40.014: Engineering Systems and Architecture

Operations Scheduling

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 Introduction

Assembly operations are the workhorses of many manufacturing systems. Imagine circuit boards populated with electronic components and vehicles taking shape on an automobile assembly line. These processes all share a common thread: they involve combining multiple inputs to create a final product, which defines an assembly operation. However, unlike simple manufacturing processes, assembly operations require all necessary components to be present before work can begin. This matching aspect creates a huge complication in the production flow. If even one component is missing, the entire assembly line will be affected. This disruption can ripple outwards to impact not only the assembly itself but also the upstream fabrication lines. Recognizing the critical influence of assemblies on overall system performance, manufacturers often prioritise scheduling and control around assembly operations by scheduling a plan to coordinate all fabrication lines such that each line is supplied with the necessary components. By ensuring timely and complete component availability, manufacturers can keep assembly lines operational and optimise their entire production process.

This project tackles the challenge of efficient assembly line scheduling in manufacturing, focusing on the incorporation of the Bill-of-Materials (BOM) to ensure accurate resource allocation. Assembly lines produce diverse products in batches, leading to complex scheduling problems due to their dynamic operations. We aim to explore heuristic methods to optimise job shop scheduling as part of the Manufacturing and Service Operations course as well as integrate concepts from the Engineering Systems Architecture course, emphasising on stakeholder analysis, exploration of design space, optimising decisions and iterating through our heuristic methods to find the best performing algorithm.

2 Problem

Assembly operations are integral to most manufacturing systems, involving the combination of multiple inputs to produce a final output. Examples include inserting electronic components into circuit boards, assembling body parts and engines into automobiles, and combining chemicals to produce other substances. These operations are characterised by their need for all necessary components to be present before processing can begin, which can complicate production flows.

The Bill-of-Materials (BOM) plays a crucial role in assembly operations, detailing all the components required at each stage of the assembly process. A shortage of any component can disrupt the entire assembly line, affecting not only the assembly operations but all other related fabrication lines. To mitigate these disruptions, it is common to subordinate the scheduling and control of the fabrication lines to the assembly operations. This is achieved by specifying a final assembly schedule and working backward to schedule the fabrication lines.

The primary goal of this project is to create a schedule that minimises the makespan (total time to complete all tasks) while considering constraints such as workstation availability, task order, and potential downtimes. The complexity of assembly line scheduling arises from the vast number of possible schedules, which grows exponentially with the number of tasks and workstations. Traditional optimization methods, while aiming for the absolute best solution, become computationally impractical for large-scale problems with numerous tasks and workstations ([Vakhania and Mamporia, 2020]). Hence, heuristic methods are preferred in this scenario.

3 Data

3.1 Analysis of Data Required

The project focuses on the development of Bills-of-Materials (BOMs), which require detailed data on parts, operations, required components, processing times, and workcentres [A. Agrawal, 1996]. Acquiring a comprehensive dataset that encompasses all these elements with real-life data presents a significant challenge. Consequently, for datasets with missing information, we will supplement these datasets by generating the missing information. Should these datasets be insufficient for our needs, we will be generating entirely new datasets using several methods as outlined in 3.3.

3.2 Industry Dataset Collection

We have identified two industry related datasets that almost align with the challenges presented by our project. Both datasets serve as benchmarks derived from simulating real-life data, which, due to confidentiality constraints, cannot be directly accessed. The first dataset pertains to a complex job shop scheduling scenario that incorporates a multi-objective approach [Deliktaş et al., 2024]. Conversely, the second dataset we evaluated originates from a remanufacturing scheduling process and also serves as a benchmark for real-life data [Jun et al., 2021].

3.2.1 Benchmark Dataset

This dataset pertains to a Flexible Job Shop scheduling problem with Sequence-Dependent Family Set-up Times and Intercellular Transportation Times (FJCS-SDFSTs-ITTs) [Deliktaş et al., 2024]. It involves:

- **Flexible allocation of jobs to multiple machines:** Jobs can be assigned to various machines based on availability and suitability.
- **Sequence-dependent setup times:** Setup times vary depending on the sequence in which job families are processed.
- **Intercellular transportation times:** Consideration of the time required to transport jobs between different cells within the manufacturing system.

