Article

Dynamic Scheduling in Identical Parallel-Machine Environments: A Multi-Purpose Intelligent Utility Approach

Mahmut İbrahim Ulucak [1,*] and Hadi Gökçen [2]

[1] Graduate School of Informatics, Gazi University, 06680 Ankara, Türkiye
[2] Department of Industrial Engineering, Faculty of Engineering and Architecture, Gazi University, 06570 Ankara, Türkiye; hgokcen@gazi.edu.tr
* Correspondence: iulucak@gmail.com

Abstract: This paper presents a robust and adaptable framework for predictive–reactive rescheduling in identical parallel-machine environments. The proposed Multi-Purpose Intelligent Utility (MIU) methodology utilizes heuristic methods to efficiently address the computational challenges associated with NP-hard scheduling problems. By incorporating 13 diverse dispatching rules, the MIU framework provides a flexible and adaptive approach to balancing critical production objectives. It effectively minimizes total weighted tardiness and the number of tardy jobs while maintaining key performance metrics like stability, robustness, and nervousness. In dynamic manufacturing environments, schedule congestion and unforeseen disruptions often lead to inefficiencies and delays. Unlike traditional event-driven approaches, MIU adopts a periodic rescheduling strategy, enabling proactive adaptation to evolving production conditions. Comprehensive rescheduling ensures system-wide adjustments to disruptions, such as stochastic changes in processing times and rework requirements, without sacrificing overall performance. Empirical evaluations show that MIU significantly outperforms conventional methods, reducing total weighted tardiness by 50% and the number of tardy jobs by 27% on average across various scenarios. Furthermore, this study introduces novel quantifications for nervousness, expanding the scope of stability and robustness evaluations in scheduling research. This work contributes to the ongoing discourse on scheduling methodologies by bridging theoretical research with practical industrial applications, particularly in high-stakes production settings. By addressing the trade-offs between improving the objective function or improving the rescheduling performance, MIU provides a comprehensive solution framework that enhances operational performance and adaptability in complex manufacturing environments.

Keywords: dispatching rules; identical parallel machines; initial scheduling; MIU; predictive–reactive rescheduling

Academic Editor: Hyokyung Bahn

Received: 28 December 2024
Revised: 22 February 2025
Accepted: 23 February 2025
Published: 25 February 2025

Citation: Ulucak, M.İ.; Gökçen, H. Dynamic Scheduling in Identical Parallel-Machine Environments: A Multi-Purpose Intelligent Utility Approach. *Appl. Sci.* **2025**, *15*, 2483. https://doi.org/10.3390/app15052483

1. Introduction

In the realm of manufacturing and production, the scheduling of tasks across identical parallel machines constitutes a fundamental aspect of operational efficiency and resource utilization. The dynamic nature of production environments, characterized by uncertainties, disruptions, and varying priorities, necessitates agile scheduling strategies capable of adapting to evolving conditions in real time. Pinedo [1] emphasizes the importance of such adaptive strategies, noting that effective scheduling must account for both the inherent uncertainties and the need for timely adjustments to maintain system efficiency. In this context, predictive–reactive rescheduling emerges as a crucial paradigm, offering the promise of enhanced adaptability and responsiveness to changing production dynamics.

Driven by the complexities of space sector production and dynamic scheduling challenges, this paper introduces a comprehensive framework for predictive–reactive rescheduling in identical parallel-machine environments. Our methodology is grounded in real-world data and scenarios, reflecting the intricacies of production dynamics and the influence of managerial decisions.

The data presented in this paper have been adapted from real-world scenarios to ensure confidentiality and protect company information. While the specific data cannot be shared for security reasons, they have been altered in a manner that preserves their relevance to practical industry settings, and the methodologies used ensure the applicability of the findings. All calculations and results are based on these adapted data sets, maintaining the integrity of the analysis.

In contrast to traditional optimization-based approaches, our methodology adopts a heuristic-driven strategy to overcome the computational complexities associated with NP-hard problems, ensuring timely and efficient solution generation.

A distinguishing feature of our approach lies in the integration of 13 distinct dispatching rules to cater to diverse production objectives and constraints. Our approach integrates 13 dispatching rules to address various production objectives and constraints. This broad selection of rules offers flexibility in scheduling solutions, allowing decision-makers to improve production outcomes while reducing computational overhead.

Crucially, our approach emphasizes periodic rescheduling over event-driven approaches, enabling proactive adaptation to changing production conditions and minimizing the impact of disruptions on schedule performance. Through complete rescheduling, we ensure comprehensive adjustments across schedules, mitigating the risk of cascading disruptions and optimizing resource utilization in dynamic manufacturing environments.

Central to our methodology is the consideration of stochastic processing time deviations and job rework occurrences, acknowledging the inherent uncertainties inherent in real-world production processes. By incorporating these factors into our rescheduling framework, we enhance the adaptability and robustness of our scheduling solutions, enabling an effective response to dynamic production conditions and uncertainties.

Drawing from the rich body of literature on predictive–reactive rescheduling, identical parallel-machine scheduling, and dispatching rules, our study contributes empirical insights and practical methodologies to inform scheduling practices in dynamic manufacturing environments. By synthesizing theoretical foundations with practical considerations, we aim to bridge the gap between research and industry practice, facilitating the adoption of predictive–reactive rescheduling strategies to enhance operational efficiency and competitiveness in modern manufacturing settings.

Instead of commonly utilized objective functions such as makespan or cost minimization [2–4], this study adopts total weighted tardiness minimization and tardy job number minimization as its primary goals. To the best of our knowledge, this study uniquely employs the simultaneous application of periodic and complete rescheduling while evaluating performance through innovative metrics for stability [5], robustness [6], and nervousness [7]. The comprehensive literature review, particularly the in-depth analysis of dispatching rules, is anticipated to serve as a valuable resource for researchers. Moreover, the unique problem structure of the production line addressed in this study is expected to provide significant insights and foster innovative perspectives within the field.

In scheduling identical parallel machines, disruptions, stochastic processing times, and job rework introduce significant inefficiencies. Existing approaches often rely on optimization-based methods that are computationally impractical for large-scale, dynamic environments. Additionally, periodic and complete rescheduling strategies are rarely combined, and total weighted tardiness (TWT) and number of tardy jobs (NTJ) are not commonly used as the primary performance objectives in predictive–reactive rescheduling studies. This paper aims to bridge these gaps by introducing a heuristic-driven predictive–reactive rescheduling framework that integrates 13 dispatching rules and evaluates performance using innovative stability, robustness, and nervousness metrics.

In the subsequent sections of this paper, we delineate the theoretical underpinnings of our predictive–reactive rescheduling framework, describe the implementation details of our methodology using Python code, and present empirical findings and case studies to demonstrate the efficacy and practical applicability of our approach. Through this comprehensive analysis, we aim to provide valuable insights and practical guidelines for the development and implementation of predictive–reactive rescheduling strategies in identical parallel-machine environments.

This paper is organized as follows. Section 2 provides a detailed literature review, emphasizing the novelty of the MIU approach. Section 3 defines the problem formally and presents the underlying assumptions. Section 4 explains the MIU approach, detailing its methodology and implementation. Section 5 presents computational experiments and results, comparing MIU with traditional methods. Finally, Section 6 concludes with findings and future research directions.

2. Literature Review

This section provides a comprehensive review of the literature and summarizes dynamic scheduling, parallel-machine scheduling, and rescheduling. The novelty of the Multi-Purpose Intelligent Utility (MIU) framework is emphasized through comparisons with previous studies, focusing on its ability to address real-world scheduling challenges.

2.1. Dynamic Scheduling in Production Systems

Scheduling in production systems ensures the coordination of activities to increase efficiency and reduce operational costs. In stochastic and dynamic production environments, traditional scheduling solutions based on classic objectives like makespan may not suffice. Additional criteria capable of addressing stochastic disruptions need to be considered due to the random disturbances that may occur in the system. Often, rescheduling is employed to enhance system performance and counteract the effects of random disruptions.

In many real-world environments, scheduling is an ongoing reactive process due to the presence of various unforeseen interruptions, making continuous the reevaluation of predetermined schedules necessary. Solutions developed to address static scheduling problems are often impractical in real-world environments, as schedules optimized based on predicted data may become invalid when deployed in the workshop. Most production systems operate in dynamic environments where unforeseeable real-time events can lead to changes in scheduled tasks, rendering a pre-implemented schedule unfeasible upon presentation to the workshop. Examples of such real-time events include machine breakdowns, the arrival of urgent tasks, changes in delivery deadlines, and more.

Dynamic scheduling strategies aim to adapt to changing conditions and uncertainties in manufacturing environments. Cowling and Johansson [8] emphasized the utilization of real-time information for effective dynamic scheduling, highlighting the importance of timely adjustments to changing conditions. Ouelhadj and Petrovic [9] provided a comprehensive survey of dynamic scheduling in manufacturing systems, outlining various approaches and challenges in dynamic scheduling environments. They laid the groundwork for understanding the complexities and requirements of dynamic scheduling systems.

As can be seen in Figure 1, dynamic scheduling, crucial for the successful implementation of scheduling systems in real-world scenarios, is categorized under three main types [7,10–13]: completely reactive scheduling, predictive–reactive scheduling, and robust proactive scheduling. The possible machine environments are also demonstrated in Figure 1 according to Pinedo's classification [1].

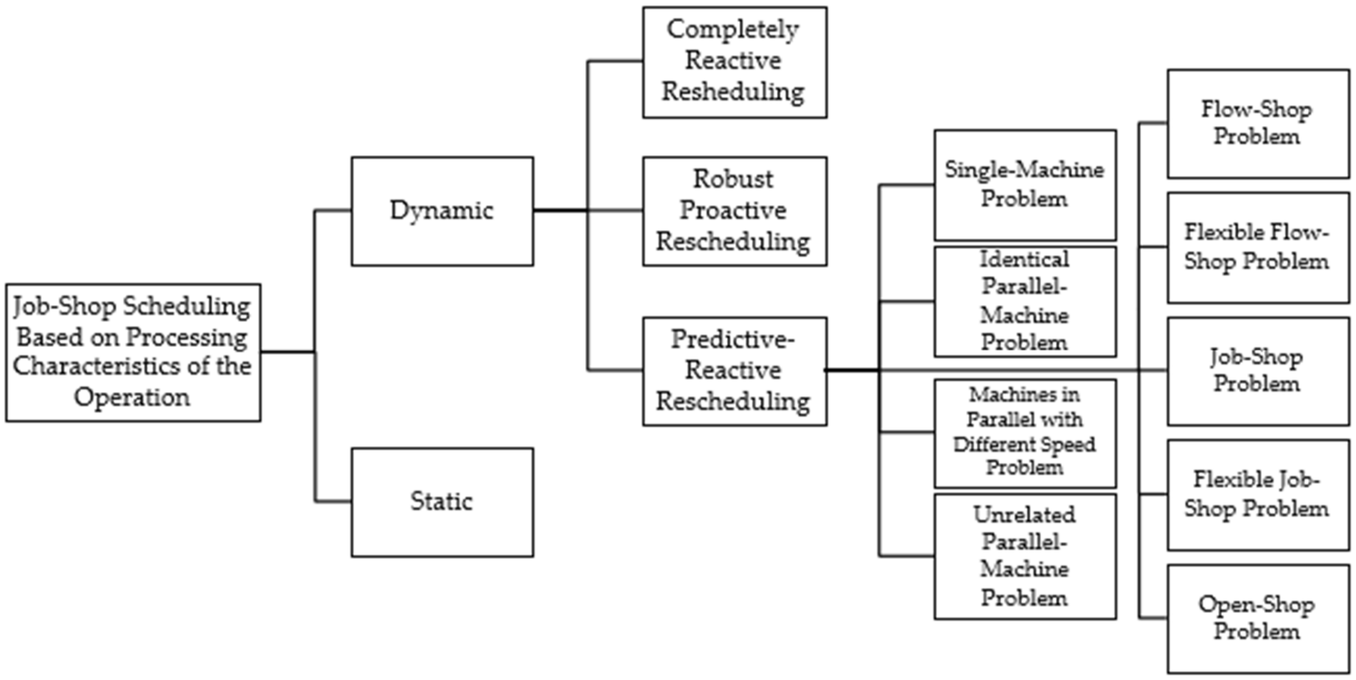
Job-Shop Scheduling Based on Processing Characteristics of the Operation
Dynamic
Static
Completely Reactive Resheduling
Robust Proactive Rescheduling
Predictive-Reactive Rescheduling
Single-Machine Problem
Identical Parallel-Machine Problem
Machines in Parallel with Different Speed Problem
Unrelated Parallel-Machine Problem
Flow-Shop Problem
Flexible Flow-Shop Problem
Job-Shop Problem
Flexible Job-Shop Problem
Open-Shop Problem

Figure 1. Dynamic scheduling problems in the literature.

