Computational and Applied Mathematics (2020) 39:107
https://doi.org/10.1007/s40314-020-1130-z

Regular scheduling measures on proportionate flowshop with job rejection

Baruch Mor[1] · **Dana Shapira**[2]

Received: 11 September 2019 / Revised: 19 February 2020 / Accepted: 4 March 2020 /
Published online: 17 March 2020

Abstract
In this article, we study the method of job rejection in the setting of proportionate flowshop, and focus on minimizing regular performance measures, subject to the constraint that the total rejection cost cannot exceed a given upper bound. In particular, we study total completion time, maximum tardiness, total tardiness, and total weighted number of tardy jobs. All the addressed problems are NP-hard as their single machine counterpart are known to be NP-hard. To the best of our knowledge, there are no detailed solutions in scheduling literature to the first two problems, whereas the last two problems were never addressed to date. For each problem, we provide a pseudo-polynomial dynamic programming solution algorithm, and furthermore, we enhance the reported running time of the first two problems. Our extensive numerical study validates the efficiency of the provided solutions.

Keywords Scheduling · Proportionate flowshop · Regular performance measures · Job rejection · Dynamic programming

Mathematics Subject Classification 90B35

1 Introduction

Flowshop setting is characterized by several *different* machines in series and each job must undergo an operation on each of the machines. A special case of the flowshop environment is the *proportionate flowshop* (PFS) setting where the processing times are *machine-independent* and the storage between consecutive machines is assumed to be *unlimited*. This setting replicates many production systems where each operation is relatively short and the goods are small in size. Thus, each machine can start its operation on the next job

Communicated by Hector Cancela.

✉ Baruch Mor
baruchm@ariel.ac.il

[1] Department of Economics and Business Administration, Ariel University, 40700 Ariel, Israel

[2] Department of Computer Science, Ariel University, Ariel, Israel

in the queue regardless of the status of the job on the succeeding machine, implying that the production line is not susceptible to blocking.

The PFS machine setting can be seen as a generalization of the single machine setting and its importance is revealed by the comprehensive review of Panwalkar et al. (2013) and the following recent studies by Mor and Mosheiov (2014), Panwalkar and Koulamas (2015, 2017), Cheng et al. (Cheng et al. 2018), Gerstl et al. (2019). Quite a few solutions to classical problems on a single machine can be applied, in a straightforward manner, to their PFS equivalents. Such problems include, among others, minimizing the total completion time, minimizing the total tardiness, and minimizing the number of tardy jobs (see Pinedo 2016). In other problems, the sequencing of the jobs may cause idle times between consecutive jobs, affecting the complexity of the solution. Several solutions thus consider a more involved approach, such that solutions to the problem of maximizing the number of just-in-time jobs (Gerstl et al. 2015), the minmax earliness problem (Mor and Mosheiov 2015a), minimizing the number of early jobs (Mor and Mosheiov 2015b), and minsum/minmax common flow allowance (Mor and Mosheiov 2016a).

Several recent articles, Shabtay and Oron (2016), Li et al. (2017), Fiszman and Mosheiov (2018) and Mor et al. (2019), combine the setting of PFS machine with the method of *job rejection*. This method is usually implemented in manufacturing systems to overcome overloaded assembly lines. Thus, the operation management is given the option to process a subset of the jobs, and reject, or alternatively out-source, the complementary subset. Utilizing this method may decrease the overload, optimize the utilization of the inputs, improve the time to market, and eventually increase the profit. To simulate real-life situations, each rejected job incurs a job-dependent penalty. Hence, the given total rejection cost is limited and may impact the profit of the production plant, suggesting solutions based on analytical methods. The intensive research on job rejection is exposed by the survey of Shabtay et al. (2013) as well as later papers: Shabtay (2014), Gerstl and Mosheiov (2015), Koulamas and Panwalkar (2015), Mor and Mosheiov (2016b), Agnetis and Mosheiov (2017), Gerstl et al. (2017), Zhong et al. (2017), and Mor and Mosheiov (2018).

Recently, Mor and Shapira (2019) improved the computational complexity of two problems studied in Shabtay and Oron (2016): minimizing the makespan subject to an upper bound on the total rejection cost, and minimizing the total rejection cost subject to a constraint on the makespan. In the current paper, we complement the results of the last two articles by addressing four regular performance measures, that is, when the objective function is a non-decreasing function of the completion times of the jobs. In more detail, the objective functions studied here are total completion time, maximum tardiness, total tardiness, and total weighted number of tardy jobs. The goal is to minimize these objective functions subject to a constraint on the total rejection cost. All the addressed problems are NP-hard as their single machine counterpart variants were proved to be NP-hard by Zhang et al. (2010). To the best of our knowledge, there are no detailed solutions in scheduling literature to the first two problems, whereas the last two problems were never addressed. For each problem, we, therefore, provide a pseudo-polynomial dynamic programming (DP) solution algorithm; furthermore, we enhance the reported running time of the first two problems. Our extensive numerical study demonstrates the DPs ability to solve real-life instances efficiently.

The paper is organized as follows. The notations are presented in Sect. 2. Sections 3–6 address our proposed solutions to the problems of minimizing the total completion time, maximum tardiness, total tardiness, and total weighted number of tardy jobs, respectively. The conclusions are finally provided in Sect. 7.

2 Notations

Assume that there are given n jobs to be processed on an m-machine PFS. The processing times of the jobs are independent of the machine they are processed on, i.e., $p_{ij} = p_j, i = 1, \ldots, m, j = 1, \ldots, n$.

