MULTI-OPERATOR HYBRID GENETIC ALGORITHM-SIMULATED ANNEALING FOR REENTRANT PERMUTATION FLOW-SHOP SCHEDULING

Achmad Pratama Rifai [1a], Putri Adriani Kusumastuti [1b], Setyo Tri Windras Mara [1c], Rachmadi Norcahyo [1d], and Siti Zawiah Md Dawal [2]

[1] Department of Mechanical and Industrial Engineering, Faculty of Engineering, Universitas Gadjah Mada, Yogyakarta, Indonesia, Tel: 62-274-521673, e-mail: [1a] achmad.p.rifai@ugm.ac.id, [1b]putriadriani99@mail.ugm.ac.id, [1c] setyotriw@ugm.ac.id, [1d] rachmadinorcahyo@ugm.ac.id

[2] Department of Mechanical Engineering, Faculty of Engineering, University of Malaya, Kuala Lumpur, Malaysia, Tel: 60-3-7967-7625, e-mail: [2] sitizawiahmd@um.edu.my

Received Date: July 8, 2020; Revised Date: December 4, 2020; Acceptance Date: March 31, 2021

Abstract

This study develops an improved hybrid genetic algorithm-simulated annealing (IGASA) algorithm to solve the reentrant flow-shop scheduling problem with permutation characteristics. The reentrant permutation flow-shop (RPFS) allows the jobs to visit certain machines more than once and has been proven to be an NP-hard problem. The proposed improved hybrid algorithm integrates the simulated annealing (SA) and genetic algorithm (GA) to obtain the near-optimal solutions by considering three objectives: minimizing the makespan, the average completion time, and total tardiness. The multi-operator mechanism is proposed for the crossover and mutation operations to improve and maintain the diversity of individuals throughout the generation. The effectiveness and robustness of the proposed method are examined in the data sets of various-sized instances with different degrees of complexity. The results highlight that the proposed hybrid algorithm is a promising alternative in solving the RPFS scheduling problem.

Keywords: Genetic algorithm, Hybrid algorithm, Multiple operators, Reentrant permutation flow-shop, Simulated annealing.

Introduction

The main issue concerning scheduling in manufacturing systems is the allocation of resources to process several jobs within a given time frame. Scheduling problems arise when there is a need to manufacture a variety of products with limited resources (e.g., machines, equipment, and skilled labour) within a specific period. Flow-shop scheduling is one of the critical issues that need to be addressed since it is applied by various industries. The classical flow-shop scheduling assumes that all jobs have identical routes, or all jobs visit the same machines according to the same order [1]. However, this assumption is sometimes violated in real-world practice when the jobs need to visit certain machines several times [2].

Owing to the constraints of classical flow-shop scheduling, a modified flow-shop configuration known as the reentrant flow-shop (RFS) has gained much interest among researchers and engineers in recent years. In the RFS, even though the jobs have the same routing sequence, each job traverses the machines several times to complete all processes.

The RFS problems are typically encountered in the high–technology industry, such as semiconductor manufacturing [3]. For instance, in semiconductor wafer fabrication, each wafer is processed multiple times by the same machine and moves around the production line repeatedly as successive layers are added onto it [4]. RFS is also common in signal processing whereby the signals are pre-processed by a computer before passing through a sensing and command system for transmission and retrieval. The signals are then returned to the computer for post-processing [5].

A challenging task in RFS scheduling is the generation of optimum solutions. Previous studies have shown that the RFS scheduling with multiple jobs belongs to *NP*-hard class [6]. Hence, the aim of this study is to develop an improved hybrid algorithm by integrating GA and SA for solving the complex nature of RFS scheduling problems with permutation characteristics, called as the reentrant permutation flow-shop (RPFS). Three objectives are to be minimized within the RPFS: (1) maximum completion time (makespan), (2) average completion time, and (3) total tardiness. These objectives are crucial because they are representing the actual scenarios in the manufacturing industry [7]. This study proposes a multi-operator mechanism for the hybrid algorithm. This mechanism allows the selection of a set of operators for both crossover and mutation operations. Hence, the various operators complement each other to maintain population diversity and avoid premature convergence.

Related Works

The RFS concept was first coined by Graves et al. [8], in which the jobs may revisit certain machines along the production line. The concept bears a resemblance to the classical flow-shop scheduling. However, the primary difference between them is that in the RFS, the jobs will be processed at the machine multiple times, which implies the creation of loops in the processing sequence. Although RFS scheduling has been introduced for over three decades, the problems pertaining to RFS scheduling have only been studied extensively since the last decade due to the critical need to increase the performance of the production line in the fabrication of integrated circuits of semiconductor industry.

The performance of RFS scheduling can be evaluated through the optimal goal of system operation. Most of the previous studies related to the flow-shop problems are concentrated on the single-criterion scheduling [7], commonly set to minimize the makespan C_{max}. This criterion is equivalent to the maximization of productivity because the makespan has a strong correlation with the throughput rate [9]. Several published studies have discussed the objective of minimizing the makespan in scheduling problems. Choi and Kim [10], for instance, focused on the development of several heuristic methods to solve the RFS problems with this criterion of various sizes. The computational time was also used as the objective function in their work since it represents the efficiency of the heuristic methods. On the other hand, Yang et al. [11] aimed to minimize the makespan time of a two-machine multi-family scheduling problem with reentrant production flows, in which a bridge construction problem was deployed as the basis to develop the formulation of the RFS problem. The term ‘multi-family’ was defined based on the scenario where the same family contains several identical jobs and the jobs within the same family were processed in sequence. The setup time was defined in their study as the time when the machines began processing jobs from different families.

Other parameters have also been used as the objectives. Demirkol and Uzsoy [12] discussed how the maximum lateness of operations could be minimized using a decomposition method, whereas Choi and Kim [13] concentrated on minimizing the tardiness in a two-machine RFS. Their work was extended by Jeong and Kim [14] who discussed on minimizing the total tardiness time within the issue of sequence-dependent setup. Kang et al. [15] proposed development on the objectives into the total weighted tardiness as an alternative formulation of tardiness. Then, the authors proposed the Revised Apparent Tardiness Cost with Setup (RATCS) to solve a scheduling problem involving both due date and sequence-dependent setup time. Kaihara et al. [16] built a model to generate a schedule that accommodating the proper proactive maintenance activities and minimizing the overall tardiness.

Previous studies have indicated that the implementation of single-criterion optimization is inadequate to represent the real-world scenarios. As noted by Sun et al. [7], multiple and conflicting objectives are typically emerged in the scheduling problems that arise in the manufacturing industry. Hence, over the years, the research community has attempted to develop models for scheduling problems with multiple objectives to address the previously mentioned situation. A model with multiple objectives to maximize the utilization rate and minimize the mean cycle time of a bottleneck facility has been proposed by Dugardin et al. [17]. Choi et al. [18] proposed a model that minimizes the makespan and the maximum allowable due date in the form of constraint sets. However, the authors did not optimize the objectives simultaneously since the makespan was set as the main priority as long as the results satisfied the due date constraint. Further, other previous studies are mainly centred on minimizing the makespan and tardiness [19-20], since reaching the optimum state for those objectives will enhance the productivity of the production line in order to satisfy the demand of customers.

