Considering Stockers in Reentrant Hybrid Flow Shop Scheduling with Limited Buffer Capacity

Chun-Cheng Lin[1], Wan-Yu Liu[2,3,*], Yu-Hsiang Chen[1]

[1]*Department of Industrial Engineering and Management, National Chiao Tung University, Hsinchu 300, Taiwan*

[2]*Department of Forestry, National Chung Hsing University, Taichung 402, Taiwan*

[3]*Innovation and Development Center of Sustainable Agriculture, National Chung Hsing University, Taichung 402, Taiwan*

Abstract

Diversification of products has increased the involvement of reentrant manufacturing processes, in which a job returns multiple times to a machine at the preceding workflow stage to continue the manufacturing process. Reentrant flow shop manufacturing can substantially improve manufacturing efficiency when scheduled properly. In practice, advanced manufacturing companies (e.g., semiconductor foundries) have introduced automated material handling system (AMHS), including stockers that serve as centralized inventory buffer space for temporarily storing the inventories owing to limited buffer capacity of each machine. However, no previous studies on reentrant flow shop scheduling have considered the impact of limited buffer capacity or stockers on scheduling efficiency. Consequently, this study investigated the application of stockers in solving the reentrant hybrid flow shop scheduling problem with limited buffer capacity.

* Corresponding author. Tel.: +886-4-22850158.
E-mail address: wyliu@nchu.edu.tw (Wan-Yu Liu)

With the objective of optimizing the makespan and mean flowtime of a schedule, this problem is NP-hard because it generalizes the flow shop problem. Therefore, this study developed a hybrid harmony search and genetic algorithm (HHSGA) for the problem, in which limited buffer capacity and stockers cause solution decoding to be non-trivial. Experimental comparison on scheduling problems with different numbers of jobs showed that the HHSGA performed better than conventional algorithms. Moreover, among three manufacturing conditions (i.e., with buffers and stockers, with buffers only, and with stockers only), the results indicated that the condition using inventory buffers and stockers was more beneficial.

Keywords: Flow shop scheduling, stocker, reentrant, harmony search algorithm, genetic algorithm

1. Introduction

Rapid advancements in manufacturing industries have resulted in complex production scheduling processes. Thus, completing all job productions in a minimum time has been the major objective in production scheduling. Factories in the manufacturing industry mostly employ either flow shop or job shop production processes. This study focused on examining flow shop production. Presently, the maturation of manufacturing technologies has led to complex internal structures in factories. To increase productivity, factories use *hybrid flow shop process* by incorporating parallel machines with single-line production processes (Marichelvam et al., 2014), as shown in Fig. 1(a) in which the process of transforming jobs into finished goods (FG) through four stations; each station has *M*

parallel machines, and each machine has an inventory buffer for works-in-process (WIPs) waiting to be routed to the machine. Under this circumstance, a scheduling problem is mainly concerned with 1) job permutation and 2) assignment of each job to a parallel machine at each station. That is, the permutation of all jobs must be properly determined; and according to this job permutation, each job passes through a sequence of stations, and is processed by the assigned parallel machine at each station. This scheduling problem is NP-hard (Hinze & Sackmann, 2016).

In the semiconductor industry, some advanced manufacturing processes are cyclic, thus giving reentrant characteristics to flow shop manufacturing processes (Fig. 1(b)). Reentrant characteristics refer to that jobs return to machines at the preceding workflow stage for subsequent processing, leading to complex production processes and considerable increases in the quantity of WIPs waiting for processing. Hence, recent studies on flow shop scheduling problems have focused on examining *reentrant flow shop processes*. A combination of parallel machines and reentrant processes constitutes a more complex process, called *reentrant hybrid flow shop process* (Fig. 1(c)). Performance indicators for evaluating this scheduling problem include the makespan, mean flowtime, and total tardiness of the manufacturing process.

From the literature, reentrant flow shop processes can be divided into two types. The first type is a reentrant flow shop process in which no parallel machines are employed (Lin et al., 2013a). The example given in Fig. 1(b) shows a process in which the job is routed three times to Stations 1 and 2. The scheduling problem in this process involves determining the sequence of jobs assigned to the machines at each station and is NP-hard (Wang et al., 1997).

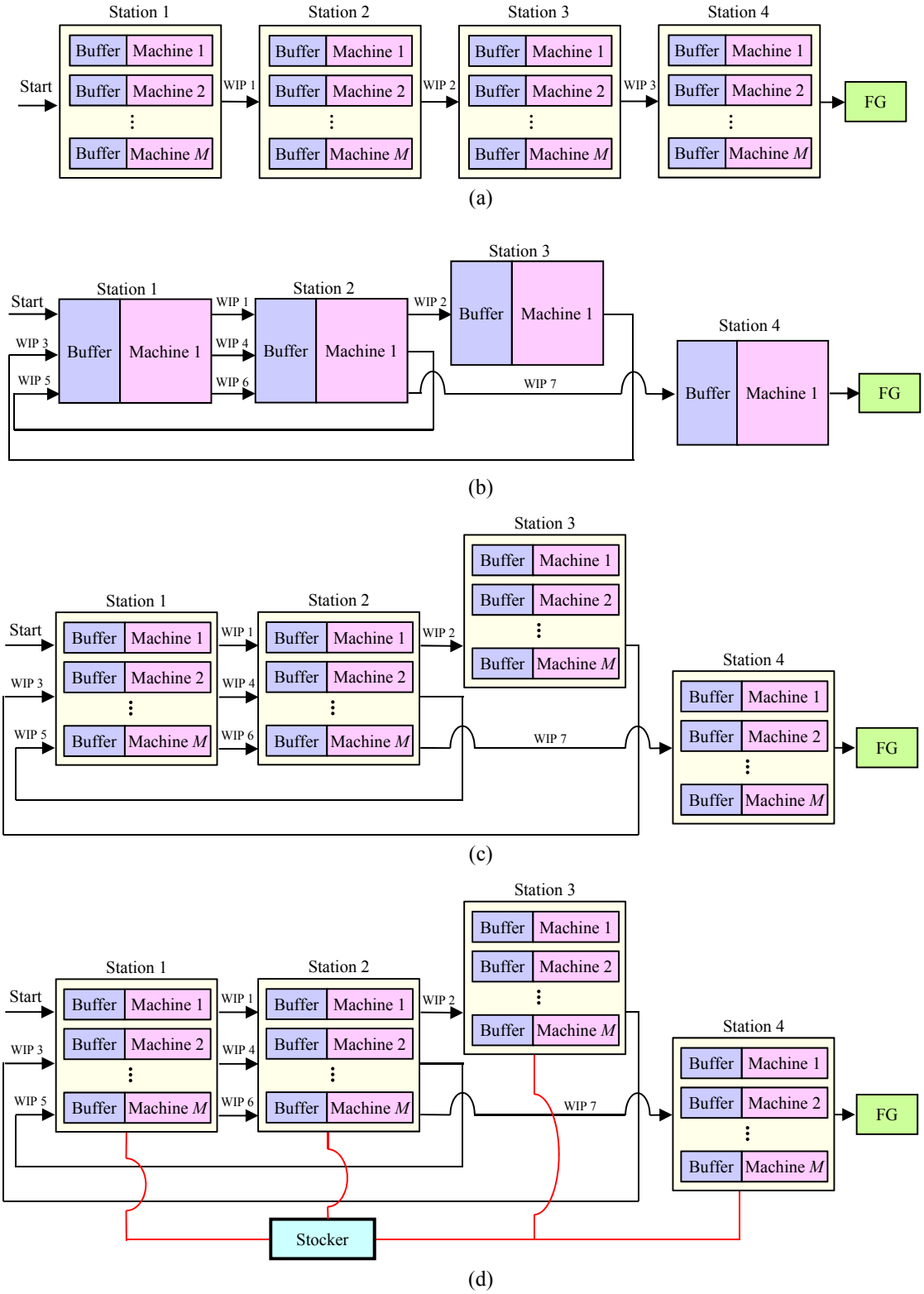


Fig. 1. Four types of production processes. (a) Hybrid flow shop. (b) Reentrant flow shop. (c) Reentrant hybrid flow shop. (d) Reentrant hybrid flow shop with a stocker.

The second type is a reentrant hybrid flow shop process, in which parallel machines are employed (EI-Khouly et al., 2009). Fig. 1(c) shows an example of this process, in which jobs are routed three times to Stations 1 and 2; each time, the job is processed by only one machine at each station. The scheduling problem in this process involves assigning parallel machines to jobs and determining all machine–job permutations.

Because these two subproblems are NP-hard (Hoogeveen et al., 1996; Wang et al., 1997; the overall scheduling problem can be inferred to be NP-hard. To solve this problem, previous studies have employed metaheuristic algorithms such as genetic algorithm (GA) (e.g., Dugardin et al., 2009; Huang & Fujimura, 2013; Lin et al., 2013a), memetic algorithm (e.g., Xu et al., 2014), harmony search algorithm (HSA) (e.g., Shen et al., 2015), and teaching-learning-based algorithms (e.g., Shen et al., 2016).

Most past studies on flow shop scheduling problems typically had only one objective of minimizing the makespan (i.e., the time from the commencement of production until the last job is completed) and have rarely considered the impact of other factors on production (EI-Khouly et al., 2009; Chamnanlor et al., 2014; Lin et al., 2013a). To simplify scheduling problems, they have generally assumed that the capacity of inventory buffers at stations is unlimited, and they have overlooked the transfer time of WIPs between stations when evaluating process efficiency (Marichelvam et al., 2014; Lin et al., 2013a). However, the transfer time between stations is a crucial consideration when reentrant characteristics in production processes are concerned. Failure to consider this factor may result in the overestimation of scheduling performance. For instance, the semiconductor industry mostly employs hybrid flow shop processes, which feature reentrant characteristics (e.g., Jing et al., 2008) and a limited capacity of inventory buffers

(e.g., Almeder & Hartl, 2013) at the stations. During production, multiple WIPs are transported among stations for processing, and the transfer time for these WIPs should be included in the makespan.

To reflect a practical factory environment, this study assumed that each inventory buffer at stations have a limited capacity. However, this assumption introduces the problem of insufficient space for WIP storage; when there is a delay in routing WIPs, they remain at the original machines instead of being stored in the inventory buffers for subsequent processing, which results in severe production stagnation. Reentrant hybrid flow shop processes in particular involve many WIPs, and these are stored in inventory buffers at station machines when awaiting further processing. In real-world practice, advanced manufacturing companies have introduced automated material handling system (AMHS), which comprises numerous subsystems such as transport, control, storage, and retrieval subsystems (Lau & Woo, 2008). In the semiconductor industry, AMHSs can further be divided into intrabay systems and interbay systems. Intrabay AMHSs route jobs between machines at a single station, whereas interbay AMHSs route jobs between stations (Kuo et al., 2007). A stocker can be employed as a link between intrabay and interbay systems (Lin et al., 2013b). In an AMHS, a stocker serves as a temporary storage area for WIPs and can be divided into a dependent or independent configurations (Agrawal & Heragu, 2006). Under a dependent configuration, each station has a stocker for storing WIPs, which are routed by the stocker to the next stocker at another station. Under an independent configuration, the stocker temporarily stores WIPs from various stations and routes them to another station for subsequent processing. In this study, a stocker with an independent configuration was employed for temporarily storing WIPs

(Fig. 1(d)).