2.1.1. Completely Reactive Rescheduling

In completely reactive scheduling, a predefined schedule is not created in advance, and decisions are made in real time locally. Dispatching rules are often utilized in this type of scheduling due to their ability to provide quick results and ease of implementation.

Notable studies on completely reactive scheduling include the following:

Ning Liu et al. [14] proposed a method that combines completely reactive scheduling with real-time decision-making, suggesting a more flexible and stable approach compared to traditional dispatching rules.

Kexin Li et al. [15] demonstrated that their method for workshop scheduling problems outperformed widely used completely reactive scheduling methods.

Bożek and Wysocki [16] presented a comprehensive case study on developing production planning solutions, including various scheduling modes and models, utilizing both offline planning and reactive scheduling.

2.1.2. Robust Proactive Rescheduling

Proactive and/or robust scheduling focuses on developing a foundational schedule that incorporates a degree of variability during project execution. The idea is to include resilience in the initial schedule.

Key studies in this area include the following:

Penz et al. [17] conducted research on sensitivity analysis for problems with maximum and total completion time objectives.

Yang and Yu [18] aimed to minimize the impact of disruptions on critical performance measures in their robust scheduling study.

Hall and Posner [19] provided a comprehensive discussion on sensitivity analysis in scheduling problems, offering connections to classical optimization and algorithms for rescheduling in response to online disruptions.

For a detailed literature review on Proactive Reactive Scheduling, the article by Rahmani and Heydari [5] can be consulted.

The difference between predictive–reactive approaches and proactive–reactive approaches is mainly due to the fact that in proactive–reactive approaches rescheduling is not conducted online; instead, one of several previously estimated schedule solutions is selected. These proactive–reactive analysis methods make it possible to create a set of static programs that facilitate the transition from one to another in case of risk.

2.1.3. Predictive–Reactive Rescheduling

Predictive–reactive scheduling, the most common dynamic scheduling approach in manufacturing systems, involves revising schedules in response to real-time events. Predictive–reactive scheduling approaches often emphasize simple adjustments to enhance workshop efficiency. While these methods are beneficial, more robust schedules can reduce the impact of disruptions and improve overall performance.

The literature on predictive–reactive scheduling has extensively considered various real-time events and their effects, covering different manufacturing systems such as single-machine systems, parallel-machine systems, flow shops, job shops, and flexible manufacturing systems.

Studies often evaluate predictive–reactive scheduling based on metrics like makespan, stability, and robustness, aiming to optimize system efficiency while maintaining stability and resilience in the face of disruptions [20–22].

Production System Types

Pinedo [1] defines system types as follows:

Single Machine: The simplest machine environment and a special case of more complex systems [13,18,22,23].

Identical Machines in Parallel: m identical machines process jobs, either on any machine or a specific subset [24–26].

Machines in Parallel with Different Speeds: m machines operate at different speeds, affecting processing times [1].

Unrelated Machines in Parallel: Each machine has a different speed for each job, leading to varying processing times [27].

Flow Shop: m machines in series, where all jobs follow the same sequence, typically under FIFO discipline [5,28].

Flexible Flow Shop: A generalized flow shop with multiple parallel machines at each stage [29].

Job Shop: Jobs follow distinct routes, possibly visiting the same machine multiple times [14,20,30,31].

Flexible Job Shop: A hybrid of job-shop and parallel machines, with multiple machines at each work center [15,29,32,33].

Open Shop: Jobs must be processed on all machines, but their processing sequence is flexible [1].

Mathematical Approaches

According to Zhang et al. [31], workshop scheduling algorithms can be examined under two main headings: Exact optimization methods include effective rule approaches, mathematical programming approaches, branch-and-bound methods, etc. Approximate methods include constructive methods, artificial intelligence, local search, and metaheuristic algorithms. Mathematical programming and operation research are applied to reach the global optimal solution or the deterministic optimal solution.

The mathematical programming approach is an exact optimization approach that is frequently used to solve predictive–reactive scheduling problems [32,34,35]. In addition to approximate methods such as the Simulated Annealing Algorithm [2], Hybrid Parallel Genetic Algorithm [27,29], Biased Randomized Iterated Greedy Algorithm [6], Tabu Search Algorithm [36,37], Directed Backjumping Algorithm, and Hybrid Multi-Objective Immune Algorithm [30], the most commonly used approximate method is the use of dispatching rules [24,38]. Constructive methods are preferred methods because they shorten the solution time.

Examples of dispatching rules that provide quick solutions include: Least Flexible Job First [24,39], longest processing time [38,40,41], Earliest Due Date [20,38,41], shortest processing time [38,40–42], First in First Out [40,41], random [40,41], and Last in First Out [41].

Uncertain and Stochastic Scenarios

Real-world manufacturing environments involve numerous states of uncertainty and stochasticity of events. Some situations are machine failure, change in delivery date, and uncertainty of the processing time of a job on a machine. An appropriate scheduling model should take into account all uncertainty conditions to approximate real-world problems [43].

Stating that stochastic scheduling has attracted great attention from both industry and academia, Xin Liu et al. [44] stated that existing jobs generally focus on random processing times, but the uncertainty in the release time when the job can start greatly affects the performance.

The first study in the field of stochastic scheduling problems was conducted by Cheng [45] in a single-machine environment. He developed his article in 1991, and in this research a common deadline was defined for all works. In the study, each job has a stochastic processing time with a certain mean and variance.

In predictive–reactive scheduling problems, entering the new job into the system [6,23,24,46,47] and machine failure [4,30,39,48] are the most common uncertain situations.

In addition to these two basic uncertainties, other uncertainties such as material shortage [37,49–51], changes in production time or uncertainty [40,41,52,53], job cancellations [28], and priority changes [38] have also been addressed in the literature.

Rescheduling Timing

Three policies regarding the timing of rescheduling have been proposed in the literature [7,54]: periodic, event-based, and hybrid.

Periodic policy is to create a schedule at regular intervals after collecting all the available information in the workshop. The dynamic scheduling problem is decomposed into a set of static problems that can be solved using classical scheduling algorithms. Scheduling is then conducted and existing information is not collected and revised until the next period begins.

Periodic policy provides greater schedule stability and less schedule tension. Following a set schedule in the face of significant changes in the shop can compromise performance as unwanted products or intermediate products may be produced. Determining the rescheduling period is also an important research topic.

In an event-driven policy, rescheduling is triggered in response to an unexpected event that changes the current system state. Most approaches to dynamic scheduling use this policy.

The hybrid policy reschedules the system periodically and also when an exception occurs. Events usually taken into account are machine failures, urgent jobs arriving, job cancellations, or job priority changes.

Periodic policy [34,35,40,41] and hybrid [37,46] policies have not been preferred much in the problems in the literature; usually Event-Oriented Policies [24,30,39,53] were applied when rescheduling.

Rescheduling Solution Methods

Regarding the question of which strategies to use for rescheduling, the literature offers two main rescheduling strategies [7,8,54]: schedule repair and complete rescheduling.

Repairing the schedule refers to editing parts of the existing schedule and may be preferable because it potentially saves CPU time and preserves the stability of the system.

Complete rescheduling creates a new schedule from scratch. Complete rescheduling may in principle be better at maintaining optimal solutions, but these solutions are rarely achievable in practice and require too much computation time. Moreover, complete rescheduling can cause instability and lack of continuity in detailed plant schedules, leading to additional production costs attributable to the so-called shop floor perturbation.

Schedule repair has been implemented in two ways in the literature [4,36,48]: Right Shift Algorithm and Partial Repair. The method that finds more space in the literature is the complete rescheduling [16,30,53,55] method.

Evaluation of Rescheduling

Rahmani and Heydari defined a new objective function called "MSR" based on Makespan (Efficiency), Stability, and Robustness. These components are explained as follows [5]:

Scheduling efficiency: This metric indicates the degree of optimization for a schedule. In Rahmani and Heydari's study, this criterion is measured by the classic target "makespan". It is stated that the actual completion times of the affected works may change due to disruptions in the system. In the studies examined, effectiveness was expressed in the same way as the objective function.

Robustness: This criterion is defined as the deviation in the performance of the actual schedule from the initial schedule. This measure indicates the closeness of the actual schedule's performance to the initial schedule. In fact, robustness is associated with the changes produced in the objective function following perturbations. If the performance of a schedule is not too poor and lacking when dealing with interruptions, it is called a robust schedule.

Stability: This criterion is the difference between the completion times of the jobs in the initial schedule and the actual schedule. When a disruption occurs in the system, the actual order may change, resulting in the cost of reallocating tools and equipment, the cost of re-ordering raw materials, etc. However, when the actual order is closer to the initial order, these costs are reduced and stability increases. Therefore, the stability criterion is about the difference between the initial schedule and the actual schedule, not about their performance. If a sequence does not change much compared to the first when faced with failures or disruptions, it is called a stable schedule.

The objective function or effectiveness was evaluated in six ways in the reviewed articles. These are cost [2,3,33,34], makespan [6,16,38,56], Total Weighted Delay and/or earliness [29,30,32,46], Total Weighted Waiting Time [26,32], Average Stability [41], Material/Machine Usage [51,57].

Stability has been examined in three basic ways. Deviation in the start time of each job [30,36,48,58], deviation in the Total Weighted Completion Time [55], and deviation in the completion time of each job [4,33,39]. These are the evaluation criteria that find their place in the literature.

Robustness did not find much space in the articles examined. There are studies that evaluate the completion time difference between the first and the last schedule [6,52], performance loss [30], and the combination of efficiency and stability [36] as robustness.

The concept of "nervousness" is mentioned as a definition in the literature [7] but as far as we know has not been expressed numerically. Scheduling nervousness was first defined as "significant changes in MRP (material requirements planning) plans" or "instability". Nervousness is the opposite of schedule stability because there is constant change in the schedule (frequent rescheduling). A "nervous system" offers little predictability. A rescheduling policy that provides fewer revisions increases schedule stability (and reduces schedule nervousness).

In this study, we considered the change in the start time of the jobs as the stability criterion. As a robustness criterion, unlike other studies in the literature, we examined the changes in delay times. Since it is certain that the work will be delayed in our problem, our performance criterion should be the completion of the work with the least delay difference. We calculated robustness, robustness2, and robustness3 as the largest difference in the change in the amount of delay, the difference in the total amount of delay, and the difference in the total weighted amount of delay. As for nervousness, although it is mentioned verbally in the literature, as far as we know, there is no numerical example. One of the things that will increase the tension the most in a production line is production on a different machine than the planned machine. If the machine job match is changed by rescheduling while all preparations have been made according to plan, the tension of the production line increases. These preparations can be exemplified as meetings of designers, technicians, and other stakeholders; machine preparations; and consumable planning. In other words, the tension would increase if a different machine than the one determined during the initial scheduling conducted the planned work. For this reason, the differences between the initially planned and the actual machine-job assignments are referred to as scheduling nervousness.

In our article, the predictive reactive scheduling problem on parallel machines will be discussed. The similarities and differences compared to similar studies in the literature are shown in Table 1.

As can be seen in Table 1, the problem we examined in our study differs from other studies in terms of objective function, random issues, reactive scheduling timing, and stability. Additionally, as far as we know, the first calculations regarding robustness on parallel machines and nervousness in whole job shops will be made in this paper.

Table 1. The similarities and differences between the literature and proposed approach.

