Powers of Two-Based Heuristic for Job-Shop Scheduling

V Mahesh*, Sandeep Dulluri**,
A Chennakesava Reddy*** and C S P Rao****

This paper deals with makespan minimization of a job shop. The job-shop scheduling problem with the makespan criterion is certainly an NP-hard case, hence optimization algorithms are ruled out in practice. It is therefore unrealistic to try obtaining a solution through a commercial solver in polynomial time. This paper proposes a computationally effective heuristic, which is based on the powers of two policy in inventory, for solving the minimum makespan problem of job-shop scheduling. It also proposes bounds for makespan, to validate the results obtained by the proposed heuristic. An increase in resource utilization is noted as an important by-product of the proposed heuristic.

Introduction

One of the most popular models in scheduling theory is that of the job shop. Job shop is considered to be a good representation of the general domain and has earned a reputation of most difficult to solve among non-deterministic polynomial-time hard (NP-hard) set of scheduling problems. In the job shop case it is more appropriate to describe an operation with *triplet* (i, j, k) in the order to denote the fact that operation i of job j requires machine k for processing.

In many cases, the combination of goals and resources exponentially increases the search space, and thus the generation of consistently good scheduling is particularly difficult because we have a very large combinatorial search space and precedence constraints between operations (Mesghouni *et al.*, 2004). Due to the NP-hard nature of the scheduling problems, it is usually difficult to find an exact optimal schedule and hence we rely on finding a near to optimal solution. In this

* Associate Professor, Department of Mechanical Engineering, S R Engineering College, Warangal, India. He is the corresponding author. E-mail: maheshvutukuru@rediffmail.com

** Portfolio Analyst, New Product Development, Demand, JDA Group Inc. (Formerly Manugistics), Hyderabad, India. E-mail: dulluri@gmail.com

*** Professor, Department of Mechanical Engineering, JNTU Ananthapur, India. E-mail: dr_ecreddy@yahoo.com

**** Professor, Department of Mechanical Engineering, National Institute of Technology, Warangal, India. E-mail: csp_rao@rediffmail.com

regard, a very fruitful approach has been to relax the notion of optimality and settle for near-optimal solutions. A near-optimal solution is one whose objective function value is within some small deviation from the globally optimal value. The near-optimal scheduling algorithms spend time on improving the schedule quality but do not continue until the optimum is found (Hussein, 2005).

This paper deals with a real time scheduling problem of a turbine manufacturing industry where each job is tagged with a priority, and has a release time, processing time and a due date (deadline). The deadline can be expressed relative to the release time or as an absolute time. In priority scheduling, each job is assigned a priority via some policy. Contention for resources is resolved in favor of job with the higher priority that is ready to run.

Next, we present the survey of relevant literature. Then, problem formulation and solution methodology are discussed. Results are presented next. Finally, managerial implications of this work and some directions for future work are provided.

Survey of Relevant Literature

The Job-shop Scheduling Problem (JSP) is proven to be NP-hard. The plethora of research in JSP is mainly focused on developing better algorithms/(meta) heuristics that are easy for deployment in the real world. The key tradeoff in designing of algorithms for JSP is considering the near-optimal solution versus the saving in computation time. In this paper, we review some relevant literature that has connections to the problem that we investigate.

This problem, having been examined for years, is also regarded as an indicator of the quality of advanced scheduling algorithms. Nowicki and Smutnicki (2005) in their work provide a new approximate algorithm that is based on the big valley phenomenon and uses some elements of the so-called path relinking technique as well as the new theoretical properties of neighborhoods. Their algorithm owns unprecedented accuracy obtainable in a quick time on a PC, which has been confirmed after wide computer tests. Zribi *et al.* (2007) present a two-phase algorithm to minimize the makespan for the Multi-Purpose Machine scheduling problem with Availability Constraints (MPMAC). In the first part, authors propose a heuristic based on priority rules to solve the assignment problem. A local search algorithm is then introduced to improve this assignment solution. In the second part, they use genetic algorithm to solve the sequencing problem. Finally, a new lower bound was developed for the problem to evaluate the quality of solutions. The two-phase optimization leads to an interesting computation time since the problem is decomposed into two stages. Tahar *et al.* (2006) consider the problem of scheduling a set of independent jobs with sequence-dependent setup times and job splitting on a set of identical parallel machines such that maximum completion time (makespan) is minimized. For NP-hard problem, a heuristic algorithm using a linear programming modeling with setup times and job splitting considerations was proposed. The performance of

the proposed new method was tested on more than 6000 instances with different sizes by comparing it with a lower bound. Tabu Search (TS) has emerged as an effective algorithmic approach for solving the JSP. However, the quality of solutions found by TS approach depends on the initial solution. To overcome this problem and to provide a robust and efficient methodology for the JSP, Zhang *et al.* (2008) propose the heuristics search approach combining Simulated Annealing (SA) and TS strategy. The main principle of this approach is that SA is used to find the elite solutions inside Big Valley (BV) so that TS can re-intensify search from the promising solutions.

