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.

How Good is my Scheduling Algorithm? A Benchmark to Compare Production Scheduling Algorithms for Real-World Production Environments

Michael Groth

michael.groth@uni-goettingen.de

University of Goettingen

Matthias Schumann

University of Goettingen

Research Article

Keywords: production scheduling, industrial scheduling, scheduling benchmark, real-world production, machine learning, algorithm performance evaluation

Posted Date: October 7th, 2025

DOI: https://doi.org/10.21203/rs.3.rs-7195769/v1

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

Additional Declarations: No competing interests reported.

Abstract

Production scheduling in modern manufacturing faces escalating complexity due to mass customization, frequent disruptions, and stringent lead-time requirements, rendering classical benchmarks inadequate. This paper introduces a realistic, multi-instance benchmark to evaluate and compare scheduling algorithms under real-world shop-floor constraints. Guided by a problem-centered design-science methodology, we derive high-level requirements—sufficient complexity, differentiation capability, objective-function variants, and resistance to overfitting—and implement ten real-world-relevant features, including sequence-dependent changeovers, limited buffers, personnel constraints, transportation resources, and job release dates. Our benchmark models a two-shop-floor environment with buffers and interconnected transportation, featuring five material types that follow hybrid routing patterns, and 100 randomly generated instances per variant, all encoded in JSON for seamless integration. Performance is assessed via average relative improvement over a "random" priority-rule baseline, which mitigates randomness and discourages overfitting. We demonstrate and validate the benchmark through the application of FIFO, due-date, changeover heuristics, and a reinforcement-learning algorithm, showing clear differentiation among methods, thus demonstrating the potential of the benchmark to assess the solution quality of production scheduling algorithms. We provide the benchmark and all necessary data for subsequent use in the supplements. We contribute to research and practice by providing a benchmark for comparing production scheduling algorithms in a standardized manner on solution quality in real-world production environments.

1 Introduction

Production scheduling is becoming increasingly complex due to rising demands for customized products, as summarized in the Mass Customization theory, resulting in smaller batch sizes that require more changeovers (Da Silveira et al., 2001; Yang & Takakuwa, 2017). Additionally, customers expect shorter lead times, which reduces flexibility (Schuh et al., 2012), while frequent disruptions, such as urgent orders or machine breakdowns, necessitate fast responses (Baykasoğlu & Ozsoydan, 2018; Freier, 2020). These influences increase the complexity of production scheduling and can require sophisticated scheduling methods, as rule-based methods may lack solution quality (Liang et al., 2022). Furthermore, manual methods, as well as genetic algorithms and other search-based methods, require too much time to respond to changing environments (Kück et al., 2016).

Simultaneously, significant advances in data-driven technologies, particularly Artificial Intelligence (AI), offer promising avenues to address these complexities. The adoption of Cyber-Physical Production Systems (CPPS) with sensor-based real-time monitoring enables the delivery of real-time production data to Manufacturing Execution Systems (MES) and Enterprise Resource Planning (ERP) systems, capturing comprehensive production-related information, including machine status, processing times, order specifics, and disruptions (Lasi et al., 2014; Monostori, 2014). By leveraging these extensive datasets, AI-driven scheduling algorithms can potentially deliver optimized and timely scheduling decisions, thereby enhancing operational efficiency and responsiveness (Waschneck et al., 2018).

This trend has led to a multitude of scheduling algorithms being proposed in research. To understand the quality of the proposed algorithms, benchmarks are required to compare solution quality on standardized problems (Taillard, 1993). Such benchmarks exist for standard optimization problems found in literature, such as the Job-Shop Scheduling Problem (JSSP) (Pinedo, 2016). However, there is a discrepancy between these idealized theoretical benchmarks and the complexities encountered in real-world production environments, such as equipment failures, additional resource constraints, or changeover times (Groth et al., 2024). Research acknowledging these complexities usually evaluates their algorithms using self-created production scenarios, which reduces comparability (Groth et al., 2024).

This methodological gap poses significant challenges for the industrial application of the developed scheduling algorithms in research. Without standardized comparisons, companies struggle to identify the most suitable algorithms for their particular production scenarios, leaving the landscape of scheduling algorithms fragmented and difficult to navigate. Furthermore, this incentivizes researchers to develop algorithms compatible with idealized theoretical benchmarks, which limits the practical applicability of their research.

To bridge this gap, a standardized and realistic benchmark based on real-world production scenarios is essential. Such a benchmark would incentivize research to focus explicitly on real-world production constraints, driving the development of more robust, adaptable, and practically applicable scheduling algorithms. Furthermore, by providing a uniform evaluation framework, it would increase transparency and comparability, thereby enhancing the adoption and effectiveness of advanced scheduling solutions in industrial practice. In addressing this issue, our research question is:

RQ

How can a benchmark based on real-world productions be designed to rank production scheduling algorithms?

The paper is structured as follows: Section 2 introduces production scheduling and discusses related research. Section 3 describes the methodological approach taken in this research. Section 4 outlines the design of the proposed benchmark, detailing problem identification, requirements, and demonstration. Section 5 critically evaluates the benchmark, followed by a discussion of the implications and limitations in Section 6. Finally, Section 7 provides concluding remarks and recommendations for future research.

2 Production Scheduling and Related Research

As part of production planning and control, production scheduling is the time-related allocation of production tasks to production resources—such as machines or workstations—for released jobs, with the aim of optimizing objectives like minimizing makespan, lead times, tardiness, or cost (Pinedo, 2016; Toader, 2017). Given its pivotal role in manufacturing and service industries, a rigorous understanding of

scheduling methods, problem types, extensions, and related benchmarks is essential for designing benchmarks that accurately assess algorithmic performance intended for real-world usage.

Approaches to solving scheduling problems fall into three broad categories (Fazel Zarandi et al., 2020): exact methods (e.g., mixed-integer linear programming), which guarantee optimality but do not scale to realistic problem sizes (Garey & Johnson, 1979); priority-rule heuristics (e.g., first-in-first-out, shortest processing time), which are fast but often produce suboptimal schedules (Pinedo, 2016); and metaheuristics (e.g., genetic algorithms, simulated annealing, tabu search), which balance solution quality and computational effort by exploring the solution space via guided randomness (Maccarthy & Liu, 1993). More recently, hybrid and learning-based methods—such as hyper-heuristics, reinforcement learning, or deep neural policies—have been proposed to select or generate scheduling rules adaptively in dynamic environments (Tamaki et al., 1999), or to make decisions on assignments directly (Waschneck et al., 2018).

The scheduling literature distinguishes several standard shop-floor topologies: single-machine, parallel-machine, flow-shop, job-shop, and open-shop settings (Pinedo, 2016). Among these, the job-shop scheduling problem (JSSP)—where each job comprises a sequence of operations on specific machines—is computationally challenging, being NP-hard (Garey & Johnson, 1979). Thus, exact solvers can effectively handle small to medium instances of these problems, but fail as the number of jobs or machines grows (Garey & Johnson, 1979).

