Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

# Predictive-reactive scheduling in dynamic flexible job shops with assembly precedence constraints: A long-term performance analysis under uncertainty

**Alejandro Uribe**

auribev1@eafit.edu.co

EAFIT University

**Urtzi Otamendi**

Vicomtech Foundation, Basque Research and Technology Alliance (BRTA), Donostia-San Sebastián

**Xabier Belaunzaran**

Vicomtech Foundation, Basque Research and Technology Alliance (BRTA), Donostia-San Sebastián

**Arkaitz Artetxe**

Vicomtech Foundation, Basque Research and Technology Alliance (BRTA), Donostia-San Sebastián

**Juan Carlos Rivera**

EAFIT University

**Research Article**

**Keywords:** Dynamic Flexible Job Shop Scheduling, Agent-Based Simulation, Predictive–Reactive Scheduling, Optimization, Uncertainty

**Posted Date:** May 25th, 2026

**DOI:** https://doi.org/10.21203/rs.3.rs-8925719/v1

**License:** This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

**Additional Declarations:** No competing interests reported.

# Predictive-reactive scheduling in dynamic flexible job shops with assembly precedence constraints: A long-term performance analysis under uncertainty

Alejandro Uribe[1,2*], Urtzi Otamendi[2,3], Xabier Belaunzaran[2,3], Arkaitz Artetxe[2], Juan Carlos Rivera[1]

[1*]School of Applied Sciences and Engineering, Analytics and Supply Chain Research Group, Universidad EAFIT, Carrera 49 N 7 Sur-50, Medellín, 05001, Antioquia, Colombia.

[2]Vicomtech Foundation, Basque Research and Technology Alliance (BRTA), Mikeletegi Pasealekua 57, Donostia-San Sebastián, 20009, Gipuzkoa, Spain.

[3]Department of Computer Sciences and Artificial Intelligence, University of the Basque Country (UPV/EHU), Manuel Lardizabal Pasealekua 1, Donostia-San Sebastián, 20018, Gipuzkoa, Spain.

*Corresponding author(s). E-mail(s): auribev1@eafit.edu.co;
Contributing authors: uotamendi@vicomtech.org;
xbelauzaran@vicomtech.org; aartetxe@vicomtech.org;
xbelauzaran@vicomtech.org;

**Abstract**

This work addresses an extended Dynamic Flexible Job Shop Scheduling Problem (DFJSSP) that incorporates non-linear precedence relations to represent assembly operations, resulting in a more complex precedence structure and increased interdependence between scheduling decisions. The problem is studied under uncertainty, where stochastic job arrivals require continuous predictive-reactive schedule updates while preserving stability with respect to prior decisions. To capture these dynamics, we develop an agent-based simulation framework conceived as a benchmarking environment capable of representing uncertainty and evaluating how scheduling decisions based on partial and evolving system information propagate over extended planning horizons. The framework enables a longitudinal assessment of solution quality, workload balance, and schedule stability, providing insights beyond traditional static or short-term analyses.

Within this environment, we benchmark mixed-integer linear programming, a memetic algorithm, and dispatching rules. Results show that optimization-based approaches achieve superior solution quality but face scalability limitations, while the memetic algorithm offers a competitive balance between performance and computational effort. Dispatching rules provide immediate responsiveness but lower global efficiency. The study demonstrates that scheduling policies exhibit fundamentally different long-term behaviors, highlighting the importance of simulation-based benchmarking to understand the cumulative impact of predictive-reactive scheduling policies in complex and uncertain manufacturing systems.

**Keywords:** Dynamic Flexible Job Shop Scheduling. Agent-Based Simulation, Predictive–Reactive Scheduling, Optimization, Uncertainty

# 1 Introduction

Scheduling is a fundamental decision-making process that directly affects the operational efficiency of organizations across multiple domains (Otamendi et al. 2025). It involves assigning jobs to resources while satisfying operational constraints and optimizing objectives such as makespan, tardiness, and resource utilization (Agnetis et al. 2025). Classical scheduling models typically assume deterministic environments in which job characteristics, machine availability, and processing parameters are known in advance; however, these assumptions are often unrealistic in dynamic, real-world settings.

Within this context, the Flexible Job Shop Scheduling Problem (FJSSP) has gained prominence as a practically relevant and computationally challenging extension of the classical Job Shop Scheduling Problem (JSP). Unlike the JSP, the FJSSP allows operations to be processed by any machine from a subset of eligible machines, increasing the complexity of both machine assignment and sequencing decisions (Dauzère-Pérès et al. 2024). Moreover, traditional FJSSP formulations usually assume linear operation sequences with fixed precedence relations, which fail to capture real production environments where jobs may exhibit non-linear precedence structures. In such cases, operations can depend on multiple predecessors, forming network-like job structures that ultimately converge to a single terminal operation marking job completion.

Dynamic environments further complicate scheduling by introducing unpredictable events such as machine breakdowns, urgent job arrivals, or rework operations, which often render static schedules infeasible and require real-time or near-real-time adjustments (Lv et al. 2025). Conventional reactive approaches frequently rely on complete rescheduling, potentially disregarding stability considerations and causing substantial disruption to ongoing operations. While flexible systems expand the range of feasible machine assignments, a central challenge remains in balancing adaptability with schedule robustness and stability. Recent research highlights that the increasing complexity of modern manufacturing systems—driven by customization, rapid technological advances, and frequent order changes—demands scheduling approaches capable of managing dynamic events in a structured and efficient manner (Yuan et al. 2025).

Traditional robustness-enhancing techniques, such as time buffering or slack insertion, are often resource-inefficient, whereas purely reactive dispatching-rule-based methods offer fast responses but may suffer from poor global coordination in highly volatile settings. Consequently, predictive–reactive scheduling strategies have emerged as a promising middle ground, combining initial schedules with systematic adjustments as disruptions occur.

The Dynamic Flexible Job Shop Scheduling Problem (DFJSSP) extends the classical FJSSP by explicitly accounting for dynamic events such as urgent job insertions, variable processing times, and equipment failures (Zhang et al. 2023). This problem is particularly relevant in highly competitive, customer-driven industries, where both responsiveness and robustness are essential. The literature predominantly explores two solution paradigms: metaheuristic approaches, which provide flexibility and strong global search capabilities, and dispatching rules, which enable rapid responses but offer limited adaptability. These methods are typically evaluated under simulated disruptions, underscoring the importance of structured experimentation in dynamic scheduling environments. In parallel, manufacturing firms are increasingly investing in intelligent scheduling systems to enhance operational performance, reduce costs, and address customer and sustainability requirements (Uribe et al. 2025). Beyond improving throughput and resource utilization, effective scheduling also supports environmental and economic sustainability by reducing waste and optimizing energy consumption.

This study addresses the DFJSSP with unplanned job arrivals using a predictive–reactive scheduling framework that accommodates non-linear routing structures with multiple precedence constraints. An agent-based simulation is employed as a benchmarking environment to evaluate how system-state-based scheduling decisions made in response to dynamic events influence long-term system performance. Three solution approaches are compared: a priority dispatching rule (DR), a Memetic Algorithm (MA), and a Mixed Integer Linear Programming (MILP) formulation. Their performance is assessed at the end of the planning horizon in terms of scheduling quality, solution stability, and system responsiveness. By explicitly linking short-term, state-dependent decisions to long-term outcomes, this study provides a structured assessment of adaptive scheduling strategies under realistic manufacturing dynamics.

The remainder of this paper is organized as follows. Section 2 reviews the literature and discusses the main approaches proposed to address the DFJSSP. Section 3 describes the simulation environment and the proposed solution methods. Section 4 outlines the experimental setup and the considerations adopted for the study. Section 5 presents the benchmarking results, and finally, Section 6 provides the conclusions and directions for future work.

## 2 Literature review

The FJSSP has been extensively studied with various extensions and solution methods addressing its increasing complexity and realistic constraints. Foundational approaches formulate the problem as mixed-integer linear programs (MILP), incorporating constraints such as alternative resources, release and due dates, sequence-dependent setup

times, and heterogeneous worker skills, reflecting operational realities (Winklehner and Hauder 2022; Manafi et al. 2025; Li et al. 2025; Perroux et al. 2024). Given the computational intractability of exact methods for large instances, metaheuristic and hybrid algorithms, including cooperative co-evolutionary strategies and evolutionary grey wolf optimizers, have been developed to efficiently solve large-scale and multi-objective variants, optimizing makespan, workload balance, and tardiness (Wang et al. 2024; Li et al. 2025; Xu et al. 2025; Zhu and Zhou 2020). Recent advancements focus on integrating dynamic and uncertain environmental factors such as energy consumption, renewable sources, and energy storage systems, employing multi-level memetic and hierarchical collaborative frameworks to minimize makespan, energy costs, and emissions in real time (Burmeister et al. 2025; Hu et al. 2025). Moreover, production system characteristics such as machine maintenance under failure rates and lot streaming constraints have been incorporated to enhance scheduling robustness and practical applicability (Wang et al. 2024; Shao et al. 2025). These developments collectively demonstrate the evolving landscape of FJSSP research, balancing methodological rigor and practical relevance. In addition, some recent studies have investigated learning-based methods, such as reinforcement learning, as alternative approaches to improve adaptability and responsiveness in dynamic scheduling contexts (Yang et al. 2025; Zhang et al. 2025; Liu et al. 2025; Zhang et al. 2025; Yu et al. 2025; Ding et al. 2025; Wang et al. 2025; Echeverria et al. 2025; Zampella et al. 2025; Li and Chai 2025; Ghasemi et al. 2024; Wang et al. 2025; Zhang et al. 2024; Zhao et al. 2024).

The integration of machine learning with real-time data-driven simulation further enhances scheduling performance under uncertainty by dynamically updating priority dispatching rules and due date predictions based on shop floor feedback, thus enabling responsive adjustments to stochastic job arrivals and machine states (Wang et al. 2023). Digital twin-driven dynamic scheduling incorporates human factors such as workers' learning and forgetting processes, simultaneously optimizing makespan, carbon emissions, costs, and quality stability through virtual workshop simulations and multi-objective metaheuristics (Li and Chen 2023). In rescheduling scenarios affected by environmental disturbances, predictive disturbance evaluation systems combined with support vector machine classifiers and improved metaheuristics provide effective adaptation mechanisms (Zhang et al. 2025).

Multi-method simulation approaches, combining discrete-event simulation, agent-based models, and Monte Carlo techniques, allow for detailed process state tracking and stochastic evaluation of production parameters, addressing uncertainty comprehensively (Ghasemi et al. 2024). Simulation optimization methods have also been effectively applied to complex industrial cases, including parcel hub scheduling with congestion modeling via FlexSim and hybrid genetic-particle swarm algorithms (Chen et al. 2023), and surgical scheduling in multi-operating room environments using discrete-event simulation to assess sequencing policies (Xiao and Yoogalingam 2022). The application of composite dispatching rules informed by fuzzy multi-criteria decision-making integrated with simulation models further supports nuanced prioritization in FJSSP (Thenarasu et al. 2022).

The DFJSSP is an extension of the classical FJSSP, designed to capture the behavior of real-world manufacturing systems where uncertainty and disruptions are

intrinsic. Unlike static models, the DFJSSP incorporates time-dependent events such as unexpected job arrivals, machine breakdowns, and processing time variability, making it both more realistic and significantly more complex. Various sources of uncertainty primarily characterize the DFJSSP. Among the most prevalent is the dynamic arrival of jobs, which reflects the real-time introduction of new production orders (Yan et al. 2025). This aspect has been addressed in a wide range of studies, either through online scheduling models (Körner et al. 2025; Wan et al. 2024; Shi et al. 2025), or via rolling time horizon rescheduling frameworks (Li et al. 2021), which aim to balance short-term responsiveness with long-term efficiency. Machine breakdowns and stochastic failures represent another critical source of uncertainty, introducing abrupt changes in resource availability (Wang et al. 2025; Tang et al. 2024). These are typically modeled either through probabilistic failure distributions or embedded maintenance policies (Wu et al. 2025; Sreekara Reddy et al. 2018). In this context, preventive and predictive maintenance strategies for machines have received growing attention as effective means to mitigate the disruptive impact of such events (An et al. 2024, 2023a,b). Uncertainty also arises in processing times, due to fluctuations in production speed or variability in operator efficiency (Gao et al. 2016; long Chen et al. 2023). Some studies address this through stochastic modeling and robust optimization strategies (Fan et al. 2024; Shi et al. 2025; Wu et al. 2025), often with the goal of minimizing worst-case performance degradation. Other dynamic factors such as job cancellations, due date modifications, and resource availability fluctuations further increase the complexity of the scheduling environment (Chen et al. 2024; Li et al. 2022; Liu et al. 2024).

To capture the dynamics of the DFJSSP, researchers have employed a variety of modeling approaches. Mathematical programming, especially MILP, remains a foundational tool to describe the problem formally. These models often include stochastic elements or scenario-based extensions to represent uncertain events (Fan et al. 2024; long Chen et al. 2023; An et al. 2024). However, due to their computational demands, they are often limited to small- or medium-scale problems. Another line of work adopts proactive or predictive and reactive frameworks, which generate an initial baseline schedule and update it dynamically as new information becomes available (Nouiri et al. 2018; long Chen et al. 2023; Gomes et al. 2013). These frameworks typically prioritize schedule stability while maintaining acceptable levels of performance. In parallel, multi-agent systems and digital twin architectures have emerged as promising paradigms to simulate real-time, decentralized scheduling environments (Wang et al. 2025, 2023). A significant modeling trend is the multi-objective formulation of the DFJSSP. Beyond classical makespan minimization, modern studies incorporate objectives such as total tardiness, machine workload balance, energy consumption, and schedule robustness or stability (Xu et al. 2025; Sun et al. 2024; Guo et al. 2025; Feng et al. 2025; Schworm et al. 2024; Zhao et al. 2025). This reflects a shift towards more comprehensive and sustainable decision-making in production planning.

