Acta Technica *61*, No. 4A/2016, 281–290

Batch delivery scheduling with simple linear deterioration on a single machine[1]

Juan Zou[2,3]

Abstract. Several single machine scheduling problems are solved, in which n independent jobs are available simultaneously and delivered to the customer together upon the completion time of the last job in the batch. The processing time of a job is a simple linear increasing function of its starting time. The objective is to minimize the scheduling cost plus the delivery cost, using several classical scheduling objectives. The several problems are to determine the optimal number of batches, the assignment of jobs to the batches and the job processing sequence so that the sum of cost is minimized. For each problem, we either derive an efficient dynamic programming algorithm that minimizes total cost, or provide some basic properties of the intractable problem.

Key words. Scheduling, batch delivery, linear deterioration, dynamic programming.

1. Introduction

Batch scheduling problems, as combinations of sequencing and partitioning problems, have attracted much attention of researchers in recent years. Motivation for the batch delivery problems comes from the assembly stage in manufacturing of very large-scale integrated circuits. In this stage, chips of various types are attached and placed on a circuit board by a pick and insertion machine. Each circuit board represents a job, upon completion, it is loaded onto a pallet. Intermittently, pallets are moved to the soldering machine and then to the test area. A set of circuit boards loaded on a pallet corresponds to a batch. In practice, however, the number of pallets in use is a cost factor which has to be taken into account. Then we obtain a situation which can be modelled as a batch delivery problem formulated. Batch delivery problems were first introduced by Cheng and Kahlbacher [1], who studied single machine batch delivery scheduling to minimize the sum of the total

[1]This paper is supported by the Science and Technology Projects of Qufu Normal University (xkj201516&sk201507).

[2]School of Mathematics and Statistics, Zhengzhou University, Zhengzhou, 450001, China

[3]School of Mathematical Sciences, Qufu Normal University, Shandong, 273165, China

weighted earliness and delivery costs. They showed that the problem is NP-complete in the ordinary sense, and the equal weight case is solved in polynomial time. Ahmadizar and Farhadi [2] studied the single machine scheduling problem in which jobs are released in different points in time but delivered to customers in batches. A mathematical formulation is developed and dominance properties providing necessary conditions for any solution to be optimal are established. Hall and Potts [3] analyzed the complexity of some single machine scheduling problems with deliveries. Yin, Ye and Zhang [4] considered the problem of scheduling $n$non-resumable and simultaneously available jobs on a single machine, the jobs are delivered in batches to the customers. The objective is to minimize the sum of total flow time and batch delivery cost. They proved the problem is NP-hard and developed two fully polynomial time approximation schemes. Mor and Mosheiov [5] studied a single machine batch scheduling problem with unit time jobs and an optional maintenance activity, the objective function is minimum total flowtime. They proposed a simple rounding procedure that guarantees an integer solution. Selvarajah and Zhang [6] considered supply chain scheduling problem, the objective is to minimize the sum of weighted flow time and the batch delivery costs. They analyzed some polynomially solvable problems and developed a heuristic algorithm for the general problem. Rostami et al. [7] proposed a single-machine scheduling problem that involves minimizing the maximum tardiness plus delivery costs in a batched delivery system with release times. They developed a MIP model and proposed a B&B algorithm with a heuristic upper bound and the LP relaxation technique was developed.

In the classical scheduling theory, the processing times of jobs are considered to be constant and independent of their starting times. However, this assumption is not appropriate for the modelling of many modern industrial processes where the processing time of a job may deteriorate while waiting to be processed. Machine scheduling problems with time dependent processing times have received increasing attention in the recent years. This model reflects some real-life situations in which the expected processing time of a task increases/decreased linearly with starting time. In fact, such situations can be found in maintenance scheduling, steel production, cleaning assignment, fire fighting, resource allocation, where any delay in processing a job may increase the time necessary for its completion. Scheduling deteriorating jobs was first considered by Browne and Yechiali [8] who assumed that the processing times of jobs are non-decreasing, start time dependent linear functions. They provided the optimal solution when the objective is to minimize the expected makespan. Mosheiov [9] considered simple linear deterioration where jobs have a fixed job-dependent growth rate but no basic processing time. He showed that most commonly applied performance criteria, such as the makespan, the total flow time, the total lateness, the sum of weighted completion times, the maximum lateness, the maximum tardiness, and the number of tardy jobs, remain polynomial solvable. Since then, Machine scheduling problems with time dependent processing times have received increasing attention. Cheng, Ding and Lin [10] presented a survey of the results on scheduling problems with time-dependent processing times. Pei et al. [11] considered serial batching scheduling problem with deteriorating jobs in a two-stage supply chain. The objective is to make decisions on job batching and batch sequenc-

