Operations Research Forum (2023) 4:49
https://doi.org/10.1007/s43069-023-00227-2

RESEARCH

An Order Scheduling Heuristic to Minimize the Total Collation Delays and the Makespan in High-Throughput Make-to-Order Manufacturing Systems

Husam Daoud[1] · Nieqing Cao[1] · Debiao Li[2] · Jaehee Kim[3] · Sang Won Yoon[1] · Daehan Won[1]

Received: 19 November 2022 / Accepted: 11 May 2023 / Published online: 6 June 2023

Abstract

This paper presents an order scheduling heuristic to minimize the total collation delays and the makespan in high-throughput make-to-order manufacturing systems. Order collation delay is the completion time difference between the first and the last processed items within the same order. Large order collation delays contribute to a reduced throughput, non-recoverable productivity loss, or even system deadlocks. In manufacturing systems with high throughput, this scheduling problem becomes computationally expensive to solve because the number of orders is very large; thus, efficient constructive algorithms are needed. To minimize both objectives efficiently, this paper proposes a novel workload balance with single-item orders (WBSO) heuristic while considering machine flexibility. Through a comparison with (1) the non-dominated sorting genetic algorithm II (NSGA-II), (2) priority-based longest processing rule (LPT-P), (3) priority-based least total workload rule (LTW-P), and (4) multi-item orders first rule (MIOF), the effectiveness of the proposed method is evaluated. Experimental results for different scenarios indicate that the proposed WBSO heuristic provides 33% fewer collation delays and 6% more makespan on average when compared to the NSGA-II. The proposed method can work on both small and large problem sizes, and the results also show that for large size problems, the WBSO generates 74%, 89%, and 62% fewer collation delays on average than LPT-P, LTW-P, and MIOF rules respectively.

Keywords Order scheduling · Make to order production · Parallel machines · Order collation delays

Extended author information available on the last page of the article

1 Introduction

Order scheduling is the problem of assigning customer orders to different machines. Orders might consist of multiple items and must be fully processed before they can be packaged and shipped. This problem can be found in many applications which include make-to-order production systems [1], assemble-to-order production systems [2], central fill pharmacies [3–5], and automotive repair shops [6]. In the literature, this scheduling problem has been studied to minimize a variety of objectives such as the makespan, the total completion time, and the total tardiness. This paper studies the order scheduling problem in high-throughput make-to-order manufacturing systems when the objective is to minimize the makespan and the total order collation delays.

Order collation delay is the completion time difference between the first and the last processed items within the same order [3, 4]. Large order collation delays contribute to a reduced throughput, non-recoverable productivity loss, or even system deadlocks. This research is motivated by an existing order collation delay problem in mail-order pharmacy automation systems (MOPA). MOPA is a high-throughput pharmacy fulfillment facility that receives large volumes of customized prescription orders from local pharmacies, fills them, and sends them back. The main component of the MOPA system is the robotic dispensing system (RDS), which consists of many auto-dispensers that count different medications in parallel, and a robot arm, which holds the vial during dispensing operation. Many of the prescription orders received consist of more than one medication, and each medication might be processed at a different RDS unit. Prescription orders with multiple medications have to be collated before packaging and shipping, which means that the first processed medication within an order has to wait until the remaining medications from the same order are processed. As the number of collated orders increases, the chances that the system might go into a deadlock or a crippled state increase. Another example of this problem can be found in automotive repair shops [6]. If a car has multiple failures, it may require more than one mechanic working simultaneously to repair it. The car will occupy a space in the shop until all repair processes are completed, which will prevent new cars from entering the system.

There are two options to avoid deadlocks: (1) increase order collation resources by investing in the material handling system or (2) schedule orders efficiently such that expected order collation delays are minimized. There have been several studies in the literature that solved the order collation delay scheduling problem by using exact methods or metaheuristics [3–5]. It was shown that the minimization of order collation delays as a sole objective function leads to an increased makespan [4]. In addition, it was noticed that the performance of the proposed methods degrades significantly as the problem size increases. In high-throughput manufacturing systems, thousands of orders are processed every day; thus, computationally efficient algorithms are needed. In this research, the workload balance with single-item orders (WBSO) heuristic is proposed to minimize the total order collation delays and the makespan. The proposed WBSO is compared with the non-dominated sorting genetic algorithm II (NSGA-II) and other greedy heuristics.

Machine flexibility is a critical factor that affects the performance of the order scheduling system. In the literature, machines are classified based on flexibility to three categories: (1) fully flexible, where machines are capable of processing all types of jobs; (2) fully dedicated, where machines are only capable of processing one type of job; and (3) multi-purpose or general, where machines are capable of processing certain types of jobs [7]. Fully flexible machines environment is expected to generate fewer total collation delays and makespan than other cases; however, in most of the applications, machines are either dedicated or multi-purpose. The performance of the proposed WBSO heuristic is evaluated based on a different amount of machine flexibility. In addition, different percentages of multi-item order jobs are incorporated in the scenarios.

The remainder of this paper is as follows: literature review is summarized in Sect. 2; the proposed heuristics are presented in detail in Sect. 3; results and analysis are discussed in Sect. 4; conclusions and future work are given in Sect. 5.

2 Literature Review

The order scheduling problem on parallel machines has been proven to be an NP-hard problem when the number of machines is larger than two [7–11]. Therefore, various heuristic methods have been proposed to provide near optimal solutions in minimal computational time for the three different machine environments: fully flexible, fully dedicated, and multi-purpose. For the fully flexible machine environment, the problem has been studied to minimize the total weighted completion time [12]. Several heuristics and rules have been proposed such as the longest processing time rule (LPT), weighted shortest total processing time (WSTP), and the weighted shortest LPT makespan first (WSLM) [12]. The WSTP-based heuristics provided better performance in terms of solution quality and computational time than the other proposed heuristics.

For the fully dedicated machine environment, the order scheduling problem has been studied to optimize different performance measures including the total weighted completion time [12, 13], the total tardiness [14], and the number of late jobs [9]. To minimize the total weighted completion times, several heuristics have been proposed such as the weighted shortest total processing time (WSTP), the weighted shortest maximum processing time (WSMP), and the weighted earliest completion time first (WECT) [12]. It was concluded that the WECT heuristic is preferable because it provides good results with short computational time [15]. A linear formulation of the problem has been proposed to generate optimal solution for the problem [13]. However, a problem with 200 orders and three machines takes up to 1 h of computational time. An approximation algorithm that incorporates a look-ahead mechanism has been proposed to minimize the total completion time [16]. Results showed that the proposed algorithm outperforms earliest completion time (ECT) heuristic while achieving similar computational performance.

