Hindawi Publishing Corporation
Journal of Applied Mathematics
Volume 2014, Article ID 304808, 5 pages
http://dx.doi.org/10.1155/2014/304808

Research Article

Single Machine Predictive Scheduling Using Inserted Idle Times

Hongli Zhu and Hong Zhou

School of Economics and Management, Beihang University, Beijing 100191, China

Correspondence should be addressed to Hong Zhou; h_zhou@buaa.edu.cn

Received 28 February 2014; Accepted 22 June 2014; Published 7 July 2014

Academic Editor: S. H. Nasseri

A single machine predictive scheduling problem is considered. The primary objective is to minimize the total completion times. The predictability of the schedule is measured by the completion time deviations between the predictive schedule and realized schedule. The surrogate measure of predictability is chosen to evaluate the completion time deviations. Both of the primary objective and predictability are optimized. In order to absorb the effects of disruptions, the predictive schedule is generated by inserting idle times. Right-shift rescheduling method is used as the rescheduling strategy. Three methods are designed to construct predictive schedules. The computational experiments show that these algorithms provide high predictability with minor sacrifices in shop performance.

1. Introduction

Production scheduling is a decision making process which is related to the allocation of resources to tasks on machines for optimization of one or more scheduling objectives [1, 2]. It plays an important role in most manufacturing and service systems [3–5]. In traditional scheduling problems, uncertainties will not be considered [1, 6]. However, in real cases, disruptions are inherently existent in every manufacturing environment. Examples of such disruptions include machine breakdowns, new job arrivals, rush job arrivals, jobs cancellation, processing time changes, due date changes, and unavailability of raw materials or tools. Dealing with these disruptions in the manufacturing environment is a very important factor in the design of scheduling algorithms.

McKay et al. [7] classified uncertainties into three main categories in practical manufacturing environment: (i) complete unknowns, (ii) suspicions about the future, and (iii) known uncertainties. Complete unknowns are those about which no advance information is available. They are unpredictable. Suspicions about the future originate from the experience and intuition of the manager. Both of the two types of uncertainties are difficult to incorporate into making production plan. Known uncertainties mean that some information is available when the predictive schedule is generated. For instance, machine breakdowns belong to the known uncertainties, and their occurrence and duration can be described using probability distributions. A large number of theoretical researches and practices indicate that there are lots of parameters with exponential distributions in the industrial production and the analysis of the reliability of equipment. In this paper, we work on the same assumption that the interval between two consecutive machine breakdowns is exponentially distributed as in [4, 8–10].

As we know, machine breakdowns are common disruptions in manufacturing systems. In order to absorb the machine breakdown disruptions' effects, a predictive schedule is generated in advance by inserting idle times in the practical production system [11–13]. The scheduling process is stated as follows. An initial schedule (called predictive schedule) is generated according to the machine environment and the information of the current orders. Then the predictive schedule is executed. When the machine breakdowns occur, the predictive schedule is modified to improve the performance and reduce the impacts of the disruptions. The finial schedule (called realized schedule) is obtained after all the jobs have been processed. Generating a predictive schedule without affecting planned activities, the difference between the predictive schedule and realized schedule should be controlled while maintaining high shop performance.

We briefly discuss some works related to predictive scheduling problems. Mehta and Uzsoy [9] presented a predictive scheduling method which inserted idle times into the schedule. The scheduling problem was minimizing maximum lateness in job shop environment with random

machine breakdowns. The results showed that the predictive scheduling improved the predictability significantly with minor sacrifices in the performance. O'Donovan et al. [10] considered a predictive scheduling problem of a single machine to minimize total tardiness with stochastic machine breakdowns. Similar to that in [9, 10], Mehta and Uzsoy [4] constructed predictive scheduling of a single machine to minimize maximum lateness with release times and machine breakdowns. Liu et al. [8] considered a predictive scheduling of a single machine to minimize the total weighted tardiness with machine breakdowns. They provided a two-stage multi-population genetic algorithm. In fact, the idle time inserting methods of the predictive scheduling are generally two-stage algorithms. An initial sequence is determined without considering machine breakdowns in the first stage. The predictive schedule is generated by inserting some amount of idle times in the second stage. In this paper, we take the total completion times as the objective. We construct predictive schedules by using similar algorithms as in [4, 9, 10] but introducing the new scheme which considers the feedback of the idle times' effects.