Data Cleaning
We automated the cleaning and reformatting of datasets that were originally in CSV format but received as text files. Our Python script iterates through datasets categorized by size in different folders, selectively processes files with a '.csv' extension, and saves them in the desired CSV format after cleaning. This process ensures data integrity and usability for further analysis.

Usable Data for the project

1. **Large Dataset Availability:** We have access to 43 datasets including multiple large instances (Instance #22-Instance #43). The last dataset, Instance #43 represents real-world data, while the other large datasets Instance #22 - Instance #42 are benchmark datasets, which are artificially yet strategically crafted problems such that the robustness of the algorithm can be evaluated. This real-world dataset will serve as an effective test set, while the remaining large datasets will be utilized to train our algorithm model.

2. **Comprehensive Benchmark Dataset:** The benchmark dataset provides comprehensive details on specific part families, including the required jobs and operations for their production, the necessary machines along with alternative options, and the processing times for each operation, all accompanied by their respective job due dates. Additionally, to further enhance our analysis, the dataset also includes extra data such as processing times, routing information, and due dates for each part family as processed by the required machines.

Data Limitations

1. **Complexity Due to Cellular Organization:** Incorporating the cellular organization of machines into the algorithm significantly increases its complexity, making it impractical to develop within a short timeframe. This approach also results in prolonged processing times due to increased algorithmic complexity. To simplify the problem, cellularization is disregarded, and the cells are treated as individual work centers instead.

2. **Relevance to Assembly Manufacturing:** Although this dataset is not as directly applicable to assembly manufacturing, it serves as a practical substitute given the scarce data available on assembly lines. One notable limitation is the absence of bills of materials, which are crucial for addressing our specific assembly line manufacturing issues.

3.2.2 Remanufacturing Dataset

This is the experimental data pertaining to a remanufacturing system, where end-of-life (EOL) products are disassembled to get the damaged components which are then fixed and reassembled to get like-new products. The remanufacturing process involves several identical workcentres to (dis)assemble products simultaneously and different reprocessing routes depending on the type and degree of damage of a component, consisting of a predefined set of operations at their respective workcentres [Jun et al., 2021]. The dataset involves:

- **Subsystem Interconnectivity:** The subsystems (disassembly, reprocessing and reassembly) process the products in batches, and a batch of the same products must finish being processed in the current subsystem before it can move on to the next.

- **Product-component-condition specific processes:** Reprocessing routes are not fixed but is dependent on the condition of the components. The condition of the component also determines the set of operations it has to undergo at various workcentres.

- **Processing Time Variations:** Real-world complexity is reflected through the variation of processing times based on the damage type. Optimisation of systems found in this dataset would be essential for realistic simulations.

Usable Data for the project
The intended use of this dataset matches our objective of the project, and contains the relevant information required for our analysis and model testing. The dataset obtained is large with 17 remanufacturing systems, each with multiple products and multiple components for each product. The detailed information on routes and timings are essential for modelling the processes involved in remanufacturing, supporting an accurate analysis of each system.

Data Limitations

1. **Specificity of Context:** One undesirable feature about this dataset is that it is extremely specific to the remanufacturing context. The type of data obtained is only a subset of the problem we are trying to solve, hence additional datasets would be used to test the extensiveness of our model.

3.3 Generating New Data

Due to the limitations of the datasets collected earlier, we need to generate new data on our own. Taking inspiration from [Hall and Posner, 2001], We developed a systematic approach to generate low-bias, representative datasets involving 5 distinct parameters (to be elaborated below). By changing the values of these parameters, we control the nature of the datasets generated. Specifically, we used the following parameter values $n = (15, 30, 50, 100, 200)$, $p = (0.15, 0.4, 0.75, 0.95)$, $M = (1, 2, 5, 10, 20)$, $D = (0.1, \ldots, 0.9)$, bottleneck=(TRUE,FALSE). Using these parameter combinations, we generated a total of $5 \times 4 \times 5 \times 9 \times 2 = 1800$ datasets.

