

A Study of Scheduling in a Mold Manufacturing Cell

P.Y. Gan, K.S. Lee, Y.F. Zhang and W.B.H. Toh

Department of Mechanical and Production Engineering
National University of Singapore
10 Kent Ridge Crescent
Singapore 119260

Abstract

As modern manufacturing plants tend to group their machinery in manufacturing cells for greater efficiency, a good scheduling technique within each cell is very critical in ensuring the smooth operation of the entire plant. With this in mind, this study is focused on scheduling jobs in a mold manufacturing cell. The present method involves many problems of coordination between departments and schedules used do not consider the optimality of the manufacturing cell. With the help of proper scheduling management, it is hoped that these problems will be eliminated and more efficient schedules are obtained. The goal of scheduling is often makespan or flow-time. However, in the manufacturing cell industries, priority weighted earliness with tardiness penalty might be more suitable in generating usable solutions. The branch and bound technique is used because of its enumerative and flexible nature. The algorithm was tested on real industrial manufacturing cell data and very promising schedules have been obtained.

Keywords: Manufacturing cell, branch and bound, priority

1 Introduction

Scheduling is a topic that has gained much attention in recent years. This is mostly caused by increasing complexities in manufacturing and also more emphasis on time and cost savings. The need for good schedules and the difficulty in obtaining them is especially true in low to medium volume production plants. Since most of the manufactured parts are unique in these plants, the situation differs greatly every time and it is difficult to use a heuristic to handle the scheduling. Due to this difficulty, many of these plants have been plagued with many problems like high work in progress inventories, high lead times and high machine idle times.

The study is focused on scheduling in a mold manufacturing plant. The plant is divided into a series of manufacturing cells where a job is passed from one cell to the other. Each of these cells functions independently by taking all incoming jobs and needs to complete each job before a stipulated deadline. To simplify things, we seek to optimize the schedule within each mold manufacturing cell. Each cell houses a certain number of parallel machines of similar type and a job only needs to be processed by one machine to be completed. However, certain jobs can be processed on multiple machines with different processing time.

For this research, we seek to implement an algorithm using branch and bound techniques for real industrial scheduling use. We have also suggested and tested priority weighted earliness as a performance measure for use in a manufacturing cell. In the next section, we will briefly present some scheduling literature and the scheduling problem. Subsequently, the implemented branch and bound algorithm will be presented followed by the industrial scheduling results and discussion. Finally the conclusion and future direction of this on-going project will be presented.

2 Background

The methodology of scheduling can be broadly categorized into two main parts, namely one pass methods and iterative methods. A dispatching rule is an example of one pass methods and is a quick solution to immediate problems, [4]. However, dispatching rules are often myopic to the overall optimality of the problem and thus more work has been done on iterative methods.

There are many scheduling problems in the industry ranging from job shops, flow shops, etc. In the situation of a manufacturing cell, the problem can be described as a multiple machine scheduling problem, [8,11]. The branch and bound method is known to be apt at solving problems of this nature, and some work has been done in similar fields by Jeng et al. [5,6]. To

better understand the branch and bound method, one can refer to the A* algorithm, [7,10].

Scheduling is mainly about optimization of a certain criteria called the performance measure. Performance measure can come in many forms. The more popular and widely researched performance measures are makespan and flow time. Though these are generally good objective functions, they cannot be applied to all situations and manufacturing environments. In the case of a manufacturing cell problem, to minimize flow time, one can simply schedule using the shortest processing time rule (SPT), for example in [1]. However, in the mold manufacturing cell environment, new jobs are constantly entering and if SPT is adopted, jobs with long processing time will never be scheduled. To ensure a more realistic schedule, other measures like weighted flow time using tabu search [2] or weighted earliness and tardiness using heuristics [3] have been used. For this research, a performance measure of priority weighted earliness with tardiness penalty is suggested and implemented.

3 The Scheduling Problem

In the mold manufacturing cell, jobs are constantly entering the cell and scheduling needs to be applied whenever a new job enters the cell. Each job that enters the manufacturing cell is assigned with a deadline and a priority. A low priority job having a tight deadline means that job should preferably finish before that deadline but should give way for higher priority jobs should their deadlines clash. In a manufacturing cell, a small job can be placed on bigger machines while bigger jobs are unable to be processed on a smaller machine. However, when a small job is being placed on a bigger machine, the process time will sometimes be greater than if it were to be done on a smaller machine. In this research the scheduling problem of the manufacturing cell will be solved with the following assumptions.

