Proceedings of the 2025 Winter Simulation Conference
E. Azar, A. Djanatliev, A. Harper, C. Kogler, V. Ramamohan, A. Anagnostou, and S. J. E. Taylor, eds.

A HEURISTIC-BASED ROLLING HORIZON METHOD FOR DYNAMIC AND STOCHASTIC UNRELATED PARALLEL MACHINE SCHEDULING

Shufang Xie[1], Tao Zhang[1], and Oliver Rose[1]

[1]Dept. of Computer Science, University of the Bundeswehr Munich, Neubiberg, GERMANY

ABSTRACT

In stochastic manufacturing environments, disruptions such as machine breakdowns, variable processing times, and unexpected delays make static scheduling approaches ineffective. To address this, we propose a heuristic-based rolling horizon scheduling method for unrelated parallel machines. The rolling horizon framework addresses system stochasticity by enabling dynamic adaptation through frequent rescheduling of both existing jobs and those arriving within a rolling lookahead window. This method decomposes the global scheduling problem into smaller, more manageable subproblems. Each subproblem is solved using a heuristic approach based on a suitability score that incorporates key factors such as job properties, machine characteristics, and job-machine interactions. Simulation-based experiments show that the proposed method outperforms traditional dispatching rules in dynamic and stochastic manufacturing environments with a fixed number of jobs, achieving shorter makespans and cycle times, reduced WIP levels, and lower machine utilization.

1 INTRODUCTION

Parallel machine scheduling is a fundamental problem in production and operations management, arising in various industries such as semiconductor manufacturing (Kim et al. 2002), healthcare (Roshanaei et al. 2017), and cloud computing (Kumar Bhardwaj et al. 2020). The problem involves assigning a set of jobs to multiple parallel machines to optimize specific performance criteria, such as minimizing makespan, total tardiness, or energy consumption. Due to its practical relevance, parallel machine scheduling has been extensively studied in operations research and scheduling theory. The literature on parallel machine scheduling problems is rich with over 2271 published articles (Ying et al. 2024).

In a parallel machine scheduling problem, a set of independent jobs must be scheduled on parallel machines. Each job has a processing time, and depending on the problem variant, additional constraints such as due dates, machine eligibility, sequence-dependent setup times, or precedence relations may apply (Zhang et al. 2025). The most common types of parallel machine environments include identical machines, where all machines have the same processing speed and the processing times of jobs are independent of the machine; uniform machines, where the machines operate at different speeds, and the processing time of each job is proportional to the speed of the assigned machine; and unrelated machines, where the processing time of each job depends on both the specific job and the machine, with no proportional relationship between them. Parallel machine scheduling involves the simultaneous processing of multiple jobs across parallel machines, presenting two key challenges: first, each job must be assigned to one of the parallel machines, and second, a schedule must be determined for each machine (Agnetis et al. 2025). This problem is widely recognized as NP-hard, meaning that finding an optimal solution is computationally difficult, particularly as the problem size increases (Garey and Johnson 1979). Among the different problem variants, unrelated parallel machine scheduling is the most generalized form, as it allows arbitrary variations in processing times across machines. Kayhan and Yildiz (2023) analyzed 80 articles published from 1995 to 2020 on machine scheduling problems from different aspects and reported that unrelated parallel machine scheduling is one of the most studied types of problems.

Existing studies in parallel machine scheduling have explored a wide range of problem constraints, such as machine setup times (Fonseca et al. 2024), due date constraints (Zhang et al. 2025) and other additional factors, including integrated scheduling with batching and heterogeneous delivery trucks (Joo and Kim 2017). Researchers have also investigated various solution approaches. Exact methods, such as branch-and-bound, guarantee optimal solutions but become computationally infeasible for large-scale or real-time scenarios. As a result, many studies turn to heuristic (So 1990) and metaheuristic techniques—including genetic algorithms, simulated annealing, and particle swarm optimization, which offer near-optimal solutions with lower computational complexity (Cao et al. 2005). For a comprehensive overview of these approaches in the context of unrelated parallel machine scheduling, readers are referred to the survey by Đurasević and Jakobović (2023), who provide a survey on heuristic and metaheuristic methods for the unrelated parallel machines. However, most of these studies are developed under static scheduling assumptions, where all job information is known in advance and system conditions are assumed stable. This assumption overlooks the dynamic and often unpredictable nature of real-world manufacturing environments. Real-world manufacturing environments are inherently stochastic, with uncertainties arising from machine breakdowns, variable processing times, and other unexpected disruptions. These factors make static scheduling approaches ineffective, as schedules created in advance can quickly become obsolete.

To address the challenges of dynamic and stochastic manufacturing environments, we propose a rolling horizon scheduling framework that enables continuous adaptation to evolving system conditions by updating the schedule at fixed intervals or certain events. Each schedule includes only the jobs that have already arrived, along with those expected to arrive within a lookahead time window from each scheduling point. The window rolls forward as scheduling occurs over time. Sub-scheduling is performed using a scoring method that accounts for job-specific characteristics, machine-related factors, and job-machine interactions that influence processing efficiency and assignment suitability. Compared to static scheduling, this approach decomposes the global scheduling problem into smaller subproblems, allowing for faster and more responsive decision-making. Unlike traditional decision rules, this method offers a broader scope and a locally "optimal" perspective, as it generates a schedule that considers future job arrivals within each subproblem.