Since RFS scheduling belongs to the class of NP-hard problems, various solution methods have been proposed, ranging from exact methods to heuristic and metaheuristic methods. The branch and bound method is commonly used for solving scheduling problems, and it was adopted by Choi et al. [18] along with other heuristic methods such as modified Nawaz, Enscore, and Ham (NEH) and Johnson, Campbell, Dudek and Smith (CDS) algorithms. Jeong and Kim [14] also integrated the branch and bound method with other heuristic methods, in which they developed two types of heuristics: (1) a list of scheduling algorithms consisting of the earliest due date (EDD), modified EDD, and modified due date rule, and (2) constructive algorithms consisting of modified NEH and Framinan and Leisten (FL) algorithm. Meanwhile, Jing et al. [21] implemented a different heuristic approach, which integrates a modified lower bound-based algorithm and a weighted idle time-based algorithm.

A variety of metaheuristic methods have been developed in recent years to solve the complexity of RFS scheduling problems. Dugardin et al. [17] developed a modified GA called Lorenz Non-Dominated Sorting Genetic Algorithm (L-NSGA) by incorporating the Lorenz dominance relationship into the GA process. Cho et al. [19] proposed a novel method named as the Minkowski distance-based Pareto GA with local search strategy (MLPGA). Three distinctive modifications were made: (1) Pareto for ranking selection, (2) Minkowski distance-based for the crossover, and (3) the addition of local search to exploit the capability of climbing upwards from a local optimum. Chamnanlor et al. [22] developed a hybrid GA

for solving RFS in the hard-disk manufacturing system by considering time window constraint. Two algorithms were added to GA: time window satisfied routine for checking and repairing the time window constraint and left shift routine for improving the offspring. The study for the same problem was extended by embedding ant colony optimization (ACO) into GA [23]. In addition, the Fuzzy Logic Controller was added to adjust the average fitness of the current and last generation for balancing the probabilities of crossover and mutation rates. Chen et al. [24] developed a hybrid tabu search method to prevent the local optimum trap. The hybrid method was introduced to explore new solution regions in situations where the best solution cannot be attained after many iterations. Fattahi et al. [25] developed a hybrid algorithm based on SA to solve a reentrant manufacturing system scheduling problem. The proposed algorithm was compared to the conventional GA, and single job approaches are used to measure the performance of all algorithms. Huang et al. [6] developed the Farness Particle Swarm Optimization (FPSO) method to solve a reentrant two-stage multiprocessor flow-shop scheduling problem, and the method was compared with the classical PSO and ACO to assess its performance. Then, Rifai et al. [26] developed an adaptive large neighbourhood search for multi-objective distributed RPFS scheduling. The study considers multiple production lines with multiple production centres in which the proposed algorithm also determined the allocation of jobs to factories. Recent studies show that metaheuristics are still the preferred methods to solve the complex problem of the RFS. Recently, various metaheuristics have been developed, such as iterated greedy algorithm [27], Improved Multi-Verse Optimizer Algorithm [28]. The latest studies also indicate that the hybrid method gains more concerns, as found in the study by Amrouche et al. [29] which combined a simulated annealing and lower bounds, and Lin et al. [30] which proposed a hybrid harmony search and genetic algorithm.

Nevertheless, the increasing complexity of RPFS problems requires the improvement of the optimization method and several studies have addressed the issue by developing hybrid methods. However, most studies are focused on the integration of metaheuristic and local search methods. A lot of their proposed methods still lack studies that focus on the integration of metaheuristic methods in RFS scheduling problems. It is known that different metaheuristic approaches will have different characteristics and learning capabilities, and therefore it is worthy to develop a hybrid algorithm that will exploit the benefits of existing methods [31]. In line with the above motivation, the main objective of this study is to develop a hybrid algorithm that integrates GA and SA to optimize the RFS scheduling problems with permutation characteristics. In addition, this study proposes multiple operators for the crossover and mutation to improve the quality and diversity of the solutions.

Problem Formulation

Globalization and an increase of competitiveness have motivated manufacturing companies to improve the facilities to respond to the market requirements [32]. In high-tech industries, those improvements are manifested as the development of RFS, which is applied for semiconductor manufacturing. This research studies the RFS scheduling problem with permutation characteristics, which is also known as reentrant permutation flow-shop (RPFS). There is a total of n jobs $J = j_1, j, \dots, j_n$ which will be processed by a set of machines $M = m_1, m_2, \dots, m_{|M|}$. Each job must be processed by each machine and the sequence of the jobs on $|M|$ machines is the same. In the reentrant scheme, the jobs will visit the

machines more than once, which is labelled as layer l. Thus, the number of layers indicates the number of times that the jobs visit the same machines. Operation o represents the processing of a job on a machine that requires the processing time ρ_{jml}. The physical configuration of the reentrant flow-shop scheduling problem is shown in Figure 1.

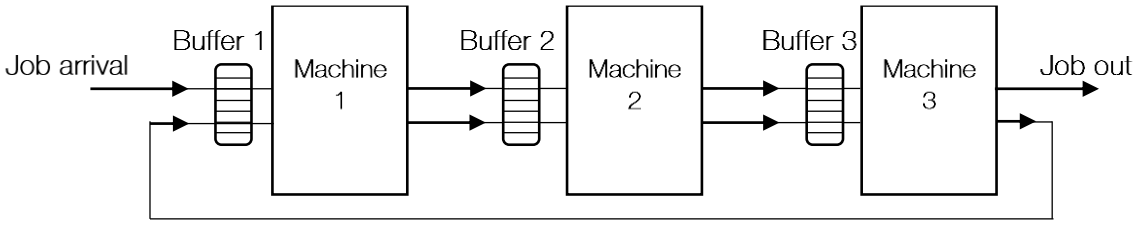

Figure 1. Physical configuration of the reentrant flow-shop scheduling problem

Since only one job can be processed by a machine at a time, the next parts will be temporarily located in a buffer, which is situated just before the machine. The usage of the buffer will lead to a delay time that represents the duration time it takes for a part to be processed by the machine. In order to prepare the readiness of a machine m prior to processing a new job j in layer l, a setup time φ_{jml} is incurred and the transportation time between machines is represented by τ_m.

