mathematics

MDPI

Article

Single-Machine Parallel-Batch Scheduling with Nonidentical Job Sizes and Rejection

Miaomiao Jin, Xiaoxia Liu and Wenchang Luo *

School of Mathematics and Statistics, Ningbo University, Ningbo 315211, China;
1711071004@nbu.edu.cn (M.J.); 1711071005@nbu.edu.cn (X.L.)
* Correspondence: luowenchang@nbu.edu.cn

Received: 30 January 2020; Accepted: 13 February 2020; Published: 16 February 2020

Abstract: We investigate the single-machine parallel-batch scheduling problem with nonidentical job sizes and rejection. In this problem, a set of jobs with different processing times and nonidentical sizes is given to be possibly processed on a parallel-batch processing machine. Each job is either accepted and then processed on the machine or rejected by paying its rejection penalty. Preemption is not allowed. Our task is to choose the accepted jobs and schedule them as batches on the machine to minimize the makespan of the accepted jobs plus the total rejection penalty of the rejected jobs. We provide an integer programming formulation to exactly solve our problem. Then, we propose three fast heuristic algorithms to solve the problem and evaluate their performances by using a small numerical example.

Keywords: parallel-batch scheduling; single machine; job sizes; rejection; makespan

1. Introduction

In this paper, we investigate the single-machine parallel-batch scheduling problem with nonidentical job sizes and rejection. This problem can be addressed as follows. We are given a set of jobs $J=\{J_1,J_2,\cdots,J_n\}$ and a single parallel-batch processing machine with a capacity 1. Each job J_j has a processing time p_j, a size $s_j\in(0,1]$, and a rejection penalty e_j, $j=1,2,\cdots,n$. A job J_j is either accepted and then processed on the machine or rejected by paying a certain penalty e_j, $j=1,2,\cdots,n$. The machine can simultaneously process a number of jobs as a batch subject to the constraint that the total size of jobs in the batch is not larger than the capacity 1 of the machine. The processing time for a parallel batch is defined as the longest processing time of the jobs in the batch. Preemption is not allowed. Our task is to choose the accepted jobs and schedule them as batches on the single machine to minimize the makespan of the accepted jobs plus the total rejection penalty of the rejected jobs. Let A denote the accepted job subset and $R=J-A$ denote the rejected job subset. Given the accepted job subset A and its corresponding schedule σ, let C_j denote the completion time of job J_j when job J_j is accepted. Then, using the general three-field notation for scheduling problem [1], the problem can be denoted by $1|p-batch,s_j,rej|C_{\max}+\sum_{j\in R}e_j$, where $C_{\max}=\max_{j\in A}C_j$. The objective function $C_{\max}+\sum_{j\in R}e_j$ addresses the total operational cost. The first part $C_{\max}$ represents the maximum completion time (i.e., makespan), which describes

the cost for working time. The second part $\sum_{j\in R} e_j$ represents the total rejection penalty, which describes the cost for outsourcing and/or for loss in revenue and customer goodwill. The second part does not have to be scaled since we may always scale the individual rejection penalties.

Our problem falls into the category of single-machine batch scheduling. Since Ikura and Gimple [2] initiated the work on the single-machine batch scheduling, single-machine batch scheduling has been investigated extensively in the past few decades. Brucker et al. [3] studied the single-machine batch processing scheduling problem with the goal to minimize regular scheduling criteria that are nondecreasing on the job completion times. Liu and Yu [4] considered the single-machine batch processing scheduling problem with jobs having release dates and the objective function is to minimize the makespan. Cheng et al. [5] investigated an unbounded batch processing machine scheduling problem with jobs having release dates and deadlines. Liu et al. [6] set the complexity of scheduling an unbounded batch machine. Zhang et al. [7] studied the single-machine batch processing scheduling problem with the goal to minimize the makespan. A survey on batch processing scheduling was given by Potts and Kovalyov [8].

