ISY5001 INTELLIGENT REASONING SYSTEMS
Group Project Report

Machine Scheduling Optimizer

Chen Liwei (A0101217B)
Raymond Djajalaksana (A0195381X)
Ng Cheong Hong (A0195290Y)
Seah Jun Ru (A0097451Y)
Lee Boon Kien (A0195175W)

1.0 EXECUTIVE SUMMARY

Long viewed as a key pillar of Singapore's growth, manufacturing accounts for nearly a fifth of the nation's GDP. (EDB, 2017) However, the industry is currently facing a lot of pressure from both domestic restructure and regional competition.

Beyond the global economic headwinds, rising operational costs, weakening of Singapore dollar and domestic labour crunch have worsened the performance of local industry. In order to resolve these mounting challenges, Singapore government has implemented a few policies to encourage local industry to revamp its manufacturing model into one that focused on the four design principles in industry 4.0 - interconnection, information transparency, technical assistance and decentralized decisions. One of the policy implemented is to provide incentive to local industry for more innovation-based and high-value production.

In manufacturing sector, it is common that the local industry faces the issue of optimizing the usage of machinery, scheduling of resources and decision making in the product flow due to complexity in solving via human knowledge. The issues mentioned have caused a huge loss in total revenue every year, which could be prevented if optimization plan is being carried in the initial stage.

In order to resolve the issues mentioned, our team has targeted the optimization issue in manufacturing industry, with the assumption of various real world constraints which could be amended according to the company's need, which consists of limitation on machinery, process lead time, cost of components and etc. We focuses on maximizing the profit earned in each month and the quantity of product to be delivered.

2.0 PROBLEM DESCRIPTION

For the production of each component, the material will needs to go through multiple processes. Some processes are repeated at different stages of the production, such as inspection of material. At times, the machinery for certain process might be in the idle state, resulting in wastage of resources since the fixed costs for machinery will still be expensed in the idle state. If we were to route other components orders to use the machinery, we would be able to increase the utilisation rate of the machinery and thereby potentially increasing profits as well since more orders can be processed within the same period of time. In theory, it is easy for us to understand and advocate the aforementioned practice. However, in practical, it is difficult for human to derive an optimal schedule by manually calculating as the number of possible permutations is simply too large. The number of possible solutions increases exponentially with an increase in the number of jobs and associated operations. The exponential growth makes it difficult to use classical computational methods and/or exhaustive search-based approaches to find the global optimum schedules. Such a problem is a combinatorial optimization problem.

2.1 PROJECT OBJECTIVE

After interviewing a domain expert in the manufacturing industry, we designed our system to one to suit the industry requirements. In our project, we have identified the following 5 different processes required to produce 8 components in a factory comprising 7 unique machines.

For simplicity, we will be referencing the processes, as per their alias, as shown in Figure 2.1A, for example P1 for CNC Machining, P2 for Deburring and so on.

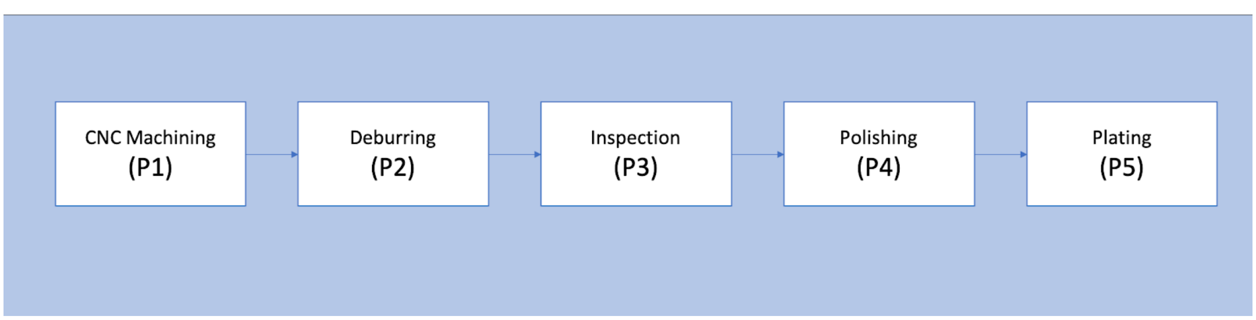


Figure 2.1A. Sequence of processes for production of components

In the manufacturing factory, a total of 7 type of machineries were procured for production. Each machinery type is unique and some of them are able to handle different manufacturing processes. The electricity consumption of each machine is also different, resulting in differing cost/min. Table A.1. in Appendix A illustrates the detailed information for each machine.

The sequence of the processes are fixed but each component may skip certain processes if not needed. Table A.2. in Appendix shows the processes required as well as the duration of each process for the production of each of the individual components.

Based on the current observation, there are quite a few areas that we could try to improve on. For example, long waiting time had been observed after finishing each process, as the machine required to carry out the next process was fully loaded. Besides, there are a few moments that the machines were in idle mode, due to poor scheduling of process. The objective of our project is to find the global optimum schedules to shorten the waiting time as well as maximizing the machine utilization rate, in order to achieve the maximum revenue generated.

While the objective result will depend on the current manufacturing equipments and the component process (Appendix A.1., A.2.), we have pre-defined these factors before in our system. The reason is because most of the time these variables will not change quite frequently. However we can always improve our system to take on these variables as user input.

2.2. ASSUMPTIONS

As there are many different variables and that could influence the behaviour and performance of the system, we have made the following assumptions when we build our system :

1. The machines are able to run without any human intervention, 24/7.
2. The operating cost of each machine varies between different machines, but is a fixed rate which will not change according to depreciation.
3. All of the components produced by the company are required to go through the production processes in the same sequence as shown in Fig.*2.1A*.
4. Each component may require different duration for different processes. An example would be that inspection might take longer for component A as compared to component B.

