based on the precedence_dict
for activity, predecessors in precedence_dict.items():
    for predecessor in predecessors:
        precedence_matrix[activity - 1][predecessor - 1] = 1

# Print the precedence matrix to verify
print("Precedence Matrix:")
print(precedence_matrix)
```

```
Precedence Matrix:
[[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0]
 [0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0]
 [0 0 0 0 0 0 0 1 0 0 0 1 0 0 0 0]]
```

[226]:

```python

```

[231]:

```python
# Define the number of jobs
num_jobs = 16

# List of different numbers of stations to try
num_stations_list = [6, 7, 8, 9]

# Dictionary to store results for each number of stations
results = {}

for num_stations in num_stations_list:
    print(f"Solving for {num_stations} stations...")

    # Create a new Gurobi model
    model = gp.Model("assembly_line_balancing")

    C = model.addVar(name="C")
    model.update()
```

```python
    model.setObjective(C, GRB.MINIMIZE)

    # Create a matrix of Gurobi binary variables X_{is} for i in [1, 16] and s␣
↪in [1, num_stations]
    X = [[model.addVar(vtype=GRB.BINARY, name="X_{}_{}".format(i+1, s+1))
          for s in range(num_stations)]
          for i in range(num_jobs)]

    # Add cycle time constraints to the model
    for s in range(num_stations):
        model.addConstr(
            gp.quicksum(X[i][s] * job_times[i+1] for i in range(num_jobs)) <= C,
            name=f"cycle_time_station_{s+1}"
        )

    # Add assignment constraints to the model
    for i in range(num_jobs):
        model.addConstr(
            gp.quicksum(X[i][s] for s in range(num_stations)) == 1,
            name=f"assignment_job_{i+1}"
        )

    # Add precedence constraints to the model
    for i in range(num_jobs):
        for j in range(num_jobs):
            if precedence_matrix[i][j] == 1:
                for k in range(num_stations):
                    lhs = X[i][k]
                    rhs = gp.quicksum(X[j][z] for z in range(k+1))
                    model.addConstr(lhs <= rhs,␣
↪name=f"precedence_{i+1}_{j+1}_station_{k+1}")

    # Update the model to integrate the new constraints
    model.update()

    # Optimize the model
    model.optimize()

    # Check if the model has an optimal solution
    if model.status == GRB.OPTIMAL:
        # Create a dictionary to store the assignment of activities to stations
        station_assignment = {s+1: [] for s in range(num_stations)}

        # Iterate over the variables to get the assignment
        for i in range(num_jobs):
            for s in range(num_stations):
```

```
                if X[i][s].x > 0.5:  # If the variable is 1 (assigned)
                    station_assignment[s+1].append(i+1)

        # Store the results
        results[num_stations] = {
            'cycle_time': C.x,
            'station_assignment': station_assignment
        }

        # Print the assignment
        print(f"Optimal cycle time for {num_stations} stations: {C.x}")
        for station, activities in station_assignment.items():
            print(f"Station {station}: Activities {activities}, Total Time␣
↪{sum(job_times[activity] for activity in activities)} seconds")
    else:
        print(f"No optimal solution found for {num_stations} stations.")

# Print the results summary
print("\nSummary of results:")
for num_stations, result in results.items():
    print(f"{num_stations} stations: Cycle time = {result['cycle_time']}")
```

```
Solving for 6 stations…
Gurobi Optimizer version 11.0.1 build v11.0.1rc0 (mac64[arm] - Darwin 23.6.0
23G93)

CPU model: Apple M2 Max
Thread count: 12 physical cores, 12 logical processors, using up to 12 threads

