IOSR Journal of Mechanical and Civil Engineering (IOSR-JMCE)
e-ISSN: 2278-1684,p-ISSN: 2320-334X, Volume 12, Issue 1 Ver. I (Jan- Feb. 2015), PP 37-46
www.iosrjournals.org

A Model for Optimum Scheduling Of Non-Resumable Jobs on Parallel Production Machines with Multiple Availability Constraints.

[1] Apalowo, R. K*. and [2] Ogedengbe, T. I.
Mechanical Engineering Department, School of Engineering and Engineering Technology, Federal University of Technology. PMB 704 Akure, Nigeria.

Abstract: *In this study, parallel machines makespan minimization problem, where the jobs are non-resumable and the machines are subject to unavailability periods of known starting times and durations, was considered. The problem was formulated as an Integer Linear Programming mathematical model. An efficient solution procedure was proposed for the problem through two algorithms. The first algorithm utilizes the Longest Processing Time dispatching rule which generates an initial solution (makespan) used to obtain an upper bound which is used to backtrack through the second algorithm. The second algorithm adopted the greedy algorithm approach to iteratively achieve an optimal solution for the scheduling problem. The developed algorithms were implemented in software using Visual Basic programming language to facilitate the solution procedure. The developed software was validated using a typical numerical example while the model was validated using two industrial problems as case studies. The makespans obtained for the first and second case studies using the developed model are respectively about 49% and 40% lower than the makespan when the jobs were carried out in the industry. This showed that, in the long run, the model would efficiently serve machine shops in machines-jobs scheduling problem.*

Keywords: *Linear Programming, Scheduling, Makespan, Algorithm, Software*

I. Introduction

In machine-shop environments, machines are usually subjected to down periods, which may be deterministic (as in the case of a planned preventive maintenance) or stochastic (as in the case of a sudden machine breakdown or unexpected material shortage). In general, intelligent scheduling methods are needed to assign activities to mechanical machines when faced with limited execution time and scarce resources. Many researchers have worked on the topic, especially in the area of machine scheduling (Lee, 1991; Brucker, 1994; Chen and Lee, 1999; Lee, 1996). However, most assume that the processors or machines are always available over the course of the production horizon (Brucker, 1994; Chen and Lee, 1999). This assumption is not realistic in many practical situations. The operation of a machine can be interrupted for a certain period of time due to accidental breakdown, preventive maintenance, periodic repair or other reasons, which render the machine non-productive for a certain period of time. This situation is referred to as the machine availability constraint and it is essential that it is considered to appropriately schedule mechanical machines for relevant activities.

Parallel machines scheduling with machines availability constraint has attracted a great interest, especially in the aspects of makespan and total completion time minimization problems. Such problems have been solved optimally by priority rules like LPT (Lee, 1991), List Scheduling (LS) (Lee, 1996) and approximation scheme (Lenstra et al., 1990), and in solving the problem, the online scheduling or resumability constraint were rarely considered, due to the complexity of the problem (Tan and Yong, 2002). Moreover, the case where the machines are related has received more attention than when they are unrelated (Vallade and Ruiz, 2011).

In the aspect of related parallel machines scheduling, Liao et al. (2005) studied two machines makespan minimization problem with availability constraint only on one of the machines. The problem was solved by partitioning it into four sub-problems and each sub-problem is solved optimally by using versions of a lexicographical search algorithm originally proposed in Ho and Wong (1995). Tan and Yong (2002) also studied the same problem with non-resumable jobs when the period of unavailability occurs on both machines at different period of time. Online algorithm was used in jobs assignment. Lee (1991) studied parallel machines makespan minimization problem with one unavailability period on all machines, except one, which is always available, and the unavailability periods are assumed to start at time zero. The study also considered unavailability periods as already scheduled jobs on the machines and are called special jobs. Modified Longest Processing Time (MLPT) dispatching algorithm was used to assign jobs on the machines with the condition that each machine cannot have more than one special job (unavailability period). The same parallel machines problem was solved in Lin et al. (1997) and Kellereer (1998) using MLPT and dual approximation algorithm

respectively. However, Lee (1996) studied parallel machines scheduling problem with unavailability periods that may not necessarily start at time zero. It also assumed that one of the machines is available and proposed the LPT2 algorithm to assign jobs on the machines.

