Proceedings of the 2005 Winter Simulation Conference
M. E. Kuhl, N. M. Steiger, F. B. Armstrong, and J. A. Joines, eds.

MINIMIZING THE TOTAL WEIGHTED COMPLETION TIME ON UNRELATED PARALLEL MACHINES WITH STOCHASTIC TIMES

Jean-Paul M. Arnaout
Ghaith Rabadi

Engineering Management and Systems Engineering Department
247 Kaufman Hall
Old Dominion University
Norfolk, VA 23529, U.S.A.

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 weighted mean completion time. Identical jobs are batched together and are available at time zero. Processing time of each job of a batch is determined according to both the machine it will be assigned to and the batch group to which the job belongs. 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, a solution heuristic is 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.

1 INTRODUCTION

In this paper we compare different heuristics for the problem of scheduling a set of independent batches (each batch is a group of identical jobs) on a set of unrelated parallel machines.

For several years now, there has been significant research involving scheduling in batches, 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; a lot 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. In this paper we address the concept of batch availability.

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. 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 will be assigned an input weight w_i indicating its importance, and each batch of jobs has the same processing time and input weight. The machine setup times are dependent only on job sequences and are machine independent.

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, Downey, and Frederickson 1981). Moreover, what differentiates this paper from most of the previous literature is the use of stochastic processing and setup times, ensuring a better depiction of the real world. Discrete event simulation will be used to model and test the problem addressed in this paper. Different heuristics will be compared using the mean weighted completion time objective; the heuristic with the lower objective function value will be the superior heuristic in this specific problem.

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

There is a lot of literature on parallel machine scheduling. The common objectives studied in this area include 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 difficulty, it became a common and acceptable practice to find suitable heuristics instead of optimal solutions for these complex scheduling problems.

Several studies discussed the unrelated parallel machine problem. Ghirardi and Potts (2004) considered the problem of scheduling jobs on unrelated parallel machines to minimize the makespan. The heuristic they used was an application of the recovering beam search. Weng, Lu and Ren (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 small fraction of a second. Low (2004) 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.

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, Schulz, and Uetz (1999).

In this paper, our objective is to develop a heuristic for the unrelated parallel machine problem with the objective of minimizing the total mean weighted completion time. Previous literature have tackled this problem but hardly with stochastic inputs.

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 was assigned. Each machine is assumed to be available at time 0 and can process one job at a time. Each job has a weight (w_i) indicating its importance, where w_i has values between 1 and 5 with 1 being less urgent than 5. The machine setup times are dependent on jobs' sequence and are machine independent. In other words, setup times depend on both the batch just completed and the next batch to be processed, but there is no setup between jobs belonging to the same batch. s_{ki} is the setup time required on a machine if batch *i* is scheduled after batch *k*; *k* refers to the previous batch processed on the machine.

The batches processing times are dependent on the machine they were assigned to; job Ji has a processing time pij when it is assigned to machine Mj. For example, the processing time of J1 on machine M2 is equal to p12. 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 Ji completion time is represented by Cij, and our objective is to find a near optimal schedule that can minimize the total mean weighted completion time. This is represented as follows:

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

where η is the total number of jobs, and the completion time of job J_i on machine M_j is given by:

$$C_{ij} = C_{kj} + p_{ij} + s_{ki}.$$

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, Na, and Chen 2003). The disadvantage of such a method is manifested in low quality solutions and extensive computational time. 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 papers, especially in parallel machines' environment. This rule was proven to obtain optimal results in the single machine weighted completion time problem and very good results in the same problem but on parallel machines (Pinedo 1995).

1. {Sort the batches in the increasing order according to their processing time over weight}
   (a) Obtain the minimum processing time $\boldsymbol{\rho_i}$ for each batch: MIN(p_{i1}), MIN(p_{i2}),…, MIN(p_{iM}). where *i* is the batches' index, and *M* is the total number of machines.
   (b) Reorder the batches in the following way: $\boldsymbol{\rho_1/w_1 \le \rho_2/w_2 \le \ldots \le \rho_B/w_B}$.
2. After sorting the batches, send them one by one to the machines. Assign each batch to the machine that has the smallest [$(p_{ij} + s_{ki})/w_i$].
3. Before a batch gets processed, separate its jobs so they can be processed one by one on the assigned machine.
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.

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, Na, and Chen (2003) noted that the weight component γ is represented as the following:

$$\gamma = [1 - (\text{weight control parameter}) \times (w_i)], \quad (1)$$

where the weight control parameter $\boldsymbol{\alpha} \in (0, 0.2)$; the selection of this range is explained in section 4.4. Due to its high-quality results, we decided to manipulate the EWDD rule so it can be used in our problem. $\boldsymbol{\alpha}$ value was determined to be 0.1 for the total tardiness minimization problem (Kim, Na, and Chen 2003); empirical tests showed that this value is also the best when used in this heuristic for the problem in hand. The updated rule that we will refer to as minimum weighted processing time (*MWP*) is calculated by multiplying the minimum processing time $\boldsymbol{\rho_i}$ of each batch by the attuned weight parameter.