There are three objectives to be minimized: (1) makespan C_{max}, (2) the average completion times $\bar{C}$ and (3) the total tardiness T. These objectives are expressed respectively in Equations (1), (2), and (3) as follows:

Minimize
$$C_{max} = \max_{o,j,m,l}(C_{ojml}) \tag{1}$$

$$\bar{C} = \frac{1}{n}\sum_{j=1}^{n} \max_{o,m,l}(C_{ojml}) \tag{2}$$

$$T = \sum_{j=1}^{n} \max\left\{\max_{o,m,l}(C_{ojml}) - td_j, 0\right\} \tag{3}$$

where C_{ojml} represents the completion time for operation o of job j on machine m at layer l in an integer value, while td_j represents the due date of job j. The total fitness value is the sum of all objectives with equal proportions and is subjected to the following constraints:

$$St_{ojml} \geq 0, \qquad \forall\, o \in O, j \in J, m \in M, l \in L \tag{4}$$

$$C_{ojml} = St_{ojml} + \rho_{jml} + \phi_{jml}, \qquad \forall\, o \in O, j \in J, m \in M, l \in L \tag{5}$$

$$C_{ojml} \geq 0, \qquad \forall\, o \in O, j \in J, m \in M, l \in L \tag{6}$$

$$C_{ojml} \leq St_{o(j+1)ml}, \qquad \forall\, m \in M, l \in L \tag{7}$$

$$St_{ojml} \geq C_{(o-1)jml} + \tau_m, \qquad \forall\, m \in M, l \in L \tag{8}$$

$$\alpha_{ojml} \in [0,1], \qquad \forall\, o \in O, j \in J, m \in M, l \in L \tag{9}$$

$$\sum_{j=1}^{n} \sum_{l=1}^{L} \sum_{o=1}^{O} \alpha_{ojml} = 1, \qquad \forall\, m \in M, t \tag{10}$$

Equation (4) describes that each operation starts in $t = 0$. Equation (5) shows that for operation o, the completion time of job j is equal to the sum of the starting time, setup time, and processing time. The completion time of operation o of all jobs, machines, and layers must be in the value of non-zero as expressed in Equation (6). The basic requirement that the starting time of job $j + 1$ on machine m must at least be the same or greater than the completion time of its predecessor job on machine m is ensured by Equation (7), while Equation (8) assures that the operation o of job j has a greater value of starting time than the sum of completion time of the operation $o - 1$ (predecessor operation) of the same job and its corresponding transportation time. Equation (9) represents the binary decision variable α_{ojml}, and finally, Equation (10) ensures that a machine can perform only one job at a given time.

Solution Algorithm

The use of conventional techniques is generally unable to attain optimum solutions in complex problem instances in an efficient manner [33]. As a result, hybrid metaheuristics are popular alternatives to solve complex problem by combining the characteristic of each constituent method. In this study, a hybrid genetic algorithm-simulated annealing is developed to minimize the objective functions in RPFS scheduling problems by exploiting the benefits of both GA and SA. The developed hybrid algorithm involves two major phases: (1) the GA and (2) the SA. In the first phase, the GA operators such as parent selection, crossover, and mutation are deployed to generate and modify the individuals in the population. Generally, the traditional GA employs only a single operator, each for the crossover and mutation operations. However, it is often observed that the use of a single operator crossover is susceptible to premature convergence. Hence, maintaining population diversity throughout generations is key to avoid premature convergence [34]. This study develops a set of crossover operators, each having a uniform probability of being selected for producing the new solutions. This multi-operator aims to reduce redundant new solutions, which are usually generated using single crossover and fitness ranking for parent selection.

After the new population generation, the GA dispatches the newly generated offspring after each generation, which will be used as the inputs of SA for further improvement. In the second phase, local search methods of SA are employed for the mutation process, thus creating a multi-operator mutation. Further, the Metropolis criterion of the SA is used to determine whether the less-fitted generated solutions can be indicted for

the next population, subject to the current temperature. Therefore, in the early generation, when the temperature high, the acceptance probability of poor solution is high to allows the method to search for a wider neighbourhood. As the temperature cooling, the acceptance probability reduces and the algorithm switch to focus on the exploitation of promising neighborhood. The process continues until the termination criterion is met. The procedure of the proposed improved hybrid genetic algorithm-simulated annealing (IGASA) is shown in Algorithm 1, which further illustrates the relative components of the two phases as well as the decision-making process.

Algorithm 1: Procedure of the IGASA

input: an instance (n, m, l), GA-SA parameters $(T_1, T_0, \alpha, K, Pc, Pm)$
initialize a set of feasible solutions $\acute{\varepsilon} \in s$;
calculate fitness of each solution $f(\acute{\varepsilon})$;
while $T_t > T_0$ **do**
  for each chromosome $\acute{\varepsilon} \in s$;
    select parent $P_1, P_2 \in s$;
    select a crossover operator $\psi \in C$;
    perform (ψ, Pc) to yield O_1, O_2
    set $\acute{\varepsilon} \leftarrow O$;
    select a mutation operator $\varphi \in Mt$;
    perform (φ, Pm) to yield $\acute{\varepsilon}'$
    calculate fitness of each offspring $f(\acute{\varepsilon}')$;
    If acceptance criteria are fulfilled
      insert $\acute{\varepsilon}$ to s
    end if
  end for
  update $T_t \leftarrow \alpha T_{t-1}$;
end while
output: $s, f(s)$

Solution Representation

The proposed algorithm begins with the solution representation. Within a single chromosome, the decision variable is represented as a permutation of integers, and each corresponds to the sequence of parts (or jobs) which will be processed by the system and needs to be optimally scheduled. The representation of chromosomes are shown in Figure 2. Since the chromosomes are integer permutation, the allele values are always unique among individuals, which curbs redundancy.

	j_1	j_2	j_3	j_4	j_5	j_6	j_7	j_8	j_9	j_{10}
Seq.	7	4	8	5	9	3	1	2	6	10

Figure 2. Integer permutation chromosomes

The initial population of solutions is randomly generated. Then, the classical roulette wheel selection is employed to select the parents for each generation. In this selection method, each individual i holds a probability p_i to be selected as a parent. The value is dependent on its corresponding fitness value f_i and is represented mathematically in Equation (11).

$$p_i = f_i / \sum_{i=1}^{n=1} f_i \tag{11}$$

The chromosome i is passed on to the next generation when the inequality $\sum_{x=0}^{i-1} p_x < \gamma \leq \sum_{x=0}^{i} p_x$ is satisfied. The ranking scheme is used to sort the population based on the fitness value of each individual, ensuring that individuals with higher fitness values have higher possibilities of being selected as parents.

Multi-Operator Crossover

The multi-operators GA has been developed by previous literatures for a wide array of optimization problems, such as routing problems [35] and traveling salesman problem [36]. In this study, the use of multi-operators of crossover and mutation is extended for the hybrid GA-SA algorithm. Specifically, for the RPFS problem with permutation solution representation, a set of crossover operators is developed. The set consists of six operations, as described in Figure 3.