Let A and $\bar{A}$ denote the subsets of accepted and rejected jobs, respectively. Clearly, their intersection is empty and their union is the complete set of jobs.

The notations p_{max} and P are used to denote the processing time of the longest job between those jobs that are processed and the total processing time of all jobs, correspondingly, i.e., $p_{max} = max_{j\in A}\{p_j\}$ and $P = \sum_{j=1}^{n} p_j$.

Each job is assigned a rejection cost e_j, and contributes this value to the total rejection cost in case it is rejected. The latter quantity is upper bounded by a predefined constant E; formally, the restriction is that $\sum_{j\in\bar{A}} e_j \leq E$.

Given a schedule of the processed jobs, the notation C_{ij} denotes the completion time of job $j, j \in A$ on machine $i, i = 1, \ldots, m$. The completion time of the last operation of job j, i.e., on machine m is denoted by C_j, i.e., $C_j \equiv C_{mj}$. The makespan on a given m-machine PFS is attained by $C_{max} = (m-1)max_{1\leq j\leq n}\{p_j\} + \sum_{j=1}^{n} p_j = (m-1)max_{1\leq j\leq n}\{p_j\} + P$ (see Pinedo 2016) and does not depend on a certain order. Thus, the makespan of the *accepted jobs* is $C_{max} = (m-1)p_{max} + \sum_{j\in A} p_j$.

Let d_j denote the due-date of job $j, j = 1, \ldots, n$. The tardiness of job j (on machine m) is defined as $T_j = \max\{0, C_j - d_j\}$, and the maximum tardiness, among all accepted jobs, is given as $T_{max} = max_{j\in A}\{T_j\}$.

Let, U_j denote the tardiness unit penalty of job $j, j = 1, \ldots n$, such that $U_j = 1$ if $T_j > 0$ and $U_j = 0$, otherwise. In addition, we denote by w_j the weight (relative importance) of job $j, j = 1, \ldots, n$. Thus, the total weighted number of the (accepted) tardy jobs is defined as $\sum_{j\in A} w_j U_j$.

All the objective functions which we focus on are regarded as regular performance measures, i.e., non-decreasing functions of the jobs' completion times. In this paper, our goal is to minimize each of these measures subject to the constraint that the total rejection cost cannot be greater than a given upper bound E. In the following, we utilize the three-field notation introduced by Graham et al. (1979), to formulate the four addressed problems.

In our first problem, the scheduling measure is total completion time, that is:

$$P1 : F_m/p_{ij} = p_j, \text{rej}, \sum_{j\in\bar{A}} e_j \leq E/\sum_{j\in A} C_j.$$

Next, our aim is to minimize the maximum tardiness:

$$P2 : F_m/p_{ij} = p_j, \text{rej}, \sum_{j\in\bar{A}} e_j \leq E/T_{\max}.$$

To formulate the last two problems, we define a common due-date, denoted by d, that is shared by all jobs. Thus, in the third problem, the objective is to minimize the total tardiness with a common due-date, that is:

$$P3 : F_m/p_{ij} = p_j, d_j = d, \text{rej}, \sum_{j\in\bar{A}} e_j \leq E/\sum_{j\in A} T_j.$$

Finally, the aim is to minimize the total weighted number of tardy jobs with a common due-date:

$$P4: F_{\mathrm{m}}/p_{ij} = p_j, d_j = d, \text{rej}, \sum_{j\in\bar{A}} e_j \le E/\sum_{j\in A} w_j U_j.$$

3 Problem $P1: F_{\mathrm{m}}/p_{ij} = p_j$, rej, $\sum_{j\in\bar{A}} e_j \le E/\sum_{j\in A} C_j$.

Problem 1/rej, $\sum_{j\in\bar{A}} e_j \le E/\sum_{j\in A} C_j$ was reported by Shabtay et al. (2012) to be solved in $O\left(n^2 \sum_{j\in\bar{A}} e_j\right)$ time, although no explicit solution was provided. In this section, we provide a DP, which is faster by a factor of n, i.e., $O\left(n \sum_{j\in\bar{A}} e_j\right)$, even for the setting of PFS and not only for a single machine environment.

The order of the jobs in an optimal instance of the classical problem $1 // \sum C_j$ is known to be in accordance with the Shortest Processing Time first (SPT) rule. The same order of the jobs is also valid for the PFS setting (see Pinedo 2016). As stated in Sect. 2, the completion time of job j on a PFS setting is sequence independent, and is given by $C_j = (m-1)\max_{1\le k\le j}\{p_k\} + \sum_{k=1}^{j} p_k = (m-1)p_j + \sum_{k=1}^{j} p_k$, where the last equation follows from the SPT order existence in an optimal schedule.

The DP for $P1$ uses a variable P_j in correspondence to every job j, which is stored in memory. P_j is used to hold the total processing times of the accepted jobs in the subset of jobs $(1, \ldots, j)$ that achieves the minimum total completion time having maximum rejection cost e, and is defined as follows:

$$P_j = \begin{cases} P_{j-1} + p_j, & \text{if job } j \text{ is processed} \\ P_{j-1}, & \text{othewise} \end{cases}, \text{ where } P_0 = 0.$$

Thus, the completion (on machine m) of job j in subset $(1, \ldots, j)$, can be calculated in a straightforward manner using the simple equation, $C_j = (m-1)p_j + P_j$.