1. The jobs are independent and only needs to be processed once.
2. Each machine can only process one job at a time.
3. The jobs are non-preemptive.
4. All process times include the loading and unloading times.
5. Waiting times for cranes or forklifts are zero as they are assumed to be always available.
6. Workers are available to load and unload at all times and they perform the same by having no effect on the loading and unloading times.

Now let,

| | |
|----------|--|
| p_{ij} | processing time of job i on machine j |
| d_i | deadline of job i |
| P_i | priority of job i |
| s_{ji} | start time of job i on machine j |
| c_{ji} | completion time of job i on machine j |
| $f(S)$ | objective function based on the schedule S |
| Ea_i | Earliness of job i |
| Ta_i | Tardiness of job i |

The scheduling problem faced in a manufacturing cell can be taken as a multiple parallel machine problem. Each job only needs to be processed once by a particular machine. However, most jobs can be processed by more than one machine but at different process times. So for all n jobs, each job i has a process time p_{ij} on machine j. The aim is to create a schedule, S of jobs to be processed on each machine such that, the earliness is maximized. Such problem is proven to be NP hard [3].

A job can either be early or tardy depending on its completion time. Early jobs finish before their deadlines and the opposite applied for tardy jobs. Earliness or tardiness can be calculated by,

$$Ea_i = d_i - c_{ji} \quad , \text{where } c_{ji} = p_{ij} + s_{ji} \quad [1]$$

$$Ta_i = (c_{ji} - d_i) \alpha \quad [2]$$

α is a penalty assigned for jobs finishing late

The objective function is then

$$f(S) = \sum_{\text{all early jobs}} Ea_i - \sum_{\text{all tardy jobs}} Ta_i \quad [3]$$

4 The Branch and Bound Algorithm

The branch and bound technique is chosen to perform the scheduling due to its robust nature and its enumeration prowess. Although the computation load for this technique is usually high, good lower bound calculations and early good upper bound attainment would ensure that the search space can be contained to a reasonable size. The research algorithm is based on classic branch and bound techniques. To better understand the nature of the technique, the A* algorithm is used to explain the final implemented algorithm.

4.1 The A* algorithm

A problem starts with an initial situation S_0 where no job is scheduled and progresses to schedule one job at a time, branching out to each possibility or intermediate schedule S_i . The aim is to reach a final schedule $S_{k(\text{final})}$ where all the jobs are scheduled. The objective function $f(S_i)$ is assigned to each situation and is defined as

$$f(S_i) = g(S_i) + h(S_i) \quad [4]$$

where $g(S_i)$ is the cost incurred to reach S_i and $h(S_i)$ is a heuristic function that calculates the estimated cost of reaching the final schedule. If $f^*(S_i)$ is the true minimum cost of reaching the solution from S , $h(S_i)$ is a lower bound for $h^*(S_i)$. Since $h^*(S_i)$ must be positive,

$$0 \leq h(S_i) \leq h^*(S_i) \quad [5]$$

If $g^*(S_i)$ is the least cost of reaching S_i from the initial S_0 , $f^*(S_i) = g^*(S_i) + h^*(S_i)$ is the least cost solution that passes through S_i while $f(S_i)$ is a conservative estimate of $f^*(S_i)$ and a lower bound. The algorithm will keep seeking the node with lowest $f(S_i)$ to branch until the final goal is achieved.

4.2 The implemented branch and bound algorithm

The implemented algorithm is similar in principle to the A* algorithm. It consists of a heuristic to calculate an upper bound value. The upper bound is compared with each successive lower bound and will discard worse lower bounds to keep the search space small. The algorithm can be briefly summarized as follows:

Begin

Step 1. $S_0 \leftarrow$ Initial situation (No jobs scheduled)
Open $\leftarrow S_0$

While Open $\neq \emptyset$

Step 2. Choose in Open, a node with situation S_i such that $f(S_i)$ is minimum.