Optimize a model with 124 rows, 97 columns and 657 nonzeros
Model fingerprint: 0xe1af4366
Variable types: 1 continuous, 96 integer (96 binary)
Coefficient statistics:
  Matrix range     [1e+00, 5e+01]
  Objective range  [1e+00, 1e+00]
  Bounds range     [1e+00, 1e+00]
  RHS range        [1e+00, 1e+00]
Found heuristic solution: objective 410.0000000
Presolve removed 17 rows and 0 columns
Presolve time: 0.00s
Presolved: 107 rows, 97 columns, 470 nonzeros
Variable types: 0 continuous, 97 integer (96 binary)

Root relaxation: objective 8.000000e+01, 64 iterations, 0.00 seconds (0.00 work
units)

    Nodes    |    Current Node    |     Objective Bounds      |     Work
 Expl Unexpl |  Obj  Depth IntInf | Incumbent    BestBd   Gap | It/Node Time
```

```
     0     0   80.00000    0   21  410.00000   80.00000  80.5%     -    0s
H    0     0                        120.0000000   80.00000  33.3%     -    0s
H    0     0                        100.0000000   80.00000  20.0%     -    0s
H    0     0                         90.0000000   80.00000  11.1%     -    0s
H    0     0                         80.0000000   80.00000  0.00%     -    0s
     0     0   80.00000    0   27   80.00000   80.00000  0.00%     -    0s

Cutting planes:
  Gomory: 5
  Cover: 5
  Implied bound: 1
  Clique: 6
  MIR: 4
  StrongCG: 2
  Zero half: 3
  RLT: 6
  Relax-and-lift: 1
  BQP: 2

Explored 1 nodes (165 simplex iterations) in 0.02 seconds (0.00 work units)
Thread count was 12 (of 12 available processors)

Solution count 5: 80 90 100 … 410

Optimal solution found (tolerance 1.00e-04)
Best objective 8.000000000000e+01, best bound 8.000000000000e+01, gap 0.0000%
Optimal cycle time for 6 stations: 80.0
Station 1: Activities [1, 4], Total Time 80 seconds
Station 2: Activities [2, 5, 7], Total Time 80 seconds
Station 3: Activities [3, 6, 8, 9], Total Time 80 seconds
Station 4: Activities [12, 13], Total Time 80 seconds
Station 5: Activities [10, 16], Total Time 80 seconds
Station 6: Activities [11, 14, 15], Total Time 80 seconds
Solving for 7 stations…
Gurobi Optimizer version 11.0.1 build v11.0.1rc0 (mac64[arm] - Darwin 23.6.0
23G93)

CPU model: Apple M2 Max
Thread count: 12 physical cores, 12 logical processors, using up to 12 threads

Optimize a model with 142 rows, 113 columns and 826 nonzeros
Model fingerprint: 0x8d604f7e
Variable types: 1 continuous, 112 integer (112 binary)
Coefficient statistics:
  Matrix range     [1e+00, 5e+01]
  Objective range  [1e+00, 1e+00]
  Bounds range     [1e+00, 1e+00]
```

```
  RHS range        [1e+00, 1e+00]
Found heuristic solution: objective 220.0000000
Presolve removed 24 rows and 4 columns
Presolve time: 0.00s
Presolved: 118 rows, 109 columns, 567 nonzeros
Variable types: 0 continuous, 109 integer (108 binary)

Root relaxation: objective 6.857143e+01, 74 iterations, 0.00 seconds (0.00 work
units)

    Nodes    |    Current Node    |     Objective Bounds      |     Work
 Expl Unexpl |  Obj  Depth IntInf | Incumbent    BestBd   Gap | It/Node Time

     0     0   68.57143    0   14  220.00000   68.57143  68.8%     -    0s
H    0     0                       80.0000000   68.57143  14.3%     -    0s
H    0     0                       70.0000000   70.00000  0.00%     -    0s
     0     0   70.00000    0   26   70.00000   70.00000  0.00%     -    0s

Cutting planes:
  Gomory: 1
  Cover: 6
  Clique: 2
  MIR: 4
  GUB cover: 1
  Zero half: 1
  RLT: 4
  BQP: 1

