Production System Modelling MEC-E7001

Assignment 4

Optimizing MTO aggregate planning

Name: Nguyen Xuan Binh

Student ID: 887799

Mail: binh.nguyen@aalto.fi

Group: 17

Date: 29.02.2024

1. INTRODUCTION

The purpose of the report is to investigate the optimization of Make-to-Order (MTO) aggregate planning, which is highly critical in an MTO environment because it directly affects a production plan to balance consumer demand with production scheduling, resource allocation, and operating expenses. Given the stochastic nature of order arrivals and the considerable unpredictability in product specifications in MTO manufacturing, the use of advanced optimization techniques is not only advantageous but also required to maintain competitive service levels and cost management.

This report compares three distinct scheduling scenarios: a standard schedule with sequential steps, a project control method that allows flexibility within timing and precedence constraints, and an advanced model that allows free allocation of work contents within process step timing windows. The examination of these situations will provide insight into resource leveling and peak load mitigation in MTO planning. We want to learn how alternative planning techniques affect the operating efficiency and responsiveness of MTO production systems by using data from ten orders that use four different resources.

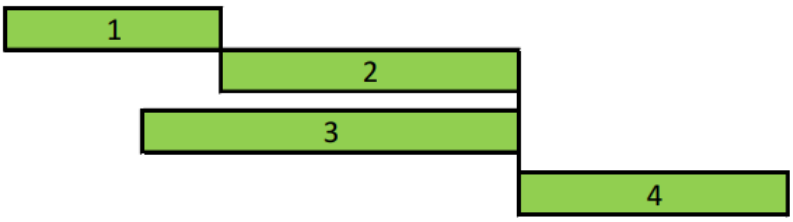
2. PROBLEM AND MODEL

Modeling method and software

The softwares that I used are Excel, Python (some light processing), and CPLEX solver. The "CPLEX Studio IDE" is the Integrated Development Environment for CPLEX, and it is the main program where we can open, edit, and solve OPL (Optimization Programming Language) files. In this assignment we would experiment with given (and modified) MTO aggregate planning optimization OPL-models using CPLEX.

Model and data:

- Process is a 4 resource job shop for a product family
- Allowed durations and work contents for process steps and due dates and earliest starting times for orders are known
- Each optimization experiment concerns 10 orders, each of which consists of 4 steps as shown below:



- Numbered steps load the four resources correspondingly
- Resource load profile peaks (sum of all 4) are the optimization criterion, which is minimized

Model description

The three MTO scheduling approaches are tested in this report: standard schedule, project control (even work content), and free work content

Approach 1: MTO aggregate planning - standard schedules

Standard project schedules where steps directly follow each other and work is distributed evenly over allowed duration. It is almost like second approach except without precedence constraint

C_ik >= C_hk + P_ik for all i, h in U(i), k

Approach 2: MTO planning optimization – project control (even WC)

Project control method where single process steps may be moved within timing and precedence constraints. Work is distributed evenly over allowed durations

$\text{Min} \sum_{\forall k} f_k$	Minimization of sum of resource load peaks
s.t.	
$C_{ik} = \sum_{t=1}^{HZ} tc_{ikt}, \quad \forall i,k$	c_{ikt} = 1if job is finished at time t, otherwise 0 C_{ik} = Finishing time of job *i*
$\sum_{t=1}^{HZ} c_{ikt} = 1, \quad \forall i,k$	Jobs end once
$D_i \geq C_{ik} \geq C_{hk} + P_{ik}, \forall i,h \in U(i),k$	Precedences and due dates not violated *U(i)* is set containing steps preceding *i*
$C_{ik} - P_{ik} \geq A_{ik}, \quad \forall i,k$	Earliest allowed starting time A_{ik}
$R_{ikt} = WC_{ik}/P_{ik} \sum_{u=t}^{t+P_{ik}-1} c_{iku}, \quad \forall i,k$	Total work content is evenly distributed over all *t* in timing window R_{ikt}
$f_k - \sum_{i=1}^{I} R_{ikt} \geq 0, \quad \forall t,k$	Peak load

- Objective Function: Minimize the sum of f_k for all k,

which aims to minimize the load peak sum

- Constraints:

1. C_ik = sum of t_c_ikt from t=1 to HZ for all i, k,

C_ik represents the completion time of the i-th job on the k-th resource. c_ikt is a binary variable that indicates job completion. This calculates the finishing time of job i on resource k.