Previous studies on stocker applications mainly discussed the optimization of job routing or vehicle dispatching (e.g., Huang et al., 2012; Lin et al., 2013b; Wu et al., 2011). Hsieh et al. (2012) proposed a segmentation strategy in which the job route was divided into various segments to avoid unnecessary routing, reduce transfer time, and increase the stocker service rate. Lin and Huang (2012) suggested that vehicles can be distributed in advance to reduce the time spent on waiting for jobs at a stocker or machine to be transported to the next stop and minimize the time of distributing these jobs without affecting the total flowtime and production volume. In contrast to previous flow shop scheduling studies, this study incorporated a stocker in a factory environment to solve the problem when WIPs storage exceeded the inventory buffer capacity. Although stockers can be used to temporarily store the WIPs from all stations to solve production stagnation, this study assumed that transferring WIPs between stations was never routed to the stocker; and WIPs were to be routed to the stocker only when the quantity of WIPs exceeded the inventory buffer capacity, after which they would be routed to another station for further processing.

To reflect a realistic factory environment, this study employed a stocker and assumed that the transfer time between stations and those from stations to the stocker are known. Note that the dispatching issues in the AMHS were not considered. Makespan and mean flowtime were employed as two indices for evaluating the performance of the production scheduling system. Specifically, the makespan was used to measure the facility utilization rate, and the mean flowtime was used to measure the quantity of WIPs in each production line. Generalizing the aforementioned scheduling problems, the scheduling problems

under this manufacturing environment are also NP-hard, and hence it is suitable to solve such problems using metaheuristic algorithms. Because HSA has been shown to be more effective than GA for independently optimizing each problem parameter (Pan et al., 2011b; Shen et al., 2015; Wang et al., 2011), this study integrated the two algorithms into the proposed algorithm to solve the concerned problem.

The rest of this study is organized as follows. Section 2 provides the literature review on previous related studies. Section 3 describes the concerned problem. Section 4 provides details of the proposed HHSGA for the problem. Section 5 shows implementation of the proposed HHSGA and experimental analysis. Section 6 conclude this study with future work.

2. Literature review

In production management, production scheduling is crucial in effectively distributing jobs, completing manufacturing procedures on time, and ensuring the timely delivery of finished goods. Numerous studies have examined the dispatching problems in production scheduling. This study introduces the internal structure of factory production environments, scheduling problems encountered in production environments, and elaborates on the approaches reported by the previous studies.

This study examined reentrant hybrid flow shop scheduling problems. Fig. 1(c) illustrates the structure of the manufacturing process, in which each station has multiple parallel machines and jobs are processed by only one machine. Therefore, the examined scheduling problems can be divided into scheduling problems associated with parallel

machines and those associated with reentrant processes. Parallel machines can be separated into related and unrelated parallel machines. Related parallel machines can be further divided into *identical* and *uniform* parallel machines. Identical parallel machines are a configuration where all machines spend the same amount of time performing the same process for the same type of job. For uniform parallel machines, the machines at a single station exhibit a proportional relationship in performing the same processing stage on the same type of job. When unrelated parallel machines are used, the machines at the same station do not exhibit any proportional relationship in performing the same processing stage on the same type of job. A reentrant process means that a job is routed multiple times to the same station for various processing tasks. In the example shown in Fig. 1(c), each job is routed to one of four stations for eight processing stages in the order of 1→2→3→1→2→1→2→4, including two reentrant processing stages.

In research on hybrid flow shop scheduling problems (i.e., with parallel machines), previous studies have proposed numerous metaheuristic algorithms, including ant colony optimization (ACO) (Alaykýran et al., 2007) and GA (Liu et al., 2008), which were employed to minimize the makespan; and particle swarm optimization (PSO), which was used to minimize the maximal flowtime (Qiao & Sun, 2011) and the total tardiness (Han et al., 2012). Mohammadi and Sahraeian (2012) proposed simulated annealing (SA) to solve single-objective hybrid flow shop scheduling problems for an unrelated parallel machine configuration. Almeder and Hartl (2013) supposed the capacity of inventory buffers at stations to be limited, and proposed a variable neighborhood search (VNS) algorithm to solve the multiobjective scheduling problems with related uniform parallel machines. Marichelvam et al. (2014) proposed a discrete firefly algorithm to solve

multiobjective flow shop scheduling problems with related uniform parallel machines. Different from metaheuristic approaches, Cheng et al. (2016) investigated the lot-streaming problem in a two-stage hybrid flow shop consisting of a machine at Stage 1 and two parallel identical machines at Stage 2. They first established a mixed-integer linear programming model, then derived closed-form expressions for finding the optimal makespan and continuous sublot sizes (supposing that the number of sublots is known a priori), and then proposed a heuristic approach for the problem with integer sublot sizes.

A large number of recent works investigated different versions of hybrid flow shop scheduling problems, and proposed a variety of methodology to address them. Lei and Zheng (2017) proposed a novel neighborhood search method with global exchange to solve a hybrid flow shop scheduling problem with assembly operations. Yu et al. (2018) proposed a GA for the hybrid flow shop scheduling problem that considers unrelated machines and machine eligibility, with the objective of minimizing the total tardiness. Shahvari Logendran (2018) modeled the hybrid flow shop batch scheduling problem with sequence- and machine-dependent family setup times, with the objective of simultaneously minimize the weighted sum of the weighted makespan and total weighted tardiness. They further proposed a tabu search and PSO to address the problem. Dios et al. (2018) proposed the hybrid flow shop scheduling problem with missing operations, and further solved the problem by four heuristics (modified from NEH and Rajendran heuristics). Fernandez-Viagas et al. (2019) analyzed the performance of different solution representations adopted in the previous works for the classic hybrid flow shop scheduling problem. Li et al. (2019) proposed a two-level imperialist competitive algorithm for the hybrid flow shop scheduling problem that considers total tardiness, makespan, and total

energy consumption.

Numerous scholars have examined reentrant flow shop scheduling problems without parallel machines. Jing et al. (2008) proposed a heuristic algorithm that minimizes the makespan to solve reentrant scheduling problems in which the flow process required two machines for repetitive reentrant processing (1→2→1→2→…→1→2). Chen and Pan (2006) proposed using integer programming to solve reentrant scheduling problems for a flow process requiring multiple machines with repetitive reentrant processing (1→2→…→M→1→2→…→M→…→1→2→…→M). Notably, reentrant characteristics have also been considered in job shop scheduling problems (e.g., Yura, 1999). To further solve reentrant hybrid scheduling problems, Xu et al. (2014) developed a memetic algorithm to minimize the makespan. Hinze and Sackmann (2016) proposed an iterated local search to minimize the makespan and total flow time. Rifai et al. (2016) built a distributed reentrant flow shop scheduling model, and proposed a multi-objective adaptive large neighborhood search algorithm. Qian et al. (2017) established operation-based and graph model for multiple-machine reentrant permutation flow-shop scheduling problem. Moreover, Lin et al. (2013a) incorporated the problem of limited resources and considered factors such as work time and material consumption to minimize the makespan by using a multilevel GA.

Because both the two types of scheduling problems above have been shown to NP-hard (Hinze & Sackmann, 2016; Wang et al., 1997), reentrant hybrid flow-shop scheduling problems that incorporate these two problems are also NP-hard. In the context of reentrant hybrid flow shop scheduling problems between two stations, Dugardin et al. (2009) proposed an improved GA for solving multiobjective problems (i.e., minimizing

the makespan and total tardiness). For reentrant flow shop scheduling problems with three stations, EI-Khouly et al. (2009) proposed a system modeling and simulation analysis method for minimizing the makespan. To solve reentrant flow shop scheduling problems with four stations, Yalaoui et al. (2009) developed a heuristic algorithm for solving the single-objective problem of minimizing the total tardiness. Other studies investigating reentrant flow shop scheduling problems between multiple stations have incorporated the analytic hierarchy process (Lin et al., 2013a) and fuzzy theory (Huang & Fujimura, 2013) with GAs in order to solve the single-objective problem of minimizing the total tardiness. In addition, Shen et al. (2015) proposed an improved HSA and Zhang and Chen (2016) proposed heuristics to solve reentrant hybrid flow shop scheduling problems where the number of production stages and uniform parallel machines varies. Chou et al. (2014) proposed a hybrid GA to solve the multiobjective TFT-LCD module assembly scheduling problem. To address reentrant flow shop scheduling problems for manufacturing solid-state disks, Chamnanlor et al. (2014) proposed a hybrid GA to solve the single-objective problem of minimizing the makespan. Cho and Jeong (2017) considered a two-level method of production planning and scheduling of reentrant hybrid flow shops, and proposed Minkowski distance-based Pareto GA with local search strategy (MLPGA) and non-dominated sorting GA (NSGA-II) for the scheduling problem. Zhang and Chen (2017) investigated a reentrant hybrid flow shop scheduling problem with machine eligibility constraints, and proposed a discrete differential evolution (DDE) algorithm with a modified crossover operator for solving the problem.

In past studies, HSAs have rarely been adopted to solve reentrant hybrid flow shop scheduling problems. However, numerous scholars have proposed improved HSAs that

have been shown to be effective than GA, PSO, ACO, SA, and tabu search algorithms in generating global optimums for flow shop scheduling problems (Pan et al., 2011b; Wang et al., 2011), and job shop scheduling problems (Yuan et al., 2013). Therefore, this study adopted a conventional HSA as a foundation for solving reentrant hybrid flow shop scheduling problems and incorporated a GA with two-point crossover for generating new solutions. Table 1 lists comparison of some key results of related studies, most of which considered only single-objective problems and made conditional assumptions pertaining to flow shop production systems.

Therefore, to reflect a realistic manufacturing environment and solve a two-objective problem (i.e., minimizing the makespan and mean flowtime), this study adjusted the conditional assumptions for the investigated production system. Moreover, this study assumed that the inventory buffers for WIPs have a limited storage capacity and that the transfer time between stations is known. Because of the limited storage capacity, the storage problem must be solved to avoid severe production stagnation. According, a stocker was applied to a typical reentrant hybrid flow shop production environment.

Table 1. Comparison of this study with previous studies.