ing so as to minimize the makespan. They presented some useful properties and a heuristic for solving it. Yin et al. [12] studied a new deterioration model, the objective is to find the optimal schedule such that makespan or total completion time is minimized. They showed that both problems are solvable in polynomial time.

In this paper, we assume that the actual processing time of each job is a simple linear increasing function of its starting time. The actual processing time of job j in a schedule is given by $p_j = b_j t$, where t represents its starting time. Note that the assumption $t_0 > 0$ is made here to avoid the trivial case of $t_0 = 0$ (when $t_0 = 0$, the completion time of each job will be 0). Each batch is to be assigned a delivery date upon which all jobs within the batch are to be delivered to the customer together. The delivery time of the batch is equal to the completion time of the last job in the batch. There is also a batch delivery cost which is a non-negative function that depends on the number of batches formed to process all the jobs. The problem is to determine simultaneously the optimal number of batches, the assignment of jobs to the batches and the job processing sequence so that the sum of batch delivery cost and the scheduling cost is minimized.

This paper is organized as follows. In section 2, we describe our notation and our scheme for the scheduling and batch delivery problems. In section 3, we provide dynamic programming algorithms for the minimization of total scheduling cost and delivery cost. For the total weighted completion time and delivery cost, we discuss several basic properties. In section 4, we conclude the paper and suggest some topics for future research.

2. Preliminaries

In this section, we describe our notation and assumptions. We begin with some notation. The jobs are processed on a single machine. Let $J = \{1, 2, ..., n\}$ denote the set of jobs to be processed. For job $j \in J$, let p_j denote its processing time, w_j denote its weight and d_j denote its due date. Pre-emption is not allowed. The actual processing time of job j in a schedule is given by $p_j = b_j t$, where t represents its starting time. Jobs form a batch if all of these jobs are dispatched to a customer together in a single delivery. The batch delivery date is equal to the completion time of the last job in the batch. The nonnegative delivery cost $C(y)$ is assumed to be a non-decreasing function of the number of batches y. For any scheduleσ, we define:

- $C_j(\sigma)$ is the time at which job j is delivered to its customer,
- $L_j(\sigma) = C_j(\sigma) - d_j$ is the lateness of job j,
- $U_j(\sigma) = 1$ if job j is late, while $U_j(\sigma) = 0$, if job j is delivered to its customer by its due date,
- $y(\sigma)$ is the number of batch deliveries and
- $C(y)$ is the delivery cost function, which is a non-decreasing function of y.

The objective functions that we consider works with the delivery cost $C(y)$ and a scheduling cost. First we denote

- $\sum C_j$ is the total completion time of the jobs,
- $\sum w_j C_j$ is the total weighted completion time of the jobs,
- $L_{\max} = \max\limits_{j \in J}\{L_j\}$ is the maximum lateness of the jobs and
- $\sum U_j$ is the total number of late jobs.

An example of the classification scheme is problem $1\,|P_j = b_j t, r_j = t_0|\sum C_i + C(y)$, which denotes the minimization of the total delivery cost and job completion times on a single machine under simple linear deterioration.

We make use of the following results which are given without proof.

Lemma 1: In any of the scheduling problems that we consider, there is no idle time between the jobs on the machine.

Lemma 2: For any optimal schedule, the sequence of jobs within each batch is immaterial.

3. Main results

3.1. Sum of completion times

Lemma 3: For problem $1\,|P_j = b_j t,\ r_j = t_0|\sum C_i + C(y)$, the cost is minimized by sequencing the jobs according to non-decreasing order of deterioration rate b_j (SDR).

