2006

A Dynamic Heuristic for the Stochastic Unrelated Parallel Machine Scheduling Problem

Jean-Paul Arnaout
Old Dominion University

Ghaith Rabadi
Old Dominion University

Ji Hyon Mun
Old Dominion University

Original Publication Citation

Arnaout, J.-P., Rabadi, G., & Mun, J. H. (2006). A dynamic heuristic for the stochastic unrelated parallel machine scheduling problem. *International Journal of Operations Research, 3*(2), 136-143.

International Journal of Operations Research

International Journal of Operations Research Vol. 3, No. 2, 136–143 (2006)

A Dynamic Heuristic for the Stochastic Unrelated Parallel Machine Scheduling Problem

Jean-Paul Arnaout*, Ghaith Rabadi, and Ji Hyon Mun

Engineering Management and Systems Engineering Department, Old Dominion University
241 Kaufman Hall, Norfolk, VA 23529, USA

Received December 2006; *Revised June 2006*; *Accepted July 2006*

Abstract—This paper addresses the problem of batch scheduling in an unrelated parallel machine environment with sequence dependent setup times and an objective of minimizing the total weighted mean completion time. The jobs' processing times and setup times are stochastic for better depiction of the real world. This is a NP-hard problem and in this paper, new heuristics are developed and compared to existing ones using simulation. The results and analysis obtained from the computational experiments proved the superiority of the proposed algorithm *PMWP* over the other algorithms presented.

Keywords—Simulation, Setup time, Unrelated parallel machine, Stochastic times, Heuristics

1. INTRODUCTION

Scheduling is considered to be a crucial factor in most manufacturing and production systems. Most of the research that has been done in this field concentrated on deterministic settings and assumed known processing and setup times with certainty. Unfortunately, most manufacturing scenarios are stochastic in nature, which leads the deterministic schedule to become obsolete once it hits the shop floor. Therefore, the proposed heuristics in this paper are tested under stochastic processing and setup times as an attempt to capture the existing randomness in a realistic environment. The stochastic jobs are scheduled on unrelated parallel machines, which are machines that have different processing times for the same job. Furthermore, the jobs are grouped in batches before being sent to the machines as it may be cheaper and faster to process jobs in batches than to process them individually (Potts and Kovalyov, 2000). One of the main benefits gained by batch scheduling is revealed in the case of setup times, where the machines incur setup times associated with processing different jobs; lots of time can be saved by scheduling identical jobs in batches, as setup will only be performed when switching batches instead of individual jobs.

There are two possible scenarios in batch scheduling environments: the first is job availability, where a job becomes available immediately after the processing of its predecessor is completed. The second is batch availability, in which a job will not be available until the complete previous batch has been processed. This paper addresses the concept of batch availability.

The scheduling objective is to minimize the total weighted mean completion time, which is at least a NP-hard problem as the simplified problem of two identical machines with no setup times is NP-hard in the ordinary sense (Bruno et al., 1981). Discrete event simulation will be used to model and test four heuristics for the problem addressed in this paper. Some preliminary promising results for this problem have been reported by Arnaout and Rabadi (2005).

The rest of this paper is organized as follows. In section 2 the related research is summarized. In section 3 the problem statement and objective function are presented. Section 4 contains description of the heuristics developed and used. The simulation model verification is presented in section 5; the computational results and output analysis are respectively described in sections 6 and 7. Finally, we conclude our results in section 8.

2. RELATED RESEARCH

The most common objectives studied in parallel machine scheduling are minimization of completion time, tardiness, and makespan. Previous research indicated that even the identical parallel machine problem with minimization of total tardiness was NP-hard (Karp, 1972). Due to this complexity, it became a common and acceptable practice to find suitable heuristics instead of optimal solutions for these complex scheduling problems.

The literature defines unrelated parallel machines as machines having different processing times for the same job (Liaw et al., 2003). They are unrelated in the sense that the processing speed depends on the job being executed and not the machine; each job will have different processing times for each of the available machines. Ghirardi and Potts (2005) considered the problem of