5. Certain components may skip certain process. An example would be that component A might not need to go through the third process in Fig.2.1 A.
6. Fixed operating expense is immaterial as compared to material and machine utilization cost.
7. There is no defective component produced and rejection rate is zero.
8. Machine can only work on one task at a time.
9. A task, once started, must run to completion.
10. Normally two or more machines can work on the same component so that the component will be produced more at the same time (example Machine 1 will produce A and another Machine 2 will also produce other A on the exact timeline, hence 2 A can be produced on the same time).

3.0 MODELING

Our system is modelled based on typical production planning process which takes production routing of products and the required quantity as input and assign to specific machines and processes based on their capacity.

Second, manufacturing businesses are likely to have multiple orders from different customers. Often the pricing strategy may vary for different customers due to order size or other business reason. Our system should be able to accommodate such business constraints to deliver maximum value with existing resources.

In the scenario when the production capacity is unable to support all the demands, our system needs to be intelligent enough to decide which order to consume and the right product mix to yield highest return of investment (ROI) with our resources.

Instead of just optimizing operational efficiency, the key goal of our system is to extract maximum business value by coupling the business process to the scheduling algorithm. This is simply because maximum operational efficiency doesn't always guarantee best return due to the influence of product pricing factor which is a dynamic variable.

3.1 BACKGROUND KNOWLEDGE AND USER CONFIGURABLE INPUTS

For our project, the following data/formulas are predefined:

1. The number of processes for production. (As observed in Fig.*2.1A*, there are 5 possible processes)
2. The number of machines available for each process type (As seen in Table 2.1C)
3. The total available number of machine hours for each machine for the month (In this model, we defined the month to have 21 working days, 8 hours per day)
4. The number of types of components produced by this company. (In this model, we defined a maximum of 8 different components, as seen in Table 2.1B)
5. The profit of each component is calculated based on formula:
   Profit = (Price of component - cost of machinery for all required processes to produce that component - cost of component),
   with the assumption that the fixed operating expense is negligible in our model.
6. The processes cannot run in parallel for each component.

3.2 CHROMOSOME REPRESENTATION

GA permutation happens for machine allocations for each component, with the aim to maximize profits. In our model, each chromosome contains 8 genes (representing each component). Each gene contains an ordered list of machines that the material will visit in order to be produced. The index positioning of each gene is fixed in order to reduce the number of unnecessary permutations, thereby reducing search space as well as the time needed for the GA to run. To compensate for the optimising of scheduling of machines, we have created another fitness function that will score the solution based on the extent of optimization of the machine schedule.

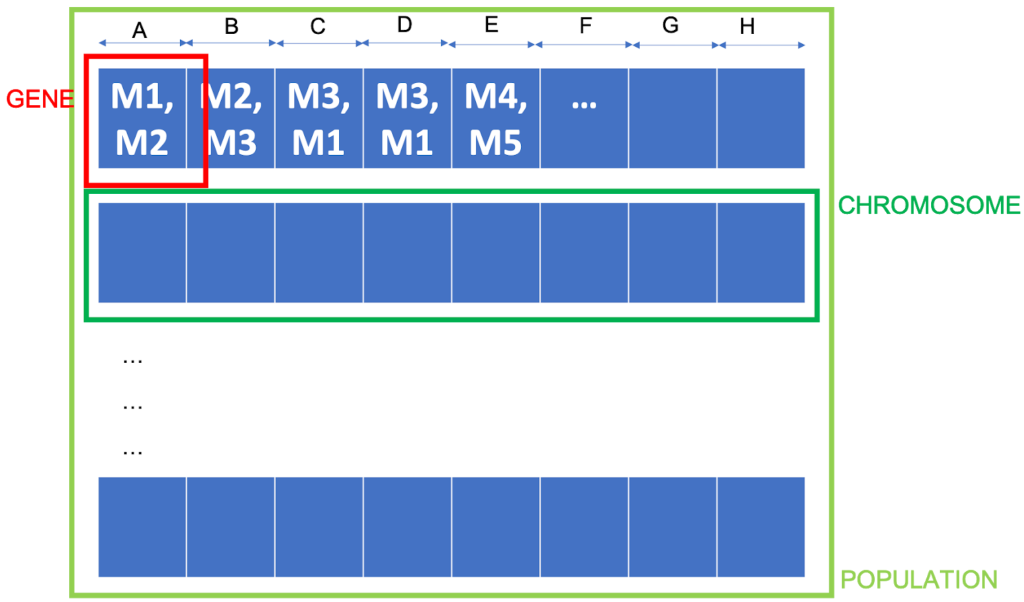


Fig.3.2 Population Diagram and chromosome representation

3.3 FITNESS FUNCTION DESIGN

In the design of fitness function, we have taken into consideration of the following objectives to optimize the schedule of machine. First, as the business profit is always in the top list when comes to manufacturing industry, we have allocated a relatively higher weightage on profit earned when computing the fitness score. Besides, the machine utilization rate is important as well. If there was machine being underutilized, the factory would have to suffer for the lost in revenue due to the fixed cost invested on the machine. Hence, we have included the number of product delivered into the fitness function, with no compromise on product delivered while optimizing the profit earned.

The table 3.3 below shown fitness priority, or so called weight consideration for fitness function integration. Since Profit Earned and Number of Product Delivered are considered almost equally important, hence the fitness priority should be in medium for such case.