Proof: Consider an optimal schedule $\sigma*$. Since all jobs within a delivery batch can be sequenced in non-decreasing order of deterioration rate b_j without affecting the cost, we assume that non-decreasing order of b_j for jobs within a batch holds for $\sigma*$. If $\sigma*$ contains jobs that are not sequenced in non-decreasing order of b_j, then we must have a job j that is the last job to be processed in some batch B, and another job i that is the first job to be processed in the next batch B', where $b_j > b_i$.

Consider another schedule σ that is created by interchanging jobs j and i, and forming delivery batches containing jobs $B \cup \{i\}\backslash\{j\}$ and $B' \cup \{j\}\backslash\{i\}$ in σ, where the first of these batches is delivered at the earlier time than batch B in $\sigma*$ and the second of these batches is delivered at the same time as batch B' in $\sigma*$. All the other delivery batches are identical and delivered at the same time in σ as in $\sigma*$. A finite number of repetitions of this argument establishes that there exists an optimal schedule in which the jobs are sequenced in SDR order.

As a result of Lemma 3, we assume that the jobs are indexed in SDR order, so that $b_1 \leq b_2 \leq \ldots \leq b_n$. In the following, we design the dynamic programming algorithm for problem $1\,|P_j = b_j t,\ r_j = t_0|\sum C_i + C(y)$. Then the total cost of the partial schedule is a function value, while the number of deliveries and the index of the current last job which is processed and delivered are state variables. More

precisely, $f(k, y)$ denotes the minimum total cost for processing and delivering jobs $1, 2, \ldots, k$ using y deliveries, with the last delivery at time $t_0(1 + b_1) \ldots (1 + b_k)$, where $0 \leq y \leq k \leq n$. We propose the following dynamic programming algorithm to solve the problem $1\,|P_j = b_j t,\ r_j = t_0|\sum C_i + C(y)$.

Algorithm DP1

- ***Step 1: (Initialization)*** Number jobs in SDR order. Set $f(0, 0) = 0$.
- ***Step 2: (Recursion relation)***

$$f(k, y) =$$

$$= \min_{0 \leq j < k} \{f(j, y - 1) + (k - j)t_0(1 + b_1) \ldots (1 + b_k) + C(y) - C(y - 1)\} .$$

  If $k = n$, go to Step 3; otherwise set $k = k + 1$ and repeat Step 2.

- ***Step 3: (Optimal solution)***

$$\min_{1 \leq y \leq n} \{f(n, y)\}$$

  and use backtracking to find the corresponding optimal schedule.

Theorem 1: Algorithm DP1 gives an optimal schedule for

$$1\,|P_j = b_j t,\ r_j = t_0|\sum C_i + C(y)$$

in $O(n^3)$ time.

Proof: There are $O(n^2)$ states (k, y), and for each state the recurrence relation requires $O(n)$ time. Therefore, the overall time complexity of Algorithm DP1 is $O(n^3)$.

3.2. Sum of weighted completion times

The NP-hardness of the problem $1\,|P_j = b_j t,\ r_j = t_0|\sum w_i C_i + C(y)$ remains open. In the following, we present several properties of the optimal schedule.

Property 1: There exists an optimal sequence for $1\,|P_j = b_j t,\ r_j = t_0|\sum w_i C_i + C(y)$ such that the delivery batch is sequenced in non-decreasing order of $(F_{\mathrm{B}} - 1)/(w_{\mathrm{B}} F_B)$, where $w_{\mathrm{B}} = \sum_{j \in B} w_j$ denotes the total weights of jobs in batch B, $F_{\mathrm{B}} = \prod_{j \in B}(1 + b_j)$.

Proof: Consider an optimal schedule $\sigma*$. If $\sigma*$ contains batches that are not sequenced in non-increasing order of $\frac{F_B - 1}{w_B F_B}$, then we must have a pair of batches B_i, B_j such that batch B_i starting at time S is followed by batch B_j, and $\frac{F_{\mathrm{B_i}} - 1}{w_{B_i} F_{B_i}} > \frac{F_{B_j} - 1}{w_{B_j} F_{B_j}}$. So we have the objective value of batch B_i and B_j in $\sigma*$, $w_{B_i} S \cdot F_{B_i} + w_{B_j} S \cdot F_{B_i} F_{B_j}$.