In the above-mentioned studies, the assumption that all the jobs are required to be accepted with processing and no rejection is allowed is made. However, in some actual situations, due to the limited production capacity and tight delivery time requirement, the decision-maker may need to reject some jobs by paying the corresponding rejection penalties. In recent years, some researchers have started to focus on batch processing scheduling with job rejection. Lu et al. [9] studied an unbounded parallel-batch processing machine scheduling problem with release dates and rejection with the goal to minimize the makespan plus the total rejection penalty. They showed that the problem is weakly NP-hard and derived a fully polynomial time approximation scheme (FPTAS). Furthermore, for the bounded parallel-batch processing machine scheduling problem with release dates and rejection, Lu et al. [10] proved that it is strongly NP-hard and presented a two-approximation algorithm and a polynomial time approximation scheme (PTAS); Cao and Yang [11] also derived a PTAS. He et al. [12] considered two bi-criteria single parallel batch processing machine scheduling problems with job rejection being allowed; one is to minimize the makespan as long as the total rejection cost is not larger than a given threshold, and the other is to minimize the total rejection cost as long as the makespan is not larger than a given threshold.

Note that, if all jobs have identical processing time and the rejection penalty for each job is infinity, then the problem is just the classical one-dimensional bin packing problem, which is strongly NP-hard [13]. Naturally, our problem $1|p-batch,s_j,rej|C_{\max}+\sum_{j\in R} e_j$ is strongly NP-hard. Our problem extends the models proposed in Lu et al. [9] and Zhang et al. [7]. By setting each job has the identical size, the model proposed in Lu et al. [9] is our special case. When we set $e_j=+\infty$, the model proposed in Zhang et al. [7] becomes our special case.

The remainder of this paper is organized as follows. In Section 2, we give the integer programming formulation to exactly solve our problem. In Section 3, we proposed three fast heuristic algorithms to solve our problem followed by a numerical example to show the solution results by running these algorithms in Section 4. Finally, we conclude the paper in Section 5.

2. Integer Programming Formulation

In this section, we derive an integer programming formulation for problem

$$1|p-batch,s_j,rej|C_{\max}+\sum_{j\in R} e_j .$$

Clearly, there are at most n batches in our problem. The decision variables of the problem are as follows:

y_i: equals 1 if the i th batch is opened; 0 otherwise, $i=1,2,\cdots,n$.

l_i: denotes the longest processing time in the i th batch if $y_i=1$; 0 otherwise, $i=1,2,\cdots,n$.

z_j : equals 1 if job J_j is rejected; 0 otherwise, $j = 1, 2, \cdots, n$.

x_{ij} : equals 1 if the i th batch contains job J_j ; 0 otherwise, $i, j = 1, 2, \cdots, n$.

Let $p_{\max} = \max_{j=1}^{n} p_j$. Then, we have the following integer programming (IP) formulation to describe problem $1 \mid p-batch, s_j, rej \mid C_{\max} + \sum_{j \in R} e_j$.

IP Formulation:

Minimize

$$\sum_{i=1}^{n} y_i l_i + \sum_{j=1}^{n} z_j e_j \tag{1}$$

Subject to

$$\sum_{i=1}^{n} x_{ij} + z_j = 1, j = 1, 2, \cdots, n; \tag{2}$$

$$\sum_{j=1}^{n} x_{ij} s_j \le 1, i = 1, 2, \cdots, n; \tag{3}$$

$$x_{ij} \le y_i, i, j = 1, 2, \cdots, n; \tag{4}$$

$$x_{ij} p_j \le l_i, i, j = 1, 2, \cdots, n; \tag{5}$$

$$y_i \le l_i \le y_i p_{\max}, i = 1, 2, \cdots, n; \tag{6}$$

$$x_{ij} \in \{0, 1\}, i, j = 1, 2, \cdots, n; \tag{7}$$

$$y_i \in \{0, 1\}, i = 1, 2, \cdots, n; \tag{8}$$

$$z_j \in \{0, 1\}, j = 1, 2, \cdots, n. \tag{9}$$

The objective function in Equation (1) tries to minimize the makespan (maximum completion time) of the accepted jobs plus the total rejection penalty of the rejected jobs. The constraint set in Equation (2) ensures that each job is either accepted to process in some batch or rejected. The constraint set in Equation (3) describes that the total size of the accepted jobs assigned into the same batch is not greater than the capacity size 1. The constraint set in Equation (4) states that there exists a job assigned into some batch if and only if this batch is opened. The constraint set in Equation (5) guarantees that the processing time of each job assigned into some batch is not greater than the longest processing time in this batch. The constraint set in Equation (6) gives the lower bound and upper bound for the longest processing time in some batch. Finally, the constraint sets in Equations (7)–(9) impose the binary restriction on x_{ij} , y_i , and z_j , $i, j = 1, 2, \cdots, n$.

Note that the problem $1 \mid p-bath, s_j, rej \mid C_{\max} + \sum_{j \in R} e_j$ includes the well-known strongly NP-hard problem Bin Packing [13] as its special case. Thus, exactly solving its IP Formulation is also strongly NP-hard [13]. It is impossible to solve it exactly in polynomial time according to the NP-completeness theory [13]. Thus, we turn to design fast heuristic algorithms to obtain an approximate feasible solution. However, if one relaxes the binary variables x_{ij} , y_i , and z_j , $i, j = 1, 2, \cdots, n$ to real variables, then we obtain its relaxed version for the IP Formulation, which can be exactly solved in polynomial time by using the well-known interior point method [14]. The optimal objective function value of the relaxed version is a lower bound for that of its IP Formulation.

3. Three Heuristics Algorithms

In this section, we focus on to design some fast heuristic algorithms to solve the approximate solution for problem $1|p-batch, s_j, rej|C_{\max}+\sum_{j\in R} e_j$. For the first two heuristic algorithms (Algorithms 1 and 2), the main idea is first to determine the rejected job subset and then construct batches for the accepted jobs to process them in an arbitrary order on the machine. For the third heuristic algorithm (Algorithm 3), the main idea is first to construct batches for the accepted jobs and then by checking each batch determine the rejected job subset. Our heuristic algorithms are listed as follows.

Algorithm 1

Step 1. Let $R=\{j \mid e_j \le s_j p_j, j=1,2,\cdots,n\} \cap \{j \mid s_j > 1/2, j=1,2,\cdots,n\}$ be the rejected job subset. Accordingly, $A=J-R$ is the accepted job subset.

Step 2. For the accepted jobs in A, first sequence them in non-increasing order of the job processing times, i.e., $p_{\pi_1} \ge p_{\pi_2} \ge \cdots \ge p_{\pi_{|A|}}$, where $|A|$ denotes the number of accepted jobs and $(\pi_1, \pi_2, \cdots, \pi_{|A|})$ is a permutation of the accepted jobs.

Step 3: If $A \neq \varnothing$, open the first batch B_1. Initially, let $B_1 := \varnothing$ and $S(B_1) := 0$. Let $k := 1$.

Step 4: From $i=1$ to $|A|$, do the following:

If $S(B_k)+s_{\pi_i} \le 1$, put job J_{π_i} into batch B_k and update $S(B_k) := S(B_k)+s_{\pi_i}$; otherwise let $k := k+1$, open a new batch B_k, put job J_{π_i} into batch B_k and update $S(B_k) := s_{\pi_i}$.

Step 5: Processing the opened batches $B_1, B_2, \cdots, B_k$ in an arbitrary order on the machine and output the solution $(B_1, B_2, \cdots, B_k \mid R)$.

Theorem 1. ***Algorithm 1*** *runs in* $O(n \log n)$ *time and output a feasible solution for problem* $1|p-batch, s_j, rej|C_{\max}+\sum_{j\in R} e_j$.

