International Journal of Production Economics 201 (2018) 26–40

Contents lists available at ScienceDirect

International Journal of Production Economics

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

Exact and heuristic methods to solve the parallel machine scheduling problem with multi-processor tasks

Lingxiao Wu, Shuaian Wang *

Department of Logistics and Maritime Studies, The Hong Kong Polytechnic University, Hung Hom, Hong Kong

ARTICLE INFO

Keywords:
Scheduling
Parallel machine
Multi-processor tasks
Branch and bound
Tabu search

ABSTRACT

This paper studies a special parallel machine scheduling problem where some tasks require more than one machine to process, known as the Parallel Machine Scheduling Problem with Multi-processor Tasks. Two mathematical models and several theoretical properties are proposed for the studied problem. To solve this problem, this paper develops an exact branch and bound algorithm and a heuristic tabu search algorithm. A series of numerical experiments are conducted to test the performance of these solution methods. The computational results show that the solution methods are effective and efficient in solving the problem with different sizes.

1. Introduction

Parallel machine scheduling problems consist of assigning and sequencing a set of tasks on a set of machines. One of the core assumptions made in classical parallel machine scheduling problems is that each task needs to be processed on exactly one machine at a time. However, this assumption is insufficient to handle many problems arising from today's production realities. Namely, tasks in some scheduling problems require more than one machine to process, and we call such problems the *multi-processor task scheduling problems* (Blazewicz et al., 1986). There are a large number of industrial applications where *multi-processor task scheduling problems* occur. For instance, (1) in diagnosable microprocessor systems, a task must be processed by at least two processors at the same time (Krawczyk and Kubale, 1985); (2) in semiconductor circuit design workforce planning, a design project may need a group of people (Chen and Lee, 1999), (3) in the quay crane scheduling problem of container terminals, a ship requires more than one crane to handle (Trkoullar et al., 2014) and (4) in parallel batch jobs scheduling problems of grid computing environments, a computing task may run on several processors that work in parallel (Switalski and Seredynski, 2015).

This paper studies a Parallel Machine Scheduling problem which allows tasks to be processed on more than one machine. The considered objective in our work is to optimize the makespan (the latest completion time of all tasks). In this paper, we use the three-field notation, $\alpha|\beta|\gamma$ which is introduced by Graham et al. (1979) for parallel machine scheduling problems. In this notation: (1) the first field α specifies the machine environment, where P represents identical, Q uniform, and R unrelated parallel machines, (2) the second field β denotes the job characteristics, and the third field γ denotes the objective function. According to Drozdowski (1996)'s classification, this problem can be denoted by $P|size_j|C_{\max}$, where P represents parallel machine, $size_j$ denotes the number of machines required by task j and $C_{\max}$ indicates that the objective is to optimize the makespan.

This paper deals with the $P|size_j|C_{\max}$ problem, and the main contribution includes:

- This paper theoretically analyzes the $P|size_j|C_{\max}$ problem and presents formal formulations. Several theoretical properties along with two different mixed integer programming models of the considered problem are proposed.
- To solve problems with real-world sizes, a branch and bound (B & B) algorithm with five efficient lower bounds and two fathoming criteria is proposed. For problems with even larger scales, we develop a tabu search (TS) method.
- Numerical experiments involving 700 instances are conducted. Computing results indicate that the proposed solution methods can obtain high-quality solutions for the $P|size_j|C_{\max}$ problem with different sizes in relatively short computational time.

This paper is organized as follows. The next section gives a review of related studies. In Section 3, we formally state the $P|size_j|C_{\max}$ problem and present several properties for the optimal schedule of the problem. Two different mixed integer programming models for the problem are

* Corresponding author.
E-mail address: wangshuaian@gmail.com (S. Wang).

https://doi.org/10.1016/j.ijpe.2018.04.013
Received 19 September 2017; Received in revised form 4 February 2018; Accepted 17 April 2018
Available online 27 April 2018
0925-5273/© 2018 Published by Elsevier B.V.

formulated in Section 4. A branch and bound method is proposed in Section 5. In Section 6 we introduce a tabu search algorithm. The performances of the proposed models and algorithms are examined in Section 7 by a series of computational experiments. Finally, our findings are summarized in Section 8.