Katsumi and Nobuto (2003) propose a makespan minimization procedure for JSP. The primal objective is to find a minimum makespan schedule by generating and evaluating active schedules implicitly and efficiently under the mechanism of branch and bound. The authors have used Lagrangean relaxation technique, which is an effective sub-optimization approach, to improve the branching decisions. Katsumi's (2004) paper deals with makespan minimization of a job shop. He proposes an effective approach which conducts the search based on the active schedule generation procedure. It tries to find a minimum makespan schedule under the mechanism of depth-first branch and bound. If two or more operations conflict on a machine when generating active schedules, then subproblems are generated by fixing the start time of each conflicting operation. To select a good subproblem, which leads to a minimum makespan schedule, a lower bound on the makespan is calculated by solving the one-machine sequencing problem for each subproblem. Cavory *et al.* (2005) consider the cyclic JSP with linear constraints. In their paper they discuss about the cyclic JSP and its open complexity. They have presented a general approach for solving it based on the coupling of a genetic algorithm and a scheduler. Gao *et al.* (2007) in their paper address the flexible JSP with three objectives—min makespan, min maximal machine workload, and min total workload. They propose a new genetic algorithm hybridized with an innovative local search procedure (bottleneck shifting) for the problem.

Arjen *et al.* (2007) in their work on job insertion problem in multistage scheduling find the feasible insertion of the additional job into the schedule that minimizes the resulting makespan. Pezzella and Merelli (2000) present computationally effective heuristic methods for solving the minimum makespan problem of job-shop scheduling. The local search method adopted by them is based on the TS technique guided by shifting bottleneck procedure. Chung *et al.* (2005) in their paper consider manufacturing industries where outsourcing of manufacturing operation is possible through subcontract. The heuristic algorithm developed by the authors repeatedly executes two steps—(1) improving the sequence of operations, and (2) picking out the operations to be subcontracted—on the bottleneck machines. Raaymakers and Weijters (2003) propose two different techniques namely regression analysis and neural network modeling for estimating the makespan of job sets in batch process industries. Both the techniques use aggregate information. The authors have concluded

that by using aggregate information the presented techniques are less tim consuming in assessing the makespan of a job set compared to the metho based on detailed information.

Girish Keshav (2001) has successfully developed a Dynamic Programming (DI model for scheduling precedence related jobs on two identical parallel processo with the objective of minimizing sum of weighted completion times. The propose DP model can be used to solve when the number of parallel machines available ar powers of 2. An interesting problem would be to develop a model for any arbitrar number of machines. Sha and Hsu (2006) propose a hybrid Particle Swarr Optimization (PSO) for the job-shop problem. As the solution space of the job-sho problem is discrete, they modified the particle position representation, particl movement, and particle velocity to better suit PSO for the job-shop problem. The also applied TS to improve the quality of the solution. Kliewer *et al.* (2006) provide polynomial-time algorithms to deal with the single machine scheduling problems i which the job processing times and/or their release dates is controllable. Possibl changes to the controllable parameters are either individual or done by controllin the relevant processing or release rate. The objective is to minimize the sum of the makespan plus the cost for changing the parameters. Wu and Zhou (2007) paper i concerned with the problems in scheduling a set of jobs associated with random due dates on a single machine so as to minimize the expected maximum delay in stochastic environment. Computation of optimal solutions are initially developed in the case of deterministic processing times, and then extended to stochastic processing times following arbitrary probability distributions.

This work can be viewed as a logical extension of our previous work on job-shop scheduling (Dulluri *et al.*, 2007). We differ from the extant literature in the following way:

- We model the re-circulation of jobs.
- We incorporate multiple jobs with parallel machines, and each job can have change in priorities at the run time.
- We investigate the linkage between makespan and capacity utilization in complex job-shop environment.