Fitness Breakdown	**Fitness Priority**	**Weightage Distributed**	**Remarks**
Profit Earned	Medium	0.6	To maximize the profit earned by optimizing the machine schedule
Quantity of Product Delivered	Medium	0.4	To maximize the number of product delivered by optimizing the machine schedule
Integrated Fitness - To Max()	N.A.	1 (Summed up)	To maximize

Table 3.3 Fitness Priority for fitness function integration

Our fitness/model design bias toward firstly satisfying the profit earned before focusing on the quantity of product delivered.

3.4 FITNESS FUNCTION FORMULA

This is how we compute the fitness score of the chromosome in details:

$$f = C_1 * \left(\sum_{i=0}^{N} \frac{UP_i}{T_i} \right) + C_2 * NX$$

Where :

UPi = Unit profit for component i that can be achieved when assigning each of component process to the respective machines.

Ti = Cycle time of component i, e.g. time duration to wait so the next same component can be produced. The inverse value of this is will indicate the production rate of the component.

NX = number of distinct product that is produced. This is range from 0 <= Nx <= N (N is total distinct component)

C1 = Weightage value for profit earned. Setting it to 0 makes the fitness score to ignore the profit in optimizing the algorithm

C2 = Weightage value for quantity of product delivered.

C1 and C2 must be adjusted depending on business requirement/scenario (eg. whether we want to optimize for range product or pure profit). Meanwhile UP and NX can actually be determined from the chromosome that we already have.

The problem will be how to calculate the Ti, that is what is the rate of production for each component with the chromosome plan that we have design. This value will be dependant on how we make a time schedule and assign it to the machine. For example in the fig 3.2. Component C and D has both been assigned to M3, and M1. But it cannot tell the exact planning. Which component Machine 3 has to process first ? Is it C or D ? Depending on the order it will create chain affect to the other process and components. Hence the overall rate production will be affected by this order.

To solve this issue, we can have 2 approaches :

1. [quickSearch] Use greedy approach directly to the chromosome. That is the first assignment in the first component / cell / gene has higher priority. Then we look into the first assignment in the second component and so on. After all first machine assignment has been finished, we move on to the second assignment and start from first component again. We will call this method as quickScan option.
   For example in fig 3.2: We will have this queue tasks from the chromosome:
   [M1 → A],[M2 → B], ….,[M4 → E],[M2 → A], and so on.
2. [GASearch] To solve greedy approach, we will try to shuffle the order of the components within the chromosome and create the schedule using greedy approach again with the new arrangement. However this is not easy because there will be N! number of permutation to completely calculate all the possible shuffle and arrangement. To balance between the accuracy and performance we are running the GA algorithm again to get the best ordered schedule. In our system we can run this with disabling quickScan option.

Another method is that we can consider the component ordering in the chromosome itself. However this will again dramatically increase the search space of our main GA (e.g. when breeding we need to produce ordered component list and the machine assignment to each component). Due to time constraint of this project, we will not consider this approach, and only proceed with method 1 and 2 only.

3.5 CONSTRAINTS

Hard constraints

1. Sequence of machinery processing
2. One machine only can process one component at any one time

Soft constraint

1. Minimum quantity of components to be produced unless no order is received for that component at all
2. Time limit

3.6 GENETIC OPERATORS

We used Genetic Algorithm for Machines and Components permutation, with the following setting being hard coded in the backend:

Population size = 100, Elite size = 20 , Generations = 100

Default Crossover Rate: 80 % , Mutation rate = 5%

There is another Genetics Algorithm to find the best schedule arrangement for each main chromosome. This GA has configurations below:

Population size = 75, Elite size =15, Generations = 50

Default Crossover Rate: 80 % , Mutation rate = 2%

As we are building a web application, we do not want to underwhelmed the search algorithm with long process because it makes a bad experience to user. As part of demonstration we choose quite small amount of size, generally around 100 population size. This is mainly due to our assumption on how many machines and component are there.

Of course these numbers actually be tuned to be dynamic number because the state search space depends on number of components and machines. But we will need to find the best parameters that can fit to all conditions which can be another task of improvement that we can do.

3.6.1. BREEDING PROCESS

Figure 3.6.1. illustrates the breeding process between two chromosome plan.

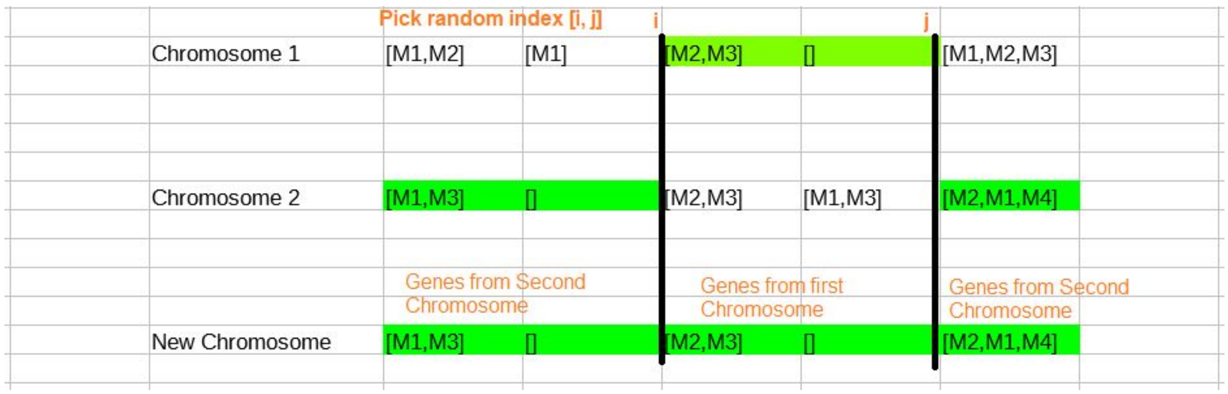


Fig 3.6.1. The breeding process of the main genetics algorithm

As for GA in 3.4 points 2, the breeding process is just a default crossover of between 2 chromosome. In fact the objective statements and constraint in this GA is almost exactly the same as TSP problem (find the best order components that has best arrangement value). See Appendix A.4. for the breeding process of the second GA model.