Few studies in the literature addressed the order scheduling problem when the machines are multi-purpose. Three heuristics were proposed to minimize the total completion time of orders and the makespan while considering a job-split property in the system in [17]. It was shown that although multi-stage optimization heuristics provide better solution quality, they are computationally inefficient for solving large size problems. The problem of minimizing total tardiness of orders when machines are multi-purpose has also been studied. Multi-stage and bottleneck-focused algorithms were proposed to solve this problem in an air conditioner manufacturing company when a job-splitting property exists [18]. It was shown that the bottleneck-focused algorithms that take the processor schedule into consideration outperform multi-stage algorithms. The multi-site order scheduling problem (MSOS) is an application of this case [19]. In MSOS, which exists in companies that have multiple manufacturing plants, the company receives many production orders that need to be manufactured in different self-owned or collaborative facilities. This problem has been solved using NSGA-II to minimize three objectives: the total tardiness, the idle time, and the throughput time. Results indicated that NSGA-II provides superior results compared to the current scheduling algorithms. However, the number of orders is relatively small in this application when compared with MOPA systems.

The order scheduling problem in MOPA systems has several interesting features: (1) machines in this system are multi-purpose and their flexibility significantly affects the overall system performance [3]; (2) the number of orders that need to be scheduled is relatively high (for some systems the daily average is 45,000 orders); (3) factors such as the percentage of multi-item orders and the amount of machine flexibility are variable throughout the days and they are not similar for all MOPA systems. There have been several studies that addressed this scheduling problem in the literature. An adaptive parallel tabu search (APTS) algorithm combined with an ϵ-constraint method has been proposed to solve the problem [4]. The proposed APTS outperformed LPT and tabu search in terms of total collation delays. This problem has also been solved using multi-objective genetic algorithms [5]. It was shown that NSGA-II provides the best Pareto front and the most stable behavior especially when the number of job in the system increases. However, both previous studies have assumed that the machines are fully flexible, which means that the RDS units are able to fill all types of medications. The effect of machine flexibility has been analyzed later in [3]. It was shown that machine flexibility significantly affects the total collation delays and the makespan. It was also concluded that the computational time of the metaheuristic increases significantly as the number of jobs, the flexibility of the machines, and the percentage of multi-item orders increase. Therefore, constructive heuristics that utilize simple priority rules are needed to efficiently schedule orders in MOPA systems.

3 Research Methodology

This research extends [3] by proposing a computationally efficient constructive heuristic that minimizes the total collation delays and the makespan. Therefore, a similar notation and mathematical model are used to provide the reader with better understanding of the problem. The proposed WBSO heuristic follows the concept of job insertion to balance workload. The details of this heuristic are presented in Sect. 3.2. The proposed algorithm is compared with NSGA-II, which is discussed in Sect. 3.3, and

Table 1 List of notations

Indexes	Description
i	Order index, $i \in \{1, 2, \ldots, I\}$
j	Item index, $j \in \{1, 2, \ldots, n_i\}$
m	Machine index, $m \in \{1, 2, \ldots, K\}$
t, r	Position index, $t, r \in \{1, 2, \ldots, T\}$
Data	**Description**
a_{ijm}	1 if item j from order i can be processed on machine m, 0 otherwise
n_i	Number of items in order i
p_{ij}	Processing time of item j from order i
A	Eligibility matrix whose component is a_{ijm}
I	The number of orders
K	Number of machines
M	A very large number
T	Number of positions
α	Percentage of items that belong to single-item orders
γ	Percentage of items that belong to multi-item orders
Variables	**Description**
c_{ij}	Completion time of item j from order i
s_{ijmt}	Starting time of item j from order i in position t on machine m
x_{ijmt}	1 if item j from order i is processed in position t of machine m; 0 otherwise
$C_{\max}$	Schedule makespan
E_i	Completion time of the first processed item of order i
L_i	Completion time of the last processed item of order i
δ_i	Collation delay of order i
Δ	Total collation delays of the schedule

several greedy heuristics that include (1) longest processing time with order priorities (LPT-P), (2) least total workload with order priorities (LTW-P), and (3) multi-item orders first (MIOF). The details of the greedy heuristics are explained in Sect. 3.4. The list of notations used in the mathematical model is presented in Table 1.

3.1 Mathematical Model

A mixed integer linear programming model is proposed in [3] to solve this problem. In this mathematical model, eligible machines for each item are defined in the eligibility matrix A. If the item j from order i can be processed on machine m, then the value of the variable a_{ijm} is equal to one. Through this mechanism, the multi-purpose machine's nature is incorporated into the problem.

$$\min \quad C_{\max} \tag{1}$$

$$\min \quad \sum_{i}^{I}(L_i - E_i) \tag{2}$$

s.t.

$$\sum_{m=1}^{K}\sum_{t=1}^{T} x_{ijmt} = 1 \qquad \forall i,j \tag{3}$$

$$\sum_{i=1}^{I}\sum_{j=1}^{n_i} x_{ijmt} \le 1 \qquad \forall m,t \tag{4}$$

$$x_{ijmt} - a_{ijm} \le 0 \qquad \forall i,j,m,t \tag{5}$$

$$s_{ijm1} = 0 \qquad \forall i,j,m \tag{6}$$

$$s_{ijm(t+1)} - \sum_{i=1}^{I}\sum_{j=1}^{n_i}\sum_{r=1}^{t} p_{ij} \cdot x_{ijmr} = 0 \qquad \forall i,j,m,t < T \tag{7}$$

$$c_{ij} - (s_{ijmt} + p_{ij}) - M(1 - x_{ijmt}) \le 0 \qquad \forall i,j,m,t \tag{8}$$

$$s_{ijmt} + p_{ij} - c_{ij} - M(1 - x_{ijmt}) \le 0 \qquad \forall i,j,m,t \tag{9}$$

$$\sum_{i=1}^{I}\sum_{j=1}^{n_i}\sum_{t=1}^{T} p_{ij} \cdot x_{ijmt} - C_{\max} \le 0 \qquad \forall m \tag{10}$$

$$c_{ij} - L_i \le 0 \qquad \forall i,j \tag{11}$$

$$E_i - c_{ij} \le 0 \qquad \forall i,j \tag{12}$$

$$s_{ijmt} \ge 0 \qquad \forall i,j,m,t \tag{13}$$

$$c_{ij} \ge 0 \qquad \forall i,j \tag{14}$$

$$E_i, L_i \ge 0 \qquad \forall i \tag{15}$$

$$x_{ijmt} \in \{0,1\} \qquad \forall i,j,m,t \tag{16}$$

Equations (1) and (2) are the objective functions to minimize the schedule makespan and the total collation delays from all orders. Equations (3) and (4) ensure that each item is processed only once and that each position is filled by one item at max, while Eq. (5) guarantees that items are processed on eligible machines. Equations (6) and (7) are used to calculate items starting times, while Eqs. (8) and (9) are used to calculate items completion times. Equation 10 is used to calculate the schedule makespan, while Eqs. (11) and (12) define the earliest and latest order completion times. Equations (13–16) state the feasible region for decision variables.

