International Journal of Industrial Engineering & Production Research June 2022 Vol. 33, No. 2: 1-17
DOI: 10.22068/ijiepr.33.2.11

RESEARCH PAPER

Customer Order Scheduling with Job-Based Processing and Lot Streaming in A Two-Machine Flow Shop

Ferda Can Çetinkaya[*1] & Günce Boran Yozgat[2]

Received 11 August 2021; Revised 10 January 2022; Accepted 26 February 2022;

ABSTRACT
This study introduces a new variant of the customer order scheduling (COS) problem in which each customer orders several products processed in a two-machine flow shop. Customers' orders are satisfied by the job-based processing approach in which the same products ordered by different customers form a product lot (job). Each customer's order for a product is processed as a sublot (batch) of identical products processed together by the same machine without intermingling the sublots of other products. A sequence-independent attached setup on each machine is required before starting the process of a product lot. Each customer order is delivered in a single shipment when processing all products in that customer order is finished. The aim is to construct an optimal schedule of product lots and the sublots' sequence in every job lot by minimizing the sum of completion times of the customer orders. In our study, a mixed-integer linear programming (MILP) model and a multi-phase heuristic algorithm are developed for solving the problem. The computational experiments reveal that the proposed model solves the small-scale problem instances with and without setups optimally within three hours of a run-time limit. However, our proposed algorithm finds optimal or near-optimal solutions for the medium and large-scale problem instances in less than five seconds.

KEYWORDS: *Customer order scheduling; Job-based processing; Lot streaming; Two-machine flow shop; Total completion time; Mixed-integer linear programming; Heuristic algorithm.*

1. Introduction

In today's manufacturing world, customer satisfaction is essential for companies due to the growing competition to survive in the marketplace. Manufacturing the products with a make-to-order or make-to-stock strategy requires effective scheduling of the resources such as machines, workers, and tools, to do a set of tasks over time. In make-to-order manufacturing environments, scheduling is usually referred to as *customer order scheduling* (COS), in which several customer demands for various products are satisfied [1].

In manufacturing environments, as mentioned in [1], two extreme processing approaches exist for producing the products: (1) *order-based processing* (OBP) and (2) *job-based processing* (JBP). In the order-based processing approach, all different products in a customer order are processed consecutively without intermingling with other customer orders [2]. That is, all products ordered by a customer should be processed on a machine before processing the products of another customer order on that machine. In order-based processing, the sequence of the customer orders and the products' sequence in each customer order are determined simultaneously. However, in the job-based processing approach, a production lot is formed for each product ordered by different customers and processed before processing another product lot. The sequence of product lots (jobs) and the customer orders' sequence in each product lot are determined simultaneously [1].

When the product lots in multi-stage shops are scheduled with the job-based processing approach, a production lot of a product can be thought of as a lot (batch) of sublots, where each sublot corresponds to different customer orders. Without waiting to complete the whole product

* **Corresponding author:** ***Ferda Can Çetinkaya***
cetinkaya@cankaya.edu.tr

1. *Department of Industrial Engineering, Faculty of Engineering, Çankaya University, Ankara, Turkey.*
2. *Defence Industries Research and Development Institute, Scientific and Technological Research Council of Turkey, Ankara, Turkey.*

lot at a stage, processing different sublots of a product lot simultaneously over different machines in multi-stage shops, is called *lot streaming* (LS).

A numerical example illustrates two extreme processing approaches. Consider a problem instance with two customer orders and two products. Customer 1 orders five units of Product 1 and 10 units of Product 2. Ten units of Product 1 and 5 units of Product 2 are ordered by Customer 2. Setup and unit processing times for operations 1 and 2 of Product 1 are (5; 1) and (10; 2), respectively. Setup and unit processing times for operations 1 and 2 of Product 2 are (10; 2) and (10; 1), respectively. The optimal schedules obtained by the order-based and job-based processing approaches are illustrated in Figures 1(a) and 1(b), respectively. Note that there is no need for a setup before processing Product 2 ordered by Customer 2 in the order-based processing case since the last job (Product 2) of the previous customer order is Product 2. In the order-based processing, the total completion time TCT of the customer orders is achieved as 150 (=60+90) time units since the completion times of the orders given by Customers 1 and 2 are 60 and 90-time units, respectively. However, the TCT value becomes 140 (=65+75) time units in the job-based processing since completion times of the orders given by Customers 1 and 2 are 65 and 75-time units, respectively.

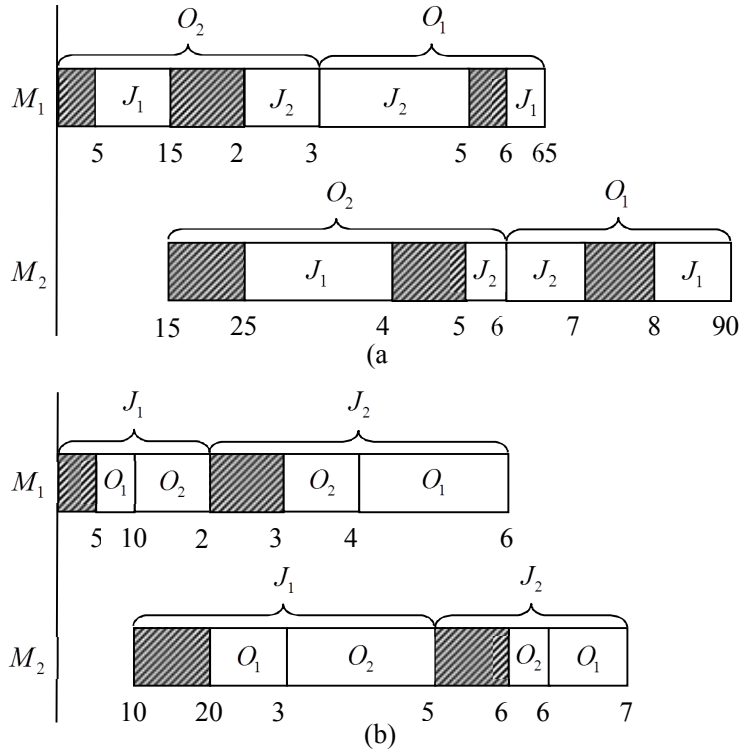


Fig. 1. (a) order-based processing with lot streaming and setup saving; and (b) job-based processing with lot streaming

The problem in this study is the scheduling of a set of customer orders in which each customer gives an order having different quantities for several types of products. The first and second operations of the products are performed on Machine 1 and Machine 2, respectively, in a two-machine flow shop environment in which job-based processing and lot streaming are used for producing the products. A customer order can only be delivered when processing all products belonging to that order is finished on Machine 2. Thus, the completion time of a customer order is the completion time of the sublot processed as the final product in that order. The aim is to construct a schedule that provides the sequence of product lots (jobs) and the customer orders' sequence in each product lot to minimize the sum of customer orders' completion times.

The rest of this paper is organized as follows. The following section presents a brief literature review on the most related works to our study. Section 3 describes the problem under study, provides the problem complexity, and presents some properties of the optimal schedule. Section

4 and 5 provide a MILP model and a heuristic algorithm, respectively. In Section 6, the computational experiments are presented to assess the effectiveness and efficiency of our proposed solution approaches. A summary and highlights of future research suggestions conclude the paper.

2. Literature Review

In the literature of manufacturing scheduling, the problem studied in this paper falls at the intersection of the COS and LS problems. We only limit our overview of the related COS and LS studies in the literature, focusing on the two-machine flow shop problems.