1. {sorting the batches}
   (a) Obtain the minimum processing time $\boldsymbol{\rho_i}$ for each batch.
   (b) Calculate for each job its MWP:

$$MWP_i = \boldsymbol{\rho_i} \times [1.0 - (0.1) \times (w_i)].$$

   (c) Reorder the batches from the smallest MWP to the largest.
2. *Step 2, 3* and *4* are exactly like in *Heuristic 1*.

4.3 Heuristic 3 (Weng's Algorithm)

Weng et al. (2001) studied the problem of unrelated parallel machine scheduling with setup consideration 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.

4.4 Heuristic 4 (PMWP)

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

Let *S* be a set containing the unscheduled batches.

1. Find batch i and machine j where the Equation (2) is at its minimum:

$$[Ckj + (pij + ski) \times (1 - (\alpha * wi))], \quad (2)$$

2. where i ϵ S (index of unscheduled batches), j is the machine index, and k is the previous batch on that specific machine j.
3. In Equation (2) above, Ckj refers to the completion time of the last batch on machine j, and the control parameter α value was determined from Figure 1 below.
4. After finding both i and j, assign batch i to machine j, and remove batch i from list S.
5. If $S = \emptyset$, STOP; else go to *Step 1*.

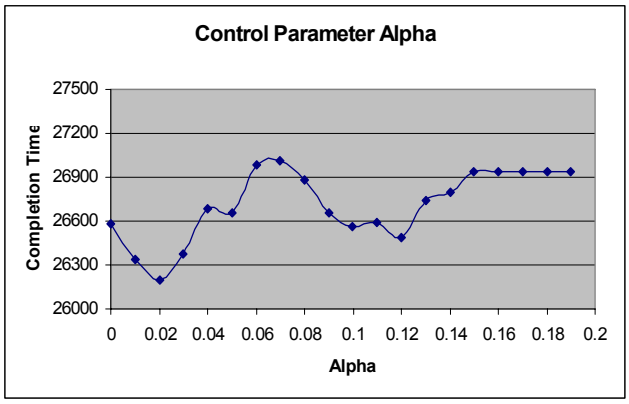

Figure 1: Control Parameter α

In Figure 1, we included a chart describing how the completion times of batches were fluctuating when **α** was changed while applying *PMWP* to the problem at hand. Recall that the values of **α** are between 0 and 0.2; it cannot be 0 because Equation (1) will then be equal to $(1 - (0 \times w_i) = 1$, meaning that the weight will not be considered in our decision. Also **α** cannot be 0.2 because w_i could be anywhere from 1 to 5; so in the case $w_i = 5$, Equation (1) 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 **α** = 0.02, and this will be the value to be used in the proposed heuristic *PMWP*. It is worth reminding here that **α** 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*.

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

The above heuristics have been modeled and compared using the simulation software Arena. 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, Sadowski, and Sturrock 2004). The reason stochastic data was used for the processing and setup times is to ensure a more real representation of a manufacturing scenario, where most of the time a job will not finish on a specific time, but on a range between two times. Simulation is considered to be one of the best approaches to deal with such source of randomness. 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, the actual system is 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 will not be known until the job is actually being processed 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 above four heuristics have been tested under 4 unrelated parallel machines, and we considered respectively

40, 80, 120, and 160 jobs. The number of replications for each of the above 4 combinations was equal to 50 replications. The number of replications was obtained by following the steps that Kelton, Sadowski, and Sturrock (2004) recommended in order to obtain good confidence intervals. We ran the simulation model for 10 replications; the half width obtained was fairly large. We decided on the tolerable half width that we want and substituted the appropriate values in the following equation:

$$n \cong n_0 \frac{{h_0}^2}{h^2},$$

where n_0 and h_0 refer respectively to the initial replication number (10) and its associated half width, h refers to the desired half width (tolerable), and n is the number of needed replications (n = 50).

7 OUTPUT ANALYSIS

The results obtained from running 50 replications of 95% confidence interval are shown in Table 1. The relative performance was calculated as follows:

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

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

As can be seen from Table 1, *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 mean weighted completion time. *Weng's Algorithm* was the second best and it outperformed the other two algorithms. These results imply that scheduling the jobs directly to the machines without arranging them in a predetermined order would lead to 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' sequences on each machine. On the other hand, when we are assigning jobs from the unscheduled ones directly before they are processed, we know which jobs already exist on the machines; hence we know the jobs' sequences on each machine and their associated setup times.

