IPM PRODUCTION ASSIGNMENT

Group 8

M. Pietruk, S. Friedlaender, N. Baehr, E. Rimkus

Tilburg University

May 2024

Part 1. Production Layout Design

Introduction

We are the managers of the company *TechnoGadget Inc.* and our goal is to create a Production Layout Design of an assembly line consisting of 15 tasks. The objectives of our Layout design are going to be: minimization of the number of workstations K^*, minimization of the idle time and balance delay D and also minimization of the workload imbalance between different workstations B. To achieve this goal we are firstly going to analyze the approach of the ranked positional weight (RPW) heuristic (in part a)), then we are going to look at possible alternatives and compare them (in part b)) and lastly we are going to simulate the production in a deterministic setting also proposing a strategy minimizing likelihood of task in-completion.

Given by the problem description the diagram/network of the production operation is given by:

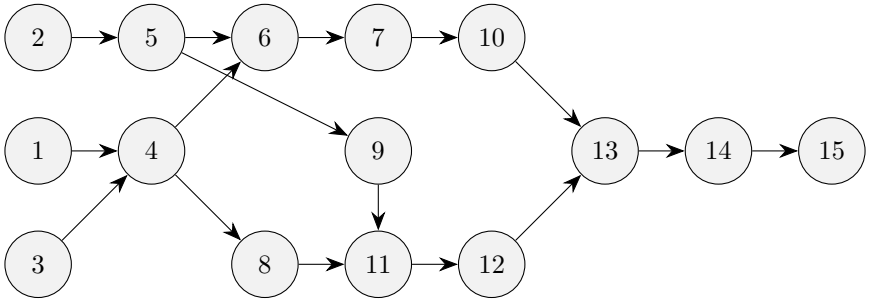


Figure 1: Diagram of the production operation

For Part C we assume some set-up parameters for the simulation of the assembly line such as that the working day of the line is 12 hours long and we compute only the products that can be done after 12 hours of work (passed through all of the WS). Moreover, we assume that the line is running the normal 5 working days a week. We setup the simulation such that there is fixed number of cycles the line can finish a day and if at any workstation the sum of simulated task times is larger than 40 minutes (cycle time) than we assume this product will continue on the belt, but it is considered to be scrap.

Quantitative models

PART A

First we define the key performance indicators so the lowest number of workstations can be determined by:

$$K^* = \left\lceil \frac{\sum_{i=1}^{n} t_i}{C} \right\rceil$$

The balance delay D, which is the fraction of total time that the workstations are idle, can be obtained as:

$$D = \frac{K^*C - \sum_{i=1}^{n} t_i}{K^*C}$$

The workload imbalance B, which measure the difference in idle times between different workstations can be determined by:

$$B = \sqrt{\frac{\sum_{i=1}^{K^*}(b_i - \hat{b})}{K^*}},$$

where b_i is the idle time of Workstation i $b_i = C - \sum_{k \in WS_i} t_k$ and $\hat{b}$ is the mean idle time among workstations.

The first discussed method is the ranked positional weight (RPW) heuristic. The algorithm of this heuristic is:

1. Determine positional weights (PW) for every node. The PW is the time of the given task and the sum of all the succesor tasks. Mathematically this means: $PW_i = t_i + \sum_{r \in S_i} t_r$, where S_i is the set of successor tasks.
2. Determine ranking of the tasks by non-increasing PW value
3. Assign tasks from the ranking to the first feasible WS j such that:
   (a) Cycle time does is not exceeded
   (b) All predecessor tasks assigned to WS 1,2,...j
   (c) Zoning constraints are satisfied

PART B

For Part B we are tasked to select a different approach for the Assembly line design either the trial-and-error method or possibly some other heuristic. From researching other heuristic methods we were able to find out another common heuristic approach, which is the Longest Task Time heuristic, which essentially considers only the individual task time as the priority rule unlike the RPW.

However, we believe that the combination of the precedence rules with the zoning constraints pose such restrictions that the common heuristic approaches such as the RPW or the Longest Task Time are not well suited for this problem. Therefore, we are going to experiment with the trial-and-error approach.

PART C

Here you can see the Python code used for the simulation of weekly production line for 5 different setups. The first two setups (a and b) represent the workstation allocation as in part A and part B of this exercise. The remaining 3 setups are the result of trying to improve the efficiency of the production line by splitting the workload at workstations with highest probability of failure (simulated in function *simulate_all_workstations*. These modifications are created in order to find the best performing setup in terms of cost-productivity. Given the number of Workstations and the average number of successfully produced products the managers can decide, whether it is beneficial to invest into extra workstations. Generally for simulation purposes we use 10 000 simulations to produce consistent and reliable results. Code can be found in the Appendix.

Results and Discussion

PART A

The number of workstation is $K = 8$, Although the theoretical lowest K given that cycle time is 40 minute is given as $K* = \lceil \frac{200}{40} \rceil = 5$. Moreover, the balance delay is $D = \frac{8 \times 40 - 200}{8 \times 40} = 0.375$. So the balance delay is 37.5%, which is a lot. Finally the workload imbalance is $B = \sqrt{67.5} = 8.21$, which captures the fact that some workstations have idle time of 28 minutes and other have idle time of 7 minutes.

PART B

Even the trial-and-error approach is very limited due to the combination of constraints on zones, precedence and the cycle time of 40 minutes. These information alone, given that we are trying to be efficient in assigning the workstations leave us with fixed outcome at the last 3 workstations. Let n denote the last workstation then: $WS_n = 15, WS_{n-1} = 14, WS_{n-2} = 11, 12, 13$. After some experimentation we can arrive at a slightly adjusted allocation to the RPW method, that however, has a better workload imbalance of B = 8.03, with the number of workstations K and the balance delay remain the same (k=8, D = 37.5%).

PART C

In this exercise we analyze the simulated output for the two workstation allocation from part A and part B. From the results we can see that on average the production efficiency of the allocation in part B is slightly better.

Table 1: Table results for part A

Task	Time	PW	Ranking	Zone
1	12	118	2	2
2	16	119	1	1
3	7	113	3	1
4	13	106	4	2
5	15	101	5	1
6	16	93	6	2
7	21	77	7	1
8	16	73	8	2
9	16	73	9	2
10	16	56	11	2
11	8	57	10	1
12	9	49	12	1
13	15	40	13	1
14	12	25	14	2
15	13	13	15	1

Table 2: Allocation to Workstations Part A

	WS 1	WS 2	WS 3	WS 4	WS 5	WS 6	WS 7	WS 8
Tasks	2, 3, 5	1, 4	6, 8	7	10, 9	11, 12, 13	14	15
Time	33	25	32	21	32	32	12	13
Idle time	7	15	8	19	8	8	28	27

Table 3: Allocation to Workstations part B

	WS 1	WS 2	WS 3	WS 4	WS 5	WS 6	WS 7	WS 8
Tasks	2, 3, 5	9, 1	6, 4	7	10, 8	11, 12, 13	14	15
Time	33	28	29	21	32	32	12	13
Idle time	7	**12**	**11**	19	8	8	28	27

Furthermore, we are considering options to increase the efficiency of this production and that is why we decided to try experiment with splitting the workstations with highest probability of failure (sum of task times > cycle time) and split them into two.

As can be seen the in table 4 in terms of purely production the best setup is setup D, which produces weekly on average 9.87 products. This calculation already takes in fact that there will be less cycles in a day since the number of Workstations increased. Still from a cost perspective of this proposal it means employing 2 extra workstations when compared to Setup B or 1 extra when compared to setup C. It is thus crucial to evaluate if the extra benefit of the production overrules the cost of the extra workstations.

Setup	# WS	Avg. # completed items	CI completed items	Incompletion likelihood
A	8	8.46	(8.41, 8.52)	0.831
B	8	8.51	(8.45, 8.56)	0.830
C	9	9.36	(9.31, 9.41)	0.792
D	10	9.87	(9.82, 9.92)	0.753
E	13	9.80	(9.76, 9.85)	0.608

Table 4: Averaged weekly simulation results for n_simulations = 10 000; CI completed items is the 95% Confidence Interval of the completed items

Note that there are also different approaches to increase efficiency of the production such as to revise the cycle time or to perhaps even adapt a dynamical cycle time that would not be constant in time but would be such that the line waits till the last workstation is done. These are further steps that could be taken be the managers of the company to increase efficiency.

Part 2

Introduction

The company AlpineThreads is selling jackets and is thinking of the best way to maximise profits (minimise costs). Given forecasted demand for the peak months of sales and known costs such as wages, material, hiring etc, we will provide the company with the optimal division of labour and distribution of factors in order to solve for the optimal solution.
In Part A, we will develop a Mixed Integer Linear Programming Model to establish the aggregate production plan for AlpineThreads. Once established, in part B1, we will implement the model into the production of the company. In Part B2, an integer requirement is relaxed an a sensitivity analysis report is produced.In Part C, AlpineThread facing competition and potential price reduction from the competitor SnowRidge, solutions for certain scenarios will be developed in order to maximise profits.