Moreover, in the aspect of unrelated parallel machines scheduling problem, Liao and Sheen (2008) studied unrelated parallel machines scheduling problem with availability constraint and solved the problem using a binary search algorithm. Blazewicz et al. (2000) also studied uniform and unrelated parallel machine problems with arbitrary patterns of unavailability with the makespan and tardiness minimization objectives. The study proposed a network flow approach for the uniform, and a linear programming approach for the unrelated machine problem. The unrelated parallel machines problem was also investigated in Yang et al. (2012) with aging effect and multi-maintenance activities considered simultaneously as availability constraints. The study proposed two efficient algorithms to optimally solve the problem when the maintenance frequencies on the machines are known. Chang et al. (2011) studied the same problem with deteriorating maintenance activities, with the objective of minimizing the total completion time or the total machine load. The study also showed that both versions of the problem considered in Yang et al. (2012) and Chang et al. (2011) can be solved with the same complexity irrespective of whether the processing time of a job after the unavailability period is greater than that before the unavailability period. Furthermore, Yang (2013) considered the problem with special availability constraint in form of simultaneous deteriorating effect and deteriorating multi-maintenance activities, with the objective of determining optimal maintenance frequencies and positions and optimal job sequences of minimized total completion time.

However, consequent upon the studies already discussed, it is essential to consider the situation in which there may be multiple unavailability periods on each machine during scheduling horizon, and with non-resumable jobs. This research therefore focuses on optimizing parallel machines scheduling problem with non-resumable jobs by considering multi-fixed machine availability constraint with the objective of minimizing the machines makespan.

II. Materials And Method

The method used involves the formulation of the problem as a mathematical model; development of efficient solution procedure; implementation of the model with its solution procedure through software; validation, and evaluation of the developed software and the model respectively. These are presented in sections 2.1 to 2.2.6.

2.1 Mathematical Formulation of the Problem

The problem is formulated as an Integer Linear Programming Mathematical (ILP) model. It is assumed that all machines can perform each operation at the same rate; each job should be processed on exactly one machine; all jobs and machines are available at time zero; processing time of each job is known in advance i.e. it is deterministic; starting time and the duration of the unavailability period of each machine are also deterministic. The indices, parameters and decision variables that are used in developing the mathematical model as well as the solution procedure are defined as thus:

Indices

i	machine index	i = 1,2,....., m
N	job index	N=1,2,....., N
c, k	unavailability indices	c = 1,....., k; k = 1,2,....., μ_i
j	job type index	j = 1,2, ... n

Parameters

p_j	processing time of job j
N	number of jobs
n	number of job types
m	number of machines
s_{ik}	starting time of k^{th} unavailability period on machine i
e_{ik}	ending time of k^{th} unavailability period on machine i
d_{ik}	duration of the k^{th} unavailability period on machine i
M	a large positive integer
μ_i	number of unavailability period(s) on machine i
v_j	the number of jobs having processing time of p_j, yet to be scheduled
u_j	number of jobs with processing time p_j which the load of the current machine can optimally accommodate
z_{bj}	number of jobs with processing time p_j in the load of b^{th} batch of the current machine
q_i	number of batches in machine i

t_i idle time of machine i, which is the space of time within a batch without jobs, which is insufficient to complete any given job.
d_i total period of time for which a machine is unavailable over the scheduling horizon
$\propto$ number of machines (including the current machine) on which jobs have already been scheduled
T_i decision parameter during assignment of jobs to machines
M_i total load of a machine (including load of jobs, unavailability and idle periods)
L_i load of the jobs scheduled on a machine (i.e. load of jobs only)
L_{imax} makespan without unavailability and idle periods
i^{**} index of machine from which a new iteration starts
l iteration counter
r Machine feasible load counter. Increases by one when a new feasible load is achieved on a machine
imax index of machine with load L_{imax}

Decision variables

$x_{ij} = \begin{cases} 1, & if\ job\ \text{j is scheduled on machine i} \\ 0, & otherwise \end{cases}$

$y_{ik} = \begin{cases} 1, & if\ all\ jobs\ scheduled\ on\ machine\ \text{i have been completed before } s_{ik} \\ 0, & otherwise \end{cases}$

h makespan

The ILP model formulated is as given in equations 1-9.

