***Dynamic Scheduling for Cyber Physical Systems***

**Harshit Kumar Singh under the supervision of Prof. Bhupesh K Lad**

**Overview**

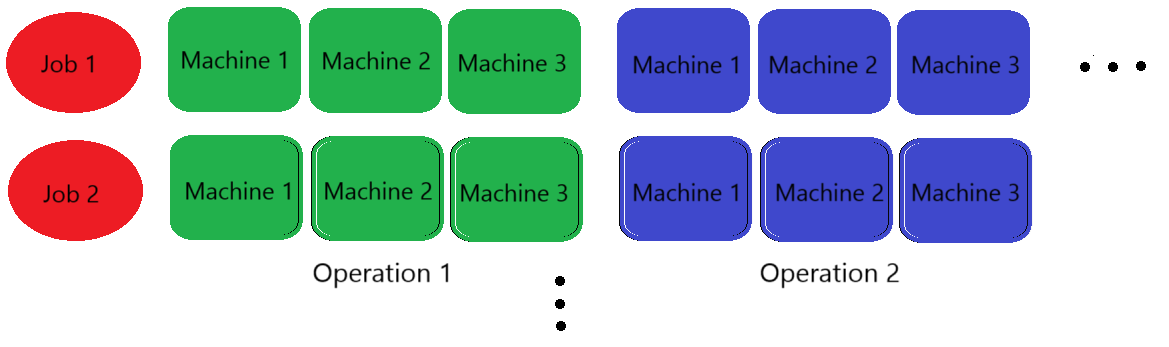
This project is focused on creating a method for scheduling a production in a high variety low volume (HVLV) type manufacturing plant. There are two types of jobs in this, one with a due date and a penalty, the other without any due date. We will call the one with penalty the critical jobs and others as non-critical. All the jobs have a priority which indicated the importance of that product for production. There are many machines which are used for specific operations on each job. Each job has at most three operations in order to be done before they are completed. Each operation has three different machines as options to be used for that particular operation.

The daily dynamics of the plant contains the schedule of operators for each machine, the machine, plant, and operator downtimes, the availability of pallets and fixtures for the machine and the availability of raw materials. All of them are pre-planned but there are chances of sudden changes in the plant which solidify the need for dynamic scheduling.

We can divide our ultimate aim in three objectives, to minimize the total production time, to minimize the penalties on critical jobs and to maximize the machine utilization for each machine.

The very first thing to do before starting was to process the data so that it is easier to work with. Then we approximated the arrival and service rate of each of the machine so that we can know the approximate time it takes to manufacture each job. Then we try to create the best sequence for the jobs to be allocation on machines. And then we try to allocate the jobs to the machines. In the end we test all the different sequencing and allocation techniques we found.

**Initial data**

The data we received were of two types, the first was the plant data which contained the information about their machines, its components such as pallets and fixtures, the data about their operators, the parts produced in the plant, their operations and the machines which can be used in each operation and daily dynamics of the plant. The second type of data was the data about their previous schedules which contain all the data about all the parts they have produced in previous several weeks, the machines which were used to produce each part and how many parts were planned to be processed per week per day and how much were actually produced.

All this data was converted into one single excel sheet in a more understandable and workable format for further calculations.

**Approach**

What we want at last in this part of the project is a static schedule for one cycle of production where each part is allocated to a single machine for that particular operation. So, we have two questions so far. Which job to choose first and to which machine to give it to? This led us to divide the problem in two parts. First, we will create a sequence of jobs to be scheduled. After that we will pick each job from that sequence in that order to assign it to a machine.

**Job Sequencing**

Because we want to minimize the production time and maximize the machine utilization of each machine, we choose a sequence based on shortest processing time as our starting point. But since in our case we have critical jobs as well which may take more time than some other non-critical job but if it were to be put under that job then it will certainly breach its due date.

But if we use their due dates to sequence them, this problem will be solved. To calculate the due date for non-critical jobs we can calculate the total time it would take for them from putting into production to the time they are manufactured. This means we have to take the total time which includes the machine waiting time, machine setup time and the actual processing time for that job. To calculate the machine waiting time we calculated the approx. arrival rate and service rate for all the machines based on the data we created from the initial data received from the plant.

