 doi:10.20944/preprints202007.0400.v1

Batch scheduling on a single machine with maintenance interval

Honglin Zhang[1], Bin Qian[2] and Yaohua Wu[1*].

1 School of Control Science and Engineering, Shandong University, Jinan 250061, China, 201920522@mail.sdu.edu.cn

2 Faculty of Information Engineering and Automation, Kunming University of Science and Engineering, Kunming 650500, China, bin.qin@vip.163.com

* Correspondence: MIKE.WU@263.net; Tel: +86 15666940680

Abstract: In the manufacturing industry, orders are typically scheduled and delivered through batches, and the probability of machine failure under high-load operation is high. On this basis, we focus on a single machine batch scheduling problem with a maintenance interval (SMBSP-MI). The studied problem is expressed by three-field representation as $1|B,MI|\sum F_j+\mu m$, and the optimization objective is to minimize total flow time and delivery costs. Firstly, $1|B,MI|\sum F_j+\mu m$ is proved to be NP-hard by Turing reduction. Secondly, shortest processing time (SPT) order is shown the optimal scheduling of SMBSP-MI, and a dynamic programming algorithm based on SPT (DPA-SPT) with the time complexity of $O(n^3T_1)$ is proposed. A small-scale example is designed to verify the feasibility of DPA-SPT. Finally, DPA-SPT is approximated to a fully-polynomial dynamic programming approximation algorithm based on SPT (FDPAA-SPT) by intervals partitioning technique. The proposed FDPAA-SPT runs in $O(\frac{n^5}{\varepsilon^2})$ time with the approximation $(1+\varepsilon)$.

Keywords: batch scheduling; single machine scheduling; maintenance interval; dynamic programming; approximation algorithm

1. Introduction

Single machine scheduling problem is ubiquitous in manufacturing. As of December 2019, the number of small and micro enterprises in China accounted for 82.5% of the national number [1]. Although modern manufacturing has shifted toward intensification and large factories, in terms of quantity, small and micro enterprises still make up a large proportion. Unlike large enterprises that have a large number of machines and assembly lines, in small and micro enterprises, especially micro enterprises, there is only one machine in the workshop production. Due to the large number of machines in large enterprises, accidental machine failure has little impact on the completion of the entire production plan. In 2017, a single machine in the large-scale wafer factory of Magnesium Group failed, and the maintenance of the machine took a lot of time, resulting in a serious lag in production progress. The lag in production progress has led to insufficient capacity of memory products using wafers as raw materials, which has triggered a surge in the prices of related products worldwide. The influence of machine failure on large enterprises is still the same. For small enterprises with only one or a few machines, once one or several machines fail, it will be under a significant negative impact on their production plans. It can be seen that despite the fact that the occurrence of machine failure is a small probability

 doi:10.20944/preprints202007.0400.v1

event, once the machine fails, it will have many adverse effects. By reasonably shutting down and inspecting the machine regularly, the probability of machine failure can be greatly reduced.

In the actual production and trading process, the finished products are usually delivered to customers in batches. Based on this fact, many scholars have conducted research in the field of batch scheduling. Ikura etc. [2] studied single machine batch scheduling problem with the goal of minimizing the maximum completion time. An approximate algorithm was proposed, while the effect of this algorithm is not ideal. Chang etc. [3] further studied the same problem and proposed a Simulated Annealing Algorithm (SAA) with better optimization effect. In Chang's paper, Longest processing time (LPT) order was applied to optimize the initial solution of SAA. Lee etc. [4] proposed a pseudo-polynomial time exact algorithm and a complete polynomial time approximate algorithm to minimize total completion time of SMBSP with dynamic job arrivals. Branch and bound algorithm (BBA) is another algorithm that is commonly used to solve single machine batch scheduling problems. For typical case, see reference XXX. In XXX, Azizoglu etc. [5] proposed a BBA to minimize total weighted completion time of SMBSP. In addition to the batch single-machine scheduling problem, research on the single machine scheduling problem with unavailable intervals (SMSP-UI) is also one of the research hotspots. Several researchers have made progress in SMSP-UI. Sanlaville etc. [6] and Ma etc. [7] independently reviewed the research in this field in recent decades. To optimize the total weighted completion time of SMSP-UI, Ma etc. [8] proposed a dynamic programming algorithm (DPA) and a BBA. both runs in pseudo-polynomial time. Although both DPA and BBA are feasible, they are both pseudo-polynomial time algorithms. In addition, Ma etc. [9] proposed a heuristic algorithm based on longest processing time (LPT) order to minimize max completion time of SMSP-UI. Xie etc. [10] designed a heuristic algorithm which runs in polynomial time to optimize delivery time of SMSP-UI with job rejection. Luo etc. [11] studied a mew branch of SMSP-UI, in which the maintenance time is related to workload. An approximate algorithm running in polynomial time was proposed to optimize total weighted completion time.