To evaluate the effectiveness of the proposed heuristic, we conduct simulation-based experiments that model a dynamic and stochastic scheduling environment. The performance of our method is assessed across various scenarios and compared with traditional decision rules, which serve as widely adopted benchmarks in scheduling. Details of the proposed approach, its implementation, and the experimental setup are provided in the following sections. The remainder of this paper is organized as follows: Section 2 introduces the rolling horizon framework and scheduling schema. Section 3 presents the scoring method and its key components. In Section 4, we assess the performance of our method through a series of experiments, benchmarking it against existing scheduling techniques. Finally, Section 5 summarizes the findings and outlines potential directions for future research.

2 ROLLING HORIZON SCHEDULING

The rolling horizon framework enhances the resilience of manufacturing systems to stochastic events by periodically updating the production schedule. This approach enables the adjustment of previously scheduled jobs in response to unforeseen disruptions, thereby maintaining operational stability. Its inherent lookahead mechanism allows the system to anticipate near-future workloads, rather than merely reacting to the current state. This anticipatory capability is essential for preserving scheduling responsiveness and decision quality in dynamic manufacturing environments, where conditions such as machine availability and processing times may change unexpectedly.

2.1 Rolling Horizon Framework

First, two key concepts are introduced: the time window and the scheduling interval. The time window defines how far into the future job arrivals are considered when generating or updating the schedule, while the scheduling interval determines how frequently scheduling occurs. Figure 1 illustrates the rolling horizon mechanism. The time window maintains a constant length of L, representing a fixed scheduling horizon. In most cases, it advances periodically by a fixed interval D, where $D < L$. This design ensures overlapping scheduling windows, enabling frequent updates and continuous adaptation. As the window shifts forward, it advances the scheduling horizon, offering a stable and continuous lookahead into future operations.

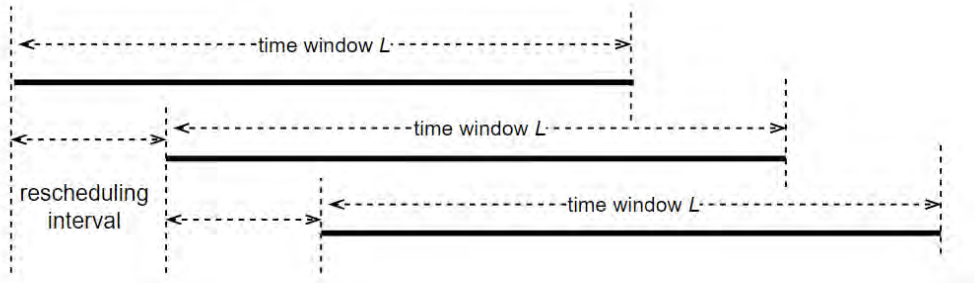

Figure 1: Rolling horizon mechanism.

A critical challenge of this approach lies in determining the optimal values for the time window length L and the rescheduling interval D. These parameters are influenced by the predefined job arrival plan specific to the problem instance. Since jobs arrive at scheduled times throughout the planning horizon, the rescheduling interval D must be carefully selected to ensure that a sufficient number of jobs are available at each update, while avoiding excessive computational overhead. Similarly, L must strike a balance between effective lookahead scheduling and uncertainty in job arrivals: if L is too short, it may fail to capture enough future jobs for meaningful decision-making, while an excessively long L may incorporate too many uncertain job arrivals.

To identify the optimal values for the time window and the rescheduling interval, a simulation-based approach can be adopted. First, a reasonable range for both parameters is defined based on the problem context, considering factors such as job processing times, expected job arrival rates, system constraints, and any other relevant properties of the scheduling environment. These factors help establish a set of plausible values for the time window and rescheduling interval. The core idea is to experiment with different parameter values, simulate the scheduling process for each configuration, and evaluate the resulting performance.

Theoretically, the most effective way is to perform scheduling precisely at the time when relevant events occur, eliminating the need for fixed-interval scheduling. However, in practice, not all critical events can be accurately captured. Therefore, this study primarily adopts a fixed time interval for scheduling. To better align the schedule with the evolving state of the manufacturing system, certain key events are still considered and trigger immediate scheduling when they occur:

- Machine breakdowns or maintenance
- Arrival of hot jobs
- Significant deviation in the actual arrival time of jobs compared to their estimated arrival time
- Significant deviation in the actual availability time of machines compared to their estimated availability time

Hot jobs refer to urgent, high-priority jobs that were not previously present in the system but appear unexpectedly and require immediate processing. Time deviations—whether earlier or later than expected—can impact the scheduling accuracy. When any of the above events occur, the system performs scheduling immediately rather than waiting for the next scheduled interval.

2.2 Scheduling/Rescheduling Schema

The flowchart in Figure 2 illustrates the scheduling process. Whenever scheduling is triggered, either at a fixed interval or by specific events, the arrival times of jobs are estimated. The time window defines the subset of jobs to be considered during scheduling. Only jobs that have already arrived and are eligible for rescheduling, along with those expected to arrive within the time window, are included in the scheduling list. Based on the current state of the machines, the listed jobs, and both local and global system conditions, a new schedule is generated.

