40.014: Engineering Systems Architecture

Operations Scheduling

Project Report 2

ESD CH01 Group 5 Project 6

Oon Eu Kuan Eugene	1006637
Kong Le'ann Norah	1007000
Georgia Karen Lau	1007153
Tan Chong Hao	1006915
Long Yan Ting	1006944
Lee Peck Yeok	1006968
Nathan Ansel	1007492

Course Instructors:
Rakesh Nagi
Sun Zeyu

SINGAPORE UNIVERSITY OF
TECHNOLOGY AND DESIGN

Contents

1 Summary of Report 1

Project Overview
Our project aims to optimise assembly line scheduling in manufacturing while achieving one of the following objectives: minimising makespan, work-in-progress (WIP) costs, or the number of tardy jobs. Report 1 focused mainly on exploring and developing our heuristic-based web application which creates efficient operation sequences, allocating resources effectively, while adhering to operational constraints. Our group came up with the following mission statement:

"Develop an intuitive and efficient web application using heuristic methods to optimise large-scale assembly line scheduling for manufacturing companies, tailored to their specific objectives."

This solution will enable the creating of near-optimal sequencing of operations, addressing the complexities and scale of large-scale assembly line scheduling challenges.

Entities

- Internal Entities: Front-end and back-end subsystems, including Bill-of Materials (BOMs) and heuristic algorithms.
- External Entities: Manufacturing companies, stakeholders, and order databases.

Owners, Customers, and Users

- Owner: Potential project clients (professors and manufacturing companies), software architects and builders (development team)
- Customers: Companies seeking to optimise their operation scheduling
- Users: Production managers and unintended users that are affected by the system changes

Requirements

- Functional Requirements: Algorithms should generate production schedules that comply with the BOM precedence constraints, and utilise Gantt charts for visualisation
- Performance Requirements: Algorithms should have specific runtimes, able to handle large datasets, and produce near-optimal schedules

We then explored different heuristics and metaheuristic methods that we could possibly implement in our back-end, and possible front-end alternatives. The full list of alternatives are listed in Appendix Section A. Our final choices will be discussed in Section 2: Optimise the Design Choices.

2 Optimise the Design Choices

In Report 1, we discussed several heuristics and front end tools while exploring the design space. Please refer to Appendix Section A for the full list of methods and tools explored. In this section, we will be discussing the reasons for implementing each specific heuristic in Section 2.1 and web app choice in Section 2.2 amongst those explored. In optimising assembly line operations scheduling, selecting appropriate heuristics and web app tools would cater to the different challenges and requirements faced. The choice of each tool was driven by their individual strengths in handling different aspects of our project.

2.1 Optimised Heuristic Choice

In this section, we explain the heuristics chosen for our system. We also present our findings after testing the performance of each heuristic.

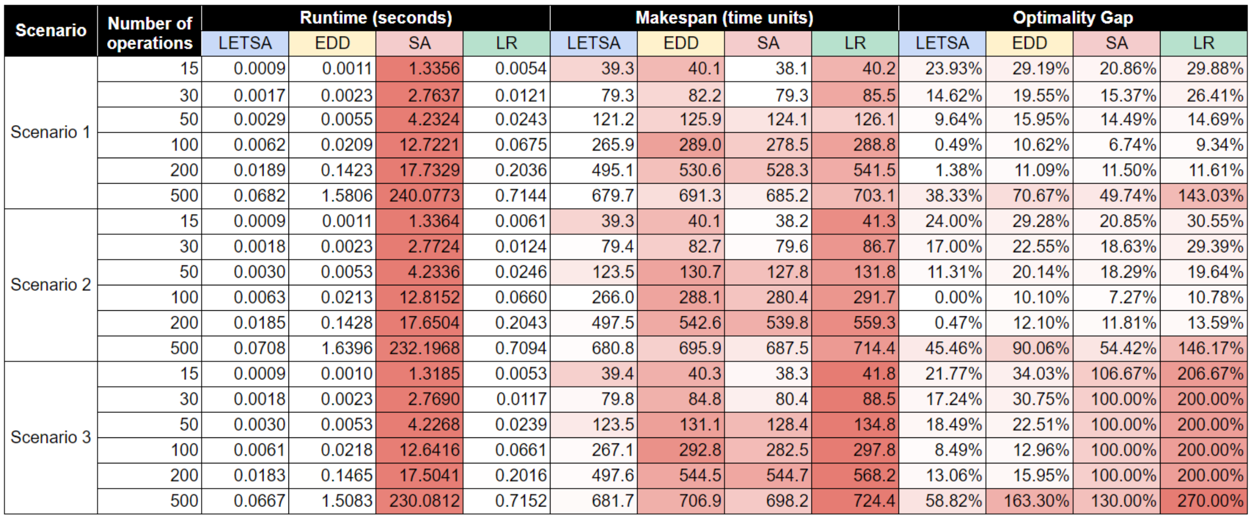
1. **Earliest Due Date (EDD)**
   The Earliest Due Date algorithm is chosen particularly for its effectiveness in environments where meeting due dates are crucial. It is straightforward and operates on a clear and intuitive principle of prioritising operations based on their due dates. EDD has been proven to minimise the number of tardy jobs, which allows for a consistent scheduling of operations to prevent significant delays in delivering tasks [Najat et al., 2019]. Its low complexity in sorting and scheduling operations based on the order is computationally inexpensive. Minimising the number of late operations was one of the objectives our problem aimed to solve, hence EDD which is widely used in many industries, was the most straightforward and intuitive choice.