Real-world production environments impose further constraints that standard models omit, necessitating benchmark enhancements to reflect practical complexity. For example, sequence-dependent setup times ("changeovers") introduce additional complexity and optimization potential (Rolf et al., 2020). Limited buffer capacities before and after machines prevent congestion and require effective storage management (Qu et al., 2016). Additionally, job priorities or release dates mandate differentiated treatment of jobs (Nie et al., 2010; Zhou & Yang, 2019). Moreover, integrated transportation scheduling (e.g., coordinating automated guided vehicles) (Li et al., 2022) and multiobjective criteria (balancing makespan, tardiness, and costs) are prevalent in factories (Lu et al., 2021). Furthermore, factory layouts can be more complex than the standard shop-floor topologies, consisting of machines with differing speeds, alternative machines, recirculation, and other complexities (Pinedo, 2016), underscoring the need for benchmarks that capture these interdependencies to create directly relevant algorithms for practice.

Classical scheduling benchmarks laid the foundation for comparative algorithm analysis but remain limited in scope. Fisher and Thompson (1963) introduced small-to-medium-sized JSSP instances that have since served as test cases for exact and heuristic methods. Taillard (1993) expanded this effort by generating larger and more varied benchmarks across flow-shop, job-shop, and open-shop configurations, enabling the evaluation of scalability and shop-floor diversity among standard scheduling problems. While valuable and often used, these benchmarks lack real-world constraints.

To capture multiobjective and cell-level scheduling challenges, Deliktas et al. (2023) introduced a benchmark dataset for flexible job-shop cell scheduling, focusing on two-machine FJSP instances with availability constraints and conflicting objectives, such as makespan versus workload balance. This dataset facilitates the evaluation of algorithms' ability to navigate the trade-offs inherent in multiobjective scheduling and limited resource availability.

Deliktaş et al. (2024) extend their work by proposing a richer benchmark for multiobjective flexible job-shop scheduling in cellular manufacturing environments, incorporating sequence-dependent family setups, transportation times between cells, and recirculation of jobs through multiple processing loops. By embedding such practical features, this benchmark demonstrates how complexity can be increased to challenge advanced algorithms and fosters the development of scheduling algorithms relevant to real-world applications. Still, complexity from practice is missing in this benchmark.

Taken together, these related research efforts provide valuable testbeds but often cover only subsets of practical requirements. Building upon their results, we develop a benchmark based on real-world productions.

3 Methodology

To answer the research question and, therefore, design a benchmark based on real-world productions to rank production scheduling algorithms, we apply the problem-centered mix-method design science research approach from Peffers et al. (2007), as depicted in Fig. 1.

First, based on a structured literature review (Webster & Watson, 2002), we identified the problem and derived the complexities required in our benchmark. Using this as a basis, in the second step, we formulated an overarching objective and requirements that the designed benchmark must fulfil. In the third step, building upon the requirements and a real-world production, we developed the benchmark.

Afterwards, in the demonstration, we used scheduling rules to create schedules for the benchmark problem, demonstrating its ability to be scheduled. Additionally, domain experts validated the resulting schedules and the plausibility of the scheduling problem. Finally, we evaluated the distribution of the performance of randomly created schedules and benchmarked multiple priority rules and a reinforcement learning-based scheduling algorithm. If the results of the demonstration and the evaluation show that the requirements were not meaningfully implemented, we refined the developed benchmark iteratively.

4 Design of the Benchmark

In the following sections, we present the results of the five-step methodology. Section 4.1 defines the problem identified. Based on this, Section 4.2 derives the requirements for designing the benchmark. In Section 4.3, we present the design process of the benchmark and its incorporated production environment. Finally, in Section 4.4, we demonstrate the benchmark before evaluating it in Section 4.5.

Sections 4.4 and 4.5 demonstrate and evaluate the benchmark, incorporating the iterations according to our methodology, presenting the final results.

4.1 Problem Identification

In prior research (Groth et al., 2024), we conducted a literature review and found that changeovers were only included in 18 % of the literature examined. Especially with a variety of products, changeovers enable improvements through effective scheduling. Limited buffer sizes were only implemented in 4 % of the literature, while job prioritization was implemented in 10 %. Only 6 % of the literature discussed the dynamics of the shop floor, and 3% discussed transportation on the shop floor, such as with automated guided vehicles. Although scheduling is a well-researched field, these results indicate that scheduling research often overlooks the complexity found in practice.

At the same time, established benchmarks focus on standard literature problems while real-world complexity is often not or only partially included (Taillard, 1993). Considering the missing benchmark, which includes real-world complexity, it is understandable that scheduling research often ignores real-world complexity, because the performance of developed algorithms, which incorporate the identified additional constraints, is not directly comparable to that of other research. Thus, we identified the problem that scheduling research with real-world complexity typically has no meaningful evaluation.

4.2 Requirements for the Design of the Benchmark

Based on the identified problem, we set the goal to develop a benchmark that includes real-world complexity. To guide the design of the benchmark, we derive requirements for creating the benchmark and those implemented within it.

To solve the identified problem, first, the benchmark must have sufficient complexity and practice relevance [B1]. The sufficient complexity is the basis for the capability to differentiate algorithms (Deliktaş et al., 2024). The benchmark must be designed in a way that it can differentiate between the ability of scheduling algorithms to create high-quality schedules [B2] (Taillard, 1993). This differentiation capability can be achieved by creating a production environment where scheduling decisions have consequences, such as buffer storage being full, or by implementing intelligent scheduling decisions that minimize idle times and changeovers. Furthermore, scheduling objectives vary in practice. Even though research often uses makespan as an evaluation criterion (Groth et al., 2024), and it serves as a good proxy for other goals, practice may have different goals, e.g., timeliness (Duenas & Petrovic, 2008). Therefore, the benchmark must have variants for different objective functions [B3]. Finally, as research increasingly applies machine learning methods (Fazel Zarandi et al., 2020), we must consider the consequences resulting from this shift in the benchmark creation. Generally, machine learning methods require training data that differs from evaluation data to prevent overfitting, which can occur when training is performed on the same data used for evaluation, as is often not provided in established

benchmarks (Taillard, 1993). Thus, the benchmark must have resistance to overfitting [B4]. Table 1 summarizes the requirements for the benchmark.

