Discrete Applied Mathematics 165 (2014) 2–11

Contents lists available at ScienceDirect

Discrete Applied Mathematics

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

The list scheduling algorithm for scheduling unreliable jobs on two parallel machines

Alessandro Agnetis, Paolo Detti[*], Marco Pranzo

Dipartimento di Ingegneria dell'Informazione, University of Siena, Via Roma 56, 53100 Siena, Italy

ARTICLE INFO

Article history:
Received 11 October 2011
Received in revised form 20 July 2012
Accepted 19 September 2012
Available online 13 October 2012

Keywords:
Unreliable jobs
List scheduling
Approximation algorithms

ABSTRACT

In this paper, we study a scheduling problem with unreliable jobs. Each job is characterized by a success probability and by a reward earned in case of success. In case of failure, the job blocks the machine that is processing it, and the jobs subsequently sequenced on that machine cannot be performed. The objective function is to maximize the expected reward. We address the problem in the case of two parallel machines, and analyze the worst-case performance of a simple list scheduling algorithm. We show that the algorithm provides an approximation ratio of $(2+\sqrt{2})/4 \simeq 0,853$, and that the bound is tight. We also provide a complexity result concerning the related *Total Weighted Discounted Completion Time Problem* on parallel machines.

1. Introduction

A set of jobs $J = \{J_1, \dots, J_n\}$ must be allocated to m parallel, identical machines, $M_1, \dots, M_m$. Each job must be assigned to a single machine and a machine can process one job at a time. Jobs are *unreliable*, i.e., while a job is being processed by a machine, a *failure* can occur, which implies losing all the work that was scheduled but not yet executed by the machine. Each job J_i is characterized by a certain *success probability* π_i (independent from other jobs) and a *reward* r_i, which is gained if the job is successfully completed. The problem is to find an allocation of jobs to the m machines and a sequence on each machine that maximizes total expected reward. In the following, we refer to this problem as *Unreliable Job scheduling Problem*, denoted by UJP. We use UJP(m) to denote UJP with m machines.

Let σ_j be a sequence of jobs assigned to machine M_j, and let $\sigma_j(k)$ be the job in k-th position in σ_j. A feasible solution $\sigma = \{\sigma_1, \sigma_2, \dots, \sigma_m\}$ for UJP is an assignment and sequencing of the n jobs on the m machines. The expected reward associated with sequence σ_j is obtained by summing up the contribution of the first, the second, …the k-th job in σ_j, taking into account that a job can be processed only if no breakdown on that machine occurred so far. Hence, if K jobs are sequenced on a machine M_j according to σ_j, recalling that the success probabilities are independent, the expected reward $ER[\sigma_j]$ is given by

$$ER[\sigma_j] = \pi_{\sigma_j(1)} r_{\sigma_j(1)} + \pi_{\sigma_j(1)} \pi_{\sigma_j(2)} r_{\sigma_j(2)} + \cdots + \pi_{\sigma_j(1)} \cdots \pi_{\sigma_j(K-1)} \pi_{\sigma_j(K)} r_{\sigma_j(K)} \quad (1)$$

and the total expected reward is therefore

$$ER[\sigma] = ER[\sigma_1] + ER[\sigma_2] + \cdots + ER[\sigma_m].$$

UJP consists in finding a solution $\sigma^* = \{\sigma_1^*, \sigma_2^*, \dots, \sigma_m^*\}$ that maximizes the total expected reward. We let z^* denote the value of such optimal solution. By a pairwise interchange argument, it can be shown [8] that the single-machine version of

[*] Corresponding author.
E-mail addresses: agnetis@dii.unisi.it (A. Agnetis), detti@dii.unisi.it (P. Detti), pranzo@dii.unisi.it (M. Pranzo).

0166-218X/$ – see front matter © 2012 Elsevier B.V. All rights reserved.
doi:10.1016/j.dam.2012.09.014

