Micron NUS-ISE Business Analytics Case Competition 2025

[BACCshot]

Contents

Question 1

Part a)

Initially we were not sure of what criteria to choose for coming up with a good loading profile but ended up using efficiency as the main criteria. Specifically, we prioritised using nodes with higher effective yields (GB per wafer * yield) as they gave us more GB per wafer, making them more efficient at meeting the TAM requirements.

We then came up with an algorithm which starts by calculating the effective yield and sorted the nodes in descending order of effective yield. Afterwards we calculated the total output of the previous loading and compared it to the TAM requirements for the quarter. If there is a need for more TAM, it increases the loading of the node with the highest effective yield. Conversely, if there was an excess, it decreases the loading of the node with the lowest effective yield; which in this case did not happen but could be adapted for another situation.

```
# Adjusting the load based on the deficit
    # Low output > we need to increase the node with the highest yield
if deficit > 0:
    for node in nodes_sorted:
        required_increase = deficit / (weeks_per_quarter * effective_yields[node])
        # Ensure that the increment is not more than prev load + 2.5k
        allowed_increase = min(max_delta, prev[node] + max_delta)
        delta = min(required_increase, allowed_increase)
        if delta > 0:
            current[node] = prev[node] + delta
            break
    # High output > we need to decrease the node with the lowest yield
else:
    for node in reversed(nodes_sorted):
        required_decrease = abs(deficit) / (weeks_per_quarter * effective_yields[node])
        # Ensure that the increment is not less than prev load - 2.5k
        allowed_decrease = min(max_delta, prev[node] - max_delta)
        delta = min(required_decrease, allowed_decrease)
        if delta > 0:
            current[node] = prev[node] - delta
            break
```

Figure 1. Code Snippet of the Algorithm used to output the loading profile

Part b)

We first identified the minute load of each node at each workstation as well as the utilization rate and copied them down in a dictionary. We then implemented our tool to help us calculate the tools per quarter at each workstation using the equation given to us.

For each quarter, we first calculate the total time at a workstation by multiplying the load values from the loading plan in Part A with the minute load for each node. To calculate the number of tools required at that workstation, we simply use the formula and divide the total tool time by the time in a week multiplied by the utilization rate. We then proceed to round up the tool requirement since Assumption 3 states that tools must be purchased to fully cover the tool requirement for each workstation and add it to a dictionary with its respective workstation key. Repeat this for every workstation at every quarter.

```
# Calculate tool requirements for each quarter
for i, quarter in enumerate(QUARTERS):
    quarter_tool_requirements = {}

    for station in workstation_minute_load:
        tool_time = 0

        # Calculate the tool_time
        for node_index, node in enumerate(["Node 1", "Node 2", "Node 3"]):
            tool_time += workstation_minute_load[station][node_index] * loading[node][i]

        # Calculate the tool requirement
        tool_requirement = tool_time / (7*24*60) * workstation_utilization[station]

        # Add the rounded up tool requirement to the dictionary
        quarter_tool_requirements[station] = math.ceil(tool_requirement)

    # Store the tool requirements for the current quarter
    tools_required_per_quarter_per_workstation[quarter] = quarter_tool_requirements

''' OUTPUT
Number of Tools per quarter per workstation
Q1'26   6   9   4   7   9   2   12  3   3   1
Q2'26   7   11  4   8   10  2   12  2   4   1
Q3'26   3   13  4   8   12  2   12  2   5   1
Q4'26   8   14  5   10  14  3   12  3   6   1
Q1'27   9   14  6   10  14  4   14  3   6   1
Q2'27   9   14  6   10  14  4   16  3   6   1
Q3'27   9   14  6   10  13  4   15  3   6   1
Q4'27   9   14  6   10  13  4   16  3   6   1
'''
```

Figure 2. Code Snippet to find the Number of Tools per Quarter per Workstation.

Afterwards we were also tasked to find the additional tool purchases required for the loading plan in part A. We noted down the initial tool count provided in Table 4 and stored that

information in a dictionary. Afterwards we simply just compare the tools required at each workstation at each quarter to that in the initial tools count. If we require more tools, indicate the additional number of required tools and store that value in another dictionary. We also should increment the initial tools count by the additional tools as the future quarters do not need to repurchase them.

```
additional_tools_required = {quarter: {} for quarter in QUARTERS}
for quarter, requirements in tools_required_per_quarter_per_workstation.items():

    for station, required in requirements.items():
        available = workstation_tools_count[station]
        additional_needed = required - available

        # Only add tools if needed
        additional_tools_required[quarter][station] = max(0, additional_needed)

        # Update the tool count
        workstation_tools_count[station] += max(0, additional_needed)

''' OUTPUT
Q1'26  0  0  0  0  0  0  0  0  0  0
Q2'26  0  0  0  0  0  0  0  0  0  0
Q3'26  0  0  0  0  0  0  0  0  1  0
Q4'26  0  0  0  0  0  1  0  0  1  0
Q1'27  0  0  1  0  0  1  0  0  0  0
Q2'27  0  0  0  0  0  0  0  0  0  0
Q3'27  0  0  0  0  0  0  0  0  0  0
Q4'27  0  0  0  0  0  0  0  0  0  0
'''
```

Figure 3. Code Snippet to find the number of additional tools we need to purchase.

Lastly, we need to find the net profit taking into account the CAPEX cost per quarter as the loss and the gain from the fixed contribution margin per GB. To find the net loss we take the sum of the number of additional tools required multiply by the CAPEX per tool, and to find the net gain, we take the sum of the GBs produced per quarter by our loading profile (stored in a dictionary during part A's process) and multiply that by the contribution margin per GB of $0.002. The net profit is net gain minus net loss.