2. Literature review

Studies of the $P|size_j|C_{\max}$ problem were motivated by research in computer systems, such as fault-tolerant systems (Krawczyk and Kubale, 1985). Lloyd (1981) is among the first who studied the $P|size_j|C_{\max}$ problem. In his paper the problem in which all tasks share equal processing time, $P\left|size_j, p_j = 1\right|C_{\max}$, where p_j represents processing time is studied and it was shown that the general $P\left|size_j, p_j = 1\right|C_{\max}$ problem is NP-complete. The author also demonstrated that the performance bound of list scheduling for solving the $P\left|size_j, p_j = 1\right|C_{\max}$ problem is bounded by $(2m-k)/(m-k+1)$, where m is the number of machines and k is the largest number of machines required by one task. Blazewicz et al. (1986) considered the preemptive and nonpreemptive parallel machine scheduling problems with multiprocessor tasks. They proposed two polynomial-time algorithms with the complexity of $O(n)$ to two problems where each task requires either 1 or k machines to process ($size_j \in \{1,k\}$, where k is integer and $k \leq m$). One problem is the $P\left|size_j \in \{1,k\}, p_j = 1\right|C_{\max}$ problem and the other can be denoted by the $P|size_j \in \{1,k\}$, $pmtn|C_{\max}$ problem in which $pmtn$ implies that tasks are preemptive. Computational complexity of the $P|size_j|C_{\max}$ problem was examined by Du and Leung (1989). Their study showed that for $m = 2, 3$, the $P|size_j|C_{\max}$ problem can be solved in pseudopolynomial time but the problem becomes NP-hard in the strong sense for each $m \geq 5$. Blkadek et al. (2015) studied the differences between contiguous schedules and non-continuous schedules of the $P|size_j|C_{\max}$. In contiguous schedules, indices of the processors assigned to a task must be a sequence of consecutive numbers while in non-continuous schedules indices of the processors assigned to a task can be arbitrary. They proved that deciding whether such difference exists for a $P|size_j|C_{\max}$ instance is NP-complete and provided bounds on the difference between the length of two schedules. Besides the parallel machine setting, there are some studies that tackle machine scheduling problem with multi-processor tasks in the flow shop environment (e.g., Chou (2013); Hidri (2016)).