The solution landscape for DFJSSP is broad, comprising exact methods, metaheuristics, and learning-based approaches. While exact methods, particularly MILP and constraint programming, offer precise and interpretable solutions, they struggle with scalability under dynamic conditions. Their role is now primarily confined to

benchmarking or integration within hybrid models (Li et al. 2025; Liu et al. 2024; Fan et al. 2024; long Chen et al. 2023). Metaheuristic algorithms are widely adopted due to their flexibility and ability to explore large search spaces effectively. Popular methods include Genetic Algorithms, NSGA III, Particle Swarm Optimization, Artificial Bee Colony, Tabu Search, and Immune Algorithms (Tang et al. 2024; Liu et al. 2024; Guo et al. 2024; Wei et al. 2024; Guo et al. 2025; Li et al. 2017; Feng et al. 2025; long Chen et al. 2023; Wang et al. 2021). These approaches are often hybridized with local search heuristics or dispatching rules to enhance exploitation capabilities and improve performance in dynamic settings (Zhou et al. 2025; Ji et al. 2025; An et al. 2023a; Sreekara Reddy et al. 2018). In response to disruptions in DFJSSP settings, a variety of rescheduling strategies have been proposed to restore feasibility and performance. Right-shift rescheduling adjusts affected operations by postponing their start times while preserving the original processing sequence, thereby minimally altering the initial plan (Fan et al. 2024; Wang et al. 2021). Other approaches, such as reassembling and insertion-based methods, aim to maintain the integrity of the original schedule while accommodating new jobs by exploiting available idle time slots, avoiding the need for full schedule regeneration (Li et al. 2017). In contrast, intersection-based approaches integrate the set of uncompleted operations from the current schedule with newly occurring stochastic events to generate a revised plan, often resulting in more substantial modifications to the original operational sequence (Li et al. 2017).

Table 1 provides an overview of existing approaches for the DFJSSP under unexpected events that require corrective actions to an initial schedule. It summarizes the objective functions considered in the literature, where schedule stability typically measures the deviation of the revised schedule from the original plan, commonly quantified through temporal shifts or changes in machine assignments. The table also identifies the types of dynamic behavior addressed, including stochastic job arrivals, unexpected machine breakdowns, and variations in job, worker, or machine characteristics throughout the scheduling horizon. In addition to the core features of the FJSSP, the literature incorporates several extended modeling aspects, such as order acceptance or rejection decisions, batching and lot-sizing policies, setup times, time windows (e.g., release and due dates), dual-resource constraints, transportation resources or travel times, and variable machine speeds. Finally, the solution method proposed in each study is reported, with the corresponding abbreviations defined in Appendix A.

Despite significant advances in addressing the DFJSSP, most existing studies tackle isolated dynamics—such as job arrivals or machine failures—within static instances, often neglecting the cumulative impact of real-time rescheduling decisions over time. Moreover, comprehensive benchmarking of diverse solution approaches remains largely unexplored. In particular, no prior work has systematically evaluated how decisions at each rescheduling interval affect overall scheduling performance across a planning horizon for different methods. To bridge this gap, this study addresses the DFJSSP with dynamic job arrivals, due dates, non-linear precedence constraints, and schedule stability (capturing changes in machine assignments and operation start times). An agent-based simulation environment is developed to emulate real manufacturing conditions beyond static benchmarks, allowing for real-time visualization and analysis of

system performance, including bottlenecks and saturation levels. Within this framework, we benchmark three representative methods a DR, an MA, and an exact MILP, to assess their long-term impact on scheduling quality, stability, and responsiveness. Although reinforcement learning methods show promising results in controlled scheduling environments, their scalability and real-time applicability remain limited (Ngwu et al. 2025). In practice, plant conditions and the characteristics of jobs, operations, and machines can diverge substantially from the training data, further limiting the applicability of RL approaches. Such discrepancies can undermine decision quality and necessitate frequent retraining, which reduces the practicality of RL for benchmarking; therefore, it is not considered in this study. This research advances the field by proposing and systematically evaluating adaptive scheduling strategies under realistic and dynamic production scenarios.

**Table 1** Approaches for solving the DFJSSP

| Reference | Objective function | | | | | | | Dynamic | | | Features | | | | | | | Solution method |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | $C_{max}$ | $T$ | $W_l$ | $E$ | \$ | *Other* | $S$ | $J_a$ | $M_F$ | $J/M$ | $A/C$ | $L/B$ | $St$ | $Tw$ | $D_r$ | $Tp$ | $Sp$ | |
| Gomes et al. (2013) | | X | | | | X | | X | | | | | | X | | | | MILP |
| Gao et al. (2016) | | | | | | X | | X | | | | | | | | | | TABC |
| Li et al. (2017) | X | | | | | | | X | X | X | X | | | | | | | HABC, TS |
| Nouiri et al. (2018) | X | | | X | | | X | | X | | | | X | | | | | GRM, PSO |
| Sreekara Reddy et al. (2018) | X | | X | | | | | | X | | | | | | | | | MOTLBO |
| Li et al. (2021) | X | | | | | | | X | X | X | X | | | X | | | | MCTS |
| Wang et al. (2021) | X | | X | X | | | | X | X | | | | X | | | | | Evolutionary game |
| Li et al. (2022) | X | | | X | | | | X | X | | | | | | X | X | | HDQN |
| An et al. (2023a) | X | | | | X | | X | X | | X | X | | | X | | | | HMOEA |
| An et al. (2023b) | X | | | | X | | X | X | X | | X | | | X | | | | INSGA-III/ARV |
| long Chen et al. (2023) | X | | | X | | X | X | X | X | | | | X | X | | X | X | Q-MOIA, Heuristics |
| Wang et al. (2023) | | | X | X | | X | | X | X | X | | | X | X | | | | IHA |
| Tang et al. (2024) | X | | | X | | | X | | X | | | | | | | X | | INSGA-III |
| Liu et al. (2024) | X | | | | | | | | X | | | X | | | X | | | GA, PSO |
| Wan et al. (2024) | X | | | | | | | X | X | X | | | | | | | | DIFFormer, DRL |
| Guo et al. (2024) | X | X | | | | X | | X | | X | | | | X | | | | GPHH |
| Liu et al. (2024) | X | | X | | | | X | X | | | | | | | X | X | | MIP, MOALNS |
| Wei et al. (2024) | X | | | X | X | | X | | X | X | | | | | X | | | IMA |
| An et al. (2024) | X | X | X | | | | X | | X | | | | | | X | | X | ACML-BCEA |
| Sun et al. (2024) | X | X | | | | | X | X | | | | | | X | | | | NSGA III, TS |
| Fan et al. (2024) | | X | | | | | | X | X | | | X | X | X | | | | Matheuristic, VNS, MILP |
| Schworm et al. (2024) | X | | | | | | X | X | X | | | | | | | | | QA |
| Chen et al. (2024) | X | | | X | | | X | | X | X | X | | | | X | X | | QNSGA-II |
| Fan et al. (2024) | X | | | | | | X | | X | | | | | | | | | GCLNTSO |
| Körner et al. (2025) | | X | | | | | | X | | | | | | X | | | | DRL |
| Wang et al. (2025) | X | | | | | | | X | X | | | | | | | | | MASC |
| Xu et al. (2025) | | X | | X | | | | X | | | | | | X | | | | GPHH |
| Shi et al. (2025) | | X | | | | X | | X | | | | | | X | | | | DEHH |
| Li et al. (2025) | X | | | | | X | X | | X | | | | | | X | | | QLIDE |
| Yuan et al. (2025) | | | | | | X | | | X | X | | | | X | | | | DRL |
| Guo et al. (2025) | | | | | | X | X | | X | | | | | | | | | VWMI, GA |
| Lv et al. (2025) | | | | | | | X | X | | | | | | | | | | DRL |
| Wu et al. (2025) | | X | | | | | | | X | | | | | X | | | | IDDQN |
| Feng et al. (2025) | X | | X | | | | X | | X | | | | | | | | | ATGA |
| Zhou et al. (2025) | X | | | | | X | | | X | | | | X | | | X | | DQN-DEMA |
| Ji et al. (2025) | X | | | | | | | X | X | X | X | | X | | X | | | MTDF, GA |
| Yan et al. (2025) | X | | | | | | | X | | | | | X | | | | | D3QN |
| Zhao et al. (2025) | X | X | X | | | X | X | | X | | | | | X | | | | AES-IBCEA |

**Objective function** $C_{max}$: makespan, $T$: tardiness, $W_l$: workingload, $E$: energy consumption, \$: cost, $S$: stability
**Dynamic attributes** $J_a$: job arrival, $M_F$: machine failure, $J/M$: change in job/machine attributes
**Features** $A/C$ job acceptance/cancelation, $L/B$: lotting/batching, $St$: setup times, $Tw$: time windows, $D_r$: dual resources
$Tp$: transportation, $Sp$: machine speed

# 3 Methodology

## 3.1 Problem Description

The *FJSSP* generalizes the classical JSSP by allowing each operation to be assigned to one machine from a set of eligible alternatives, with possibly different processing times.

This flexibility adds complexity, as the problem involves both machine assignment and sequencing decisions. Let:

- $J = \{1, \ldots, n\}$ be the set of jobs,
- $R = \{1, \ldots, K\}$ be the set of machines,
- $O = \{1, \ldots, N\}$ be the set of operations, with each job $j \in J$ composed of a sequence of $n_j$ operations.

Each operation $i \in O$ can be executed on a subset $R_i \subseteq R$ of eligible machines. The processing time of operation $i$ on machine $k \in R_i$ is denoted by $p_{k,i}$. Let $pr(i)$ denote the immediate predecessor of operation $i$ in its technological route. The standard assumptions are:

1. All machines and jobs are available at time zero.
2. No preemption is allowed: once started, an operation must run to completion.
3. Each machine can process at most one operation at a time.

To incorporate the dynamics of real-world production environments, two additional sets are introduced. Let $S$ denote the set of operations that must remain fixed in the schedule, as they are already being processed on specific machines at the time of optimization. Similarly, let $F$ represents the set of operations that were previously scheduled on specific machines at assigned times but may still be reallocated during rescheduling. Furthermore, the following parameters are considered:

- $dd_i$: the due date associated with operation $i$, representing its required completion time;
- $fi_i$: the originally assigned start time for operation $i \in F \cup S$;
- $fm_i$: the machine to which operation $i \in F \cup S$ was previously assigned.

## 3.2 Agent-based simulation for representing the DFJSSP

To describe the agent-based simulation of a production environment with the characteristics of the DFJSSP, the ODD (Overview, Design concepts, and Details) protocol was chosen as the formal structure (Grimm et al. 2010). Its systematic organization facilitates transparency and reproducibility by guiding the documentation of complex models in a structured and comprehensive manner.

Given the multi-agent nature of the system and the interactions among heterogeneous entities, the ODD protocol provides an effective framework to explicitly define agent behaviors, inter-agent interactions, and emergent dynamics. It also allows for a clear separation between high-level objectives, underlying design principles, and technical implementation details, which is essential for both academic rigor and interdisciplinary communication.

By adhering to the ODD protocol, the model's structure, logic, and assumptions are made accessible to researchers, practitioners, and reviewers, enhancing the model's credibility and facilitating its extension or replication in future studies. In this section we are going to describe each feature of the simulation in the framework of the ODD protocol.

### 3.2.1 Model purpose

The objective of this model is to simulate the behavior of a dynamic manufacturing environment to evaluate different job scheduling strategies across machines. The model addresses a DFJSSP, where products must follow a predefined sequence of operations performed on various machines. Consequently, each operation must be assigned to a machine and appropriately sequenced to optimize overall factory performance. Factory performance is assessed based on the stability of production schedules over time and the system's ability to meet customer demand. The model is expected to exhibit a shift in previously scheduled orders in response to the arrival of new jobs. Each implemented scheduling strategy seeks to adapt the current plan to meet estimated delivery dates by modifying the schedule in a way that respects operational constraints while optimizing performance indicators.

### 3.2.2 Entities and state variables

System entities are summarized in Table 2 and described below. The interaction among agents takes place within a networked environment. The job generator is connected to the job entities, which are in turn composed of a series of operations. These operations are linked to the solver agent, which communicates scheduling decisions to all machines. The simulation proceeds in discrete time intervals, which serve as the temporal scale of the system. This approach allows for generalization across a wide range of manufacturing contexts, where processing times may be measured in days, hours, minutes, or even finer resolutions.

- **Job Generator**: Responsible for creating jobs to represent the changing demand in a production environment. At each interval, it probabilistically generates a number of manufacturing orders with randomized characteristics tailored to the production setting. At each iteration, through a Poisson distribution, a number of jobs are generated according to an arrival rate $\lambda$.
- **Jobs**: Agents representing individual production orders. Each job consists of a predefined sequence of operations and is associated with a due date, both of which are treated as parameters. A job has a priority level that evolves as the simulation progresses. Additionally, each job has information on route precedence. That is, there may be several lines of operations converging on a main line (multiple precedence or non-linear paths).
- **Operations**: These are the fundamental components to be scheduled on machines. Each operation has a unique identifier, information about the set of machines capable of processing it, and the corresponding processing times for each machine; features treated as parameters. On the other hand, it has a counter of the number of temporary displacements of the activity and machine changes to measure the stability of the scheduling. Decision variables include the machine to which the operation is assigned and the time at which it will be completed.
- **Machines**: Entities where operations are executed. Each machine is characterized by a unique identifier. Additionally, machines maintain a production plan list of operations to be processed, order of completed operations, and a state indicator showing which activity is being performed over time.

- **Optimizer**: This agent is responsible for organizing jobs and operations in a manner that respects system constraints while optimizing performance metrics such as schedule adherence and the degree of change from the existing plan. Its state variables include the selected scheduling strategy, problem-specific information (e.g., processing time matrices, fixed operations, machine availability, precedence list and due dates), and the hyperparameters of the solver employed. Additionally, it maintains a record of the current solution and an up-to-date value of the objective function.

**Table 2** Entities and state variables information

| Entity | Variable name | Summary | Values | Units |
|---|---|---|---|---|
| Job generator | Generated jobs | Generated jobs until the actual time interval | N | List |
| Job | Priority | Priority of sequencing the job | 0-100 | % |
| Operations | Assignation | The machine where it is assigned | 1-M<br>Unassigned | Numeric |
| | Operation state | Actual state of operation in the current iteration | Under development<br>Finished | Option |
| | Processing time | Processing time of the operation in the current machine | 0-T | Time units |
| | Machine changes | Count of machine changes in the operations plan | 0-inf | Units |
| | Init time changes | Time offsets in the operations plan | 0-inf | Time units |
| Machines | Plan | Operations assigned to the machine in processing order | (Operation, Time) | List of tuples |
| | Completed operations | Operations completed by machine | Operation objects | List |
| Optimizer | Type | Optimization strategy to solve the problem | MILP<br>MA<br>DR | Option |
| | Hyperparameters | Parameters for the problem and solver | Problem and environment data | List |
| | Current solution | Schedule of activities | (Job, Operation, Init, End) | Tuple |
| | Objective function | Objective value metrics | 0 - inf | Dimensionless |

### 3.2.3 Process summary and scheduling

At the beginning of the simulation, a set of machines is initialized to perform the processing tasks, a planning horizon is defined, and an initial set of production orders is generated to represent a pre-existing schedule. At each simulation step (i.e., time interval), a decision-making sequence is executed as illustrated in Figure 1. The processing order of machines and jobs does not affect the outcome; thus, they are evaluated in ordinal fashion. The following section outlines the sequence of steps followed throughout the simulation.

1. **Job creation**: At each time step, the job generator evaluates whether to create a set of jobs based on a Poisson distribution with a $\lambda$ job arrival rate. For each job, its priority, estimated delivery time, and number of operations are randomly determined. Subsequently, the respective operations are generated for each job, including the machines on which they can be processed and their estimated processing times. In addition, information is generated on the routes to be followed by each operation and the main route. These jobs and operations are then made available.
2. **Optimization criterion**: At each time step, the optimization criterion is evaluated to determine whether it has been met. This criterion is based on whether "t" time intervals have passed or if a set of orders requires processing (i.e., if their priority value is 100%). If the criterion is not met, the process proceeds to step 4.

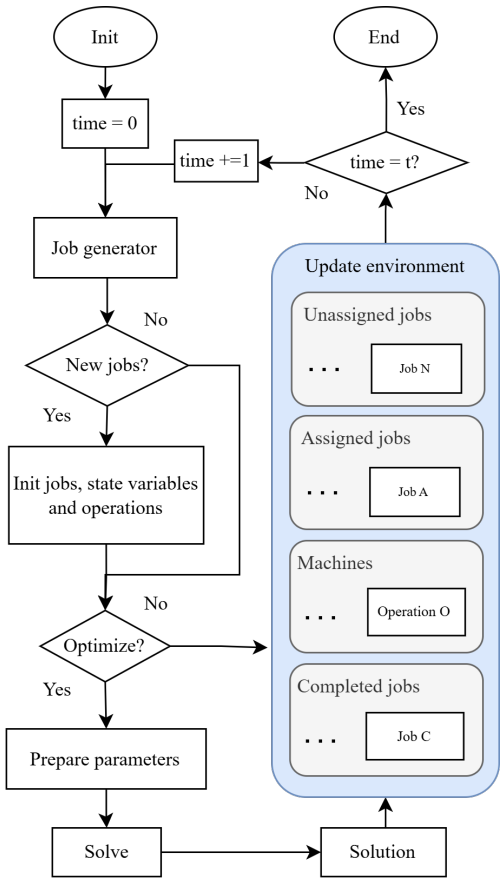



**Fig. 1** Agent-Based Modeling (ABM) process summary

3. **Optimization**: When optimizing the schedule, information from the environment is gathered, including the current plan, jobs in progress, unscheduled orders, and the status of the machines. This data is organized so that a specialized program, the 'solver,' can identify the best solution. Additionally, environment parameters are adjusted to fine-tune the solver,' and ultimately, the program optimizes the schedule based on all the collected information. A new schedule is generated, along with its corresponding objective function.
4. **Environment update**: The agents are updated based on the results of the optimization process, and at every timestep. The status of the machines changes once the operations scheduled have finished and when new operations are assigned. In this way, their planning list is updated along with its operations. When all operations of a job have been processed is added to the set of completed jobs. On the other hand, if a job has not yet been assigned, its priority increases over time. This mechanism ensures that, if it needs to be assigned quickly to meet estimated delivery times, the optimizer can react accordingly to fulfill the objectives.

### 3.2.4 Design concepts

In this section, we are going to explain the design concepts used in the Agent-Based Modeling (ABM) framework.

***Basic principles***

This model is built on the principle that production scheduling in manufacturing environments must dynamically adapt to the arrival of new orders without disrupting ongoing operations or compromising customer delivery promises. This challenge is particularly relevant in industries where demand is uncertain, and decisions must be made based on constantly changing conditions. The system is composed of several agents that represent key components of the production process: demand patterns (which capture the stochastic nature of manufacturing orders), machines (including their availability and capacity), and scheduling control (an agent responsible for allocating jobs to machines and sequencing operations).

The scheduling decisions, such as job allocation and operation sequencing, can be made using exact methods when the problem is computationally manageable or heuristics when the complexity of the system demands approximate solutions in reasonable execution times. The model adheres to optimization principles in dynamic environments, implementing strategies like predictive-reactive scheduling and real-time optimization to improve performance metrics, particularly schedule stability and delivery adherence.

***Objectives***

The model is designed to achieve multiple objectives, weighted according to the production strategy. Performance is evaluated at two levels: detailed operation scheduling, including start times and machine assignments, and global indicators such as maximum job completion time and schedule stability. The key objectives are: (i) minimizing operation idle time by starting tasks as early as possible (earliness), (ii) ensuring timely order fulfillment by reducing delays, measured as the difference between estimated and actual job completion dates (tardiness), and (iii) maintaining scheduling stability, which quantifies changes from previous decisions. Together, these objectives balance smooth operational adjustments with timely order delivery, both critical for overall system effectiveness.

***Adaptation***

The system is designed to dynamically adjust to new orders or disruptions. Optimization algorithms are triggered periodically or in response to critical events, applying pre-established logic to readjust the current schedule. The goal is to optimize key performance indicators while orders' priorities demand it.

***Emergence***

The global behavior of the system, including compliance with deadlines and performance metrics, emerges from the interaction between agents and the production

environment. Specifically, patterns like machine occupancy, delay accumulation, bottleneck formation, and waiting times arise from factors such as the stochastic arrival of new orders, job allocation, and the ongoing prioritization and rescheduling decisions.

While the model defines explicit rules for operation assignments (via exact or heuristic algorithms), the efficiency effects are not directly programmed but emerge as a result of the interaction between demand variability, finite resource capacity, and decision-making logic. These emergent phenomena are crucial for evaluating different scheduling strategies and understanding how small changes in allocation logic can significantly affect system stability and performance.

***Prediction***

The model's learning process includes predicting pessimistic assignment times based on current planning conditions. These predictions help update the priority of orders, providing a forward-looking mechanism to prioritize orders that are at risk of missing deadlines, further enhancing the system's ability to adapt.

***Sensing***

The system uses dynamic sensing to continually update scheduling decisions. Each job order tracks critical variables such as proximity to the delivery deadline. The solver continuously monitors the schedule and status of operations to determine if rescheduling or postponing jobs is necessary.

***Interaction***

The communication among agents in the system is event-driven and regulates the updating of the schedule. At each time interval, the solver queries the system's state and the priority of work orders, detecting important changes in new orders, and accumulated delays. If necessary, the solver processes these updates to optimize operation sequencing and reassign resources. Work orders also indirectly interact by competing for limited resources, which can lead to bottlenecks and machine congestion. These interactions affect the overall efficiency and flow of the system.

***Stochasticity***

The model introduces stochastic elements in three key areas: First, each operation has a variable execution time, modeled using probability distributions to account for uncertainty in machine performance. Second, variability exists in the number of machines capable of processing a given operation, simulating dynamic constraints on resource availability. Finally, new orders arrive based on a stochastic process, with delivery dates assigned with some degree of randomness, reflecting uncertainty in customer commitments.

***Observation***

The system's performance is evaluated by analyzing the objective functions over time, alongside agent-level variables such as order priority evolution and the solver's reaction

capability. To facilitate interpretation, simulation-time Gantt charts are used to visualize how the schedule evolves and to detect emerging patterns in resource allocation. This visualization aids in understanding the system's dynamics and performance.

### 3.2.5 Initialization

Upon initializing the model, the following agents and system parameters are created:

- A set of $K$ machines is instantiated, each identified by a unique indicator.
- An initial set of jobs is generated with a Poisson distribution with $\lambda$ job arrival rate. Each job consists of a number of operations between $\frac{o_{max}}{2}$ and $o_{max}$.
- Each operation is randomly assigned to $k$ machines from the available $K$. A processing time is then assigned to each operation, drawn randomly from a uniform distribution between 1 and $dur_{max}$.
- The time horizon of $T$ time intervals is defined for the scheduling process.
- A solver is instantiated based on the specified method, and an initial scheduling is executed based on the defined hyperparameters:
  - Exact Methods: Minimum GAP and maximum execution time must be specified.
  - Heuristic methods: Parameters include population size, stopping criteria, among others.

## 3.3 Optimization approaches

This section presents three optimization approaches for the DFJSSP. First, an MILP formulation is introduced. Then, two heuristic methods are described: a greedy-based DR and a MA.

### 3.3.1 MILP formulation

To address the DFJSSP, we build upon the formulation proposed by Dauzère-Pérès et al. (2024). In this model, the decision variable $t_i$ represents the start time of operation $i \in O$. Two binary variables are defined: $\alpha_{k,i}$, which takes the value 1 if operation $i \in O$ is assigned to machine $k \in R_i$ and 0 otherwise; and $\beta_{ii'}$, which equals 1 if operation $i$ is scheduled before operation $i'$, and 0 otherwise.

We extend this formulation by incorporating additional features commonly found in dynamic scheduling literature, such as schedule stability, due date adherence, and nonlinear precedence constraints. To model these aspects, we introduce three additional variables: $e_i$, representing the positive delay of operation $i$ with respect to its due date $dd_i$; $v_i$, which measures the number of time intervals by which the start time of operation $i \in F$ deviates from its previously assigned value $fi_i$; and $m_i$, a indicator that measures if $i \in F$ has been reassigned to a different machine compared to the original plan $fm_i$.

In the proposed formulation, Equation (1) defines a multi-objective function expressed as a weighted sum scaled by a weight factor $w$. The objective aims to minimize three key aspects: the total tardiness of operations with respect to their due dates, the schedule instability—quantified as the sum of deviations in start times and machine assignments relative to the previous schedule—and a regularization term that

$$\min \sum_{i \in O} t_i + w \cdot \left( \sum_{i \in F} (v_i + m_i) + \sum_{i \in O} e_i \right) \tag{1}$$

$$\sum_{k \in R_i} \alpha_{k,i} = 1 \qquad \forall\, i \in O \tag{2}$$

$$t_i \geq t_j + \sum_{k \in R_j} p_{k,j} \cdot \alpha_{k,j} \qquad \forall\, j \in pr(i),\ i \in O \tag{3}$$

$$t_i \geq t_{i'} + p_{k,i'} - (2 - \alpha_{k,i} - \alpha_{k,i'} + \beta_{i,i'}) \cdot H \qquad \forall\, i \neq i' \in O, k \in R_i \cap R_{i'} \tag{4}$$

$$t_{i'} \geq t_i + p_{k,i} - (3 - \alpha_{k,i} - \alpha_{k,i'} - \beta_{i,i'}) \cdot H \qquad \forall\, i \neq i' \in O, k \in R_i \cap R_{i'} \tag{5}$$

$$\alpha_{fm_i,i} = 1 \qquad \forall\, i \in S \tag{6}$$

$$t_i = fi_i \qquad \forall\, i \in S \tag{7}$$

$$e_i \geq \left( t_i + \sum_{k \in R_i} p_{k,i} \cdot \alpha_{k,i} \right) - dd_i \qquad \forall\, i \in O \tag{8}$$

$$e_i \geq 0 \qquad \forall\, i \in O \tag{9}$$

$$v_i \geq t_i - fi_i \qquad \forall\, i \in F \tag{10}$$

$$v_i \geq -(t_i - fi_i) \qquad \forall\, i \in F \tag{11}$$

$$m_i \geq 1 - \alpha_{fm_i,i} \qquad \forall\, i \in F \tag{12}$$

$$\alpha_{k,i} \in \{0, 1\} \qquad \forall\, i \in O,\ k \in R_i \tag{13}$$

$$\beta_{i,i'} \in \{0, 1\} \qquad \forall\, (i, i') \in O \times O \tag{14}$$

$$t_i \geq 0 \qquad \forall\, i \in O \tag{15}$$

minimizes the sum of start times. The latter encourages early initiation of operations or earliness, which enhances the comparability of algorithmic performance in scenarios involving the dynamic arrival of new jobs.

Constraints (2) guarantee that each operation is assigned to at most one eligible machine. Constraints (3) enforce the precedence relationships among operations within the same job, supporting multiple precedences by allowing each operation $i$ to be preceded by a set of operations denoted by $pr(i)$. Constraints (4) and (5) prevent the overlap of operations assigned to the same machine, using a sufficiently large constant $H$ to linearize the disjunctive constraints. Constraints (6) and (7) are used to fix the start time and machine assignment of operations currently in progress, ensuring consistency with the partial schedule at the time of optimization. Constraints (8) and (9) model the tardiness of operations as the positive difference between their completion times and their due dates, allowing the formulation to penalize jobs with multiple delayed operations more heavily. Constraints (10) and (11) capture the deviation in start time by computing the absolute difference between the current and previous schedules, thus quantifying temporal instability. Constraints (12) identify whether an operation has been reassigned to a different machine compared to the original plan,

measuring structural instability. Finally, Constraints (13) to (15) define the binary domain for the decision variables used in the model.

### 3.3.2 Dispatching rule

This heuristic, depicted in Algorithm 1, follows an earliest completion time strategy, where operations are scheduled on the machine that allows them to finish as early as possible. The process begins by sorting the jobs according to their due dates, establishing a priority order for scheduling. Operations that are currently in execution—referred to as fixed operations—are first identified and directly inserted into the schedule to initialize the schedule for each machine and preserve their original timing.

**Algorithm 1** Priority-based Insertion Heuristic

```
1:  Input: Problem data
2:  Sort jobs by ascending due dates
3:  Initialize schedule for each machine
4:  Initialize precedence time counter
5:  for each fixed operation do
6:      Add operation to the machine at the specified time
7:      Update precedence time counter
8:  end for
9:  for each prioritized not scheduled job j do
10:     for each operation o of job j do
11:         Define incumbent
12:         for each feasible machine k for operation o do
13:             Find a feasible insertion time window
14:             if operation finish time < incumbent then
15:                 Update incumbent
16:             end if
17:         end for
18:         Insert operation o in best machine and position
19:         Update precedence time counter
20:     end for
21: end for
22: Output: solution formatted
```

A precedence time counter is initialized to keep track of the earliest feasible start time for each operation, based on the completion times of its immediate predecessors. The heuristic then proceeds to schedule the remaining (non-fixed) operations following the priority of their corresponding jobs. For each operation $i$, the eligible machines in $R_i$ are evaluated to find the earliest feasible start time that meets the following criteria: precedence constraints are respected, all predecessors are completed before the operation begins, and a non-overlapping time window exists on the machine. Among all valid options, the operation is scheduled at the machine and time that results in the minimum completion time. After each assignment, the precedence counters of the

successor operations are updated to reflect the new scheduling dependencies. Once all operations have been scheduled, the algorithm returns a complete solution matrix containing the machine assignments, start times, and finish times for every operation and the associated objective function value.

### 3.3.3 Memetic algorithm

We developed a memetic algorithm that achieves a critical balance between exploration and exploitation. By combining the global search capabilities of evolutionary algorithms with the intensification strengths of local search, this hybrid approach enables an effective traversal of the solution space. It ensures both diversification across the landscape and intensification in promising areas, addressing the challenges of the DFJSSP through the framework outlined in Algorithm 2. This routine is executed until one of two stopping criteria is met: either a predefined number of generations occurs without improvement, or a specified time limit is reached. Each phase of the algorithm is described in detail below.

**Algorithm 2** Memetic Algorithm for the DFJSSP

**Input:** Problem data, Algorithm hyperparameters
Initialize population $\mathcal{P}$
Evaluate fitness for the initial population
Find best fitness value $f_{\text{inc}}$ of the solution `best_sol`
Set counter `no_improvement` $\leftarrow 0$
**while** elapsed time $< t_{max}$ **do**
    **if** `no_improvement` $> gen_{max}$ **then**
        **break**
    **end if**
    Apply selection procedure to $\mathcal{P}$
    Recombine-mutate $\mathcal{P}$ to generate the offspring $\mathcal{P}'$
    Apply VND to $\mathcal{P}'$ given a probability
    Evaluate $\mathcal{P}'$ fitness values and update $\mathcal{P}$
    **if** a new solution fitness improves $f_{\text{inc}}$ **then**
        Update `best_sol` and $f_{\text{inc}}$
        `no_improvement` $\leftarrow 0$
    **else**
        Increment `no_improvement`
    **end if**
**end while**
**Output:** `best_sol` formatted

#### *Encoding*

To represent a solution, we use a two-vector encoding scheme. The first vector defines the operation sequence for each job, ensuring that job-specific precedence constraints are respected. The latter only considers eligible machines for each operation.

Figure 2 illustrates an example of a feasible solution for a problem instance with three jobs, nine operations, and three machines. The first job consists of three operations located at positions 0, 1, and 2, which are executed in the global order 5, 2, and 8, respectively. These operations follow a non-linear route: the operations at positions 0 and 1 can be performed concurrently, and both are predecessors of the operation at position 2. For the second job, there are four operations. The first operation (position 3) is followed by two operations (positions 4 and 5) that can be executed in parallel, which then converge into the final operation (position 6). A feasible global sequence that respects all precedence constraints is 1, 6, 3, 7, ensuring that the initial and final operations are correctly executed as the first and last in the sequence. For the third job, there are only two sequential operations, making the precedence analysis straightforward.

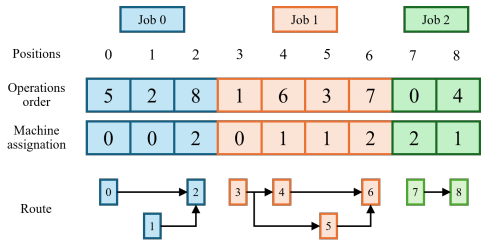



**Fig. 2** Solution encoding for MA

### *Initialization*

To generate each solution for the initial population, we first create a vector whose length corresponds to the total number of operations. Each operation is randomly assigned to one of its eligible machines. The vector is then shuffled to introduce randomness in the operation sequence. However, this randomization may result in infeasibilities due to violated precedence constraints, as some operations may appear before their required predecessors.

To address this issue, a schedule correction mechanism is applied to each solution in the population. This procedure traverses the precedence list in reverse order, identifying any violations where an operation is scheduled before one of its predecessors. When such a violation is detected, the two operations are swapped. This process is repeated iteratively until all precedence constraints are satisfied.

This procedure is illustrated in Figure 3 using the same example as above, with three jobs, nine operations, and three machines, starting from an infeasible sequence. In this initial sequence, each job violates its internal precedence constraints. For instance, in the first job, the operation at position 0 is scheduled in global order 8, which occurs after the operation at position 2, executed in order 5. However, the latter operation has the operations at positions 0 and 1 as predecessors, rendering the sequence infeasible. For the second job, operation 3 must precede operation 4 according to its precedence relation. In the initial sequence, however, operation 3 is scheduled at global order 6 while operation 4 occurs at order 1, requiring both operations to be swapped to restore

feasibility. Finally, the third job has its operations executed in reverse order: the last operation is performed first, and the first operation is performed last, which also violates precedence constraints. After applying the correction procedure, the solution is transformed into the feasible sequence presented in the previous example.

Additionally, for operations that are currently being executed (i.e., already in progress), the assigned machine is forcibly maintained, even if it differs from the machine previously chosen during the random assignment.

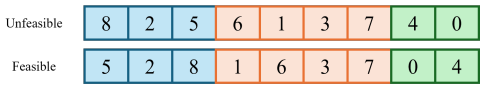



**Fig. 3** Fix solution procedure for operations order list

### *Evaluation*

To evaluate the objective function, each solution in the population must first be decoded. This process begins by setting each machine at their initial stage with only fixed operations assigned. Operations are then assigned sequentially, following the order defined in the first vector, to their corresponding machines specified in the second vector. During this process, precedence constraints are enforced by ensuring that an operation can only be scheduled after the completion of all its predecessors. For example, suppose an operation is assigned to machine 1 and its preceding operation finishes at time unit 5. In that case, it can only be scheduled in the next available time slot on machine 1 that can accommodate its processing time, starting no earlier than time 5.

Once the solution has been decoded, each machine's schedule is traversed, and for every operation, the following metrics are computed: start time, tardiness concerning its due date, deviation from its start time in the previous schedule, and the number of machine changes. These components are then weighted accordingly and combined to compute the overall fitness value, which is stored for the solution.

### *Selection*

We implemented a roulette-wheel selection mechanism that assigns higher selection probabilities to solutions with lower fitness values. Specifically, the inverse of each fitness value is computed and then normalized to obtain the cumulative selection probabilities. Parents are randomly sampled according to the cumulative distribution derived from these probabilities, so that better solutions have a proportionally higher chance of being chosen.

### *Recombination*

In the recombination process, offspring are generated by applying Order Crossover (OX) to the operation sequence and direct recombination to the machine assignments. For each selected pair of parents, two crossover points are chosen.

The offspring's operation sequence is constructed by inserting the middle segment from one parent into the corresponding position of the other, and then filling the

remaining positions with operations from the original parent in their relative order, avoiding duplicates.

Machine assignments are recombined by segment, aligned with the crossover points. After recombination, each offspring may undergo mutation according to a predefined probability. Subsequently, a feasibility correction procedure—identical to that used during population initialization—is applied to ensure that all precedence constraints are satisfied. The validated offspring are then added to the new population.

***Mutation***

The mutation operator introduces additional diversity into the population by probabilistically altering parts of the solution. For each offspring, a mutation is applied according to a predefined probability. When triggered, the algorithm randomly decides whether to modify the operation sequence or the machine assignment.

If the operation sequence is selected, two positions corresponding to operations from different jobs are randomly chosen, and one of several neighborhood-based perturbations is applied, each selected with equal probability. These include:

- Swap: exchanges the positions.
- Inversion: reverses the order of all operations between the selected positions.
- Insertion: removes one operation from its position and reinserts it into another, shifting the remaining elements.
- Shuffle Segment: randomly reorders the operations within the selected segment, breaking local patterns.

If the machine assignment is selected, an operation with multiple eligible machines is chosen and reassigned randomly to a different valid machine, ensuring feasibility.

***Variable Neighborhood Descent (VND)***

To further refine the search and improve local optimality, a VND algorithm is applied. This VND is conditionally activated for each offspring generated through recombination and mutation, based on a predetermined probability. The solution is iteratively improved by exploring a set of predefined neighborhoods, including:

- Intra-job operation swaps respecting precedence constraints.
- Inter-job operation swaps with full schedule repair.
- Feasible operation insertions within and across jobs.
- Machine reassignments for flexible operations.

The current solution is evaluated using the weighted fitness function defined in previous sections, and the search proceeds through the neighborhood structures in a deterministic order. Upon improvement, the process restarts from the first neighborhood; otherwise, it advances to the next. The descent terminates when no further improvement is found across all neighborhoods.

***Population updating***

After generating and evaluating the offspring, the population is entirely replaced by the new set of solutions. This generational replacement strategy maintains algorithmic

simplicity while allowing continuous renewal of the search space. Although no explicit elitism is implemented, the best solution found so far is preserved externally as the incumbent and is updated whenever an offspring achieves a lower fitness value. This ensures that the global best solution is never lost across generations, while maintaining diversity and avoiding premature convergence.

# 4 Experimental setup

To assess the effectiveness of the proposed DJSSP environment and the associated decision-making strategies, we designed a series of simulation-based experiments implemented in Python 3.9.5. The simulation models a production system in which jobs arrive dynamically over time and their operations must be scheduled on a set of available machines under real-time constraints. For the MILP model, we used the Python API of the Gurobi Optimizer (version 11.0.1), running on a machine equipped with an Intel(R) Xeon(R) E5-2670 v3 @ 2.30 GHz 16-core CPU, 377 GB DDR4 memory, and 1 TB HDD storage.

Scenario selection is a crucial step in this setup, as the behavior of a dynamic scheduling environment is highly sensitive to parameters such as order arrival rates, processing times, and overall system load. Inadequate configurations may lead either to trivial instances with underutilized resources or to overloaded systems where queues grow uncontrollably, preventing fair comparison across methods. Consequently, identifying stable and representative scenarios is essential to ensure meaningful, controlled, and reproducible evaluation of the proposed approaches

## 4.1 Scenario design and selection

To evaluate the performance of the proposed methods under uncertainty generated by the arrival of new orders, we consider two types of experimental scenarios. Static scenarios are used to compare all methods under exactly the same conditions, ensuring that the outcome is not influenced by past decisions. In contrast, dynamic scenarios, which align with the main objective of this work, evaluate the cumulative effect of successive scheduling decisions. In this setting, direct one-to-one comparisons at specific moments are not meaningful, as each decision alters the system state and affects all subsequent choices.

Both type of scenarios require a consistent specification of the key parameters of the production environment: the number of machines, the maximum number of operations per job (with a minimum of two operations to define a valid product route), the maximum processing time per operation (with the shortest duration corresponding to a single time interval), and the order-arrival rate, expressed as a fixed parameter. Once these bounds are defined, the specific characteristics of each job and its operations—such as the actual number of operations and its processing times—are generated using discrete uniform distributions over their respective minimum and maximum values. In addition, job arrivals are modeled as a Poisson process, meaning that the number of arrivals within each time interval follows a Poisson distribution

parameterized by the specified arrival rate. The objective is to determine a stable configuration that prevents unbounded queue growth while ensuring a realistic workload for all evaluated methods.

To identify suitable values for these parameters, we first conduct a sensitivity analysis using the DR presented in Section 3.3.2, chosen for its simplicity and interpretability. Following previous studies, we fix the system to six machines, as this level of resource availability is commonly used to represent an adequate level of complexity in FJSSP research (Li et al. 2021). On the other hand, increasing the number of machines without increasing the job load in unrelated machine settings, such as the FJSSP, typically relaxes the scheduling constraints, allowing solution methods to navigate the search space more effectively (Uribe et al. 2025).

The simulation horizon is set to 1000 time intervals, allowing us to observe the system's long-term behavior. A rescheduling frequency of every 10 intervals is adopted. For each configuration, simulations were run with 10 independent seeds. The following calibration parameters were varied:

- Order arrival rate (Poisson $\lambda$): 0.85 and 0.95, following the workload levels reported in Xu et al. (2025).
- Maximum number of operations per job: 4, 5, and 6, selected to reflect the structure of small- and medium-sized benchmark instances widely used in the FJSSP literature (Brandimarte 1993; Kacem et al. 2002; Fattahi et al. 2007; Behnke and Geiger 2012).
- Maximum operation duration: 3 or 4 time units, determined after calibrating the two previous factors to ensure stable system behavior without excessive congestion.

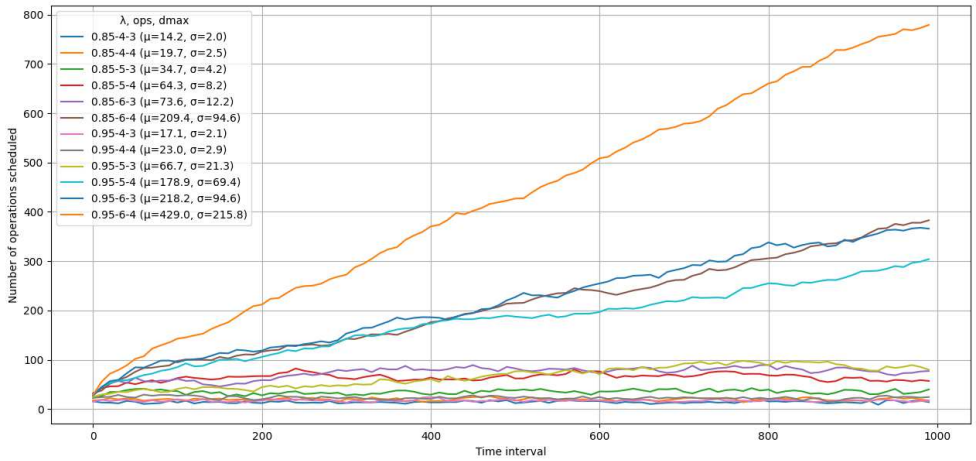



**Fig. 4** Behavior of the system under different parameter values

Figure 4 shows the evolution of the number of scheduled operations observed across all configurations. For each configuration, we computed the mean and standard deviation of the number of scheduled operations over the entire simulation horizon, and

from these values we derived the coefficient of variation. We then selected the configurations with the lowest coefficient of variation. The most stable behavior occurs with a Poisson arrival rate $\lambda$ of 0.85, a limit of 5 operations per job (permitting multiple routes and precedence relations), and a maximum duration of 3 time units, producing an average of 34.7 scheduled operations per rescheduling interval with a standard deviation of 4.2.

## 4.2 Scalability evaluation with static instances

To evaluate the scalability of the proposed scheduling methods, we developed a static instance generator aligned with the principles of the agent-based simulation environment. The generator produces FJSSP instances that include partially completed jobs as well as newly released jobs that must be scheduled, enabling the study of methods under realistic intermediate system states. Each instance comprises a user-specified number of jobs, with each job consisting of between two and five operations. For each operation, a subset of eligible machines is randomly selected according to a discrete uniform distribution over the range from 1 to the total number of available machines. Similarly, the processing time for each operation–machine pair is then drawn from a discrete uniform distribution between 1 and 3 time units.

To represent ongoing production activities, a random subset of jobs was marked as 'in progress'. For each, operations were assigned to randomly selected machines according to a discrete uniform distribution over the set of eligible machines and scheduled as early as possible, respecting both precedence constraints and machines availability based on previously allocated operations. This procedure produces a partial schedule with machine assignments and start times for ongoing activities. Job due dates are derived from an upper-bound estimate of their total processing time, computed using the duration of each operation on the machine on which it requires the longest processing time: already-started jobs receive tighter windows (30–70%), while the remaining jobs are assigned wider ranges (70–130%), capturing different urgency levels. Job structures incorporate non-linear precedence patterns by allowing parallel operation routes, reflecting non-trivial industrial workflows. Operation indices allow to model precedences while avoiding cyclic relationships.

Four levels were considered for the number of jobs: 5, 15, 30, and 50. To test the benefit of warm-starting, the DR heuristic was used to initialize both the MA and the MILP approaches. A 5-minute time limit was imposed, balancing solution quality and computational feasibility in fast rescheduling contexts. Within this horizon, the MILP solver achieved optimality gaps below 0.05% for most cases, while the MA completed sufficient evolutionary and local-search steps to reach comparable results.

Based on preliminary tests, the MA was configured with a population of 20 individuals, a mutation rate of 30%, a VND activation probability of 5%, and a stopping criterion of 100 generations without improvement. Larger populations were discarded as they reduced the number of generations achievable within the time limit due to the cost of VND. Each configuration was evaluated using 10 random seeds to ensure variability and reproducibility.

**Table 3** Impact of job increase on solution methods for static configurations

| Method | GAP | | | | Time (s) | | | |
|---|---|---|---|---|---|---|---|---|
| | **5** | **15** | **30** | **50** | **5** | **15** | **30** | **50** |
| **DR** | 30,25% | 82,07% | 98,56% | 84,42% | 0,00 | 0,00 | 0,00 | 0,00 |
| **HMA** | 4,72% | 72,21% | 96,65% | 83,29% | 14,05 | 300,22 | 301,42 | 304,70 |
| **HMILP** | 0,00% | 31,98% | 59,80% | 75,04% | 2,61 | 301,00 | 304,30 | 310,22 |
| **MA** | 6,83% | 72,87% | 105,83% | 101,53% | 14,66 | 300,27 | 301,63 | 304,23 |
| **MILP** | 0,00% | 29,02% | 71,85% | - | 3,47 | 300,93 | 303,46 | 307,61 |

### 4.3 Dynamic experiments under successive decision-making

For the dynamic experiments, we adopted the stable configuration identified in the scenario selection stage (see Section 4.1). MA and MILP methods preserve the same settings used in the static analysis. As previously observed, system behavior stabilizes around time interval 100. Therefore, a warm-up period of 100 intervals is introduced, during which the DR generates all schedules. After this period, the evaluated methods operate over an additional 100 intervals.

To capture variability under uncertainty, each scenario is executed with 10 random seeds. From interval 100 onward, all methods begin from the same system state for a given seed, ensuring identical order arrivals and job characteristics. However, due to every decision can alter the subsequent system configuration, the methods inevitably face different states over time. Consequently, point-wise comparisons are not meaningful; evaluation focuses instead on their overall performance across the entire dynamic horizon.

To further examine scalability and observe algorithmic behavior under challenging conditions, we include a more challenging setup in which all operations have a duration of one time unit, jobs may contain up to 10 operations, and the arrival rate is increased to $\lambda = 0.95$. This setting naturally generates congestion and steadily growing workloads, providing insight into how each method responds under tightening operational pressure.

## 5 Results

### 5.1 Performance on static instances

To summarize the results on static scenarios, we report the average gap percentage and execution time (in seconds) computed over 10 random instances. It is worth noting, however, that the number of operations may vary slightly between instances due to the use of a random distributions in the generation process. Table 3 presents the performance of all evaluated methods in terms of these two metrics. Each row corresponds to a different solution method, while the columns report the average gap and execution time for instances with increasing numbers of jobs. Thus, the table allows for a direct comparison of both solution quality and computational efficiency across problem sizes. In this context, HMA and HMILP refer to the hybrid approaches—i.e., those initialized with the deterministic DR solution—for the MA and the MILP model, respectively.

The reported gap values were computed based on the best bound provided by Gurobi, ensuring a consistent measure of optimality across all solution methods. Additionally, experiments were conducted using multiple weight configurations ($w$), and the results revealed that performance trends remained consistent across all tested values. Consequently, $w = 1$ was selected as the representative configuration for the static analysis.

The DR progressively closes its gap difference with respect to the other methods as the problem size increases. In contrast, the MA experiences a clear degradation in performance as the number of jobs increases. This is mainly because the high computational cost of the VND neighborhoods severely limits the number of evolutionary and local-search iterations it can complete within the time limit. As problem size grows, the algorithm performs progressively fewer refinement steps, which restricts its ability to effectively exploit and improve solutions.

The hybrid approaches, HMA and HMILP, benefit substantially from being initialized with the DR method. In particular, HMA achieves better results than its standalone version because it starts from a meaningful baseline that is integrated into the evolutionary process, allowing the algorithm to leverage this initial structure during its search.

Similarly, HMILP exploits the initial feasible solution as an incumbent to strengthen the branch-and-bound process, allowing early pruning of suboptimal nodes and leading to smaller GAP values and faster convergence. Among all methods, HMILP demonstrates the best overall balance between solution quality and computational effort, whereas the pure MILP model faces scalability challenges, failing to obtain feasible solutions for the largest instances due to the exponential growth of the solution space.

## 5.2 Performance in dynamic conditions

### 5.2.1 Stable dynamic scenario

This section examines the performance of the evaluated methods under a stable dynamic scenario, in which new orders arrive at a controlled rate that does not overload or destabilize the production system.

After executing the simulation for each seed and method, we collected several performance indicators, excluding all jobs completed before the warm-up period. The recorded data include:

- The number of completed jobs and operations, defined as those fully processed and ready for delivery by the end of the planning horizon.
- Assigned jobs and operations, which were scheduled but not fully processed by the machines.
- Delayed jobs, a category that may overlap with the previous two and corresponds to jobs that were or will be delivered after their estimated due date.
- Moved operations, referring to tasks that were initially assigned to a specific time interval but were subsequently rescheduled to another.
- Transferred operations, defined as tasks initially planned on one machine but ultimately executed on a different one.

**Table 4** Number of jobs and operations at the end of the simulation (averaged across instances) for each base-state scenario: completed, assigned, delayed, rescheduled, or transferred

| Method | Jobs | | | Operations | | | |
|---|---|---|---|---|---|---|---|
| | C | A | D | C | A | M | T |
| **DR** | 85,3 | 8,6 | 63,5 | 299,0 | 31,4 | 49,4 | 28,4 |
| **HMA** | 86,3 | 7,6 | 40,1 | 302,7 | 27,7 | 20,1 | 24,0 |
| **HMILP** | 86,1 | 7,8 | 38,4 | 302,1 | 28,3 | 13,6 | 16,3 |
| **MA** | 86,3 | 7,6 | 41,6 | 302,7 | 27,7 | 19,1 | 24,0 |
| **MILP** | 86,2 | 7,7 | 39,8 | 302,4 | 28,0 | 14,2 | 14,2 |

C: completed, A: assigned, D: delayed, M: moved, T: transferred

A summary of these indicators is presented in Table 4. The results indicate that the number of completed jobs remains relatively stable across all methods, suggesting that each approach is similarly capable of processing the assigned workload within the given time horizon. Nonetheless, some minor variations can be observed. For instance, the DR method tends to complete slightly fewer jobs on average, leading to a higher accumulation of pending work on the machines, whereas the MA and HMA methods achieve marginally better throughput and resource utilization. In terms of delays, while the overall scheduling performance remains consistent, the DR method shows a higher proportion of delayed jobs (68%), whereas the HMILP approach performs best, meeting 59% of order deadlines.

The number of operations with temporal changes and machine reassignments further highlights the performance gap: exact methods yield the lowest values, followed by memetic approaches. The DR modifies the start time of 15% of operations and reassigns 9% of them, while the other methods alter on average only 5% and 6% of operations for the same metrics. These results are consistent with expectations: exact methods typically reach optimal solutions, with an average GAP of 0.02% across intervals for all instances, requiring 26.2s on average for MILP and 27.6s for HMILP. Memetic methods also exploit the available time effectively through evolutionary processes and local search, requiring 18.5s on average for MA and 15.7s for HMA.

These findings, however, should be interpreted with caution. While Table 4 illustrates the relative capability of the methods to manage job flow and prevent workload accumulation, this does not necessarily imply superior performance in the main metrics of interest (stability and accumulated delays over time). In other words, achieving a lower number of delayed jobs or modified operations does not preclude the presence of significant delays or substantial rescheduling actions. To properly evaluate the overall effectiveness of each approach, it is therefore necessary to analyze their impact on the cumulative target metrics at the final simulation snapshot. For this purpose, Table 5 summarizes the emergent patterns observed over all processed jobs at the end of the simulation horizon, using the same notation introduced for the MILP model in previous sections. It does not represent the objective function as such, but rather provides a snapshot of the cumulative compliance- and stability-related metrics across the simulation. To evaluate the average contribution of each component of the objective function, Table 6 is used instead, as it reports the average values computed at each

solver call across all instances, capturing the objective's components in a meaningful way during the dynamic execution.

**Table 5** Average values of emergent simulation indicators at the final snapshot (averaged across instances) for the base scenarios

| **Method** | $e_f$ | $v_f$ | $m_f$ |
|---|---|---|---|
| **DR** | 206,8 | 115,0 | 35,8 |
| **HMA** | 94,2 | 30,4 | 26,2 |
| **HMILP** | 84,9 | 23,1 | 17,4 |
| **MA** | 98,1 | 30,5 | 27,0 |
| **MILP** | 86,9 | 23,9 | 15,0 |

$e_f$: sum of all jobs delay
$v_f$: sum of all operations time movements
$m_f$: number of all machines transfers

**Table 6** Average objective-function values across simulations and instances throughout the execution for base scenario

| **Method** | $e$ | $v$ | $m$ | $t$ | $of$ |
|---|---|---|---|---|---|
| **DR** | 35,11 | 2,79 | 0,86 | 59,23 | 98,01 |
| **HMA** | 8,14 | 0,66 | 0,62 | 27,30 | 36,74 |
| **HMILP** | 7,09 | 0,48 | 0,40 | 26,20 | 34,19 |
| **MA** | 8,54 | 0,67 | 0,64 | 27,77 | 37,64 |
| **MILP** | 7,23 | 0,50 | 0,34 | 26,27 | 34,36 |

$e$: sum of delay
$v$: sum of time movements
$m$: sum of machines transfers
$t$: sum of earliness
$of$: objective function

Table 5 shows a consistent pattern: the DR produces the highest levels of delay and instability, whereas exact methods achieve the best outcomes. Specifically, the DR accumulates 144% more delay than the best method (HMILP) and induces up to 398% more reallocation of previously assigned operations. In comparison, memetic methods increase these values by only 13% and 32% on average, respectively. For machine reassignments, MILP achieves the best overall result, while HMILP, MA, HMA, and the DR exhibit increases of 16%, 80%, 75%, and 138%, respectively.

It is also important to evaluate not only the total accumulated delay and instability but also their relative intensity per affected job or operation. To this end, each metric was normalized by the number of jobs or operations involved. The results show that the exact and memetic approaches achieve similar average delay rates of approximately 2.27 time units per job, while the DR method exhibits a higher rate of 3.3. Trends for

other metrics, such as rescheduled operations and machine changes, are similar across methods.

Table 6 shows that, for this base configuration, the MILP-based approaches achieve the best overall objective values as well as the best performance in the individual objective components, closely followed by the MA methods, which also perform very well across all metrics. In contrast, the DR method exhibits the lowest performance, both in the overall objective and its constituent components.

Additionally, the agent-based simulation provides machine utilization metrics which, although not explicitly included in the objective function, offer valuable insight into the workload balance achieved by each method. In this regard, the DR method exhibits the highest imbalance, with a gap of 13.6 time units between the least and most utilized machines (measured by total active operating time). In contrast, the remaining methods show a much narrower average difference of about 6 time units. This suggests that the alternative approaches promote a more homogeneous distribution of workloads across machines, whereas the heuristic nature of the DR leads to less balanced utilization.

### 5.2.2 High-load dynamic scenarios

Table 7 presents the results for a high-load dynamic scenario, where the problem becomes increasingly complex as jobs arrive more frequently. In this configuration, the MILP failed to produce feasible solutions within the five-minute time limit and was excluded from the comparison. Overall performance deteriorates slightly for the memetic and hybrid approaches, narrowing the performance gap with the DR method. Although completion rates remain comparable across methods, around 92% of all jobs experience delays, clearly indicating that in such scenarios companies should reconsider how due dates are estimated unless increases in production capacity are possible. The DR method exhibits greater stability, as its decision logic yields consistent scheduling patterns under changing conditions. In contrast, the other methods introduce more variability, rescheduling over 80% of operations and reassigning more than half of them to different machines. It is also observed that, in this setup, the HMILP reports an average optimality gap of 35% (as indicated by Gurobi), while the memetic algorithms manage to execute on average only one generation and a single VND iteration. This limited search depth constrains their exploration capability and prevents the local search process from being effectively applied to more promising regions of the solution space.

Following the same approach as in previous experiments, Table 8 reports the cumulative behavior of all jobs throughout the simulation, averaged across instances, in terms of delay and scheduling stability. The results indicate that the memetic approaches accumulate higher delays and instability compared to the HMILP. Specifically, MA and HMA incur 57% and 30% more delay, respectively, than HMILP, while DR increases delay by only 6%. When analyzing scheduling stability, the DR maintains lower instability levels toward the end of the planning horizon. Similarly, Table 9 shows the average values of the objective metrics and the overall objective function at each solver call, computed across all instances.

**Table 7** Number of jobs and operations at the end of the simulation (averaged across instances) for each high-load-state scenario: completed, assigned, delayed, rescheduled, or transferred

| Method | Jobs | | | Operations | | | |
|---|---|---|---|---|---|---|---|
| | **C** | **A** | **D** | **C** | **A** | **M** | **T** |
| **DR** | 73,5 | 45,2 | 110,2 | 516,0 | 317,0 | 522,9 | 321,0 |
| **HMA** | 61,9 | 56,8 | 109,4 | 434,2 | 398,8 | 682,1 | 632,6 |
| **HMILP** | 78,4 | 40,3 | 109,3 | 550,9 | 282,1 | 737,1 | 418,8 |
| **MA** | 55,7 | 63,0 | 109,2 | 388,3 | 444,7 | 745,7 | 758,2 |

**Table 8** Average values of emergent simulation indicators at the final snapshot (averaged across instances) for the high-load scenarios

| **Method** | $e_f$ | $v_f$ | $m_f$ |
|---|---|---|---|
| **DR** | 3218,5 | 1831,9 | 506,7 |
| **HMA** | 3926,1 | 29126,1 | 4948,0 |
| **HMILP** | 3012,9 | 7800,4 | 921,3 |
| **MA** | 4750,4 | 56401,5 | 8257,0 |

**Table 9** Average objective-function values across simulations and instances throughout the execution for high-load scenario

| **Method** | $e$ | $v$ | $m$ | $t$ | $of$ |
|---|---|---|---|---|---|
| **DR** | 10102,69 | 22,00 | 6,16 | 6422,05 | 16552,91 |
| **HMA** | 8664,61 | 448,48 | 75,57 | 5079,10 | 14267,77 |
| **HMILP** | 8794,37 | 115,27 | 12,65 | 5588,92 | 14511,23 |
| **MA** | 7544,38 | 874,66 | 127,27 | 3908,32 | 12454,65 |

To better understand the evolution of these behaviors, Figure 5 depicts the variation of the objective function values throughout the simulation for a fixed seed, capturing each rescheduling decision rather than only the final snapshot. This dynamic analysis reveals that DR consistently attains higher objective function values at each decision step. In contrast, HMILP, HMA, and MA exhibit smoother, more homogeneous growth patterns, primarily driven by the continuous arrival of new jobs resulting from the accumulation of unprocessed orders under the bottleneck conditions of the hard scenario.

When comparing the results in Table 8 with the trends shown in Figure 5 and Table 9, some differences become evident. To better interpret these results, the Figure 6 decomposes the objective function into its individual components over the same simulation period (after the warm-up). The analysis shows that the overall increase in the objective function is mainly driven by delays and the sum of operation start times—terms in which DR obtains higher values. However, in the components associated with scheduling stability, such as machine reassignments and temporal shifts,

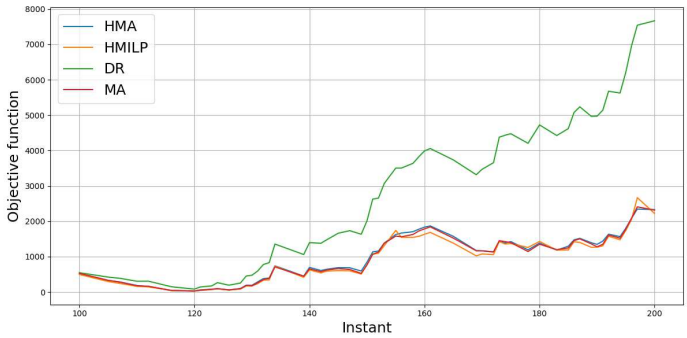



**Fig. 5** Objective function behavior for the different methods with seed fixed

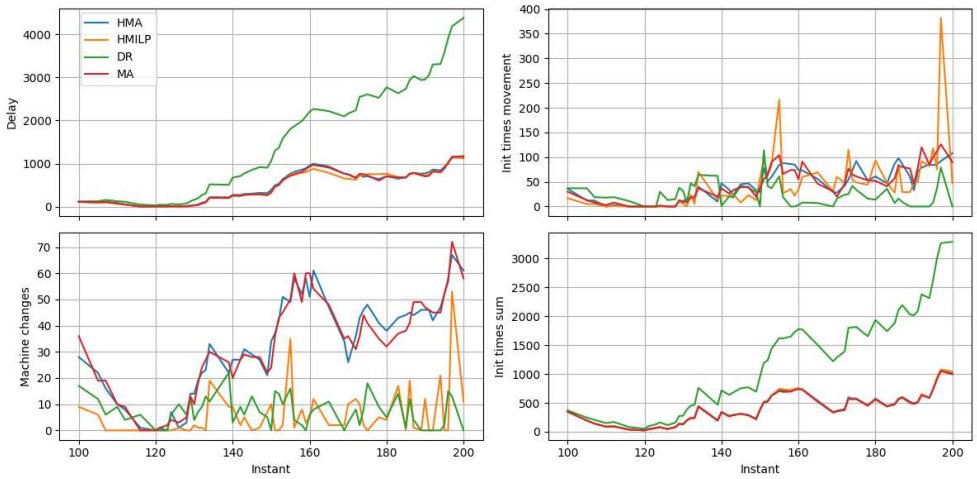



**Fig. 6** Performance of the disaggregated values of the objective function for the different methods with seed fixed

DR maintains lower values, reflecting the consistency of its decision logic. In contrast, HMILP, HMA, and MA improve the objective function more effectively as a whole, achieving lower global values largely influenced by the regularization term that promotes early processing (earliness).

The observed deterioration in the performance of the memetic approaches, as an emergent pattern at the end of the simulation, can be attributed to several factors. One key factor is the difference in delay granularity: while the objective function evaluates delays at the operation level, the final performance summaries report delays aggregated at the job level. Consequently, small operation-level variations can accumulate unevenly, whereas DR behaves differently, as its logic focuses on grouping the operations of each job closely together rather than exploring alternative sequencing patterns. Second, instability accumulates over successive rescheduling steps. Operations reassigned multiple times contribute repeatedly to the summary metrics, amplifying

instability values even when the objective function registers only incremental changes. Finally, in the hard scenario, the homogenization of processing times reduces variability among machines, weakening the advantage of evolutionary search mechanisms. Combined with the problem size, this limitation restricts the memetic algorithms' ability to effectively diversify and intensify the search.

# 6 Conclusions

This study emphasizes the importance of evaluating scheduling strategies in dynamic environments over extended planning horizons. We addressed the DFJSSP by explicitly modeling random job arrivals and non-linear precedence constraints. The proposed agent-based simulation framework proved effective in capturing emergent system behaviors and quantifying the cumulative impact of scheduling decisions. Beyond enabling benchmarking under uncertainty, the simulation also provided valuable insights into machine utilization and workload balance, two key factors for the overall efficiency of manufacturing systems.

Within this framework, we compared three representative strategies: a MILP model, a MA, and a priority DR. The results confirmed that scheduling decisions have significant and lasting effects on system performance, and that trade-offs between solution quality, stability, and responsiveness must be carefully considered in practice.

The experimental analysis demonstrated the superiority of MILP-based optimization in both static and dynamic settings. These methods not only minimized tardiness but also improved workload balance and schedule stability. In baseline scenarios, MILP achieved an average optimality gap of just 0.02%, and even under heavy machine saturation, the hybrid version initialized with a feasible solution remained competitive with gaps around 35%. However, scalability emerged as a critical limitation, since computational effort increased sharply with problem size.

MA algorithms demonstrated competitive performance across most evaluation metrics. The incorporation of feasible initializations contributed to reducing the optimality gap under the same computational budget. In the dynamic baseline scenario, their results were comparable to those of the exact methods, showing moderate increases in delay, temporal changes, and machine reassignments, while requiring only about 63% of the computation time. In more demanding configurations, however, the complexity of the local search mechanisms, together with the size of the problem instances, constrained their effectiveness under the selected parameter settings.

The DR provided near-instant solutions, showing consistent performance in terms of scheduling stability. However, its short-term decision logic limited its effectiveness regarding other objective metrics, leading to delays in up to 68% of jobs, along with workload accumulation and less balanced resource utilization compared with optimization-based methods. In static instances, it also exhibited larger optimality gaps relative to exact and memetic approaches. Nonetheless, under increased problem complexity—where exact methods often failed to produce feasible solutions—the DR remained a competitive alternative, delivering stable and immediate scheduling responses within the available computation time.

In summary, this study provides an integrated perspective on the performance of exact, metaheuristic, and heuristic scheduling strategies under dynamic and uncertain conditions. These findings emphasize the importance of balancing responsiveness, robustness, and long-term efficiency when designing scheduling strategies for real-world manufacturing systems.

Future research could extend the proposed framework by incorporating additional sources of uncertainty that more accurately reflect real manufacturing environments. In particular, subsequent studies may consider stochastic variations in operation processing times and machine availability. Additionally, integrating machine and operator calendars could enable the representation of maintenance schedules, shift patterns, and workforce constraints, providing a more realistic evaluation of scheduling strategies under limited resource availability. Another promising direction would be the exploration of proactive scheduling approaches aimed at enhancing schedule robustness rather than focusing solely on reactive performance. By embedding robustness-oriented objectives and uncertainty modeling within the agent-based simulation, future work could support the design and assessment of scheduling policies capable of sustaining high performance even under significant disruptions.

# Appendix A Abbreviations of Table 1

***ARV:*** Adaptive reference vector
***ATGA:*** Adaptive tripartite game theory-based genetic algorithm
***AES-IBCEA:*** Adaptive environment selection improved bi-population cooperative evolutionary algorithm
***ACML-BCEA:*** Adaptive clustering and Meta-Lamarckian learning-based bi-population cooperative evolutionary algorithm
***DRL:*** Deep reinforcement learning
***DQN-DEMA:*** Deep Q-network based dynamic efficient memetic algorithm
***DIFFormer:*** Diffusion-based transformer
***DEHH:*** Diversity-Enhanced Hyper-Heuristics
***D3QN:*** Dueling double deep Q-network
***GPHH:*** Genetic programming hyper-heuristic
***GA:*** Genetic algorithm
***GCLNTSO:*** Genetic chaos levy nonlinear tuna swarm optimization
***GRM:*** Green rescheduling method
***HMOEA:*** hybrid multi-objective evolutionary algorithm
***HABC:*** hybrid artificial bee colony algorithm
***HDQN:*** Hybrid deep Q-network
***IMA:*** Improved memetic algorithm
***INSGA-III:*** Improved non-dominated sorting genetic algorithm-III
***IDDQN:*** Improved double deep Q-network
***IHA:*** Improved Hungarian algorithm
***MILP:*** Mixed integer linear programming
***MCTS:*** Monte Carlo tree search
***MASC:*** Multi-agent scheduling chain

***MOALNS:*** Multi-objective adaptive large neighbourhood search
***MOTLBO:*** Multi-objective teacher learning-based optimization
***MTDF:*** Multi-type data-driven framework
***NSGA:*** Non-dominated sorting genetic algorithm
***PSO:*** Particle swarm optimization
***QLIDE:*** Q-learning improved differential evolution
***QNSGA-II:*** Q-Learning non-dominated sorting genetic algorithm-II
***Q-MOIA:*** Q-learning multi-objective immune algorithm
***QA:*** Quantum annealing
***TS:*** Tabu search
***TABC:*** Two-stage artificial bee colony algorithm
***VNS:*** Variable neighborhood search
***VWMI:*** Variable window multi-interval optimization

# References


Agnetis, A., Billaut, J.-C., Pinedo, M., Shabtay, D.: Fifty years of research in scheduling — theory and applications. European Journal of Operational Research (2025) https://doi.org/10.1016/j.ejor.2025.01.034

An, Y., Chen, X., Gao, K., Zhang, L., Li, Y., Zhao, Z.: A hybrid multi-objective evolutionary algorithm for solving an adaptive flexible job-shop rescheduling problem with real-time order acceptance and condition-based preventive maintenance. Expert Systems with Applications **212**, 118711 (2023) https://doi.org/10.1016/j.eswa.2022.118711

An, Y., Chen, X., Gao, K., Zhang, L., Li, Y., Zhao, Z.: Integrated optimization of real-time order acceptance and flexible job-shop rescheduling with multi-level imperfect maintenance constraints. Swarm and Evolutionary Computation **77**, 101243 (2023) https://doi.org/10.1016/j.swevo.2023.101243

An, Y., Zhao, Z., Gao, K., Dong, Y., Chen, X., Zhou, B.: A self-adaptive co-evolutionary algorithm for multi-objective flexible job-shop rescheduling problem with multi-phase processing speed selection, condition-based preventive maintenance and dynamic repairman assignment. Swarm and Evolutionary Computation **89**, 101643 (2024) https://doi.org/10.1016/j.swevo.2024.101643

Behnke, D., Geiger, M.J.: Test instances for the flexible job shop scheduling problem with work centers (2012) https://doi.org/10.24405/436

Burmeister, S.C., Guericke, D., Schryen, G.: A two-level approach for multi-objective flexible job shop scheduling and energy procurement. Cleaner Energy Systems **10**, 100178 (2025) https://doi.org/10.1016/j.cles.2025.100178

Brandimarte, P.: Routing and scheduling in a flexible job shop by tabu search. Ann. Oper. Res. **41**(3), 157–183 (1993) https://doi.org/10.5555/160231.160247

Chen, J.C., Chen, T.-L., Lee, Y.-H.: Simulation optimization for parcel hub scheduling problem in closed-loop sortation system with shortcuts. Simulation Modelling Practice and Theory **124**, 102728 (2023) https://doi.org/10.1016/j.simpat.2023.102728

Chen, R., Wu, B., Wang, H., Tong, H., Yan, F.: A q-learning based nsga-ii for dynamic flexible job shop scheduling with limited transportation resources. Swarm and Evolutionary Computation **90**, 101658 (2024) https://doi.org/10.1016/j.swevo.2024.101658

Ding, L., Luo, D., Mudassar, R., Yue, L., Meng, L.: A novel deep self-learning method for flexible job-shop scheduling problems with multiplicity: Deep reinforcement learning assisted the fluid master-apprentice evolutionary algorithm. Swarm and Evolutionary Computation **94**, 101907 (2025) https://doi.org/10.1016/j.swevo.2025.101907

Dauzère-Pérès, S., Ding, J., Shen, L., Tamssaouet, K.: The flexible job shop scheduling problem: A review. European Journal of Operational Research **314**(2), 409–432 (2024) https://doi.org/10.1016/j.ejor.2023.05.017

Echeverria, I., Murua, M., Santana, R.: Leveraging constraint programming in a deep learning approach for dynamically solving the flexible job-shop scheduling problem. Expert Systems with Applications **265**, 125895 (2025) https://doi.org/10.1016/j.eswa.2024.125895

Fattahi, P., Saidi Mehrabad, M., Jolai, F.: Mathematical modeling and heuristic approaches to flexible job shop scheduling problems. J. Intell. Manuf. **18**(3), 331–342 (2007) https://doi.org/10.1007/s10845-007-0026-8

Fan, C., Wang, W., Tian, J.: Flexible job shop scheduling with stochastic machine breakdowns by an improved tuna swarm optimization algorithm. Journal of Manufacturing Systems **74**, 180–197 (2024) https://doi.org/10.1016/j.jmsy.2024.03.002

Feng, Z., Zou, Z., Liang, X.: Solving machine overload for re-scheduling of dynamic flexible job shop by adaptive tripartite game theory-based genetic algorithm. Swarm and Evolutionary Computation **96**, 101938 (2025) https://doi.org/10.1016/j.swevo.2025.101938

Fan, J., Zhang, C., Yang, F., Shen, W., Gao, L.: A matheuristic with re-lot-sizing strategies for flexible job-shop rescheduling problem with lot-streaming and machine reconfigurations. European Journal of Operational Research **319**(3), 747–762 (2024) https://doi.org/10.1016/j.ejor.2024.07.030

Grimm, V., Berger, U., DeAngelis, D.L., Polhill, J.G., Giske, J., Railsback, S.F.: The odd protocol: A review and first update. Ecological Modelling **221**(23), 2760–2768 (2010) https://doi.org/10.1016/j.ecolmodel.2010.08.019

Gomes, M.C., Barbosa-Póvoa, A.P., and, A.Q.N.: Reactive scheduling in a make-to-order flexible job shop with re-entrant process and assembly: a mathematical programming approach. International Journal of Production Research **51**(17), 5120–5141 (2013) https://doi.org/10.1080/00207543.2013.793428 https://doi.org/10.1080/00207543.2013.793428

Ghasemi, A., Farajzadeh, F., Heavey, C., Fowler, J., Papadopoulos, C.T.: Simulation optimization applied to production scheduling in the era of industry 4.0: A review and future roadmap. Journal of Industrial Information Integration **39**, 100599 (2024) https://doi.org/10.1016/j.jii.2024.100599

Guo, H., Liu, J., Wang, Y., Zhuang, C.: An improved genetic programming hyper-heuristic for the dynamic flexible job shop scheduling problem with reconfigurable manufacturing cells. Journal of Manufacturing Systems **74**, 252–263 (2024) https://doi.org/10.1016/j.jmsy.2024.03.009

Gao, K.Z., Suganthan, P.N., Pan, Q.K., Tasgetiren, M.F., Sadollah, A.: Artificial bee colony algorithm for scheduling and rescheduling fuzzy flexible job shop problem with new job insertion. Knowledge-Based Systems **109**, 1–16 (2016) https://doi.org/10.1016/j.knosys.2016.06.014

Guo, Z., Wei, L., Li, X., Yang, S., Zhang, J.: A variable window multi-interval rescheduling optimization algorithm for dynamic flexible job shop problem. Applied Soft Computing **176**, 113157 (2025) https://doi.org/10.1016/j.asoc.2025.113157

Hu, M., Qin, S., Wang, S., Zhang, J., Ding, G.: An energy-saving real-time scheduling method based on bi-level multi-agent architecture with bargaining game for flexible job shops. Expert Systems with Applications **269**, 126527 (2025) https://doi.org/10.1016/j.eswa.2025.126527

Ji, S., Wang, Z., Yan, J.: A multi-type data driven framework for solving flexible job shop scheduling problem considering multiple production resource states. Computers & Industrial Engineering **200**, 110835 (2025) https://doi.org/10.1016/j.cie.2024.110835

Kacem, I., Hammadi, S., Borne, P.: Pareto-optimality approach for flexible job-shop scheduling problems: hybridization of evolutionary algorithms and fuzzy logic. Mathematics and Computers in Simulation **60**(3), 245–276 (2002) https://doi.org/10.1016/S0378-4754(02)00019-8 . Intelligent Forecasting, Fault Diagnosis, Scheduling, and Control

Körner, A., Pasterk, D., Stadler, F., Zeh, C.: Insight-driven optimization to improve dynamic scheduling for flexible job-shops. IFAC-PapersOnLine **59**(1), 451–456 (2025) https://doi.org/10.1016/j.ifacol.2025.03.077 . 11th Vienna International Conference on Mathematical Modelling MATHMOD 2025

Li, Z., Chen, Y.: Dynamic scheduling of multi-memory process flexible job shop problem based on digital twin. Computers & Industrial Engineering **183**, 109498 (2023) https://doi.org/10.1016/j.cie.2023.109498

Li, L., Chai, Z.: Energy-saving distributed flexible job-shop scheduling with fuzzy processing time in iiot: A novel evolutionary multitasking algorithm. Journal of Industrial Information Integration **45**, 100829 (2025) https://doi.org/10.1016/j.jii.2025.100829

Liu, X., Chen, X., Chau, V., Musial, J., Blazewicz, J.: Flexible job shop scheduling problem using graph neural networks and reinforcement learning. Computers & Operations Research **182**, 107139 (2025) https://doi.org/10.1016/j.cor.2025.107139

Chen, X.-l., Li, J.-q., Xu, Y.: Q-learning based multi-objective immune algorithm for fuzzy flexible job shop scheduling problem considering dynamic disruptions. Swarm and Evolutionary Computation **83**, 101414 (2023) https://doi.org/10.1016/j.swevo.2023.101414

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. Computers & Industrial Engineering **155**, 107211 (2021) https://doi.org/10.1016/j.cie.2021.107211

Lv, L., Fan, J., Zhang, C., Shen, W.: Schedule repair for flexible job shops under machine breakdowns by deep reinforcement learning. Computers & Industrial Engineering **207**, 111256 (2025) https://doi.org/10.1016/j.cie.2025.111256

Li, X., Guo, A., Yin, X., Tang, H., Wu, R., Zhao, Q., Li, Y., Wang, X.: A q-learning improved differential evolution algorithm for human-centric dynamic distributed flexible job shop scheduling problem. Journal of Manufacturing Systems **80**, 794–823 (2025) https://doi.org/10.1016/j.jmsy.2025.04.001

Li, Y., Gu, W., Yuan, M., Tang, Y.: Real-time data-driven dynamic scheduling for flexible job shop with insufficient transportation resources using hybrid deep q network. Robotics and Computer-Integrated Manufacturing **74**, 102283 (2022) https://doi.org/10.1016/j.rcim.2021.102283

Li, Z.-K., Li, N., Pan, Q.-K., Gao, L., Li, W.-M.: A parallel cooperative co-evolutionary algorithm for flexible job-shop scheduling with workers' heterogeneity. Expert Systems with Applications **290**, 128362 (2025) https://doi.org/10.1016/j.eswa.2025.128362

