Abbott Nutrition Network Capacity Optimisation

IE3100M Systems Design Project AY2021/2022

Group 1: Chua Wee Kian, Huang Siyuan, Mitchell Kwong, Yue Lanxin

Module Advisor: A/Prof He Shuangchi

Industrial Supervisor: Lester Eng

NUS National University of Singapore | Department of Industrial Systems Engineering & Management, Faculty of Engineering

Abbott

INTRODUCTION

ABOUT ABBOTT

Abbott is a global healthcare technology and manufacturing company that is committed to transferring life-changing technologies to create breakthrough healthcare products, diagnostic tools, and medical devices to help people over 160 countries to live a healthier life. Our project is under the Singapore regional office, which mainly focuses on project management, manufacturing support and industrial engineering. We are looking to optimise the capacity of the Abbott Nutrition Network, in particular their product line of milk powders.

PROBLEM DESCRIPTION

The project seeks to develop an analytical model for plant M, one of Abbott's manufacturing plants in the Europe and Asia region. We are required to **create an integrated applet** that would **first convert annual product demands** for plant M **into production schedules**, then **simulate the dynamics** of the manufacturing system. The applet should be easily used to maximise plant M's production capacity within the planned production time in a year. Following which, we then **generate key statistical results** for both individual machines, and the entire plant. This would help understand the manufacturing dynamics and answer key business questions with different scenarios.

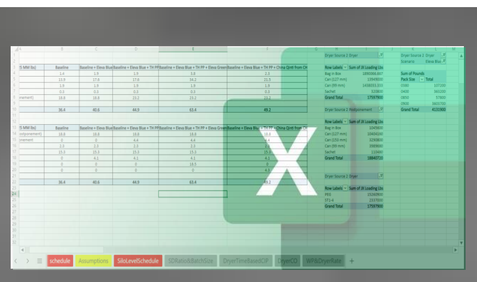
PROJECT OBJECTIVES

- The model should be robust to account for changes
- The model should be easy to use i.e. in Excel
- The model has to be a good representative of the actual manufacturing dynamics
- The model should be sustainable, through incorporating necessary functions such as data cleaning, documentation and format standardisation
- The model should track relative statistics in the simulation process, so that further output analysis, such as data visualisation, can be executed to explore the process bottlenecks.

STREAMLINED WORKFLOW

Data Analytics → Data Cleaning → Schedule Optimisation → Simulation Model → User Testing / Acceptance

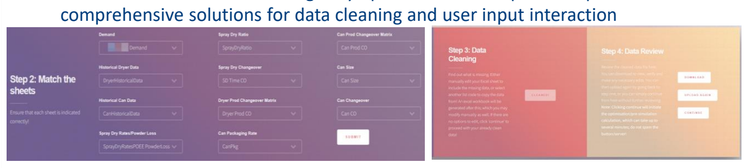
Current Solution: Half-automated Excel-based Simulation



Issues Identified ↓ **Proposed Approach**

Pre-Simulation Issued Identified

- ❑ **Repetitive and time-consuming work on data cleaning and standardization**
- ❑ ➔ Web User Interface allowing easy updates on user inputs and provide comprehensive solutions for data cleaning and user input interaction



- ❑ Scheduling heuristics is not robust
- ❑ ➔ Deployed different scheduling heuristics such as Product Group Clustering, Travelling Salesman Problem (TSP), Job Shop Problem (JSP) and TSP is found to be the most relevant and efficient

Simulation Issued Identified

- ❑ Simulation run on Excel (with VBA) is slow, and models cannot be reused
- ❑ ➔ Set up a modelling pipeline serving as infrastructure for future model building & Construct simulation logic using Python

Web User Interface for Data Cleaning

Upload of Demand Workbook:
The user accesses the applet via his web browser. He is then prompted to upload the demand workbook, which includes the demand sheet, changeover data, and other relevant information.

Matching of Workbook sheets:
The sheets then have to be matched to its corresponding function, by selecting the sheet name in a dropdown selection for each function..

Automated Data Cleaning:
Data Cleaning is done automatically, and missing values are flagged out for manual correction. The system also recommends similar data points to copy data from as an option

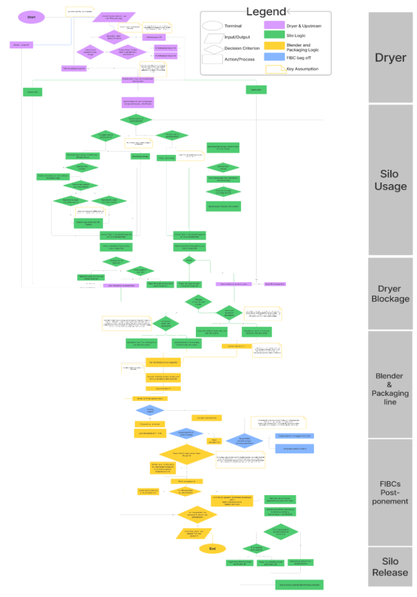
Optimized Schedule Generation

Travelling Salesman Problem: TSP helps to find the shortest path (changeover) between powder, and this method could be evaluated to be more optimized compared with the random ordering logic.