Problem Description

We consider a typical problem faced by a job-shop manufacturer. The manufacturer is a world class leader in turbine manufacturing. There is a wide scope and variety of products that are being manufactured in the job-shop. The manufacturer uses a priority mechanism to resolve the resource contentions that arise on special purpose machines. The key objective of the manufacturer is to improve the resource utilization. This will correspond to minimizing the makespan and reducing the idle time of machines/resources. It is assumed that the manufacturer has at least one order and hence there is no possibility (theoretically), that the resources are left idle. It should

be noted that preventive maintenance is scheduled separately and is outside the scope of the current paper.

From the time a customer places an order for a particular product, the order is uniquely identified by a Work Order Number (WONO). Each work order (i.e., product) consists of several sub-assemblies and they are identified by a unique number known as Product Group Main Assembly (PGMA). Further each product group consists of individual parts identified by a part number and the operation sequence of each part is identified by Operation Number (OPNO). The industry may have thousands of parts to be processed on various machines in a given time horizon and a part may 'recirculate' a machine more than once.

Mathematical Formulation

Here, we discuss the mathematical formulation for minimizing the makespan in a typical job-shop environment with multiple machines and multiple jobs that are with distinct priorities/weights. The formulation presented here is a case of static and deterministic problem, it can be later extended to stochastic and dynamic setting.

Notation:

N	:	Set of jobs
M	:	Set of machines
O	:	Set of operations
n	:	Index for job ($n = 1, 2, \ldots, N$)
m	:	Index for machine ($m = 1, 2, \ldots, M$)
o	:	Index for operation ($o = 1, 2, \ldots, O$)
wo	:	Index for work order ($wo = 1, 2, \ldots, WO$)
$partno(wo)$	:	Index for part number of work order 'wo' ($partno = 1, 2, \ldots, PTNO(wo)$)
$pt(n,m)$	:	Processing time of n^{th} job on m^{th} machine
$st(n,m)$	:	Setup time of n^{th} job on m^{th} machine
$c(n,m)$	:	Completion time of n^{th} job on m^{th} machine (The authors ignore the recirculation index here and consider that recirculation would be counted as a separate job.)
$r(n,m)$	:	Release time of n^{th} job on the m^{th} machine
$Ft(n,m)$	:	Flow time of n^{th} job on the m^{th} machine – $Ft(n,m) = c(n,m) - r(n,m)$
$Pri(n)$	:	Priority of the n^{th} job (All jobs belonging to a work order will have the same priority.)
$Wc(m,n)$	:	Weighted completion time for n^{th} job indexed on m^{th} machine

$Pred(n,wo)$: The set of all predecessor jobs for n^{th} job that belong to work order 'wo'

$Predecessor(n)$: The set of all predecessor jobs to n^{th} job across all work orders

$Succ(n,wo)$: The set of all successor jobs for n^{th} job that belong to work order 'wo'

$Successor(n)$: The set of all successor jobs to n^{th} job across all work orders

$Pred(n,m)$: The set of all predecessor jobs for n^{th} job on mth machine.

$Mc(m,w)$: The set of all the machines (work places) in a work center 'w'

$x(n,m)$: Binary variable, takes a value = 1, if n^{th} job is assigned on to the m^{th} machine and equal to 0 otherwise

$$\min\left\{\max_{\{o_j, j=0,1,\ldots,n+1\}\in N}\left[\sum_{m=1}^{M}\left(r\left(0,m\right)+pt\left(n,m\right)\right)^{*}pri\left(m\right)^{*}x\left(n,m\right)\right]\right\} \quad \ldots(1)$$

Subject to:

$$r\left(o\left(n\right),m\right)^{*}x\left(n,m\right)\geq 0 \quad \ldots(2)$$

$$\forall o\left(n\right)\in O\left(n\right)\subseteq O,\ n\in N,\ m\in M$$

$$r\left(o\left(n\right),m\right)^{*}x\left(n,m\right)-r\left(o\left(n'\right),m\right)^{*}x\left(n',m\right)\geq pt\left(n,m\right) \quad \ldots(3)$$

$$\forall\left(o\left(n\right),o\left(n'\right)\right)\in PREC\left(n,m\right),m\in M$$

$$\sum_{n\in N}\sum_{m\in M}x\left(o\left(n\right),m\right)\leq 1 \quad \ldots(4)$$

$$\forall o\left(n\right)\in O\left(n\right)\subseteq O$$

We have thus formulated the JSP as an instance of integer programming problem. In particular, Equation 1 is the objective, and Equations 2, 3 and 4 form the constraints.

The objective function (Equation 1) indicates the expression for weighted makespan. In the current setting the weights are taken as the priorities of the respective work orders (equivalently jobs). Our aim is to minimize the maximum time taken for completion of all jobs across all possible machines. Simultaneously, our intent is to consider the priorities of work orders. In other words, a work order with higher priority should not be kept waiting compared to a work order with lower priority.