Li, X., Peng, Z., Du, B., Guo, J., Xu, W., Zhuang, K.: Hybrid artificial bee colony algorithm with a rescheduling strategy for solving flexible job shop scheduling problems. Computers & Industrial Engineering **113**, 10–26 (2017) https://doi.org/10.1016/j.cie.2017.09.005

Liu, J., Sun, B., Li, G., Chen, Y.: Multi-objective adaptive large neighbourhood search algorithm for dynamic flexible job shop schedule problem with transportation resource. Engineering Applications of Artificial Intelligence **132**, 107917 (2024) https://doi.org/10.1016/j.engappai.2024.107917

Liu, Q., Zhu, L., Gao, Z., Wang, J., Li, J.: Research on flexible job shop scheduling based on improved two-layer optimization algorithm. Computers, Materials & Continua **78**(1), 811–843 (2024) https://doi.org/10.32604/cmc.2023.046040

Manafi, E., Domenech, B., Tavakkoli-Moghaddam, R., Ranaboldo, M.: A self-learning whale optimization algorithm based on reinforcement learning for a dual-resource flexible job shop scheduling problem. Applied Soft Computing **180**, 113436 (2025) https://doi.org/10.1016/j.asoc.2025.113436

Nouiri, M., Bekrar, A., Trentesaux, D.: Towards energy efficient scheduling and rescheduling for dynamic flexible job shop problem. IFAC-PapersOnLine **51**(11), 1275–1280 (2018) https://doi.org/10.1016/j.ifacol.2018.08.357 . 16th IFAC Symposium on Information Control Problems in Manufacturing INCOM 2018

Ngwu, C., Liu, Y., Wu, R.: Reinforcement learning in dynamic job shop scheduling: a comprehensive review of ai-driven approaches in modern manufacturing. Journal of Intelligent Manufacturing (2025) https://doi.org/10.1007/s10845-025-02585-6

Otamendi, U., Martinez, I., Belaunzaran, X., Artetxe, A., Franco, J., Uribe, A., Olaizola, I.G., Sierra, B.: An analytics-based framework for optimizing resource allocation and preemptive scheduling in manufacturing. Decision Analytics Journal **16**, 100596 (2025) https://doi.org/10.1016/j.dajour.2025.100596

Perroux, T., Arbaoui, T., Merghem-Boulahia, L.: The flexible job shop scheduling problem with setups and operator skills: an application in the textile industry. IFAC-PapersOnLine **58**(19), 1090–1095 (2024) https://doi.org/10.1016/j.ifacol.2024.09.125 . 18th IFAC Symposium on Information Control Problems in Manufacturing INCOM 2024

Sreekara Reddy, M.B.S., Ratnam, C., Rajyalakshmi, G., Manupati, V.K.: An effective hybrid multi objective evolutionary algorithm for solving real time event in flexible job shop scheduling problem. Measurement **114**, 78–90 (2018) https://doi.org/10.1016/j.measurement.2017.09.022

Schworm, P., Wu, X., Klar, M., Aurich, J.C.: Multi-objective rescheduling of job shop scheduling problems in manufacturing using quantum annealing. Manufacturing Letters **42**, 5–10 (2024) https://doi.org/10.1016/j.mfglet.2024.09.066

Shao, S., Xu, G., Li, J., Meng, X., Xu, X., Wang, H., Jin, Z.: A lot streaming scheduling algorithm with variable sub-lots for solving a multi-objective flexible job shop scheduling problem. Applied Soft Computing **177**, 113184 (2025) https://doi.org/10.1016/j.asoc.2025.113184

Shi, Y., Yang, Y., Li, B., Qian, H., Hao, H., Zhou, A.: Diversity-enhanced hyper-heuristics for multi-objective dynamic flexible job shop scheduling. Swarm and Evolutionary Computation **96**, 101994 (2025) https://doi.org/10.1016/j.swevo.2025.101994

Sun, J., Zhang, Z., Zhang, G., Huang, Z.: Multi-objective evolutionary algorithm based flexible assembly job-shop rescheduling with component sharing for order insertion. Computers & Operations Research **169**, 106744 (2024) https://doi.org/10.1016/j.cor.2024.106744

Thenarasu, M., Rameshkumar, K., Rousseau, J., Anbuudayasankar, S.P.: Development and analysis of priority decision rules using mcdm approach for a flexible job shop scheduling: A simulation study. Simulation Modelling Practice and Theory **114**, 102416 (2022) https://doi.org/10.1016/j.simpat.2021.102416

Tang, H., Xiao, Y., Zhang, W., Lei, D., Wang, J., Xu, T.: A dql-nsga-iii algorithm for solving the flexible job shop dynamic scheduling problem. Expert Systems with Applications **237**, 121723 (2024) https://doi.org/10.1016/j.eswa.2023.121723

Uribe, A., Otamendi, U., Artetxe, A., Rivera, J.C.: Mathematical formulations for the scheduling of unrelated parallel machines considering machine availability and eligibility. Expert Systems with Applications **284**, 127773 (2025) https://doi.org/10.1016/j.eswa.2025.127773

Wan, L., Cui, X., Zhao, H., Fu, L., Li, C.: A novel method for solving dynamic flexible job-shop scheduling problem via difformer and deep reinforcement learning. Computers & Industrial Engineering **198**, 110688 (2024) https://doi.org/10.1016/j.cie.2024.110688

Winklehner, P., Hauder, V.A.: Flexible job-shop scheduling with release dates, deadlines and sequence dependent setup times: a real-world case. Procedia Computer Science **200**, 1654–1663 (2022) https://doi.org/10.1016/j.procs.2022.01.366 . 3rd International Conference on Industry 4.0 and Smart Manufacturing

Wang, H., Lin, W., Peng, T., Xiao, Q., Tang, R.: Multi-agent deep reinforcement learning-based approach for dynamic flexible assembly job shop scheduling with uncertain processing and transport times. Expert Systems with Applications **270**, 126441 (2025) https://doi.org/10.1016/j.eswa.2025.126441

Wang, J., Liu, Y., Ren, S., Wang, C., Wang, W.: Evolutionary game based real-time scheduling for energy-efficient distributed and flexible job shop. Journal of Cleaner Production **293**, 126093 (2021) https://doi.org/10.1016/j.jclepro.2021.126093

Wang, J., Liu, Y., Ren, S., Wang, C., Ma, S.: Edge computing-based real-time scheduling for digital twin flexible job shop with variable time window. Robotics and Computer-Integrated Manufacturing **79**, 102435 (2023) https://doi.org/10.1016/j.rcim.2022.102435

Wang, H., Peng, T., Li, X., He, J., Liu, W., Tang, R.: An integrated simulation–optimization method for flexible assembly job shop scheduling with lot streaming and finite transport resources. Computers & Industrial Engineering **200**, 110790 (2025) https://doi.org/10.1016/j.cie.2024.110790

Wang, H., Peng, T., Nassehi, A., Tang, R.: A data-driven simulation-optimization framework for generating priority dispatching rules in dynamic job shop scheduling with uncertainties. Journal of Manufacturing Systems **70**, 288–308 (2023) https://doi.org/10.1016/j.jmsy.2023.08.001

Wei, S., Tang, H., Li, X., Lei, D., Wang, X.V.: An improved memetic algorithm for multi-objective resource-constrained flexible job shop inverse scheduling problem: An application for machining workshop. Journal of Manufacturing Systems **74**, 264–290 (2024) https://doi.org/10.1016/j.jmsy.2024.03.005

Wang, Z., Wan, C., Liu, J., Zhang, X., Wang, H., Hu, Y., Hu, Z.: Masc: Large language model-based multi-agent scheduling chain for flexible job shop scheduling problem. Advanced Engineering Informatics **67**, 103527 (2025) https://doi.org/10.1016/j.aei.2025.103527

Wang, Y., Xia, T., Xu, Y., Ding, Y., Zheng, M., Pan, E., Xi, L.: Joint optimization of flexible job shop scheduling and preventive maintenance under high-frequency production switching. International Journal of Production Economics **269**, 109163 (2024) https://doi.org/10.1016/j.ijpe.2024.109163

Wu, R., Zheng, J., Li, X., Tang, H., Wang, X.V., Li, Y.: Dynamic scheduling for flexible job shop under machine breakdown using improved double deep q-network. Expert Systems with Applications **288**, 128280 (2025) https://doi.org/10.1016/j.eswa.2025.128280

Xu, B., Cao, X., Zhang, S., Shen, J., Wang, C., Wang, S., Huang, D., Wang, C., Mei, Y., Wang, Y.: Terminal normalization in genetic programming for dynamic flexible job shop scheduling. Swarm and Evolutionary Computation **96**, 101970 (2025) https://doi.org/10.1016/j.swevo.2025.101970

Xiao, Y., Yoogalingam, R.: A simulation optimization approach for planning and scheduling in operating rooms for elective and urgent surgeries. Operations Research for Health Care **35**, 100366 (2022) https://doi.org/10.1016/j.orhc.2022.100366

Xu, Y., Zhang, M., Wang, D., Yang, M., Liang, C.: Hybrid gaussian quantum particle swarm optimization and adaptive genetic algorithm for flexible job-shop scheduling problem. Engineering Applications of Artificial Intelligence **154**, 110882 (2025) https://doi.org/10.1016/j.engappai.2025.110882

Yang, Z., Li, X., Gao, L., Liu, Q.: A heterogeneous graph attention-enhanced deep reinforcement learning framework for flexible job shop scheduling problem with variable sublots. Engineering Applications of Artificial Intelligence **157**, 111375 (2025)

https://doi.org/10.1016/j.engappai.2025.111375

Yan, B., Liu, X., Lu, S., Hu, C., Wang, X., Zhou, Z.: Deep reinforcement learning approach for a dynamic flexible job shop problem with sequence dependent setup times. Computers & Industrial Engineering **207**, 111310 (2025) https://doi.org/10.1016/j.cie.2025.111310

Yuan, M., Yu, Q., Zhang, L., Lu, S., Li, Z., Pei, F.: Deep reinforcement learning based proximal policy optimization algorithm for dynamic job shop scheduling. Computers & Operations Research **183**, 107149 (2025) https://doi.org/10.1016/j.cor.2025.107149

Yu, C., Zhang, C., Fan, J., Shen, W.: An integrated mathematical programming and reinforcement learning algorithm for the flexible job shop scheduling with variable lot-sizing. Journal of Manufacturing Systems **82**, 210–223 (2025) https://doi.org/10.1016/j.jmsy.2025.05.002

Zhao, Z., Chen, X., An, Y.: Multi-objective flexible flow-shop rescheduling with rigid–flexible hybrid constraints and preventive maintenance. Computers & Industrial Engineering **200**, 110813 (2025) https://doi.org/10.1016/j.cie.2024.110813

Zhang, Y., Dong, S., Yuan, Z., Wen, T., Xiao, J., Diao, Z.: Meta-relation-based heterogeneous graph neural network with deep reinforcement learning for flexible job shop scheduling. Expert Systems with Applications **291**, 128411 (2025) https://doi.org/10.1016/j.eswa.2025.128411

Zhang, L., Feng, Y., Xiao, Q., Xu, Y., Li, D., Yang, D., Yang, Z.: Deep reinforcement learning for dynamic flexible job shop scheduling problem considering variable processing times. Journal of Manufacturing Systems **71**, 257–273 (2023) https://doi.org/10.1016/j.jmsy.2023.09.009

Zampella, M., Otamendi, U., Belaunzaran, X., Artetxe, A., Olaizola, I.G., Sierra, B., Longo, G.: Exploring multi-agent reinforcement learning for unrelated parallel machine scheduling. The Journal of Supercomputing **81**(4) (2025) https://doi.org/10.1007/s11227-025-07030-2

Zhang, Q., Shao, W., Shao, Z., Pi, D.: Graph-based reinforced multi-objective optimization for distributed heterogeneous flexible job shop scheduling problem under nonidentical time-of-use electricity tariffs. Expert Systems with Applications **290**, 128428 (2025) https://doi.org/10.1016/j.eswa.2025.128428

Zhou, X., Wang, F., Wu, B., Li, Y., Shen, N.: Deep reinforcement learning-based memetic algorithm for solving dynamic distributed green flexible job shop scheduling problem with finite transportation resources. Swarm and Evolutionary Computation **94**, 101885 (2025) https://doi.org/10.1016/j.swevo.2025.101885

Zhang, W., Yan, S., Tang, H., Li, X., Lei, D.: An improved shuffled frog leaping

algorithm based on support vector machine for hybrid flow shop rescheduling with disturbance prediction. Swarm and Evolutionary Computation **93**, 101843 (2025) https://doi.org/10.1016/j.swevo.2025.101843

Zhang, L., Yan, Y., Yang, C., Hu, Y.: Dynamic flexible job-shop scheduling by multi-agent reinforcement learning with reward-shaping. Advanced Engineering Informatics **62**, 102872 (2024) https://doi.org/10.1016/j.aei.2024.102872

Zhu, Z., Zhou, X.: An efficient evolutionary grey wolf optimizer for multi-objective flexible job shop scheduling problem with hierarchical job precedence constraints. Computers & Industrial Engineering **140**, 106280 (2020) https://doi.org/10.1016/j.cie.2020.106280

Zhao, S., Zhou, H., Zhao, Y., Wang, D.: Dql-assisted competitive evolutionary algorithm for energy-aware robust flexible job shop scheduling under unexpected disruptions. Swarm and Evolutionary Computation **91**, 101750 (2024) https://doi.org/10.1016/j.swevo.2024.101750