Let $f(j, e)$ denote the minimum total completion time for the partial schedule of jobs $1, \ldots, j$ with maximum rejection cost e. At each iteration of the DP, the scheduling cost of jobs 1 to j having an upper bound $e (0 \le e \le E)$ on the rejection cost, is computed, based on the processing costs of jobs 1 to $j-1$, with an upper bound on the rejection cost of either e or $e - e_j$. Thus, at each iteration of the algorithm, the scheduler needs to decide whether to accept or reject job j, as follows:

- Job j must be accepted in case its rejection cost exceeds the current rejection cost limit e.
- Job j may be accepted in case its cost is not greater than the minimum processing cost of jobs 1 to $j-1$, and upon acceptance, the cost is increased by $(m-1)p_j + P_j$.
- Job j may be rejected in case this minimizes the processing cost.

Now, we are ready to present the DP.

Dynamic programming algorithm DP1

$$f(j, e) = \begin{cases} \min\begin{pmatrix} f(j-1, e) + (m-1)p_j + P_j \\ f\left(j-1, e-e_j\right) \end{pmatrix}, & e_j \le e \\ f(j-1, e) + (m-1)p_j + P_j, & \text{othewise} \end{cases} \quad (1)$$

The boundary conditions are:

$$f(j, 0) = (m-1)p_j + \sum_{k=1}^{j} p_k, \quad j, 1 \le j \le n$$

$$f(0, e) = 0, \quad e, 0 \le e \le E.$$

The optimal solution is given as $f(n, E)$.

Theorem 1 *Algorithm* DP1's *complexity is* $O(nE)$.

Proof The recursive formula for $f(j, e)$ given in (1) is processed by a nested loop for each job $j, 1 \leq j \leq n$, and each rejection cost $e \leq E$. The solution is reconstructed by means of backtracking, starting at the last cell in which $j = n$ and $e = E$, and ending at the first cell where $j = e = 0$. Summing up the processing times of both stages, the computational complexity of $DP1$ is $O(nE) + O(n + E) = O(nE)$. □

Example 1 The following illustration is given for clarity.

Let $m = 4,\ n = 6, E = 54$, and assume that the jobs are sequenced in SPT order and renumbered.

The processing times are $p = (13, 14, 17, 26, 27, 45)$.

The job-dependent rejection costs are $e = (18, 22, 14, 10, 9, 38)$.

The solution attained from running DP1 is accepting the first, second, and sixth jobs, having $\sum_{j\in A} C_j = 328$, while rejecting the third, fourth, and fifth jobs with $\sum_{j\in \bar{A}} e_j = 33 \leq 35 = E$.

Numerical study To evaluate our proposed solution in practice, we ran DP1 on several numerical, randomly generated, instances. As the running times are uninfluenced by the number of machines, we restricted the illustrations to four machines only. However, being the number of jobs the main parameter of the running time, we considered an increasing size of the set of input jobs: $n = 500,\ 1000,\ 1500$ and 2000 jobs. The processing times of all jobs and their corresponding rejection costs were uniformly generated in the range of [1, 50]. To avoid trivial instances, i.e., solutions where the majority of the jobs are either accepted or rejected, we generated the values of E in three different intervals. These intervals guaranty challenging instances with approximately equal numbers of accepted and rejected jobs, as expected in real-life circumstances. As the maximum rejection cost in this setting is restricted to $\bar{e} = 50$, the total rejection upper bound cost, E, was generated uniformly in the intervals $[0.02, 0.03]n\bar{e}$, $[0.050, 0.055]n\bar{e}$, and $[0.08, 0.10]n\bar{e}$, simulating rejection of approximately 20%, 30%, and 40% jobs, respectively. Twenty random instances were created for each pair of n and E and then fed to the proposed algorithm.

C ++ executed on an Intel (R) Core ™ i7-8650U CPU @ 1.90 GHz 16.0 GB RAM platform. Table 1 presents the average- and worst-case running times in milliseconds. The number of jobs, n, is given in the first column, and the intervals controlling the maximal rejection cost, E, are given in the second column. The third and fourth columns present the average and worst-case running times, respectively. Although it is not unexpected, it is noteworthy to observe that the running times increase as the values of n and E increase. This phenomenon is due to the tabulation approach, which is standard practice in solving DP algorithms. In this approach, all sub-problems are solved and stored in memory. In the particular case of DP1, the size of used memory is $n \times E$ (See Theorem 1). This characteristic is true for all the numerical studies presented throughout this study. The results indicate that DP1 is extremely efficient and can solve large-sized problems. In particular, the worst-case running time for instances of 2000 jobs and 40% of rejected jobs did not exceed 138 milliseconds (ms).

4 Problem $P2 : F_m/p_{ij} = p_j$, rej, $\sum_{j\in \bar{A}} e_j \leq E/T_{\max}$

The second problem dealt within this paper is minimizing the maximum tardiness, limited by an upper bound on the cost of all rejected jobs, and improve the latest results reported in

Table 1 Average- and worst-case running times of DP1 algorithm for Problem $P1$

n	E	Average running time (ms)	Worst-case running time (ms)
Approximately 20% rejected jobs			
500	[500, 750]	2	3
1000	[1000, 1500]	8	10
1500	[1500, 2250]	17	22
2000	[2000, 3000]	33	47
Approximately 30% rejected jobs			
500	[1250, 1375]	4	5
1000	[2500, 2750]	18	21
1500	[3750, 4125]	39	45
2000	[5000, 5500]	70	89
Approximately 40% rejected jobs			
500	[2000, 2500]	8	9
1000	[4000, 5000]	29	33
1500	[6000, 7500]	69	85
2000	[8000, 10000]	122	138

scheduling theory. Shabtay and Oron (2016) suggest utilizing their $O\left(n^2PE\right)$ time, Pareto optimal solution, to solve $P2$. In the following, we prove that $P2$ can be solved in $O(nPE)$.