3.2 Workload Balancing by Single-Item Orders (WBSO)

There are two types of heuristics to solve scheduling problems: constructive and improvement heuristics. Constructive heuristics attempt to build a schedule from scratch through adding jobs each step, while improvement heuristics start with an initial solution and then improve it through steps of schedule alteration. Although improvement heuristics, including local search methods, usually provide the decision maker with higher quality solutions, they are criticized for their large computational time. In contrast, constructive heuristics can obtain high-quality solutions in short computational time, if designed well. The complexity of this problem arises from three aspects: the order scheduling nature of the problem, job assignment restrictions, and the fact that there are multiple objectives. These three aspects make it challenging to design an effective constructive heuristic that minimizes both objectives simultaneously.

In this research, a heuristic that is based on the concept of job insertion is proposed to solve this order scheduling problem. The concept of this heuristic, which is called the workload balancing with single-item orders (WBSO), is illustrated in Fig. 1. When a multi-item order is assigned, the workload difference is calculated, and a single-item order is inserted to balance the current workload before another multi-item order is assigned.

The WBSO algorithm can be summarized in the following steps:

Step 1. Calculate the maximum workload possible per each machine EW_m

$$EW_m = \sum_{i=1}^{I} \sum_{j=1}^{n_i} a_{ijm} \cdot p_{ij} \tag{17}$$

Step 2. Choose the least flexible machine m_l, which is the machine with the minimum expected workload, $min\{EW_1, EW_2, ..., EW_K\}$

Step 3. For each order i, set order priority O_i as

$$O_i = \sum_{j=1}^{n_i} a_{ijm_l} \tag{18}$$

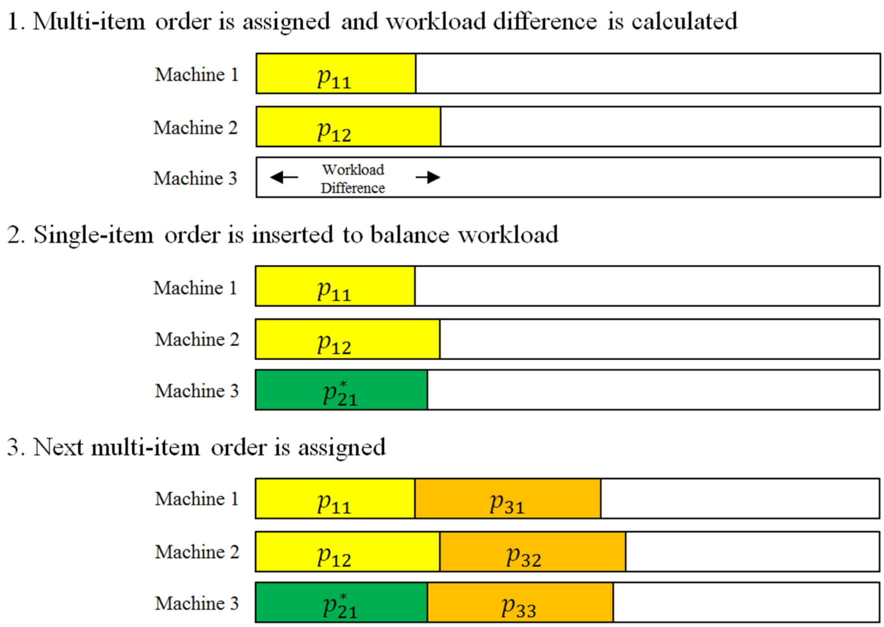


Fig. 1 Illustration of WBSO algorithm concept

Step 4. Sort the orders based on their priority O_i in descending order. Ties are broken arbitrarily.

$$O_1 \geq O_2 \geq O_3 \geq \dots \geq O_I \quad (19)$$

Step 5. Schedule the next multi-item order based on the priority list in Step (4) by following Steps (5.1−5.5):

5.1. Choose the least flexible item j_l, min $\{\sum_{m=1}^{K} a_{i1m}, \sum_{m=1}^{K} a_{i2m}, \cdots, \sum_{m=1}^{K} a_{in_i m}\}$. Ties are broken based on LPT and then arbitrarily.

5.2. Find the set of eligible machines (A_{j_l}) for item j_l

5.3. Calculate the current workload for each machine in set A_{j_l}

$$W_m = \sum_{i=1}^{I} \sum_{j=1}^{n_i} \sum_{t=1}^{T} p_{ij} \cdot x_{ijmt} \qquad \forall m \in A_{j_l} \quad (20)$$

5.4. Assign item j_l on the machine with the minimum current workload

5.5. Repeat from Step (5.1) until all the items within the order are scheduled

Step 6. Calculate the current workload for each machine W_m in the system

$$W_m = \sum_{i=1}^{I} \sum_{j=1}^{n_i} \sum_{t=1}^{T} p_{ij} \cdot x_{ijmt} \tag{21}$$

Step 7. Calculate the current largest workload difference D

$$D = \max\{W_1, W_2, ..., W_K\} - \min\{W_1, W_2, ..., W_K\} \tag{22}$$

Step 8. While $D > s$, where s is the shortest processing time in the system, do

8.1. Choose the machine with minimum current workload f
8.2. Find the set of unscheduled single-item orders that can be processed on machine f
8.3. If $f \neq \phi$, do
8.3.1. Choose a single-item order with $a_{ijf} = 1$ based on SPT
8.3.2. Schedule that single-item order on f to reduce the workload gap
8.3.3. Calculate the workload difference D

Step 9. Repeat from Step (5) until all multi-item orders are scheduled. If all multi-item orders are scheduled, go to Step (10)
Step 10. Schedule the remaining single-item orders based on O_i and the LPT rule

To illustrate the algorithm steps, consider the orders in Table 2. In this example, there are five single-item orders (1, 2, 3, 4, and 8) and three multi-item orders (5, 6, and 7). The corresponding binary values in the table represent the eligibility matrix. First, the maximum possible workload for each machine EW_m is calculated using Eq. (17). Based on these values (136, 137, 74), Machine 3 is picked as the least flexible machine (m_l); thus, the priority of orders with items that can be processed on this machine is increased. In Step 5, the next multi-item orders on the priority list are orders (5 and 7). Ties can be broken arbitrarily; thus, Order 5 is chosen to be scheduled in this step. For this order, Item 1 is assigned on Machine 3, while Item 2 can be assigned on Machine 1 or 2. At this step, because the workload of the two machines is equal, Machine 1 is chosen. In Step 6, the current workload (W_m) for each machine is calculated (14, 0, 15) and the workload difference (D) is determined (15).

Then, the workload difference is compared to a predefined value s, which can be assumed as the shortest processing time in the system (14). Because $D > s$, the machine with the minimum current workload w_{min} is picked (Machine 2), and the single-item order with shortest processing time is assigned on it (Order 1). Because the current workload on the three machines is now less than s, the next multi-item order in the list is scheduled (Order 7). After assigning all multi-item orders, single-item orders are assigned based on LPT rule. The resulting schedule for these orders is presented in Fig. 2.

The performance of the WBSO is compared to the NSGA-II and the other greedy heuristics in this research.