2. **Lead Time Evaluation and Scheduling Algorithm (LETSA)**
   LETSA is chosen for its ability to optimise lead times, manage dependencies, and adaptability in making changes, improving assembly line scheduling and efficiency. Assembly lines often have complex dependencies between operations and LETSA's efficient use of resources allows for operation scheduling that avoids bottlenecks and idle times. LETSA was designed to be robust against changes and disruptions in operation times and machine availability, ensuring operation schedules remain feasible and efficient. By minimising the amount of work-in-progress inventory by scheduling operations just-in-time, LETSA reduces inventory holding costs and enhances the overall flow of materials through the production process. [Agrawal et al., 1996]

3. **Lagrangian Relaxation (LR)**
   Lagrangian Relaxation is chosen for this assembly line scheduling problem due to its strength in providing both lower and upper bounds, which are essential for evaluating and guiding other heuristics like simulated annealing. By relaxing constraints and incorporating them into the objective function through Lagrange multipliers, the method transforms a difficult constrained problem into a more manageable unconstrained one. This transformation allows for efficient exploration of the solution space, yielding a lower bound that indicates the best possible performance under relaxed conditions. Additionally, by subsequently fixing the solution to respect the original constraints, Lagrangian Relaxation helps generate an upper bound, offering a practical, feasible solution. This dual capability is particularly valuable in complex scheduling problems where precise constraint handling and optimal resource utilisation are critical. The method's robustness in dealing with multiple constraints and its ability to inform and enhance the performance of other heuristics make it an indispensable tool for optimising assembly line scheduling. [Xu and Nagi, 2013]

4. **Simulated Annealing (SA)**
   Simulated Annealing is a proven heuristic and is chosen for assembly line scheduling problems due to its effectiveness in exploring large and complex solution spaces to find near-optimal solutions. It excels at escaping local optima by accepting worse solutions with a probability that decreases over time, allowing it to navigate though local optimums. This ability is crucial for minimising makespan while adhering to various constraints such as precedence, machine availability, and work centre capacities. The algorithm's flexibility, robustness, and simplicity make it particularly suitable for handling the intricacies of assembly line scheduling, providing high-quality solutions within a reasonable computational time. [Kirkpatrick et al., 1983]

The four heuristics above have been widely discussed in the fields of operations research. However, it is equally important for our team to ensure that the final product implements these heuristics not only appropriately but efficiently too. As such, we performed a systematic performance testing for all the heuristics, aimed at measuring both the solution quality and the algorithm runtime. The results are summarised in Figure 1.



Scenario	Number of operations	Runtime (seconds)				Makespan (time units)				Optimality Gap			
		LETSA	EDD	SA	LR	LETSA	EDD	SA	LR	LETSA	EDD	SA	LR
Scenario 1	15	0.0009	0.0011	1.3356	0.0054	39.3	40.1	38.1	40.2	23.93%	29.19%	20.86%	29.88%
	30	0.0017	0.0023	2.7637	0.0121	79.3	82.2	79.3	85.5	14.62%	19.55%	15.37%	26.41%
	50	0.0029	0.0055	4.2324	0.0243	121.2	125.9	124.1	126.1	9.64%	15.95%	14.49%	14.69%
	100	0.0062	0.0209	12.7221	0.0675	265.9	289.0	278.5	288.8	0.49%	10.62%	6.74%	9.34%
	200	0.0189	0.1423	17.7329	0.2036	495.1	530.6	528.3	541.5	1.38%	11.09%	11.50%	11.61%
	500	0.0682	1.5806	240.0773	0.7144	679.7	691.3	685.2	703.1	38.33%	70.67%	49.74%	143.03%
Scenario 2	15	0.0009	0.0011	1.3364	0.0061	39.3	40.1	38.2	41.3	24.00%	29.28%	20.85%	30.55%
	30	0.0018	0.0023	2.7724	0.0124	79.4	82.7	79.6	86.7	17.00%	22.55%	18.63%	29.39%
	50	0.0030	0.0053	4.2336	0.0246	123.5	130.7	127.8	131.8	11.31%	20.14%	18.29%	19.64%
	100	0.0063	0.0213	12.8152	0.0660	266.0	288.1	280.4	291.7	0.00%	10.10%	7.27%	10.78%
	200	0.0185	0.1428	17.6504	0.2043	497.5	542.6	539.8	559.3	0.47%	12.10%	11.81%	13.59%
	500	0.0708	1.6396	232.1968	0.7094	680.8	695.9	687.5	714.4	45.46%	90.06%	54.42%	146.17%
Scenario 3	15	0.0009	0.0010	1.3185	0.0053	39.4	40.3	38.3	41.8	21.77%	34.03%	106.67%	206.67%
	30	0.0018	0.0023	2.7690	0.0117	79.8	84.8	80.4	88.5	17.24%	30.75%	100.00%	200.00%
	50	0.0030	0.0053	4.2268	0.0239	123.5	131.1	128.4	134.8	18.49%	22.51%	100.00%	200.00%
	100	0.0061	0.0218	12.6416	0.0661	267.1	292.8	282.5	297.8	8.49%	12.96%	100.00%	200.00%
	200	0.0183	0.1465	17.5041	0.2016	497.6	544.5	544.7	568.2	13.06%	15.95%	100.00%	200.00%
	500	0.0667	1.5083	230.0812	0.7152	681.7	706.9	698.2	724.4	58.82%	163.30%	130.00%	270.00%