Quantitative models

The model used for this exercise is the Mixed Integer Linear Programming Model (MILP).
We will propose an objective function (a profit function), which will be set to be maximised. The whole model is presented and explained in detail in part A below.

Results and Discussion

Part A

The objective function of the MILP is a profit function, which is set to be maximised. A list of variables is given: Some decision variables have to be defined:

Decision variable	Explanation
h_t	employees hired during period t
l_t	employees laid off during period t
w_t	total workforce during period t
o_t	total overtime hours during period t
i_t	outstanding inventory at the end of period t
s_t	number of stockouts during period t
p_t	number of produced jacket during period t

Table 5: Decision variables

- $i_t = i_{t-1} + p_t - d_t$
  current inventory is inventory at the end of previous period plus number of produced jackets take away jackets sold (demand)
- $w_t = w_{t-1} + h_t - f_t$
  current workforce is workforce at the end of previous period plus number of hired workers take away fired workers
- $p_t = w_t \cdot \frac{20 \cdot 8}{4} + \frac{o_t}{4}$
  current production is number of workers times number of working days (20) multiplied by number of working hours (8) divided by hours needed to produce at jacket (4) plus overtime hours worked divided by hours needed to produce a jacket.

List of parameters, their abbreviations and values are listed in a table:

Abbreviation	Explanation	Value
RT	Standard Work Rate	\$17 per hour
OT	Overtime Rate	\$27 per hour
	Labour Hours per jacket	4
	Working days per month	20
	Working hours per day	8
OL	Overtime limit per employee	27
RE	Number of regular employees	54
TE	Maximum number of temporary employees	19
RC	Hiring costs per temporary employee	543
TC	Firing costs per temporary employee	1010
MC	Material Cost per Jacket	335
IC	Inventory Holding Cost per Jacket	14
II	Initial Inventory	1094
EIC	Excess Inventory Cost per Jacket	461
P	Selling Price per Jacket	847

Table 6: Parameters

The objective function is given by:

maximise $\sum_{\mathbf{i=1}}^{\mathbf{6}} \mathbf{d_t} \cdot \mathbf{P} - \mathbf{20} \cdot \mathbf{8w_t} + \mathbf{h_t} \cdot \mathbf{RC} + \mathbf{f_t} \cdot \mathbf{TC} + \mathbf{o_t} \cdot \mathbf{OT} + \mathbf{i_t} \cdot \mathbf{IC} + \mathbf{s_t} \cdot \mathbf{P} + \mathbf{p_t} \cdot \mathbf{MC}$

Subject to:

$i_t = i_{t-1} + p_t - d_t$	for t=1,2,3,4,5,6
$w_t = w_{t-1} + h_t - f_t$	for t=1,2,3,4,5,6
$p_t = w_t \cdot \frac{20 \cdot 8}{4} + \frac{o_t}{4}$	for t=1,2,3,4,5,6
$o_t \leq w_t \cdot OL$	for t=1,2,3,4,5,6
$h_t \leq TE$	for t=1,2,3,4,5,6
$f_t \leq TE$	for t=1,2,3,4,5,6
$RE \leq w_t \leq RE + TE$	for t=1,2,3,4,5,6
$w_t, h_t, f_t, o_t, i_t, s_t, p_t \geq 0$	for t=1,2,3,4,5,6
$w_t, h_t, f_t, o_t, i_t, s_t, p_t$ are integer	for t=1,2,3,4,5,6
$i_t = 0$	t=6

Outstanding inventory, workforce and production per period t are such as the corresponding definitions; the overtime hours cannot be greater than the product of workforce and the overtime limit per worker; number hired and fired workers cannot be greater than the maximum number of temporary employees; the workforce cannot be lower than the number of permanent workers and cannot be greater than the maximum possible workforce (permanent and temporary workers); the variables have to nonnegative integers and there cannot be any outstanding inventory in the last period. We intend to have to no idle inventory and workers.

Part B1

Having implemented the objective function for the given demand:

Month	Demand
October	2841
November	2539
December	4187
January	3762
February	2200
March	2167

Table 7: Demand

and used Solver in Excel the following results are in:

Month	**Hired**	**Laid Off**	**Workforce**	**Overtime**	**Inventory**	**Stockout**	**Production**
Sep	0	0	54	0	1094	0	0
Oct	0	0	54	0	1820	0	2160
Nov	18	0	72	0	2161	0	2880
Dec	0	0	72	0	854	0	2880
Jan	0	0	72	112	0	0	2908
Feb	0	17	55	0	0	0	2200
Mar	0	1	54	28	0	0	2167

Table 8: Variables' optimal values

Month	**Hiring**	**Lay off**	**Workforce costs**	**Overtime**	**Inventory**	**Stockout**	**Production**	**Total**
Oct	-	-	$146,880	-	$25,480	-	$723,600	$895,960
Nov	$9,774	-	$195,840	-	$30,254	-	$964,800	$1,200,668
Dec	-	-	$195,840	-	$11,956	-	$964,800	$1,172,596
Jan	-	-	$195,840	$3,024	-	-	$974,180	$1,173,044
Feb	-	$17,170	$149,600	-	-	-	$737,000	$903,770
Mar	-	$1,010	$146,880	$756	-	-	$725,945	$874,591

Table 9: Costs for variables' optimal values

Revenue	$13,796,783.00
Cost	$6,220,629.00
Profit	$7,576,154.00

The optimal production plan will yield profit of $7,576,154.00.
There were 18 hired workers for the busiest part of the season (November - February). Overtime was needed in January and March for the production to meet the demand.

Part B2

In Part B2, the integer requirement is relaxed. As seen in the answers, even though the result of the objective function has improved, it doesn't make sense as a noninteger number workers cannot be hired.

Month	Hired	Laid Off	Workforce	Overtime	Inventory	Stockout	Production
Sep	0	0	54	0	1094	0	0
Oct	0	0	54	0	1820	0	2160
Nov	16.7	0	70.7	0	2109	0	2828
Dec	2.3	0	73	0	842	0	2920
Jan	0	0	73	0	0	0	2920
Feb	0	18	55	0	0	0	2200
Mar	0	0.825	54.175	0	0	0	2167

Table 10: Variables' optimal values

Month	Hiring	Lay off	Workforce costs	Overtime	Inventory	Stockout	Production	Total
Oct	-	-	$146,880.00	-	$25,480.00	-	$723,600.00	$895,960.00
Nov	$9,068.10	-	$192,304.00	-	$29,526.00	-	$947,380.00	$1,178,278.10
Dec	$1,248.90	-	$198,560.00	-	$11,788.00	-	$978,200.00	$1,189,796.90
Jan	-	-	$198,560.00	-	-	-	$978,200.00	$1,176,760.00
Feb	-	$18,180.00	$149,600.00	-	-	-	$737,000.00	$904,780.00
Mar	-	$833.25	$147,356.00	-	-	-	$725,945.00	$874,134.25

Table 11: Costs for variables' optimal values

Revenue	$13,796,783.00
Cost	$6,219,709.25
Profit	$7,577,073.75

The sensitivity report is provided in the Appendix. A few takeaways from it are:

- In the reduced costs section, many variables regarding the hiring or laying off costs have higher negative value than corresponding objective coefficient. So if the costs were reduced below 0 (negative costs positively contribute to the revenue), it would mean that the company is actually being paid to hire/fire workers, which doesn't make sense in the real life. A similar case could be made for inventory costs; if the costs became negative (someone would be paying for the company to store its excess production), the company would opt for that option, but it doesn't make sense for that case to occur.

- Regarding reduced costs for overtime, company would likely opt for extra overtime, given it was cheaper. In some months, it could make sense as the overtime rate would still be higher than the standard work rate (e.g. in January or December). In some other months, the overtime rate would match or even be lower than the standard work rate (e.g. November, March), which very unlikely would incentivise workers to work extra hours.

- In the shadow pricing section, we find that relaxing the constraint regarding the workforce (having one extra worker) in December, would improve profits of the company by $127. Similarly in February and March, had the company increased the number of idle workers by up to 18, it could have saved $1010 per worker. Other shadow prices are either 0 (so the constraint is binding) or negative, which wouldn't improve the objective function.

Part C

In part C, 6 certain scenarios are created depending on decisions from AlpineThread (AT) and SnowRidge (SR).

1. Both AT and SR implement the promotion in October
2. Only AT implements the promotion in October
3. Only SR implements the promotion in October
4. Knowing that SR implemented the promotion in October, AT implements the strategy in December
5. Knowing that SR did not implement the promotion in October, AT implements the strategy in December
6. Neither company implements the promotion (scenario from part A).