#	Requirements for the benchmark
B1	Sufficient complexity and practice relevance
B2	Differentiation capability
B3	Variants for different objective functions
B4	Resistance to overfitting

Table 1: Requirements for the benchmark

Additionally, to achieve the targeted practice relevance [B1], we derive requirements that must be implemented in the benchmark. Table 2 summarizes these requirements. Real-world productions are typically more complex than scheduling problem types defined in research (Pinedo, 2016). Instead, we find complex routing options and operation dependencies. Thus, the benchmark must extend on and implement all problem classes found in research [I1]. Furthermore, changeover times exist when machines or workstations need to be prepared to process a specific operation (Rolf et al., 2020). We decided to implement sequence-dependent changeovers [I2], where the required time is dependent on the combination of the material before and after the changeover. For example, ovens require sequence-dependent changeovers if the materials being produced require different temperatures. In this case, changeovers represent the time required to transition from one temperature to another. Using sequence-dependent changeovers, product-dependent changeovers are also mappable. Limited buffer space for storing intermediary materials necessitates scheduling that takes this constraint into account (Qu et al., 2016). Thus, the benchmark must include buffers at production resources with a flexible size to account for different production environments [I3].

#	Requirements implemented in the benchmark
I1	All problem classes
I2	Sequence-dependent changeovers
I3	Support buffers of flexible size at production resources
I4	Personnel
I5	Minimum storage periods for intermediary materials
I6	Prioritisation of jobs
I7	Transportation
I8	Production aids
I9	Simultaneous production conditions
I10	Jobs with release dates

Table 2: Requirements implemented in the benchmark

Skilled personnel might be required for certain production tasks (Patel et al., 1999). Personnel may not be available at all times in sufficient quantities, which could render schedules invalid if not taken into account at schedule creation. Thus, the benchmark must include personnel required for specific production tasks in limited quantities [I4]. After certain operations, materials may require a minimum storage period before subsequent operations may begin [I5] (Jian et al., 2004). For example, this period can represent cooling or drying times. Furthermore, jobs can have priorities based on the consequences of a missed deadline [I6] (Zhou et al., 2019). For example, a production job linked to an important customer should have a high priority, while production jobs aimed at filling stock should have a low priority. These priorities should be taken into account for benchmark variants that include the timeliness of the generated schedules.

In some productions, transportation between production resources is a limiting factor, for example, when distances are long between different shop floors or there is only limited capacity to transport goods. Thus, the production environment in the benchmark must be designed to accommodate the scheduling of transportation resources [I7] (Li et al., 2022). Additionally, limited capacity may also occur due to a shortage of production aids required for certain operations (Graham et al., 1979). For example, specific tools may be available in limited quantities, such that some operations may not be scheduled simultaneously [I8]. Furthermore, the simultaneous production of certain operations may also be limited by other reasons, which formulate simultaneous production conditions [I9] (Xue et al., 2004). For example, an oven capable of processing multiple operations can only process operations requiring the same temperature. Finally, production jobs have release dates defining the earliest time production of the job may begin [I10] (Nie et al., 2010). This release date may be provided by a delivery date of required components or used to influence the production order.

4.3 Design of the Benchmark

We use the identified requirements to guide the design process of the benchmark. Furthermore, we applied domain knowledge and incorporated real-world elements to enhance the real-world relevance of the developed benchmark. Due to confidentiality requirements, we cannot provide details on the real-world productions referenced. We refined the following results through design cycles in accordance with our methodology and present the final results.

Regarding the requirements for the benchmark, we ensure sufficient complexity and practice relevance [B1, see Table 1] by implementing the requirements identified to be implemented in the benchmark. Along with the requirement for differentiation capability [B2], we aim to include as many production resources and materials as necessary to create a solution space large enough to demonstrate the solution quality of an algorithm, while not adding unnecessary complexity that would reduce the interpretability of the results. To provide variants of the benchmarks with different objective functions [B3], we found makespan, machine utilization, prioritized timeliness, and a combination of makespan and prioritized timeliness as relevant optimization goals (Groth et al., 2024; Tavakkoli-Moghaddam et al., 2008; Vela et al., 2010; Zhao et al., 2006).

To mitigate overfitting of algorithms to the benchmark problem and, thus, increase robustness of the benchmark, we suggest the following benchmarking process: Instead of one benchmark instance to be scheduled, we provide 100 randomly generated instances to be used for benchmarking. Then, the KPI used for benchmarking is the provided average relative improvement to the random priority rule. For the training of algorithms, researchers should randomly generate training situations. This benchmarking process hinders the training of algorithms on specific situations and ensures solution quality in changing environments. Additionally, this provides the added benefit of including situations with high and low workloads in the benchmark.

For the production environment, we decided to include two shop floors connected by a transportation resource (e.g., a forklift), shuttling between both shop floors [I7, see Table 2]. Both shop floors have limited storage spaces; therefore, they each have a shared buffer, implementing I3. Afterwards, we decided on the remaining production resources and implemented the identified requirements. Besides the transportation resource, we identified six groups of production resources as relevant: Production resource group 1 (R1) implements I8 by requiring limited production aids for each production step. Additionally, it implements recirculation, included in I1. I1 is also implemented in R2, which represents an optional production step that is not required for all material types. R3 includes two resources and implements I9, limiting which materials can be produced simultaneously. R4 and R5 share a limited number of personnel required for the production steps [I4], implemented using production aids. Finally, R6 includes production resources with slower and faster production speeds, extending standard literature problems (Pinedo, 2016) [I1].

We decided to implement five materials (M1-M5) because this number strikes a balance between necessary changeovers and variety while still offering the potential to reduce the amount of changeovers if the same material is scheduled in succession on a production resource. This number balances the required complexity and interpretability for humans. The materials follow different paths through the production process, combining elements of flow shop, job shop, and open shop, while introducing more complex dependencies that allow operations to occur in parallel on different resources [I1]. Furthermore, materials require changeovers at production resources [I2], which are primarily randomly defined. Minimum storage periods are required for all materials exiting R2 [I5]. The production speed of materials on production resources was initially set randomly and then checked to ensure a balanced distribution between resource groups, thereby guaranteeing sufficient scheduling quality across all production resources. Nevertheless, we decided against random production speed for every benchmark instance, as this could cause imbalances and would not be relevant in practice, as scheduling algorithms will be trained on a given production scenario, where usually only the available jobs change, while production speeds and available production resources generally remain the same. Figure 2 visualizes the resulting production layout.

A job requires 1-19 materials of one type to be produced. We assume that operations are only processable in their entirety, and subsequent operations may only begin once the previous operations are fully completed (Graham et al., 1979). If production batches are to be split, we assume that multiple jobs are required. The material type is chosen randomly, while M2 is chosen half as often to simulate production requirements that are not evenly balanced. Jobs have a priority value [I6] between 0.5 and 3, where 1 equals normal priority. Half of the generated jobs have a priority of 1, while 25% have a lower priority and 25% have a higher priority. These priority values are only relevant for benchmark variants with a goal function that includes prioritized timeliness. To make the benchmark more realistic (Shaw et al., 2000), we included 36 work-in-progress (WIP) jobs, where some operations were already completed, thereby filling the buffer at the beginning of the scheduling horizon. At the same time, all production resources have a chance to be blocked in the beginning, simulating an already running production. Afterwards, we add 36 jobs with all operations required. Finally, 18 jobs are added with a release date [I10]. The release date and deadline of the jobs increase randomly with each job added.