Consider a new schedule σ which is obtained from $\sigma*$ by interchanging batches B_i and B_j. Under σ, we have $w_{B_j}S \cdot F_{B_j} + w_{B_i}S \cdot F_{B_i}F_{B_j}$. Then we obtain that

$$w_{B_i}S \cdot F_{B_i} + w_{B_j}S \cdot F_{B_i}F_{B_j} - (w_{B_j}S \cdot F_{B_j} + w_{B_i}S \cdot F_{B_i}F_{B_j}) =$$

$$= S\left(w_{B_i}F_{B_i}\left(1 - F_{B_j}\right) - w_{B_j}F_{B_j}\left(1 - F_{B_i}\right)\right) \geq 0\,.$$

Since all other completion times are unchanged, this contradicts the optimality of $\sigma*$. We conclude that the non-decreasing order of $F_B - 1/(w_B F_B)$ is the optimal schedule.

Property 2: For any two jobs $i, j \in J$ to be scheduled consecutively, if $b_i > b_j, w_j > w_i$ and $w_i b_i > w_j b_j$, there is an optimal schedule in which job j immediately precedes job i.

Proof: Consider an optimal schedule $\sigma*$. Obviously, all jobs within a delivery batch can be sequenced in order from Property 2. Then there exists a pair of (i, j) in $\sigma*$ such that job i is followed by j. A job i that is the last job to be processed in some batch B_k, and another job j that is the first job to be processed in the next batch B_{k+1}. Let S denotes the starting time of job i, $\overline{W}$ denotes the total weight in B_k except job i, D_{k+1} denotes the delivery time of B_{k+1}. Then

$$\sum_{l \in B_k} w_l C_l = (w_i + \overline{W})C_{B_k}(\sigma*) = (w_i + \overline{W})S \cdot (1 + b_i)\,,$$

$$w_j C_j(\sigma*) = w_j D_{k+1}\,.$$

Consider a new schedule σ that is created by interchanging jobs i and j, and form delivery batches containing jobs $B_k^{'} = B_k \cup \{j\} \backslash \{i\}$ and $B_{k+1}^{'} = B_{k+1} \cup t\{i\} \backslash \{j\}$. Thus,

$$\sum_{l \in B_k^{'}} w_l C_l = (w_j + \overline{W})C_{B_k^{'}}(\sigma) = (w_j + \overline{W})S \cdot (1 + b_j)\,,$$

$$w_i C_i(\sigma) = w_i D_{k+1}\,.$$

It follows that

$$\sum_{l \in B_k} w_l C_l + w_j C_j(\sigma*) - \sum_{l \in B_k^{'}} w_l C_l - w_i C_i(\sigma) =$$

$$= (w_i + \overline{W})S \cdot (1 + b_i) + w_j D_{k+1} - (w_j + \overline{W})S \cdot (1 + b_j) - w_i D_{k+1} =$$

$$= \overline{W}S(b_i - b_j) + (D_{k+1} - S)\,(w_j - w_i) + S\,(w_i b_i - w_j b_j) \geq 0\,.$$

A finite number of repetitions of this argument establishes that there exists an optimal schedule in which the jobs are sequenced by Property 2.

3.3. Maximum lateness

Lemma 4: For problem $1\,|P_j = b_j t,\ r_j = t_0|\,L_{\max}+C(y)$, there exists an optimal schedule in which the jobs are sequenced according to an earliest due date (EDD) rule.

As a result of Lemma 4, we assume that the jobs are indexed in EDD order, so that $d_1 \le d_2 \le \ldots \le d_n$. In the following, we design the dynamic programming algorithm for problem $1\,|P_j = b_j t,\ r_j = t_0|\,L_{\max} + C(y)$, the maximum lateness of the partial schedule is a function value, while the number of deliveries and the index of the current last job which is processed and delivered are state variables. More precisely, $f(k, y)$ denotes the minimum value of the maximum lateness for processing and delivering jobs $1, 2, \ldots, k$, using y deliveries, with the last delivery at time $t_0(1 + b_1)\ldots(1 + b_k)$, where $0 \le y \le k \le n$. A formal statement of this dynamic programming algorithm is as follows.