Paper	Objective Function & Efficiency					Resolution Method	Stochastic Issues	When Rescheduling		How Rescheduling		Stability	Robustness	Nervousness
	CM	MM	TWWTM	TWTM	TJNM			Event-driven	Periodical	Complete Rescheduling	Partial Repair			
Tighazoui et al. [55]			✓			LMM	ANJ	✓		✓		TWCTD	ND	ND
Tighazoui et al. [26]			✓			MIP	ANJ	✓		✓		TWCTD	ND	ND
Turkcan et al. [4]	✓	✓				LPR & TIM	MD	✓		✓	✓	DCEJ	ND	ND
Li et al. [3]	✓	✓				MIP	ANJ, MD	✓		✓		ND	ND	ND
Ghaleb and Taghipour [2]	✓	✓				SAA	MD	✓		✓		DCEJ	ND	ND
Duenas and Petrovic [24]		✓				DR	ANJ, MD	✓		✓	✓	ND	ND	ND
Petrovic and Duenas [39]		✓				DR	MD	✓		✓	✓	DCEJ	ND	ND
Proposed Approach				✓	✓	DR	PTC, RN		✓	✓		DSTEJ	MTD, TWTD, TTD	NMC

CM: Cost Minimization, MM: Makespan Minimization, TWWTM: Total Weighted Waiting Times Minimization, TWTM: Total Weighted Tardiness Minimization, TJNM: Tardy Job Number Minimization, LMM: Linear Mathematical Model, MIP: Mixed Integer Programming, LPR: LP-Relaxation Based Two-Stage Algortihm, TIM:Time Indexed LpModel, SAA: Simulated Annealing Algorithm, DR: Dispatching Rules, ANJ: Arrivals of New Jobs, MD: Manufacturer Disruptions, PTC: Production Time Changes, RN: Rework Needs, TWCTD: Total Weighted Completion Time Deviation, DCEJ: Deviation of Completion time of each job, DSTEJ: Deviation of Starting Time Of Each Job, MTD: Maximum Tardiness Deviation, TWTD: Total Weighted Tardiness Deviation, TTD: Total Tardiness Deviation, ND: Not Discussed, NMC: Number of Machine Changes.

3. Problem Definition and Solution Method

This section provides a comprehensive explanation of the system under examination, defines the core problem, and presents the methodology to be employed in addressing the identified challenges. The methodology is explained in detail, highlighting the specific approach to be followed in solving the problem and ensuring an effective resolution of the production scheduling issues.

3.1. Current Status of the System

Our problem addresses an issue frequently encountered by a company operating in the space sector within its production line. Production in the space sector is carried out in "clean rooms" with highly sensitive equipment and qualified personnel. Projects in this sector are consistently impacted by schedule congestion, and any malfunction or production flaw leads to substantial delays in the overall timeline.

Figure 2 defines the production process, showing the sequential workflow from order initiation to final approval. The process begins when orders are received from designers, followed by scheduling to allocate resources and define timelines. The technicians conduct production activities as per the work sequence. Once production is completed, the output moves to the quality and functionality control phase. If defects are found, the order is rescheduled for rework, ensuring corrections before final approval. If the product meets quality standards, the order is closed, marking the successful completion of the process.

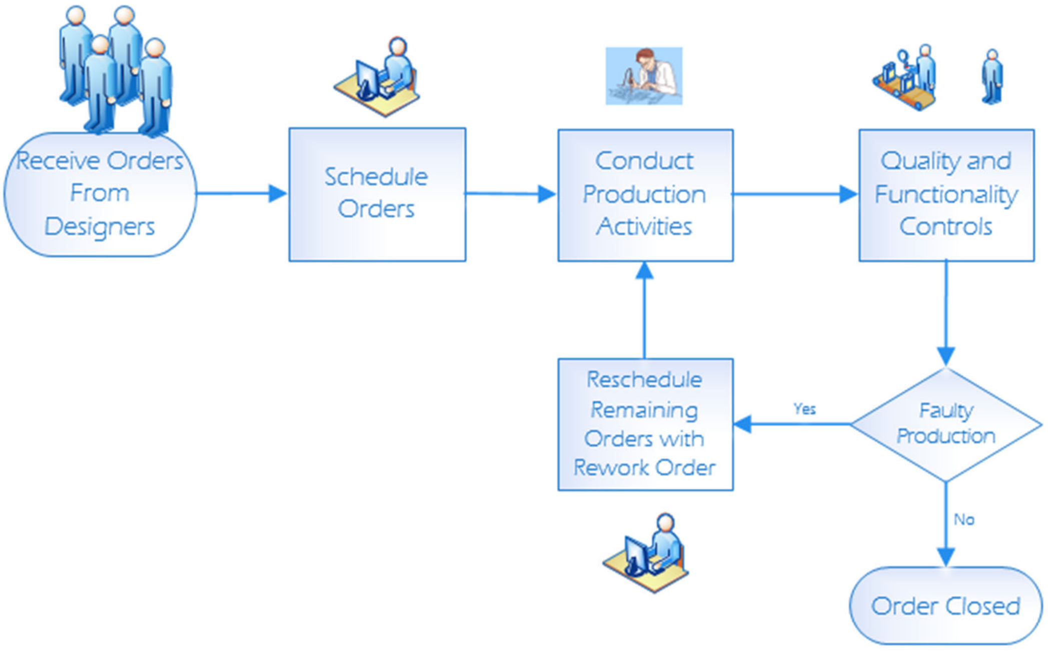
Receive Orders From Designers
Schedule Orders
Conduct Production Activities
Quality and Functionality Controls
Faulty Production
Yes
No
Reschedule Remaining Orders with Rework Order
Order Closed

Figure 2. Production process of company.

Figure 3 represents the information flow within the production system, detailing how various roles exchange data to ensure a well-coordinated manufacturing process. The Production Planner serves as the central node, receiving information from management and designers while coordinating work with technicians and the Quality Manager and Designer.

Management provides the Order Strategy (total weighted tardiness—TWT, number of tardy jobs—NTJ) and project schedule to the Production Planner for execution. Once production is underway, the Production Planner reports back to management with estimated completion times and performance metrics, allowing for oversight and performance evaluation. Designers supply the Bill of Materials (BOM), specifying the type and quantity of required components, which is directed to the Production Planner to facilitate resource planning. The Production Planner, after processing inputs from management and designers, generates the work sequence for each machine and forwards it to the technicians, who carry out the production activities.

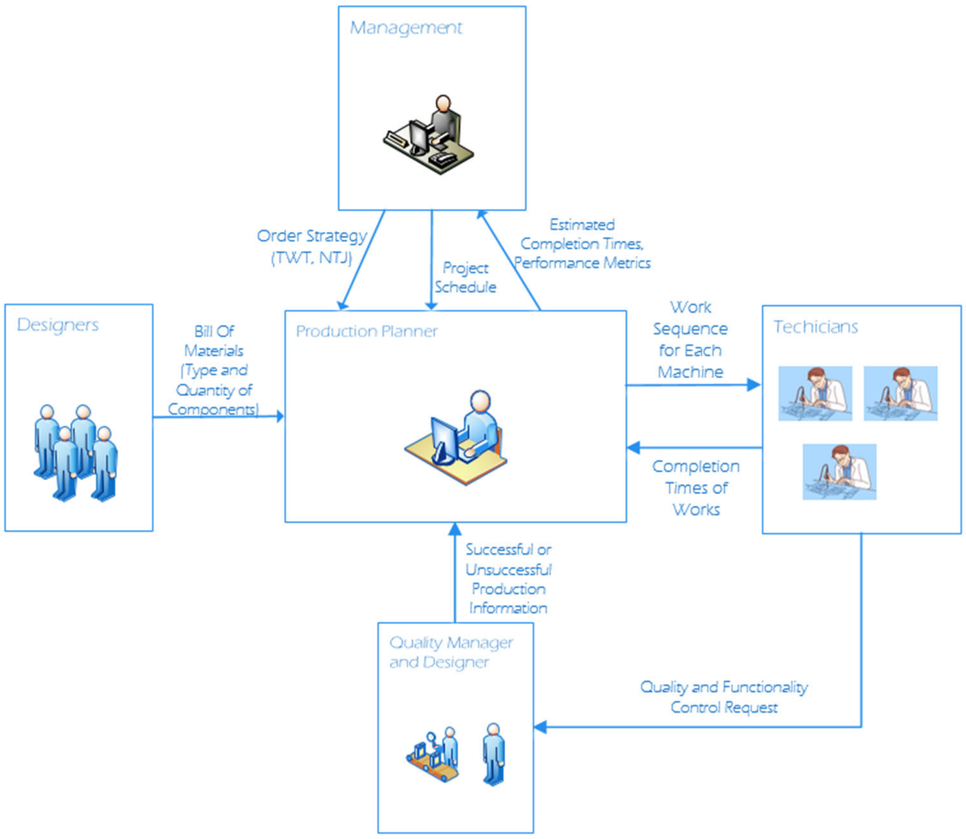
Management
Order Strategy (TWT, NTJ)
Project Schedule
Estimated Completion Times, Performance Metrics
Designers
Bill Of Materials (Type and Quantity of Components)
Production Planner
Work Sequence for Each Machine
Techicians
Completion Times of Works
Successful or Unsuccessful Production Information
Quality Manager and Designer
Quality and Functionality Control Request

Figure 3. Information flow of company.

Once production tasks are completed, technicians report the completion time of the works to the Production Planner, signaling that the manufacturing phase is finished. The Quality Manager and Designer then take over by conducting quality and functionality controls based on the completed work. They determine whether production is successful or unsuccessful and provide this information to the Production Planner for rescheduling or finalizing the order.

Although the production is precise, in practice no work is perfect or error-free. Mistakes are inevitable in every job where people are involved. For this reason, after the production of the produced electronic boards (Printed Circuit Board–PCB) is completed, they are first subjected to quality control and then to a series of functional tests by the designers. When a problem is encountered in both quality control and functional tests, the PCB returns to the production line and the production plan is shifted. While making this change, no optimization or heuristic method is used, and priority is given to the returned card.

PCBs are the fundamental components in production. The parameters taken into consideration are the release date and production time. The release date represents the earliest time components can be assorted, depending on supply availability and design readiness. The production time varies based on the type and quantity of the components on the PCB.

The release date is considered deterministic. The production time varies depending on the number of components, and the production time/number of components ratio has a distribution function that can be obtained based on historical data. The production time is considered stochastic.

No other work can be conducted on the same machine before the production of a card is finished. Technicians (machinery) in production have equal skills and conduct the same job in the same time. It is known that the hand tools used by technicians do not make a distinctive difference to the process. This shows that technicians can be considered as identical parallel machines. Each machine (technician) can only do one job at a time.

A faulty manufactured card discovered during quality control or functional testing is returned to the production line and must be reprocessed. This process is called "Rework". Errors that occur during production due to technician error, material, or equipment problems are noticed during quality control or functional tests after production, and this time increases in direct proportion to the number of components. This information is important as it directly affects the date of return of the card to the system. The release time parameter of the job to be reworked is the sum of the end date of the main job and the quality and testing time of that job. This period can be considered as two units for the number of components between 50 and 100, three units for the number of components between 101 and 150, and four units for the number of components between 151 and 200. The production time of the Rework Card also varies depending on the number of components to be reprocessed. Reprocessing has a distribution function whose production time/number of components ratio can be obtained from historical data.

Table 2 (part a) shows the number of components the cards have, the estimated production time based on historical data, and the release date of each card. Table 2 (part b) shows the actual production time of each production and the parameters of the productions that need rework.

Table 2. (**a**) Parameters in the beginning; (**b**) realized parameters.