It is clear from the above literature review, however, despite the in-depth study in the field of SMBSP and SMSPUI, single machine batch scheduling with maintenance intervals (SMBSP-MI) is still a blind spot for research. Since the outbreak of the COVID-19 virus in January 2020, the production capacity of major mask manufacturers worldwide has been unable to meet demand, and a large number of household mask factories have appeared in this context. Most of these small mask factories work independently or independently from each other. Due to the serious shortage of production capacity, mask machines usually need to work at full load or overload, which greatly increases the probability of machine failure. Therefore, it is necessary to carry out regular shutdown and maintenance of the mask machine. This paper studies SMBSP-MI of a single mask machine scheduling environment. First, the mathematical programming model of SMBSP-MI is established, and then the SMBSP-MI is proved to be NP-hard. On this basis, shortest processing time (SPT) order is proved the optimal scheduling rule of SMBSP-MI, and a DPA based on SPT (DPA-SPT) is running in $O(n^3T_1)$ is proposed. Finally, an approximate method is added to reduce the time complexity of DPA-SPT to polynomial time.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 17 July 2020 doi:10.20944/preprints202007.0400.v1

2. Problem description

Since the outbreak of the COVID-19 virus in January 2020, the production capacity of major mask manufacturers worldwide has been unable to meet demand, and a large number of household mask factories have appeared in this context. Most of these small mask factories work independently or independently from each other. Due to the serious shortage of production capacity, mask machines usually need to work at full load or overload, which greatly increases the probability of machine failure. Therefore, it is necessary to carry out regular shutdown and maintenance of the mask machine. This paper studies the maintenance scheduling problem in the environment of a masking machine. The problem is described as follows: Mask orders for multiple customers are processed on one mask machine. The factory needs to process n quantities of raw materials into masks. Because different customers have different delivery times, the raw materials are processed in batches. Due to a large number of orders, the mask machine needs high-load work. To reduce and avoid the probability of machine failure due to high-load work and ensure smooth production, the factory regularly shuts down the mask machine for maintenance. To ensure that the delivery schedule is not affected while overhauling, the total process time and delivery cost of mask production are used as evaluation indicators

The mathematical description of SMBSP-MI is: n jobs are processed in batches on single machine. The set of jobs is $J = \{j_1, j_2, \dots, j_n\}$. Jobs in J is divided into m batches, i.e., $B = \{B_1, B_2, \dots B_k, \dots, B_m\}$, where B_k represents a batch of jobs, and B is the set of all batches. Once a batch of jobs is processed, it is delivered to the customer. The unit batch delivery cost is μ. A maintenance interval (MI) is set during the machining process, where MI begins at time T_1 and ends at time T_2. No job is allowed to be processed during MI. The goal is to optimize schedule of batches and jobs, so that the sum of total flow time $\sum_{j=1}^{n} F_j$ and delivery cost total μm is minimized. SMBSP-MI is expressed by three-field representation as $1|B, MI|\sum F_j + \mu m$.

The mathematical programming model of SMBSP-MI is expressed as follows.

$$\min Z = \sum_{j=1}^{n} F_j + \mu m \tag{1}$$

s.t.

$$\sum_{j=1}^{n} F_j = \sum_{k=1}^{m} F_{J_k} \tag{2}$$

$$F_{J_k^i} = \begin{cases} F_{J_k^{i-1}} + C_{J_k^i}, & i > 1 \\ F_{J_{k-1}^{last}} + C_{J_k^i}, & i < 1 \end{cases} \tag{3}$$

$$C_{J_k^i} = \begin{cases} C_{J_k^{i-1}} + p_{J_k^i} + \gamma(T_2 - T_1), & i > 1 \\ C_{J_{k-1}^{last}} + p_{J_k^i} + \gamma(T_2 - T_1), & i < 1 \end{cases} \tag{4}$$

$$\gamma = \begin{cases} 1, C_{J_k^{i-1}} + p_{J_k^i} > T_1 \\ 0, C_{J_k^{i-1}} + p_{J_k^i} < T_1 \end{cases}, or \quad \gamma = \begin{cases} 1, C_{J_{k-1}^{last}} + p_{J_k^i} > T_1 \\ 0, C_{J_{k-1}^{last}} + p_{J_k^i} < T_1 \end{cases} \tag{5}$$

$$p_{max} \le T_1 < T_2 \le \tau \sum_{j=1}^{n} p_j \tag{6}$$

Formula 1 is the objective function, in which F_{J_k} is flow time of the kth scheduled batch, and μm is total delivery cost. Formula 2 indicates that the total flow time of all jobs is equal to the sum of the flow time of all batches. This formula makes it convenient to calculate the total flow time in an efficient way. Formula 3 indicates that the flow time of

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 17 July 2020 doi:10.20944/preprints202007.0400.v1

the *i*th job in the *k*th batch is equal to the sum of the flow time of the previous processed job and the completion time of J_k^i. *i*=1 means that J_k^i is the first job to be processed in the *k*th batch, thus the previous job of J_k^i is J_{k-1}^{last} i.e. the last job processed in the (*k*-1)th batch . The previous job of J_k^i is J_k^{i-1} while i >1, respectively. Formula 4 and 5 explains the calculation of J_k^i's completion time, where γ is a 0-1 variable to judge whether J_k^i is allowed to be processed before MI. i.e. J_k^i is allowed to be processed immediately while $C_{J_k^{i-1}} + p_{J_k^i} < T_1$ (or $C_{J_{k-1}^{last}} + p_{J_k^i} < T_1$), otherwise it is to be processed after MI. The strong constraint of formula 6 ensures that the processing time of any job is not greater than T_1, and the end of MI is not earlier than the sum of the processing time of all jobs.

3. The NP-hard attribute of SMBSP-MI

If μ=0 and $|\mathrm{MI}| = 0$, this problem is reduced to $1|B|\sum F_j$, which is proved to be NP-hard by Yang [12]. Obviously $1|B, MI|\sum F_j + \mu m$ is more complicated than $1|B|\sum F_j$, hence is supposed to be an NP-hard problem. In what follows, $1|B, MI|\sum F_j + \mu m$ is proved an NP-hard problem by reducing of equal-size partition problem [13-15].