Scenario 1

Given both companies will have opted for a promotion, the demand for AT's jackets will change for the promotion month (October) and so will the price of the jacket ($797). The demand for other months is the same

Old Demand	New Demand
1434	2841
2539	2539
4187	4187
3762	3762
2200	2200
2167	2167

Table 12: Demand Comparison

Month	Hired	Laid Off	Workforce	Overtime	Inventory	Stockout	Production
Sep	0	0	54	0	1094	0	0
Oct	19	0	73	0	1173	0	2920
Nov	0	0	73	0	1554	0	2920
Dec	0	0	73	252	350	0	2983
Jan	0	0	73	1968	0	0	3412
Feb	0	18	55	0	0	0	2200
Mar	0	1	54	28	0	0	2167

Table 13: Variables' optimal values

The company has increased the revenue as well as faced higher costs. There were 19 extra workers hired for the majority of the season and overtime was primarily used in January, when the number of produced jackets was the biggest.

Month	Hiring	Lay off	Workforce costs	Overtime	Inventory	Stockout	Production	Total
Oct	$10,317	-	$198,560	-	$16,422	-	$978,200	$1,203,499
Nov	-	-	$198,560	-	$21,756	-	$978,200	$1,198,516
Dec	-	-	$198,560	$6,804	$4,900	-	$999,305	$1,209,569
Jan	-	-	$198,560	$53,136	-	-	$1,143,020	$1,394,716
Feb	-	$18,180	$149,600	-	-	-	$737,000	$904,780
Mar	-	$1,010	$146,880	$756	-	-	$725,945	$874,591

Table 14: Costs for variables' optimal values

Revenue	$ 14,103,712.00
Cost	$6,785,671.00
Profit	$7,318,041.00

Scenario 2

AT faces new demand and a changed price of the jacket ($797).

Old Demand	New Demand
1434	3200
2539	2285
4187	3768
3762	3386
2200	2200
2167	2167

Table 15: Demand Comparison

Month	Hired	Laid Off	Workforce	Overtime	Inventory	Stockout	Production
Sep	0	0	54	0	1094	0	0
Oct	4	0	58	0	214	0	2320
Nov	15	0	73	0	849	0	2920
Dec	0	0	73	0	1	0	2920
Jan	0	0	73	1860	0	0	3385
Feb	0	18	55	0	0	0	2200
Mar	0	1	54	28	0	0	2167

Table 16: Variables' optimal values

Month	Hiring	Lay off	Workforce costs	Overtime	Inventory	Stockout	Production
Oct	$2,172	-	$157,760	-	$2,996	-	$777,200
Nov	$8,145	-	$198,560	-	$11,886	-	$978,200
Dec	-	-	$198,560	-	$14	-	$978,200
Jan	-	-	$198,560	$50,220	-	-	$1,133,975
Feb	-	$18,180	$149,600	-	-	-	$737,000
Mar	-	$1,010	$146,880	$756	-	-	$725,945

There were 2 rounds of hiring workers and the limit of overtime was almost reached in January as production was highest then.

Revenue	$13,553,782.00
Cost	$6,475,819.00
Profit	$7,077,963.00

Table 17: Profit for Scenario 2

Scenario 3

AT faces a new demand, whilst the price is unchanged.

Old Demand	New Demand
1434	1004
2539	2285
4187	3768
3762	3386
2200	2200
2167	2167

Table 18: Demand Comparison

Values	Hired	Laid Off	Workforce	Overtime	Inventory	Stockout	Production
Sep	0	0	54	0	1094	0	0
Oct	0	0	54	0	2239	0	2149
Nov	6	0	60	0	2354	0	2400
Dec	0	0	60	0	986	0	2400
Jan	0	0	60	0	0	0	2400
Feb	0	5	55	0	0	0	2200
Mar	0	0	55	0	0	0	2167

Table 19: Variables' optimal values

Month	Hiring	Lay off	Workforce costs	Overtime	Inventory	Stockout	Production
Oct	-	-	$146,880	-	$31,346	-	$719,915
Nov	$3,258	-	$163,200	-	$32,956	-	$804,000
Dec	-	-	$163,200	-	$13,804	-	$804,000
Jan	-	-	$163,200	-	-	-	$804,000
Feb	-	$5,050	$149,600	-	-	-	$737,000
Mar	-	-	$149,600	-	-	-	$725,945

Table 20: Costs for variables' optimal values

Revenue	$12,544,070.00
Cost	$5,616,954.00
Profit	$6,927,116.00

There was very little hiring of extra workers and no overtime was needed for the production, which remained almost constant throughout the whole production plan.

Scenario 4

SR has opted for promotion in October and AT is opting for promotion in December. As a result, there's a decreased demand in October and November and increased decreased demand in December and January (given the consumer actions described in the problem) with changed demand in February and March. The price is $797.

Old Demand	New Demand
1434	1004
2539	2285
4187	6428
3762	3047
2200	1980
2167	1950

Table 21: Demand Comparison

Values	Hired	Laid Off	Workforce	Overtime	Inventory	Stockout	Production
Sep	0	0	54	0	1094	0	0
Oct	16	0	70	0	2890	0	2800
Nov	3	0	73	0	3525	0	2920
Dec	0	0	73	0	17	0	2920
Jan	0	0	73	440	0	0	3030
Feb	0	19	54	0	0	0	1980
Mar	0	0	54	0	0	0	1950

Table 22: Variables' optimal values

Month	Hiring	Lay off	Workforce costs	Overtime	Inventory	Stockout	Production
Oct	$8,688	-	$190,400	-	$40,460	-	$938,000
Nov	$1,629	-	$198,560	-	$49,350	-	$978,200
Dec	-	-	$198,560	-	$238	-	$978,200
Jan	-	-	$198,560	$11,880	-	-	$1,015,050
Feb	-	$19,190	$146,880	-	-	-	$663,300
Mar	-	-	$146,880	-	-	-	$653,250

Revenue	$13,305,118.00
Cost	$6,437,275.00
Profit	$6,867,843.00

Production was almost steady for the first 4 months and then decreased by given demand decreased substantially compared to previous months. Limit of extra workers was reached and overtime hours were needed in January. The profit from this scenario was the lowest one compared to other possibilities of promotion strategies.

Scenario 5

SR decided not to implement the promotion in October, whilst AT decided to implement it in December. Demand for October and November remains unchanged, whilst demand for the following months is changed. The price is $797.

Old Demand	New Demand
1434	1434
2539	2539
4187	7093
3762	3386
2200	1980
2167	1950

Table 23: Demand Comparison

Values	Hired	Laid Off	Workforce	Overtime	Inventory	Stockout	Production
Sep	0	0	54	0	1094	0	0
Oct	19	0	73	912	2808	0	3148
Nov	0	0	73	1968	3681	0	3412
Dec	0	0	73	1968	0	0	3412
Jan	0	0	73	1864	0	0	3386
Feb	0	19	54	0	0	0	1980
Mar	0	0	54	0	0	0	1950

Table 24: Variables' optimal values

Month	Hiring	Lay off	Workforce costs	Overtime	Inventory	Stockout	Production
Oct	$10,317	-	$198,560	$24,624	$39,312	-	$1,054,580
Nov	-	-	$198,560	$53,136	$51,534	-	$1,143,020
Dec	-	-	$198,560	$53,136	-	-	$1,143,020
Jan	-	-	$198,560	$50,328	-	-	$1,134,310
Feb	-	$19,190	$146,880	-	-	-	$663,300
Mar	-	-	$146,880	-	-	-	$653,250

Revenue	$14,650,454.00
Cost	$7,181,057.00
Profit	$7,469,397.00

The revenue was the highest out of all possible scenarios and so were the costs. Limit of extra workers was reached in October and plenty of overtime hours were needed, which lead to extra costs in the first 4 months.The production was almost cut in half in months of February and March compared to production in months October-January.

Best case scenario for AT would be such that no company decides to implement the promotion. Worst case scenario for AT would be to implement the promotion in December after RS had done so in October. The company could implement the promotion in October and possibly have up to $390,925 more in profits had it not implemented it or AT miss out on $498,194 had it implemented it. Choosing to promote in December would not make any sense as the profits from the promotion would be less it than opting for a promotion in October or no promotion at all.

Scenario	**Expected profit**
1	$7,318,041.00
2	$7,077,963.00
3	$6,927,116.00
4	$6,867,843.00
5	$7,469,397.00
6	$7,576,154.00

Table 25: Scenarios' profits

Part 3. Lot sizing and MRP

Part 3A

There is a manufacturing process of Widget Corporation, which produces product, called Part A. To produce the key product, one has to integrate 3 components: Part B, C, and D. Table 26 below presents the part explosion details.

Table 26: Part explosion details

Part	Required Quantity	Procurement status	Lead Time (Weeks)
A	-	Manufactured	1
B	5	Purchased	2
C	2	Manufactured	3
D	4	Purchased	5

Each part has different costs and annual demands. Inventory costs are calculate by assuming that each year has 50 weeks (Table 27).

