algorithms

MDPI

Article

Dynamic Events in the Flexible Job-Shop Scheduling Problem: Rescheduling with a Hybrid Metaheuristic Algorithm

Shubhendu Kshitij Fuladi and Chang-Soo Kim *

Department of Information Systems, Pukyong National University, Busan 608737, Republic of Korea; shubhendu06@gmail.com
* Correspondence: cskim@pknu.ac.kr

Abstract: In the real world of manufacturing systems, production planning is crucial for organizing and optimizing various manufacturing process components. The objective of this paper is to present a methodology for both static scheduling and dynamic scheduling. In the proposed method, a hybrid algorithm is utilized to optimize the static flexible job-shop scheduling problem (FJSP) and dynamic flexible job-shop scheduling problem (DFJSP). This algorithm integrates the genetic algorithm (GA) as a global optimization technique with a simulated annealing (SA) algorithm serving as a local search optimization approach to accelerate convergence and prevent getting stuck in local minima. Additionally, variable neighborhood search (VNS) is utilized for efficient neighborhood search within this hybrid algorithm framework. For the FJSP, the proposed hybrid algorithm is simulated on a 40-benchmark dataset to evaluate its performance. Comparisons among the proposed hybrid algorithm and other algorithms are provided to show the effectiveness of the proposed algorithm, ensuring that the proposed hybrid algorithm can efficiently solve the FJSP, with 38 out of 40 instances demonstrating better results. The primary objective of this study is to perform dynamic scheduling on two datasets, including both single-purpose machine and multi-purpose machine datasets, using the proposed hybrid algorithm with a rescheduling strategy. By observing the results of the DFJSP, dynamic events such as a single machine breakdown, a single job arrival, multiple machine breakdowns, and multiple job arrivals demonstrate that the proposed hybrid algorithm with the rescheduling strategy achieves significant improvement and the proposed method obtains the best new solution, resulting in a significant decrease in makespan.

Keywords: flexible job-shop scheduling; dynamic scheduling; genetic algorithm; simulated annealing algorithm; variable neighborhood search

Citation: Fuladi, S.K.; Kim, C.-S.. Dynamic Events in the Flexible Job-Shop Scheduling Problem: Rescheduling with a Hybrid Metaheuristic Algorithm. *Algorithms* **2024**, *17*, 142. https://doi.org/10.3390/a17040142

Academic Editors: Frank Werner and Maciej Drozdowski

Received: 8 January 2024
Revised: 21 March 2024
Accepted: 25 March 2024
Published: 28 March 2024

1. Introduction

The flexible job-shop scheduling problem (FJSP) is an expanded version of the traditional job-shop scheduling problem (JSSP), with additional flexibility and complexity in the scheduling process [1]. The FJSP involves determining the operation sequence for executing operations with satisfied constraints and the selection of suitable machines to carry out these operations, considering factors such as compatibility, efficiency, and resource utilization [2]. Consequently, the FJSP emerges as a highly demanding problem, characterized by its status as an extremely nondeterministic polynomial-time hard (NP-hard) problem that demands intensive computational effort to derive optimal solutions [3]. Scheduling has received a lot of research attention, but most of it has focused on static scheduling. In real-world smart factories, production environments must deal with uncertainty, as various unpredictable and dynamic occurrences frequently arise. In industry, such events are referred to as dynamic scheduling problems [4]. These include issues such as machine breakdowns [5], unexpectedly arriving jobs [6], and order cancellations [7]. Such unpredictable events add complexity to the scheduling and coordination of operations, highlighting the need for scheduling methodologies to address the dynamic flexible job-shop scheduling problem

(DFJSP), which deals with two types of datasets: single-purpose machine datasets and multi-purpose machine datasets. The method that can help to solve the dynamic event problem is rescheduling; this approach involves rescheduling when a new event occurs [8].

The GA is widely used for scheduling and optimization problems due to its ability to explore the solution space and find the near-optimal solution, but it has a low convergence rate and is capable of falling into the local optima, which can result in longer computational times and may impact performance [9,10]; to avoid these conditions, many studies have proposed the hybrid algorithms explained above. Therefore, this study suggests using a hybrid genetic algorithm, simulated annealing, and variable neighborhood search (GASAVNS) algorithms to enable strong local search ability with global optimization. Another contribution is the influence of the hybrid algorithm on various dynamic scenarios observed in the context of flexible job-shop scheduling. The method that is used for dynamic scheduling is rescheduling with the hybrid GASAVNS algorithm by extracting the static scheduling data. All observations for dynamic events will be carried out on the obtained datasets and datasets from [11]. The simulation results show that the proposed method works well in both scheduling and rescheduling for the FJSP and DFJSP, respectively.

The structure of the remaining manuscript is as follows: A literature review is presented in Section 2. Section 3 gives a brief description of the FJSP formulation of both datasets, namely the single-purpose machine and multi-purpose machine datasets, along with the required assumptions to initiate the FJSP. Section 4 explains the proposed method, the hybrid GASAVNS algorithm, while Section 5 elaborates on the proposed rescheduling method for dynamic events such as machine breakdown and job arrival. Section 6 presents the experimental results and analysis for the FJSP and various dynamic events using both datasets. The conclusion is provided in Section 7.

2. Literature Review

Over the last few years, in response to the strong NP-hard FJSP, researchers have introduced various heuristic approaches aimed at providing effective and practical solutions that involve complex combinatorial optimization and multiple constraints. Some dispatching rules like first-in–first-out (FIFO), shortest processing time (SPT), and longest processing time (LPT) are easy to use, but they are not accurate enough for solving tough scheduling problems. Heuristic approaches are proposed to find the near-optimal solution for the static FJSP. Giovanni et al. [12] proposed an improved genetic algorithm (GA) to solve distributed and flexible job-shop scheduling. Lim et al. [13] implemented a hyperheuristic-based simulated annealing (SA) method to solve the FJSP. This method utilized single-point selection that maintains a single candidate solution from the search space. Saidi-Mehrabad et al. [14] utilized the tabu search (TS) algorithm to tackle the FJSP. This heuristic approach implements a sequence-dependent setup and is implemented using the visual fortune language. Han et al. [15] adapt the reinforcement learning algorithm for solving the FJSP. A variety of improved and hybrid methods have emerged to enhance optimization objectives. These approaches integrate diverse techniques, aiming to achieve superior performance in scheduling. For example, Gao et al. [16] presented an effective hybrid approach by combining GA with the local improvement capacity of TS. Their study focused on addressing the FJSP with the objective of minimizing the makespan. Escamilla-Serna et al. [17] made a hybrid GA-RRHC algorithm using a GA and random-restart hill-climbing (RRHC) to perform a local search. Additionally, the authors used cross-over and mutation operators with a cellular automaton (CA)-inspired neighborhood to perform a global search, with implementation carried out using MATLAB. Tang et al. [18] introduced a hybrid algorithm combining chaos particle swarm optimization and a GA, addressing the optimization mechanism of the FJSP.

The dynamic job-shop scheduling problem has recently captured significant research attention even though it involves uncertain events. Zhang et al. [19] proposed a hybrid tabu search and genetic algorithm combination to address the dynamic job-shop scheduling problem, which involves the difficulties of machine breakdowns and job arrivals. Wang

et al. [11] proposed a variable interval rescheduling strategy to deal with the dynamic flexible job-shop scheduling problem; this approach involves the improvement of a genetic algorithm to obtain an effective solution. A heuristic approach for a dynamic flexible job-shop scheduling problem considering variable processing times was proposed by Shahgholi et al., inspired by the artificial bee colony algorithm, that aimed to achieve a near-optimal solution [20]. Zhang et al. [21] proposed a framework of dynamic scheduling based on an improved gene expression programming algorithm combined with effective neighborhood structures. Fattahi et al. [22] considered two objectives, efficiency and stability, for dynamic scheduling in flexible job-shop systems, including various strategies for handling arrival disturbances using a genetic algorithm. Kundakci et al. [23] combined a genetic algorithm and tabu search to address the optimization problem of the dynamic job-shop scheduling problem with machine breakdown for a parallel machine environment. Wei et al. [24] proposed a hybrid genetic algorithm with a simulated annealing algorithm for flow-shop scheduling with makespan criteria. The implementation of a hybrid genetic algorithm (GA) and a simulated annealing (SA) approach was employed by Al-milli et al. to tackle timetabling challenges under specific constraints, showing promising results [25]. The study by Shady et al. [26] introduces the integration of GA with the NSGA-II to automatically generate dispatching rules for the DJSP with machine breakdown, achieving the minimizing mean flow time and makespan simultaneously. Industry 4.0 is witnessing a transformative shift where manufacturing systems are becoming increasingly intelligent and capable of optimization [27,28]. Researchers have explored the application of scheduling; for example, Hadi et al. [29] used flexible flow-shop scheduling in the domain of a waste-to-energy system, which reduced the total cost significantly in the equipment-driven industry, and Soroush et al. [30] investigated a dynamic scheduling problem within a job-shop robotic cell for material handling. This study proposes the utilization of a hybrid approach comprising genetic algorithms, simulated annealing, and variable neighborhood search (GASAVNS) to enhance local search capabilities within the context of global optimization.

3. Problem Formulation

In the Problem Formulation Section, the two types of datasets encountered within manufacturing environments are described and formulated: single-purpose machine datasets and multi-purpose machine datasets. Foundational assumptions are established under the FJSP framework. These constraints ensure the feasibility of a solution. The primary objective of the FJSSP is to minimize the maximum completion time expressed in Equation (1)

$$\text{Objective} = \text{Min}\ (Cmax) \tag{1}$$

Here, *Cmax* represents the maximum completion time required to complete jobs among all jobs processed by machines.