Figure 1: Summary of Runtime and Makespan for each Heuristic

Our performance testing was done while ensuring that a wide range of scenarios are taken account for. As such, the test cases used follows the principles and methodologies introduced in our project proposal. In total, there were 4 parameters used to generate different BOM test cases, namely:

- n: Number of operations in the BOM. This parameter affects the size of the BOM and, with it, the complexity of the problem. The values of n selected were 15 and 30 for small scale operations; 50 and 100 for medium scale operations; 200 for large scale operations, and 500 for colossal scale operations (mainly used for stress testing).
- D: BOM network density (between 0 and 1). This parameter affects the shape of the BOM. A high D creates a 'long' BOM shape with few end-product operations whereas a low D creates a 'wide' BOM shape with more end-product operations. The values selected were 0.1, 0.5, and 0.9.
- p: Logarithmic distribution parameter (between 0 and 1). The processing time of each operation follows an i.i.d. logarithmic distribution with parameter p, where increasing this parameter increases the variance of the distribution. The p values selected are 0.15 and 0.95.
- M: The number of distinct machines used in total by all operations. M follows a discrete uniform distribution between 1 and 15.
- `bottleneck`: A Boolean parameter that indicates whether a BOM includes a bottleneck machine. When set to true, a machine is randomly designated as a bottleneck within the production line.

Additionally, we also needed to generate information representing the factory's production capacity. This includes generating information on how many workcenters and machines exist in the factory. For this purpose, we create three scenarios to represent the possible scenarios that commonly emerge in industrial situations.

- **Scenario 1**: In this scenario, we assume that each workcenter in the factory has a very abundant number of machines available for use. This means that the workcenters have a very high capacity for processing the operations at parallel.
- **Scenario 2**: Here, we assume that each workcenter in the factory has a mild number of machines available. In other words, some workcenters may have more machines as compared to others, making it more challenging to schedule operations smoothly and achieve a low makespan.
- **Scenario 3**: In this scenario, each workcenter only has one machine for each functionally identical machine listed in the BOM. In other words, it becomes extremely challenging to schedule operations and achieve the optimal makespan.

By varying these parameters, we created 1944 test cases to systematically evaluate our heuristics. Our testing reveals that all heuristics satisfy the functional requirements stipulated in Report 1. Indeed, for a BOM with 200 or less operations, the heuristics successfully terminate within 30 seconds. This is so much faster than the initial target of 30 minutes maximum runtime. Furthermore, it can be seen that the heuristics mostly yield a reasonable optimality gap for all the test cases. Here, the optimality gap is calculated by:

$$\text{Optimality Gap} = \frac{f_H - f_L}{f_H}$$

where f_H is the makespan found by heuristic H and f_L is the lower bound found via the LR heuristic.

Figure 1 also shows that LETSA is a superior choice compared to the remaining heuristics, both in minimising makespan and algorithm runtime. SA ranks second in solution quality requires a relatively longer (but still reasonable) computational time. Meanwhile, EDD and LR are very quick to execute, though they tend to produce solutions with larger optimality gaps. Nevertheless, our product integrates these four heuristics to provide comparisons and alternatives, ensuring that production managers can have confidence in any schedule generated by the product.

2.2 Optimised Web App Choice

Why ReactJS over the rest:

Given our limited time, we chose to develop our web application using React, leveraging HTML, CSS, and JavaScript. ReactJS, a popular JavaScript library for dynamic and interactive user interfaces, proved to be the best fit for our project due to its community support, and rapid rendering via the virtual Document Object Model (DOM) which improved user experience. This also makes the application lightweight, boosting performance and productivity.

Moreover, there are already pre-built components made by other frameworks that builds on ReactJS, which makes it super easy for us to utilise and incorporate them into our designs (e.g. AntDesign, MUI). It is also easier for team members to incorporate their own parts since ReactJS allows for components to be made separately.

Hence, even though our team is familiar with R, a fundamental part of Rshiny, the online resources for implementing ReactJS are more robust, better supporting our team's needs. Additionally, while Docker provides excellent application isolation and consistent performance, our lack of expertise led us to opt out of using it.

3 Develop the Architecture

3.1 Subsystem within Project

In this section, we explain some subsystem components our product is constructed from. We also explain how these subsystems interact with each other.

3.1.1 Front-End Subsystem

1. **User Interface (UI)**
   UI refers to the screens, buttons, toggles, icons, and other visual elements that a user interacts with when using a website, app, or other electronic devices. UX refers to the entire interaction a user has with a product, including how they feel about the interaction. This subsystem ensures that the web application is aesthetically pleasing and functionally accessible. Key features include:

   - **Responsive Design:**
     The UI dynamically adapts to different screen sizes. For instance, below 800px width, the layout transitions to a vertical flow, enhancing usability on mobile devices.
   - **Color Scheme:**
     A consistent and visually appealing color palette is applied to enhance the overall look and feel of the application.

2. **User-experience (UX)**
   The User Experience subsystem focuses on the overall interaction users have with the product, emphasizing ease of use and satisfaction. Key elements include:

   - **Integrated Functionality:**
     Essential functions, such as instructions, file uploads, and Gantt chart displays, are consolidated on a single "manage" page. This centralization ensures users can access necessary features without navigating through multiple pages.
   - **Visual Guidance:**
     A stepper which shows the steps to obtain a operation schedule remains at the top of the "manage" page even when scrolling, and shows which step the user is on. This ensures users know how to interact with the various sections on the page.