Algorithm DP2

- ***Step 1: (Initialization)*** Number jobs in EDD order. Set $f(0, 0) = -\infty$.
- ***Step 2: (Recursion relation)***

$$f(k, y) = \min_{0 \le j < k} \{\max\{t_0(1 + b_1)\ldots(1 + b_k) - d_{j+1}, f(j, y - 1)\}\}\ .$$

  If $k = n$, go to Step 3; otherwise set $k = k + 1$ and repeat Step 2.

- ***Step 3: (Optimal solution)***

$$\min_{1 \le y \le n} \{f(n, y) + C(y)\}$$

  and use backtracking to find the corresponding optimal schedule.

Some remarks should be made about algorithm DP2. In step 2, the recurrence relation selects a batch $\{j + 1, \ldots, k\}$ of jobs to be delivered at time $t_0(1+b_1)\ldots(1+b_k)$. From the EDD indexing of the jobs, job $j + 1$ has the smallest due date and hence also the maximum lateness in this batch, the lateness of job $j+1$ is compared with the maximum lateness of the jobs that have been scheduled earlier.

Theorem 2: Algorithm DP2 gives an optimal schedule for

$$1\,|P_j = b_j t,\ r_j = t_0|\,L_{\max} + C(y)$$

in $O(n^3)$ time.

Proof: There are $O(n^2)$ states (k, y), and for each state the recurrence relation requires $O(n)$ time. Therefore, the overall time complexity of Algorithm DP2 is $O(n^3)$.

3.4. Number of late jobs

Lemma 5: For problem $1\,|P_j = b_j t, r_j = t_0|\sum U_j + C(y)$, there exists an optimal schedule in which the on-time jobs are sequenced according to an earliest due date (EDD) rule.

We assume that a job which would be late is neither product nor delivered. As a result of Lemma 5, we assume that the on-time jobs are indexed in EDD order. For problem $1\,|P_j = b_j t,\ r_j = t_0|\sum U_j + C(y)$, we describe a dynamic programming algorithm that either appends a job to some previous schedule of on-time jobs, or specifies that this job is late. A subsequent decision is made about whether a batch delivery is scheduled on completion of an appended on-time job. The completion time of the current partial schedule is stored as the function value, more precisely, we recursively compute value function $f(k, y, u, j)$ which represents the minimum completion time for processing the on-time jobs $J_1, J_2, ..., J_j, ..., J_k$. All processed jobs $J_1, J_2, ..., J_{j-1}$ are delivered using y deliveries, J_j is the first job in the last batch of the current partial schedule that is not yet scheduled for delivery, and the total number of late jobs is u, where $0 \le y \le k \le n$, $0 \le u \le n$, and $0 \le j \le k$.

Algorithm DP3

- ***Step 1: (Initialization)*** Number jobs in EDD order. Set $f(k, y, u, j) = \infty$, if $j > 0$, $f(k, y, u, j) > d_j$. Set $f(0, 0, 0, 0) = 0$.
- ***Step 2: (Recursion)***

$$f(k,y,u,j) = \min \begin{cases} f(k-1,y,u-1,j), \\ f(k-1,y,u,j) + p_k \\ \quad \text{if} \ \ 0 < j < k \ \ \text{and} \ \ f(k-1,y,u,j) + p_k \le d_j, \\ \min\limits_{j' \in J} \{f(k-1,y-1,u,j') + p_k\} \quad \text{if} \ \ j = k, \end{cases}$$

  where $J = \{j' \,|1 \le j' \le k-1, f(k-1, y-1, u, j') + p_k \le d_k\}$.

  If $k = n$, go to Step 3; otherwise set $k = k + 1$ and repeat Step 2.