Open \leftarrow Open - S_i

Step 3. Sequence all the possibilities starting from S_i and for each possibility k , use a heuristic to schedule the remaining jobs $S_{k(\text{final})}$. Since $h(S_{k(\text{final})}) = 0$, the objective function can then be calculated as:

$$f(S_{k(\text{final})}) = g(S_{k(\text{final})})$$

The upper bound solution is the best known solution so far.

if $f(S_{k(\text{final})})$ is better,

then upper bound = $f(S_{k(\text{final})})$

Step 4. For each possibility k , calculate the lower bound:

$$f(S_k) = g(S_k) + h(S_k)$$

Where $h(S_k)$ is the best possible solution from S_i

Step 5. **if** $f(S_k)$ better than upper bound

then put the node S_k into Open

Step 6. Discard all nodes in Open with $f(S)$ worse than or equal to upper bound

End While

End

4.2.1 The largest job heuristic

When the next possibility is sequenced, the schedule becomes S_k . There is a need to compute a good solution from S_k so that a good upper bound solution can be obtained. If a good upper bound value is found very early, it helps to reduce search space by omitting many unpromising nodes.

The heuristic is based on how scheduling is done manually. Whenever a machine becomes free, the next loaded job will be the largest sized job the machine can hold because smaller jobs can always be done when a smaller machine is freed up but not the reverse. It is found that the heuristic based on this rule of thumb, obtains good early upper bounds which helps in containing the search space.

4.2.2 Upper bound calculation

When a schedule $S_{k(\text{final})}$ is obtained, each job i has a certain start time s_{ji} on a certain machine j , and a completion time c_{ji} . The objective function is a composite index of earliness, tardiness and job priority. To include the effects of job priority, the calculation of earliness and tardiness can be modified to

$$Ea_i = (d_i - c_{ji})P_i \quad [6]$$

$$Ta_i = (c_{ji} - d_i)P_i \alpha \quad [7]$$

Where α is a constant for tardiness penalty ranging about 3 to 5.

The upper bound can thus be calculated as

$$g(S_{k(\text{final})}) = \sum_{\text{all early jobs}} Ea_i - \sum_{\text{all tardy jobs}} Ta_i \quad [8]$$

4.2.3 Lower bound calculation

The lower bound value is an estimate of the best possible solution that can arise from what have already been sequenced in S_k . As the best solution $h^*(S_k)$ is impossible to obtain unless one lists out all the possibilities, one can only give a good estimate by calculating the process times of the remaining jobs.

$$h(S_i) = \sum_{\text{for all } i \text{ remaining jobs}} [\max(Ea_i) \text{ or } \min(Ta_i)] \quad [9]$$

A good lower bound is important as $h(S_i)$ should be as close to $h^*(S_i)$ as possible to allow $f(S_i)$ to give a good estimate on the potential of the node.

5 Results and Discussion

The algorithm was programmed in C++ and was implemented on a Pentium III 450 computer. The aim of the experiment is to study the computational behavior and effectiveness of the algorithm. In the first experiment, we vary the parameter of the implemented algorithm and observe the corresponding effect on the results obtained. In the other experiment, the algorithm is tested and contrasted in the mold manufacturing cell for a more thorough evaluation.

5.1 The value of α

The main parameter in the algorithm is the tardy job penalty α . The greater the value of α , the more it penalizes tardy jobs. The value of α is varied from 2 to 8 in a series of runs on several problems. The problem presented here is an 11 machines 20 jobs problem. The number of jobs that are early or late are recorded in each run to see the effects α has on the quality and nature of the solution. The attributes for comparison are number of early jobs, mean earliness (ME), mean tardiness (MT) and overall mean earliness (OME) which are calculated as such,

$$ME = [\sum_{\text{for all early jobs}}(d_{ji} - c_{ji})]/n_{\text{early jobs}} \quad [10]$$

$$MT = [\sum_{\text{for all tardy jobs}}(c_{ji} - d_{ji})]/n_{\text{tardy jobs}} \quad [11]$$

$$OME = [\sum_{\text{for all jobs}}(d_{ji} - c_{ji})]/n_{\text{total}} \quad [12]$$

Table 1. Earliness and Tardiness with varies α values

| α | ME | MT | OME | Early jobs |
|----------|------|------|-------|------------|
| 2 | 3.29 | 4.00 | -1.45 | 7 |
| 3 | 2.63 | 4.25 | -1.5 | 8 |
| 4 | 1.67 | 5.38 | -1.15 | 12 |
| 5 | 1.67 | 5.38 | -1.15 | 12 |
| 6 | 1.73 | 4.56 | -1.1 | 11 |
| 7 | 1.73 | 4.56 | -1.1 | 11 |
| 8 | 1.73 | 4.56 | -1.1 | 11 |

From Table 1, it is found that generally, a greater value of α , yields less tardy jobs. Besides that, an increase in α also reduces mean earliness, mean tardiness and combined earliness and tardiness. In most of the experiments, α values greater than 7 do not have any changes to the solutions. Smaller problems are found to be more insensitive to the α value.