3.1. Single-Purpose Machine Dataset

Considering the FJSP for the single-purpose machine, a set of 'n' jobs is represented by $J = \{J_1, J_2, J_3, \ldots, J_n\}$ and a set of 'g' machine groups is represented by $MG = \{MG_1, MG_2, MG_3, \ldots, MG_g\}$. Each machine group can have m machines $MG_g = \{M_1, M_2, \ldots, M_m\}$, and every operation has 'm' machine choices. Each group of machines is dedicated to one unique operation, and the number of operations will be equal to the number of machines. This type of dataset has single-purpose machines (SPMs). Each operation in the job can be processed by any one machine out of a dedicated machine group. Each job consists of a sequence of operations represented as $O = (O_{i1}, O_{i2}, \ldots, O_{ik}, \ldots O_{i,ni})$. Each machine Mi becomes available at time $AM\text{i} \geq AM = \{AM_1, AM_2, \ldots, AM_m\}$. Each operation needs to be completed by a functional machine within a particular time $T = \{T_{i1}, T_{i2}, \ldots, T_{ik}\}$. The processing time varies depending on the resources used for processing operations. Each machine has a different execution time, and every operation has a different resource usage time. Therefore, the processing time of machines, even within the same group, differs. In

other words, different machines may have different processing times for the same operation; some machines are faster, and some are slower.

Considering the textile factory environment, many products are involved in production, such as pants (Product 1), shirts (Product 2), and t-shirts (Product 3), which correspond to *J1*, *J2*, and *J3*, respectively. Each product undergoes a series of operations, such as sizing, weaving, knitting, dying, and printing [31,32]. Table 1 shows the initial information for the FJSP. There are three jobs (*J1, J2, J2*), and each job has multiple operations. As shown in Table 1 there are four machine groups (MG_1, MG_2, MG_3, MG_3), and every group performs a specific operation, which means all machines present in *MG1* perform the same operation. An operation OP_{11} has a choice of four machines, M_1, M_2, M_3, and M_4, whereas *J*1 has two operations (OP_{11}, OP_{12}). Depending on the product specifications, some of these operations may not be required. The symbol "-" indicates that the corresponding operation does not apply to that job. Each operation has a machine group to perform that operation; OP_{11} (job1, operation1) is processed on machine group 1 (MG_1), and MG_1 has a choice of four machines: *M1* (8-unit processing time), *M*2 (7-unit processing time), M3 (8-unit processing time), and *M*4 (9-unit processing time). OP_{12} (Job1, operation2) is processed on machine group 2 (MG_2), and MG_2 has a choice of two machines, i.e., *M*5 and *M*6, with processing times of 4 and 5, respectively. Operations 3 (OP_{13}) and 4 (OP_{14}) are not available for *J*1. On other hand, *J2* has two operations (OP_{22}, OP_{24}) and job 3 has three operations (OP_{31}, OP_{33}, OP_{34}).

Table 1. Processing information of FJSP with single purpose-machine.

Jobs \ Ops	OP_1 (MG_1)			OP_2 (MG_2)			OP_3 (MG_3)			OP_4 (MG_4)		
	M_1	M_2	M_3	M_4	M_5	M_6	M_7	M_8	M_9	M_{10}	M_{11}	M_{12}
J_1	8	7	8	9	4	5	-	-	-	-	-	-
J_2	-	-	-	-	8	7	-	-	-	5	8	3
J_3	7	4	6	5	-	-	9	7	6	9	6	7

3.2. Multi-Purpose Machine Dataset

Considering the FJSP for multi-purpose machines, there is set of n jobs $J = \{J_1, J_2, J_3, \ldots, Jn\}$, and each J_i job consists of n_i operations $O = (O_{i1}, O_{i2}, \ldots, O_{ik}, \ldots O_{i,n_i})$. There is a set of m multi-purpose machines $M = \{M_1, M_2, \ldots, M_m\}$, and one machine is capable of processing multiple operations, but at the same time, only one operation can proceed [33]. Each machine can perform p_m operations at a given time $P_{m,i} = \{P_{m,1}, P_{m,2}, \ldots P_{mpm}\}$. Each machine *M*i is available at time *AM*i => $AM = \{AM_1, AM_2, \ldots, AM_m\}$. Each operation needs to be processed by a functional machine within a particular time $T = \{T_{i1}, T_{i2}, \ldots, T_{ik}\}$.

As shown in Table 2, there are three jobs $\{J_1, J_2, J_3\}$ and three machines $\{M_1, M_2, M_3\}$. Each job has three operations that need to be performed, and every operation has many machine choices. Operation O_{11} from J_1 can be followed by a choice of three machines: M_1 with a processing time of 7 units, M_2 with a processing time of 2 units, or M_3 with a processing time of 3 units. The third operation of J_1 has a choice of two machines, M_1 and M_3. On the other hand, J_2 and J_3 have three operations each, with their number of machine choices mentioned in Table 2.

Table 2. Processing information of FJSP with multi-purpose machine.

Jobs	Operations	Processing Time		
		M_1	M_2	M_3
J_1	O_{11}	7	2	3
	O_{12}	3	4	5
	O_{13}	3	-	6
J_2	O_{21}	2	-	5
	O_{22}	4	4	-
	O_{23}	2	-	5
J_3	O_{31}	4	2	3
	O_{32}	3	-	-
	O_{33}	2	8	5

"-" shows that the machine cannot proceed with the corresponding operation.

3.3. Assumptions for Static Scheduling

The hypotheses or assumptions considered for the static initial scheduling are as follows [11]:

1. Every machine in the system is available for the scheduling operation at time zero.

$$\forall i \in m, Ami = 0 \tag{2}$$

2. Every job within the system can be started at the initial time point, which is time zero, as mentioned in equation [34].

$$C_{ik} = T_{ik}(i,k) \in O \text{ and } k = 1 \tag{3}$$

3. A machine can perform one operation at a time and is not capable of performing many operations at once. The scheduling process needs to ensure that operations assigned to a particular machine do not overlap in time.

$$\forall i \in m, p_i = 1 \tag{4}$$

4. An operation on a machine cannot be stopped or interrupted once it has started and must continue until finished.
5. Due dates and release times are unspecified.
6. Job setup time and transportation time are ignored.

4. Proposed Hybrid GASAVNS Algorithm

In this paper, a hybrid algorithm is used which includes the GA, SA, and the variable neighborhood search (VNS) algorithm. The SA is inserted into the procedure of GA for local searching. The GA can fall into the local optimal solution, which is why the proposed hybrid algorithm uses the local search algorithm, i.e., SA, to avoid being trapped in the local minima and accelerate the convergence speed. To diversify the solution, the VNS is used with SA with the aim to escape the local optimum and reach a global optimum. VNS works on the principle of using two or more neighborhood structures and systematically changing the neighborhood within the local search [35]. Figure 1 shows the workflow of the proposed hybrid algorithm. The following is a description of the GASA and VNS steps that are used in the proposed hybrid algorithm. The '1' in Figure 1 flowchart represents the link between the scheduling and rescheduling methods allowing for retrieval of scheduling data for use in the rescheduling process.

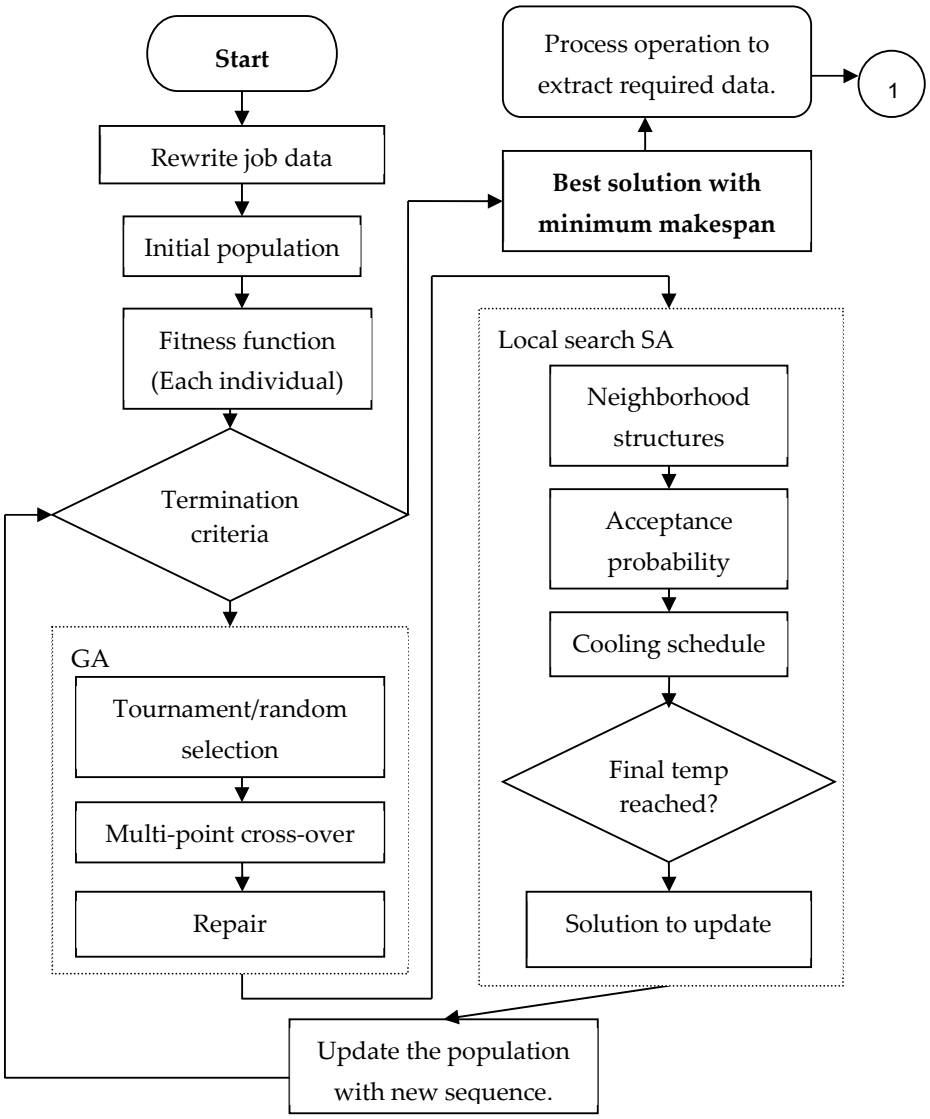

Figure 1. Proposed hybrid GASAVNS algorithm flowchart for scheduling.