2. sum of C_ikt from t=1 to HZ equals 1 for all i, k,

Ensures that each process i on resource k must be completed once

3. D_i >= C_ik for all i, k

D_i is the due date for order i. C_hk is the completion time of process h on resource k. This constraint ensures that the due date must be respected.

4. C_ik >= C_hk + P_ik for all i, h in U(i), k

P_ik is the processing time for order i on resource k.

This ensures that the completion time for each process must be after the sum of the completion time of any preceding process h and its own processing time, following the order given by U(i) which is the precedence.

5. C_ik - P_ik >= A_ik for all i, k,

A_ik is the earliest start time for process i on resource k. This constraint ensures that the start time of each process is not before its earliest allowed start time.

6. R_ik = WC_ik / P_ik * sum of t_c_iku from u=t to t+P_ik-1 for all i, k,

Indicates that the total work content is evenly distributed over all t in the timing window.

7. f_k - sum of R_ikt from i=1 to I' >= 0 for all t, k,

R_ikt represents the resource load at time t for resource k. This constraint ensures that there is workload only if work i is ongoing.

Approach 3: MTO planning optimization – free WC allocation within process step timing window

Same as approach 2, but work contents are freely allocated within process step timing windows

$$\text{Min} \sum_{\forall k} f_k$$

s.t.

$$C_{ik} = \sum_{t=1}^{HZ} tc_{ikt}, \quad \forall i$$

$$\sum_{t=1}^{HZ} c_{ikt} = 1, \quad \forall i,k$$

$$D_i \geq C_{ik} \geq C_{hk} + P_{ik}, \forall i, h \in U(i), k$$

$$C_{ik} - P_{ik} \geq A_{ik}, \quad \forall i,k$$

$$X_{ikt} = \sum_{u=t}^{t+P_{ik}-1} c_{iku}, \quad \forall i,t$$ X_{ikt} = 1, if t in allowed timing window, otherwise 0

$$R_{ikt} \leq MX_{ikt}, \quad \forall i,k,t$$ Work may only be allocated to timing window

$$\sum_{t=1}^{HZ} R_{ikt} = WC_{ik}, \quad \forall i,k$$ All work content must be allocated

$$f_k - \sum_{i=1}^{I} R_{ikt} \geq 0, \quad \forall t,k$$

- Objective Function: Minimize the sum of f_k for all k,

which aims to minimize the load peak sum

- Subject to Constraints:

1. C_ik = sum of t_c_ikt from t=1 to HZ for all i,

C_ik represents the completion time of the i-th job on the k-th resource. c_ikt is a binary variable that indicates job completion. This calculates the finishing time of job i on resource k.

2. sum of C_ikt from t=1 to HZ equals 1 for all i, k,

Ensures that each process i on resource k must be completed once

3. D_i >= C_ik for all i, k

D_i is the due date for order i. C_hk is the completion time of process h on resource k. This constraint ensures that the due date must be respected.

4. C_ik >= C_hk + P_ik for all i, h in U(i), k

P_ik is the processing time for order i on resource k.

This ensures that the completion time for each process must be after the sum of the completion time of any preceding process h and its own processing time, following the order given by U(i) which is the precedence.

5. C_ik - P_ik >= A_ik for all i, k,

A_ik is the earliest start time for process i on resource k. This constraint ensures that the start time of each process is not before its earliest allowed start time.

6. X_ikt = sum of t_c_iku from u=t to t+P_ik-1 for all i, t and X_ikt = 1, if t in allowed timing window, otherwise 0,

X_ikt is a binary variable that indicates whether the process i is in the timing window at time t on resource k. This defines X_ikt based on whether the process is within the allocated timing window.

7. R_ikt <= M * X_ikt for all i, k, t,

R_ikt represents the resource load at time t for resource k. This constraint ensures that there is workload only if work i is ongoing.

8. sum of R_ikt from t=1 to HZ equals WC_ik for all i, k,

WC_ik represents the total work content for process i on resource k.