Job Shop Problem: JSP is more resource oriented in constrains compared to TSP. However, because JSP requires more computational power and time compared to TSP. We finalized the scheduling logic to follow TSP in the end.

Simulation Logic Construction

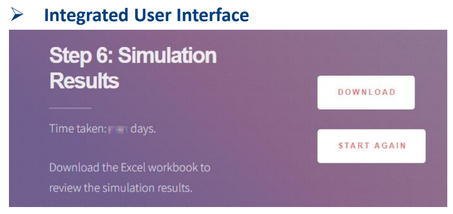
The team adopts a Simulation Flow Chart approach to visualize the simulation dynamics for the manufacturing process at our target plant.



Simulation Solution Development

After aligning and validating the simulation logic and key assumptions with Abbott, the team implemented a native simulation software in python that can compile itself into an equivalent simulation model on excel.

➢ **Integrated User Interface**



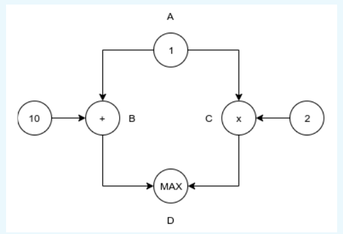
➢ **Python Simulation Model**

```
# Load data                    YOU, Now • Uncommitted changes
file = request.files['file']
data = pd.read_excel(file, sheet_name=None)
data = ad_hoc_data_cleaning(data)

# Run model
params, inputs = parse_excel_input(data)
simulation = build_simulation(params, inputs)

# Export static results
result = simulation.simulate(num_iter=len(simulation.inputs_quantity.values))
```

The integrated solution frees up analysts from repetitive work and speed up the entire analytical process from a few days to around 5 minutes (both inclusive of time waiting for missing inputs) .

SOLUTION DEVELOPMENT

A1: "1"
B1: "= 10 + A1"
C1: "= A1 * 2"
D1: "= MAX(B1, C1)"



Simulation platform
A key hurdle of implementing a fully automated web interface was interfacing with excel, which was a requirement provided to us by Abbott. To address this, the team developed a custom simulation software in python that is guaranteed to function identically to excel. Additionally, any models built with this software can also be exported to excel files with equivalent excel formulas filled in.

This is achieved by allowing developers to express simulation logic in excel syntax, which is then parsed into a dependency graph. This graph can then be traversed using simple algorithms to query the value of any cell in the simulation, or the excel formula that should be written to that cell.

Advantages of this platform
Compared to using excel as a simulation platform, this solution is far more modular and encourages reuse and standardization of logic across different simulation models. Additionally, as it runs natively in python, it enables Abbott to fully automate the simulation process, either through the web application built by our team, or through any scripts of applications they may develop in the future.

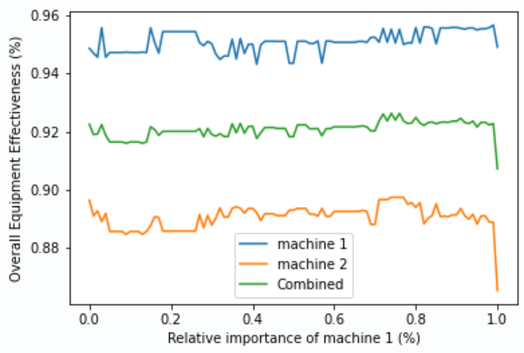
Backwards compatibility
Most importantly, adopting the platform remains compatible with current business processes as any logic implemented in python can be automatically exported to excel files, the format Abbott's current solution are using. Additionally, as excel formulas are automatically generated, it is also much easier to update formulas across multiple cells or simulations without manually editing of each cell.