As for the solution methods of the $P|size_j|C_{\max}$ problem, exact solution methods can be found for some special cases of the problem, for example, when all the tasks share the same processing time or there are only two different types of machine requirements (Blazewicz et al., 1986), when tasks have various ready times and require either one or all machines (Blazewicz et al., 2003), or when only two or three machines are involved (Du and Leung, 1989; Chen and Lee, 1999). When the number of machines is fixed and not a parameter of the problem, Amoura et al. (2002) proposed a polynomial time approximation scheme for the problem $Pm|size_j|C_{\max}$, where Pm means that the number of identical machines (m) is fixed. When it comes to the general $P|size_j|C_{\max}$ problem, to the best of our knowledge, the only exact method was proposed by Blkadek et al. (2015). The authors developed a simple branch and bound (B & B) algorithm to solve the $P|size_j|C_{\max}$. The algorithm searches the solution space by enumerating all possible permutations of tasks. Partial permutations of tasks which represents sequences of completed tasks are expanded step by step adding undone tasks. For each newly generated partial solution, only one simple lower bound (the makespan of the partial solution) is used in the algorithm to determine the validity of the partial solution. Using a cluster of 30 PCs, the algorithm was reported to be able to deliver optimal solutions for instances with up to 11 tasks. Several approximation methods have been proposed to solve the problem. Lin and Chen (1994) obtained a heuristic algorithm based on the well-known *largest processing time first scheduling, LPT* rule. The performance bound of this algorithm is proved to be $\frac{4}{3}k - \frac{k(k+1)}{6m}$, where k equals the largest number of machines required by each task. A similar approximation method for the $P|size_j|C_{\max}$ problem was developed by Li (1999). The method was shown to have a performance guarantee of $\frac{31}{18}$. Later on, Johannes (2006) proved that no approximation algorithm for the $P|size_j|C_{\max}$ problem can provide with a performance guarantee better than $\frac{3}{2}$, unless P = NP. The study also demonstrated the performance guarantee of any list-scheduling algorithm for the $P|size_j|C_{\max}$ problem should be no less than 2. Switalski and Seredynski (2015) proposed an evolutionary metaheuristic algorithm called the generalized extremal optimization (GEO) to solve a machine scheduling problem with multi-processor tasks. In their study, each machine is defined to have a batch of processors that can work in parallel, and different machines may have different numbers of processors. Tasks with different processor requirements are scheduled among these machines (each task can only be assigned to one machine). Instances with up to 500 tasks and 48 machines are solved and compared with the genetic algorithm (GA). The computational results showed that the proposed GEO outperforms the GA for most of the tested instances.

For a comprehensive understanding of this problem, refer to the studies of Drozdowski (1996), Lee et al. (1997), Brucker (2006) and Leung (2004).

The $P|size_j|C_{\max}$ problem has wide-range industrial applications and has been studied by a number of scholars. In this study, we analyze several properties of the optimal solution of the $P|size_j|C_{\max}$ problem, formulate two different programming models and propose an exact branch and bound (B & B) method and a tabu search (TS) heuristic to solve the problem. Numerical experiments show that the proposed algorithms are effective and efficient to solve the $P|size_j|C_{\max}$ problem with different sizes.

3. Problem description

This section first gives a formal description of the parallel machine scheduling problem with multi-processor tasks, and then presents several properties of an optimal schedule to this problem.

This problem can be described as follows: assume there is a set $N = \{1, 2, \ldots, n\}$ of n tasks to be processed by the set $M = \{1, 2, \ldots, m\}$ of m identical machines. All the tasks are available and all the machines are ready to work from time $t = 0$. Each machine can process at most one task at a time, and each task j needs to be processed by $size_j$ arbitrary machines working in parallel ($size_j$ is integer and $1 \leq size_j \leq m$) in a continuous period of p_j units of time (p_j is integer and $p_j \geq 1$). Preemption is not allowed. The objective is to find a schedule that produces the minimum makespan ($C_{\max}$).

The optimal schedule of the $P|size_j|C_{\max}$ problem has several properties, which can help decrease the search space for the solution algorithms. The first property examines the effectiveness of List Scheduling (LS) rule for the considered problem. List Scheduling rule assigns the earliest available machines to each task in a given sequence.

Theorem 3.1. *There exists an optimal schedule for the $P|size_j|C_{\max}$ problem where machines are assigned to tasks by the List Scheduling rule.*

Proof. *Let δ denote a schedule of all tasks in our considered problem. In addition, we assume in schedule δ, $C(\delta)_j$ is the completion time of task j and $C_{\max}(\delta)$ is the makespan. Since $C_{\max}(\delta_1) > C_{\max}(\delta_2)$ implies $C(\delta_1)_j > C(\delta_2)_j$ for at least one j, the objective function $C_{\max}(\delta)$ is a regular function of δ. Further, for any scheduling problem with a regular objective function, there exists an optimal schedule that is generated by the List Scheduling rule* (Schutten, 1996).□

The second property specifies the sequence of certain tasks in an

Download English Version:

https://daneshyari.com/en/article/7355097

Download Persian Version:

https://daneshyari.com/article/7355097

Daneshyari.com