4.1. Encoding and Decoding

In the flexible job-shop scheduling problem (FJSP), the following scheme is utilized for the selection of available machines and the scheduling of operations based on job numbers. Each job number is linked with the available machines, options, and operations, along with their corresponding processing times. The assignment of operations and machines is then carried out according to the feasible solution to satisfy the constraints [36,37].

$$SEQUENCE = (Job, Machine, Operation, Option, Processing\ Time)$$

4.2. Initial Population

Creating an initial population of solutions is an essential step at the beginning of a genetic algorithm. To form the initial population, we must generate a set of solutions by randomly selecting an optional machine from a set of machines and a job from a set of jobs. In this method, 300 numbers of random populations are created, each of which represents a potential solution to the problem.

4.3. Selection

The selection operator used in the GA to select the individual plays a crucial role in determining which members of a population will be selected for further generations. In

this paper, two selection operators are adopted, with the capacity for either to be utilized in a single iteration. One is tournament selection. In tournament selection, the tournament size is initialized to 5. We must select the two individual sequences (chromosomes) with the highest fitness (minimum makespan) from the b randomly selected sequences from the current population. The other is random selection. In random selection, two sequences are randomly selected without considering their fitness, which allows any sequence to participate in further steps to produce offspring and allows for the possibility of discovering diverse solutions.

4.4. Cross-Over

In this research, two cross-over operators are adopted with the ability to be applied in an iteration. One is two-point cross-over. In this case, the two selected parents make a cross-over with each other and produce two offspring sequences. In a two-point cross-over, we exchange the position of two randomly selected operations (genes) between the two parents and generate two offspring. The other is multi-point cross-over. In multi-point cross-over, more than two operations are selected from two selected parents to exchange the position, which will generate two offspring.

4.5. Repair

It is possible that after cross-over, the generated offspring will violate the problem constraints; to repair this kind of problem, a repair operator is used to make sure that the newly generated offspring maintain the validity of the solution.

4.6. Local Search

In this paper, the heuristic method of SA is used as a local search algorithm. The SA can be successfully applied to the optimization scheduling problem [38]. SA will be applied to the output obtained from the cross-over in each iteration. SA is run through several iterations until the stopping requirement is satisfied.

The procedure is described as follows:

1. Set the initial parameters: In this proposed method, the initial values set for the parameters of SA are an initial temperature of 50.0, a cooling rate of 0.90, and a final temperature of 10.
2. Neighborhood structures: For neighborhood structures, we use the VNS. The idea behind VNS is to use two or more neighborhood structures and systematically mutate them in a neighborhood within a local search [39]. In each iteration of the local search, one neighborhood structure is chosen at random and applied to the cross-over output sequence (offspring).

   The following are the six types of neighborhood structures that are based on VNS:

- Swap neighborhood: Two positions are randomly selected from the sequence, and their positions are swapped.
- Insert neighborhood: Two elements are chosen at random and the latter element is placed in front of the earlier element.
- Inverse neighborhood: The elements are rearranged between the two randomly chosen positions in reverse order.
- Four positions are randomly selected from the sequence and their positions shuffled (rearrangement).
- Adjacent neighborhood: A position is randomly selected and switched to its adjacent position.
- Inverse neighborhood: Two positions are chosen from the sequence randomly, and then, the element order between those two positions is reversed.

3. Acceptance criteria and update: first, we find the delta makespan, i.e., the difference between the neighborhood makespan and the current makespan. Acceptance probability functions are crucial for accepting and rejecting new solutions based on the

makespan difference and current temperature. If the new solution is better, i.e., the makespan is less than the current makespan, it is accepted; otherwise, it is rejected.

4. Cooling schedule: This uses an exponential cooling scheme. The temperature steadily and gradually drops with the cooling rate (temperature*cooling rate).
5. Final temperature: The final temperature is a stopping condition. When the current temperature is equal to the final temperature, the local search loop stops, and we obtain the new solution. After obtaining the new solution, we update the population and control go to the HA next iteration.

4.7. Termination Criteria

The proposed hybrid GASAVNS algorithm terminates when the number of generations reaches the maximum iteration value; in this study, the iteration value is set to 300, and we obtain the most feasible solution with the optimal makespan.

5. Proposed Rescheduling Methods

In this section, rescheduling approaches are presented to address dynamic events in the FJSP. Specifically, we explore two distinct scenarios.

5.1. Rescheduling for Machine Breakdown (Dynamic Event)

Figure 2 shows the workflow for machine breakdown. A machine breakdown, referred to as a dynamic event, occurs when a machine fails unexpectedly at a specific time of scheduling. When it comes to flexible job-shop scheduling, if there is sudden (unplanned) unavailability of a machine, it can have a significant impact on the production process and lead to a delay in processing. In the proposed method, initially, basic information on machine breakdown is required, like the broken machine number and breakdown time of the machine. At the time of machine breakdown, all upcoming operations are rescheduled, and the starting time of operations is defined immediately after the breakdown time with interruption of the operations that had already started to process on the broken machine. After eliminating the machine breakdown, with the help of extracted data from static scheduling, the remaining operations that need to take place are rescheduled.

The conditions to filter out the operation that participates in rescheduling are as follows:

The operations involved in the machine breakdown:

- Operations that end before breakdown time (i.e., stop time of operation < breakdown time) will not be involved in the rescheduling strategy.
- Operations that end after breakdown time (i.e., stop time operation > breakdown time) will be involved in the rescheduling strategy.

The operations that are not involved in the machine breakdown:

- Operations that had already started (i.e., start time of operation > breakdown time) will not be involved in the rescheduling strategy.
- Operations that started after the breakdown time (i.e., start time operation > breakdown time) will be involved in the rescheduling strategy.

After eliminating the non-required operation, we must reschedule all remaining operations using the proposed hybrid optimization method with a new start time for the machines to obtain the optimization solution for rescheduling the sequence.

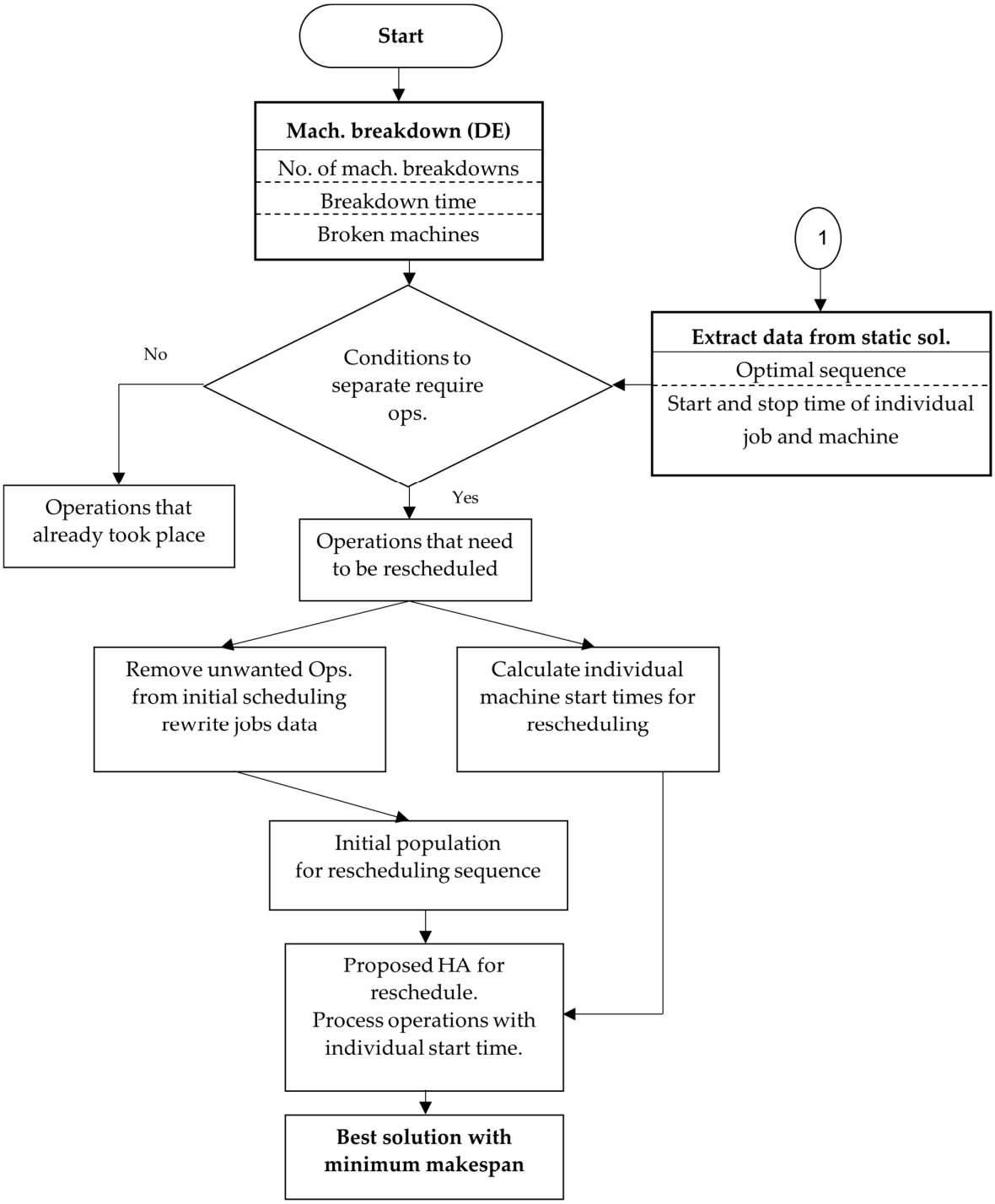

Figure 2. Proposed rescheduling method flowchart for machine breakdown.

