Information Processing Letters 112 (2012) 743–747

Contents lists available at SciVerse ScienceDirect

Information Processing Letters

www.elsevier.com/locate/ipl

Scheduling on parallel identical machines with job-rejection and position-dependent processing times

Enrique Gerstl, Gur Mosheiov *

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

ARTICLE INFO

Article history:
Received 30 January 2012
Received in revised form 17 June 2012
Accepted 19 June 2012
Available online 21 June 2012
Communicated by Jinhui Xu

Keywords:
Scheduling
Parallel machines
Flow-time
Job-rejection
Position-dependent processing times

ABSTRACT

We solve scheduling problems which combine the option of job-rejection and general position-dependent processing times. The option of rejection reflects a very common scenario, where the scheduler may decide not to process a job if it is not profitable. The assumption of position-dependent processing time is a common generalization of classical settings, and contains the well-known and extensively studied special cases of "learning" and "aging". The machine setting is parallel identical machines, and two scheduling measures are considered: total flow-time and total load. When the number of jobs is given, both problems are shown to be solved in polynomial time in the number of jobs. The special case of non-decreasing job-position processing times ("aging") is shown to be solved much faster.

1. Introduction

In a recent paper, Li and Yuan [1] studied scheduling problems on parallel machines with job deterioration and the option of job rejection. Specifically, they assumed: (i) parallel identical machines, (ii) job-processing times which are linear increasing functions of their starting times (with job-independent deterioration factor), and (iii) the option to reject jobs for a job-dependent penalty. The objective function in each problem is a sum of a scheduling measure (makespan, total weighted completion times) and the total rejection cost.

In the setting of *scheduling with rejection*, the scheduler/supplier may decide not to process a job if it is not profitable. While saving some direct production cost, a rejected job incurs some penalty due to the rejection. This penalty may reflect e.g. revenue or reputation loss. Thus, the scheduler should decide prior to the scheduling decisions, which of the jobs are processed and which are rejected. Some papers dealing with scheduling with the option of job rejection are: Sengupta [2], Bartal et al. [3], Epstein et al. [4], Hoogeveen et al. [5], Engels et al. [6], Cao and Yang [7], Cheng and Sun [8], Mosheiov and Sarig [9], Shabtay et al. [10], Shabtay and Gasper [11], Zhang et al. [12], among others.

Scheduling with position-dependent processing times has received increasing attention in recent years. Two well-known special cases are: (i) *job deterioration* (or *aging effect*), where the processing times increase as a function of the position (see e.g. Mosheiov [13]) and (ii) *learning effect*, where the processing times decrease as a function of the position (see e.g. Biskup [14]).

In this note, we combine scheduling with rejection and position-dependent processing times. We consider the general setting studied by Li and Yuan [1], i.e. parallel identical machines and the option of job rejection. However, we assume *position-dependent* processing times, rather than the *time-dependent* assumption of Li and Yuan [1]. Moreover, we extend the setting to *general* position-dependent processing times, thus our model is not restricted to any specific job-position function, and in particular is not restricted to a monotonic function (i.e. either to aging or to learning).

* Corresponding author. Tel.: +972 2 588 3108; fax: +972 2 588 1341.
E-mail address: msomer@mscc.huji.ac.il (G. Mosheiov).

0020-0190/$ – see front matter
http://dx.doi.org/10.1016/j.ipl.2012.06.009

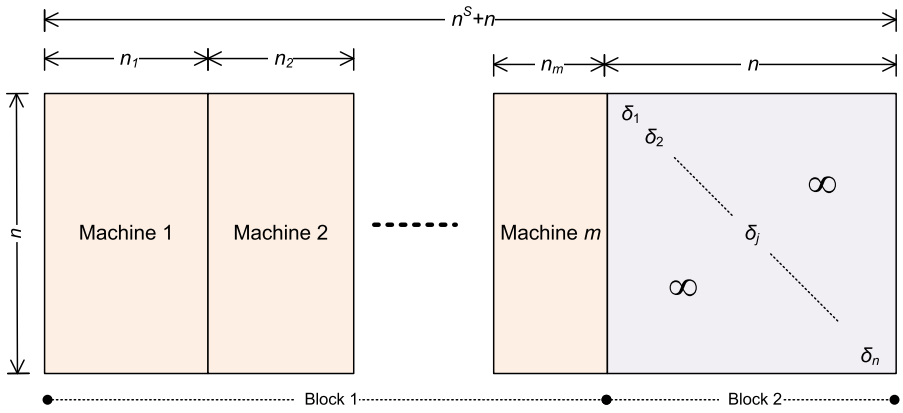

Fig. 1. General two-block structure of the cost matrix.