Equal-size partition problem is described as: Given a set $I = (a_1, a_2, \ldots, a_j, \ldots, a_{2u}), a_j \in N^*$, and $\sum_{j=1}^{2u} a_j = 2M$. Are there two disjoint subsets $I_1$and I_2, where $|I_1| = |I_2|$ and $\sum_{j\in I_1} a_j = \sum_{j\in I_2} a_j = M$?

Theorem 1. SMBSP-MI is an NP-hard problem.

Proof. The NP-hard attribute of SMBSP-MI is proved by the reduction of the Equal-Size Partition Problem. Let $M \geq u + 2$. The Equal-Size Partition Problem is obviously meaningless when $M < u + 2$, so the situation of $M < u + 2$ is not considered. An example of equal-size partition problem is established bellow, variables of example see table 1.

Table 1. variables of equal-size partition example

variables	equations
n	$n = 2u + 1$
p_j	$p_j = \begin{cases} M + a_j, j = 1,2,\ldots,2u \\ (u+2)M, j = 2u+1 \end{cases}$
T_1	$(u + 1)M$
T_2	$M3 + (u + 1)M$
μ	$\mu = (u\text{-}1)\, M^3 + 2(2u2+3u+1)M$
U (Z threshold)	$U = (3u\text{-}1)M^3 + 4(3u^2+5u+2)M$

In what follows, we will prove that the equal-size partition problem has a feasible solution, if and only if the constructed example has a solution which does not exceed U. Let the sets I_1 and I_1 be the solutions to Equal-Size Partition Problem. Let collection B represent the set of jobs corresponding to set I_1, A represent the set of artifacts corresponding to collection I_2.

Consider a situation where jobs in B are divided into one batch and processed before T_1, and jobs in A are divided into one batch and processed after T_2. The objective function

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 17 July 2020 doi:10.20944/preprints202007.0400.v1

is

$Z = \sum_{j=1}^{n} F_j + 2\mu = u\sum_{j\in B}(M + a_j) + (+1)\big(M^3 + (u+1)M + \sum_{j\in A-\{u+1\}}(M + a_j) + (u+2)M\big) = u(u+1)M + (u+1)(M^3 + 2(u+1)M + (u+2)M) + 2\big((u-1)M^3 + 2(2u^2 + 3u + 1)M\big) = (3u-1)M^3 + 4(3u^2 + 5u + 2)M$

In this case the function value is exactly U.

Conversely, assume that there is a feasible schedule with the objective value not exceeding U. Let B and A be the sets of jobs scheduled before $T_1$and after T_2, respectively. Then the following assertions hold:

1) The $(2u+1)$th job is processed after T_2;
2) There are only two batches in this feasible schedule, one batch is processed before T_1 and another batch is processed after T_2, respectively.
3) $|B| = u, |A| = u + 1$;
4) $\sum_{j\in B} p_j = M$.

The proof of these assertions is as follows:

1) The $(2u+1)$th job is processed after T_2 because $p_{2u+1} + M^3 > T$;
2) Three possible cases are considered to prove assertion 2.

Case 1: There is one batch in this schedule, i.e. all jobs are processed after T_2, and thus

$\sum_{j=1}^{n} F_j + \mu m = (2u+1)M(M^3 + 3(u+1)M + (u+2)M) + \big((u-1)M^3 + 2(2u^2 + 3u + 1)M\big) = 3uM^2 + (12u^2 + 20u + 7)M > U$,

which contradicts the constraint that $Z = \sum_{j=1}^{n} F_j + \mu m \le U$.

Case 2: Jobs are divided into more than two batches, i.e. there are at least two jobs processed after T_2.

$\sum_{j=1}^{n} F_j + \mu m > (2u+1)M(M^3 + 3(u+1)M + (u+2)M) + \big((u-1)M^3 + 2(2u^2 + 3u + 1)M\big) = 3uM^2 + (12u^2 + 20u + 7)M > U$,

which contradicts the constraint that $Z = \sum_{j=1}^{n} F_j + \mu m \le U$.

Case 3: Jobs are divided into two batches and both batches are processed after T_2.

$\sum_{j=1}^{n} F_j + 2\mu - U > (2u+1)M^3 + (u+1)M - (u+1)(M^3 + 4(u+1)M) = M(uM^2 - 2u^2 - 5u - 3) > 0$,

which contradicts the constraint that $Z = \sum_{j=1}^{n} F_j + \mu m \le U$. The inequality holds because $M \ge u + 2$.

3) As shown in table 1, T_1=$(u+1)M$ and $p_j > M$, thus $|B| \le u$. Take the case that $|A| \ge u + 1$ under consideration. Let $\sum_{j\in B} p_j = C \le M$ and $|A| = u + 1 + \vartheta$, $\vartheta \ge 1$.

$\sum_{j=1}^{n} F_j + 2\mu - U = (u-\vartheta)\big((u-\vartheta)M\big) + C + (u+1+\vartheta)(M^3 + (u+1)M + (u+\vartheta)M + (2M - C) + (u+2)M) - ((u+1)M^3 + 4(u+1)^2M) = \vartheta M^3 + (2u\vartheta + 2\vartheta^2 + 6\vartheta + 1)M - (2\vartheta + 1)C \ge \vartheta M^3 + (2u\vartheta + 2\vartheta^2 + 6\vartheta + 1)M - (2\vartheta + 1)M > 0$,

which contradicts the constraint that $Z = \sum_{j=1}^{n} F_j + \mu m \le U$. Thus $|A| \le u + 1$. Considering the former assumption that $|A| \ge u + 1$, we get $|A| = u + 1$, and $|B| = u$, respectively.