First, we prove that, although idle time between consecutive jobs may exist, similarly to $1/T_{\max}$, also $F_{\mathrm{m}}/p_{ij} = p_j/T_{\max}$ can be optimally solved by sequencing the jobs in Earliest Due-Date first (EDD) order.

Property 1 *There exists an optimal schedule for $F_{\mathrm{m}}/p_{ij} = p_j/T_{\max}$ in which the jobs are sequenced in EDD order.*

Proof Let schedule π_1 be an optimal schedule which is not sequenced in EDD order. Therefore, there exist a pair of consecutive jobs in π_1 that are scheduled in reverse due-date order. Let k and l denote the first pair of jobs that violates the EDD order and are scheduled at positions i and $i+1$, respectively. Recall that d_k and d_l denote the due-dates of jobs k and l, respectively.

We differentiate between several cases depending on the values of p_l and p_k.

Case 1 $\max_{1\le j\le i-1}\left\{p_j\right\} \ge p_k, p_l$. Due to the dominance of the largest job on the completion times of jobs k and l, there is no idle time between the jobs in schedule π_1.

Let t be the starting time of job k on machine m in the optimal schedule π_1. The contribution cost of jobs k and l in π_1 is

$$Z_{\pi_1}(k,l) = \max\{t + p_k - d_k, t + p_k + p_l - d_l\}.$$

Construct a schedule π_2 by a standard pair-wise interchange of jobs k and l and similar to schedule π_1, there is no idle time between these jobs:

$$Z_{\pi_2}(k,l) = \max\{t + p_l - d_l, t + p_l + p_k - d_k\}.$$

We prove that $Z_{\pi_1}(k,l) \ge Z_{\pi_2}(k,l)$ by showing that:

$$t + p_k + p_l - d_l \geq \max\{t + p_l - d_l, t + p_l + p_k - d_k\}.$$

The latter is true due to the following inequalities:

i. $t + p_k + p_l - d_l \geq t + p_l - d_l$;
ii. $t + p_k + p_l - d_l \geq t + p_l + p_k - d_k$, since, p_k, d_l and d_k are all positive, and from our assumption that $d_k \geq d_l$.

Case 2 $\max_{1\leq j\leq i-1}\{p_j\} < p_l < p_k$.

Since $p_l < p_k$, there is no idle time between the jobs in schedule π_1.

Let t be the starting time of job k on machine 1 in schedule π_1, and thus, the cost contribution of jobs k and l in π_1 is:

$$Z_{\pi_1}(k, l) = \max\{t + m \times p_k - d_k, t + m \times p_k + p_l - d_l\}.$$

Obtain a schedule π_2 by swapping jobs k and l. Since $p_l < p_k$, there is an idle time between the jobs in schedule π_2.

$$Z_{\pi_2}(k, l) = \max\{t + m \times p_l - d_l, t + p_l + m \times p_k - d_k\}.$$

We prove that $Z_{\pi_1}(k, l) \geq Z_{\pi_2}(k, l)$ by showing that:

$$t + m \times p_k + p_l - d_l \geq \max\{t + m \times p_l - d_l, t + p_l + m \times p_k - d_k\}.$$

The latter is true due to:

i. $(i)\ t + m \times p_k + p_l - d_l \geq t + m \times p_l - d_l$, as $p_k > p_l$;
ii. $(ii)\ t + m \times p_k + p_l - d_l \geq t + p_l + m \times p_k - d_k$, as $d_l \leq d_k$.

Case 3 $\max_{1\leq j\leq i-1}\{p_j\} < p_k < p_l$.

Since $p_k < p_l$ there is an idle time between the jobs in schedule π_1.

Let t be the starting time of job k on machine 1 in schedule π_1, and thus, the cost contribution of jobs k and l in π_1 is:

$$Z_{\pi_1}(k, l) = \max\{t + m \times p_k - d_k, t + p_k + m \times p_l - d_l\}.$$

Obtain a schedule π_2 by interchanging jobs k and l. Since $p_l > p_k$, there is no idle time between the jobs, and the cost contribution of jobs k and l in π_2 is:

$$Z_{\pi_2}(k, l) = \max\{t + m \times p_l - d_l, t + m \times p_l + p_k - d_k\}.$$

We prove that $Z_{\pi_1}(k, l) \geq Z_{\pi_2}(k, l)$ by showing that:

$$t + p_k + m \times p_l - d_l \geq \max\{t + m \times p_l - d_l, t + m \times p_l + p_k - d_k\}.$$

The latter is true due to:

i. $(i)\ t + p_k + m \times p_l - d_l \geq t + m \times p_l - d_l$;
ii. $(ii)\ t + p_k + m \times p_l - d_l \geq t + m \times p_l + p_k - d_k$.

The remaining cases, i.e., Case 4: $p_l < \max_{1\leq j\leq i-1}\{p_j\} < p_k$ and Case 5: $p_k < \max_{1\leq j\leq i-1}\{p_j\} < p_l$ are similar to Case 2 and Case 3, respectively, and are thus omitted.