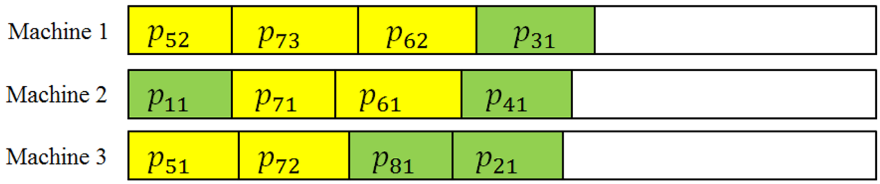


Fig. 2 Calculated schedule based on WBSO

3.3 Non-dominated Sorting Genetic Algorithm (NSGA-II)

Multi-objective optimization has received considerable attention in the literature over the past years. This is related to the fact that solutions in real-world problems are evaluated based on multiple objectives. The multi-objective optimization problem can be mathematically expressed as follows [20]:

$$\min \quad f = [f_1(x), f_2(x), \ldots, f_m(x)] \tag{23}$$

$$\text{subject to} \quad x \in \Omega \tag{24}$$

where Ω is the decision variable space. In some problems, the objectives do not conflict with each other and it is easy to find a single compromise solution for the problem; however in most of the practical problems, there are usually conflicting objectives and it is difficult to find a single optimal solution [19]. In this situation, the objective is to find the set of Pareto optimal solutions, which are the solutions that cannot be improved in any objective without degrading another objective. These solutions are also referred to as non-dominated solutions in the problem.

Methods used to solve multi-objective optimization problems can be classified into two approaches. The first approach, which is the classical approach, attempts to combine multiple objectives into a single aggregating function. Examples of

Table 2 Sample of random orders for WBSO numerical example

Order	Item	p_{ij}	a_{ij1}	a_{ij2}	a_{ij3}
1	1	14	1	1	0
2	1	15	1	0	1
3	1	16	1	1	0
4	1	15	0	1	1
5	1	15	1	1	1
5	2	14	1	1	0
6	1	17	0	1	0
6	2	16	1	0	0
7	1	14	0	1	0
7	2	15	1	1	1
7	3	17	1	1	0
8	1	14	1	0	1

this approach include the weighted sum, goal programming, min-max Pareto point, and the ε-constraint methods [20]. The second approach, by contrast, attempts to generate the Pareto front, which is the set of non-dominated solutions in the problem. Pareto-based evolutionary algorithms have been widely applied in solving multi-objective optimization problems because they utilize a population-based approach which allows them to generate multiple solutions in one simulation run. Another advantage of using multi-objective evolutionary algorithms is that they are not affected by the shape or the continuity of the Pareto front in contrast to traditional optimization techniques.

The non-dominated sorting genetic algorithm II (NSGA-II), proposed by [21], is considered one of the most effective multi-objective evolutionary algorithms. It utilizes a fast non-dominated sorting algorithm, a crowded distance estimation procedure, and a crowded comparison operator to overcome some of the problems encountered by other previous MOEAs. These problems include non-elitism and the need to specify a sharing parameter. In this section, NSGA-II will be briefly described.

The first feature of the NSGA-II is the fast sorting algorithm, which can be summarized in the following steps:

Step 1. For each solution p in the population, calculate the domination count n_p, which represents the number of individuals that dominate the solution, and create S_p, which is the set of individuals dominated by the solution. Solutions that belong to the first non-dominated front will have an n_p equal to zero.

Step 2. Initialize the front counter i to one. For each solution p that belongs the frontier F_i, visit each solution q from its set S_p and decrement its domination count n_q by one. If n_q becomes zero, put q in a separate list Q. The set Q will be used to store individuals for the $(i+1)^{th}$ front.

Step 3. Set $F_i = Q$ and repeat Step two until $Q = \phi$.

NSGA-II employs a crowding distance assignment process and crowded-comparison operator to guide the selection process to achieve better approximation of the Pareto front. The crowding distance assignment process starts by sorting the population in an ascending order, and assigning the boundary solutions (solutions with largest and smallest objective function values) an infinite distance value. Solutions in between ($i = 2$ to $i = l - 1$) are assigned distance value according the following formula:

$$d_i = d_i + \frac{d_{(i+1)}.m - d_{(i-1)}.m}{f_m^{max} - f_m^{min}} \tag{25}$$

where $d_i.m$ is the m^{th} objective function value and f_m^{max} and f_m^{min} are the maximum and the minimum m^{th} objective function values. After the individuals are sorted based on non-domination and the crowding distance for each solution is assigned, a selection process is executed by using a crowded-comparison operator. The operator will check the ranking and choose the solution with the better ranking. If rankings are equal, the solution with the largest distance value is chosen (i.e., the solution located in a less crowded region).

The chromosome used in this problem is a 2D chromosome. Each row represents a machine, and each column represents a position. Each gene is an assigned item, and the sequence of genes represents the scheduling order of these items on the machine. The main loop of NSGA-II starts by creating a random parent population P_0 by assigning items randomly to machines while taking into consideration eligibility constraints. A fast non-dominated sorting is then applied to the initial population and crowding distance is assigned to each solution. An offspring set Q_0 is then created through applying binary selection, crossover, and mutation to the initial population. The crossover operator is used to create new solutions by combining two or more parent solutions, which helps to explore the search space by combining different elements of the parent solutions. A partially mapped crossover operator is used in this GA to produce offspring. This crossover process can be summarized briefly in the following steps: (1) randomly pick a segment from Parent 1 and copy it to the offspring; (2) locate the segment in Parent 2 and find the set of elements that were not copied i; (3) find the set of elements j from the offspring that do not exist in the segment of Parent 2; (4) place elements i in the location of elements j in the offspring. This process is restricted by machine assignment constraints. The mutation operator is used to introduce small random changes into the existing solutions, which can maintain diversity in the population and help avoid premature convergence of the algorithm. Mutation strategy applied in this research is a gene-swap mutation, where two genes are swapped randomly given that eligibility constraints are not violated. Figure 3 illustrates crossover and mutation operators.

The population set P_0 and the offspring set Q_0 are combined to form R_0, and a fast non-dominated sorting is applied to it. After a crowding distance assignment process is applied, solutions are selected to fill the new population P_{t+1}. An offspring set Q_{t+1} is then created through binary selection, crossover, and mutation. If the stopping criteria are met, Q_{t+1} is returned; else, parents and offspring sets are combined and the process continues. The main loop is described as follows:

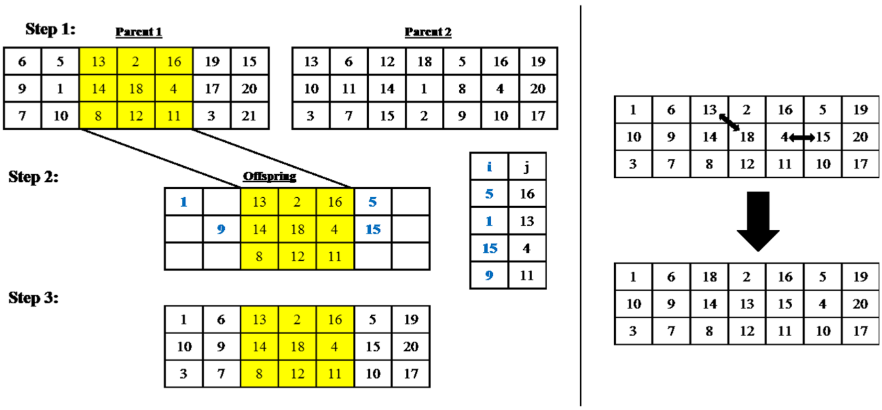


Fig. 3 GA operators: (left) crossover (right) mutation

Step 1. Initialize a random population P_0
Step 2. Apply fast non-dominated sorting to P_0 and calculate d_i for each solution i
Step 3. Use binary selection, crossover, and mutation to create offspring Q_0
Step 4. Set counter $t = 1$
Step 5. Combine P_t and Q_t to one set $R(t)$
Step 6. Apply fast non-dominated sorting to R_t and calculate d_i for each solution i
Step 7. Generate population P_{t+1}
Step 8. Terminate if stopping criteria are met and output Q_t
Step 9. Use binary selection, crossover, and mutation to create offspring Q_{t+1}
Step 10. Set $t = t + 1$ and go to Step 5

3.4 Greedy Heuristics

The proposed WBSO heuristic is evaluated against the following greedy heuristics.

3.4.1 Longest Processing Time with Order Priorities (LPT-P)

The longest processing time (LPT) heuristic, which prioritizes items based on the length of their processing time, reports optimal or near optimal makespan in parallel machine scheduling. The LPT heuristic is modified to consider the nature of the order scheduling problem studied. This modified heuristic can be summarized in the following steps:

Step 1. Pick the item with the longest processing time j_{LP}
Step 2. Assign item j_{LP} to the first available eligible machine
Step 3. Increase the priority of the remaining items in item j_{LP} order
Step 4. Assign the items that have higher priority
Step 5. Repeat until all items are assigned

3.4.2 Least Total Workload with Order Priorities (LTW-P)

The least total workload heuristic considers the scheduled and the expected unassigned workload for each machine [22]. This heuristic is also modified to consider order scheduling. Let S_m and W_m represent the current and the unassigned workloads of machine m. The modified heuristic can be summarized in the following steps:

Step 1. Find machine m with minimum $S_m + W_m$
Step 2. List all the items that can be processed on machine m
Step 3. Pick the item with the LPT and assign it to machine m
Step 4. Increase the priority of the remaining items in the order
Step 5. Assign the items that have higher priority
Step 6. Repeat until all items are assigned

3.4.3 Multi-item Orders First (MIOF)

In this heuristic, items that belong to multi-item orders will be given higher priority, and therefore will be scheduled first. This heuristic can be summarized in the following steps:

Step 1. Take a multi-item order (starting with three-item orders)
Step 2. Schedule all items within this order
Step 3. Repeat until all multi-item orders are scheduled
Step 4. Schedule single-item orders

4 Results and Analysis

Six scenarios with different problem sizes are designed to evaluate the performance of the proposed WBSO heuristic. In the first three scenarios, the amount of machine flexibility F_p is varied between 0, 50, and 100%, respectively, while the percentage of items that belong to multi-item orders γ is fixed at 50%. Machine flexibility is calculated based on the measure proposed by [22], which is illustrated in Eq. (26). In this equation, a_{jm} is a binary variable that indicates if item j can be processed on machine m, n is the number of items, and K is the number of machines. A system with $F_p = 0\%$ is a fully dedicated system, while a system with $F_p = 100\%$ is a fully flexible system. In the second three scenarios, γ is varied between 25, 50, and 75%, respectively, while F_p is fixed at 50%. Processing times are generated from a discrete random uniform distribution ($U[14, 17]$), and the number of machines analyzed is three. The number of items in multi-item orders follows a discrete random uniform distribution ($U[2, 3]$). The term "number of items" refers to the accumulated items that reach the dynamic scheduling system at the same moment, with a large-scale problem being defined as the number of accumulated items reaching 96. Instances and CPLEX results for these scenarios are obtained from [3] and provided here as a performance reference.

$$F_p = \frac{\sum_{j,m} a_{jm} - n}{n(K-1)} \cdot 100\% \tag{26}$$

It is difficult to compare the results of the WBSO and the NSGA-II, because the latter provides a set of solutions rather than a single one. To assess the convergence of both algorithms towards an ideal reference point, we employed the modified mean ideal distance measure (MMID). This metric is widely used in evaluating the performance of the NSGA-II algorithm and comparing it with other heuristics that yield single or multiple optimal solutions in the scheduling domain. MMID measure is calculated based on Eq. (27) [23]. In this equation, NPS is the number of Pareto solutions, s is the solution index, r is the objective index, f_r is the solution objective value, and z is the reference point. The reference point z is considered as $(C^*_{max}, 0)$, where C^*_{max} is the optimal makespan and 0 is the ideal value of total collation delays.

Table 3 NSGA-II operators and parameters

Population size	100
Stopping criteria	No change in solution for 40 iterations
Mutation method	2-opt swap
Mutation rate	0.01
Crossover method	Partially mapped crossover
Crossover rate	1.0
Parent selection method	Binary selection
Number of replications per instance	20

$$MMID = \frac{1}{NPS} \sum_{s=1}^{NPS} \sqrt{\sum_{r=1}^{2} (f_{sr} - z_r)^2} \tag{27}$$

The NSGA-II parameters and operators used in these experiments are presented in Table 3. These parameters have been optimized using a design of experiments approach to achieve the best performance in terms of solution quality and computational time. The results that illustrate the comparison between WBSO and the NSGA-II based on the *MMID* measure are shown in Table 4. For these experiments, F_p and γ are fixed at 50%. It can be observed that for large size problems (96, 120, and 160 items), the WBSO outperforms the NSGA-II by 33.7% on average. The experimental results also showed that the difference between MMID values and the average total collation delays is not significant. Therefore, the NSGA-II average values will be used to benchmark the WBSO performance.

The experimental results for different values of F_p are presented in Table 5.