4) As shown in the proof of assertion 3, $|A| = u + 1$, $|B| = u$. If $C < M$, then

$\sum_{j=1}^{n} F_j + 2\mu - U = u(uM + C) + (u+1)(M^3 + (u+1)M + uM + (2M - C) + (u+2)M) - ((u+1)M^3 + 4(u+1)^2M) = (4u^2 + 8u + 5)M - C - 4(u+1)^2M = M - C > 0$,

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 17 July 2020 doi:10.20944/preprints202007.0400.v1

which contradicts the constraint that $Z = \sum_{j=1}^{n} F_j + \mu m \le U$. Thus $C \ge M$. Considering the former assumption that $C \le M$, we get $C = M$, and $\sum_{j\in B} p_j = C = M$.

The proof of proposed assertions is completed, Equal-Size Partition Problem has a feasible solution. ----- is NP-hard.

3. DPA-SPT

In this section, the Shortest Processing Time (SPT) order[16-19] is proved to be the optimal scheduling rule of ------. And a Dynamic Programming Algorithm (DPA) based on SPT is proposed after that.

Theorem 2. SPT order is the optimal scheduling rule of SMBSP-MI.

Proof. Assume that there is an optimal schedule that does not follow SPT order

$$S = (B_1, B_2, \dots, B_k, \dots, B_l, \dots, B_u, MI, B_{u+1} \dots, B_m),$$

in which $p_{i\in B_1} > p_{i\in B_2} > \dots > p_{i\in B_k} > \dots > p_{i\in B_l} > \dots > p_{i\in B_u} > p_{i\in B_{u+1}} > \dots > p_{i\in B_m}$. Take $j_{B_k}^f$, i.e. the fth job of the kth batch, and $j_{B_l}^g$, i.e. the gth job of the lth batch with the inequality $p_{j_{B_k}^f} > p_{j_{B_l}^g}$. Exchange $j_{B_k}^f$ and $j_{B_l}^g$, i.e. $j_{B_l}^g \in B_k$ and $j_{B_k}^f \in B_l$. A new schedule S' is established. The objective Z with S' is increased by $Z_{S'} - Z_S$, comparing to it is with S.

$$Z_{S'} - Z_S = -(|B_k| + |B_{k+1}| + \dots + |B_{l-1}|)(p_{j_{B_k}^f} - p_{j_{B_l}^g}) > 0,$$

which contradicts the assumption that S is the optimal schedule of ----, and thus $p_{j_{B_k}^f} \le p_{j_{B_l}^g}$. S' is the optimal schedule of ----, that is

$$S' = (B_1, B_2, \dots, B_k, \dots, B_l, \dots, B_u, MI, B_{u+1} \dots, B_m),$$

in which $p_{i\in B_1} \le p_{i\in B_2} \le \dots \le p_{i\in B_k} \le \dots \le p_{i\in B_l} \le \dots \le p_{i\in B_u} \le p_{i\in B_{u+1}} \le \dots \le p_{i\in B_m}$. Note that batches of schedule S' are processed following SPT order, and sequence of jobs in batch does not affect the flow time of this batch.

As is known, DPA is an exact algorithm, which implies that any step of DPA meet the best choice. Since SPT is shown the optimal scheduling order of ---, a DPA based on SPT order (DPA-SPT) is established as follows.

Since there are n jobs in total, DPA-SPT contains n cycles of iterations. Several states are produced while containing the jth into S'. Let R_j=$(s_j^1, \dots, s_j^i, \dots, s_j^x)$ be the set of the states produced in the jth cycle of iteration, in which $s_j^i = (l, c_1, c_2, a, t)$ is one of the feasible states. Variables and parameters in s_j^i are defined as follows: l denotes finish time of the last job processed before T_1; c_1 denotes number of jobs in the last batch processed before T_1; c_2 denotes number of jobs in the first batch processed after T_2 (these jobs may be in the same batch with those processed before T_1, if processing of the last batch is interrupt by MI); a denotes number of jobs processed after T_2; t denotes the sum of current total flow time and delivery cost.

Since DPA select the optimal job in every cycle of iteration, the current optimal schedule of SMBSP-MI before the jth iteration is $S_{j-1} = (1_{best}, 12_{best}, \dots, j-1_{best})$. And by comparing t values of all states produced in the following cycle of iteration, the one with minimal t value is chosen as j_{best}. And j_{best} is at one of the possible position on the machine, i.e. j_{best} is located in the last batch processed before T_1; j_{best} is located in one

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 17 July 2020 doi:10.20944/preprints202007.0400.v1

of other batches processed before T_1; j_{best} is located in the first batch processed after T_2; j_{best} is located in one of other batches processed after T_2, respectively. The optimal value Z^*_{min} will be found after all jobs are sequenced, and the optimal schedule of SMBSP-MI will be decided by backtracking, respectively.

The pseudo code of DPA-SPT is as follows.

Algorithm 1: DPA-SPT

Input: J, m, μ, T_1, T_2
Output: Z^*_{min}
Initialization.
$R_1 = \{(p_1, 1,0,0, p_1 + \mu), (0,0,1,1, T_2 + p_1 + \mu)\}$;
Iteration.
For $j = 2$ to n do
Set $R_j = \emptyset$;
For $(l, c_1, c_2, a, t) \in R_{j-1}$
If $l + p_1 \le T_1$ and $c_1 > 1$ then
$s_j = (l + p_j, c_1 + 1, c_2, a, t + l + (c_1 + 1)p_j)$;
If $l + p_1 \le T_1$ then
$s_j = (l + p_j, 1, c_2, a, t + l + p_j + \mu)$;
If $c_2 > 1$ then
$s_j = (l, c_1, c_2 + 1, a + 1, t + T_2 + \sum_{k=1}^{j} p_k - l + c_2 p_j)$;
Else then
$s_j = (l, c_1, 1, a + 1, t + T_2 + \sum_{k=1}^{j} p_k - l + \mu)$.
End if;
End for;
For $s_j^1 = (l, c_1, c_2, a, t) \in R_j$, $s_j^2 = (l, c_1, c_2, a', t') \in R_j$
If $t < t'$ then
Eliminate s_j^2 from R_j;
Else then
Eliminate s_j^1 from R_j;
End if;
End for;
End for;
Optimization. $Z^*_{min} = min_{(l,c_1,c_2,a,t) \in R_n}\{t\}$.
The optimal schedule is obtained by backtracking.

