Computers & Operations Research 40 (2013) 2970–2974

Contents lists available at SciVerse ScienceDirect

Computers & Operations Research

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

Minimizing resource consumption on uniform parallel machines with a bound on makespan

Min Ji [a], Jen-Ya Wang [b], Wen-Chiung Lee [c,*]

[a] School of Computer Science and Information Engineering, Contemporary Business and Trade Research Center, Zhejiang Gongshang University, Hangzhou 310018, PR China
[b] Department of Computer Science and Information Management, Hungkuang University, Sha Lu 43302, Taiwan
[c] Department of Statistics, Feng Chia University, 100 Wenhua Road, Taichung 40724, Taiwan

ARTICLE INFO

Available online 13 July 2013

Keywords:
Scheduling
Uniform parallel machine
Makespan
Resource consumption

ABSTRACT

With the crucial issue of environmental protection, managing natural resources efficiently and/or reducing the amount of carbon emissions have become more important than ever. In this paper, we introduce a uniform parallel machine scheduling problem where the objective is to minimize resource consumption given that the maximum completion time does not exceed a certain level. We show that the problem is strongly NP-hard. A tight lower bound and a particle swarm optimization algorithm are then developed. Finally, some computational results are provided.

1. Introduction

Parallel machine scheduling problems have been studied intensively in the past decades. Pinedo [1] pointed out that a bank of machines in parallel is a setting that is important from both a theoretical and a practical point of view. From a theoretical viewpoint, it is a generalization of the single machine and a special case of the flexible flow shop. From a practical point of view, it is important because the occurrence of resources in parallel is common in the real world. Also, techniques for machines in parallel are often used in decomposition procedures for multistage systems.

However, classical scheduling problems typically strike for increased efficiency in terms of "time". For instance, minimizing the makespan is achieved by finishing all the jobs in the shortest possible time. In response to the effects of global warming, environmental protection and carbon emission cuts have become more crucial issues. Moreover, the problems of pollution and excessive use of natural resources are prevalent in many industries. For instance, the liquid crystal display industry uses an excessive amount of water, and the petrochemical industry produces severe air pollution. Often, newer machines have faster processing speeds in manufacturing systems. At the same time, the amounts of resource consumption might be different between machines. Motivated by these observations, we introduce a uniform parallel machine scheduling problem where the objective is to minimize the resource consumption given that all the jobs must be finished within a period of time, i.e., the makespan cannot exceed an upper bound.

Cheng and Sin [2] and Mokotoff [3] provided extensive reviews of the research on parallel machine scheduling. The majority of the research used the makespan as the objective function [4–7]. In addition, some researchers considered the same problem under various environments such as assuming the machines have the maintenance activities [8,9], or assuming the job processing is shortened due to learning effects [10], or assuming the machines are unrelated [11–13]. In these articles, the makespan is the primary objective to be minimized. Due to the environmental protection issue, it might be reasonable to sacrifice a little bit time efficiency in exchange for energy savings, reducing the pollutions or excessive usages of natural resources. This motivates us to move the makespan (time efficiency) to the secondary objective as a constraint, and the resource consumption becomes the primary objective to be minimized. To the best of our knowledge, this problem has never been studied before. The remainder of this paper is organized as follows. In Section 2, we formulate the problem. In Section 3, we show that the problem is strongly NP-hard and derive a tight lower bound for this problem. In Section 4, we develop a particle swarm optimization algorithm. The computational experiments are given in Section 5, and the conclusion is presented in Section 6.

2. Problem formulation

The description of the proposed problem is as follows. There are a set of n independent jobs $J=\{J_1,\ldots,J_n\}$ to be scheduled on a set of uniform parallel machines $M=\{M_1,M_2,\ldots\}$. All the jobs are available for processing at time 0. Each job has to be processed on either one of the machines. Each machine can process one job at a

* Corresponding author. Tel.: 886 4 24517250x4016; fax: 886 4 24517092.
E-mail address: wclee@fcu.edu.tw (W.-C. Lee).

0305-0548/$ - see front matter © 2013 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.cor.2013.06.011

time, and once a job starts to be processed, it must be completed without interruption. Each job J_j has a processing time p_j, and each machine M_i has a speed v_i. Moreover, there is a cost of β_i of processing jobs on machine M_i per unit time. For instance, β_i is the amount of carbon emission, water consumption, or electricity usage of producing one unit of production on machine M_i. Without loss of generality, we assume that $\beta_1/v_1 \le \beta_2/v_2 \le \ldots$. Given that all the jobs must be finished before time B, the objective is to find a schedule that minimizes the total resource consumption (TRC). A mixed integer linear programming formulation for the proposed problem can be expressed as follows:

$$\text{Min } TRC = \sum_{j=1}^{n} \sum_{i=1}^{\infty} \beta_i p_j x_{ij} / v_i \quad (1)$$

subject to

$$\sum_{i=1}^{\infty} x_{ij} = 1 \text{ for } j = 1, \ldots, n$$

$$\sum_{j=1}^{n} p_j x_{ij} / v_i \le B \text{ for } i = 1, 2, \ldots$$

where x_{ij} is an indicator function that is 1 if job J_j is assigned to machine M_i, and 0 otherwise. Using the conventional three-field notation, the problem is denoted as $Q/C_{\max} \le B/TRC$.

Another application of the proposed model is the firm outsourcing. In this case, the jobs are the orders, and the machines are the outsourcing suppliers, who have different speeds of processing and charges due to their facilities or capabilities. Our goal is to determine an assignment of the orders to the suppliers such that the total cost or expense is minimized given that all the orders must be ready within a certain period of time.

