Scheduling with Centralized and Decentralized Batching Policies in Concurrent Open Shops

B. M. T. Lin,[1] T. C. E. Cheng[2]

[1] *Department of Information and Finance Management, Institute of Information Management, National Chiao Tung University, Hsinchu, Taiwan*

[2] *Department of Logistics and Maritime Studies, The Hong Kong Polytechnic University, Kowloon, Hong Kong*

Received 1 January 2010; revised 30 September 2010; accepted 4 October 2010
DOI 10.1002/nav.20437
Published online 2 December 2010 in Wiley Online Library (wileyonlinelibrary.com).

Abstract: This article considers batch scheduling with centralized and decentralized decisions. The context of our study is concurrent open shop scheduling where the jobs are to be processed on a set of independent dedicated machines, which process designated operations of the jobs in batches. The batching policy across the machines can be centralized or decentralized. We study such scheduling problems with the objectives of minimizing the maximum lateness, weighted number of tardy jobs, and total weighted completion time, when the job sequence is determined in advance. We present polynomial time dynamic programming algorithms for some cases of these problems and pseudo-polynomial time algorithms for some problems that are NP-hard in the ordinary sense.
 Naval Research Logistics 58: 17–27, 2011

Keywords: batch scheduling; centralized and decentralized decisions; concurrent open shop; dynamic programming

1. INTRODUCTION

This article considers concurrent open shop scheduling with batching policies. The concurrent open shop model is defined as follows: A set of n jobs $\mathcal{J} = \{J_1, J_2, \ldots, J_n\}$ is to be processed. Each job $J_i \in \mathcal{J}$ has m tasks or operations O_{ki}, $1 \leq k \leq m$, to be processed on m independent dedicated machines $M_1, M_2, \ldots, M_m$, subject to the condition that operation $O_{ki}, 1 \leq k \leq m$, can only be processed on machine M_k and requires a processing time p_{ki}. The m operations of a job are independent and their processing can overlap, i.e., they can be simultaneously processed on their dedicated machines. Each machine can process at most one operation at a time and no preemption is allowed. This production model is similar to open shops except that the operations of the same job can be simultaneously processed. Leung et al. [15] provided several practical applications of concurrent open shop scheduling. In this study, we assume that the job operations are processed in batches on each machine. A practical example of the scheduling model is as follows: A manufacturer produces different types of foam products, like pillows and bed mattresses, on dedicated production lines. Due to differences in material properties, each dedicated machine requires a setup time to prepare the molds and mixture of raw materials, and to adjust the machine setting (e.g., temperature) whenever a batch of products of the same type is to be processed. A client who places an order that consists of several different products would like to receive the products in a single shipment. The manufacturer wants to optimize certain performance metrics by batching and sequencing the production of all the products on the dedicated machines.

The batching decision considered in this study is described as follows: Whenever a batch is formed on machine $M_k, 1 \leq k \leq m$, a machine-dependent setup time s_k will be incurred. Batch processing is assumed to be sequential, i.e., the processing length of a batch is equal to the setup time plus the processing times of all the operations contained in the batch. Completion of an operation follows the batch availability mode, which means that the completion time of an operation coincides with the time when the batch it belongs to is finished. If all the machines have the same batching decision, then the model is called centralized, which signifies the fact that a centralized decision is applied to all the machines. On the other hand, if each machine has its own batching decision, then the model is called decentralized.

Correspondence to: B.M.T. Lin (bmtlin@mail.nctu.edu.tw)

Setup $s_1 = s_2 = 5$.

	J_1	J_2	J_3	J_4
p_{1i}	9	9	8	8
p_{2i}	4	5	15	11

Decentralized Batching	*Centralized Batching*
Machine M_1: $\{O_{11}\}, \{O_{12}\}, \{O_{13}, O_{14}\}$	Machine M_1: $\{O_{11}, O_{12}\}, \{O_{13}, O_{14}\}$
Machine M_2: $\{O_{21}, O_{22}\}, \{O_{23}\}, \{O_{24}\}$	Machine M_2: $\{O_{21}, O_{22}\}, \{O_{23}, O_{24}\}$
$C_{11} = 14, C_{12} = 28, C_{13} = C_{14} = 49$	$C_{11} = C_{12} = 23, C_{13} = C_{14} = 44$
$C_{21} = C_{22} = 14, C_{23} = 34, C_{24} = 50$	$C_{21} = C_{22} = 14, C_{23} = C_{24} = 45$
$C_1 = 14, C_2 = 28, C_3 = 49, C_4 = 50$	$C_1 = C_2 = 23, C_3 = C_4 = 45$

Figure 1. Centralized vs. decentralized batching.

Denote the completion time of operation O_{ki} on machine M_k by C_{ki}. Because a job is completed only if all of its operations are finished, the completion time of job J_i is defined as $C_i = \max_{1 \le k \le m}\{C_{ki}\}$. Associated with each job $J_i \in \mathcal{J}$ are a weight w_i and a due date d_i, which specifies the time by which the job is expected to be completed. In a particular schedule, the lateness of job J_i is defined as $L_i = C_i - d_i$ and the maximum lateness of the schedule is $L_{\max} = \max_{J_i \in \mathcal{J}}\{L_i\}$. Subject to the due date constraints, we use the binary variable U_i to denote the status of job J_i as being tardy or not by letting $U_i = 1$ if $C_i > d_i$, and $U_i = 0$ otherwise.

Scheduling of jobs in the proposed production model consists of sequencing, as well as grouping the jobs, so as to optimize a certain objective function. We use the three-field notation $ID|batch^*|\gamma$ to denote the concurrent open shop scheduling problem with batching considerations. The first field ID stands for independent dedicated machines. For our problem, $batch^*$ is present in the second field as a short form of "s-batch$(b/a, s_k)$", i.e., sequential batch processing of the jobs in the batch availability mode with different machine setup times given by s_k $(k = 1, \ldots, m)$. The last field γ is the objective function to be minimized. In this article, we study three objective functions, namely $\gamma = L_{\max}, \gamma = \sum w_i U_i$, and $\gamma = \sum w_i C_i$ (i.e., the total weighted job completion time). The objective $\min \sum w_i C_i$ will be studied based upon the assumption that the operation sequences on all the machines are known a priori. Figure 1 shows an instance of four jobs to be processed on two dedicated machines. With decentralized batching, the operations on different machines are grouped in different ways. While operations O_{11} and O_{12} are in different batches and have different completion times 14 and 28, operations O_{21} and O_{22} are in the same batch and have the same completion time 14. Therefore, the completion times of jobs J_1 and J_2 are 14 and 28, respectively. When centralized batching is adopted, all the machines have the same grouping of jobs and the jobs of a batch have the same completion time.