3.1.2 Database Subsystem

1. **Application Programming Interface (API)**
   The API component facilitates communication between the front-end and back-end of the web application. It ensures data is correctly fetched and posted, maintaining the application's functionality.

- **Data Retrieval and Storage:**
  The API handles requests for retrieving and storing data in the database, enabling real-time updates and synchronization between user actions and the database.

2. **File Management**
   This component manages the upload and storage of CSV files, essential for tasks like generating Gantt charts.

   - **File Upload:**
     Users can upload CSV files through the "upload file" system, which are then stored in the "static/files" directory within the server folder.
   - **Gantt Chart Integration:**
     Scheduled CSV files are processed to generate and display Gantt charts, providing visual project management tools within the application.

3.1.3 Heuristic Subsystem

The heuristic subsystem's role in our product is analogous to a brain. The heuristics solves the scheduling problem to cater for certain objectives selected by the user.

1. **Lead Time Evaluation and Scheduling Algorithm (LETSA)**
   - **What it does:** LETSA focuses on minimising the makespan in a project.
   - **How it helps:** Consider having a list of tasks that need to be completed as efficiently as possible. The LETSA algorithm evaluates each task, determines the required completion time, and organizes them to optimize overall efficiency. It ensures that the most critical tasks are prioritized and completed promptly, thereby aiding in the effective adherence to deadlines.
2. **Earliest Due Date (EDD)**
   - **What it does:** EDD prioritises tasks based on their due dates to minimise the number of tardy jobs.
   - **How it helps:** The algorithm prioritizes tasks with the nearest deadlines, ensuring they are completed first. This approach helps avoid missing crucial due dates and keeps the project on schedule.
3. **Simulated Annealing (SA)**
   - **What it does:** SA is inspired by a process used in metalworking to find good solutions to complex problems.
   - **How it helps:** The goal is to identify the optimal way to schedule tasks, despite the numerous possible arrangements. SA explores different task sequences, occasionally accepting less optimal ones to investigate potentially better solutions. Over time, SA increasingly focuses on the most promising arrangements, helping to develop a high-quality schedule that balances various factors effectively.
4. **Lagrangian Relaxation (LR)**
   - **What it does:** LR simplifies complex scheduling problems by relaxing certain rules temporarily to make the problem easier to solve.

- **How it helps:** The LR heuristic temporarily relaxes some constraints to solve a more relaxed, unconstrained optimisation problem. Then, any solution found is adjusted to approach the optimal solution. Further, this method enables us to know the lower bound of the makespan, effectively informing us of the optimality gap of solutions found by any of the heuristics.

3.1.4 Deployment and Infrastructure Subsystem

The deployment and infrastructure subsystem ensures the web application is hosted and accessible to users.

1. **Hosting Environment:**
   The application is hosted on a virtual environment, with the front-end served from localhost:3000 for local server and GitHub Pages for public hosting, and the database interactions managed through localhost:8000.

   - **localhost:3000:**
     Typically used for the front-end development server. This is where the user interface (UI) and user experience (UX) components of the web application are served. Technologies like React, Angular, or Vue.js commonly use this port by default for development purposes.
   - **localhost:8000:**
     Generally used for the back-end server. This port is where the server-side application runs, handling API requests, database interactions, and other server-side logic. It is often associated with back-end frameworks such as Express.js for Node.js applications or Django for Python applications.
   - **GitHub Pages:**
     GitHub Pages is a popular tool for hosting local sites publicly and reliably, directly from a repository on GitHub.com. One of the main reasons for its widespread use is the free hosting service it offers. This makes it an excellent choice for developer like our group looking to share their projects and websites without incurring hosting costs.

2. **Scalability and Maintenance:**
   The virtual environment allows for easy scaling and maintenance, ensuring the application can handle increasing user loads and updates without significant downtime.

To identify the interfaces and interactions between subsystems from internal entities, we utilize design structure matrix as per taught in class [Jackson, 2009].

(Event) Relates Subsystem to	UI System	UX System	API System	Deployment and Infrastructure System	Integrated Functionality System	File Management System	Database System	LETSA Algorithm	EDD Algorithm	SA Algorithm	LR Algorithm
UI System	▨	works with	needs	hosted by	requires	interacts with		interacts with	interacts with	interacts with	interacts with
UX System		▨	needs		depends on			interacts with	interacts with	interacts with	interacts with
API System			▨	hosted by	interacts with	interacts with	accesses	interacts with	interacts with	interacts with	interacts with
Deployment and Infrastructure System	hosts		hosts	▨				interacts with	interacts with	interacts with	interacts with
Integrated Functionality System	requires	depends on	interacts with		▨	relies on		interacts with	interacts with	interacts with	interacts with
File Management System			uses			▨	stores metadata	interacts with	interacts with	interacts with	interacts with
Database System						interacts with	▨	interacts with	interacts with	interacts with	interacts with
LETSA Algorithm								▨			
EDD Algorithm									▨		
SA Algorithm										▨	
LR Algorithm											▨

Table 1: Design Structure Matrix

3.2 Subsystem between Projects

Our project, which focuses on operation scheduling using heuristic approach and web application development, is part of a larger system comprised of several interrelated projects. Notably, our project has the capability to scale with project 5—Operation Scheduling (Optimal) group, and project 3 and 4—Materials Requirements Planning (MRP) groups. Below, we identify subsystems between these projects and explore their interactions and dependencies with our own subsystem

3.2.1 Database Subsystem (Project 3)

- **Data Provider from Forecasted Demand:**
  Provides a centralised platform that connects the data from all groups to form a streamline and efficient backend system as the backbone of the full scale project by ensuring information consistency and version management control. They play a role in allowing us to retrieve the latest BOM dataset for further analysis of makespan based on the objective chosen and allow us to form a gantt chart.