This ensures that the total allocated work content for each process on each resource equals the required work content.

9. f_k - sum of R_ikt from i=1 to n >= 0 for all t, k,

Ensures that the cost or load f_k for resource k is at least as large as the sum of the loads R_ikt across all processes i.

3. EXPERIMENT

a. The used values in this modeling assignment

Here are the first few rows of the orders.

i (Index Column): This column represents the unique identifier for each task in the table

Order (Order Column): This column seems to be grouping tasks that belong to the same job or order

Each task has its own A (earliest starting time), D (due date), P (duration), and WC (work content) for different resources (k=1 to k=4).

					WC			
i	Order	A	D	P	k=1	k=2	k=3	k=4
1	1	7	18	3	134.5817	0	0	0
2	1	7	18	3	0	227.0393	0	0
3	1	7	18	1	0	0	79.56843	0
4	1	7	18	4	0	0	0	45.49319
5	2	3	15	1	17.12197	0	0	0
6	2	3	15	3	0	231.6176	0	0
7	2	3	15	3	0	0	186.1477	0
8	2	3	15	4	0	0	0	277.8746
9	3	7	26	4	166.4834	0	0	0
10	3	7	26	1	0	14.92198	0	0
11	3	7	26	4	0	0	262.2905	0
12	3	7	26	4	0	0	0	313.497

b. Model testing

The model is tested as follows:

1. OPL-project is created and MILP models are tested.

2. I would then experiment with my own group data, which is read directly from the Excel data file. Repetitions are not conducted and the tests with given ”order backlog” is reported

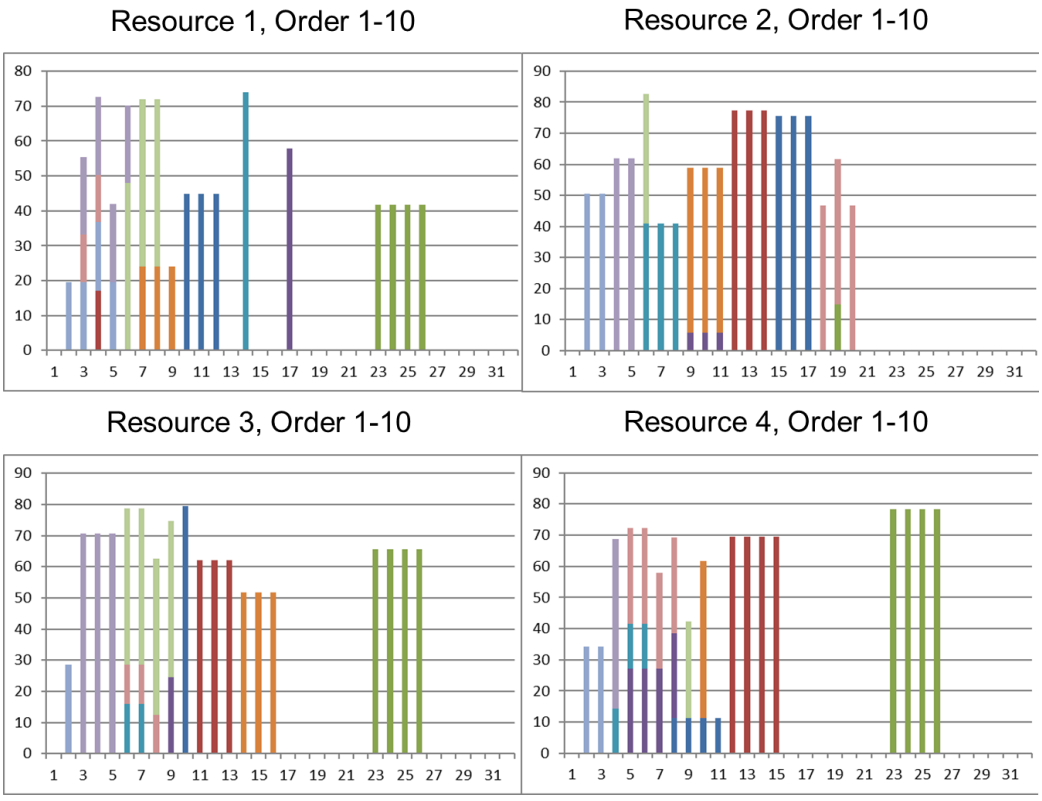
3. The Excel file ”MTO-AggregatePlanningAssignmentData.xls” contains group specific .dat file. I copied my data to the yellow area so that the models can read the data from the right cells (whose references can be observed in .dat file)

