TWO-STAGE DYNAMIC SCHEDULING MODEL FOR A FLEXIBLE FLOW SHOP USING A GENETIC ALGORITHM: A CASE STUDY IN A TRUCK BODY MANUFACTURING COMPANY

Huynh Nhat Trieu[1], Nguyen Hong Phuc[2]*, Phan Thi Mai Ha[3]

[1]*Faculty of Engineering and Technology, Binh Duong Economics and Technology University, Vietnam*
[2]*Faculty of Industrial Management, College of Engineering, Can Tho University, Vietnam*
[3]*Department of Industrial System Engineering, Ho Chi Minh City University of Technology, Vietnam*

*Corresponding author: nguyenhongphuc@ctu.edu.vn

(Received: December 17, 2025; Revised: January 28, 2026; Accepted: March 11, 2026)

DOI: 10.31130/ud-jst.2026.24(3).721E

Abstract - Dynamic flexible flow shop scheduling problems present a complex challenge, where new orders arrive stochastically during the current schedule; this uncertainty degrades schedule stability and resource utilization, highlighting the need for responsive scheduling mechanisms. This study proposes a two-stage scheduling framework. The "master schedule" drives global coordination across all stages, while the "reactive" schedule enables immediate rescheduling decisions in response to uncertain job arrivals within the current "master schedule" through the use of a dispatching rule. The problem is modeled using mixed-integer programming (MIP). The objective is to minimize the makespan, and the problem is solved by a genetic algorithm (GA). Experimental results demonstrate that the proposed model outperforms traditional dispatching rules. It achieves a makespan reduction of up to 4.9%, a tardiness reduction of over 40%, and a flow time improvement of over 24%. The study provides a practical and scalable solution for dynamic flexible flow shop environments.

Key words - Flexible Flow Shop Scheduling; Dynamic Scheduling; Metaheuristic Algorithms; Genetic Algorithm; Rescheduling

1. Introduction

Scheduling is one of the most crucial aspects of manufacturing operations, which involves assigning resources such as materials, equipment, tools, and manpower to specific jobs to achieve targeted outcomes [1], [2]. A well-organized schedule helps organizations boost productivity and resource utilization while meeting the real constraints of manufacturing environments that face increasing market pressure for shorter lead times and diverse product specifications. Furthermore, a good schedule stabilizes the production flow, especially in environments where demand is uncertain and delivery commitments are tight.

The flexible flow shop scheduling (FFSS) problem represents a multistage production process consisting of two or more sequential stages, as illustrated in Figure 1. Each stage contains at least one machine, with at least one stage having multiple machines. One machine is selected to process a given operation at each stage. FFSS problems aim to optimize sequencing and assignments to improve makespan, tardiness, and idleness, bound by capacity, precedence, and time constraints. In dynamic environments, these systems encounter additional disruptions, including stochastic job arrivals, varying processing times, and workload imbalances. As a result, dynamic flexible flow shop scheduling (DFFSS) requires not only optimizing sequencing but also enabling prompt reactive adjustments to maintain performance [3]-[5].

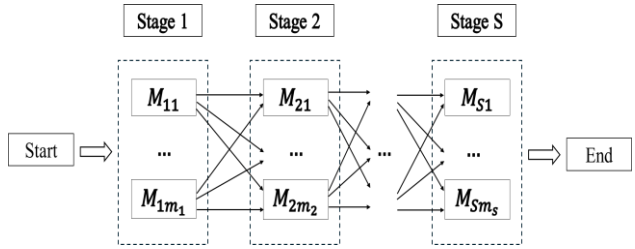

Figure 1. *A flexible flow shop layout*

In a truck body manufacturing company, customer orders appear unpredictably, differing greatly in size, priority, and processing requirements. Such randomness leads to idle times, bottlenecks, delays, and unstable utilization across stages. Existing reactive rescheduling techniques, such as right-shift or simple event-driven insertion, can mitigate disruptions but often lack alignment with long-term production planning.

To address these limitations, this study proposes a two-stage dynamic scheduling framework consisting of: (1) a periodic master schedule that provides global coordination over a fixed horizon; and (2) an event-driven reactive scheduling mechanism that inserts newly arriving jobs without reconstructing the master plan. A GA is used to generate high-quality master schedules, while dispatching rules support fast adjustments. This approach aims to improve makespan, tardiness, and flow time while ensuring responsiveness in dynamic multi-stage production systems.

2. Literature review

DFFSS is a critical and complex area of study in operations research and industrial engineering, focusing on optimizing production processes in environments characterized by flexibility and unpredictability. In DFFSS problems, jobs can be processed on multiple machines at each stage, adding significant complexity to scheduling decisions. The dynamic nature of DFFSS further heightens the challenge, as instantaneous adjustments must be made to accommodate unforeseen disruptions, such as machine breakdowns, urgent job arrivals, or variations in processing times [6] - [8].