Table 27: Costs and other data for parts

Part	Setup Cost $(/order)	Lot Sizing Rule	Current Inventory	Weekly Inventory costs	Scheduled receipts
A	55	M	1	0.38	-
B	55	P	2	0.3	-
C	55	M	3	0.18	-
D	55	P	5	0.12	25

Note: scheduled receipts are for period 10.

The order for production is as follows: to produce item A, items B and items C are required, whereas producing item C requires item D.

Item A

It is given that to manufacture item A, 5 items B and 2 items C are needed. As the production of item B has the strategy Lot for Lot (L4L), the amount produced at each period is exactly equal to the demand at that period. Hence, the final MRP for item A is shown in Table 28.

L4L method does not have any inventory holding costs, however the setup costs have to be paid each period, therefore the method is relatively expensive and not the economically optimal one. Nevertheless, it gives the freedom for management to produce as many items as wanted in each period.

Table 28: MRP schedule for Item A

Period	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**
Gross requirements				11	22	30	38	34	19	11
Scheduled receipts										
Planned order release (L4L)			11	22	30	38	34	19	11	
Planned deliveries				11	22	30	38	34	19	11
On-hand inventory										

Item B

For item B to be purchased, the orders have to be made 2 weeks in advance. For one item A one requires 5 items B. This item is purchased according to the EOQ rule, i.e. to determine the optimal quantity we use the EOQ formula:

$$EOQ = \sqrt{\frac{2AD}{h}}$$

where the setup cost $A = 55€$, the demand $D = \frac{1}{7}\sum_{t=13}^{19} D_t$ (average gross requirement from period 13 to period 19), and inventory cost per unit per week $h = \frac{15}{50} = 0.30€$. After computing it with the following formula, we get that the order quantity is equal to 208. Hence, order 208 every time when the ending inventory is coming negative in the subsequent period. The MRP schedule for Item B can be seen in table 29.

Under EOQ, total inventory costs are minimized. As seen in our case the demand is fluctuating quite a lot, whereas EOQ assumes that the demand is constant. Therefore, the EOQ approach might not be the optimal one and the setup as well as inventory costs could potentially be reduced using other methods.

Table 29: MRP schedule for Item B

Period	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**
Gross requirement			55	110	150	190	170	95	55	
Scheduled receipts										
Planned order release (EOQ)	208		208	208	208					
Planned deliveries			208		208	208	208			
Ending inventory	20	20	153	43	101	119	157	62	7	7

Item C

To produce item A, there is a need for 2 items C to be manufactured. The waiting time until its manufactured is 3 weeks. We use the Silver-meal heuristic approach to determine the schedule of manufacturing.

Before computing the schedule we determine inventory cost for item C per unit per week $h = \frac{9}{50} = 0.18€$.

Below is the algorithm of how to compute this schedule.

1) Start in period 11. Then, $C(1) = A = 55$.

$C(2) = \frac{A+h\times D_{15}}{2} = \frac{55+0.18\times 60}{2} = 32.9$

$C(3) = \frac{55+0.18\times 60+2\times 0.18\times 76}{3} = 31.05$

$C(4) = \frac{55+0.18\times 60+2\times 0.18\times 76+3\times 0.18\times 68}{4} = 32.47$

Since $C(4) > C(3)$, at period 11 we produce $y_{11} = 36 + 60 + 76 = 172$

2) Now, we start in period 14 $C(1) = 55$

$C(2) = 30.92$

$C(3) = 23.25$

Since average cost per period is diminishing, we produce $y_{14} = 128$ at period 14. The final MRP schedule for item C is shown in table 30.

Table 30: MRP schedule for Item C

Period	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**
Gross-requirements				22	44	60	76	68	38	22	
Scheduled receipts											
Planned order release (S-M)		172			128						
Planned deliveries					172			128			
On-hand inventory	30	30	30	8	136	76	0	60	22	0	

The Silver-Meal heuristic makes use of determining the average cost per period based on how long the current order lasts.

Item D

For one unit of item C to be manufactured we need to purchase 4 items D 5 weeks in advance. Also, for item D we have 25 scheduled receipts for period 10. To purchase this item we make use of Wagner-Whitin algorithm.

Having inventory cost of $h = 0.12€$ for item C per unit per week, and setup costs of $A = 55€$, we determine the arc cost matrix to be:

i,j	**6**	**7**	**8**	**9**	**10**
6		55	55	55	128.4
7			55	55	104
8				55	79.48
9					55

Usually, for Wagner-Whitin algorithm, we solve the system of equations of $f_k = \min_{j>k}(c_{kj} + f_j)$ for $k = 1, ..., n$ with the initial condition of $f_{n+1} = 0$, where f_k is defined as the minimum cost starting at node k. However, since ordering is required only in two periods, it is clearly seen from the matrix above that the costs are optimal when the quantity required is ordered at exactly the same period when it is required with time-phased requirements. The final MRP table for item D is seen in Table 31.

Table 31: MRP for Item D

Period	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**
Gross requirements							663	0	0	512
Scheduled receipts						25				
Planned order release (WW)		598	0	0	204	0				
Planned deliveries							598	0	0	204
On-hand inventory	40	40	40	40	40	65	0			

Part 3B

We are introduced with the capacity constraint of 30 per period (Table 32)

Table 32: Table with capacity constraint introduced

Period	12	13	14	15	16	17	18	19	20
Demand	0	0	11	22	30	38	34	19	11
Time-phased production	0	11	22	30	38	34	19	11	0
Capacity	30	30	30	30	30	30	30	30	30

Let capacity be c_i and demand d_i for period i. Since the cumulative capacity is larger than the cumulative demand for periods 13-19, i.e. $\sum_{i=13}^{19} d_i \leq \sum_{i=13}^{19} c_i$, the production schedule is feasible.

We rework the production to adjust for the capacity constraint to obtain Table 33

Table 33: Table with capacity constraint introduced

Period	12	13	14	15	16	17	18	19	20
Demand	0	0	11	22	30	38	34	19	11
Time-phased production	0	15	30	30	30	30	19	11	0
Capacity	30	30	30	30	30	30	30	30	30
End inventory	0	4	12	12	4	0	0	0	0

To calculate the total costs of this production schedule, we need to compute holding costs and fixed order costs. Given that a inventory cost per unit per week is -.345. Then, the total cost T is equal to $0.345 \times (4 + 12 + 12 + 4) + 7 \times 55 = 396.1\$$.

Then, we try to enhance the feasible solution by reducing fixed order costs, hence we try to stop production in period 19 (Table 34).

Table 34: Table without production in period 19

Period	12	13	14	15	16	17	18	19	20
Demand	0	0	11	22	30	38	34	19	11
Time-phased production	0	15	30	30	30	30	30	0	0
Capacity	30	30	30	30	30	30	30	30	30
End inventory	0	4	12	12	4	0	0	11	0

Applying the same calculation, we get that the total cost is equal to \$344.9 (calculated in the same way as previously by summing fixed costs and inventory holding costs). Hence, the cost is reduced by shutting off production in period 19. Since due to the capacity constraint it is not possible to shut off production at any of the periods, Table 34 is the final enhanced feasible schedule of production with reduced costs.

4.A

Introduction:
In this exercise we are tasked to organize work order for a programmer who is tasked with completing 8 different programming project. Each of the programs has a different due date expressed with a date and a time it takes to complete it. Assuming that the programmer starts working on 1st of June and does not work during weekends we are asked to find an order of the jobs which minimizes different factors specified later. Firstly, however from the due dates we can easily obtain amount of workdays remaining until the due date. We will do that in order to simplify calculations. That can be achieved by simply counting the amount of working days between first of June and the due date given for a particular job. With this in mind the following table represents the current problem:

Table 35: Project Schedule

Job F	Time Required (days)	Due Date	Working Days Available
1	4	June 8	5
2	10	June 15	10
3	1	June 10	5
4	1	June 12	7
5	8	July 1	20
6	3	July 6	25
7	1	June 25	16
8	6	June 29	20

Minimize mean lateness

In this part we are asked to minimize mean lateness, where lateness is a difference between the flowtime of the jobs and the due dates.

Methodology: Minimizing mean lateness is equivalent with minimizing the mean flow (completion) time, which can be done by sequencing the jobs using the SPT (shortest processing time). That means ordering jobs with regard to the time required to complete them in an increasing manner.
Results:

Table 36: Project Schedule sorted Using SPT (Shortest Processing Time) to Minimize Mean Lateness

Job	Time Required (days)	Due Date	Working Days Available	Flowtime	Lateness
3	1	June 10	5	1	-4
4	1	June 12	7	2	-5
7	1	June 25	16	3	-13
6	3	July 6	25	6	-19
1	4	June 8	5	10	5
8	6	June 29	20	16	4
5	8	July 1	20	24	-4
2	10	June 15	10	34	24

Such an ordering minimizes the mean lateness, which is equal to -1.25

Minimize amount of tardy jobs

In this part we are asked to find a schedule of jobs, which minimizes the amount of jobs that end up being late. Late jobs are those for which the difference between Flowtime and due date is positive.
Methodology: To minimize this particular variable we can apply the Moore Algortihm. This algorithm consists of several different steps:

1. Order the jobs using the EDD, meaning ordering them with regard to the due date in such a way that the earliest date is first and the latest is last.