Min h (1)

Subject to

$\sum_{j=1}^{N} p_j x_{ij} + \sum_{k=1}^{\mu_i - 1}(\sum_{c=1}^{k}(d_{ic} + t_{ic})y_{i(k+1)} + \sum_{k=1}^{\mu_i}(d_{ik} + t_{ik})(1 - (\sum_{k=1}^{\mu_i} y_{ik})) \leq \sum_{k=1}^{\mu_i} s_{ik} y_{ik} + M(1 -$
$(k=1\mu iyik))$ $\forall i$ (2)

$\sum_{j=1}^{N} p_j x_{ij} + \sum_{k=1}^{\mu_i - 1}(\sum_{c=1}^{k}(d_{ic} + t_{ic})y_{i(k+1)} + \sum_{k=1}^{\mu_i}(d_{ik} + t_{ik})(1 - (\sum_{k=1}^{\mu_i} y_{ik})) \leq h$ $\forall i$ (3)

$\sum_{i=1}^{m} x_{ij} = 1$ $\forall j$ (4)

$\sum_{k=1}^{\mu_i} y_{ik} \leq 1$ $\forall i$ (5)

$x_{ij} \in \{0, 1\}$ $\forall i, \forall j$ (6)

$y_{ik} \in \{0, 1\}$ $\forall i$ (7)

$h \geq 0$ (8)

The objective function (eq. 1) is the makespan to minimize. Constraint set (eq. 2) ensures that the duration of unavailability period(s) is/are only added to the total processing times of jobs assigned to machine i, when the starting time of the unavailability period(s) is/are before the completion of the jobs. Constraint set (eq. 3) ensures that the makespan of each machine, when unavailability period is considered, is at most as large as the makespan. Constraint set (eq. 4) ensures that each job is assigned to exactly one machine. Constraint set (eq. 5) ensures that no more than one y_{ik} is equal to one for each job. Constraint set (eq. 6) ensures that no more than one z_{ijk} is equal to one for each job and machine. Constraint sets (eq. 7) and (eq. 8) show that x_{ij} and y_{ik} are binary variables (i.e. can either be 0 or 1). Constraint set (eq. 9) is a non-negativity constraint.

2.2 Solution Procedure

Longest Processing Time (LPT) rule for job dispatching is known to perform better for minimizing makespan (Lee, 1996; Baker and Trietsch, 2009). The LPT dispatching rule is first applied to develop a suitable solution procedure for the problem. Then, the solution of the LPT-based algorithm is used as the initial feasible solution to iteratively backtrack better feasible solutions in the main algorithm, which is formulated using the greedy algorithm approach, until the optimal solution is obtained.

At any point in time, assume that the current feasible solution (makespan) is h, a better solution, h^*, is expected to be less than h by at least one unit, in the case of an integer processing time of jobs.

Hence, an upper bound of a feasible makespan is defined as thus:

$UB = h - 1$ (9)

2.2.1 Loading of the Machines

Jobs are scheduled on a machine in batches such that each batch refers to the period of availability of the machine which is the period of time between the end of an unavailability period and the start of the next unavailability period.

Machines loads are determined in lexicographical order in which jobs are grouped into job types, according to the processing time of each job, in descending order and the number of each job type (except the first job type) scheduled on a machine depends on the loads of other job types already scheduled on the machine.
For the first job type:

$$u_1 = \min\left\{\left\lfloor\frac{L_{UB}}{p_1}\right\rfloor, v_1\right\} \quad (10)$$

For other job types:

$$u_j = \min\left\{\left\lfloor\frac{L_{UB} - \sum_{a=1}^{j-1} p_a u_a}{p_j}\right\rfloor, v_j\right\} \qquad j = 2,3,\ldots\ldots\ldots n \quad (11)$$

Where p_a and u_a are respectively the processing time and the number of job type(s) already loaded on the machine prior to loading the current job type on the machine.
The number of each job type scheduled in each batch of a machine is likewise determined as thus:
For the first batch:

$$z_{1j} = min\left\{\left\lfloor\frac{s_1 - \sum_{\theta=1}^{j-1} p_\theta z_{1\theta}}{p_j}\right\rfloor, u_j\right\} \quad \forall j \quad (12)$$

For other batches:

$$z_{bj} = min\left\{\left\lfloor\frac{(s_{ib} - e_{i(b-1)}) - \sum_{\theta=1}^{j-1} p_\theta z_{b\theta}}{p_j}\right\rfloor, \left(u_j - \sum_{\tau=1}^{b-1} z_{\tau j}\right)\right\} \quad b = 2,3,\ldots\ldots q;\ \forall j \quad (13)$$