DFFSS are NP-hard problems, presenting significant computational challenges and demanding continued research into more efficient, adaptive, and scalable algorithms. GAs have proven remarkably effective for complex scheduling challenges [9]. Their evolutionary framework, encompassing selection, crossover, and mutation, enables them to adapt readily to real-time disturbances such as new job arrivals, machine failures, and fluctuating processing times. GAs can systematically identify highly flexible and efficient scheduling solutions by encoding machine assignments in their chromosome structures [10], [11]. The algorithms minimize makespan, reduce tardiness, and maintain strong performance in dynamic environments where production constraints and objectives shift continually. This adaptability, coupled with their capacity to manage multiple objectives simultaneously, confirms GAs as a compelling choice for optimizing DFFSS [6], [12], [13].

Table 1. *Comparison of rescheduling strategies for DFFSS problems*

Studies	Disturbance type	Right shift rescheduling	Event-driven rescheduling	Utilizing available resources
Tang et al. [8]	Machine Breakdown	X		
Tang et al. [14]	Time delays & new job insertions		X	
Ren and Liu [15]	Machine Breakdown	X		
Lu et al. [16]	Machine Breakdown	X		
Wang, Zhang and Si [17]	New Job Arrivals	X	X	
Nguyen, Ngo and Nguyen [18]	New Job Arrivals		X	

Previous studies have primarily focused on rescheduling when disturbances like machine breakdowns or new job arrivals occur, by reactive methods like right-shift rescheduling or event-driven adjustment [6], [19]. While these methods are effective in mitigating the immediate effects of dynamic events, they often operate in isolation from long-term production planning and scheduling. Table 1 reveals that although various rescheduling strategies have been developed, the majority lack mechanisms for strategic resource utilization during disruptions. Specifically, there is limited research that integrates rescheduling with a global view of available capacity and machine balancing. In addition, many of these studies rely heavily on reactive or rule-based mechanisms without explicitly linking local rescheduling decisions to an underlying optimization framework. As a result, rescheduling actions are often myopic, focusing on immediate feasibility rather than system-wide performance over the planning horizon. The absence of a clear connection between long-term scheduling models and real-time adjustment strategies limits the ability of existing approaches to balance responsiveness and global efficiency in dynamic flexible flow shop environments. Overall, many existing models only use simple dispatching rules or fixed schedules.

To bridge this gap, this study proposes a novel two-stage scheduling model that combines:

(1) A periodic "master schedule" for global coordination, which provides a high-level production plan and resource allocation over a fixed planning horizon.

(2) An event-driven reactive schedule to handle unforeseen disruptions (e.g., new job arrivals) in a responsive and localized manner, while preserving the integrity of the master plan.

This two-layer architecture enables the scheduling system to be adaptive, ensuring improved responsiveness, resource utilization, and overall scheduling performance in dynamic manufacturing environments. The integration of GA with this two-stage scheduling framework represents a practical and scalable approach to addressing the DFFSP more comprehensively than existing solutions.

3. Problem formulation

3.1. Problem description

This case study focuses on a truck body manufacturing system. The workflow encompasses multiple sequential stages, including bending, processing, rough assembly, painting, base vehicle assignment, specialized equipment installation, electrical installation, and finishing, as shown in Figure 2. In this case study, we apply the proposed model to schedule 71 jobs, comprising three job types, across seven stages, each with two machines, in a manufacturing environment. Table 2 provides the processing routes and processing times for each job type. The primary objective in this scenario is to minimize makespan, ensuring efficient utilization of resources.

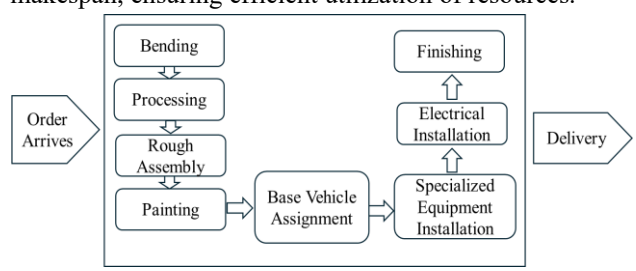

Figure 2. *Flowchart of the production of specialized equipment*

Table 2. *The processing routes and processing times (hours) for three job types*

Job type	Stage 0		Stage 1		Stage 2		Stage 3		Stage 4		Stage 5		Stage 6	
	M0	M1	M0	M1	M0	M1	M0	M1	M0	M1	M0	M1	M0	M1
1	3	2	4	5	2	1	3	4	2	3	4	3	3	5
2	4	3	0	0	4	5	4	2	2	3	5	3	4	6
3	3	5	2	1	0	0	6	8	5	3	4	6	5	4

The proposed DFFSS problem introduces dynamic characteristics to the static FFSS, specifically unpredictable job arrivals that interfere with the initially created schedule. In the static FFSS, the jobs are priorly known, and the schedule remains constant after generation. In contrast, the DFFSS problem assumes jobs arrive randomly over time, necessitating a two-stage framework: (1) a periodic master schedule, generated every 7 days to provide global coordination and resource allocation over a fixed period; and (2) a reactive schedule, triggered by events (e.g., new arrivals) and using dispatching rules

(e.g., First Come First Serve, Earliest Due Date) to insert jobs locally without altering the master schedule. In this approach, other dynamic variables, such as machine breakdowns, are not considered, as the stochastic job arrival was the core problem analyzed in the case study.