The arrival rate was calculated by dividing the total parts to be processed on that machine by the number of days until the next order. Then the average of all such numbers for the given several weeks was taken as the average arrival rate. The service rate was calculated by dividing all the jobs processed on that machine by the number of days it took them to process. Then the average of all these rates for the given several weeks were taken as the average service rate. The waiting time was then calculated using the formula:

**Waiting time = (AR/SR) / (SR – AR)**

AR – Arrival rate, SR – Service rate

For the jobs with no waiting time (*machine x*) the service rate was updated by taking the ratio of the arrival rate by the service rate of all the machines with positive waiting time (*w value*) and dividing the arrival rate of *machine x* with the average of all *w values.* This was done to be on the conservative side.

Not just that but we also had priorities assigned to each job which is an indication of the importance of that job in the production. Now we can manage this by simply sequencing all jobs with different priorities separately and then combine it according to the decreasing order of their priority. But this will not be the most efficient way of doing it considering we want to minimize the average production time for each job and also our greater goal is to create a dynamic scheduling algorithm where there is no ending point of a sequence.

Considering this a new formula was derived which calculated what we call prioritized due date.

**Prioritized Due Date = (CDD/NP)**

**NP = Priority / 10**

CDD – calculated due date, NP – normalised priority

This PDD (Prioritized Due Date) is calculated for every job for the maximum time and then a buffer of 20 percent is added to each of them. After that the critical jobs’ PDD are compared by their actual customer due date and if the later is earlier then the PDD value is changed to that. Finally, we arrange that into an ascending order sequence which gives us a sequence. We will call this sequence as our **default sequence.**

But we can also use one more sequencing technique in which we can modify the default sequence. We can shift all the critical jobs to the top without changing the internal order of critical or non-critical jobs.

In the end we have two sequences with us, the default sequence and the critical jobs shifted sequence. We will develop one more sequence to compare the two with that one which is created using brute force computation such that it gives us the best sequence based on our constraints. Now we are not taking that sequence to integrate into our sequence and only using it for comparison because it is very time consuming and computationally intensive which many small manufactures will find hard to afford.

**Machine Allocation**

After having a sequence, we can allocate the jobs to the machines. Our goal and focus in this part are to maximize the machine utilisation and minimize the total production time and total penalty. To achieve this, we used the method where the job is allocated to the machine (among the options) which after adding the job production time to its current flow time is minimum among the other options. And the job will go to next operation immediately after the completion of current operation in similar way.

This is a simple way to allocate jobs to machines which also promises good machine utility and close to least production time for the given sequence but it is hard to say same for the job penalty. Therefore, we developed another method whose primary focus is to minimize the critical job penalty. the way we can do this if we somehow keep the machine on which a critical job will be processed free at the time it will arrive for operation. Coming up with a method which can do something like this without using intensive computation is difficult but we came up with a method which can do this to some degree. We gave each machine a number we called **criticality index** which is the measure of how many critical jobs can be produced on that machine. This number is calculated for a particular machine by taking the ratio of sum of all the critical jobs’ production time (setup time and processing time) which can be processed on that machine for all operations and the sum of all such numbers. Based on that we will that allocate all the non-critical jobs to machine with the lowest CI (criticality index) among its options and the critical jobs to those machines with lowest CI given that it does not cross its due date in that machine. If it crosses than it moves to one machine above in the CI table which can process that operation and so on. Now this CI is a dynamic index for it will change after each job is allocated from the sequence. Therefore, we can have two methods in this, one in which update the CI after the allocation of every job and the other in which we calculate CI only one at the start and use it for all the jobs.

In the end we have three methods to allocate jobs to the machine, lowest flowtime-based allocation (LFA), Criticality index-based allocation (CIA) and Dynamic criticality index-based allocation (DCIA).

Since CI based allocation is primarily focused on reducing the due date breach and it is designed in such a way that machine utilisation and total production time will increase compared to the first allocation method, it is expected to see the average machine utilisation and production time in the order LFA > DCIA > CIA and LFA < DCIA < CIA respectively.

Now we have developed two ways to create sequence and three ways to allocate jobs to the machines. Therefore, we can have total 6 schedules based on that and three more based on the sequence created by brute force computation for comparison.

**Testing**

**Data**

We created total 500 datasets to the testing of all the nine scheduling methods. Each dataset had 20 jobs to process and only single operation. These datasets were created such that they were very similar to the data we received from the Bengaluru plant so that we can test it on the data which can exist in real life. All the elements of the data were in the same range as that of the real data, the priority of job to be critical was calculated and random jobs were made critical based on that, same was for the priority. The due dates were in the same range as in the real life for the given quantity of jobs to be produced.