According to the rolling horizon mechanism, jobs are typically scheduled either in advance or at the time of their arrival. Some of these jobs may already be in progress on machines, while others remain in the queue awaiting processing. In this study, we assume a non-preemptive scheduling policy, meaning that jobs already in progress cannot be rescheduled. The reschedulability of jobs that have not yet started depends on practical operational considerations commonly encountered in real manufacturing environments. For example, if preparatory activities—such as material staging or tooling—have already begun for a specific job on a particular machine, rescheduling may lead to unnecessary disruptions or inefficiencies. Conversely, jobs that remain flexible in terms of their execution timing, or for which no setup activities have yet been performed, can be rescheduled to better reflect the current state of the system and enhance overall resource utilization.

This selective rescheduling mechanism strikes a balance between adaptability to dynamic changes and the stability required for predictable and efficient execution on the shop floor. In practice, jobs that are about to start imminently are no longer considered reschedulable. To simplify this determination, a time-based constraint is introduced: if the difference between a job's scheduled start time and the current time is less than a predefined threshold, the job is deemed non-reschedulable.

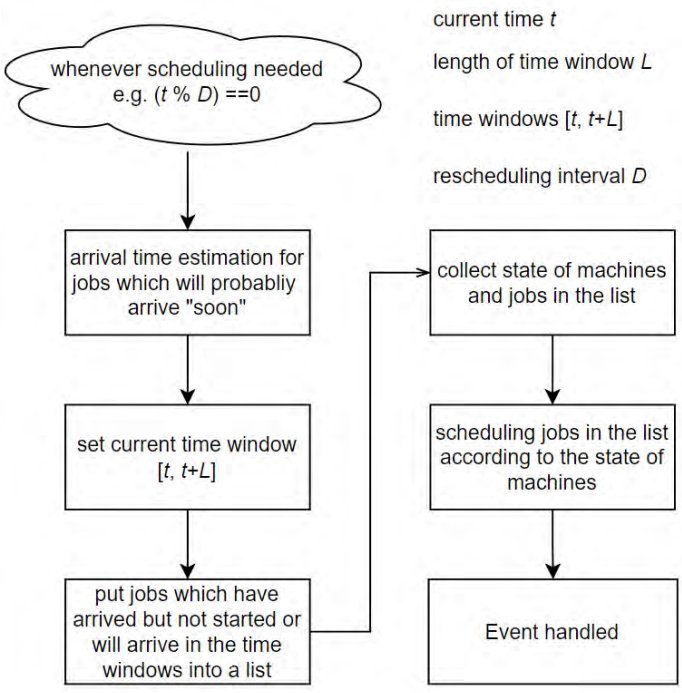

Figure 2: Periodic rolling horizon scheduling.

It is worth noting that, in estimating job arrival times, manufacturing environments differ from systems with unpredictable input flows. These environments often operate under a predefined production plan,

allowing job arrival times to be forecasted with reasonable accuracy. Instead of relying on pattern recognition or statistical inference from historical data, the scheduling system can directly reference the production plan to determine the expected release times of upcoming jobs. This deterministic approach simplifies the forecasting process and enhances the reliability of scheduling decisions within the rolling horizon framework.

3 SUB-SCHEDULING HEURISTICS

Rolling horizon scheduling determines the timing of scheduling and defines the time window in which the jobs should be considered at each decision point. It decomposes the large-scale, complex scheduling problem into a sequence of smaller, more manageable subproblems, each solved independently at regular scheduling points. While the reduced problem size might suggest that exact optimization methods (e.g., mathematical programming) could be used to find optimal solutions within each period, this is not always practical in real-time settings. In a dynamic system with frequent rescheduling and continuous uncertainty, repeatedly solving even small-scale mathematical models can lead to non-negligible computational delays. To ensure responsiveness and maintain real-time performance, we adopt a heuristic approach that provides high-quality, near-optimal solutions with significantly lower computational overhead. This allows the system to react quickly to changing conditions without incurring intolerable delays, making the heuristic more suitable for our rolling scheduling context.

3.1 Job-machine Pair Scoring

At each scheduling point, the actual scheduling is carried out by a dedicated job assignment heuristic, which optimizes job allocation based on the current system state. The key idea behind this heuristic is to simultaneously consider job characteristics, machine properties, and job-machine interactions when making scheduling decisions. Unlike traditional reactive methods that typically rely on simple job ordering rules and treat machines as interchangeable, our approach evaluates the suitability of each job-machine assignment by incorporating multiple factors.

The heuristic begins by generating all possible job-machine combinations. A suitability score is then calculated for each pair, combining job-specific attributes, machine characteristics, and their interactions to support more informed and context-aware scheduling decisions. The range of attributes considered may vary depending on the specific implementation and decision-making context. Table 1 summarizes a general set of job-, machine-, and job-machine-specific attributes that may be included in the evaluation of suitability scores.

Table 1: Attributes potentially used in suitability score calculation.