Part c)

We chose this loading plan as it meets the TAM demand while also being extremely cost efficient. To ensure that the TAM demand is being fulfilled without causing overproduction, we have improvised a plan in which it dynamically adjusts the loading per quarter. It prioritizes high-yield nodes when increasing wafer starts, maximizing production output with minimal tool expansion, reducing the deficit of the tools available against the tool requirement. Furthermore our solution only increments it by 0 to 2500, preventing sudden fluctuations from occurring that could disrupt manufacturing. Code wise, it modifies only one node per iteration, breaks the loop when no further changes are possible and rounds the wafer starts to whole numbers. Minimizing tool requirements while maintaining production targets allows the plan to become optimal for reaching the TAM with minimal CAPEX. Making it the most efficient and financially sustainable approach.(143 Words)

Question 2

Part a)

Previously, a static expected value was used to determine capacity governing equations. However, with the use of a random distribution when considering RPT as well as using probability and statistical concepts, this can help to better our approach when determining the risk associated with the plan.

Firstly, by determining the mean and the standard deviation of each node's RPT distribution, using the sample provided(Assumption1) and treating the RPT as a random distribution(Assumption 2), we can better comprehend the RPT variability. Using these tools help to make a better estimation of processing times rather than to use a static median value. To provide further context, the mean RPT helps to give a central estimate whereas the standard deviation creates the expected range of variation which proves useful to identify potential deviations.

Using the aforementioned tool requirement formula, using the assumption that all the wafers' RPT are independent (Assumption 3), we can use the new mean RPT calculated beforehand which can then replace the static median RPT giving a better estimation on total tool demand.

Since RPT follows a probability distribution, the total processing time required per week is a random variable. The summation of processing times as well as the workstations give the cumulative processing demand as a probability distribution. An inequality is formed, P(Total Processing Time > Available Tool Time), which is the probability of the required processing exceeding the available tool hours in a given week which results in a failure to meet demand(Assumption 4). Since we are using a very large sample size in our simulations, by the Central Limit Theorem, we are able to approximate its distribution to be about normal.

Using the fact that all weeks in a quarter share the same designed loading and tool availability(Assumption 5), the weekly failure probability to estimate the chance of at least one failure occurring in a quarter is formulated using: P(At least 1 week fails) = 1 - (1 - P(failure in a single week)). This equation assumes that weekly failures are independent. Furthermore, the probability of at least one quarter failing across all quarters is: P(At least 1 quarter fail) = 1- (1 - P(quarter)). Providing a long-term risk assessment associated with RPT variability. Needless to say, the higher the probability, the greater likelihood that at least one failure is experienced.

Finally, median RPT only represents the middle value of distribution disregarding the variations and extreme values that affect total processing time. Using the mean and standard deviation helps give a more accurate estimate. However, other methods such as mode or percentile-based estimates can prove more effective if there are significant outliers. To conclude, using mean RPT provides a comprehensive risk assessment. Ensuring resources are optimized and meeting wafer output.

Part b)

When estimating RPT, mean is a better choice for analysis compared to median, this is because the curve distribution is normal and there are few outliers.The mean would consider every RPT data point and this is particularly useful for resource allocation, as it allows you to calculate the total processing time by multiplying the mean by the number of steps in each process. The

mean would account for every data point hence it would take into account outliers and the effect it would have on the output of the nodes.

Another advantage of the mean is the ability to detect changes in RPT data. If certain steps start taking longer or shorter over time, the mean will capture these shifts, making it easier to spot inefficiency and chokepoints for the system.This sensitivity is important for efficiency and identifying slow chokepoints in the system. On the other hand, the median, while stable, will not highlight these changes as rapidly, hence the manufacturer will not be able to promptly react to the changing situation.

In summary, since the data achieved is normally distributed with few outliers, the mean is the better measure for estimating RPT. It offers a complete and precise way of identifying choke points in the different nodes and steps. By using the mean, all data points are taken into account and we are able to observe even the smallest shift in processing time which could affect performance drastically, especially in this microelectronics field. Thus allowing the manufacturer to gain accurate and prompt insights into their recipe processing times. (261 Words)

Part c)

By using a fixed median RPT in Q1(b), our loading and tool purchase plan assumes a constant processing time across all wafers. However, from the new insights gained in Q2, we know that RPT fluctuates from the median. Given that the curve distribution is normal when estimating RPT data values with relatively few outliers, this means the mean is a more representative statistic than the median. However, RPT can still fluctuate and differ from the original forecast.

If the recalculated mean RPT is higher than the original median estimate, we risk underproduction. This would translate to unplanned production shortfalls and thus not meeting the demand, reducing revenue. Conversely, if the mean is lower, we would have over-allocated tools, driving up CAPEX and reducing net profit. Hence, we should adjust Q1 to mean-based calculations to refine our overall profit projections .

An acceptable risk level would depend on the strategic priorities of the company. If we target near 100% certainty of meeting demand, we must increase tool count or reduce loading in each quarter to accommodate worst-case RPTs. This will add to CAPEX but reduces the chances of expensive shortfalls. Conversely, if we accept a moderate failure rate (e.g. 5-10%), it might keep CAPEX at a minimum and still meet our targets.

To reduce risk, we could revisit our loading plan and tooling purchases in Q1(b) and substitute the mean for the median in our capacity governing equations. Any under-resourced workstations where mean-based RPT exceeds the assumed tool hours can be addressed by modest additional tool purchases or minor downward adjustment in wafer starts. This can help to sustain output without over inflating CAPEX. This helps to maintain a profitable level of output without over-investing in equipment. (288 Words)