In the next experiment, the aim is to test the effect α has on how the solution converges. The algorithm is again tested on a variety of problems and the showcased problem is the same earlier 11 machines 20 jobs problem. A plot on how fast the upper bound value evolve with different values of α is shown in figure 1.

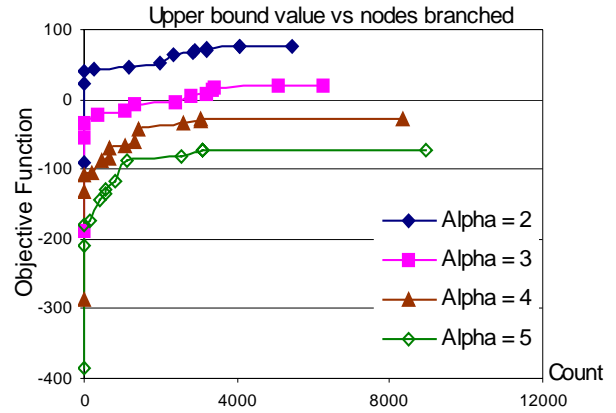


Figure 1. A plot of evolution of upper bound values with different α values

From figure 1, it can be seen that a larger α causes the solution to converge faster. The other observation is that greater α causes more nodes to be considered and hence greater computation times. From figure 1, when $\alpha=2$ the program terminates in about 6800 nodes and when $\alpha=5$, the program ends after 9000 nodes. The difference in the number of nodes increases as the problems becomes greater.

The findings indicate that the choice of α will actually change the type of solutions desired. If moderate lateness is tolerated, α can be reduced to a lower value to allow higher priority jobs to be completed earlier and faster computation times. The parameter α allows a little free play in determining which type of solution is optimal.

5.2 Application of the branch and bound algorithm

The branch and bound algorithm was tested in the mold manufacturing cell which consists of six CNC machines. At any one time there are usually about 10 to 20 jobs queuing in different machines and the schedule is determined manually by the cell supervisor. The algorithm was tested alongside with the actual schedule for several days and the effectiveness of it is compared against the actual schedule. The main objective is to obtain schedules which maximizes job earliness and to make sure that high priority jobs do not finish too late. The basis for comparison is thus overall jobs' mean earliness and priority weighted earliness.

For that particular manufacturing plant, the jobs are passed from one department to another and job deadlines determined by the cell supervisor. As the deadlines set are quite conservative and a late job can be passed to the next cell with a higher priority, we set the α value as 3. In the manufacturing cell, new jobs are constantly entering the cell and processed jobs are usually completed earlier or later than the estimated time and not exactly on time. Due to this

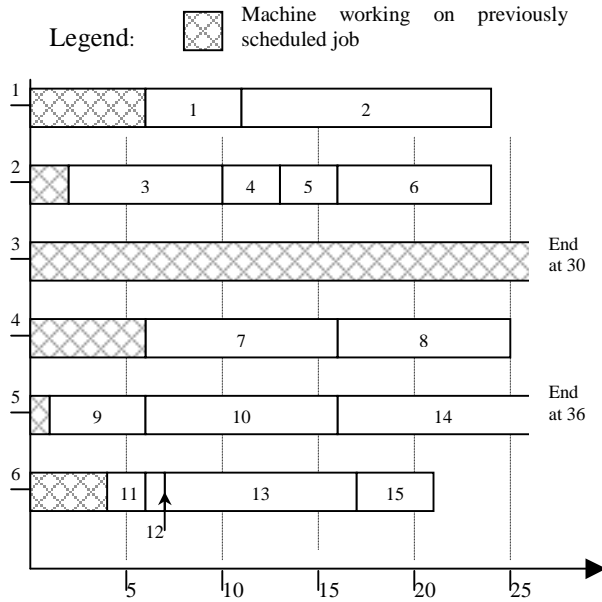


Figure 2. Gantt chart of the actual schedule from a mold manufacturing cell for all jobs starting before $t=20$ hrs

