Computers & Operations Research 79 (2017) 109–118

Contents lists available at ScienceDirect

Computers & Operations Research

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

Scheduling nonpreemptive jobs on parallel machines subject to exponential unrecoverable interruptions

Alessandro Agnetis[a], Paolo Detti[b,*], Patrick Martineau[c]

[a] Dipartimento di Ingegneria dell'Informazione, Università di Siena, Italy
[b] Dipartimento di Ingegneria dell'Informazione, Università di Siena, Via Roma 56, 53100, Italy
[c] Laboratoire d'Informatiqued'Informatique – Polytech'Tours, Université de Tours, France

ARTICLE INFO

Keywords:
Unrecoverable interruptions
Parallel machines scheduling
NP-hardness
Exact and heuristic algorithms

ABSTRACT

In this paper we consider the problem of scheduling n independent jobs on m parallel machines. If, while a machine is processing a job, a failure (*unrecoverable interruption*) occurs, the current job as well as subsequently scheduled jobs on that machine cannot be performed, and hence do not contribute to the overall revenue or throughput. The objective is to maximize the expected amount of work done before an interruption occurs. In this paper, we investigate the problem when failures are exponentially distributed. We show that the problem is NP-hard, and characterize a polynomially solvable special case. We then propose both an exact algorithm having pseudopolynomial complexity and a heuristic algorithm. A combinatorial upper bound is also proposed for the problem. Experimental results show the effectiveness of the heuristic approach.

1. Introduction

In this paper we address the problem of scheduling n jobs on m parallel *machines* which are subject to unrecoverable interruptions. More and more computer systems are based on multiple computer servers or dedicated data storage. When the number of computers is large, the probability of *failures* is not negligible, and allocation and scheduling policies must therefore adequately take it into account [3]. Classical grid schedulers plan jobs on parallel machines for one or several windows [4], i.e., each machine receives a set of jobs to execute in a certain time window. If the machine interrupts at time t during the window, all jobs that are unfinished at time t are lost. Another possibility to access CPU resources is to be connected to a cloud system. In this setting, some computer can be shutdown or extracted from the cloud without sending a message and the jobs it was processing are lost. In all the above cases, a job can be interrupted without the possibility to continue on another machine. The situation we consider is therefore similar to the model described by Benoit et al. [5,6]. In their work, a certain computing workload has to be allocated to a number of remote computers, each of which is subject to interruptions (of known likelihood) that kill all work in progress on it. The problem is how to allocate the workload on the remote computers so that the *expected amount of completed work* is maximized. Unlike [5,6], here we are not concerned with a "divisible" workload, but rather we have a *finite* set of jobs, each of which requires a certain processing time. Also, we do not consider job checkpointing [10], as this requires some context-switching and is therefore relatively expensive.

In general, the chances of successfully carrying out a job depend on the failure process as well as on job features. Typically, the shorter a job, the higher the probability of success. As for the failure process, in this paper we address the case in which the time between machine failures is *exponentially distributed*. This implies that the success probability of a job does not depend on its starting time, but only on its duration. The problem, which we denote as *Exponential Unrecoverable Interruption Scheduling Problem* (EUISP), consists in allocating the jobs to the machines and sequencing them in order to maximize the *expected amount of completed work* achieved by the system.

In this paper we present a number of results concerning the complexity of EUISP and viable solution approaches. First we elaborate on the fact that, if the machines have identically distributed failure processes, the problem can be formulated as a special case of another problem known in the literature as *Unreliable Job scheduling Problem* (UJP) [2]. We then show that EUISP is in general NP-hard even for m=2 identical machines, and we characterize a special class of instances of EUISP which can be efficiently solved. Thereafter we consider two solution approches, namely (i) an exact, dynamic-programming-based solution algorithm and (ii) a list scheduling heuristic algorithm. The exact algorithm has pseudopolynomial complexity (for a

* Corresponding author.
E-mail addresses: agnetis@dii.unisi.it (A. Agnetis), detti@dii.unisi.it (P. Detti), patrick.martineau@univ-tours.fr (P. Martineau).

http://dx.doi.org/10.1016/j.cor.2016.10.013
Received 8 January 2015; Received in revised form 25 October 2016; Accepted 26 October 2016
Available online 28 October 2016
0305-0548/

fixed number of machines), showing indeed that EUISP is not strongly NP-hard. We perform computational experiments for both approaches. Thanks to an upper bound which can be quickly computed, we show that, at least on instances of moderate size, the list scheduling algorithm provides good quality solutions for EUISP in a very limited computation time.