Category	Attribute Name	Description
Job	Arrival time	Time the job becomes available
	Due date	Target completion time for the job
	Priority	Importance level of the job
	Job type	Product family
Machine	Current workload	Total remaining processing time for jobs in the queue and the job being processed
	Queue length/time	Number /working time of jobs currently waiting
	Availability Status	Indicates if the machine is operational
	Idle time	Time the machine has been idle from the last process
	Maintenance schedule	Scheduled upcoming downtime or maintenance
Interaction	Processing time	Time required for one job on the specific machine
	Setup time	Time to prepare the machine for one job

Failure risk	Probability of machine failure when processing one job under the current state
Energy cost	Estimated energy consumption for running one job on the specific machine

The suitability score $S_{j,m}$ is calculated as a sum of selected attributes associated with a specific job-machine pair. Each attribute is assigned a corresponding weight to control its influence on the overall score. The general form of the suitability score is given by:

$$S_{j,m} = \sum_k \lambda_k \bullet a_k(j,m)$$

where $a_k(j,m)$ represents the value of the k-th attribute for job j on machine m, and λ_k is the weight associated with that attribute. This flexible formulation allows the heuristic to adapt to different objectives or constraints by adjusting the attribute set and their corresponding weights. Weights can be defined based on expert knowledge, statistical analysis, optimization methods, or empirical tuning through simulation experiments. The specific attributes and weights used in our experimental setup are detailed in Section 4.

3.2 Job-machine Pairs to Schedule

A score is calculated for each job-machine pair as described in the previous section. This section explains how these scored job-machine pairs are translated into an actual schedule. The procedure is illustrated in Figure 3.

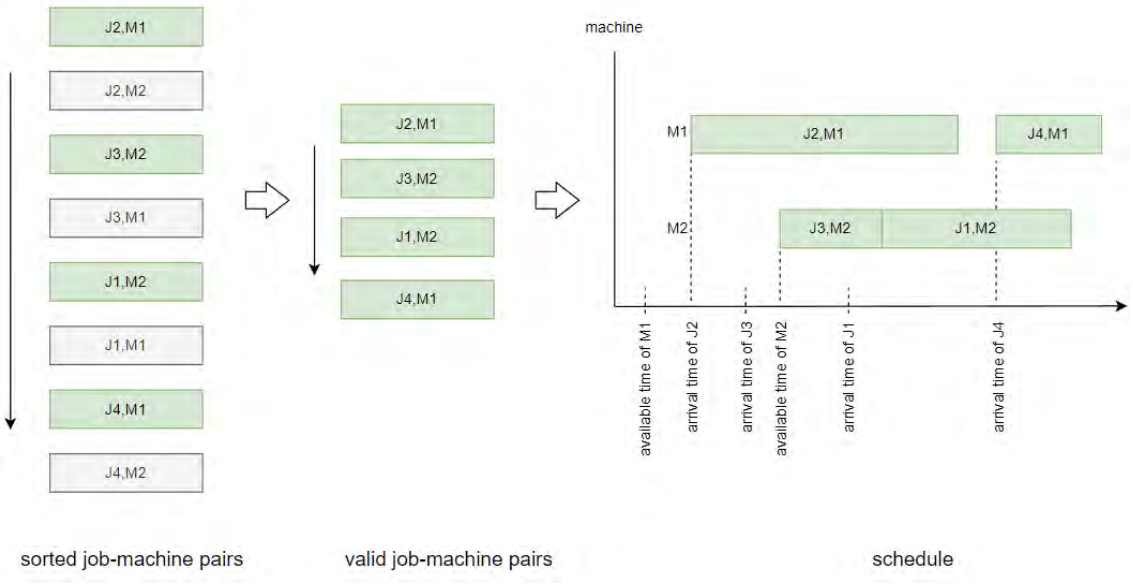

Figure 3: From a sorted sequence of job-machine pairs to the corresponding schedule.

First, all the job-machine pairs are sorted by their suitability score in descending order, from highest to lowest. This ensures that the most optimal job-machine assignments are considered first. Next, invalid or redundant assignments are filtered out. For each job, only its first occurrence in the sorted list is retained, ensuring that each job is assigned to a single machine. Any subsequent appearances of the same job in the list are discarded, preventing duplicate assignments.

Once the list has been filtered, a schedule is generated using the following algorithm:

```
For each machine in all machines, loop
    For each pair assigned to the machine in the valid sequence, loop
        start time of the job in the pair = max (job arrival time, machine available
         time)
        machine available time = start time of the job + processing time of the job
         on the machine
```

Because the system is subject to various uncertainties, the generated schedule cannot be strictly followed. For example, the scheduled start time of a job may be missed due to the random arrival time of the job or the unpredictable processing time of the previous job on the machine. In this study, instead of adhering strictly to the scheduled times, we focus on the sequence of jobs assigned to each machine. As a result, the schedule is executed reactively, as shown in Figure 4. The system checks the schedule only when a machine becomes idle or a job arrives, determining which job the machine will process next or which machine should process the arriving job. If a machine becomes idle but the first scheduled job has not yet arrived, the machine remains idle and waits for the job, even if other jobs are already scheduled and waiting on the machine. Similarly, if a job arrives at an idle machine but is not the next job in the machine's schedule, it cannot be processed immediately.