The remainder of this paper is organized as follows. In Section 2, we present the problem formulation of the single machine scheduling problem to minimize the total completion times. In Section 3, we provide some preliminaries of the problem. We choose a surrogate measure of predictability to evaluate the completion time deviations. In Section 4, we provide three predictive schedule algorithms by inserting idle times. In Section 5, extensive experiments are conducted to compare the performance of the three algorithms. Finally, we give a summary of the paper and a discussion of future work in Section 6.

2. Problem Formulation and Notations

The problem is stated as follows. Let $N = \{1, \ldots, n\}$ denote the set of jobs to be processed on a single machine. We use p_j to denote the processing time of job j, $j = 1, \ldots, n$. The machine can process only one job at the same time and the jobs are processed without preemption. Once the processing of the job starts, the machine is occupied until the process is completed. The machine is subject to random breakdowns. The primary objective is to minimize the total completion times. In this model, we should optimize not only the primary objective, but also the total completion time deviations between the predictive schedule and realized schedule. Denote S_p as the predictive schedule and S_r as the realized schedule. Let $C_j(S_p)$ denote the completion time of job j in predictive schedule S_p and let $C_j(S_r)$ denote the completion time of job j in realized schedule S_r. We define $\sum_{j=1}^{n} C_j(S_p)$ as the total completion times in the predictive schedule S_p and $\sum_{j=1}^{n} C_j(S_r)$ as the total completion times in the realized schedule S_r. Let S_p^0 be a predictive schedule to minimize the primary objective without machine breakdowns and let S_p^p be a predictive schedule by inserting idle times.

3. Preliminaries

In this section, we develop a surrogate measure of predictability, which will be used in subsequent sections.

Let the jobs be indexed in the order in which they appear in the predictive schedule. We evaluate the predictability of a predictive schedule S_p by the total completion time deviations between the predictive schedule and realized schedule. It is calculated as

$$\sum_{j=1}^{n} \left(E_j(S_r) + D_j(S_r) \right), \tag{1}$$

where $E_j(S_r) = \max\{C_j(S_p) - C_j(S_r), 0\}$ and $D_j(S_r) = \max\{C_j(S_r) - C_j(S_p), 0\}$ [9]. Lower value of total time deviations means that the schedule has better predictability performance. Hence, it can generate predictive schedule with better predictability performance to minimize the total completion time deviations. However, it is hard to optimize predictability performance because of the random machine breakdowns. Mehta and Uzsoy [9] presented five surrogate measures of predictability to evaluate the completion time deviations. According to computational experiments, they showed that surrogate measure $M5$ provides significantly higher correlation with the expected completion time deviations than other measures. Therefore, we choose $M5$ as the surrogate measure of predictability. The surrogate measure $M5$ is defined as follows:

$$M5_j\left(S_p\right) = \max\left\{C_j\left(S_p^*\right) - C_j\left(S_p\right), 0\right\}, \tag{2}$$

where S_p^* is the schedule formed by increasing the processing time of job j by $p_j(R_m/\lambda_h)$ while maintaining the same sequence with S_p, λ_h is the mean rate at which machine breakdowns occur, and R_m is the mean repair duration.

Adiri et al. [3] and Lee and Liman [14] proved that the single machine scheduling problem to minimize the total completion times with a single breakdown under a deterministic environment is NP-complete. Hence, the predictive scheduling to optimize the total completion times and predictability with random machine breakdowns is fairly difficult to solve.

4. Heuristic Predictive Scheduling Methods

In this section, three heuristic predictive algorithms are developed. In Section 4.1, an optimized surrogate measure heuristic algorithm is designed to construct a predictive schedule. In Section 4.2, a linear programming based heuristic algorithm for the predictive schedule is provided. In Section 4.3, a feedback algorithm is presented.