5.2. Rescheduling for Job Arrival (Dynamic Event)

Figure 3 shows the workflow for the job arrival dynamic event. In this dynamic event, there can be the arrival of one or multiple urgent jobs. At the start of this method, the job arrival dataset and job arrival timing (i.e., rescheduling timing) are required. In the next step, we filter out all operations into operations that have already taken place and operations that need to be rescheduled with newly arrived jobs.

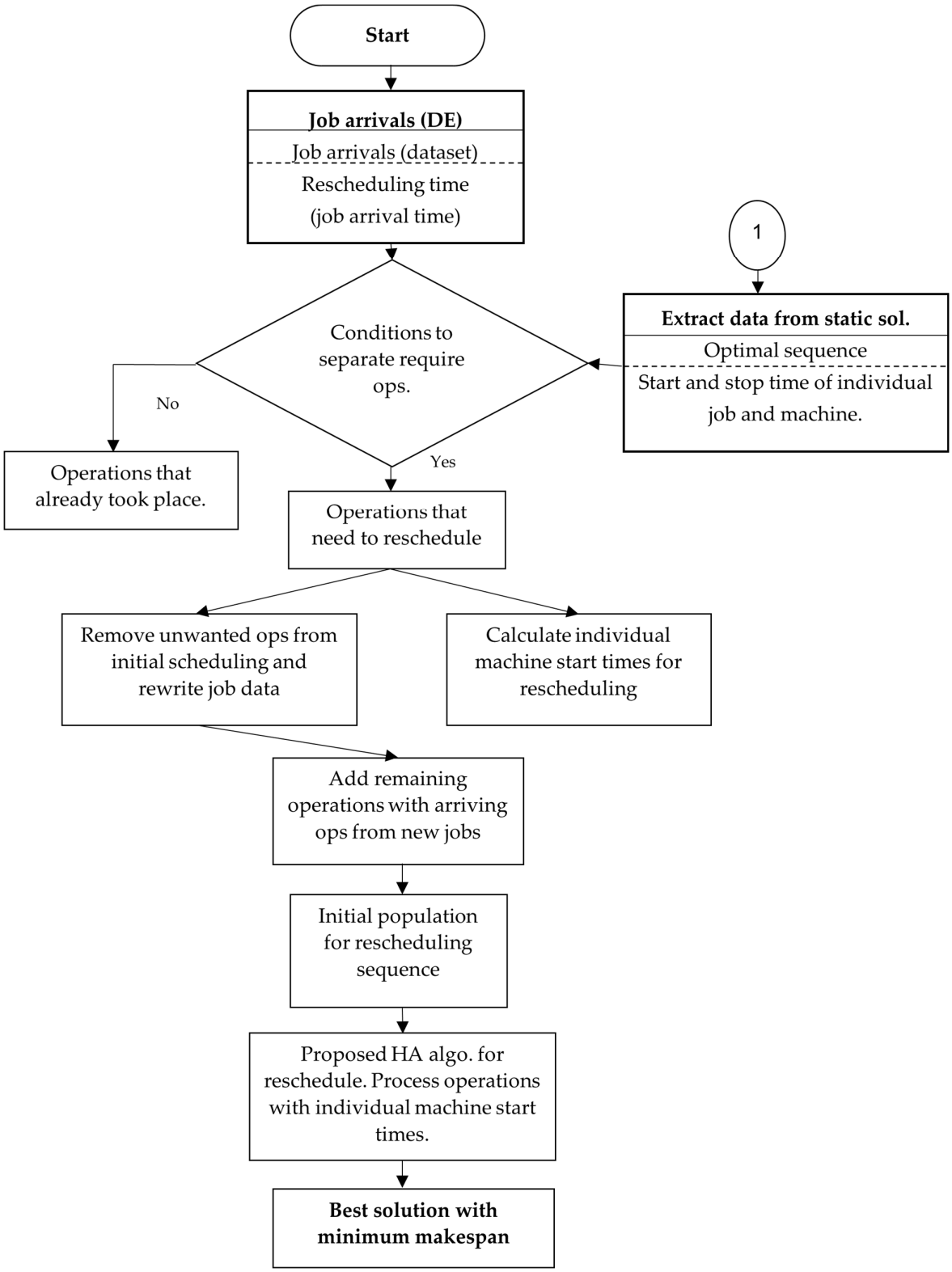

Figure 3. Proposed rescheduling method flowchart for job arrival.

The conditions to filter out the operations that participate in rescheduling are as follows:

- Operations that already started before the rescheduling time (i.e., the start time of operation < rescheduling time) will not be involved in the rescheduling strategy.

- Operations that did not start before the rescheduling time (i.e., the start time of operation > rescheduling or breakdown time) will be involved in the rescheduling strategy.

After filtering out the operations, we remove unwanted operations from the initial static scheduling sequence. The remaining operations that participate in rescheduling are added to the operations from the new job arrival dataset. Then, we produce a population of 300 for the new operation sequence and optimize the solution with the proposed hybrid algorithm. After applying the proposed algorithm, we obtain the best solution with the minimum makespan.

6. Results and Discussion

The result of this study shows both the static and dynamic event results, along with their respective analyses.

6.1. Static Scheduling Experiment Results

To illustrate the effectiveness and performance of the proposed hybrid GASAVNS algorithm on a static scheduling dataset, this study utilizes the benchmark problem dataset adapted from [40]. Table 3 shows the results of the experiment, providing a comparison of the makespan obtained using our algorithm with that obtained using other algorithms. The proposed algorithm is compared with the following DRL-based methods: DDQN [15], SPT, LPT, SA, and GA. The results indicate that the proposed hybrid GASAVNS algorithm shows the best result for 38 instances out of 40 compared to the other algorithms. This means that the proposed method is highly efficient for solving the static FJSP.

Table 3. Comparative results for different JSSP instances.

Instances	SPT	LPT	DQN	SA	GA	Proposed
La01 (10 × 5)	920	889	**666**	889	717	**666**
La02 (10 × 5)	901	894	**655**	871	751	**655**
La03 (10 × 5)	770	748	**597**	788	677	606
La04 (10 × 5)	916	848	**609**	804	658	**609**
La05 (10 × 5)	827	787	**593**	785	593	**593**
La06 (15 × 5)	1369	1105	**926**	1020	946	**926**
La07 (15 × 5)	1128	1145	**890**	1125	1007	**890**
La08 (15 × 5)	1168	1061	**863**	1089	992	**863**
La09 (15 × 5)	1289	1105	**951**	1129	980	**951**
La10 (15 × 5)	1345	1136	**958**	1065	963	**958**
La11 (20 × 5)	1654	1476	**1222**	1543	1880	**1222**
La12 (20 × 5)	1352	1222	1047	1402	1286	**1039**
La13 (20 × 5)	1747	1298	1151	1466	1239	**1150**
La14 (20 × 5)	1757	1360	**1292**	1485	1315	**1292**
La15 (20 × 5)	1476	1510	1221	1551	1432	**1219**
La16 (10 × 10)	1588	1238	**980**	1230	1135	1000
La17 (10 × 10)	1094	1157	799	1291	947	**794**
La18 (10 × 10)	1259	1264	**859**	1264	1063	**859**
La19 (10 × 10)	1339	1140	872	1256	1089	**860**
La20 (10 × 10)	1331	1293	**924**	1375	1107	**924**
La21 (15 × 10)	1707	1545	1162	1672	1458	**1132**
La22 (15 × 10)	1257	1409	1021	1489	1327	**1000**
La23 (15 × 10)	1522	1330	1053	1417	1423	**1034**
La24 (15 × 10)	1554	1472	1029	1500	1336	**1000**
La25 (15 × 10)	1624	1382	1067	1429	1355	**1061**
La26 (20 × 10)	2137	1616	1327	1696	1742	**1277**
La27 (20 × 10)	2048	1776	1397	1863	1837	**1345**
La28 (20 × 10)	2034	1668	1386	1748	1746	**1305**
La29 (20 × 10)	2048	1649	1323	2048	1691	**1290**
La30 (20 × 10)	2081	1783	1417	1848	1806	**1370**

Table 3. *Cont.*

Instances	SPT	LPT	DQN	SA	GA	Proposed
La31 (30 × 10)	2379	2394	1854	2415	2360	**1784**
La32 (30 × 10)	2823	2571	1900	2694	2587	**1850**
La33 (30 × 10)	2487	2372	1782	2445	2348	**1719**
La34 (30 × 10)	2500	2425	1880	2515	2402	**1748**
La35 (30 × 10)	2440	2514	1941	2514	2499	**1888**
La36 (15 × 15)	2070	1884	**1355**	1821	1806	1395
La37 (15 × 15)	2075	1940	1540	1983	1964	**1504**
La38 (15 × 15)	1944	1841	**1348**	1826	1832	1392
La39 (15 × 15)	1790	2064	1357	1902	1730	**1281**
La40 (15 × 15)	2003	1829	1336	1800	1814	**1300**

The bolded makespan in the table corresponds to the minimum makespan compared to the others.

In Figure 4, a Gantt chart illustrates the scheduling solution for problem instance LA28. It displays 20 jobs allocated across 10 machines. Each color represents a specific job; for example, all green sections indicate job number 17. On machine 1, the green-colored section represents job number 17, indicated within brackets as (17,1,1) (75). Here, 17 denotes the job number, 1 signifies the operation number, 1 indicates the chosen machine option from the set of machines, and 75 denotes the processing time. The maximum completion time (makespan) for scheduling all operations is 1305 units.