(a)				(b)					
Parameters				**Realized Parameters**					
PCB No	**Component Amount**	**Release Date**	**Expected Processing Time**	**PCB No**	**Realized Processing Time**	**Rework Need?**	**Rework Need Component Amount**	**Expected Rework Time**	**Realized Rework Time**
P01	93	0	9	P01	8				
P02	87	30	8	P02	9				
P03	64	40	6	P03	5	Yes	8	3	4
P04	116	60	11	P04	12				
P05	68	0	6	P05	6				
P06	121	60	11	P06	12				
P07	168	0	15	P07	15				
P08	194	20	18	P08	19				
P09	103	0	9	P09	10	Yes	15	5	5
P10	54	30	5	P10	5				
P11	102	60	9	P11	9				
P12	82	10	8	P12	8	Yes	18	6	6
P13	157	30	14	P13	12				
P14	124	20	11	P14	11				
P15	120	30	11	P15	12				
P16	109	60	10	P16	11	Yes	18	6	7
P17	128	20	12	P17	11				
P18	165	90	15	P18	17				

Table 2. *Cont.*

(a)				(b)					
Parameters				Realized Parameters					
PCB No	Component Amount	Release Date	Expected Processing Time	PCB No	Realized Processing Time	Rework Need?	Rework Need Component Amount	Expected Rework Time	Realized Rework Time
P19	154	20	14	P19	13				
P20	89	30	8	P20	9				
P21	173	10	16	P21	14				
P22	96	0	9	P22	10	Yes	25	7	6
P23	190	60	17	P23	15				
P24	176	20	16	P24	18				

As can be seen in Table 2 (parts a,b), the sometimes realized processing times differ the from expected processing times, and reworks also make important disturbance for the system as a completely new job.

Initial scheduling is conducted before production starts. The main purpose of scheduling is to maximize capacity utilization and minimize makespan. For this purpose, the longest processing time first (LPT) rule is used, which is among the most frequently used distribution rules for makespan minimization in the literature [1,59]. In cases where rescheduling is required, fluctuations in production times are not taken into account, and in cases where rework is required, priority is given to reworks, so the Right Shift Algorithm is applied.

3.2. Problem Definition

As mentioned in Section 3.1., when a PCB returns to the production line priority is given to the returned card. This can be mathematically defined as increasing the weight of that card. This is a valid approach; however, the priority of cards already in the system may sometimes take precedence. This requires a comprehensive reassessment of all tasks and the creation of a revised schedule.

In today's environment, the growing intensity of project schedules and customer pressure necessitate considering weight and due date parameters when prioritizing PCBs for production. Weight should reflect the criticality of the card, while due date should be determined by decision-makers based on project timelines.

When project schedules are less constrained, and delays or shifting priorities are not critical, minimizing makespan can serve as a sufficient objective function. However, in cases where project delays are inevitable, it becomes crucial to determine which tasks—and how many—will be late. In such scenarios, objective functions such as total weighted tardiness (TWT) minimization or number of tardy jobs (NTJ) minimization should be prioritized.

Minimizing makespan does not account for job weight or due dates. Similarly, the longest processing time (LPT) algorithm disregards these parameters, often leading to suboptimal results when optimizing for the TWT or NTJ. Therefore, more sophisticated heuristics should be employed to effectively address these objectives.

When performing rescheduling, the Right Shift Algorithm increases stability and reduces nervousness, but it is more useful to perform complete rescheduling and reconsider the entire system in order to improve the objective function. However, complete rescheduling causes less stability and higher nervousness. At the same time, performing rescheduling in every disruption will cause calculation and implementation difficulties. The decision-maker needs to find a balance between improving the objective function or improving the rescheduling performance (stability, robustness, nervousness). In order to achieve this balance, rescheduling should be performed at certain periods, not complete rescheduling in every disruption.

3.3. Choosing a Reactive Scheduling Approach

Our problem requires basic scheduling at the beginning of the production schedule and rescheduling as a result of disruptions caused by production errors or the incorrect estimation of production time.

In our article, periodic and complete rescheduling methods will be studied using the scheduling approach known as predictive–reactive scheduling in the literature. The predictive reactive scheduling approach was preferred because it responds more flexibly to real-time events and allows strategy transitions.

The completely reactive scheduling approach is not suitable for our problem because the main purpose of our thesis is to provide the manager with a production plan and estimated performance evaluations at time t_0. In other words, it should include a result prediction before production begins.

Again, robust proactive scheduling is also not suitable for our problem. The difference between predictive–reactive approaches and robust–proactive approaches is mainly due to the fact that in robust–proactive approaches rescheduling is not conducted online; instead, one of several previously estimated schedule solutions is selected. The fact that there are too many variables and random situations in our problem makes it impossible to make robust scheduling for every possibility in advance.

3.4. Mathematical Model

The following mathematical model is suggested to find the solution the company is looking for in periodic meetings. Here, two separate objective functions are given at once. The company should look for the objective function solution it wants according to the strategy it has determined.

Data sets and indices are as follows:

N: Work Set (1, 2, n);
K: Row Set (1, 2, n);
M: Machine Set (1, 2, m);
J: work index j = 1, 2, n;
K: row index k = 1, 2, n;
I: machine index i = 1, 2, m.

Parameters are as follows:

W_j = weight of job j;
R_j = release date of job j;
P_j = processing time of job j;
D_j = due date of job j.

Decision variables are as follows:

X_{jki} = {1: if job j is assigned to row k of machine i; 0: o.w.};
CT_{ki}: completion time of job in row k on machine i;
ST_{ki}: starting time of job in row k on machine i;
T_j = max ($CT_j - d_j$; 0);

$$U = \sum_{k=0}^{n} \delta(T_J) \;\; \delta(x) = \begin{cases} 1, & x > 0; \\ 0, & o.w \end{cases}$$

Objective Functions are as follows:

Min $\sum_{j=1}^{n} w_J T_J$: minimizing total weighted tardiness;

Min U: minimizing the number of tardy jobs.

Constraints are as follows:

$$\sum_{i=1}^{m}\sum_{k=1}^{n} x_{Jki} = 1 \qquad \forall j \in N \tag{1}$$

$$\sum_{k=1}^{n} x_{Jki} \leq 1 \qquad \forall k \in K \quad \forall i \in M \tag{2}$$

$$ST_{ki} \geq \sum_{j=1}^{n} r_J x_{Jki} \qquad \forall k \in K \quad \forall i \in M \tag{3}$$

$$ST_{(k+1)i} \geq ST_{ki} + \sum_{j=1}^{n} p_J x_{Jki} \qquad \forall k = 1, \ldots n-1 \quad \forall i \in M \tag{4}$$

$$CT_{(k+1)i} \geq ST_{ki} + \sum_{j=1}^{n} p_J x_{Jki} \; \forall k \in K \quad \forall i \in M \tag{5}$$

$$x_{jki} = \{0,1\} \qquad \forall k \in K \quad \forall i \in M \; \forall j \in N \tag{6}$$

$$ST_{ki},\ CT_{ki} \geq 0 \; \forall k \in K \quad \forall i \in M \tag{7}$$

Assignment Constraint (1): Each job must be assigned to exactly one position in the schedule. This ensures that all jobs are scheduled without duplication or omission.

Uniqueness Constraint (2): Each position can be occupied by at most one job. This prevents multiple jobs from being assigned to the same position.

Release Time Constraint (3): A job cannot start before its predefined release time. This ensures that production does not begin prematurely.

Sequencing Constraint (4): Ensures that when a job is scheduled in a sequence, the next job can only start after the completion of the previous one. This maintains the order of execution.

Completion Time Constraint (5): Defines the completion time of a job as the sum of its start time and processing duration. This ensures that the model correctly accounts for processing times.

Variable Definitions (6): x_{jki} is a binary variable ensuring that job assignments follow the given constraints.

Non-Negativity Constraints (7): ST_{ki} and CT_{ki} are non-negative, ensuring a feasible scheduling timeline.

3.5. Proposed Methodology

The problem has two stages. The first stage is to predict, and the second stage is to decide what will happen after the failure. In the literature, there are studies that use the same algorithm for both parts, as well as studies that apply different solution methods for both stages.

In the first stage, an initial schedule is created, and in the second stage, rescheduling activities are carried out after the failure. A schedule will be created for the first stage, and periodic updates to the schedule will be allowed.

The mathematical model created can be used at the beginning of production, without taking into account any fluctuations in processing times or rework, by using only the expected processing times values when creating the initial scheduling. Even in this case, our problem is NP-hard.

The minimizing total weighted tardiness (TWT) problem involves scheduling a set of jobs, each with a given processing time, weight, and due date, such that the total weighted tardiness is minimized. The tardiness of each job is defined as the difference between its completion time and its due date, if the job is completed late. The Job-Shop Scheduling Problem (JSSP), the well-known NP-hard problem of scheduling, can be reduced to TWT. In the JSSP, a set of jobs must be scheduled on a set of machines, with each job having a predefined sequence of operations. The objective in the JSSP is to minimize the makespan, the total time required to complete all jobs. To reduce the JSSP to TWT, one conducts the following:

- Each operation in the JSSP corresponds to a job in TWT.
- The processing time of each operation in the JSSP is mapped directly to the processing time of the corresponding job in TWT.
- The weight of all jobs in TWT is set to one (i.e., no job has a higher priority than another).
- The due date for each job in TWT is set to the completion time of the corresponding operation in the JSSP.

By this reduction, solving the TWT problem minimizes the total weighted tardiness, and the tardiness of a job in TWT corresponds to the delay in completing a job beyond its due date, which is determined by the JSSP instance. Thus, an optimal solution to TWT also provides an optimal schedule for the JSSP, proving that TWT is NP-hard.

Similarly, the minimizing number of tardy jobs (NTJ) problem involves scheduling jobs so that the number of tardy jobs is minimized, which can also be reduced from the JSSP:

- Each operation in the JSSP corresponds to a job in the NTJ.
- The processing time of each operation in the JSSP is mapped to the corresponding job's processing time in the NTJ.
- The due date for each job in the NTJ is set to the completion time of the corresponding operation in the JSSP.

The goal of the NTJ is to minimize the number of tardy jobs, which is equivalent to minimizing the number of jobs whose completion time exceeds their due date. This objective directly relates to minimizing the number of operations that are completed after their due dates in the JSSP instance. Therefore, an optimal solution to the NTJ will correspond to an optimal solution for the JSSP, proving that the NTJ is NP-hard.

When actual production times deviate from expected values or rework operations become necessary, the continuous optimization model must be re-executed using updated data. This adjustment is essential, as machine availability changes dynamically with each completed job, requiring real-time recalibration of the schedule. This makes the structure of the problem even more difficult. The fact that the structure of the problem is NP-hard and the amount of disturbance is very high has caused the need for a new methodology that is practical and gives good results.

The methodology we developed is called Multi-Purpose Intelligent Utility (MIU). This methodology provides fast and good results to the targeted strategy based on a large number of distribution rules. Each dispatching rule has its own advantages and disadvantages. We calculate all of these rules at once and present the one that gives the best results to the decision-maker. Dispatching rules are frequently used in the literature because they have low computational complexity, are easily applicable and adaptable, and can provide good answers to high-input problems [1,24,25,39,40].

The advantages and disadvantages of the dispatching rules to be used in MIU are given in Table 3. These rules, along with others, have been extensively discussed in the following studies: Panwalkar and Iskander (1977) conducted a comprehensive survey of scheduling rules [60]. Rajendran and Holthaus (1999) provided a comparative analysis of dispatching rules in dynamic flow shops and job shops [61]. Chiang and Fu (2007) examined the use of dispatching rules for job-shop scheduling with due-date-based objectives [62].

Table 3. 13 Dispatching rules used in MIU.