In order to maintain high predictability of the schedule with minor sacrifices in primary objective performance, we use the right-shift rescheduling (RSR) method after the machine breakdowns occur. The right-shift rescheduling method implies that the jobs' starting time is right-shift to the end of the machine breakdown, while keeping the job sequence in the predictive schedule.

4.1. SPT-OSMH Algorithm. Mehta and Uzsoy [9] provided an optimized surrogate measure heuristic (OSMH) for

the predictive scheduling of a job shop to minimize the maximum lateness. In the OSMH heuristic, a predictive schedule is generated to minimize the primary objective assuming no breakdowns. Then keep the same job sequence and insert idle time into the schedule to minimize $M5$ without considering the effects on the primary objective. Using the idea of OSMH, we design the SPT-OSMH algorithm for the predictive scheduling on a single machine to minimize the total completion times. The SPT-OSMH algorithm is given as the follows.

Algorithm H1

Step 1. Generate the predictive schedule S_p^0 to minimize $\sum_{j=1}^{n} C_j$ using shortest processing time first (SPT) rule assuming no machine breakdowns.

Step 2. Calculate the jobs' idle time $id_j = p_j(R_m/\lambda_h)$. Generate the predictive schedule S_p^p by inserting the idle time id_j into S_p^0 while keeping the same job sequence as in S_p^0, $j = 1, \ldots, n$.

Step 3. Use right-shift rescheduling method when machine breakdown occurs.

4.2. Linear Programming Based Algorithm. Mehta and Uzsoy [4] presented a linear programming based heuristic for the predictive scheduling on a single machine with release times to minimize maximum lateness. In the linear programming based heuristic, a predictive schedule is also generated to minimize the primary objective assuming no breakdowns. But the amount of inserted idle time is constrained by a linear programming to control the realized schedule primary objective degradation. Using the idea of linear programming based algorithm, we provide H2 algorithm for the predictive schedule to minimize the total completion times.

Algorithm H2

Step 1. Generate the predictive schedule S_p^0 to minimize $\sum_{j=1}^{n} C_j$ using shortest processing time first (SPT) rule assuming no machine breakdowns.

Step 2. Compute the completion time by linear programming (LP) while keeping the same sequence as in S_p^0. The predictive schedule S_p^p is obtained.

Step 3. Use right-shift rescheduling method when machine breakdown occurs:

$$\text{LP: min} \quad \sum_{j=1}^{n} \max\left\{C_j(\text{H1}) - C_j\left(S_p^p\right), 0\right\}, \tag{3}$$

$$\text{s.t.} \quad C_j\left(S_p^p\right) - C_{j-1}\left(S_p^p\right) \geq p_j, \quad j = 1, 2, \ldots, n$$

$$C_0\left(S_p^p\right) = 0, \tag{4}$$

$$\sum_{j=1}^{n} C_j\left(S_p^p\right) \leq \sum_{j=1}^{n} C_j\left(S_p^0\right) + \alpha\left(\sum_{j=1}^{n} C_j(\text{H1}) - \sum_{j=1}^{n} C_j\left(S_p^0\right)\right). \tag{5}$$

Let $C_j(\text{H1})$ denote the completion time of job j in the schedule which is obtained by Algorithm H1. Constraint (3) guarantees the precedence relationship. Constraint (5) controls the degradation in realized schedule. We define α as the control parameter.

4.3. Feedback Algorithm. The initial sequences generated by Algorithms H1 and H2 are both optimizing the total completion times without considering the random machine breakdowns. The sequence of the predictive schedule will not change after inserting the idle times in the schedule. In order to consider the feedback effects of the idle time on the initial sequence, we present the feedback Algorithm H3.

Algorithm H3

Step 1. Generate the predictive schedule S_p^0 to minimize $\sum_{j=1}^{n} C_j$ using shortest processing time first (SPT) rule assuming no machine breakdowns.