Parent	P_1	7 4 8 5 9 |3 1 2 6	P_1	7 4 8 |5 9 3 1| 2 6	P_1	7 4 8 |5 9 3 1| 2 6
	P_2	9 2 5 1 8 |7 6 4 3	P_2	9 2 5 |1 8 7 6| 4 3	P_2	9 2 5 |1 8 7 6| 4 3
Offspring	O_1	**1 2** 8 5 9 |7 6 4 3	O_1	**5** 4 **9** |1 8 7 6| 2 **3**	O_1	4 2 8 |5 9 3 1| 7 6
	O_2	9 **7** 5 **4** 8 |3 1 2 6	O_2	**8** 2 **7** |5 9 3 1| 4 **6**	O_2	2 4 5 |1 8 7 6| 9 3
	(a) One-point		(b) PMX		(c) OX			
Parent	P_1	7 4 8 5 9 |3 1 2 6	P_1	7 4 8 |5 9 3 1| 2 6	P_1	7 4 8 |5 9 3 1| 2 6
	P_2	9 2 5 1 8 |7 6 4 3	P_2	9 2 5 |1 8 7 6| 4 3	P_2	9 2 5 |1 8 7 6| 4 3
Offspring	O_1	**1 2** 8 5 9 |3 4 6 7	O_1	**5** 4 **9** |6 7 8 1| 2 **3**	O_1	6 7 8 |5 9 3 1| 2 4
	O_2	9 **7** 5 **4** 8 |6 2 1 3	O_2	**8** 2 **7** |1 3 9 5| 4 **6**	O_2	3 9 5 |1 8 7 6| 4 2
	(d) reverse one-point		(e) reverse PMX		(f) reverse OX			

Figure 3. Crossover operators

The cutting points are marked with a vertical bar sign (|). The swapped parts of chromosome are highlighted in the coloured section, while the adjusted parts are highlighted with bold font. The one-point crossover simply randomly select a cutting point and exchange the subsequent parts between two parent chromosomes. The partially matched crossover (PMX) operator splits the chromosomes into three sections and then exchanges the parts between the cutting points from P_1 to O_2 and P_2 to O_1. The order crossover (OX) builds offspring by preserving the parts in-between the cutting points, and exchanging the parts outside. To fulfil the exchanged parts of P_1, the parts of P_2 after the second cut points are selected, i.e. $4 \rightarrow 3 \rightarrow 9 \rightarrow 2 \rightarrow 5 \rightarrow 1 \rightarrow 8 \rightarrow 7 \rightarrow 6$. Since the middle section is preserved, the inserted parts become $4 \rightarrow 2 \rightarrow 8 \rightarrow 7 \rightarrow 6$, then is mapped to O_1 from the first bit. The reverse one-point, PMX, and OX operators follow the same procedure of each operator, but the swapped parts are flipped before exchange, i.e. $5 \rightarrow 9 \rightarrow 3 \rightarrow 1$ to $1 \rightarrow 3 \rightarrow 9 \rightarrow 5$ for the reverse PMX example, and $4 \rightarrow 2 \rightarrow 8 \rightarrow 7 \rightarrow 6$ to $6 \rightarrow 7 \rightarrow 8 \rightarrow 2 \rightarrow 4$ for the reverse OX example.

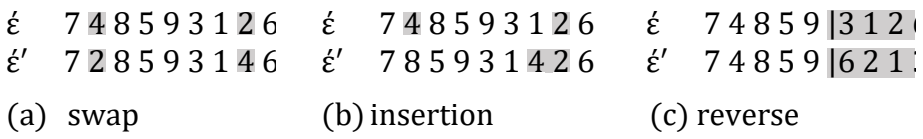

Figure 4. Mutation operators

Multi-Operator Mutation

The implementation of crossover and roulette wheel selection techniques may reduce the variability of chromosomes within the population since the better-fitted solutions tend dominating the population. Therefore, mutation operators are used to enhancing the variability and prevent the same chromosome values. Here, the searching procedure of SA is adapted for the non-uniform mutation techniques. Three operators are used, as illustrated in Figure 4, thus creating a multi-operator mutation process.

The SA process begins with an initial solution $\acute{\epsilon}$ that belongs to solution space Ω. A neighbouring solution $\acute{\epsilon}'$ is modified using some pre-defined rule from the neighbourhood function $N(\acute{\epsilon})$. The first function is swap mutation which starts with randomly selecting two genes, then their positions are exchanged. The insertion is performed by randomly select two points, then the first selected gen is inserted just before the second point, or vice versa. The reverse mutation follows a similar procedure as a one-point crossover where the genes after the cutting point are flipped. However, it is performed within its own genes and not exchanged with other chromosomes. The allele value of a gene in a chromosome can be changed when the value of a uniformly distributed random number $U(0,1)$ is less than the mutation probability Pm. Otherwise, the allele value will remain its initial value.

Acceptance Criteria and Population Update

The last section of the algorithm is the new solution acceptance and population update procedure which utilizes the annealing characteristic of the SA. Annealing refers to a process of altering the internal metal structure by heating and cooling to change its physical properties. The new structure becomes fixed as the metal cools, and this process enables the metal to preserve its newly obtained properties. Several parameters are considered in which their values crucially influence the performance of SA. Therefore, they should be carefully selected. Within the framework of SA, the parameters are the initial temperature, the cooling rate of the temperature, the number of iterations at each temperature, and the termination criterion of the algorithm.

At high temperatures, there is a high probability that the SA can accede a new state which has higher energy compared to that of the previous state. The probability that the state is accepted decreases as cooling proceeds. The neighbourhood solution space becomes small eventually, which finally converges to a global optimum or near-optimum solution. However, the initial temperature is a trade-off variable, whereby a high initial temperature results in extensive computational time whereas a low initial temperature exempts the possibility of ascending steps which in turn misses the global minimum solution.

The cooling rate is also a trade-off variable besides the initial temperature and has a substantial effect on SA performance. However, the cooling rate has the opposite effect compared to the initial temperature. A low cooling rate results in extensive computational time, which slows down the alteration process. Vice versa, a high cooling rate reduces the processing time significantly but leads to the possibility of getting stuck in a local minimum. Hence, a suitable cooling rate should be chosen to ensure the reliability and efficiency of the searching process of SA. The next iteration temperature is determined by using the cooling schedule given by Equation (12).

$$T_t = \alpha T_{t-1} \tag{12}$$

where T_t and T_{t-1} represent the temperatures at t and $t-1$ respectively, and α represents the cooling rate with a value within [0, 1]. The number of iterations at each temperature L_t is defined as follow:

$$L_t = \beta_t \tag{13}$$

where β is a constant variable. The acceptance procedure is built based on the Metropolis criterion. The candidate solution $\acute{\varepsilon}'$ is taken as the current solution $\acute{\varepsilon}$ based on the probability P as given in Equation (14).

$$P = \begin{cases} exp[-(f(\acute{\varepsilon}') - f(\acute{\varepsilon}))/t_k] & if\ f(\acute{\varepsilon}') - f(\acute{\varepsilon}) > 0 \\ 1 & if\ f(\acute{\varepsilon}') - f(\acute{\varepsilon}) \leq 0 \end{cases} \tag{14}$$

where $f(\acute{\varepsilon})$ is the energy value and t_k is the temperature at iteration k. If the temperature is reduced sufficiently slowly, the system will reach equilibrium (steady-state condition) at each iteration k. The probability function P must be positive even if $f(\acute{\varepsilon}')$ is greater than $f(\acute{\varepsilon})$. This criterion ensures that the search space to be explored is large at high temperatures. Likewise, the search space of the solutions shrinks with a decrease in temperature, as the algorithm opts for the exploitation of promising neighborhood. The accepted solutions are then sent back to GA phase as a new generation. The iteration process continues until the termination criterion is met, i.e., the solution reaches the global optimum, or the iteration exceeds the pre-determined maximum number of generations, which is determined by the final value of temperature.