Where:
p_θ and $z_{b\theta}$ are respectively the processing time and the number of job types already loaded on a batch prior to loading the current job type on the batch.
$z_{\tau j}$ is the number of jobs with processing time p_j already scheduled in the previous batches before the current batch.
The machine's makespan is then determined as thus:

$$h_i = L_i + d_i + t_i \quad (14)$$

Where:

$$L_i = \sum_{b=1}^{q_i} \sum_{j=1}^{n} p_j z_{bj} \quad (15)$$

$$d_i = \sum_{k=1}^{q_i - 1} d_{ik} \quad (16)$$

and

$$t_i = \sum_{b=1}^{q_i} (s_b - e_{b-1}) - \sum_{j}^{n} p_j z_{bj} \quad (17)$$

2.2.2 Feasibility Test of Machine Loading
As each machine is loaded, the feasibility of the load is tested using the following lemmas
(a) The makespan of the machine load must not be greater than the current upper bound
i.e $h_i \leq UB$ (18)
(b) At any point in time during loading of a particular machine, if the remaining average load is less than the current load upper bound L_{UB},
That is;
If

$$\frac{\sum_{j=1}^{N} p_j - \sum_{i=1}^{\propto} L_i}{m - \propto} \leq L_{UB} \quad (19)$$

is satisfied, then the load is feasible and the next machine is loaded. Else;
(c) Test will be carried out if there may still be a feasible solution for the current upper bound. This is done by assuming that the machines are loaded to their current upper bound load, L_{UB_i}. If remaining average load in this condition is less than L_{UB}, that is;
If

$$\frac{\sum_{j=1}^{N} p_j - \left(\sum_{i=1}^{\propto} L_{UB\,i}\right)}{m - \propto} \leq L_{UB} \quad (20)$$

then feasible solution may be possible. Otherwise, feasible solution is no more possible.
(d) Also, if the feasibility test conducted in (c) is feasible, there is need to determine whether the possible feasible solution is on the current machine. If the remaining average load, when the machines are loaded to their upper bound except the current machine, is less than L_{UB},
That is;
If

$$\frac{\sum_{j=1}^{N} p_j - (\sum_{i=1}^{\propto - 1} L_i + L_{UB\,\propto})}{m - \propto} \leq L_{UB} \quad (21)$$

then the current machine is reloaded. Else, the immediate previous machine is reloaded.
The machine reloading is done as thus:
where $\gamma = max\{j | u_j > 0 \ and \ j \neq n\}$

$$\bar{u}_j = u_j, \qquad j = 1, 2, \dots \dots \gamma - 1; \tag{22}$$

$$\bar{u}_j = u_j - 1, \qquad j = \gamma; \tag{23}$$

$$\bar{u}_j = min\left\{\left\lfloor \frac{L_{UB} - \sum_{a=1}^{j-1} p_a \bar{u}_a}{p_j} \right\rfloor, \ v_j\right\} \qquad j = \gamma + 1, \dots . n; \tag{24}$$

The job type with the highest number of jobs, in the load of the machine to be reloaded, is determined and denoted as γ. The number of each job type(s) before γ will be left as they were before reloaded. This is done using equation (22). The number of jobs in γ is decreased by one (equation 23). The job type(s) after γ are determined, in decreasing order of their processing time using equation (24), in the same manner as in equation (11).

2.2.3 Algorithm Development

The steps involved in the LPT-based algorithm, henceforth referred to as the initial solution algorithm and the main algorithm developed for the scheduling problem are presented as thus:

Initial Solution Algorithm

Step 1. Specify values for: m; n; $p_N (N = 1,2, \dots . N)$; $s_{ik}, d_{ik} (i = 1,2, \dots m; k = 1,2, \dots . \mu_i)$
Step 2. Arrange the N jobs in decreasing order of their processing time
Step 3. Set:

$$M_i = 0 \qquad (i = 1,2, \dots \dots \dots . , m)$$
$$j = 1$$

Step 4. Determine the job assignment decision variable, T_i

$$T_i = \begin{cases} p_N + s_{ik} + d_{ik} & M_i < s_{ik} < (M_i + p_N) \\ M_i + p_N & otherwise \end{cases}$$

Step 5. Assign job j to machine i, such that;

$$i = min(T_i: \qquad \forall i)$$

Step 6. Set:

$$\begin{cases} M_i = T_i & for \ i = min(T_i: \ \ step \ 5) \\ M_i = M_i & otherwise \end{cases}$$

and

$$j = j + 1$$

Step 7. If $j \leq N$, Go to 4

Step 8. $d_i = \sum_{k=1}^{q} d_{ik} \qquad (\forall i)$
$L_i = M_i - d_i, \qquad (\forall i)$
Step 9. $h = max(M_i: \qquad \forall i)$
$L_{imax} = max(L_i: \qquad \forall i)$

Main Algorithm

Step 1. Set: $l = 1; \qquad r = 0$
Step 2. Determine the upper bound of makespan and the upper bound of makespan load (i.e. makespan without unavailability and idle periods) using the solution obtained by the initial solution algorithm

$$UB = h - 1; \qquad L_{UB} = L_h - 1$$

Step 3. Determine i^{**} using

$$i^{**} = imax \quad \text{OR} \, i^{**} = 1 \ (if \ l = 1)$$

Step 4. If $i^{**} = m$, then $i = i^{**} - 1$
Else, $i = i^{**}$
Step 5. Load machine i using equations (10) to (13)
Step 6. Determine the values of $L_i, \ h_i, \ d_i, \ and \ t_i$ using equations (15), (14), (16) and (17) respectively.
Step 7. If $h_i \leq UB$, Go to 10,
Step 8. If $i = m$, set $i = i - 1$. Otherwise, set $i = i$
Step 9. If $\sum_{j=1}^{n-1} u_j > 0 \quad for \ machine \ i$, Go to 18. Else, Go to 21
Step 10. If $i = m$, Go to 20
Step 11. Determine the feasibility of the load using equation (19)
Step 12. If equation (19) is feasible, set:

$$i = i + 1; \qquad r = r + 1$$

Go to 5
Step 13. Else, if equation (19) is infeasible, determine, using equation (3.31), whether there may still be a feasible solution for the current upper bound.
Step 14. If equation (20) is infeasible, Go to 21
Step 15. Else, if equation (20) is feasible, check the feasibility of other possible loads on the current machine using equation (21).
Step 16. If equation (21) is feasible, Set $i = i$. Go to 18
Step 17. Else, if equation (21) is infeasible; set:
If $r > 0$, Set $i = i - 1$. Else, Go to 21
Step 18. Reload machine i using equations (22) to (24), then (12) to (13). Go to 6
Step 19. Determine the new feasible solution (makespan) as thus:
$h = max(h_i: \ i = 1,2, \ldots\ldots\ldots, m)$
$L_{imax} = max(L_i: \quad \forall i)$
Step 20. Set:
$l = l + 1; \qquad r = 0$
Go to 2
Step 21. Set:
$h = UB + 1$

2.2.4 Software Development

The solution procedure, demonstrated through the initial solution algorithm and the main algorithm, is implemented through a software that was developed using Microsoft Visual Basic Programming Language.

2.2.5 Software Validation

A typical numerical example was used to validate the developed software. This numerical example is solved manually using the proposed solution procedure demonstrated through the initial solution algorithm and the main algorithm. The same numerical example problem is then run on the developed software and the result obtained is compared against that of the manual solution.
A problem of ten jobs on three machines with availability constraints on each machine is used in this example. The processing times of the jobs are as in Table I.
Machine 1 is unavailable for 4 units of time at every 20 units of time after the end of last unavailability period.
Machine 2 is unavailable for 4 units of time at every 15 units of time after the end of last period of unavailability.
Machine 3 is unavailable for 4 units of time at every 10 units of time after the end of last unavailability period.
The parameters of the numerical example are:
$m = 3 \qquad n = 4 \qquad N = 10 \ d_{ik} = 4 \quad (\forall i, \forall k)$
$s_{11} = 20; \ s_{12} = 44; \ s_{13} = 68$ etc.
$s_{21} = 15; \ s_{22} = 34; \ s_{23} = 53$ etc.
$s_{31} = 10; \ s_{32} = 24; \ s_{33} = 38$ etc.
Solving through the numerical example, manually, using the solution procedure;
Optimal makespan, $h = 34$.
The solution schedule obtained by the manual calculation is presented in Table II.

2.2.6 Model Validation

Industrial data were collected, for validation of the developed model, at Dinehin Engineering Workshop situated at Ondo-Ore road, Akure Nigeria and Afolabi and Sons Engineering Limited, Akure, Nigeria. The workshops respectively specialize in grinding of crankshaft and boring of engine block of different vehicles. Data collected include the standard time, to bore an engine block and that to grind a crankshaft to various sizes. These data were obtained by direct observation (using a stop watch) during operations on the machines. Also, information on the preventive maintenance schedules of the machines was obtained.