Proof. For the running time, **Algorithm 1** mainly costs in Step 2, which is $O(n \log n)$ time. For other steps, it costs at most $O(n)$ time. Thus, **Algorithm 1** runs in $O(n \log n)$ time. From Step 4, the constructed batches are subject to the batch capacity constraint, which implies the outputted solution is feasible. Hence, it is proven. □

Algorithm 2

Step 1. First sort all the jobs in J according to the non-increasing order of the job processing times, i.e., $p_{\pi_1} \ge p_{\pi_2} \ge \cdots \ge p_{\pi_n}$, where $(\pi_1, \pi_2, \cdots, \pi_n)$ is a permutation of $(1, 2, \cdots, n)$.

Step 2. Construct a series of rejected job subsets R_i as follows:

$$R_0 = \varnothing, R_i = \{J_{\pi_1}, J_{\pi_2}, \cdots, J_{\pi_i}\}, i = 1, 2, \cdots, n.$$

Accordingly, define the corresponding accepted job subset $A_i = J - R_i, i = 0, 1, 2, \cdots, n$.

Step 3: From $i=0$ to n do the following:

Step 3.1: For the accepted jobs in A_i, first sequence them in non-increasing order of the job processing times, i.e., $p_{\pi_1^i} \geq p_{\pi_2^i} \geq \cdots \geq p_{\pi_{|A_i|}^i}$, where $|A_i|$ denotes the number of accepted jobs in A_i and $(\pi_1^i, \pi_2^i, \cdots, \pi_{|A_i|}^i)$ is a permutation of the accepted jobs in A_i.

Step 3.2: If $A_i \neq \varnothing$, open the first batch B_1^i. Initially, let $B_1^i := \varnothing$ and $S(B_1^i) := 0$. Let $k_i := 1$.

Step 3.3: From $j = 1$ to $|A_i|$, do the following:

If $S(B_{k_i}^i) + s_{\pi_j^i} \leq 1$, put job $J_{\pi_j^i}$ into batch $B_{k_i}^i$ and update $S(B_{k_i}^i) := S(B_{k_i}^i) + s_{\pi_j^i}$; otherwise let $k_i := k_i + 1$, open a new batch $B_{k_i}^i$, put job $J_{\pi_j^i}$ into batch $B_{k_i}^i$ and update $S(B_{k_i}^i) := s_{\pi_j^i}$.

Step 3.4: Processing the opened batches $B_1^i, B_2^i, \cdots, B_{k_i}^i$ in an arbitrary order on the machine.

Step 3.5: Compute the objective function value $Z_i = C_{\max}(A_i) + \sum_{j \in R_i} e_j$ for the accepted jobs A_i and the rejected jobs R_i, where $C_{\max}(A_i)$ denotes the makespan for the constructed parallel-batch schedule $B_1^i, B_2^i, \cdots, B_{k_i}^i$ of A_i and $\sum_{j \in R_i} e_j$ denotes the total rejection penalty.

Step 4: Choose the solution $(B_1^q, B_2^q, \cdots, B_{k_q}^q \mid R_q)$ with minimum Z_q as the output, i.e., $Z_q = \min_{i=0,1,2,\cdots,n} Z_i$.

Theorem 2. ***Algorithm 2*** *runs in* $O(n^2)$ *time and output a feasible solution for problem* $1 \mid p-batch, s_j, rej \mid C_{\max} + \sum_{j \in R} e_j$.

Proof. For the running time, **Algorithm 2** mainly costs in Step 3, which is $O(n^2)$ time. For the other steps, it costs at most $O(n \log n)$ time. Thus, **Algorithm 2** runs in $O(n^2)$ time. From Step 3.3, the constructed batches are subject to the batch capacity constraint, which implies the outputted solution is feasible. Hence, it is proven. □

Algorithm 3

Step 1. First sort all the jobs in J according to the non-increasing order of the job processing times, i.e., $p_{\pi_1} \geq p_{\pi_2} \geq \cdots \geq p_{\pi_n}$, where $(\pi_1, \pi_2, \cdots, \pi_n)$ is a permutation of $(1, 2, \cdots, n)$.