Explored 1 nodes (376 simplex iterations) in 0.02 seconds (0.01 work units)
Thread count was 12 (of 12 available processors)

Solution count 3: 70 80 220

Optimal solution found (tolerance 1.00e-04)
Best objective 7.000000000000e+01, best bound 7.000000000000e+01, gap 0.0000%
Optimal cycle time for 7 stations: 70.0
Station 1: Activities [2, 5], Total Time 70 seconds
Station 2: Activities [1, 3], Total Time 70 seconds
Station 3: Activities [4, 6, 9], Total Time 70 seconds
Station 4: Activities [7, 13], Total Time 60 seconds
Station 5: Activities [10, 12], Total Time 70 seconds
Station 6: Activities [11, 14], Total Time 70 seconds
Station 7: Activities [8, 15, 16], Total Time 70 seconds
Solving for 8 stations…
Gurobi Optimizer version 11.0.1 build v11.0.1rc0 (mac64[arm] - Darwin 23.6.0
23G93)

CPU model: Apple M2 Max
```

```
Thread count: 12 physical cores, 12 logical processors, using up to 12 threads

Optimize a model with 160 rows, 129 columns and 1012 nonzeros
Model fingerprint: 0xa29d5b87
Variable types: 1 continuous, 128 integer (128 binary)
Coefficient statistics:
  Matrix range     [1e+00, 5e+01]
  Objective range  [1e+00, 1e+00]
  Bounds range     [1e+00, 1e+00]
  RHS range        [1e+00, 1e+00]
Found heuristic solution: objective 300.0000000
Presolve removed 21 rows and 2 columns
Presolve time: 0.00s
Presolved: 139 rows, 127 columns, 725 nonzeros
Variable types: 0 continuous, 127 integer (126 binary)

Root relaxation: objective 6.000000e+01, 85 iterations, 0.00 seconds (0.00 work
units)

    Nodes    |    Current Node    |     Objective Bounds      |     Work
 Expl Unexpl |  Obj  Depth IntInf | Incumbent    BestBd   Gap | It/Node Time

     0     0   60.00000    0   38  300.00000   60.00000  80.0%     -    0s
H    0     0                     110.0000000   60.00000  45.5%     -    0s
H    0     0                      90.0000000   60.00000  33.3%     -    0s
H    0     0                      80.0000000   60.00000  25.0%     -    0s
H    0     0                      70.0000000   60.00000  14.3%     -    0s
     0     0   70.00000    0   39   70.00000   70.00000  0.00%     -    0s

Cutting planes:
  Gomory: 12
  Cover: 7
  Clique: 11
  MIR: 5
  StrongCG: 1
  GUB cover: 1
  Zero half: 11
  RLT: 8
  Relax-and-lift: 1
  BQP: 2

Explored 1 nodes (217 simplex iterations) in 0.02 seconds (0.01 work units)
Thread count was 12 (of 12 available processors)

Solution count 5: 70 80 90 … 300

Optimal solution found (tolerance 1.00e-04)
Best objective 7.000000000000e+01, best bound 7.000000000000e+01, gap 0.0000%
```

```
Optimal cycle time for 8 stations: 70.0
Station 1: Activities [1, 3], Total Time 70 seconds
Station 2: Activities [4, 6, 9], Total Time 70 seconds
Station 3: Activities [10], Total Time 40 seconds
Station 4: Activities [14], Total Time 50 seconds
Station 5: Activities [2, 5], Total Time 70 seconds
Station 6: Activities [13], Total Time 50 seconds
Station 7: Activities [7, 8, 11, 15], Total Time 60 seconds
Station 8: Activities [12, 16], Total Time 70 seconds
Solving for 9 stations…
Gurobi Optimizer version 11.0.1 build v11.0.1rc0 (mac64[arm] - Darwin 23.6.0
23G93)

CPU model: Apple M2 Max
Thread count: 12 physical cores, 12 logical processors, using up to 12 threads