In these experiments, γ is fixed at 50%. Figure 4 illustrates the WBSO performance in terms of total collation delays when $F_p = 50\%$. For small size problems (12, 24, and 48 items), WBSO generates larger makespan and total collation delays compared to NSGA-II. However compared to the other heuristics (LPT-P, LTW-P, MIOF), WBSO generates (63%, 90%, 65%) fewer collation delays on average. For large size problems, WBSO outperforms all other heuristics in generating fewer collation delays. When compared to the NSGA-II, WBSO generates 28% fewer collation delays and 6% more makespan on average. This indicates that when the number

Table 4 Comparison between WBSO and NSGA-II based on the MMID performance measure

Problem		NSGA-II			WBSO		
Items	C^*_{max}	C_{max}	Δ	MMID	C_{max}	Δ	MMID
12	62	**62**	**3.5**	**3.5**	74	16	20.0
24	123	**123.1**	**13.3**	**14.3**	135	18	21.6
48	245	**245.1**	**29**	**29.0**	259	35	37.7
96	493	493.1	158.1	159.8	**505**	**120**	**120.6**
120	620	621	239.5	227.7	**699**	**130**	**152.1**
160	828	828	348.5	348.9	**853**	**196**	**197.6**

Values in bold highlight the system performance under the best MMID value

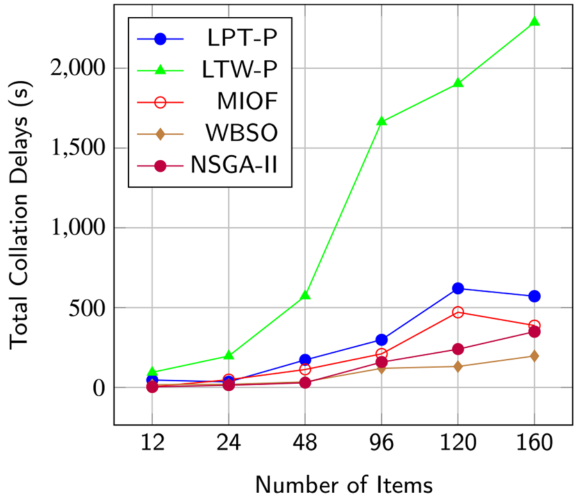
Table 5 Experimental results for different amounts of machine flexibility

Problem			**CPLEX**				**LPT-P**		**LTW-P**		**MIOF**		**WBSO**		**NSGA-II**		
F_p	Items	C^*_{max}	C_{max}	Δ	CT	Duality gap	C_{max}	Δ	C_{max}	Δ	C_{max}	Δ	C_{max}	Δ	Cmax	Δ	CT
0%	12	79	79	31	3.4	0.00%	79	50	79	53	79	45	79	35	79.0	31.0	6.7
	24	140	140	30	902.0	0.00%	140	125	140	257	140	113	140	39	140.0	31.5	9.2
	48	352	352	78	3600*	18.14%	352	413	352	607	352	218	352	100	352.0	81.9	17.7
	96	728	728	335	3600*	31.51%	728	1680	728	2093	728	1363	728	315	728.0	266.7	60.6
	120	771	771	780	7200*	50.29%	771	2846	771	3504	771	1375	771	306	771.0	283.6	76.2
	160	972	972	4533	10,800*	82.34%	972	4582	972	5009	972	1924	972	376	972.0	395.3	147.5
50%	12	62	62	3	3600*	4.62%	75	45	78	93	63	3	74	16	62.0	3.5	8.0
	24	123	123	2	3600*	1.60%	124	34	127	196	123	48	135	18	123.1	13.3	11.2
	48	245	245	49	3600*	16.67%	255	172	265	572	256	112	259	35	245.1	29.0	35.6
	96	493	502	198	3600*	29.57%	494	298	503	1664	501	209	505	120	493.1	158.1	101.3
	120	620	635	347	7200*	36.86%	636	620	623	1904	624	471	699	130	622.7	239.5	198.7
	160	828	847	3845	10,800*	85.67%	839	571	828	2289	841	387	853	196	828.0	348.5	354.6
100%	12	61	61	2	3600*	3.17%	61	59	61	59	63	3	61	5	61.0	2.6	11.2
	24	123	123	3	3600*	2.38%	124	104	124	104	123	46	123	16	123.0	11.4	16.6
	48	245	248	56	3600*	19.41%	246	207	246	207	246	69	247	20	245.0	27.4	48.5
	96	493	497	369	3600*	43.07%	495	417	495	417	499	161	493	35	493.0	133.6	203.2
	120	620	624	464	7200*	58.73%	620	528	620	528	620	184	620	49	620.1	179.4	364.3
	160	828	832	3912	10,800*	93.43%	835	716	835	716	834	259	836	47	828.0	317.3	482.8

CT computational time

*CPLEX reached time limit

Fig. 4 Algorithms performance when $F_p = 50\%, \gamma = 50\%$



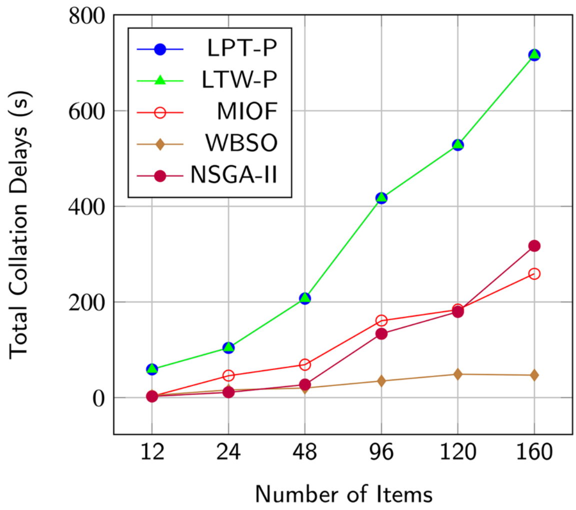
of items increases, the performance of local search methods degrades significantly. Moreover, when compared to other heuristics (LPT-P, LTW-P, MIOF), WBSO generates (68%, 94%, 55%) fewer collation delays on average. The primary reason is that the WBSO is designed to combine the features of the three scheduling rules.

Figure 5 illustrates the performance of the proposed WBSO heuristic when the system is fully flexible ($F_p = 100\%$). All the proposed heuristics generate optimal/near optimal solutions for the makespan. The WBSO heuristic generates only 0.27% more makespan compared to NSGA-II. For the total collation delays, the WBSO heuristic outperforms all other heuristics. Compared to the NSGA-II, the WBSO heuristic generates 70–85% fewer collation delays in large-size problems (96, 120, and 160 items). Compared to the other heuristics (LPT-P, LTW-P, MIOF), the WBSO heuristic generates (92%, 92%, 80%) fewer total collation delays on average in large size problems.

The performance of the proposed heuristic in the fully dedicated environment ($F_p = 0\%$) is illustrated in Fig. 6. All heuristics generate the same makespan, because each item can be processed on one machine only. For the total collation delays, the NSGA-II outperformed the WBSO heuristic in the 12, 24, 48, 96, and 120 items problems; however, both heuristics provided similar solution quality in the 160 items problem. Compared to the other heuristics (LPT-P, LTW-P, MIOF), the WBSO heuristic generates (87%, 90%, 78%) fewer total collation delays on average in large size problems.

To evaluate the performance of the proposed heuristic for different values of γ, the following scenarios are designed. In the these scenarios, γ is varied between 25%, 50%, and 75% while F_p is fixed at 50%. The obtained results are presented in Table 6. Figure 7 illustrates the WBSO performance when γ is 25%. Compared to the NSGA-II, the WBSO achieves 27% fewer collation delays on average in large size problems. Compared to the other heuristics, the WBSO heuristic generates (80%, 95%, 65%) fewer collation delays on average in large size problems.