Note that decentralized batching does not necessarily mean that we make an individually optimal decision on each machine. A combination of individual optimal decisions may turn out to be an unfavorable decision. For example, to minimize the number of tardy jobs, we can easily construct an example in which all the machines have early operations, but no job is early. Suppose there are two jobs J_1 and J_2 with $p_{11} = 4, p_{12} = 7, p_{21} = 7, p_{22} = 4$, and $d_1 = d_2 = 7$. Applying Moore-Hodgson algorithm [21], we determine that operation O_{11} is scheduled early on machine M_1 and operation O_{22} is scheduled early on machine M_2. Based on this result, no job is early because one operation of each job is completed later than the given due date. Here an operation is early if it is completed on its dedicated machine no later than the due date of the job it belongs to and a job is early if all of its operations are completed no later than its due date. Therefore, if we treat each machine as an individual decision maker, then the decision makers need to coordinate their decisions so as to avoid causing trouble to one another. Moreover, similar to the relationship between an integer program and its corresponding linear program, the solution space of centralized batching is a subset of that of decentralized batching. For any specific objective function studied in this article, the optimal solution value for centralized batching cannot be better than that for decentralized batching.

The first study of scheduling in the concurrent open shop appears to be due to Ahmadi and Bagchi [2]. The assembly-type flowshops proposed by Lee et al. [13] and Potts et al. [25] are a generalization by including a second-stage assembly machine. Wagneur and Sriskandarajah [29] studied the same model using the term "concurrent open shops with job overlaps." Wang and Cheng [30] and Cheng et al. [5] provided another interpretation: jobs and operations are replaced by orders and products, respectively. An order specifies a set of product items that can be produced simultaneously and the order is fulfilled when all the product items are produced. We refer the reader to [26] for an up-to-date and comprehensive summary of results on scheduling in the concurrent open shop. To the best of our knowledge, scheduling in the concurrent open shop consisting of batching machines has not been studied in the literature. The critical issue

concerning centralized or decentralized batching decisions across concurrent machines is new and reflects realistic manufacturing environments. If we extend the notion of machines to companies, then the model can deal with coordination of suppliers in supply chain management. Under the circumstances where a company that is relatively dominant in the supply chain places orders (jobs) with manufacturers (machines) for different products, the company can dictate the production rates of the manufacturers. Or the company, intending to optimize its own operations performance, can pay the manufacturers to offset their losses resulting from coordination Li and Xiao [17]. The batching issue considered in Kovalyov et al. [11] is probably the most relevant to our model. They investigated an assembly-type flowshop in which the stage-one machines are assumed to produce components in batches. The objective is to minimize the maximum completion time, i.e., the makespan. They considered two different types of batching, namely sequential and parallel batching. For the parallel-batching mode, the processing length of a batch is defined as the maximum of the processing times of the jobs in it. Cheng et al. [4] and Potts and Kovalyov [24] are two excellent reviews on scheduling with batching and related results.

The rest of this article is organized as follows: We study the problem to minimize the maximum lateness in Section 2. We discuss previous works, along with the development of dynamic programming algorithms for the studied problems. In Section 3, we study the problem to minimize the weighted number of tardy jobs. In Section 4, we discuss the problem to minimize the total weighted completion time. In Section 5, we conclude the paper and suggest potential research topics for further study.

2. PRELIMINARIES AND $ID|batch^*|L_{\max}$

In this section we study the concurrent open shop batch scheduling problem to minimize the maximum lateness, i.e., $ID|batch^*|L_{\max}$. We start by presenting some known related results on concurrent open shop scheduling and batch scheduling.

2.1. Preliminary Properties

We assume that the cost function $f_j(C_j)$ is defined for job J_j, where f_j is nondecreasing in C_j. The following results are fundamental to concurrent open shop scheduling.

LEMMA 1: [29] For concurrent open shop scheduling with the max-objective $\max_{1\le j\le n} f_j(C_j)$, as well as the sum-objective $\sum_{j=1}^{n} f_j(C_j)$, there is an optimal schedule in which all the machines have the same processing sequence.

In view of Lemma 1, we only need to consider permutation schedules, in which all the machines process operations in the same sequence of job indices, for concurrent open shop scheduling with separable functions of the "sum" and "max" types. It is not hard to see that Lemma 1 holds for any regular objective function (to minimize). When batching is incorporated, the order of operations in the same batch is immaterial. Nevertheless, a sequence of operations is still required for each dedicated machine. The next lemma reveals the implication of the property of identical sequence for $ID|batch^*|\gamma$.

LEMMA 2: It suffices to consider only permutation schedules for $ID|batch^*|\gamma$, where γ is an arbitrary regular objective function to minimize.

PROOF: Assume that there is an optimal schedule for the $ID|batch^*|\gamma$ problem that is nonpermutational. There exist two jobs J_i and J_j that do not abide by the requirement of a permutation schedule. Assume that $C_i \le C_j$ and that $C_j = C_{kj}$, i.e., the last operation of job J_j is completed on machine M_k. We then have two operations O_{li} and O_{lj} such that operation O_{lj} precedes O_{li} on machine M_l. Operation O_{lj} is moved to the batch containing operation O_{li} and make operation O_{lj} the immediate successor of operation O_{li}. The processing sequences of jobs J_i and J_j are now the same on machines M_k and M_l, and the completion time of any operation will not be increased by the move. Repeating the above process, if necessary, we eventually come up with a permutation schedule without increasing the optimal objective value. □

With Lemma 2, we can determine a single sequence of job indices that is applied across all the machines for the three objectives $L_{\max}$, $\sum w_i U_i$, and $\sum w_i C_i$ studied in this article.

2.2. $L_{\max}$ with Decentralized Batching

In what follows, we introduce some known results on minimizing the maximum lateness in the concurrent open shop.

LEMMA 3: [15] The earliest due date (EDD) rule solves the $ID||L_{\max}$ problem.

LEMMA 4: [31] There is an optimal schedule for the $1|batch^*|L_{\max}$ problem in which the jobs are sequenced in the EDD order.

Based upon Lemma 4, Webster and Baker [31] developed a backward dynamic programming algorithm for the $1|batch^*|L_{\max}$ problem. Assume that the jobs are indexed in non-decreasing order of their due dates. Let $G(i)$ denote the optimal maximum lateness of the single-machine problem for jobs $J_i, J_{i+1}, \ldots, J_n$. The dynamic program defines a recursive formulation to derive $G(i)$ from $G(j), 1 \le i < j \le n$,

by inserting a batch of jobs $\{J_i, \ldots, J_{j-1}\}$ in front of the schedule associated with $G(j)$. The algorithm is given as follows:

ALGORITHM WB

Initialization:
Set $G(n+1) = -\infty$.

Recursion: Compute the recursion in the order of i: $i = n, n-1, \ldots, 1$.

$$G(i) = \min_{i+1 \le j \le n+1} \left\{ s_1 + \sum_{r=i}^{j-1} p_{1r} + \max\{-d_i, G(j)\} \right\}. \quad (1)$$

The optimal maximum lateness is $G(1)$. Algorithm WB has a time complexity of $O(n^2)$. To study the $ID|batch^*|L_{\max}$ problem, we generalize Lemma 4 to the concurrent open shop model.