3.6.2. EARLY RETURN

We will return the GA prematurely in case the growth is stagnant. That is: when over 10 continuous generations the improvement of fitness score is always less than 2 % then we will end the GA algorithm as we will conclude that the result is stagnant and further evolution will only cost performance with little benefits of new discovery.

3.7 GENETICS ALGORITHM TO SOLVE THE PROBLEM STATEMENT

So far our GA have not solved the constraint for time limit and desired number of unit which actually the main concern of our business model.

There are two possible approach to solve these constraints. First, take into account the time limit and desired number into GA model. However as the model itself has been complicated, adding other constraints might not be feasible.

The second approach is to use divide and conquer method. The result might not be optimal when compared to the first approach, but it is very easy to be adapted with current genetics algorithm model.

Using this approach, we will get the schedules divided into some **batches**. This is actually quite convenient for manufacturing side to oversee the process, as repetitive cycle is something that is preferred in manufacturing. The steps to implement the solutions can be explained as below:

1. Create the list of components that we want to produced. Hence if desired unit for component X is 0, we ignore this and will not be included into genetics algorithm.
2. Run the GA with the list to be produced. We will get the best schedule. From here we know the production rate of each component. Using this, we will know the minimum time when one of the component will reach desired number. We call this minimum batch duration (Tb).
3. Given the Tb now, we can backtrack for other components how much we have produced during this duration. Subtract the desired number with this value.
4. Now we move the start time after Tb + 1 Max cycle time. 1 Max cycle time is necessary for one purpose: to put a buffer time in between batches (consider it delay as safety measurement in a case there is delay/breakdown).
5. Repeat no 1 for the next batch, as long as there is component that has to be produced or we are still within time limit given (start time < end time).

4.0 SYSTEM ARCHITECTURE

The system can be divided into 2 parts, frontend and backend.

The frontend is a web based application for user to interact with backend. We used React javascript framework to easily create and deploy interactive user interface. Users will key in the required values in the user-interface and the values are subsequently relayed to the backend via a REST request. After the backend is done analyzing and computing, the recommended results are then sent back to the frontend in the REST response and displayed in the user interface. With the myriad of libraries available for React, we were able to use libraries such as Google Charts to display our results from the backend in an interactive manner, such as the Timeline component of Google Charts.

On the other hand, we use Golang to as our backend to process the frontend request and run the genetics algorithm to find the best results. The reason for golang is because it is very good at concurrency and we can utilize concurrency to speed up the genetics algorithm process. This is because normally the algorithm time will linearly scale up based on population size since for every generation we need to calculate the fitness score for each individual or chromosome. By using concurrency we can simultaneously calculate all individual score at once. Running with some set of test that requires the backend to run GA 6 times, the non-concurrency requires 88.1 ms on average while concurrency only takes about 48.1 ms (with using greedy method to calculate fitness score). Hence on average we have optimize the speed by twice.

4.1 SYSTEM'S FEATURES

Despite of the limitations mentioned in the previous sections, we were able to implement some key features as illustrated in this section.

4.1.1 SYSTEM REUSABILITY & SCALABILITY

Firstly our frontend and backend are decoupled by REST Application Programming Interface (API). This mean that our frontend can be used with any other backend easily and vice versa. For example if we deploy several different rules engine on many backend server, frontend just need to change the API to get result from different engines. On the other hand our backend can be used by many other frontend too as long as the api contract is satisfied.

Furthermore all the machines, process, components metadata are stored in json format file, hence it allows to add, edit, or remove any changes to match with business/client need without any change of the code. This will ease the server deployment as the changes can be made in runtime without any effort of downtime due to redeployment.

4.1.2 EASE OF USAGE

As the system is a web based application, it can be easily accessed by any electronic device nowadays as long as the device has web browser and internet connection.

4.2 LIMITATIONS

The knowledge model that are used to build the system has been simplified This is mainly happen because of the complexity of the problem that we would like to tackle. The result of knowledge acquisition is insufficient to give all the insights needed to solve the problem. Hence the system tend to focus to solve on one small part of the area. However going forward we can expand our system to cover more cases from different perspectives.

5.0 CONCLUSION & REFERENCES

The group discussion session was fruitful and rewarding as we have learnt much from each other. Even though every one of us has personal commitment on our work, we still committed a substantial portion of our rest time and weekend to deliver the project. The time spent on this group project was not wasted, the joy of witnessing the completion of final product was unforgettable and we are glad that we have successfully achieved our objective of optimizing the machine schedule.

5.1 IMPROVEMENTS

If we were given a longer duration to work on this project, we would have worked upon the following points of improvement:

5.1.1 REJECTION RATE

In our system design, we assumed that all goods produced by the machines through the processes are without defects. However in reality, it is almost impossible to achieve zero defect and rejection rate. If we had a longer project timeframe, we could take into account of rejection rate and set minimizing of rejection rate as a soft constraint. Including reworking scenario would enhance the real-time planning capability to guide user in deciding rework schedule for defective parts with the least impact to existing production schedule.

