Order Scheduling On a Single Batch Processor

Feng Zhou*
California State University, Stanislaus, Turlock, California, USA

We consider the problem of scheduling customer orders on a single batch processor machine. The batch processor machine can process up to a certain number of jobs at the same time. Different customer orders may be grouped into the same batch for processing and a customer order may also be split into multiple batches for processing. Three scheduling objective measures are studied: total completion time, the number of late orders and the maximum lateness of orders. For each objective, we either show characteristics of optimal schedules or provide a simple polynomial time algorithm to develop an optimal schedule.

* Corresponding Author. E-mail address: fzhou@csustan.edu

I. INTRODUCTION

There are two types of processing technology in manufacturing industry, discrete processing and batch processing. In discrete processing, a machine can only process one job at a time. Jobs are scheduled to be processed on the machine sequentially. In batch processing, a machine is capable of processing up to a certain number of jobs (usually of the same type) simultaneously.

The majority of job/machine scheduling literature focus on discrete processing setting. This mainstream research can be classified according to the configuration of machines and the flow of jobs, i.e. single machine models, parallel machine models, flow shop models and job shop models. For each type of model, literature can be further classified in term of scheduling objective measures, like total completion time, the number of late jobs and the maximum lateness, etc. Readers who are interested in this area can refer to Chen et al. (1998) and Pinedo (2008).

There are a small amount of discrete processing scheduling models that also include batching operations. In those models, jobs of the same family are grouped in the same batches and are processed consecutively. Setups are performed whenever a machine is switched to process a new batch of jobs. Batching jobs of the same family will save setup time/cost and result in improved efficiency for a manufacturer. Santos and Magazine (1985), Dobson et al. (1987), Baker (1988), Coffman et al. (1989), Vickson et al. (1993), Gerodimos et al. (2000), Lin (2002) and Zhou et al. (2014) studied a variety of single machine scheduling problems with batching decisions. This research stream is still considered as discrete processing in scheduling literature because jobs in the same batch are processed one at a time consecutively.

Batch processing in scheduling draws much less attention over the years. One reason may be related to its limited application in manufacturing industry. Hou et al. (2014) and Ahmadi et al. (1992) provided multiple batch processing examples. One of the examples occurs in a semiconductor factory where chips are tested simultaneously in a burn-in oven. Hou et al. (2014) studied a batch/lot scheduling problem on a single machine. In their model, a manufacturer needs to group customer orders into multiple batches and schedule them on a single batch processor to minimize total completion time. They showed a key property regarding optimal schedules. Their model restricts each customer's order size to be less than the capacity of the batch processor. Furthermore, a customer order may be split into two consecutive batches and its completion time is assumed to be the totals of the weighted completion time of the two consecutive batches. Yang et al. (2017) studied a similar problem as Hou et al. (2014). However, they require each customer order to be processed in the same batch, thus eliminating order split. They proposed a binary integer programming method and four simple heuristics to solve the problem. Ahmadi et al. (1992) presented a new class of two-machine flow shop scheduling problems in which one machine is a discrete processor while the other is a batch processor. Two scheduling objective measures are considered in their models, the makespan and total completion time. They analyzed complexities of their models and provided either polynomial time procedures or heuristics to solve the proposed problems.

In this paper, we consider a similar setting as Hou et al. (2014). However, we do not restrict customer order sizes to be less than the capacity of a batch processor. We require each order to be delivered as a whole instead of allowing partial shipments. Therefore the completion time of an order is when the last batch that contains that order completes. Furthermore, we study three important scheduling objective measures instead of one as in Hou et al. (2014): total completion time, the number of late orders and the maximum lateness of orders. All of the above would make our research a nice addition to the existing literature on batch processing scheduling.

II. PROBLEM DESCRIPTION

We study a setting where a manufacturer has received N customer orders, varying in sizes and needs to schedule them on a batch processor machine. The machine is capable of processing a maximum of b units of jobs at the same time, where b is the machine's capacity (or batch size limit). A customer's order size, σ_i, where $i = 1, ..., N$, may exceed the capacity of the machine, $\sigma_i \geq b$. Therefore the order has to be split into multiple batches. The processing time of each batch on the machine, u, is constant, regardless of types of customer orders or sizes. There is no setup time or cost between any of two consecutive batches. To better utilize the machine's capacity, if multiple batches are required, all batches should be full except the very last batch. An order will only be delivered to its customer when all of its units are completed. Therefore the completion time of an order, $C(i)$, $i = 1, ..., N$, is defined as the time when the last batch that contains that order is completed. The manufacturer needs to group all customer orders into multiple batches and schedule them on the machine. Three common scheduling objectives are considered, including the total completion time of all orders, the number of late orders, and the maximum lateness of orders. The following sections will address each objective accordingly.