Customer order scheduling was first studied by Jullien and Magazine [3]. This study was followed by several studies considering various manufacturing environments such as the single-machine, parallel machines, and job shop. However, the studies on COS problems for flow shop environment are very few in the literature. One study for the two-machine flow shop environment, namely by Yang [2], has similar characteristics but not precisely as the one introduced in our study. He considered the order-based processing approach and developed a polynomial-time algorithm giving the optimal schedule for the makespan objective. He also provided the problem complexity to minimize the total completion time and developed a simple heuristic algorithm. The main difference between the problem studied by Yang [2] and the one investigated in our study is the processing approaches, which are the order-based and job-based processing in his and our study, respectively. Furthermore, no setup times are assumed between different products (jobs) in [2], whereas sequence-independent attached setups exist between products (jobs) in our study. A thorough review of COS problems is provided by Xu et al. [4].

On the other hand, the idea of lot streaming was first used by Reither [5] and rediscovered in the late 1980s. In the past three decades, the use of the LS idea in scheduling problems of multi-stage manufacturing environments has received much attention from researchers. Different aspects of the LS problem have been studied since LS has several advantages in make-to-order manufacturing environments to improve delivery times, especially when significantly large setup times are needed before starting the process of the products. Based on the number of job lots, the literature on LS problems for various job and shop characteristics can easily be divided into two categories: one deals with a single-job (single-lot), and the other addresses the multi-job (multi-lot) case. While the number of sublots and their sizes are determined in single-job problems, the number of sublots, sublot sizes, and sublots' sequence are determined in multi-job problems. Here, we limit our literature review on the LS studies to the multi-job case with a two-machine flow shop to ease the understanding of the proper place of the study under consideration.

Vickson and Alfredsson [6] considered the multi-job case with unit-sized sublots (i.e., sublots with a single item of a product). They investigated the benefits of sublots in two and three-machine flow shop environments. To solve the makespan minimization problem in the two-machine flow shop and the special case of the three-machine flow shop problem, they proposed a modification in Johnson's algorithm [7]. Çetinkaya and Kayalıgil [8] obtained a combined procedure, modifying Johnson's algorithm, handling both sequence-independent attached and detached setup cases. Çetinkaya [9] studied the two-machine flow shop problem with sequence-independent detached setups and removal times. He showed that the sublot-sizing and job-sequencing problems are solved independently and provided an optimal schedule to minimize the makespan by modifying Johnson's algorithm. Vickson [10] considered the same problem studied in [9] and provided an optimal solution procedure with sequence-independent attached or detached setups on the machines and transportation times between the machines. Glass and Possani [11] considered the same problem studied in [10], showed that sublot-sizing and job-sequencing problems are solved independently as in [9], and developed an optimal solution procedure with attached setup times and transportation times. Sriskandarajah and Wagneur [12] studied the problem with no-wait restriction between machines. Pranzo [13] extended the work in [9] to the case in which a limited buffer exists between machines. The makespan minimization problem for a two-machine flow shop with a single transport agent and sequence-independent attached/detached setups on the machines was considered solved in [14]. Comprehensive reviews of scheduling problems with the LS concept for single-job lot and multi-job cases can be found in [15-18].

A brief overview of the literature indicates that the study by Liu [19] is the only one to apply the LS concept with order-based processing to the COS problem in job shops. Thus, the study under consideration will be the second one in the

literature that combines the LS concept and the COS problem. The contributions of our study to literature are as follows:

- To the best of our knowledge, a study that considers the COS problem with lot streaming for a flow shop to minimize the sum of the customers' completion times does not exist. Hence, we intend to contribute to the literature on COS and LS in this direction.
- A MILP model has been presented to solve the problem under study optimally.
- A multi-phase heuristic algorithm has been proposed to solve medium and large-scale problems.
- Our proposed heuristic algorithm can be easily implemented, and its solutions are optimal for small-scale problems. It also provides satisfactory solutions (optimal or near-optimal) for problem instances where the MILP model cannot achieve the optimal solution.

3. Problem Definition and Preliminary Results

In this section, the problem under study and its assumptions are first described in detail; next, its complexity is discussed. Finally, some preliminary results regarding the properties of the optimal schedule are provided.

3.1. Problem definition and assumptions

Consider a scheduling problem $\boldsymbol{P}$ of K customer orders $(k = 1,2,\dots,K)$. Customer order k is composed of several products from a set of N products. Products $(j = 1,2,\dots,N)$ are processed in a two-machine flow shop in which each product has one operation on each machine. The first and second operations of all products are performed by Machines 1 and 2, respectively. Unit processing time of the product j on machine m $(m = 1,2)$ is $p_{j,m}$, and $t_{j,m}$ units of time are required to set up the machine m before starting to process the first sublot of product j. Customer k orders $Q_{j,k}$ units of product j, which is called the *product sublot size*. While processing the products, all sublots of the same product are processed together on each machine. Furthermore, a sublot of a product processed on Machine 1 is moved to Machine 2, while other sublots of the same product are being processed on Machine 1. That is, overlapping the two operations of the same product through the creation of sublots (i.e., lot streaming) is allowed without intermingling the sublots of other products, as shown in Fig. 2(b) for the numerical example discussed in Section 1.

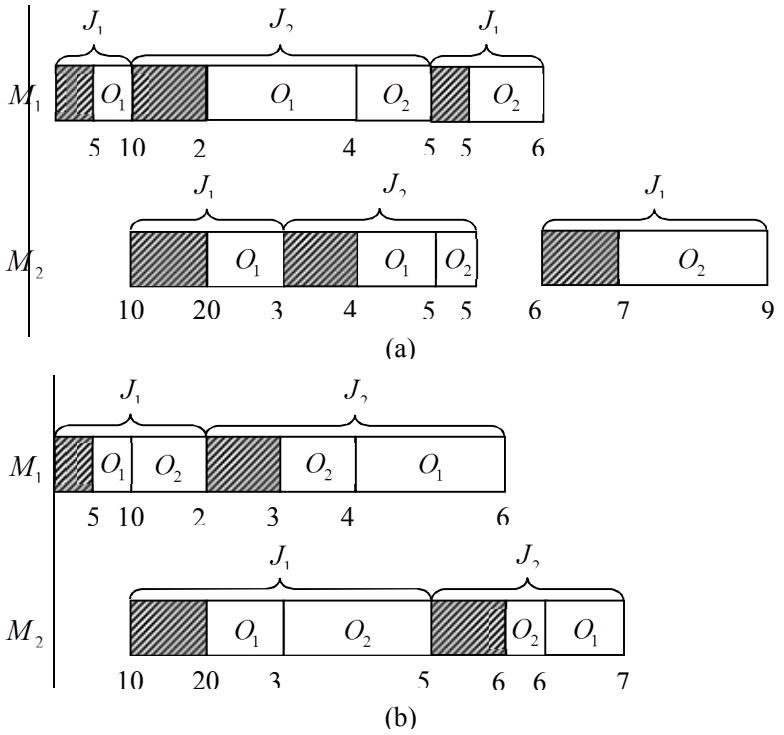


Fig. 2. Schedules (a) with and (b) without intermingling the sublots of different products (jobs)

Besides, the following additional assumptions are considered:

- All customer orders are available for processing simultaneously.
- Machines are ready to process the products and operate independently.
- The product processing sequence is the same at both machines; i.e., permutation flow shop is considered.
- Both machines can process only one sublot at a time, and each sublot can be processed by only one machine at any given time; i.e., machines are not batch-processing machines.
- Setup times are *sequence-independent and attached*; i.e., the setup on a machine cannot be performed before the first sublot belonging to a product physically arrives at that machine. However, there is no need to make a setup between sublots of the same product.
- Sublots of a product are immediately sent to Machine 2 without waiting to complete all other sublots of the same product; i.e., sublot availability is considered.
- Transportation times from Machine 1 to Machine 2 are negligibly short; thus, they are ignored.
- Storage space between machines is sufficient to stock the processed sublots on Machine 1.
- Each customer order is delivered when processing all products in the customer order is completed; i.e., no partial delivery is allowed. Thus, the completion time of the sublot processed as the final product in a customer order determines the completion time of that customer order.

The aim is to find a sequence of the product lots and the sublots' sequence in each product lot so that the sum of the customer orders' completion times is minimized.