* Corresponding author's e-mail: jarna002@odu.edu

scheduling jobs on unrelated parallel machines to minimize the makespan. The heuristic they used was an application of the recovering beam search. Weng et al. (2001) addressed the problem of scheduling a set of independent jobs on unrelated parallel machines with sequence dependent setup times so as to minimize the weighted mean completion time. They presented in their paper seven heuristic algorithms and tested them. In their algorithms, they either assigned a job to the machine with the least cost contribution, or to the machine on which the job has the shortest processing time. They also introduced an algorithm where they first assigned the job with the smallest ratio of processing time plus setup time to weight; this strategy outperformed the rest significantly. The authors claimed that their algorithms are extremely fast and can find solutions for up to 120 jobs and 12 machines in a fraction of a second. Low (2005) solved a multi-stage flow shop scheduling problem with unrelated parallel machines and an objective of minimizing total flow time in the system. A simulated annealing (SA)-based heuristic was proposed to solve the addressed problem in a reasonable running time. Mosheiov and Sidney (2003) addressed the case of job-dependent learning curves and applied it to the problem of unrelated parallel machines with the objective of minimizing total flow time. Rabadi et al. (2006) addressed the same problem with sequence dependent setup times to minimize the makespan, where they introduced a new heuristic (Meta-RaPS) for the deterministic problem and compared it to an existing heuristic called the Partitioning Heuristic, which was introduced by Al-Salem (2004). Meta-RaPS outperformed the existing Partitioning Heuristic in almost all cases.

Considerable research has also been published on batch scheduling. Schwindt and Trautmann (2000) proposed a new solution approach using models and methods of resource constrained project scheduling to minimize the makespan, subject to sequence dependent setup times in process industries. Cheng et al. (2001) addressed the single machine batch scheduling problem with resource dependent setup and processing times. They considered a common setup time for all batches, and their objective was to minimize either the maximum job lateness or the total weighted resource consumption. Brucker et al. (1998) solved for the parallel machine batch scheduling problem with sequence independent setup times and an objective of finding feasible schedules with respect to deadlines. For an extensive review on the topic, please refer to Potts and Kovalyov (2000). It is important to note that the addition of sequence dependent setup times adds a lot of complexity to the problem. Allahverdi et al. (1999, 2006) reviewed the literature that involved setup times, and reported that Dietrich (1989) developed a heuristic for the unrelated parallel machine problem with major and minor setups, Guinet (1990) developed heuristic algorithms for the same problem but with sequence dependent setup times, and Elmaghraby et al. (1993), after extending Guinet's (1990) results, introduced an improved iterative heuristic to minimize makespan.

Stochastic machine scheduling problems have been considered, among others, by Glazebrook (1979), Weiss and Pinedo (1980), Bruno et al. (1981), Weber et al. (1986), Weiss (1992), and Mohring et al. (1999). However, and up to our knowledge, no previous research has addressed the generation of schedules in unrelated parallel machines scenarios with stochastic processing and sequence dependent setup times, and this is where the contribution of this paper lies.

3. PROBLEM STATEMENT

The scheduling problem considered in this paper can be described as follows. There are M unrelated parallel machines and B batches, where a batch refers to a lot containing n identical jobs, and different batches have different job types. In the case where there are not enough identical jobs to form a full batch, a partial one will be produced. As we are assuming the concept of batch availability, all jobs in a specific batch should be processed on the same machine to which the batch is assigned. Each machine is assumed to be available at time 0 and can process one job at a time.

The jobs are simultaneously available at the beginning of the scheduling horizon (at time zero). Further more, each job can be processed on any of the machines but needs to be processed by one machine only, and each machine is capable of processing only one job at a time. Job preemption is not allowed and there is no processing precedence on any of the machines. Each job has a weight (w_i) indicating its importance, where w_i has values between 1 and 5 with 1 being the least urgent and 5 the most urgent. The machine setup times are dependent on jobs' sequence where setup times depend on both the batch just completed and the next batch to be processed, and there is no setup between jobs belonging to the same batch. Setup times are assumed to be machine independent such that regardless to which machine batches k and i are assigned, s_{ki} would be the setup time required if batch i is scheduled after batch k.

The batch processing times are dependent on the machine they are assigned to; job J_i has a processing time p_{ij} when it is assigned to machine M_j. For example, the processing time of J_1 on machine M_2 is equal to p_{12}. However, jobs in the same batch are assumed to have the same processing times when processed on the same machine. For a given schedule, job J_i completion time on machine M_j is represented by C_{ij}, and our objective is to find a near optimal schedule that can minimize the total weighted mean completion time. This is represented as follows:

$$\text{Minimize } Z = \frac{1}{(\eta)} \sum_{i=1}^{\eta} w_i C_{ij}, \tag{1}$$

where η is the total number of jobs, and the completion time of job J_i on machine M_j is given by: $C_{ij} = C_{kj} + p_{ij} + s_{ki}$, where C_{kj} refers to the completion time of the job J_k that preceded job J_i on machine M_j.

It is important to note that Equation (1) is in function

of jobs instead of batches because each batch must be separated before being processed on a machine, as the latter can not process more than one job at a time.