To communicate the benchmark, we used the problem representation presented by Groth and Schumann (2023). We transformed the data structure presented there into a JSON format. Each JSON file includes one benchmark instance, while all 100 instances of a benchmark variant are cumulated in a folder in the supplements. There, we also provide the results of the "random" priority rule to use as a baseline for the benchmarking process and the generator to create training data for machine learning-based scheduling algorithms. We provide all the results necessary to use the benchmark in the supplements.

4.4 Setup and Demonstration of the Benchmark

According to our methodology, in the demonstration, we used the first-in-first-out (FIFO) priority rule to create schedules on multiple benchmark instances generated and conducted expert plausibility checks. To verify this, we defined criteria for the requirements to determine if they are implemented in a manner that has an impact, but only to an extent expected in real-world production. If a criterion was not met, we refined the relevant design details iteratively to fulfill the criterion.

The designed production environment incorporates elements of various problem classes [I1], including flow shop elements, parallel machine groups, open shop elements (M4 on resources R4, R5, R6), circular production (R2), and parallel tasks. The goal with due dates was to create situations where it is difficult to fulfill all jobs within the due date, typically resulting in multiple jobs being delayed, thus providing improvement potential. In the demonstration, we found a tardiness rate of 2% - 20%, depending on the random generation. Changeovers [I2] are implemented with varying significance at the different production resources. While R4 and R6 have high changeover times, R1 and R2 have medium changeover times. R5 requires low changeover times, and R3 has no changeovers. We aimed to distribute the load evenly across all production resources to prevent any bottlenecks. Therefore, our criterion was to maintain an equal total load while varying the significance of the changeovers between the production resources. To achieve this, we slightly adjusted the production times on production resources with high changeover times. For buffer storage, we aimed for a low impact, as otherwise we expect the shop floor design to be altered [I3]. Thus, we aimed for an impact of 3% in tests with priority rules compared to schedules with infinite buffer sizes. We iteratively found that 275 is a suitable buffer size that fulfills these aims in the designed production environment and for the available jobs.

Limited available personnel are modeled as production aids on R4 and R5 [I4], while minimum storage periods are required for production steps after R2 [I5]. Prioritizations of jobs [I6] are intended to simulate an environment where most jobs are of normal priority, while specific jobs are of high importance, such as those for high-margin customers. Other jobs represent make-to-stock orders with low priority. Transportation is modeled as a production resource, connecting both shop floors [I7]. For production aids, we decided to model tools specifically required for a material on R1, where each tool is available only once, thereby limiting scheduling options. We implemented production aids [I8] with a similar goal to the buffer spaces, as otherwise, we expect the company to acquire further tools.

We assume R3 to be a machine that handles two batches simultaneously. Due to the material handling, two material combinations cannot be processed at the same time, thus, implementing simultaneous production conditions. We aim not to create a bottleneck in the benchmark at R3; therefore, we only added two material combinations [I9]. Still, in our demonstration with priority rules, we observed idle times resulting from poor scheduling decisions, indicating options to create higher-performing schedules if scheduling algorithms handle these constraints intelligently. Finally, we added release dates for jobs to enable us to measure the ability of scheduling algorithms to incorporate later tasks into earlier decisions [I10]. We assume in our scenario that a production planner plans to start these jobs with release dates later in time to limit WIP.

To demonstrate the complexity [B1], we estimate that ~10^{1800} schedules are possible for each generated situation, presenting a solution space that is not trivially solvable. Due to the random generation of jobs and WIP jobs, the resulting situations can be more challenging and easier to solve. To demonstrate the differentiation capability [B2], we applied the priority rule "random" 100 times to the generated problem. We found a standard deviation in the resulting goal value of ~8%, indicating that even with random decisions, the resulting quality varies significantly, presenting opportunities for improvement.

4.5 Evaluation of the Benchmark

We evaluated the benchmark in a two-step process. First, we used the "random" priority rule and evaluated the distribution of the results. Then, we applied priority rules and a reinforcement-learning-based approach to evaluate the performance of these algorithms.

For the first step, we applied the "random" priority rule 100 times to each benchmark instance. Averaged over the 100 benchmark instances, we found a standard deviation of 8.6% for the combination variant (makespan: 6.9%, machine utilization: 5.2%, prioritized timeliness: 6.3%). These results show that the benchmark problem can effectively distinguish between good and poor scheduling approaches. Furthermore, we analyzed whether we found duplicate resulting goal values for a benchmark instance. Frequent duplicates would indicate that too many scheduling decisions would not impact the performance of the resulting schedule. We found duplicates in only 0.47% of the generated schedules. Therefore, we derive that the options included in the benchmark are relevant to the performance of the resulting schedule.

Afterwards, we applied scheduling algorithms to the benchmark problem. We applied the priority rules first-in-first-out (FIFO; schedule the operation that first arrived out of all the available operations), due date (schedule the operation with the most urgent due date), and changeover (schedule the operation with the least changeover time) (Kistner & Steven, 2001). Furthermore, we applied a reinforcement-learning-based scheduling approach that was trained on a different real-world production. We expected the priority rule FIFO to perform best, as the jobs are already ordered according to their due date, and FIFO often yields decent results in various scenarios (Park et al., 2019; Wang et al., 2017). At the same time, it prevents any idle time on production resources. Those idle times may occur for the other priority rules. Thus, we expect worse performance for the priority rules due date and changeover. For the reinforcement-learning-based approach, we expect good performance but not inferior performance as the algorithm is trained on a different production.

Figure 3 presents the benchmark results for the variants. The results show the average relative improvement of the benchmark algorithm compared to the average performance of the "random" priority rule. We find that the FIFO priority rule performs the best in the benchmark, while due date performs worse than a randomly generated schedule in some variants. Otherwise, our results match the expectations.

5 Discussion

To benchmark one's algorithm, users must schedule all 100 benchmark instances of the chosen variant and then calculate the relative distance between the provided benchmark baseline for every instance, where the baseline is the average result of the "random" priority rule. The average relative distance of all 100 instances is the benchmark performance. Through this benchmarking process, we aim to prevent overfitting of algorithms by extensive training on the given benchmark, as well as reduce the impact of randomness in random-guided algorithms (e.g., genetic algorithms). If chance were to play a significant role, we would expect researchers to optimize their published results by repeated benchmark executions. The developed benchmarking process makes this more difficult as the randomness averages over the 100 benchmark instances. Furthermore, we suggest that developers of machine learning-based algorithms generate training data using the data generator to prevent overfitting during training and create more robust scheduling algorithms. To maintain the meaning of the results, we decided against directly using the results of the goal function (e.g., makespan), as good results will differ between benchmark instances due to the random generation of these instances. Thus, we found the relative improvement over the average random result to be a more meaningful metric.