The first constraint (see Equation 2) indicates that release times for all jobs across all possible machine set are always non-negative. The second constraint (Equation 3) indicates that the difference in release times of two successive operations

should at the least differ by the processing time of preceding jobs—this constraint holds across all the machines. It should also be noted that we consider the successive processing tasks at an operation level, and therefore we resort to the operation index instead of job index. The third constraint (see Equation 4) indicates that at the most one job can be assigned to a given machine. We can alternately make this constraint restrictive by changing the inequality sign to equality. In other words, our intent is to assign exactly one job to one machine and that all machines are assigned with jobs.

The computation time for any LP (Linear Programming) solver depends on the size of the problem. As it is evident that the JSP problem is NP-hard. It is therefore unrealistic to try obtaining a solution through a commercial solver in polynomial time. Therefore, we propose our heuristic which is based on the powers of two policy in inventory (Mucksdat and Roundy, 1993).

Powers of Two-Based Priority Based Scheduling

The crux of this heuristic is to accommodate the manufacturer's/planner's intelligence blended into the generalized job-shop makespan minimization framework for achieving good results. It is well-known that powers of two concept has been widely useful in the inventory control literature. It was proposed by Mucksdat and Roundy (1993).

The concept of powers of two in inventory control can be mapped with job-shop scheduling application. The mapping in this case is a cost to time map. In inventory control setup, we deal with cost structure while in the job-shop scheduling the key variable is time/makespan. The planners are given arbitrary release dates for work orders (hence the underlying jobs) from the planning department. The planners take this as input and try to match it to the nearest powers of two release date. We consider three options in adjusting the release date of a job to the nearest powers of two. They are: pure preponement, pure postponement, and partial summation to nearest powers of 2.

Flow of the Heuristic

1. At time $t = 0$, set the setup time for all jobs with no predecessor(s) as $st(n,m) = 0$. Mathematically, $st(n,m) = 0|_{\text{no predecessor}}$

   In addition, set the completion time of all the jobs which has no predecessor(s) equal to the processing time. In other words, $C(n,m) = pt(n,m)|_{\text{no predecessor}}$

2. Break the work orders into corresponding jobs.

3. Arrange the jobs coming on to a machine in the increasing order of priority (for example, a job with priority 3 should be loaded on to a machine before loading a job with priority > 3).

4. Mapping the nearest release date: we map that the maximum/minimum adjustment to the release date. This will not exceed half the nearest powers of two. We fix a threshold of 3 (this can be configured as per user's choice and convergence criteria).

The release date adjustment can be considered as valid, if and only if the difference between the actual release date and the adjusted release date lies with in the power of two of the threshold.

Alternately, there could be a simple mechanism for checking the adjustment in release dates. For example, if the work order has a release date on the 19th day. We would prepone it to not less than the 16th day (2 ^ 4). Or we could alternately consider to postpone it to the 32nd day (2 ^ 5). In the current case, preponing to 16th day will be considered since it is closer (19 – 16 = 3 vs. 32 – 19 = 13) to the actual release date.

a) Preponement mapping: In this case, the planner always tries to prepone the release date(s) of one or more jobs to the nearest powers of two. For example, consider a work order whose release date is 19th day—applying the preponement mapping would adjust the release date of this work order to 16th day (2 ^ 4). The difference between actual release date and adjusted release date is 19 – 16 = 3 days (<2 ^ 3). Hence, we treat this adjustment as acceptable.

b) Postponement mapping: In this case, the planner always tries to postpone the release date(s) of one or more jobs to the nearest powers of two. For example, consider a work order whose release date is 19th day—applying postponement would take the release date to 32nd day (2 ^ 5). Consider the difference 32 – 19 = 13, this difference is greater than the permissible threshold (2 ^ 3) 8 days. Hence, a postponement mapping might not be valid. However if the planner still wants to go for postponement mapping then she should consider 32nd day as the adjusted release date.

Note: In cases (4a) and (4b), we do not allow for partial powers i.e., the adjusted release date is always in the set of integer powers of two.

Adjusted release date $r(n,m) \in 2^{Z^+ \bigcup \{0\}} \forall n \in \{1,2,\ldots,N\}, m \in \{1,2,\ldots,M\}$.

c) Partial summation to the nearest powers of 2: In this case, planner is allowed to use sums of powers of two. However, the integrality of powers of two should still be maintained.