III. MINIMIZING THE TOTAL COMPLETION TIME OF ALL ORDERS

In a typical manufacturing setting, the completion time of an order or a job is used to measure how long that order or job and all of its associated raw material, parts and components stay in the system. The longer the order remains in the system, the higher inventory cost it will be. In that sense, minimizing the total completion time of all orders is the same as minimizing the total work-in-process (WIP) inventory cost, which has great appeal to manufactures in business practice.

Theorem 1. *There exists an optimal schedule in which orders are sequenced in nondecreasing order of their sizes, σ_i, then arranged to fill up all the batches and processed sequentially.*

Proof. Without loss of generality, assume order i and j are any two orders in a schedule S, where $S = (..., i, ..., j, ...)$ and $\sigma_i \geq \sigma_j$.

Interchange the position of order i and j in the schedule S and it results in a new schedule S'. It is clear that the completion time of order i in the new schedule S', $C_{S'}(i)$, is the same as the completion time of order j in the original schedule S, $C_S(j)$, i.e. $C_{S'}(i) = C_S(j)$. It is also clear that $C_{S'}(j) \leq C_S(i)$ because of $\sigma_j \leq \sigma_i$. Therefore, $C_{S'}(i)+C_{S'}(j) \leq C_S(j)+C_S(i)$. Furthermore, for all of the orders in between order i and j, their total completion time in the new schedule S' is no more than in the original schedule S because $\sigma_i \geq \sigma_j$. For all the remaining orders, their completion time is unaffected by the interchange of order i and j. Therefore the total completion time of all orders in the new schedule S' will be no more than in the original schedule S, i.e., $\sum_{i=1}^{N} C_{S'}(i) \leq \sum_{i=1}^{N} C_S(i)$.

Repeatedly apply the same interchange argument for all the other orders, our proof is completed.

According to Theorem 1, a manufacturer would schedule customer orders based on their order sizes, from the smallest to the largest, and then assign them to fill up each batch in sequence.

IV. MINIMIZING THE NUMBER OF LATE ORDERS

Minimizing the number of late customer orders has great practical implication. It is equivalent to maximize the percentage of on-time order shipments, which is an important performance measure a manager needs to constantly monitor and evaluate in business practice. A manufacturer may not complete all of its orders on time due to capacity constraint or other restrictions, but it needs to find a way to schedule orders to limit the number of late customer orders.

Each customer order is associated with a due date, d_i, where $i = 1,2, ..., N$. If the completion time of an order is later than its due date, $C(i) > d_i$, that order is late. We are to use a forward algorithm by Pinedo (2008) to build a schedule that would result in the minimum number of late orders for the manufacturer. Before the introduction of the algorithm, reorder all customer orders according to their due dates in nondecreasing order, which is often referred to as the earliest due date (EDD) rule, such that $d_1 \leq d_2 \leq ... \leq d_N$. The algorithm will go through N iterations. During each iteration, the algorithm will select the first order from an unscheduled customer order set, J^w, and assign it to either an order set, J^e, in which all of its orders would meet their due dates or the other order set, J^l, in which all of its order would be considered as late orders. The algorithm is presented as follows.

Step 1. Set $J^e = \emptyset$, $J^l = \emptyset$ and $J^w = \{1, 2, ..., N\}$. Set the counter $n = 1$.

Step 2. Add customer order n to set J^e, and delete order n from set J^w. Go to the next step.

Step 3. If

$$\left\lceil \frac{\sum_{i \in j^e} \sigma_i}{b} \right\rceil u \leq d_n$$

where $\lceil x \rceil$ is denoted as the smallest integer that is greater than or equal to x. Go to the next step.

Otherwise, denote order m as the largest order in set J^e, then remove order m from set J^e and reassign it to set J^l.

Step 4. Set $J_n = J^e$. If $n = N$, stop; otherwise, set $n = n + 1$ and go to step 2.

Once the algorithm terminates, it results in three order sets: J^e, a set in which all of its orders would be completed on time; J^l, a set in which all of its orders would be completed late; and J^w, which should become an empty set. The resulting schedule from the algorithm contains two portions: the first portion is from set J^e, in which orders are scheduled in nondecreasing order of their due dates; the second portion of the schedule is from set J^l in which orders can be sequenced arbitrarily.

Lemma 2. *Define an order set J as a feasible set, if all of its orders would meet their due dates when they are sequenced according to EDD rule, like $d_1 \le d_2 \le \ldots \le d_N$. Then, order sets J_n in the forward algorithm are feasible, where n = 1, ..., N.*

Proof. We prove by induction.

It is obvious when $n = 1$, J_1 is feasible if $\sigma_1 \le d_1$. Otherwise, J_1 is an empty set.

Assume it is true for $n = k - 1$, thus order set J_{k-1} is feasible. Add order k to order set J_{k-1} results in a new set J'.

