Contents lists available at ScienceDirect

Int. J. Production Economics

journal homepage: www.elsevier.com/locate/ijpe

A heuristic for scheduling jobs on two identical parallel machines with a machine availability constraint

Xiuli Wang[a,*], T.C.E. Cheng[b]

[a] School of Economics and Management, Nanjing University of Science and Technology, Nanjing 210094, People's Republic of China
[b] Department of Logistics and Maritime Studies, The Hong Kong Polytechnic University, Hung Hom, Kowloon, Hong Kong, China

ARTICLE INFO

Article history:
Received 21 February 2014
Accepted 25 November 2014
Available online 6 December 2014

Keywords:
Scheduling
Identical parallel machines
Heuristic
Worst-case ratio bound

ABSTRACT

We study a two identical parallel-machine scheduling problem in which one machine is available to process jobs in a limited time interval while the other machine is always available over the scheduling horizon. The objective is to maximize the number of on-time jobs. As the problem is NP-hard, we develop a heuristic to tackle it by incorporating the backward adjusting and two-step look-ahead strategies into some existing heuristics for similar problems without the machine availability constraint. We show that our heuristic has a worst-case ratio bound of 4/3 and the bound is tight.

1. Introduction

We consider two identical parallel-machine scheduling to maximize the number of on-time jobs, where one machine is continuously available over the scheduling horizon while the other machine can only process jobs within a pre-determined time interval. The problem captures the common production scenario in which a manufacturing firm receives a pool of orders (jobs), each of which has a due date. The firm can either process a job in-house or subcontract it to another firm for processing. However, since the subcontractor has other order commitments, it can only offer an available time window to process the jobs of the firm. If both the firm and subcontractor use similar production technologies to process jobs, then they can be approximately considered as two plants with the same processing speed, i.e., two identical parallel machines. Although we assume that each order requires the same processing time on either the in-house machine or subcontractor's machine, different orders have different processing times due to their heterogeneity. At the beginning of the scheduling period, the firm needs to decide the orders to process in-house and to subcontract out with a view to maximizing the number of on-time jobs.

The model can be easily extended to the situation that the firm needs to pay a booking cost to the subcontractor for using the given time window. On the other hand, it is reasonable to assume that the subcontractor is willing to follow the job processing schedule specified by the firm when the job setup times are negligible or the subcontractor only simply sells its machine time to the firm.

In this paper we first consider applying a few existing heuristics for similar problems without the machine availability constraint to solve our problem. We find that they admit larger worst-case ratio bounds for our problem than their original problems. We then design a heuristic by adapting the existing heuristics and show that it admits a better and tight worst-case ratio bound of 4/3.

The remainder of the paper is organized as follows: In Section 2 we give a brief literature review. In Section 3 we formally describe the problem under study. In Section 4 we derive the worst-case ratio bounds of applying some existing heuristics for similar problems without the machine availability constraint to solve our problem. In Section 5 we design a heuristic by adapting the existing heuristics. In Section 6 we analyze the worst-case ratio bound of our proposed heuristic. We conclude the paper and suggest topics for future research in Section 7.

2. Literature review

Using the standard three-field notation for describing scheduling problems, with an extension to indicate the availability interval on one machine, we denote the problem under study as $P2|interval|\sum(1-U_j)$, where $P2$ represents the two identical parallel-machine environment, *interval* denotes that job processing is limited within an available time interval on one machine, $U_j=0$ and 1 if job J_j is on-time and late, respectively, and "$\sum(1-U_j)$" denotes the number of on-time jobs to be maximized. The problem is an extended version of the classical scheduling problem $P2||\sum(1-U_j)$, in which both machines are available to process jobs over the scheduling horizon.