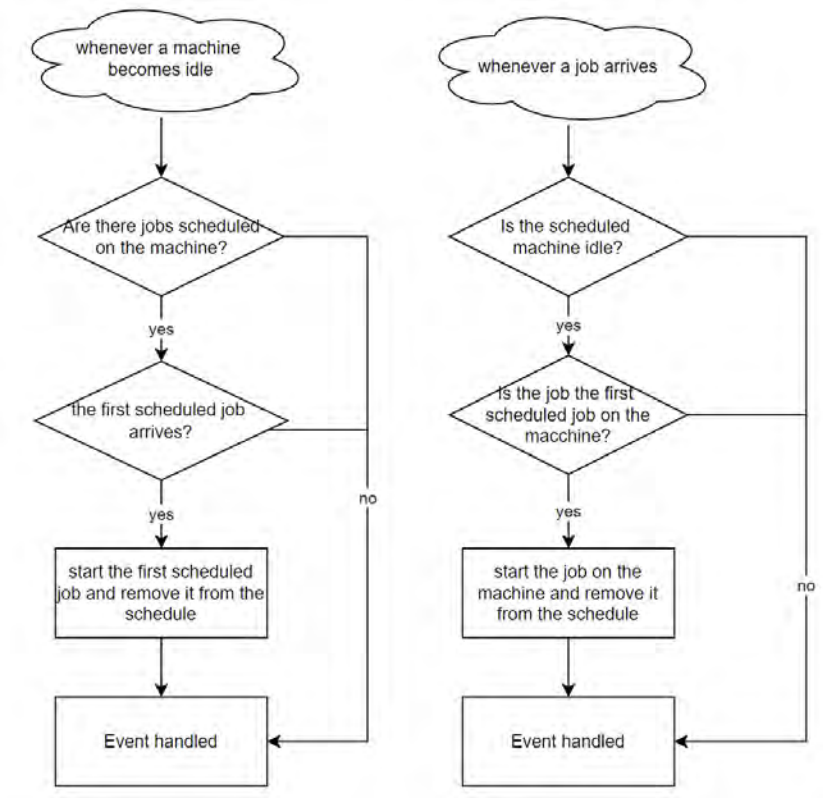

Figure 4: Reactive execution of schedules.

4 EXPERIMENTS

To evaluate the effectiveness of the proposed scheduling approach, we conduct a series of simulation-based experiments in an unrelated parallel machine scheduling environment. All simulation experiments were conducted in AnyLogic, a versatile platform for modeling discrete-event and stochastic systems.

4.1 Simulation Environment

The simulation model consists of 5 machines and 100 dynamically arriving jobs, where each job has different processing times on different machines, following an unrelated machine setting. All job processing times are drawn from a uniform distribution between 1.8 and 16.8 hours, ensuring a balanced mix of short and long tasks. All machines are assumed to be idle at time zero. Jobs arrive dynamically according to a predefined plan, with arrival times determined in advance. A total of 100 jobs are introduced over the planning horizon based on this arrival schedule. We assume no preemption—once a machine starts a job, it completes it before switching to another. Stochastic events, such as machine breakdowns, are also incorporated into the simulation. Each machine is assigned an initial failure time drawn from a uniform distribution between 0 and 30 hours. Subsequent failure intervals and repair durations are modeled using triangular distributions, with modes of 30 hours and 10 hours, respectively, and a variability of 10%. The simulation continues until all 100 jobs are processed. Upon completion, performance metrics such as makespan, cycle time, and machine utilization are recorded. Since jobs flow continuously into the manufacturing line in real-world scenarios, there is typically no clear completion point. The primary objective of scheduling is to minimize the average cycle time rather than the makespan.

The selection of factors and their corresponding weights in the scoring method was guided by domain knowledge and refined through preliminary simulation experiments. This section presents the final scoring scheme adopted in our experimental setup. Based on these considerations, the following factors are included in the scoring method:

- **Processing time**: Jobs with longer processing times on a given machine receive lower scores.
- **Machine status**: Idle machines are favored with higher scores, while busy machines are penalized. The status is represented as follows: 0 for idle, 1 for busy, and 2 for repair/maintenance.
- **Machine workload**: Defined as the sum of the processing times of all jobs scheduled on the machine, plus the remaining processing time of the job currently being processed (if any). Machines with higher workloads receive lower scores.

The score for each job-machine pair is calculated using the following equation:

$$Score = -processing\ time - 100 * machine\ status - 100 * machine\ workload .$$

4.2 Parameter Selections

Before comparing the proposed heuristic with baseline methods, we first determine the optimal values for the time window (L) and scheduling interval (D). Regardless of computational resource limitations, the scheduling interval should be as small as possible. Therefore, we begin by assigning a very short interval of 0.01 hours and conduct a sensitivity analysis for the time window, ranging from 0 to 8 hours with a step size of 0.5 hours. This results in a total of 16 scenarios, each run with 100 replications. The average makespan and cycle time for each scenario are shown in Figure 5. From this figure, we observe that the best scheduling performance—i.e., the shortest cycle time—is achieved when the time window is approximately 4.5 hours. Therefore, 4.5 hours is chosen as the optimal value for L in this case.