SOLUTION VALIDATION: TEST CASES

1	Check silo changeover conditions: - Dynamic silo capacity reached - List code changes	A	REST	16
		A	REST	16
		A	REST	16
		B	REST	16
		A	REST	16
		B	REST	16
2	Test silo allocation	A	NORMAL	16
		A	NORMAL	16
		A	NORMAL	16
		B	NORMAL	16
		A	NORMAL	16
		B	NORMAL	16
		B	NORMAL	16
3	Check for consistent behaviour with non-perfect quantities	A	NORMAL	14
		A	NORMAL	14
		A	NORMAL	14
		B	NORMAL	14
		A	NORMAL	14
		B	NORMAL	14
		B	NORMAL	14

6	Sanity check for odd resolutions	A	NORMAL	6
		A	NORMAL	6
		A	NORMAL	4
		B	REST	6
		B	REST	6
		B	REST	4
		B	REST	6
		B	REST	6
		B	REST	4
		A	NORMAL	6
		A	NORMAL	6
		A	NORMAL	4
		A	NORMAL	6
		A	NORMAL	6
		A	NORMAL	4
		B	REST	6
		B	REST	6
		B	REST	4
		A	NORMAL	6
		A	NORMAL	6
		A	NORMAL	4

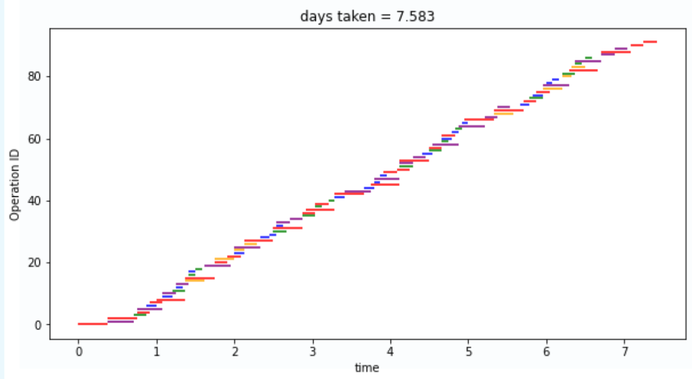
SCHEDULING RESULTS



TSP Scheduler
With the TSP scheduler, The plot on the left shows how overall equipment effectiveness (OEE) varies with the importance weights. From the results, we can see that the combined average OEE (in green) is maximized when the importance of machine 1 was set to 76% and the importance of machine 2 was set to 24%. This highlights another key benefit, the importance weights produced by our solution can yield insightful information as to what machines are most impactful on how the master order is created. These machines could then be potentially targeted for further improvements.

Additionally, in contrast to manual scheduling, our team's optimization script takes less than a second to run and is able to easily respond to real life changes such as updates to machine changeover times or the introduction of new products.

COMPARISON OF TSP AND JSP



JSP Scheduler
The plot shows the JSP scheduling results for a small sample of annual demand. The y axis represent unique operations, which correspond to powder from one demand order being processed by a particular machine, and the x axis represents time. Bars are also colored based on their demand order.

Inspecting the blue bars, we can see that they cluster closely in 5 groups that appear 5 times across the entire plot. The tight clustering is indicative of the lack of delay between machines that is achieved with a FIFO ordering, and the equally spaced 5 clusters is indicative of a successful production frequency of 5.

Limitations

- ❑ Scheduling model logic lacks validation on its optimization level
- ❑ Simulation model logic has gaps with real-world manufacturing process
- ❑ Limited test cases to test simulation model

Future Improvements

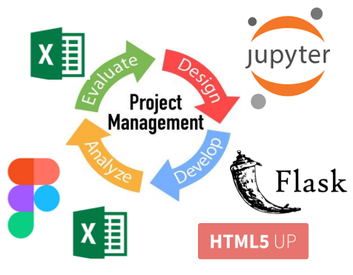
- ❑ Further optimization of scheduling model
- ❑ Complete to increase the model accuracy of simulation model
- ❑ Further improve on the user web interface with optimized model coded in
- ❑ User manual documentation

Key Skillsets

- ❑ Excel and VBA: Data Analytics, Model Validation
- ❑ Python: Data Cleaning, Schedule Optimization, Simulation Model & Modelling Pipeline Development
- ❑ Figma: Simulation Logic Flow Chart Construction
- ❑ HTML5UP: User Web Interface Development

- ❑ Key ISE Skillsets Applied:

Operations Research, Data Analytics & Visualization
Discrete Event Simulation, Project Management



Conclusion

Overall, the team has managed to set up the infrastructure for a fully automated end-to-end pipeline to perform data cleaning, scheduling and simulation in a quick convenient and replicable manner. Additionally, the modular design of the solution allows for easy modification or extension of the solution. Finally, the pyxl simulation platform used in the solution is also backwards compatible with existing processes through its ability to export simulation models to equivalent excel files. With the limitations due to time constraints and model complexity, further improvement and implementation would be proceeded with considerations, and the team has also provided documentations, user guides and other essential materials to ensure a smooth handover for future work.