5.1.2 TIMELINE OF GOODS

When companies submit purchase orders, usually they are not timed perfectly to be received at the start of the month and expect the components to be received at the end of the month. If we had more time to enhance the algorithm, we could have taken into account of the timeline of the goods.

5.1.3 DOWNTIME FOR MACHINES

In a happy scenario, we could assume that machines do not break down at all but it is seldom the case in reality. To consider the possible downtime of machines by taking in a parameter to indicate whether the machine is down, we could make our system more realistic.

5.1.4 HUMAN OPERATOR COST

In our system, human operator cost and availability is not included in our problem definition. Depending on the level of automation, there are operations that may still require human to operate and monitor. For example, material loading/unloading into machine and in-line inspections that cannot be done by machine.

6.0 BIBLIOGRAPHY

https://www.researchgate.net/publication/236004224_A_genetic_algorithm-based_approach_for_optimization_of_scheduling_in_job_shop_environment

Appendix

APPENDIX A BUSINESS AND KNOWLEDGE MODELLING

A.1. Detailed capability and running cost of each machine

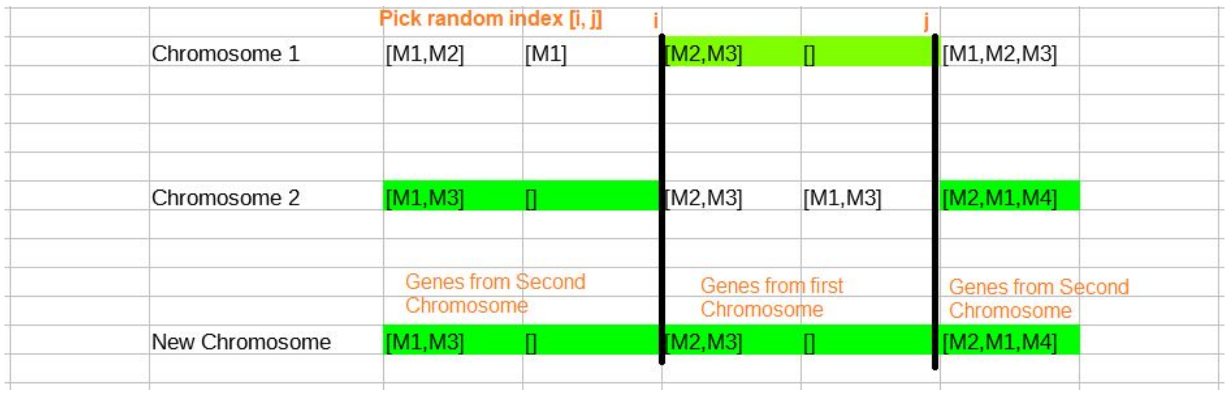
Machine Type	Process	Number of Machines	Operating cost ($/min)
CNC 5-axis turning/milling machine	CNC Machining (P1)	3	0.05
CNC 3-axis milling machine	CNC Machining (P1)	2	0.03
Deburring machine	Deburring (P2)	2	0.02
Polishing machine	Polishing (P4)	2	0.03
Coordinate measuring machine	Inspection (P3-1)	2	0.02
Vision measuring system	Inspection (P3-2)	1	0.02
Electroplating machine	Plating (P5)	1	0.1

A.2. Process duration for production of each component

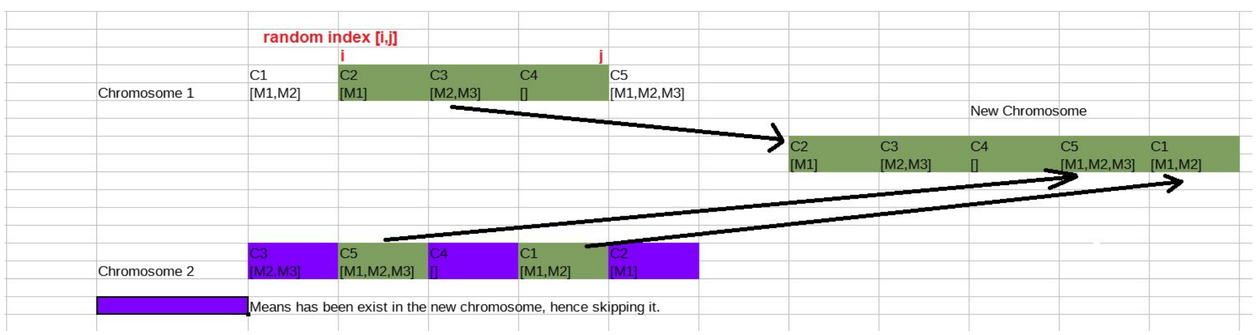
Components	Processes required	Duration required for process (min)
Samsung S9 Silicone Case (Coloured)	P1	5.0
	P2	3.0
	P3-1	1.0
	P5	3.0
	P3-2	1.0
Huawei P30 Clear Case	P1	5.0
	P2	3.0
	P3-1	1.0

	P4	2.0
Samsung Galaxy Tab A.10.1 Case	P1	7.0
	P2	3.0
	P3-1	1.0
Microsoft Surface Pro 5 Protective Case	P1	7.0
	P2	5.0
	P3-1	1.0
	P4	2.0
Iphone X Normal Case	P1	4.0
	P2	3.0
	P3-1	1.0
Iphone XS SE Case (Gold Colour)	P1	4.0
	P2	2.0
	P3-1	1.0
	P5	3.0
	P3-2	1.0
Iphone XS Colour	P5	5.0
	P3-2	2.0
Ipad Pro 12.9 inch Case	P2	2.0
	P3-1	1.5

A.3.The breeding process of the main genetics algorithm



A.4. The breeding process of the 2nd genetics algorithm to solve the best machine schedules, given the assignment in first genetics algorithm of chromosome plan.