Another interesting observation from Figure 5 is that good performance is also achieved when L is set to 0 hours. In this case, only the jobs that have already arrived are considered, and machines do not wait for future jobs, leading to more immediate job assignments and higher utilization. However, when L is slightly increased (e.g., to 0.5 hours), performance deteriorates significantly. This is likely because machines may

choose to wait for jobs expected to arrive soon, resulting in idle time. When L increases further, more jobs are visible during scheduling, reducing unnecessary waiting and improving decisions. However, if L becomes too large, performance drops again due to over-planning or excessive anticipation of future jobs.

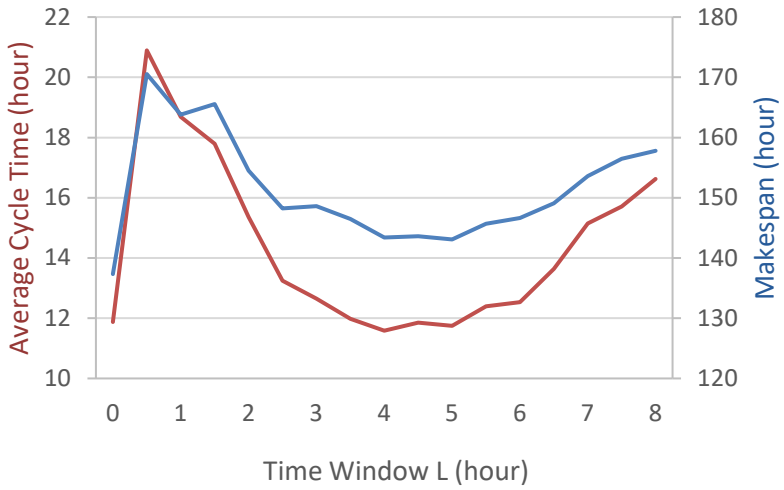

Figure 5: Sensitivity analysis of the time window.

By setting L to 4.5 hours, a sensitivity analysis for the scheduling interval is conducted, where the interval ranges from 0.1 to 1.7 hours with a step size of 0.1. A total of 17 scenarios are simulated, with 100 replications each. The makespan and average cycle time are reported and illustrated in Figure 6. From the trends of both curves, it is clear that smaller intervals lead to better performance. To balance the speed of the following experiment with minimal performance loss, the scheduling interval is set to 0.2 hours.

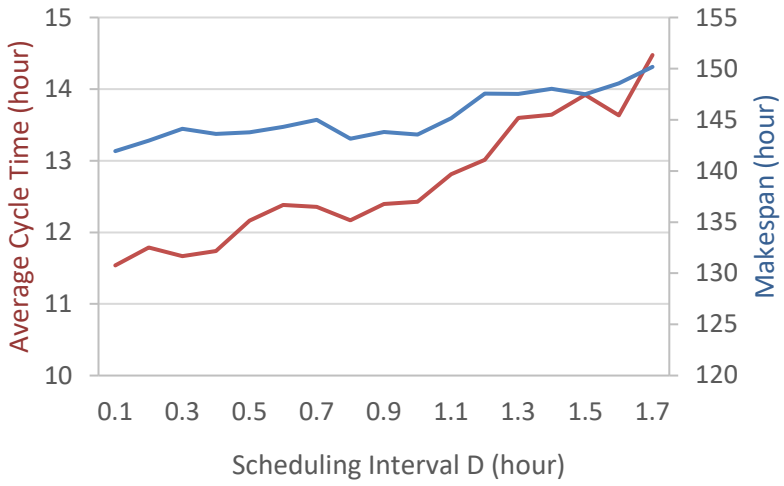

Figure 6: Sensitivity analysis of the scheduling interval.

4.3 Performance Comparison

We compare the performance of our proposed approach against several widely used decision rules: Shortest Processing Time First (SPT), First In First Out (FIFO), and the random rule. These rules are used to select a job or machine to start processing when a machine with a queue becomes idle, or when a job arrives at

multiple idle machines. SPT prioritizes jobs/machines based on processing time, always selecting the job/machine with the shortest processing time. FIFO prioritizes earlier-arriving jobs or machines, and can also be viewed as "First Idle, First Operated" on the machine side. The random rule selects jobs and machines randomly. Additionally, two combined approaches are considered: SPT+FIFO and FIFO+SPT. In the SPT+FIFO approach, jobs are selected using SPT, and machines are chosen using FIFO. Conversely, in FIFO+SPT, the process is reversed.

Since the performance of the approaches is heavily dependent on the system's capacity loading, defined as the ratio of total job demand to overall machine processing capacity, four loading levels (40%, 60%, 70%, and 80%, which is the original level) are designed by randomly removing some jobs from the set of 100 jobs. Sensitivity analysis is conducted for these levels. The optimal time window for the 40% loading level is 4.0 hours, while 4.5 hours is optimal for the other levels. A total of 24 scenarios are designed, with each scenario running 100 replications. Additional performance indicators are calculated, such as machine utilization, work in progress (WIP), and the deviation and margin of error (half-width confidence interval) of the makespan and average cycle time across the 100 replications. To facilitate a clear comparison, the mean average cycle time for all scenarios grouped by loading level is shown in Figure 7. The figure demonstrates that the proposed approach (OUR) outperforms the other rules across all loading levels.