3.2.2 Material Requirement Planning (Project 4)

- **Explosion Calculus:**
  Provides a visualisation of the BOM (Bill of Materials) structure and accounts for lead times for each part. The Material Requirements Planning (MRP) process helps in creating a Master Production Schedule (MPS) when combined with the scheduled operations derived from heuristic scheduling methods.

3.2.3 Exact Methods Scheduling Algorithm Subsystem (Project 5)

- **Integration with Optimal Operation Scheduling:**
  Comparison could be made to evaluate the performance of heuristic methods versus exact methods. However this is only applicable to smaller dimension datasets (both size and parameter complexity) to ensure viable computational time for the optimal algorithms.

4 Validate the Design

4.1 Product Testing

To validate our design, we develop multiple test plans and manage design risks. Our test plan consists of two components: behavioral tests and non-behavioral tests.

Non-behavioural methods are descriptive models in which no hypothesis on user behavior is proposed, whereas behavioral methods try to reproduce the workings of the user's mind when making their choices. In addition to that, both behavioral and non-behavioral method require test plans and entry and exit conditions. Test plan includes a list of test procedures, a summary of the test facilities, and the entry and exit conditions for each test, whereas entry condition describes a requirement that must be satisfied before conducting the test and exit condition describes a minimum requirement for passing the test. Several subsystems in the product are needed for testing before publishing them:

Test number	Test method	Test facilities	Entry condition	Exit condition
TP.1	User uploads CSV files	User (in testing phase, it will be our team)	There are upload files area in the web app for the user to upload	Able to show the dataset after uploading the CSV files
TP.2	User chooses the objectives	User (in testing phase, it will be our team)	There are buttons for the user to choose the objectives	Web App should show the calculated results of the objectives the user chose

Table 2: Behavioral Test Methodology for Heuristic and Web App

Test number	Test method	Test facilities	Entry condition	Exit condition
TP.1	Whitebox Testing	Computer, Python	There is a Python code that implements the heuristics.	The program follows the algorithm, solves basic test cases, and respects precedence and operational constraints.
TP.2	Systematic Testing	Computer, Python	Whitebox testing has been completed and systematic test cases have been generated.	The program solves basic to edge test cases without violating any constraints or requirements.
TP.3	Runtime Testing	Computer, Python	Whitebox testing has been completed and systematic test cases have been generated.	The program fulfills runtime requirements as stipulated in Report 1.
TP.4	Gantt Chart Generation	Computer, dummy datasets	The program is able to translate and parse CSV files into the gantt chart specification.	The program generates the gantt chart in the web app.
TP.5	Codes Linkage	Computer, front-end and back-end codes	The heuristic implementation in Python is properly and accurately functional.	The web app is able to integrate the Python code into the React code.
TP.6	Gantt Chart Generation	Computer, scheduled datasets	Gantt chart should be able to visualize the dummy datasets.	Web app is able to use the dataset generated from the algorithm code to generate the gantt chart accurately

Table 3: Non-Behavioural Test Methodology for Heuristic and Web App

4.2 Failure Modes and Effects Analysis (FMEA)

To identify the project's uncertainties and potential risks, we implement design risk management, which is the activity of proactively identifying risks and attempting to prevent or mitigate the most serious risks. Our risk mitigation strategy is based on failure modes and effects analysis (FMEA). This is a formal technique for identifying potential design failures and implementing corrective actions while the product is still being developed [Aeronautics and (NASA), 2000].

Component	Failure Mode	Effect of Failure	Cause of Failure	Severity (S)	Occurrence (O)	Current Controls	RPN (S*O)
User uploads CSV files	File upload fails	User cannot use the app	Server or network issues	3	2	Upload retries	6
CSV files fetched to algorithm	Files not fetched correctly	Algorithm doesn't run	API or file path issues	4	1	Error logging, status messages	4
Algorithm calculates solutions	Incorrect calculations	Incorrect scheduling solutions	Bugs in algorithm, incorrect data inputs	4	2	Code reviews, unit testing	8
Algorithm generates dataset	Dataset generation fails	No output for Gantt chart	Algorithm logic issues, data format errors	4	2	Error logging, status messages	8
Web app generates Gantt chart	Gantt chart not generated or incorrect	User cannot see schedule	Incorrect data handling, UI issues	2	2	UI testing, error messages	4

Table 4: FMEA Table

The severity scale for our risk assessment is:

- Severity 5: Catastrophic
- Severity 4; Major
- Severity 3: Moderate
- Severity 2: Minor
- Severity 1: Negligible

The Occurrence scale for our risk assessment is:

- Occurrence 5: Almost Certain
- Occurrence 4; Frequent
- Occurrence 3: Occasional
- Occurrence 2: Remote
- Occurrence 1: Rare

For each component, after calculating the Risk Priority Number (RPN), we will prioritize actions to reduce the risk to minimise waste of time and resources:

1. File Upload Fails (RPN 6):
   (a) improve server reliability by using a more robust hosting service.
   (b) Provide clearer error messages and retry options for users.
2. File Not Fetched Correctly (RPN 4):
   (a) Implement better error handling and ensure file paths are validated.
   (b) log detailed error messages for debugging.
3. Incorrect Calculations (RPN 8):
   (a) Increase code review frequency and coverage of unit tests.
   (b) Add validation checks for input data to prevent incorrect calculations by providing the lower and upper bound of the feasible solution.
4. Dataset Generation Fails (RPN 8):
   (a) Enhance the robustness of the algorithm to handle edge cases.
   (b) Validate intermediate data formates to ensure correctness.
5. Gantt Chart Not Generated (RPN 4):
   (a) Perform comprehensive UI testing.
   (b) Ensure input data for Gantt chart is thoroughly validated

5 Execute the Design

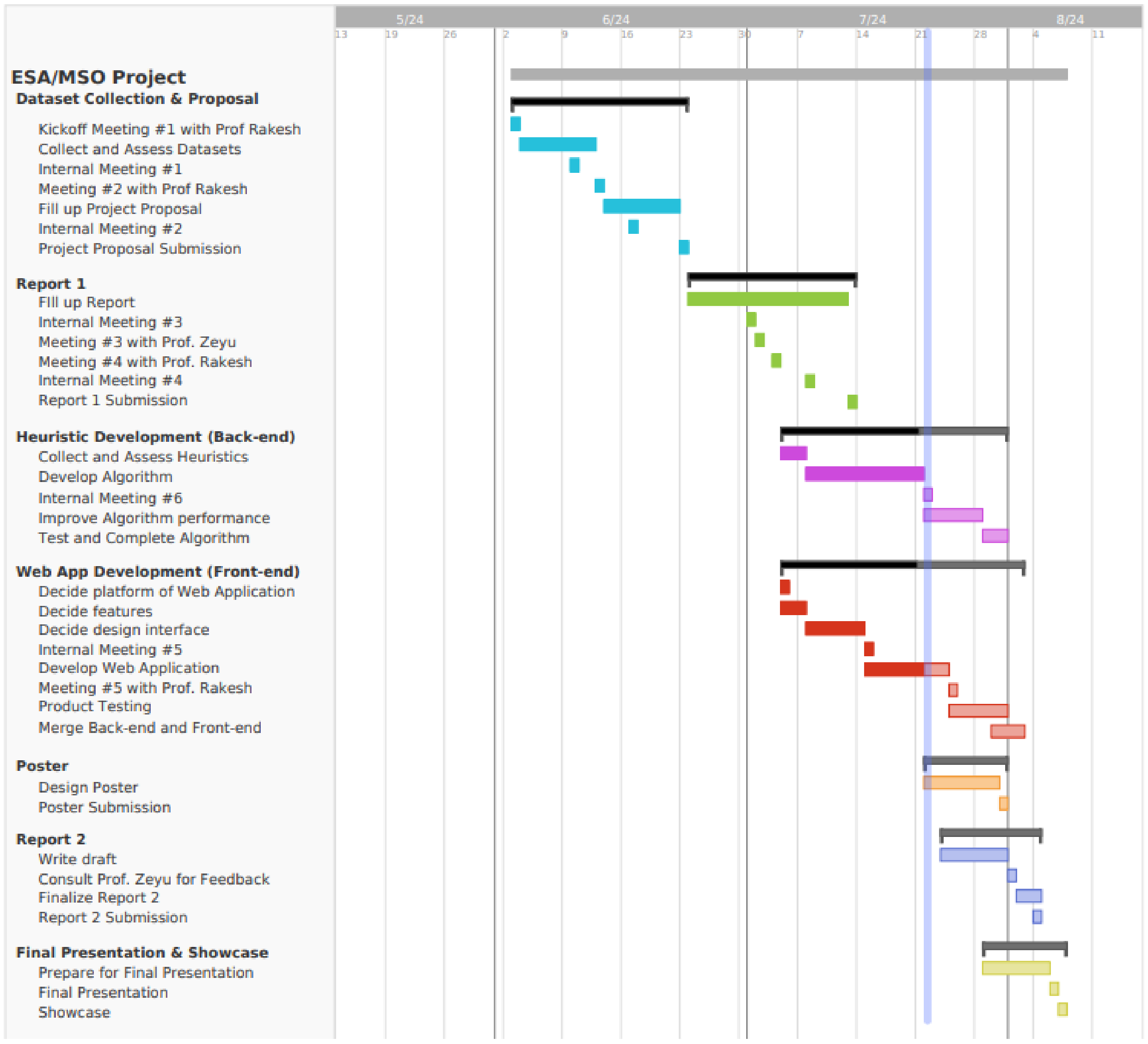


Figure 2: Project Gantt Chart (at Week 11)

This Gantt chart, which is updated to Week 11, serves to keep us on track on our progress for the project. The chart starts at Week 1 and ends by Week 14. Week 1 to Week 3 is kept blank as there is the project has not been explored in that time yet. The project is scheduled to start on June 3rd and end by Aug 11th. We categorised the project by certain key milestones such as the development of the heuristics and web application, as well as the due dates of the deliverables, namely Report 1 and Report 2.