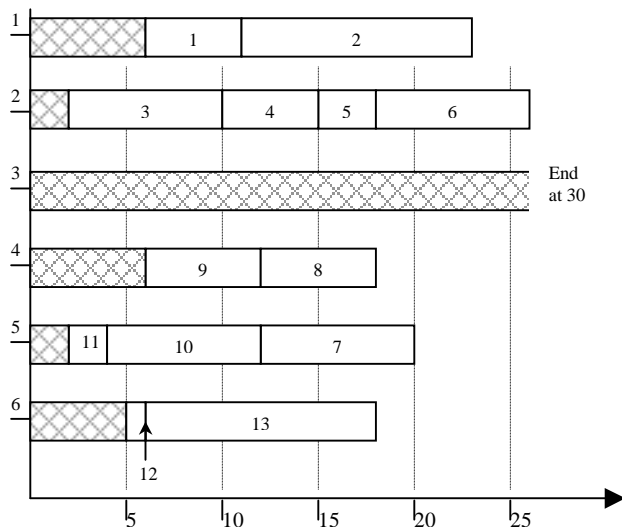


Figure 3. Gantt chart of the schedule obtained from the algorithm based on estimated times at $t=0$

dynamic environment, the optimum situation will constantly be

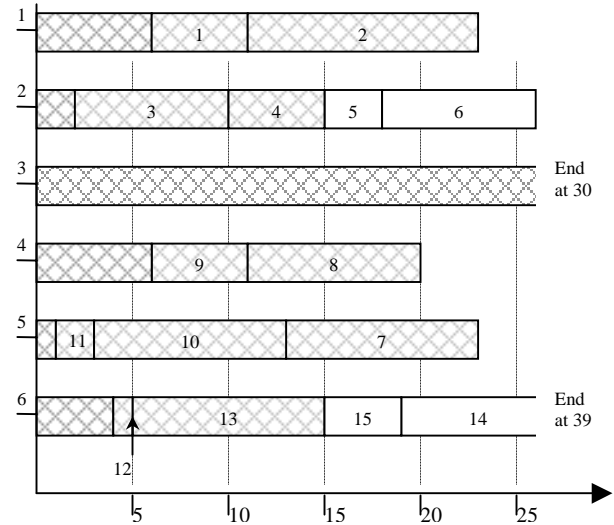


Figure 4. Gantt chart of the schedule obtained from the algorithm based on estimated times at $t=10$

changed. To ensure that the schedule keeps up with the changes, the algorithm is ran every 10 hours and the jobs previously scheduled for the next 5 hours will not be changed. For this case study, we consider only jobs starting before $t=20$ and the time is rounded off to the nearest hour for simplicity. The details on the deadlines and priorities for jobs 1 to 15 are summarised in table 2.

Table 2. Deadlines and priorities for jobs 1 to 15

| | | | | | | | | |
|-------|----|----|----|----|----|----|----|----|
| Job | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| d_i | 20 | 25 | 12 | 15 | 20 | 25 | 22 | 20 |
| P_i | 10 | 7 | 8 | 8 | 5 | 5 | 5 | 7 |
| Job | 9 | 10 | 11 | 12 | 13 | 14 | 15 | |
| d_i | 5 | 10 | 7 | 5 | 7 | 8 | 10 | |
| P_i | 13 | 15 | 5 | 20 | 15 | 38 | 22 | |

Figures 2 and 3 show the Gantt charts of the schedules from the actual manufacturing cell and from the algorithm at $t=0$ respectively. Figure 3 shows the Gantt chart of the schedule when the algorithm is ran again after 10 hours taking in new jobs. When the algorithm is run at this point of time, some of the previous time estimates are altered to represent more accuracy.

The performance of the algorithm is compared against the actual schedule by comparing the job's overall mean earliness (OME) and priority weighted earliness. As the algorithm uses estimates to work out the process times, the comparison will convert estimated time to the actual processed time before earliness is calculated. For a more objective view of the results, earliest due-date (EDD) dispatching rule is also used in the comparison.