For any state $s_j = (l, c_1, c_2, a, t)$, the upper bound of l is T_1, the upper bound of c_1 is n, the upper bound of c_2 is (n+1), and the upper bound of j is (n+1). Hence the complexity of DPA-SPT is $O(n^3 T_1)$, and thus DPA-SPT is a pseudo-polynomial time exact algorithm.

To verify the feasibility of DPA-SPT, we present an illustrative 20-scale example. The result shows that DPA-SPT achieves the optimal solution. In this sample,$m = 4$, $\mu = 500$,

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 17 July 2020 doi:10.20944/preprints202007.0400.v1

$T_1 = 90$, $T_2 = 120$, and processing time of jobs are shown in table 2.

Table 2. Processing time of jobs

j	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
p	22	13	23	13	30	30	15	5	8	23	23	22	17	26	24	19	27	23	7	6

Sort jobs according to SPT order. The sorted schedule is shown in table 3.

Table 3. SPT schedule

j	8	20	19	9	2	4	7	13	16	1	12	3	10	11	18	15	14	17	5	6
p	5	6	7	8	13	13	15	17	19	22	22	23	23	23	23	24	26	27	30	30

Run iteration and elimination phase, result is shown in figure 1. The third job in B_2 is interrupted by MI, thus the starting time of $j_{B_2}^3$ is T_2. Detail of $j_{B_2}^3$ see figure 2.

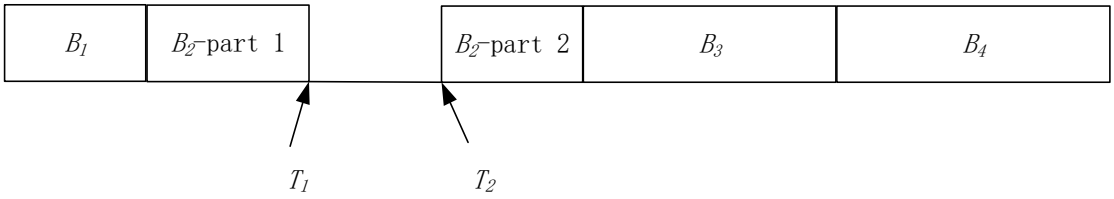

Figure 1. DPA-SPT schedule

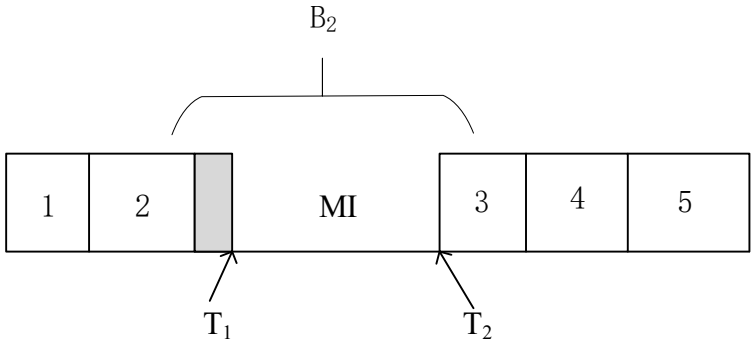

Figure 2. Detail of B_2

When DPA-SPT is finished, the optimal schedule π is obtained, where

$\pi = \{(8, 20, 19, 9, 2), (4, 7, 13, 16, 1), (12, 3, 10, 11, 18), (15, 14, 17, 5, 6)\}$.

Values of F_j see table 4. The optimal value is Z_{min}^* =5784.

Table 4. Values of F_j

F_1	F_2	F_3	F_4	F_5	F_6	F_7	F_8	F_9	F_{10}	F_{11}	F_{12}	F_{13}	F_{14}	F_{15}	F_{16}	F_{17}	F_{18}	F_{19}	F_{20}	$\sum F_j$
5	11	18	26	39	52	77	137	156	178	200	223	246	269	292	316	342	369	399	429	3784

The example shows the feasibility of DPA-SPT.

4. FDPAA-SPT

Although DPA-SPT achieve best solution of small and medium scale problems, it is not feasible for large-scale problems. With the expansion of the problem scale, the

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 17 July 2020 doi:10.20944/preprints202007.0400.v1

complexity of DPA-SPT faces exponential dimension explosion. Taking into account this problem, the use of elimination is the core of approximation to reduce the complexity down to polynomial time. By pruning some bad states and the application of state pruning technology[17], the solution space that the algorithm needs to search is reduced, so as to reduce the complexity. In addition, due to elimination operation, the optimal solution may be missed, hence the improved algorithm is an approximate algorithm. In what follows, a fully-polynomial dynamic programming approximation algorithm based on SPT (FDPAA-SPT) is designed.