Reference	Reentrant	Parallel machines	Performance measure	Buffer capacity	Stocker	Method
Tseng & Lin, 2010	No	No	Makespan	Unlimited	No	Hybrid GA and local search
Alaykýran et al., 2007	No	Yes	Makespan	Unlimited	No	ACO
Pan et al., 2011a	No.	No	Earliness and tardiness	Unlimited	No.	Artificial bee colony (ABC)
Pan et al., 2011b	No.	No	Makespan	Limited	No.	Chaotic HSA
Almeder & Hartl, 2013	No	Yes	Buffer utilization and makespan	Limited	No	VNS
Marichelvam et al., 2014	No	Yes	Makespan and mean flowtime	Unlimited	No	Firefly algorithm
Li & Pan, 2015	No	Yes	Makespan	Limited	No	Hybrid ABC and tabu search
Jing et al., 2008	Yes	No	Makespan	Unlimited	No	Heuristic
Lin et al., 2013a	Yes	No	Makespan	Unlimited	No	Multilevel GA
Chamnanlor et al., 2014	Yes	Yes	Makespan	Unlimited	No	Hybrid GA
Shen et al., 2015	Yes	Yes	Makespan and total tardiness	Unlimited	No	Improved HSA
Cheng et al., 2016	No	Yes	Makspan	Unlimited	No	Close-form expressions and a heuristic
Zhang & Chen, 2016	Yes	Yes	Total tardiness	Unlimited	No	Greedy and NEH
Zhang & Chen, 2017	Yes	Yes	Total tardiness	Unlimited	No	DDE
Cho & Jeong, 2017	Yes	Yes	Makespan and total tardiness	Unlimited	No	MLPGA, NSGA-II
Lei & Zheng, 2017	No	Yes	Total tardiness, maximum tardiness, and makespan	Unlimited	No	Neighborhood search with global exchange
Yu et al., 2018	No	Yes	Total tardiness	Unlimited	No	GA
Shahvari & Logendran, 2018	No	Yes	Weighted makspan and total tardiness	Unlimited	No	Tabu search and PSO
Dios et al., 2018	No	Yes	Makespan	Unlimited	No	NEH and Rajendran heuristic
Fernandez-Viagas et al., 2019	No	Yes	Makespan	Unlimited	No	22 approaches
Li et al., 2019	No	Yes	Total tardiness, makespan, and total energy consumption	Unlimited	No	Two-level imperialist competitive algorithm
This study	Yes	Yes	Makespan and mean flowtime	Limited	Yes	HHSGA

3. Problem Description

All notations adopted in the problem concerned in this study are given in Table 2.

Table 2. Notation used in this study.

Parameter	Definition
N	Number of jobs
G	Number of processing stages
S	Number of all stations
M	Number of parallel machines in each station
B	Maximal inventory buffer capacity
Index	**Definition**
i	Job index (i.e., $i \in \{1, 2, \ldots, N\}$)
g	Stage index (i.e., $g \in \{1, 2, \ldots, G\}$)
s	Station index (i.e., $s \in \{1, 2, \ldots, S\}$)
m	Machine index (i.e., $m \in \{1, 2, \ldots, M\}$)
Variable	**Definition**
R_i	Start time of releasing raw materials into the operations for job i
S_{ig}	Start time for stage g of job i
C_{ig}	Completion time of stage g of job i
P_{ig}	The time period spent on processing stage g of job i
T_{ps}	The time period spent on transferring a job from station p to station s
τ_s	The time period spent on transferring a job from the machine at station s to the stocker, which is equal to that from the stocker to the machine at station s
$C_{\max}$	Makespan
$\overline{f}$	Mean flowtime
Variable used in algorithms	**Definition**
B_{sm}	Set of the jobs that require being processed by machine m of station s but are not completed yet (including the jobs at the machine, the jobs waiting in the buffer, and the jobs moved to the stocker)
F_{sm}	The completion time of the latest job in machine m of station s

This study considers a reentrant hybrid flow shop with S stations (e.g., $S = 4$ in Fig. 1(d)) each having M parallel machines (owing to "hybrid") in which each job must be processed for a certain time period on the stations in the same order (owing to "flow shop"); and a job could visit a station more than once (owing to "reentrant"). Because of the "reentrant" feature, this study employs the term "stage" to record the order in which each job enters different stations. For example, in Fig. 1(d), each job must be processed in

the order of Stations 1, 2, 3, 1, 2, 1, 2, and 4. Hence, each job in this example has 8 stages to be processed (i.e., G = 8 in Fig. 1(d)). That is, each job at Stages 1, 4, and 6 is processed at Station 1; each job at Stages 2, 5, and 7 is processed at Station 2; and so on.

Different from previous works, this study additionally considers stockers and the limited buffer capacity in the reentrant hybrid flow shop. Each machine is associated with a buffer with a limited capacity B (see also Fig. 1(d)). Hence, when a job is assigned to a machine whose buffer is fully occupied, this job is transported to a stocker for temporary storage. The scheduling problem of G stages of N jobs in the reentrant hybrid flow shop has two groups of decision variables: the sequence of G stages of N jobs to be processed; and one of M parallel machine assigned to each stage g of each job i. Supposing that all the decision variables have been determined, a schedule for processing the N jobs is obtained. The problem concerned in this study considers following objective:

$$\text{Minimize} \qquad \omega_1 \cdot C_{\max} + \omega_2 \cdot \overline{f} = \omega_1 \cdot \max_{i \in \{1,..,N\}} \{C_{iG}\} + \omega_2 \cdot \frac{\sum_{i=1}^{N} (C_{iG} - R_i)}{N} \qquad (1)$$

The above objective represents minimizing the makespan $C_{\max}$ and the mean flowtime $\overline{f}$, which are assigned weighted values ω_1 and ω_1, respectively, ranging between 0 and 1 so that $\omega_1 + \omega_1 = 1$. Makespan C_{max} of the schedule is the maximal time among the completion time C_{iG} for each job i at the last stage G. The mean flowtime $\overline{f}$ of each job is the mean of the time period spent from releasing materials for each job i (R_i) to completion (C_{iG}).

The following assumptions are considered in this study:

- The inventory buffer for each machine m at each station s has a limited storage

capacity B.

- Each machine can process only one job at a time.
- Each job can be processed by only one machine at a time.
- Each job can be routed many times to the same station, and each time is called a stage.
- The processing time P_{ig} for each job i at each stage g is known.
- The transfer time T_{ps} between any two stations p and s is known.
- The transfer time τ_s between the stocker and each station s is known.
- The setup time for machine is not considered.
- Once initiated, a procedure cannot be halted until the job is fully completed.

4. Proposed Hybrid HSA and GA (HHSGA)

This section first introduces the basic version of the HSA, and then proposes the HHSGA for the concerned problem.

4.1. HSA

The HSA is a metaheuristic algorithm (Geem et al., 2001) that was developed using multiple musical improvisations to solve optimization problems. In the original HSA, which was designed to mimic the process of improvisation in playing music, each note played by a musician corresponds to a decision variable of the problem. A combination of all the generated notes can be used to produce a candidate solution, namely a *harmony*. In addition, an evaluation function must be predefined to determine the harmony performance. Because this study examined minimization problems, the evaluation

function was considered as a cost. The HSA is generally used to record a fixed number of historical harmonies searched so far. These harmonies and their costs are stored in a harmony memory (*HM*) matrix. To generate the final harmony solution, *HM* is iteratively adjusted multiple times by selecting, adjusting, and randomly generating harmonies from old notes until a termination criterion is obtained (i.e., the optimal harmony).

After the cost function and all parameters are set, the basic steps of the HSA are listed as follows:

Step 1. Generate *hms* random harmonies in the harmony memory *HM*, in which the *i*th harmony is expressed as $x^i = (x_1^i, x_2^i, ..., x_n^i)$ and its cost as $f(x^i)$. The *HM* is represented as follows:

$$HM = \left[\begin{array}{ccc|c} x_1^1 & \cdots & x_n^1 & f(x^1) \\ \vdots & \ddots & \vdots & \vdots \\ x_1^{hms} & \cdots & x_n^{hms} & f(x^{hms}) \end{array}\right].$$

Step 2. Generate a new harmony $x' = (x_1', x_2', ..., x_n')$, in which each note x_i' can be generated in two possible ways. If a random number generated from [0, 1] is not greater than a given parameter called harmony memory consideration rate *HMCR*, note x_i' is assigned with a random historical note value from the *i*th column $\{x_i^1, ..., x_i^{hms}\}$ of matrix *HM*. Otherwise, it is assigned to a random note value.

Step 3. If note x_i' in the previous step is assigned with a historical note value from *HM*, and a random number generated from [0, 1] is not greater than a given parameter called pitch adjustment rate *PAR*, then this note is adjusted by $x_i' = x_i' \pm \delta$.

Step 4. Calculate the cost $f(x')$ of harmony x'. If this cost is less than that of the worst harmony in *HM*, this worst harmony is replaced by x'.

Step 5. Repeat Steps 2 to 4 until the terminal criterion is met.

4.2. Representation of a harmony

To employ the HSA to solve the reentrant hybrid flow shop scheduling problem, the first step is to determine how to encode a solution for the problem into a harmony in the HSA. This problem is a complex discrete problem because a reentrant hybrid flow shop process is characterized by production stage limitations, the distribution of machines at each stage, and most importantly, the reentrant nature of the production stages. In a reentrant hybrid flow shop scheduling problem involving N jobs, G stages, and M machines at each stage/station, the harmony encoding includes job permutation and machine assignment, expressed as $\langle a_1, a_2, \ldots, a_{(N\cdot G)} \,||\, b_{11}, b_{12}, \ldots, b_{1G} \,|\, b_{21}, b_{22}, .., b_{2G} \,|\, \ldots \,|\, b_{N1}, b_{N2}, \ldots, b_{NG}\rangle$, where the first part $\langle a_1, a_2, \ldots, a_{(N\cdot G)}\rangle$ is a job permutation of G 1's, G 2's, …, and G N's; and each note $b_{ij} \in \{1, 2, \ldots, M\}$ in the second part $\langle b_{11}, b_{12}, \ldots, b_{1G} \,|\, b_{21}, b_{22}, .., b_{2G} \,|\, \ldots| \, b_{N1}, b_{N2}, \ldots, b_{NG}\rangle$ suggests that machine b_{ij} is assigned to stage j of job i. For instance, Fig. 2 corresponds to the harmony encoding of the scheduling problem with 3 jobs, 8 stages, and 2 machines (i.e., $N = 3$, $G = 8$, $M = 2$).

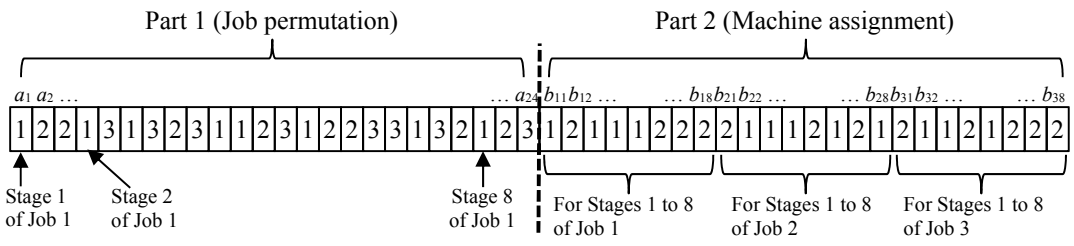


Fig. 2. An example of the harmony with 3 jobs, 8 stages, and 2 machines at each stage.