The plan of the paper is as follows. In Section 2 we define the problem and introduce the notation. In Section 3 we discuss the relationship between EUISP and UJP, and establish some results needed by the subsequent complexity analysis, carried out in Section 4. In Section 5, we propose an exact, dynamic-programming-based solution algorithm and a heuristic algorithm. In Section 6 an upper bound is presented, which is then used in the computational experiments of Section 7 to assess the quality of heuristic solutions. Finally, some conclusions are drawn.

2. Problem definition and notation

Let $J = \{J_1, \ldots, J_n\}$ and $M = \{M_1, \ldots, M_m\}$ be the sets of jobs and machines, respectively, with $|J| = n$ and $|M| = m$. To be carried out, a job J_j requires a *processing time* p_j on any machine. We suppose the jobs numbered in SPT order, i.e., $p_j \le p_k$ if $j < k$. Jobs cannot be preempted. Machines are subject to *unrecoverable interruptions*. This means that if a machine fails while processing a job, the work related to that job and all subsequently scheduled jobs on that machine are lost and cannot be accounted for. Here we consider the case in which, for machine M_i, the time between failures is *exponentially distributed* with parameter λ_i, i.e., if M_i is up at a certain time, the probability that it is still up after t time units is given by $e^{-\lambda_i t}$. In fact, thanks to the well-known memoryless property of the exponential distribution, such probability value does not depend on the time elapsed since the last failure, but only on t. Hence, if machine M_i is *up* when starting job J_j, the probability that it does not fail during the execution of J_j is given by $\pi_j = e^{-\lambda_i p_j}$. We call π_j the *success probability* of job J_j. If a job J_j is successfully processed by a machine, we achieve an *amount of work* given by p_j. The problem addressed in this paper, EUISP, consists in allocating the jobs to the m machines and sequencing them on each machine in such a way that the total *expected amount of work done* is maximized.

In the following we use $\sigma = \{\sigma_1, \sigma_2, \ldots, \sigma_m\}$ to denote a *schedule*, i.e., an assignment of the n jobs to the m machines, along with a job sequence σ_i for each machine M_i ($i = 1, \ldots, m$). Given a subset of K jobs assigned to a certain machine M_i, having failure parameter λ_i, and given a sequence σ_i of such jobs, we denote by $J_{\sigma_i(h)}$ the job in h-th position on the machine. The expected amount of work done by machine M_i, denoted as $EAW_i[\sigma_i]$, is given by:

$$EAW_i[\sigma_i] = p_{\sigma_i(1)} e^{-\lambda_i p_{\sigma_i(1)}} + p_{\sigma_i(2)} e^{-\lambda_i (p_{\sigma_i(1)} + p_{\sigma_i(2)})} + \ldots + p_{\sigma_i(K)} e^{-\lambda_i (p_{\sigma_i(1)} + p_{\sigma_i(2)} + \ldots p_{\sigma_i(K)})} = \sum_{h=1}^{K} p_{\sigma_i(h)} e^{-\lambda_i \sum_{j=1}^{h} p_{\sigma_i(j)}}. \tag{1}$$

EUISP consists in finding σ such that

$$EAW[\sigma] = \sum_{i=1}^{m} EAW_i[\sigma_i]$$

is maximized. When referring to the problem with m machines, we denote the problem as EUISP (m). When all m machines have identical failure processes (i.e., $\lambda_i = \lambda$ for $i = 1, \ldots, m$), we denote the problem as EUISP*(m). Clearly, if m=1, we use indifferently EUISP(1) or EUISP*(1).