LEMMA 5: There is an optimal schedule for the $ID|batch^*|L_{\max}$ problem, regardless of whether centralized or decentralized batching is in effect, in which the jobs are sequenced in the EDD order.

PROOF: Assume that in some optimal schedule there are two consecutive jobs violating the EDD rule. When centralized batching is in effect, the proof is similar to that of Lemma 4 because the operations of a job follow the same way as batching. For decentralized batching, let J_i and J_j be the first pair of such jobs. Consider operations O_{ki} and O_{kj} on machine M_k. If they are in the same batch, we can swap their positions without increasing the maximum lateness. Without loss of generality, let $d_i < d_j$ and the batch containing O_{kj} precede that containing O_{ki}. Since J_i and J_j are the first pair of such jobs, the two batches must be consecutive and O_{kj} is the last operation of its batch and O_{ki} is the first operation of its batch. We move operation O_{kj} to the batch containing operation O_{ki}. The lateness of all the jobs, except J_j, will not increase. The completion time C_j could increase due to the movement of operation O_{kj}. Since $d_i < d_j$, the lateness of job J_j after the movement is less than the lateness of job J_i. Therefore, the maximum lateness will not increase after the movement. Continuing the process of operation movements until the EDD order emerges, we obtain the result. □

With Lemma 5, we can re-index the jobs in the EDD order. While the job ordering issue is resolved, the grouping issue remains to be addressed. We need to determine how the jobs are partitioned into batches on the machines. In the following, we first develop a solution algorithm for decentralized batching.

Algorithm DB-$L_{\max}$ for decentralized batching:

Step 1: Apply Algorithm WB to the n operations on each machine $M_k, 1 \le k \le m$, individually.
Step 2: Return $\max_{1 \le k \le m}\{L_{k,\max}\}$, where $L_{k,\max}$ is the optimal lateness obtained for machine M_k in Step 1.

THEOREM 1: Algorithm DB-$L_{\max}$ optimally solves in $O(mn^2)$ time the $ID|batch^*|L_{\max}$ problem subject to decentralized batching.

PROOF: Let L^*_{ki} denote the lateness of operation O_{ki} in an optimal schedule. Denote the optimal objective value as $L^*_{\max} = \max_{1 \le k \le m, 1 \le i \le n}\{L^*_{ki}\}$. Let L_{ki} be the lateness of operation O_{ki} returned by Algorithm DB-$L_{\max}$. By the optimality of Algorithm WB, we have

$$\max_{1 \le i \le n}\{L_{ki}\} \le \max_{1 \le i \le n}\{L^*_{ki}\}$$

on each machine M_k. The solution value given by Algorithm DB-$L_{\max}$ is

$$L_{\max} = \max_{1 \le i \le n}\{\max_{1 \le k \le m}\{L_{ki}\}\}.$$

We re-write the above expression to obtain the following inequality

$$L_{\max} = \max_{1 \le k \le m}\{\max_{1 \le i \le n}\{L_{ki}\}\} \le \max_{1 \le k \le m}\{\max_{1 \le i \le n}\{L^*_{ki}\}\} = L^*_{\max}.$$

With regard to the computational time, since Algorithm WB is invoked just m times, the total running time of the algorithm is $O(mn^2)$. □

2.3. $L_{\max}$ with Centralized Batching

When centralized batching is in effect, we need to make decisions jointly across all the machines. The first idea to approach this problem is to modify Algorithm WB. Unfortunately, this approach will not work. In the backward formulation of Algorithm WB, no detailed information on each machine is revealed to determine the completion times of the jobs. Therefore, the lateness cannot be determined in the backward recursive process. To acquire and keep track of the necessary information, we circumvent this difficulty by designing a dynamic program using forward chaining. We need to know the exact completion time of the last batch, so we incorporate two extra parameters into our algorithm, namely the number of batches formed and the size of the last batch. The proposed algorithm therefore has a higher complexity than that of Algorithm WB. However, our algorithm solves the case subject to centralized batching.

Recall that the jobs are indexed and sequenced in the EDD order. Define function $G(i, u, v)$, $0 \le u \le i, 0 \le v \le$

$i-u+1$, as the optimal maximum lateness of the first i jobs $J_1, J_2, \ldots, J_i$, given that they are partitioned into exactly u batches and the last batch contains exactly v jobs. When a decision on job J_i is to be made, we either insert J_i in the last batch or form a new batch containing only J_i. In the former case, we need to calculate the lateness of the first job in the last batch. This goal is attainable because we know the completion times of the last batches on all the machines and the due date of the first job in the last batch. The dynamic programming algorithm is given as follows:

Algorithm CB-$L_{\max}$ for centralized batching:

Initialization:

$$G(i,u,v) = \begin{cases} 0, & \text{if } i = u = v = 0; \\ \infty, & \text{otherwise.} \end{cases}$$

Recursion: Compute the recursion for $i = 1, \ldots, n; u = 1, \ldots, i; v = 1, \ldots, i-u+1$:

$$G(i,u,v) = \begin{cases} \min\limits_{1 \le v' \le i-u+1} \left\{ \max\left\{ G(i-1,u-1,v'), C_i^u - d_i \right\} \right\}, & \text{if } v = 1; \\ \max\left\{ G(i-1,u,v-1), C_i^u - d_{i-v+1} \right\}, & \text{if } v > 1, \end{cases}$$

where $C_i^u = \max_{1 \le k \le m}\{us_k + \sum_{r=1}^{i} p_{kr}\}$.

The optimal maximum lateness subject to centralized batching is

$$\min\{G(n,u,v) : 1 \le u \le n, 1 \le v \le n-u+1\}.$$

We justify the algorithm as follows: When the last batch consists of a single job J_i, the recursion will proceed to $G(i-1,u-1,v')$ for all the feasible v' and the lateness of job J_i can be determined by its due date d_i and completion time C_i^u. On the other hand, $v > 1$ implies that including job J_i in the last batch will increase the lateness of any job contained in the last batch. Specifically, the leading job J_{i-v+1} in the last batch has the largest lateness within this batch. Therefore, we take the maximum between $G(i-1,u,v-1)$ and L_{i-v+1}, which is equal to $C_i^u - d_{i-v+1}$. The minimum value is selected from among all the feasible $g(n,u,v)$ values.

To determine the time complexity of the above algorithm, we first note that in the formulation, function $G(\cdot,\cdot,\cdot)$ consists of $O(n^3)$ entries. Once the input instance is given, the partial sums $\sum_{r=1}^{i} p_{kr}$ for all i and a fixed k can be incrementally computed in $O(n)$ time. Therefore, for all i and k, we can derive all the partial sums $\sum_{r=1}^{i} p_{kr}$ in $O(mn)$ time. Knowing $\sum_{r=1}^{i} p_{kr}$, we can further compute C_i^u in $O(m)$ time for a fixed i and a fixed u. In other words, an $O(mn + mn^2) = O(mn^2)$ preprocessing procedure can calculate C_i^u for all i and u. With all C_i^u known in advance, the time required to determine the value of each entry $G(i,u,v)$ is $O(n)$ by examining different v' values when $v = 1$ and $O(1)$ when $v > 1$. There are $O(n^2)$ entries for $v = 1$ and $O(n^3)$ entries for $v > 1$. Therefore the overall running time of the algorithm, including the preprocessing step, is $O(\max\{n^3, mn^2\})$. The result is summarized in the following theorem.