4.3. Decoding a harmony and evaluating its cost

When a harmony $\langle a_1, a_2, \ldots, a_{(N\cdot G)} \parallel b_{11}, b_{12}, \ldots, b_{1G} \mid b_{21}, b_{22}, .., b_{2G} \mid \ldots \mid b_{N1}, b_{N2}, \ldots, b_{NG}\rangle$ is given, the following procedures can be followed to decode the harmony into a solution of the problem:

- The first part $\langle a_1, a_2, \ldots, a_{(N\cdot G)}\rangle$ in the harmony consists of G 1's, G 2's, …, and G N's, and determines the ordering of stages of all jobs to be processed. Consider each note a_j in $\langle a_1, a_2, \ldots, a_{(N\cdot G)}\rangle$ from the left to the right. If note a_j is considered at the gth time, then the stage g of job i is processed. For instance, Fig. 2 shows that stage 1 of job 1, stage 1 of job 2, stage 2 of job 2, stage 2 of job 1, stage 1 of job 3, and so on are processed sequentially.
- In the second part $\langle b_{11}, b_{12}, \ldots, b_{1G} \mid b_{21}, b_{22}, .., b_{2G} \mid \ldots \mid b_{N1}, b_{N2}, \ldots, b_{NG}\rangle$ in the harmony, each note $b_{ij} \in \{1, 2, \ldots, M\}$ represents that machine b_{ij} is assigned to stage j of job i. For instance, Fig. 2 shows that machines 1, 2, 1, 1, 1, 2, 2, and 2 are assigned to the 8 stages of job 1, respectively.

Given a harmony, the harmony is decoded to calculate its cost by Algorithm 1, which is explained as follows. Lines 1 and 2 initialize the set B_{sm} and the machine time F_{sm} for each machine m of each station s. The major loop of Lines 3–30 iteratively considers each note a_k in part 1 of harmony x from the left to the right. According to note a_k, Lines 4–6 determine the current job index i, stage index g, and station index s. Then, Line 7 determines machine index m according to note b_{ig} in part 2 of harmony x. Then, consider two cases depending on whether the current stage g is the first stage in Line 8. If true, this algorithm further checks whether set B_{sm} is empty. If the set is empty, it means that no job

was ever processed in machine m of station s. Hence, the start time of job i (R_i) and the start time of stage g of job i (S_{ig}) is zero (Line 10); otherwise, they are set as the completion time of the latest job in machine m of station s (F_{sm}) (Line 12).

Algorithm 1. Cost_Evaluation(harmony $x = \langle a_1, \ldots, a_{(N \cdot G)} \,||\, b_{11}, \ldots, b_{1G} \,|\, \ldots \,|\, b_{N1}, \ldots, b_{NG} \rangle$)

```
1:  B_sm = ∅ for each station s and machine m
2:  F_sm = 0 for each station s and machine m
3:  for k = 1 to N · G do
4:      i = a_k
5:      g = index of the current stage for job i
6:      s = index of the station for stage g
7:      m = b_ig
8:      if g = 1 then
9:          if B_sm = ∅ then
10:             S_ig = R_i = 0
11:         else
12:             S_ig = R_i = F_sm
13:         end if
14:     else
15:         if | B_sm | > B and C_i(g-1) + T_ps < C_jt where p = index of the station for stage g – 1, j = index of
            the latest (B + 1)-th job in B_sm, and t = index of the current stage of job j then
16:             if C_i(g-1) + τ_p < C_jt then
17:                 S_ig = max{C_i(g-1) + τ_p + τ_s, F_sm}
18:             else
19:                 S_ig = max{C_jt + τ_s, F_sm}
20:             end if
21:         else
22:             S_ig = max{C_i(g-1) + T_ps, F_sm}
23:         end if
24:         Record that stage g of job i is completed
25:         Remove job i from B_ph where h = b_i(g-1)
26:     end if
27:     Add job i at stage g to B_sm
28:     C_ig = S_ig + P_ig
29:     F_sm = C_ig
30: end for
31: Compute C_max = max{C_1G, C_2G, …, C_NG}
32: Compute f̄ = Σ_{i=1}^{N} (C_iS − R_i) / N
33: Output ω_1 · C_max + ω_2 · f̄
```

When the current stage g is not the first stage (Line 14), Line 15 checks whether the size of set B_{sm} is greater than B (i.e., the buffer of machine m of station s is fully occupied), and the time of job i arriving at stage g (i.e., $C_{i(g-1)} + T_{ps}$) is less than the completion time C_{jt} of the latest (B + 1)-th job j in set B_{sm}, in which p represents the index

of the station for stage $g - 1$, and t represents the index of the current stage of job j. If true, job j requires moving to the stocker (Lines 16–20). If $C_{i(g-1)} + \tau_p < C_{jt}$ (Line 16), it means that job i arrives at the stocker before job j is completed, and hence, job i does not stay at the stocker any longer and returns to station s immediately. Then, the start time S_{ig} of stage g of job i is calculated as follows:

$$S_{ig} = \max\{C_{i(g-1)} + \tau_p + \tau_s, F_{sm}\}. \quad (2)$$

Note that if the start time S_{ig} is less than F_{sm} (i.e., the completion time of the latest job in this machine), then this job is queued at the buffer and is delayed to start at F_{sm}.

The case of Line 18 means that job j has been completed before job i arrives at the stocker. In this case, job i waits at the stocker, and returns to station s when completion of job j is informed. Then, the start time S_{ig} of stage g of job i in this case is calculated as follows:

$$S_{ig} = \max\{C_{jt} + \tau_s, F_{sm}\}. \quad (3)$$

In the case when the buffer of machine m of station s is not fully occupied (Line 21), job i can transfer to station s to be processed. Then, the start time S_{ig} of stage g of job i in this case is calculated as follows:

$$S_{ig} = \max\{C_{j(g-1)} + T_{ps}, F_{sm}\}. \quad (4)$$

Subsequently, Line 24 records completion of the concerned stage g of job i, and Line 25 removes this job from the set B_{ph} (i.e., the job set for the machine at the earlier stage of job i). Line 27 adds this job to the set B_{sm} at the current machine m of the current station s. After the start time S_{ig} is determined, Line 28 adds the processing time P_{ig} to S_{ig} to obtain

the completion time C_{ig}, and Line 29 updates F_{sm} as C_{ig}. Finally, Lines 31 – 33 compute $C_{\max}$ and $\overline{f}$, and output the objective value.

The Gantt chart generated by decoding the harmony in Fig. 2 is shown in Fig. 3, in which the vertical axis shows four station labels "s1"–"s4" and two parallel machine labels "m1" and "m2" for each station; the horizontal axis is the time axis; the processing times of three jobs are expressed by three different-colored rectangles; each colored rectangle is associated with two numbers (in which the numbers attached to the left and the right sides of each rectangle are the start and the completion times, respectively). From this chart, the makespan and the mean flowtime for each job can be obtained to further calculate the final cost value. The total cost of the schedule is $\omega_1 \cdot C_{\max} + \omega_2 \cdot \overline{f}$ $= 0.50 \times 195 + 0.50 \times 175.00 = 185.00$, as shown in the below of Fig. 3.

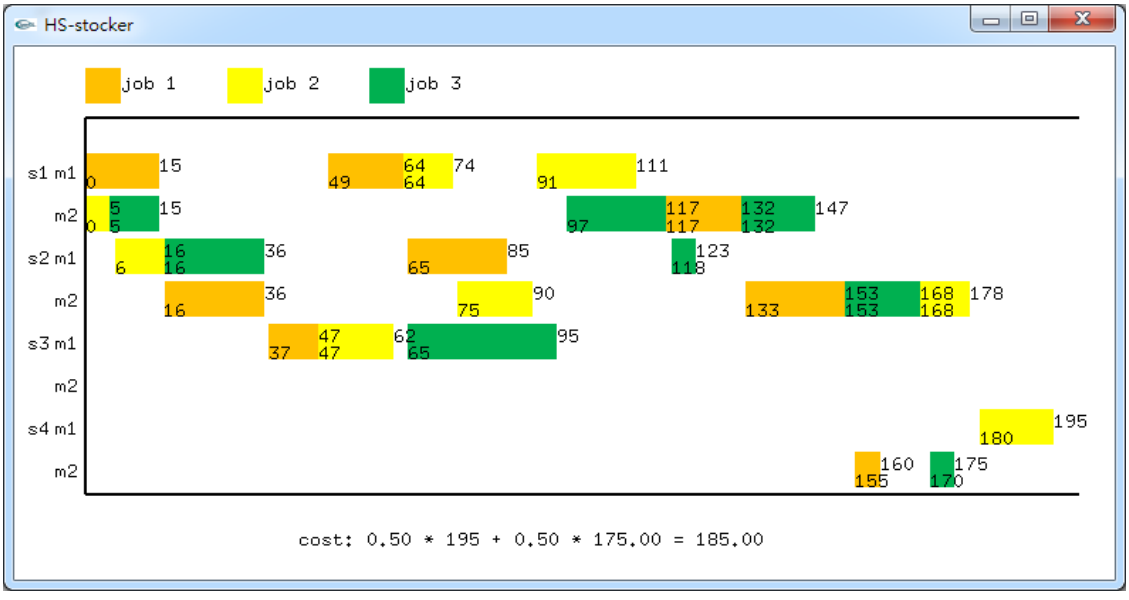


Fig. 3. The Gantt chart generated by decoding the harmony in Fig. 2.

It is worthy to notice a gap between jobs 2 and 3 in machine “m1” of station “s3”. This gap occurs because job 3 takes too much time to make a round trip to the stocker. It should also be noticed that this gap does not imply inefficiency. The stocker allows an additional buffer to avoid production stagnation. If there is no stocker, stage 3 of job 3 will wait for longer time to start the production process in machine “m1” of station “s3”.

4.4. Algorithm design

According to the no free lunch theorem for optimization (Wolpert & Macready, 1997), there is no metaheuristic algorithm that is universal to solve all optimization problems. Therefore, this study develops a hybrid metaheuristic algorithm to solve the examined problems, in which a GA is incorporated into an HSA. From the context of evolution and natural selection, the solutions of GA are generated using crossover and mutation operators on chromosomes. The advantage of GA is that nearly-optimal solutions for a complex problem can be generated in a timely manner. However, GA may prematurely converge to a local optimum in some cases. By contrast, HSA is suitable for solving global optimization problems, but they are less effective in local search ability compared with GA. Thus, this study incorporates GA and HSA to propose a HHSGA to improve the local search ability, increase the diversity of solutions, and thereby enhance the efficiency of generating global optimums.

The HHSGA is given in Algorithm 2, which is explained as follows. The main loop of the algorithm considers H iterations by increasing the iteration number η (Lines 1, 2, and 37). For each iteration, different from the conventional HSA that applies a fixed PAR value, Line 3 considers flexible PAR_1 and PAR_2 values according to the following linear

formula of iteration number η (Mahdavi et al., 2007):

$$PAR_j = PAR_j^{\max} - \frac{PAR_j^{\max} - PAR_j^{\min}}{H} \cdot \eta \quad \text{for } j = 1, 2. \tag{5}$$

where $PAR_j^{\min}$ and $PAR_j^{\max}$ are the minimum and maximum of the PAR_j value. When the iteration number η increases, the PAR_j value decreases dynamically, i.e., the earlier iterations emphasize more diversity; whereas the later iterations emphasize more stability.