Step 2. Compute the idle time re_id_j of job j, $j = 1, 2, \ldots, n$ by the LP optimization of Algorithm H2.

Step 3. Insert re_id_j into the schedule S_p^0. Schedule the jobs in nondecreasing order of $p_j + re\_id_j$ and generate the new predictive schedule S_p^p.

Step 4. Use right-shift rescheduling method when machine breakdown occurs.

In the next section, computational experiments are conducted for the predictive scheduling problem.

5. Experimental Results

In order to examine the performance of the predictive schedule, a series of computational experiments using randomly generated test problems are conducted. These algorithms are coded in Matlab and run on an Intel Core Quad PC with 2.66 GHz CPU and 4.0 GB RAM. The test instances are generated as in [4]. There are six levels of job numbers; $n = 10, 30, 50, 70, 90, 110$. The processing times are from two discrete uniform distributions, where $P1$ = uniform (1,11) and $P2$ = uniform (4,8). Therefore, we have a total of 12 problem parameter combinations. For each problem parameter combination, 20 instances are generated. There are a total of 240 instances in the problem set (see Table 1).

The time between machine breakdowns is exponentially distributed with mean $\theta E[p_j]$, where $E[p_j]$ is the expected job processing time and $\theta = 10, 5, 3$. Higher value of θ indicates less frequent machine breakdowns. The machine breakdown durations are generated from a uniform distribution between

Table 1: Problem parameter.

Parameter	Value	Number of values
Number of jobs	$n = 10, 30, 50, 70, 90, 110.$	6
Processing time	$P1$ = uniform(1, 11) $P2$ = uniform(4, 8)	2
	Total combinations	12
	Problem combination	20
	Total problems	240

Table 2: Type of machine breakdown.

Type of machine breakdown B	Time between breakdowns exponential $\theta E[p_j]$	Breakdown durations uniform $[\beta_1 E[p_j], \beta_2 E[p_j]]$
B1	$\theta = 10$	$(\beta_1, \beta_2) = (0.1, 0.5)$
B2	$\theta = 5$	$(\beta_1, \beta_2) = (0.1, 0.5)$
B3	$\theta = 3$	$(\beta_1, \beta_2) = (0.1, 0.5)$
B4	$\theta = 10$	$(\beta_1, \beta_2) = (1, 2)$
B5	$\theta = 5$	$(\beta_1, \beta_2) = (1, 2)$
B6	$\theta = 3$	$(\beta_1, \beta_2) = (1, 2)$

$\beta_1 E[p_j]$ and $\beta_2 E[p_j]$. Six types of machine breakdown parameter combinations are generated (see Table 2). Let the control parameter α be 0.5. Therefore, we have 240 instances subject to 6 types of machine breakdowns and a total of 1440 combinations of the problem and breakdown type.

For each instance, predictive schedules are generated by Algorithms H1, H2, and H3. We simulate the execution of each predictive schedule 30 times to calculate the average completion time deviation and average realized schedule total completion times using the right-shift rescheduling method when machine breakdowns occur. Let $ACD_{(H,Q)}$ and $AC_{(H,Q)}$ denote the average completion time deviation and average realized schedule $\sum_j C_j$ for problem Q using predictive Algorithm H. Denote $ACD_{(PRS,Q)}$ and $AC_{(PRS,Q)}$ as the average completion time deviation and average realized schedule $\sum_j C_j$ for problem Q using predictive-reactive algorithm (i.e., schedule the jobs in SPT order without inserting idle times) and right-shift rescheduling method. We define $ACI(H, \delta)$ as the average percentage total completion time deviation improvement for the same problem class δ and $ACD(H, \delta)$ as the average percentage total completion times improvement for the same problem class δ (the positive value indicates improvement while the negative value means degradation). The problem class δ is defined as a set of instances, in which some parameter has a fixed value. Let (B, n, P) denote the problem class where B represents the breakdown type, n represents the number of jobs, and P represents the processing times. Let $*$ denote all possible values of a parameter. For instance, $(*, *, P1)$ indicates the set of all instances with processing time $P1$:

$$ACI(H, \delta) = \frac{\sum_{Q \in \delta} ACD_{(PRS,Q)} - \sum_{Q \in \delta} ACD_{(H,Q)}}{\sum_{Q \in \delta} ACD_{(PRS,Q)}},$$
$$ACD(H, \delta) = \frac{\sum_{Q \in \delta} AC_{(PRS,Q)} - \sum_{Q \in \delta} AC_{(H,Q)}}{\sum_{Q \in \delta} AC_{(PRS,Q)}}. \quad (6)$$

Table 3: ACI values for the problem.