Step 2: Open the first batch B_1. Initially, let $B_1 := \varnothing$ and $S(B_1) := 0$. Let $k := 1$.

Step 3: From $i = 1$ to n, do the following:

If $S(B_k) + s_{\pi_i} \leq 1$, put job J_{π_i} into batch B_k and update $S(B_k) := S(B_k) + s_{\pi_i}$; otherwise let $k := k + 1$, open a new batch B_k, put job J_{π_i} into batch B_k and update $S(B_k) := s_{\pi_i}$.

Step 4: For all the constructed batches B_q, $q = 1, 2, \cdots, k$, if $\max_{j \in B_q} p_j \geq \sum_{j \in B_q} e_j$, we remove batch B_q and reject all the jobs in B_q; otherwise we reserve the batch B_q.

Step 5: Processing the reserved batches in an arbitrary order on the machine.

Theorem 3. ***Algorithm 3*** *runs in* $O(n \log n)$ *time and output a feasible solution for problem* $1 \mid p-batch, s_j, rej \mid C_{\max} + \sum_{j \in R} e_j$.

Proof. For the running time, **Algorithm 3** mainly costs in Step 1, which is $O(n \log n)$ time. For other steps, it costs at most $O(n)$ time. Thus, **Algorithm 3** runs in $O(n \log n)$ time. From Steps 3 and 5, the reserved batches are subject to the batch capacity constraint, which implies the outputted solution is feasible. Hence, it is proven. □

The above three algorithms are combinatorial algorithms. Their running times are faster than some meta-heuristic approach proposed by Shojafar et al. [15] for cloud job scheduling problem and bio-inspired greedy or clustering algorithms proposed by Alizadeh et al. in [16] for batch scheduling problems and Amato et al. in [17] for multimedia stories' creation problem from online social networks.

4. A numerical Example

In this section, we present a small numerical example to show how Algorithms 1, 2 and 3 work and use the exact solution with solving the IP to evaluate the quality of solutions returned by the three heuristics algorithms.

Numerical example. Consider the following $n = 10$ jobs to be possible processed on a single batch processing machine. The capacity for the machine is 1. The corresponding processing times, sizes and reject penalties for the 10 jobs are shown in Table 1.

Table 1. The processing times, sizes and reject penalties for the given 10 job.

Job	J_1	J_2	J_3	J_4	J_5	J_6	J_7	J_8	J_9	J_{10}
Processing time	5	29	20	15	24	4	13	27	24	28
Size	0.7	0.4	0.2	0.1	0.5	0.8	0.6	0.3	0.9	0.5
Rejection penalty	3	12	17	8	14	3	10	14	18	7

Solution results. The IP solver used was Lingo 11. Algorithms 1, 2 and 3 were implemented in MATLAB. All tests were run on an Intel Core i5-4200 PC with a 2.5 GHz dual core processor, 4 GB of RAM, and the Windows 10 operating system. The collected times are the actual run times for the program, in seconds.

We first solved the IP formulation for the numerical example with 10 jobs by using Lingo 11, which takes a very long time, about 14 h. We obtained the optimal values for Z_j^*, $j = 1, 2, \cdots, 10$, as shown in Table 2.

Table 2. The values for Z_j^*, $j = 1, 2, \cdots, 10$.

Z_j^*	Z_1	Z_2	Z_3	Z_4	Z_5	Z_6	Z_7	Z_8	Z_9	Z_{10}
Value	1	0	0	0	1	1	1	0	1	1

According to the values of Z_j^*, $j = 1, 2, \cdots, 10$, we obtained the rejected job subset

$$R^* = \{J_1, J_5, J_6, J_7, J_9, J_{10}\}$$

and the accepted job subset

$$A^* = \{J_2, J_3, J_4, J_8\}$$

with one batch $B_1^* = \{J_2, J_3, J_4, J_8\}$. The objective function value for the optimal solution is 84.