Subsequently, Line 4 adopts the $HMCR_1$ value to respectively applying GA operators (Lines 5–11) and HSA operators (Lines 13–33) to generate a new harmony x^{new}. In the GA part, Line 5 generates two parent harmony x_1 and x_2 through tournament selection. Then, if a random number between 0 and 1 is less than the PAR_1 value, then Line 7 generates two children harmonies x_1^{new} and x_2^{new} through the following two-point crossover on the two chosen parent harmonies x_1 and x_2. Randomly generate two cut points c_1 and c_2 from $\{2, 3, \ldots, N \cdot G\}$. Each of the two parts of each parent harmony has $N \cdot G$ notes, and is divided into three regions: notes 1 to $c_1 - 1$, notes c_1 to $c_2 - 1$, and notes c_2 to $N \cdot G$. The first child harmony x_1^{new} consists of region 1 of part 1 of x_1, region 2 of part 1 of x_2, and region 3 of part 1 of x_1, region 1 of part 2 of x_1, region 2 of part 2 of x_2, and region 3 of part 2 of x_1. The second child harmony x_2^{new} can be generated symmetrically.

Algorithm 2. HHSGA

1: $\eta = 1$
2: **while** $\eta \le H$ **do**
3: Adjust PAR_1 and PAR_2 values according to (5)
4: **if** *rand*(0, 1) < $HMCR_1$ **then**
5: Employ two times of tournament selection to choose two parent harmonies x_1 and x_2 from *HM*
6: **if** *rand*(0, 1) < PAR_1 **then**
7: Apply the two-point crossover on the two chosen parents x_1 and x_2 to generate two children harmonies x_1^{new} and x_2^{new}
8: Repair the two children harmony by Algorithm 3, and then let x^{new} be the children harmony with the better cost value
9: **else**
10: Apply the mutation operator on the parent harmony with the better cost value to obtain a new harmony x^{new}
11: **end if**
12: **else**
13: **for** $k = 0$ to $N \cdot G$ **do**
14: **if** *rand*(0, 1) < $HMCR_2$ **then**
15: a_k^{new} is assigned a random note from the kth column of the *HM*
16: **if** *rand*(0, 1) < PAR_2 **then**
17: $a_k^{new} = a_k^{new} \pm 1$ within the feasible range $\{1, 2, \ldots, N\}$
18: **end if**
19: **else**
20: a_k^{new} is assigned a random note within the feasible range $\{1, 2, \ldots, N\}$
21: **end if**
22: **end for**
23: **for** $k = 0$ to $N \cdot G$ **do**
24: **if** *rand*(0, 1) < $HMCR_2$ **then**
25: b_k^{new} is assigned a random note from the kth column of the *HM*
26: **if** *rand*(0, 1) < PAR_2 **then**
27: $b_k^{new} = b_k^{new} \pm 1$ within the feasible range $\{1, 2, \ldots, M\}$
28: **end if**
29: **else**
30: a_k^{new} is assigned a random note within the feasible range $\{1, 2, \ldots, M\}$
31: **end if**
32: **end for**
33: $x^{\text{new}} = \left\langle a_1^{\text{new}}, \ldots, a_{(N \cdot G)}^{\text{new}} \,||\, b_{11}^{\text{new}}, \ldots, b_{1G}^{\text{new}} \,|\ldots|\, b_{N1}^{\text{new}}, \ldots, b_{NG}^{\text{new}} \right\rangle$
34: **end if**
35: Repair harmony x^{new} to be feasible by Algorithm 3
36: The feasible harmony x^{new} replaces the worse harmony in the *HM* if it has a lower cost value
37: $\eta = \eta + 1$
38: **end while**
39: Decode the best harmony in the *HM* as the solution

Because the two new children harmonies may not be feasible, Line 8 repairs them by Algorithm 3, and then lets x^{new} be the children harmony with the better cost value.

Algorithm 3 is explained as follows. Since part 2 of harmony x (assigning stages of each job to machines) is always feasible, it suffices to repair part 1 of harmony x. Hence, Algorithm 3 first scans part 1 of harmony x from left to right, and records illegal notes. Then, we count the frequency f_i of illegal notes for each job ID i, and then construct a random sequence consisting of $(N - f_i)$ legal notes of each job ID i. Finally, all illegal notes are replaced by legal notes according to the legal note sequence.

Algorithm 3. Repar_Infeasibility(harmony $x = \langle a_1, \ldots, a_{(N \cdot G)} \,||\, b_{11}, \ldots, b_{1G} \,|\, \ldots |\, b_{N1}, \ldots, b_{NG} \rangle$)

1: **for** $k = 1$ to $N \cdot G$ **do**
2: For $a_1, \ldots, a_k$, if the job ID of a_k has appeared more than N times, then record that a_k is illegal
3: **end for**
4: Count the frequency f_i of illegal notes for each job ID i in the first part of harmony x
5: Construct a random sequence consisting of $(N - f_i)$ legal notes of each job ID i
6: **for** $k = 1$ to $N \cdot G$ **do**
7: If a_k is illegal, a_k is replaced by a legal note according to the above legal note sequence
8: **end for**

Continue Algorithm 2. In the case when the random number is no less than PAR_1 (Line 9), Line 10 chooses the parent harmony with the better cost value (i.e., one of harmony x_1 and x_2), and then conducts the mutation operator on this harmony to obtain a new harmony x^{new}. The mutation operator randomly selects a note from harmony x^{new}, and then modifies this note to a number in the feasible range (i.e., $\{1, 2, \ldots, N\}$ and $\{1, 2, \ldots, M\}$ for parts 1 and 2 of the harmony, respectively).

Subsequently, Lines 13–33 conducts HSA to obtain a new harmony x^{new}. That is, we determine each note by adopting *MHCR* and *PAR* values to conduct three adjustment schemes: assigning a random note from the same column in the *HM* matrix (Lines 15 and 25), pitch adjustment (Lines 17 and 27), and assigning a random note (Lines 20 and 30).

After the above GA and HSA operators, the generated harmony x^{new} may not be

feasible. Hence, Line 35 repairs it by Algorithm 3. Then, Line 36 finds the worse harmony x^{worst} in the *HM*, and replaces it by harmony x^{new} if x^{new} has a lower cost value than x^{worst}. Finally, Line 39 decodes the best harmony in the *HM* as the solution of the concerned problem.

5. Implementation and Experimental Results

To examine the reentrant hybrid flow shop scheduling problems with stockers (see Section 2), this section conducts an experimental analysis. In Section 5.1, the proposed approach is applied to the benchmark problems without stockers to analyze whether this approach is suitable for reentrant hybrid flow shop scheduling problems. Section 5.2 describes the experimental and environmental parameter settings. A sensitivity analysis of the parameter settings of the HHSGA is given in Section 5.3, and the effectiveness of the HSA, GA, and HHSGA in solving scheduling problems in a thin-film chip resistor manufacturing process is reported in Section 5.4 along with an analysis of their differences. Section 5.5 presents an analysis of the differences between production processes with and without a stocker.

5.1. Experimental analysis of the benchmark problems without stockers

Chamnanlor et al. (2014) proposed an adaptive hybrid GA (AHGA) for 40 problem instances for minimizing the makespan C_{max} of a conventional disk manufacturing process without stockers. Table 3 shows experimental comparison of the current practice, the previous AHGA (Chamnanlor et al., 2014), and the proposed HHSGA for the 40 problem instances, in which the ‘Average’ and ‘Best’ indicate the average and the best of

the makespan results of running 100 times of the algorithm, respectively; and the 'Relative improvement' is the percentage of the improvement of the proposed HHSGA over the other two approaches. From Table 3, without stockers, the proposed HHSGA performs best than the other two approaches for almost all problem instances. Thus, the proposed HHSGA is suitable for solving the reentrant hybrid flow shop scheduling problems.

Table 3. Experimental comparison of the scheduling problem without stockers.

Problem instance			Current practice	AHGA		HHSGA		Relative improvement of HHSGA over current practice	Relative improvement of HHSGA over AHGA
Process times	No. of jobs	Type	C_{max} (m)	C_{max} (m)		C_{max} (m)			
				Average	Best	Average	Best		
ProcT1	7 jobs	L1	611	488	487	488	486	20.5%	0.2%
		L2	605	489	488	488	486	19.7%	0.4%
		L3	605	490	488	489	486	19.7%	0.4%
		L4	609	489	488	489	486	20.2%	0.4%
		L5	631	491	489	488	486	23.0%	0.6%
	11 jobs	L1	910	714	713	706	702	22.9%	1.5%
		L2	923	708	704	707	702	23.9%	0.3%
		L3	891	713	713	709	702	21.2%	1.5%
		L4	930	711	710	710	702	24.5%	1.1%
		L5	944	716	715	709	702	25.6%	1.8%
ProcT2	7 jobs	L1	1828	1627	1624	1627	1624	11.2%	0.0%
		L2	1854	1629	1627	1629	1624	12.4%	0.2%
		L3	1854	1627	1624	1626	1624	12.4%	0.0%
		L4	1828	1627	1624	1626	1624	11.2%	0.0%
		L5	1854	1624	1624	1625	1624	12.4%	0.0%
	11 jobs	L1	2800	2468	2464	2467	2464	12.0%	0.0%
		L2	2800	2469	2464	2467	2464	12.0%	0.0%
		L3	2826	2469	2467	2467	2464	12.8%	0.1%
		L4	2826	2474	2473	2466	2464	12.8%	0.4%
		L5	2878	2476	2474	2468	2464	14.4%	0.4%

Note that the benchmark problems without stockers in this subsection are from (Chamnanlor et al., 2014), and each of them only involves 7 or 11 jobs. It is reasonable because this subsection aims to compare the performance of our proposed HHSGA with the state-of-the-art metaheuristic method for the same problem setting except for not having stockers (i.e., the AHGA proposed by Chamnanlor et al. (2014)). Specifically, the statistical results for the 'AHGA' in Table 3 are from (Chamnanlor et al., 2014). For

performance comparison, it would be fair to directly refer to their statistical results rather than implementing their method by ourselves. Note that the other recent works on the reentrant hybrid flow shop (i.e., Cho and Jeong, 2017; Zhang and Chen, 2017) did not focus on only the scheduling problem nor investigated the same problem setting.

5.2. Experiment setting for the problems with stockers

This study experimentally investigates the scheduling problems of a thin-film chip resistor manufacturing process (YAGEOmkt, 2013) and considers the effect of using stockers. The manufacturing process is illustrated in Fig. 4, and the experimental setting of this process is given in Table 4.

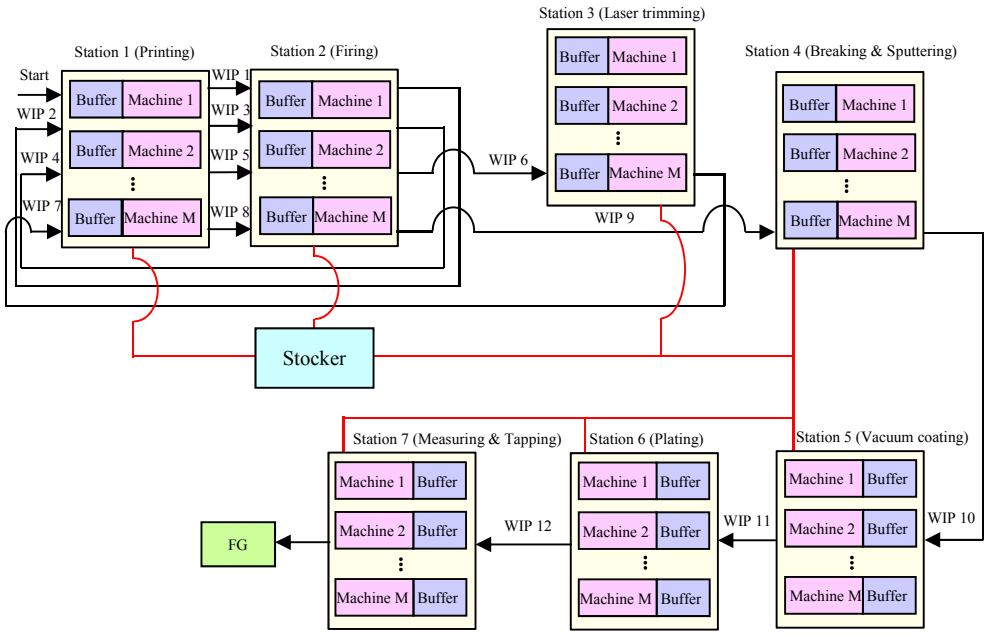


Fig. 4. Illustration of a thin-film chip resistor manufacturing process.

