Information Processing Letters 132 (2018) 39–43

Contents lists available at ScienceDirect

Information Processing Letters

www.elsevier.com/locate/ipl

Minimizing total load on a proportionate flowshop with position-dependent processing times and job-rejection

Shir Fiszman, Gur Mosheiov [*]

School of Business Administration, The Hebrew University, Jerusalem 91905, Israel

A R T I C L E   I N F O

Article history:
Received 15 January 2017
Received in revised form 13 November 2017
Accepted 15 December 2017
Available online 19 December 2017
Communicated by P. Wong

Keywords:
Scheduling
Proportionate flowshop
Total load
Position-dependent processing times
Job-rejection

A B S T R A C T

We study the scheduling problem of minimizing total load on a proportionate flowshop. We consider position-dependent job processing times in the most general way. We show that this problem is solved in $O(n^4)$ time, where n is the number of jobs. We then extend the setting to allow job-rejection, where the scheduler may decide to process only a subset of the jobs, and the rejected jobs are penalized. This extension is shown to be solved in $O(n^5)$ time.

1. Introduction

Scheduling problems with the objective function of minimum *total load* have been rarely studied. This measure becomes relevant usually in multi-machine settings, and is equivalent to the sum of the largest completion times on all the machines. Minimizing total load is justified in settings where the cost is a function of the time that the machines are busy. In such cases the scheduler tries to finish the work on all machines as early as possible. When focusing on the classical setting of parallel identical machines, it is clear that total load is independent of the job allocation to the machines and of the job-sequence on each machine. However, in settings of time-dependent or position-dependent job processing times, minimizing total load becomes less trivial. Mosheiov [5] studied the problem of minimizing total load on parallel identical machines with time-dependent processing times (linear deterioration). He proved that the problem is NP-hard, and introduced and tested a heuristic and a lower bound. Mosheiov [7] studied the same machine setting and objective function with position-dependent processing times (position-based deterioration). This case was shown to be polynomially solvable in the number of jobs and the number of machines. Yu et al. [11] extended the latter model to the case that the job processing times are both position- and machine-dependent. This setting was shown to be polynomially solvable as well. Cheng et al. [3] studied a similar objective function on a two-stage flowshop with a common critical machine in stage one and two independent dedicated machines in stage two. Their goal was to minimize the weighted sum of machine completion times (where, clearly, only stage-two machine completion times are considered in the objective function). They proved that the problem is strongly NP-hard, but the case of a given job-sequence is polynomially solvable.

In this note we study the measure of total load on an *m-machine proportionate flowshop*. In a proportionate flowshop, the job processing times are assumed to be machine-independent. As above, we focus on the case of position-dependent job-processing time. We allow the processing

[*] Corresponding author.
E-mail address: msomer@huji.ac.il (G. Mosheiov).

https://doi.org/10.1016/j.ipl.2017.12.004
0020-0190/© 2017 Published by Elsevier B.V.

times to be position-dependent in the most general way. In particular, no monotonicity is required: neither increasing processing time (reflecting deterioration or aging), nor decreasing processing time (reflecting learning). As mentioned, total load refers to the sum of the completion times of the last jobs on all the machines of the flowshop. We first introduce a polynomial ($O(n^5)$) time solution for this problem, where n is the number of jobs. Then, we consider a recently published improved procedure for makespan minimization (Agnetis and Mosheiov [1]). Using this faster procedure, the running time is reduced to $O(n^4)$.

In the second part of the paper, we further extend this setting to allow job-rejection, a topic which has become popular among scheduling researchers in the last decade. In many real-life situations, processing all jobs may cause a delay in the completion of orders and may lead to tardiness cost. A decision on rejecting some of the jobs (either outsourcing them or rejecting them all together) becomes reasonable. We refer the reader to the recently published survey on this topic of Shabtay et al. [10]. We show that this problem, i.e., that of minimizing total load plus the rejection cost on a proportionate flowshop with general position-dependent processing times is solved in $O(n^6)$ time. Then, based on the improved procedure mentioned above (Agnetis and Mosheiov [1]), the running time is shown to be reduced to $O(n^5)$.

Section 2 provides the notation and the formulation. In Section 3 we introduce the solution to the problem of minimum total load. Section 4 contains the solution for the extension in which the objective is minimum total load plus rejection cost. In Section 5 we show how to reduce the above procedures by a factor of n.