Fig. 5 Algorithms performance when $F_p = 100\%, \gamma = 50\%$



The WBSO performance when γ is 75% is illustrated in Fig. 8. The WBSO heuristic generates (50%, 78%, 30%) fewer collation delays on average compared to the LPT-P, LTW-P, and the MIOF heuristics in the large size problems. It can also be noticed that the NSGA-II performance degrades significantly as the number of items increase.

Fig. 6 Algorithms performance when $F_p = 0\%, \gamma = 50\%$

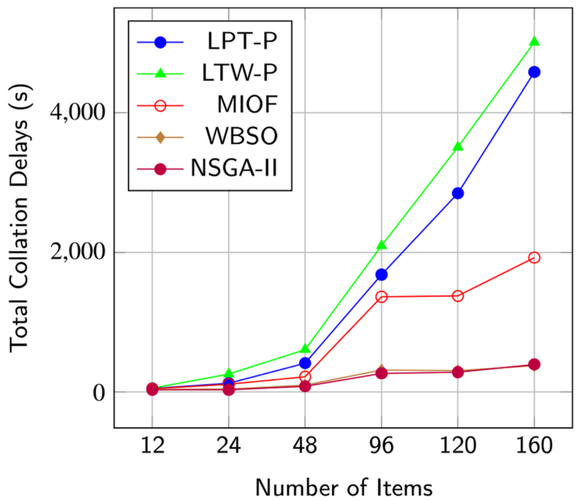


Table 6 Experimental results for different values of γ

Problem			CPLEX				LPT-P		LTW-P		MIOF		WBSO		NSGA-II		
γ	Items	C^*_{max}	C_{max}	Δ	CT	Duality gap	C_{max}	Δ	C_{max}	Δ	C_{max}	Δ	C_{max}	Δ	C_{max}	Δ	CT
25%	12	62	62	1	3600*	1.59%	75	15	78	31	62	15	75	15	62	2.0	8.1
	24	123	123	0	426.2	0.00%	124	20	126	109	136	45	135	12	123	4.6	11.5
	48	245	246	33	3600*	12.19%	246	94	261	296	247	31	261	12	245.0	19.5	22.8
	96	493	505	76	3600*	15.15%	505	186	498	1103	500	117	522	45	493.0	47.8	57.4
	120	620	633	346	7200*	36.67%	627	305	657	1747	630	246	698	80	620.7	109.2	98.1
	160	828	848	3643	10,800*	86.54%	835	276	833	1024	834	225	895	75	828	146.0	146.5
50%	12	62	62	3	3600*	4.62%	75	45	78	93	63	3	74	16	62.0	3.5	8.0
	24	123	123	2	3600*	1.60%	124	34	127	196	123	48	135	18	123.1	13.3	11.2
	48	245	245	49	3600*	16.67%	255	172	265	572	256	112	259	35	245.1	29.0	35.6
	96	493	502	198	3600*	29.57%	494	298	503	1664	501	209	505	120	493.1	158.1	101.3
	120	620	635	347	7200*	36.86%	636	620	623	1904	624	471	699	130	622.7	239.5	198.7
	160	828	847	3845	10,800*	85.67%	839	571	828	2289	841	387	853	196	828.0	348.5	354.6
75%	12	62	62	5	3600*	7.46%	74	72	78	81	62	18	62	18	62.0	8.1	8.8
	24	123	125	27	3600*	19.08%	123	125	127	297	123	72	123	23	123.0	38.9	14.9
	48	245	247	137	3600*	36.20%	264	291	255	598	260	217	248	98	245.0	114.9	42.5
	96	493	503	401	3600*	45.46%	520	655	498	1137	501	422	523	409	493.0	404.8	153.7
	120	620	635	694	7200*	53.51%	638	757	660	1654	627	837	733	269	620.8	543.7	224.8
	160	828	839	3823	10,800*	87.89%	831	837	828	1844	836	610	837	460	828.0	693.2	397.4

CT computational time

*CPLEX reached time limit

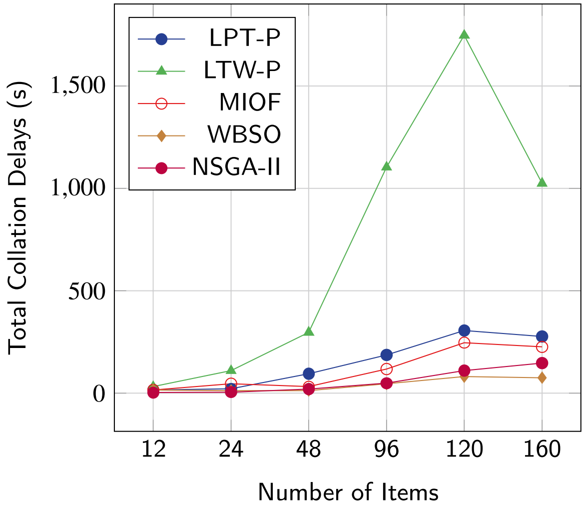


Fig. 7 Algorithms performance when $F_p = 50\%, \gamma = 25\%$

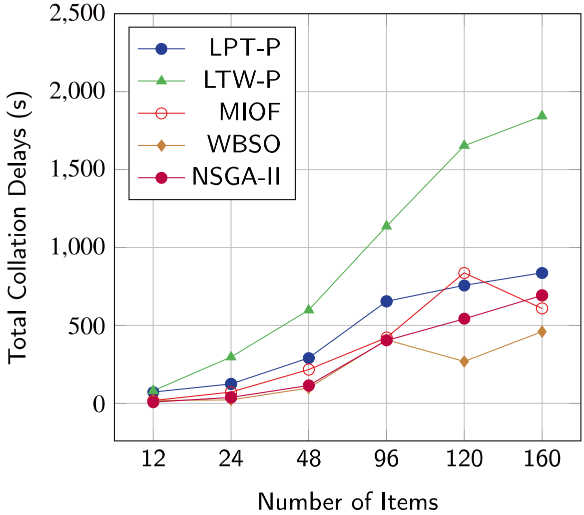


Fig. 8 Algorithms performance when $F_p = 50\%, \gamma = 75\%$

5 Conclusions and Future Work

In this research, an order scheduling heuristic is proposed to minimize the makespan and the total collation delays in high-throughput make-to-order manufacturing systems. The proposed heuristic, which is called the WBSO, is based on the concept of workload balancing by inserting single-item orders. The performance of WBSO is compared to NSGA-II, LPT-P, LTW-P, and MIOF. The experimental results show that the proposed provides 33% fewer collation delays and 6% more makespan on average when compared to the NSGA-II. The results also show that for large size problems, the WBSO generates (74%, 89%, and 62%) less collation delays on average than LPT-P, LTW-P, and MIOF rules. A major advantage of the WBSO algorithm is the ability to generate minimum total collation delays with minimum computational time (one second) for large size problems. On the contrary, the performance of meta-heuristic methods, such as the NSGA-II, degrades significantly as the problem size increases. In MOPA systems, the number of orders per day can reach 40,000 to 50,000 orders; therefore, computationally efficient algorithms, such as the WBSO, are needed.