3.3.1 Precedence Constraints

A crucial aspect of the dataset is the precedence relationships between different operations. [Hall and Posner, 2001] presents a method to generate precedence relationships and BOM networks by utilizing probability and arc density. Using this approach, we are able to control the shape and density of the network according to our needs. To do this, let P_{ij} be the probability of an arc existing between node i to j, where $1 \leq i < j \leq n$ and D be the target arc density of the network:

$$P_{ij} = \frac{D(1-D)^{j-i-1}}{1 - D(1-(1-D)^{j-i-1})} \qquad D = \frac{2 \cdot \text{Total Number of Arcs}}{n(n-1)}$$

By varying D and n (the targeted number of nodes in the network), we can control the shape and density of a network according to our needs. To do this, for all $1 \leq i < j \leq n$, calculate P_{ij} and generate a random number γ_{ij} from a continuous uniform distribution $U(0, 1)$. If $P_{ij} \leq \gamma_{ij}$, then include the arc (i, j) in the network. Note that the condition $i < j$ ensures that the graph remains acylical, which means that no chain of operations will ever form a closed loop.

3.3.2 Operation Times

Most studies such as [Ow, 1985] generate machine speeds through the use of a uniform distribution. However, this may bring up biases into the dataset [Hall and Posner, 2001]. As such, a logarithmic distribution is preferable to generate the machine speeds of a dataset. In the context of this project, we need to generate the processing times of each operation to execute the scheduling algorithm. As such, we take the inverse of the machine speeds to generate this data. Let S be a random variable representing the speed of a randomly selected machine. Then, the processing time of an operation, T, is given by:

$$S \sim \text{logarithmic}(p), \quad 0 < p < 1$$
$$T = l/S$$

where p is the parameter of the logarithmic distribution and l is an arbitrary number representing the 'length' or amount of work involved in an operation. l can be chosen as a constant value or generated using a continuous uniform distribution, depending on the nature of the operation.
Through Python, we generate a visualization of the network too as shown below.

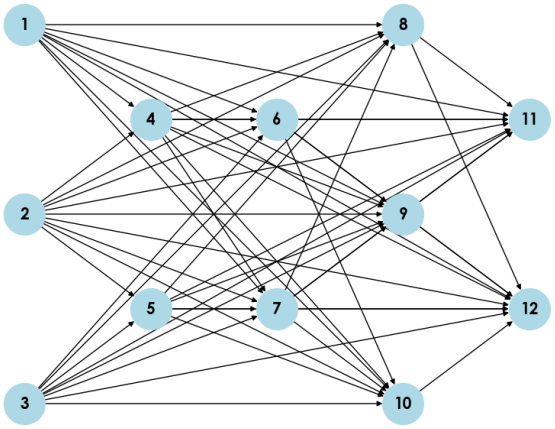

Figure 1: A Sample Graph Generated and Visualized Using Python ($D = 0.95$, $n = 12$)

3.3.3 Machine Routing

As a component travels through the assembly line, it may partake in multiple operations involving various types of machines. This sequence of machines is known as the machine routing: it determines the type of machine to use in each operation before a component transitions into a higher level product. To generate this data, [Hall and Posner, 2001] suggest using transition probability matrices—a concept in Markov Chains—to simulate the machine routing of all operations.

Given M machines, then the transition probability matrix A is a $M \times M$ matrix with each entry in row i and column j representing the probability that an operation on machine i is followed by an operation on machine j. At the start of the assembly process, if each machine is equally likely to be used, then each entry takes the value of $1/M$. This assumption can be modified as needed by following the machine's usage distribution.

For every operation, the probability values are used to determine which machine to use. Afterwards, we can update the matrix A in several ways. For example, after a job visits a machine, the corresponding matrix entry can be assigned a value of 0 to ensure the same machine is not used again in the next consecutive operation. Following the laws of probability, each entry in a row must sum up to 1 and as such, the remaining entries in the row can then be normalized so that they again sum to one.

We are also able to control what machine is more likely to be involved in any step of the assembly given a set of predecessor operations. Specifically, we can generate bottlenecks within the machine routing (bottleneck=TRUE), enabling us to test our system for a greater variety of cases. The generated machine routing would be in such a way that all or most routings would convert onto a single designated machine, thereby creating a bottleneck.

At present, we have generated both machine routings with and without bottlenecks. A component is also not restricted from being processed by the same machine more than once, whether consecutively or not. This assumption can be changed along the way as necessary, depending on the nature of the operations to schedule.