UJP is solved by sequencing the jobs in nonincreasing order of the following *Z-ratio* [8]:

$$Z_i = \frac{\pi_i r_i}{1-\pi_i}. \tag{2}$$

UJP is related to the *Total Weighted Discounted Completion Time Problem* (TWDCTP) on m parallel machines [10]. In TWDCTP, each job i has processing time p_i and weight w_i (suppose integers). If job i is completed at time C_i, we incur the cost $w_i(1-e^{-\gamma C_i})$, where $\gamma > 0$ is a given discount factor. The problem is that of minimizing total costs, which is equivalent to maximizing $\sum w_i e^{-\gamma C_i}$. Also TWDCTP on a single machine can be solved by a simple index rule, namely sequencing the jobs by nondecreasing values of the ratio

$$\frac{w_i e^{-\gamma p_i}}{1-e^{-\gamma p_i}}. \tag{3}$$

If $\gamma \ll 1$, one has $\sum w_i e^{-\gamma C_i} \approx \sum w_i - \sum w_i C_i$, and therefore TWDCTP tends to become equivalent to the *Total Weighted Completion Time Problem* (TWCTP), i.e., the problem of minimizing the weighted sum of job completion times. Such problem is solved by the well-known Smith's rule, i.e., jobs must be processed in nonincreasing order of the ratio

$$\frac{w_i}{p_i} \tag{4}$$

and as $\gamma \to 0$ in (3), the two rules (3) and (4) tend to become the same. (In fact, it was proved by Rothkopf and Smith [11] that the single-machine versions of TWDCTP and TWCTP are the *only* scheduling problems of the form $1 \parallel \sum_i f_i(C_i)$ which are solved by a simple priority index sequencing rule.) Note that all these problems (UJP, TWDCTP, TWCTP) essentially consist in deciding how to partition the n jobs among the m machines, since on each machine the sequencing is then dictated by the priority rule.

The plan of the paper is as follows. In Section 2, some results and applications from the literature are reviewed. In Section 3, the relationship between UJP, TWDCTP and TWCTP is investigated. In Section 4, the list scheduling heuristic algorithm for UJP is proposed, and the worst case behavior is analyzed. Finally, conclusions follow.

2. Literature review and applications

UJP has been introduced in [1], motivated by an application in highly automated manufacturing facilities. In particular, in preparation for an unsupervised shift, jobs are selected for processing on a set of parallel machines. During the unsupervised operation, the jobs are automatically loaded on the machine whenever the machine becomes idle. If a failure occurs during the processing of a job, all subsequent jobs on that machine cannot be processed, and processing of that set of jobs is suspended until the blocked machine is cleared at the end of the unsupervised shift. In [1], the NP-hardness proof for UJP(2) is given, and a round-robin algorithm for UJP(m) is analyzed. Such algorithm simply consists in ordering the jobs according to nonincreasing Z-ratios and then assign them to machine $1, 2, \ldots, m$, and then again $1, 2, \ldots,$ until all jobs are assigned. It is shown that for any m the round-robin algorithm is 1/2-approximate, and the bound is tight. This motivates the research for fast algorithms having better worst-case bound.

There are several applications in which a function like (1) has to be either minimized or maximized. Several researchers [2,3,12] analyzed data acquisition and processing problems in sensor networks. To minimize data transmission, it is desirable to perform "in-network" query processing, i.e., to do some data processing at intermediate nodes, by placing filters on them. A filter i has a selectivity s_i, defined as the fraction of data stream captured by the filter, and a per-tuple execution cost c_i. Once n filters have been located at a node, the problem is to decide in which sequence they should be applied. The cost of sequence σ is given by:

$$EC[\sigma] = c_{\sigma(1)} + s_{\sigma(1)}c_{\sigma(2)} + \cdots + s_{\sigma(1)} \cdots s_{\sigma(n-1)}c_{\sigma(n)} \tag{5}$$

which is formally identical to (1), identifying s_i with π_i and c_i with $\pi_i r_i$. Another similar application arises in the management of queries in databases [4]. The queries may have an arbitrary number of restriction predicates, each of which may be a complicated Boolean function, possibly containing expensive (i.e., time-consuming) subqueries. The problem is to sequence the predicates (each having a certain selectivity and cost) in such a way as to minimize the expense of applying them to the tuples of the relation being scanned. Several other applications related to component testing problems having a cost structure equivalent to (5) are described by Ünlüyurt [13]. In [6], a series repairable system with n components has been considered, which may *fail* because one of its components has failed. Probability of each component to be responsible for the failure is given. Each component can be tested and repaired at given costs, and the problem is to find a sequence of testing and repairing operations such that the system is always repaired, and the total expected cost of testing and repairing the components is minimized. Polynomially solvable special cases are presented and a fully polynomial time approximation scheme is given for the general case.

3. Relationship with TWDCTP and TWCTP

In this section, we analyze the relationships between UJP and problems TWDCTP and TWCTP.