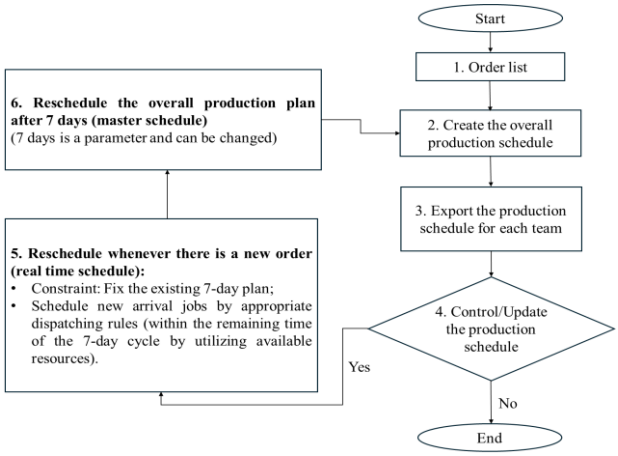

Figure 3. *The proposed scheduling approach*

Figure 3 illustrates the proposed rescheduling process, as follows.

Step 1: Compilation of the order list. The rescheduling process begins by compiling an order list, which helps collect customer orders and understand current workloads for more effective planning.

Step 2: Create the overall production schedule. This schedule assigns jobs and machines based on the order list and current system status. It works as the master plan for the next 7-day period, helping to organize production activities and balance resource usage across all stages.

Step 3: Export the production schedule for each team. Each production team receives a detailed schedule with their assigned jobs, machines, and deadlines.

Step 4: Control/update the production schedule. The schedule is regularly checked to detect delays or changes in production. If problems arise, updates are made to keep the schedule aligned with actual conditions.

Step 5: "Reactive schedule" for new orders. This step plays a key role in making the production system more flexible. When new job orders arrive unexpectedly, the reactive scheduling mechanism is triggered. Unlike the master schedule, which is fixed for a given period (e.g., 7 days), the reactive schedule allows the system to respond immediately. Based on the current status of machines and jobs, dispatching rules such as First Come First Serve (FCFS), Earliest Due Date (EDD), Shortest Processing Time (SPT), and Longest Processing Time (LPT) are used to insert the new job into the existing schedule with minimal disruption. This rapid response helps avoid bottlenecks, reduces idle time, and ensures on-time delivery for both existing and new jobs. This ensures that the system remains responsive and can adapt to changes without requiring a complete rebuild of the entire schedule.

Step 6: Periodically reschedule the overall production plan. After completing each 7-day cycle, the process undertakes a comprehensive review and rescheduling of the overall production plan, referred to as the **"master schedule".** The periodic rescheduling is designed to realign the production plan in response to cumulative changes and evolving demands.

This two-stage scheduling approach, comprising a "reactive schedule" and a "master schedule", ensures that the production schedule remains both structured and adaptable, effectively balancing routine planning with the capacity to respond promptly to emergent demands. The process concludes once all required adjustments to the production schedule have been finalized, resulting in a dynamic and resilient scheduling system tailored to the complexities of the DFFSS.

The GA is applied only in the master scheduling stage, which defines a stable production plan over a fixed planning horizon. Re-optimizing the schedule each time a new job arrives is not appropriate, as frequent changes to the master plan would propagate disruptions to workforce allocation, material preparation, and upstream coordination. Therefore, the master schedule is intentionally kept stable, while dynamic job arrivals are handled through localized adjustments without reconstructing the global schedule.

3.2. Problem formulation

Assumptions:

- Interruptions or preemptive scheduling of operations are prohibited.

- Machines can encounter new jobs at any point following the rescheduling process.

- The processing time for each operation on any machine is known.

- Set-up times are small and included in the processing times.

- There is unlimited intermediate storage available between machines.

Set of indices

j Job index (j = 1, 2,..., J)

l Stage index (l = 1, 2,..., L)

k Machine index at each stage (k = 1, 2,..., K)

Parameters

p_{jlk}: Processing time of job j at stage l on machine k

M: A sufficiently large number (big-M)

Decision Variables

$$x_{jlk} = \begin{cases} 1, \text{if job } j \text{ is assigned to machine } k \text{ at stage } l \\ 0, \text{otherwise} \end{cases}$$

s_{jl}: Start time of job j at stage l

$$y_{ijlk} = \begin{cases} 1, \text{if job } i \text{ is processed before job } j \text{ on machine } k \text{ at stage } l \\ 0, \text{otherwise} \end{cases}$$