It follows that in all cases, π_2 contradicts the optimality of π_1, which completes the proof. □

To facilitate the DP, we use a variable $(p_{max})_j$, that holds the largest processing time of a subset $(1, \ldots, j)$ that achieves the minimal tardiness with maximum rejection cost e. The variable $(p_{max})_j$ is defined as follows:

$$(p_{max})_j = \begin{cases} \max\{(p_{max})_{j-1}, p_j\}, & \text{if job } j \text{ is processed} \\ (p_{max})_{j-1}, & \text{othewise} \end{cases}, \text{ where } (p_{max})_{j-1} = 0.$$

Let $f(j, t, e)$ denote the minimal tardiness for the partial schedule of jobs $1, \ldots, j$ with total processing time, t, and maximum rejection cost, e. Likewise to DP1 at each iteration of the DP, the scheduling cost of jobs 1 to j having an upper bound $e(0 \le e \le E)$ on the rejection cost is computed, based on the processing times and costs of jobs 1 to $j-1$, with an upper bound rejection cost of either e or $e - e_j$, depending on whether it is possible to reject job j (i.e., its rejection cost does not exceed the current rejection cost limit e), and whether the resulting minimum tardiness cost can only be improved by this rejection. Intuitively, variable t is used to track the total time of the processed jobs in the examined schedule.

The completion time of job j (on machine m) in the partial set $(1, \ldots, j)$, with total processing time, t, is given as $C_j = (m-1)(p_{max})_j + t$.

Next, we present the formal DP by its recursion formula.

Dynamic programming algorithm DP2

$$f(j,t,e) = \begin{cases} \min\left(\begin{array}{l} \max\left(\begin{array}{l} f\big(j-1, t-p_j, e\big), \\ \max\{0, C_j - d\} \end{array} \right), \\ f\big(j-1, t, e-e_j\big) \end{array} \right), & p_j \le t \text{ and } e_j \le e \\ \max\left(\begin{array}{l} f\big(j-1, t-p_j, e\big), \\ \max\{0, C_j - d\} \end{array} \right), & p_j \le t \text{ and } e_j > e \\ f\big(j-1, t, e-e_j\big), & p_j > t \text{ and } e_j \le e \\ \infty, & p_j \text{ and } e_j > e. \end{cases} \quad (2)$$

The first condition of the recursion reflects the option to accept or reject job j. The second condition refers only to the option of processing job j as e_j exceeds the current rejection cost limit e, implying that job j must be accepted. The third condition refers only to the option of rejecting job j, as t is smaller than p_j. The last line addresses invalid cases implying a cost of ∞.

The boundary conditions are:

$$f(0,0,e) = 0, \quad 0 \le e \le E$$
$$f(0,t,e) = \infty, \quad 0 < t \le P.$$

The optimal solution is given by $\min_{0 \le t \le P, 0 \le e \le E}\{f(n,t,e)\}$

Theorem 2 *The computational complexity of DP2 is $O(nPE)$.*

Proof Using the recursive function in (2), the DP is calculated for every job j, $1 \le j \le n$, for every t, $1 \le t \le P$, and every rejection cost $e \le U$, resulting in an $O(nPE)$ processing time. Reconstructing the solution is done by backtracking, finding the minimum cost, $f(n,t,e)$, in $O(PE)$ time, and starting at the found minimum and ending at $f(0,0,0)$, for an addition of $O(n+P+U)$ operations. We conclude that the total processing time is $O(nPE)$. □

Example 2 Consider the following instance of the problem, $m = 4$, $n = 6$, $E = 27$, and jobs with due-dates $d = (38, 57, 76, 90, 118, 123)$, sequenced in EDD order, and renumbered.

The processing times are $p = (18, 34, 18, 17, 8, 44)$.

The job-dependent rejection costs are $e = (3, 31, 8, 31, 25, 46)$.

Executing DP2, we obtain that in an optimal solution, the set of accepted jobs is $A = (2, 4, 5, 6)$. The tardiness of the jobs is $T_2 = 79$, $T_4 = 63$, $T_5 = 43$, $T_6 = 112$, implying that $T_{max} = 112$.

Table 2 Average- and worst-case running times of DP2 algorithm for Problem $P2$

n	E	Average running time (s)	Worst-case running time (s)
Approximately 5% rejected jobs			
25	[6, 12]	0.002	0.003
50	[12, 25]	0.013	0.014
75	[18, 37]	0.041	0.058
100	[25, 50]	0.089	0.124
125	[31, 62]	0.179	0.221
Approximately 10% rejected jobs			
25	[18, 25]	0.003	0.003
50	[37, 50]	0.024	0.031
75	[56, 75]	0.089	0.096
100	[75, 100]	0.185	0.227
125	[93, 125]	0.352	0.412
Approximately 15% rejected jobs			
25	[31, 37]	0.004	0.005
50	[62, 75]	0.035	0.041
75	[93, 112]	0.114	0.128
100	[125, 150]	0.273	0.318
125	[156, 187]	0.553	0.646

The set of rejected jobs is $\bar{A} = (1, 3)$ and $\sum_{j\in\bar{A}} e_j = 11 \leq 27 = E$.

