Computers & Industrial Engineering 91 (2016) 66–84

Contents lists available at ScienceDirect

Computers & Industrial Engineering

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

Event driven strategy based complete rescheduling approaches for dynamic *m* identical parallel machines scheduling problem with a common server

Alper Hamzadayi [a,*], Gokalp Yildiz [b]

[a] Department of Industrial Engineering, Yuzuncu Yil University, 65080 Van, Turkey
[b] Department of Industrial Engineering, Dokuz Eylül University, 35397 Izmir, Turkey

ARTICLE INFO

Article history:
Received 24 December 2014
Received in revised form 4 November 2015
Accepted 9 November 2015
Available online 14 November 2015

Keywords:
Dynamic scheduling
m identical parallel machines scheduling with a common server
Simulation
Simulated annealing
Dispatching rules
Sequence dependent setup times

ABSTRACT

This paper addresses the dynamic *m* identical parallel machine scheduling problem in which the sequence dependent setup operations between the jobs are performed by a single server. An event driven rescheduling strategy based simulation optimization model is proposed by inspiration from limited order release procedure (Bergamaschi, Cigolini, Perona, & Portioli, 1997) for being able to tackle the changing environment of the system. The proposed event driven rescheduling strategy is based on the logic of controlling the level of the physical work-in-process on the shop floor. A simulated annealing and dispatching rules based complete rescheduling approaches as the simulation based optimization tools are proposed and adapted to the developed simulation model for generating new schedules depending on the proposed event driven rescheduling strategy. The objective of this study is to minimize the length of schedule (makespan). The performances of the approaches are compared on a hypothetical simulation case. The results of the extensive simulation study indicate that simulated annealing based complete rescheduling approach produces better scheduling performance.

1. Introduction

Scheduling is the allocation of resources (e.g. machines) to tasks (e.g. jobs) in order to ensure the completion of these tasks in a reasonable amount of time; and the scheduling strategies are usually classified into two categories (Ouelhadj & Petrovic, 2009): static and dynamic.

In static scheduling, all jobs are available and ready for processing. Once the schedule is prepared, the processing sequence of jobs is determined and is not changed during processing (Fang & Xi, 1997). The static parallel machine scheduling problem is one of the most difficult classes of the scheduling problem, and a considerable number of studies have been conducted on various commercial, industrial, and academic fields. Parallel machine scheduling problems can be roughly classified into three categories (Cheng & Sin, 1990): (1) identical parallel machines, (2) uniform parallel machines, and (3) unrelated parallel machines. On the other hand, there are two common types of setup time in classical scheduling problems: sequence independent or sequence dependent (Allahverdi, Aldowaisan, & Gupta, 1999). In the first case, the setup time is usually added to the job processing time, whereas, in the second case the setup time depends not only on the job currently being scheduled but also the last scheduled job. The studies of parallel machine scheduling problems with sequence-dependent setup times to minimize makespan have attracted a special attention in recent years. A comprehensive literature reviews on solution methods for different types of parallel machine scheduling problems can be found at Allahverdi, Ng, Cheng, and Kovalyov (2008), Allahverdi et al. (1999), and Pfund, Fowler, and Gupta (2004). The parallel machine scheduling problem is *NP*-hard (Ho & Chang, 1995). Due to the complexity of the problem, it is a general practice to find an appropriate heuristic rather than an optimal solution for the parallel-machine scheduling problem (Bilge, Kirac, Kurtulan, & Pekgun, 2004; Kim, Kim, Jang, & Chen, 2002; Mendes, Muller, França, & Moscato, 2002; Ventura & Kim, 2003).

In dynamic scheduling, jobs arrive dynamically over time and are processed on machines continuously. Jobs whose operations are finished are moved out of the system continuously. There may be random and unpredictable events in the system, such as machine breakdown and repair, and the due date of jobs may be changed during processing. So, a previously feasible static schedule may turn infeasible when it is released to the shop floor (Fang & Xi,

* Corresponding author. Tel.: +90 505 279 73 77; fax: +90 432 225 17 30.
E-mail address: alperhamzadayi@yyu.edu.tr (A. Hamzadayi).

http://dx.doi.org/10.1016/j.cie.2015.11.005
0360-8352/

1997). Most scheduling problems in real-world production systems occur in such a dynamic environment.

Rescheduling is the process of updating an existing production schedule in response to unpredictable real-time events to minimize its impact on the system performance and it needs to address two issues: how and when to react to real-time events. Regarding the first issue, the literature provided two main rescheduling strategies (Cowling & Johansson, 2002; Sabuncuoglu & Bayiz, 2000; Vieira, Herrmann, & Lin, 2003), namely schedule repair and complete rescheduling. Schedule repair refers to some local adjustments of the current schedule and complete rescheduling refers to regenerating a new schedule from scratch (Ouelhadj & Petrovic, 2009). Regarding the second issue, when to reschedule, three policies have been proposed in the literature (Sabuncuoglu & Bayiz, 2000; Vieira et al., 2003): periodic, event-driven, and hybrid. Under the periodic strategy, the rescheduling occurs regularly with a constant time interval (the rescheduling period) between consecutive rescheduling events. No other events trigger the rescheduling. In event-driven strategy, the rescheduling is triggered in response to an unexpected event alters the current system status. Under the hybrid strategy, the rescheduling occurs not only periodically but also whenever an unexpected event that alters the current system status occurs.