For comparative analysis, three approaches (i.e., HSA, GA, and HHSGA) are adopted to generate solutions for the aforementioned scheduling problems. The three approaches

are implemented in the C++ programing language, and the experiments are operated on a PC with Intel i5-3400 CPU and 8-GB RAM. Table 4 presents the experimental setting of the scheduling problems of the reentrant hybrid flow shop process with stockers; the scheduling problems are analyzed for three job quantities (i.e., 20, 50, and 100) in the same manufacturing environment. Each job is processed in 13 stages. Each station contains four parallel machines and each machine has a buffer that can inventory at most four jobs. The stocker capacity is set as 10.

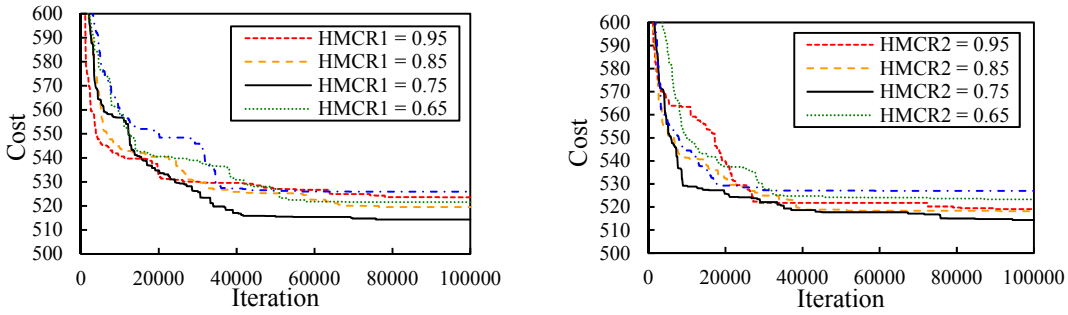
Table 4. Experimental setting for the process in Fig. 4.

Parameter	Value
Number of jobs	20, 50, 100
Number of stages	13
Number of stations	7
Number of parallel machines at a station	4
Buffer capacity	4
Stocker capacity	10

5.3. Parameter sensitivity analysis of the proposed HHSGA

This subsection presents the parameter sensitivity analysis of the size of *HM* and parameters $HMCR_1$, $HMCR_2$, $PAR_1^{\min}$, $PAR_1^{\max}$, $PAR_2^{\min}$, and $PAR_2^{\max}$ of the proposed HHSGA. Initially, $HMCR_1 = 0.9$, $HMCR_2 = 0.9$, $PAR_1^{\min} = 0.2$, $PAR_1^{\max} = 0.9$, $PAR_2^{\min} = 0.2$, and $PAR_2^{\max} = 0.9$. Fig. 5(a) shows parameter sensitivity analyses on $HMCR_1$ values set at 0.55, 0.65, 0.75, 0.85, and 0.95. From Fig. 5(a), the best solutions are obtained when $HMCR_1 = 0.75$. Thus, $HMCR_1$ is set at 0.75. Similar analysis is conducted for $HMCR_2$ with the values set at 0.55, 0.65, 0.75, 0.85, and 0.95 (Fig. 5(b)). The results

suggest that the best solutions are obtained when $HMCR_2 = 0.75$. Table 5 illustrates that the best solutions are obtained when $PAR_1^{\min} = 0.1$ and $PAR_1^{\max} = 0.4$. Table 6 indicates that the best solutions are obtained when $PAR_2^{\min} = 0.4$ and $PAR_2^{\max} = 0.8$. Therefore, the parameter sensitivity analysis results are employed to derive the experimental parameter settings for the proposed HHSGA (Table 7).



(a) (b)

Fig. 5. Comparison of the results under different $HMCR_1$ values and $HMCR_2$ values.

5.4. Analyzing the experimental results of the problems with stockers

This subsection compares the initial solutions and the solutions generated by three approaches (i.e., HSA, GA, and HHSGA) for the scheduling problems of the thin-film chip resistor manufacturing processes with various job quantities (i.e., 20, 50, and 100). Fig. 6 shows the box plots of the initial solutions and the solutions generated by three approaches for the three different-scale problems.

Table 5. Comparison of the results under different combinations of $PAR_1^{\max}$ and $PAR_1^{\min}$ values.

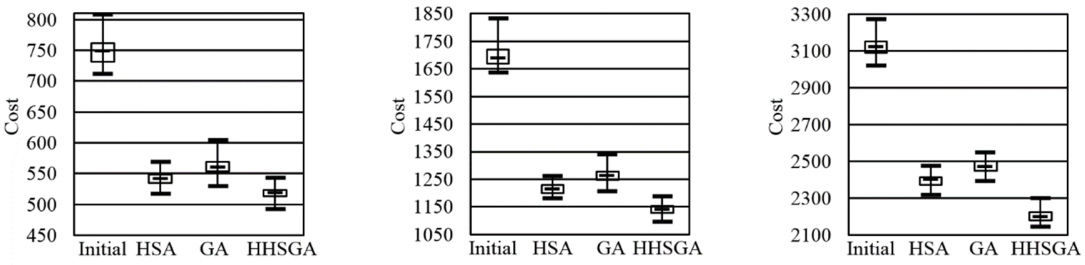
$PAR_1^{\max}$	$PAR_1^{\min}$	HHSGA	
		Average	StdDev
0.1	0.1	521.32	13.16
0.2	0.1	521.45	8.77
	0.2	519.76	9.60
0.3	0.1	521.70	8.36
	0.2	521.50	10.14
	0.3	519.06	9.32
0.4	**0.1**	517.67	9.88
	0.2	518.31	10.12
	0.3	520.29	8.74
	0.4	521.61	11.25
0.5	0.1	522.36	7.84
	0.2	520.05	9.29
	0.3	523.33	8.79
	0.4	526.10	9.83
	0.5	522.78	9.28
0.6	0.1	519.03	9.41
	0.2	520.30	9.16
	0.3	520.15	9.65
	0.4	522.23	10.47
	0.5	520.79	9.07
	0.6	520.95	8.77
0.7	0.1	526.01	7.58
	0.2	519.76	9.60
	0.3	520.96	7.93
	0.4	521.92	6.73
	0.5	522.07	8.42
	0.6	521.86	7.58
	0.7	525.52	10.43
0.8	0.1	522.80	11.16
	0.2	523.48	8.17
	0.3	520.76	10.23
	0.4	525.05	11.00
	0.5	528.13	9.57
	0.6	520.47	12.07
	0.7	526.36	8.12
	0.8	527.99	10.19
0.9	0.1	524.27	8.89
	0.2	520.90	7.58
	0.3	524.00	6.74
	0.4	521.21	6.16
	0.5	524.80	7.80
	0.6	530.25	12.79
	0.7	528.84	5.83
	0.8	528.30	8.35
	0.9	534.53	8.60

Table 6. Comparison of the results under different combination of $PAR_2^{\max}$ and $PAR_2^{\min}$ values.

$PAR_2^{\max}$	$PAR_2^{\min}$	HHSGA	
		Average	StdDev
0.1	0.1	520.24	8.73
0.2	0.1	521.21	9.42
	0.2	520.76	9.02
0.3	0.1	519.40	10.57
	0.2	519.49	8.73
	0.3	520.90	9.32
0.4	0.1	522.24	7.72
	0.2	520.97	8.45
	0.3	522.11	9.69
	0.4	519.43	8.66
0.5	0.1	522.66	11.86
	0.2	519.68	10.05
	0.3	519.13	8.90
	0.4	519.07	9.53
	0.5	519.68	10.52
0.6	0.1	519.06	9.13
	0.2	520.43	9.10
	0.3	519.94	9.47
	0.4	520.10	9.20
	0.5	518.02	9.98
	0.6	518.95	8.66
0.7	0.1	520.53	7.38
	0.2	519.99	8.24
	0.3	520.91	9.33
	0.4	519.25	8.98
	0.5	519.36	9.71
	0.6	518.53	10.98
	0.7	519.94	7.91
0.8	0.1	520.92	7.85
	0.2	520.51	7.99
	0.3	520.55	10.09
	0.4	517.72	9.55
	0.5	520.56	9.55
	0.6	519.67	10.85
	0.7	520.26	8.27
	0.8	522.38	9.20
0.9	0.1	520.17	10.24
	0.2	520.72	8.24
	0.3	519.57	10.55
	0.4	518.97	11.45
	0.5	521.34	10.14
	0.6	521.30	7.39
	0.7	520.49	12.06
	0.8	520.55	9.22
	0.9	521.90	8.32

Table 7. Parameter setting in the proposed HHSGA

Parameter	Value
Harmony memory size (*hms*)	12
$HMCR_1$	0.75
$HMCR_2$	0.75
$PAR_1^{\min}$	0.1
$PAR_1^{\max}$	0.4
$PAR_2^{\min}$	0.4
$PAR_2^{\max}$	0.8
Number of iterations (*H*)	100000



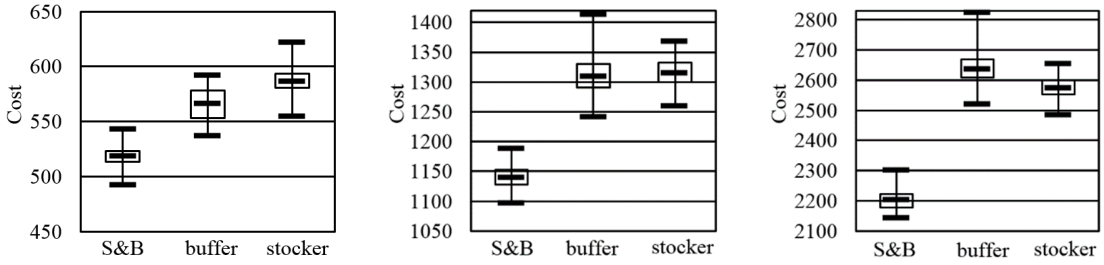
(a) 20 jobs (b) 50 jobs (c) 100 jobs

Fig. 6. Box plots of the initial solutions and the solutions generated by three approaches for three different-scale problems.