3.2. Problem complexity

When there is a single customer order, problem ***P*** becomes the maximum completion time (makespan) minimization problem since the completion time of the final sublot processed determines the customer order completion. The optimal solution to this reduced problem is trivial, and the algorithm by Johnson [8] gives the optimal sequence of the jobs. Therefore, to investigate the complexity of problem ***P***, it is assumed that there is more than one customer order.

Theorem 1 provides the complexity of the problem ***P***.

Theorem 1. *The problem **P** is NP-hard in the strong sense.*

Proof: Consider a particular case of problem ***P***, where a single customer orders one unit of each product (job), and the setup times are omitted. This special case is equivalent to the classical two-machine flow shop problem $F2//\sum C_j$, where the total completion time of the products (jobs) is minimized and proven to be *NP*-hard in the strong sense by Gonzalez and Sahni [20]. Hence, problem ***P*** is also *NP*-hard in the strong sense.

3.3. Preliminary results

In this section, some definitions and theorems are given to derive the structural properties of the optimal solution to problem ***P***.

Definition 1 (Smith et al. [21]): *The M-machine N-job flowshop is called an ordered flow shop if the following two properties are satisfied:*

(i) If a particular job has a smaller processing time on any machine than does a second job on the same machine, this implies that the processing time of this first job is less than or equal to the processing time of the second job on all corresponding machines.

(ii) If a job has its rth smallest processing time on some machine m where $m = 1,2,\dots,M$, this implies that every other job will have its rth smallest processing time on the same machine m where $r = 1,2,\dots,M$.

Using the results in [21], Panwalker and Khan [22] provided the following result for the ordered flow shops.

Lemma 1. *In the optimal schedule for the total completion time minimization problem in the ordered flow shop, jobs are sequenced in non-decreasing order of their processing times.*

Çetinkaya and Gupta [23] presented the following result for the single job-lot streaming problem in the M-machine flow shop.

Lemma 2. *The single job-lot streaming problem with the total completion time minimization objective satisfies the characteristics of the ordered flow shops.*

When all customers order a single product (job), Theorem 2 describes the optimal schedule for this special case of problem ***P***.

Theorem 2. *In an optimal schedule of the*

*problem **P**, the customer orders (sublots) are processed in non-decreasing order of their sublot sizes if all customers order a single product.*

Proof: When all customers order a single product, this special case of problem ***P*** is equivalent to the single job-lot streaming problem with the total completion time minimization objective. From Lemmas 1 and 2, it is clear that the customer orders (sublots) are processed in non-decreasing order of their sublot sizes.

Theorem 3 describes the optimal schedule for all sublots of the product processed as the last in the product (job) sequence and will be used to increase the efficiency of our proposed heuristic algorithm in Section 5.

Theorem 3. *In an optimal schedule of problem **P**, sublots of the final product (job) in the job sequence are processed in non-decreasing order of sublot sizes.*

Proof: The sum of the completion times for the customer orders (sublots) having no demand for the final product (job) in the job sequence does not depend on the sublots of the last job in the job sequence. The problem of finding the optimal sequence for the sublots of the last job in the job sequence can be considered the single product case as given in Theorem 2. Thus, sublots of the last job in the job sequence are processed in non-decreasing order of the sublot sizes.

4. Mathematical Programming Model

In this section, a MILP model is presented for solving problem ***P***. This model is an extension of the model developed for the classical two-machine multi-job lot streaming problem in [17]. The MILP model provides the optimal schedule with the *job sequence* (i.e., sequence of the products) and the *sublot sequence* (i.e., customer orders' sequence) in each job.

The following parameters, indices, sets, and decision variables are used to develop the MILP model.

4.1. Parameters, indices and sets

K Number of orders given by different customers.
k Index for customer orders ($k = 1, \dots, K$).
N Number of products (jobs).
j Index for jobs ($j = 1, \dots, N$).
O_k Set of jobs ordered by customer k.
J_j Set of customers ordering for job j.
n_j Number of customers ordering job j.
$Q_{j,k}$ Quantity (number of identical items) of product (job) j ordered by customer k.
L_j Lot size (total order quantity) for job j, where $L_j = \sum_{k \in J_j} D_{j,k}$.
i Sublot index ($i = 1, \dots, n_j$).
m Machine index ($m = 1,2$).
$p_{j,m}$ Unit processing time of job j on machine m.
$t_{j,m}$ Time for the attached setup performed before processing job j on machine m.
V A sufficiently large number.

4.2. Decision variables

$X_{i,j,k}$ 1 if ith sublot of job j is processed for customer order k; otherwise, 0.
$Y_{h,j}$ 1 if job h precedes j; otherwise, 0.
$S_{i,j}$ Size of the ith sublot of job j.
$C_{i,j,m}$ Completion time of the ith sublot of job j on machine m.
CT_k Completion time of the customer order k.

4.3. The MILP model

The MILP model for the problem ***P*** is as follows:

$$\text{Minimize} \quad TCT = \sum_{k=1}^{K} CT_k \tag{1}$$

$$\text{Subject to} \quad \sum_{i=1}^{n_j} S_{i,j} = L_j \qquad \text{for } j = 1,2, \dots, N \tag{2}$$

$$\sum_{i=1}^{n_j} X_{i,j,k} = 1 \quad \text{for } k = 1,2,\dots,K;\ j \in O_k \tag{3}$$

$$\sum_{k \in J_j} X_{i,j,k} = 1 \quad \text{for } j = 1,2,\dots,N;\ i = 1,2,\dots,n_j \tag{4}$$

$$s_{i,j} = \sum_{k \in J_j} Q_{j,k} X_{i,j,k} \quad \text{for } j = 1,2,\dots,N;\ i = 1,2,,\dots,n_j \tag{5}$$

$$C_{1,j,1} - p_{j,1}\, s_{1,j} \geq t_{j,1} \quad \text{for } j = 1,2,\dots,N \tag{6}$$

$$C_{i+1,j,1} - p_{j,1}\, s_{i+1,j} \geq C_{i,j,1} \quad \text{for } j = 1,2,\dots,N;\ i = 1,2,\dots,n_j - 1 \tag{7}$$

$$C_{1,j,2} - p_{j,2}\, s_{1,j} \geq C_{1,j,1} + t_{j,2} \quad \text{for } j = 1,2,\dots,N \tag{8}$$

$$C_{i+1,j,2} - p_{j,2}\, s_{i+1,j} \geq C_{i+1,j,1} \quad \text{for } j = 1,2,\dots,N;\ i = 1,2,\dots,n_j - 1 \tag{9}$$

$$C_{i+1,j,2} - p_{j,2}\, s_{i+1,j} \geq C_{i,j,2} \quad \text{for } j = 1,2,\dots,N;\ i = 1,2,\dots,n_j - 1 \tag{10}$$

$$\left(C_{i,j,m} - p_{j,m}\, s_{i,j}\right) - \left(C_{e,h,m} - p_{h,m}\, s_{e,h}\right) + V\left(1 - Y_{h,j}\right) \geq \left(L_h - \sum_{r=1}^{e-1} s_{r,h}\right) p_{h,m} + t_{j,m} + p_{j,m} \sum_{r=1}^{i-1} s_{r,j} \quad \text{for } h \neq j;\ e = 1,2,\dots,n_h;\ h = 1,2,\dots,N;\ i = 1,2,\dots,n_j;\ j = 1,2,\dots,N;\ m = 1,2 \tag{11}$$

$$\left(C_{e,h,m} - p_{h,m}\, s_{e,h}\right) - \left(C_{i,j,m} - p_{j,m}\, s_{i,j}\right) + V \cdot Y_{h,j} \geq \left(L_j - \sum_{r=1}^{i-1} s_{r,j}\right) p_{j,m} + t_{h,m} + p_{h,m} \sum_{r=1}^{e-1} s_{r,h} \quad \text{for } h \neq j;\ e = 1,2,\dots,n_h;\ h = 1,2,\dots,N;\ i = 1,2,\dots,n_j;\ j = 1,2,\dots,N;\ m = 1,2 \tag{12}$$

$$CT_k \geq C_{i,j,2} - V(1 - X_{i,j,k}) \quad \text{for } k = 1,2,\dots,K;\ j \in O_k;\ i = 1,2,\dots,n_j \tag{13}$$

$$X_{i,j,k}, Y_{h,j} \in \{0,1\} \quad \text{for } \forall h,i,j,k \tag{14}$$

$$s_{i,j}, C_{i,j,m}, CT_k \geq 0 \quad \text{for } \forall i,j,k,m \tag{15}$$

The objective in (1) of the MILP model is the sum of customer orders' completion times. Constraint set (2) guarantees that the sum of the items in the sublots of a job must be equal to the total number of items in that job. i.e., the sum of the sublot sizes of a job must be equal to the lot size (total demand) for that job. Constraint set (3) ensures that each job of a customer order is assigned to only one sublot of that job. Constraint set (4) guarantees that each sublot of a job can be assigned to only one customer order. Constraint set (5) satisfies that the sum of the items in a sublot of a job must be equal to the demand for that job in the customer order assigned to the sublot. Constraint set (6) ensures that the processing of the first sublot of any job on Machine 1 begins after the setup on the same machine has been finished. Constraint set (7) guarantees that a sublot (except the first sublot) of a job begins processing on Machine 1 after the previous sublot is completed on the same machine. Constraint set (8) ensures that the first sublot of a job begins processing on Machine 2 after its processing on Machine 1 and the setup on Machine 2 have been finished. Constraint set (9) ensures that all the sublots (except the first sublot) of a job begin processing on Machine 2 after they are completed on Machine 1. Constraint set (10) guarantees that a sublot (except the first sublot) of a job begins processing on Machine 2 after the previous sublot is finished on the same machine. The terms on the right-hand side of the constraint set (11) ensure that the difference between the start times of sublots e and i is at least equal to the sum of the processing times of the sublots e to n_h of job h and 1 to $i-1$ of job j and the setup time for job j. Note that either constraint set (11) or (12) is valid for an optimal solution. Constraint set (13) ensures that the completion time of a customer order is the maximum of completion times for the jobs belonging to that customer order. Constraint sets (14) and (15) are binary and non-negativity restrictions.

4.4. Parameter V

Selecting a suitable value of the parameter V in Constraint sets (11), (12), and (13) affects the computational effort of the model since the parameter V defines the feasible region. One should restrict the value of parameter V with a sufficiently smallest positive number to reduce the feasible region of the MILP model. A sufficiently smallest positive number for parameter V is $V = \sum_{j=1}^{N} \sum_{m=1}^{2} (t_{j,m} + p_{j,m} L_j)$, which is equivalent to the sum of all jobs' setup and processing times on both machines, and a

reasonable value in Constraint sets (11), (12), and (13). In solving the MILP model, the rounded-up value of V to the nearest number, which is the multiple of a hundred, will be used.

5. Heuristic Algorithm

The experiments in Section 6 show that the optimal solution of the MILP model cannot be obtained by the solver GAMS in a reasonable time limit for medium and large-scale problems. Furthermore, developing a polynomial-time algorithm that provides the optimal solution for all problem instances is not possible as problem ***P*** is *NP*-hard. Thus, a fast heuristic algorithm to provide optimal or near-optimal solutions within relatively short times has been developed. The proposed heuristic is a multi-phase algorithm having four phases. An initial schedule is generated in the first phase. The insertion algorithm then improves this schedule in the second phase. The third phase improves the schedule obtained in the second phase by pairwise exchanging the positions of the customer orders. In the last phase, a tabu search algorithm improves the schedule obtained in the third phase. Now, the following sections give detailed descriptions of the phases.

5.1. Phase 1 (An initial schedule generation)

Before providing the steps of Phase 1, it is better to give the following definition of the run-in time of a job since an initial schedule is found by considering the run-in times of the jobs.

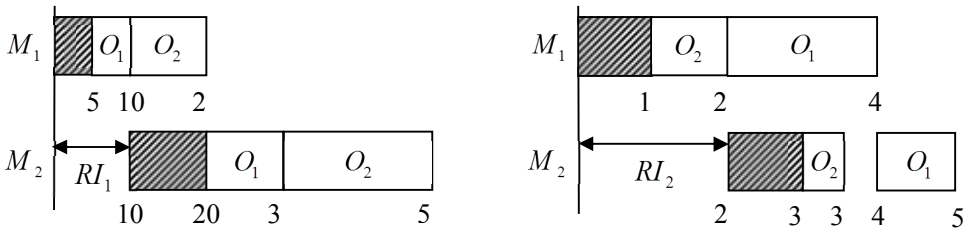
Definition 2 (Run-in Time). *Run-in time* RI_j of the job j in the problem ***P*** is the time that elapses between the starting times of the setups on machines 1 and 2. RI_j is calculated as $RI_j = t_{j,1} + p_{j,1}s_{[1],j}$, where

$t_{j,1}$ = sequence-independent attached setup time on Machine 1,

$p_{j,1}$ = unit processing time on Machine 1, and

$s_{[1],j}$ = size of the first sublot (customer order) among the sublots processed in non-decreasing order of their sublot sizes in job j.

To illustrate the run-in times, consider the numerical example discussed in Section 1. When the customer orders of Product (Job) 1 are sorted in non-decreasing order of their demands, the order belonging to Customer 1 is the first sublot to be processed. The run-in time for Product 1 becomes 10 (=5+5) time units, as illustrated by Fig. 3(a). Similarly, when the customer orders of Product 2 are sorted in non-decreasing order of their demands, the order belonging to Customer 2 is processed as the first sublot. The run-in time for Product 2 becomes 20 (=10+10) time units, as illustrated by Fig. 3(b).



(a) Job J_1 (b) Job J_2

Fig. 3. Run-in times for the products (jobs)

The steps of Phase 1 are presented below.

Step 1. Construct the customer order list and the job list, where the customer orders (sublots) in each job are sorted in non-decreasing order of sublot sizes.

Step 2.

(a) Consider the list of customer orders and sort them in non-decreasing order of their number of jobs.

(b) If at least two customer orders have the same number of jobs, then calculate their completion times independently and sort the customer orders in non-decreasing order of their completion times.

Step 3. If the customer order has more than one job, then

(a) Calculate the run-in times of these jobs as $RI_j = t_{j,1} + p_{j,1}s_{[1],j}$, where the run-in time of a job is the sum of the setup time on Machine 1 and the total processing time of the first sublot of this job.

(b) Sort the jobs in non-decreasing order of their run-in times.

(c) Select the first job in the sorted job list as the first job of the initial job sequence *IJS*

and remove this selected job from the order list.

Step 4. If all jobs are placed in IJS, calculate the TCT value for IJS, and go to Phase 2; otherwise, go to Step 2(a).

5.2. Phase 2 (Improving the initial schedule by the insertion algorithm)

Insertion algorithm, also known as NEH algorithm, is a kind of neighbourhood algorithm proposed by Nawaz et al. [24] to solve the classical *m*-machine flow shop scheduling problem to minimize the makespan. In this phase, we have adapted the insertion algorithm to our heuristic algorithm to improve the initial schedule obtained in Phase 1. Although Phase 2 is the same as the modified NEH procedure in Phase 1 of the heuristic algorithm in [1], a stepwise description of the NEH procedure is also given below for completeness.

Step 1.

(a) Let IJS be the initial job sequence obtained in Phase 1 and select the first two jobs a_1 and a_2 from this sequence.

(b) Form two partial sequences $a_1 - a_2$ and $a_2 - a_1$.

(c) Calculate the TCT value for each partial sequence and select the partial sequence with minimum TCT value as the best partial sequence BPS.

Step 2.

(a) Consider the job in the following position of the sequence IJS, and generate partial sequences by placing this new job in all possible beginning, between, and ending positions of the sequence BPS.

(b) Calculate the TCT value for each partial sequence and select the partial sequence with minimum TCT value as the best partial sequence BPS.

Step 3. If all jobs of the sequence IJS are considered, then go to Phase 3; otherwise, go to Step 2(a).

5.3. Phase 3 (Improving the schedule obtained in the second phase by pairwise exchanging the positions of the customer orders)

When we generate the initial schedule in Phase 1, we assume that the customer orders (sublots) in each job are sequenced in non-decreasing order of their sizes. However, TCT value may be improved by pairwise exchanging the positions of the customer orders in each job. Below is the stepwise description of the third phase of our heuristic algorithm.

Step 1. Let the first job of the job sequence obtained in Phase 2 be the current job.

Step 2.

(a) Check whether there is a customer order in the current job such that this customer order does not have the jobs processed after the current job in the job sequence obtained in Phase 2.

(b) If there is such a customer order, then go to Step 3a; otherwise, consider the next job in the job sequence as the current job. If the current job is in the last position of the job sequence, go to Phase 4; otherwise, go to Step 2(a).

Step 3.

(a) Let the first customer order, which does not have the jobs processed after the current job of the job sequence, be the current customer order.

(b) Temporarily pairwise exchange the positions of the current customer order and the customer order immediately succeeding the current customer order.

(c) Check whether the pairwise exchange in Step 3b improves the total completion time.

(d) If the pairwise interchange does not improve the total completion time, then do not make this exchange and go to Step 3(e); otherwise,

(i) Make this pairwise exchange.

(ii) If the new position of the current customer order is the first position in the current job, then go to Step 3(e); otherwise, go to Step 3(b).

(e) Check for the next customer order, which does not have the jobs processed after the current job of the job sequence, and go to Step 2(b).

5.4. Phase 4 (Improving the schedule obtained in the third phase by a Tabu Search algorithm)

Tabu Search (TS) is a widely used metaheuristic algorithm searching a global optimum for many industrial engineering related problems [1, 27-28]. TS algorithm, which was first developed by Glover [25-26], takes an initial solution (schedule) generated randomly or obtained by a simple rule or a constructive heuristic algorithm. The solution achieved by Phase 3 of our heuristic algorithm will be the initial solution of our TS algorithm described below. In the classical application of the TS algorithm, all possible mutations, which are all solutions produced from

the current solution by a solution generation mechanism, are determined. The objective function value TCT is calculated for each mutation. The mutation having the TCT value is selected as the candidate solution. The candidate solution becomes the best solution if the candidate solution's TCT value is better than the current best solution's TCT value. The next tabu search iteration continues with the job sequence of the new solution. However, to reduce the TCT value, we modify the application of the TS algorithm by inserting a new step before selecting the candidate solution at any iteration. Phase 3 of our heuristic algorithm is inserted as this new step and implemented before selecting the candidate solution in the neighbourhood of a current solution. The process continues with the previous solution if the TCT value is not improved after applying Phase 3.

Tabu search iterations in Phase 4 of our heuristic algorithm are run until one of the two stopping conditions is reached. We let the TS algorithm run for five iterations as the first stopping condition. The second stopping condition is that the TS algorithm ends when all possible mutations are worse than the parent.

The size of the tabu list, which is the list for keeping the history of moves and avoiding the return to a solution visited before, is also an important parameter. Our preliminary experiments set the tabu list size to different values ranging from 2 to 10, and some pilot runs were made. It has been observed that better solutions are obtained when the tabu list size is set to 5; thus, it was used for the rest of the experiments.

The following stepwise description of Phase 4 is the same as the TS algorithm in Phase 2 of the heuristic algorithm in [1]. However, Phase 3 above is applied in Step 2(b) of the TS algorithm instead of the SCO algorithm in [1].

Step 1. Set the iteration counter ic to 1, i.e., set $ic = 1$. Set the initial schedule σ_1 to the schedule obtained in Phase 3 above. Set the best schedule σ_B to σ_1, i.e., set $\sigma_B = \sigma_1$.

Step 2.

(a) Generate the neighbourhood of the schedule σ_{ic} by adjacent pairwise interchanges of the jobs in the schedule σ_{ic}.

(b) For each of the mutation in the neighbourhood of the schedule σ_{ic}, apply Phase 3 above.

(c) If the total completion time value of each mutation is bigger than the total completion time of the parent schedule σ_{ic}, then stop; otherwise, from the neighbourhood of the schedule σ_{ic}, select the schedule with the lowest total completion time value as the candidate schedule σ_C.

Step 3.

(a) If the move $\sigma_{ic} \rightarrow \sigma_C$ is prohibited by a mutation on the tabu list, set $\sigma_{ic+1} = \sigma_{ic}$ and go to Step 4; otherwise,

(i) Delete the entry at the bottom of the tabu list.

(ii) Push all other entries in the tabu list one position down.

(iii) Enter reverse mutation at the top of the tabu list.

(iv) Set $\sigma_{ic+1} = \sigma_C$.

(v) Set the best schedule to the candidate schedule (i.e., set $\sigma_B = \sigma_C$), if the total completion time value of the candidate schedule is smaller than the total completion time vale of the best schedule, i.e., $TC(\sigma_C) < TC(\sigma_B)$.

(vi) Go to Step 4.

Step 4.

(a) Increment the iteration counter ic by 1. i.e., set $ic = ic + 1$.

(b) If the iteration counter ic is equal to the pre-specified value NI for the number of iterations (i.e., $ic = NI$), then stop; otherwise, go to Step 2.

5.5. Numerical example

Consider a problem instance with five customer orders and five products to illustrate the proposed heuristic algorithm. In Table 1, order quantities of the products, setup times, and the unit processing times are given.

Tab. 1. Data for the numerical example

	$Q_{j,k}$					$t_{j,m}$		$p_{j,m}$	
	O_1	O_2	O_3	O_4	O_5	m_1	m_2	m_1	m_2
J_1	-	6	1	-	-	89	8	1	6

			0				7		
J_2	3	4	10	2	8	73	55	5	9
J_3	-	-	-	9	5	2	60	10	8
J_4	9	8	8	4	7	38	80	4	9
J_5	4	-	-	-	-	68	31	3	7

Phase 1. The lists of customer orders and jobs are constructed as follows:

List of Customer Orders: $O_1 = \{J_2, J_4, J_5\}, O_2 = \{J_1, J_2, J_4\}, O_3 = \{J_1, J_2, J_4\}, O_4 = \{J_2, J_3, J_4\}, O_5 = \{J_2, J_3, J_4\}$

List of Jobs (Products): $J_1 = \{O_2[6], O_3[10]\}$, $J_2 = \{O_4[2], O_1[3], O_2[4], O_5[8], O_3[10]\}$, $J_3 = \{O_5[5], O_4[9]\}$, $J_4 = \{O_4[4], O_5[7], O_2[8], O_3[8], O_1[9]\}$, $J_5 = \{O_1[4]\}$

Note that the number in a bracket denotes the demand for a job in a customer order and all customer orders have the same number of jobs, which is three. The completion times for the customer orders are independently determined as $CT(1) = 390$, $CT(O_2) = 461$, $CT(O_3) = 543$, $CT(O_4) = 423$, and $CT(O_5) = 483$. When the customer orders are sorted in non-decreasing order of their completion times, the list of customer orders is obtained as $O_1 - O_4 - O_2 - O_5 - O_3$. The first customer order in the sorted list of customer orders obtained in Step 2 is the order O_1, which has three jobs J_2, J_4, and J_5. The run-in times for the jobs J_2, J_4, and J_5 equal to 83, 54, and 80 time units, respectively. When these jobs are sorted in non-decreasing order of their run-in times, the sorted job list becomes $J_4 - J_5 - J_2$. Thus, job J_4 will be the first job of the initial job sequence since it has the minimum run-in time. When job J_4 is removed from all customer orders having this job, the updated list of customer orders becomes $O_1 = \{J_2, J_5\}$, $O_2 = \{J_1, J_2\}$, $O_3 = \{J_1, J_2\}$, $O_4 = \{J_2, J_3\}$, $O_5 = \{J_2, J_3\}$. When the remaining steps in Phase 1 are applied, the initial job sequence is obtained as $J_4 - J_5 - J_2 - J_3 - J_1$, and the customer orders' sequences in each job becomes $J_4 = \{O_4[4], O_5[7], O_2[8], O_3[8], O_1[9]\}$, $J_5 = \{O_1[4]\}$, $J_2 = \{O_4[2], O_1[3], O_2[4], O_5[8], O_3[10]\}$, $J_3 = \{O_5[5], O_4[9]\}$, and $J_1 = \{O_2[6], O_3[10]\}$. The associated TCT value for the initial job schedule is 4,799 time units.

Phase 2. From the initial job sequence $IJS = J_4 - J_5 - J_2 - J_3 - J_1$ obtained in Phase 1, the first two jobs J_4 and J_5 are selected. Two partial sequences $J_4 - J_5$ and $J_5 - J_4$, where the customer order sequences in each job are as given in the last Step 4 of Phase 1 above, are formed. TCT values in these partial sequences are $TCT(J_4 - J_5) = 1{,}602$ and $TCT(J_5 - J_4) = 1{,}968$. The partial sequence $J_4 - J_5$ is selected since its TCT value is smaller than TCT value of the partial sequence $J_5 - J_4$. Job J_2 is selected as the next job from the sequence IJS, and three partial sequences $J_2 - J_4 - J_5$, $J_4 - J_2 - J_5$, and $J_4 - J_5 - J_2$ are formed. TCT values in these partial sequences are $TCT(J_2 - J_4 - J_5) = 3{,}237$, $TCT(J_4 - J_2 - J_5) = 3{,}362$ and $TCT(J_4 - J_5 - J_2) = 3{,}400$. The partial sequence $J_2 - J_4 - J_5$ is selected since it has the minimum total completion time among the three partial sequences. When Step 2 is repeated by considering the next two jobs (J_3 and J_1) from the initial job sequence obtained in Phase 1, an improved job sequence $J_3 - J_2 - J_4 - J_5 - J_1$, which has a TCT value of 4,605 time units, is obtained.

Phase 3. The first job of the job sequence $J_3 - J_2 - J_4 - J_5 - J_1$ obtained in Phase 2 is $J_3 = \{O_5[5], O_4[9]\}$, which is considered the current job. The sublots of job $J_3 = \{O_5[5], O_4[9]\}$ are customer orders O_5 and O_4, and these customer orders have jobs J_2, J_4 and J_5, which are processed after job J_3 in the job sequence $J_3 - J_2 - J_4 - J_5 - J_1$. Thus, the next job in the job sequence is job $J_2 = \{O_4[2], O_1[3], O_2[4], O_5[8], O_3[10]\}$ ordered by all customers, and these customer orders have jobs J_4 and J_5, which are processed after job J_2 in the job sequence $J_3 - J_2 - J_4 - J_5 - J_1$. Thus, pairwise interchanging of the customer orders is not possible. Therefore, the next job in the sequence $J_3 - J_2 - J_4 - J_5 - J_1$, which is job $J_4 = \{O_4[4], O_5[7], O_2[8], O_3[8], O_1[9]\}$, should be considered. The sublots O_4 and O_5 of job J_4 do not have the jobs processed after job J_4 in the job

sequence to pass to the next step. Customer order O_5 follows the customer order O_4 in job J_4. Pairwise interchange of customer orders O_4 and O_5 increases the total order completion time TCT to 4,632 so that this interchange is not made. Step 2 is repeated again since the remaining sublots O_1, O_2 and O_3 of job J_4 exist in jobs J_5 and J_1 of the sequence $J_3 - J_2 - J_4 - J_5 - J_1$.The next job in the sequence $J_3 - J_2 - J_4 - J_5 - J_1$ is job J_5 having only one customer order, so there is no need to make any pairwise interchange. Now, the new current job becomes job J_1. Phase 2 is terminated since this job is the last job of the sequence $J_3 - J_2 - J_4 - J_5 - J_1$.

Phase 4. The initial schedule $J_3 - J_2 - J_4 - J_5 - J_1$ for the tabu-search algorithm is taken from the solution obtained in Phase 3 of the heuristic algorithm and its TCT value is 4,605. TS algorithm terminates with an improved schedule $J_3 - J_2 - J_5 - J_4 - J_1$ in two iterations.

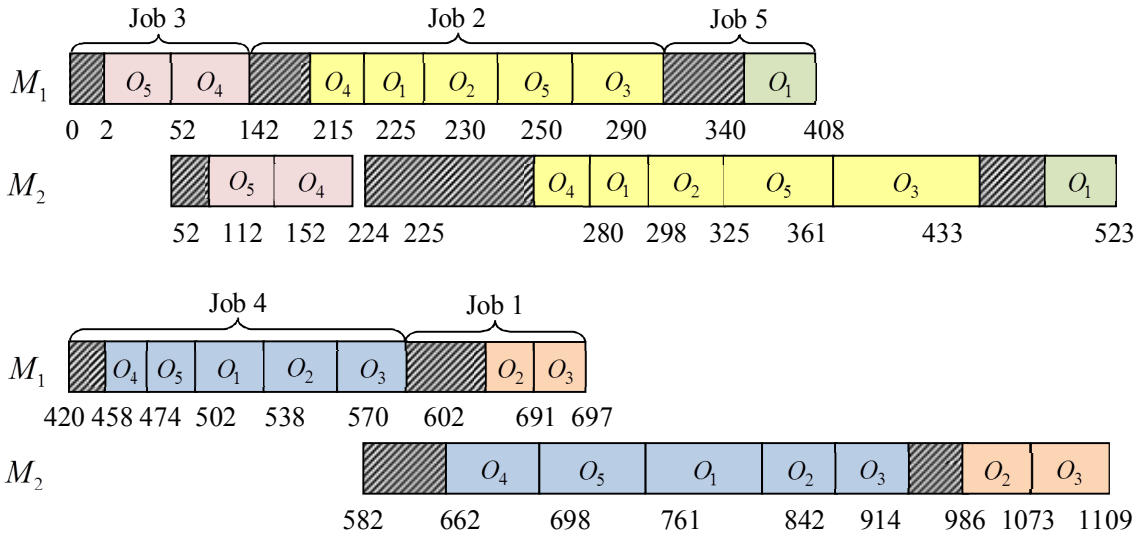


Fig. 4. Gantt chart of the schedule obtained by the heuristic algorithm

Fig. 4 illustrates the Gantt chart of the schedule obtained by Phase 4. Note that TCT value of this schedule is 4579 (=842+1109+1169+698+761) and equivalent to the TCT value of the optimal schedule achieved by solving the MILP model.

6. Computational Experiments

This section presents the computational tests to examine the effectiveness and efficiency of our solution approaches MILP model and the heuristic algorithm. Our mathematical model is solved using the CPLEX solver of the software package GAMS with version 24.1. The proposed heuristic algorithm is coded in Java programming language. A computer with 128 GB RAM and a processor running at 2.00 GHz under the operating system Windows 10 is used for all experiments.

6.1. Parameter settings and problem instances generation

The values of the parameters used in the experiments are generated as in [29]. The number of customer orders (K) is taken as 5 and 10; the number of jobs (N) is taken as 5, 10, 15, and 20. The number of customers ordering job j (n_j) and the quantity of product (job) j ordered by customer k ($Q_{j,k}$) are randomly generated from the discrete uniform distributions DU[1, K] and DU[1, 10], respectively. The processing times ($p_{j,m}$) and the setup times ($t_{j,m}$) are randomly generated from the discrete uniform distributions DU[1, 10] and DU[0, 100f], respectively, where f is taken as 0.5, 1.0, 1.5, and 2.0. Thus, for each of 8 possible combinations of the parameters K and N, 25 problem instances (replicates) are randomly generated. A total of 400 problem instances, 200 for setup and 200 for no-setup cases, are tested.