With the 10 requirements implemented in the benchmark, we extend existing benchmarks by incorporating real-world constraints to evaluate how scheduling algorithms perform when these constraints are present. Nevertheless, some of the included constraints will not be relevant in some productions. Still, we expect algorithms that perform well in this benchmark to perform equally well in production environments where fewer constraints are present. Thus, the results of this benchmark provide a starting point for identifying suitable algorithms for production and guiding the development of real-world, relevant scheduling algorithms.

For research, our benchmark creates value by providing a means to evaluate the performance of scheduling algorithms that incorporate real-world constraints. Hereby, it offers the possibility to guide the development of scheduling algorithms more relevant to real-world production environments as it provides a data structure, training data, and the possibility to evaluate the performance of developed scheduling algorithms. For practice, the benchmark has the potential to offer further algorithms that can be directly implemented due to the included constraints. Thus, we aim to foster an ecosystem of production scheduling research that is directly relevant to real-world production environments.

6 Conclusion

This paper uses a design-science approach to create a realistic benchmark for comparing production scheduling algorithms under real-world constraints. We began by identifying the gap between idealized, standard benchmarks and the complexities faced in practice—such as sequence-dependent changeovers, limited buffers, personnel constraints, transportation, and release dates—through a structured literature review. Guided by the problem-centered mix-method methodology, we derived high-level requirements (B1–B4) for sufficient complexity, differentiation capability, objective-function

variants, and resistance to overfitting, as well as implementation requirements (I1–I10) capturing diverse practical constraints. Building on these requirements, we designed a two-shop-floor scenario that implements the identified requirements, generated 100 random instances for each of the four variants, and defined a benchmarking process based on relative improvement over the "random" priority-rule baseline. We demonstrated and validated the benchmark using priority rules and a reinforcement-learning approach, and evaluated its ability to distinguish between algorithm performance and mitigate overfitting.

In answering our research question—"How can a benchmark based on real-world productions be designed to rank production scheduling algorithms?"—we show that a benchmark built on realistic shop-floor modeling, multiple randomized instances, and a relative-improvement metric provides both the complexity and comparability required for rigorous evaluation. The 100-instance approach ensures robustness against overfitting and randomness, while the JSON-based data structure enables easy integration into scheduling algorithms. The resulting benchmark is available in the supplements.

Despite its breadth, our benchmark reflects a single production layout and set of parameter distributions; it may not incorporate all constraints found in practice or accurately depict all performance measurements observed in practice (e.g., energy usage). If other variants or constraints are found relevant in future research, the present benchmark can be extended or further variants added. The latter is straightforward, as only a new goal function needs to be defined.

Additionally, our demonstration focused on priority-rule heuristics and a pre-trained reinforcement-learning model, leaving metaheuristics and other machine learning-based methods less explored. Future research should extend and apply state-of-the-art scheduling algorithms such as reinforcement learning algorithms and transformer models to the benchmark to measure their performance.

Furthermore, the benchmark has a limitation due to its size. We aimed for a benchmark size that can differentiate the solution quality of scheduling algorithms while still yielding resulting schedules that are interpretable by humans. Due to this goal, our benchmark does not have an extensive size and is not meant for researchers competing to find the best schedule. Instead, the benchmark is intended to evaluate the real-world performance of scheduling algorithms.

For researchers, our benchmark provides a uniform evaluation framework that encourages the development of algorithms tailored to practical constraints and enables direct comparison across studies. Practitioners gain a tool for pre-testing scheduling solutions under realistic conditions, facilitating informed selection and implementation of algorithms in their specific production contexts.

Declarations

Author Contribution

M.G. wrote the manuscript text. M.S. reviewed the results and the manuscript.

Data Availability

Data generator, benchmark instances, and baseline comparison values are published in the supplementary material.

References

1. Baykasoğlu, A., & Ozsoydan, F. B. (2018). Dynamic scheduling of parallel heat treatment furnaces: A case study at a manufacturing system. *Journal of Manufacturing Systems*, *46*, 152–162. https://doi.org/10.1016/j.jmsy.2017.12.005
2. Da Silveira, G., Borenstein, D., & Fogliatto, F. S. (2001). Mass customization: Literature review and research directions. *International Journal of Production Economics*, *72*(1), 1–13. https://doi.org/10.1016/S0925-5273(00)00079-7
3. Deliktas, D., Özcan, E., Üstün, Ö., & Torkul, O. (2023). *A Benchmark Dataset for Multiobjective Flexible Job Shop Cell Scheduling*. https://doi.org/10.2139/ssrn.4545801
4. Deliktaş, D., Özcan, E., Ustun, O., & Torkul, O. (2024). A benchmark dataset for multiobjective flexible job shop cell scheduling. *Data in Brief*, *52*, 109946. https://doi.org/10.1016/j.dib.2023.109946
5. Duenas, A., & Petrovic, D. (2008). Multiobjective genetic algorithm for single machine scheduling problem under fuzziness. *Fuzzy Optimization and Decision Making*, *7*(1), 87–104. https://doi.org/10.1007/s10700-007-9026-6
6. Fazel Zarandi, M. H., Sadat Asl, A. A., Sotudian, S., & Castillo, O. (2020). A state of the art review of intelligent scheduling. *Artificial Intelligence Review*, *53*(1), 501–593. https://doi.org/10.1007/s10462-018-9667-6
7. Fisher, H., & Thompson, G. L. (1963). Probabilistic Learning Combinations of Local Job-Shop Scheduling Rules in Industrial Scheduling. *Industrial Scheduling*.
8. Freier, P. (2020). *Empirische Erkenntnisse und Gestaltungsansätze für Entscheidungsunterstützungssysteme in der Ablaufplanung im Kontext von Cyber-Physischen Systemen*. Cuvillier Verlag.
9. Garey, M. R., & Johnson, D. S. (1979). *Computers and intractability* (Vol. 174). freeman San Francisco.
10. Graham, R. L., Lawler, E. L., Lenstra, J. K., & Kan, A. (1979). Optimization and Approximation in Deterministic Sequencing and Scheduling: a Survey. In *Annals of Discrete Mathematics. Discrete Optimization II, Proceedings of the Advanced Research Institute on Discrete Optimization and Systems Applications of the Systems Science Panel of NATO and of the Discrete Optimization Symposium co-sponsored by IBM Canada and SIAM Banff, Aha. and Vancouver* (Vol. 5, pp. 287–326). Elsevier. https://doi.org/10.1016/S0167-5060(08)70356-X
11. Groth, M., & Schumann, M. (2023). Design of a Reference Architecture for Production Scheduling Applications based on a Problem Representation including Practical Constraints. *ACIS 2023 Proceedings*. https://aisel.aisnet.org/acis2023/8/

12. Groth, M., Schumann, M., & Nickerson, R. C. (2024). Characteristics of Production Scheduling Problems in the Era of Industry 4.0 – A Review of Machine Learning Algorithms for Production Scheduling. In F. J. G. Silva, L. P. Ferreira, J. C. Sá, M. T. Pereira, & C. M. A. Pinto (Eds.), *Lecture Notes in Mechanical Engineering. Flexible Automation and Intelligent Manufacturing: Establishing Bridges for More Sustainable Manufacturing Systems* (pp. 119–127). Springer Nature Switzerland. https://doi.org/10.1007/978-3-031-38165-2_15
13. Jian, W., Xue, Y., & Du Hongbin (2004). Optimum steelmaking cast plan using improved genetic algorithm. In *2004 IEEE International Conference on Systems, Man and Cybernetics (IEEE Cat. No.04CH37583)* (pp. 4284–4289). IEEE. https://doi.org/10.1109/ICSMC.2004.1401204
14. Kistner, K.-P., & Steven, M. (2001). *Produktionsplanung*. Physica-Verlag HD. https://doi.org/10.1007/978-3-642-57546-4
15. Kück, M., Ehm, J., Hildebrandt, T., Freitag, M., & Frazzon, E. M. (2016). Potential of data-driven simulation-based optimization for adaptive scheduling and control of dynamic manufacturing systems. In T. M. K. Roeder, P. I. Frazier, R. Szechtman, E. Zhou, T. Huschka, & S. E. Chick (Eds.), *Winter Simulation Conference* (pp. 2820–2831). IEEE. https://doi.org/10.1109/WSC.2016.7822318
16. Lasi, H., Fettke, P., Kemper, H.-G., Feld, T., & Hoffmann, M. (2014). Industry 4.0. *WIRTSCHAFTSINFORMATIK*, *56*(4), 239–242. https://doi.org/10.1007/s11576-014-0424-4
17. Li, W [Wangming], Han, D., Gao, L., Li, X., & Li, Y. (2022). Integrated Production and Transportation Scheduling Method in Hybrid Flow Shop. *Chinese Journal of Mechanical Engineering*, *35*(1). https://doi.org/10.1186/s10033-022-00683-7
18. Liang, Y., Sun, Z., Song, T., Chou, Q., Fan, W., Fan, J., Rui, Y., Zhou, Q., Bai, J., Yang, C., & Bai, P. (2022). Lenovo Schedules Laptop Manufacturing Using Deep Reinforcement Learning. *INFORMS Journal on Applied Analytics*, *52*(1), 56–68. https://doi.org/10.1287/inte.2021.1109
19. Lu, C., Zhang, B., Gao, L., Yi, J., & Mou, J. (2021). A Knowledge-Based Multiobjective Memetic Algorithm for Green Job Shop Scheduling With Variable Machining Speeds. *IEEE Systems Journal*, 1–12. https://doi.org/10.1109/JSYST.2021.3076481
20. Maccarthy, B. L., & Liu, J. (1993). Addressing the gap in scheduling research: a review of optimization and heuristic methods in production scheduling. *International Journal of Production Research*, *31*(1), 59–79. https://doi.org/10.1080/00207549308956713
21. Monostori, L. (2014). Cyber-physical Production Systems: Roots, Expectations and R&D Challenges. *Procedia CIRP*, *17*, 9–13. https://doi.org/10.1016/j.procir.2014.03.115
22. Nie, L., Shao, X., Gao, L., & Li, W [Weidong] (2010). Evolving scheduling rules with gene expression programming for dynamic single-machine scheduling problems. *The International Journal of Advanced Manufacturing Technology*, *50*(5-8), 729–747. https://doi.org/10.1007/s00170-010-2518-5
23. Park, I.-B., Huh, J., Kim, J., & Park, J. (2019). A Reinforcement Learning Approach to Robust Scheduling of Semiconductor Manufacturing Facilities. *IEEE Transactions on Automation Science and Engineering*, 1–12. https://doi.org/10.1109/TASE.2019.2956762

24. Patel, V., ElMaraghy, H. A., & Ben-Abdallah, I. (1999). Scheduling in dual-resources constrained manufacturing systems using genetic algorithms. In *1999 7th IEEE International Conference on Emerging Technologies and Factory Automation. Proceedings ETFA '99 (Cat. No.99TH8467)* (pp. 1131–1139). IEEE. https://doi.org/10.1109/ETFA.1999.813116
25. Peffers, K., Tuunanen, T., Rothenberger, M. A., & Chatterjee, S. (2007). A Design Science Research Methodology for Information Systems Research. *Journal of Management Information Systems*, *24*(3), 45–77. https://doi.org/10.2753/MIS0742-1222240302
26. Pinedo, M. L. (2016). *Scheduling*. Springer International Publishing. https://doi.org/10.1007/978-3-319-26580-3
27. Qu, S., Wang, J [Jie], Govil, S., & Leckie, J. O. (2016). Optimized Adaptive Scheduling of a Manufacturing Process System with Multi-skill Workforce and Multiple Machine Types: An Ontology-based, Multi-agent Reinforcement Learning Approach. *Procedia CIRP*, *57*, 55–60. https://doi.org/10.1016/j.procir.2016.11.011
28. Rolf, B., Reggelin, T., Nahhas, A., Lang, S., & Müller, M. (2020). Assigning dispatching rules using a genetic algorithm to solve a hybrid flow shop scheduling problem. *Procedia Manufacturing*, *42*, 442–449. https://doi.org/10.1016/j.promfg.2020.02.051
29. Schuh, G., Potente, T., Thomas, C., Hauptvogel, A., Mueller, C., & Stollwerk, A. (2012). Web-based value stream oriented simulation of production control. In C. Laroque (Ed.), *2012 Winter Simulation Conference* (pp. 1–10). IEEE. https://doi.org/10.1109/WSC.2012.6465080
30. Shaw, K. J., Lee, P. L., Nott, H. P., & Thompson, M. (2000). Genetic algorithms for multiobjective scheduling of combined batch/continuous process plants. In *Proceedings of the 2000 Congress on Evolutionary Computation. CEC00 (Cat. No.00TH8512)* (pp. 293–300). IEEE. https://doi.org/10.1109/CEC.2000.870309
31. Taillard, E. (1993). Benchmarks for basic scheduling problems. *European Journal of Operational Research*, *64*(2), 278–285. https://doi.org/10.1016/0377-2217(93)90182-M
32. Tamaki, H., Kryssanov, V. V., & Kitamura, S. (1999). A simulation engine to support production scheduling using genetics-based machine learning. In K. Mertins, O. Krause, & B. Schallock (Eds.), *Global Production Management* (pp. 482–489). Springer US. https://doi.org/10.1007/978-0-387-35569-6_59
33. Tavakkoli-Moghaddam, R., Rahimi-Vahed, A. R., & Mirzaei, A. H. (2008). Solving a multiobjective no-wait flow shop scheduling problem with an immune algorithm. *The International Journal of Advanced Manufacturing Technology*, *36*(9-10), 969–981. https://doi.org/10.1007/s00170-006-0906-7
34. Toader, F. A. (2017). Production scheduling in flexible manufacturing systems: A state of the art survey. *Journal of Electrical Engineering, Electronics, Control and Computer Science*, *3*(1), 1–6.
35. Vela, C. R., Varela, R., & González, M. A. (2010). Local search and genetic algorithm for the job shop scheduling problem with sequence dependent setup times. *Journal of Heuristics*, *16*(2), 139–165. https://doi.org/10.1007/s10732-008-9094-y