Computational Results

Experimental Design

Several test problems are generated to measure the performance of the proposed algorithm. The problem sets consist of the number of machines $|M|$, number of jobs n, number of layers $|L|$, due date td, processing time ρ, setup time φ, and transportation time τ. The setup time is uniformly distributed within the interval $\{1,50\}$ whereas the processing time is uniformly distributed within the interval $\{1,50\}$. The values of all parameters used in the problem instances are based on data from real manufacturing [14]. The tardiness factor Ω and due date range R are used to generate the due dates which are uniformly distributed within the interval $\{X(1-\Omega-R/2), X(1-\Omega+R/2)\}$ [26]. X represents the lower bound of the makespan which is obtained from trial scheduling. The problem instances consist of twenty-

one datasets, classified into three categories based on their size. A total of ten instances are generated for each dataset. This procedure was done to ensure the randomness and to compare the robustness of the methods. The performance of the proposed IGASA is compared against benchmark methods of standard SA, GA, and hybrid GA-SA.

Analysis of Performance

A comparative study is performed to determine the performance of the proposed IGASA. The improvement rate IR is used to evaluate the performance of the proposed algorithm and it indicates the effectiveness of the IGASA relative to other methods. The IR of the proposed algorithm over algorithm A is defined in Equation (15). In addition, the relative deviation index RDI is used to measure the relative performance of the proposed method as compared to the benchmark methods. The calculation of RDI is proposed by Choi and Kim [13], as described in Equation (16).

$$IR = \frac{f(A) - f(IGASA)}{f(A)} \times 100\% \tag{15}$$

$$RDI_Y = \frac{f(Y) - f_b}{f_w - f_b} \tag{16}$$

where $f(hGASA)$ represents the fitness obtained from the proposed algorithm whereas $f(A)$ represents the obtained fitness value from the comparison method A. The improvement rate is computed on the best fitness f_{min} and average fitness $\bar{f}$ due to the stochastic nature of the methods involved. The RDI_Y represents the relative deviation index for algorithm Y, $f(Y)$ represents the fitness of algorithm value from algorithm Y, while f_b and f_w respectively describe the best and worst objective function value. The results of the proposed IGASA and benchmark methods in small, medium, and large size problems are summarized in Table 1. The results of IR and RDI test are described in Table 2. The obtained average IR both for the best fitness f_{min} and average fitness $\bar{f}$ shows the superiority of IGASA as compared to benchmark methods, indicated by positive IR. The average IR based on the best fitness f_{min} are 3.3%, 0.1%, and 0.3%, while the average IR based on the average fitness $\bar{f}$ are 5.7%, 0.4%, and 0.8% against SA, GA, and hybrid GA-SA, respectively. In general, the IR in instances 1-10 is zero since all methods can obtain the optimal solutions. As the number of machines $|M|$, number of jobs n, number of layers $|L|$ increase, there is tendency that the IR of IGASA is increasing as well. This indicates the effectiveness of the proposed IGASA in finding better solutions in complex instances.

Figure 5 presents the boxplot of average RDI of the conventional GA, hybrid GA-SA, and IGASA. The RDI of SA is significantly much larger than other methods in all instances, which indicates that it perform the worst. Especially in small instances, the RDI of GA, hybrid GA-SA, and IGASA are low since the three methods can obtain the optimal solution while the SA often fails. In general, the IGASA scores improvement over the conventional SA, GA, and hybrid GA-SA in the instance with various complexity. The ability to explore the wider neighbourhood of the solution space and varying its population is vital to improve the effectiveness of the algorithm during the searching process since the methods have similar characteristics and are based on the population. It is evident from the

results that the IGASA yields a significant improvement rate over traditional methods, which indicates that GA and hybrid GA-SA is rather weak in exploring the search space and maintaining the population diversity due to a single operator of crossover and mutation.

Robustness Test

Due to its stochastic nature, robustness is an essential criterion in assessing the stochastic searching heuristics. Robustness ensures the stability of the algorithm when it encounters different types of data. The robustness improvement ratio (RIR) of the IGASA relative to the benchmark method A is defined in Equation (17). The metric is used to examine the stability of the proposed algorithm in delivering optimal solutions over ten repetitions. The results of the robustness test are presented in Table 2.

$$RIR = 1 - \frac{\sigma^2(hGASA)}{\sigma^2(A)} \times 100\% \tag{17}$$

It can be observed from the results that the proposed IGASA outperforms other methods in terms of robustness with average RIR at 79.75%, 42.41%, 43.26% as compared to the SA, GA, and hybrid GA-SA, respectively. In addition, there is a correlation between the RIR and the complexity of the problem. The problem complexity increases exponentially with an increase in the combination of number of jobs, machines, and layers, which in turn, significantly widen the search space. The results indicate that the GA often fails to vary its population and consequently becomes trapped in the local optima. In contrast, the SA focuses on exploring other neighbourhoods to improve the current solution, which will sometimes alter the current neighbourhoods radically. Hence, it lacks the ability of neighborhood exploitation as compared to population-based algorithms. Meanwhile, the hybrid algorithm overcomes the common pitfalls of both GA and SA by integrating the characteristics of both methods to enhance the strength of the algorithm. However, although the hybrid GA-SA also combines the benefit of both GA and SA, it is still susceptible to premature convergence. In contrast, the use of multi-operator significantly increases IGASA to maintain the diversity of the population. Thus, the IGASA method is more stable and robust in generating solutions, as demonstrated by the robustness test.