3. Relationship with the unreliable job scheduling problem

In this section we show that EUISP*(m) is a special case of the Unreliable Job scheduling Problem (UJP) [2]. In UJP(m), there are n jobs that must be allocated and sequenced on m machines. No processing times are specified, but for each job J_j there is a certain *success probability* π_j and a *revenue* r_j. As in EUISP*(m), if a job fails, the machine on which the job was allocated is blocked and all subsequently scheduled jobs cannot be performed. When a set of K jobs is sequenced according to a sequence σ_i on machine M_i, the *expected revenue* $z_i(\sigma_i)$ for M_i is given by

$$z_i(\sigma_i) = r_{\sigma_i(1)}\pi_{\sigma_i(1)} + r_{\sigma_i(2)}\pi_{\sigma_i(1)}\pi_{\sigma_i(2)} + \cdots + r_{\sigma_i(K)}\pi_{\sigma_i(1)}\pi_{\sigma_i(2)} \cdots \pi_{\sigma_i(K)} = = \sum_{h=1}^{K} r_{\sigma_i(h)} \left(\prod_{j=1}^{h} \pi_{\sigma_i(j)} \right). \tag{2}$$

UJP(m) consists in allocating the jobs to the machines and sequencing them so that the total expected revenue is maximized. Notice that EUISP*(m) is a special case of UJP(m) in which r_j=p_j and $\pi_j = e^{-\lambda p_j}$. When m=1, UJP(1) can be efficiently solved by sequencing the jobs in nonincreasing order of the following *Z-ratio*, originally introduced by Mitten [9]:

$$Z_j = \frac{\pi_j r_j}{1 - \pi_j}. \tag{3}$$

When referring to EUISP*(1), the ratio Z_j becomes

$$Z_j = \frac{p_j e^{-\lambda p_j}}{1 - e^{-\lambda p_j}}. \tag{4}$$

Now, simple calculus shows that for any $\lambda > 0$, the function

$$f(x) = \frac{x e^{-\lambda x}}{1 - e^{-\lambda x}} \tag{5}$$

is monotonically decreasing in $[0, +\infty)$. As a consequence, if two jobs J_j, J_k are such that $p_j < p_k$, then $Z_j > Z_k$ and hence the following theorem holds:

Theorem 3.1. *EUISP(1) is solved by sequencing the jobs in nondecreasing order of shortest processing time (SPT order).*

As a consequence, in EUISP*(m), the problem concerns indeed the allocation of the jobs to the machines, since the optimal sequencing on each machine is then given by SPT. We next briefly recall the known complexity results for UJP. In [2], a polynomial reduction of PRODUCT PARTITION to UJP(2) is presented. We note here that since PRODUCT PARTITION was shown to be strongly NP-hard by Ng et al. [8], so is UJP(m) for any fixed $m \ge 2$.

Heuristic approaches for UJP have been considered [1]. In particular, for many parallel-machine scheduling problems a viable approach is the *List Scheduling Algorithm* (LSA), which consists in ordering all jobs in a list, and sequentially assigning them to the machines according to a given criterion. A list scheduling algorithm for UJP works as follows:

- Order the jobs by nonincreasing Z_j;
- Assign the first not assigned job to the machine i having largest *cumulative probability* $\Pi(i)$, i.e., largest product of the π_j of the jobs already allocated to that machine.

Note that the complexity of LSA is essentially due to the job sorting phase, i.e., $O(n \log n)$. For the case of m=2, it was proved [1] that LSA provides a solution of value at least $(2 + \sqrt{2})/4 \simeq 0, 853$ times the optimal expected revenue for UJP. Notice that, in the case of EUISP*, the Z-ordering coincides with the SPT-ordering, and the largest cumulative probability indeed coincides with the smallest cumulative processing time. Hence, LSA can be simplified as in Fig. 1, in which we use P_i to denote the total processing time of the jobs currently assigned to machine M_i. Recalling that the jobs are numbered in SPT order, we note that in this case LSA coincides with the simple round robin algorithm, consisting in allocating job J_k to machine M_u where $u = (k \bmod m)$, for all $k = 1, \ldots, n$.

LSA can be modified to solve EUISP (m), i.e., when the failure