2. Formulation

We consider an n-job m-machine proportionate flowshop problem. The processing time of job j which is performed in position r on machine i is denoted (in a general flowshop) by p_{ijr}, $i = 1, \dots, m$, $j, r = 1, \dots, n$. In a proportionate flowshop, job processing times are machine-independent, i.e., $p_{ijr} = p_{jr}$, $i = 1, \dots, m$, $j, r = 1, \dots, n$. The job-position processing times (p_{jr}) are given by a general $n \times n$ matrix, and as mentioned, no specific function and no monotonicity in r are assumed.

For a given schedule, C_{ij} denotes the completion time of job j on machine i; $i = 1, \dots, m$; $j = 1, \dots, n$. $C_{max}^{(i)} = \max\left\{C_{ij};\ j = 1, \dots, n\right\}$, $i = 1, \dots, m$, denotes the completion time of the last processed job on machine i. The objective function is minimizing *Total Load*, i.e., $TL = \sum_{i=1}^{m} C_{max}^{(i)}$. Using the classical notation for defining scheduling problems, the problem studied here is:

$$F/p_{ijr} = p_{jr}/\sum_{i=1}^{m} C_{max}^{(i)}.$$

When the option of job-rejection is considered, let e_j denote the rejection cost of job j, $j = 1, \dots, n$. If the entire set of jobs is denoted by N, let S denote the set of scheduled (non-rejected) jobs, and R denote the set of the rejected jobs ($N = S \cup R$). The *Total Rejection* cost is $TR = \sum_{j \in R} e_j$. The objective function becomes minimum total load and total rejection cost: $TL + TR$. Thus, the problem solved in this section is:

$$F/p_{ijr} = p_{jr}/\sum_{i=1}^{m} C_{max}^{(i)} + \sum_{j \in R} e_j.$$

3. A polynomial-time solution for minimizing total load

In the classical makespan minimization problem on a proportionate flowshop, it is well-known (see e.g., Pinedo [9]) that the optimum is sequence-independent. The makespan value (i.e., the completion time of the last job on machine m) is given by $\sum_{j=1}^{n} p_j + (m-1)p_{max}$, where p_{max} is the largest processing time. Note that the critical path follows the largest job along all m machines, implying that the contribution of this largest job to the makespan is mp_{max}. When general position-dependent job processing times are assumed, the largest processing time is not known in advance, and is clearly sequence-dependent. Mosheiov [6] proposed a solution procedure, in which all job-positions are tested as being the largest processing time. Specifically, in each iteration of the algorithm, one of the n^2 job-positions is defined as a pseudo-job, and its processing time is multiplied by m. At the end of each iteration, the resulting schedule is checked, and if feasible (i.e., the pseudo-job is indeed the largest), the solution is registered as a candidate for being optimal. The best of all the feasible schedules is clearly the optimum.

When the objective function is that of total load $\left(\sum_{i=1}^{m} C_{max}^{(i)}\right)$, an extension of this idea can be employed. Assume first that the job sequence is known, and that the (actual) job processing time of the job in position j is denoted by $p_{(j)}$, $j = 1, \dots, n$. Assume also that the largest processing time is that of the job in position l, i.e., $p_{max} = p_{(l)}$. Thus, the job sequence is given by: $p_{(1)}, p_{(2)}, \dots, p_{(l)} (= p_{max}), \dots, p_{(n)}$. It follows that the completion time of the last job on each machine is given by:

Machine 1: $C_{max}^{(1)} = \sum_{j=1}^{n} p_{(j)}$;
Machine 2: $C_{max}^{(2)} = \sum_{j=1}^{n} p_{(j)} + p_{max}$;
Machine 3: $C_{max}^{(3)} = \sum_{j=1}^{n} p_{(j)} + 2p_{max}$;
...
Machine m: $C_{max}^{(m)} = \sum_{j=1}^{n} p_{(j)} + (m-1)p_{max}$.

Total load (for a given job sequence) is thus the following:

$$TL = \sum_{i=1}^{m} C_{max}^{(i)} = m\sum_{j=1}^{n} p_{(j)} + \frac{(m-1)m}{2} p_{max}.$$

As a result, the problem can be formulated as a non-linear integer program. Let X_{jr} be a binary variable: $X_{jr} = 1$ if job j is assigned to position r, $X_{ij} = 0$ otherwise. The program is: