Available online at: http://e-jurnal.lppmunsera.org/index.php/JSMI

Jurnal Sistem dan Manajemen Industri
(Journal System and Management of Industries)

ISSN (Print) 2580-2887 ISSN (Online) 2580-2895

A genetic algorithm-based predictive–reactive scheduling for no-wait flow shop with sequence-dependent setup times under dynamic job arrivals

Muhammad Ridwan Andi Purnomo[1*], Azmi Hassan[2]
[1]Department of Industrial Engineering, Universitas Islam Indonesia, Jl. Kaliurang KM 14.5, Sleman, Yogyakarta, 55584, Indonesia
[2]Electrical, Electronics & Automation Section, Universiti Kuala Lumpur Malaysian Spanish Institute, Kulim Hi-Tech Park, 09000, Kulim, Kedah, Malaysia

ARTICLE INFORMATION

Article history:

Received: November 26, 2025
Revised: June 28, 2026
Accepted: June 30, 2026

Keywords:

Genetic algorithm
Multi-objective scheduling
No-wait flow shop
Predictive-reactive scheduling
Sequence dependent setup time

ABSTRACT

Modern electronics controller manufacturing operates in highly dynamic environments where production schedules must rapidly adapt to newly arriving customer orders while maintaining production efficiency and delivery performance. However, conventional predictive scheduling alone is often insufficient because schedule revisions are required for unexpected order arrivals, which may increase production completion time and customer lateness. Therefore, this study aims to develop a two-stage predictive–reactive scheduling framework for a no-wait flow shop with sequence-dependent setup times (SDST), involving 20 initial jobs with varying release times and 7 dynamically arriving jobs with distinct due dates. In Stage 1, a Genetic Algorithm (GA) is employed to maximize the number of initial jobs completed before a 300-minute production cut-off, establishing a predictive baseline schedule. In Stage 2, the proposed GA performs reactive rescheduling by integrating the unprocessed initial jobs with the newly arriving jobs to minimize the makespan deviation from the predictive schedule and reduce the new jobs' lateness. Based on a case study, the results demonstrate that the proposed model and GA optimization effectively balance schedule stability and responsiveness. The results show that while reactive rescheduling introduces a makespan deviation of 183-time units, incoming job lateness is substantially reduced, providing a trade-off between operational continuity and service-level performance.

*Corresponding Author

Muhammad Ridwan Andi Purnomo
E-mail: ridwan_ie@uii.ac.id

This is an open access article under the CC–BY-NC-SA license.

© 2026 Some rights reserved

1. INTRODUCTION

Improving productivity has long been a primary objective in manufacturing and service systems. Productivity can be defined as the ratio between the output produced and the resources used. In operational contexts, increasing productivity often entails either maximizing output or minimizing time and cost. *Makespan*, which refers to the total time required to complete a set of jobs, is a performance criterion for evaluating operational efficiency in machine scheduling. Minimizing the *makespan* directly contributes to higher production throughput and better resource use. In addition, customer satisfaction must be maintained not only by reducing costs due to high throughput but also by avoiding excessive earliness or lateness, which can increase hidden costs such as inventory build-up or lateness penalties.

Scheduling plays a vital role in aligning production activities with operational goals and customer requirements. Specifically, developing schedules that ensure high productivity while maintaining service-level commitments is a challenge. Technically, earliness and lateness penalties are often introduced to capture deviations from *debates*, which are critical in order-based production systems. Hence, an integrated scheduling objective that maximizes output and

deviations from debates is a beneficial formulation in a mixed production system [1].

In real-world systems, scheduling rarely occurs in a static environment. New jobs may arrive, machine availability may change, or demand may fluctuate, prompting the need for reactive scheduling. Reactive scheduling refers to updating an existing schedule in response to unforeseen disruptions or new order arrivals [2]. Unlike predictive scheduling, which assumes complete information at the planning stage, reactive scheduling must accommodate dynamic changes while preserving stability and efficiency [3]. This makes the scheduling problem more complex, requiring more complicated models and many decision variables to reflect changing priorities and constraints.

Reactive scheduling involves a variety of decisions, such as job sequencing, timing, machine assignment, and rescheduling effort, all of which contribute to its increased complexity in dynamic manufacturing systems [4]. The situation becomes more complicated when multiple objectives, such as maximizing job completions within the available time window and aligning jobs with *due dates*, are combined, as this leads to conflicting trade-offs and a combinatorial explosion of possibilities [5]. Traditional optimization techniques are often unable to efficiently handle the nonlinearity and complexity inherent in dynamic, multiobjective scheduling problems [6]. Thus, metaheuristic algorithms, such as the Genetic Algorithm (GA), have emerged as promising solutions due to their ability to navigate large and complex search spaces and find optimum or near-optimum solutions in reasonable time [7]. GA simulates natural evolution via populations of solutions that evolve over generations through selection, crossover, and mutation, making it particularly well-suited to scheduling applications where job sequences must be optimized under dynamic conditions [8].

Previous studies have shown that GA can solve scheduling problems under dynamic or reactive conditions. For example, Panda et al. applied GA to solve the multi-objective dynamic job-shop scheduling problem to enhance system responsiveness and stability. The adaptability of GA makes it suitable for handling sequence-dependent setup times, job release constraints, and job priorities. By using crossover, mutation, and selection operators tailored to scheduling, GA can efficiently explore a wide variety of feasible solutions [9].

The manufacturing system under consideration in this study is a high-mix, low-volume electronic controller production line operating under a no-wait *flow shop* configuration. In practice, the manufacturer simultaneously serves Make-to-Stock (MTS) and Make-to-Order (MTO) demands. A predictive schedule is generated at the beginning of each production shift to maximize the number of MTS products completed within the available production window. However, urgent MTO orders may arrive unexpectedly during production, necessitating the revision of the existing schedule while minimizing disruption to the remaining scheduled jobs and satisfying customer due dates. Although reactive scheduling, no-wait *flow shops*, sequence-dependent setup times, or GA-based optimization have been investigated individually, the integration of these characteristics into a unified predictive-reactive scheduling framework for electronics manufacturing has received limited attention.

In this study, two-stage scheduling is evaluated. The first stage involves constructing a predictive schedule for 20 products to maximize the output. The second stage considers 7 new jobs arriving simultaneously and reschedules the unprocessed jobs together with the new jobs, with the objective of minimizing the *makespan* and lateness of the 7 new jobs. This study contributes to the literature by highlighting a practical and scalable GA-based predictive-reactive scheduling model under multi-objective criteria. This study also demonstrates how intelligent optimization allows manufacturers to maintain high levels of productivity and customer satisfaction in dynamic situations

2. RELATED WORK

Reactive scheduling has received major attention by researchers as manufacturing systems become dynamic, producing the product to stock and receiving orders from other consumers. Györgyi *et al*. [10] addressed the reactive scheduling of jobs with uncertain task durations and maximum time lag constraints motivated by real-time bio-manufacturing scenarios. The authors propose a polynomial-time proactive–reactive method that incrementally reschedules jobs upon their release or completion of uncertain tasks, accounting for all possible realizations of uncertain durations, ensuring robustness. Their robust scheduling algorithm determines feasible job start times under resource and temporal constraints and outperforms a baseline heuristic in minimizing job waiting time. While the study effectively handles resource-constrained scheduling with uncertainty and lag limits, it does not explore hybrid objectives involving both *makespan* minimization and *due date* adherence for dynamically inserted jobs, which will be a key focus in this study using GA.

Qiu *et al*. [11] focused on predictive–reactive scheduling in printed circuit board (PCB) drilling systems that involve parallel batch processors with multiple spindles and sequence-dependent setup times under dynamic conditions. The authors propose a hybrid algorithm called S-AGAIG, which combines a splitting heuristic to form sublots with an adaptive GA enhanced by iterated greedy search to balance spindle use and order tardiness. Additionally, a schedule repair strategy based on sublot co-processing is introduced,

http://dx.doi.org/10.30656/jsmi.v10i1.11538

along with a scheduling stability metric to guide rescheduling decisions. Experimental results across 36 case sets show that S-AGAIG improves average tardiness by 21.27% over competing methods and maintains robustness despite disturbances. Although this study demonstrates the effectiveness of GA-based rescheduling in PCB manufacturing, its primary focus is on reducing tardiness through sublot formation and parallel batch processing. Consequently, the scheduling characteristics differ substantially from those of no-wait *flow shop* environments, where strict inter-machine synchronization and sequence-dependent setup times impose additional operational constraints. Moreover, the study does not consider balancing schedule stability with due-date performance under dynamic job arrivals, leaving room for further investigation.

Otero-Caicedo *et al*. [12] presented a preventive–reactive approach to the nurse scheduling problem (NSP), addressing both labor regulation constraints and nurses' personal preferences while considering unexpected absenteeism. The authors propose a multi-objective linear programming model that proactively assigns daily backup nurses and apply a GA to efficiently solve the NP-hard scheduling problem. To handle real-time disruptions, two reactive rescheduling policies are introduced: one preserves the baseline schedule and the other exclusively relies on backup nurses. Monte Carlo simulations across various settings show that both proposed policies outperform the conventional no preventive approach, particularly when the probability of nurses accepting additional shifts is low. This study focuses on personnel rostering and does not explore reactive rescheduling in manufacturing *flow shops* with hybrid objectives, such as maximizing job completions within the available time window and earliness/lateness.

Su *et al*. [13] addressed the complex problem of reactive multi-objective multi-skilled project scheduling under resource disruptions by incorporating overtime and proactive pre-emption strategies and accounting for the impact of resource skill levels on setup time. The authors developed a reactive scheduling model that minimizes deviations from the baseline in terms of activity start times and resource allocations while also minimizing resource usage costs. To solve the model effectively, a hybrid multi-objective evolutionary algorithm (HMOEA) enhanced by adaptive fitness assignment, hybrid evolution, and local search was introduced. Numerical experiments using five performance metrics confirm the superiority of HMOEA over five state-of-the-art algorithms in terms of convergence, diversity, and distribution. This study is limited to project environments and does not consider reactive scheduling in no-wait *flow shop* systems with hybrid objectives, such as maximizing job completions within the available time window and earliness/lateness penalties, which will be the focus of this proposed approach.

Wenfeng *et al*. [14] studied a digital twin-driven proactive-reactive scheduling for a multi-equipment port system. The authors propose an innovative scheduling framework that leverages a digital twin (DT) to handle the integrated scheduling of multiple port equipment, namely quay cranes, intelligent guided vehicles (IGVs), and yard cranes under a real-world U-shaped port layout and double cycling mode. This framework integrates both proactive (predictive) planning and reactive rescheduling mechanisms, enabling dynamic adjustment to uncertainties such as variability in operation time, equipment failures, and IGV routing conflicts. Through a virtual container port simulation, the authors demonstrate that their DT-driven approach achieves up to a 19.56 **%** reduction in *makespan* and a 3.67 % saving in energy consumption under large-scale conditions compared to nonDT methods. The sensitivity analysis further shows that the energy savings become even more significant than the uncertainty (fluctuation index) increases. This study focuses on port terminal equipment and does not directly address *flow shop* production systems with sequence-dependent setup times, multistage Kuwait constraints, or combined earliness/lateness objectives.

Peng *et al*. [15] present a novel critical chain-based proactive–reactive scheduling approach tailored for the resource-constrained project scheduling problem (RCPSP) under uncertainty. The framework integrates a proactive schedule generated using the critical chain method (CCM) and a reactive rescheduling mechanism using a discrete-time markov decision process (DTMDP). During execution, dynamic adjustments are made using a look-up table inspired by reinforcement learning, which considers the buffer occupation (project and feeding buffers) originally embedded in the baseline schedule. This study emphasizes robustness and reliability by preserving critical chain characteristics while adaptively responding to disruptions. The proposed model is efficient and competitive for small- to medium-sized problem instances, according to computational experiments using benchmark datasets. Although this study focuses on project scheduling, its proactive–reactive duality and learning-based adaptation are highly relevant to *flow shop* production scheduling, especially in managing uncertainties and real-time disruptions. Unlike our *flow shop* context with no-wait and sequence-dependent setup times, this study assumes precedence and resource constraints in a project network, making the operational environment and optimization variables significantly different.

Takeda Berger *et al*. [16] addressed the dynamic nature of modern manufacturing environments, where unplanned events can disrupt predetermined production schedules. A reactive production scheduling approach

that dynamically adjusts job sequencing based on inventory availability is proposed. The core of the method is simulation-based optimization, which evaluates multiple sequencing options in response to the real-time inventory situation. The research is grounded in a real-world automotive component manufacturing case, emphasizing its practical applicability. The results demonstrate that this approach effectively manages stochastic events, enhances disruption responsiveness, and improves production performance across key indicators. While the study focuses on material availability as the main trigger for rescheduling, it shares common ground with reactive scheduling frameworks in other domains, such as *flow shop* scheduling, which drives adaptation to job arrival and machine availability. The main difference with this study lies in the trigger of reactivity (inventory level vs. order arrivals). However, both studies highlight the importance of real-time adaptation mechanisms in modern production systems.

Chakrabortty *et al*. [17] proposed an event-based reactive approach (EBRA) that dynamically adapts schedules when resource disruptions occur to address the resource-constrained project scheduling problem (RCPSP) in uncertain environments. The model integrates a mathematical formulation for reactive scheduling and is validated for large instances using both exact optimization (LINGO) and a metaheuristic enhanced iterated greedy (EnIG) algorithm. Through simulation experiments based on PSPLIB benchmark data, the study demonstrates that compared with conventional reactive methods, EBRA effectively manages single and multiple resource failures, reduces computational complexity, and enhances schedule resilience. However, while this study contributes significantly to project-based scheduling by emphasizing robustness under resource uncertainty, it differs from this study, which focuses on *flow shop* manufacturing systems with sequence-dependent setup times, no-wait constraints, and multi-objective optimization.

Echsler Minguillon and Stricker [18] explore a robust predictive–reactive scheduling strategy for matrix-structured production systems, which are characterized by high flexibility and complex routing. Their research emphasizes the relationship between the robustness of predictive schedules and the adaptability of reactive rescheduling under machine disturbance conditions. The proposed approach aims to sustain high logistic performance despite uncertainties by integrating predictive planning with disruption-responsive adjustments. Experimental analysis shows that enhancing robustness in the predictive phase improves a system's capacity to efficiently recover from disturbances, thereby maintaining stability and minimizing performance degradation. While the study makes a valuable contribution by linking robustness and reactivity in a manufacturing context, it focuses primarily on matrix-structured layouts and robustness analysis rather than detailed operational flow with sequence-dependent setup times.

Joo *et al*. [19] proposed a coordination-based reactive scheduling model for the resource-constrained project scheduling problem (RCPSP) in construction environments with limited on-site storage space. The study is motivated by the increasing adoption of prefabricated building blocks, which introduces new challenges in synchronizing material deliveries and construction activities. The authors develop a reactive scheduling procedure that revises project timelines after disruptions by coordinating delivery schedules from suppliers while considering storage capacity limitations. The model demonstrates its ability to minimize project duration and effectively manage material flow during unexpected events through computational experiments. The authors also highlight that the proposed coordination mechanism is extendable to engineer-to-order (ETO) and MTO manufacturing systems with constrained storage capacity. Although this study offers valuable insights into coordination-driven reactive scheduling, its scope is limited to project environments with distinct task dependencies and material logistics constraints.

Takeda Berger *et al*. [20] introduce a conceptual data-driven predictive–reactive scheduling model that adjusts production schedules based on real-time inventory data using machine learning and simulation-based optimization. The approach captures the dynamic states of raw materials, work-in-process, and finished goods inventories to anticipate and respond to disruptions, enhancing flexibility and decision-making in job-shop environments. While their work demonstrates how data integration supports predictive and reactive control in production systems, it mainly emphasizes information-driven responsiveness rather than operational sequence optimization. The current research differs by addressing reactive scheduling through an algorithmic optimization perspective, where sequencing logic and timing decisions are optimized under dynamic order arrivals, rather than driven by inventory state analysis.

Panda and Ramteke [21] developed a structure-adapted GA for reactive scheduling of crude oil operations under multiple uncertainties, such as tank unavailability and routing conflicts. Their model is applied to a marine-access refinery context, where continuous crude oil flows must be managed to minimize delays and maintain operational efficiency. They demonstrated that the adapted GA effectively handles uncertain system states and provides better responsiveness than static scheduling methods. However, their focus is on crude oil supply chain and pipeline scheduling, whereas the present research targets a no-wait *flow shop* manufacturing system with sequence-dependent setup times and dynamic new job arrivals, using a GA to handle multi-objective

http://dx.doi.org/10.30656/jsmi.v10i1.11538

Table 1. State of the art of this study

Author(s)	System type/domain	Uncertainty /disturbance	Scheduling Type	Objective(s)	Method / Algorithm	No-wait / SDST	GA-based optimization
Györgyi *et al*. [10]	Bio-manufacturing	Task duration uncertainty	Proactive–Reactive	Min waiting time	Polynomial-time algorithm	x	x
Qiu *et al*. [11]	PCB drilling system	Machine and lot disturbances	Predictive –Reactive	Min tardiness	S-AGAIG (GA + Iterated Greedy)	x	√
Otero-Caicedo *et al*. [12]	Nurse scheduling	Absenteeism	Preventive –Reactive	Min tardiness and workload	GA + Monte Carlo simulation	x	√
Su *et al*. [13]	Multiskilled project scheduling	Resource disruption	Reactive	Min deviation and resource cost	HMOEA	x	x
Wenfeng *et al*. [14]	Port multi-equipment	Operation and routing uncertainty	Proactive–Reactive	Min *makespan* and energy use	Digital Twin + Simulation	x	x
Peng *et al*. [15]	Project scheduling	Duration and resource uncertainty	Proactive–Reactive	Min deviation, robustness	DTMDP + Critical Chain	x	x
Takeda-Berger *et al*. [16]	Automotive manufacturing	Inventory fluctuation	Reactive	Improve responsiveness	Simulation-based Optimization	x	x
Chakrabortty *et al*. [17]	Project scheduling	Resource reliability	Reactive	Min deviation and complexity	Event-based Reactivity + EnIG	x	x
Echsler Minguillon and Stricker [18]	Matrix production	Machine disturbances	Predictive –Reactive	Improve robustness	Robust scheduling	x	x
Joo *et al*. [19]	Construction (RCPSP)	Delivery delay	Reactive	Min project duration	Coordination-based procedure	x	x
Takeda Berger *et al*. [20]	Job-shop manufacturing	Inventory uncertainty	Predictive –Reactive	Improve flexibility	ML + simulation-based optimization	x	x
Panda and Ramteke [21]	The crude oil supply chain	Tank unavailability and routing	Reactive	Min delay	Structure-adapted GA	x	√
Sousa *et al*. [22]	Smart grid energy system	Demand fluctuation	Reactive	Min cost, loss, and voltage dev	Multi-objective Optimization	x	x
Rahmani and Ramezanian [23]	Flexible *flow shop*	Machine breakdown	Reactive	Min tardiness, max stability	Metaheuristic Optimization	√	x
Li *et al*. [24]	Distributed assembly *flow shop*	Order acceptance	Predictive –Reactive	Multi-objective (efficiency and revenue)	Knowledge-driven MOEA	√	x
Khare and Agrawal [25]	Hybrid *flow shop*	Sequence-dependent setup	Predictive	Min weighted earliness and tardiness	Hybrid GA	√	√
This study	No-wait *flow shop* (electronics manufacturing)	Dynamic Job Arrivals	Predictive -Reactive	Min makespan, makespan deviation, and lateness	Genetic algorithm (GA)	√	√

scheduling. Thus, the operational context, routing constraints, objective functions, and scheduling environment are significantly different despite both studies using GA for reactive scheduling under uncertainty.

Sousa *et al*. [22] presented a multi-objective optimization framework that addresses the scheduling of active and reactive resources within a smart grid distribution system. Their approach integrates operational and energy management considerations to coordinate distributed energy resources, including renewable generation and storage units. The problem is described as a multi-objective optimization model to minimize power losses, operational costs and voltage deviations using an intelligent scheduling mechanism that balances active and reactive power flow. This coordinated scheduling strategy enhances overall grid stability and efficiency while maintaining the quality of power delivery. Computational experiments have shown that the proposed approach provides effective trade-offs between cost efficiency and system reliability in smart grid operations. While this paper contributes valuable insights into optimization under uncertainty and resource coordination, it differs from the present research, which applies GA to manufacturing scheduling problems focusing on job sequencing, maximizing job completions within the available time window and *due date* alignment under dynamic order arrivals rather than energy distribution management.

Rahmani and Ramezanian [23] proposed a stable reactive scheduling approach to handle unexpected disruptions in dynamic flexible flow shop environments. Their research focuses on the maintenance of schedule stability while adapting to real-time changes, such as machine breakdowns or job delays, which are common in modern manufacturing systems. The proposed method uses a two-stage process: first, an initial predictive schedule is generated using a metaheuristic optimization model, and second, reactive adjustments are applied to restore feasibility and minimize deviations from the original plan. The results of the case study show that the proposed approach effectively balances system responsiveness and stability, thereby reducing total tardiness and maintaining efficient machine use. Stability, often overlooked in reactive scheduling, plays a crucial role in maintaining operational consistency under disruptions. Compared with this study, Rahmani and Ramezanian's work focuses on reactive stability and disruption recovery in flexible flow shops, whereas the current research prioritizes dynamic order insertion in a no-wait flow shop with sequence-dependent setup times, optimizing both output and earliness/lateness through a GA-based approach.

Li *et al*. [24] studied a distributed assembly blocking *flow shop* with order acceptance decisions, formulating a setting in which jobs must be scheduled across distributed assembly lines subject to blocking constraints while simultaneously deciding which orders to accept. They developed a knowledge-driven multiobjective algorithm that exploits problem structure to guide search and balance competing goals typical of order-acceptance scheduling (e.g., production efficiency vs. revenue/penalty trade-offs). The proposed approach targets the combined sequencing and acceptance problem under assembly coupling and blocking and reports performance gains over comparative methods on benchmarked experiments, demonstrating its effectiveness for large instances in distributed assembly environments. However, while their study focuses on incorporating order acceptance into distributed *flow sho*ps, the current research deals with dynamic job arrivals in a single no-wait *flow shop* setting, emphasizing reactive rescheduling and multi-objective optimization for maximizing job completions within the available time window and *due date* alignment rather than acceptance decision-making.

Khare and Agrawal [25] investigated a hybrid *flow shop* scheduling problem incorporating sequence-dependent setup times and due windows to minimize the total weighted earliness and tardiness of jobs. This study models a realistic manufacturing environment where setup times depend on job sequences, and due windows are introduced to represent customer delivery flexibility. A mixed-integer linear programming model is formulated, and a metaheuristic approach based on a hybrid GA is developed to efficiently handle large-scale problem instances. The proposed algorithm achieves superior performance in balancing early and late deliveries compared with conventional heuristics, according to computational experiments, providing practical insights for manufacturing systems with strict delivery commitments. The key contribution lies in the integration of due window constraints and weighted penalties within a hybrid *flow shop* context. The authors successfully incorporate sequence-dependent setup times and due-window considerations into a hybrid *flowshop* model, demonstrating the importance of delivery-related objectives in scheduling. Nevertheless, their study assumes a static scheduling environment and therefore does not address the rescheduling challenges arising from orders arriving dynamically. The interaction between predictive schedules and subsequent reactive adjustments remains unexplored. Based on the literature survey, a state-of-the-art table can be developed, as shown in Table 1.

3. RESEARCH METHODS

3.1. System description

System modelling and analysis must begin with a clear and structured system description. This step helps define the boundary, elements, constraints, and flow of a system to be analyzed [26], [27], [28], [29]. This study considers a no-wait *flow shop* scheduling system

http://dx.doi.org/10.30656/jsmi.v10i1.11538

in a manufacturing environment that produces electronic circuits. The production line consists of five sequential machines where each product must be processed in the same order across all machines without waiting time between machines. This no-wait flow shop characteristic imposes tighter constraints than classical flowshop because each job must proceed directly from one machine to the next without idle time.

Each product goes through 5 machines in sequence: Surface Mount Technology (M_1), manual assembly (M_2), firmware uploading (M_3), quality inspection (M_4), and packaging (M_5). Initially, the system schedules 20 different products labelled P1 to P20. Each product may require different component setups, testing configurations, or calibration routines, implying that the transition from one product to another involves distinct setup operations, which is known as sequence-dependent setup time (SDST). For instance, switching from a product that uses through-hole components to one with surface-mount technology requires cleaning, retooling, and testing adjustments, all of which contribute to additional time costs. Furthermore, each product has a defined release time, which represents when it becomes available for processing. The objective of the initial scheduling phase is to maximize the output of these 20 products before new orders are placed.

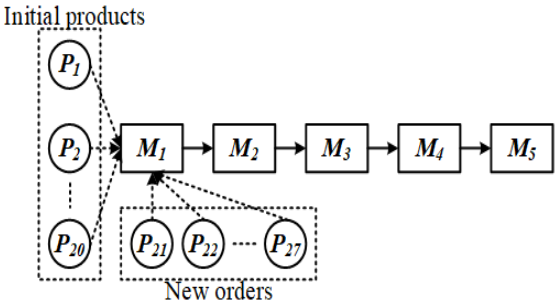


Fig. 1. No-wait *flow shop* environment

The manufacturing system under consideration is highly dynamic, where sudden orders from key clients or urgent replacements for industrial systems can arrive at unpredictable times. In this case study, the arrival of seven new jobs (P21 to P27) represents such disturbances that require immediate integration into the existing production schedule to maintain responsiveness and customer satisfaction. These new orders, often with strict *debates*, can disrupt the original plan by competing for limited machine capacity and setup resources. Therefore, a reactive scheduling mechanism is required to adjust start times and job sequences without causing excessive delays or idle periods for previously scheduled products. Revising the predictive schedule allows the system to adapt dynamically, ensuring minimal deviation from the original plan while accommodating new demands. This predictive–reactive balance is essential for modern electronics manufacturers to maintain operational agility, reduce delivery penalties, and sustain long-term client trust in a volatile, demand-driven market. The system environment is illustrated in Fig. 1.

Each of the 20 initial electronic products must undergo a set of preliminary processes that influence their respective release or ready times before entering the production shop floor. These preliminary steps include the following: (1) PCB fabrication, where raw copper-clad boards are processed into circuit-ready PCBs; (2) component picking, in which required components are collected from storage based on the bill of materials; (3) incoming quality control (IQC), where the picked components are checked for compliance and defects; (4) kitting tray preparation, where components are organized into trays to support efficient downstream assembly; and (5) pre-test and validation, which applies to critical or complex designs requiring early-stage functional testing. Each of these processes has its own lead time and is subject to resource availability and line delays, resulting in varying release times for the 20 jobs. This variability must be considered because it directly impacts the sequencing and scheduling results.

3.2. Mathematical formulation

Existing mathematical formulations for scheduling cannot be directly applied because the scheduling environment considered in this study combines several practical characteristics that have rarely been simultaneously modelled, including no-wait *flow shop* production, sequence-dependent setup times, varying release times, dynamic job arrivals, and a two-stage predictive-reactive scheduling strategy. Therefore, a new mathematical formulation is developed to accurately represent the proposed scheduling problem and provide the basis for the subsequent optimization process using the GA. The following assumptions are established to define the scope of the problem before presenting the mathematical formulation. The mathematical model will be developed based on Mixed Integer Linear Programming (MILP).

Assumptions:

a. Each product must be processed on all five machines in the same order (M_1 to M_5) with no waiting time between machines.
b. The processing times of each job on each machine are known and deterministic.
c. Setup times between jobs on each machine depend on the job sequence and are nonzero.
d. Only one product is processed on each machine at any given time (no parallel processing).
e. The release times for each job are known and may differ, including for initial and dynamically arriving jobs.

f. The system does not allow job pre-emption; once a job starts processing on a machine, it must be completed before the next job begins.
g. The shop operates continuously without planned downtime.
h. The GA optimization process does not guarantee a global optimum, but it should find high-quality solutions that are close to or equal to the best.

Sets:

j : 1, 2, …, 27 set of products/jobs
m : 1, 2, …, 5 be set of machines
o : 1, 2, …, 20 set of initial jobs
n : 1, 2, …, 7 set of new jobs
s : 1, 2 be the set scheduling stage

Parameters:

p_j^m : processing time of job *j* on machine *m*
s_{ab}^m : setup time from job *a* to job *b* on machine *m*
r_j : release time of job *j*
d_j : *due date* of job *j* (only for new jobs)
M : a large constant number for scheduling models
G : a large constant number for chromosome fitness

Decision variables:

$x_{ab} \in \{0,1\}$: 1 if job *a* precedes job *b*
s_j^m : starting time of job *j* on machine *m*
c_j : completion time of a job
c_{max}^1 : scheduling *makespan* of stage 1
c_{max}^2 : scheduling *makespan* of stage 2
e_j : earliness of job *j*
l_j : lateness of job *j*
fit_k : fitness function of chromosome-*k*
nj : number of jobs that have been processed before new jobs arrive

Objective function:

Stage 1:

$$Max\ Z_1 = nj \quad (1)$$

Stage 2:

$$Min\ Z_2 = \alpha \cdot (c_{max}^2 - c_{max}^1) + \sum_{j \in n} \beta \cdot l_j \quad (2)$$

Constraints:

$$s_j^{m+1} = s_j^m + p_j^m \quad (3)$$

$$s_b^m \geq s_a^m + p_a^m + s_{ab}^m - M(1 - x_{ab});\ \forall m \in M; \forall ab, a \neq b$$

$$s_a^m \geq s_b^m + p_b^m + s_{ba}^m - Mx_{ab};\ \forall m \in M; \forall ab, a \neq b \quad (4)$$

$$x_{ab} + x_{ba} = 1;\ \forall a, b, a \neq b$$

$$c_{max} = s_j^5 + p_j^5 \quad (5)$$

$$s_j^1 \geq r_j \quad (6)$$

$$nj = arg\ min_{i=1}^{n} |b - c_j| \quad (7)$$

$$c_j = s_j^5 + p_j^5;\ \forall j, j \in n \quad (8)$$

$$e_j \geq d_j - c_j;\ \forall j, j \in n \quad (9)$$

$$l_j \geq c_j - d_j;\ \forall j, j \in n \quad (10)$$

There will be 2-stages of scheduling. Stage 1 aims to maximize the number of jobs that have been processed, while stage 2 aims to minimize the *makespan* for jobs that have not been processed yet from the results of stage 1 scheduling and minimize lateness for newly arrived jobs to avoid consumer dissatisfaction. The combination of objective functions for scheduling at stage 2 is shown by Equation (2), with the weights for minimizing *makespan* and minimizing lateness will be α = 0.3 and β = 0.7, respectively. The objective function weights were selected according to the manufacturer's production policy. Since newly arriving MTO jobs directly affect customer delivery performance, minimizing their lateness is prioritized over minimizing the MTS jobs' makespan deviation, which mainly affects internal operational efficiency.

In no-wait *flow shop*, the starting time of a job in the next machine equals the starting time of the job in a machine plus its processing time, as shown in Equation (3). Equation (4) ensures that each machine will process only one job at a time, preventing any overlap in the processing of multiple jobs on a machine.

The *makespan* of the scheduling result, which is the sum of the starting time of the last job on the last machine and its processing time, is denoted by Equation (5). All jobs will be scheduled after they are available for processing, as indicated by Equation (6). The number of processed jobs is expressed by Equation (7), and the completion time of each job is delineated in Equation (8) by summing the starting time of the job on the last machine and its processing time. In Equation (9), the term earliness is defined as the difference between a job's due date and completion time. The lateness of a job is represented by Equation (10), which is calculated by subtracting the job completion time from its *due date*.

3.3. GA development

GA is a general-purpose metaheuristic whose performance depends primarily on the representation, fitness function, and genetic operators tailored to the optimization problem's characteristics rather than on adopting the latest GA variant. Therefore, this study employs a classical yet robust GA framework with a problem-specific chromosome representation, repair mechanism in the crossover, and insertion mutation specifically designed for the problem, which constitutes the methodological contribution of this study.

In this case, the GA must handle both the initial scheduling of jobs with varying release times and the dynamic insertion of new orders. The algorithm iteratively evolves a population of candidate solutions (chromosomes) through selection, crossover, and mutation operators guided by a fitness function that reflects the scheduling objectives.

http://dx.doi.org/10.30656/jsmi.v10i1.11538

3.3.1. *Chromosome representation*

In GA, the solution to be found must be represented in the form of a chromosome. Here, the solution to be found is the job sequence. To effectively model the job sequences, this study adopts a combinatorial representation for chromosomes in both Stage 1 (initial scheduling of 20 jobs) and Stage 2 (rescheduling of unfinished jobs combined with 7 new orders). In this representation, a chromosome is encoded as a permutation of job indices, where each job's position in the list determines its processing order across all machines. This representation is suitable for no-wait *flowshop* problems because it directly captures job precedence without the need for time-based encoding. For Stage 1, the chromosome consists of a permutation of the initial 20 jobs. In Stage 2, the chromosome includes only the unfinished jobs from Stage 1 and the new arrived 7 jobs. The GA chromosome for Stage 1 is depicted in Fig. 2. Stage 2 uses a similar chromosome design; however, the chromosomes only represent unprocessed jobs and newly arrived jobs.

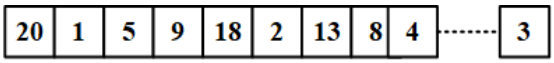


Fig. 2. The chromosome represents the job sequence

3.3.2. *Fitness function*

GA is blind; it does not know what problem it is attempting to resolve. The only information used by GA to find a solution to the problem is the fitness value calculated using the fitness function. The objective of scheduling in this study is to minimize *makespan* while minimizing the earliness or lateness of the new jobs. In finding a solution, GA will maintain fit chromosomes from generation to generation. This is identical to the maximization function and contrary to the minimization function in this case study. Thus, the fitness function used in this study is shown in Equation (10).

$$fit_k = G - Z_s \quad (10)$$

3.3.3. *Crossover mechanism*

In permutation-based optimization a 1-cut-point crossover operator is widely used because of its simplicity and ability to exchange adjacent genetic material between parents. However, this crossover often produces illegal children containing duplicate jobs and missing jobs when applied to permutation chromosomes. Therefore, a repair strategy is typically required to restore feasibility by identifying duplicates, detecting the missing elements, and replacing them while preserving as much of the inherited order as possible. Recent studies have emphasized the importance of such repair mechanisms: Hvattum [30] demonstrated that naive order-based crossover can introduce structural defects in permutation encodings and proposed corrective adjustments to enhance offspring quality. Cicirello [31] highlighted that to prevent invalid permutations, crossover operators for permutations must explicitly handle feasibility or employ post-processing repairs.

More recent work by Ciepliński and Golak [32] further supports the need for controlled recombination and repair to avoid destructive disruptions in task sequences. These findings justify the integration of a repair mechanism in permutation-based optimization when applying 1-cut-point crossover. Therefore, in this study, 1-cut-point crossover is implemented with the mechanism shown in Fig. 3.

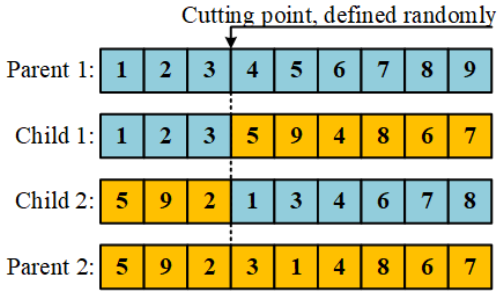


Fig. 3. The crossover mechanism to produce legal children

3.3.4. *Mutation mechanism*

In this case study, insertion mutation is one of the most effective mutation operators because it preserves most of the job sequence structure while introducing meaningful diversity into the population. The operator removes one job from its current position and inserts it into another chromosome location, resulting in a small yet impactful perturbation that respects permutation feasibility without requiring repair. Recent studies have confirmed that insertion mutations improve convergence stability and solution quality in optimization settings.

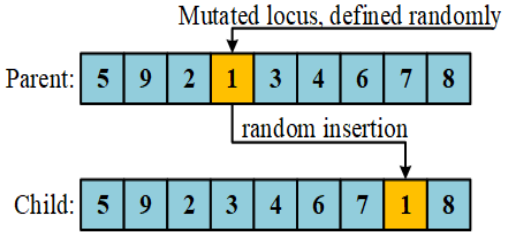


Fig. 4. Insertion mutation mechanism implemented

For example, Ciepliński and Golak [32] demonstrated that insertion-style permutation perturbation maintains structural consistency and enhances exploration when combined with robust crossover schemes in sequencing algorithms. Similarly, Li *et al.* [24] applied insertion-based neighborhood moves within a multi-objective metaheuristic to effectively optimize complex distributed assembly *flow shops*, demonstrating competitive performance on

large-scale scheduling instances. These findings support the suitability of insertion mutation for this study, where maintaining a valid job sequence while requiring flexible adjustments under dynamic order arrivals is essential. These findings indicate that insertion mutation is a suitable and robust choice for the patient in this study. Fig. 4 shows the mechanism of the insertion mutation implemented in the proposed GA.

The proposed GA procedure for the two-stage PRS problem is summarized in the following pseudo-code.

```
Stage 1: Predictive Scheduling
Generate initial population
   Each chromosome is a permutation of jobs
   in Stage 1
        For each chromosome in the population:
           Decode chromosome into a no-wait
           flowshop schedule
           Compute completion time of all jobs
           Count number of jobs completed
           before predetermined time
           Evaluate fitness of each chromosome
For iteration = 1 to gen do:
   Select parent chromosomes from the
   population
   Apply 1-cut-point crossover
   Repair offspring if duplicate or missing
   jobs occur
   Apply insertion mutation
   Decode the chromosome into no-wait
   schedule
   Evaluate chromosome fitness
   Form the next generation using selected
   parents and offspring
   Select the best chromosome Stage 1
Obtain Stage 1 best sequence
Determine completed jobs before the
predetermined time
Determine unprocessed jobs after the
predetermined time
Record Stage 1 baseline makespan

Stage 2: Reactive Scheduling
Define reactive job set:
Let JR = unprocessed jobs from Stage 1
Generate initial population
   Each chromosome is a permutation of JR
        For each chromosome in Stage 2:
           Decode each chromosome into a no-
           wait flowshop schedule
           Compute completion time of all jobs
           in JR
           Compute Stage 2 makespan
           Compute makespan deviation:
           makespan of Stage 2 – makespan of
           Stage 2
           Compute lateness of each new job
           Compute fitness the chromosome
           according to the objective function
For iteration = 1 to gen do:
   Select parent chromosomes from the
   population
   Apply 1-cut-point crossover
   Repair offspring if duplicate or missing
   jobs occur
   Apply insertion mutation
   Decode the chromosome into no-wait
   schedule
   Compute makespan deviation and lateness of
   new jobs
   Evaluate chromosome fitness
   Form the next generation using selected
   parents and offspring
   Select the best chromosome
Obtain Stage 2 job sequence
```

4. RESULTS AND DISCUSSION

The collected data in this case study are SDST, processing, and release time of the initial jobs in each machine, as shown in Tables 1, Table 2, Table 3, Table 4, Table 5, and Table 6.

Table 1. SDST of initial jobs in M_1

Job	Initial Setup	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	P11	P12	P13	P14	P15	P16	P17	P18	P19	P20
P1	15	0	15	15	9	15	14	12	14	13	15	9	5	10	8	10	15	15	12	11	8
P2	8	6	0	15	12	9	7	5	14	10	11	8	5	7	6	7	6	11	6	11	11
P3	10	7	10	0	15	12	15	15	13	14	13	10	8	15	10	6	13	5	13	8	11
P4	8	10	8	8	0	13	8	10	6	11	15	10	9	7	8	15	6	11	10	15	9
P5	12	5	12	12	13	0	15	5	15	13	15	8	12	15	9	15	6	8	9	15	8
P6	12	10	12	14	12	12	0	5	15	5	15	6	10	14	9	15	5	9	11	15	11
P7	11	13	11	14	12	5	15	0	7	13	15	13	8	14	13	11	8	6	7	6	5
P8	5	15	5	8	15	14	11	11	0	6	9	7	5	6	6	7	14	14	12	13	10
P9	11	14	11	12	5	7	8	7	9	0	13	8	11	12	8	11	13	8	10	12	10
P10	13	5	13	7	13	12	11	15	11	9	0	11	14	10	10	8	15	11	5	11	9
P11	10	6	10	14	7	7	15	8	8	7	11	0	9	9	5	15	14	6	9	10	8
P12	5	15	5	7	14	12	10	13	6	10	9	12	0	6	10	8	13	12	6	10	10
P13	7	9	7	10	5	7	8	12	15	8	11	14	15	0	8	8	9	7	7	15	13
P14	15	9	15	7	15	9	8	9	11	7	10	13	6	11	0	10	8	10	10	13	11
P15	15	12	15	12	7	10	6	14	5	6	11	9	8	15	7	0	14	10	6	5	6
P16	12	12	12	5	6	14	11	15	6	11	15	11	7	14	14	5	0	12	13	15	12
P17	5	7	5	14	15	13	14	8	13	9	11	7	8	6	10	12	6	0	5	7	12
P18	14	10	14	12	15	9	12	9	9	15	5	6	14	7	13	12	13	13	0	12	14
P19	15	11	15	6	5	8	7	6	8	8	8	15	5	9	14	10	10	9	15	0	10
P20	7	12	7	9	9	7	7	11	11	10	7	8	11	14	15	12	14	10	7	15	0

http://dx.doi.org/10.30656/jsmi.v10i1.11538

Table 2. SDST of initial jobs in M_2

Job	Initial Setup	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	P11	P12	P13	P14	P15	P16	P17	P18	P19	P20
P1	12	0	5	12	6	15	14	8	6	10	11	13	7	7	7	13	11	11	5	10	12
P2	15	9	0	15	7	6	14	6	15	13	14	9	15	12	7	7	7	13	11	11	5
P3	5	8	7	0	6	14	13	13	7	8	8	10	14	6	11	12	6	11	10	9	10
P4	8	12	11	8	0	15	12	9	9	11	5	9	6	5	12	7	8	5	14	12	7
P5	14	12	10	14	6	0	10	6	6	12	8	12	12	14	9	9	15	5	6	9	10
P6	9	15	13	9	15	8	0	5	14	7	8	6	10	7	14	10	9	7	15	9	11
P7	9	8	7	9	5	9	8	0	7	13	8	12	13	14	13	9	14	7	9	15	10
P8	14	13	9	14	6	7	5	12	0	13	15	13	6	14	7	12	7	6	7	7	15
P9	12	7	7	12	6	13	10	13	13	0	13	11	5	6	11	15	13	14	10	14	15
P10	11	8	11	11	10	8	13	12	15	5	0	7	7	6	7	8	14	11	15	10	13
P11	6	8	5	6	11	13	11	15	6	11	12	0	5	10	12	9	12	7	7	11	12
P12	7	5	6	7	11	10	10	8	5	12	10	10	0	15	10	7	12	7	10	10	6
P13	14	6	7	14	13	15	5	10	7	5	7	7	7	0	14	5	9	9	12	5	10
P14	11	11	11	11	6	10	14	10	15	6	8	8	14	11	0	8	11	14	12	5	8
P15	10	14	9	10	5	8	5	13	12	6	6	8	14	11	5	0	5	9	13	5	6
P16	9	15	13	9	15	13	7	9	8	6	5	11	8	5	5	8	0	15	14	9	5
P17	5	12	8	5	11	7	10	7	10	12	15	10	11	5	14	5	7	0	11	9	13
P18	9	12	7	9	9	14	8	11	5	6	10	8	10	5	5	8	7	15	0	8	13
P19	10	12	9	10	13	6	10	13	13	15	12	15	5	10	15	10	9	12	10	0	10
P20	6	8	9	6	9	11	8	9	5	12	5	15	12	7	12	7	11	7	5	6	0

Table 3. SDST of initial jobs in M_3

Job	Initial Setup	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	P11	P12	P13	P14	P15	P16	P17	P18	P19	P20
P1	13	0	11	13	5	13	6	8	6	13	5	13	5	14	12	11	6	6	13	6	7
P2	14	11	0	14	11	15	13	15	9	7	7	7	11	14	11	7	7	12	9	14	6
P3	9	9	15	0	5	14	6	6	6	11	15	15	12	11	6	11	15	14	15	12	14
P4	15	12	10	15	0	11	8	12	12	11	7	15	12	15	12	10	12	9	15	11	10
P5	10	15	15	10	14	0	15	15	14	15	10	8	9	6	9	11	8	11	5	9	9
P6	12	13	14	12	12	7	0	14	15	13	11	6	9	5	10	7	11	13	9	5	15
P7	9	7	6	9	10	5	9	0	7	9	10	5	6	5	10	15	11	15	14	13	6
P8	7	9	8	7	10	13	9	13	0	6	15	14	9	10	10	15	15	15	11	11	14
P9	14	12	5	14	13	14	14	5	12	0	11	14	14	8	10	10	10	5	11	7	6
P10	12	7	12	12	6	15	6	9	9	12	0	8	9	9	6	5	7	10	8	5	6
P11	14	14	7	14	7	8	12	9	9	7	6	0	12	14	6	12	8	8	9	8	14
P12	10	10	11	10	14	12	5	5	15	12	12	11	0	7	12	8	11	13	13	12	7
P13	14	10	15	14	9	5	14	14	14	7	6	8	11	0	5	12	12	9	9	6	6
P14	8	7	11	8	6	15	12	10	5	15	5	13	9	5	0	7	13	9	5	12	12
P15	7	15	14	7	14	7	12	8	13	13	6	7	9	15	12	0	9	11	13	7	12
P16	10	6	14	10	5	11	14	9	14	10	5	15	8	9	15	8	0	6	14	12	5
P17	13	8	10	13	5	6	13	10	12	15	7	10	7	12	10	8	10	0	14	11	12
P18	11	12	13	11	12	5	7	15	15	12	9	6	8	11	14	9	13	12	0	13	15
P19	9	13	9	9	15	13	15	5	11	9	15	13	12	7	11	8	8	6	8	0	11
P20	10	15	12	10	12	5	14	8	9	5	9	13	7	6	9	5	13	5	6	14	0

Table 4. SDST of initial jobs in M_4

Job	Initial Setup	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	P11	P12	P13	P14	P15	P16	P17	P18	P19	P20
P1	13	0	10	13	5	13	10	13	15	13	15	9	12	5	12	12	10	5	9	15	5
P2	9	9	0	9	9	8	7	8	10	10	7	7	11	11	15	10	15	5	9	5	12
P3	15	7	7	0	13	12	15	14	6	9	10	12	9	9	11	14	11	14	7	12	11
P4	10	8	8	10	0	11	8	15	8	10	11	11	11	12	14	11	10	15	7	10	11
P5	11	11	6	11	14	0	15	5	5	11	9	7	13	8	15	7	13	5	6	12	10
P6	6	12	11	6	14	11	0	11	13	8	14	11	15	6	12	15	11	10	13	8	5
P7	9	6	13	9	6	12	12	0	6	13	14	6	8	8	12	11	6	11	13	11	10
P8	11	5	10	11	12	5	9	10	0	12	6	7	5	7	6	13	5	8	11	10	8
P9	6	14	15	6	10	7	6	9	7	0	14	10	9	12	12	15	7	11	11	5	10
P10	11	14	5	11	11	11	9	11	12	15	0	14	9	15	14	10	15	9	11	5	12
P11	7	11	8	7	12	10	13	15	5	10	5	0	14	13	9	6	14	15	8	12	7
P12	7	11	15	7	11	9	8	14	13	15	14	8	0	13	13	13	13	12	13	11	11
P13	10	15	8	10	9	8	11	11	5	12	10	7	6	0	6	6	13	14	14	14	7
P14	10	15	7	10	9	14	7	5	11	9	15	15	6	8	0	12	9	9	6	11	11
P15	8	15	7	8	9	11	13	8	9	9	11	13	7	8	8	0	5	14	13	12	8
P16	15	11	13	15	9	8	8	9	8	12	9	6	12	12	12	8	0	11	8	5	14
P17	9	5	6	9	7	10	10	8	11	12	12	9	11	9	15	10	15	0	11	5	7
P18	9	14	7	9	6	15	15	13	12	7	8	9	5	6	8	10	9	14	0	8	15
P19	5	9	5	5	15	13	12	14	5	13	8	9	13	5	5	12	15	9	13	0	10
P20	12	14	13	12	5	13	7	13	10	9	11	5	13	6	9	9	6	15	6	12	0

Table 5. SDST of initial jobs in M_5

Job	Initial Setup	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	P11	P12	P13	P14	P15	P16	P17	P18	P19	P20
P1	7	0	14	7	7	6	12	9	14	9	8	9	6	12	15	15	8	10	13	9	10
P2	9	15	0	9	7	10	15	7	6	6	6	5	9	10	9	7	13	12	9	8	12
P3	13	14	14	0	9	14	9	6	8	12	11	14	6	15	13	14	13	10	11	12	14
P4	10	14	11	10	0	11	13	14	11	12	7	9	10	9	15	12	9	15	10	9	9
P5	15	13	9	15	14	0	11	7	5	12	8	14	7	10	14	12	11	13	14	10	7
P6	13	15	12	13	14	5	0	9	8	9	8	13	11	13	10	5	12	15	14	15	6
P7	12	9	10	12	5	9	9	0	10	14	10	8	9	15	15	13	8	12	14	9	7
P8	10	5	12	10	6	14	8	10	0	13	7	7	14	14	9	8	9	5	11	15	9
P9	6	13	7	6	8	9	12	5	15	0	11	5	9	12	8	10	5	10	15	8	14
P10	8	10	11	8	15	15	15	11	10	8	0	6	12	13	11	8	10	10	8	8	14
P11	12	7	14	12	12	5	10	8	5	6	7	0	14	12	15	5	8	9	7	14	15
P12	9	11	5	9	15	9	10	14	11	15	10	7	0	7	11	12	10	5	12	9	7
P13	14	7	15	14	14	13	11	13	14	14	15	5	12	0	5	15	13	8	13	6	11
P14	12	9	6	12	12	5	5	6	15	5	6	6	9	6	0	15	8	11	6	8	11
P15	13	11	13	13	14	8	9	6	10	13	14	9	13	15	10	0	9	5	12	8	12
P16	11	13	5	11	8	11	12	6	9	5	11	14	15	11	10	9	0	6	6	7	11
P17	6	10	12	6	7	7	7	5	13	6	15	12	9	5	8	10	11	0	13	12	13
P18	10	15	13	10	6	10	14	10	5	12	9	8	8	7	10	15	10	12	0	9	6
P19	6	13	14	6	15	7	13	6	6	7	7	9	7	14	9	15	12	11	14	0	11
P20	14	10	12	14	10	13	6	11	8	8	15	7	7	12	8	6	13	13	13	12	0

http://dx.doi.org/10.30656/jsmi.v10i1.11538

The manufacturing system operates under a fixed internal policy in which new incoming jobs are scheduled starting at 1:00 PM; therefore, only a time window of 300 minutes is available to process as many initial jobs as possible before transitioning to the reactive phase. In stage 1 scheduling, the collected data are used to construct the predictive schedule for the 20.

Table 6. Processing and release times of the initial jobs in each machine

Job	M1	M2	M3	M4	M5	Release Time
P1	13	10	6	18	8	13
P2	20	16	7	15	8	6
P3	20	17	6	17	7	14
P4	12	13	11	16	6	27
P5	20	22	6	12	7	7
P6	18	12	11	9	11	10
P7	16	17	11	18	6	20
P8	18	12	14	12	8	10
P9	17	15	10	7	5	0
P10	20	25	6	16	9	22
P11	13	20	8	6	12	20
P12	9	16	10	15	8	23
P13	13	13	8	9	7	26
P14	12	11	8	13	6	30
P15	14	24	7	14	6	1
P16	19	17	10	17	12	29
P17	20	19	14	18	5	23
P18	16	14	15	17	8	28
P19	16	10	12	12	8	10
P20	11	13	15	10	12	19

initial jobs, with the primary objective being to maximize the number of jobs completed within the time window. A sensitivity analysis was performed to evaluate the influence of the GA parameters on the scheduling results before conducting the Stage 1 scheduling experiment. Four parameters were examined: population size (20, 30, and 40), crossover rate (0.4, 0.5, and 0.6), mutation rate (0.05, 0.10, and 0.15), and number of generations (300, 500, and 700). The GA consistently generated the same job sequence and objective value across all tested parameter combinations, indicating that the proposed optimization framework is relatively insensitive to moderate parameter variations. Therefore, the parameter setting of population size (*pop_size*) = 30, crossover rate (*c_rate*) = 0.5, mutation rate (*m_rate*) = 0.1, and 500 generations (*gen*) was adopted for the subsequent experiments because it provides stable solutions while maintaining reasonable computational effort.

Under these parameters, the GA generated the job sequence 8-16-7-18-12-3-14-20-9-19-13-5-2-11-1-10-15-4-6-17, which can complete 11 jobs within the time window. The effectiveness of the proposed scheduling framework was validated by comparing the Stage 1 result with the current scheduling practice implemented by the manufacturer. Under the existing scheduling approach, only 8 jobs can be completed before the production cut-off of 1:00 PM. This represents a 37.5% improvement in the number of completed jobs without requiring any additional production resources or extending the available processing time.

Although *makespan* is not the main optimization criterion in this stage, the resulting finishing time of the last job, which is 581-time units, is still recorded as the baseline predictive *makespan*. Later, it will be used in stage 2 to evaluate and minimize *makespan* deviation when new jobs arrive dynamically. This baseline schedule provides a stable foundation for the subsequent reactive scheduling phase, where both deviation from the initial plan and *new job adherence* must be optimized simultaneously.

Table 7. SDST of unprocessed initial jobs and new incoming jobs in M_1

Job	P5	P2	P11	P1	P10	P15	P4	P6	P17	P21	P22	P23	P24	P25	P26	P27
P5	0	12	8	5	15	15	13	15	8	12	11	7	11	11	6	8
P2	9	0	8	6	11	7	12	7	11	14	10	6	8	10	8	8
P11	7	10	0	6	11	15	7	15	6	11	5	12	11	5	15	5
P1	15	15	9	0	15	10	9	14	15	5	9	9	7	13	6	10
P10	12	13	11	5	0	8	13	11	11	7	14	6	10	9	5	8
P15	10	15	9	12	11	0	7	6	10	11	5	15	9	5	8	14
P4	13	8	10	10	15	15	0	8	11	7	10	14	7	8	12	8
P6	12	12	6	10	15	15	12	0	9	14	6	6	8	13	7	11
P17	13	5	7	7	11	12	15	14	0	6	10	5	15	5	7	12
P21	6	12	7	6	15	11	11	10	11	0	6	7	7	6	10	10
P22	5	5	10	14	8	15	9	11	11	7	0	9	9	7	9	11
P23	11	13	9	11	12	6	6	10	9	7	14	0	9	10	12	12
P24	13	9	8	6	10	7	5	6	14	13	8	12	0	12	6	12
P25	9	13	7	5	14	10	5	15	10	14	7	9	6	0	11	7
P26	7	14	9	7	7	11	5	9	10	15	5	12	9	11	0	12
P27	5	14	13	10	14	8	13	15	14	5	6	12	7	14	6	0

Table 8. SDST of unprocessed initial jobs and new incoming jobs in M_2

Job	P5	P2	P11	P1	P10	P15	P4	P6	P17	P21	P22	P23	P24	P25	P26	P27
P5	0	10	12	12	8	9	6	10	5	5	8	14	10	13	15	9
P2	6	0	9	9	14	7	7	14	13	9	5	7	14	6	7	5
P11	13	5	0	8	12	9	11	11	7	13	13	6	9	8	8	10
P1	15	5	13	0	11	13	6	14	11	7	5	9	11	13	12	15
P10	8	11	7	8	0	8	10	13	11	7	14	15	8	10	13	10
P15	8	9	8	14	6	0	5	5	9	15	14	8	10	9	11	5
P4	15	11	9	12	5	7	0	12	5	10	8	11	9	14	10	9
P6	8	13	6	15	8	10	15	0	7	9	5	8	9	7	13	15
P17	7	8	10	12	15	5	11	10	0	7	5	8	9	8	13	13
P21	13	8	7	15	6	9	7	9	13	0	9	15	8	10	9	14
P22	9	6	13	5	14	7	13	6	9	9	0	14	15	13	12	8
P23	13	12	14	8	6	10	8	12	14	14	10	0	5	12	5	12
P24	5	8	6	8	7	10	13	9	13	9	11	7	0	14	15	15
P25	6	9	5	9	12	10	14	6	11	5	10	12	7	0	11	6
P26	5	6	6	8	10	11	10	9	5	11	8	13	10	8	0	12
P27	8	14	8	13	15	14	15	6	6	11	7	13	12	9	15	0

Table 9. SDST of unprocessed initial jobs and new incoming jobs in M_3

Job	P5	P2	P11	P1	P10	P15	P4	P6	P17	P21	P22	P23	P24	P25	P26	P27
P5	0	15	8	15	10	11	14	15	11	5	11	7	8	13	5	11
P2	15	0	7	11	7	7	11	13	12	7	11	7	8	10	8	15
P11	8	7	0	14	6	12	7	12	8	11	9	15	7	7	10	10
P1	13	11	13	0	5	11	5	6	6	9	9	15	11	5	6	11
P10	15	12	8	7	0	5	6	6	10	7	7	7	6	11	6	14
P15	7	14	7	15	6	0	14	12	11	9	8	15	7	8	7	15
P4	11	10	15	12	7	10	0	8	9	8	5	9	14	6	7	15
P6	7	14	6	13	11	7	12	0	13	9	5	7	11	5	7	7
P17	6	10	10	8	7	8	5	13	0	14	13	7	9	13	13	9
P21	5	6	13	7	8	15	14	10	6	0	15	6	9	10	7	10
P22	12	9	14	8	13	13	11	13	7	5	0	7	13	13	10	12
P23	13	12	10	10	12	7	13	11	7	9	9	0	14	5	8	14
P24	8	7	10	10	9	12	9	10	5	13	8	12	0	5	15	15
P25	6	8	15	13	7	14	5	5	7	10	12	5	11	0	8	8
P26	8	6	9	8	7	13	15	9	5	9	11	7	13	8	0	7
P27	11	9	12	5	7	12	11	5	7	15	5	7	7	10	12	0

Table 10. SDST of unprocessed initial jobs and new incoming jobs in M_4

Job	P5	P2	P11	P1	P10	P15	P4	P6	P17	P21	P22	P23	P24	P25	P26	P27
P5	0	6	7	11	9	7	14	15	5	10	13	5	8	10	5	9
P2	8	0	7	9	7	10	9	7	5	11	11	11	12	9	8	15
P11	10	8	0	11	5	6	12	13	15	15	6	11	11	13	13	11
P1	13	10	9	0	15	12	5	10	5	5	14	14	5	6	15	13
P10	11	5	14	14	0	10	11	9	9	8	10	7	5	9	14	15
P15	11	7	13	15	11	0	9	13	14	8	11	6	14	7	8	12
P4	11	8	11	8	11	11	0	8	15	10	8	14	11	8	5	13
P6	11	11	11	12	14	15	14	0	10	7	6	6	12	9	11	13
P17	10	6	9	5	12	10	7	10	0	7	6	8	7	12	7	13
P21	9	11	8	11	10	10	11	8	5	0	13	7	11	14	12	9
P22	10	9	11	9	15	5	6	13	14	11	0	15	9	13	13	11
P23	10	6	14	13	11	9	13	10	10	14	5	0	7	14	10	7
P24	5	5	7	12	14	10	11	6	9	12	10	6	0	14	7	6
P25	6	9	15	8	14	9	6	6	15	14	9	12	7	0	13	7
P26	10	5	11	12	5	13	10	15	6	5	14	10	8	6	0	7
P27	11	11	8	5	6	9	7	5	9	11	5	10	6	6	9	0

http://dx.doi.org/10.30656/jsmi.v10i1.11538

Table 11. SDST of unprocessed initial jobs and new incoming jobs in M_5

Job	P5	P2	P11	P1	P10	P15	P4	P6	P17	P21	P22	P23	P24	P25	P26	P27
P5	0	9	14	13	8	12	14	11	13	12	11	5	10	11	15	12
P2	10	0	5	15	6	7	7	15	12	9	6	14	13	7	5	9
P11	5	14	0	7	7	5	12	10	9	7	5	12	12	9	6	7
P1	6	14	9	0	8	15	7	12	10	7	7	5	5	5	6	8
P10	15	11	6	10	0	8	15	15	10	11	8	15	9	10	12	12
P15	8	13	9	11	14	0	14	9	5	14	9	5	8	14	13	11
P4	11	11	9	14	7	12	0	13	15	9	14	9	12	12	9	12
P6	5	12	13	15	8	5	14	0	15	8	13	5	8	8	15	9
P17	7	12	12	10	15	10	7	7	0	9	12	9	10	15	8	15
P21	7	12	9	9	6	12	14	12	7	0	7	15	8	6	9	12
P22	13	5	6	11	9	15	6	11	13	10	0	15	12	13	9	13
P23	7	6	12	7	8	14	9	9	5	9	12	0	11	7	7	14
P24	15	9	14	11	12	10	13	14	6	14	8	13	0	13	9	8
P25	15	11	13	7	11	7	8	11	15	8	11	7	6	0	8	7
P26	7	10	14	14	13	15	9	7	15	13	5	14	5	10	0	13
P27	10	6	7	8	11	5	10	12	6	10	7	13	5	15	11	0

In stage 2 scheduling, data regarding SDST, processing time, and debate of the incoming jobs are collected, as shown in Table 7, Table 8, Table 9, Table 10, Table 11 and Table12.

Table 12. Processing time of new jobs and due dates

Jobs	M_1	M_2	M_3	M_4	M_5	*Due date*
P21	22	14	7	5	7	565
P22	12	20	10	19	14	389
P23	22	20	13	5	3	398
P24	8	12	15	20	7	431
P25	10	19	12	6	10	539
P26	12	18	12	11	14	409
P27	22	10	5	18	9	505

Using the same GA parameter settings as in Stage 1, the reactive scheduling phase produced the updated job sequence 26–22–23–24–25–27–11–21–4–15–17–6–2–1–10–5. In this stage, the optimization focuses on minimizing the deviation from the baseline *makespan* for the remaining initial jobs while simultaneously reducing lateness for the newly arrived jobs. Based on this objective, the resulting sequence yields the lateness for each new job (Table 13). The updated schedule yields a new *makespan* of 764-time units, representing a deviation of 183-time units from the initial baseline, reflecting the necessary extension caused by the insertion of new job arrivals.

Table 13. Lateness of new jobs

New Jobs	*Lateness*
P21	0
P22	11
P23	13
P24	26
P25	0
P26	0
P27	0

The Gantt chart of the scheduling results in stages 1 and 2 is shown in Fig. 5 and Fig. 6. The 183-unit increase in *makespan* indicates a clear stability

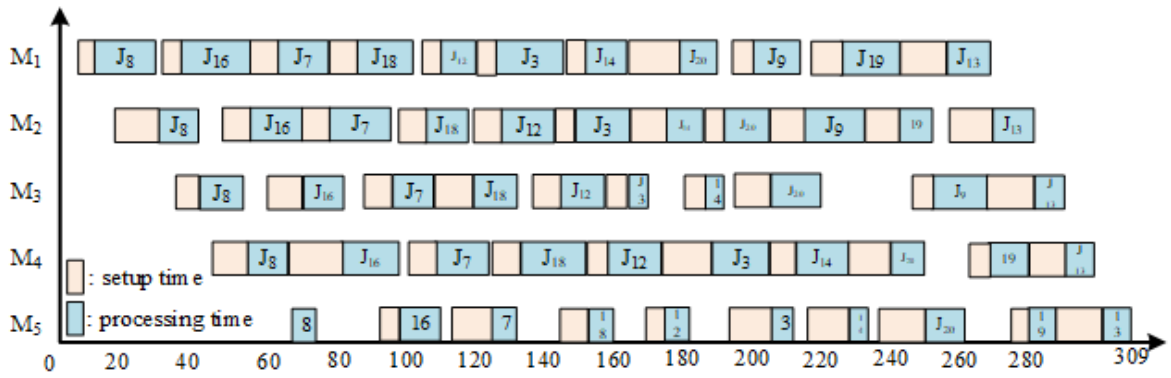


Fig. 5. Scheduling Gantt chart for stage 1

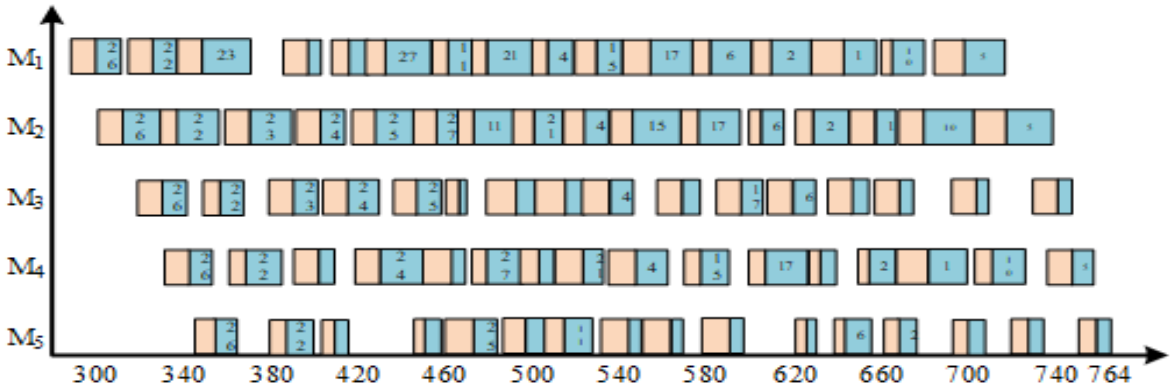


Fig. 6. Scheduling Gantt chart for stage 2

responsiveness trade-off between the seven new -date performance, but at the cost of elongating the *makespan*. One potential approach to further reduce waiting time is the implementation of SMED, which has been widely reported to reduce setup durations in manufacturing systems. Although this strategy was not experimentally evaluated in this study, it may provide a practical direction for future industrial implementation.

From a scheduling perspective, waiting time can be reduced by integrating setup-aware sequencing rules and setup-time balancing mechanisms within the optimization process. This includes prioritizing job orders that minimize large SDST transitions, grouping similar jobs into families to reduce sequence variability, and dynamically adjusting machine start times so that downstream machines become available exactly when upstream processing finishes. Combining these approaches helps synchronize machine operations, enabling the system to move closer to an ideal no-wait *flow shop* with minimal idle or buffer periods.

5. CONCLUSION

This study addresses the challenge of maintaining high scheduling performance in a dynamic no-wait *flow shop* environment where a batch of 20 predictive jobs is followed by the insertion of 7 new incoming jobs. By developing a two-stage GA–based predictive–reactive scheduling framework, the study fills a clear gap in the literature, which typically examines reactive scheduling either without predictive baselines, without sequence-dependent setup times, or without multi-objective criteria combining *makespan* deviation and lateness. The findings demonstrate that incorporating the predictive *makespan* as the baseline criterion in stage 2 rescheduling supports schedule stability while simultaneously ensuring customer service quality for new orders.

The results further reveal that complete no-wait feasibility is difficult to achieve when sequence-dependent setup times are long and both schedule stability and lateness minimization are simultaneously required. These insights confirm the practical relevance of hybrid objectives in real electronic manufacturing systems, where operational constraints, such as unavoidable setups, workstation synchronization, and variable release times, interact with dynamic demand patterns. The developed framework successfully handled this complexity, offering a scalable and efficient decision-support tool compared with fixed-rule or single-objective approaches. Beyond the framework performance, the study highlights managerial implications, which are the need to understand trade-offs between responsiveness and stability, coordinate predictive capacity planning with real-time order handling, and adopt operational improvements.

Unlike studies proposing new optimization algorithms, this study aims to develop and validate a predictive–reactive scheduling methodology for a realistic electronics manufacturing environment. Therefore, rather than conducting extensive benchmark comparisons, the emphasis is placed on accurately modelling the industrial scheduling process and demonstrating the applicability of the proposed framework through a representative industrial case study. The proposed GA serves as the optimization engine for the mathematical model and scheduling methodology.

Although this study contributes to the development of PRI under dynamic job arrivals and multi-objective criteria, several opportunities for future research remain. The model can be extended to incorporate sustainability metrics (e.g., energy use, worker load, and emission cost), stochastic processing times, and machine breakdown uncertainty. Hybridizing GA with reinforcement learning or digital twin-based real-time feedback may further enhance responsiveness. Additionally, integrating order acceptance decisions or production and inventory coordination could broaden the framework's applicability to more complex factory environments. Overall, this study indicates that advanced metaheuristics can effectively bridge the gap between predictive planning and reactive execution, offering a practical scheduling solution for modern high-variability manufacturing systems.

http://dx.doi.org/10.30656/jsmi.v10i1.11538

ACKNOWLEDGMENT

The author would like to express the highest appreciation to the Department of Industrial Engineering, Universitas Islam Indonesia, which has funded the publication of this manuscript.

REFERENCES

[1] H. Wang, R. Li, and W. Gong, "Minimizing tardiness and makespan for distributed heterogeneous unrelated parallel machine scheduling by knowledge and Pareto-based memetic algorithm," *Egypt. Informatics J.*, vol. 24, no. 3, p. 100383, Sep. 2023, doi: 10.1016/j.eij.2023.05.008.

[2] K. A. Bakon and T. Holczinger, "S-Graph-Based Reactive Scheduling with Unexpected Arrivals of New Orders," *Machines*, vol. 12, no. 7, p. 446, Jun. 2024, doi: 10.3390/machines12070446.

[3] D. Ouelhadj and S. Petrovic, "A survey of dynamic scheduling in manufacturing systems," *J. Sched.*, vol. 12, no. 4, pp. 417–431, Aug. 2009, doi: 10.1007/s10951-008-0090-8.

[4] S. Anderer, A. Hochreiter, and T. Pajdla, "Meta heuristics for dynamic machine scheduling," in *Proceedings of the International Multi-Conference on Engineering and Technological Innovation (IMETI)*, 2018, pp. 111–118. [Online]. Available: https://www.scitepress.org/Papers/2018/69307/69307.pdf

[5] W. Zhang, X. Bao, X. Hao, and M. Gen, "Metaheuristics for multi-objective scheduling problems in industry 4.0 and 5.0: a state-of-the-arts survey," *Front. Ind. Eng.*, vol. 3, p. 1540022, Jan. 2025, doi: 10.3389/fieng.2025.1540022.

[6] F. Werner, "Genetic algorithms for shop scheduling problems: A survey," Otto-von-Guericke-Universität Magdeburg, 2010. [Online]. Available: https://mat.uab.cat/~Alseda/MasterOpt/p11-31.pdf

[7] H. F. Rahman, R. Sarker, and D. Essam, "A genetic algorithm for permutation flowshop scheduling under practical make-to-order production system," *Artif. Intell. Eng. Des. Anal. Manuf.*, vol. 31, no. 1, pp. 87–103, Feb. 2017, doi: 10.1017/S0890060416000196.

[8] M. Mirabi, S. M. T. Fatemi Ghomi, and F. Jolai, "A novel hybrid genetic algorithm to solve the make-to-order sequence-dependent flow-shop scheduling problem," *J. Ind. Eng. Int.*, vol. 10, no. 2, p. 57, Jun. 2014, doi: 10.1007/s40092-014-0057-7.

[9] D. Panda, F. Bayu, and M. Ramteke, "Discrete time reactive scheduling of gasoline blending and product delivery in presence of demand and component uncertainties using graphical genetic algorithm," *Comput. Chem. Eng.*, vol. 143, p. 107100, Dec. 2020, doi: 10.1016/j.compchemeng.2020.107100.

[10] P. Györgyi, T. Kis, and E. Szögi, "Reactive scheduling of uncertain jobs with maximum time lags," *Eur. J. Oper. Res.*, vol. 326, no. 1, pp. 69–77, Oct. 2025, doi: 10.1016/j.ejor.2025.04.013.

[11] J. Qiu, J. Liu, C. Peng, and Q. Chen, "A novel predictive–reactive scheduling method for parallel batch processor lot-sizing and scheduling with sequence-dependent setup time," *Comput. Ind. Eng.*, vol. 189, p. 109985, Mar. 2024, doi: 10.1016/j.cie.2024.109985.

[12] R. Otero-Caicedo, C. E. M. Casas, C. B. Jaimes, C. F. G. Garzón, E. A. Y. Vergel, and J. C. Z. Valdés, "A preventive–reactive approach for nurse scheduling considering absenteeism and nurses' preferences," *Oper. Res. Heal. Care*, vol. 38, p. 100389, Sep. 2023, doi: 10.1016/j.orhc.2023.100389.

[13] Y. Su, Z. Xu, and D. Liu, "A model and algorithm for reactive multi-objective multi-skilled project scheduling under resource disruptions," *Comput. Ind. Eng.*, vol. 203, p. 111043, May 2025, doi: 10.1016/j.cie.2025.111043.

[14] W. Li, H. Fan, L. Cai, W. Guo, Z. Wu, and P. Yang, "Digital twin-driven proactive-reactive scheduling framework for port multi-equipment under a complex uncertain environment," *Simul. Model. Pract. Theory*, vol. 136, p. 103011, Nov. 2024, doi: 10.1016/j.simpat.2024.103011.

[15] W. Peng, X. Lin, and H. Li, "Critical chain based Proactive-Reactive scheduling for Resource-Constrained project scheduling under uncertainty," *Expert Syst. Appl.*, vol. 214, p. 119188, Mar. 2023, doi: 10.1016/j.eswa.2022.119188.

[16] S. L. Takeda-Berger, Í. R. S. Agostino, M. R. F. da Silva, and E. M. Frazzon, "Reactive production scheduling approach based on inventory availability," *IFAC-PapersOnLine*, vol. 55, no. 10, pp. 418–423, 2022, doi: 10.1016/j.ifacol.2022.09.429.

[17] R. K. Chakrabortty, H. F. Rahman, K. M. A. Haque, S. K. Paul, and M. J. Ryan, "An event-based reactive scheduling approach for the Resource Constrained Project Scheduling Problem with unreliable resources," *Comput. Ind. Eng.*, vol. 151, p. 106981, Jan. 2021, doi: 10.1016/j.cie.2020.106981.

[18] F. Echsler Minguillon and N. Stricker, "Robust predictive–reactive scheduling and its effect on

machine disturbance mitigation," *CIRP Ann.*, vol. 69, no. 1, pp. 401–404, 2020, doi: 10.1016/j.cirp.2020.03.019.

[19] B. J. Joo, T. J. Chua, T. X. Cai, and P. C. Chua, "Coordination-based reactive resource-constrained project scheduling," *Procedia CIRP*, vol. 81, pp. 51–56, 2019, doi: 10.1016/j.procir.2019.03.010.

[20] S. L. Takeda Berger, R. M. Zanella, and E. M. Frazzon, "Towards a data-driven predictive-reactive production scheduling approach based on inventory availability," *IFAC-PapersOnLine*, vol. 52, no. 13, pp. 1343–1348, 2019, doi: 10.1016/j.ifacol.2019.11.385.

[21] D. Panda and M. Ramteke, "Reactive scheduling of crude oil using structure adapted genetic algorithm under multiple uncertainties," *Comput. Chem. Eng.*, vol. 116, pp. 333–351, Aug. 2018, doi: 10.1016/j.compchemeng.2018.04.005.

[22] T. Sousa, H. Morais, Z. Vale, and R. Castro, "A multi-objective optimization of the active and reactive resource scheduling at a distribution level in a smart grid context," *Energy*, vol. 85, pp. 236–250, Jun. 2015, doi: 10.1016/j.energy.2015.03.077.

[23] D. Rahmani and R. Ramezanian, "A stable reactive approach in dynamic flexible flow shop scheduling with unexpected disruptions: A case study," *Comput. Ind. Eng.*, vol. 98, pp. 360–372, Aug. 2016, doi: 10.1016/j.cie.2016.06.018.

[24] T. Li, J. -q. Li, X. -l. Chen, and J. -k. Li, "Solving distributed assembly blocking flowshop with order acceptance by knowledge-driven multiobjective algorithm," *Eng. Appl. Artif. Intell.*, vol. 137, p. 109220, 2024, doi: 10.1016/j.engappai.2024.109220.

[25] A. Khare and S. Agrawal, "Scheduling hybrid flowshop with sequence-dependent setup times and due windows to minimize total weighted earliness and tardiness," *Comput. Ind. Eng.*, vol. 135, pp. 780–792, Sep. 2019, doi: 10.1016/j.cie.2019.06.057.

[26] G. Muller, *System Modeling and Analysis: A Practical Approach*. University of South-Eastern Norway-NISE, 2023. [Online]. Available: https://www.gaudisite.nl/info/SystemModelingAndAnalysis.info.html

[27] K. Medini, S. Peillon, M. Orellano, S. Wiesner, and A. Liu, "System Modelling and Analysis to Support Economic Assessment of Product-Service Systems," *Systems*, vol. 9, no. 1, p. 6, Jan. 2021, doi: 10.3390/systems9010006.

[28] F. Wilking, D. Horber, S. Goetz, and S. Wartzack, "Utilization of system models in model-based systems engineering: definition, classes and research directions based on a systematic literature review," *Des. Sci.*, vol. 10, p. e6, Feb. 2024, doi: 10.1017/dsj.2024.3.

[29] A. Dér, L. Hingst, P. Nyhuis, and C. Herrmann, "Concept for modeling and quantitative evaluation of life cycle dynamics in factory systems," *Prod. Eng.*, vol. 17, no. 3–4, pp. 601–611, Jun. 2023, doi: 10.1007/s11740-023-01189-4.

[30] L. M. Hvattum, "Adjusting the order crossover operator for capacitated vehicle routing problems," *Comput. Oper. Res.*, vol. 148, p. 105986, Dec. 2022, doi: 10.1016/j.cor.2022.105986.

[31] V. Cicirello, "A Survey and Analysis of Evolutionary Operators for Permutations," in *Proceedings of the 15th International Joint Conference on Computational Intelligence*, SCITEPRESS - Science and Technology Publications, 2023, pp. 288–299. doi: 10.5220/0012204900003595.

[33] P. Ciepliński and S. Golak, "Crossover Operator Inspired by the Selection Operator for an Evolutionary Task Sequencing Algorithm," *Appl. Sci.*, vol. 14, no. 24, p. 11786, Dec. 2024, doi: 10.3390/app142411786.

http://dx.doi.org/10.30656/jsmi.v10i1.11538