5. Applying any of the powers of two adjustment (4a or 4b or 4c), we compute the optimal makespan on m^{th} machine $WC_{m,\max} = \max\{WC_{m,1}, WC_{m,2}, \ldots, WC_{m,n}\}$.

We generalize this schedule across the set of all jobs (N).

We apply the precedence and successor constraints on the optimal job-shop sequence to ensure the feasibility of the solution in real time. These constraints are discussed in Steps 6 and 7.

6. a) Precedence constraints: $(r(n2,m) - r(n1,m) - pt(n1,m) >= 0)$. The release time of job "$n2$" belonging to the predecessor set of $n1$ (i.e., $pred(n1)$), should differ by the processing time of job "$n1$" on machine "m".

b) Similarly this constraint should also hold across all machines.

$r(n,m1) - r(n,m2) - pt(n,m1) >= 0$, machine "m2" comes after "$m1$" in the processing list.

7. a) Successor constraints: $(c(n2,m) - c(n1,m) - pt(n1,m) >= 0)$. The completion time of job "$n2$" belonging to the predecessor set of $n1$ (i.e., $succ(n1)$), should differ by the processing time of job "$n1$" on machine "m".

   b) Similarly this constraint should also hold across all machines.

   $c(n,m1) - c(n,m2) - pt(n,m1) >= 0$, machine "$m2$" comes after "$m1$" in the processing list.

8. Go to Step 3 and repeat the entire process till there are no more jobs left for scheduling.

9. STOP: The resulting sequence of release times, completion times is the optimal result.

Problem Size

The complexity and magnitude of the scheduling problem can be better understood by considering a product like steam turbine. The steam turbine is a heavy machine consisting of approximately 600 to 800 components, which may be classified into, major, sub-major, and minor components. Out of all these components there are about 15 to 30 critical components that are manufactured in the heavy machine shop. The number of operations that are performed for each major component may range from 25 to 125. For example, the number of operations to be performed on an outer casing of a turbine is around 125 and for valve chest it is about 25. These parts have to be processed through various machine shops like blade shop, heavy machine shop, medium machine shop and rotor bay. In the current research work, the authors have experimented on the real time data available for processing few parts of a turbine in the heavy machine shop, which may be extensible for all the parts and all machines shops where they require processing.

The problem size under consideration is as follows: 10 work orders, with each work order consisting of 15 or more part numbers. Each part on an average undergoes 30 operations. There are 20 work centers and considering the parallel machines, the total number of machines amount to 82.

Results and Discussion

We apply first the priority heuristic to obtain an initial solution. The makespan obtained by the priority heuristic for 10 work orders of the problem under consideration is reported in Table 1. The release dates of work orders are arbitrary as given by the planning department. While scheduling these work orders on the available machines, if two jobs are waiting to be processed on the same machine, the job with the higher priority is given preference and will be processed first. The job with the lower priority is kept waiting till the processing of higher priority job finishes.

Table 1: Makespan of Work Orders with Random Release Time				
Sl. No.	Work Order No.	Initial Priority	Release Date (in Days)	Makespan (in Days)
1.	WONO 1	1	1	32
2.	WONO 2	2	19	55
3.	WONO 3	6	28	73
4.	WONO 4	3	45	100
5.	WONO 5	5	27	128
6.	WONO 6	4	49	125
7.	WONO 7	7	56	155
8.	WONO 8	8	70	180
9.	WONO 9	9	82	205
10.	WONO 10	10	110	228

We report, in Table 2, the nearest powers of 2 release dates as well as the makespan results of 10 work orders. The partial summation of the nearest powers of 2 is taken as release date so that it approaches nearer to the arbitrary value considered in Table 1. We see that the maximum/minimum adjustment to the release date will not exceed half the nearest powers of two. Work order number 4 of Table 1 has an arbitrary release date on the 48th day. The nearest powers of 2 release date for the same is considered as $((2 \wedge 5)+(2 \wedge 4) = 48)$. The net saving in makespan of all the work orders under consideration is observed as 25 days in comparison to arbitrary release days.

Table 2: Makespan of Work Orders with Nearest Powers of 2 Release Time				
Sl. No.	Work Order No.	Initial Priority	Release Date (in Days)	Makespan (in Days)
1.	1	1	1	32
2.	2	2	16	49
3.	3	6	24	69
4.	4	3	48	101
5.	5	5	24	120
6.	6	4	48	126
7.	7	7	48	152
8.	8	8	68	178
9.	9	9	80	203
10.	10	10	96	226

Table 3 represents makespan of work orders with preponed powers of 2 release time. In this case, the planner always tries to prepone the release date(s) of one or

more jobs to the nearest powers of two. Applying the preponement mapping would adjust the release date of the work order number 4 (which is 45 in Table 1) to the 32nd day (2 ^ 5). It is observed that the net saving of makespan is about 133 days compared to the random release date.