4. HEURISTIC ALGORITHMS

The basic and easiest method to obtain a solution for the parallel machine problem is by randomly scheduling the jobs to the machines (Kim et al., 2003), which will result in low quality solutions most of the time. Since this is a hard problem, obtaining optimal solutions will be very time consuming and could be computationally infeasible. From here came the need to invest more time in developing appropriate heuristics. In the following sections, different heuristics are presented and compared in order to determine the most appropriate one for our problem. Recall that the jobs' processing and setup times are stochastic and drawn from different uniform distributions. Whenever a job is called by any algorithm to be sorted with the other jobs or sent to a machine, it will be assigned a processing time and setup time following some uniform distribution; this is discussed more in section 6.

4.1 Heuristic 1 (*WSPT*)

In the *weighted shortest processing time first (WSPT) rule*, batches (containing identical jobs) will be sorted from the smallest $[p_{ij}/w_i]$ to the largest, and then they will be assigned to the different machines according to the smallest $[(p_{ij} + s_{ki})/w_i]$. *WSPT* has been used by a great number of researchers, especially when the objective is to minimize the total completion time in parallel machine environments. This rule was proven to obtain optimal results in the single machine total weighted completion time problem and very good results in the same problem but on identical parallel machines (Pinedo, 1995). The heuristic can be summarized as follows:

Step 1. {Sort the batches in the increasing order of their processing time over weight}
  (a) Obtain the minimum processing time ϱ_i for each batch: $\varrho_i = \min\{p_{i1}, p_{i2}, ..., p_{iM}\}$, where i is the batches' index, and M is the total number of machines.
  (b) Reorder the batches as follows: $\varrho_1/w_1 \leq \varrho_2/w_2 \leq \ldots \leq \varrho_B/w_B$.

Step 2. After sorting the batches, send them one by one to the machines. Assign each batch to the machine that has the smallest $[(p_{ij} + s_{ki})/w_i]$.

Step 3. Before a batch gets processed, separate its jobs so they can be processed one by one on the assigned machine.

Step 4. STOP once all the jobs are assigned.

As one can see, *WSPT* neglects the setup time when sorting the batches, which could lead to low quality solutions if the setup times mount to a considerable portion of the processing times. The computational complexity of *WSPT* is $O(B^2 + BM + Blog(M))$, where we recall that B and M are respectively the number of batches and machines.

4.2 Heuristic 2 (*MWP*)

Heuristic 2 works similar to *WSPT*, except that in *Step* 1, the batches are sorted according to the smallest $[(p_{ij} + s_{ki}) \times$ attuned weight component].

In the total tardiness minimization problems, the earliest weighted due date (EWDD) rule has been used quite often. The weighted due date is calculated by multiplying the due date by an attuned weight component which we will refer to as γ in this paper. Kim et al. (2003) noted that the weight component γ is represented as follows:

$$\gamma = [1 - \alpha a \times (w_i)] \quad (2)$$

where $\alpha \in (0, 0.2)$ is the weight control parameter; the selection of this range is explained in section 4.4. Due to its high-quality results, we decided to alter the EWDD rule so it can be used in our problem. α value was determined to be 0.1 for the total tardiness minimization problem (Kim et al., 2003); empirical tests showed that this is also the best value when used in this heuristic for the problem at hand. The altered rule will be referred to as the minimum weighted processing time (*MWP*), where the MWP_i index for batch i is calculated by multiplying the minimum processing time ϱ_i of each batch by the attuned weight parameter:

Step 1. {sorting batches}
  (a) Obtain the minimum processing time ϱ_i for each batch i.
  (b) Calculate for each batch its *MWP*:

$$MWP_i = \varrho_i \times [1.0 - (0.1) \times (w_i)].$$

  (c) Reorder the batches from the smallest *MWP* to the largest.

Step 2. *Step* 2, 3 and 4 are exactly like in Heuristic 1.

MWP is more complex ($O(B^2 + BM + B + Blog(M))$) than *WSPT* due to the calculation of MWP_i.

4.3 Heuristic 3 (*Weng's Algorithm*)

Weng et al. (2001) studied the problem of unrelated parallel machine scheduling with sequence dependent setup times and a total weighted mean completion time objective. They presented in their paper seven heuristics, and showed through extensive computational experiments that their heuristic *algorithm* 7 significantly outperformed the other seven heuristics presented. *Algorithm* 7 does not sort the jobs according to a predetermined order; instead, among the unscheduled jobs, it next assigns the job with the smallest ratio of processing time plus setup time to weight. So every time a job needs assignment, the algorithm looks at all the unscheduled jobs, determines which one has the

smallest $[(p_{ij} + s_{ki})/w_i]$ on which machine, and it assigns this job to the associated machine.

Weng's Algorithm was modeled through simulation and compared with the proposed heuristics in this paper. The algorithm's complexity ($O(BM)$) is less than both *MWP* and *WSPT*, as it assigns the batches directly to the machines without presorting them.