Table 3. The improvement of using branch and bound algorithm

| | OME | Priority Weighted Earliness |
|-------------------------------|------|-----------------------------|
| Actual Schedule of cell | 2.47 | 72 |
| EDD Dispatching rule | 2.6 | 231 |
| B&B algorithm, ($\alpha=3$) | 3.2 | 299 |

From Table 3, it can be seen that the quality of the schedules obtained using branch and bound algorithm is significantly better than the original schedule and a schedule found by EDD. It is thus clear that the algorithm can obtain better schedules for the manufacturing cell. In practical usage, the algorithm can be manually triggered to run whenever a new job enters the cell and frequency of running the program can also be altered to suit the nature of the manufacturing cell. The time unit should also be changed to every minute or ten minutes depending on the situation.

6 Conclusions

The implemented branch and bound algorithm has shown itself to be practical for scheduling in a mold manufacturing cell. Usually, the improvements are more significant in larger problems where the deviation of the actual schedule from optimality is more severe. The superior schedules obtained from the branch and bound algorithm will ensure improved productivity and job earliness. With better schedules and deadline, coordination between manufacturing cells is made easier as completion times can be more accurately approximated and provisions can be made in the other cells.

The suggested performance measure of priority weighted earliness or tardiness has provided a means of judging the effectiveness of a schedule. It is shown that good and practical schedules have been obtained by using this performance measure. The branch and bound algorithm can also be used to optimize other performance measures in other areas of scheduling or even process planning. The main difference is in altering the calculation of the objective function and the lower bound.

The computation load for a typical branch and bound algorithm is inherently high and the implemented algorithm has striven to reduce search space by having good early upper bounds and tight calculations of lower bounds. The resulting search space and computation times are very reasonable for problems with sizes typical in the mold manufacturing cell.

Computation times for a 6 machines 20 jobs problem rarely exceeded 10 seconds.

7 Future Directions

In this research, all jobs are assumed to have arrived and waiting in the queues. It will be more practical to incorporate arrival times for jobs that are due to arrive later. It is hoped that by doing this, the schedule will make provisions to wait for arriving jobs which are high in priority. The inclusion of arrival time for new jobs will introduce a host of new considerations.

1. There are very few cases of preemption of jobs in the manufacturing cell. Although not allowed in this research, it would be more practical to make provisions for this when new jobs are entering the cell.
2. Presently, as there are no arriving jobs, waiting is not required and a job is scheduled the moment a machine is free. When there are arriving jobs, job waiting must be incorporated in the algorithm for machines to remain idle to wait for a more suitable job to arrive.

Scheduling within a single manufacturing cell is only focused on the optimality of that particular cell. It is hoped that eventually, this algorithm will be incorporated into solving the schedule of the whole plant, across multiple manufacturing cells.

References

1. Baker KR. "Introduction to sequencing and scheduling." Wiley, New York (1974)
2. Barnes J.W. and M. Laguna, "Solving the multiple-machine weighted flow time problem using tabu search," IIE transaction, vol. 25, n2, pp121-128 (1993).
3. Biskup D. and T.C. Cheng, "Multiple machine scheduling with earliness, tardiness and completion time penalties," Computers and Operations Research, vol 26, pp 45-57 (1999).
4. Hoiomt D.J, P.B. Luh and Krishna R. Pattipati, "A practical approach to Job-Shop Scheduling problems," IEEE transaction on robotics and automation, vol. 9, n1, pp1-13 (1993).
5. Jeng W.D. and James T. Lin, "A Branch and Bound algorithm for scheduling problem of a robot centered manufacturing cell," Journal of the Chinese Institute of Engineers, vol.16, n3, pp431-440 (1993).
6. Jeng W.D. and James T. Lin, "Algorithms for sequencing robot centered parallel processor

- workcell,” *Computers and Operations Research*, vol.20, n2, pp185-197 (1993).
7. Laurilere J.L., “Problem solving and artificial intelligence,” New York, Prentice Hall (1990).
 8. Lee H.F., “Production planning for flexible manufacturing systems with multiple machine types: a practical approach”, *International Journal of Prod. Research*, vol.36, n10, pp 2911-2927 (1998).
 9. Luh P.B. and D.J Hoitomt, “Scheduling of Manufacturing Systems Using the Lagrangian Relaxation Technique,” *IEEE transaction on automatic control*, vol. 38, n7, pp1-13 (1993).
 10. Nilsson J., “Problems solving methods in artificial intelligence,” New York, McGraw Hill (1971).
 11. Zhou C., P. Egbelu, “Scheduling in a manufacturing shop with sequence dependent setups”, *Robotics and Computer Intergrated Manufacturing*, vol. 5 n1, pp 73-81 (1989).