APPENDIX B PERFORMANCE RESULTS OF GENETICS ALGORITHM

Due to complexity of the task and design, we have created several models and we can actually freely switch between configurations. Hence it is natural for us to find the best configurations from all aspect, or at least to get jack of all trades setup.

As described above, we have 2 main configs:

1. Whether to use quickScan (greedy to calculate score) or do full search (use another GA to calculate score)
2. Whether to use concurrency or not

And we will have 2 main set of test:

1. Simple test set, that is giving the all the machines and components information to get the best schedule without any constraint on the time and desired unit to be produced. The GA algorithm will be runned only once.
2. Test set with mocks of real problem. That is, given all the machines and components to be produced with time limit and desired unit constraint, the system must now generate all the plans within the time limit to fulfill all desired unit constraint.

Each config has 2 possible states, giving it 4 possible states for each test cases. The table below show the average result of the test:

	Simple Test Case		Real Problem Test Case	
	With concurrency	No concurrency	With concurrency	No concurrency
Quick search	Best score: 9.4 Duration: 58	Best score: 9.6 Duration: 100	TP*: 24749 Duration: 82	TP*: 24819 Duration: 144
Thorough search	Best score: 21.4 Duration: 23477	Best score: 21.14 Duration: 29432	TP: 37356 Duration: 116867	TP: 38293 Duration: 139831

*TP : Total Profit
Duration is in ms

From the table we can easily conclude that concurrency gives better performance while keeping the score the same. Another point is that the thorough search using another GA will give much better local maxima in trade of its duration performance (~50% more on real problem set).

Hence it is obvious that it is better for us to utilize concurrency. However in case we are running the system with small memory capacity, we might consider to use non concurrency.

APPENDIX C USERS MANUAL

SYSTEM OVERVIEW

Our Machine Scheduling Optimizer is mainly targeting manufacturing area. While in this demonstration we cover a basic manufacturing process, our system can easily be extended and customized to any manufacturing process. It is a simple application that will create and plan the best machine scheduler for you to produced the desired components. The system will try to maximize your profit will creating such plan. All you need to do is just to fill in price and number of components to be produced.

USER INTERFACE

Our user interface runs using react js framework. Once our backend engine returns the computation of machine schedule optimization, the web application will show all the details of machine schedule process, including unit produced, profitability, and schedule visualisation to help imagining the schedule process.

RECOMMENDED BROWSERS

Credit Card Recommendation System supports the following Web Browsers:

- Internet Explorer 11
- Google Chrome Version 72 and above
- Safari Version 12 and above

REQUIREMENTS

- nodejs and npm should be installed. Otherwise please download and install from the following website: https://www.npmjs.com/get-npm
- To run the backend system, you can just run the binary file (src/go/main.exe). But it is also recommended to always install Golang version 1.12.4 or later. Please follow the installation in https://golang.org/dl/

INSTALLATION AND DEPLOYMENT

- [Node.js](https://nodejs.org/en/ "Node.js ")
- [Optional] [Golang](https://golang.org/dl/ "Golang")

```
# 1. install all front end dependencies
cd SystemCode/company-order-form
npm i react-scripts
```

```
npm install


# 2. Run both web app and backend system
start_server.sh # to start backend system
web_app.sh # to start web app
start.sh # to run both start_server.sh and web_app.sh
```

APPENDIX D SAMPLE INPUT & SYSTEM OUTPUT

D.1 Scenario 1

<table>
<tr><td>Characteristic of sales</td><td>Received order on 3 type of casings and produced all orders</td></tr>
<tr><td>Input</td><td>Quick Scan: Off

<table>
<tr><th></th><th>Cost ($/unit)</th><th>Price ($/unit)</th><th>Order Received (unit)</th></tr>
<tr><td>Samsung S9 Silicone Case (Coloured)</td><td>1</td><td>3</td><td>1500</td></tr>
<tr><td>Huawei P30 Clear Case</td><td>1</td><td>2</td><td>2000</td></tr>
<tr><td>Samsung Galaxy Tab A.10.1 Case</td><td>1</td><td>2</td><td>1000</td></tr>
</table>
</td></tr>
</table>

System output:

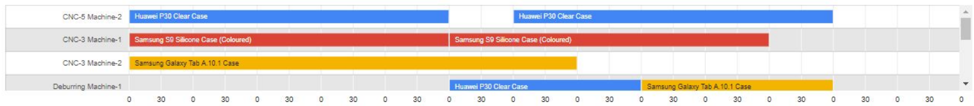
Result

Total profit is $4176.74.

Components that cannot be produced within the month

- Samsung S9 Silicone Case (Coloured): 0
- Huawei P30 Clear Case: 0
- Samsung Galaxy Tab A.10.1 Case: 0
- Microsoft Surface Pro 5 Protective Case: 0
- Iphone X Normal Case: 0
- Iphone XS SE Case (Gold Colour): 0
- Iphone XS Colour: 0
- Ipad Pro 12.9 inch Case: 0

Batch No. 1

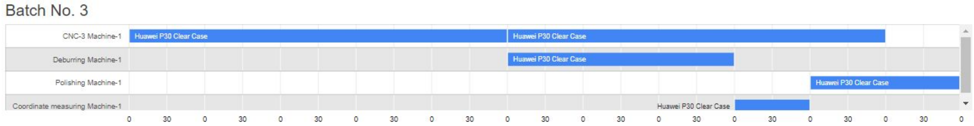


	Samsung S9 Silicone Case (Coloured)	Huawei P30 Clear Case	Samsung Galaxy Tab A.10.1 Case
Units Produced	1400	1166	1000
Unit Profit	1.45	0.6	0.71
Cycle Time	5	6	7