4.4 Heuristic 4 (*PMWP*)

The *Pick Minimum Weighted Processing Time (PMWP)* algorithm is introduced in this paper and is similar to *Weng's Algorithm* in the sense that it will not sort the batches according to a predetermined order. However, it will pick from the unscheduled batches the one with the smallest $[(p_{ij} + s_{ki}) \times \gamma]$ and assigns it to the machine where this minimum exists:

Let S be a set containing the unscheduled batches.

Step 1. Find batch i and machine j where the Eq. (3) is at its minimum:

$$[C_{kj} + (p_{ij} + s_{ki}) \times (1 - (\alpha \times w_i))], \quad (3)$$

where $i \in S$ (index of unscheduled batches), j is the machine index, and k is the previous batch on that specific machine M_j.

In Equation (3) above, C_{kj} refers to the completion time of the last batch on machine j, and the control parameter a value was determined from Figure 1 and design of experiments (section 4.4.1).

Step 2. After finding both i and j, assign batch i to machine M_j, and remove batch i from list S.

Step 3. If $S = \emptyset$, STOP; else go to *Step* 1.

PMWP complexity ($O(BM)$) is equal to *Weng*'s, i.e. it is less than both *MWP* and *WSPT*.

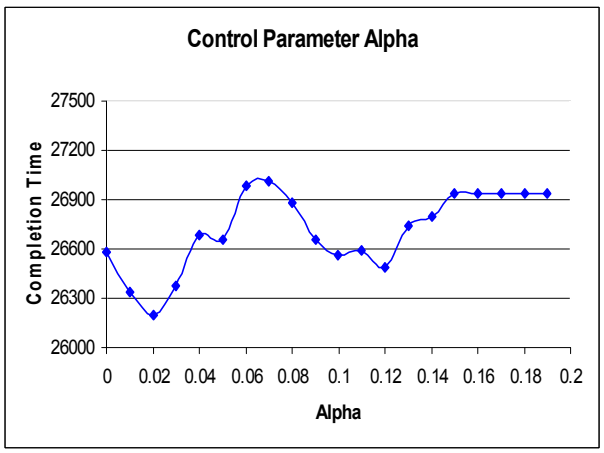

Figure 1. Control Parameter α.

The chart in Figure 1 describes how the completion times of batches fluctuate when α is changed while applying *PMWP* to the problem at hand. Recall that the values of α are between 0 and 0.2; α cannot be 0 because Equation (2) will then be equal to $(1 - (0 \times w_i)) = 1$, meaning that the weight will not be considered in the decision. Also α cannot be 0.2 because w_i could be anywhere from 1 to 5; and therefore when $w_i = 5$, Eq. (2) will then be $(1 - (0.2 \times 5)) = 0$, which will lead to incorrect decisions, as the algorithm will assign the wrong jobs first assuming that they have the smallest $[(p_{ij} + s_{ki}) \times \gamma]$. We can conclude from Figure 1 that the algorithm is giving the best solution when $\alpha = 0.02$, and this will be the value to be used in the proposed heuristic *PMWP*. It is worth reminding here that a was equal to 0.1 when used with the *MWP* heuristic, and it was not used with neither the *WSPT* heuristic nor *Weng's algorithm*.

4.4.1 *PMWP* design of experiments

Although the best value of a was determined to be 0.02 after running several replications of the same problem design (4 machines and 120 jobs), a single problem design is not sufficient to decide on a values for different problem configurations, i.e. we can not guarantee that the best a is 0.02 for all problem combinations. Therefore, Design of Experiments (DoE) was used to determine the appropriate levels (parameters) of α that will contribute to better objective function values in the various problem configurations. Numerous publications provide a good review about DoE (e.g., Fisher, 1960; Taguchi, 1993).

Four main factors along with their interactions were considered for analysis. The factors are described as follows: 1. *Machines*, referring to the number of machines; 2. *Batches*, referring to the total number of batches (which is equal to the number of jobs divided by the batch size); 3. *Weight to Processing*, indicating the percentage that the weight accounts to the processing time; 4. α, the control parameter.

A 2-level full factorial design was carried out, with a total number of experiments equal to 2^k (2 referring to the number of levels and k the number of factors) = $2^4 = 16$ experiments. The factors' levels are shown in Table 1.

The parameter α levels were respectively 0.02 and 0.18 based on Figure 1, where 0.02 indicated the best α value, and 0.18 is a poor a located on the other extreme of the parameter range [0, 0.2]. The reason these two levels were chosen is to capture the two extremes in α range. The results shown in Table 3 proved the correctness of the approach and the choice of a two-level factorial design versus a three (or more) level design that would have required a much more extensive number of experiments.