THEOREM 2: The $ID|batch^*|L_{\max}$ problem subject to centralized batching can be solved in $O(n^2 \max\{m,n\})$ time.

3. WEIGHTED NUMBER OF TARDY JOBS

In this section, we address another due-date related objective function—the weighted number of tardy jobs. Hochbaum and Landy [7] proposed a pseudo-polynomial time dynamic programming algorithm for the $1|batch^*|\sum w_i U_i$ problem. The running time is $O(n^2 W)$, where $W = \sum_{j=1}^{n} w_j$. The algorithm solves the case where the jobs have equal weights in $O(n^3)$ time. Brucker and Kovalyov [3] developed a dynamic programming algorithm for the general case. By scaling the weights used in the dynamic programming algorithm, they proposed a fully polynomial time approximation scheme, each $(1+\epsilon)$-approximation algorithm of which requires $O(n^3\epsilon + n^3 logn)$ time. The algorithm solves the unit-weighted case in $O(n^3)$ time. In the context of the concurrent open shop, Wagneur and Sriskandarajah [29] showed that $ID||\sum w_i U_i$ is NP-hard in the ordinary sense even if $m = 2, d_i = d$, and $w_i = 1$. Ng et al. [22] proved the strong NP-hardness of the $PD|p_{ki} \in \{0,1\}, d_i = d|\sum U_i$ problem.

From the above review, we see that scheduling with the objective function $L_{\max}$ and scheduling with the objective function $\sum w_i U_i$ exhibit different computational complexity, although the optimal solutions for both problems are based on the EDD rule (as will be shown in this section for the second function). The $\sum U_i$ problem remains intractable because of the difficulty from not only sequencing but also selecting jobs. A simple proof of the strong NP-hardness of the $ID|p_{ki} \in \{0,1\}, d_i = 1|\sum U_i$ problem by a reduction from three-Dimensional Matching [19] reflects the intractability of the job selection issue. As a generalization of $ID||\sum U_i$, the $ID|batch^*|\sum w_i U_i$ problem of interest in this paper is strongly NP-hard, too. In the following we develop solution algorithms, which have exponential running times, for scheduling with different batching policies. The following properties inspire the development of our algorithms.

LEMMA 6: [5] There exists an optimal solution for the $ID||\sum w_i U_i$ problem in which the early jobs are sequenced in the EDD order and the late jobs are sequenced in an arbitrary order.

LEMMA 7: Regardless of whether centralized or decentralized batching is in effect, there is an optimal solution for the $ID|batch^*|F(U_1,\ldots,U_n)$ problem in which the early jobs are sequenced in the EDD order, where $F(U_1,\ldots,U_n)$ is any regular function.

PROOF: Assume in some optimal schedule there are two early jobs J_i and J_j such that $d_i < d_j$ and J_j precedes J_i on all the machines. We move operation O_{kj}, for each $k, 1 \le k \le m$, to make it the immediate successor of O_{ki} in the same batch. The completion times of all the operations, except operations $\{O_{kj}\}$, do not increase by the move. Furthermore, since New $C_j =$ New $C_i = C_i \le d_i < d_j$, the equality $U_j = 0$ remains true. In other words, no early job will become late after the move. Repeating the same process, we eventually obtain a schedule in which all the early jobs are arranged by the EDD rule, and no early job in the original schedule will be tardy in the new schedule. We thus complete the proof. □

Common examples of the objective function $F(U_1,\ldots,U_n)$ specified in Lemma 7 include $\sum_i w_iU_i$, $\max_i w_iU_i$, $\max\{\sum_{i=1}^{n/2} w_iU_i, \sum_{i=n/2+1}^{n} w_iU_i\}$, etc.

3.1. $\sum w_iU_i$ with Centralized Batching

We now tackle the $ID|batch^*|\sum w_iU_i$ problem subject to centralized batching. For convenience of description, we seek to maximize the weighted number of early jobs instead. Again recall that the jobs are first sequenced in non-decreasing order of their due dates. The recursive function $G(i,t_1,\ldots,t_m,j)$ for $\max_{1\le k\le m}\{t_k\} \le d_j$ is defined as the maximum weighted number of early jobs out of the first i jobs $\{J_1,J_2,\ldots,J_i\}$, subject to the conditions that the completion time of the last early job on machine $M_k, 1 \le k \le m$, is exactly t_k and that the leading job in the last batch on all the machines is $J_j, 0 \le j \le i$. The recursion proceeds by considering job J_i. If $\max_{1\le k\le m}\{t_k\} \le d_i$ and J_i solely constitutes the last batch, then in the recursion we consider the first $i-1$ jobs with different leading jobs in the last batch. On the other hand, if job J_i does not form the last batch alone, then it can be in the last batch or discarded as tardy. The dynamic programming algorithm is given as follows:

Algorithm CB-Sum-of-wiUi for centralized batching:

Initialization:

$$G(i,t_1,\ldots,t_m,j) = \begin{cases} 0, & \text{if } i = t_1 = \cdots = t_m = j = 0;\\ -\infty, & \text{otherwise.}\end{cases}$$

Recursion: Compute the recursion for $i = 1,\ldots,n; j = 1,\ldots,i; t_k = 1,\ldots,d_j (1 \le k \le m)$:

$$(i,t_1,\ldots,t_m,j) = \begin{cases} \max\{G(i-1,t_1-p_{1i},\ldots,t_m-p_{mi},j)+w_i, \\ \quad G(i-1,t_1,\ldots,t_m,j)\}, & \text{if } j<i;\\ \max\limits_{0\le j'<i} G(i-1,t_1-s_1-p_{1i},\ldots, \\ \quad t_m-s_m-p_{mi},j')+w_i, & \text{if } j=i.\end{cases}$$

The maximum weighted number of early jobs is

$$\max\{G(n,t_1,\ldots,t_m,j): 0 \le t_k \le d_j, 1 \le k \le m; 1 \le j \le n\}.$$

We justify the formulation as follows: In the recursion, the condition "$j < i$" dictates that job J_i can be early if it is included in the last early batch but it is not the leading job in the last batch. Subject to this condition, we can either select job J_i for processing or discard it. The second condition is given for the scenario that job J_i is early and solely forms the last batch. For further recursion, we need to consider all the possible jobs $J_{j'}$ acting as the leading job in the last batch after removing job J_i.

With regard to the required running time, we note that each state needs $O(m)$ time to calculate $t_k - p_{ki}$ or $t_k - s - p_{ki}, 1 \le k \le m$, for the recursions. For the case $j < i$, there are $O(n^2 d_{\max}^m)$ states, each of which can be computed in $O(1)$ time. On the other hand, for the case $j = i$, there are $O(nd_{\max}^m)$ states, each of which requires $O(n)$ time to consider different leading jobs $J_{j'}$. The overall running time is thus $O(mn^2 d_{\max}^m)$, which becomes pseudo-polynomial when the number of machines m is fixed or is not part of the input. We thus have the following theorem.

THEOREM 3: The $ID|batch^*|\sum w_iU_i$ problem subject to centralized batching can be solved in pseudo-polynomial time if the number of dedicated machines is constant. □

As discussed before, Brucker and Kovalyov [3] used the rounding technique to design an $O(n^3/\varepsilon + n^3 \log\log n)$ algorithm with a performance ratio of $1+\varepsilon$ for $1|batch^*|\sum w_iU_i$, which is ordinary NP-hard. Although Algorithm CB-Sum-of-wiUi solves problem $IDm|batch^*|\sum w_iU_i$ in pseudo-polynomial time for a fixed m, it cannot be improved to be a polynomial time algorithm, since Lin and Kononov [20] proved that unless $NP = P$, there exists no FPTAS for $ID2|d_i = d|\sum U_i$. Therefore, the above algorithm gives a tight result on the computational complexity of $IDm|batch|\sum w_iU_i$ subject to centralized batching and a fixed number of machines. It is also tight in another sense: when the number of machines is part of the input, there is no pseudo-polynomial time solution, due to Theorem 3.

3.2. $\sum w_i U_i$ with Decentralized Batching

In this subsection, we modify Algorithm CB-Sum-of-wiUi to cope with the decentralized case. Here we need such detailed information as the completion time of each machine and the leading job in the last batch on each machine. Define $G(i, t_1, \ldots, t_m, j_1, \ldots, j_m)$ for $t_k \le d_{j_k}, 1 \le k \le m$, as the optimal weighted number of early jobs for the first i jobs $\{J_1, \ldots, J_i\}$, subject to the conditions that the completion time of the last early job on machine M_k is exactly t_k and that the leading job in the last batch on machine M_k is J_{j_k}. The most crucial part in the design of an exact algorithm is to handle the situation in which job J_i must be included in the solution when J_i is considered for recursion and it solely forms the last batch on some machine M_k, i.e., $i = j_k$.

Algorithm DB-Sum-of-wiUi for decentralized batching:

Initialization:

$$G(i, t_1, \ldots, t_m, j_1, \ldots, j_m) = \begin{cases} 0, & \text{if } i = t_1 = \cdots = t_m = j_1 = \cdots = j_m = 0; \\ -\infty, & \text{otherwise.} \end{cases}$$

Recursion: Compute the recursion for $i = 1, \ldots, n; j_k = 1, \ldots, i; t_k = 1, \ldots, d_{j_k} (1 \le k \le m)$:

$$G(i, t_1, \ldots, t_m, j_1, \ldots, j_m) = \begin{cases} \max\{G(i-1, t_1 - p_{1i}, \ldots, t_m - p_{mi}, j_1, \ldots, j_m) + w_i, \\ \quad G(i-1, t_1, \ldots, t_m, j_1, \ldots, j_m)\}, \text{ if } j_k < i \text{ for all } k; \\ \max_{\mathcal{D}} G(i-1, t'_1, \ldots, t'_m, j'_1, \ldots, j'_m) + w_i, \\ \quad \text{if } i = j_k \text{ for some } k; \end{cases}$$

where $\mathcal{D}$ is the set of feasible $2m$-tuple vectors $(t'_1, \ldots, t'_m, j'_1, \ldots, j'_m)$ defined in the following way:

If $j_k < i$, then $t'_k = t_k - p_{ki}, j'_k = j_k$.
If $j_k = i$, then $t'_k = t_k - s_k - p_{ki}, j'_k \in \{0, 1, \ldots, i-1\}$.

The maximum weighted number of early jobs subject to decentralized batching is

$$\max\{G(n, t_1, \ldots, t_m, j_1, \ldots, j_m) : 0 \le t_k \le d_{j_k}, 1 \le j_k \le n, 1 \le k \le m\}.$$

In the first case of the recursion formula, job J_i does not form the last batch alone on any machine so we either accept job J_i as early or discard it. The size of the space with such states is $O(n^{m+1} d_{\max}^m)$ and the value of each state can be computed in $O(m)$ time, which is required for computing the values $t_1 - p_{1i}, \ldots, t_m - p_{mi}$. In the second case, job J_i solely forms the last batch on at least one machine and it is anyway accepted as an early job. Set $\mathcal{D}$ contains all the admissible combinations of $(t'_1, \ldots, t'_m, j'_1, \ldots, j'_m)$ for further recursions. There are $O(\binom{m}{l} n^{m-l+1} d_{\max}^m)$ states where exactly l j_k's are equal to i and the cardinality of set $\mathcal{D}$ is $O(n^l)$. Therefore, the total (over all the states) time complexity is $O(m 2^m n^{m+1} d_{\max}^m)$, which is exponential and becomes pseudo-polynomial when the number of machines m is constant. We conclude the result in the following theorem.

THEOREM 4: The $ID|batch^*|\sum w_i U_i$ problem subject to decentralized batching can be solved in pseudo-polynomial time if the number of dedicated machines is constant. □

4. TOTAL WEIGHTED COMPLETION TIME

In the context of batch scheduling, Albers and Brucker [1] proved that the single-machine problem to minimize the total weighted completion time, i.e., $1|batch^*|\sum w_i C_i$, is NP-hard in the ordinary sense. If the job sequence is fixed, an $O(n)$ dynamic program can be designed to optimally solve the problem. The case with equal processing times can be solved by sequencing the jobs in non-increasing order of their weights. Coffman et al. [6] investigated the unit-weighted version $1|batch^*|\sum C_i$ and obtained the following result.

LEMMA 8: [6] There exists an optimal schedule for the $1|batch^*|\sum C_i$ problem in which the jobs are sequenced in the shortest processing time (SPT) order.

In view of this property, for the unweighted single-machine case, we can re-index the jobs in the SPT order. Based upon this property, Coffman et al. [6] proposed a backward dynamic programming algorithm. They defined $G(i)$ as the optimal total completion time of the jobs $J_i, J_{i+1}, \ldots, J_n$ and sought the value of $G(1)$. The algorithm is outlined in the following:

Algorithm CYMS

Initialization:

$$\text{Set } G(n+1) = 0.$$

Recursion: Compute the recursion for $i = n, n-1, \ldots, 1$:

$$G(i) = \min_{i+1 \le j \le n+1} \left\{ G(j) + (n - i + 1) \left(s_1 + \sum_{r=i}^{j-1} p_{1r} \right) \right\}.$$

The optimal total completion time is given by $G(1)$. The algorithm is justified by considering the increase in the total completion time caused by a prefix batch $\{J_i, \ldots, J_{j-1}\}$ to all of the jobs in the schedule. The overall running time of the algorithm is $O(n^2)$. Coffman et al. [6] provided an efficient $O(n \log n)$ implementation by using a queue to store the

candidate jobs that can start the next batch. As reviewed in Potts and Kovalyov [24], the geometric approach developed by van Hoesel et al. [8] can attain the same reduction in time complexity.

With regard to the minimization of the total completion time in concurrent open shop scheduling, Wagneur and Sriskandarajah [29] first showed that the $ID||\sum C_i$ problem is strongly NP-hard. Due to the intractability of the problem, Lann et al. [12] proposed a simple heuristic and proved its asymptotic optimality in a probabilistic sense. Sung and Yoon [28] studied the weighted case of the two-machine problem, i.e., $ID2||\sum w_iC_i$, and showed that the worst case performance ratio of the weighted shortest processing time (WSPT) rule is 2. Nevertheless, Leung et al. [16] gave a counter-example to the proof of Wagneur and Sriskandarajah [29]. Leung et al. [14] further showed that the problem is strongly NP-hard for $m \geq 3$. Roemer [26] further sharpened the complexity boundary by showing that the problem remains strongly NP-hard even if there are only two machines. Therefore, the $ID|batch^*|\sum C_i$ problem is hard to solve, too.

In this article, we assume that a job sequence is given and the decision is how to group the jobs into batches. A fixed sequence of the jobs can be admitted by, e.g., an agreeable condition that for any two jobs $J_i, J_j \in \mathcal{J}$, $p_{1i} \leq p_{1j} \Leftrightarrow p_{2i} \leq p_{2j} \Leftrightarrow \cdots \Leftrightarrow p_{mi} \leq p_{mj}$. In most scheduling problems, schedules are easily implied from sequences for most scheduling problems. However, for some problems it is nontrivial to determine the optimal solution to the instance associated with a given job sequence. The assumption of a fixed job sequence has been considered in several scheduling problems. Due to technical constraints on specific machines, Shafransky and Strusevich [27] investigated open shop scheduling problems subject to the condition that the job sequence on each machine is fixed. Studies on scheduling problems with fixed job sequences also inspire different theoretical interests. Lin and Cheng [18] proposed an algorithm to determine the optimal makespan of a sequence of jobs in a two-machine flowshop with conditional stage-two processing times. Solving the special case with a fixed job sequence facilitates the development of lower bounds. Ng and Kovalyov [23] proposed a dynamic programming algorithm for batch scheduling in a flowshop with a given sequence of jobs. Hwang et al. [9] studied three flowshop scheduling problems with decisions of batching and idle time insertion subject to a given job sequence. The development of dynamic programs demands the exploration of several theoretical properties. Based upon the concept of optimal blocks, they developed polynomial-time dynamic programming algorithms. Hwang and Lin [10] introduced several intriguing properties of coupled-task scheduling subject to a fixed job sequence. In this paper we denote the problem to minimize the total weighted completion time as $ID|batch^*, fixed\_seq|\sum w_iC_i$, where "fixed_seq" in the second field indicates that a job sequence is given.

4.1. $\sum w_iC_i$ with a Fixed Sequence and Centralized Batching

Our discussion begins with centralized batching. An attempt to deploy Algorithm CYMS to solve the problem may fail because the increase in the total completion time caused by the insertion of a prefix batch cannot be determined. Consider a partial schedule of the jobs $\{J_j, J_{j+1}, \ldots, J_n\}$. Jobs J_i, J_{i+1}, J_{j-1} constitute a batch to be added in front of the partial schedule. While the completion times of the jobs J_i, J_{i+1}, J_{j-1} are easily determined, there is no information on how the completion times of the jobs $J_j, J_{j+1}, \ldots, J_n$ will increase because the operation completion times of any job are different on different machines and they are not recorded along with each state. Thus the contribution to the objective value made by the new batch cannot be calculated. To circumvent this difficulty, we adopt forward recursion. Define $G(i,u)$ as the optimal total weighted completion time of the jobs $J_1, \ldots, J_i$, given that the jobs are grouped into exactly $u, 1 \leq u \leq i$, batches. To find the value of $G(i,u)$, we consider different numbers of jobs (or sizes) in the last batch of the associated partial schedule of jobs $J_1, \ldots, J_i$. If $J_{j+1}, \ldots, J_i$ constitute the last batch, then the problem reduces to solving $G(j, u-1)$. The increase in the total weighted completion time caused by the jobs in this batch can be calculated as follows:

$$\max_{1\leq k\leq m}\left\{us_k + \sum_{r=1}^{i} p_{kr}\right\} \times \sum_{r=j+1}^{i} w_r,$$

where $\max_{1\leq k\leq m}\{us_k + \sum_{r=1}^{i} p_{kr}\}$ calculates the completion time of the last batch. Therefore, we can develop the following dynamic program accordingly.

Algorithm CB-WTCT for centralized batching:

Initialization:

$$G(i,u) = \begin{cases} 0, & \text{if } i = u = 0; \\ \infty, & \text{otherwise.} \end{cases}$$

Recursion: Compute the recursion for $i = 1, \ldots, n; u = 1, \ldots, i$:

$$G(i,u) = \min_{\max\{0,u-1\}\leq j<i}\left\{G(j,u-1) + \max_{1\leq k\leq m}\left\{us_k + \sum_{r=1}^{i} p_{kr}\right\} \times \sum_{r=j+1}^{i} w_r\right\}.$$

The total weighted completion time is $\min\{G(n,u) : 1 \le u \le n\}$. The partial sums $\sum_{r=1}^{i} p_{kr}$ for all i and a fixed k can be incrementally computed in $O(n)$ time in a preprocessing procedure. With these partial sums, the terms $\max_{1\le k\le m}\{us_k + \sum_{r=1}^{i} p_{kr}\}$ can be computed for all i and for all u by an $O(mn^2)$-time preprocessing procedure. Similarly, for all i and j, the terms $\sum_{r=j+1}^{i} w_r$ can be derived in $O(n^2)$ time. The preprocessing step requires $O(mn^2)$ time in all. To compute for each state $G(i,u)$, $O(n)$ iterations are required to examine all the different jobs J_j and each iteration takes $O(1)$ time. The main body of Algorithm CB-WTCT depends on $O(n^2)$ states for all the possible values of i and u. Therefore, the overall running time of the algorithm is $O(n^2 \max\{m,n\})$.

THEOREM 5: The $ID|batch^*, fixed\_seq|\sum w_iC_i$ problem subject to centralized batching can be solved in $O(n^2\max\{m,n\})$ time.