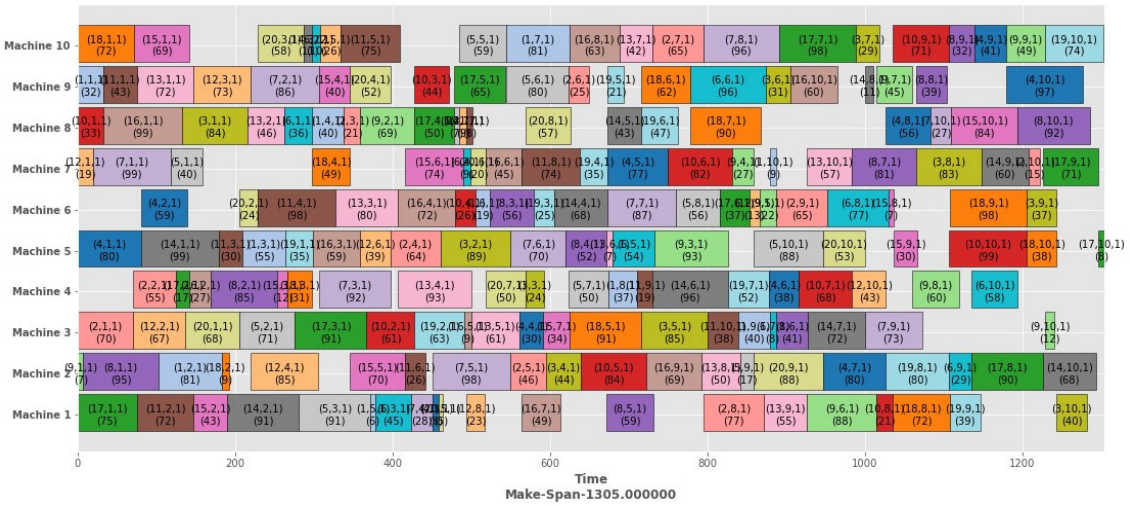

Figure 4. Gantt chart of problem la28 for static scheduling.

6.2. Machine Breakdown (Dynamic Event)

In this event, machine breakdown is considered a dynamic event. Table 4 shows the processing data adapted from [11] to demonstrate this event. These data come under the multi-purpose machine dataset type. Figure 5 displays a Gantt chart for static scheduling (before the machine breakdown) obtained using the proposed HA, with a makespan of 29 units. Then, one machine fails, i.e., machine 6 at a time of 20 units. Figure 6 shows a Gantt chart after rescheduling (before the machine breakdown) using the proposed HA strategy for the dynamic event, resulting in a makespan of 28 units, which is less than the comparative result [17,32]. The comparative results are shown in Table 5. In the processes of rescheduling, Operations O14, O34, O45, and O54, which were not on the broken machine and already started before breakdown time, are completed without any changes in rescheduling. All remaining operations participate in the rescheduling process, forming a new sequence.

Table 4. Processing information for 5-job and 6-job machine FJSP.

Jobs	Operations	Processing Time					
		M1	*M2*	*M3*	*M4*	*M5*	*M6*
J1	*O11*	2	3	4	-	-	-
	O12	-	3	-	2	4	-
	O13	1	4	5	-	-	-
	O14	4	-	-	3	5	-
	O15	-	6	8	7	6	9
J2	*O21*	3	-	5	5	2	-
	O22	4	3	-	-	6	-
	O23	-	-	4	4	7	11
	O24	-	5	-	-	-	5
	O25	4	5	7	7	5	-
J3	*O31*	5	6	-	-	-	-
	O32	-	4	3	3	5	-
	O33	-	-	-	-	9	12
	O34	6	5	4	4	8	-
	O35	8	6	-	-	7	8
J4	*O41*	9	-	7	9	-	-
	O42	-	6	-	4	-	5
	O43	1	-	3	-	-	3
	O44	6	-	9	7	5	4
	O45	-	8	7	8	8	-
J5	*O51*	4	3	7	9	3	6
	O52	5	6	-	4	-	5
	O53	6	-	4	-	-	3
	O54	-	5	-	7	-	7
	O55	7	-	8	7	8	-

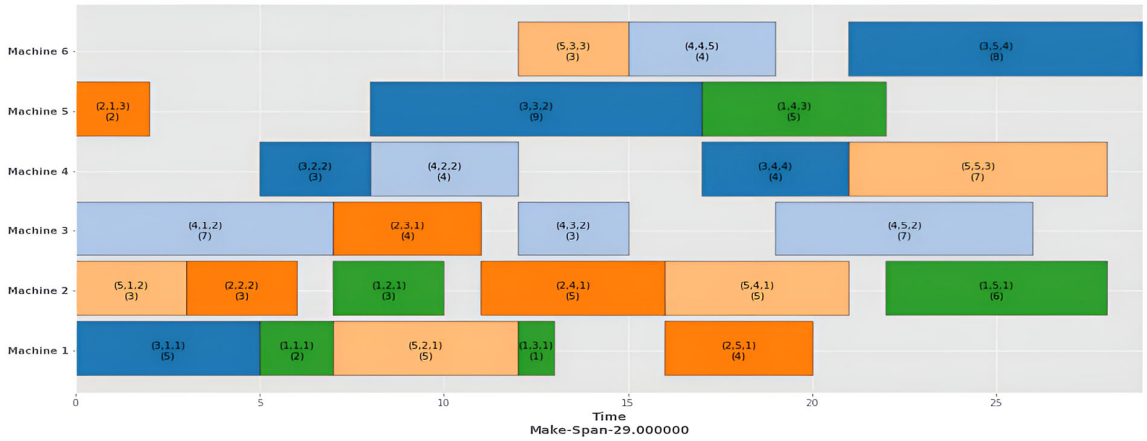

Figure 5. Gantt chart for static scheduling (before the dynamic event).

Table 5. Comparison of results for rescheduling.

Events	Method Proposed in [41]	Method Proposed in [11]	Proposed Method
Failure time	20	20	20
Machine repair time	0	0	0
Ideal makespan	37	35	29
Actual makespan	37	35	28

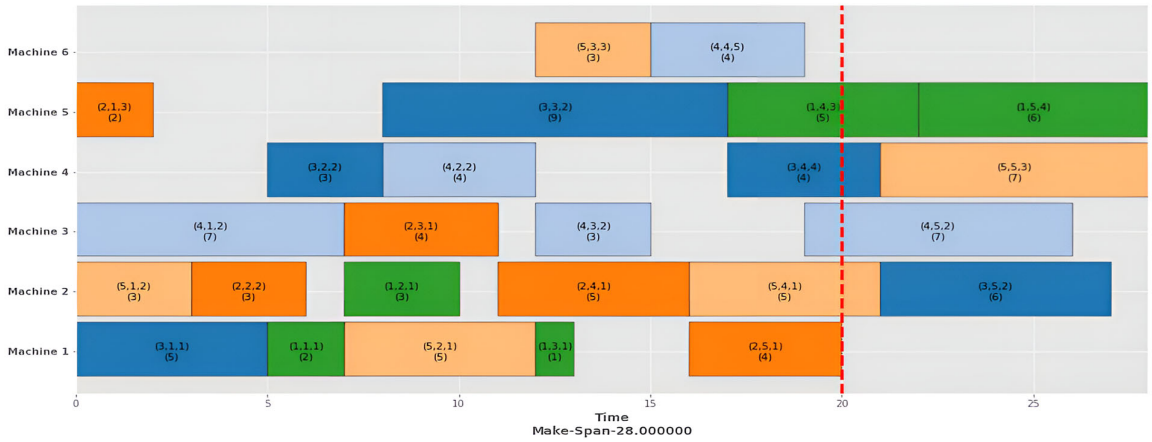

Figure 6. Gantt chart after optimized rescheduling (after the dynamic event).

6.3. Urgent Job Arrival

In this dynamic event, an urgent job arrives at a specific time. Table 6 shows the processing information of the arrival of a single job, i.e., job 6, consisting of five operations and six machines. The data from Table 4 are utilized for static scheduling. Job 6 arrives at a time of 20 units. Figure 7 shows a Gantt chart of the static scheduling of five jobs before job six arrives, with a makespan of 29. Figure 8 illustrates a Gantt chart of rescheduling after arriving at a new job without optimizing, with a makespan 37. After applying the proposed rescheduling HA strategy, the makespan is minimized to 31, as illustrated in the Gantt chart in Figure 9.

Table 6. Processing information of a job arrival event.

Job	Operations	Processing Time					
		M1	**M2**	**M3**	**M4**	**M5**	**M6**
J6	O61	6	7	3	6	2	5
	O62	6	7	2	6	7	2
	O63	2	2	5	5	2	3
	O64	3	4	7	7	2	3
	O65	3	3	5	7	5	5

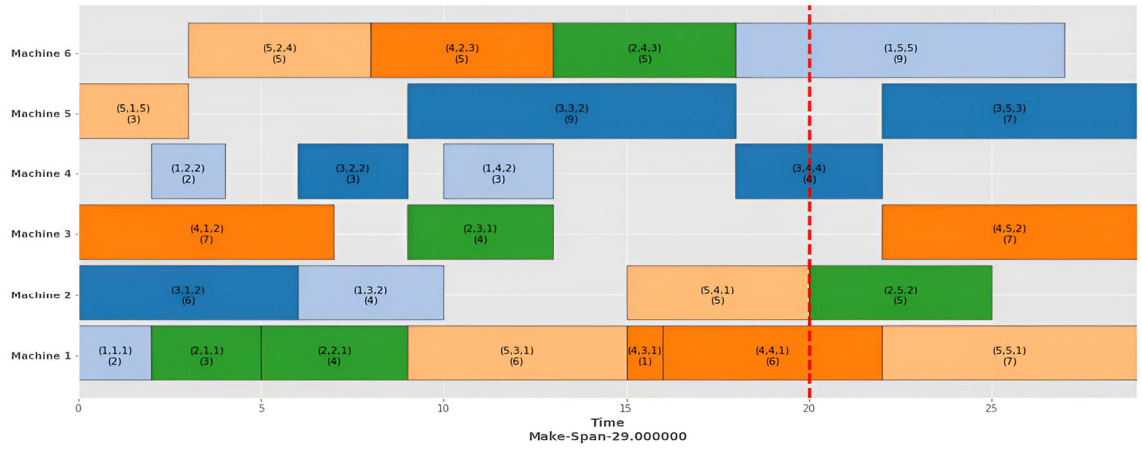

Figure 7. Gantt chart of scheduling before job arrival.