The first objective function considered in this note is total flow-time. Thus, the goal of the scheduler is to find the set of the rejected jobs, and to process the remaining jobs, such that the sum of the flow-time and cost of the rejected jobs is minimized. The second objective function is total load, i.e. the sum of the job-processing times on all the machines. This non-classical measure has been studied by several researchers; see e.g. the recent paper of Mosheiov [15]. Again, the scheduler needs to find the set of the rejected jobs, and to process the remaining jobs, such that the sum of the total load and the cost of the rejected jobs is minimized. Both cases are shown to be solved in polynomial time in the number of jobs (for a given number of machines). We also study the special case of aging: again, for a given number of machines, both problems (total flow-time and total load) are shown to be solved in $O(n^3)$ time, where n is the number of jobs.

In Section 2 we present the notation and formulation of the problems. In Sections 3 and 4 we solve the flow-time and the total load problems, respectively. In Section 5 we focus on the special case of aging.

2. Notation and formulation

We consider m parallel identical machines and n jobs. P_{jr} denotes the processing time of job j if assigned to position r (on any of the machines), $j, r = 1, \ldots, n$. (P_{ijr} denotes the processing time of job j if assigned to position r on machine i, $j, r = 1, \ldots, n$, $i = 1, \ldots, m$.) δ_j denotes the penalty cost for rejecting job j, $j = 1, \ldots, n$. For a given schedule, the completion time of job j is denoted by C_j. Let C_i^{max}, $i = 1, \ldots, m$, denote the completion time of the last scheduled job on machine i. The scheduling measures considered in this paper are: total completion time, i.e. $TC = \sum_{j=1}^{n} C_j$, and total load, i.e. $TL = \sum_{i=1}^{m} C_i^{max}$.

For a given schedule, let S denote the set of non-rejected jobs, and R denote the set of the remaining (rejected) jobs. Similarly, let n^S and n^R denote the number of processed jobs and rejected jobs, respectively. Clearly, $n^S + n^R = n$. Let n_i denote the number of jobs processed on machine i, $i = 1, \ldots, m$. (Clearly, $\sum_{i=1}^{m} n_i = n^S$.) We denote the total rejection cost by $RC = \sum_{j \in R} \delta_j$. The goal of the scheduler is to find the sets S and R, and to schedule the jobs in S such that the sum of the scheduling measure and the rejection cost are minimized. Thus, the objective functions considered here are:

$$TC + RC = \sum_{j \in S} C_j + \sum_{j \in R} \delta_j, \tag{1}$$

$$TL + RC = \sum_{i=1}^{m} C_i^{max} + \sum_{j \in R} \delta_j. \tag{2}$$

For convenience, we denote these two problems *TCRC* and *TLRC*, respectively.

3. Total completion time with rejection (*TCRC*)

For a given number of rejected jobs and a given allocation of the processed jobs to the machines, we show that the problem can be reduced to a (non-standard) linear assignment problem. We create an assignment matrix, where the rows represent the jobs and the columns represent their potential positions. Thus there are n rows (one for each job). For the given allocation to n^S processed jobs and n^R rejected jobs, the matrix contains two blocks, respectively. For a given vector $(n_1, n_2, \ldots, n_m)$, there are n_i columns for the positions allocated to machine i (implying that the total number of columns in the first block is n^S, and the size of this block is $n \times n^S$). Since we do not know which of the jobs will be rejected, the second block contains n columns (one for each job). The size of this block is $n \times n$. Thus, the size of the entire assignment matrix is $n \times (n^S + n)$.

In the following we define the cost values in the matrix. The first block contains the processing times of the jobs multiplied by their positional weight on the relevant machine. Recall that the positional weight of position r on machine i is $n_i - r + 1$, $i = 1, \ldots, m$, $r = 1, \ldots, n_i$. It follows that the positional weights in our matrix are: $w_{ir} = n_i - r + 1$, $1 \leqslant r \leqslant n_i$. The second $n \times n$ block contains the δ_j values on the diagonal ($j = 1, \ldots, n$), and infinity in all other entries. Thus, each one of the n jobs can be potentially included in the set R of the rejected jobs. For convenience we define the second block (of the rejected jobs) as machine $m + 1$. This machine contains n potential positions, implying that this block contains n columns, from position $n^S + 1$ to position $n^S + n$ (i.e. $n_{m+1} = n$). Fig. 1 demonstrates the structure of the entire matrix.