Objective function

$$\text{Min } C_{max} \quad (1)$$

Constraints

$$\sum_{k=1}^{K} x_{jlk} = 1, \forall j, l \quad (2)$$

$$s_{jl} \geq s_{jl'} + p_{jl'k}, \forall j, k, l, l' \text{ and } l > l' \quad (3)$$

$$s_{il} + p_{ilk} \leq s_{jl} + M(1 - y_{ijlk}), \forall i, j, k, l \text{ and } i \neq j \quad (4.1)$$

$$s_{jl} + p_{jlk} \leq s_{il} + M y_{ijlk}, \forall i, j, k, l \text{ and } i \neq j \quad (4.2)$$

$$y_{ijlk} \geq x_{ilk} + x_{jlk} - 1, \forall i, j, k, l \text{ and } i \neq j \quad (4.3)$$

$$C_{max} \geq s_{jl} + p_{jlk}, \forall j, k, l \quad (5)$$

Description of constraints:

(1) The objective is to minimize the makespan;

(2) Each job is assigned to only one machine at each stage.

(3) The start time of job j at stage l is at least equal to the completion time of the same job at the previous stage.

(4) No two jobs overlap on the same machine at the same stage.

(5) Makespan is the maximum completion time among all jobs after processing through all stages.

3.3. Genetic Algorithm

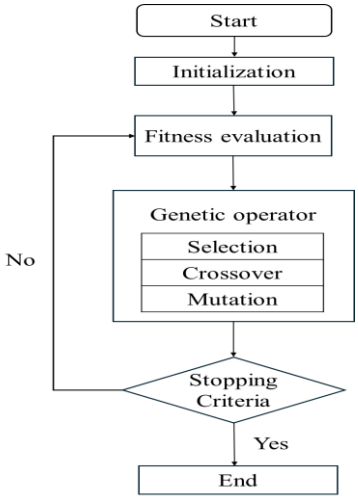

Figure 4. *The flowchart of the Genetic Algorithm*

GA is employed to solve the proposed scheduling model. The algorithm starts with a randomly generated population of candidate solutions. Each solution is evaluated using a fitness function, such as makespan. Based on fitness, better solutions are selected to form the next generation. Crossover combines parts of chosen individuals to generate new solutions, while mutation introduces random changes to maintain diversity and prevent the solution from becoming trapped in a local optimum. Through repeated selection, crossover, and mutation, the GA evolves toward near-optimal solutions for complex scheduling problems (Figure 4).

3.3.1. Structure of the chromosome

Chromosomes are encoded as permutations representing job sequences that respect precedence constraints; for instance, in a scenario with nine jobs across two machines, Figure 5 depicts a solution vector combining these jobs. The chromosome focuses solely on job sequencing.

Job 1	Job 3	Job 5	Job 2	Job 4	Job 8	Job 6	Job 9	Job 7
1	3	5	2	4	8	6	9	7

Figure 5. *An example of solution representation*

3.3.2. Crossover

The crossover strategy in this study is an order-based crossover. As illustrated in Figure 6, it generates offspring by taking a segment from Parent 1 and placing it into the offspring at the same positions. Then, it fills the remaining positions with genes from Parent 2 that have not been used yet, in the order they appear in Parent 2. This approach ensures the production of valid offspring while effectively combining parental traits.

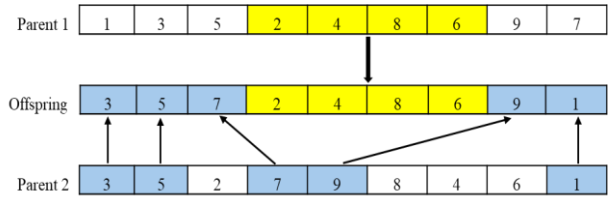

Figure 6. *An example of an order-based crossover operator*

3.3.3. Mutation

The mutation strategy employed in this study is a swap mutation, involving the random selection of two distinct positions within the chromosome, each corresponding to a different job, followed by the swapping of the jobs at these positions (Figure 7). By selectively swapping different jobs, this method introduces genetic diversity without disrupting the internal stage order of each job.

In the flexible flow shop considered in this study, all jobs follow the same fixed processing order across stages. The chromosome encodes only the job sequence; thus, crossover and mutation operators modify job positions without changing the internal operation order of any job. Consequently, all offspring solutions automatically satisfy precedence constraints, and no repair mechanism is required.

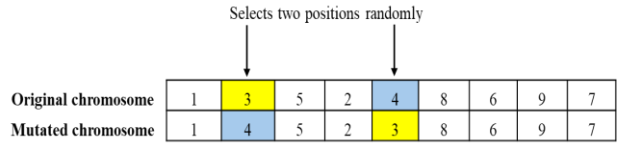

Figure 7. *Mutation operator in genetic algorithm*

4. Experiments and results

To evaluate the performance of the proposed two-stage scheduling model, a simulation program was developed based on the operational characteristics of a truck body manufacturing system. The simulation considers a total of 71 jobs (with three distinct job types) processed across seven production stages, each equipped with two identical parallel machines. Jobs arrive randomly over time and differ in routing sequences, processing times on each machine, and individual due dates, reflecting the dynamic nature of real production environments. Machine conditions are assumed to be ideal, with full availability, and job processing is non-preemptive.