References

[A. Agrawal, 1996] A. Agrawal, G. Harhalakis, I. M. . R. N. (1996). 'just-in-time' production of large assemblies. pages 653–667. https://doi.org/10.1080/15458830.1996.11770710.

[Deliktaş et al., 2024] Deliktaş, D., Özcan, E., Ustun, O., and Torkul, O. (2024). A benchmark dataset for multi-objective flexible job shop cell scheduling. *Data in Brief*, 52:109946. https://www.sciencedirect.com/science/article/pii/S2352340923009770.

[Hall and Posner, 2001] Hall, N. G. and Posner, M. E. (2001). Generating experimental data for computational testing with machine scheduling applications. *Operations Research*, 49(6):854–865. https://doi.org/10.1287/opre.49.6.854.10014.

[Jun et al., 2021] Jun, W., Liu, X., Zhang, W., and Xu, J. (2021). A New Remanufacturing System Scheduling Model with Diversified Reprocessing Routes Using a Hybrid Meta-heuristic Algorithm. https://figshare.com/articles/dataset/A_New_Remanufacturing_System_Scheduling_Model_with_Diversified_Reprocessing_Routes_Using_a_Hybrid_Meta-heuristic_Algorithm/17026007.

[Ow, 1985] Ow, P. S. (1985). Focused scheduling in proportionate flowshops. *Management Science*, 31(7):852–869. https://doi.org/10.1287/mnsc.31.7.852.

[Vakhania and Mamporia, 2020] Vakhania, N. and Mamporia, B. (2020). Fast algorithms for basic supply chain scheduling problems. *Mathematics*, 8(11). https://doi.org/10.3390/math8111919.

A Appendix

A.1 Benchmark Dataset Data

Dataset Contents: The dataset comprises:

- n: Total number of jobs
- m: Total number of machines
- C: Total number of cells
- opi: Total number of operations for job i
- N_Cmax: nadir point of the makespan objective
- N_TT: nadir point of the total tardiness objective
- I_Cmax: ideal point of the makespan objective
- I_TT: ideal point of the total tardiness objective
- MaxFES: maximum number of fitness evaluations

Figure 2: Data Cleaning for Benchmark Dataset

			1	2	2	3	4	4	Machine
Part Family	Job	Operation	1	1	2	1	1	2	Alternative(for machine)
1	2	1		4	4				
1	2	2	3						
1	2	3		4	4				
1	9	1		5	5				
1	9	2	4						
1	9	3		5	5				
2	4	1				6			
2	4	2					2	2	
2	4	3				6			
2	22	1					3	3	
2	22	2				8			
2	22	3					3	3	
2	22	4				8			

Figure 3: Instance #1 from dataset

Job	Due Date
2	77
9	50
4	3
22	11

Figure 4: Due dates of Jobs in Instance #1

Machines	Parts										
	P_1	P_2	P_3	P_4	P_5	P_6	P_7	P_8	P_9	P_{10}	P_{11}
M_1	4	3	–	–	6	1	6	3	4	–	–
M_2	5	4	7	–	2	–	–	9	5	7	–
M_3	6	–	2	6	–	5	–	–	–	9	5
M_4	–	–	4	2	3	7	5	–	–	3	–
M_5	–	–	–	–	–	–	5	4	–	–	–
M_6	–	–	–	–	–	–	–	–	–	–	3
Route (Except Ins#31)	1-2-3-2[a]	2-1-2[a]	4-2-3[a]	3-4-3	4-2-1-4[a]	1-4-3-1	4-1-5[a]	1-2-5	2-1-2	3-2-4	6-3-6-3[b]
Route for Ins#31*	1-2-3-2-3-1	2-1-2-1-2	4-2-3-2-4	3-4-3-4-3	4-2-1-4-1-4-1-4	1-4-3-1-4-1	4-1-5-4-1	1-2-5-1-2	–	3-2-4-2-4	6-3-6-3-6-3
Due dates	98	77	75	3	48	2	5	110	50	8	53

Figure 5: Processing times, routing and due dates belonging to each part family processed by each machine