An illustration chart for the makespan of the optimal solution is shown in Figure 1.

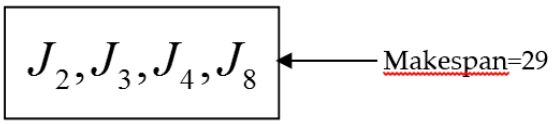

Figure 1. The makespan of the optimal solution.

Running Algorithms 1, 2 and 3 we obtained the solution results in 5 s. All solution results are shown in Table 3, where "OBJ" denotes the objective function value.

Table 3. The solution results for three heuristic algorithms.

Algorithm	The Accepted Jobs and Its Batches	The rejected jobs	OBJ
1	$B_1=(J_2,J_{10}), B_2=(J_8,J_5,J_3), B_3=(J_4,J_7)$	$R=\{J_1,J_6,J_9\}$	95
2	$B_1=(J_3,J_4,J_7), B_2=(J_1), B_3=(J_6)$	$R=\{J_2,J_5,J_8,J_9,J_{10}\}$	94
3	$B_1=(J_5,J_8), B_2=(J_3,J_4,J_7)$	$R=\{J_1,J_2,J_6,J_9,J_{10}\}$	90

The illustration charts for the obtained makespan by Algorithms 1, 2 and 3 are shown in Figures 2–4.

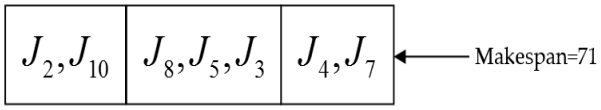

Figure 2. The obtained makespan by Algorithm 1.

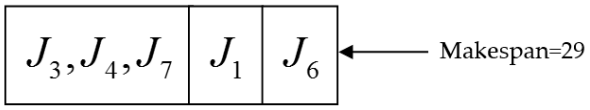

Figure 3. The obtained makespan by Algorithm 2.

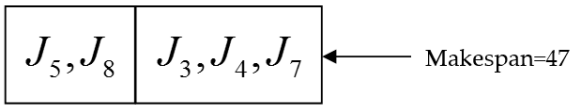

Figure 4. The obtained makespan by Algorithm 3.

Performance Ratio. For the performance ratio of **Algorithm 1**, it is at most $\frac{95}{84} \approx 1.14$. For the performance ratio of **Algorithm 3**, it is $\frac{94}{84} \approx 1.12$. For the performance ratio of **Algorithm 3**, it is $\frac{90}{84} \approx 1.08$. In the given numerical example, **Algorithm 3** output the best solution.

In the actual application, we should run all the three proposed heuristic algorithms and choose the best solution.

Remarks. The above-proposed Algorithms 1, 2 and 3 cannot been applied to other objective functions. If one changes the objective function, then we should redesign the algorithms for the new objective function.

5. Concluding Remarks

In this paper, we investigate the single-machine parallel-batch scheduling problem with nonidentical job sizes and rejection. Our proposed problem generalizes many existing batch

scheduling problems. Our proposed heuristic algorithms are combinatorial and run more quickly than some existing bio-inspired algorithms. It would be interesting to further explore other objective functions and design some fast algorithms with good performance ratio.

Author Contributions: Conceptualization, M.J. and X.L.; methodology, M.J.; software, X.L.; validation, M.J., X.L. and W.L.; writing—original draft preparation, M.J.; writing—review and editing, W.L.; supervision, W.L. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by Zhejiang Provincial Natural Science Foundation of China, grant number LY19A010005; the Humanities and Social Sciences Planning Foundation of the Ministry of Education, grant number 18YJA630077; and Natural Science Foundation of China, grant number 11971252.