The industrial data collected were run on the developed, and already validated, software for the purpose of validating the model. Two case studies were used. One case representing validation through the data collected on the grinding machines and the other case represents that of the boring machines. The data collected for the validation of the model, on the two cases, are as presented in Tables III to VI.

Case Study 1: Data on Crankshaft Grinding Machine

Twenty jobs are considered on three parallel crankshaft grinders. The details of the jobs and the unavailability periods of the machines are presented in Tables III and IV respectively.

Case Study 2: Data on Engine Block Boring Machine

Fifteen jobs are considered on three parallel engine block borers. The details of the jobs and the unavailability periods of the machines are presented in Tables V and VI respectively.

III. Results And Discussion

Model for scheduling non-resumable jobs on parallel machines subject to predictive unavailability constraints has been developed and its solution procedure has been implemented in computer software. Illustration 1 shows the input data interface of the software used for inputting jobs and machines details.

Validation of the developed software which was done using a typical numerical example showed that the software generates a makespan of 34 units of time (see Illustration 2) which is the same as the one obtained from manual calculation. The initial and optimal schedules obtained by the software and the manual calculation are as presented in Illustration 3 and Table II respectively.

In order to ascertain the potential of the developed model in production or process optimization, the model developed with its solution procedure is validated using the data collected for the two case studies. Case study one is a problem of twenty jobs on three crankshaft grinders. An optimal makespan of 160 mins is obtained by the model using the software against a makespan of 315 mins used in the company when the jobs were done. Case study two is a problem of fifteen jobs on three engine block borers. Optimal makespan of 187 mins is obtained by the model using the software while makespan of 310 mins was used when the jobs are carried in the company. Tables VII and VIII give the initial and optimal schedule obtained by the software for case studies 1 and 2 respectively.

Comparatively, the model developed produced a better (lower) makespan than what is being expended when the jobs are carried out in the company from which the data were obtained. Hence, the company can increase its level of productivity of completing jobs in a shorter period of time than its present scenario, if it can make use of the developed model and its software for jobs scheduling.

Moreover, it can also be said that the model developed has been effectively facilitated through efficient computer software. Through the software, a scheduling problem involving three machines and ten jobs was scheduled within a period of five to ten seconds, while the same problem took about two hours when solved manually using the developed solution procedure. In this sense, the developed software will save a great deal of time when used in an industrial setting.

IV. Conclusion

Although there have been various studies in the field of machine scheduling especially in recent years, there are still many unexplored problems. In this research, parallel machines makespan minimization problem, where the jobs are non-resumable and the machines are subjected to preventive maintenance activities of known starting times and durations, was considered.

The problem was formulated as an integer linear programming and an effective algorithm was developed to solve the problem. The algorithm developed is in two parts; the first part (initial solution algorithm) generates an initial solution which was employed in the second part (main algorithm) to iteratively generate better feasible solution. The second part is a backtracking algorithm which seeks for all the possible better feasible solution until the optimum solution is attained.

The model developed with its solution procedure (algorithm) was then implemented in flowcharts which were used to develop computer software. The computer software was validated using a numerical example. In validating the model developed, the validated software was used to solve two different real life industrial scheduling problems; grinding of twenty crankshafts of different vehicles on three crankshaft grinding machines and boring of fifteen engine blocks on three engine block boring machines, when the machines were subject to preventive maintenance activities of known starting times and durations. A better makespan were obtained by the software, in the two problems presented, than what were expended in the company where the jobs were carried out.

The use of the developed model and software will assist industries in time management, elimination of process delay, meeting jobs due dates towards production/process optimization and maintenance schedule management.

V. Tables And Illustrations

5.1 Tables

Table I: Numerical Example Job's Processing Times

j	1	2	3	4	5	6	7	8	9	10
P_j	10	10	9	9	7	7	6	6	6	6

Table II: Solution Schedules Obtained by the Manual Calculation

S/N	Job description		No. of jobs	Processing time of each job (mins)
	Vehicle	Grinding work		
1	Mazda	Standard to 010	4	17
2	Mazda	Standard to 040	6	13
3	Toyota	020 to 030	4	23
4	Bedford	030 to 040	4	37
5	Toyota	010 to 020	2	17