This section is divided into three parts: (1) tuning the GA's parameters to improve its efficiency for the proposed DFFSS; (2) applying the proposed model to a case study to illustrate its application; and (3) evaluating the proposed model's performance. All experimental tests were coded in Python and executed on a PC equipped with an Intel Core i5-12500 processor operating at 3.00 GHz and 16 GB of RAM.

4.1. Parameter tuning for GA

In this experiment, four key parameters of the GA, namely, population size, crossover rate, mutation rate, and

selection rate, were selected as experimental factors. The first three factors were tested at three levels, and the last factor was examined at two levels, as shown in Table 3.a. A full factorial design was employed, yielding a total of 72 combinations. Each combination was replicated 50 times. The number of generations was fixed at 500 for all runs. The fitness evolution chart, as shown in Figure 8, illustrates the performance trajectory of the GA over 500 generations.

Table 3. *Design of experiment and fitness values*

a. Tuning parameters

Factors	**Levels**
Population size	50, 100, 200
Crossover rate	0.6, 0.7, 0.8, 0.9
Mutation rate	0.05, 0.1, 0.15
Selection rate	0.05, 0.1

b. ANOVA analysis for fitness values

Source	**DF**	**Adj SS**	**Adj MS**	**F-Value**	**P-Value**
Population size	2	2914.78	1457.39	70.58	0.000
Crossover rate	3	84.67	28.22	1.37	0.267
Mutation rate	2	351.44	175.72	8.51	0.001
Selection rate	1	410.89	410.89	19.90	0.000
Population size*Crossover rate	6	27.00	4.50	0.22	0.969
Population size*Mutation rate	4	48.89	12.22	0.59	0.670
Population size*Selection rate	2	63.44	31.72	1.54	0.228
Crossover rate*Mutation rate	6	195.67	32.61	1.58	0.178
Crossover rate*Selection rate	3	74.89	24.96	1.21	0.319
Mutation rate*Selection rate	2	31.44	15.72	0.76	0.474
Error	40	826.00	20.65		
Total	71	5029.11			

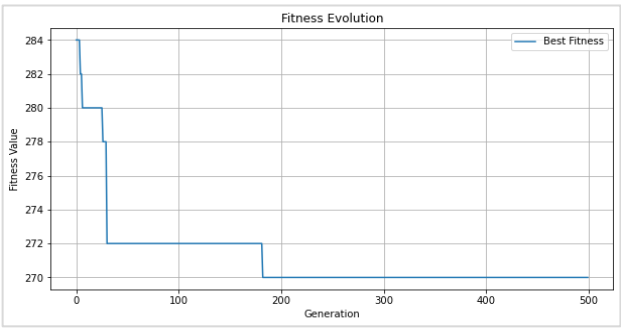

Figure 8. *Fitness evolution of the proposed GA*

The experimental results suggest the following: First, from the ANOVA table (Table 3.b) it implies that population size, mutation rate and selection rate significantly influence the fitness value. Moreover, the ANOVA results indicate that the interaction effects among GA parameters are not statistically significant (p-values > 0.05), suggesting that each parameter mainly affects the fitness value independently. Second, the main effects plot (Figure 9) and interaction plot (Figure 10) reveal that higher levels of population size, mutation rate, and selection rate will improve the fitness values. In conclusion, the selected parameter levels are a population size of 200, a crossover rate of 0.8, a mutation rate of 0.15, and a selection rate of 0.1. Thus, these values will be applied to the following experiments.

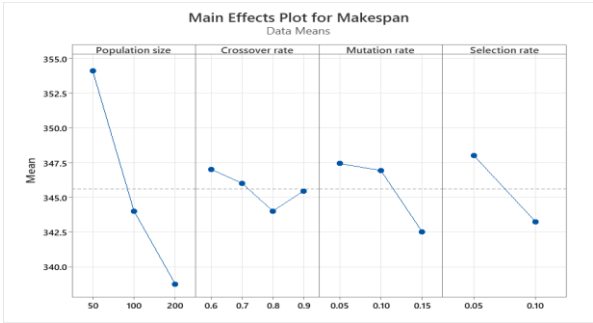

Figure 9. *Main effects plot for fitness value*

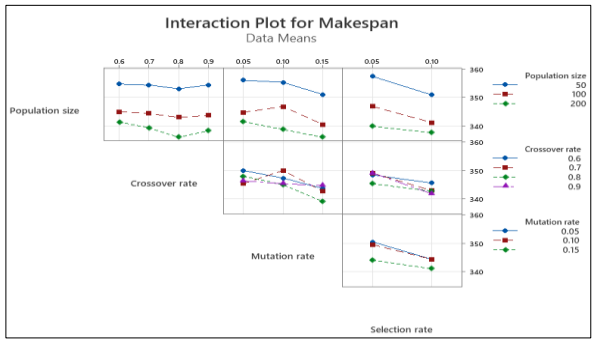

Figure 10. *Interaction plot for fitness value*

4.2. A case study of a truck body manufacturing system

A simulation program was developed to evaluate the performance of the proposed two-stage scheduling model in a truck body manufacturing system. During the simulation, a total of 71 jobs are generated and processed. These jobs arrive randomly over time and differ in several aspects, including routing sequences, processing times on each machine, and individual deadlines. This setting aims to replicate real-world production conditions, where variability and uncertainty frequently occur.