If the completion time of order k in J' satisfies

$$C_{j'}(k) = \left\lceil \frac{\sum_{i \in J'} \sigma_i}{b} \right\rceil u \le d_k$$

then order k will also be completed on time in J'. And therefore the order set $J_k = J'$ is feasible.

If the completion time of order k in J' satisfies

$$C_{J'}(k) > d_k$$

then remove the largest order m from set J', which results in a new set J'', where $\sigma_m = \max_{i \in j'} \{\sigma_i\}$. Since $\sigma_k \le \sigma_m$ and $d_m \le d_k$, the resulting set J'' is also feasible, and $J_k = J''$.

Therefore it is also true for when $n = k$, J_k is feasible.

Our proof by induction is now complete.

Lemma 3. *Define an order set J as l−optimal, if it is a feasible subset of orders 1, 2, ..., l and if it has, among all feasible subsets of orders 1,2, ..., l, the maximum number of orders. Then, for any l > n, there exists an l−optimal order set that consists of a subset of J_n and a subset of orders n + 1, ..., l.*

Proof. We also prove it by using induction.

When $n = 1$, it is clearly true that there exists an l−optimal set that consists of a subset of J_1 and a subset of orders 2, ..., l.

Now assume it is true for $n = k - 1$. Therefore there exits an l−optimal order set, J^*, which consists of a subset of J_{k-1} and a subset of orders k, ..., l.

We are to prove it is also true for $n = k$, where there exits an l−optimal order set, J^{**}, which consists of a subset of J_k and a subset of orders $k + 1$, ..., l. And we can show J^{**} can be constructed according to the following three cases.

Case 1: Order set J_k consists of J_{k-1} plus order k. To create set J^{**}, just use set J^*.

Case 2: Order set J_k consists of J_{k-1} plus order k, and subtract order m which does not belong to set J^*. Then to create set J^{**}, use set J^* again.

Case 3: Order set J_k consists of J_{k-1} plus order k, and subtract order m which does belong to set J^*. Since J_{k-1} plus order k is not feasible, there must exist an order p in a set that consists of J_{k-1} and k, such that p does not belong to J^*. Now to create set J^{**}, use set J^* plus order p and subtract order m. It is clear that J^{**} is a subset of J_k and $k + 1$, ..., l. Since J^{**} has the same number of orders as set J^*, we only need to show J^{**} is also feasible. Since J^{**} differs from J^* in its intersection with {1, ..., k}, it suffices to verify the following two properties: the orders that are in the intersection of J^{**} and {1, ..., k} is also feasible, and the total processing time of orders that are in the intersection of J^{**} and {1, ..., k} is no more than the total processing time of orders that are in the intersection of J^{*} and {1, ..., k}. The first property is true because the intersection of J^{**} and {1, ..., k} is a subset of J_k, which is feasible according to Lemma 2. The second property is also true due to $\sigma_p \leq \sigma_m$.

Therefore it is also true for n = k.
Our induction proof completes.

Theorem 4. *The algorithm yields an optimal schedule that minimizes the number of late orders.*

Proof. We only need to prove the resulting order set J_n in the algorithm is an n–optimal for n = 1, ..., N. By induction, it is clearly true for n = 1.

Assume it is also true for $n = k - 1$, thus J_{k-1} is a $(k - 1)$–optimal. According to Lemma 3, the set that consists of J_{k-1} and k must contain a k–optimal set. If J_k, a feasible set by Lemma 2, consists of the entire set J_{k-1} plus k, it must be k–optimal. If set J_{k-1} plus order k is not feasible, the k–optimal set must be a smaller set within the set of J_{k-1} and k; but it must have at least the same number of orders as set J_{k-1}. Set J_k clearly satisfies this condition and J_k is a k–optimal set. Therefore it is also true for $n = k$.

Our induction proof now completes.

The algorithm examines each order in set J^w and then place them in either set J^e or J^l. If a new added order in set J^e is considered as late order, the largest order in set J^e would be removed and then placed in set J^l. This process would ensure small orders with early due dates be scheduled earlier while large orders with late due date would be scheduled at later time. In such a way, the number of late orders is minimized.

V. MINIMIZING THE MAXIMUM LATENESS OF ORDERS

Another due date related objective is the maximum lateness of customer orders. The maximum lateness measures a manufacturer's worst performance regarding meeting due dates. When a manufacturer is unable to complete all orders by their due dates, it has great incentive of limiting the maximum lateness of its orders. That is because a customer may cancel its order if it is too late and may never order again from that manufacturer. The lateness of a customer order is the difference between its completion time and due date, i.e., $L(i) = \max[C(i) - d_i, 0]$, where i = 1, ..., N.

Theorem 5. *There exists an optimal schedule in which all of the orders are sequenced in nondecreasing order of their due dates, such that* $d_1 \leq d_2 \leq \ldots \leq d_N$.