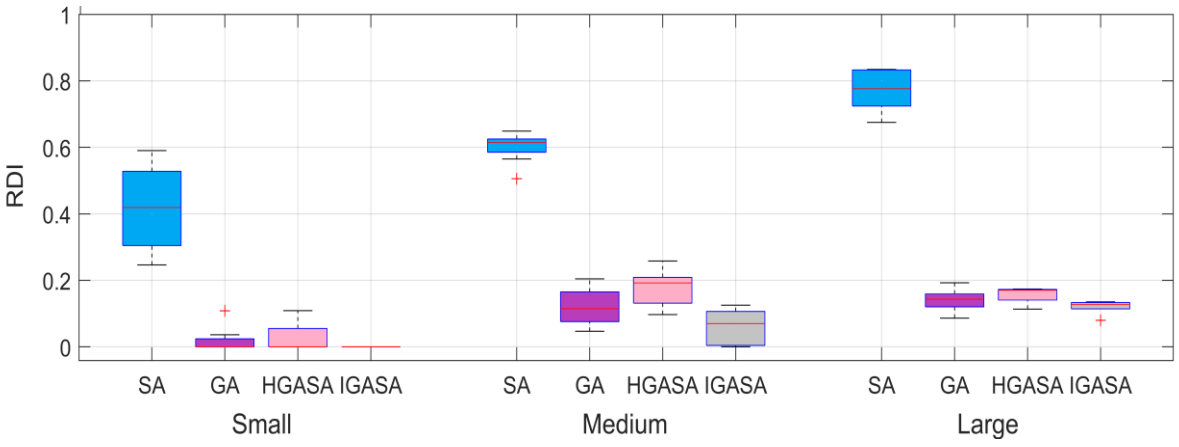

Figure 5. The boxplot of average RDI

Table 1. Results of Computational Experiments

Instance	$n, \|M\|, \|L\|$	no. of sub-jobs	Best fitness f_{min}				Average fitness $\bar{f}$				Standard deviation σ			
			SA	GA	HGASA	IGASA	SA	GA	HGASA	IGASA	SA	GA	HGASA	IGASA
1	3,3,3	27	3188	3188	3188	3188	3191	3188	3188	3188	4.41	0.00	0.00	0.00
2	3,3,4	36	3628	3628	3628	3628	3677	3628	3628	3628	78.60	0.00	0.00	0.00
3	3,4,3	36	2573	2573	2573	2573	2593	2573	2573	2573	30.34	0.00	0.00	0.00
4	4,3,3	36	3857	3857	3857	3857	4018	3861	3857	3857	114.78	10.73	0.00	0.00
5	4,4,3	48	4205	4205	4205	4205	4253	4205	4205	4205	34.62	0.00	0.00	0.00
6	4,5,3	60	5415	5415	5415	5415	5461	5429	5424	5415	40.40	21.31	18.60	0.00
7	4,4,4	64	6230	6230	6230	6230	6475	6230	6246	6230	164.75	0.00	31.30	0.00
8	4,5,4	80	6963	6963	6963	6963	7194	6982	7020	6963	177.27	38.30	46.91	0.00
9	6,6,2	72	5119	5105	5105	5105	5427	5135	5237	5110	196.80	46.62	53.39	6.70
10	6,8,5	240	15183	15183	15183	15183	16017	15330	15340	15183	447.98	120.26	186.43	0.00
11	6,9,3	162	8814	8674	8674	8674	9101	8710	8783	8674	192.25	72.40	101.77	0.00
12	7,7,5	245	17704	17457	17457	17457	18474	17651	17613	17588	395.79	174.01	152.54	121.07
13	7,8,4	224	13008	12600	12727	12600	13626	12852	12948	12797	389.09	104.22	141.16	116.37
14	8,8,3	192	11076	10772	10772	10772	11556	11032	11036	10847	312.28	152.59	204.13	114.76
15	9,9,2	162	8161	7563	7767	7563	8549	7834	7955	7754	305.37	142.82	162.04	136.25
16	10,10,2	200	9090	8348	8381	8369	9726	8596	8751	8561	419.62	271.28	269.22	108.50
17	12,12,10	1440	39000	36934	37077	36911	39895	37762	37680	37496	665.52	478.22	472.77	332.59
18	15,15,5	1125	24256	21797	21875	21868	25038	22395	22508	22322	525.24	313.88	521.91	233.73
19	20,20,4	1600	29627	25578	25652	25335	31020	26232	26509	26200	806.80	529.95	634.36	376.17
20	25,25,8	5000	98414	91432	91349	91043	100681	92756	92774	92609	1372.95	1038.15	1182.58	883.08
21	30,30,5	4500	85878	79893	79734	79609	89138	80718	81068	80635	1872.86	883.37	1042.05	911.69

Table 2. Results of IR, RDI, and RIR Test

Instance	IR based on f_{min}			IR based on $\bar{f}$			RDI				RIR		
	vs. SA	vs. GA	vs. HGASA	vs. SA	vs. GA	vs. HGASA	SA	GA	HGASA	IGASA	vs. SA	vs. GA	vs. HGASA
1	0	0	0	0.11%	0	0	0.400	0	0	0	100%	0%	0%
2	0	0	0	1.32%	0	0	0.249	0	0	0	100%	0%	0%
3	0	0	0	0.75%	0	0	0.246	0	0	0	100%	0%	0%
4	0	0	0	4.00%	0.09%	0	0.528	0.012	0	0	100%	100%	0%
5	0	0	0	1.14%	0	0	0.527	0	0	0	100%	100%	0%
6	0	0	0	0.85%	0.26%	0.17%	0.361	0.109	0.072	0	100%	100%	100%
7	0	0	0	3.78%	0	0.25%	0.590	0	0.038	0	100%	0%	100%
8	0	0	0	3.22%	0.27%	0.82%	0.438	0.036	0.109	0	100%	100%	100%
9	0.27%	0	0	5.84%	0.47%	2.41%	0.506	0.047	0.207	0.009	96.6%	85.6%	87.5%
10	0	0	0	5.21%	0.96%	1.03%	0.565	0.100	0.107	0	100%	100%	100%
11	1.58%	0	0	4.69%	0.42%	1.24%	0.613	0.052	0.156	0	100%	100%	100%
12	1.40%	0	0	4.80%	0.36%	0.14%	0.630	0.120	0.097	0.082	69.4%	30.4%	20.6%
13	3.13%	0	1.00%	6.08%	0.43%	1.16%	0.619	0.152	0.210	0.119	70.1%	-11.7%	17.6%
14	2.74%	0	0	6.14%	1.68%	1.72%	0.616	0.204	0.207	0.058	63.3%	24.8%	43.8%
15	7.32%	0	2.62%	9.30%	1.02%	2.54%	0.649	0.178	0.258	0.125	55.4%	4.6%	15.9%
16	7.93%	-0.25%	0.14%	11.98%	0.40%	2.17%	0.606	0.109	0.177	0.094	74.1%	60.0%	59.7%
17	5.36%	0.06%	0.45%	6.01%	0.70%	0.49%	0.675	0.193	0.174	0.132	50.0%	30.5%	29.6%
18	9.84%	-0.33%	0.03%	10.85%	0.33%	0.83%	0.776	0.143	0.170	0.126	55.5%	25.5%	55.2%
19	14.48%	0.95%	1.23%	15.54%	0.12%	1.17%	0.834	0.132	0.172	0.127	53.4%	29.0%	40.7%
20	7.49%	0.43%	0.33%	8.02%	0.16%	0.18%	0.832	0.148	0.149	0.135	35.7%	14.9%	25.3%
21	7.30%	0.36%	0.16%	9.54%	0.10%	0.53%	0.741	0.086	0.113	0.080	51.3%	-3.2%	12.5%

Table 3. The Paired Sample T-Test Between the IGASA and Benchmark Methods

Criteria	Benchmark methods					
	SA		GA		HGASA	
	p-value	description	p-value	description	p-value	description
$\Delta_{f_{min}}$	0.0027	H_0 is rejected, IGASA performs better	0.2908	Fail to reject H_0	0.0553	Fail to reject H_0
$\Delta_{\bar{f}}$	9.2×10^{-5}	H_0 is rejected, IGASA performs better	0.3072	Fail to reject H_0	0.0681	Fail to reject H_0
RDI	4.6×10^{-13}	H_0 is rejected, IGASA performs better	0.0423	H_0 is rejected, IGASA performs better	0.0064	H_0 is rejected, NSGA-II performs better
T	0.0470	H_0 is rejected, SA is significantly faster	0.4941	Fail to reject H_0	0.8526	Fail to reject H_0

Statistical Analysis

Hypothesis tests were performed to demonstrate the significance of the improvement obtained by the proposed IGASA as compared to the benchmark methods. A two-tailed t-test is performed with the following hypotheses.

$$H_0: \mu_{IGASA}^{\gamma} = \mu_A^{\gamma} \tag{18}$$

$$H_1: \mu_{IGASA}^{\gamma} \neq \mu_A^{\gamma} \tag{19}$$

where μ_{IGASA}^{γ} is the mean obtained by the IGASA, and μ_A stands for the mean of the benchmark algorithm A on criterion γ. The test is performed for four criteria: the $\Delta_{f_{min}}$, $\Delta_{\bar{f}}$, RDI, and computational time T. The $\Delta_{f_{min}}$, and $\Delta_{\bar{f}}$ criteria which represents the deviation of the obtained best fitness toward the best-known fitness f_{min}^* are calculated as follow.

$$\Delta_{f_{min}} = (f_{min} - f_{min}^*)/f_{min}^* \tag{20}$$

$$\Delta_{\bar{f}} = (\bar{f} - f_{min}^*)/f_{min}^* \tag{21}$$

Before the t-test, F-test was performed to check the equality of variances of the two compared method. The significance level is set at 0.05, which means that if the null hypothesis is rejected with p-value < 0.05, there is a significant difference between the results of both algorithms. The results of statistical tests are depicted in Table 3.