The experiments along with their results are shown in Table 2, where Y (located in the last column) refers to the total weighted mean completion time.

Table 1. DoE factors and parameters

Factors	Level 1 (-1)	Level 2 (+1)
Machines (M)	2	4
Batches (B)	40	120
Weight to Processing (W)	4%	20%
Parameter α	0.02	0.18

Table 2. Experiments' designs and their results

Run	W	B	M	α	$W\times B$	$W\times M$	$W\times\alpha$	$B\times M$	$B\times\alpha$	$M\times\alpha$	$W\times B\times M\times\alpha$	Y
1	-1	-1	-1	-1	1	1	1	1	1	1	1	41444.35
2	1	-1	-1	-1	-1	-1	-1	1	1	1	-1	9507.56
3	-1	1	-1	-1	-1	1	1	-1	-1	1	-1	476962.6
4	1	1	-1	-1	1	-1	-1	-1	-1	1	1	113899.5
5	-1	-1	1	-1	1	-1	1	-1	1	-1	-1	20487.32
6	1	-1	1	-1	-1	1	-1	-1	1	-1	1	4685.78
7	-1	1	1	-1	-1	-1	1	1	-1	-1	1	218125.8
8	1	1	1	-1	1	1	-1	1	-1	-1	-1	48722.11
9	-1	-1	-1	1	1	1	-1	1	-1	-1	-1	41401.73
10	1	-1	-1	1	-1	-1	1	1	-1	-1	1	10251.83
11	-1	1	-1	1	-1	1	-1	-1	1	-1	1	481516.9
12	1	1	-1	1	1	-1	1	-1	1	-1	-1	116288.5
13	-1	-1	1	1	1	-1	-1	-1	-1	1	1	20534.43
14	1	-1	1	1	-1	1	1	-1	-1	1	-1	4754.92
15	-1	1	1	1	-1	-1	-1	1	1	1	-1	221890.4
16	1	1	1	1	1	1	1	1	1	1	1	50388

Table 3. Factors' coefficients

Constant	W	B	M	α	$W\times B$	$W\times M$	$W\times\alpha$	$B\times M$	$B\times\alpha$	$M\times\alpha$	$W\times B\times M\times\alpha$
117553	-72741	98420	-43855	824	-60908	26180	-216	-37337	722	-131	52

4.4.2 Analysis and interpretation of the results

A 95% confidence interval analysis was conducted using regression, and the factors coefficients are shown in Table 3.

The following can be observed from the regression results:

- R-squared = 0.978; this is a very good value, indicating the success of the regression in predicting the values of the dependent variable Y (the total weighted mean completion time) within the sample. Also the adjusted R-squared was equal to 0.92.
- At a significance level of 0.05 and from the t-statistics and p-values reported (less than 0.05), we can reject the null hypothesis that all (or any) of the regression coefficients are zero.
- All the factors coefficient signs are as expected, signifying the model's logic and soundness.
- The generated prediction model from the regression was successful in estimating the completion time.
- The α coefficient is positive, indicating that when α level is -1 (α = 0.02), the completion time will decrease. On the other hand, when α level is +1 (α = 0.18), the completion time will increase, i.e. the solution would worsen.

The above observations indicated two essential conclusions: the first is the soundness and validity of the implemented Design of Experiments, and the second is that α value of 0.02 is in fact a suitable parameter to be used in all of the proposed problem combinations.

5. MODEL VERIFICATION

Verification is the process of ensuring that the simulation model behaves in the way it was intended according to the modeling assumptions made (Kelton et al., 2004).

Different methods were applied in verifying the behavior of our models:

1. We used first deterministic data instead of stochastic data for both the processing and setup times; this allowed us to predict the system's behavior.
2. We let only a single entity enters the system, and then followed this entity through all the decisions nodes to ensure that the model's logic is correct.
3. We monitored the model's animation, which made it easier to detect any errors in our logic.
4. Finally, we put several variable animations, which enabled us to determine which batch number is first scheduled, and which batch is separated.

6. COMPUTATIONAL TESTS

Simulation is one of the best approaches to deal with stochastic problems and therefore, the heuristics discussed above were implemented and compared using the simulation software Arena 7.0 from Rockwell Software. The popularity of simulation has been increasing over the past decade mainly due to its ability to deal with very complicated models of correspondingly complicated systems (Kelton et al., 2004). Another advantage of stochastic simulation is its ability to provide the user with an assessment of the robustness of the model, due to the fact that randomness is taken into account; after all, most actual systems are unlikely to work under ideal deterministic conditions, but rather in a stochastic uncertain environment (Reuter and Hulsmann, 2000). The jobs' processing times and machines' setup times are

stochastic; the processing times can take any value of four different uniform distributions: U[55, 75], U[35, 65], U[45, 70], and U[70, 90], and the setup times can take any value of the following distributions: U[6, 10], U[4, 9], U[3, 8], and U[1, 7]. Recall that these values are not known until the job is actually started on the machine; this is how the algorithms' robustness is being tested. The reason uniform distributions were used is due to their high variances, ensuring that the presented heuristics are being tested under unfavorable conditions (Weng et al., 2001). The jobs' input weights were discrete values that were randomly generated between 1 and 5.