2. Calculate the lateness of each of those jobs (not to confuse with tardiness, which can also take positive values) and find the first job that is late. If there are none go to the last step

3. Consider all the jobs before the one that is late first and the late job itself. Out of these jobs remove the one with the highest processing time.

4. Repeat the process until none of the jobs are late. The deleted jobs should be ordered after the jobs that remained. The order of those deleted jobs is insignificant as they will be the only ones that are late so their order does not impact the amount of late jobs.

Results of this algorithm are presented below.

Step 1

Order the jobs with regards to the Due date in an increasing manner. Find the job which is late and out of the jobs preceding it and itself find a job with highest processing time and delete it.

Table 37: Project Schedule Sorted by Due Date

Job	Time Required (days)	Due Date	Working Days Available	Flowtime	Tardiness
1	4	June 8	5	4	0
3	1	June 10	5	5	0
4	1	June 12	7	6	0
2	10	June 15	10	16	6
7	1	June 25	16	17	1
8	6	June 29	20	23	3
5	8	July 1	20	31	11
6	3	July 6	25	34	9

The first job that is late is job 2, and among jobs 1,3,4 and 2 it has the highest processing time and thus should be taken out.

Step 2

After removing job 2 the table representing the project schedule looks like it follows:

Table 38: Project Schedule Sorted by Due Date after removing job 2

Job	Time Required (days)	Due Date	Working Days Available	Flowtime	Tardiness
1	4	June 8	5	4	0
3	1	June 10	5	5	0
4	1	June 12	7	6	0
7	1	June 25	16	7	0
8	6	June 29	20	13	0
5	8	July 1	20	21	1
6	3	July 6	25	24	0

This time the only tardy job is job 8 and and among all the proceeding jobs it has the highest processing time so it will be removed.

Step 3

After removing job 8 we are left with the following table:

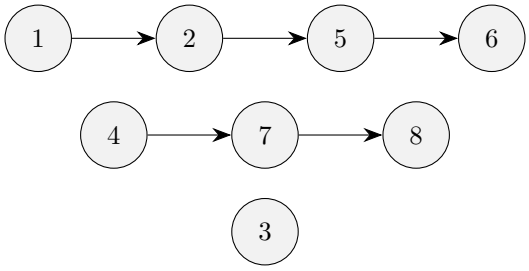
Table 39: Project Schedule Sorted by Due Date after removing job 2 and 8

Job	Time Required (days)	Due Date	Working Days Available	Flowtime	Tardiness
1	4	June 8	5	4	0
3	1	June 10	5	5	0
4	1	June 12	7	6	0
7	1	June 25	16	7	0
5	8	July 1	20	15	0
6	3	July 6	25	18	0

No more jobs are late now so we have reached the final order. Therefore, the final optimal order that minimizes amount of tardy jobs is 1-3-4-7-5-6-(2-8), where the 2 jobs in the brackets can be put in any order. The minimal amount of tardy jobs is 2.

Minimize maximum latnesss subject to precedence condtions

In this part we aim to minimize the maximum lateness subject to precedence conditions as presented below. The precedence conditions dictate that certain jobs have to be done earlier in order for some different jobs to be doable. The graph below presents the precedence conditions in this exercise:



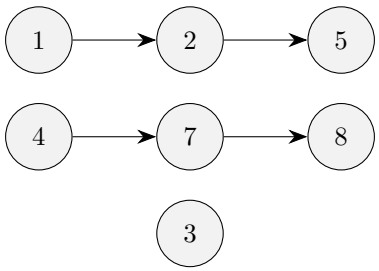
An algorithm that is used in order to minimize maximum lateness subject to the precedence constraints is called Lawler's Algorithm. This algorithm has the following :

1. Start each stage with determining the set of jobs V, which are the jobs which are available (meaning all their successors have been already completed or there are no successors).
2. From this set of jobs V, choose one job k for which the equation $\tau - d_i$ equals the smallest amount. In this case τ is the time required to complete all the jobs and d_i is the amount of working days available of job i.
3. Put job k in the last available position
4. Set $\tau := \tau - t_k$
5. Update set V
6. Repeat those steps until all the jobs are scheduled.

Below we can observe how this process looks in the given example.

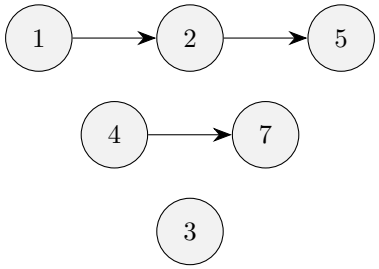
Step 1

$\tau = 34$
Possible candidates in this step are 3 ,6 and 8 : min{34-5, 34-23, 34-20}=11
Therefore, put job 6 in spot 8 The newly update precedence conditions look like that:



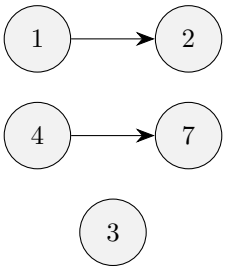
Step 2

$\tau = 34 - 3 = 31$
Possible candidates in this step are 3 ,5 and 8 : min{31-5, 31-20, 31-20}=11
Since both 5 and 8 result in the same value we are indifferent. In this case we will put job 8 in spot 7, however either one would work. The newly update precedence conditions look like that:



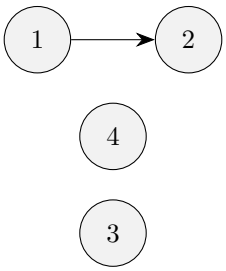
Step 3

$\tau = 31 - 6 = 25$
Possible candidates in this step are 3 ,5 and 7 : min{25-5, 25-20, 25-16}=5
Therefore, put job 5 in spot 6 The newly update precedence conditions look like that:



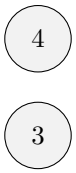
Step 4

$\tau = 25 - 8 = 17$
Possible candidates in this step are 2 ,3 and 7 : min{17-10, 17-5, 17-16}=1
Therefore, put job 7 in spot 5 The newly update precedence conditions look like that:



Step 5

$\tau = 17 - 1 = 16$
Possible candidates in this step are 2 ,3 and 4 : min{16-10, 16-5, 16-7}=6
Therefore, put job 2 in spot 4 The newly update precedence conditions look like that:



So from this point on the precedence conditions are no longer relevant.

Step 6

$\tau = 16 - 10 = 6$
Possible candidates in this step are 1 ,3 and 4 : min{6-5, 6-5, 6-7}=-1
Therefore, put job 4 in spot 3

Step 7

$\tau = 6 - 1 = 5$
Possible candidates in this step are 1 ,3 : min{5-5, 5-5}=0
The 2 remaining jobs have the same due date so in this case we are indifferent to which one we place as the first one and which one as the second one.

Thus the final order is 1-3-4-2-7-5-8-6, 3-1-4-2-7-5-8-6, 1-3-4-2-7-8-5-6 or 3-1-4-2-7-8-5-6
In such a schedule maximum lateness equals 11.

Table 40: Project schedule using order 1-3-4-2-7-5-8-6

Job	Time Required (days)	Due Date	Working Days Available	Flowtime	Lateness
1	4	June 8	5	4	-1
3	1	June 10	5	5	0
4	1	June 12	7	6	-1
2	10	June 15	10	16	6
7	1	June 25	16	17	1
5	8	July 1	20	25	5
8	6	June 29	20	31	11
6	3	July 6	23	34	11

Table 41: Project schedule using order 3-1-4-2-7-5-8-6

Job	Time Required (days)	Due Date	Working Days Available	Flowtime	Lateness
3	1	June 10	5	1	-1
1	4	June 8	5	5	0
4	1	June 12	7	6	-1
2	10	June 15	10	16	6
7	1	June 25	16	17	1
5	8	July 1	20	25	5
8	6	June 29	20	31	11
6	3	July 6	23	34	11

Table 42: Project Schedule sorted Using order 1-3-4-2-7-8-5-6

Job	Time Required (days)	Due Date	Working Days Available	Flowtime	Lateness
1	4	June 8	5	4	-1
3	1	June 10	5	5	0
4	1	June 12	7	6	-1
2	10	June 15	10	16	6
7	1	June 25	16	17	1
8	6	June 29	20	23	3
5	8	July 1	20	31	11
6	3	July 6	23	34	11

Table 43: Project Schedule sorted Using order 3-1-4-2-7-8-5-6

Job	Time Required (days)	Due Date	Working Days Available	Flowtime	Lateness
3	1	June 10	5	1	-4
1	4	June 8	5	5	0
4	1	June 12	7	6	-1
2	10	June 15	10	16	6
7	1	June 25	16	17	1
8	6	June 29	20	23	3
5	8	July 1	20	31	11
6	3	July 6	23	34	11

4.B

In this exercise a software developer is preparing to do a series of software development tasks for 6 different clients. Before the programmer can start working on a task an assistant has to review the task. In the following table we are given estimated time for completing each job for both the assistant an the programmer. Given those estimated times we are asked to find an optimal order of those jobs in order to minimize the total makespan, that is the total time the whole process takes. The estimated times are giving in the following table:

Table 44: Programming task schedule

Item	Client Assistant Time (min)	Programmer Time (min)
1	59	111
2	91	219
3	113	83
4	105	373
5	186	298
6	23	67

We can treat this problem as a minimizing the makespan problem for 2 "machines". Theory tell us that for both the assistant and the programmer the order of tasks will be the same and it can be determined using the Johnson's algorithm, which works as follows:

1. List the values of A_is(estimated time it will take the assistant to complete job i) and B_is (estimated time it will take the programmer to complete job i) in two columns.