4.2. $\sum w_iC_i$ with Decentralized Batching

We proceed to deal with the decentralized case. The success of the solution approach for the $ID|batch^*|L_{\max}$ problem suggests the use of Algorithm CYMS for scheduling operations on each machine. Unfortunately, this approach fails to work for the $ID|batch^*|\sum C_i$ problem subject to decentralized batching. Consider a two-machine, 10-job instance defined by $s_1 = 5, p_{1,1} = \cdots = p_{1,9} = 2, p_{1,10} = 3; s_2 = 10, p_{2,1} = \cdots = p_{2,10} = 2$. Optimally solving for each machine individually, we have the optimal batching $\{J_1,\ldots,J_6\},\{J_7,\ldots,J_{10}\}$ on M_1 and $\{J_1,\ldots,J_{10}\}$ on M_2. Taking the maximum completion time of each operation on the two machines, the total completion time is calculated as $6\times 30 + 4\times 31 = 304$. On the other hand, if we group the operations on each machine into a single batch, then all the operations have a completion time of 26 on M_1 and a completion time of 30 on M_2, thus achieving a total completion time of 300, which is smaller than 304.

To deal with the decentralized case, we first note that there are $\binom{n-1}{u_k-1}$ ways to group n operations into u_k batches on machine M_k. Therefore the total number of possible solutions is $\prod_{k=1}^{m}\sum_{u_k=1}^{n}\binom{n-1}{u_k-1}$, which grows exponentially. Without the successful development of a polynomial time approach, we again resort to the design of a dynamic programming algorithm, which however exhibits an exponential time complexity when the number of machines m is a variable. The dynamic program runs using backward chaining. Define $G(i,u_1,\ldots,u_m,j_1,\ldots,j_m)$ as the optimal contribution of the jobs $J_i,\ldots,J_n$ to the total weighted completion time, given that there are exactly $u_k, 1\le u_k\le i-1$, setups before operation O_{ki} and that operation O_{kj_k} is the last operation in the batch containing operation O_{ki}. Note that here we use the term "optimal contribution" instead of the optimal total weighted completion time of the jobs $J_i,\ldots,J_n$ because function G does not compute their optimal total weighted completion time. A dummy job J_{n+1} with $w_{n+1} = 0$ and $p_{k,n+1} = 0$ for all k is added to define the boundary condition.

Subject to the scenario defining $G(i,u_1,\ldots,u_m,j_1,\ldots,j_m)$, operations O_{ki} and O_{kj_k} have the same completion time $u_ks_k + \sum_{r=1}^{j_k} p_{kr}$. Therefore, the completion time of job J_i is given by

$$\max_{1\le k\le m}\left\{u_ks_k + \sum_{r=1}^{j_k} p_{kr}\right\}.$$

The recursion of the dynamic programming algorithm presented below proceeds by jobs, instead of by batches as in the centralized case.

Algorithm DB-WTCT for decentralized batching:

Initialization:

$$G(n+1,u_1,\ldots,u_m,j_1,\ldots,j_m) = \begin{cases} 0, & \text{if } 1\le u_k\le n+1, 1\le k\le m;\\ & j_1=\cdots=j_m=n+1;\\ \infty, & \text{otherwise.}\end{cases}$$

Recursion: Compute the recursion for $i = n,\ldots,1$; $u_k = 1\ldots,i$; $j_k = 1,\ldots,i-u_k+1$ $(1\le k\le m)$:

$$G(i,u_1,\ldots,u_m,j_1,\ldots,j_m) = w_i\times\max_{1\le k\le m}\left\{u_ks_k + \sum_{r=1}^{j_k} p_{kr}\right\} + \min_{\mathcal{D}}\left\{G\left(i+1,u_1',\ldots,u_m',j_1',\ldots,j_m'\right)\right\},$$

where $\mathcal{D}$ is the set of feasible $2m$-tuple vectors $(u_1',\ldots,u_m',j_1',\ldots,j_m')$ defined in the following way:

If $i < j_k$, then $u_k' = u_k, j_k' = j_k$.
If $i = j_k$, then $u_k' = u_k+1, j_k' \in \{i+1, i+2,\ldots,n+1\}$.

The optimal total weighted completion time subject to decentralized batching is

$$\min\{G(1,1,\ldots,1,j_1,\ldots,j_m) : 1\le j_k\le n \text{ for } 1\le k\le m\}.$$

For each j_k, the sum $\sum_{r=1}^{j_k} p_{kr}$ can be computed by an $O(mn)$ preprocessing procedure before the dynamic program starts. Under the specifications $i,u_1,\ldots,u_m,j_1,\ldots,j_m$, the first term $w_i\times\max_{1\le k\le m}\{u_ks_k+\sum_{r=1}^{j_k} p_{kr}\}$ calculates the contribution of job J_i to the total weighted completion time, which can be derived in $O(m)$ time. There are $O(\binom{m}{l}n^{2m-l+1})$ different states of function G where there are exactly l j_ks equal

Table 1. Summary of the results.

Objective, batching mode	DP's complexity	Remark
$L_{\max}$, decentralized	$O(mn^2)$	Theorem 1
$L_{\max}$, centralized	$O(n^2 \max\{m,n\})$	Theorem 2
$\sum w_i U_i$, centralized	$O(mn^2 d_{\max}^m)$	Theorem 3: Pseudo-polynomial for fixed m
$\sum w_i U_i$, decentralized	$O(m2^m n^{m+1} d_{\max}^m)$	Theorem 4: Pseudo-polynomial for fixed m
$\sum w_i C_i$(fixed_seq), centralized	$O(n^2 \max\{m,n\})$	Theorem 5
$\sum w_i C_i$(fixed_seq), decentralized	$O(2^m n^{2m+1})$	Theorem 6 Polynomial for fixed m

to i and the cardinality of set $\mathcal{D}$ is $O(n^l)$. The overall running time of Algorithm DB-WTCT is $O(2^m n^{2m+1})$.

THEOREM 6: The $ID|batch^*, \text{fixed\_seq}|\sum w_i C_i$ problem subject to decentralized batching can be solved in $O(2^m n^{2m+1})$, which is polynomial if the number of dedicated machines is constant.

Before closing this section, we note that using the same design scheme of Algorithm DB-WTCT, we can design a backward chaining dynamic programming algorithm for the case subject to centralized batching. Nevertheless, we cannot use the forward scheme of Algorithm CB-WTCT to design an algorithm for the case subject to decentralized batching. Without the information on the last operation in the batch containing the current operation O_{ki}, we cannot determine the contribution of job J_i.

5. CONCLUSIONS

In this article, we considered a new model that includes batching decisions in concurrent open shop scheduling. Batching decisions can be centralized or decentralized. We studied three objectives, namely the maximum lateness, number of tardy jobs, and total weighted completion time. We proposed polynomial time exact algorithms to minimize the maximum lateness subject to either centralized or decentralized batching. When the objective is to minimize the number of tardy jobs, the problem remains strongly NP-hard even if a fixed job sequence is given. To minimize the total weighted completion time of a fixed or predetermined job sequence, we presented a polynomial time algorithm for the case where centralized batching is in effect. The algorithm designed for decentralized batching however has an exponential running time. When the number of dedicated machines is constant, the time complexity becomes polynomial. We summarize the results of this article in Table 1.