Numerical study We performed numerical experiments to measure the running times of $DP2$. For $m = 4$, we generated random instances having $n = 25, 50, 75, 100$ and 125 jobs. The job-processing times and the job-rejection costs were generated uniformly in the interval $[1, 50]$. For each instance, the maximal makespan was calculated: $\overline{C_{max}} = (m-1)\max_{1\leq j\leq n}\{p_j\} + \sum_{j=1}^{n} p_j$. The job *due-dates* were generated uniformly in the interval $[0.10, 0.50]\overline{C_{max}}$. Similar to Sect. 3, $\bar{e} = 50$, and the total rejection upper bound cost, E, was generated uniformly in the intervals $[0.005, 0.010]n\bar{e}$, $[0.015, 0.020]n\bar{e}$ and $[0.025, 0.030]n\bar{e}$, reflecting rejection of approximately 10%, 15%, and 20% jobs, respectively. The rest of the scheme is identical to that given in Sect. 3. The results are presented in Table 2 and demonstrate the efficiency of DP2 to solve real-life instances. We note that the worst-case running time for problems of 125 jobs and 20% rejection rate did not exceed 0.646 seconds (s).

5 Problem $P3 : F_m/p_{ij} = p_j, d_j = d$, rej, $\sum_{j\in\bar{A}} e_j \leq E/\sum_{j\in A} T_j$

To the best of our knowledge, the problem, $F_m/p_{ij} = p_j, d_j = d/\sum T_j$, was not addressed in scheduling theory to date. Recall that the SPT rule is optimal for $1/d_j = d/\sum T_j$. We start by proving that the SPT order is optimal for $Fm/p_{ij} = p_j, d_j = d/\sum T_j$ as well.

Property 2 *There exists an optimal schedule for* $Fm/p_{ij} = p_j, d_j = d/\sum T_j$ *in which the jobs are sequenced in SPT order.*

Proof By a pair-wise interchange argument.

Consider an optimal schedule π_1, where the tardy jobs are not SPT. As above, let k (at position i) and l (at position $i+1$) denote the first pair of consecutive jobs that violates the SPT order, that is $p_l < p_k$. Let $T_j(\pi_1)$ and $P_{i-1}(\pi_1)$ denote the tardiness value of job j and the total processing time of jobs at positions $1, \ldots, i-1$, i.e., $P_{i-1} = \sum_{j=1}^{i-1} p_j$, in schedule π_1:

$$\begin{aligned} T_k(\pi_1) + T_l(\pi_1) &= C_k - d + C_l - d \\ &= \left[\left(P_{i-1}(\pi_1) + p_k + (m-1)\max\left\{\max_{1\le j\le i-1}\{p_j\}, p_k\right\}\right) - d\right] \\ &\quad + \left[\left(P_{i-1}(\pi_1) + p_k + p_l + (m-1)\max_{1\le j\le i+1}\{p_j\}\right) - d\right]. \end{aligned}$$

Obtain a schedule π_2 by a standard pair-wise interchange of jobs k and l.
It follows that:

$$\begin{aligned} &T_k(\pi_1) + T_l(\pi_1) - (T_l(\pi_2) + T_k(\pi_2)) \\ &= p_k - p_l + (m-1)\left[\max\left\{\max_{1\le j\le i-1}\{p_j\}, p_k\right\} - \max\left\{\max_{1\le j\le i-1}\{p_j\}, p_l\right\}\right] > 0. \end{aligned}$$

We conclude that π_2 is optimal as well, which completes the proof. □

We conclude from Property 2 that $(p_{max})_j = p_j$.

Let $f(j,t,e)$ denote the minimal total tardiness for the partial schedule of jobs $1, \ldots, j$, having total processing time t and maximum rejection cost e.

The dynamic programming for $f(j,t,e)$ is as follows:

Dynamic programming algorithm DP3:

$$f(j,t,e) = \begin{cases} \min\left(\begin{array}{l} f\big(j-1, t-p_j, e\big) + \max\big\{0, (m-1)p_j + t - d\big\} \\ f\big(j-1, t, e-e_j\big) \end{array}\right), & p_j \le t \text{ and } e_j \le e \\ f\big(j-1, t-p_j, e\big) + \max\big\{0, (m-1)p_j + t - d\big\}, & p_j \le t \text{ and } e_j > e \\ f\big(j-1, t, e-e_j\big), & p_j > t \text{ and } e_j \le e \\ \infty, & p_j > t \text{ and } e_j > e. \end{cases}$$

The first line of the recursion reflects the option to accept or reject job j. The second line refers to the option of production of job j only and the third line refers to the option of rejection of job j only. The last line addresses unacceptable cases implying a cost of ∞.

The boundary conditions are:

$$f(0,0,e) = 0, 0 \le e \le E$$
$$f(0,t,e) = \infty, 0 < t \le P.$$

The optimal solution is given by $\min_{0\le t\le P, 0\le e\le E}\{f(n,t,e)\}$

Theorem 3 *The computational complexity of DP3 is $O(nPE)$.*

(The proof is similar to that given for DP2 in Sect. 4.)

Example 3 Consider the following instance of the problem, $m = 4, n = 6, E = 28, d = 16$, and the jobs are sequenced in SPT order and renumbered.

Table 3 Average- and worst-case running times of DP3 algorithm for Problem $P3$

n	E	Average running time (s)	Worst-case running time (s)
Approximately 5% rejected jobs			
25	[6, 12]	0.002	0.003
50	[12, 25]	0.010	0.014
75	[18, 37]	0.032	0.044
100	[25, 50]	0.078	0.102
125	[31, 62]	0.143	0.190
Approximately 10% rejected jobs			
25	[18, 25]	0.002	0.003
50	[37, 50]	0.017	0.022
75	[56, 75]	0.062	0.072
100	[75, 100]	0.145	0.171
125	[93, 125]	0.274	0.325
Approximately 15% rejected jobs			
25	[31, 37]	0.003	0.003
50	[62, 75]	0.025	0.028
75	[93, 112]	0.083	0.110
100	[125, 150]	0.195	0.220
125	[156, 187]	0.418	0.481

The processing times are $p = (10, 10, 21, 37, 39, 43)$.