6.2. Performance measures

In the computational experiments, the run time for GAMS to solve the MILP model of each problem instance is limited by 3 hours. When the run time of the solver GAMS for obtaining the optimal solution is limited, GAMS gives one of three types of solutions for the MILP models:

(1) *Best Integer Solution* (BIS), which might be non-optimal

(2) *Optimal Solution* (OS) which is the desired

one and equals the best integer solution

(3) *No-Solution*

To assess the effectiveness of the heuristic algorithm for the problem instances having the optimal solution, TCT value obtained by the proposed heuristic algorithm is compared with TCT value of the optimal solution of the MILP model obtained by GAMS. However, for the problem instances in which the best integer solution exists but is not necessarily optimal, the TCT value obtained by the proposed heuristic algorithm is compared with the TCT value of the best integer solution.

For the problem instances having the optimal solutions obtained by solving the MILP model, the percent deviation of the TCT value provided by our heuristic algorithm from the TCT value of the optimal solution is calculated. The percent deviation from the optimal solution is $PD = 100 \times (TCT^H - TCT^O)/TCT^O$, where TCT^H is the TCT value of the solution obtained by the heuristic algorithm and TC^O is the TCT value of the optimal solution obtained by solving the MILP model. We replace TCT^O with TCT^B in PD calculation, where TCT^B is the TCT value of the best integer solution obtained by solving the MILP model when the best integer solution exists but an optimal solution is not achieved.

We measure the efficiency of our proposed heuristic algorithm using the computational time required to solve the problem instances. The computational time for the proposed heuristic algorithm is relatively minimal, less than 5 seconds, for all problem instances. Also, note that the computational time increases as the number of products or customer orders increases. Nevertheless, the computational time is minimal again, less than 5 seconds. Thus, the computational times are not reported here.

6.3. Discussion on the performance of the MILP model

This section investigates the performance of the MILP model for the setup and no-setup cases. As shown in Table 2, the MILP gives the optimal solution for every problem instance for setup and no-setup cases when the number of customer orders and the number of jobs are 5 (i.e., $K = 5$ and $N = 5$). However, when $K = 5$ and $N = 10$, the MILP finds the optimal solution for one and two problem instances for the setup and no-setup cases, respectively. No optimal solution is obtained, but the best integer solutions are achieved for setup and no-setup cases when there are five customer orders and 15 or 20 jobs. On the other hand, the MILP brings the optimal solution for four problem instances when $K = 10$ and $N = 10$ for both setup and no-setup cases. However, no optimal solution is obtained for all problems instances with 10, 15, or 20 jobs.

The quality of the best integer solutions, which is not necessarily optimal, are also investigated by examining the gap value (percent difference) between the best-integer and optimal solutions. As the number of iterations increases, integer solutions are expected to become closer to the optimal solution. However, GAMS's solver CPLEX may terminate before reaching the optimal solution because of the 3-hour time limit. Nevertheless, this case is the best since the gap values are close to zero. On the other hand, for some non-optimally solved problem instances, branching becomes very difficult and time-consuming. When branching is slow, the number of iterations is moderate, leading to higher gap values than the best case and lower gap values than the worst case. The gap values are as in Table 2. When $K = 5$ and $N = 5$ for setup and no-setup cases, all gap values equal zero, which means that MILP can achieve the optimal solution for all problem instances; furthermore, the average gap values for the no-setup case are slightly bigger than average gap values for the setup case. However, when the number of customer orders is increased from 5 to 10 for the setup case, it is observed that the average gap value of the non-optimally solved problem instances increases from 52.5 percent to 77 percent. Similarly, the average gap value of the non-optimally solved problem instances increases from 53.2 percent to 80 percent when the number of customer orders is increased from 5 to 10 for the no-setup case. These results conclude that problem complexity increases directly as customer orders or jobs increase.

Tab. 2. Performance of the MILP model

			Setup case			*No-setup case*		
K	*N*	*NPI*	*NOS*	*NBIS*	*AG*	*NOS*	*NBIS*	*AG*
5	5	25	25	0	0	25	0	0
	10	25	1	24	45	2	23	42

	15	25	0	25	78	0	25	80
	20	25	0	25	87	0	25	89
Total		100	26	74		27	73	
Average					52.5			53.2
10	5	25	4	21	38	4	21	38
	10	25	0	25	85	0	25	90
	15	25	0	25	91	0	25	94
	20	25	0	25	93	0	25	96
Total		100	4	96		4	96	
Average					77.0			80.0
Total		200	30	170		31	169	
Grand Average					64.7			66.4

Notes: *K*: number of customer orders; *N*: number of jobs; *NPI*: number of problem instances; *NOS*: number of optimally solved instances; *NBIS*: number of best integer solutions obtained only; *AG*: average gap value in percentage.

6.4. Discussion on the performance of the heuristic algorithm

This section investigates the performance of the proposed heuristic algorithm for the setup and no-setup cases. The average and maximum percent deviations from the optimal or best integer solution obtained by solving the MILP model are reported in Table 3. One can observe that average percent deviations decrease as the number of products increases from 5 to 10, 15, or 20. This situation indicates that the heuristic algorithm is powerful to provide optimal or near-optimal solutions, especially for large-scale problems. A second observation is that the grand average percent deviations for all 200 problem instances with and without setups are 0.57 and 1.24, respectively, which are low to some degree. This result indicates that the performance of the heuristic algorithm is better for the setup case.

In summary, the results above conclude that the performance of the proposed heuristic algorithm is better for large-scale problems and is more successful for the setup case than the no-setup case. Furthermore, practitioners are suggested to solve the problems with up to 10 customer orders, using the MILP model for setup and no-setup cases. For the problems with more than ten customer orders, the heuristic algorithm could be preferred when the time to obtain a solution by the MILP model is limited.

7. Conclusions

This study introduces a new customer order scheduling problem with job-based processing and lot streaming in a two-machine flow shop. The aim is to construct a sequence of the product lots and the sublots' sequence in each product lot to minimize the sum of the customer orders' completion times. We have proved that the problem under study is *NP*-hard in the strong sense. Thus, a MILP model has been presented to solve the problem optimally.

The results of our experiments indicate that the GAMS achieves optimal solutions to the MILP model for problem instances with 5 or 10 customer orders and five products in less than 3 hours of the time limit. However, no problem instance was solved optimally as the number of customer orders or products is increased. From these observations, it has been concluded that solving the MILP model cannot handle the large-scale problem instances in reasonable computational times. Thus, a multi-phase heuristic algorithm with tabu search has been developed.

Tab. 3. Performance of the heuristic algorithm

K	*N*	*NPI*	*Setup case*		*No-setup case*	
			AVE	*MAX*	*AVE*	*MAX*
5	5	25	0.36	2.59	2.26	12.64
	10	25	1.09	9.34	3.00	14.73
	15	25	0.52	3.52	1.55	8.26
	20	25	0.08	1.62	0.18	1.76

Total		100				
Average			0.51	4.27	1.75	9.35
10	5	25	1.27	7.66	0.96	7.93
	10	25	0.82	4.54	1.00	3.98
	15	25	0.44	7.11	0.50	3.71
	20	25	0.00	0.00	0.46	6.57
Total		100				
Average			0.63	4.83	0.73	5.55
Total		200				
Grand Average			0.57	4.55	1.24	7.45

Notes: K: number of customer orders; N: number of jobs; NPI: number of problem instances; AVE: average percent deviation, i.e., $\overline{PD}$; MAX: maximum percent deviation.

In our computational experiments, 200 problem instances have been generated and solved separately with and without setups. The computational experiments revealed that the proposed heuristic algorithm could optimally solve small and medium-scale problems among a total of 400 problem instances and provide near-optimal solutions in less than five seconds to large-scale instances.