Acknowledgments: The authors would like to thank the reviewers for their useful suggestions and comments.

Conflicts of Interest: The authors declare no conflict of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript, or in the decision to publish the results.

References

1. Graham, R.L.; Lawler, E.L.; Lenstra, J.K.; Rinnooy Kan, A.H.G. Optimization and approximation in deterministicsequencing and scheduling: A survey. *Ann. Discret. Math.* **1979**, *5*, 287–326.
2. Ikura, Y.; Gimple, M. Scheduling algorithms for a single batching processing machine. *Oper. Res. Lett.* **1986**, *5*, 61–65.
3. Brucker, P.; Gladky, A.; Han, H.; Kovalyov, M.Y.; Potts, C.N., Tautenhahn, T.; Van de Velde, S.L. Scheduling a batching machine. *J. Sched.* **1998**, *1*, 31–54.
4. Liu, Z.; Yu, W. Scheduling one batch processor subject to job release dates. *Discret. Appl. Math.* **2000**, *105*, 129–136.
5. Cheng, T.C.E.; Liu, Z.; Yu, W. Scheduling jobs with release dates and deadlines on a batch processing machine. *IIE Trans.* **2001**, *33*, 685–690.
6. Liu, Z.; Yuan, J.; Cheng, T.C.E. On scheduling an unbounded batch machine. *Oper. Res. Lett.* **2003**, *31*, 42–48.
7. Zhang, G.; Cai, X.; Lee, C.Y.; Wong, C.K. Minimizing makespan on a single batch processing machine with nonidentical job sizes. *Nav. Res. Logist.* **2001**, *48*, 226–240.
8. Potts, C.N.; Kovalyov, M.Y. Scheduling with batching: A review. *Eur. J. Oper. Res.* **2000**, *120*, 228–249.
9. Lu, L.; Zhang, L.; Yuan, J. The unbounded parallel batch machine scheduling with release dates and rejection to minimize makespan. *Theor. Comput. Sci.* **2008**, *396*, 283–289.
10. Lu, L.; Cheng, T.C.E.; Yuan, J.; Zhang, L. Bounded single-machine parallel-batch scheduling with release dates and rejection. *Comput. Oper. Res.* **2009**, *36*, 2748–2751.
11. Cao, Z.; Yang, X. A PTAS for parallel batch scheduling with rejection and dynamic job arrivals. *Theor. Comput. Sci.* **2009**, *410*, 2732–2745.
12. He, C.; Leung, Y.T.; Lee, K.; Pinedo, M.L. Scheduling a single machine with parallel batching to minimize makespan and total rejection cost. *Discret. Appl. Math.* **2016**, *204*, 150–163.
13. Garey, M.R.; Johnson, D.S. *Computers and Intractability: A Guide to the Theory of NP-Completeness*; W.H. Freeman and Company: San Francisco, CA, USA, 1979.
14. Vanderbei, R.J. *Linear Programming: Foundations and Extensions*, 3rd ed.; Springer: Berlin/Heidelberg, Germany, 2008.
15. Shojafar, M.; Javanmardi, S.; Abolfazli, S.; Nicola, C. FUGE: A joint meta-heuristic approach to cloud job scheduling algorithm using fuzzy theory and a genetic method. *Clust. Comput.* **2015**, *18*, 829–844.
16. Alizadeh, N.; Kashan, A.H. Enhanced grouping league championship and optics inspired optimization algorithms for scheduling a batch processing machine with job conflicts and non-identical job sizes. *Appl. Soft Comput.* **2019**, *83*, 105657.
17. Amato, F.; Castiglione, A.; Mercorio, F.; Mezzanzanica, M.; Moscato, V.; Picariello, A.; Sperlì, G. Multimedia story creation on social networks. *Future Gener. Comput. Syst.* **2018**, *86*, 412–420.