Part Family	Job	Operation	Processing Time	Machine	Components Required
1	2	1	4	2	
		2	3	1	
		3	4	2	
	9	1	5	2	
		2	4	1	
		3	5	2	
2	4	1	6	3	
		2	2	4	
		3	6	3	
	22	1	3	4	
		2	8	3	
		3	3	4	
		4	8	3	

Figure 6: BOM Table of Instance #1

B Appendix

B.1 Remanufacturing Dataset Reformatting

Dataset Contents: The dataset comprises:

- Product_category: Number of unique products
- totalroute_number: Number of unique workcentres
- productnum_number: Quantity of each product
- componentcat_number: Number of components for each product
- The type of the ith product jth component damage type number is: Classifies the damaged components by the type and degree of its damage
- The disassemble time of the ith product is: Time taken to disassemble a EOL product into its components
- The reassemble time of the ith product is: Time taken to reassemble a like-new product using the fixed components
- The type of the ith product of the jth component the kth damage type can select the route is: The sequence of workcentres the damaged component has to visit
- The type of the ith product of the jth component the kth damage type select the the nth route restore time is: The processing time at the nth workcenter in its route, based on the damage type and component

A screenshot of an example of the raw CSV file is shown in the figure below.

The type of the 0th product 0th component damage type number is	3
The type of the 0th product 1th component damage type number is	3
The type of the 1th product 0th component damage type number is	3
The type of the 1th product 1th component damage type number is	3
The disassemble time of the0th product is	2
The disassemble time of the1th product is	3
The reassemble time of the0th product is	1
The reassemble time of the1th product is	2
The type of the 0th product of the0th component the0th damage type can select the route is	[[9, 10, 5, 7]]
The type of the 0th product of the0th component the1th damage type can select the route is	[[9, 1, 4]]
The type of the 0th product of the0th component the2th damage type can select the route is	[[9, 4, 10, 3]]
The type of the 0th product of the1th component the0th damage type can select the route is	[[9, 7, 2]]
The type of the 0th product of the1th component the1th damage type can select the route is	[[9, 1, 4]]
The type of the 0th product of the1th component the2th damage type can select the route is	[[5, 9, 10]]
The type of the 1th product of the0th component the0th damage type can select the route is	[[9, 10, 5, 7]]
The type of the 1th product of the0th component the1th damage type can select the route is	[[6, 4, 2, 5]]
The type of the 1th product of the0th component the2th damage type can select the route is	[[9, 4, 10, 3]]
The type of the 1th product of the1th component the0th damage type can select the route is	[[10, 5, 6, 9]]
The type of the 1th product of the1th component the1th damage type can select the route is	[[10, 5, 6, 9]]
The type of the 1th product of the1th component the2th damage type can select the route is	[[8, 7, 2]]
The type of the 0th product of the0th component the0th damage type select the the0th route restore time is	[2]
The type of the 0th product of the0th component the0th damage type select the the1th route restore time is	[2]
The type of the 0th product of the0th component the0th damage type select the the2th route restore time is	[2]
The type of the 0th product of the0th component the0th damage type select the the3th route restore time is	[2]
The type of the 0th product of the0th component the1th damage type select the the0th route restore time is	[2]

Figure 7: Raw Data of Instance from Remanufacturing Data

By extracting the relevant information and formatting it into the desired format, we obtained the final dataset to be processed as follows:

Product 0	Part	Operation	Components Required	Processing Time	Workcenter
	D	D.10		2	WC#0
	A1	A1.10	D	2	WC#9
		A1.20	D	2	WC#10
		A1.30	D	2	WC#5
		A1.40	D	2	WC#7
	A2	A2.10	D	2	WC#9
		A2.20	D	2	WC#1
		A2.30	D	2	WC#4
	A3	A3.10	D	2	WC#9
		A3.20	D	2	WC#4
		A3.30	D	2	WC#10
		A3.40	D	2	WC#3
	B1	B1.10	D	3	WC#9
		B1.20	D	4	WC#7
		B1.30	D	4	WC#2
	B2	B2.10	D	3	WC#9
		B2.20	D	5	WC#1
		B2.30	D	5	WC#4
	B3	B3.10	D	4	WC#5
		B3.20	D	5	WC#9
		B3.30	D	5	WC#10
	C	C.10	A, B	1	WC#11

Figure 8: BOM Table of Instance from Remanufacturing Data