As shown in Table 3, the proposed IGASA generated the solutions that were significantly better than the SA, in both $\Delta_{f_{min}}$ and $\Delta_{\bar{f}}$. The tests failed to reject H_0 in the comparison between IGASA and GA, and IGASA and hybrid GA-SA for $\Delta_{f_{min}}$ and $\Delta_{\bar{f}}$ criteria. However, further test for RDI indicated that the proposed IGASA was significantly better than the three benchmark methods in generating the near-optimal solutions. In terms of the algorithm efficiency, the average computational time T of IGASA is comparable to hybrid GA-SA as the parameters were set equal. The standard GA is slightly faster since it encompassed simpler crossover and mutation operations. Meanwhile, SA is significantly faster since it is a trajectory-based algorithm. Overall, the proposed IGASA offers an effective and efficient algorithm for solving the reentrant permutation flow shop scheduling problem.

Conclusions

This study develops an improved hybrid genetic algorithm-simulated annealing to solve the RPFS scheduling problem. The proposed algorithm is developed in order to optimize multiple objective functions simultaneously, these are: (1) to minimize the makespan, (2) to minimize the average completion time, and (3) to minimize the total tardiness. The applicability of the IGASA is tested using various size of datasets, whereby each dataset has a different number of jobs, machines, and reentrant layers. The results are then compared with the results of the benchmark methods.

It has been proven that the proposed IGASA outperforms the GA, SA, and hybrid GA-SA, indicated by the positive value of average IR and lower value of RDI. The average improvement rates of IGASA based on average fitness are 5.7%, 0.4%, and 0.8% as compared to SA, GA, and hybrid GA-SA, respectively. The average RDI of IGASA is 5.2%, which is significantly lower than other benchmark methods which are 57.1%, 8.7%, 11.5% for SA, GA, and hybrid GA-SA, respectively. In addition, the IR is increasing and RDI of IGASA is decreasing in instances with a higher number of sub-jobs. Hence, it indicates effectiveness of the proposed IGASA in solving problem with high complexity. Further, the robustness test is also executed to exhibit the robustness of IGASA in solving the RPFS problem. The obtained average RIR of IGASA are 79.8%, 42.4%, and 43.3% against SA, GA, and hybrid GA-SA, respectively. Thus, it clearly shows the robustness of the proposed IGASA in delivering the near-optimal solution.

Finally, based on the current state-of-the-art research on the optimization methods, the dynamic nature of the real-world problem may be the focus of manufacturing research in the future. A dynamic environment is concerned with the variations of capacity constraints, resource consumption, or the addition of novel entities within the environment (e.g., jobs, machines, resources). These uncertain conditions will pose additional challenges in solving RPFS scheduling problems and would be a highly interesting extension for this study. In addition, the study can be extended for reentrant hybrid flow shop as well as by considering other factors such as the sequence-dependent setup time and machine breakdown to mimic some real-world scenario on the reentrant flow shop.

References

[1] M. Pinedo, and X. Chao, *Operation Schedulling*, McGraw Hill, Singapore, 1999.

[2] M. Hekmatfar, S.M.T.F. Ghoumi, and B. Karimi, "Two stage reentrant hybrid flow shop with setup times and the criterion of minimizing makespan," *Applied Soft Computing*, Vol. 11, pp. 4530-4539, 2011. doi: 10.1016/j.asoc.2011.08.013.

[3] J.S. Chen, "A branch and bound procedure for the reentrant permutation flow-shop scheduling problem," *International Journal of Advance Manufacturing Technology*, Vol. 29, pp. 1186-193, 2006. doi: 10.1007/s00170-005-0017-x.

[4] M.L. Pinedo, *Scheduling: Theory, Algorithms, and Systems*, 5th Edition, Springer International, 2016.

[5] M.Y. Wang, S.P. Sethi, and S.L.V.D. Velde, "Minimizing makespan in a class of reentrant shops," *Operations Research*, Vol. 45, pp. 702-712, 1997. doi: 10.1287/opre.45.5.702.

[6] R.H. Huang, S.C. Yu, and C.W. Kuo, "Reentrant two-stage multiprocessor flow shop scheduling with due windows," *International Journal of Advance Manufacturing Technology*, Vol. 71, pp. 1263-1276, 2014, doi: 10.1007/s00170-013-5534-4.

[7] Y. Sun, C. Zhang, L. Gao, and X. Wang, "Multi-objective optimization algorithms for flow shop scheduling problem: A review and prospects," *International Journal of Advance Manufacturing Technology*, Vol. 55, pp. 723-739, 2011. doi: 10.1007/s00170-010-3094-4.

[8] S.C. Graves, H.C. Meal, D. Stefek, and A.H. Zeghmi, "Scheduling of re-entrant flow shops," *Journal of Operations Management*, Vol. 3, pp. 197-207, 1983. doi: 10.1016/0272-6963(83)90004-9.