**Methodology**

The simulation was done using python programming language. One dataset was taken at a time and was used to create all 9 schedules using the sequencing and allocation methods discussed above. From these schedules the testing metrics were measured. This was repeated for all the 500 datasets. This gave us 1500 different metrics set each for one dataset and in that one schedule. The average of those values was taken as the performance metrics for that pair of sequence and allocation.

**Testing metrics**

We calculated three metrics to test out methods. These were the average flowtime, average lateness or breach and average machine utilisation. Average flowtime is the measure of on an average how for what interval a job is being processed in the plant since it started. It was calculated by first averaging the flowtime of all the jobs in one dataset and then calculating the grand average for all 500 datasets. All other metrics were calculated in similar way. Average lateness is the average time interval for the critical job in process from the due date to the time it was completed or how late it was. Average late count tells the on an average how many critical jobs breached their due date. Machine utilisation for a machine is the ratio of the total machine operating time and the total schedule time.

**Results**

The table below gives the result of the simulation. It can be seen that the Critical job shifted sequence has lesser average lateness or breach and slightly better average machine utilisation among all the allocation methods than default sequence. But the average flowtime is higher than the default sequence in all the allocation methods. If we compare the two sequences with the sequence created by repeated calculation of different

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Sequencing** | **Allocation** | **Average Utilisation (%)** | **Average Lateness**  **(minutes)** | **Average Late Count** | **Average Flowtime**  **(minutes)** |
| **Default Sequence** | LFA | 56% | 10419.70774 | 1.264 | 31431.63414 |
| CIA | 38% | 10511.61572 | 0.822 | 41727.60882 |
| DCIA | 42% | 9792.649089 | 0.774 | 39630.51174 |
| **Critical Jobs Shifted Sequence** | LFA | 57% | 5506.939844 | 0.772 | 32413.71426 |
| CIA | 35% | 7687.520871 | 0.598 | 44838.53958 |
| DCIA | 40% | 6869.934697 | 0.606 | 42678.42636 |
| **Calculated Sequence** | LFA | 57% | 3415.153979 | 0.642 | 31836.42703 |
| CIA | 36% | 4652.527227 | 0.492 | 42801.8499 |
| DCIA | 41% | 4846.6745 | 0.484 | 40843.52101 |

sequences or the brute force generated sequence, we see that the average machine utilisation is not much different but the average flow time is better in the default sequence. if we look at the average lateness then there is a big difference between the default sequence and the brute force generated sequence whereas the critical job shifted sequence is not very far compared to the default sequence. Since the brute force sequence was created such that the lateness is as low as possible in the given constraints, the default sequence is not performing well to prevent the penalties of critical jobs even though it has lesser average flowtime.

If we look at the allocation methods, we see an expected result in case of average machine utilisation and average flowtime which is LFA > DCIA > CIA and LFA < DCIA < CIA respectively. But if we look at the average lateness, in case of default sequence the lateness for DCIA is less than LFA as expected but CIA is more than LFA which was unexpected. Also, in case of critical job first sequence there was an unexpected result as LFA had the least average lateness compared to other two. Although the average number of jobs which were late was always as expected which was LFA > CIA > DCIA.

Upon further investigation into why the average lateness was more in case of CIA for default sequence and for both CIA and DCIA on case of critical jobs first sequence compared to LFA, it was found that there are two reasons for this. First, the machines which was lowest in CI among the options for a non-critical job which was above a critical job would process that job but the same machine had the highest CI among the options for that critical machine therefore the critical machine which should have been processed before the non-critical one end up being processed after it and many time breach its due date. If this was not the case then the average lateness for CIA and DCIA would have been even better in the default sequence. The second reason was that in case of LFA the jobs lateness is more spread among the critical jobs, in other words there are more jobs which are late but their lateness is less. In a was this is not good as more jobs getting late is worse compared to fewer jobs getting a little later.

**Conclusion**

Overall, we can say that the best combination depends on the manufacturer but two best combinations which we think are the critical job shifted LFA and critical job shifted DCIA. Among them the LFA is best on all the performance metrics but the number of jobs breaching due date is significantly lesser in DCIA which could be a deal breaker for some manufacturers.