On the other hand, the average percentage deviations of the total completion time of the solution obtained by the heuristic algorithm from that of the optimal or best integer solution obtained by the MILP model are 0.57 and 1.24 for all problem instances with and without setups, respectively. The associated average percentages of maximum deviations are 4.55 and 7.45 for all problem instances with and without setups, respectively. Those results altogether reveal the excellent performance of the proposed heuristic algorithm over 400 problem instances solved.

The study in this paper opens up the opportunities to do new research in the future on several extensions:

- The MILP model in Section 4 could be easily adapted to the flow shop environment having more than two machines so that more complex and realistic problems could also be studied.
- In the problem under study, intermingling the sublots of different products (jobs) is not allowed. Relaxing this no-intermingling assumption and comparing the problems with and without intermingling could be a future research issue.
- Moreover, sequence-independent setups are considered in this study. However, setups could be sequence-dependent, as in [30].
- Studying the problem under investigation in this paper for a due-date-based performance measure could be the subject of another future study.
- In addition to the above extensions, considering the job-based processing and lot streaming on different manufacturing environments, including flow shops having more than two machines (stages) and hybrid flow shops with several machines at one or more stages, could be other promising research topics.

8. Acknowledgements

We thank the referees for their constructive suggestions and comments that we have used in improving the quality of our article.

References

[1] Çetinkaya, F.C., Yeloğlu, P., Çatmakaş, H.A., "Customer order scheduling with job-based processing on a single-machine to minimize the total completion time," International Journal of Industrial Engineering Computations, Vol. 12, No. 3, (2021), pp. 271-292.

[2] Yang, J., "Customer order scheduling in a two machine flowshop," International Journal of Management Science, Vol. 17, No. 1, (2011), pp. 921-939.

[3] Jullien, F.M., Magazine, M.J., "Scheduling customer orders: an alternative production scheduling approach," Journal of Manufacturing and Operations Management, Vol. 3, (1990), pp. 177-199.

[4] Xu, X., Ma, Y., Zhou, Z., Zhao, Y., "Customer order scheduling on unrelated parallel machines to minimize total completion time," IEEE Transactions on Automation Science and Engineering, Vol.

12, No. 1, (2015), pp. 244-257.

[5] Reiter, S., "A system for managing job-shop production," The Journal of Business, Vol. 39, (1966), pp. 371-393.

[6] Vickson, R.G., Alfredsson, B.E., "Two- and three-machine flow shop scheduling problems with equal sized transfer batches," International Journal of Production Research, Vol. 30, No. 7, (1992), pp. 1551-1574.

[7] Johnson, S.M., "Optimal two- and three-stage production schedules with setup times included," Naval Research Logistics Quarterly, Vol. 1, No. 1, (1954), pp. 61-67.

[8] Çetinkaya, F.C., Kayaligil M.S., "Unit sized transfer batch scheduling with setup times," Computers and Industrial Engineering, Vol. 22, No. 2, (1992), pp. 177-182.

[9] Çetinkaya, F.C., "Lot streaming in a two-stage flow shop with set-up, processing and removal times separated," Journal of Operational Research Society, Vol. 45, No. 12, (1994), pp. 1445-1455.

[10] Vickson, R.G., "Optimal lot streaming for multiple products in a two-machine flow shop," European Journal of Operational Research, Vol. 85, No. 3, (1995), pp. 556-575.

[11] Glass, C.A., Possani, E., "Lot streaming multiple jobs in a flow shop," International Journal of Production Research, Vol. 49, No. 9, (2011), pp. 2669-2681.

[12] Sriskandarajah, C., Wagneur, E., "Lot streaming and scheduling multiple products in two-machine no-wait flow shops," IIE Transactions, Vol. 31, No. 8, (1999), 695-707.

[13] Pranzo, M., "Batch scheduling in a two-machine flow shop with limited buffer and sequence-independent setup times and removal times," European Journal of Operational Research, Vol. 153, No. 3, (2004), pp. 581-592.

[14] Çetinkaya, F.C., "Unit sized transfer batch scheduling in an automated two-machine flow-line cell with one transport agent," International Journal of the Advanced Manufacturing Technology, Vol. 29, No. 1-2, (2006), pp. 178-183.

[15] Chang, J.H., Chiu, H.N., "A comprehensive review of lot streaming," International Journal of Production Research, Vol. 43, No. 8, (2005), pp. 1515-1536.

[16] Sarin, S.C., Jaiprakash, P., Flow shop lot streaming, Springer, (2007).

[17] Cheng, M., Mukherjee, N.J., Sarin, S.C., "A review of lot streaming," International Journal of Production Research, Vol. 51, No. 23-24, (2013), pp. 7023-7046.

[18] Gomez-Gasquet, P., Segura-Andres, R., Andres-Romano, C., "A review of lot streaming in a flow shop environment with makespan criteria," Journal of Industrial Engineering and Management, Vol. 6, No. 3, (2013), pp. 761-770.

[19] Liu, C.H., "Lot streaming for customer order scheduling problem in job shop environments," International Journal of Computer Integrated Manufacturing, Vol. 22, No. 9, (2009), pp. 890-907.

[20] Gonzales, T., & Sahni, S. (1978). Flow shop and job shop scheduling: complexity and approximation. Operations Research, 26(1), 36-52.

[21] Smith, M.L., Panwalker, S.S., Dudek, R.A., "Flowshop sequencing problem with ordered processing time matrices: A general case," Management Science, Vol. 21, No. 5, (1975), pp. 44-549.

[22] Panwalker, S.S., Khan, A.W., "An ordered flow-shop sequencing problem with mean completion time criterion," International Journal of Production Research, Vol. 14, No. 5, (1976), pp. 631-635.

[23] Çetinkaya, F.C., Gupta, J.N.D., "Flowshop lot streaming to minimize total weighted

flow time," Research Memorandum No. 94-24, School of Industrial Engineering, Purdue University, West Lafayette, Indiana, (1994).

[24] Nawaz, M., Enscore, E., Ham, I., "A heuristic for the m-machine, n-job flow shop sequencing problem," Omega, Vol. 5, No. 11, (1983), pp. 91-95.

[25] Glover, F., "Future paths for integer programming and links to artificial intelligence," Journal of Computers and Operations Research, Vol. 13, No. 5, (1986), pp. 533-549.

[26] Glover, F., "Tabu-search: a tutorial," Interfaces, Vol. 20, No. 4, (1990), pp. 79-94.

[27] Mazdeh, M.M., Nakhjavani, A.K., Zareei, A., "Minimizing Total Weighted Tardiness with Drop Dead Dates in Single Machine Scheduling Problem," International Journal of Industrial Engineering & Production Research, Vol. 21, No. 2, (2010), pp. 89-95.

[28] Roshani, A., Giglio D., "A tabu search algorithm for the cost-oriented multi-manned assembly line balancing problem," International Journal of Industrial Engineering & Production Research, Vol. 31, No. 2, (2020), pp. 189-202.

[29] Çetinkaya, F.C., Çatmakaş, H.A., Görür, A.K., "Single-machine scheduling of indivisible multi-operation jobs," South African Journal of Industrial Engineering, Vol. 30, No.1, (2019), pp. 78-93.

[30] Mousavipour, S., Farughi, H, Ahmadizar, F., "A Job Shop Scheduling Problem with Sequence-Dependent Setup Times Considering Position-Based Learning Effects and Availability Constraints," International Journal of Industrial Engineering & Production Research, Vol. 30, No. 3, (2019), pp. 329-340.

Follow This Article at The Following Site:

ÇETİNKAYA F C, BORAN YOZGAT G. Customer order scheduling with job-based processing and lot streaming in a two-machine flow shop. IJIEPR. 2022; 33 (2) :1-17
URL: http://ijiepr.iust.ac.ir/article-1-1308-en.html