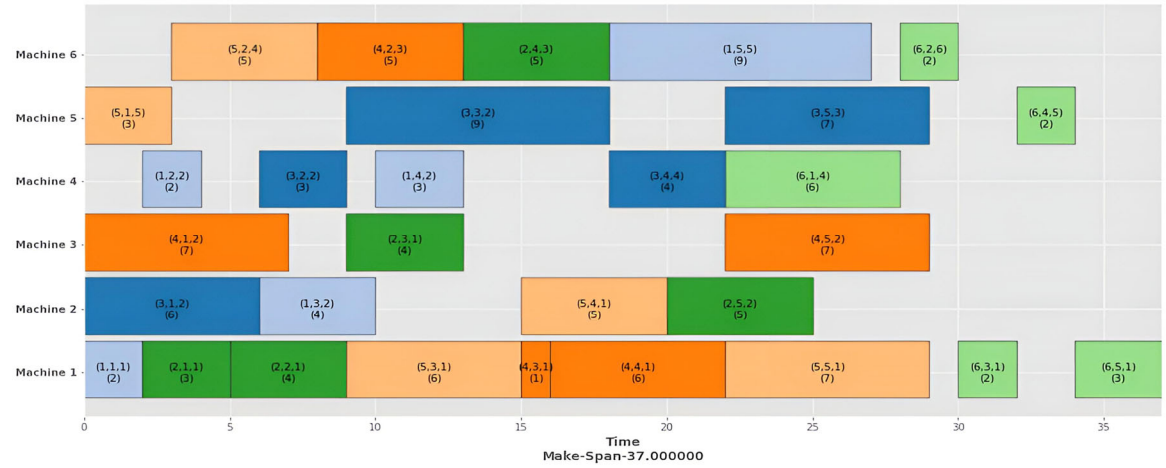

Figure 8. Gantt chart of scheduling after job arrival without optimization.

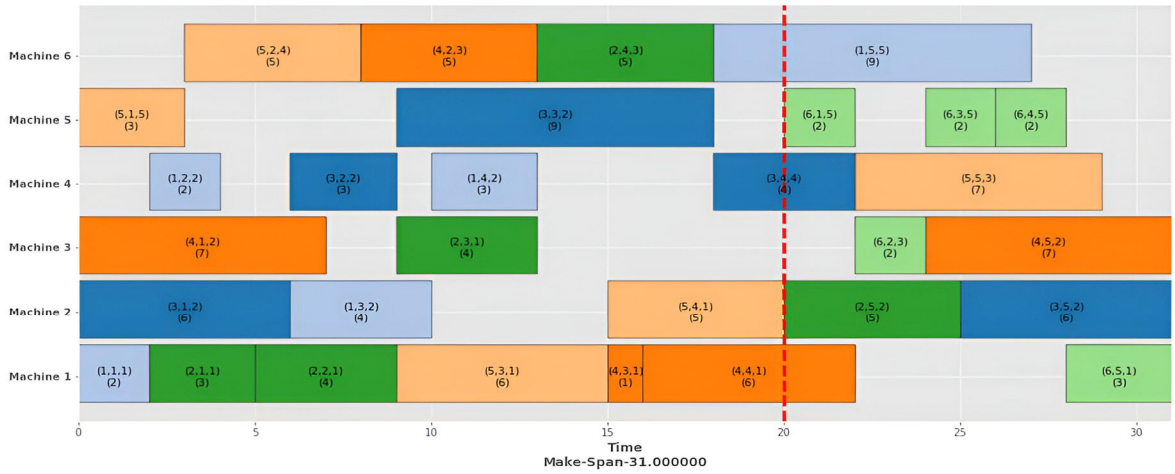

Figure 9. Gantt chart of rescheduling after optimization with the proposed algorithm.

As shown in Table 7, the proposed hybrid algorithm demonstrated better performance in both scheduling and rescheduling when compared to the other methods.

Table 7. Comparison of job arrival events.

Events	Method Proposed in [11]	Proposed HA Method
Int. scheduling makespan	35	29
Makespan before optimization (rescheduling)	50	37
Makespan after optimization (rescheduling)	37	31

6.4. Multiple Machine Breakdowns

To demonstrate this dynamic event, the single-purpose machine dataset mentioned in Table 8 is utilized, consisting of eight jobs and five machine groups, with each machine group allocated a specific set of machines. The first machine group offers four machine choices, the second group provides two choices, the third group presents three choices, the fourth group offers three machine choices, and the fifth machine group provides a choice of four machines. Each machine group is exclusively assigned to execute a specific operation.

Table 8. Single-purpose machine data with 8 jobs and 5 machine groups.

Jobs \ Ops	*OP1* (*MG1*)				*OP2* (*MG2*)		*OP3* (*MG3*)			*OP4* (*MG4*)			*OP5* (*MG5*)			
J1	-	-	-	-	44	42	33	25	36	37	29	27	35	31	33	32
J2	46	41	42	40	18	11	-	-	-	46	43	42	18	28	17	19
J3	22	15	14	19	45	44	41	37	33	43	41	49	-	-	-	-
J4	18	21	24	13	25	23	11	12	21	-	-	-	40	34	38	30
J5	43	38	37	45	37	43	-	-	-	10	15	21	28	40	31	41
J6	46	41	48	38	25	19	-	-	-	13	14	15	28	26	34	27
J7	32	28	21	20	-	-	23	28	26	39	46	41	48	38	39	40
J8	16	17	13	12	22	17	-	-	-	49	42	45	25	33	29	27

"-" shows that the machine cannot proceed with the corresponding operation.

In this dynamic event, multiple machines become unavailable due to service or material unavailability. Then, multiple machines need to break down at a time. Specifically, machines numbers 5, 10, and 15 will be unavailable at the time of 80 units. Figure 10 shows a Gantt chart of static scheduling with a makespan of 163 units, and after rescheduling with the proposed HA, the makespan is 200 units, as demonstrated in the Gantt chart mentioned in Figure 11. Operations O_{52} and O_{73}, which already started on the broken machine before the breakdown time, proceed again following rescheduling.

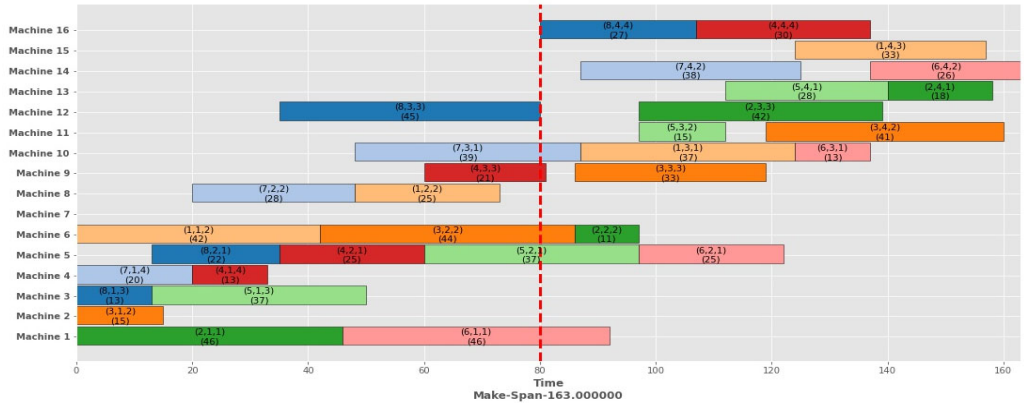

Figure 10. Gantt chart of scheduling before dynamic event.

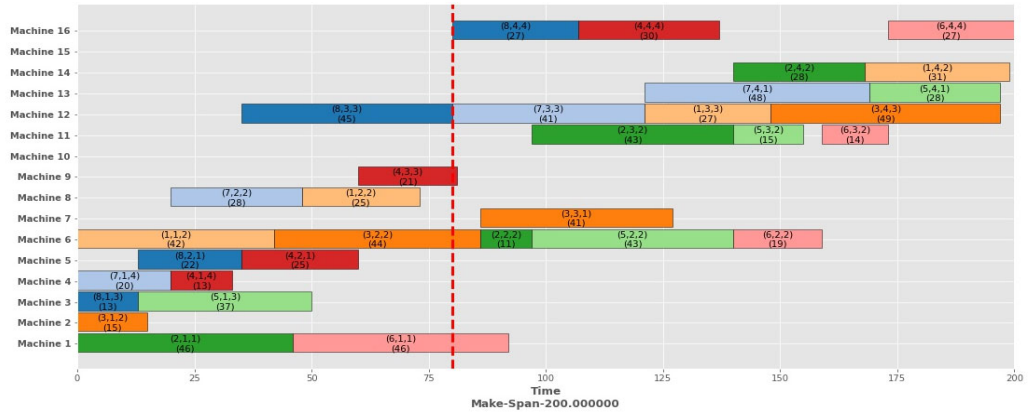

Figure 11. Gantt chart of rescheduling by using HA rescheduling strategy.

6.5. Multiple Job Arrivals

In this dynamic event, an urgent batch of multiple jobs arrives at a specific time. Table 9 shows the processing data of the arrival of multiple jobs consisting of five jobs and five machine groups. The data from Table 8 are utilized for static scheduling. The

batch of five jobs arrives on time with 80 units. Figure 12 shows a Gantt chart of static scheduling with HA of eight jobs before a new batch of jobs arrives, with a makespan of 160 units. Figure 13 illustrates a Gantt chart of rescheduling (without optimization) after the arrival of a new batch of five jobs, with a makespan of 379 units. After applying the proposed rescheduling HA strategy, the makespan is minimized to 217 units, as shown in Figure 14, reaching a makespan from 379 to 217 units over iteration. Figure 14 shows the minimum makespan after every iteration. Figure 15 shows a Gantt chart of the final optimized rescheduling sequence.