Dispatching Rule	Definition	Advantages	Disadvantages
Apparent Tardiness Cost (ATC)	Prioritizes jobs based on a calculated cost of apparent tardiness, which considers the difference between the job's current completion time and its due date.	Helps in minimizing overall tardiness and meeting due dates.	Requires accurate estimation of apparent tardiness costs and may not consider other factors such as job priorities or processing times.
Earliest Due Date (EDD)	Prioritizes jobs based on their due dates, with jobs having earlier due dates processed first.	Effective for minimizing the maximum lateness of jobs.	May lead to suboptimal utilization of machines if jobs with later due dates could be processed earlier.
Shortest Processing Time (SPT)	Prioritizes jobs based on the shortest processing time, with shorter jobs processed first.	Reduces the average completion time of jobs.	Can lead to starvation of longer jobs, especially if there are many short jobs.
Longest Processing Time (LPT)	Prioritizes jobs based on the longest processing time, with longer jobs processed first.	Useful in balancing the load across machines when job lengths vary significantly.	May increase the average completion time of jobs and can delay short jobs.
Weighted Shortest Processing Time (WSPT)	Prioritizes jobs based on their weighted processing time, calculated as processing time divided by job weight.	Minimizes the weighted sum of completion times.	Requires accurate weights for each job and can be computationally intensive.
First Slack Time Last Processed (FSTLP)	Prioritizes jobs based on their slack time, calculated as due date minus (current time plus processing time).	Reduces the risk of job lateness by prioritizing jobs close to their due dates.	May deprioritize already late jobs, making them even later.
Minimum Slack (MS)	Prioritizes jobs based on their slack time, calculated as the maximum of zero or due date minus (current time plus processing time).	Ensures that late jobs are still given priority, reducing maximum lateness.	Similar to FSTLP but may not always optimize other performance metrics.
Least Flexible Job First (LFJF)	Prioritizes jobs based on their flexibility, calculated as due date minus release date.	Effective in environments where job due dates are critical.	Jobs with more flexibility may be unnecessarily delayed.
First in, First out (FIFO)	Processes jobs in the order they arrive, based on release date.	Simple and fair, easy to implement.	May not optimize any specific performance metric and can lead to inefficiencies.
Last in, First out (LIFO)	Processes the most recently arrived job first.	Can be useful in specific scenarios where recent jobs are more urgent.	Rarely used due to potential for high lateness and job starvation.
Random Scheduling	Jobs are processed in a random order.	Simple to implement, useful for testing.	Inefficient and may lead to suboptimal scheduling performance.

Table 3. *Cont.*

Dispatching Rule	Definition	Advantages	Disadvantages
Weighted Shortest Processing Time with Due Date (WSPTDD)	Prioritizes jobs based on a combination of processing time, weight, and due date, calculated as processing time multiplied by (due date/weight).	Balances due dates and processing times, optimizing a weighted objective.	Requires accurate weights and due dates for each job, computationally intensive.
Critical Ratio (CR)	Prioritizes jobs based on the critical ratio, calculated as time until due date divided by processing time.	Helps in managing jobs close to their due dates and balances processing times and due dates.	May not prioritize jobs effectively if processing times and due dates vary widely.

Each dispatching rule has its own trade-offs and is suited to different production environments and objectives. Choosing the right rule depends on factors such as job characteristics, machine capabilities, and production goals. Using multiple dispatching rules at the same time will cover each other's disadvantages and ensure that the most advantageous one is selected.

In the literature, various resolution methods have been employed to address predictive–reactive rescheduling problems in parallel-machine environments, including Linear Mathematical Models [55], Mixed Integer Programming (MIP) [26], LP-Relaxation-Based Two-Stage Algorithms, the Time-Indexed LP Model [4], and Simulated Annealing Algorithms [2]. While these approaches provide structured optimization techniques, they also present significant limitations, particularly in large-scale and dynamic production environments.

Mathematical programming models, such as MIP and the Time-Indexed LP Model, can theoretically provide optimal solutions but often become impractical for real-world applications due to their excessive computational requirements. As problem size increases, the solution time grows exponentially, making them unsuitable for large-scale industrial scheduling. The LP-Relaxation-Based Two-Stage Algorithm attempts to mitigate this issue by simplifying the problem space, but this comes at the cost of deviating from true optimality. Simulated Annealing and other heuristic-based approaches, while effective in certain conditions, may suffer from inconsistent performance and require extensive parameter tuning to achieve satisfactory results.

In contrast, our approach employs Python-based dispatching rules, which offer significant advantages in terms of computational efficiency, adaptability, and scalability. Unlike complex mathematical models that demand high processing power, dispatching rules provide fast, heuristic-based scheduling decisions. Python's (version 3.11.5) robust ecosystem, including libraries such as NumPy, Pandas, and SciPy, allows for efficient data handling and rapid simulations, making it particularly well suited for large-scale scheduling problems.

One of the key strengths of our method lies in its computational speed. Dispatching rules operate with minimal processing overhead, enabling real-time decision-making in production environments where quick responsiveness is crucial. Additionally, our approach implements 13 different dispatching rules, ensuring adaptability to various production scenarios without requiring extensive modifications. Moreover, Python's modular structure allows for seamless integration with existing production planning systems, facilitating a smooth transition from theoretical models to practical implementation.

Despite these advantages, our approach is not without limitations. This study does not use the model to calculate LB-UB (lower and upper bounds), which could present an opportunity for integration with researchers working on exact models. Despite the advantages of using dispatching rules, the absence of LB-UB calculations means that global optimality cannot be guaranteed. Furthermore, handling complex constraints—such as machine-specific priorities, maintenance schedules, and intricate sequencing requirements—is more straightforward in mathematical programming approaches than in dispatching-rule-based heuristics.

In conclusion, our Python-based dispatching rule approach offers significant advantages in terms of speed, adaptability, and real-world applicability while avoiding the computational burdens associated with traditional optimization models. However, to further enhance its effectiveness, a hybrid approach that integrates heuristics with advanced optimization techniques could be explored to address its limitations, particularly in terms of optimality and complex constraint handling.

4. MIU Methodology and Its Computations

In this section, the proposed MIU approach is introduced as a heuristic framework to address the scheduling problem. The methodology combines multiple dispatching rules to improve scheduling efficiency and adaptability in dynamic environments. The implementation details, including rule selection and their integration into the scheduling algorithm, are explained in depth.

Computations were made by applying the steps in the flow chart given in Figure 4.

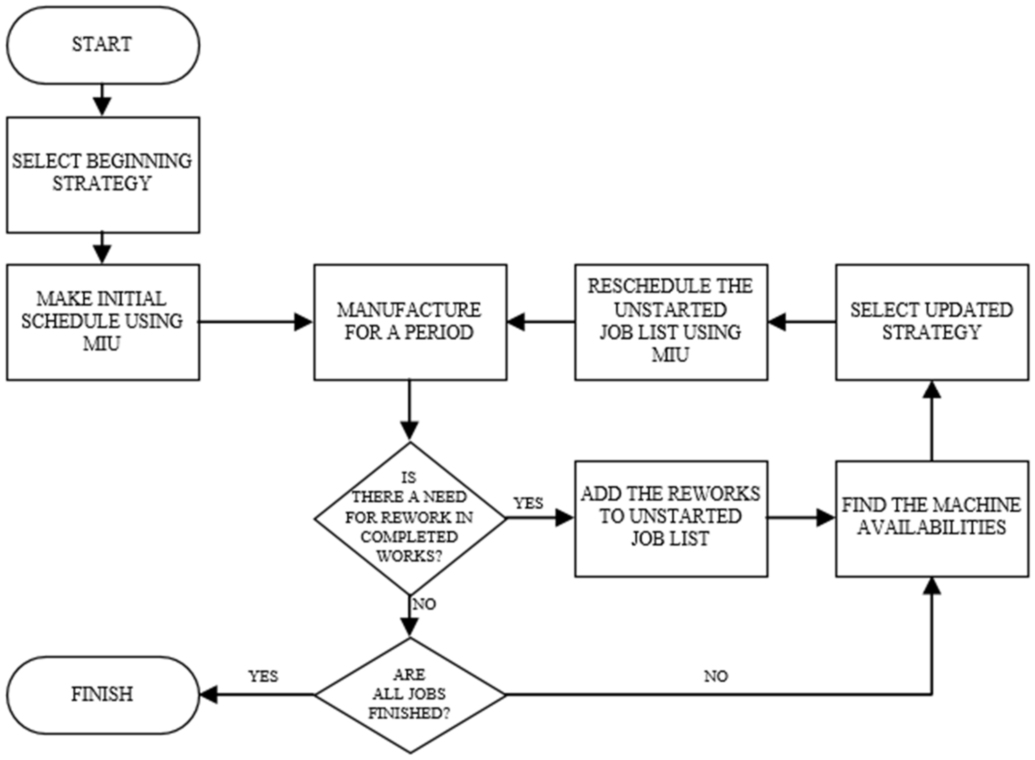
START
SELECT BEGINNING STRATEGY
MAKE INITIAL SCHEDULE USING MIU
MANUFACTURE FOR A PERIOD
RESCHEDULE THE UNSTARTED JOB LIST USING MIU
SELECT UPDATED STRATEGY
IS THERE A NEED FOR REWORK IN COMPLETED WORKS?
YES
ADD THE REWORKS TO UNSTARTED JOB LIST
FIND THE MACHINE AVAILABILITIES
NO
FINISH
YES
ARE ALL JOBS FINISHED?
NO

Figure 4. Predictive–reactive rescheduling using MIU.

The flowchart represents a decision-making framework for scheduling and rescheduling processes. It begins with selecting an initial strategy by management, followed by creating an initial schedule using MIU (details given in Figure 5). The manufacturing process runs for a set period, after which the system evaluates whether rework is required for completed jobs. If rework is needed, the jobs are added to the unstarted job list.

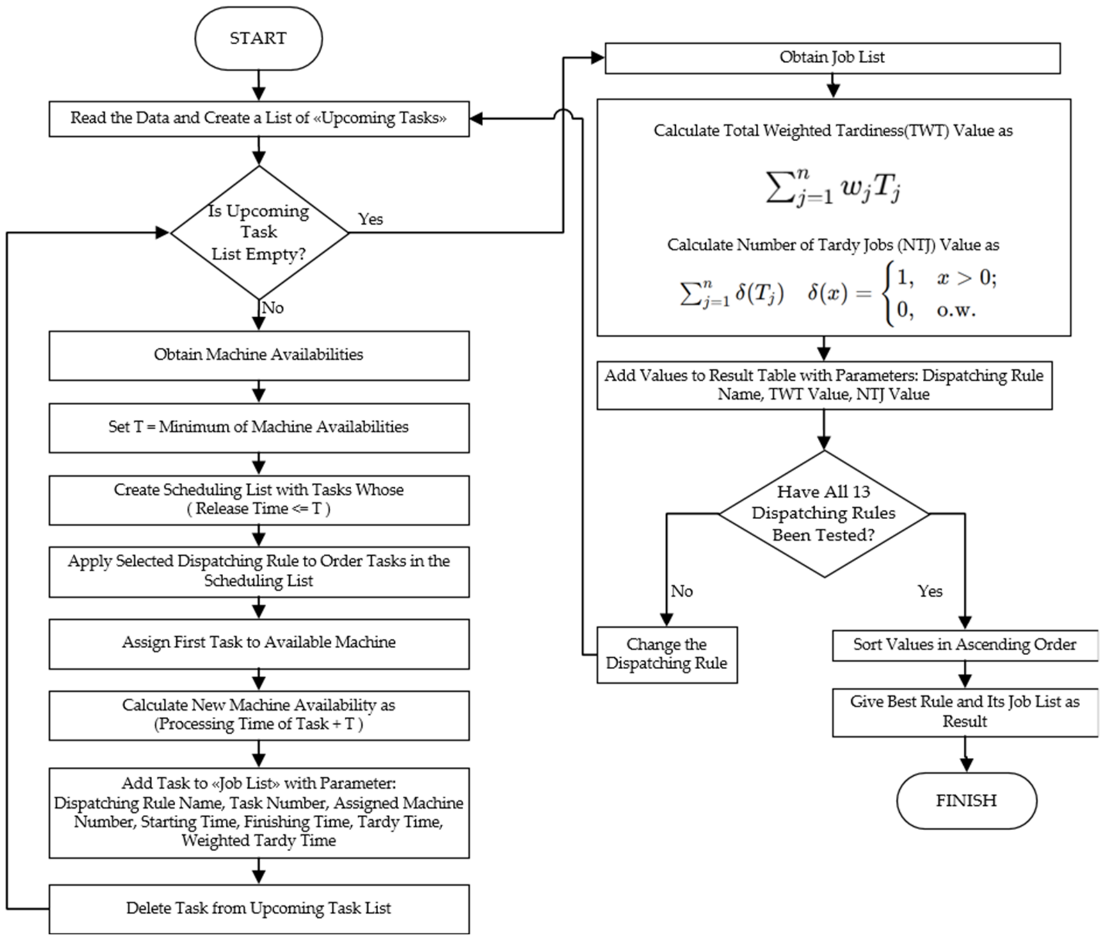
START
Read the Data and Create a List of «Upcoming Tasks»
Is Upcoming Task List Empty?
Yes
No
Obtain Machine Availabilities
Set T = Minimum of Machine Availabilities
Create Scheduling List with Tasks Whose (Release Time <= T)
Apply Selected Dispatching Rule to Order Tasks in the Scheduling List
Assign First Task to Available Machine
Calculate New Machine Availability as (Processing Time of Task + T)
Add Task to «Job List» with Parameter: Dispatching Rule Name, Task Number, Assigned Machine Number, Starting Time, Finishing Time, Tardy Time, Weighted Tardy Time
Delete Task from Upcoming Task List
Obtain Job List
Calculate Total Weighted Tardiness(TWT) Value as
$\sum_{j=1}^{n} w_j T_j$
Calculate Number of Tardy Jobs (NTJ) Value as
$\sum_{j=1}^{n} \delta(T_j) \quad \delta(x) = \begin{cases} 1, & x > 0; \\ 0, & \text{o.w.} \end{cases}$
Add Values to Result Table with Parameters: Dispatching Rule Name, TWT Value, NTJ Value
Have All 13 Dispatching Rules Been Tested?
No
Yes
Change the Dispatching Rule
Sort Values in Ascending Order
Give Best Rule and Its Job List as Result
FINISH

Figure 5. MIU procedure.

If there are still unstarted jobs, machine availability is assessed, and an objective function target may be updated by management at this moment. Management may define the objective function at the outset of the scheduling process and update it at the beginning of each period if necessary. The unstarted job list is then rescheduled using MIU, and the cycle repeats with another manufacturing period. This iterative process continues until all jobs are completed, at which point the process concludes.

Figure 5 illustrates the procedure of MIU. The proposed scheduling approach begins by reading the data set and generating a list of upcoming tasks. If the list is empty, the process moves directly to performance evaluation. Otherwise, the algorithm determines machine availability and sets the scheduling time T as the minimum available time among all machines. At this point, a scheduling list is created, including all tasks with a release time less than or equal to T. The selected dispatching rule is then applied to determine the task order, and the first task in the ordered list is assigned to an available machine. Once assigned, the machine's availability is updated by adding the task's processing time to the current time. The task is then recorded in the job list with its relevant parameters, including the dispatching rule name, task number, assigned machine number, starting and finishing times, tardy time, and weighted tardy time. Afterward, the scheduled task is removed from the upcoming task list, and the process continues iteratively until all tasks are scheduled. Upon completing the scheduling process, the algorithm evaluates performance by calculating the total weighted tardiness (TWT) and the number of tardy jobs (NTJ). These metrics are stored for each dispatching rule, and if all 13 predefined dispatching rules have not yet been tested, the algorithm switches to the next rule and repeats the process. Once all dispatching rules have been evaluated, the results are sorted, and the best-performing rule—based on the lowest TWT value—is selected as the optimal scheduling approach. Finally, the selected rule and its corresponding job sequence are presented as the final output.

It is accepted that the periodic meeting is held once a month, each day is two units of time, each week has 5 days, each month has 4 weeks. This calculation makes the period length 40 units.

The machine availability time is calculated based on the expected values of the planned jobs in the previous period. If only the rework process remains after the main transactions are completed in the last period, this process is completed without waiting for the periodic meeting. The rework process of the delayed job is considered late even if the main process is completed on time. If rework is needed within a period, no complete or partial rescheduling should be conducted without a periodic meeting and a decision should be waited after the meeting. If only the rework process remains after the main transactions are completed in the last period, this process is completed without waiting for the periodic meeting.

Initial scheduling was prepared for TWT and NTJ problems separately, and the Gantt charts which are given below as Figures 6 and 7 were obtained.

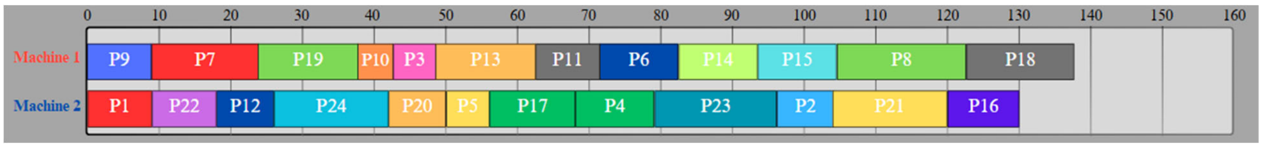

Figure 6. Initial scheduling for TWT problem.

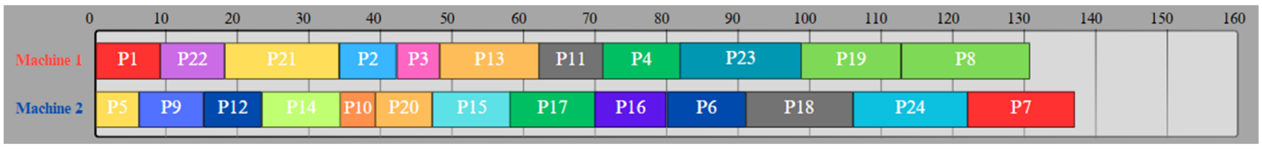

Figure 7. Initial scheduling for NTJ problem.

After a period of production based on the initial schedule, in accordance with the workflow chart, rework jobs, if any, are included in the list of unfinished jobs and the lists are updated for rescheduling. Rescheduling operations are carried out through two strategies that can be selected in periodic meetings, and the results are given separately below.

4.1. Total Weighted Tardiness (TWT) Minimization

The Gantt chart and detailed result table obtained when the decision-maker determines the strategy as TWT are given below as Figure 8 and Table 4.

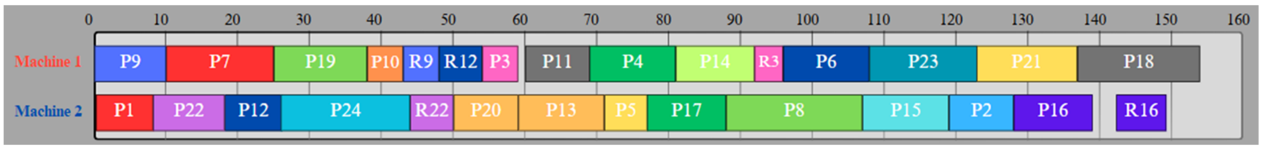

Figure 8. Completed TWT strategy Gantt chart.

Table 4. TWT strategy results and output details.

T = 160; TWT = 858										
PCB No	**Release Date**	**Due Date**	**Weight**	**Expected Processing Time**	**Realized Processing Time**	**Rework Need?**	**Starting Time**	**Finishing Time**	**Delay**	**Weighted Tardiness**
P01	0	30	3	9	8		0	8	0	0
P02	30	70	1	8	9		119	128	58	58
P03	40	70	2	6	5	Yes	54	59	0	0
P04	60	80	3	11	12		69	81	1	3
P05	0	50	1	6	6		71	77	27	27
P06	60	100	3	11	12		96	108	8	24
P07	0	30	1	15	15		10	25	0	0
P08	20	50	1	18	19		88	107	57	57
P09	0	30	3	9	10	Yes	0	10	0	0
P10	30	60	3	5	5		38	43	0	0
P11	60	80	3	9	9		60	69	0	0
P12	10	50	3	8	8	Yes	18	26	0	0
P13	30	60	3	14	12		59	71	11	33
P14	20	50	1	11	11		81	92	42	42
P15	30	50	1	11	12		107	119	69	69
P16	60	100	1	10	11	Yes	128	139	39	39
P17	20	50	2	12	11		77	88	38	76
P18	90	110	1	15	17		137	154	44	44
P19	20	40	3	14	13		25	38	0	0
P20	30	60	3	8	9		50	59	0	0
P21	10	50	1	16	14		123	137	87	87
P22	0	30	3	9	10	Yes	8	18	0	0
P23	60	90	3	17	15		108	123	33	99
P24	20	40	3	16	18		26	44	4	12
Rew03	62	70	2	3	4		92	96	26	52
Rew09	13	30	3	5	5		43	48	18	54
Rew12	28	50	3	6	6		48	54	4	12
Rew16	142	100	1	6	7		142	149	49	49
Rew22	20	30	3	7	6		44	50	20	60

4.2. Number of Tardy Job (NTJ) Minimization

The Gantt chart and detailed result table obtained when the decision-maker determines the strategy as the NTJ are given below as Figure 9 and Table 5.

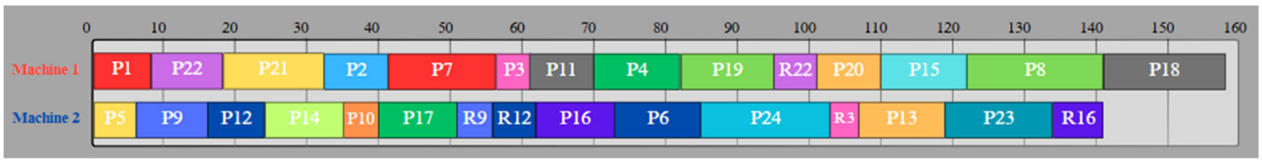

Figure 9. Completed NTJ strategy Gantt chart.

Table 5. NTJ strategy results and output details.

T = 160; NTJ = 16										
PCB No	Release Date	Due Date	Weight	Expected Processing Time	Realized Processing Time	Rework Need?	Starting Time	Finishing Time	Delay	Weighted Tardiness
P01	0	30	3	9	8		0	8	0	0
P02	30	70	1	8	9		32	41	0	0
P03	40	70	2	6	5	Yes	56	61	0	0
P04	60	80	3	11	12		70	82	2	6
P05	0	50	1	6	6		0	6	0	0
P06	60	100	3	11	12		73	85	0	0
P07	0	30	1	15	15		41	56	26	26
P08	20	50	1	18	19		122	141	91	91
P09	0	30	3	9	10	Yes	6	16	0	0
P10	30	60	3	5	5		35	40	0	0
P11	60	80	3	9	9		61	70	0	0
P12	10	50	3	8	8	Yes	16	24	0	0
P13	30	60	3	14	12		107	119	59	177
P14	20	50	1	11	11		24	35	0	0
P15	30	50	1	11	12		110	122	72	72
P16	60	100	1	10	11	Yes	62	73	0	0
P17	20	50	2	12	11		40	51	1	2
P18	90	110	1	15	17		141	158	48	48
P19	20	40	3	14	13		82	95	55	165
P20	30	60	3	8	9		101	110	50	150
P21	10	50	1	16	14		18	32	0	0
P22	0	30	3	9	10	Yes	8	18	0	0
P23	60	90	3	17	15		119	134	44	132
P24	20	40	3	16	18		85	103	63	189
Rew03	61	70	2	3	4		103	107	37	74
Rew09	13	30	3	5	5		51	56	26	78
Rew12	28	50	3	6	6		56	62	12	36
Rew16	142	100	1	6	7		134	141	41	41
Rew22	20	30	3	7	6		95	101	71	213

5. Evaluations and Comparisons with the Current Situation

This section presents the experimental setup, including test scenarios based on real-world data, and evaluates the performance of the MIU approach. Results are compared with traditional scheduling methods, highlighting improvements in minimizing tardiness and optimizing resource utilization. Key performance metrics and visualizations, such as Gantt charts, are provided to support the analysis.

In the current situation, the company performs makespan minimization by taking into account the expected production times before the production process starts, and for this purpose, it accepts the LPT rule results as an initial schedule. Jobs that cannot be produced within the expected time are not updated subsequently. If rework is needed, these jobs are given priority, and these jobs are conducted when the machines are first empty and the remaining jobs are shifted so that their order is not disrupted. When the problem in our article was solved with the company's current method, it gave the result of LPT 142 for the initial schedule that minimizes the makespan. When the company's rescheduling method was implemented using the same errors and actual processing times as those of the initial schedule based on the LPT rule, the resulting Gantt charts (Figures 10 and 11) and outcome table (Table 6) were obtained.

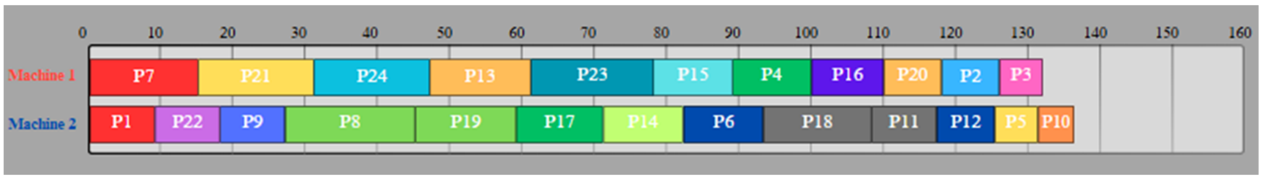

Figure 10. Initial schedule when applying company's current scheduling method.