4. The auxiliary variables S[i][t] in the model are written to the same Excel file (green area). With this data, resource profiles are then sorted first on the Resource column, then sorted by Order column in the spreadsheet. After that, the resource profiles would then be plotted as barplots

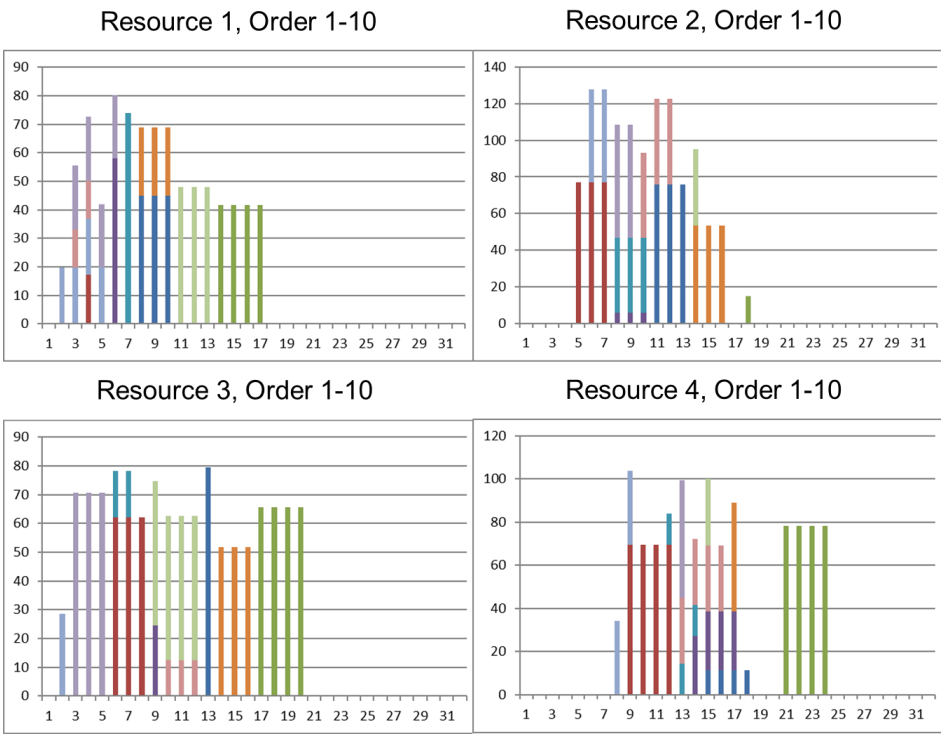
c. Modeling results:

Here are the resource allocation graphs of the 3 schedules

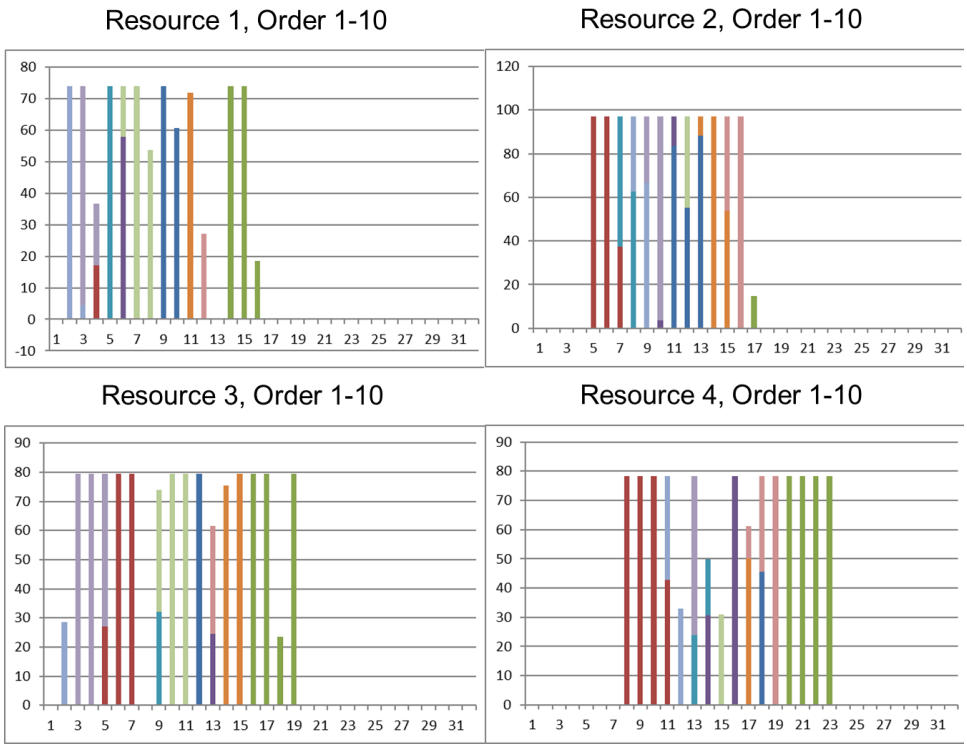
- **Standard schedule**



- **Even work content**



- **Free work content**



The resource load peaks are outputted from the model
dvar float+ f[1..K]; // Resource load peaks

	Standard schedule	Even WC	Free WC
k = 1	73.98366212	80.20598237	73.98366212
k = 2	82.52252769	127.702799	97.07719221
k = 3	79.56843435	79.56843435	79.56843435
k = 4	78.37425773	103.7048991	78.37425773

Next, we can report the completion time, flow time and waiting time across all jobs (I = 40)

1. Completion time is outputted from the model

dvar int C[1..I];// Completion times of jobs, auxiliary variable

2. Flow time is completion time minus starting time
3. Waiting time is flow time minus processing time

	Standard schedule	Even WC	Free WC
max completion time	26	24	26
mean completion time	10.6	11.425	11.45
max flow time	19	17	19
mean flow time	6.6	7.425	7.45
max waiting time	15	13	15
mean waiting time	4.025	4.85	4.875

Brief analysis of the results

- Resource Load Peaks

Standard Schedule and Even WC have nearly identical resource allocations for the four resources. On the other hand, Standard Schedule and Free WC show the same resource load at k=1 and k=4, implying that these techniques are equally efficient in exploiting these specific

resources without overloading them. For resources k=2 and k=3, however, the Free WC method marginally raises the load peak compared to the Standard schedule, indicating the possibility of increased resource usage but also the risk of overflowing.

Even WC has much greater resource demand maxima at k=2 and k=4 than the Standard and Free WC tactics. This suggests that the Even WC method aims to distribute workload more equally across resources, but at the expense of greater peak loads, which may result in delays due to resource saturation.

- Completion Time

Compared to the Standard and Free WC tactics, the Even WC approach has a two-unit shorter maximum completion time. This implies that even WC can accomplish all jobs faster, perhaps resulting in a more efficient overall operation. The Even WC and Free WC methods have a little greater mean completion time than the Standard schedule. This rise might be attributed to task redistribution with the goal of achieving balance at the expense of individual job efficiency.

- Flow Time

The Even WC method reduces maximum flow time, which indicates a more efficient processing path for the longest-waiting activities. The mean flow time is longer in the Even WC and Free WC strategies, indicating a modest increase in the average time jobs spend in the system from start to finish.

- Waiting Time

The Even WC method reduces the maximum waiting time, implying that it can shorten the wait for the longest-waiting operation. The average waiting time is larger under the Even WC and Free WC strategies, suggesting that tasks have somewhat longer idle times before processing.

Extra study: Test forward and backward scheduling (see lecture notes) and compare results with the earlier ones.

In the lecture notes, the two schedules are mentioned as an extreme version of MTO planning. Forward planning means that the planners must use the resources to run the orders as soon as possible, forcing the jobs to start when all constraints are satisfied. This method would increase WIP and it also allows accommodating new orders. However, this forward planning can lead to a complex reschedule, which can cause confusion and unwanted administrative expenses that arise during the rescheduling. In contrast, backward planning only begins to process the orders as late as possible with the constraints of capacity, release dates, and due dates. This strategy is less flexible to accept new orders compared to forward planning. However, it is more suitable

where customers may ask the manufacturers to revise their requirements, as corrections are more manageable closer to the manufacturing date.

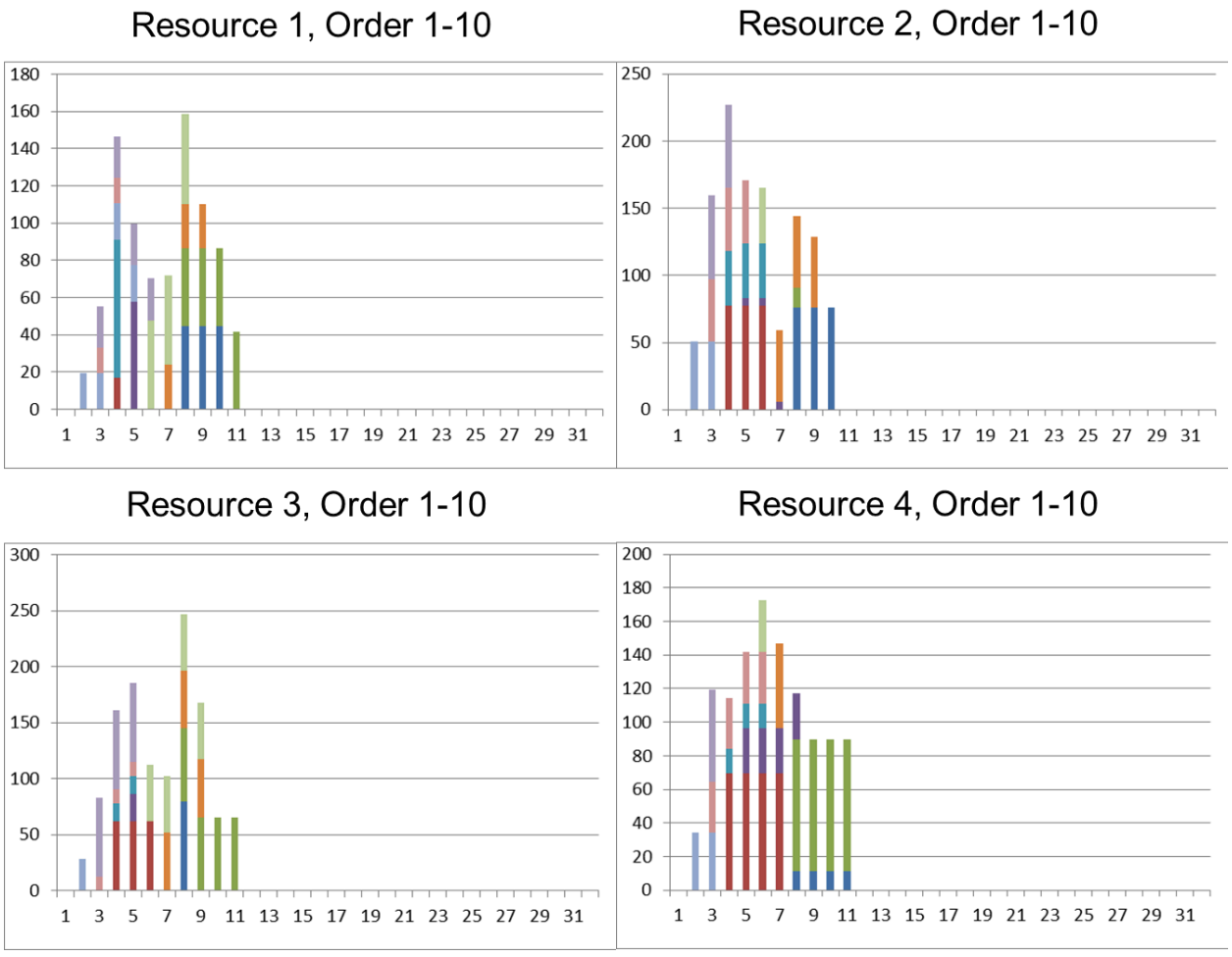
Forward planning and backward planning have the same constraints as the standard schedule, except that their objective function is to minimize or maximize sum of starting time