Leung and Yu (1994) show that the problem $P2||\sum(1-U_j)$ is NP-hard and propose a heuristic (denoted as the *L–Y heuristic*)

[*] Corresponding author. Tel./fax: +86 25 84261056.
E-mail address: wangdu0816@163.com (X. Wang).

http://dx.doi.org/10.1016/j.ijpe.2014.12.001
0925-5273/© 2014 Elsevier B.V. All rights reserved.

with a worst-case ratio bound of 4/3. Bar-Noy et al. (2001) develop solution algorithms for variants of the more general problem $R|r_j|\sum w_j(1-U_j)$, where R denotes the unrelated parallel-machine environment, and r_j and w_j are the release time and weight of job J_j, respectively. They develop a heuristic algorithm with a worst-case ratio bound of $2+\sqrt{3}$ for arbitrary weights. They also provide a greedy algorithm (denoted as the *Bar-Noy et al. heuristic*) with a worst-case ratio bound of 2 for the case of identical weights. For the special case of $P2||\sum(1-U_j)$, the Bar-Noy et al. heuristic admits a worst-case ratio bound of 9/5, which is inferior to the 4/3 bound of the L–Y heuristic. For the problem $Q2||\sum(1-U_j)$, where $Q2$ represents the uniform parallel-machine environment, Koulamas and Kyparisis (2006) develop a heuristic (denoted as the *K–K heuristic*) by embedding a look-ahead feature in the L–Y heuristic. They prove that their heuristic has the tight worst-case ratio bound of 3/2. In addition, Leung (2004) develop heuristics for online versions of the problems $P2||\sum(1-U_j)$ and $Q2||\sum(1-U_j)$, and prove that they have worst-case ratio bounds of $2(1+1/\sqrt{6})$ and $1+1/s$, respectively, where $s\ (>1)$ represents the speed ratio of the fast machine over the slow machine. In the m parallel-machine environment, Ho and Chang (1995) propose heuristics for the problem $Pm||\sum(1-U_j)$ and conduct experiments to test their computing performance. M'Hallah and Bulfin (2005) propose a branch-and-bound algorithm to solve the problem $Pm||\sum w_j(1-U_j)$. These studies involve both the job acceptance and scheduling decisions at the same time. For detailed research results on this topic, the reader is referred to the papers of Oğuz et al. (2010) and Wang et al. (2013a, 2013b), and the survey of Slotnick (2011).

In the two parallel-machine environment, the aforementioned offline heuristics can be straightforwardly applied to deal with the problem $P2|interval|\sum(1-U_j)$. However, we show in the next section that these heuristics admit larger worst-case ratio bounds for our problem than their original problems. In this paper we develop a heuristic for solving the problem $P2|interval|\sum(1-U_j)$ by adapting the L–Y heuristic (and the K–K heuristic) and show that it has a better worst-case ratio bound than the two heuristics.

Another line of research related to our study concerns identical parallel-machine scheduling with machine maintenance constraints, which are characterized as unavailable time intervals on the machines due to preventive maintenance activities. Most results in this research stream focus on the problem to minimize the makespan or the total (weighted) completion time under the assumption that all the jobs are accepted for processing.

For the objective of minimizing the makespan, when the jobs are resumable, i.e., the processing of a job interrupted by an unavailable interval of a machine can continue its processing after the machine becomes available again, Lee (1996) considers identical parallel-machine scheduling under the assumption that there is at least one machine that is always available. He shows that the worst-case ratio bounds of list scheduling and longest processing time (LPT) scheduling are m and $(m+1)/2$, respectively, where m is the number of identical parallel machines. Hwang and Chang (1998), and Hwang et al. (2005) analyze the worst-case ratio bounds of LPT scheduling for the problem where each machine may have an unavailable interval. The former show that the tight worst-case ratio bound is 2 if the number of simultaneous unavailable machines is no more than $m/2$. The latter generalize this result and show that a worst-case ratio bound is $1+\lceil m/(m-k)\rceil/2$, where k is the maximum number of simultaneous unavailable machines and $\lceil x\rceil$ denotes the smallest integer that is no less than x. For the objective of minimizing the total completion time, Lee and Liman (1993) consider two identical parallel-machine scheduling where one machine becomes unavailable from a particular instant and the other machine is always available. They propose a shortest processing time (SPT)-based heuristic with the tight worst-case ratio bound of 3/2. Liao et al. (2009) propose a branch-and-bound algorithm to treat the problem. Tan et al. (2011) consider m identical parallel-machine scheduling where each of the first k machines has unavailable intervals, while the other $m-k$ machines are always available, where $1\le k\le m$. They prove that SPT scheduling has a worst-case ratio bound of $1+(m-1)/(m-k)$ when $k<m$. If there is exactly one unavailable interval on each of the first k machines and the unavailable intervals do not overlap, then SPT scheduling has a worst-case ratio bound of $1+(k-1)/(m-1)$. For the objective of minimizing the total weighted completion time, Zhao et al. (2011) develop a fully polynomial-time approximation scheme (FPTAS) for m identical parallel-machine scheduling under the assumptions that m is a fixed number, only one machine is unavailable in a fixed interval, and the other machines are always available. For detailed research results in this area, the reader is referred to the surveys provided by Sanlaville and Schmidt (1998), Schmidt (2000), and Ma et al. (2010). However, to the best of our knowledge, no research has considered the problem to maximize the number of on-time jobs.