36. Wang, J [Jinzhi], Qu, S., Wang, J [Jie], Leckie, J. O., & Xu, R. (2017). Real-Time Decision Support with Reinforcement Learning for Dynamic Flowshop Scheduling. In *Smart SysTech 2017: European Conference on Smart Objects, Systems and Technologies : 20-21 June 2017* (pp. 1–9). VDE.
37. Waschneck, B., Reichstaller, A., Belzner, L., Altenmuller, T., Bauernhansl, T., Knapp, A., & Kyek, A. (2018). Deep reinforcement learning for semiconductor production scheduling. In *2018 29th Annual SEMI Advanced Semiconductor Manufacturing Conference (ASMC)* (pp. 301–306). IEEE. https://doi.org/10.1109/ASMC.2018.8373191
38. Webster, J., & Watson, R. T. (2002). Analyzing the Past to Prepare for the Future: Writing a Literature Review. *MIS Quarterly*, *26*(2), xiii–xxiii.
39. Xue, Y., Yang, Q., & Ban, W. (2004). Improved genetic algorithm with adaptive operators for integrated steelmaking optimum cast plan. In *30th Annual Conference of IEEE Industrial Electronics Society, 2004. IECON 2004* (pp. 1279–1283). IEEE. https://doi.org/10.1109/IECON.2004.1431760
40. Yang, W., & Takakuwa, S. (2017). Simulation-based dynamic shop floor scheduling for a flexible manufacturing system in the industry 4.0 environment. In W. K. V. Chan (Ed.), *Winter Simulation Conference* (pp. 3908–3916). IEEE. https://doi.org/10.1109/WSC.2017.8248101
41. Zhao, F., Zhang, Q., & Yang, Y. (2006). An Improved Particle Swarm Optimization(PSO) Algorithm and Fuzzy Inference Systems Based Approach to Process Planning and Production Scheduling Integration in Holonic Manufacturing System (HMS). In *2006 International Conference on Machine Learning and Cybernetics* (pp. 396–401). IEEE. https://doi.org/10.1109/ICMLC.2006.259102
42. Zhou, Y., & Yang, J.-J. (2019). Automatic design of scheduling policies for dynamic flexible job shop scheduling by multiobjective genetic programming based hyper-heuristic. *Procedia CIRP*, *79*, 439–444. https://doi.org/10.1016/j.procir.2019.02.118
43. Zhou, Y., Yang, J.-J., & Zheng, L.-Y. (2019). Multi-Agent Based Hyper-Heuristics for Multiobjective Flexible Job Shop Scheduling: A Case Study in an Aero-Engine Blade Manufacturing Plant. *IEEE Access*, *7*, 21147–21176. https://doi.org/10.1109/ACCESS.2019.2897603