When the job quantity is 20, the initial solutions, HSA, GA, and HHSGA attain median cost values of 749, 541, 559, and 519, respectively (Fig. 6(a)). The corresponding median values are 1685, 1213, 1262, and 1140 when the job quantity is set at 50 (Fig. 6(b)), and 3124, 2400, 2471, and 2200 when the job quantity is set at 100. The experimental results show that compared with the initial solutions and the other two approaches, the proposed HHSGA generates the best cost values and, when the number of jobs increases, exhibits more favorable performance.

5.5. Comparing the experimental results with and without stockers

This subsection compares the following three manufacturing environment conditions: with inventory buffers and stockers, with inventory buffers only, and with stockers only. These three conditions are applied to the scheduling of a thin-film chip resistor manufacturing process with various job quantities (i.e., 20, 50, and 100). To examine the resulting differences under various manufacturing environments, Fig. 7 shows the box plots of the results generated from these three conditions, in which 'S&B' denotes the condition of 'with inventory buffers and stockers', 'buffer' denotes the condition of 'with inventory buffers only', and 'stocker' denote the condition of 'with stockers only'.



(a) 20 jobs (b) 50 jobs (c) 100 jobs

Fig. 7. Box plots of the results for three different-scale problems in three manufacturing environments.

As shown in Fig. 7(a), when the job quantity is 20, median cost values of 519, 566, and 586 are obtained for the manufacturing conditions with inventory buffers and stockers, with inventory buffers only, and with stockers only. The figure also indicates that under these conditions, manufacturing with inventory buffers and stockers generates the best cost values, followed that with buffers only, and finally that with stockers only.

From Fig. 7(b), the corresponding median cost values are 1140, 1309, and 1313 when the job quantity is set at 50. The order of the optimality of the median cost values is the same as that for previous condition. Finally, Fig. 7(c) shows that the corresponding median cost values are 2200, 2635, and 2572 when the job quantity is set to 100. The figure also shows that the order of optimality changes for this condition, with the manufacturing condition with inventory buffers and stockers generating the best cost values, followed by the condition with stockers only, and then the condition with inventory buffers only.

Note that all the experimental results and discussion above (i.e., those for Figs. 5–7 and Tables 5–7) are based on the manufacturing process setting in Fig. 4. If the experiments are conducted on a different manufacturing process setting, similar discussion results would be expected, as explained as follows. After testing multiple parameter settings for the different manufacturing process (i.e., similar to Fig. 5 and Tables 5–6), a parameter setting in the proposed HHSGA (i.e., similar to Table 6) can be obtained. Because the proposed HHSGA integrates the HSA and GA, it would perform better than the other two methods on average, and hence would obtain the box plot results similar to Figs. 6 and 7.

Several conclusions can be derived from the comparison of these three manufacturing conditions (Fig. 7). Regardless of the different job quantities, the condition with the inventory buffers and stockers performs the best among the three conditions. Figs. 8 and 9 are Gantt charts for the condition with inventory buffers and stockers and that with stockers only, respectively (when job quantity is set as 20). Note that the Gantt chart for the condition with inventory buffers only are not given because the time of a production bottleneck in this condition cannot be expressed in this type of chart. However, the box

plot of this manufacturing condition (Fig. 7(a)) presents the differences in flowtime. Because the condition with inventory buffers only may generate production stagnation due to the absence of an inventory buffer, the total flowtime under this condition can be improved.

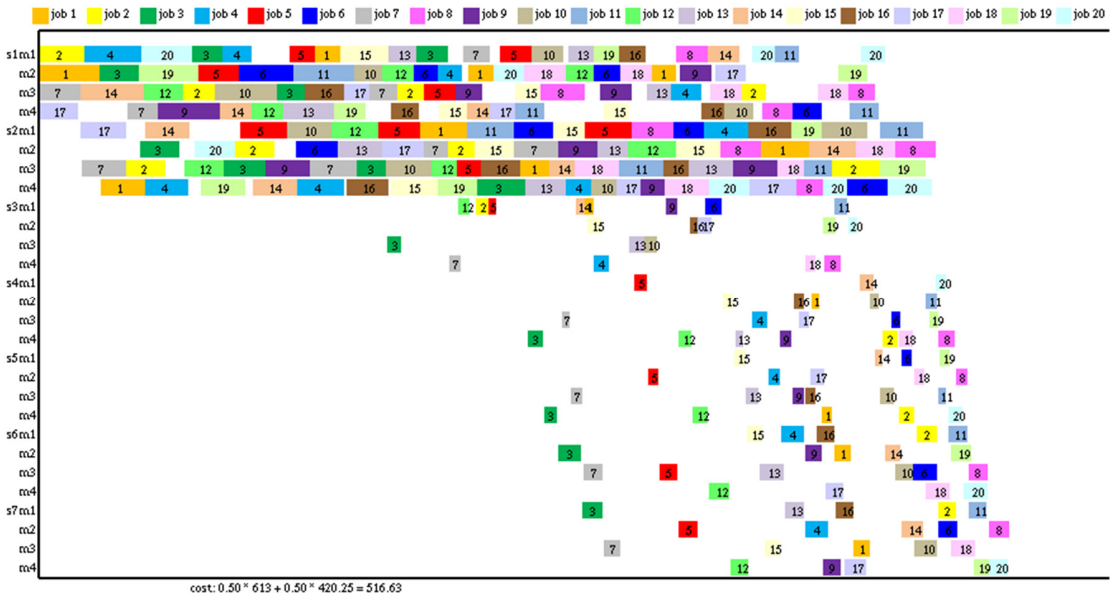


Fig. 8. Gantt chart for the condition with inventory buffers and stockers when job quantity is set as 20.

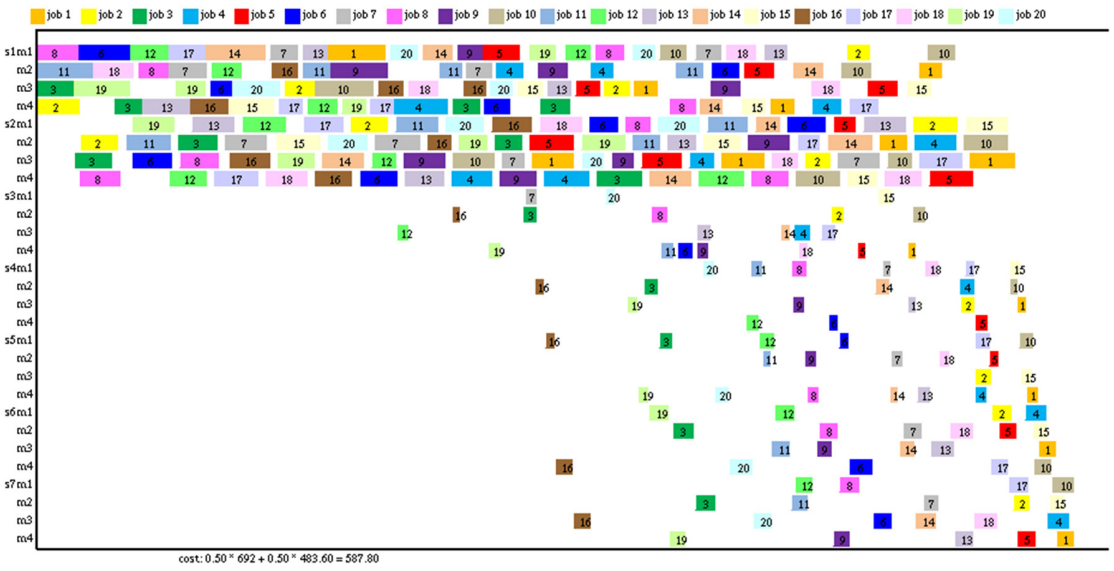


Fig. 9. Gantt chart for the condition with stockers only when job quantity is set as 20.

A comparison of Figs. 8 and 9 indicates that under the condition with stockers only, the total makespan is longer because of the excessive transfer time for the routing of WIPs to the stocker. Fig. 7 suggests that, when the job quantity is reduced, the manufacturing condition with inventory buffers only generates less costs as compared with those obtained under the condition with stockers only. This difference might be explained by the reduced possibility of exceeding the buffer capacity, because smaller job quantities are less likely to generate production stagnation under the condition with inventory buffers only, thus leading to less cost values. At larger job quantities, the manufacturing condition with inventory buffers only generates cost values that are worse than those obtained under the condition with stockers only. This difference might be explained by the increased likelihood of the buffer capacity being exceeded when the job quantity is increased, resulting in more production stagnation, longer total flowtimes, and worse cost values.

6. Conclusion and Future Work

An HHSGA has been proposed to solve the reentrant hybrid flow shop scheduling problems with stockers, while considering job permutation, transfer time between stations, and limited inventory buffer capacity. Because a limited inventory buffer capacity can cause WIP storage problems, a stocker was used to overcome this problem. Moreover, considering the additional transfer time and limited buffer capacity replicated a realistic manufacturing environment. The two-objective optimization problem of minimizing the makespan and mean flowtime was examined to evaluate scheduling problems. Thus, to achieve these two objectives, the machine utilization and job quantity in a production

process were evaluated to identify the trade-off between the two. Because this scheduling problem is NP-hard, the HHSGA is developed for addressing the problem.

Under different job quantities (i.e., 20, 50, and 100), simulation results showed that the proposed HHSGA generated more favorable scheduling results than did the HSA and GA, as evidenced by the higher median cost values obtained using the proposed method. In addition, under higher job quantities, the proposed algorithm outperformed the other two algorithms in scheduling performance, indicating that the proposed algorithm was more effective in handling the examined scheduling problems and generating good solutions. Moreover, differences in cost values generated under the three manufacturing conditions (i.e., with inventory buffers and stockers, with inventory buffers only, and with stockers only) were analyzed, revealing that, regardless of the job quantity, the best solutions were obtained under the condition with inventory buffers and stockers. Therefore, when examining scheduling problems resulting from a limited inventory buffer capacity, future studies should consider incorporating stockers because of their effectiveness in preventing production stagnation from a lack of buffer capacity; furthermore, they are widely used in manufacturing, and thus incorporating them is a more realistic representation of such environments.

This study supposed that all stockers are located in a centralized position. To meet the practice of more complex manufacturing processes, a line of the future work is to consider multiple stocker positions to solve the problems of limited WIP storage capacity and different transfer times between stockers and stations. Secondly, some products have much complex manufacturing processes, which may include assembling processes of more than two different WIPs. Hence, some WIPs after preprocessing stages may be

moved to stockers and stored there for long time. Thus, future studies are suggested to incorporate the scheduling and distribution problems of WIPs to stockers. Thirdly, if stockers are not deployed properly, they would result in suboptimal routing paths when transferring WIPs between stockers and stations. Therefore, it would be of future interest to incorporate scheduling and facility location problems of multiple stockers. Finally, because this study proposes a new setting for the reentrant hybrid flow shop scheduling problem with stockers and limited buffer capacity and proposes a metaheuristic solution for the problem, it would be of interest to further establish a mathematical programming solution for this problem.

References

Agrawal, G. K., Heragu, S. S., 2006. A survey of automated material handling systems in 300-mm semiconductor fabs. *IEEE Transactions on Semiconductor Manufacturing*, 19(1), 112–120.

Alaykýran, K., Engin, O., Döyen, A., 2007. Using ant colony optimization to solve hybrid flow shop scheduling problems. *International Journal of Advanced Manufacturing Technology*, 35, 541–550.