Batch No. 2

	Samsung S9 Silicone Case (Coloured)	Huawei P30 Clear Case
Units Produced	100	100
Unit Profit	1.45	0.71
Cycle Time	5	5

Batch No. 3



	Huawei P30 Clear Case
Units Produced	734
Unit Profit	0.71
Cycle Time	5

Analysis of system output

Features	Output	Explanation of system's output
Profit	$4176.74	Based on the units produced and unit profit, the factory is able to achieve the optimum profit of $4176.74 after optimizing the schedule.
Outstanding component	0	The factory is able to manufacture all the orders received within the month
Recommended machine schedule	Refer to the graph attached above	Based on divide and conquer method, the system recommends to carry out the manufacturing in 3 batches.

D.2 Scenario 2

<table>
<tr><td>Characteristic of sales</td><td>Received order on 3 type of casings with outstanding orders</td></tr>
<tr><td>Input</td><td>Quick Scan: Off
<table>
<tr><td></td><td>Cost ($/unit)</td><td>Price ($/unit)</td><td>Order Received (unit)</td></tr>
<tr><td>Samsung S9 Silicone Case (Coloured)</td><td>1</td><td>3</td><td>3000</td></tr>
<tr><td>Huawei P30 Clear Case</td><td>1</td><td>2</td><td>2000</td></tr>
<tr><td>Samsung Galaxy Tab A.10.1 Case</td><td>1</td><td>2</td><td>1000</td></tr>
</table>
</td></tr>
</table>

System output:

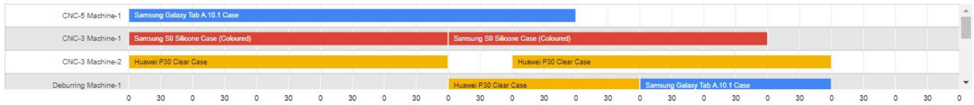
Result

Total profit is $6147.3.

Components that cannot be produced within the month

- Samsung S9 Silicone Case (Coloured): 126
- Huawei P30 Clear Case: 0
- Samsung Galaxy Tab A.10.1 Case: 0
- Microsoft Surface Pro 5 Protective Case: 0
- Iphone X Normal Case: 0
- Iphone XS SE Case (Gold Colour): 0
- Iphone XS Colour: 0
- Ipad Pro 12.9 inch Case: 0

Batch No. 1

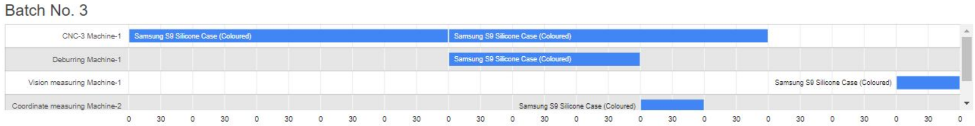


	Samsung S9 Silicone Case (Coloured)	Huawei P30 Clear Case	Samsung Galaxy Tab A.10.1 Case
Units Produced	1400	1166	1000
Unit Profit	1.45	0.71	0.56
Cycle Time	5	6	7

Batch No. 2

	Samsung S9 Silicone Case (Coloured)	Huawei P30 Clear Case
Units Produced	834	834
Unit Profit	1.45	0.71
Cycle Time	5	5

Batch No. 3



	Samsung S9 Silicone Case (Coloured)
Units Produced	640
Unit Profit	1.45
Cycle Time	5

Analysis of system output	Features	Output	Explanation of system's output
	Profit	$6147.3	Based on the units produced and unit profit, the factory is able to achieve the optimum profit of $6147.3 after optimizing the schedule.
	Outstanding component	Samsung S9 Silicone Case (Coloured): 126	Due to limited time and machine, 126 nos. of Samsung S9 Silicone Case (Coloured) remained as outstanding component.
	Recommended machine schedule	Refer to the graph attached above	Based on divide and conquer method, the system recommends to carry out the manufacturing in 3 batches.

D.3 Scenario 3

<table>
<tr><td>Characteristic of sales</td><td>Received order on 5 type of casings with outstanding orders</td></tr>
<tr><td>Input</td><td>Quick Scan: On
<table>
<tr><th></th><th>Cost ($/unit)</th><th>Price ($/unit)</th><th>Order Received (unit)</th></tr>
<tr><td>Samsung S9 Silicone Case (Coloured)</td><td>1</td><td>3</td><td>1200</td></tr>
<tr><td>Huawei P30 Clear Case</td><td>1</td><td>2</td><td>800</td></tr>
<tr><td>Samsung Galaxy Tab A.10.1 Case</td><td>1</td><td>2</td><td>1000</td></tr>
<tr><td>Microsoft Surface Pro 5 Protective Case</td><td>1</td><td>4</td><td>700</td></tr>
<tr><td>Iphone X Normal Case</td><td>1</td><td>5</td><td>500</td></tr>
<tr><td>Iphone XS SE Case (Gold Colour)</td><td>2</td><td>4</td><td>1000</td></tr>
<tr><td>Iphone XS Colour</td><td>2</td><td>5</td><td>600</td></tr>
<tr><td>Ipad Pro 12.9 Inch Case</td><td>2</td><td>6</td><td>1000</td></tr>
</table>
</td></tr>
</table>

System output:

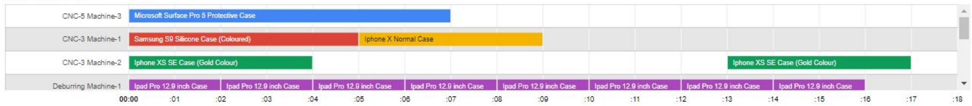
Result

Total profit is $13647.039999999997.