The previous four heuristics were tested under 2, 3, and 4 unrelated parallel machines, as well as different job and batch size combinations. The number of replications for each of the problem instances was equal to 50 replications. The number of replications was obtained using the approach presented by Kelton et al. (2004) in order to obtain good confidence intervals, where the simulation was initially run for 10 replications. The half width obtained was fairly large, and therefore more runs were needed to reach a tolerable half width. Eq. (4) was used to determine the number of runs:

$$k \cong k_0 \times \frac{h_0^2}{h^2} \tag{4}$$

where k_0 and h_0 refer respectively to the initial replication number (10) and its associated half width, h refers to the desired (tolerable) half width, and k is the number of needed replications ($k = 50$)

7. OUTPUT ANALYSIS

The results obtained from running 50 replications at 95% confidence level are shown in Table 4, where the Batch Size column indicates the number of jobs included in a batch. The relative performance was calculated as follows:

$$\text{Relative Performance} = Z_l / Z_{min}, \text{ for } l = 1, 2, 3, \text{ and } 4, \tag{5}$$

where Z_l refers to the weighted mean completion time obtained when using heuristic l, and Z_{min} refers to the minimum weighted mean completion time among all 4 heuristics.

Table 4. Results from computational experiments

Machines	Jobs	Batch Size	*WSPT*	*MWP*	*Weng's Algorithm*	*PMWP*
2	80	1	1.10181	1.0766	1.035245	1
	120	1	1.10236	1.08625	1.022129	1
	480	4	1.04673	1.04532	1.004147	1
		10	1.01399	1.0103	1.001289	1
	1200	10	1.04714	1.04231	1.005242	1
		15	1.04254	1.04156	1.005776	1
3	80	1	1.10912	1.10911	1.024873	1
	120	1	1.11884	1.12231	1.026609	1
	480	4	1.09338	1.09781	1.002877	1
	1200	10	1.0874	1.09518	1.00273	1
		15	1.09747	1.09655	1.003905	1
4	80	1	1.124	1.12198	1.016727	1
	120	1	1.13969	1.14115	1.028273	1
	480	4	1.10445	1.12529	1.002489	1
	2400	20	1.11831	1.13627	1.006042	1

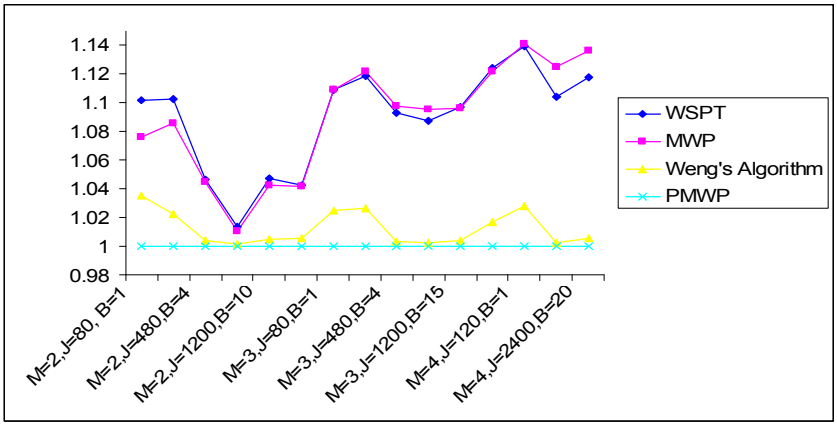

Figure 2. Comparison of the 4 algorithms.

As the number of machines increases, the load on each machine decreases. Therefore, for the problem instances of 4 machines, the size of batches was increased to 20. Table 4 clearly confirms that *PMWP* significantly outperformed the other algorithms for all experiments. Even when the number of jobs per batch is one, which changes the problem from batch scheduling to job scheduling (because every job is a batch now), *PMWP* still reached the lowest weighted mean completion time. *Weng's Algorithm* was the second best and it outperformed the other two algorithms. These results imply that assigning jobs directly to machines without arranging them in a predetermined order results in better results. This is a valid reasoning because when we sort the jobs ahead of time, it is very difficult to predict the setup times as we do not know the jobs' sequence on each machine. On the other hand, when we are assigning jobs from the unscheduled ones directly before they are processed, we already know which jobs exist on the machines; hence we know the jobs' sequence on each machine and their associated setup times.