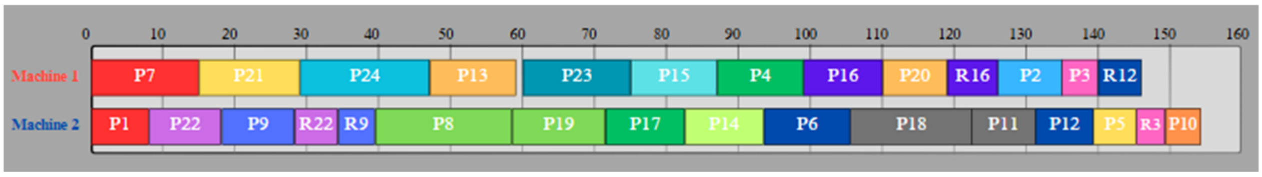

Figure 11. Final schedule when applying company's current rescheduling method.

Table 6. Result table of company's current rescheduling method.

T = 160; TWT = 1633, NTJ = 19										
PCB No	**Release Date**	**Due Date**	**Weight**	**Expected Processing Time**	**Realized Processing Time**	**Rework Need?**	**Starting Time**	**Finishing Time**	**Delay**	**Weighted Tardiness**
P01	0	30	3	9	8		0	8	0	0
P02	30	70	1	8	9		126	135	65	65
P03	40	70	2	6	5	Yes	135	140	70	140
P04	60	80	3	11	12		87	99	19	57
P05	0	50	1	6	6		139	145	95	95
P06	60	100	3	11	12		93	105	5	15
P07	0	30	1	15	15		0	15	0	0
P08	20	50	1	18	19		39	58	8	8
P09	0	30	3	9	10	Yes	18	28	0	0
P10	30	60	3	5	5		149	154	94	282
P11	60	80	3	9	9		122	131	51	153
P12	10	50	3	8	8	Yes	131	139	89	267
P13	30	60	3	14	12		47	59	0	0
P14	20	50	1	11	11		82	93	43	43
P15	30	50	1	11	12		75	87	37	37
P16	60	100	1	10	11	Yes	99	110	10	10
P17	20	50	2	12	11		71	82	32	64
P18	90	110	1	15	17		105	122	12	12
P19	20	40	3	14	13		58	71	31	93
P20	30	60	3	8	9		110	119	59	177
P21	10	50	1	16	14		15	29	0	0
P22	0	30	3	9	10	Yes	8	18	0	0
P23	60	90	3	17	15		60	75	0	0
P24	20	40	3	16	18		29	47	7	21
Rew03	142	70	2	3	4		145	149	79	158

Table 6. *Cont.*

T = 160; TWT = 1633, NTJ = 19										
PCB No	**Release Date**	**Due Date**	**Weight**	**Expected Processing Time**	**Realized Processing Time**	**Rework Need?**	**Starting Time**	**Finishing Time**	**Delay**	**Weighted Tardiness**
Rew09	31	30	3	5	5		34	39	9	27
Rew12	141	50	3	6	6		140	146	96	288
Rew16	113	100	1	6	7		119	126	26	26
Rew22	20	30	3	7	6		28	34	4	12

The company's current strategy and the results obtained by applying our MIU method are given in Table 7 below.

Table 7. Comparison between current method and MIU approach performances.

	Total Weighted Tardiness Minimization	**Number of Tardy Jobs Minimization**
Current Method	1633	19
MIU Approach	858	16
Improvement	**47.46%**	**15.79%**

The proposed method outperformed the company's current approach for both objective functions. This improvement stems from the implementation of complete rescheduling and a flexible scheduling method.

Apart from data which are given in Section 3 and Results and which are given above, 20 different data sets are worked on and results are given in Table 8 altogether. These 20 scenarios were evaluated by applying random rework requirements, ranging from 0 to 5, derived from past company data. The analysis was conducted on 16, 20, and 24 job batches used in company planning. Scenario generation continued until the results converged to a stable average.

Table 8. 20 Different scenario improvement results.

Scenario	**TWT**			**NTJ**		
	Current Method	**MIU Approach**	**Improvement**	**Current Method**	**MIU Approach**	**Improvement**
1	1633	858	47.46%	19	16	15.79%
2	1127	844	25.11%	17	13	23.53%
3	1019	540	47.01%	19	14	26.32%
4	1050	629	40.10%	18	12	33.33%
5	1319	806	38.89%	18	15	16.67%
6	624	292	53.21%	8	7	12.50%
7	478	115	75.94%	8	6	25.00%
8	676	213	68.49%	9	9	0.00%
9	767	563	26.60%	14	13	7.14%
10	552	295	46.56%	10	9	10.00%
11	686	354	48.40%	11	9	18.18%

Table 8. *Cont.*

Scenario	TWT			NTJ		
	Current Method	MIU Approach	Improvement	Current Method	MIU Approach	Improvement
12	235	16	93.19%	7	3	57.14%
13	322	164	49.07%	8	5	37.50%
14	263	37	85.93%	8	3	62.50%
15	360	193	46.39%	8	7	12.50%
16	446	337	24.44%	17	10	41.18%
17	457	245	46.39%	15	8	46.67%
18	576	378	34.38%	18	10	44.44%
19	695	502	27.77%	16	12	25.00%
20	692	426	38.44%	15	11	26.67%
Average			**48.19%**			**27.10%**

To ensure transparency and reproducibility, the complete data set, including input parameters and corresponding results for all 20 scenarios, has been made publicly available at https://github.com/miulucak/Dynamic-Scheduling-in-Identical-Parallel-Machines-Environments (accessed on 21 December 2024). Readers can access the data set through this repository for further analysis and verification.

As can be seen in the table, an average improvement of approximately 50 percent in TWT and 27 percent in the NTJ was observed. The main reason for this improvement is that the weight and due date parameters are specifically taken into account in the dispatching rules we use in the MIU approach. After objective function assessment, performance criteria which are given in Section 2 have been calculated. In this study, the change in the start times of jobs is considered as the stability criterion. Robustness is assessed through three metrics: the largest difference in the amount of delay as robustness1, the difference in the total amount of delay as robustness2, and the difference in the total weighted amount of delay as robustness3. Lastly, nervousness is defined as the difference between the initially planned and resulting machine job assignments. According to our definitions, lower values for stability, robustness, and nervousness indicate a positive characteristic. In Table 9, the company's current strategy and MIU approach results in terms of stability, robustness, and nervousness are given.

Table 9. Comparison between current method and MIU approach in terms of evaluation criteria.

	Current Method	MIU Approach	
		Total Weighted Tardiness Minimization	Total Tardy Jobs Minimization
stability	158	227	454
robustness	18	27	81
robustness2	133	149	127
robustness3	276	261	318
nervousness	1	8	7

It is important to note that the numerical values in Table 9 present a comparative result rather than absolute quantities. For instance, doubling the stability value does not imply that the solution is twice as bad but rather indicates reduced stability. The illustrative nature of these figures is crucial in this context. These numbers could vary significantly depending on the characteristics of the data set; however, the underlying meaning of the metrics remains consistent.

In the current production system, disruptions are addressed by promptly performing partial rescheduling rather than complete rescheduling, ensuring minimal changes to the remaining orders. This approach leads to better stability and nervousness values compared to the MIU approach.

By definition and methodology, complete rescheduling typically results in greater changes in job ordering and machine assignments. Therefore, it is natural for stability and nervousness values to deteriorate. However, no significant differences were observed in robustness values. This is because the MIU approach directly considers delay amounts and the number of delayed jobs as objective functions, ensuring that the parameters within the robustness criteria (e.g., changes in tardiness) remain controlled to prevent deterioration in the objective function values with each rescheduling.

With the MIU approach, the deterioration of stability and nervousness values is deemed acceptable because the primary goal is to minimize the number of delayed jobs and the overall delay amount. Decision-makers must weigh the trade-offs between these approaches when selecting a strategy. The classical method provides an agile and stable solution without focusing on outcomes, while the MIU approach offers a planned, result-oriented framework that prioritizes outcomes over stability and robustness by embracing changes.

6. Conclusions and Future Works

In our study, a scheduling problem frequently encountered by a company manufacturing in the space sector is discussed. With a process and method that we developed, we tried to solve the need for rescheduling caused by the difference between expected and actual values in the production period and the return of finished work to the system due to production errors. There are two strategies that company management can choose. These are minimization of the total weighted tardiness time and minimization of the number of delayed jobs. For these strategies, visible improvements have been achieved compared to the current system. Through rigorous analysis and comparison with traditional methods, MIU emerges as a powerful tool for optimizing scheduling strategies and enhancing overall efficiency.

A key strength of MIU lies in its dynamic approach to scheduling, offering tailored solutions for each strategy while seamlessly adapting to evolving production scenarios. This flexibility empowers decision-makers to navigate uncertainties and optimize scheduling outcomes in real time, a stark departure from the rigid solutions of the past.

Our process gives management the opportunity to assess strategy in periodic meetings and allows rework operations to be completely rescheduled along with all remaining work. Since the problems were NP-hard, there was a need to develop a heuristic procedure. The MIU Approach we developed is structured to give good results very quickly and is designed to use the advantages and avoid the disadvantages of dispatching rules for each situation by using 13 different rules.

In our study, the stability, robustness, and nervousness value changes between the results obtained by the current application of the workplace and the results given by the developed algorithm are also shown. The trade-off between the improvement in objective function values and the worsening of the evaluation criteria was mentioned. In addition, expressing numerically the inverse proportion of the nervousness value to stability in the literature is another important contribution of our study.

As we chart the course forward, we may also analyze the impact of varying period durations across diverse scenarios to further validate the robustness of the methodology. Additionally, investigations into scenarios involving variable technician speeds could offer deeper insights into real-world production dynamics. In future work, it is planned to conduct additional analyses to further validate the improvements introduced by the MIU approach. This will include statistical tests and significance analysis to quantify enhancements compared to existing methods, providing a more robust evaluation of the results.

Author Contributions: Writing—original draft, M.İ.U.; Supervision, H.G. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Data is contained within the article.