[9] F. Chu, C. Chu, and C. Desprez, "Series production in a basic re-entrant shop to minimize makespan or total flow time," *Computers & Industrial Engineering*, Vol. 58, pp. 257-268, 2010. doi: 10.1016/j.cie.2009.02.017.
[10] S.W. Choi, and Y.D. Kim, "Minimizing makespan on an m-machine re-entrant flowshop," *Computers & Operations Research*, Vol. 35, pp. 1684-1696, 2008. doi: 10.1016/j.cor.2006.09.028.
[11] D.L. Yang, W.H. Kuo, and M.S. Chern "Multi-family scheduling in a two-machine reentrant flow shop with setups," *European Journal of Operational Research*, Vol. 187, pp. 1160-1170, 2008. doi: 10.1016/j.ejor.2006.06.065.
[12] E. Demirkol, and R. Uzsoy "Decomposition methods for reentrant flow shops with sequence-dependent setup times," *Journal of Scheduling*, Vol. 3, pp. 155-177, 2000. doi: 10.1002/(SICI)1099-1425(200005/06)3:3<155::AID-JOS39>3.0.CO;2-E .
[13] S.W. Choi, and Y.D. Kim, "Minimizing total tardiness on a two-machine re-entrant flowshop," *European Journal of Operational Research*, Vol. 199, pp. 375-384, 2009. doi: 10.1016/j.ejor.2008.11.037 .
[14] B. Jeong, and Y.D. Kim, "Minimizing total tardiness in a two-machine re-entrant flowshop with sequence-dependent setup times," *Computers & Operations Research*, Vol. 47, pp. 72-80, 2014. doi: 10.1016/j.cor.2014.02.002.
[15] Y.H. Kang, S.S. Kim, and H.J. Shin, "A scheduling algorithm for the reentrant shop: An application in semiconductor manufacture," *International Journal of Advance Manufacturing Technology*, Vol. 35, pp. 566-574, 2007. doi: 10.1007/s00170-006-0736-7.
[16] T. Kaihara, N. Fujii, A. Tsujibe, and Y. Nonaka, "Proactive maintenance scheduling in a re-entrant flow shop using Lagrangian decomposition coordination method," *CIRP Annals*, Vol. 59, pp. 453-456, 2010. doi: 10.1016/j.cirp.2010.03.031.
[17] F. Dugardin, F. Yalaoui, and L. Amodeo, "New multi-objective method to solve reentrant hybrid flow shop scheduling problem," *European Journal of Operational Research*, Vol 203, pp. 22-31, 2010. doi: 10.1016/j.ejor.2009.06.031.
[18] H.S. Choi, H.W. Kim, and D.H. Lee, "Scheduling algorithms for two-stage reentrant hybrid flow shops: Minimizing makespan under the maximum allowable due dates," *International Journal of Advance Manufacturing Technology*, Vol. 42, pp. 963-973, 2009. doi: 10.1007/s00170-008-1656-5.
[19] H.M. Cho, S.J. Bae, J. Kim, and I.J. Jeong, "Bi-objective scheduling for reentrant hybrid flow shop using Pareto genetic algorithm," *Computers & Industrial Engineering*, Vol. 61, pp. 529-541, 2011. doi: 10.1016/j.cie.2011.04.008.
[20] M. Ebrahimi, S.M.T.F. Ghomi, and B. Karimi, "Hybrid flow shop scheduling with sequence dependent family setup time and uncertain due dates," *Applied Mathematical Modelling*, Vol. 38, pp. 2490-2504, 2014. doi: 10.1016/j.apm.2013.10.061.
[21] C. Jing, W. Huang, and G. Tang, "Minimizing total completion time for re-entrant flow shop scheduling problems," *Theoretical Computer Science*, Vol. 412, pp. 6712-6719, 2011, doi: 10.1016/j.tcs.2011.08.030.
[22] C. Chanmanlor, K. Sethanan, C.F. Chien, and M. Gen, "Hybrid Genetic Algorithms for Solving Reentrant Flow-Shop Scheduling with Time Windows," *Industrial Engineering and Management Systems*, Vol. 12, pp. 306-316, 2013. doi: 10.7232/iems.2013.12.4.306.
[23] C. Chanmanlor, K. Sethanan, M. Gen, and C.F. Chien, "Embedding ant system in genetic algorithm for re-entrant hybrid flow shop scheduling problems with time window constraints," *Journal of Intelligent Manufacturing*, Vol. 28, pp. 1915-1931, 2017. doi: 10.1007/s10845-015-1078-9.

[24] J.S. Chen, J.C.H. Pan, and C.K. Wu, "Hybrid tabu search for re-entrant permutation flow-shop scheduling problem," *Expert Systems with Applications*, Vol. 34, pp. 1924-1930. 2008, doi: 10.1016/j.eswa.2007.02.027.
[25] P. Fattahi, N.B. Tavakoli, A.J. Nejad, and F. Jolai, "A hybrid algorithm to solve the problem of re-entrant manufacturing system scheduling," *CIRP Journal of Manufacturing Science and Technology*, Vol. 3, pp. 268-278, 2010. doi: 10.1016/j.cirpj.2011.01.001.
[26] A.P. Rifai, H.T. Nguyen, and S.Z.M. Dawal, "Multi-objective adaptive large neighborhood search for distributed reentrant permutation flow shop scheduling," *Applied Soft Computing*, Vol. 40, pp. 42-57, 2016. doi: 10.1016/j.asoc.2015.11.034.
[27] Q. Zhang, Z. Tian, S. Wang, and S. Liu, "Iterated Greedy Algorithm for Solving a Hybrid Flow Shop Scheduling Problem with Reentrant Jobs," In: *2020 Chinese Control and Decision Conference (CCDC)*, IEEE, pp. 5636-5641, 2020, doi: 10.1109/CCDC49329.2020.9164464.
[28] K. Geng, C. Ye, L. Cao, and L. Liu, "Multi-objective reentrant hybrid flowshop scheduling with machines turning on and off control strategy using improved multi-verse optimizer algorithm," *Mathematical Problems in Engineering*, Article No. 2573873, 2019, doi: 10.1155/2019/2573873.
[29] K. Amrouche, M. Boudhar, and N. Sami, "Two-machine chain-reentrant flow shop with the no-wait constraint," *European Journal of Industrial Engineering*, Vol. 14, No. 4, pp. 573-597, 2020. doi: 10.1504/EJIE.2020.108577.
[30] C.C. Lin, W.Y. Liu, and Y.H. Chen, "Considering stockers in reentrant hybrid flow shop scheduling with limited buffer capacity," *Computers & Industrial Engineering*, Vol. 139, p. 106154, 2020. doi: 10.1016/j.cie.2019.106154.
[31] F.T.S. Chan, and H.K. Chan, "A comprehensive survey and future trend of simulation study on FMS scheduling," *Journal of Intelligent Manufacturing*, Vol. 15, pp. 87-102, 2004. doi: 10.1023/B:JIMS.0000010077.27141.be.
[32] S.Z.M. Dawal, N. Yusoff, H.T. Nguyen, and H. Aoyama, "Multi-attribute decision-making for CNC machine tool selection in FMC based on the integration of the improved consistent Fuzzy AHP and TOPSIS," *ASEAN Engineering Journal Part A*, Vol. 3, pp. 16-31, 2013.
[33] O.A. Joseph, and R. Sridharan, "Analysis of dynamic due-date assignment models in a flexible manufacturing system," *Journal of Manufacturing Systems*, Vol. 30, pp. 28-40, 2011. doi: 10.1016/j.jmsy.2011.02.005.
[34] S.H. Chen, M.C. Chen, P.C. Chang, and V. Mani, "Multiple parents crossover operators: A new approach removes the overlapping solutions for sequencing problems," *Applied Mathematical Modelling*, Vol. 37, pp. 2737-2746, 2013. doi: 10.1016/j.apm.2012.06.005.
[35] E. Osaba, E. Onieva, and R. Carballedo, "An adaptive multi-crossover population algorithm for solving routing problems," *Studies in Computational Intelligence*, Vol. 512, pp. 113-124, 2014. doi: 10.1007/978-3-319-01692-4_9.
[36] C.C. Bolton, and V. Parada, "Automatic combination of operators in a genetic algorithm to solve the traveling salesman problem," *PLoS ONE*, Vol. 10, pp. 1-25, 2015. doi: 10.1371/journal.pone.0137724.