Components that cannot be produced within the month

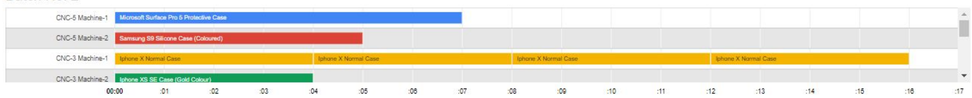
- Samsung S9 Silicone Case (Coloured): 0
- Huawei P30 Clear Case: 0
- Samsung Galaxy Tab A.10.1 Case: 0
- Microsoft Surface Pro 5 Protective Case: 0
- Iphone X Normal Case: 0
- Iphone XS SE Case (Gold Colour): 0
- Iphone XS Colour: 0
- Ipad Pro 12.9 inch Case: 0

Batch No. 1



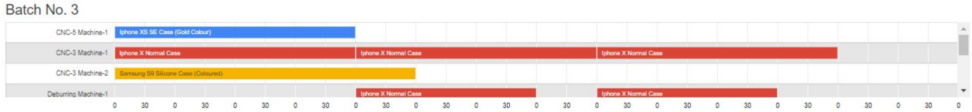
	Samsung Galaxy Tab A.10.1 Case	Microsoft Surface Pro 5 Protective Case	Iphone X Normal Case	Iphone XS SE Case (Gold Colour)	Iphone XS Colour	Ipad Pro 12.9 inch Case
Units Produced	153	153	153	153	153	1000
Unit Profit	1	2.46	3.8	1.49	2.46	3.93
Cycle Time	13	13	13	13	13	2

Batch No. 2



	Samsung S9 Silicone Case (Coloured)	Samsung Galaxy Tab A.10.1 Case	Microsoft Surface Pro 5 Protective Case	Iphone XS SE Case (Gold Colour)	Iphone XS Colour
Units Produced	168	168	547	218	168
Unit Profit	1.35	1	2.46	1.49	2.46
Cycle Time	13	13	4	10	13

Batch No. 3



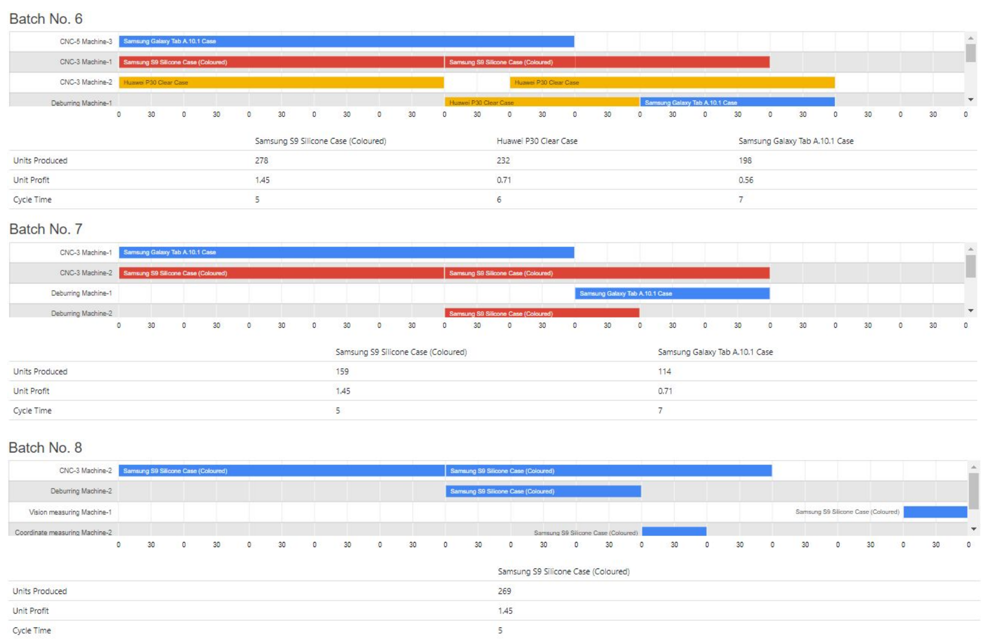
	Samsung S9 Silicone Case (Coloured)	Huawei P30 Clear Case	Iphone XS SE Case (Gold Colour)	Iphone XS Colour
Units Produced	193	193	629	193
Unit Profit	1.45	1	1.41	2.46
Cycle Time	13	13	4	13

Batch No. 4

	Samsung S9 Silicone Case (Coloured)	Huawei P30 Clear Case	Samsung Galaxy Tab A.10.1 Case	Iphone XS Colour
Units Produced	133	86	120	86
Unit Profit	1.35	0.6	1	2.46
Cycle Time	9	14	10	14

Batch No. 5

	Huawei P30 Clear Case	Samsung Galaxy Tab A.10.1 Case	Iphone X Normal Case
Units Produced	289	247	347
Unit Profit	1	0.56	3.8
Cycle Time	6	7	5



	Features	Output	Explanation of system's output
Analysis of system output	Profit	$13647.04	Based on the units produced and unit profit, the factory is able to achieve the optimum profit of $13647.04 after optimizing the schedule.
	Outstanding component	0	The factory is able to manufacture all the orders received within the month
	Recommended machine schedule	Refer to the graph attached above	Based on divide and conquer method, the system recommends to carry out the manufacturing in 8 batches.