3. Some results

In this section, we will first prove the $Q/C_{\max} \le B/TRC$ problem is strongly NP-hard. The complexity of the $Q/C_{\max} \le B/TRC$ problem can be established by a reduction from the 3-partition problem, which is known to be strongly NP-hard [14] in polynomial time.

Theorem 1. *The $Q/C_{\max} \le B/TRC$ problem is strongly NP-hard.*

Proof. At first an instance I of the 3-partition problem is given:

Given a set of $3n+1$ positive integers a_1, a_2, …, a_{3n} and b such that $b/4 < a_j < b/2$ for $1 \le j \le 3n$ and $\sum_{j=1}^{3n} a_j = nb$, is there a partition of $\{a_1, \ldots, a_{3n}\}$ into n disjoint subsets of 3 integers, such that the sum of each subset exactly equals to b?

For any given instance I of the 3-partition problem, we construct a corresponding instance II of the corresponding decision version of our parallel-machine problem as follows:

- Number of machines: n
- Speed of the machines: $v_i = 1$ for $i = 1, \ldots, n$
- Resource consumption of the machines: arbitrary β_i for $i = 1, \ldots, n$
- Number of jobs: $3n$
- Bound of the maximum completion time: $B = b$
- Processing times: $p_j = a_j$ for $j = 1, \ldots, 3n$
- The threshold: $G = B\sum_{i=1}^{n} \beta_i$.

It is easy to verify that there exists a schedule with an objective value less than or equal to G if and only if there exists a solution for the 3-partition problem.

For an NP-hard problem, developing a heuristic algorithm might be an alternative approach. Before constructing the algorithm, we first provide a lower bound to evaluate the performance of the proposed algorithm.

Property 1. Let k be the number such that $B\sum_{i=1}^{k} v_i \le \sum_{j=1}^{n} p_j \le B\sum_{i=1}^{k+1} v_i$, then $B\sum_{i=1}^{k} \beta_i + \beta_{k+1}(\sum_{j=1}^{n} p_j - B\sum_{i=1}^{k} v_i)/v_{k+1}$ is a tight lower bound of the amount of the resource consumption.

Proof. Since β_i/v_i is the unit cost of processing jobs on machine M_i, it is natural to assign as many jobs as possible to the machine with the lowest unit cost rate β_i/v_i. It is machine 1 since we have the assumption of $\beta_1/v_1 \le \beta_2/v_2 \le \ldots$. Split the last job if necessary to make machine 1 fully loaded, and process the remaining part of the split job on machine 2. The process is repeated until all the jobs have been processed. Thus, the lower bound is derived.

To show the lower bound is tight, consider an example of two jobs and two machines where $\beta_1/v_1 \le \beta_2/v_2$. Let $p_1 = Bv_1$, and $p_2 = 0.5Bv_2$. The optimal schedule is to assign job 1 to machine 1 and job 2 to machine 2, and the total resource consumption is $B\beta_1 + 0.5B\beta_2$, which is equal to the lower bound.

4. Heuristic algorithm

In this section, we will develop a heuristic algorithm for this NP-hard problem. Since β_i/v_i is the unit cost of processing jobs on machine M_i, it is natural to assign as many jobs as possible to the machine with the lowest rate β_i/v_i. If this machine is full or no more jobs can be inserted, we continue to assign jobs to the machine with the second lowest rate. The process is repeated until all jobs have been assigned. In addition, let K be the sufficient number of machines such that all the jobs can be processed. The details of the proposed algorithm are as follows for a given sequence $\sigma = (\sigma(1), \ldots, \sigma(n))$.

Algorithm 1. Input: n, K, B, p_j for $j = 1, \ldots, n$, β_i, and v_i for $i = 1, \ldots, K$.

Step 1. Let $\sigma = (\sigma(1), \ldots, \sigma(n))$ be an ordering of jobs. Set $TRC = 0$, $S_1 = \ldots = S_K = \phi$, $CT_1 = \ldots = CT_K = 0$, and $j = 1$.
Step 2. Set $i = 1$.
Step 3. If $CT_i + p_{\sigma(j)}/v_i \le B$, set $CT_i = CT_i + p_{\sigma(j)}/v_i$, $TRC = TRC + \beta_i p_{\sigma(j)}/v_i$, put job $\sigma(j)$ into set S_i, and go to Step 4. Otherwise, go to Step 5.
Step 4. If $j \le n$, set $j = j + 1$, and go to Step 2. Otherwise, go to *Step 6*.
Step 5. If $i < K$, set $i = i + 1$, and go to Step 3.
Step 6. Output $(S_1, \ldots, S_K, TRC)$.

In the output of the algorithm, S_i contains the set of jobs processed on machine M_i, and TRC is the total cost associated with this schedule.

Many evolutionary heuristic algorithms have been proposed and successfully applied to combinatorial optimization problems [15–19]. In order to produce random sequences for Algorithm 1, we adopt the particle swarm optimization (PSO) method, a kind of evolutionary optimization technique advocated by Kennedy and Eberhart [20]. The biological inspiration for this technique is based on the collective behaviors of insect colonies, bird flocks, fish schools, and other animal societies. The standard PSO procedure maintains a swarm of particles that represent the potential solutions of the problem studied. Each particle is embedded with relevant information regarding the decision variables and a fitness value that provides an indication of the performance of this particle. Basically, the trajectory of each particle is updated according to its own flying experience as well as to that of the best particle in the swarm. The procedure of the PSO implemented in the problem is as follows.