- ***Step 3: (Optimal solution)***

$$\min \left\{ u + C(y) \,\middle|\, \min_{1 \le j \le n} \ \{f(n,y,u,j)\}, 0 \le u \le n, 0 \le y \le n \right\}$$

  and use backtracking to find the corresponding optimal schedule.

Some remarks should be made about algorithm DP3. In step 2, the first term of the minimization in the recurrence relation schedules job k to be late; the second term schedules job k to be on time and belonging to the current batch of jobs, provided that job k can be completed no later than time d_j, job j can be dispatched by its due date, no decision has yet been made about when to delivery the batch

containing jobs j and k; the third term schedules a delivery for a batch containing j' as its first job, if $j' > 0$, starts a new batch with job k which is scheduled to be on time.

Theorem 3: Algorithm DP3 gives an optimal schedule for

$$1\,|P_j = b_j t,\ r_j = t_0|\sum U_j + C(y)$$

in $O(n^4)$ time.

Proof: There are $O(n^4)$ states (k, y, u, j). The first and second terms in the recurrence relation require constant time for each state, the third term requires $O(n)$ time for each of the $O(n^3)$ states for $j = k$. Therefore, the overall time complexity of Algorithm DP3 is $O(n^4)$.

4. Conclusion

We have studied a single machine batch delivery problem under simple linear deterioration, and presented several models for scheduling problems which include a delivery cost. The aim is to find simultaneously a number of batches, a partition of the jobs into batches and a job sequence so as to minimize the scheduling cost and the delivery cost. We provided several algorithms for scheduling jobs on machine and forming batches for delivery. Therefore, our work has practical implications for the way in which scheduling and delivery decisions are made. For future research, it would be interesting to focus on the scheduling problems with batch delivery costs on parallel machines.

References

[1] T. C. E. Cheng, H. G. Kahlbacher: *Scheduling with delivery and earliness penalties.* Asia Pacific Journal of Operational Research *10* (1993), No. 2, 145–152.

[2] F. Ahmadizar, S. Farhadi: *Single-machine batch delivery scheduling with job release dates, due windows and earliness, tardiness, holding and delivery costs.* Computers & Operations Research *53* (2015), 194–205.

[3] N. G. Hal, C. N. Potts: *Supply chain scheduling: Batching and delivery.* Operations Research *51* (2003), No. 4, 566–584.

[4] Y. Q. Yin, D. S. Ye, G. C. Zhang: *Single machine batch scheduling to minimize the sum of total flow time and batch delivery cost with an unavailability interval.* Information Sciences *274* (2014), 310–322.

[5] B. Mor, G. Mosheiov: *Batch scheduling with a rate-modifying maintenance activity to minimize total flowtime.* IJ Production Economics *153* (2014), 238–242.

[6] E. Selvarajah, R. Zhang: *Supply chain scheduling at the manufacturer to minimize inventory holding and delivery costs.* IJ Production Economics *147* (2014) 117–124.

[7] M. Rostami, O. Kheirandish, N. Ansari: *Minimizing maximum tardiness and delivery costs with batch delivery and job release times.* Applied Mathematical Modelling *39* (2015), No. 16, 4909–4927.

[8] S. Browne, U. Yechiali: *Scheduling deteriorating jobs on a single processor.* Operations Research *38* (1990), No. 3, 495–498.

[9] G. Mosheiov: *Scheduling jobs under simple linear deterioration.* Computers & Operations Research *21* (1994), No. 6, 653–659.

[10] T. C. E. Cheng, Q. Ding, B. M. T. Lin: *A concise survey of scheduling with time-dependent processing times.* European Journal of Operational Research *152*, (2004), No. 1, 1–13.

[11] J. Pei, P. M. Pardalos, X. B. Liu, W. J. Fan, S. L. Yang: *Serial batching scheduling of deteriorating jobs in a two-stage supply chain to minimize the makespan.* European Journal of Operational Research *244* (2015), No. 1, 13–25.

[12] Y. Q. Yin, W. H. Wu, T. C. E. Cheng, C. C. Wu: *Single-machine scheduling with time-dependent and position-dependent deteriorating jobs.* IJ Computer Integrated Manufacturing *28* (2015), No. 7, 781–790.

Received December 6, 2016