The job-dependent rejection costs are $e = (32, 8, 23, 32, 18, 48)$.

Applying $DP3$, the set of accepted jobs is $A = (1, 3, 4, 6)$.

The tardiness of the accepted jobs is $T_1 = 24$, $T_3 = 78$, $T_4 = 163$, $T_6 = 224$, achieving $\sum_{j\in A} T_j = 489$.

The set of rejected jobs is $\bar{A} = (2, 5)$ and $\sum_{j\in \bar{A}} e_j = 26 \leq 28 = E$.

Numerical study We adapted the arrangement depicted in Sect. 4, except that the common due-date was generated uniformly in the interval $[0.10, 0.50]\overline{C_{max}}$. The results presented in Table 3 reflect the ability of $DP3$ to solve real-life instances, especially as the worst-case running time for problems of 125 jobs and 20% rejection rate did not exceed 0.481 s.

6 Problem $P4 : F_m/p_{ij} = p_j, d_j = d, \text{rej}, \sum_{j\in \bar{A}} e_j \leq E/\sum_{j\in A} w_j U_j$

It is well known that the classical problem $1/d_j = d/\sum w_j U_j$ is equivalent to the *knapsack* problem. Solving $1/d_j = d/\sum w_j U_j$, similarly to knapsack, does not require an initial sequencing of the jobs prior to executing the DP. Unlike the single machine setting, in the PFS setting, the scheduler has to consider the inherent idle times between consecutive jobs and, subsequently, keep track of the longest job in the subset of accepted jobs, $(p_{max})_j$. To simplify the DP and avoid excessive variables, we start by sequencing the jobs in accordance to the SPT rule. From Property 2 in Sect. 5, this step guarantees that $(p_{max})_j = p_j$. Let $f(j, t, e)$ denote the minimal total weighted number of tardy jobs for the partial schedule of jobs $1, \ldots, j$, having processing time t and maximum rejection cost e. At each iteration

of the DP, the scheduling cost of jobs 1 to j, having an upper bound $e(0 \le e \le E)$ on the rejection cost is computed, based on the processing costs of jobs 1 to $j-1$, with an upper bound rejection cost of either e or $e-e_j$.

From the above, the tardiness unit penalty of job j can be determined by the following updated definition:

$$U_j = \begin{cases} 1, T_j = t + (m-1)p_j - d > 0 \\ 0, otherwise \end{cases}.$$

Dynamic programming algorithm DP4:

$$f(j,t,e) = \begin{cases} \min\begin{pmatrix} f(j-1, t-p_j, e) + w_j U_j, \\ f(j-1, t, e-e_j) \end{pmatrix}, & p_j \le t \text{ and } e_j \le e \\ f(j-1, t-p_j, e) + w_j U_j, & p_j \le t \text{ and } e_j > e \\ f(j-1, t, e-e_j), & p_j > t \text{ and } e_j \le e \\ \infty, & p_j > t \text{ and } e_j > e \end{cases}.$$

As for DP3, the first condition of the recursion reflects the option to accept or reject job j. The second condition refers to the option of acceptation only, whereas the third condition refers to the option of rejection only. The last line addresses illegal cases implying a cost of ∞.

The boundary conditions are:

$$f(0,0,e) = 0, \quad 0 \le e \le E$$
$$f(0,t,e) = \infty, \quad 0 < t \le P.$$

The optimal solution is given by $\min_{0 \le t \le P, 0 \le e \le E}\{f(n,t,e)\}$.

Theorem 4 The computational complexity of $DP4$ is $O(nPE)$.

(The proof is similar to that given for DP2 in Sect. 4.)

Example 4 Consider the following instance of the problem, $m = 4, n = 6, E = 33, d = 22$ and the jobs are sequenced in SPT order and renumbered.

The processing times are $p = (31, 31, 37, 40, 44, 45)$.
The job-dependent rejection costs are $e = (42, 43, 21, 25, 31, 6)$.
The job-dependent weights are $w = (20, 7, 10, 11, 24, 20)$.
Executing $DP4$, we obtain that in an optimal solution, the set of accepted jobs is $A = (1, 2, 3, 5)$, obtaining total weighted number of tardy jobs of $\sum_{j \in A} w_j U_j = 61$.
The set of rejected jobs is $\bar{A} = (4, 6)$ and $\sum_{j \in \bar{A}} e_j = 31 \le 33 = E$.

Numerical study We adapted the scheme presented in Sect. 5, with the addition of job-dependent weights that were generated uniformly in the interval [1, 25]. The results presented in Table 4 validate the efficiency of $DP4$ to solve real-life instances. Specifically, the worst-case running time for problems of 125 and 20% rejection rate jobs did not exceed 0.468 s.