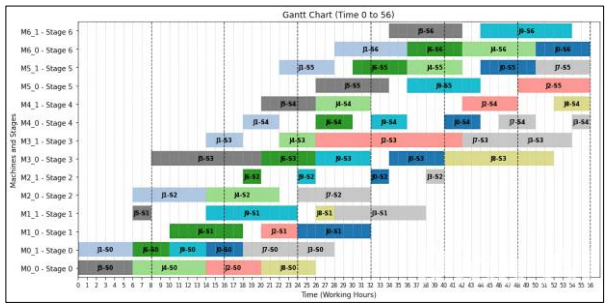

Figure 11. *The schedule for the first week*

Figure 11 presents the Gantt chart generated from the master schedule, which includes 14 jobs planned over the first 7-day production period. Subsequently, Figure 12 shows how the system adapts when three new jobs (J15, J16, and J17) arrive unexpectedly during execution. Instead of reconstructing the entire schedule, the system applies a reactive scheduling rule to insert new jobs into available machine slots while keeping the original job sequence unchanged. The FCFS rule is selected due to its simplicity.

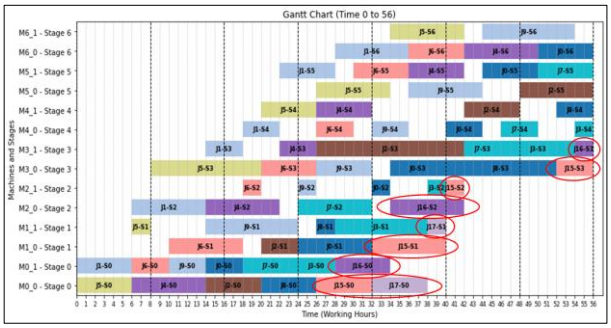

Figure 12. *The schedule of the first week with the arrival of new jobs*

4.3. Performance comparisons

4.3.1. Comparison with fixed periodic scheduling approaches

In this section, the proposed two-stage scheduling model is compared with several benchmark approaches under a fixed scheduling policy. Specifically, a fixed rescheduling cycle of seven days is adopted, meaning that the production schedule is globally regenerated only at the beginning of each seven-day planning horizon. All customer orders arriving within a cycle are accumulated and postponed until the next rescheduling point. Under this setting, conventional dispatching rules such as FCFS, EDD, SPT, and LPT, as well as a GA-based flexible flow shop scheduling model (FFS+GA), are employed to generate schedules at each rescheduling epoch.

Table 4 compares the performance of the proposed two-stage model with such fixed periodic approaches. The results show that the proposed model consistently outperforms fixed periodic scheduling models across all evaluation criteria. Although FFS+GA achieves a comparable makespan, it suffers from significantly higher total tardiness and flow time, indicating limited responsiveness to dynamic job arrivals within the planning horizon as reflected by an increase in total tardiness from 906 hours (proposed model) to 1,587 hours in FFS+GA, caused by postponing newly arriving jobs until the next rescheduling point. FFS+GA lacks flexibility in handling local operational variations. By integrating a dispatching rule in the second stage, the proposed model enhances local adaptability without sacrificing the global schedule achieved by the GA. Similarly, dispatching-based fixed periodic schedules (FCFS, EDD, SPT, and LPT) exhibit longer completion times and substantial increases in tardiness and flow time.

Table 4. *Compared to fixed periodic scheduling models*

Scheduling approaches	**Makespan** *(hours)*	**Total tardiness** *(hours)*	**Total flow time** *(hours)*	**% Difference in makespan** *(compared to *)*	**% Difference in tardiness** *(compared to *)*	**% Difference in flow time** *(compared to *)*
Proposed model *(*)*	**410**	**906**	**3,815**			
FFS by GA	410	1,587	4,523	0.0	75.0	18.6
FCFS	422	1,932	4,851	3. 0	113.2	27.0
EDD	424	1,972	4,913	3.4	117.7	28.8
SPT	424	1,954	4,887	3.4	115.7	28.1
LPT	424	2,128	5,055	3.4	135.0	32.5

4.3.2. Comparison with rule-based two-stage scheduling approaches

For the two-stage heuristic-based models, the same dispatching rule is consistently applied in both stages. Specifically, FCFS, SPT, EDD, and LPT are used to generate the periodic master schedule in stage 1 and are again employed to sequence and assign newly arriving jobs in stage 2. No optimization model or metaheuristic is involved in either stage, and scheduling decisions are entirely rule-based.