δ	Algorithm H1	H2	H3
Breakdown type			
(B1, *, *)	0.9976	0.9457	0.9455
(B2, *, *)	0.9874	0.7729	0.7720
(B3, *, *)	0.8603	0.4869	0.4857
(B4, *, *)	0.9992	0.9858	0.9852
(B5, *, *)	0.9986	0.9577	0.9563
(B6, *, *)	0.9961	0.8308	0.8282
Number of jobs			
(*, 10, *)	0.8401	0.5859	0.5466
(*, 30, *)	0.9464	0.7515	0.7388
(*, 50, *)	0.9670	0.7976	0.7942
(*, 70, *)	0.9740	0.8108	0.8106
(*, 90, *)	0.9787	0.8332	0.8326
(*, 110, *)	0.9802	0.8350	0.8337
Processing time			
(*, *, $P1$)	0.9704	0.8181	0.8160
(*, *, $P2$)	0.9801	0.8270	0.8257
Overall			
(*, *, *)	0.9757	0.8229	0.8213

Table 4: ACD values for the problem.

δ	Algorithm H1	H2	H3
Breakdown type			
(B1, *, *)	−0.166382	−0.061220	−0.060909
(B2, *, *)	−0.100296	−0.014628	−0.014423
(B3, *, *)	−0.028129	−0.001218	−0.001198
(B4, *, *)	−0.514067	−0.306969	−0.306278
(B5, *, *)	−0.423533	−0.184976	−0.184379
(B6, *, *)	−0.284821	−0.049659	−0.049741
Number of jobs			
(*, 10, *)	−0.399131	−0.151821	−0.148474
(*, 30, *)	−0.416704	−0.135519	−0.133822
(*, 50, *)	−0.427604	−0.134320	−0.133558
(*, 70, *)	−0.428795	−0.132986	−0.132273
(*, 90, *)	−0.431004	−0.129646	−0.129444
(*, 110, *)	−0.431383	−0.129731	−0.129434
Processing time			
(*, *, $P1$)	−0.423863	−0.127176	−0.126877
(*, *, $P2$)	−0.434810	−0.134043	−0.133486
Overall			
(*, *, *)	−0.429880	−0.130951	−0.130510

Tables 3 and 4, respectively, show $ACI(H, \delta)$ and $ACD(H, \delta)$ values for various problem classes. The running times of the algorithms are not reported since the majority of the instances are finished in a few seconds. According to Tables 3 and 4, we can draw the conclusions as follows.

(i) The predictability of the predictive schedules generated by Algorithms H1, H2, and H3 improved significantly over those without inserting idle time. Meanwhile, inserting idle time sacrifices minor shop performance.

(ii) The predictive schedule obtained by Algorithm H2 effectively controls the objective degradation, which is obviously better than that by Algorithm H1. When the types of machine breakdown are B1, B4, and B5, the predictive approach H2 yields substantial predictability improvements over PRS at the cost of very slight degradation in the performance of realized schedule $\sum_j C_j$. Moreover, the greater number of jobs indicates higher predictability of the predictive schedule.

(iii) The predictability and shop performance generated by Algorithm H3 are both better than that by H2. It shows that the predictive schedule constructed by feedback based algorithm which considers the effects of idle time on the initial sequence has great advantages in predictability and shop performance.

6. Conclusions

In this paper, we address a single machine predictive scheduling using idle times. From the performance and stability measure, we optimize both the total completion times and the predictability of the schedule.