Based on the above we obtain the following cost matrix:

$$cost_{ijr} = \begin{cases} w_{ir}P_{ijr}, & i = 1,\ldots,m,\ j = 1,\ldots,n, \\ & r = 1,\ldots,n_i, \\ \infty, & i = m+1,\ j = 1,\ldots,n, \\ & r = 1,\ldots,n,\ r \neq j, \\ \delta_j, & i = m+1,\ j = 1,\ldots,n,\ r = j. \end{cases} \quad (3)$$

Let X_{ijr} denote a binary variable reflecting whether job j is assigned to position r on machine i ($X_{ijr} = 1$ if job j is at position r on machine i; $X_{ijr} = 0$ otherwise). (Note that position $n^S + j$ for job j means that the job has been rejected.) Then, we formulate the following assignment problem (for a given $(n_1, n_2, \ldots, n_m)$ vector):

$$MIN \quad \left\{ \sum_{i=1}^{m+1} \sum_{j=1}^{n} \sum_{r=1}^{n_i} X_{ijr} cost_{ijr} \right\}, \quad (4)$$

$$S.T. \quad \sum_{i=1}^{m+1} \sum_{r=1}^{n_i} X_{ijr} = 1, \quad j = 1,\ldots,n, \quad (5)$$

$$\sum_{j=1}^{n} X_{ijr} = 1, \quad i = 1,\ldots,m,\ r = 1,\ldots,n_i, \quad (6)$$

$$\sum_{j=1}^{n} X_{ijr} \leqslant 1, \quad i = m+1,\ r = 1,\ldots,n, \quad (7)$$

$$X_{ijr} \geqslant 0, \quad i = 1,\ldots,m+1,\ j = 1,\ldots,n, \quad r = 1,\ldots,n_i. \quad (8)$$

The set of constraints (5) guarantees that each job is either assigned or rejected. The set of constraints (6) guarantees that all the positions defined by the vector $(n_1, n_2, \ldots, n_m)$ are occupied. (7) reflects the fact that each job can be rejected (in which case the left-hand side equals 1), or not rejected (and then the left-hand side equals 0). [Note that the set (7) is redundant due to the fact that in each column of the second block, there is only one finite value.] We conclude with the following theorem:

Theorem 1. *For a given number of machines, problem TCRC is solved in polynomial time in the number of jobs.*

Proof. It is clear that the above assignment problem solves *TCRC* for a given vector $(n_1, n_2, \ldots, n_{m+1})$. We note that the size of the assignment matrix is $n \times (n^s + n)$, i.e. $O(n \times n)$, implying that the computational effort required for the solution is $O(n^3)$. For a given n^S value, the number of allocation vectors $(n_1, n_2, \ldots, n_m)$ is bounded by $\frac{(2n^S)^m}{(m-1)!}$; see Stirzaker [16]. [Note that due to the fact that the machines are identical, only monotonic vectors are relevant; see e.g. Mosheiov [13].] It follows that an upper bound on the number of assignment problems to be solved (for a given n^S value) is $O(n^m)$. In fact, given the allocation to the first $m-1$ machines, the remaining number of jobs assigned to machine m is completely determined. Hence, this upper bound can be slightly reduced to $O(n^{m-1})$; see Ji and Cheng, 2010 [17]. This procedure needs to be repeated for all possible n^S values, i.e., $n^S = 0, 1, \ldots, n$. Thus, the total number of the assignment problems to be solved is bounded by $nO(n^{m-1})$, which is $O(n^m)$. It follows that an upper bound on the total running time is $O(n^{m+3})$. Hence, for a given number of machines m, the running time is polynomial in the number of jobs n. □

Table 1
Job-processing times and rejection costs for Example 1.

j	p_{j1}	p_{j2}	p_{j3}	p_{j4}	p_{j5}	p_{j6}	p_{j7}	δ_j
1	9	16	23	135	9	16	23	39
2	4	13	27	99	4	13	27	22
3	3	13	28	87	3	13	28	26
4	10	16	24	120	10	16	24	32
5	9	18	30	178	9	18	30	11
6	3	13	24	319	3	13	24	16
7	6	20	21	240	6	20	21	35

Example 1. For the demonstration of the solution procedure, we solve a 3-machines 7-jobs problem. The processing time and the rejection costs are given in Table 1. [It is easily verified that for this input, no more than 3 jobs are assigned to any of the machines. The reason is the very large processing times of all the jobs in position 4. It follows that a total of only 17 assignment problems need to be solved.] The resulting optimal job sequence is: (2, 1) on machine 1, (3, 4) on machine 2, job 7 on machine 3, and jobs 5 and 6 are rejected. The total cost (flow-time plus rejection cost) is 79.