Table 3: Makespan of Work Orders with Preponed Powers of 2 Release Time

Sl. No.	Work Order No.	Initial Priority	Release Date (in Days)	Makespan (in Days)
1.	1	1	1	32
2.	2	2	16	49
3.	3	6	16	60
4.	4	3	32	86
5.	5	5	16	105
6.	6	4	32	111
7.	7	7	32	139
8.	8	8	64	164
9.	9	9	64	189
10.	10	10	64	213

Finally, we report the results of makespan of work orders with postponed powers of 2 release time in Table 4. In this case, the planner always tries to postpone the release date(s) of one or more jobs to the nearest powers of two. For the work order number 4 with arbitrary release date 45, the postponed powers of release date is taken as 64 which is (2 ^ 6). In this case, the net increase in the makespan of all the 10 work orders under consideration is obtained as 80 in comparison to random release dates.

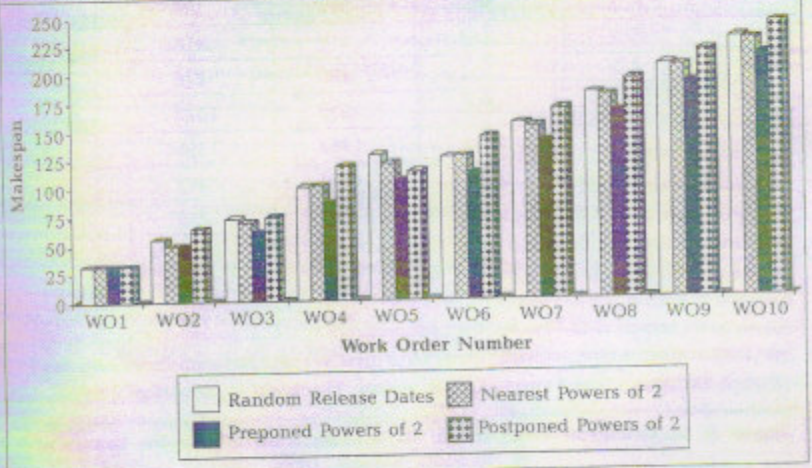
Table 4: Makespan of Work Orders with Postponed Powers of 2 Release Time

Sl. No.	Work Order No.	Initial Priority	Release Date (in Days)	Makespan (in Days)
1.	1	1	1	32
2.	2	2	32	64
3.	3	6	32	74
4.	4	3	64	118
5.	5	5	32	113
6.	6	4	64	143
7.	7	7	64	167
8.	8	8	128	192
9.	9	9	128	217
10.	10	10	128	241

From the results presented in Tables 2, 3 and 4, it is quite evident that preponement mapping always reduces the makespan. This is obvious because early starting should always result in an early ending. The makespan values obtained by preponement mapping can be taken as the lower bounds for the makespan. Postponement mapping has the exact opposite impact. Postponement mapping tends to increase the makespan and hence this forms the upper bound for the makespan.

We notice that makespan results obtained by the proposed heuristic are better than the random release dates and also these results are validated with that obtained by preponed and postponed powers of 2, which represent the lower and upper bounds of the makespan (Figure 1).

Figure 1: A Graphical Comparison of Makespan with Nearest Powers of 2 vs. Random Release Dates, Preponed Powers of 2 and Postponed Powers of 2



To evaluate the effectiveness of the proposed heuristic, the Computation Time(s) (CPU) were also collected. Table 5 represents the computation time required to obtain the detail schedule of each work order. The algorithms are coded in Visual Basic with Oracle as database and implemented on a personal computer (Pentium IV, 3.07 GHz CPU with 1.0 GB RAM). The average computation time for 10 work orders with 30 machines is around 1200 seconds, with the median computation time being 1500 seconds.

It is observed that there is an overall saving of 308 seconds of computation time in the case of nearest powers of 2 release dates when compared to that of random release dates. In the case of preponed powers of 2 release dates, the computation time is about 180 seconds more than that obtained by random release dates. The reason for this is due to the fact that more number of resource contentions have occurred by shifting the dates to preponed powers of 2 and therefore more CPU time is consumed in resolving them.

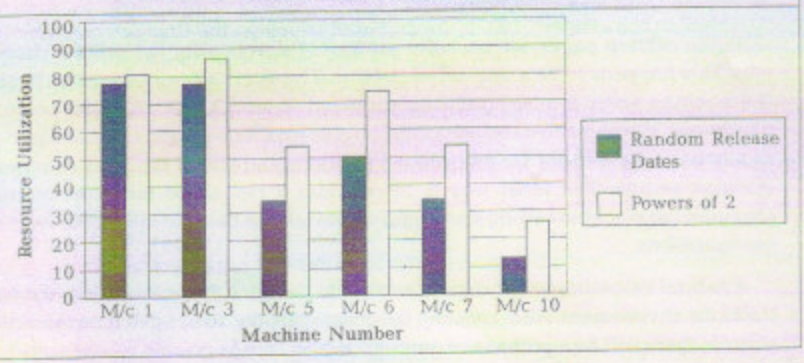
But, in the case of postponed powers of 2 the saving of time is about 324 seconds compared to random release dates. This is due to the occurrence of lesser number of resource contentions by postponing all the dates. Though, the overall saving in computation time is more in the case of postponed powers of 2, it is not preferred due to its inability to minimize the makespan, the primary criterion for an industry.

Table 5: Comparison of Computation Time (in Seconds) with Nearest Powers of 2 vs. Random Release Dates, Preponed Powers of 2 and Postponed Powers of 2

WONO	Priority	Computation Time (s)			
		Random Release Dates	Nearest Powers of 2 Dates	Preponed Powers of 2 Dates	Postponed Powers of 2 Dates
1.	1	396	396	396	396
2.	2	449	418	418	408
3.	6	424	456	478	447
4.	3	1025	1030	1072	1011
5.	5	1132	1084	1198	1107
6.	4	1893	1702	1877	1709
7.	7	760	748	772	703
8.	8	789	755	761	763
9.	9	751	692	791	706
10.	10	728	758	764	773

Minimum makespan corresponds to minimum idle time on the resources and hence, maximum overall capacity utilization. The nearest powers of 2 scheduling

Figure 2: A Graphical Comparison of Resource Utilization with Powers of 2 vs. Random Release Dates



not only produce minimal makespan but also a noticeable improvement in the machine utilization (see Figure 2). The resource utilization graph is shown for the high cost/critical machines of the heavy machine shop of the industry under study. The utilization is shown for a typical time bucket of 30 calendar days, and all the 10 work orders are being processed on the shop floor during this period.

Implications for Management

Scheduling deals with assigning multiple jobs on to multiple resources with each job requiring some processing time on all/some resources. The resources are often constrained in terms of capacity. With the increase in the number of jobs or/and resources, the problem of optimally allocating the jobs on to the resources in a time-phased manner becomes much more tough and challenging. Many resource contention problems are modeled as a JSP. The proposed heuristic attempts to solve a large scale JSP with a reasonable level of accuracy and with huge savings in the computation time. Further, the heuristic is easy to implement and can be readily adapted in various industrial settings for decision making.

Future Work

Based on our interaction with the user industry, the job-shop personnel are not inclined towards postponement mapping due to the threat of not meeting the due dates. On the other hand, they are not interested in preponement mapping either, as the preponement would put extra load on the resources (including human resources). In this regard, we had to think of an industrially feasible solution. Hence, we had to consider the idea of partial summation of powers of two. This can be viewed as splitting one job into two equivalent dummy jobs that are to be processed sequentially. For the first dummy job, the release time will be the first nearest power of two and with a processing time equal to the second nearest power of two. The second dummy job starts at the end of processing of the first dummy job. In the context of inventory control, we can view this as a case of partial shipments for the same order in order to ensure full-truck load conditions.

In the current paper, we consider postponing/preponing the release dates of jobs. This happens to be a one-sided control. The alternate way could be to align the machine settings (configuration problem) to match the jobs that come for processing. The one-sided methods might not always form a good choice, especially in a setting where there are engineered products and which requires changes in machine setups. The ideal way is to consider a two-sided mix that includes changing the machine setting and aligning the release dates to match the machine configurations.

A natural extension to our current work is to consider the problem in a dynamic stochastic environment. Also, consider the geographically distributed manufacturing, wherein there will be a problem of gaming. ❖

References

1. Arjen P A Vestjens, M Wennink and Woeginger G J (2007), "Complexity of the Job Insertion Problem in Multi-Stage Scheduling", *Operations Research Letters*, Vol. 35, No. 6, pp. 754-758.