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 1 clearly show the superiority of *PMWP*, we will conduct simulation output analysis to compare between *PMWP* and *Weng's Algorithm* as it was the second best. The appropriate statistical methods will be applied to ensure that valid conclusions are drawn. This comparison will be done under 40 jobs (each batch has one job only) and 4 machines. The reason we chose 40 jobs 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; a screenshot of the output is shown in Figure 2. The output analyzer calculates the mean difference between the two algorithms as follows:

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

As can be seen in Figure 2, the difference is negative because *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. From Figure 2, you can see that this interval misses zero, and H_0 is rejected, concluding that *PMWP* performs better than *Weng's Algorithm*.

8 CONCLUSIONS

In this paper, we have introduced an effective heuristic algorithm, *PMWP*, for minimizing the total mean weighted 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. Computational experiments 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 mean weighted 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.

Table 1: Relative Performance Obtained from Computational Experiments

Number of jobs	Number of jobs per batch	WSPT	MWP	Weng's Algorithm	PMWP
40	**1**	1.1021146	1.054535	1.0072343	1
80	**2**	1.0907348	1.048003	1.0117412	1
120	**3**	1.1166439	1.079203	1.0143247	1
160	**4**	1.120799	1.083521	1.0166956	1

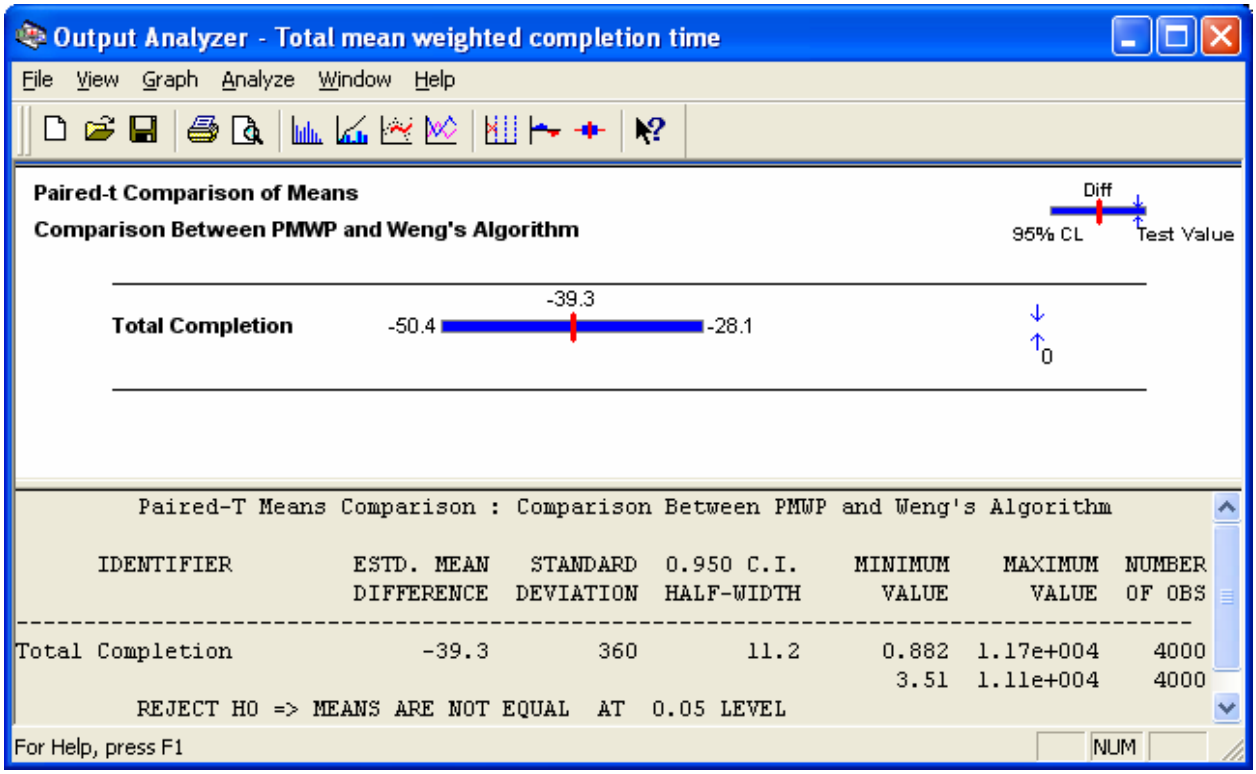

Figure 2: Arena Output Analysis

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

Akkiraju, R., S. Murthy, P. Keskinocak, and F. Wu. 1998. Multi machine scheduling: an agent-based approach. In *Proceedings of Innovative Applications of Artificial Intelligence, July 1998*: 1013-1018.