It can be noted from Figure 2 that the larger the problem size, the better *PMWP* performs when compared to the other rules. Even though *Weng's Algorithm* approaches *PMWP* in some problem instances, it never performed better. Moreover, as the problem size increases, it can be noticed that *Weng's algorithm* departs from *PMWP*.

As we are comparing different models or logics for the same problem, output analysis becomes crucial to ensure the soundness of the results obtained. Even though the results stated in Table 4 clearly indicate the superiority of *PMWP*, simulation output analysis was conducted to compare between *PMWP* and *Weng's Algorithm* as it was the second best. Both algorithms were compared under 40 jobs (each batch has one job only) and 4 machines. The reason for having a batch size of one is to be as fair as possible to *Weng's Algorithm*, which was developed for job scheduling and not batch scheduling. We ran both models for 100 replications each, and the outputs were studied through Arena Output Analyzer, which calculates the mean difference between the two algorithms to test the following null hypothesis:

$$H_0: \text{Mean}_{PMWP} - \text{Mean}_{Weng's\ Algorithm} = 0.$$

The obtained difference was negative, confirming that *PMWP* leads to smaller completion time than *Weng's Algorithm*. To see if the obtained difference is statistically significant (because of the stochastic input, we need to ensure that the difference is far from zero in order to draw sound conclusions), the output analyzer gives 95% confidence interval on the expected difference and H_0 is rejected, concluding that *PMWP* significantly performs better than *Weng's Algorithm*.

The data and results for all four heuristics are available at SchedulingResearch (2005) for other researchers to compare any newly developed solution methodologies for this problem.

8. CONCLUSIONS

In this paper, we have introduced an effective heuristic algorithm, *PMWP*, for minimizing the total weighted mean completion time on unrelated parallel machines with sequence dependent setup times. *PMWP* was compared to three other algorithms, including *Weng's Algorithm* 7 in Weng et al. (2001). All four algorithms were modeled and tested through simulation, and our conclusions were drawn using a large number of replications and several statistical tests. These tests revealed that the algorithms are extremely fast as they found solutions for 120 batches and 4 machines in a small fraction of a second. Computational experiments also showed that *PMWP* significantly outperformed the other algorithms, especially as the number of jobs increased. Moreover, we were able to draw the conclusion that in problems dealing with unrelated parallel machines with setup times and the objective of minimizing the total weighted mean completion time, it is better to schedule the jobs directly to the machines according to some rule rather than sorting them in a predetermined order.

It is worth noting here that the four heuristics presented in this paper were also tested in a deterministic environment, and the results obtained were similar to the stochastic environment in the sense that *PMWP* significantly outperformed the other algorithms, and *Weng's algorithm* was the second best.

REFERENCES

1. Akkiraju, R., Murthy, S., Keskinocak, P., and Wu, F. (1998). Multi machine scheduling: an agent-based approach. *Proceedings of Innovative Applications of Artificial Intelligence*, 1013-1018.
2. Allahverdi, A., Gupta, J.N.D., and Aldowaisan, T. (1999). A review of scheduling research involving setup considerations. *Omega*, 27: 219-39.
3. Allahverdi, A., Ng, C.T., Cheng, T.C.E., and Kovalyov, M.Y. (2006). A survey of scheduling problems with setup times or costs. *European Journal of Operational Research* (to appear).
4. Al-Salem, A. (2004). Scheduling to minimize makespan on unrelated Parallel machines with sequence dependent setup times. Engineering *Journal of the University of Qatar*, 17: 177-187.
5. Arnaout, J-P. and Rabadi, G. (2005). Minimizing the total weighted completion time on unrelated parallel machines with stochastic times. *Proceedings of the 2005 Winter Simulation Conference*, Orlando: 2141-2147.
6. Brucker, P., Kovalyov, M., Shafransky, Y., and Werner, F. (1998). Batch scheduling with deadlines on parallel machines. *Annals of Operations Research*, 83: 23-40.
7. Bruno, J.L., Downey, P.J., and Frederickson, G.N. (1981). Sequencing tasks with exponential service times to minimize the expected flow time or makespan. *Journal of the ACM*, 28: 100-113.
8. Cheng, T.C.E., Janiak, A., and Kovalyov, M.Y. (2001). Single machine batch scheduling with resource