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

References

1. Pinedo, M.L. *Scheduling: Theory, Algorithms, and Systems*; Springer: Berlin/Heidelberg, Germany, 2016.
2. Ghaleb, M.; Taghipour, S. Dynamic shop-floor scheduling using real-time information—A case study from the thermoplastic industry. *J. Manuf. Syst.* **2023**, *69*, 271–286. [CrossRef]
3. Li, Y.; Li, Y.; Cheng, J.; Wu, P. Order Assignment and Scheduling for Personal Protective Equipment Production During the Outbreak of Epidemics. *IEEE Trans. Eng. Manag.* **2022**, *69*, 1285–1298. [CrossRef]
4. Turkcan, A.; Akturk, M.S.; Storer, R.H. Predictive reactive scheduling with controllable processing times and earliness tardiness penalties. *Int. J. Prod. Res.* **2009**, *47*, 1233–1258. [CrossRef]
5. Rahmani, D.; Heydari, M. Robust and stable flow shop scheduling with unexpected arrivals of new jobs and uncertain processing times. *J. Manuf. Syst.* **2014**, *33*, 84–92. [CrossRef]
6. Al-Behadili, M.; Ouelhadj, D.; Jones, D. Multi objective biased randomised iterated greedy for robust permutation flow shop scheduling problem under disturbances. *J. Manuf. Syst.* **2019**, *51*, 224–235. [CrossRef]
7. Vieira, G.E.; Herrmann, J.W.; Lin, E. Rescheduling Manufacturing Systems: A Framework of Strategies, Policies, and Methods. *J. Sched.* **2003**, *6*, 39–62. [CrossRef]
8. Cowling, P.I.; Johansson, M. Using real-time information for effective dynamic scheduling. *Eur. J. Oper. Res.* **2002**, *139*, 230–244. [CrossRef]
9. Ouelhadj, D.; Petrovic, S. A survey of dynamic scheduling in manufacturing systems. *J. Sched.* **2009**, *12*, 417–443. [CrossRef]
10. Aytug, H.; Lawley, M.A.; McKay, K.; Mohan, S.; Uzsoy, R. Executing production schedules in the face of uncertainties: A review and some future directions. *Eur. J. Oper. Res.* **2005**, *161*, 86–110. [CrossRef]
11. Herroelen, W.; Leus, R. Project scheduling under uncertainty: Survey and research potentials. *Eur. J. Oper. Res.* **2005**, *165*, 289–306. [CrossRef]
12. Mehta, S.V.; Uzsoy, R. Predictable scheduling of a single machine subject to breakdowns. *Int. J. Comput. Integr. Manuf.* **1999**, *12*, 15–38. [CrossRef]
13. Vieira, G.E.; Herrmann, J.W.; Lin, E. Analytical models to predict the performance of a single machine system under periodic and event-driven rescheduling strategies. *Int. J. Prod. Res.* **2000**, *38*, 1899–1915. [CrossRef]
14. Liu, N.; Abdelrahman, M.; Ramaswamy, S. A Complete Multiagent Framework for Robust and Adaptable Dynamic Job Shop Scheduling. *IEEE Trans. Syst. Man Cybern. Part C Appl. Rev.* **2007**, *37*, 904–916. [CrossRef]
15. Li, K.; Deng, Q.; Zhang, L.; Fan, Q.; Gong, G.; Ding, S. An effective MCTS-based algorithm for minimizing makespan in dynamic flexible job shop scheduling problem. *Comput. Ind. Eng.* **2021**, *155*, 107157. [CrossRef]
16. Bożek, A.; Wysocki, M. Off-Line And Dynamic Production Scheduling—A Comparative Case Study. *Manag. Prod. Eng. Rev.* **2016**, *7*, 21–32. [CrossRef]
17. Penz, B.; Rapine, C.; Trystram, D. Sensitivity analysis of scheduling algorithms. *Eur. J. Oper. Res.* **2001**, *134*, 606–615. [CrossRef]

18. Yang, J.; Yu, G. On the robust single machine scheduling problem. *J. Comb. Optim.* **2002**, *6*, 17–33. [CrossRef]
19. Hall, N.G.; Posner, M.E. Sensitivity analysis for scheduling problems. *J. Sched.* **2004**, *7*, 49–83. [CrossRef]
20. Leon, V.J.; Wu, S.D.; Storer, R.H. Robustness measures and robust scheduling for job shops. *IIE Trans.* **1994**, *26*, 32–41. [CrossRef]
21. Wu, S.D.; Storer, R.H.; Chang, P.C. A Rescheduling Procedure for Manufacturing Systems Under Random Disruptions. In Proceedings of the Joint USA/German Conference on New Directions for Operations Research in Manufacturing, Gaithersburg, MD, USA, 30–31 July 1991; pp. 292–306.
22. Wu, S.D.; Storer, R.H.; Chang, P.C. One machine rescheduling heuristics with efficiency and stability as criteria. *Comput. Oper. Res.* **1993**, *20*, 1–14. [CrossRef]
23. Yang, B.; Geunes, J. Predictive-reactive scheduling on a single resource with uncertain future jobs. *Eur. J. Oper. Res.* **2008**, *189*, 1263–1282. [CrossRef]
24. Duenas, A.; Petrovic, D. An approach to predictive-reactive scheduling of parallel machines subject to disruptions. *Int. J. Prod. Econ.* **2007**, *103*, 31–42. [CrossRef]
25. Kaya, S.; Karaçizmeli, İ.H. Hazırlık Zamanlı Ortak Teslim Tarihli Özdeş Paralel Makine Çizelgeleme Problemlerinin Çok Amaçlı Çözümü. *Harran Üniv. Mühendis. Derg.* **2018**, *3*, 205–213.
26. Tighazoui, A.; Sauvey, C.; Sauer, N. Predictive-reactive strategy for identical parallel machine rescheduling. *J. Manuf. Syst.* **2021**, *59*, 230–245. [CrossRef]
27. Geurtsen, M.; Adan, J.; Akçay, A. Integrated maintenance and production scheduling for unrelated parallel machines with setup times. *Flex. Serv. Manuf. J.* **2024**, *36*, 1046–1079. [CrossRef]
28. Tighazoui, A.; Sauvey, C.; Sauer, N. Predictive-reactive Strategy for Flowshop Rescheduling Problem—Minimizing the Total Weighted Waiting Times and Instability. *J. Manuf. Syst.* **2021**, *60*, 28–40. [CrossRef]
29. Luo, J.; Fujimura, S.; El Baz, D.; Plazolles, B. GPU based parallel genetic algorithm for solving an energy efficient dynamic flexible flow shop scheduling problem. *J. Parallel Distrib. Comput.* **2019**, *123*, 184–197. [CrossRef]
30. Paprocka, I. Evaluation of the Effects of a Machine Failure on the Robustness of a Job Shop System—Proactive Approaches. *Sustainability* **2019**, *11*, 65. [CrossRef]
31. Zhang, J.; Ding, G.; Zou, Y.; Qin, S.; Fu, J. Review of job shop scheduling research and its new perspectives under industry 4.0. *J. Intell. Manuf.* **2017**, *30*, 1809–1830. [CrossRef]
32. Gomes, M.; Barbosa-Póvoa, A.; Novais, A. Reactive scheduling in a make-to-order flexible job shop with re-entrant process and assembly: A mathematical programming approach. *Comput. Oper. Res.* **2013**, *40*, 1891–1900. [CrossRef]
33. Nouiri, M.; Bekrar, A.; Trentesaux, D. Towards Energy Efficient Scheduling and Rescheduling for Dynamic Flexible Job Shop Problem. *J. Manuf. Syst.* **2018**, *47*, 78–90. [CrossRef]
34. Moghaddam, S.K.; Saitou, K. A novel predictive-reactive rescheduling method for products assembly lines with optimal pegging. *IEEE Trans. Autom. Sci. Eng.* **2022**, *19*, 360–375. [CrossRef]
35. Valledor, P.; Gomez, A.; Puente, J.; Fernandez, I. Solving Rescheduling Problems in Dynamic Permutation Flow Shop Environments with Multiple Objectives Using the Hybrid Dynamic Non-Dominated Sorting Genetic II Algorithm. *Mathematics* **2022**, *10*, 2395. [CrossRef]
36. Ouelhadj, D.; Cowling, P.I.; Petrovic, S. Utility and Stability Measures for Agent-Based Dynamic Scheduling of Steel Continuous Casting. *IEEE Trans. Autom. Sci. Eng.* **2003**, *1*, 74–88.
37. Tang, L.; Wang, X. A predictive reactive scheduling method for color-coating production in steel industry. *J. Manuf. Syst.* **2008**, *27*, 83–91. [CrossRef]
38. Abuhasel, K. A Dynamic Approach of Using Dispatching Rules in Scheduling. *J. Teknol.* **2016**, *78*, 179–184. [CrossRef]
39. Petrovic, D.; Duenas, A. A fuzzy logic based production scheduling-rescheduling in the presence of uncertain disruptions. *Int. J. Prod. Res.* **2006**, *44*, 3571–3587. [CrossRef]
40. Sobaszek, L.; Gola, A.; Swic, A. Predictive Scheduling as a Part of Intelligent Job Scheduling System. *J. Intell. Manuf.* **2018**, *29*, 1817–1835.
41. Valledor, P.; Gomez, A.; Priore, P.; Puente, J. Solving multi objective rescheduling problems in dynamic permutation flow shop environments with disruptions. *Comput. Ind. Eng.* **2018**, *124*, 354–367. [CrossRef]
42. Geurtsen, M.; Adan, J.; Akçay, A. Quick dispatching-rules-based solution for the two parallel machines problem under mold constraints. *Flex. Serv. Manuf. J.* **2024**, *36*, 224–249.
43. Ebrahimi, M.; Ghomi, F.; Karimi, B. Hybrid flow shop scheduling with sequence dependent family setup time and uncertain due dates. *Appl. Math. Model.* **2014**, *38*, 2490–2504. [CrossRef]
44. Xin, L.; Chu, F.; Zheng, F.; Chu, C.; Liu, M. Parallel machine scheduling with stochastic release times and processing times. *Int. J. Prod. Res.* **2021**, *59*, 6327–6346.
45. Cheng, T.C.E. Optimal due-date assignment for a single machine sequencing problem with random processing times. *Int. J. Syst. Sci.* **1986**, *17*, 1139–1144. [CrossRef]

46. Brusaferri, A.; Leo, E.; Nicolosi, L.; Ramin, D.; Spinelli, S. Integrated automation system with PSO based scheduling for PCB remanufacturing plants. *Comput. Ind. Eng.* **2019**, *129*, 179–190.
47. Shokraneh, K.; Moghaddam, S.; Saitou, K. Predictive-Reactive Rescheduling for New Order Arrivals with Optimal Dynamic Pegging. *Int. J. Prod. Res.* **2020**, *58*, 4232–4246.
48. Barták, R.; Vlk, M. Machine Breakdown Recovery in Production Scheduling with Simple Temporal Constraints. *Int. J. Prod. Res.* **2015**, *53*, 106–122.
49. Barták, R.; Vlk, M. Hierarchical Task Model for Resource Failure Recovery in Production Scheduling. *Comput. Oper. Res.* **2017**, *83*, 11–21.
50. Takeda-Berger, S.; Frazzon, E. An inventory data driven model for predictive reactive production scheduling. *J. Manuf. Syst.* **2023**, *69*, 324–335. [CrossRef]
51. Takeda-Berger, S.; Zanella, R.; Frazzon, E. Towards a data-driven predictive-reactive production scheduling approach based on inventory availability. *Comput. Ind. Eng.* **2019**, *133*, 11–23. [CrossRef]
52. Heydari, M.; Soudi, A. Predictive/Reactive Planning and Scheduling of a Surgical Suite with Emergency Patient Arrival. *Comput. Ind. Eng.* **2016**, *98*, 92–106. [CrossRef]
53. Manzini, M.; Demeulemeester, E.; Urgo, M. A predictive–reactive approach for the sequencing of assembly operations in an automated assembly line. *Int. J. Prod. Res.* **2022**, *60*, 30–44. [CrossRef]
54. Sabuncuoglu, I.; Bayiz, M. Analysis of reactive scheduling problems in a job shop environment. *Eur. J. Oper. Res.* **2000**, *126*, 567–586. [CrossRef]
55. Tighazoui, A.; Sauvey, C.; Sauer, N. New efficiency-stability criterion in a rescheduling problem with dynamic jobs weights. *Int. J. Prod. Econ.* **2020**, *229*, 107855.
56. Sun, M.; Liu, M.; Zhang, X.; Ling, L.; Ge, M.; Liu, C.; Rui, Z. Real-time rescheduling for smart shop floors: An integrated method. *Flex. Serv. Manuf. J.* **2024**, 1–34. [CrossRef]
57. Qiao, F.; Ma, Y.; Zhou, M.; Wu, Q. A Novel Rescheduling Method for Dynamic Semiconductor Manufacturing Systems. *IEEE Trans. Semicond. Manuf.* **2020**, *33*, 543–555. [CrossRef]
58. Wang, B.; Han, X.; Zhang, X.; Zhang, S. Predictive-reactive scheduling for single surgical suite subject to random emergency surgery. *Int. J. Prod. Res.* **2015**, *53*, 761–777. [CrossRef]
59. Baker, K.R.; Trietsch, D. *Principles of Sequencing and Scheduling*; John Wiley & Sons: Hoboken, NJ, USA, 2018.
60. Panwalkar, S.S.; Iskander, W. A Survey of Scheduling Rules. *Oper. Res.* **1977**, *25*, 45–61. [CrossRef]
61. Rajendran, C.; Holthaus, O. A comparative study of dispatching rules in dynamic flowshops and jobshops. *Eur. J. Oper. Res.* **1999**, *116*, 156–170. [CrossRef]
62. Chiang, T.-C.; Fu, L.-C. Using dispatching rules for job shop scheduling with due date-based objectives. *Int. J. Prod. Res.* **2007**, *45*, 3245–3273. [CrossRef]