4. Total load with rejection (*TLRC*)

The objective function in this section is minimum total load (i.e. minimum work time of all the machines) plus the rejection cost. As in the previous section, we show that the problem is reduced to a linear assignment problem for a given number of rejected jobs and a given allocation of the scheduled jobs to the machines. We use the same notation: n^S the number of scheduled jobs, n^R the number of the rejected jobs, and $(n_1, n_2, \ldots, n_m)$, the allocation vector of jobs to machines. The size of the assignment matrix is identical to that of minimum total completion time (i.e. $n \times (n^S + n)$), however the cost values are clearly different. The first block of the matrix contains the job-position processing times, i.e. P_{ijr}, $i = 1,\ldots,m$, $j = 1,\ldots,n$, $r = 1,\ldots,n_i$, where the second $(n \times n)$ block of the matrix is identical to the second block of the minimum total completion time matrix.

It follows that the cost matrix is

$$cost_{ijr} = \begin{cases} P_{ijr}, & i = 1,\ldots,m,\ j = 1,\ldots,n, \\ & r = 1,\ldots,n_i, \\ \infty, & i = m+1,\ j = 1,\ldots,n, \\ & r = 1,\ldots,n,\ r \neq j, \\ \delta_j, & i = m+1,\ j = 1,\ldots,n,\ r = j. \end{cases} \quad (9)$$

Given the cost matrix (9), the assignment problem is (4) subject to (5), (6), (7), (8).

Theorem 2. *For a given number of machines, problem TLRC is solved in polynomial time in the number of jobs.*

Proof. As in the previous case, it is clear that the above assignment problem solves *TLRC*. Again, the size of the assignment matrix is bounded by $O(n \times n)$. The upper bound on the number of assignment problems to be solved is $O(n^{m-1})$. Hence, an upper bound on the total running time remains $O(n^{m+3})$. This running time is polynomial in n for a given m value. □

5. General deterioration with rejection

Starting with Browne and Yechiali [18], many researchers have studied scheduling problems in which the job-processing times deteriorate, i.e. increase when they are delayed. Mosheiov [13] introduced a special type of deterioration as a function of the job position (also known as *aging* effect). In this section we focus on this special case, i.e. we assume that P_{jr} is a non-decreasing function of the position r for $j = 1, \ldots, n$. It should be noted that we allow any monotonic function to reflect the job-processing times, and do not restrict ourselves to a specific form as commonly assumed in many previous studies. In the following we solve both *TCRC* and *TLRC* assuming an aging effect.

A trivial property of an optimal schedule in the case of (general) aging, is the *equal allocation* of jobs to machines, i.e. $|n_i - n_k| \leqslant 1$, $i, k = 1, \ldots, m$. Thus, the maximum number of jobs per machine is $\lceil n/m \rceil$. This important property enables us to obtain an optimal solution for this special case by solving a *single* assignment problem.

As in the previous sections, the assignment matrix contains two blocks. The first block consists of $\lceil n/m \rceil$ potential positions for some machines, and $\lfloor n/m \rfloor$ potential positions for the others (such that the total number of potential positions in this block equals n). [Specifically, we solve the equation $k\lceil n/m \rceil + (m-k)\lfloor n/m \rfloor = n$ for k, and the resulting block sizes are: $n_i = \lceil n/m \rceil$, $i = 1, \ldots, k$, and $n_i = \lfloor n/m \rfloor$, $i = k+1, \ldots, m$.]

First, we focus on the cost matrix used to minimize the *total completion time*. Again, the values in the first block are the job-processing times multiplied by their positional weights ($w_{ir} = n_i - r + 1$, $1 \leqslant r \leqslant n_i$). The $n \times n$ second block is identical to the one described in Section 3 (containing the δ_i values on the diagonal, and infinity in all other entries). Given the definitions of n_i and of w_{ir}, we obtain the same assignment matrix (3), and the assignment problem is minimum (4), subject to (5), (6), (7), (8).

It is easy to show that the equal allocation property is valid for the *total load* problem as well. We thus define the n_i values as above: $n_i = \lceil n/m \rceil$, $i = 1, \ldots, k$, and $n_i = \lfloor n/m \rfloor$, $i = k+1, \ldots, m$. Given these, we create the relevant assignment matrix (9). The single assignment problem that needs to be solved is minimum (4) subject to (5), (6), (7), (8).