Forward planning: minimize sum(i in 1..I)(C[i] - P[i]);

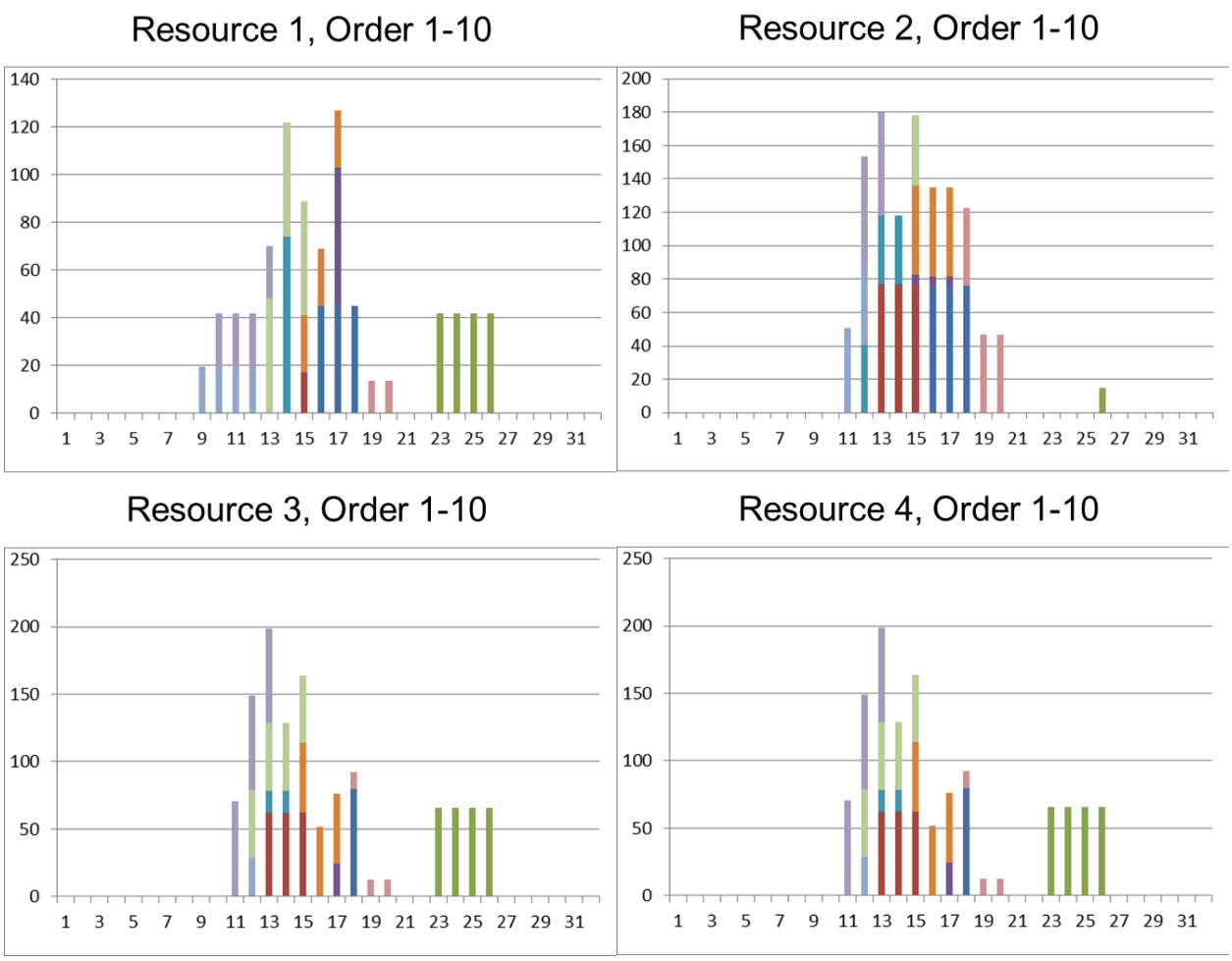
Backward planning: maximize sum(i in 1..I)(C[i] - P[i]);