Take two large number L and V, where $L \le Z^* \le V$. Take a $\varepsilon > 0$, let $\delta_1 = \frac{L\varepsilon}{2n}$, $\delta_2 = \frac{T\varepsilon}{2n}$. Divide $[0,V]$ and $[0,T_1]$ into $\left\lfloor \frac{V}{\delta_1} \right\rfloor$ and $\left\lfloor \frac{T_1}{\delta_2} \right\rfloor$sub intervals, respectively. In this way, $[0,V] \times [0,T_1]$ is divided into $\left\lfloor \frac{V}{\delta_1} \right\rfloor \times \left\lfloor \frac{T_1}{\delta_2} \right\rfloor$ subintervals.

To get a polynomial-time approximation schedule, the state set R_j obtained by the jth iteration of DPA-SPT is pruned. A new set R_j^* is produced, and R_j^* satisfies the following attribute:

(1) $R_j^* \in R_j$;

(2) There is only one R_j^* in the same subinterval of $\left\lfloor \frac{V}{\delta_1} \right\rfloor \times \left\lfloor \frac{T_1}{\delta_2} \right\rfloor$;

(3) For any eliminated state (l, c_1, c_2, a, t), there is a state $(l', c_1, c_2, a', t') \in R_j^*$ locating in the same subinterval with it.

The pseudo code of FDPAA-SPT is as follows.

Algorithm 2: FDPAA-SPT

Input: J, m, μ, T_1, T_2, L, V, δ_1, δ_2

Output: Z_{min}^*

Initialization.

$$[0,\delta_1),[\delta_1,2\delta_1),\dots,\left[\left(\left\lfloor \frac{V}{\delta_1} \right\rfloor - 1\right)\delta_1, V\right) \leftarrow [0,V];$$

$$[0,\delta_2),[\delta_2,2\delta_2),\dots,\left[\left(\left\lfloor \frac{V}{\delta_2} \right\rfloor - 1\right)\delta_1, T_1\right) \leftarrow [0,T_1];$$

$$R_1 = \{(p_1,1,0,0,p_1+\mu),(0,0,1,1,T_2+p_1+\mu)\};$$

Iteration.

For $j = 2$ to n do

Set $R_j^* = \emptyset$;

For $(l, c_1, c_2, a, t) \in R_{j-1}^*$

If $l + p_1 \le T_1$ and $c_1 > 1$ then

$s_j^* = (l + p_j, c_1 + 1, c_2, a, t + l + (c_1 + 1)p_j)$;

If $l + p_1 \le T_1$ then

$s_j^* = (l + p_j, 1, c_2, a, t + l + p_j + \mu)$;

If $c_2 > 1$ then

$s_j^* = (l, c_1, c_2 + 1, a + 1, t + T_2 + \sum_{k=1}^{j} p_k - l + c_2 p_j)$;

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 17 July 2020 doi:10.20944/preprints202007.0400.v1

Else then

$s_j^* = (l, c_1, 1, a+1, t+T_2+\sum_{k=1}^{j} p_k - l + \mu)$.

End if;

End for;

For any $s_j^* = (l, c_1, c_2, a, t) \in R_j^*$

If $l > V$ then

Eliminate s_j^2 from R_j^*;

Else

Keep s_j^*;

End for;

For $s_j^{*1} = (l, c_1, c_2, a, t) \in R_j^*$, $s_j^{*2} = (l', c_1, c_2, a', t') \in R_j^*$

If $l < l'$ and $t < t'$ then

Eliminate s_j^{*2} from R_j^*;

Else then

Eliminate s_j^{*1} from R_j^*;

End if;

End for;

End for;

Optimization. $Z_{min}^* = min_{(l,c_1,c_2,a,t)\in R_n^*}\{t\}$.

The optimal schedule is obtained by backtracking.

The approximation process of FDPAA-SPT reduces its complexity to 1, and the gap between the optimization solution and the approximate solution of FDPAA-SPT is 2. In what follows, the two conclusions are proved.

Theorem XXX. For any eliminated state $s_j^i = (l, c_1, c_2, a, t) \in R_j$, there exists a state $s_j^* = (l', c_1, c_2, a', t') \in R_j^*$, where $l < l'$ and $t < t + n\delta + na\delta$.

Proof. Mathematical induction is used to prove theorem XXX. When $n = 1$, $s_1 = (l, c_1, c_2, a, t) = s_j^*$, theorem XXX holds. Assume that theorem XXX holds while $n = j - 1$, i.e. for any eliminated state $s_{j-1}^i = (l, c_1, c_2, a, t) \in R_{j-1}$, there exists a state $s_{j-1}^* = (l', c_1, c_2, a', t') \in R_{j-1}^*$, where $l < l'$ and $t < t + (j-1)\delta + (j-1)a\delta$. When it comes to $n = j$, considering the diversity of states in R_j, s_j may be produced from one of the following former cases:

Case 1: $s_{j-1} = (l - p_j, c_1 - 1, c_2, a, t - l - c_1 p_j) \in R_{j-1}$;

Case 2: $s_{j-1} = (l - p_j, x, c_1, c_2, t - l - p_j - \mu) \in R_{j-1}$, $0 \le x \le n - c_1 - 1$;

Case 3: $s_{j-1} = (l, c_1, c_2 - 1, a - 1, t + l - \sum_{k=1}^{j} p_k - (c_1 - 1)p_j) \in R_{j-1}$;

Case 4: $s_{j-1} = (l, c_1, y, a - 1, t - T_2 + l - \sum_{k=1}^{j} p_k - \mu) \in R_{j-1}$;

In what follows, we prove that no matter how s_j is produced, it satisfies the assumption of theorem XXX.