For further study, it will be interesting to determine the complexity status of the $ID|batch^*, fixed\_seq|\sum w_i C_i$ problem subject to decentralized batching with a variable number of machines. Moreover, at present we cannot find approximation approaches admitting performance analysis. Another complexity issue is related to $ID|batch^*, agr|\sum U_i$. While the restricted problem $ID2|d_i = d, p_{ik} \in \{0,1\}|\sum U_i$ is strongly NP-hard, problem $ID|agr|\sum U_i$ with agreeable conditions, which were defined in Section 4 on $\sum w_i U_i$, can be solved in polynomial time [15]. If the agreeable conditions are satisfied, then an optimal schedule can be obtained by first scheduling the jobs in the EDD order. It will be interesting to investigate the complexity status of $ID|batch^*, agr|\sum U_i$. The reduction from 3DM used in the proof of Lin and Kononov [19] does not work here because the constructed instance does not meet the agreeable condition. Extending the study of the concurrent open shop to simultaneously include other batching models, such as simultaneous batch processing, is another direction worthy of research.

ACKNOWLEDGMENTS

This research was supported in part by the Logistics Research Centre of The Hong Kong Polytechnic University. Lin was partially supported by the National Science Council of Taiwan under grant number 96-2416-H-009-001 as well. The authors are grateful to the anonymous referees for their constructive comments that have helped improve the presentation of the article.

REFERENCES

[1] S. Albers and P. Brucker, The complexity of one-machine batching problems, Discrete Appl Math 47 (1993), 87–107.
[2] R.H. Ahmadi and U. Bagchi, Scheduling of multi-job customer orders in multi-machine environments, ORSA/TIMS Philadelphia, 1990.
[3] P. Brucker and M.Y. Kovalyov, Single machine batch scheduling to minimize the weighted number of late jobs, Math Methods Oper Res 43 (1996), 1–8.
[4] T.C.E. Cheng, J.N.D. Gupta, and G. Wang, A review of flowshop scheduling research with setup times, Prod Oper Manage 9 (2000), 262–282.
[5] T.C.E. Cheng, Q. Wang, and J.J. Yuan, "Customer order scheduling on multiple facilities," Working Paper, Department of Logistics, The Hong Kong Polytechnic University, 2006.
[6] E.G. Coffman Jr., M. Yannakakis, M.J. Magazine, and C.A. Santos, Batch sizing and job sequencing on a single machine, Ann Oper Res 26 (1990), 135–147.
[7] D.S. Hochbaum and D. Landy, Scheduling with batching: Minimizing the weighted number of tardy jobs, Oper Res Lett 16 (1994), 79–86.
[8] S. van Hoesel, A. Wagelmans, and B. Moerman, Using geometric techniques to improve dynamic programming algorithms for the economic lot-sizing problem and extensions, Eur J Oper Res 75 (1994), 312–331.

[9] F.J. Hwang, M.Y. Kovalyov, and B.M.T. Lin, "Total completion time minimization in flow shop scheduling problems with a fixed job sequence," Presented at The 12th International Workshop on Project Management and Scheduling (PMS2010), Tours, France, 2010.
[10] F.J. Hwang and B.M.T. Lin, Coupled-task scheduling on a single machine subject to a fixed job sequence (in press).
[11] M.Y. Kovalyov, C.N. Potts, and V.A. Strusevich, Batching decisions for assembly production systems, Eur J Oper Res 157 (2004), 620–642.
[12] A. Lann, G. Mosheiov, and Y. Rinott, Asymptotic optimality in probability of a heuristic schedule for open shops with job overlaps, Oper Res Lett 22 (1998), 63–68.
[13] C.Y. Lee, T.C.E. Cheng, and B.M.T. Lin, Minimizing the makespan in three-machine assembly type flow shop problem, Manage Sci 39 (1993), 616–625.
[14] J.Y.T. Leung, H. Li, and M. Pinedo, Order scheduling in an environment with dedicated resources in parallel, J Scheduling 8 (2005), 355–386.
[15] J.Y.T. Leung, H. Li, and M. Pinedo, Scheduling orders for multiple product types with due date related objectives, Eur J Oper Res 168 (2006), 370–389.
[16] J.Y.T. Leung, H. Li, M. Pinedo, and S. Sriskandarajah, Open shops with jobs overlap—Revisited, Eur J Oper Res 163 (2005), 569–571.
[17] C.L. Li and W.Q. Xiao, Lot streaming with supplier-manufacturer coordination, Nav Res Logist 51 (2004), 522–542.
[18] B.M.T. Lin and T.C.E. Cheng, Two-machine flowshop scheduling with conditional deteriorating second operations, Int Trans Oper Res 13 (2006), 91–174.
[19] B.M.T. Lin and A.V. Kononov, A study on customer order scheduling, Technical Report 2006-NSC-95-2416-H-009-003, National Chiao Tung University, Taiwan, 2006.
[20] B.M.T. Lin and A.V. Kononov, Customer order scheduling to minimize the number of late jobs, Eur J Oper Res 183 (2007), 944–948.
[21] J.M. Moore, An *n* Job, one machine sequencing algorithm for minimizing the number of late jobs, Manage Sci 15 (1968), 102–109.
[22] C.T. Ng, T.C.E. Cheng, and J.J. Yuan, Concurrent open shop scheduling to minimize the weighted number of tardy jobs, J Scheduling 6 (2003), 405–412.
[23] C.T. Ng and M.Y. Kovalyov, Batching and scheduling in a multi-machine flow shop, J Scheduling 10 (2007), 353–364.
[24] C.N. Potts and M.Y. Kovalyov, Scheduling with batching: A review, Eur J Oper Res 120 (2000), 228–249.
[25] C.N. Potts, S.V. Sevastyanov, V.A. Strusevich, L.N. Van Wassenhove, and C.M. Zwaneveld, The two-stage assembly scheduling problem: Complexity and approximation, Oper Res 43 (1995), 346–355.
[26] T.A. Roemer, A note on the complexity of the concurrent open shop problem, J Scheduling 9 (2006), 389–396.
[27] Y.M. Shafransky and V.A. Strusevich, The open shop scheduling problem with a given sequence of jobs on one machine, Nav Res Logist 45 (1998), 705–731.
[28] C.S. Sung and S.H. Yoon, Minimizing total weighted completion time at a pre-assembly stage composed of two feeding machines, Int J Prod Econ 54 (1998), 247–255.
[29] E. Wagneur and C. Sriskandarajah, Open shops with jobs overlap, Eur J Oper Res 71 (1993), 366–378.
[30] G. Wang and T.C.E. Cheng, Customer order scheduling to minimize total weighted completion time, Proceedings of the 1st Multidisciplinary Conference on Scheduling Theory and Applications, 2003, pp. 409–416.
[31] S.T. Webster and K.R. Baker, Scheduling groups of jobs on a single machine, Oper Res 43 (1995), 692–703.