Figures

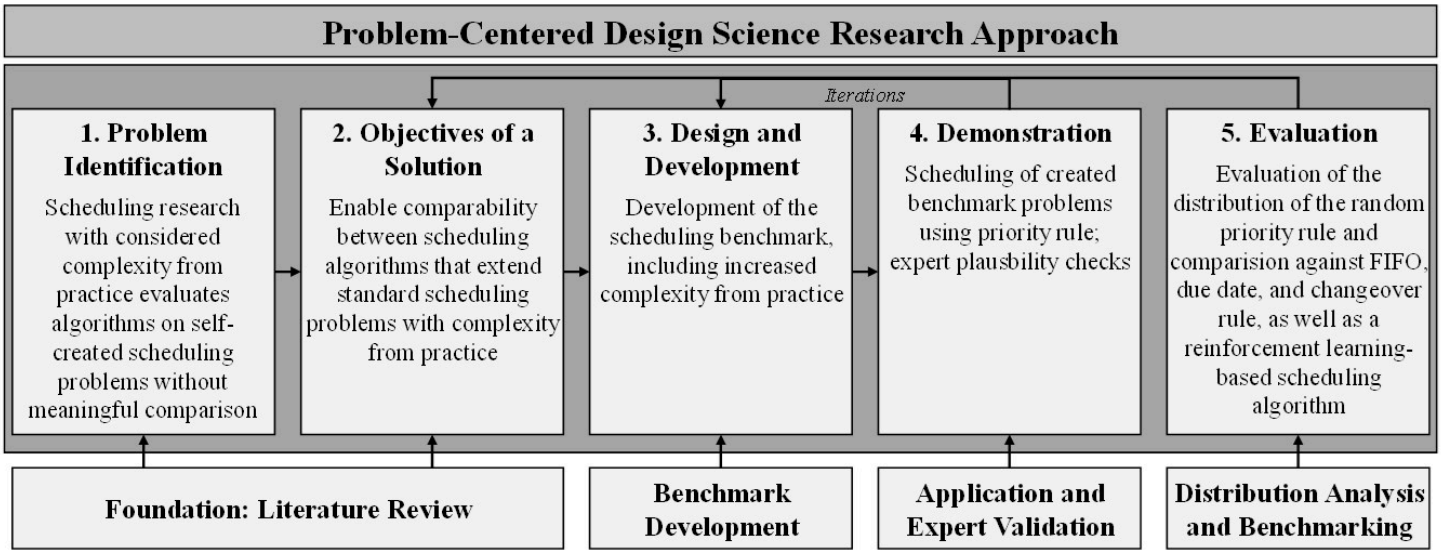


Figure 1

Problem-Centered Design Science Research Approach

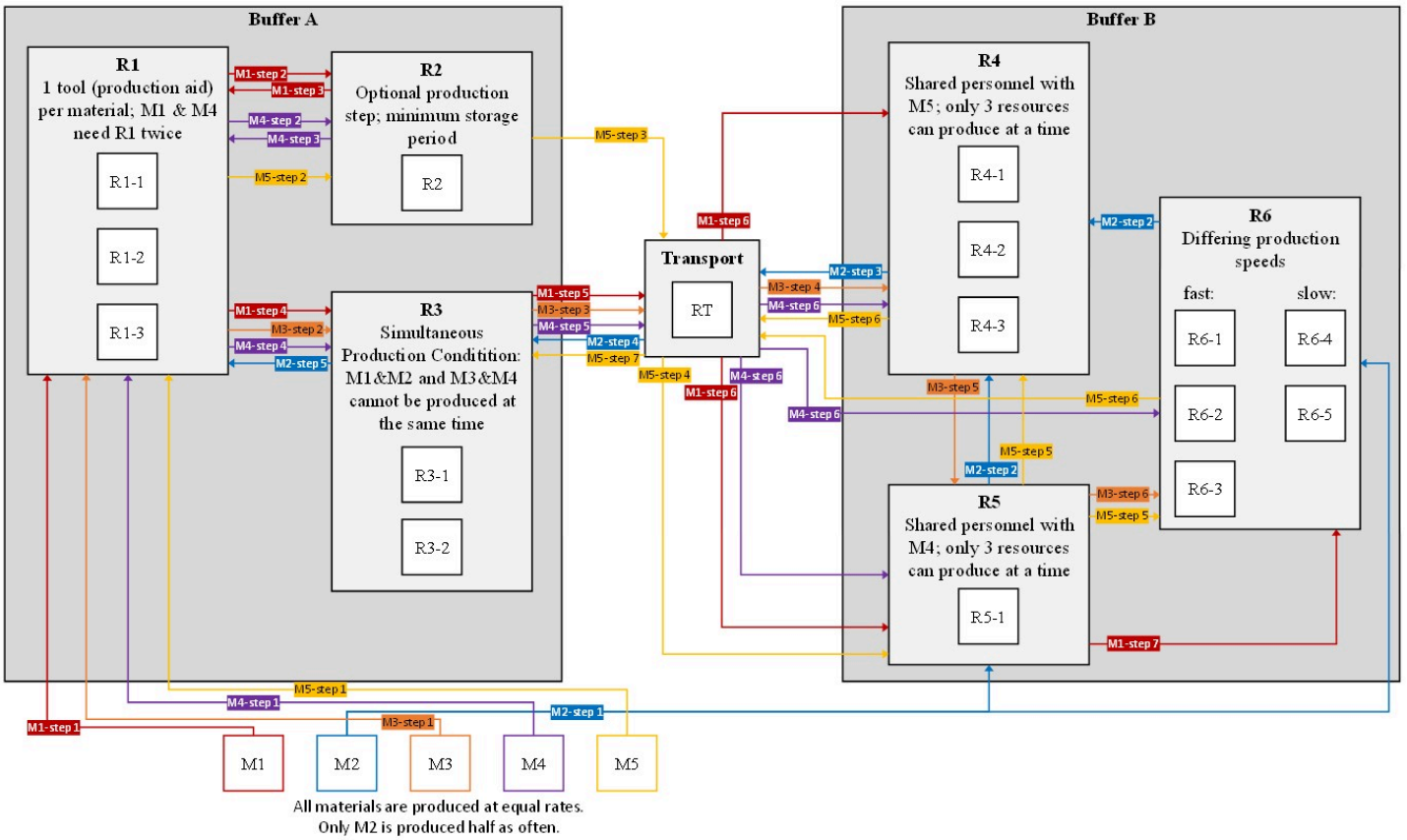


Figure 2

Production layout

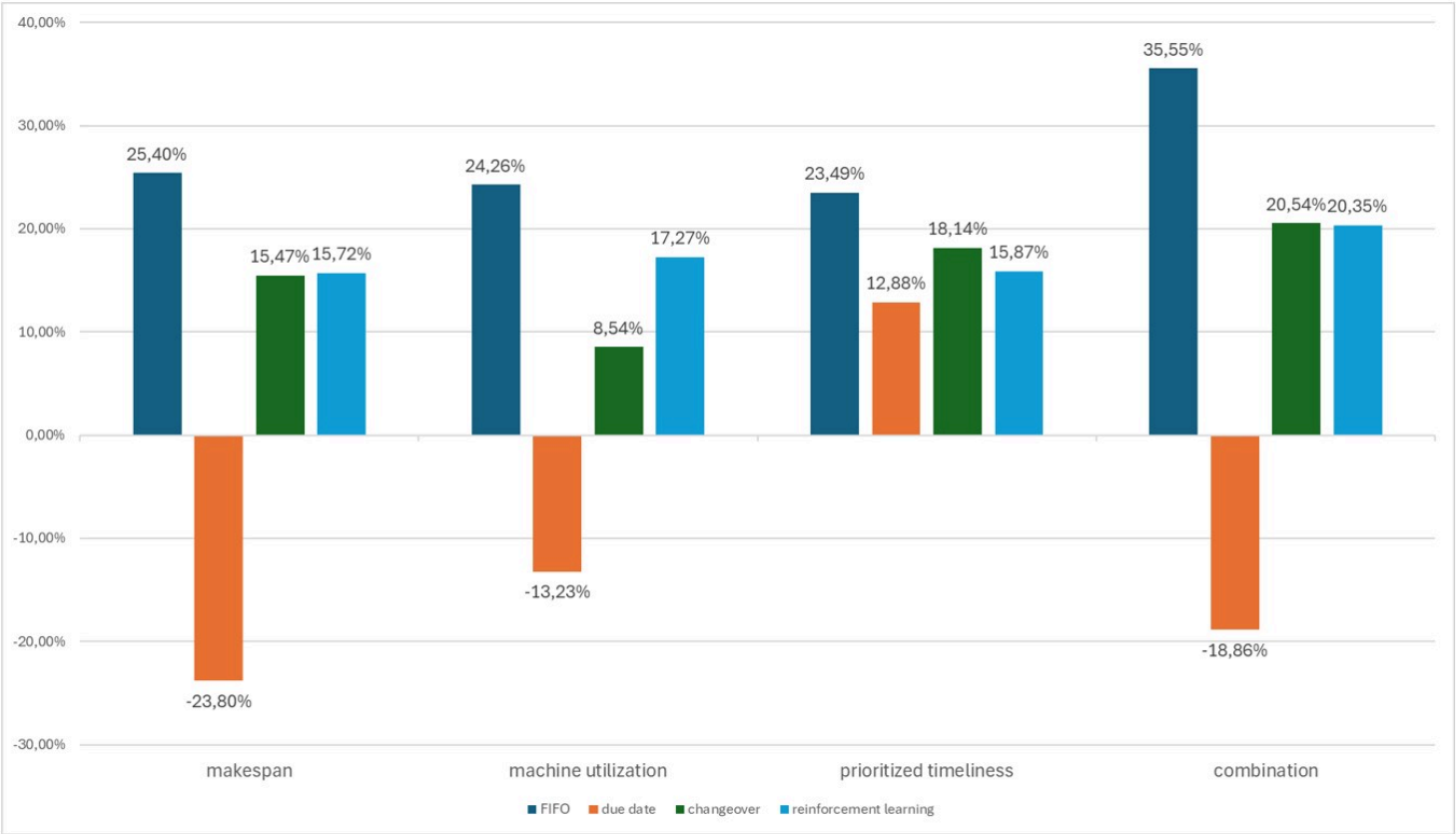


Figure 3

Results of benchmarking scheduling algorithms

Supplementary Files

This is a list of supplementary files associated with this preprint. Click to download.

- baselinecomparisonvalues.csv
- generator.zip
- generator.zip
- data.zip