3. Problem description

We formally describe the problem $P2|interval|\sum(1-U_j)$ as follows: There is a set of n jobs $N=\{J_1,J_2,\cdots,J_n\}$ that are available for processing at time zero. There are two identical parallel machines M_1 and M_2, where M_1, as the in-house machine, is always available for processing jobs from the beginning of the scheduling horizon (time zero) while M_2, as the subcontractor's machine, can only process jobs within a given time interval $[t_0, t_0+\Delta]$. Associated with each job J_j are its processing time p_j and due date d_j. We assume that preemption is not allowed, i.e., processing a job on the machine cannot be interrupted until it is finished. A job J_j is late if it is completed after its due date d_j, denoted as $U_j=1$; otherwise, the job J_j is on time, denoted as $U_j=0$. The decisions are to determine the jobs to accept for processing and to schedule the accepted jobs on either machine so as to maximize the number of on-time jobs.

The problem is NP-hard because it is an extended version of the NP-hard problem $P2||\sum(1-U_j)$. Similar to Leung and Yu (1994), we provide the following dynamic programming algorithm to solve the problem in pseudo-polynomial time. Re-index the jobs $J_1,J_2,\cdots,J_n$ such that $d_1\le d_2\le\cdots\le d_n$. Let $F(t_1,\ t_2;j)$ denote the maximum number of on-time jobs in $\{J_1,J_2,\cdots,J_j\}$ where the completion times of the last jobs on M_1 and M_2 are t_1 and t_2, respectively. Given the boundary conditions: $F(0,t_0;0)=0$, $F(t_1,t_2;0)=-\infty$ for $0<t_1\le d_n$ or $t_0<t_2\le \min\{t_0+\Delta,d_n\}$, and $F(t_1,t_2;j)=-\infty$ for $t_1<0$ or $t_2<t_0$, the forward current relation is

$$F(t_1,t_2;j)=\max\begin{cases}F(t_1-p_j,t_2;j-1)+1 & \text{if } 0<t_1\le d_j\\ F(t_1,t_2-p_j;j-1)+1 & \text{if } t_0<t_2\le \min\{t_0+\Delta,d_j\}\\ F(t_1,t_2;j-1)\end{cases}$$

The optimal solution is the largest value of $F(t_1,t_2;n)$ for all $0<t_1\le d_n$ and $t_0<t_2\le\min\{t_0+\Delta,d_n\}$. The dynamic programming algorithm is pseudo-polynomial as its time complexity is $O(nd_n^2)$. So the problem $P2|interval|\sum(1-U_j)$ is ordinary NP-hard and it is justified to develop a fast heuristic to tackle it.

The computational performance of a heuristic is often measured by its worst-case ratio bound. Let n_A and n_O denote the numbers of on-time jobs produced by a heuristic and an optimal algorithm, respectively. For the problem under study, we define the worst-case ratio bound ρ of a heuristic as the smallest number such that $n_O/n_A\le\rho$ holds for any problem instance. If there exists at least one problem instance such that $n_O/n_A=\rho$, then we say that the worst-case ratio bound of the heuristic is tight.