Table 9. Single-purpose machine data with 5 jobs and 5 machine groups (job arrivals).

Ops / Jobs	*OP1* (*MG1*)				*OP2* (*MG2*)		*OP3* (*MG3*)			*OP4* (*MG4*)			*OP5* (*MG5*)			
J9	32	35	29	30	46	47	21	16	17	17	16	11	-	-	-	-
J10	19	17	24	22	-	-	29	38	30	47	34	43	13	20	14	16
J11	46	34	44	43	-	-	12	13	14	14	21	18	27	29	39	28
J12	44	47	46	41	29	31	-	-	-	46	40	49	21	32	30	33
J13	40	47	44	49	28	23	28	23	26	26	24	33	15	16	11	19

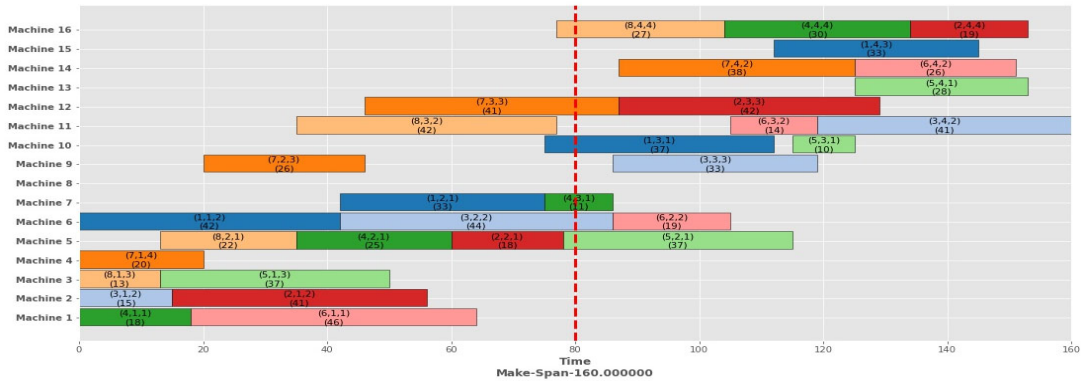

Figure 12. Gantt chart of rescheduling by using HA rescheduling strategy.

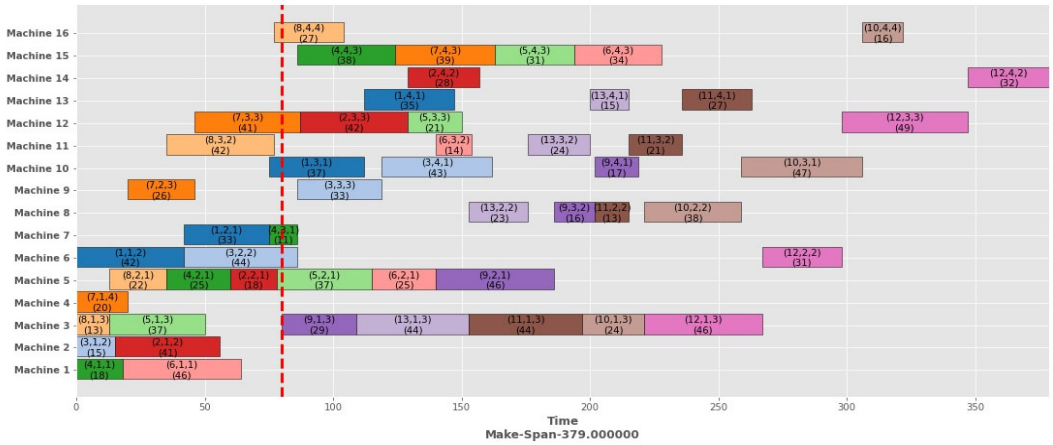

Figure 13. Gantt chart of rescheduling by using HA rescheduling strategy.

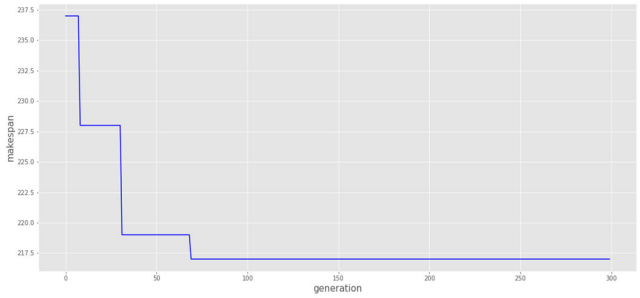

Figure 14. The process of reaching the optimal solution for rescheduling using proposed HA.

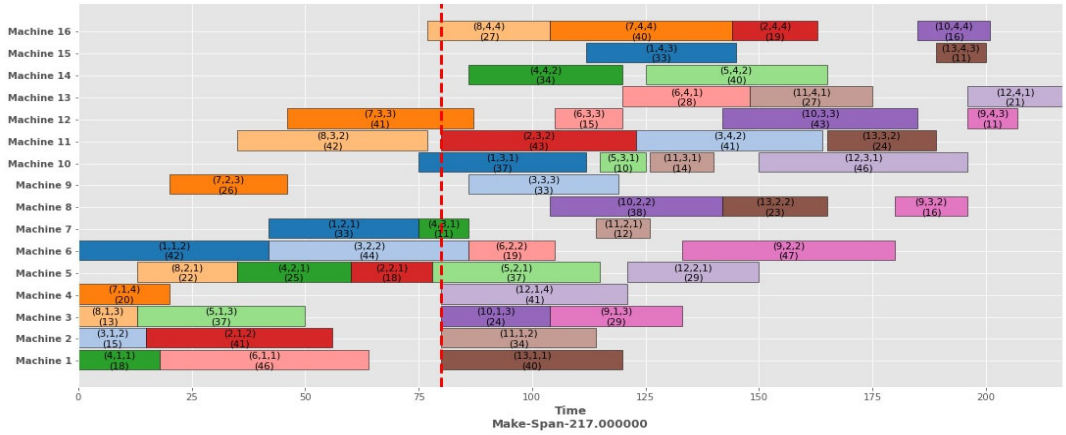

Figure 15. Gantt chart of rescheduling by using HA rescheduling strategy.

7. Conclusions

The DFJSP plays a crucial role in the operation of real-world manufacturing systems. This manuscript proposed a rescheduling method to address dynamic scheduling in the FJSP with the goal of minimizing the makespan. The GA, SA, and VNS were used to obtain optimized solutions. To demonstrate that the proposed hybrid GASAVNS algorithm is efficient, 40 static benchmark datasets were utilized. Out of 40 instances, 38 perform better than the other algorithms mentioned in the study. This study confirms that the proposed rescheduling hybrid GASAVNS algorithm works efficiently for a single machine breakdown, multiple machine breakdowns, a single job arrival, and multiple job arrivals, regardless of whether single-purpose machine or multi-purpose machine datasets are used. For single-machine breakdown, the makespan after scheduling is 29 units. Then, after the dynamic event triggers rescheduling using the proposed method, the makespan decreases to 28 units. This reduction is significant when compared to the other algorithm, which results in a makespan of 35 units for static scheduling and remains at 35 units after rescheduling. For urgent job arrival, the makespan before optimization for rescheduling is 37 units. After being optimized by the proposed algorithm, it reduces to 31 units, which is a better result compared to the other algorithm, where the makespan decreases from 50 units before optimization to 37 units after optimization for rescheduling. For multiple machine breakdowns, the results demonstrate a dynamic event involving the breakdown of three machines. After rescheduling, the makespan is 200 units. The experiment shows that after the breakdown, other jobs efficiently change their positions to achieve an optimized solution. For multiple job arrivals, the makespan after static scheduling is 160 units. Then, after the dynamic event (five jobs arrival) triggers rescheduling, the makespan increases to 379 units before optimization. However, after applying the proposed rescheduling algorithm, the makespan is significantly reduced to 217 units. As a result, the objective of minimizing the makespan is significantly achieved after applying the proposed rescheduling algorithm for the DFJSP.

Author Contributions: Conceptualization, S.K.F. and C.-S.K.; methodology, C.-S.K.; writing—original draft preparation, S.K.F.; writing—review and editing, S.K.F. and C.-S.K.; visualization, S.K.F.; supervision, C.-S.K. All authors have read and agreed to the published version of the manuscript.

Funding: This research was supported by the Basic Science Research Program through the National Research Foundation of Korea (NRF), funded by the Ministry of Education (2021R1I1A3052605).