Future work of this research would include improving the WBSO heuristic multi-objective performance and testing it on different values of processing times, levels of machine flexibility, and percentages of multi-item orders. In addition, the WBSO heuristic can be combined with local search methods to provide superior solutions. Future work would also include integrating this scheduling problem with the RDS planogram design problem [24]. In the planogram design problem, medication assignments to machines are optimized, which leads to more machine flexibility. The integrated problem will have more objectives; thus, the design of a scheduling rule will be a more challenging task.

Acknowledgements This study was supported by the Watson Institute of Systems Excellence (WISE) at Binghamton University and was partially supported by "Research Base Construction Fund Support Program" funded by Jeonbuk National University in 2022.

Author Contribution HD did the conceptualization, data curation, formal analysis, investigation, methodology development, coding, result validation, and visualization and was a major contributor in writing the original manuscript. NC did the result validation and manuscript review and editing. DL did the data curation, result validation, and manuscript review and editing. JK did the data curation, resources providing, supervision, result validation, and manuscript review and editing. SY did the methodology development, project administration, supervision, and manuscript review and editing. DW did the conceptualization, methodology development, project administration, supervision, and manuscript review and editing. All authors read and approved the final manuscript.

Data Availability The datasets analyzed during the current study are not publicly available due to the requirement of funder. Data are however available from the authors upon reasonable request and with permission of the funder.

Code Availability The codes developed during the current study are not publicly available due to the requirement of funder. The codes are however available from the authors upon reasonable request and with permission of the funder.

Declarations

Ethical Approval

1. This material is the authors' own original work, which has not been previously published elsewhere.
2. The paper is not currently being considered for publication elsewhere.
3. The paper reflects the authors' own research and analysis in a truthful and complete manner.
4. The paper properly credits the meaningful contributions of co-authors and co-researchers.
5. The results are appropriately placed in the context of prior and existing research.
6. All sources used are properly disclosed (correct citation). Literally copying of text must be indicated as such by using quotation marks and giving proper reference.
7. All authors have been personally and actively involved in substantial work leading to the paper, and will take public responsibility for its content.

Consent to Participate Not applicable.

Consent for Publication Not applicable.

Competing Interests The authors declare no competing interests.

References

1. Julien F, Magazine M (1990) Scheduling customer orders: an alternative production scheduling approach. Journal of Manufacturing and Operations Management 3(3):177–199
2. Wang B, Guan Z, Chen Y et al (2013) An assemble-to-order production planning with the integration of order scheduling and mixed-model sequencing. Front Mech Eng 8(2):137–145
3. Dauod H, Li D, Yoon SW et al (2018) Multi-objective optimization of the order scheduling problem in mail-order pharmacy automation systems. Int J Adv Manuf Technol 99(1):73–83
4. Li D, Yoon SW (2015) A novel fill-time window minimisation problem and adaptive parallel Tabu search algorithm in mail-order pharmacy automation system. Int J Prod Res 53(14):4189–4205
5. Mei K, Li D, Yoon SW et al (2016) Multi-objective optimization of collation delay and makespan in mail-order pharmacy automated distribution system. Int J Adv Manuf Technol 83(1):475–488
6. Yang J (2005) The complexity of customer order scheduling problems on parallel machines. Comput Oper Res 32(7):1921–1939
7. Leung JY, Li H, Pinedo M (2005) Order scheduling models: an overview. Multidisciplinary Scheduling: Theory and Applications. pp 37–53
8. Alharkan I, Saleh M, Ghaleb MA et al (2020) Tabu search and particle swarm optimization algorithms for two identical parallel machines scheduling problem with a single server. J King Saud Univ Eng Sci 32(5):330–338
9. Lin BM, Kononov AV (2007) Customer order scheduling to minimize the number of late jobs. Eur J Oper Res 183(2):944–948
10. Meng L, Zhang C, Shao X et al (2019) Mathematical modelling and optimisation of energy-conscious hybrid flow shop scheduling problem with unrelated parallel machines. Int J Prod Res 57(4):1119–1145
11. Sung CS, Yoon SH (1998) Minimizing total weighted completion time at a pre-assembly stage composed of two feeding machines. Int J Prod Econ 54(3):247–255
12. Leung JYT, Li H, Pinedo M (2008) Scheduling orders on either dedicated or flexible machines in parallel to minimize total weighted completion time. Ann Oper Res 159(1):107–123
13. Shi Z, Wang L, Liu P et al (2015) Minimizing completion time for order scheduling: formulation and heuristic algorithm. IEEE Trans Autom Sci Eng 14(4):1558–1569
14. Lee IS (2013) Minimizing total tardiness for the order scheduling problem. Int J Prod Econ 144(1):128–134
15. Leung JYT, Li H, Pinedo M (2007) Scheduling orders for multiple product types to minimize total weighted completion time. Discret Appl Math 155(8):945–970
16. Framinan JM, Perez-Gonzalez P (2017) New approximate algorithms for the customer order scheduling problem with total completion time objective. Comput Oper Res 78:181–192
17. Xu X, Ma Y, Zhou Z et al (2013) Customer order scheduling on unrelated parallel machines to minimize total completion time. IEEE Trans Autom Sci Eng 12(1):244–257
18. Choi YC, Kim YD, Bang JY (2010) Scheduling algorithms for an air conditioner manufacturing system composed of multiple parallel assembly lines. Int J Adv Manuf Technol 51(9):1225–1241
19. Guo Z, Wong WK, Li Z et al (2013) Modeling and pareto optimization of multi-objective order scheduling problems in production planning. Comput Oper Res 64(4):972–986
20. Belegundu AD, Chandrupatla TR (2019) Optimization concepts and applications in engineering. Cambridge University Press
21. Deb K, Pratap A, Agarwal S et al (2002) A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Trans Evol Comput 6(2):182–197
22. Vairaktarakis GL, Cai X (2003) The value of processing flexibility in multipurpose machines. IIE Trans 35(8):763–774

23. Ahmadi E, Zandieh M, Farrokh M et al (2016) A multi objective optimization approach for flexible job shop scheduling problem under random machine breakdown by evolutionary algorithms. Comput Oper Res 73:56–66
24. Khader N, Lashier A, Yoon SW (2016) Pharmacy robotic dispensing and planogram analysis using association rule mining with prescription data. Expert Syst Appl 57:296–310

Authors and Affiliations

Husam Dauod[1] · Nieqing Cao[1] · Debiao Li[2] · Jaehee Kim[3] · Sang Won Yoon[1] · Daehan Won[1]

✉ Daehan Won
dhwon@binghamton.edu

1 Department of Systems Science and Industrial Engineering, State University of New York at Binghamton, 4400 Vestal Parkway East, Binghamton 13902, NY, USA

2 Department of Management Science and Engineering, Fuzhou University, Fuzhou, Fujian, China

3 Department of Business Administration, Jeonbuk National University, Jeonju, Jeonbuk, South Korea