Table III: Jobs details for data collected on crankshaft grinders

INITIAL SOLUTION			OPTIMALSOLUTION		
Job	Processing time	Machine assigned to	Jobs	Processing time	Machine assigned to
1	10	1	1	10	1
2	10	2	2	10	1
3	9	3	3	9	2
4	9	1	4	9	1
5	7	3	5	7	2
6	7	2	6	7	2
7	6	1	7	6	3
8	6	2	8	6	3
9	6	3	9	6	3
10	6	1	10	6	2

Table IV: Machines details for data collected on crankshaft grinders

Machine	Maintenance schedule	
	Duration (mins)	Interval of Time till the next delay period (mins)
1	5	60
2	5	55
3	5	50

Table V: Jobs details for data collected on engine block borers

S/N	Job description		No. of jobs	Processing time of each job (mins)
	Vehicle	Grinding work		
1	Nissan	Standard to 020	3	27
2	J5	Standard to 020	6	22
3	Nissan	Standard to 010	3	26
4	Bedford	030 to 040	3	35

Table VI: Machines details for data collected on engine block borers

Machine	Maintenance schedule	
	Duration (mins)	Interval of Time till the next delay period (mins)
1	5	60
2	5	55
3	5	50

Table VII: Solution Schedules Obtained for Case Study 1

INITIAL SOLUTION			OPTIMAL SOLUTION		
Job	Processing time	Machine assigned to	Jobs	Processing time	Machine assigned to
1	37	1	1	37	2
2	37	2	2	37	2
3	37	3	3	37	1
4	37	3	4	37	1
5	23	1	5	23	2
6	23	1	6	23	1
7	23	2	7	23	1
8	23	1	8	23	1
9	17	2	9	17	3
10	17	1	10	17	3
11	17	3	11	17	3
12	17	2	12	17	2
13	17	3	13	17	2
14	17	1	14	17	2
15	13	2	15	13	3
16	13	3	16	13	3
17	13	1	17	13	3
18	13	2	18	13	3
19	13	1	19	13	3
20	13	3	20	13	3

Table VIII: Solution Schedules Obtained for Case Study 2

INITIAL SOLUTION			OPTIMAL SOLUTION		
Job	Processing time	Machine assigned to	Jobs	Processing time	Machine assigned to
1	35	1	1	35	1
2	35	2	2	35	2
3	35	3	3	35	3
4	27	3	4	27	3
5	27	2	5	27	2
6	27	1	6	27	1
7	26	2	7	26	2
8	26	1	8	26	1
9	26	3	9	26	3
10	22	2	10	22	2
11	22	1	11	22	1
12	22	3	12	22	3
13	22	2	13	22	2
14	22	1	14	22	1
15	22	3	15	22	3