2. Cavory G, Dupas R and Goncalves G (2005), "A Genetic Approach to Solving the Problem of Cyclic Job Shop Scheduling with Linear Constraints", *European Journal of Operational Research*, Vol. 161, No. 1, pp. 73-85.

3. Chung S H, Lee A H L and Pearn W L (2005), "Analytic Network Process (ANP) Approach for Product Mix Planning in Semiconductor Fabricator", *International Journal of Production Economics*, Vol. 96, pp. 15-36.

4. Dulluri S, Mahesh V and Rao C S P (2007), "A Priority Based Scheduling Heuristic for Turbine Manufacturing Job-Shop", accepted for publication in the *International Journal of Industrial Systems and Engineering*.

5. Gao J, Gen M, Sun L and Zhao X (2007), "A Hybrid of Genetic Algorithm and Bottleneck Shifting for Multiobjective Flexible Job Shop Scheduling Problems", *Computers & Industrial Engineering*, Vol. 53, No. 1, pp. 149-162.

6. Girish Keshav P (2001), "Simulated Annealing: A Heuristic Optimization Algorithm", *Dr. Dobb's Journal, Software Tools for the Professional Programmer*, Vol. 26, No. 9, pp. 121-123.

7. Hussein M (2005), "New Utilization Criteria for Online Scheduling", Ph.D. Thesis Dortmund University.

8. Katsumi M (2004), "Makespan Minimization of a Job Shop Based on Active Schedule Generation Procedure", *Journal of Japan Industrial Management Association*, Vol. 55, No. 1, pp. 9-15.

9. Katsumi M and Nobuto N (2003), "Finding a Minimum Makespan Schedule for the Job-Shop Problem by Branch and Bound Using the Lagrangean Relaxation Technique", *Journal of Japan Industrial Management Association*, Vol. 53, No. 6, pp. 466-473.

10. Kliewer Natalia, Mellouli Taleb and Suhl Leena (2006), A Time-Space Network Based Exact Optimization Model for Multi-Depot Bus Scheduling", *European Journal of Operational Research*, Vol. 175, No. 3, pp. 1616-1627.

11. Mesghouni K, Hammadi S and Borne P (2004), "Evolutionary Algorithm for Job Shop Scheduling", *International Journal of Applied Mathematics Computer Sciences*, Vol. 14, No. 1, pp. 91-103.

12. Muckstadt J A and Roundy R O (1993), "Analysis of Multistage Production Systems", in Graves S C, Rinnooy Kan A H G and Zipkin P H (Eds.), *Handbooks in Operations Research and Management Science*, Vol. 4, pp. 59-131, North-Holland, Amsterdam.

13. Nowicki E and Smutnicki C (2005), "An Advanced Tabu Search Algorithm for the Job Shop Problem", *Journal of Scheduling*, Vol. 8, No. 2, pp. 145-159.

14. Pezzella F and Merelli E (2000), "A Tabu Search Method Guided by Shifting Bottleneck for the Job Shop Scheduling Problem", *European Journal of Operational Research*, Vol. 120, No. 2, pp. 297-310.

15. Raaymakers W H M and Weijters A J M M (2003), "Makespan Estimation in Batch Process Industries: A Comparison Between Regression Analysis and Neural Networks", *European Journal of Operational Research*, Vol. 145, pp. 14-30.

16. Sha D Y and Hsu C Y (2006), "A Hybrid Particle Swarm Optimization for Job Shop Scheduling Problem", *Computers & Industrial Engineering*, Vol. 51, No. 4, pp. 791-808.

17. Tahar D N, Yalaoui F, Chu C and Amodeo L (2006), "A Linear Programming Approach for Identical Parallel Machine Scheduling with Job Splitting and Sequence-Dependent Setup Times", *International Journal of Production Economics*, Vol. 99, Nos. 1-2, pp. 63-73.

18. Wu X and Zhou X (2007), "Stochastic Scheduling to Minimize Expected Maximum Lateness", *European Journal of Operational Research* (in Press), corrected proof available online, June 16.

19. Zhang C Y, Li P G, Rao Y Q and Guan Z L (2008), "A Very Fast TS/SA Algorithm for the Job Shop Scheduling Problem", *Computers & Operations Research*, Vol. 35, No. 1, pp. 282-294.

20. Zribi N *et al.* (2007), "Minimizing the Makespan for the MPM Job-Shop with Availability Constraints", *International Journal of Production Economics*, doi: 10.1016/j.ijpe.2007.01.014.

Reference # 07J-2008-02-02-01