2. Find the smallest remaining elements in two columns:

(a) If the value belonged to column A schedule that job as the next one

(b) If the value belonged to column A schedule that job last

(c) If the values in both columns are the same the placement can either be as the next one or at the end.

3. Cross of the already scheduled jobs and continue the process until all the jobs are scheduled.

Using this algorithm we can easily calculate the optimal schedule for those task:

1. Firstly the minimal job time in the table is 23, which is the assistants time, for job 6. Therefore 6 goes to spot 1.
2. Next minimal job time is 59, which is once again the assistants time for job 1, hence job 1 goes to the second spot.
3. Now the minimal job time is 83 , the programers time, for job 3, so it ends up in the sixth spot.
4. Next, the minimal time is 91 for job 2, again for the assistant, which causes it to be in the third spot
5. Next, the job to consider is job 4 with a time of 105 again for the assitant, so it goes in the fourth spot.
6. So finally job 5 goes to spot number 5 as it is the only left spot

Therefore the final order that minimizes makespan is 6-1-2-4-5-3 In this case the makespan is 1174 minutes

Figure 2: Gantt chart for ordering 6-1-2-4-5-3

and the idle time of the assistant is $1174-577 = 597$, and the idle time of the programmer is $1174-23 = 1151$

4.C

In this exercise we are still considering the problem described in part B, however this time we assume that the processing times are not deterministic but they are exponentially distributed random variables with their expected times equal to the times provided in the previous part.
Given this information we can easily transform the expected times into rates, which can be used to determine the optimal order. To do that we will simply use the formula Rates=(1/Expected Times)

Table 45: Programming task schedule - with exponentially distributed process times

Item	Assi. Time	Prog. Time	Rates for assi. (a)	Rates for prog. (p)	$a-p$
1	59	111	$\frac{1}{59}$	$\frac{1}{111}$	0.007940
2	91	219	$\frac{1}{91}$	$\frac{1}{219}$	0.006423
3	113	83	$\frac{1}{113}$	$\frac{1}{83}$	-0.003199
4	105	373	$\frac{1}{105}$	$\frac{1}{373}$	0.006843
5	186	298	$\frac{1}{186}$	$\frac{1}{298}$	0.002021
6	23	67	$\frac{1}{23}$	$\frac{1}{67}$	0.028553

The optimal order can be determined, based on the Johnson's Algorithm in a Stochastic Setting, by sequencing the jobs in a decreasing manner dependent on the a-p, so in this case the optimal schedule is 6-1-4-2-5-3

Having established this optimal order, using the knowledge of the rates of the exponentially distributed times for each job we are able to simulate the times of those jobs.
With the rates already calculated the simulation is a simple programming matter. The code that was used for this part can be found in the Appendix

The results of this simulation are showcased in this histogram:

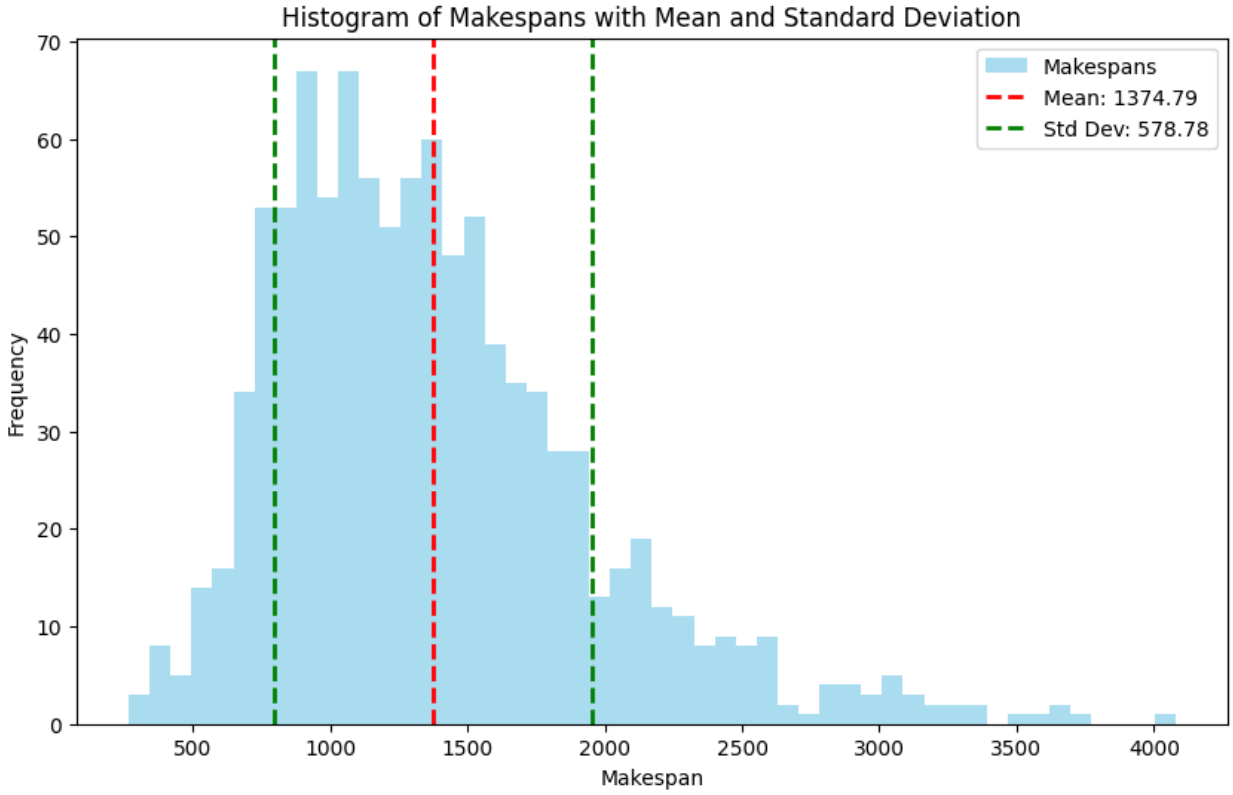


Figure 3: Histograms with the results of the simulation

The simulation resulted in the following statistics :
Average makespan: 1374.79
Variance of makespan: 334986.32
Minimum makespan: 266.27
Maximum makespan: 4077.59
Using the same simulation we can also easily calculate the % chance that the makespan from this simulation exceeds the one found in the deterministic case, which is 1174 minutes. In this case that % chance is 57.60%

4.D

In this task we consider a single machine, to which jobs arrive randomly and independently. The arrival process is a Poisson process with a mean arrival rate $\lambda = 12$ per hour. The processing time are exponentially distributed with a mean of $\frac{1}{\mu} = \frac{1}{17}$ per hour. The fact that the arrival and processing times are specified per hour is an assumption that is made, however it is not specified in the question. Those processing times are also independent are identically distributed exponential random variables. We are asked to consider different selection disciplines namely FCFS(First come first serve), LCFS(Last come last serve) and a random selection disciplines. For each of those selection disciplines we are asked to calculate the probability that the flowtime in the system exceeds 40 minutes. To calculate that we will utilize the assumption that

the wait time W is equivalent with the flow time.

FCFS

In this selection discipline, as the name suggests, whoever comes first is served first. We are asked to calculate the probability that the flow time in the system exceeds 40 minutes. In this case the flow time is equivalent to the wait time W which follows an exponential distribution with rate $\mu - \lambda$

In this case the value of $\mu - \lambda$=5. Knowing that we have to calculate

$$P\{W \geq t\} = 1 - (1 - e^{-(\mu-\lambda)t})$$

We can plug in the number to obtain the following equation:

$$P\{W \geq 40\} = e^{-(5)*\frac{2}{3}} = 0.035674$$

So in case of FCFS the probability of the process exceeding 40 minutes is 3.5674%.