The dynamic identical parallel machine scheduling problem with sequence-dependent setup time has been studied for many years and many solution techniques have been proposed for these problems. Firstly, Ovacik and Uzsoy (1995) presented a family of rolling horizon procedures (*RHP*s) in order to solve the dynamic m identical parallel machines scheduling problem for minimizing the maximum lateness. It was found from the results that both on average and in the worst case, *RHP*s consistently outperformed dispatching rules (*DR*s) combined with local search methods. Kim and Shin (2003) proposed a restricted Tabu search (*RTS*) approach for the same problem. Experimental results indicated that, in general, the *RTS* seems to outperform some heuristic algorithms such as the *RHP*, the basic Tabu search, and simulated annealing (*SA*); however, one drawback is the requirement of a certain level of tuning and customization. Lee, Lin, and Ying (2010) proposed a restricted *SA* (*RSA*) for solving the same problem. They reported that *RSA* algorithm is highly effective when compared to the basic *SA*. Ying and Cheng (2010) and Lin, Lee, Ying, and Lu (2011) proposed an iterated greedy heuristic for the same problem presented by Ovacik and Uzsoy (1995). Their computational results revealed that iterated greedy heuristic is highly effective when compared to state-of-the-art algorithms on the same benchmark problem data set. Yang (2009) proposed a genetic algorithm based simulation optimization approach to solve the m identical parallel machines scheduling problem under the dynamic job arrival and static machine availability constraints for minimizing makespan. More recently, Yang and Shen (2012) proposed job-driven scheduling heuristic and machine-driven scheduling heuristic to solve a practical-size the m identical parallel machines scheduling problem with stochastic failures. The reader can refer to Vieira et al. (2003) and Ouelhadj and Petrovic (2009) for the detailed survey on dynamic scheduling in manufacturing systems.

In modern manufacturing systems, some resources such as a piece of equipment or a team of setup workers or a single operator may be required throughout the setup process. Each of these situations defines a scheduling problem with a single server. The type of server may vary according to the production environment. Examples of this problem arise frequently in production environments such as manufacture of automobile components and printing industry (Huang, Cai, & Zhang, 2010). Numerous research efforts have been attempted for solving the static parallel machine scheduling problem with a single server over the years in the literature. Some examples from the literature indicate the importance of the problem in the industry can be found at Kravchenko and Werner (1997), Hall, Potts, and Sriskandarajah (2000), Glass, Shafransky, and Strusevich (2000), Guirchoun, Souhkal, and Martineau (2005), Huang et al. (2010), and Kim and Lee (2012).

Since the two identical parallel machines scheduling problem with a common server in which setup times are sequence dependent is *NP*-hard (Abdekhodaee & Wirth, 2002; Abdekhodaee, Wirth, & Gan, 2004; Koulamas, 1996; Kravchenko & Werner, 1997) therefore m identical parallel machines version of the same problem is also *NP*-hard.

When a dynamic and stochastic manufacturing environment is encountered in which static scheduling may be impractical, the use of real time scheduling approaches is required (Xiang & Lee, 2008). Simulation is a powerful tool to analyze complex, dynamic and stochastic systems and a simulation-based real time scheduling system usually consists of four main components (Yoon & Shen, 2006): "a monitoring system to collect data from the physical shop floor; a simulator to generate simulation models, run the models, and analyze their results; a decision-making system to generate decisions such as schedules and priority rules; and an execution system to control the shop floor". The reader can refer to Negahban and Jeffrey (2014) for the recently published survey on simulation approaches for manufacturing systems.

One of the common problems of the production system is that continuously arriving customer orders cannot be released to shop floor as soon as the planning system releases the job order due to the physical work-in-process (*PWIP*) constraint. The *PWIP* requires a storage space and most companies strive to keep the actual amount of the *PWIP* as low and constant as possible, so as to reduce the amount of capital tied up in the production or manufacturing process and to reduce the risk of obsolescence, especially in fast-moving sectors such as technology and consumer electronics. Therefore, the *PWIP* is an important constraint that must be taken into consideration in terms of production systems.

Before giving a concise literature review, the notations used for convenience and readability are summarized as shown in Table 1.

The three-field notation $\alpha/\beta/\gamma$ of Graham, Lawler, Lenstra, and Rinnooy Kan (1979) is used to describe a scheduling problem in the literature. The α field denotes the shop (machine) environment. The β field describes the setup information, other shop conditions, and details of the processing characteristics, which may consist of one or more entries. Finally, the γ field contains the objective to be minimized. According to the standard three-field notation, P_m,$S/ST_{sd}/C_{max}$ can be used to denote the m identical parallel machines scheduling problem with a common server and sequence dependent setup times under the objective of minimizing the makespan.

Koulamas (1996) showed that the problem of P_2,S/ST_{si} with the objective of minimizing the machine idle time resulting from the unavailability of the server is *NP*-hard in the strong sense.

Table 1
The notations used for convenience and readability.

Notation	Description
PD_m, S	m dedicated parallel machines with a common server
P_m, S	m identical parallel machines with a common server
ST_{sd}	Sequence dependent setup time
ST_{si}	Sequence independent setup time
C_{max}	Makespan
L_{max}	Maximum lateness
$\sum C_j$	Total completion time
$\sum w_j C_j$	Total weighted completion time
$\sum T_j$	Total tardiness
$\sum w_j T_j$	Total weighted tardiness
$\sum U_j$	Number of tardy jobs; if $T_j > 0 \Rightarrow U_j = 1$, 0 otherwise
$\sum w_j U_j$	Weighted number of tardy jobs