dependent setup and processing times. *European Journal of Operational Research*, 135: 177-183.
9. Dietrich, B.L. (1989). *A two-phase heuristic for scheduling on parallel unrelated machines with setups*, IBM TJ Watson Research Center, York Town Heights, NY: RC 14330.
10. Elmaghraby, S.E., Guinet, A., and Schellenberger, K.W. (1993). *Sequencing on parallel processors: an alternate approach*, OR Technical Report No. 273, Raleigh, NC: North Carolina State University.
11. Fisher, R.A. (1960). *The Design of Experiments*. Hafner Publishing Company, New York.
12. Ghirardi, M. and Potts, C.N. (2005). Makespan minimization for scheduling unrelated parallel machines: a recovering beam search approach. *European Journal of Operational Research*, 165: 457-467.
13. Glazebrook, K.D. (1979). Scheduling tasks with exponential service times on parallel machines. *Journal of Applied Probability*, 16: 685-689.
14. Guinet, A. (1990). Textile production systems: a succession of non-identical parallel processor shops. *Journal of the Operational Research Society*, 42: 655-671.
15. Karp, R.M. (1972). *Reducibility among combinatorial problems. Complexity of Computer Communications*, Plenum Press, New York.
16. Kelton, D., Sadowski, R., and Sturrock, D. (2004). *Simulation with Arena*, 3rd ed, McGraw-Hill Companies, New York.
17. Kim, D-W., Kim, K-H., Jang, W., and Chen, F. (2002). Unrelated parallel machine scheduling with setup times using simulated annealing. *Robotics and Computer Integrated Manufacturing*, 18: 223-231.
18. Kim, D-W., Na, D., and Chen, F. (2003). Unrelated parallel machine scheduling with setup times and a total weighted tardiness objective. *Robotics and Computer Integrated Manufacturing*, 19: 173-181.
19. Liaw, C., Lin, Y., Cheng, C., and Chen, M. (2003). Scheduling unrelated parallel machines to minimize total weighted Tardiness. *Computers and Operations Research*, 30: 1777-1789.
20. Low, C. (2005). Simulated annealing heuristic for flow shop scheduling problems with unrelated parallel machines. *Computers and operations research*, 32: 2013-2025.
21. Mohring, R., Shulz, A., and Uetz, M. (1999). Approximation in stochastic scheduling: the power of LP-based priority policies. *Journal of the ACM* (*JACM*), 46: 924-942.
22. Mosheiov, G. and Sidney, J. (2003). Scheduling with general job-dependent learning curves. *European Journal of Operational Research*, 147: 665-670.
23. Pinedo, M. (1995). *Scheduling: Theory, Algorithms, and Systems*, Prentice Hall international series in industrial and systems engineering, New Jersey.
24. Potts, C. and Kovalyov, M.Y. (2000). Scheduling with batching: A review. *European Journal of Operational Research*, 120: 228-249.
25. Pourbabai, B. (1985). One stage scheduling of preemptive jobs on parallel machines with setup times and due dates. *Proceedings of American Institutions of Industrial Engineering*, *Annual Conference Convention*, 258: pp.525-528.
26. Rabadi, G., Moraga, R., and Al-Salem, A. (2006). Heuristics for the Unrelated Parallel Machine Scheduling Problem with Setup Times. To appear in: *Journal of Intelligent Manufacturing*; 17.
27. Reuter, R. and Hulsmann, J. (2000). Achieving design targets through stochastic simulation. *Proceedings of the Madymo Users' Conference*, Paris. Available: http://www.easi.de/company/publications/mad_2000/mad_2000.pdf.
28. Ross, P. (1996). *Taguchi Techniques for Quality Engineering*, McGraw Hill: New York.
29. SchedulingResearch. (2005). Data and solutions for scheduling problems. Available: http://www.schedulingresearch.com/.
30. Schwindt, C. and Trautmann, N. (2000). Batch scheduling in process industries: an application of resource-constrained project scheduling. *OR Spectrum*, 22: 501-524.
31. Taguchi, G. (1993). *Taguchi Methods: Design of Experiments*, American Supplier Institute, Inc., Michigan.
32. Weber, R.R., Varaiya, P., and Walrand, J. (1986). Scheduling jobs with stochastically ordered processing times on parallel machines to minimize expected flowtime. *Journal of Applied Probability*, 23: 841-847.
33. Weiss, G. (1992). Turnpike optimality of Smith's rule in parallel machines stochastic scheduling. *Mathematics of Operations Research*, 17: 255-270.
34. Weiss, G., and Pinedo, M. (1980). Scheduling tasks with exponential service times on non-identical processors to minimize various cost functions. *Journal of Applied Probability*, 17: 187-202.
35. Weng, M., Lu, J., and Ren, H. (2001). Unrelated parallel machine scheduling with setup consideration and a total weighted completion time objective. *International Journal of Production Economics*, 70: 215-226.