5.2 Illustrations

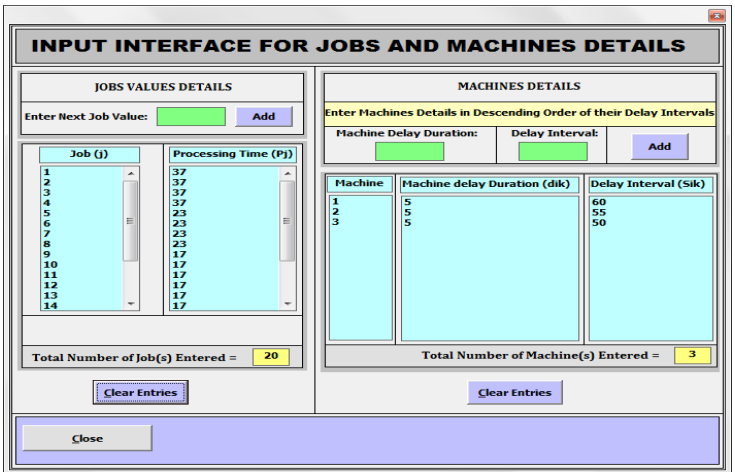

Illustration 1: Input Data Interface of the Software

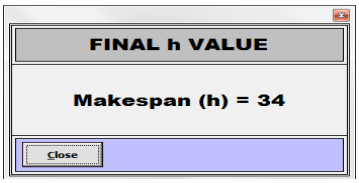

Illustration 2: Optimal Makespan for the Numerical Example

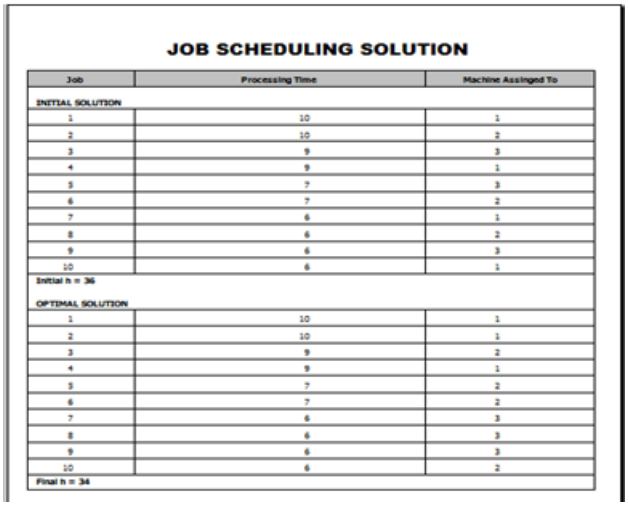

Illustration 3: Obtained Initial and Optimal Schedules for the Numerical Example

References

[1]. Baker, K.R. and Trietsch, D. (2009): Principles of Sequencing and Scheduling, Wiley, Hoboken, New Jersey.

[2]. Blazewicz, J., Drozdowski, M., Formanowicz, P., Kubiak, W. and Schmidt, G. (2000): Scheduling Pre-emptable Tasks on Parallel Processors with Limited Availability. Parallel Computing, 26:1195-1211.

[3]. Brucker, P. (1994): A polynomial algorithm for the two machine job-shop scheduling problem with a fixed number of jobs. Operations Research Spectrum, 16(1):5–7.

[4]. Chang, T.C.E., Hsu, C. and Yang, D. (2011): Unrelated parallel machines scheduling with deteriorating maintenance activities, Computers & Industrial Engineering Vol. 60, p. 602-605.

[5]. Chen, J. and Lee, C.Y. (1999): General multiprocessor task scheduling. Naval Research Logistics, 46(1):57–74.

[6]. Ho, J. C. and Wong, J. S. (1995): Makespan Minimization for m-parallel Identical Processors. Naval Research Logistics, 42:935-942.

[7]. Kellereer, H. (1998): Algorithms for Multiprocessor Scheduling with Machine Release Times, IIE Transactions, Vol. 30, pp.991-999.

[8]. Lee. C.Y. (1991): Parallel Machines Scheduling with Non-simultaneous Machine Available Time, Discrete Applied Mathematics, Vol. 30, pp.53-61.

[9]. Lee. C.Y. (1996): Machine Scheduling with an Availability Constraint, Journal of Global Optimization, Vol. 9. pp. 395-416.

[10]. Lenstra J. K., Shmoys, D. B. and Tardis E. (1990): Approximation algorithm for scheduling unrelated parallel machines. Springer. Vol 46, issue 1-3, pp 259-271.

[11]. Liao, L.W. and Sheen, G.J. (2008): Parallel Machine Scheduling with Machine Availability and Eligibility Constraints, European Journal of Operation Research, Vol. 184, pp.458-467.

[12]. Liao, C.J., Shyur, D.L., and Lin, C.H. (2005): Makespan Minimization for Two Parallel Machines with an Availability Constraint, European Journal of Operation Research, Vol. 160, pp.445-456.

[13]. Lin, G., He, Y., Yao, Y., and Lu, H. (1997): Exact bounds of the modified LPT Algorithms Applying to Parallel Machines Scheduling with Non-simultaneous Machine Available Times, Applied Mathematics- A Journal of Chinese Universities, Vol.12, pp.109-116

[14]. Tan, Z. and Yong, H. (2002): Optimal Online Algorithm for Scheduling on Two Identical Machines with Machine Availability Constraints, Information Processing Letters, Vol. 83, pp.323-329.

[15]. Vallade, E. and Ruiz, R. (2011): A genetic algorithm for the unrelated parallel machines scheduling problem with sequence dependent set up times. European Journal of Operations Research. Vol 211 (3). Pp 612-622.

[16]. Yang, S. (2013): Unrelated parallel-machine scheduling with deterioration effects and deteriorating multi-maintenance activities for minimizing the total completion time, Applied Mathematical Modelling, vol. 37-5, pp. 2995-3005.

[17]. Yang, D., Cheng T.C.E., Yang, S. and Hsu, C. (2012): Unrelated parallel-machine scheduling with aging effects and multi-maintenance activities, Computers & Operations Research, Vo. 39, p. 1458-1464.