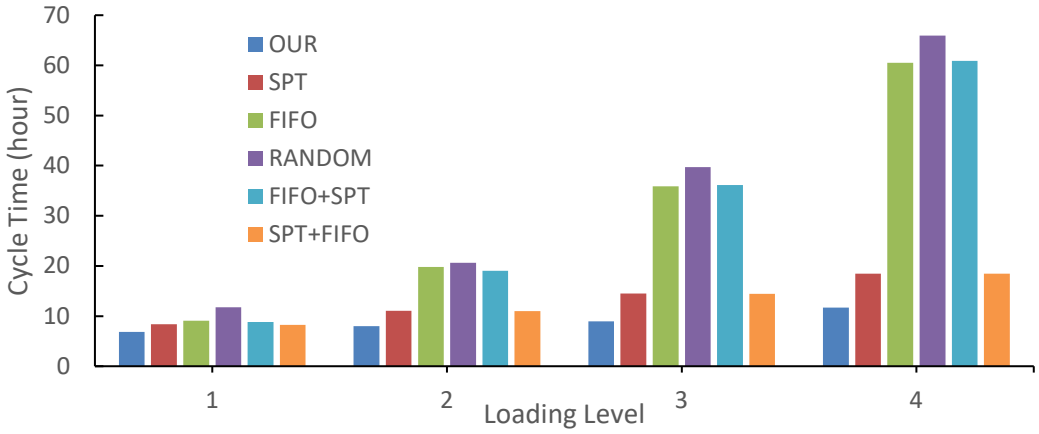

Figure 7: Comparison of mean average cycle time.

More comprehensive results are listed in Table 2. The proposed approach also outperforms the other methods in most indicators, except for the deviation and confidence interval of the makespan at the first loading level. It is important to note that the jobs included at each loading level are identical for all approaches. The desired performance is to complete these jobs as quickly as possible while using fewer resources, i.e., achieving lower machine utilization rather than the highest utilization.

Table 2: Performance comparison.

Load level	Method	Number of Jobs	WIP	Utili.	Makespan (hour)			Cycle Time (hour)		
					Mean	Dev.	MoE (95%)	Mean	Dev.	MoE (95%)
1	OUR-4.0	34	2.17	44%	118.07	4.98	0.99	6.84	0.91	0.18
	SPT	34	2.54	53%	123.08	4.52	0.90	8.39	1.17	0.23
	FIFO	34	2.67	55%	125.57	6.35	1.26	9.07	1.99	0.39
	RANDOM	34	3.40	65%	128.00	6.23	1.24	11.75	1.84	0.37

	FIFO+SPT	34	2.62	54%	124.80	6.23	1.24	8.82	1.76	0.35
	SPT+FIFO	34	2.49	52%	123.58	4.82	0.96	8.26	1.11	0.22
2	OUR-4.5	57	3.53	63%	129.78	4.51	0.89	7.98	0.83	0.16
	SPT	57	4.62	79%	136.87	6.27	1.24	11.03	1.28	0.25
	FIFO	57	7.11	84%	158.79	9.42	1.87	19.77	3.45	0.68
	RANDOM	57	7.42	86%	158.24	9.68	1.92	20.61	3.98	0.79
	FIFO+SPT	57	6.94	84%	156.31	9.47	1.88	19.04	4.04	0.80
	SPT+FIFO	57	4.66	80%	135.27	5.71	1.13	10.98	1.29	0.26
3	OUR-4.5	76	5.14	72%	133.28	4.53	0.90	8.97	1.02	0.20
	SPT	76	7.58	87%	145.95	7.88	1.56	14.48	1.82	0.36
	FIFO	76	13.88	90%	196.88	9.60	1.90	35.85	4.47	0.89
	RANDOM	76	14.97	91%	201.64	11.74	2.33	39.68	5.48	1.09
	FIFO+SPT	76	13.88	90%	198.16	10.35	2.05	36.10	4.42	0.88
	SPT+FIFO	76	7.51	87%	146.82	7.45	1.48	14.42	1.95	0.39
4	OUR-4.5	100	8.22	79%	142.77	6.08	1.21	11.67	1.12	0.22
	SPT	100	11.83	91%	156.84	6.33	1.25	18.44	1.43	0.28
	FIFO	100	24.03	92%	252.35	11.40	2.26	60.48	5.86	1.16
	RANDOM	100	25.53	93%	258.95	12.87	2.55	65.95	6.26	1.24
	FIFO+SPT	100	23.96	92%	254.63	11.14	2.21	60.87	5.40	1.07
	SPT+FIFO	100	11.74	90%	158.32	6.45	1.28	18.48	1.54	0.31

5 CONCLUSION

In this study, we implemented a heuristic-based rolling horizon method for unrelated parallel machine scheduling in dynamic and stochastic environments. This strategy updates the schedule at fixed rescheduling intervals or in response to special events while considering upcoming jobs expected to arrive within a specified time window. Our findings highlight the importance of carefully selecting both time window and the scheduling interval to balance responsiveness with planning scope, ultimately enhancing overall system performance.