Table 5. *Experimental results from applying a two-stage scheduling approach*

Scheduling approaches	**Makespan** *(hours)*	**Total tardiness** *(hours)*	**Total flow time** *(hours)*	**% Difference in makespan** *(compared to *)*	**% Difference in tardiness** *(compared to *)*	**% Difference in flow time** *(compared to *)*
Proposed model *(*)*	**410**	**906**	**3,815**			
FCFS	414	1,258	4,101	0.9	38.8	7.5
EDD	422	1,568	4,503	2.9	73.0	18.0
SPT	424	1,270	4,125	3.4	40.0	8.1
LPT	430	2,121	5,027	4.9	134.0	31.8

As reported in Table 5, although all methods benefit from event-driven job insertion, their performance varies significantly depending on the scheduling logic applied in both stages. This superiority stems from the complementary roles of the two stages, where the GA is responsible for generating a globally optimized master schedule, while the dispatching rule refines local sequencing decisions during execution. The proposed model consistently outperforms heuristic-based two-stage approaches across all performance indicators. Compared with FCFS, SPT, EDD, and LPT, the proposed model achieves shorter makespan, substantially lower total tardiness, and reduced flow time, indicating better coordination between long-term planning and short-term responsiveness.

Overall, the comparative results highlight clear trade-offs between fixed periodic scheduling and two-stage scheduling approaches. Fixed periodic scheduling offers simplicity and stable coordination within a predefined planning horizon; however, its inability to adapt during the execution period makes it highly vulnerable to uncertain job arrivals, leading to accumulated tardiness. Two-stage scheduling enhances responsiveness by allowing event-driven job insertion; however, its effectiveness depends significantly on the quality of the master schedule and the logic governing local rescheduling decisions. The proposed model effectively integrates the strengths of both approaches by combining a globally optimized master schedule with constrained reactive adjustments. This integration ensures both structural stability and adaptive flexibility, allowing the system to respond to dynamic disturbances without sacrificing overall performance. Consequently, the proposed model demonstrates superior robustness and consistently outperforms the benchmark scheduling strategies in dynamic flexible flow shop environments.

5. Conclusions and recommendations

This research proposed a two-stage scheduling method for solving the DFFSS problems. The model combines a

master schedule for long-term planning with a reactive schedule that helps the system react quickly to unexpected job arrivals. A GA was used to solve the problem, and its parameters were adjusted to improve performance. The simulation results from a case study in truck body manufacturing showed that the proposed model yields better results than traditional scheduling approaches. It helps reduce makespan, tardiness, and flow time, and can be helpful for factories that need to handle uncertain job arrivals and frequent changes in production.

Although the proposed model demonstrates good performance, several limitations should be considered. First, the current model does not account for setup times between jobs or machine breakdowns, which often occur in real factories. Second, the GA was used alone without combining with other techniques that might improve the search process. Finally, the experiments were tested on a limited number of jobs, so the model's ability to handle huge problems was not thoroughly examined. For future research, additional factors such as setup time, maintenance activities, and job priority can be incorporated to further refine the model's accuracy in simulating real-world production. For instance, setup times can be incorporated by considering sequence-dependent setup durations, allowing setup-related trade-offs to be explicitly captured in the scheduling model. Machine breakdowns can be addressed by modeling machine availability as an uncertain parameter using stochastic or scenario-based representations, with the two-stage framework enabling robust master scheduling and adaptive reactive adjustments. It is also interesting to try hybrid algorithms to improve the solution quality and speed. Finally, applying the model to real production data in different industries will help to test its flexibility and practical value. From a practical perspective, the proposed two-stage scheduling framework can be readily applied in manufacturing environments where orders arrive dynamically, and rescheduling costs are high. Practitioners may use the master schedule to stabilize medium-term production plans, while the reactive stage supports rapid decision-making when disruptions occur. This structure is particularly suitable for make-to-order under high demand uncertainty.

REFERENCES

[1] M. Geurtsen, J. B. H. C. Didden, J. Adan, Z. Atan, and I. Adan, "Production, maintenance and resource scheduling: A review," *European Journal of Operational Research*, vol. 305, no. 2, pp. 501–529, Mar. 2023, doi: 10.1016/j.ejor.2022.03.045..

[2] S. Chorghe, R. Kumar, M. S. Kulkarni, V. Pandhare, and B. K. Lad, "Smart scheduling for next generation manufacturing systems: a systematic literature review," *Journal of Intelligent Manufacturing*, 2024, doi: 10.1007/s10845-024-02484-2.

[3] Y. Zhang and L. Wang, "A Dynamic Scheduling Method for Logistics Supply Chain Based on Adaptive Ant Colony Algorithm," *International Journal of Computational Intelligence Systems*, vol. 17, no. 1, Dec. 2024, doi: 10.1007/s44196-024-00606-5.

[4] J. Jin, H. Huang, Y. Li, Y. Dong, G. Zhang, and J. Chen, "Variable speed limit control strategy for freeway tunnels based on a multi-objective deep reinforcement learning framework with safety perception," *Expert Systems With Applications*, vol. 267, Apr. 2025, doi: 10.1016/j.eswa.2024.126277.