Under each category, we further split the milestones into subtasks and assigned them manageable deadlines so as to track our progress for each milestone. Each subtask is scheduled to start after the preceding sub task so as to respect the precedence relationship between subtasks. Developing the heuristics and the web application can be done concurrently as the two tasks do not have any precedence relationships. As such, we scheduled them to start at the same time and to be worked on in parallel.

The Gantt chart also includes progress bars which serves as indicators on the progress of each subtask. For example, the subtask "Develop Web Application" is only 60% complete. All in all, the Gantt chart effectively indicates to the team the progress of the project and whether we are lagging behind on certain tasks to be completed.

6 Demonstrate Your Understanding of MSO Materials

We applied and expanded upon concepts from 40.012: Manufacturing and Service Operations to tackle the NP-hard problem of assembly line scheduling. Due to the inherent complexity and computational difficulty of NP-hard problems, finding an exact optimal solution within a reasonable time frame is often impractical, especially for large-scale problems. Exact methods, while theoretically capable of providing optimal solutions, are computationally expensive and can become infeasible as the size and dimension of the problem grows, resulting in excessive computation times and resource requirements.

Given these challenges, we opted to use heuristics, which provide approximate solutions more efficiently. Heuristic methods, such as the Lead Time Evaluation and Scheduling Algorithm (LETSA), Lagrangian Relaxation (LR), Simulated Annealing (SA), and Earliest Due Date (EDD), offer a practical balance between solution quality and computational feasibility. They allow us to explore a broad solution space and arrive at near-optimal solutions within a manageable timeframe. These heuristics were chosen to address different objectives within the scheduling problem, providing us a different perspective in handling the various constraints and requirements inherent in assembly line scheduling. This approach enabled us to effectively manage the complexity of the problem and deliver actionable schedules that meet the needs of real-world manufacturing operations.

LETSA was employed with the dual objective of minimising both makespan and work-in-progress costs, providing an efficient schedule that balances throughput and inventory holding costs. EDD focused on minimising the number of tardy jobs, prioritising tasks with the earliest due dates to reduce penalties associated with late deliveries. SA aimed to minimise the makespan, with potential to outperforming LETSA in optimising total operation time. LR served a critical role in validating our results by establishing lower and upper bounds, thus providing a benchmark against which the other heuristics could be measured.

The parameters involved in our study include the precedence constraints between operations, processing times, work centres, machine types, and due dates, all of which are essential in defining the constraints and objectives for the scheduling algorithms. Our analysis encompassed 1944 test cases, assessing the performance of each heuristic through calculated makespan, total number of tardy jobs, and the runtime of each algorithm.

While MSO primarily focused on job shop scheduling problem (a subset of assembly line scheduling), our model's broader applicability allows it to also handle job shop scheduling problems given appropriate data formatting. The course content covered heuristic lot sizing schemes (Silver-Meal heuristic, least unit cost (LUC) heuristic, part period balancing), as well as other simple sequencing rules (first come first serve, shortest processing time, earliest due date, critical ratio). However, our project necessitated more advanced methodologies, as most of the taught heuristics were insufficient for the complexity of our scheduling problem, which included sequencing across multiple machines and work centres.

The final output of our project includes the calculated result and a detailed Gantt chart for each heuristic, illustrating the makespan derived from the algorithm (dependant on objective chosen) and the start and end schedule for each operation across different work centre machines. This visual representation showcases the optimal or near-optimal schedules derived from each heuristic.

References

[Aeronautics and (NASA), 2000] Aeronautics, N. and (NASA), S. A. (2000). Failure modes and effects analysis (fmea): A bibliography. Nasa/sp–2000-6110, NASA Scientific and Technical Information (STI) Program Office. Retrieved from NASA Technical Reports Server (NTRS).

[Agrawal et al., 1996] Agrawal, A., Minis, I., and Nagi, R. (1996). 'just-in-time' production of large assemblies. *IIE Transactions*, 28(8):653–667. `https://doi.org/10.1080/15458830.1996.11770710`.

[Jackson, 2009] Jackson, P. (2009). *Getting Design Right: A Systems Approach.* CRC Press.

[Kirkpatrick et al., 1983] Kirkpatrick, S., Gelatt, C. D., and Vecchi, M. P. (1983). Optimization by simulated annealing. *Science*, 220(4598):671–680. `https://doi.org/10.1126/science.220.4598.671`.

[Najat et al., 2019] Najat, A., Yuan, C., Gursel, S., and Tao, Y. (2019). Minimizing the number of tardy jobs on identical parallel machines subject to periodic maintenance. *Procedia Manufacturing*, 38:1409–1416. 29th International Conference on Flexible Automation and Intelligent Manufacturing (FAIM 2019), June 24-28, 2019, Limerick, Ireland, Beyond Industry 4.0: Industrial Advances, Engineering Education and Intelligent Manufacturing.

[Xu and Nagi, 2013] Xu, J. and Nagi, R. (2013). Solving assembly scheduling problems with tree-structure precedence constraints: A lagrangian relaxation approach. *IEEE Transactions on Automation Science and Engineering*, 10(3):757–771. `http://dx.doi.org/10.1109/TASE.2013.2259816`.