Data Availability Statement: Data are contained within the article.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Amjad, M.K.; Butt, S.I.; Kousar, R.; Ahmad, R.; Agha, M.H.; Faping, Z.; Anjum, N.; Asgher, U. Recent Research Trends in Genetic Algorithm Based Flexible Job Shop Scheduling Problems. *Math. Probl. Eng.* **2018**, *2018*, 9270802. [CrossRef]
2. Shen, X.N.; Yao, X. Mathematical Modeling and Multi-Objective Evolutionary Algorithms Applied to Dynamic Flexible Job Shop Scheduling Problems. *Inf. Sci.* **2015**, *298*, 198–224. [CrossRef]
3. Garey, M.R.; Johnson, D.D.; Sethi, R. The Complexity of Flowshop and Jobshop Scheduling. Your Use of the JSTOR Archive Indicat. *Math. Oper. Res.* **1976**, *1*, 117–129. Available online: https://www.jstor.org/stable/3689278 (accessed on 28 March 2016). [CrossRef]
4. Mohan, J.; Lanka, K.; Rao, A.N. A Review of Dynamic Job Shop Scheduling Techniques. *Procedia Manuf.* **2019**, *30*, 34–39. [CrossRef]
5. Al-Hinai, N.; Elmekkawy, T.Y. Robust and Stable Flexible Job Shop Scheduling with Random Machine Breakdowns Using a Hybrid Genetic Algorithm. *Int. J. Prod. Econ.* **2011**, *132*, 279–291. [CrossRef]
6. Wang, Z.; Zhang, J.; Si, J. *Dynamic Job Shop Scheduling Problem with New Job Arrivals: A Survey*; Springer: Singapore, 2020; Volume 586, ISBN 9789813290495.
7. Rahmani, D.; Ramezanian, R. A Stable Reactive Approach in Dynamic Flexible Flow Shop Scheduling with Unexpected Disruptions: A Case Study. *Comput. Ind. Eng.* **2016**, *98*, 360–372. [CrossRef]
8. Valledor, P.; Gomez, A.; Priore, P.; Puente, J. Solving Multi-Objective Rescheduling Problems in Dynamic Permutation Flow Shop Environments with Disruptions. *Int. J. Prod. Res.* **2018**, *56*, 6363–6377. [CrossRef]
9. Zhang, G.; Gao, L.; Shi, Y. An Effective Genetic Algorithm for the Flexible Job-Shop Scheduling Problem. *Expert Syst. Appl.* **2011**, *38*, 3563–3573. [CrossRef]
10. Luo, X.; Qian, Q.; Fu, Y.F. Improved Genetic Algorithm for Solving Flexible Job Shop Scheduling Problem. *Procedia Comput. Sci.* **2020**, *166*, 480–485. [CrossRef]
11. Wang, L.; Luo, C.; Cai, J. A Variable Interval Rescheduling Strategy for Dynamic Flexible Job Shop Scheduling Problem by Improved Genetic Algorithm. *J. Adv. Transp.* **2017**, *2017*, 1527858. [CrossRef]
12. De Giovanni, L.; Pezzella, F. An Improved Genetic Algorithm for the Distributed and Flexible Job-Shop Scheduling Problem. *Eur. J. Oper. Res.* **2010**, *200*, 395–408. [CrossRef]
13. Lim, K.C.W.; Wong, L.P.; Chin, J.F. Simulated-Annealing-Based Hyper-Heuristic for Flexible Job-Shop Scheduling. *Eng. Optim.* **2022**, *55*, 1635–1651. [CrossRef]
14. Saidi-Mehrabad, M.; Fattahi, P. Flexible Job Shop Scheduling with Tabu Search Algorithms. *Int. J. Adv. Manuf. Technol.* **2007**, *32*, 563–570. [CrossRef]
15. Han, B.A.; Yang, J.J. Research on Adaptive Job Shop Scheduling Problems Based on Dueling Double DQN. *IEEE Access* **2020**, *8*, 186474–186495. [CrossRef]
16. Li, X.; Gao, L. An Effective Hybrid Genetic Algorithm and Tabu Search for Flexible Job Shop Scheduling Problem. *Int. J. Prod. Econ.* **2016**, *174*, 93–110. [CrossRef]
17. Escamilla-Serna, N.J.; Seck-Tuoh-Mora, J.C.; Medina-Marin, J.; Barragan-Vite, I.; Corona-Armenta, J.R. A Hybrid Search Using Genetic Algorithms and Random-Restart Hill-Climbing for Flexible Job Shop Scheduling Instances with High Flexibility. *Appl. Sci.* **2022**, *12*, 8050. [CrossRef]
18. Tang, J.; Zhang, G.; Lin, B.; Zhang, B. A Hybrid Algorithm for Flexible Job-Shop Scheduling Problem. *Procedia Eng.* **2011**, *15*, 3678–3683. [CrossRef]
19. Zhang, L.; Gao, L.; Li, X. A Hybrid Genetic Algorithm and Tabu Search for a Multi-Objective Dynamic Job Shop Scheduling Problem. *Int. J. Prod. Res.* **2013**, *51*, 3516–3531. [CrossRef]
20. Shahgholi Zadeh, M.; Katebi, Y.; Doniavi, A. A Heuristic Model for Dynamic Flexible Job Shop Scheduling Problem Considering Variable Processing Times. *Int. J. Prod. Res.* **2019**, *57*, 3020–3035. [CrossRef]
21. Zhang, C.; Zhou, Y.; Peng, K.; Li, X.; Lian, K.; Zhang, S. Dynamic Flexible Job Shop Scheduling Method Based on Improved Gene Expression Programming. *Meas. Control* **2021**, *54*, 1136–1146. [CrossRef]
22. Fattahi, P.; Fallahi, A. Dynamic Scheduling in Flexible Job Shop Systems by Considering Simultaneously Efficiency and Stability. *CIRP J. Manuf. Sci. Technol.* **2010**, *2*, 114–123. [CrossRef]
23. Kundakci, N.; Kulak, O. Hybrid Genetic Algorithms for Minimizing Makespan in Dynamic Job Shop Scheduling Problem. *Comput. Ind. Eng.* **2016**, *96*, 31–51. [CrossRef]

24. Wei, H.; Li, S.; Jiang, H.; Hu, J.; Hu, J. Hybrid Genetic Simulated Annealing Algorithm for Improved Flow Shop Scheduling with Makespan Criterion. *Appl. Sci.* **2018**, *8*, 2621. [CrossRef]
25. Al-Milli, N.R. Hybrid Genetic Algorithms with Simulating Annealing for University Course Timetabling Problems Publication of Little Lion Scientific R & D, Islamabad Pakistan. *J. Theor. Appl. Inf. Technol.* **2011**, *29*, 100–106.
26. Shady, S.; Kaihara, T.; Fujii, N.; Kokuryo, D. Evolving Dispatching Rules Using Genetic Programming for Multi-Objective Dynamic Job Shop Scheduling with Machine Breakdowns. *Procedia CIRP* **2021**, *104*, 411–416. [CrossRef]
27. Pocol, C.B.; Stanca, L.; Dabija, D.C.; Câmpian, V.; Mișcoiu, S.; Pop, I.D. A QCA Analysis of Knowledge Co-Creation Based on University–Industry Relationships. *Mathematics* **2023**, *11*, 388. [CrossRef]
28. Pelau, C.; Dabija, D.C.; Ene, I. What Makes an AI Device Human-like? The Role of Interaction Quality, Empathy and Perceived Psychological Anthropomorphic Characteristics in the Acceptance of Artificial Intelligence in the Service Industry. *Comput. Human Behav.* **2021**, *122*, 106855. [CrossRef]
29. Gholizadeh, H.; Fazlollahtabar, H.; Fathollahi-Fard, A.M.; Dulebenets, M.A. Preventive Maintenance for the Flexible Flowshop Scheduling under Uncertainty: A Waste-to-Energy System. *Environ. Sci. Pollut. Res.* **2021**, *28*, 1–20. [CrossRef] [PubMed]
30. Fatemi-Anaraki, S.; Tavakkoli-Moghaddam, R.; Foumani, M.; Vahedi-Nouri, B. Scheduling of Multi-Robot Job Shop Systems in Dynamic Environments: Mixed-Integer Linear Programming and Constraint Programming Approaches. *Omega* **2023**, *115*, 102770. [CrossRef]
31. Madhav, S.; Ahamad, A.; Singh, P.; Mishra, P.K. A Review of Textile Industry: Wet Processing, Environmental Impacts, and Effluent Treatment Methods. *Environ. Qual. Manag.* **2018**, *27*, 31–41. [CrossRef]
32. He, Z.; Xu, J.; Tran, K.P.; Thomassey, S.; Zeng, X.; Yi, C. Modeling of Textile Manufacturing Processes Using Intelligent Techniques: A Review. *Int. J. Adv. Manuf. Technol.* **2021**, *116*, 39–67. [CrossRef]
33. Hurink, J.; Jurisch, B.; Thole, M. Tabu Search for the Job-Shop Scheduling Problem with Multi-Purpose Machines. *OR Spektrum* **1994**, *15*, 205–215. [CrossRef]
34. Shao, X.; Kshitij, F.S.; Kim, C.S. GAILS: An Effective Multi-Object Job Shop Scheduler Based on Genetic Algorithm and Iterative Local Search. *Sci. Rep.* **2024**, *14*, 2068. [CrossRef]
35. Pěnička, R.; Faigl, J.; Saska, M. Variable Neighborhood Search for the Set Orienteering Problem and Its Application to Other Orienteering Problem Variants. *Eur. J. Oper. Res.* **2019**, *276*, 816–825. [CrossRef]
36. Liu, Q.; Li, X.; Gao, L.; Li, Y. A Modified Genetic Algorithm with New Encoding and Decoding Methods for Integrated Process Planning and Scheduling Problem. *IEEE Trans. Cybern.* **2021**, *51*, 4429–4438. [CrossRef] [PubMed]
37. Fathollahi-Fard, A.M.; Woodward, L.; Akhrif, O. Sustainable Distributed Permutation Flow-Shop Scheduling Model Based on a Triple Bottom Line Concept. *J. Ind. Inf. Integr.* **2021**, *24*, 100233. [CrossRef]
38. Kolonko, M. Some New Results on Simulated Annealing Applied to the Job Shop Scheduling Problem. *Eur. J. Oper. Res.* **1999**, *113*, 123–136. [CrossRef]
39. Liu, W.; Dridib, M.; Fathollahi-Fard, A.M.; El Hassani, A.H. A Customized Adaptive Large Neighborhood Search Algorithm for Solving a Multi-Objective Home Health Care Problem in a Pandemic Environment. *Swarm Evol. Comput.* **2024**, *86*, 101507. [CrossRef]
40. Lawrence, S. *Resouce Constrained Project Scheduling: An Experimental Investigation of Heuristic Scheduling Techniques (Supplement)*; Graduate School of Industrial Administration, Carnegie-Mellon University: Pittsburgh, PA, USA, 1984.
41. Wu, Z.; He, H.; Huang, C. Flexible Job Shop Dynamic Scheduling Problem Research with Machine Fault. *Mach. Des. Res.* **2015**, *31*, 94–98. [CrossRef]