LCFS

In this selection discipline, the last person to come is served first. Once again we are asked to calculate the probability of the process exceeding 40 minutes. In this case we can no longer utilize the exponential distribution of the flow time so instead we will use normal approxmation to calculate the desire probability. To do so we firstly must calculate the mean and variance of the flowtime under the LCFS distribution. The mean of flow time in that case is the same as in the FCFS case and it is:

$$E_{LCFS}(W) = \frac{1}{\mu-\lambda}$$

and the variance is equal to:

$$Var_{LCFS}(W) = \frac{1}{(\mu-\lambda)^2}(\frac{2}{1-\rho} - 1), \quad \text{where } \rho = \frac{\lambda}{\mu}$$

After plugging in the values we get that:

$$E_{LCFS}(W) = \frac{1}{5}$$
$$Var_{LCFS}(W) = \frac{1}{5^2}(\frac{2}{1-\frac{12}{17}} - 1) = 0.232$$

Knowing that we can calculate the desired probability:

$$P\{W \geq \frac{2}{3}\} = P\{\frac{W-\frac{1}{5}}{\sqrt{0.232}} \geq \frac{\frac{2}{3}-\frac{1}{5}}{\sqrt{0.232}}\} = 1 - \Phi(\frac{\frac{2}{3}-\frac{1}{5}}{\sqrt{0.232}}) \approx 1\text{-}\Phi(0.97) = 1\text{-}0.83398 = 16.602\%.$$

Random selection discipline

In this discipline people to be served or tasks to be processed are chosen randomely. To calculate probability that the process time will exceed 40 minutes we will follow very similar steps to the previous part.
Firstly we will once again calculate the mean and the variance of the waiting time under this discipline The mean of flow time in that case is the same as in the FCFS case and it is:

$$E_{Random}(W) = \frac{1}{\mu-\lambda}$$

and the variance is equal to:

$$Var_{Random}(W) = \frac{1}{(\mu-\lambda)^2}(\frac{2}{1-\frac{\rho}{2}} - 1), \quad \text{where } \rho = \frac{\lambda}{\mu}$$

After plugging in the values we know we obtain:

$$E_{Random}(W) = \frac{1}{5}$$
$$Var_{Random}(W) \approx 0.0837$$

Hence we have: Normal approximation $W_{Random}(\frac{1}{5}, 0.0837)$
Now we can easily calculate the probability:

$$P\{W \geq \frac{2}{3}\} = P\{\frac{W-\frac{1}{5}}{\sqrt{0.0837}} \geq \frac{\frac{2}{3}-\frac{1}{5}}{\sqrt{0.0847}}\} = 1 - \Phi(\frac{\frac{2}{3}-\frac{1}{5}}{\sqrt{0.0837}}) \approx 1\text{-}\Phi(1.61) = 1\text{-}0.94630 = 0.05370 = 5.37\%$$

Appendix

Part 1

Below is the code for python simulations in Part 1:

```
##### import libraries
import numpy as np
import scipy.stats as stats
import pandas as pd
import random


##### set-up

random.seed(42)

# Tasks grouped by workstation with mean times
workstations_part_a= {
    1: [16, 7, 10],  # Tasks 2, 3, 5
    2: [12, 13],     # Tasks 1, 4
    3: [16, 16],     # Tasks 6, 8
    4: [21],         # Task 7
    5: [16, 16],     # Tasks 9, 10
    6: [8, 9, 15],   # Tasks 11, 12, 13
    7: [12],         # Task 14
    8: [13]          # Task 15
}

workstations_part_b = {
    1: [16, 7, 10],  # Tasks 2, 3, 5
    2: [12, 16],     # Tasks 1, 9
    3: [16, 16],     # Tasks 6, 4
    4: [21],         # Task 7
    5: [16, 13],     # Tasks 8, 10
    6: [8, 9, 15],   # Tasks 11, 12, 13
    7: [12],         # Task 14
    8: [13]          # Task 15
}

workstations_part_c = {
    1: [16],         # Tasks 2
    2: [7, 10],      # Tasks 3, 5
    3: [12, 16],     # Tasks 1, 9
    4: [16, 16],     # Tasks 6, 4
    5: [21],         # Task 7
    6: [16, 13],     # Tasks 8, 10
    7: [8, 9, 15],   # Tasks 11, 12, 13
    8: [12],         # Task 14
    9: [13]          # Task 15
}

workstations_part_d = {
    1: [16],         # Tasks 2
    2: [7, 10],      # Tasks 3, 5
    3: [12, 16],     # Tasks 1, 9
    4: [16],     # Tasks 6
    5: [16],     # Tasks 4
    6: [21],         # Task 7
    7: [16, 13],     # Tasks 8, 10
    8: [8, 9, 15],   # Tasks 11, 12, 13
    9: [12],         # Task 14
    10: [13]          # Task 15
}

workstations_part_e = {
    1: [16],         # Tasks 2
    2: [7, 10],      # Tasks 3, 5
```

```
    3: [12],          # Tasks 1
    4: [16],          # Tasks 9
    5: [16],          # Tasks 4
    6: [16],          # Tasks 6
    7: [21],          # Task 7
    8: [16],          # Tasks 8,
    9: [13],          # Task 10
    10: [8, 9],       # Tasks 11, 12
    11: [15],         # Tasks 13
    12: [12],         # Task 14
    13: [13]          # Task 15
}

workstation_setups = {
    "A": workstations_part_a,
    "B": workstations_part_b,
    "C": workstations_part_c,
    "D": workstations_part_d,
    "E": workstations_part_e
}

results =[]

# Simulation parameters
cycle_time = 40  # minutes
work_days = 5    # per week
hours_per_day = 12 #12 hour work day of the assembly line
minutes_per_day = hours_per_day * 60
n_simulations = 10000

##### simulations functions

# simulates the work of the line for one day
def simulate_day(workstations):
    cycles_per_day = minutes_per_day / cycle_time - len(workstations)
    products_completed = 0
    products_failed = 0
    for i in range(0,int(cycles_per_day)):
        all_within_cycle_time = True
        for times in workstations.values():
            total_time = sum(np.random.exponential(scale=time) for time in times)
            if total_time > cycle_time:
                all_within_cycle_time = False
                break
        if all_within_cycle_time:
            products_completed += 1
        else:
            products_failed += 1
    return products_completed, products_failed

# simulates the work of the line for one week
def simulate_week(workstations):
    total_completed = 0
    total_failed = 0
    for k in range(work_days):
        completed, failed = simulate_day(workstations)
        total_completed += completed
        total_failed += failed
    likelihood_incompletion = total_failed/(total_failed+total_completed)
    return total_completed, total_failed, likelihood_incompletion

# used to calculate the output for simulations of n weeks
def multiple_weekly_simulations(workstations, n):
    completions = []
    avg_likelihood_incompletion = 0
    for _ in range(n):
        completed, _, likelihood = simulate_week(workstations)
        avg_likelihood_incompletion += likelihood
```

```
        completions.append(completed)
    avg_completed = np.mean(completions)
    std_completed = np.std(completions, ddof=1)
    z_score = stats.norm.ppf(0.975)
    ci_completed = z_score * (std_completed / np.sqrt(n))
    confidence_interval_completed = (avg_completed - ci_completed, avg_completed + ci_completed)
    return avg_completed, avg_likelihood_incompletion / n, confidence_interval_completed


# This function is used to caculate the average fail rate at each workstation to determine which workstation to split
def simulate_all_workstations(workstations, n_simulations):
    failure_probabilities = {}
    for workstation_id, times in workstations.items():
        failures = 0
        for _ in range(n_simulations):
            total_time = sum(np.random.exponential(scale=time) for time in times)
            if total_time > cycle_time:
                failures += 1
        failure_probability = failures / n_simulations
        failure_probabilities[workstation_id] = failure_probability
    # Sort by failure probability in descending order
    sorted_failures = sorted(failure_probabilities.items(), key=lambda item: item[1], reverse=True)
    return "; ".join(f"Workstation {wid}: {prob:.3f}" for wid, prob in sorted_failures)

##### Weekly simulation execution

for label, workstations in workstation_setups.items():
    avg_completed, avg_likelihood, ci_completed = multiple_weekly_simulations(workstations, n_simulations)
    results.append({
        "Setup": label,
        "# Workstations": len(workstations),
        "Avg Completed": f"{avg_completed:.2f}",
        "CI Completed": f"({ci_completed[0]:.2f}, {ci_completed[1]:.2f})",
        "Avg Likelihood of Incompletion": f"{avg_likelihood:.3f}"
    })

results_df = pd.DataFrame(results)
```