Bruno, J.L., P.J. Downey, and G.N. Frederickson. 1981. Sequencing tasks with exponential service times to minimize the expected flow time or makespan. *Journal of the ACM* 28 (1): 100 –113.

Ghirardi, M., and C.N. Potts. 2004. Makespan minimization for scheduling unrelated parallel machines: A recovering beam search approach. *European Journal of Operational Research.* In Press.

Glazebrook, K. D. 1979. Scheduling tasks with exponential service times on parallel machines. *Journal of Applied Probability* 16: 685– 689.

Karp, R.M. 1972. *Reducibility among combinatorial problems*. Complexity of Computer Communications. Plenum Press, New York: 85-103.

Kelton, D., R. Sadowski, and D. Sturrock. 2004. *Simulation with Arena.* 3rd ed. McGraw-Hill Companies, New York.

Kim, D-W, K-H Kim, W. Jang, and F. Chen. 2002. Unrelated parallel machine scheduling with setup times using simulated annealing. *Robotics and Computer Integrated Manufacturing* 18: 223-231.

Kim, D-W, Na, D. & Chen, F. 2003. Unrelated parallel machine scheduling with setup times and a total weighted tardiness objective. *Robotics and Computer Integrated Manufacturing* 19: 173-181.

Liaw, C., Y. Lin, C. Cheng, and M. Chen. 2003. Scheduling unrelated parallel machines to minimize total weighted Tardiness. *Computers & Operations Research* 30: 1777-1789.

Low, C. 2004. Simulated annealing heuristic for flow shop scheduling problems with unrelated parallel machines. *Computers and operations research.* In Press.

Mohring, R., A. Shulz, and M. Uetz. 1999. Approximation in stochastic scheduling: the power of LP-based priority policies. *Journal of the ACM (JACM)* 46 (6): 924-942.

Mosheiov, G., and J. Sidney. 2003. Scheduling with general job-dependent learning curves. *European Journal of Operational Research* 147: 665-670.

Pinedo, M. 1995. *Scheduling: Theory, Algorithms, and Systems.* Prentice Hall international series in industrial and systems engineering, New Jersey.

Potts, C., and M.Y. Kovalyov. 2000. Scheduling with batching: A review. *European Journal of Operational Research* 120: 228-249.

Pourbabai, B. 1985. One stage scheduling of preemptive jobs on parallel machines with setup times and due dates. In *Proceedings of American Institutions of Industrial Engineering, Annual Conference Convention 1985,* 258: 525-528.

Reuter, R., and J. Hulsmann. 2000. Achieving Design Targets through Stochastic Simulation. In *Proceedings of the Madymo Users' Conference, Paris 2000*. Available online via < http://www.easi.de/company/publications/mad_2000/mad_2000.pdf > [accessed March 20, 2005].

Weber, R.R., P. Varaiya, and J. Walrand. 1986. Scheduling jobs with stochastically ordered processing times on parallel machines to minimize expected flowtime. *Journal of Applied Probability* 23: 841– 847.

Weiss, G. 1992. Turnpike optimality of Smith's rule in parallel machines stochastic scheduling. *Mathematics of Operations Research* 17: 255–270.

Weiss, G., and M. Pinedo. 1980. Scheduling tasks with exponential service times on non-identical processors to minimize various cost functions. *Journal of Applied Probability* 17: 187–202.

Weng, M., J. Lu, and H. Ren. 2001. Unrelated parallel machine scheduling with setup consideration and a total weighted completion time objective. *International Journal of Production Economics* 70: 215-226.

AUTHOR BIOGRAPHIES

JEAN-PAUL M. ARNAOUT is a PhD student at the Department of Engineering Management and Systems engineering at Old Dominion University. He also has been a graduate and research assistant at this department since 2003. He received a Master's degree in Engineering Management from Old Dominion University, Norfolk, Virginia, in 2003, and a bachelor's degree in Mechanical Engineering from the University of Balamand, Lebanon. His Research interests include Optimization Techniques, Simulation and Modeling, Scheduling and Rescheduling. His e-mail address is < jarna002@odu.edu >.

GHAITH RABADI is an assistant professor at the Department of Engineering Management and Systems Engineering at Old Dominion University. Prior to that, he was a visiting assistant professor at the Department of Industrial Engineering and Management Systems at the University of Central Florida. He received his Ph.D. and M.S. in Industrial Engineering from the University of Central Florida, Orlando, Florida, in 1999 and 1996 respectively. He received his B.S degree in Industrial Engineering from the University of Jordan, Amman, Jordan. He has been involved in research projects funded by NASA and Department of Homeland Security. His research interests include Operations Research, Scheduling, Simulation Modeling and Analysis, and Optimization. His e-mail address is < grabadi@odu.edu >.