7 Conclusions

In this study, we combined the method of job rejection and the setting of proportionate flow-shop, and focused on minimizing regular performance measures, subject to the constraint that the total rejection cost cannot exceed a given upper bound. In particular, we considered the

Table 4 Average- and worst-case running times of DP4 algorithm for Problem $P4$

n	E	Average running time (s)	Worst-case running time (s)
Approximately 5% rejected jobs			
25	[6, 12]	0.002	0.004
50	[12, 25]	0.011	0.014
75	[18, 37]	0.035	0.043
100	[25, 50]	0.075	0.107
125	[31, 62]	0.142	0.174
Approximately 10% rejected jobs			
25	[18, 25]	0.002	0.003
50	[37, 50]	0.018	0.022
75	[56, 75]	0.058	0.076
100	[75, 100]	0.130	0.155
125	[93, 125]	0.272	0.318
Approximately 15% rejected jobs			
25	[31, 37]	0.003	0.004
50	[62, 75]	0.025	0.033
75	[93, 112]	0.079	0.093
100	[125, 150]	0.195	0.225
125	[156, 187]	0.396	0.468

problems of total completion time, maximum tardiness, total tardiness, and total weighted number of tardy jobs. For each problem, we introduced an efficient pseudo-polynomial dynamic programming solution algorithm. We also conducted extensive numerical studies that demonstrate the DPs ability to solve real-life instances. Challenging future objective functions in the setting of proportionate flowshop with rejection, include, among others, makespan with release dates, total weighted tardiness with a common due-date, and total tardiness. These problems will probably necessitate a different approach, such as applying advanced metaheuristic techniques.

Acknowledgements This research did not receive any specific grant from funding agencies in the public, commercial, or not-for-profit sectors.

References

Agnetis A, Mosheiov G (2017) Scheduling with job-rejection and position-dependent processing times on proportionate flowshops. Optim Lett 11(4):885–892

Cheng C-Y, Ying K-C, Chen H-H, Lin J-X (2018) Optimization algorithms for proportionate flowshop scheduling problems with variable maintenance activities. Comput Ind Eng 117:164–170

Fiszman S, Mosheiov G (2018) Minimizing total load on a proportionate flowshop with position-dependent processing times and job rejection. Inf Process Lett 132:39–43

Gerstl E, Mosheiov G (2015) Single machine scheduling problems with generalised due-dates and job-rejection. Int J Prod Res 55(11):3164–3172

Gerstl E, Mor B, Mosheiov G (2015) A note: maximizing the weighted number of just-in-time jobs on a proportionate flowshop. Inf Process Lett 115(2):159–162

Gerstl E, Mor B, Mosheiov G (2017) Minmax scheduling with acceptable lead-times: extensions to position-dependent processing times, due-window and job rejection. Comput Oper Res 83:150–156

Gerstl E, Mor B, Mosheiov G (2019) Scheduling on a proportionate flowshop to minimise total late work. Int J Prod Res 57(2):531–543
Graham RL, Lawler EL, Lenstra JK (1979) Optimization and approximation in deterministic sequencing and scheduling: a survey. Ann Discr Math 4:287–326
Koulamas C, Panwalkar SS (2015) On the equivalence of single machine earliness/tardiness problems with job rejection. Comput Ind Eng 87:1–3
Li SS, Qian DL, Chen RX (2017) Proportionate Flow Shop Scheduling with Rejection. Asia-Pac J Oper Res 34(4):1750015
Mor B, Mosheiov G (2014) Polynomial time solutions for scheduling problems on a proportionate flowshop with two competing agents. J Oper Res Soc 65(1):151–157
Mor B, Mosheiov G (2015a) A note: minimizing maximum earliness on a proportionate flowshop. Inf Process Lett 115(2):253–255
Mor B, Mosheiov G (2015b) Minimizing the number of early jobs on a proportionate flowshop. J Oper Res Soc 66(9):1426–1429
Mor B, Mosheiov G (2016a) Minsum and minmax scheduling on a proportionate flowshop with common flow-allowance. Eur J Oper Res 254(2):360–370
Mor B, Mosheiov G (2016b) Minimizing maximum cost on a single machine with two competing agents and job rejection. J Oper Res Soc 67(12):1524–1531
Mor B, Mosheiov G (2018) A note: minimizing total absolute deviation of job completion times on unrelated machines with general position-dependent processing times and job-rejection. Ann Oper Res 271(2):1079–1085
Mor B, Mosheiov G, Shapira D (2019) Flowshop scheduling with learning effect and job rejection. J Sched. https://doi.org/10.1007/s10951-019-00612-y
Mor B, Shapira D (2019) Improved algorithms for scheduling on proportionate flowshop with job-rejection. J Oper Res Soc 70(11):1997–2003
Panwalkar SS, Koulamas C (2015) Proportionate flow shop: new complexity results and models with due date assignment. Naval Res Log 62(2):98–106
Panwalkar SS, Koulamas C (2017) On the dominance of permutation schedules for some ordered and proportionate flow shop problem. Comput Ind Eng 107:105–108
Panwalkar SS, Smith ML, Koulamas C (2013) Review of the ordered and proportionate flow shop scheduling research. Naval Res Log 60(1):46–55
Pinedo ML (2016) Scheduling: theory, algorithms and systems, 5th edn. Springer, New-York
Shabtay D (2014) The single machine serial batch scheduling problem with rejection to minimize total completion time and total rejection cost. Eur J Oper Res 233(1):64–74
Shabtay D, Oron D (2016) Proportionate flow-shop scheduling with rejection. J Oper Res Soc 67(5):752–769
Shabtay D, Gaspar N, Yedidsion L (2012) A bicriteria approach to scheduling a single machine with job rejection and positional penalties. J Comb Optim 23:395–424
Shabtay D, Gaspar N, Kaspi M (2013) A survey on offline scheduling with rejection. J Sched 16:3–28
Zhang L, Lu L, Yuan J (2010) Single-machine scheduling under the job rejection constraint. Theoret Comput Sci 411:1877–1882
Zhong X, Pan Z, Jiang D (2017) Scheduling with release times and rejection on two parallel machines. J Comb Optim 33:934–944