In case 1, there is a state $s_{j-1}^* = (l', c_1 - 1, c_2, a', t') \in R_{j-1}^*$ satisfying inequalities $l' < l - p_j$ and $t' < t - l - c_1 p_j + (j-1)\delta_1 + (j-1)a\delta_2$. This judgment is clear due to the assumption of $n = j - 1$. And in this case, inequality $l' + p_j \le l \le T_1$ holds. By iteration

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 17 July 2020 doi:10.20944/preprints202007.0400.v1

phase of FDPAA-SPT, $s_j = (l' + p_j, c_1, c_2, a, t' + l' + ac_1) \in R_j$ is produced, and inequalities $l' < l - p_j$ and $t' < t - l - c_1 p_j + (j-1)\delta_1 + (j-1)a\delta_2$ holds. According to elimination phase of FDPAA-SPT, there is a state $s_j^* = (l'', 1, c_2, a'', t'') \in R_j^*$ locating in the same subinterval of R_j with s_j. And $l'' \le l' + p_j \le l$ holds. Hence $t'' \le t' + l' + c_1 p_j + \delta_1 \le t - l - c_1 p_j + (j-1)\delta_1 + (j-1)a\delta_2 + l' + c_1 p_j + \delta_1 < t + j\delta_1 + ja\delta_2$.

In case 2, there is state $s_{j-1}^* = (l', c_1 - 1, c_2, a', t') \in R_{j-1}^*$ satisfying inequalities $l' < l - p_j$ and $t' < t - l - p_j - \mu + (j-1)\delta_1 + (j-1)a\delta_2$. By iteration phase of FDPAA-SPT, $s_j = (l' + p_j, 1, c_2, a, t' + l' + \mu) \in R_j$ is produced. According to elimination phase of FDPAA-SPT, there is a state $s_j^* = (l'', 1, c_2, a'', t'') \in R_j^*$ locating in the same subinterval of R_j with s_j. And $l'' \le l' + p_j \le l$ holds. Hence $t'' \le t' + l' + \mu + \delta_1 \le t - l - \mu + (j-1)\delta_1 + (j-1)a\delta_2 + l' + \mu + \delta_1 < t + j\delta_1 + ja\delta_2$.

In case 3, there is state $s_{j-1}^* = (l', x, c_2, a', t') \in R_{j-1}^*$ satisfying inequalities $l' < l$ and $t' < t + l - \sum_{k=1}^{j} p_k - c_2 p_j + (j-1)\delta_1 + (j-1)a\delta_2$. By iteration phase of FDPAA-SPT, $s_j = (l', c_1, c_2, a' + 1, t' - l' + \sum_{k=1}^{j} p_k + (c_2 - c_1)p_j) \in R_j$ is produced. According to elimination phase of FDPAA-SPT, there is a state $s_j^* = (l'', 1, c_2, a'', t'') \in R_j^*$ locating in the same subinterval of R_j with s_j. And $l'' \le l' \le l$ holds. Hence $t'' \le t' - l' + \sum_{k=1}^{j} p_k + (c_2 - 1)p_j + \delta_1 \le t + l - \sum_{k=1}^{j} p_k + (c_2 - 1)p_j + (j-1)\delta_1 + (j-1)a\delta_2 - l' + \sum_{k=1}^{j} p_k + (c_2 - 1)p_j + \delta_1 = t + (l - l') + j\delta_1 + (j-1)(a-1)\delta_2$. Since $l' \ge l - j\delta_2$, $t'' \le t + j\delta_1 + j\delta_2 + (j-1)(a-1)\delta_2 < t + j\delta_1 + ja\delta_2$.

In case 4, there is state $s_{j-1}^* = (l', c_1, y, a', t') \in R_{j-1}^*$ satisfying inequalities $l' < l$ and $t' < t - T_2 + l - \sum_{k=1}^{j} p_k - \mu + (j-1)\delta_1 + (a-1)\delta_2$. By iteration phase of FDPAA-SPT, $s_j = (l', c_1, 1, a' + 1, t' + T_2 - l' + \sum_{k=1}^{j} p_k + \mu) \in R_j$ is produced. According to elimination phase of FDPAA-SPT, there is a state $s_j^* = (l'', c_1, 1, a'', t'') \in R_j^*$ locating in the same subinterval of R_j with s_j. Hence $t'' \le t' + T_2 - l' + + \sum_{k=1}^{j} p_k + \mu \le t - T_2 + l - \sum_{k=1}^{j} p_k - \mu + (j-1)\delta_1 + (j-1)a\delta_2 + T_2 - l' + \sum_{k=1}^{j} p_k + \mu \le t + j\delta_1 + ja\delta_2$.

All the four cases meet theorem XXX, thus theorem XXX holds. Hence there is a state $s_n^* = (l', c_1, c_2, a', t') \in R_n^*$ satisfying inequalities $l' < l$ and $t < t + n\delta_1 + na\delta_2 = t + \varepsilon\frac{L}{2} + \varepsilon\frac{aT_1}{2}$. Note that job n is processed after T_2, thus $t > aT_1$. Hence $t < t + \varepsilon\frac{L}{2} + \varepsilon\frac{aT_1}{2} = (1 + \varepsilon)$t. FDPAA-SPT get a $(1 + \varepsilon)$ approximate solution.

Theorem XXX. The complexity of FDPAA-SPT is $O(\frac{n^5}{\varepsilon^2})$.

Proof. In every iteration phase of FDPAA-SPT, $\left\lceil\frac{V}{\delta_1}\right\rceil \times \left\lceil\frac{T_1}{\delta_2}\right\rceil$ subintervals which occupying $\frac{4n^2}{\varepsilon^2}\frac{V}{L}$ space are produced. Since n is the maximum value of c_1 and c_2, there are $\frac{4n^4}{\varepsilon^2}\frac{V}{L}$ cases in every iteration of FDPAA-SPT. And after n cycle of iteration, $\frac{4n^5}{\varepsilon^2}\frac{V}{L}$ states

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 17 July 2020 doi:10.20944/preprints202007.0400.v1

are produced. Hence the complexity of FDPAA-SPT is $O(\frac{n^5}{\varepsilon^2})$. It is very clear that FDPAA-SPT runs in polynomial time.