[5] J. Zhao, Y. Long, B. Xie, G. Xu, and Y. Liu, "A matheuristic solution for efficient scheduling in dynamic truck–drone collaboration," *Expert Systems With Applications*, vol. 267, Apr. 2025, doi: 10.1016/j.eswa.2024.126218.

[6] J. Luo, S. Fujimura, D. El Baz, and B. Plazolles, "GPU based parallel genetic algorithm for solving an energy efficient dynamic flexible flow shop scheduling problem," *Journal of Parallel and Distributed Computing*, vol. 133, pp. 244–257, Nov. 2019, doi: 10.1016/j.jpdc.2018.07.022.

[7] M. Kim and S. Kwon, "Application of supervised and unsupervised learning for enhancing energy efficiency and thermal comfort in air conditioning scheduling under uncertain and dynamic environments," *Energy and Buildings*, vol. 325, Dec. 2024, doi: 10.1016/j.enbuild.2024.115028.

[8] D. Tang, M. Dai, M. A. Salido, and A. Giret, "Energy-efficient dynamic scheduling for a flexible flow shop using an improved particle swarm optimization," *Computers in Industry*, vol. 81, pp. 82–95, Sep. 2016, doi: 10.1016/j.compind.2015.10.001.

[9] D. Rooyani and F. Defersha, "A Two-Stage Multi-Objective Genetic Algorithm for a Flexible Job Shop Scheduling Problem with Lot Streaming," *Algorithms*, vol. 15, no. 7, Jul. 2022, doi: 10.3390/a15070246.

[10] L. Cai and Z. Chen, "Multi-objective optimization Genetic algorithm for flow-shop Scheduling module by using fuzzy-AHP," in *Proceedings - 2019 2nd World Conference on Mechanical Engineering and Intelligent Manufacturing, WCMEIM 2019*, Institute of Electrical and Electronics Engineers Inc., Nov. 2019, pp. 372–375. doi: 10.1109/WCMEIM48965.2019.00080.

[11] M. Wang and B. Xin, "A Genetic Algorithm for Solving Flexible Flow Shop Scheduling Problem with Autonomous Guided Vehicles," *2019 IEEE 15th International Conference on Control and Automation (ICCA)*, Edinburgh, UK, 2019, pp. 922-927, doi: 10.1109/ICCA.2019.8899914.

[12] J. Zhang and J. Cai, "A Dual-Population Genetic Algorithm with Q-Learning for Multi-Objective Distributed Hybrid Flow Shop Scheduling Problem," *Symmetry (Basel)*, vol. 15, no. 4, Apr. 2023, doi: 10.3390/sym15040836.

[13] M. Zandieh and N. Karimi, "An adaptive multi-population genetic algorithm to solve the multi-objective group scheduling problem in hybrid flexible flowshop with sequence-dependent setup times," *Journal of Intelligent Manufacturing*, vol. 22, no. 6, pp. 979–989, Dec. 2011, doi: 10.1007/s10845-009-0374-7.

[14] Z. Wang, W. Liao, and Y. Zhang, "Rescheduling optimisation of sustainable multi-objective fuzzy flexible job shop under uncertain environment," *International Journal of Production Research*, 2024, doi: 10.1080/00207543.2024.2354830.

[15] F. Ren and H. Liu, "Dynamic scheduling for flexible job shop based on MachineRank algorithm and reinforcement learning," *Scientific Reports*, vol. 14, no. 1, Dec. 2024, doi: 10.1038/s41598-024-79593-8.

[16] J. Lu, J. Zhang, J. Cao, X. Xu, Y. Shao, and Z. Cheng, "Flexible Job Shop Dynamic Scheduling and Fault Maintenance Personnel Cooperative Scheduling Optimization Based on the ACODDQN Algorithm," *Mathematics*, vol. 13, no. 6, Mar. 2025, doi: 10.3390/math13060932.

[17] Z. Wang, J. Zhang, and J. Si, "Dynamic Job Shop Scheduling Problem with New Job Arrivals: A Survey," in *Lecture Notes in Electrical Engineering*, Springer Verlag, 2020, pp. 664–671. doi: 10.1007/978-981-32-9050-1_75.

[18] H.-P. Nguyen, Q.-H. Ngo, and V.-C. Nguyen, "Multi-Agent communication for dynamic Job-Shop scheduling: a robust Single-Machine scheduling model with genetic algorithm optimization," *IEEE Access*, vol. 13, pp. 87183–87192, Jan. 2025, doi: 10.1109/access.2025.3569568.

[19] H. Wang, W. Lin, T. Peng, Q. Xiao, and R. Tang, "Multi-agent deep reinforcement learning-based approach for dynamic flexible assembly job shop scheduling with uncertain processing and transport times," *Expert Systems with Applications*, vol. 270, Apr. 2025, doi: 10.1016/j.eswa.2025.126441.