Optimize a model with 178 rows, 145 columns and 1215 nonzeros
Model fingerprint: 0xb4366da3
Variable types: 1 continuous, 144 integer (144 binary)
Coefficient statistics:
  Matrix range     [1e+00, 5e+01]
  Objective range  [1e+00, 1e+00]
  Bounds range     [1e+00, 1e+00]
  RHS range        [1e+00, 1e+00]
Found heuristic solution: objective 260.0000000
Presolve removed 21 rows and 2 columns
Presolve time: 0.00s
Presolved: 157 rows, 143 columns, 890 nonzeros
Variable types: 0 continuous, 143 integer (142 binary)

Root relaxation: objective 5.333333e+01, 120 iterations, 0.00 seconds (0.00 work
units)

    Nodes    |    Current Node    |     Objective Bounds      |     Work
 Expl Unexpl |  Obj  Depth IntInf | Incumbent    BestBd   Gap | It/Node Time

     0     0   53.33333    0   25  260.00000   53.33333  79.5%     -    0s
H    0     0                     120.0000000   53.33333  55.6%     -    0s
H    0     0                     100.0000000   53.33333  46.7%     -    0s
H    0     0                      90.0000000   53.33333  40.7%     -    0s
H    0     0                      80.0000000   53.33333  33.3%     -    0s
     0     0   60.00000    0    8   80.00000   60.00000  25.0%     -    0s
H    0     0                      70.0000000   60.00000  14.3%     -    0s
H    0     0                      60.0000000   60.00000  0.00%     -    0s
     0     0   60.00000    0    8   60.00000   60.00000  0.00%     -    0s

Cutting planes:
  Gomory: 2
```

```
  Cover: 6
  Implied bound: 4
  Clique: 1
  MIR: 2
  RLT: 4

Explored 1 nodes (366 simplex iterations) in 0.02 seconds (0.01 work units)
Thread count was 12 (of 12 available processors)

Solution count 7: 60 70 80 … 260

Optimal solution found (tolerance 1.00e-04)
Best objective 6.000000000000e+01, best bound 6.000000000000e+01, gap 0.0000%
Optimal cycle time for 9 stations: 60.0
Station 1: Activities [2], Total Time 50 seconds
Station 2: Activities [1, 8], Total Time 50 seconds
Station 3: Activities [4], Total Time 50 seconds
Station 4: Activities [3, 6, 9], Total Time 60 seconds
Station 5: Activities [5, 7, 12], Total Time 60 seconds
Station 6: Activities [10, 11], Total Time 60 seconds
Station 7: Activities [14, 15], Total Time 60 seconds
Station 8: Activities [13], Total Time 50 seconds
Station 9: Activities [16], Total Time 40 seconds

Summary of results:
6 stations: Cycle time = 80.0
7 stations: Cycle time = 70.0
8 stations: Cycle time = 70.0
9 stations: Cycle time = 60.0
```

[233]:
```
import csv

# Define the file path
output_file_path = '/Users/warrenweissbluth/Documents/
↪CMOR-464-INDE-543-MANUFACTURING-PROCESSES-AND-SYSTEMS/results.csv'

# Write the results to the CSV file
with open(output_file_path, 'w', newline='') as file:
    writer = csv.writer(file)

    # Write the header
    writer.writerow(["Number of Stations", "Cycle Time", "Station",␣
↪"Activities", "Total Time (seconds)"])

    # Write the data
    for num_stations, result in results.items():
        for station, activities in result['station_assignment'].items():
```

```python
            total_time = sum(job_times[activity] for activity in activities)
            writer.writerow([num_stations, result['cycle_time'], station,␣
 ↪activities, total_time])

print(f"Results written to {output_file_path}")
```

```
Results written to
/Users/warrenweissbluth/Documents/CMOR-464-INDE-543-MANUFACTURING-PROCESSES-AND-
SYSTEMS/results.csv
```