Listing 1: Python code for simulation of the production line

Part 2

Cell	Name	Final Value	Reduced Cost	Objective Coefficient	Allowable Increase	Allowable Decrease
$E3$	Oct Hired	0	-560	-543	560	1E+30
$F3$	Oct Laid Off	0	-993	-1010	993	1E+30
$G3$	Oct Workforce	54	0	-2720	560	993
$H3$	Oct Overtime	0	-13.5	-27	13.5	1E+30
$I3$	Oct Inventory	1820	0	-14	14	24.825
$J3$	Oct Stockout	0	-847	-847	847	1E+30
$K3$	Oct Production	2160	0	-335	14	24.825
$E4$	Nov Hired	16.7	0	-543	63.5	280
$F4$	Nov Laid Off	0	-1553	-1010	1553	1E+30
$G4$	Nov Workforce	70.7	0	-2720	63.5	433
$H4$	Nov Overtime	0	-10	-27	10	1E+30
$I4$	Nov Inventory	2109	0	-14	1.5875	10.825
$J4$	Nov Stockout	0	-847	-847	847	1E+30
$K4$	Nov Production	2828	0	-335	1.5875	10.825
$E5$	Dec Hired	2.3	0	-543	433	42.33333333
$F5$	Dec Laid Off	0	-1553	-1010	1553	1E+30
$G5$	Dec Workforce	73	0	-2720	1E+30	127
$H5$	Dec Overtime	0	-6.5	-27	6.5	1E+30
$I5$	Dec Inventory	842	0	-14	3.175	10.825
$J5$	Dec Stockout	0	-847	-847	847	1E+30
$K5$	Dec Production	2920	0	-335	26	3.175
$E6$	Jan Hired	0	-433	-543	433	1E+30
$F6$	Jan Laid Off	0	-1120	-1010	1120	1E+30
$G6$	Jan Workforce	73	0	-2720	433	127
$H6$	Jan Overtime	0	-3	-27	3	1E+30
$I6$	Jan Inventory	0	-42	-14	42	1E+30
$J6$	Jan Stockout	0	-847	-847	847	1E+30
$K6$	Jan Production	2920	0	-335	10.825	3.175
$E7$	Feb Hired	0	-1553	-543	1553	1E+30
$F7$	Feb Laid Off	18	0	-1010	433	127
$G7$	Feb Workforce	55	0	-2720	1570	1600
$H7$	Feb Overtime	0	-10	-27	10	1E+30
$I7$	Feb Inventory	0	-39.25	-14	39.25	1E+30
$J7$	Feb Stockout	0	-847	-847	847	1E+30
$K7$	Feb Production	2200	0	-335	39.25	42
$E8$	Mar Hired	0	-1553	-543	1553	1E+30
$F8$	Mar Laid Off	0.825	0	-1010	785	1600
$G8$	Mar Workforce	54.175	0	-2720	1710	1570
$H8$	Mar Overtime	0	-16.3125	-27	16.3125	1E+30
$I8$	Mar Inventory	0	-391.75	-14	391.75	1E+30
$J8$	Mar Stockout	0	-847	-847	847	1E+30
$K8$	Mar Production	2167	0	-335	391.75	39.25

Cell	Name	Final Value	Shadow Price	Constraint R.H. Side	Allowable Increase	Allowable Decrease
$E2$	Sep Hired	0	0	19	1E+30	19
$E3$	Oct Hired	0	0	19	1E+30	19
$E4$	Nov Hired	16.7	0	19	1E+30	2.3
$E5$	Dec Hired	2.3	0	19	1E+30	16.7
$E6$	Jan Hired	0	0	19	1E+30	19
$E7$	Feb Hired	0	0	19	1E+30	19
$E8$	Mar Hired	0	0	19	1E+30	19
$F2$	Sep Laid Off	0	0	19	1E+30	19
$F3$	Oct Laid Off	0	0	19	1E+30	19
$F4$	Nov Laid Off	0	0	19	1E+30	19
$F5$	Dec Laid Off	0	0	19	1E+30	19
$F6$	Jan Laid Off	0	0	19	1E+30	19
$F7$	Feb Laid Off	18	0	19	1E+30	1
$F8$	Mar Laid Off	0.825	0	19	1E+30	18.175
$G2$	Sep Workforce	54	0	73	1E+30	19
$G3$	Oct Workforce	54	0	73	1E+30	19
$G4$	Nov Workforce	70.7	0	73	1E+30	2.3
$G5$	Dec Workforce	73	127	73	1.11022E-16	0.766666667
$G6$	Jan Workforce	73	0	73	1E+30	1.11022E-16
$G7$	Feb Workforce	55	0	73	1E+30	18
$G8$	Mar Workforce	54.175	0	73	1E+30	18.825
$O3$	Oct Inventory	0	-389	0	92	668
$O4$	Nov Inventory	0	-403	0	92	668
$O5$	Dec Inventory	0	-417	0	92	668
$O6$	Jan Inventory	0	-431	0	92	668
$O7$	Feb Inventory	0	-403	0	720	33
$O8$	Mar Inventory	0	-377.75	0	33	47
$P3$	Oct Overtime	1458	0	0	1458	1E+30
$P4$	Nov Overtime	1908.9	0	0	1908.9	1E+30
$P5$	Dec Overtime	1971	0	0	1971	1E+30
$P6$	Jan Overtime	1971	0	0	1971	1E+30
$P7$	Feb Overtime	1485	0	0	1485	1E+30
$P8$	Mar Overtime	1462.725	0	0	1462.725	1E+30
$Q3$	Oct Production	0	-54	0	92	668
$Q4$	Nov Production	0	-68	0	92	668
$Q5$	Dec Production	0	-82	0	92	668
$Q6$	Jan Production	0	-96	0	92	668
$Q7$	Feb Production	0	-68	0	720	33
$Q8$	Mar Production	0	-42.75	0	33	47
$R3$	Oct Workforce	0	17	0	1	8.35
$R4$	Nov Workforce	0	-543	0	2.3	16.7
$R5$	Dec Workforce	0	-543	0	16.7	2.3
$R6$	Jan Workforce	0	-110	0	2.3	1.11022E-16
$R7$	Feb Workforce	0	1010	0	18	1
$R8$	Mar Workforce	0	1010	0	0.825	18.175

Table 46: Sensitivity report

Part 3

Here is the python code for Part 3:

```
1
2  import numpy as np
3  import pandas as pd
4  import matplotlib.pyplot as plt
5  makespans=[]
6  num_simulations = 1000
7  for i in range (1000):
8      assistant_time1=np.random.exponential(23)
9      assistant_time2=np.random.exponential(59)
10     assistant_time3=np.random.exponential(105)
11     assistant_time4=np.random.exponential(91)
12     assistant_time5=np.random.exponential(186)
13     assistant_time6=np.random.exponential(113)
14     prog_time1=np.random.exponential(67)
15     prog_time2=np.random.exponential(111)
```

```
    prog_time3=np.random.exponential(373)
    prog_time4=np.random.exponential(219)
    prog_time5=np.random.exponential(298)
    prog_time6=np.random.exponential(83)
    ass_finish_6=assistant_time6
    ass_finish_1=ass_finish_6+assistant_time1
    ass_finish_4=ass_finish_1+assistant_time4
    ass_finish_2=ass_finish_4+assistant_time2
    ass_finish_5=ass_finish_2+assistant_time5
    ass_finish_3=ass_finish_5+assistant_time3
    prog_finish_6=ass_finish_6+prog_time6
    prog_finish_1=max(ass_finish_1 ,prog_finish_6)+prog_time1
    prog_finish_4=max(ass_finish_4 ,prog_finish_1)+prog_time4
    prog_finish_2=max(ass_finish_2 ,prog_finish_4)+prog_time2
    prog_finish_5=max(ass_finish_5 ,prog_finish_2)+prog_time5
    prog_finish_3=max(ass_finish_3 ,prog_finish_5)+prog_time3
    makespans.append(prog_finish_3)



makespan_mean = np.mean(makespans)
makespan_variance = np.var(makespans)
print(f"Average makespan: {makespan_mean:.2f}")
print(f"Variance of makespan: {makespan_variance:.2f}")
print(f"Minimum makespan: {np.min(makespans):.2f}")
print(f"Maximum makespan: {np.max(makespans):.2f}")


prob_exceeding_1174 = np.mean(np.array(makespans) > 1174)
print(f"Probability of makespan exceeding 1174: {prob_exceeding_22:.4f}")


plt.figure(figsize=(10, 6))
plt.hist(makespans, bins=50, color='skyblue', alpha=0.7, label='Makespans')
plt.axvline(makespan_mean, color='red', linestyle='dashed', linewidth=2, label=f'Mean: {
    makespan_mean:.2f}')
plt.axvline(makespan_mean - np.sqrt(makespan_variance), color='green', linestyle='dashed',
    linewidth=2, label=f'Std Dev: {np.sqrt(makespan_variance):.2f}')
plt.axvline(makespan_mean + np.sqrt(makespan_variance), color='green', linestyle='dashed',
    linewidth=2)
plt.title('Histogram of Makespans with Mean and Standard Deviation')
plt.xlabel('Makespan')
plt.ylabel('Frequency')
plt.legend()
plt.show()
```

Listing 2: Python code for simulating makespan