Through the horizon strategy, the global scheduling problem is decomposed into smaller subproblems, each of which considers a selective subset of previously scheduled but unprocessed jobs, along with future jobs expected to arrive within the time window. This selective rescheduling strategy enhances schedule stability while improving the system's responsiveness to stochastic events, such as machine breakdowns. By incorporating anticipated future jobs, the scheduler becomes more proactive and better equipped to manage dynamic system changes. Each subproblem is solved using a scoring heuristic that evaluates job-machine pairs based on various attributes, including job characteristics, machine conditions, and job-machine interactions. This distinguishes our approach from traditional dispatching rules, which rely on fixed priorities and treat machines as interchangeable.

Experimental results demonstrate that both the rolling horizon strategy and the proposed scoring method are critical to effective scheduling in dynamic and uncertain environments. While the approach

does not guarantee global optimality, its low computational overhead and responsiveness make it highly suitable for real-world applications where adaptability and decision speed are essential. Future work could focus on analyzing the performance of the proposed approach under different distributions of random variables, such as processing times, breakdown and repair times, and so on. Additionally, arrival time estimation could be an important direction for scheduling in the absence of a predefined arrival plan.

REFERENCES

Agnetis, A., J.-C. Billaut, M. Pinedo, and D. Shabtay. 2025. "Fifty Years of Research in Scheduling – Theory and Applications". *European Journal of Operational Research* 327(2):367–393.

Cao, D., M. Chen, and G. Wan. 2005. "Parallel Machine Selection and Job Scheduling to Minimize Machine Cost and Job Tardiness". *Computers and Operations Research* 32(8):1995–2012.

Cheng, T. C. E., and C. C. S. Sin. 1990. "A State-of-the-Art Review of Parallel-Machine Scheduling Research". *European Journal of Operational Research* 47(3):271-292.

Đurasević, M., and D. Jakobović. 2023. "Heuristic and Metaheuristic Methods for the Parallel Unrelated Machines Scheduling Problem: A Survey". *Artificial Intelligence Review* 56(4):3181–3289.

Fonseca, G. H. G., G. B. Figueiroa, and T. A. M. Toffolo. 2024. "A Fix-and-Optimize Heuristic for the Unrelated Parallel Machine Scheduling Problem". *Computers & Operations Research* 163:106504.

Joo, C. M., and B. S. Kim. 2017. "Rule-Based Meta-Heuristics for Integrated Scheduling of Unrelated Parallel Machines, Batches, and Heterogeneous Delivery Trucks". *Applied Soft Computing* 53:457–476.

Kayhan, B. M., and G. Yildiz. 2023. "Reinforcement Learning Applications to Machine Scheduling Problems: A Comprehensive Literature Review". *Journal of Intelligent Manufacturing* 34(3):905–929.

Kim, D.-W., K.-H. Kim, W. Jang, and F. F. Chen. 2002. "Unrelated Parallel Machine Scheduling with Setup Times Using Simulated Annealing". *Robotics and Computer-Integrated Manufacturing* 18(3):223–231.

Kumar Bhardwaj, A., Y. Gajpal, C. Surti, S. S. Gill, and L. M. Thapar. 2020. "HEART: Unrelated Parallel Machines Problem with Precedence Constraints for Task Scheduling in Cloud Computing Using Heuristic and Meta-Heuristic Algorithms". *Software: Practice and Experience* 50(12):2231–2251.

Roshanaei, V., C. Luong, D. M. Aleman, and D. Urbach. 2017. "Propagating Logic-Based Benders' Decomposition Approaches for Distributed Operating Room Scheduling". *European Journal of Operational Research* 257(2):439–455.

So, K. C. 1990. "Some Heuristics for Scheduling Jobs on Parallel Machines with Setups". *Scientia* 36(4):361–369.

Ying, K. C., P. Pourhejazy, and X. Y. Huang. 2024. "Revisiting the Development Trajectory of Parallel Machine Scheduling". *Computers & Operations Research* 168:106709.

Zhang, W., M. Kong, Y. Zhang, A. M. Fathollahi-Fard, and G. Tian. 2025. "A Revised Deep Reinforcement Learning Algorithm for Parallel Machine Scheduling Problem under Multi-Scenario Due Date Constraints". *Swarm and Evolutionary Computation* 92:101808.

AUTHOR BIOGRAPHIES

SHUFANG XIE is a Research Assistant and Ph.D. student at Universität der Bundeswehr at the Chair of Modeling and Simulation. Her focus is on simulation-based scheduling and optimization of production systems. She received her M.S. degree in Metallurgical Engineering from Chongqing University, China. Her email address is shufang.xie@unibw.de.

TAO ZHANG is a Research Assistant at the Universität der Bundeswehr München, and he holds an M.S. in Metallurgical Engineering from Chongqing University, China and a Ph.D. in Computer Science from the Universität der Bundeswehr München, Germany. His research interest is working on production planning and scheduling, the main focus of his research is on the modelling and simulation of complex systems and intelligent optimization algorithms. His email address is tao.zhang@unibw.de.

OLIVER ROSE holds the Chair for Modeling and Simulation at the Department of Computer Science of the Universität der Bundeswehr, Germany. He received an M.S. degree in applied mathematics and a Ph.D. degree in computer science from Würzburg University, Germany. His research focuses on the operational modelling, analysis, and material flow control of complex manufacturing facilities such as semiconductor factories. He is a member of INFORMS Simulation Society, ASIM, and GI. His email address is oliver.rose@unibw.de.