Here are the resource allocation graphs of the 2 forward and backward planning schedules

- **Forward planning schedule**



- **Backward planning schedule**



The resource load peaks are outputted from the model

dvar float+ f[1..K]; // Resource load peaks

	Forward planning	Backward planning
k = 1	157	127
k = 2	227	180
k = 3	248	199
k = 4	173	139

Next, we can report the completion time, flow time and waiting time across all jobs (I = 40)

1. Completion time is outputted from the model

dvar int C[1..I];// Completion times of jobs, auxiliary variable

2. Flow time is completion time minus starting time
3. Waiting time is flow time minus processing time

	Forward planning	Backward planning
max completion time	11	26
mean completion time	6.575	16.7
max flow time	4	19
mean flow time	2.575	12.7
max waiting time	0	18
mean waiting time	0	10.125

Brief analysis of the result

- Completion time: Forward planning reduces both the maximum and mean completion times compared to backward planning. So it appears that forward planning is better in processing and completing jobs sooner.

- Flow time: Similar to completion times, forward planning has considerably lower maximum and mean flow times, meaning that jobs spend less time in the system before completion.

- Waiting time: Forward planning has zero waiting time, suggesting there are almost no delays within the production process. In contrast, backward planning shows long waiting times, as it prioritizes jobs starting to be as late as possible.

4. ANALYSIS

Analysis and comparison between standard schedule, even WC and free WC schedules

The even WC schedule focuses on balancing the workload across resources, which helps to reduce the maximum completion and flow times. However, this balance comes at the cost of higher resource load peaks and slightly increased average times (completion, flow, and waiting).

Additionally, the Standard and Free WC schedules manage resource load peaks more better for k=2 and k=3. However, the even WC's approach to use resource capacities at a constant rate could be good in scenarios where resources are underutilized, despite planners may have problems with lower utilization rate due to possible bottlenecks.

While the even WC schedule reduces the time to complete all jobs and the time for the longest-waiting jobs, it also increases the resource load and the average times slightly. The choice between these schedules should be influenced by the priorities: whether the goal is to maximize overall throughput, balance resource loads, or minimize waiting times.

Comparison of forward/backward planning with the previous three schedules

Forward planning naturally has lowest completion, flow, and waiting times, outperforming the Standard, Even WC, and Free WC strategies. Despite these advantages, forward planning may lead to a higher need for rescheduling compared to backward planning, which inherently depends on the due dates and capacity constraints. Backward planning, while providing late start times to accommodate order changes, risks losing orders due to capacity constraints

Additionally, even WC and free WC schedules reduce the resource load peaks, which are not considered at all in forward and backward planning. The capacity utilization and flexibility in handling jobs. As a result, they do not measure up to forward planning in all timing metrics.

5. CONCLUSION

a. General and specific conclusions

The five different scheduling approaches give planners insights into MTO aggregate planning practices. The even WC prioritizes workload balance across resources, optimizing throughput but at the expense of higher peak loads. Conversely, standard and free WC schedules have better resource utilization, aiming to reduce the risk of bottlenecks. Forward planning is shown to be a time-efficient method, significantly reducing completion, flow, and waiting times, although

it may frequently require rescheduling in theory. Backward planning is a kind of more cautious strategy to prioritizing late starts to accommodate sudden changes in the orders. This analysis shows the importance of a suitable schedule for different priorities.

b. Practical value of this assignment

This assignment provides a framework for manufacturing planners to assess MTO scheduling approaches that match the constraints while trying to minimize the resource load peaks. This criteria is chosen to measure the performance of the MTO aggregate planning because it helps avoid overutilization and ensures smoother production flows. This comparative analysis can provide references to improve MTO scheduling.

c. Reliability of results

The reliability of the results from this assignment can be considered highly good because it was solved with MILP models in CPLEX, following the models in the lecture notes. However, the focus on specific metrics such as completion time, flow time, and waiting time, may not reveal the full effectiveness of each schedule. Utilization rate and throughput time could also be considered when measuring MTO planning.