As the predictive schedule plays important role in the manufacturing system, it should be generated to serve as a basis for planning activities. We provide three heuristic predictive scheduling algorithms for the single machine scheduling problem. The key of the heuristic is the strategies of inserting idle times. The experiment results show that predictive schedule provides significant improvement in predictability with minor sacrifices in shop performance. The feedback algorithm has advantages in both predictability and shop performance.

For future research, we can consider several extensions of the predictive scheduling. First, it is interesting to develop algorithms for scheduling problems in complex machine environment, such as flow shop and open shop machine environment. Second, we can consider other disruptions. It is possible to deal with new job arrivals, rush job arrivals, processing time changes, and due date changes in the predictive scheduling problems. Furthermore, probabilistic analysis can be employed to get the length of idle time.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

The authors would like to thank the reviewers for their constructive suggestions and kind help. This research is supported by the Natural Science Foundation of China (Grants nos. 71071008 and 91224007), China Scholarship Council, and the Innovation Foundation of BUAA for PhD Graduates.

References

[1] P. Brucker, *Scheduling Algorithms*, Springer, Berlin, Germany, 5th edition, 2006.

[2] M. Pinedo, *Scheduling: Theory, Algorithms and Systems*, Prentice-Hall, Englewood Cliffs, NJ, USA, 1995.

[3] I. Adiri, J. Bruno, E. Frostig, and A. H. G. Rinnooy Kan, "Single machine flow-time scheduling with a single breakdown," *Acta Informatica*, vol. 26, no. 7, pp. 679–696, 1989.

[4] S. V. Mehta and R. Uzsoy, "Predictable scheduling of a single machine subject to breakdowns," *International Journal of Computer Integrated Manufacturing*, vol. 12, no. 1, pp. 15–38, 1999.

[5] X. Rong, Y. Lu, R. Yin, and J. Zhang, "A robust optimization approach to emergency vehicle scheduling," *Mathematical Problems in Engineering*, vol. 2013, Article ID 848312, 8 pages, 2013.

[6] D. Mou and W. Zhao, "An irregular flight scheduling model and algorithm under the uncertainty theory," *Journal of Applied Mathematics*, vol. 2013, Article ID 361926, 8 pages, 2013.

[7] K. N. McKay, F. R. Safayeni, and J. A. Buzacott, "The scheduler's knowledge of uncertainty: the missing link," in *Knowledge Based Production Management Systems*, Elsevier, Amsterdam, The Netherlands, 1989.

[8] L. Liu, H. Y. Gu, and Y. G. Xi, "Robust and stable scheduling of a single machine with random machine breakdowns," *International Journal of Advanced Manufacturing Technology*, vol. 31, no. 7-8, pp. 645–654, 2007.

[9] S. V. Mehta and R. M. Uzsoy, "Predictable scheduling of a job shop subject to breakdowns," *IEEE Transactions on Robotics and Automation*, vol. 14, no. 3, pp. 365–378, 1998.

[10] R. O'Donovan, R. Uzsoy, and K. N. McKay, "Predictable scheduling of a single machine with breakdowns and sensitive jobs," *International Journal of Production Research*, vol. 37, no. 18, pp. 4217–4233, 1999.

[11] H. Aytug, M. A. Lawley, K. McKay, and R. Uzsoy, "Executing production schedules in the face of uncertainties: a review and some future directions," *European Journal of Operational Research*, vol. 161, no. 1, pp. 86–110, 2005.

[12] D. Briskorn, J. Leung, and M. Pinedo, "Robust scheduling on a single machine using time buffers," *IIE Transactions*, vol. 43, no. 6, pp. 383–398, 2011.

[13] R. Leus and W. Herroelen, "The complexity of machine scheduling for stability with a single disrupted job," *Operations Research Letters*, vol. 33, no. 2, pp. 151–156, 2005.

[14] C. Lee and S. D. Liman, "Single machine flow-time scheduling with scheduled maintenance," *Acta Informatica*, vol. 29, no. 4, pp. 375–382, 1992.