A Explored Design Space

A.1 Heuristic Choices

- **Lead Time Evaluation and Scheduling Algorithm (LETSA)**
  A heuristic that generates a close-to-optimal schedule by prioritizing the execution of jobs which belong to a dynamically computed critical path. The proposed method, evaluated based on cumulative lead-time and work-in-process (WIP) inventory costs, outperforms various scheduling heuristics developed by past researchers, be it simple sequencing rules such as SPT (Shortest Processing Time), EDD (Earliest Due Date), FCFS (First-Come-First-Served) or more sophisticated algorithms such as SIMFACTORY, a commercially available shop-floor simulation tool. LETSA largely outperforms these heuristics in terms of producing a near-optimal solution, as well as minimizing WIP holding costs via a just-in-time production approach.

- **Lagrangian Relaxation (LR)**
  An approach utilising lagrangian relaxation to solve the MILP formulation for the assembly scheduling problem. What sets apart the lagrangian relaxation, apart from being able to produce fast and good solutions, is the method's ability to produce the solution's gap towards optimality. This makes it easier to determine the algorithm's stopping criteria for problems where the optimal solution is strenuous to find.

- **Shifting Bottleneck Procedure (SBP)**
  A scheduling algorithm which specialises in solving assembly scheduling problems where bottleneck machines complicate the search for an optimal solution. This heuristic outperforms other methods in terms of solving bottleneck situations, but perhaps not as well when it comes to solving other objectives such as minimising makespan.

- **Memetic Algorithms (MA)**
  Memetic algorithms are one of the most effective and flexible metaheuristic approaches for tackling hard optimisation problems. A heuristic based on memetic algorithms to solve the assembly scheduling problem. MA searches the global and local solution space in order to reduce the chance of a local convergence. This enables the algorithm to efficiently explore and exploit the search space, providing high-quality solutions for complex optimisation problems.

- **Genetic and Evolutionary Algorithms (GA & EA)**
  A genetic and evolutionary algorithms used to solve the job shop scheduling problem. The algorithms draw inspiration from biological processes, namely genetic mutations and evolution, to find a near-optimal solution. These algorithms simulate the process of natural selection whereby only the "fittest" survives: that is, weak solutions are discarded as the algorithm runs. This approach enables the genetic algorithm to generate an optimal solution effectively and efficiently.

- **Variable Neighbourhood Search Algorithm (VNS)**
  Search-based approach centred on exploring more than one type of neighbourhood structure. VNS iterates in an analogous way over some neighbourhood structure until some stopping criterion is met. Either a local minimum of one neighbourhood structure or the global optimum of all neighbourhood structures are observed. VNS starts from an initial solution produced by Shortest Processing Times rule for job shops, and utilises 3 types of Neighbourhood Search Structure (NSS) that are based on insertion neighbourhood. A significant advantage of VNS

compared to other local search-based metaheuristic algorithms is that VNS is equiped with several neighbourhood search structures which diversifies the search space. This prevents the metaheuristic from getting stuck in the local optima and hence a more accurate solution.

- **Multi-type Particle Swarm Optimisation (MPSO)**
  MPSO enhances traditional Particle Swarm Optimisation algorithm and incorportates multi-type individual enhancement schemes . This prevents the solution from converging to the local optima prematurely. MPSO adopts the real space as the search space called random-key (RK) and does not require a heuristic algorithm to initialise a population to speed the convergence rate. Implementing a heuristic algorithm increases the computational load. MPSO is able to reach the optimal area in the search space with a smaller population size and iterations.

- **Simulated Annealing (SA)**
  Simulated annealing is inspired by the annealing process in metallurgy. A control parameter, temperature, is used to determine the probability of accepting non-improving solutions. The temperature is gradually decreased according to a cooling schedule such that few non-improving solutions are accepted at the end of the search such that it converges to an optimal or near-optimal solution. Simulated annealing has the flexibility to handle a wide range of optimisation problems and is generally more capable of escaping the local optima, giving it a better chance of finding the global optimum. However, the cooling schedule needs to be tuned, if it is too fast, the algorithm may converge to a local optimum, and if it is too slow, the algorithm may take too long to converge. It may also require a large number of iterations to find a good solution with increased complexity.

- **Tabu Search (TS)**
  Tabu search enhances the performance of local search by relaxing constraints to explore a broader solution space. Unlike hill climbing algorithms that only accept improving solutions, tabu search allows for non-improving solutions to escape from local optima, allowing the algorithm navigate out of local optima when all neighbouring solutions are non-improving. A tabu table is introduced to prohibit the revisitation of recently explored solutions, thus leading the search away from previously visited solutions. This approach can lead to cycles where the solution repeats itself within a local optimum. To mitigate this, a larger tabu list can be used (determined by varying the table size or allocated a dynamic tabu list). This maintains a history of a broader set of visited solutions, preventing the formation of cycles. By doing so, the tabu search algorithm can explore new areas of the solution space more effectively, thereby enhancing the overall performance in finding a high-quality solution.

A.2 Web App Choices

- **ReactJS**
  React, widely used by developers globally, excels in creating interactive user interfaces. It's a dominant library among web developers, with over 40% reporting extensive usage based on 2023 Stack Overflow surveys. React's declarative views enhance code predictability and simplify debugging. Additionally, its encapsulated components, managing their own state, promote better code comprehension for team members.

- **RShiny**
  RShiny, built upon the widely acclaimed R language for statistical modeling and data analysis, provides a viable option. It enables the development of interactive web applications that execute R code on the back end.

- **Docker**
  Docker offers a platform that streamlines the deployment and management of applications. Its ability to isolate applications and their dependencies within containers ensures consistent performance across diverse environments. This feature makes Docker a compelling candidate for enhancing our front-end development processes.