Proof. Assume there are two consecutive customer orders, i and j, in a schedule S, such that $S = \{..., i, j, ...\}$ and $d_i \geq d_j$. We are to show interchange order i and j in the schedule would either decrease or not affect the maximum lateness.

It is obvious when order i and j are swapped, it would not affect the completion time of all the other orders, and therefore would not affect the lateness of all the other orders. Denote L^* as the maximum lateness of all the other orders, i.e., $L^* = \max\{L_n\}$, where $n \neq i, j$.

Before order i and j are swapped, we have

$$L_{max}=\max\{C(i)-d_i, C(j)-d_j, L^*\}$$

After order i and j are swapped, we have

$$L'_{max}=\max\{C'(i)-d_i, C'(j)-d_j, L^*\}$$

where $C'(i) = C(j)$ and $C'(j) \leq C(j)$.

Since $d_i \geq d_j$, it leads to $C'(i) - d_i \leq C(j) - d_j$. Because $C'(j) \leq C(j)$, it results in $C'(j) - d_j \leq C(j) - d_j$. Therefore we can conclude $L'_{max} \leq L_{max}$.

Repeatedly applying the pairwise interchange argument for all the other orders completes the proof.

Theorem 5 shows to minimize the maximum lateness of orders, a manufacturer needs to prioritize orders with earlier due dates in its schedule, i.e. process customer orders with the earliest due date first, then followed by orders with increasing due dates.

VI. CONCLUSION

This paper studied a customer order scheduling problem for a manufacturer who is equipped with a single batch processor machine. Customer orders vary in sizes and may be larger or smaller than the capacity of the batch processor machine. Different customer orders may be grouped into the same batch for processing and an order may be split into multiple batches. A customer order cannot be delivered until all of its units are completed. Three popular scheduling objective measures are studied in the paper, including the total completion time of all orders, the number of late orders and the maximum lateness of orders. For each objective measure, we either show properties of optimal schedules or develop a simple polynomial time algorithm that would result in an optimal schedule.

There has been limited research in batch processing scheduling compared to discrete processing scheduling. The main reason is its limited industry applications. However, as examples shown in Ahmadi et al. (1992) and Hou et al. (2014), there are noticeable industry applications for batch processing, like semiconductor manufacturing, production of adhesives and paint, bakery and water purifying industry. As new technology/machinery being developed, it is reasonable to believe batch processing will find more usage in practice. This paper extends the existing literature on batch processing and hopefully would draw more research interests in future.

As for future research, it is worthwhile considering the case when a manufacturer does not permit grouping different customer orders into the same batch for processing. This may occur because of the varying processing requirements (like processing time or processing temperature) for different customer orders.

REFERENCES

Ahmadi, J. H., Ahmadi, R. H., Dasu, S. and Tang, C. S., “Batching and scheduling jobs on batch and discrete processors”, *Operations Research* 40 (4), 1992, 750–763.

Baker, K., “Scheduling the production of components at a common facility”, *IIE Transactions* 20 (1), 1988, 32–35.

Chen, B., Potts, C. and Woeginger, G., *A Review of Machine Scheduling: Complexity, Algorithms and Approximability*, Springer, Boston, MA, 1998.

Coffman, E., Nozari, A. and Yannakakis, M., “Optimal scheduling of products with two subassemblies on a single machine”, *Operations Research* 37 (3), 1989, 426–436.

Dobson, G., Karmarkar, U. and Rummel, J., “Batching to minimize flow times on one machine”, *Management Science* 33 (6), 1987, 784–799.

Gerodimos, A., Glass, C. and Potts, C., “Scheduling the production of two-component jobs on a single machine”, *European Journal of Operational Research* 120 (2), 2000, 250–259.

Hou, Y., Yang, D. and Kuo, W., “Lot scheduling on a single machine”, *Information Processing Letters* (114), 2014, 718–722.

Lin, B., "Fabrication scheduling on a single machine with due date constraints", *European Journal of Operational Research* 136 (1), 2002, 95–105.

Pinedo, M. L., *Scheduling: theory, algorithms and systems*, Springer, New York, NY, 2008

Santos, C. and Magazine, M., "Batching in single operation manufacturing systems", *Operations Research Letters* 4 (3), 1985, 99–103.

Vickson, R., Magazine, M. and Santos, C., "Batching and sequencing of components at a single facility", *IIE Transactions* 25 (2), 1993, 65–70.

Yang, D., Hou, Y. and Kuo, W., "A note on a single-machine lot scheduling problem with indivisible orders", *Computers and Operations Research* (79), 2017, 34–38.

Zhou, F., Blocher, J., Heese, H. and Hu, X., "Optimal single machine scheduling of products with components and changeover cost", *European Journal of Operational Research* (233), 2014, 75–83.