Almeder, C., Hartl, R. F., 2013. A metaheuristic optimization approach for a real-world stochastic flexible flow shop problem with limited buffer. *International Journal of Production Economics*, 145(1), 88–95.

Chamnanlor, C., Sethanan, K., Chien, C. F., Gen, M. 2014. Re-entrant flow shop scheduling problem with time windows using hybrid genetic algorithm based on

auto-tuning strategy. *International Journal of Production Research*, 52(9), 2612–2629.

Chen, J. S., Pan, J. C. H., 2006. Integer programming models for the re-entrant shop scheduling problems. *Engineering Optimization*, 38(5), 577–592.

Cheng, M., Sarin, S. C., Sing, S., 2016. Two-stage, single-lot, lot streaming problem for a 1+2 hybrid flow shop. *Journal of Global Optimization*, 66(2), 1–28.

Cho, H.-M., Jeong, I.-J., 2017. A two-level method of production planning and scheduling for bi-objective reentrant hybrid flow shops. *Computers & Industrial Engineering* 106, 174–181.

Chou, C. W., Chien, C. F., Gen, M., 2014. A multiobjective hybrid genetic algorithm for TFT-LCD module assembly scheduling. *IEEE Transactions on Automation Science and Engineering* 11(3), 692–705.

Dios, M., Fernandez-Viagas, V., Framinan, J. M., 2018. Efficient heuristics for the hybrid flow shop scheduling problem with missing operations. *Computers & Industrial Engineering*, 115, 88–99.

Dugardin, F., Amodeo, L., Yalaoui, F., 2009. Multiobjective scheduling of a reentrant hybrid f1owshop. In *Proc. of 2009 IEEE International Conference on Computers & Industrial Engineering*, pp. 193–195.

EI-Khouly, l. A., EI-Kilaoy, K. S., EI-Sayed, A. E., 2009. Modelling and simulation of re-entrant flow shop scheduling: An application in semiconductor manufacturing. In *Proc. of 2009 IEEE International Conference on Computers & Industrial Engineering*, pp. 211–216.

Fernandez-Viagas, V., Perez-Gonzalez, P., Framinan, J. M., 2019. Efficiency of the solution representations for the hybrid flow shop scheduling problem with makespan objective. *Computers & Operations Research*, 109, 77–88.

Geem, Z. W., Kim, J. H., Loganathan, G. V., 2001. A new heuristic optimization algorithm: Harmony search. *Journal of Simulation*, 76(2), 60–68.

Han, Z., Shi, H., Yuan, W., Zhang, Z., 2012. Hybrid PSO/DE solution for the earliness/tardiness case of hybrid flow-shop scheduling problem. In *Proc. of 2012 International Conference on Modelling, Identification and Control*, IEEE Press, pp. 277–282.

Hinze, R., Sackmann, D., 2016. An iterated local search for a re-entrant flow shop scheduling problem. In *Operations Research Proceedings 2014*, Springer International Publishing, pp. 221–226.

Hoogeveen, J. A., Lenstra, J. K., Veltman, B., 1996. Preemptive scheduling in a two-stage flow shop NP-hard. *European Journal of Operational Research*, 89, 172–175.

Hsieh, C. H., Cho, C., Yang, T., Chang, T. J., 2012. Simulation study for a proposed segmented automated material handling system design for 300-mm semiconductor fabs. *Simulation Modelling Practice and Theory*, 29, 18–31.

Huang, C. J., Chang, K. H., Lin, J. T., 2012. Optimal vehicle allocation for an automated materials handling system using simulation optimisation. *International Journal of Production Research*, 50(20), 5734–5746.

Huang, I. H., Fujimura, S., 2013. A fuzzy-based multi-term genetic algorithm for reentrant flow shop scheduling problem. In *Proc. of 2013 IEEE International*

Conference on Industrial Engineering and Engineering Management, pp. 10–13.

Jing, C., Tang, G., Qiana, X., 2008. Heuristic algorithms for two machine re-entrant flow shop. *Theoretical Computer Science*, 400, 137–143.

Kuo, Y., Yang, T., Peters, B. A., Chang, I., 2007. Simulation metamodel development using uniform design and neural networks for automated material handling systems in semiconductor wafer fabrication. *Simulation Modelling Practice and Theory*, 15, pp. 1002–1015.

Lau, H. Y., Woo, S. O., 2008. An agent-based dynamic routing strategy for automated material handling systems. *International Journal of Computer Integrated Manufacturing*, 21(3), 269–288.

Lei, D., Zheng, Y., 2017. Hybrid flow shop scheduling with assembly operations and key objectives: A novel neighborhood search. *Applied Soft Computing*, 61, 122–128.

Li, M., Lei, D., Cai, J., 2019. Two-level imperialist competitive algorithm for energy-efficient hybrid flow shop scheduling problem with relative importance of objectives. *Swarm and Evolutionary Computation*, 49, 34–43.

Li, J., Pan, Q., 2015. Solving the large-scale hybrid flow shop scheduling problem with limited buffers by a hybrid artificial bee colony algorithm. *Information Sciences* 316, 487–502.

Lin, D., Lee, C. K. M., Ho, W., 2013a. Multi-level genetic algorithm for the resource-constrained re-entrant scheduling problem in the flow shop. *Engineering Applications of Artificial Intelligence*, 26, 1282–1290.

Lin, J. T., Huang, C. W., 2012. A novel vehicle pre-dispatching method for automated

material handling system in semiconductor manufacturing. In *Proc. of 2012 3rd International Asia Conference on Industrial Engineering and Management Innovation*, pp. 861–872.

Lin, J. T., Wu, C. H., Huang, C. W., 2013b. Dynamic vehicle allocation control for automated material handling system in semiconductor manufacturing. *Computers & Operations Research*, 40(10), 2329–2339.

Liu, X., Zou, F., Zhang, X., 2008. Mathematical model and genetic optimization for hybrid flow shop scheduling problem based on energy consumption. In *Proc. of IEEE Control and Decision Conference*, pp. 1002–1007.

Mahdavi, M., Fesanghary, M., Damangir, E., 2007. An improved harmony search algorithm for solving optimization problems. *Applied Mathematics and Computation*, 188(2), 1567–1579.

Marichelvam, M. K., Prabaharan, T., Yang, X. S., 2014. A discrete firefly algorithm for the multi-objective hybrid flowshop scheduling problems. *IEEE Transactions on Evolutionary Computation*, 18(2), 301–305.

Mohammadi, H., Sahraeian, R., 2012. Bi-objective simulated annealing and adaptive memory procedure approaches to solve a hybrid flow shop scheduling problem with unrelated parallel machines. In *Proc. of 2012 IEEE International Conference on Industrial Engineering and Engineering Management*, pp. 528–532.

Qian, B., Li, Z., Hu, R., 2017. A copula-based hybrid estimation of distribution algorithm for m-machine reentrant permutation flow-shop scheduling problem. *Applied Soft Computing* 61, 921–934.

Qiao, P., Sun, C., 2011. Research on hybrid flow-shop scheduling problem based on improved immune particle swarm optimization. In *Proc. of 2011 IEEE International Conference on Artificial Intelligence, Management Science and Electronic Commerce*, pp. 4240–4243.

Pan, Q.-K., Tasgetiren, M. F., Suganthan, P. N., Chua, T. J., 2011a. A discrete artificial bee colony algorithm for the lot-streaming flow shop scheduling problem. *Information Sciences* 181(12), 2455–2468.

Pan, Q.K., Wang, L., Gao, L., 2011b. A chaotic harmony search algorithm for the flow shop scheduling problem with limited buffers. *Applied Soft Computing*, 11, 5270–5280.

Rifai, A. P., Nguyen, H.-T., Dawal, S. Z. M., 2016. Multi-objective adaptive large neighborhood search for distributed reentrant permutation flow shop scheduling, *Applied Soft Computing* 40, 42–57.

Shahvari, O., Logendran, R., 2018. A comparison of two stage-based hybrid algorithms for a batch scheduling problem in hybrid flow shop with learning effect. *International Journal of Production Economics*, 195, 227–248.

Shen, J., Wang, L. Deng, J., Zheng, X., 2015. A Pareto-based discrete harmony search algorithm for bi-objective reentrant hybrid flowshop scheduling problem. In *Proc. of Proceedings of the 2nd International Conference on Harmony Search Algorithm (ICHSA2015)*, vol. 382 of Advances in Intelligent Systems and Computing, 435–445.

Shen, J., Wang, L., Zheng, H., 2016. A modified teaching–learning-based optimisation algorithm for bi-objective re-entrant hybrid flowshop scheduling. *International*

Journal of Production Research, 54(12), 3622-3639.

Tseng, L.-Y., Lin, Y.-T., 2010. A hybrid genetic algorithm for no-wait flowshop scheduling problem. *International Journal of Production Economics*, 128(1), 144–152.

Wang, M. Y., Sethi, S. P., van de Velde, S. L., 1997. Minimizing makespan in a class of reentrant shops. *Operations Research*, *45*(5), 702–712.

Wang, L., Pan, Q. K., Tasgetiren, M. F., 2011. A hybrid harmony search algorithm for the blocking permutation flow shop scheduling problem. *Computers & Industrial Engineering* 61, 76–83.

Wolpert, D. H., Macready, W. G., 1997. No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, 1(1), 67–82.

Wu, L. H., Mok, P. Y., Zhang, J., 2011. An adaptive multi-parameter based dispatching strategy for single-loop interbay material handling systems. *Computers in Industry*, 62(2), 175–186.

Xu, J., Yin, Y., Cheng, T. C. E., Wu, C. C., and Gu, S., 2014. A memetic algorithm for the re-entrant permutation flowshop scheduling problem to minimize the makespan. *Applied Soft Computing* 24, 277-283.

YAGEOmkt, 2013. YAGEO Chip Resistors Manufacturing Process. Available at: https://www.youtube.com/watch?v=wshRwO0MCSU

Yalaoui, N., Camara, M., Amodeo, L., Yalaoui, F., Mahdi, H. 2009. New heuristic for scheduling re-entrant production lines. In *Proc. of 2009 IEEE International Conference on Computers & Industrial Engineering*, pp. 199-204.

Yu, C., Semeraro, Q., Matta, A., 2018. A genetic algorithm for the hybrid flow shop scheduling with unrelated machines and machine eligibility. *Computers & Operations Research*, 100, 211–229.

Yuan, Y., Xu, H., Yang, J., 2013. A hybrid harmony search algorithm for the flexible job shop scheduling problem. *Applied Soft Computing*, 13, 3259–3272.

Yura, K., 1999. Cyclic scheduling for re-entrant manufacturing systems. *International Journal of Production Economics*, 60–61, 523–528.

Zhang, X. Y. and Chen, L., 2016. Heuristics for minimizing the total tardiness in a re-entrant hybrid flow shop with non-identical machines in parallel. In *Proc. of 2016 IEEE International Conference on Industrial Engineering and Engineering Management*, pp. 987-991.

Zhang, X. Y., Chen, L., 2017. A re-entrant hybrid flow shop scheduling problem with machine eligibility constraints. *International Journal of Production Research*, 56(16), 5293–5305.