5. conclusion

In the manufacturing industry, orders are typically scheduled through batch production and delivery mode, and the probability of machine failure under high-load operation is high. On this basis, we focus on a single machine batch scheduling problem with machine maintenance intervals. To solve this problem, the mathematical programming model is established. SMBSP-MI is proved to be an NP-hard problem, i.e., there is no exact algorithm that can obtain the optimal solution of large-scale problems in polynomial time. Hence, we consider designing an approximation algorithm that can obtain a high-precision solution in polynomial time. Our approach is to design an accurate algorithm based on optimal rules in the first place. Considering the dimensional explosion when solving large-scale problems, some of the poor solutions obtained by DPA-SPT are eliminated to reduce the complexity down to polynomial time. A small-scale example verifies the feasibility of DPA-SPT. As for the proposed approximation algorithm, no verification of the small and medium-scale is conducted, for the verification of the small and medium-scale problem is meaningless. Besides, although the approximation algorithm theoretically has a solution in polynomial time, it still consumes a lot of calculation time, thus no large-scale calculation examples are designed for verification. On the contrary, we proved that the approximate ratio of the algorithm is $(1+\varepsilon)$.

Author contributions: Investigation, Zhang, H. L.; Writing—original draft, Zhang, H. L. and Qian, B.; Writing—review & editing, Zhang, H. L. and Wu, Y. H. All authors have read and agreed to the published version of the manuscript.

Acknowledgments: The authors would like to thank Professor Yin Y. Q. (Doctoral candidate, School of Management and Economics, University of Electronic Science and Technology of China, China) for providing technical support, discussions and review. First author would like to thank professor Qian, B., for providing guidance at the beginning of the study and the initial review of abstract.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Xuan, T., Wei, Z., Weiwen, L., Huihong, L. Low-carbon sustainable development of China's manufacturing industries based on development model change. *Science of the Total Environment*,2020,737.
2. Ikura, Y., Gimple, M. Efficient scheduling algorithms for a single batch processing machine. *Operations Research Letters*,1986, 5(2), 61-65.
3. Chang, P. Y., Damodaran, P., Melouk, S. Minimizing makespan on parallel batch processing machines. *International Journal of Production Research*, 2004, 42(19), 4211-4220.

4. Lee, C. Y. Minimizing makespan on a single batch processing machine with dynamic job arrivals. *International Journal of Production Research*, 1999, 37(1), 219-236.
5. Azizoglu, M., Webster, S. Scheduling a batch processing machine with non-identical job sizes. *International Journal of Production Research*, 2000, 38(10), 2173-2184.
6. Sanlaville, E., Schmidt, G. Machine scheduling with avalibility constraints. *Acta Information*, 1998, (35):795-811.
7. Ma, Y., Chu, C., Zuo, C. A survey of scheduling with deterministic machine avalibility constraints. *Computers& Industrial Engineering*, 2010, (58):199-211.
8. Ma, Y., Chu, C. B., Yang, S. L. Minimizing total weighted completion time in semiresumable case of single-machine scheduling with an availability constraint. *Systems Engineering-Theory & Practice*, 2009, 29(2), 134-143
9. Ma, Y., Yang, S. L., C. Cheng-Bin. Single machine scheduling with an availability constraint and deteriorating jobs. *Journal of Hefei University of Technology (Natural Science)* 25.3(2007):778-781.
10. Xie, X., Xingyao, W., Xiaoli, L., Xiangyu, K. Production and transportation coordinated single machine scheduling problems with unavailability interval and rejection. J*ournal of Shenyang University (Natural Science)*. 2015, 27(3):222-225
11. Luo, W. C., Liu, F. On single-machine scheduling with workload-dependent maintenance duration. *Omega*,2017,68:119-122.
12. Yu, Y., Tang, J. F., Li, J., Sun, W., Wang, J. W. Reducing carbon emission of pickup and delivery using integrated scheduling. *Transportation Research Part D 47*, 2016, 237-250.
13. Corentin, L. H., Anne, L. L. Operations scheduling for waste minimization: a review. *Journal of Cleaner Production*, 2019, 206, 211-226.
14. Yin, Y. Q., Cheng, T. C. E., Hsu, C. J., Wu, C. C. Single-machine batch delivery scheduling with an assignable common due window. *Omega*, 2013, 41, 216-225.
15. Yin, Y. Q., Wang, Y., Cheng, T. C. E., Wang, D. J., Wu, C. C. Two-agent single-machine scheduling to minimize the batch delivery cost. *Computers & Industrial Engineering*, 2016, 92, 16-30.
16. Qi, X., Bard, J. F., Yu, G. Disruption management for machine scheduling: the case of spt schedules. *International Journal of Production Economics*, 103(1), 166-184.
17. Tang, G.C. Theory of modern scheduling. Shanghai science popularization press, 2003.
18. Brucker, P. Scheduling: Theory and Applications. *Operations Research Proceedings 1997*. 1998.
19. Pinedo, M. Scheduling: theory, algorithms, and systems. *Tsinghua university press*, 2005.
20. Yin, Y. Q., Cheng, T. C. E., Wang, D. J. Rescheduling on identical parallel machines with machine disruptions to minimize total completion time. *European Journal of Operational Research*, 2016, 252, 737-749.