Theorem 3. *Problems TCRC and TLRC with an aging effect are solved in $O(n^3)$ time.*

Proof. The above assignment problems solve *TCRC* and *TLRC*, respectively. In both cases, the size of the assignment matrix is $n \times (2n)$. A single assignment problem of this size should be solved, implying a total running time of $O(n^3)$. □

6. Conclusion

We studied scheduling with rejection and general position-dependent processing times on parallel identical machines. Two objective functions were considered: (i) minimum total flow-time and the cost of the rejected jobs, and (ii) minimum total load and the cost of the rejected jobs. We introduced efficient algorithms that require a solution of a set of non-standard linear assignment problems. In both cases, the algorithms run in $O(n^{m+3})$ time (which is polynomial for a given number of machines). For the special case of aging, faster ($O(n^3)$) algorithms are introduced.

To the best of our knowledge, scheduling problems with both job-rejection and position-dependent processing times have never been studied in the past. Future research may thus include other combinations of scheduling measures and rejection cost, or other machine settings.

Acknowledgement

The second author is the Charles Rosen Chair of Management, The School of Business Administration, The Hebrew University. This paper was supported in part by the Recanati Fund of The School of Business Administration, The Hebrew University, Jerusalem, Israel.

References

[1] S. Li, J. Yuan, Parallel-machine scheduling with deteriorating jobs and rejection, Theoretical Computer Science 411 (2010) 3642–3650.
[2] S. Sengupta, Algorithms and approximation schemes for minimum lateness/tardiness scheduling with rejection, Algorithms and Data Structures 2748 (2003) 79–90.
[3] Y. Bartal, S. Leonardi, A. Marchetti-Spaccamela, J. Sgall, L. Stougie, Multiprocessor scheduling with rejection, SIAM Journal on Discrete Mathematics 13 (2000) 64–78.
[4] L. Epstein, J. Noga, G.J. Woeginger, On-line scheduling of unit time jobs with rejection: minimizing the total completion time, Operations Research Letters 30 (2002) 415–420.
[5] H. Hoogeveen, M. Skutella, G.J. Woeginger, Preemptive scheduling with rejection, Mathematical Programming 94 (2003) 361–374.
[6] D.W. Engels, D.R. Karger, S.G. Kolliopoulos, S. Sengupta, R.N. Uma, J. Wein, Techniques for scheduling with rejection, Journal of Algorithms 49 (2003) 175–191.
[7] Z. Cao, X. Yang, A PTAS for parallel batch scheduling with rejection and dynamic job arrivals, Theoretical Computer Science 410 (2009) 2732–2745.
[8] Y. Cheng, S. Sun, Scheduling linear deteriorating jobs with rejection on a single machine, European Journal of Operational Research 194 (2009) 18–27.
[9] G. Mosheiov, A. Sarig, Scheduling and due-date assignment problems with job rejection, Foundations of Computing and Decision Sciences 34 (2009) 193–208.
[10] D. Shabtay, N. Gaspar, L. Yedidsion, A bicriteria approach to scheduling a single machine with job rejection and positional penalties, Journal of Combinatorial Optimization 23 (4) (2012) 395–424.
[11] D. Shabtay, N. Gasper, Two-machine flow-shop scheduling with rejection, Computers & Operations Research 39 (2012) 1087–1096.
[12] L.Q. Zhang, L.F. Lu, C.T. Ng, The unbounded parallel-batch scheduling with rejection, Journal of the Operational Research Society 63 (2012) 293–298.

[13] G. Mosheiov, Parallel machine scheduling with a learning effect, Journal of the Operational Research Society 52 (2001) 1165–1169.
[14] D. Biskup, Single-machine scheduling with learning considerations, European Journal of Operational Research 115 (1999) 173–178.
[15] G. Mosheiov, A note: Multi-machine scheduling with general position-based deterioration to minimize total load, International Journal of Production Economics 135 (2012) 523–525.
[16] D. Stirzaker, Elementary Probability, Cambridge University Press, Cambridge, 1995.
[17] M. Ji, T.C.E. Cheng, Scheduling with job-dependent learning effects and multiple rate-modifying activities, Information Processing Letters 110 (2010) 460–463.
[18] S. Browne, U. Yechiali, Scheduling deteriorating jobs on a single processor, Operations Research 38 (1990) 495–498.