12-2-2023

Design of a Reference Architecture for Production Scheduling Applications based on a Problem Representation including Practical Constraints

Michael Groth
University of Goettingen, Germany, michael.groth@uni-goettingen.de

Matthias Schumann
University of Goettingen, Germany, mschuma1@uni-goettingen.de

Design of a Reference Architecture for Production Scheduling Applications based on a Problem Representation including Practical Constraints

Full research paper

Michael Groth
Chair of Application Systems and E-Business
University of Goettingen
Goettingen, Germany
Email: michael.groth@uni-goettingen.de

Matthias Schumann
Chair of Application Systems and E-Business
University of Goettingen
Goettingen, Germany
Email: mschuma1@uni-goettingen.de

Abstract

Changing customer demands increase the complexity and importance of production scheduling, requiring better scheduling algorithms, e.g., machine learning algorithms. At the same time, current research often neglects practical constraints, e.g., changeovers or transportation. To address this issue, we derive a representation of the scheduling problem and develop a reference architecture for future scheduling applications to increase the impact of future research. To achieve this goal, we apply a design science research approach and, first, rigorously identify the problem and derive requirements for a scheduling application based on a structured literature review. Then, we develop the problem representation and reference architecture as design science artifacts. Finally, we demonstrate the artifacts in an application scenario and publish the resulting prototypical scheduling application, enabling machine learning-based scheduling algorithms, for usage in future development projects. Our results guide future research into including practical constraints and provide practitioners with a framework for developing scheduling applications.

Keywords Production Scheduling, Scheduling Application, Problem Representation, Reference Architecture, Machine Learning.

1 Introduction

Customers of manufacturing companies increasingly demand individualized products, summarized in the Mass Customization Theory (Da Silveira et al. 2001). This trend requires more changeovers in production, which increases the relevance of good production scheduling (Yang and Takakuwa 2017). In addition, dynamic influences on production (e.g., rush orders, machine breakdowns) render schedules invalid, requiring the ability to create new schedules quickly (Baykasoğlu and Ozsoydan 2018). Not reacting quickly to these dynamic effects can lead to decreased efficiency in production (Freier 2020). Scheduling algorithms can meet these challenges.

Currently, scheduling is primarily done manually by experts or with the help of heuristics, e.g., priority rules. The quality of manually created schedules depends on the scheduler's intuition and experience (Li and Olafsson 2005) and, thus, is not instant and error-prone (Liang et al. 2022). In turn, priority rules can create schedules in a sufficiently fast time but often lack quality, especially in complex scheduling problems (Kück et al. 2016). Alternatively, mathematical programming or evolutionary algorithms cannot cope with uncertainty and unexpected incidents, making them not applicable to modern, complex scheduling problems (Kuhnle et al. 2019).

The increase in complexity and insufficient current solutions require improved scheduling algorithms. Machine learning algorithms can learn from past situations and create schedules fast enough to react well to unforeseen situations (Waschneck et al. 2018). Such algorithms require accurate data representing the current production and, depending on the algorithm, training data (Shalev-Shwartz and Ben-David 2014). Cyber-physical systems, manufacturing equipment embedded with sensors and actors to realize the real-time availability of data on the equipment and communication between equipment, can provide the data required by these algorithms (Monostori 2014). The technological advances are summarized in the concept of “Industry 4.0” (Lasi et al. 2014) and “Advanced Manufacturing Systems” (Tao et al. 2017). Still, current research (e.g., Saqlain et al. 2022) on using machine learning for production scheduling primarily focuses on the standard Operations Research scheduling problem (Taillard 1993) that ignores the increase in complexity and additional constraints to consider (Baykasoğlu and Ozsoydan 2018; c.f., Section 4.1).

For future research to impact practice, research must consider the increased complexity. Therefore, as a basis for future research on scheduling algorithms, a representation of a scheduling problem, including the current complexity, is required. For practice to adopt newly developed scheduling algorithms based on the problem representation, a reference architecture is needed to include the newly developed algorithms in the existing production planning and control systems. Such a reference architecture would ease the adoption of new scheduling algorithms (e.g., machine learning algorithms) and allow researchers to evaluate newly developed algorithms quickly in comparison with existing ones. Following this aim, we address the following research questions in this paper:

RQ1: *How can a problem representation be designed, that includes previously unconsidered complexity from practice?*

RQ2: *How can an application for continuous production scheduling in practice be designed, that includes previously unconsidered complexity from practice and enables machine learning algorithms for scheduling?*

To answer these questions, we first cover the theoretical foundations and related research to introduce the topic and provide a common understanding (Section 2). Afterwards, we present our research design (Section 3), followed by the results of our research in Section 4. Namely, in this Section, we derive the requirements of a scheduling application, develop a representation of a practical scheduling problem, create a reference architecture for continuous production scheduling in practice, and demonstrate the developed artifacts in an application scenario. Finally, we conclude our research in Section 5.

2 Scheduling and Related Research

Production scheduling is part of production planning and control used in many manufacturing and service industries (Pinedo 2016). It is the time-related allocation of production tasks to production resources, e.g., machines, for released production jobs (Pinedo 2016). Furthermore, production scheduling is responsible for monitoring the plan's fulfillment and, in the case of deviations, taking countermeasures such as repairing or rescheduling (Sabuncuoglu and Goren 2009; Schneeweiß 1999). Research distinguishes between the following problem classes, describing the shop floor: single machine, parallel machine, flow shop, job shop, and open shop (Pinedo 2016).

The objectives vary in companies. Often, minimization of lead times, tardiness, or costs are used as objectives (Toader 2017). Because scheduling is an np-hard problem, an optimal schedule for realistic problem sizes cannot be found in polynomial time (Garey and Johnson 1979), requiring heuristic methods, e.g., machine learning algorithms.

Besides the standard scheduling problem (Taillard 1993), additional constraints exist in practice (Groth et al. 2024): Changeovers between operations on machines are commonly required (Vela et al. 2010). Buffers before and after production resources are often neglected (Kardos et al. 2021) but are essential to realistic schedules (Tamaki et al. 1999). Some productions prioritize jobs, thus, scheduling must also consider the prioritization of jobs (Nguyen et al. 2019). Dynamic influences (e.g., rush orders, machine breakdowns) may invalidate static schedules. Scheduling can consider potential dynamic influences to foster the creation of low-risk schedules (Baykasoğlu and Ozsoydan 2018). For some productions, the transportation on the shop floor (e.g., automated guided vehicles, forklifts) is closely tied to the manufacturing scheduling problem and must also be scheduled. In such cases, production scheduling should include the transportation problem (Li et al. 2022). Scheduling literature rarely includes these constraints (Groth et al. 2024).

For the standard scheduling problem, a formal problem representation already exists, containing information on the machine environment, job characteristics, and optimality criteria (Graham et al. 1979). Varela et al. (2005) extend on this and present how this formal problem representation can be used for an XML-based application programming interface. Similarly, Schmidt (1996) suggests an object-oriented data structure for the problem representation. For scheduling problems to be solved using Genetic Algorithms, Yingzi et al. (2009) name priority rule-based, random keys, and operation-based representation as possibilities. Finally, Maravelias (2012) discusses the problem representation of the chemical industry, drawing similarities to discrete manufacturing in the case of sequential processing. These representations assume fixed assignments of operations to production resources, not allowing alternative paths of jobs through production, and do not include the constraints mentioned above from practice. Additionally, ontologies exist for production scheduling. Smith and Becker (1997) and Rajpathak et al. (2001) provide general ontologies for scheduling. Both ontologies are general but lack concrete applicability in developing scheduling systems, thus not providing a standard data format. Sanko and Kotkas (2016) present an ontology that describes the result of scheduling usable in the execution of the scheduling plan, therefore making it not applicable to describing the scheduling problem. Consequently, to answer RQ1, we extend the problem representation of the scheduling problem to include constraints from practice and to make the problem representation directly usable in developing scheduling systems.

Reference architectures often have a different scope than just scheduling, considering upstream (e.g., a reference architecture for machine resource planning; Howard et al. 1999) or downstream processes (e.g., a reference architecture to transfer scheduling results to machines and enterprise resource planning software; Cândido et al. 2013). Framinan and Ruiz (2010) propose an architecture of a scheduling system after identifying requirements. This architecture pools multiple business tasks into the scheduling system (e.g., user interface, database), limiting its use in a service-oriented software landscape in a company. Vidoni and Vecchietti (2016) present a reference architecture for scheduling and other planning processes. It does not include specific requirements through the use of machine learning for scheduling algorithms. In contrast, Cunha et al. (2020) present an architecture for scheduling with deep reinforcement learning agents, following the requirements of a reinforcement learning architecture. Lastly, Shukla et al. (2018) propose a multi-agent-based architecture, working with different types of agents, including scheduling agents, that create schedules. These reference architectures are not generally applicable to work with scheduling algorithms based on machine learning. Thus, we develop, answering RQ2, a reference architecture for developing scheduling applications that include the constraints from practice and are suitable for machine learning algorithms for scheduling.

3 Research Design

To answer the research questions and, thus, develop a reference architecture including the problem representation that represents the previously unconsidered complexity, we use the problem-centered mix-method design science research approach adapted from Peffers et al. (2007). This research method guides the creation of artifacts, e.g., implementations or reference models. In our research, we followed the research progress depicted in Figure 1.

First, to ensure practical relevance and rigor theoretical grounding (Gregor and Hevner 2013), we conducted a structured literature review according to Webster and Watson (2002) and vom Brocke et

al. (2009). The structured literature review aimed to determine the state of the art of using machine learning for production scheduling and to identify constraints from practice. We used these results to derive requirements for the problem representation and reference architecture. Due to the aim of finding relevant concepts and a large amount of available research on the production scheduling problem, the literature scope is selective.

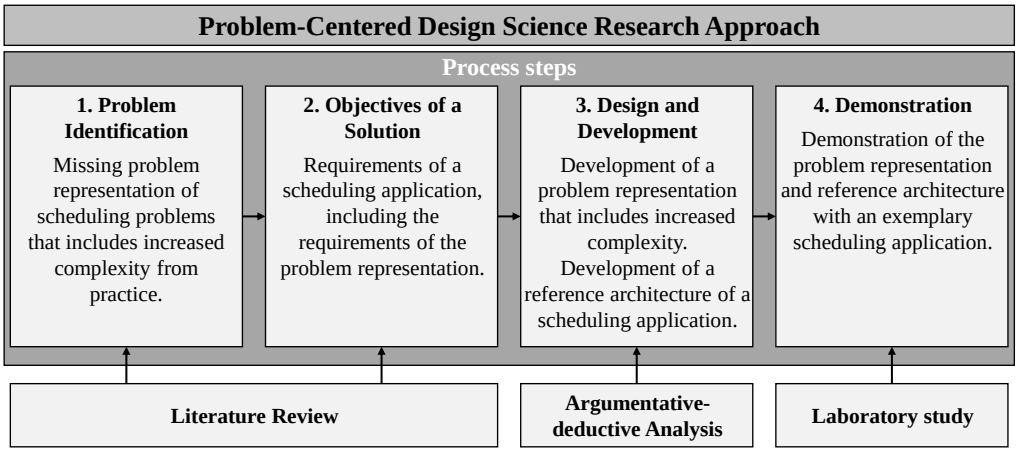

Figure 1. Research design adapted from Peffers et al. (2007)

We searched scientific and practice-oriented databases (IEEE Xplore, SpringerLink, ScienceDirect, AISeL, ACM, EbscoHost, Emerald, WISO) not limited to highly ranked journals to include practical case studies. To extend the existing problem representations and reference architectures for the applicability of machine learning-based scheduling algorithms, we used the search term *"machine learning" AND "production scheduling"*. We find papers relevant for our analysis if the research uses machine learning to solve the production scheduling problem, or discusses requirements, the problem representation, or the architecture of a scheduling application. We excluded literature that is not specifically about production scheduling but other application areas. We found 1721 papers and excluded 1561 papers due to missing relevance or as duplicates after screening the title and abstract. After a full-text analysis of the remaining 160 papers and a forward and backward search, 117 relevant publications remained. We present the details of the methodology and results of the structured literature review in Groth et al. (2024).

In the first step, we analyze the literature review results to identify constraints to production scheduling that are required in practice. In the second step, we build on this to define the objective of a solution and derive requirements that the artifacts must fulfill.

In the third step, we derived the artifacts based on the previous results and set objectives. First, we developed the problem representation, which implements the requirements identified in step 2. The problem representation served as a theoretical foundation for the subsequent development of the reference architecture. In step 4, we demonstrated the artifacts in a laboratory study. Therefore, we implemented the reference architecture in a priority rule scheduling application prototype and published the resulting scheduling application prototype to enable the use in other development projects to maximize the impact of this research project and to ease the inclusion of the identified constraints in future research.

4 Design of a Scheduling Application Framework Including Practical Constraints

In the following, we present the results of the structured literature review and design the artifacts. First, we identify the problem (Section 4.1). Afterwards, we use our findings to derive the objectives as requirements for a scheduling application (Section 4.2). Based on the previous results, in Section 4.3, we design the problem representation and the reference architecture of a scheduling application in Section 4.4. Finally, in Section 4.5, we demonstrate the resulting artifacts in an exemplary application scenario, thus, completing our research process.

4.1 Problem Identification

For manufacturing companies, it is essential to quickly generate good production schedules to achieve high efficiency and meet production deadlines driven by changes in customer demand described in Section 1. Nevertheless, we find, as the result of our literature review, that constraints from practice (c.f., Section 2) are often neglected in research. Including those constraints in research is required to be of relevance for producing companies that handles with these constraints.

As constraints, extending the literature scheduling problem (Taillard 1993), we found changeovers, buffers at production resources, prioritization of jobs, dynamics of the production and inclusion of the transportation scheduling problem to be of relevance but often missing in research. Out of these constraints, changeovers are most considered in in the literature that applied machine learning for scheduling. Still, only 18 % of the literature included changeovers in their algorithm (Vela et al. 2010). Buffers at production resources were only considered in 4 % (Tamaki et al. 1999), while prioritization of jobs was included in 10 % of publications (Nguyen et al. 2019). 6 % of the literature considered dynamics (Wang et al. 2013), and only 3 % included the transportation problem (Li et al. 2022). For research to be useful for practice, it must consider the above constraints.

Research is often conducted without considering the constraints because the standard literature problem is well-defined, and the problem representation is provided in the form of benchmark problems (Taillard 1993). Therefore, as a step to guide future research to include the practical constraints, a problem representation is required that includes the practical constraints. Additionally, to foster the adoption of the representation by researchers and practitioners, a reference architecture could guide the development of a scheduling application that uses the problem representation.

4.2 Objectives of a Solution

To solve the identified problem, we first develop a problem representation of the scheduling problem, including the practical constraints discussed above. This problem representation structures the scheduling problem and should be used to describe a scheduling problem in scheduling applications. Then, using the problem representation, we develop a reference architecture of a scheduling application to use the problem representation in scheduling problems.

#	Requirement Description	Examplary Reference
R1	Support all problem classes	Pinedo 2016
R2	Support sequence-dependent changeovers	Rolf et al. 2020
R3	Support buffers of flexible size at production resources	Qu et al. 2016
R4	Support a no-wait condition of operations	Tavakkoli-Moghaddam et al. 2008
R5	Support prioritization as goal weights, priority classes, and continuous priority values	Niemeyer and Shiroma 1996; Riley et al. 2016; Zhou et al. 2019
R6	Support planned buffers in operations to reduce risk through dynamic production speed	Zhang et al. 2020
R7	Support dynamics by integrating uncertainties	Zheng et al. 2010
R8	Support dynamics by simulation uncertainties	Aytug et al. 1998
R9	Support scheduling of transportation resources	Li et al. 2022
R10	Support simultaneous production on a production resource with exclusion conditions	Xue et al. 2004
R11	Support minimum storage periods of intermediary products	Jian et al. 2004
R12	Support limited amount of production aids	Graham et al. 1979
R13	Support jobs with release dates	Nie et al. 2010

Table 1. Requirements of a general production scheduling application

To achieve these objectives, we use the results of the structured literature review to identify requirements that the problem representation and reference architecture must fulfill and extend the standard scheduling problem. Therefore, the neglected constraints from practice served as a starting point and are extended with additional requirements found in the literature specific to extending the standard scheduling problem (Taillard 1993). Table 1 summarizes the resulting 13 requirements.

First, to be generally applicable, the scheduling application must support all shop floor classes [R1], summarized in the problem classes (Pinedo 2016). Additionally, it must support sequence-dependent changeovers (Rolf et al. 2020) [R2], thus including other, more straightforward changeover types like product-dependent changeovers (e.g., Qu et al. 2016). To ensure practical resulting schedules, the algorithm must support buffers of flexible sizes at production resources (Qu et al. 2016) [R3]. If materials may not be stored between operations, but instead, the subsequent operation must start immediately after the competition of the previous operation, a no-wait condition is required (Tavakkoli-Moghaddam et al. 2008) [R4]. In the literature, we found that prioritization of jobs is handled in two ways: First, prioritization can be considered in the goal function as weights, usually when minimizing tardiness (Riley et al. 2016). Second, the algorithm can integrate priorities as priority classes (Niemeyer and Shiroma 1996) or as continuous priority values (Zhou et al. 2019). A general scheduling application must include both cases [R5].

R6-8 handle the topic of dynamics of the production. To reduce risk through dynamic influences, we found literature that plans buffers into the production time of operations (Zhang et al. 2020) [R6]. As with the prioritization of jobs, dynamics can be used in the algorithm (Zheng et al. 2010) [R7], or it can be evaluated, in this case, by simulation of the uncertain influences (Aytug et al. 1998) [R8]. Furthermore, if transportation on the shop floor is a success factor, the scheduling application must support the simultaneous scheduling of transportation resources (Li et al. 2022) [R9]. Some production resources (e.g., ovens) can be used simultaneously by multiple operations with exclusion conditions (e.g., if operations require different temperatures) (Xue et al. 2004). A scheduling application must also support this [R10]. Intermediary products may require a minimum storage period (e.g., for cooling), which must also be supported (Jian et al. 2004) [R11]. Furthermore, certain operations may need production aids only available in limited quantities (Graham et al. 1979) [R12]. Finally, jobs can have release dates [R13]. It is useful to consider jobs with release dates in scheduling as it allows for better schedules because more information about the future is available (Nie et al. 2010).

4.3 Design of a Problem Representation Including Practical Constraints

Using the problem representation of Graham et al. (1979) and the derived requirements as a basis, we developed a representation of the scheduling problem, including the identified practical constraints. We employ an Entity Relationship Model according to Chen (1976) because the relevant entities, their attributes, and the relations between them describe the scheduling problem. Figure 2 presents the resulting Entity Relationship Model.

A **scheduling problem** has a goal in the form of a function and consists of multiple **production jobs**. Each production job has a priority [R5], a due date to determine tardiness, and a release date representing the earliest start of production [R13]. It embodies multiple **operations**, each representing one production step. If all operations of a production job are completed, the job is fulfilled. Each operation can have multiple predecessors and define which operations cannot be produced simultaneously. A minimum storage time can be used to define a timespan before the next operation may be started (e.g., for cooling) [R11].

Materials can be any initial, intermediary, or final products. An operation produces one material in a specific amount; if multiple resulting materials are provided, they are alternative results. Combined with the exclusion of simultaneous production of operations, the sequence can be flexible, thus supporting open shop productions. The predecessor definitions support the remaining problem classes [R1]. The flexible definition of operations, their dependency, and resulting materials also allow transportation to be represented [R9]. Therefore, the transportation resources must be modeled as production resources. Suppose the transportation times used for calculation vary between different production resources. In this case, one physical material may be modeled as multiple materials in the scheduling representation, thus allowing different production times defined in the relationship between material and production resource. For example, the *physical material A* may be modeled as *material A at machine 1* and *material A at machine 2*. Additionally, a no-wait condition can be defined for each predecessor definition of an operation [R4]. Furthermore, operations may need **production aids** (e.g., special tools) only available in limited quantities [R12].

Production resources, e.g., machines, can produce materials, where times and costs are defined for each production resource and material, potentially as a function instead of static values representing complex dependencies (e.g., *time = 10 + 3*produced_amount)*. Planned buffers in production speed and costs can be considered in these values [R6]. In addition, each machine defines sequence-dependent **changeovers**, i.e., from and to a material, with times and costs as functions [R2]. Each material has a buffer usage, and each production resource can have a **buffer** with a given size before and after it [R3]. Furthermore, each production resource may have failure and time/cost deviation probabilities to allow the algorithm to integrate uncertainties [R7].

Finally, if a physical production resource can produce multiple operations simultaneously, it must be modeled as two production resources. **Simultaneous production conditions** can be defined for those production resources, consisting of the type of the condition (e.g., material exclusions) and the associated values [R10].

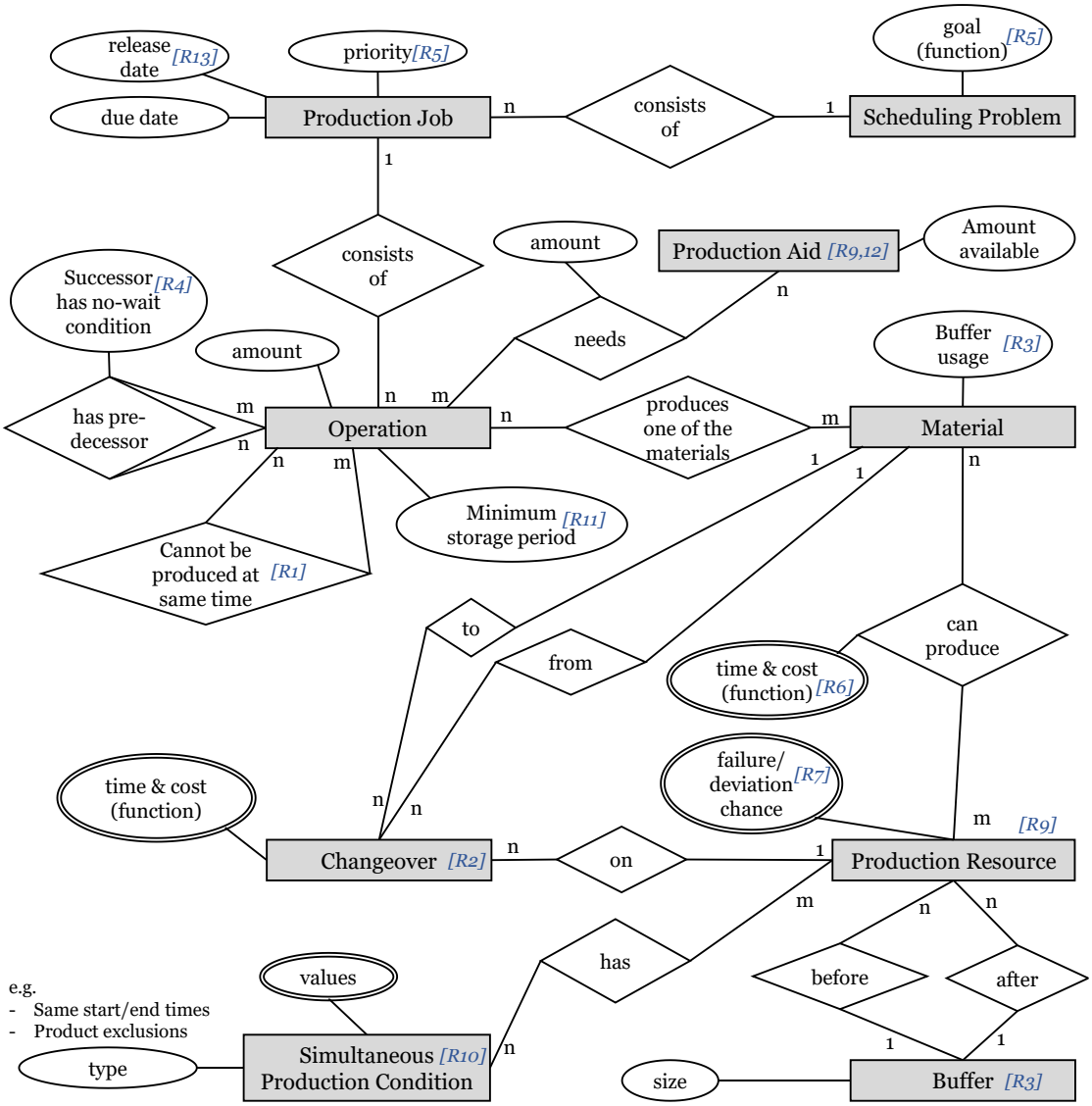

Figure 2. Entity Relationship Model of the problem representation

This problem representation meets the requirements set in Section 4.2 and, thus, answers RQ1. R8 (Support dynamics by simulation of uncertainties) is not mentioned in this Section as it is only a requirement regarding functionality and not relevant to the data structure. Thus, it is just included in the reference architecture of a scheduling application.

4.4 Design of a Reference Architecture for a Scheduling Application

As a second artifact, using the problem representation, we developed a reference architecture for a scheduling application that includes the practical constraints, thus answering RQ2. Figure 3 presents the resulting reference architecture, where each box depicts a service.

In developing the reference architecture, we followed the service-oriented architecture (OASIS 2006) to ensure high flexibility, allowing a seamless exchange of the scheduling algorithm. Additionally, the architecture aims to support machine learning scheduling algorithms and, thus, supports the training of algorithms (Shalev-Shwartz and Ben-David 2014). The derived problem representation serves as a data structure used between the services.

The **API layer** handles requests for a new schedule, requiring the complete problem representation discussed in Section 4.3 as input (1). When the API layer receives the generated schedule for the given problem, it returns it to the requester. Additionally, the API layer offers an input to receive training data (2). Algorithms that can train themselves to adapt to changing situations require training data representing the current production situation. If and how training data is handled depends on the scheduling algorithm.

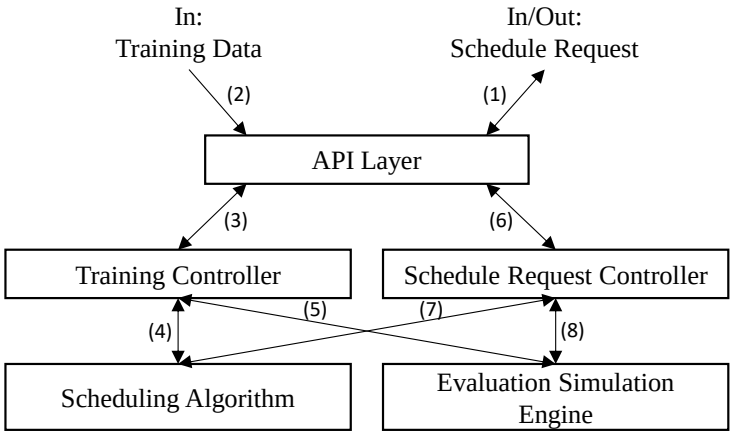

Figure 3. Reference architecture of a scheduling application

The API layer passes training data, as a scheduling problem, to the underlying **training controller** (3). The training controller uses this training data for continuous training of the scheduling algorithm (4). This continuous training allows the algorithm to adapt to the current situation (Shalev-Shwartz and Ben-David 2014). The training controller alters the scheduling problem to train the algorithm with similar problems. Depending on the type of algorithm used (reinforcement learning, supervised learning), the training data received can be annotated if required. Additionally, the training controller is responsible for evaluating generated training results, and it controls the adaptions of the algorithm, e.g., in the form of rewards with reinforcement learning (Groth et al. 2021) or weight recalculations with Neural Networks (Zhao et al. 2005). The training controller uses the evaluation simulation engine to evaluate the generated schedules (5).

Regarding schedule requests, the API layer provides the received problem to the **schedule request controller** (6). The schedule request controller is responsible for creating schedules, evaluating the generated schedule, and returning the result to the API layer. Therefore, it passes the scheduling problem to the scheduling algorithm (7) and sends the resulting schedule to the evaluation simulation engine (8).

The **scheduling algorithm** performs the schedule generation. It either implements the algorithm directly or serves as a gateway to an independently deployed scheduling algorithm.

The **evaluation simulation engine** is responsible for evaluating schedules. For this, the training controller and schedule request controller pass generated schedules to the evaluation simulation engine. A simulation of the schedule is required if uncertainties (e.g., machine failures) should be simulated [R8]. For this purpose, a discrete event simulation could be used (Jeffrey and Seaton 1995), allowing the incorporation of random events into the evaluation (Banks 1996). In the case of an external simulation engine (e.g., Plant Simulation 2023), the evaluation simulation engine can act as a gateway and send the problem to the external simulation engine, potentially adjusting the data structure if required.

The presented reference architecture is practical if used in the context of a service-oriented architecture that consists of other applications. Enterprise Resource Planning software and cyber-physical systems provide the initially required data. Only the data relevant to scheduling is needed. Therefore, an intermediary system, e.g., a Production Planning and Control software, could aggregate the data and create the scheduling problem before sending it to the scheduling application (Kistner and Steven 2001). Such a system could also send regular updates on the current production situation as training data to the scheduling application. The resulting schedules can either be automatically used in production or passed to a Decision Support System (Freier and Schumann 2020) in which a person in charge can modify the schedule or decide on alternative schedules. The first method allows for quick reactions to unforeseen situations, thus reducing the costs of slow reactions (Freier 2020). The latter method is helpful if the generated schedules are not the optimal solution and an experienced person is available who can decide on the best schedule for the production.

4.5 Demonstration

To revise the requirements and developed artifacts, we demonstrate the developed artifacts in an application scenario according to the research design presented in Section 3. Therefore, we conducted a laboratory study in which we implemented the reference architecture and the problem representation in a fictional scheduling problem that includes all constraints discussed above and used the priority rule first-in-first-out (FIFO) as a scheduling algorithm (Pinedo 2016). We decided to use FIFO as it is easy to comprehend and widely used (Schuh 2007).

For the demonstration, we used TypeScript, which extends JavaScript with types (TypeScript 2023). We decided on TypeScript because the types guide and ease the development, and we have previous experience with it. Furthermore, it is a well-established and popular programming language (Stack Overflow 2022). Finally, the ecosystem around TypeScript allows us to publish the developed artifact as a module that other researchers or practitioners can import into their development projects (npm 2023).

The prototypical scheduling application implements the reference architecture depicted in Figure 3. A REST API represents the API layer (Masse 2011), providing the POST methods *schedule* and *training*. Each method accepts the scheduling problem described in Figure 2 as a JSON object following an interface definition. Jobs, materials, simultaneous production conditions, production resources, production aids, and buffers are provided as attributes of the scheduling problem, operations are attributes of a job, and changeovers are attributes of a production resource. Function (i.e., goal, time, and cost function) are modeled as objects representing arithmetic operations, constants, and variables, enabling an efficient calculation of the resulting values at runtime. The other attributes of the entities are implemented as defined in Figure 2. To enable subsequent use of the schedules, we extended the above definition by identifiers for the entities. Furthermore, to reduce the number of definitions of changeovers and production times and costs required, we added material groups to the definition of materials, changeovers, and production times and costs.

We developed the remaining parts of the architecture as TypeScript services, implementing the necessary functionalities. The scheduling algorithm implements the priority rule first-in-first-out for demonstration purposes (Pinedo 2016). The resulting application is available for use in other applications in the following git project: https://gitlab.com/michael-groth/acis-2023-design-of-a-reference-architecture-for-production-scheduling-applications-based-on-a-problem-representation-including-practical-constraints.

We tested the scheduling application with a fictional production scenario of a bike manufacturer. Figure 4 depicts the chosen production scenario. The scenario included parallel operations with a common predecessor and an optional production step not required by all jobs. The assembly step required material-dependent tools only available in limited quantities. Furthermore, the packaging step takes place at one packaging machine that can package two products simultaneously, thus modeled as two production resources. To test the simultaneous production exclusion conditions, we assumed that it could not package some products simultaneously. Additionally, after lacquering, the products must not be processed further for a given timeframe. The production scenario has two buffer zones limiting the amount of work in progress. For jobs, we also defined jobs with later release dates.

The demonstration scenario purposely did not include the requirement of uncertainties, thus enabling us to verify the proposed concept in application areas that only have a subset of the above-defined requirements. In the development, we left out all parts only relevant for uncertainties. Still, the problem representation and reference architecture were applicable.

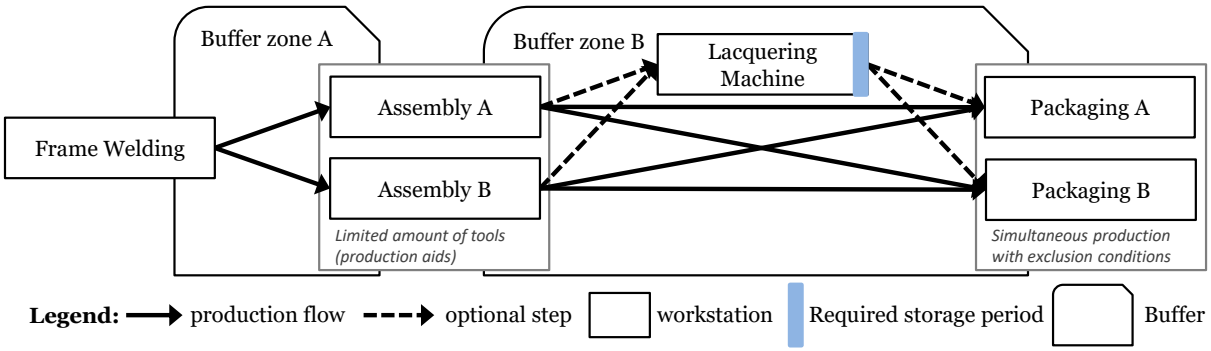

Figure 4 Production scenario used in the demonstration

As a result of the demonstration, we found that the presented problem representation and reference architecture helped us develop the prototypical scheduling application by providing the structure of the scheduling problem and defining the responsibilities of the architectural layers. With this, we enable researchers and practitioners to use the problem representation and reference architecture implemented in the published prototypical scheduling application as a starting point for developing scheduling applications focused on meaningful scheduling algorithms applicable in practice. Additionally, the reference architecture provides a framework for the development of scheduling applications and allows a seamless exchange of scheduling algorithms, thus easing the evaluation of scheduling algorithms.

5 Discussion and Conclusion

The present work aimed to increase the practical relevance of future production scheduling research by including practical constraints in future research. Therefore, we derived a representation of the scheduling problem (RQ1) and developed a reference architecture for future scheduling applications (RQ2). To answer the research questions, we used a problem-oriented design science research approach adopted from Peffers et al. (2007). Following this methodology, we used a structured literature review to identify 13 requirements specific to extending the literature scheduling problem that the problem representation and reference architecture must fulfill. Afterwards, we answered RQ1, developing a representation of the extended scheduling problem. We presented the representation as an Entity Relationship Model. After this, we answered RQ2 and designed a reference architecture of a scheduling application using the problem representation. It follows a service-oriented design providing flexibility (OASIS 2006). To ensure the requirements were met with the two developed artifacts, we demonstrated them in a laboratory study using a fictional problem scenario. Employing this scenario, we implemented the problem representation and reference architecture in a prototype and published the resulting prototypical implementation as a git project for subsequent use by researchers and practitioners. We found both artifacts helpful in the laboratory study because they provided a framework for the development and ensured a consistent data structure.

The proposed architecture would fit into a manufacturing application landscape because of the service-oriented design. Primarily, a manufacturing execution system (MES) would be responsible for initiating the creation of a schedule whenever rescheduling is required (e.g., delays in production, the arrival of new jobs) and transferring the resulting schedule to production. The MES would collect the required information from connected systems (e.g., enterprise resource planning, edge devices for monitoring machines). Alternatively, this could be done by a middleware layer between the MES and the scheduling application. Due to the service-oriented design, the responsibilities are separated, and the proposed architecture would fit into different manufacturing application landscapes.

However, we acknowledge that our research has some limitations. First, the literature review might not have found all relevant practical constraints and, therefore, requirements, even though we followed a structured approach and tried to minimize subjective influences. Second, we only conducted a demonstration of the artifacts. An evaluation using a machine learning algorithm that requires training and a real scheduling problem is missing. Such an evaluation could add insight from a scheduling application development point of view. Additionally, an evaluation with experts from the industry to validate the results is missing. Consequently, we plan an evaluation of the developed artifacts in future studies.

Nevertheless, the results can help guide future scheduling research to include practical constraints and help practitioners with more meaningful future research, closing the gap between research and practice. Therefore, the artifact of the problem representation serves as a starting point for scheduling applications that include practical constraints. The published prototypical scheduling application, including interfaces of the problem representation, increases the speed of scheduling application development and allows developers to focus on the development of the algorithm itself. Furthermore, the results serve as a starting point for standardizing the scheduling problem to make scheduling algorithms developed in research applicable in most practice scenarios, as following the architecture and problem representation makes scheduling algorithms exchangeable. The artifact of the reference architecture holds as a framework for developing scheduling applications for practice and research, focused, given the service-oriented architecture, on a seamless exchange of scheduling algorithms and guides standardization, mainly if the problem representation is used within the application. The main contributions of this work are the aid in developing scheduling systems through standardization and generally applicable scheduling architecture, with its focus on enabling machine learning-based scheduling systems. For future research, the proposed architecture could also be generalized to support scheduling algorithms other than machine learning algorithms. Overall, from a methodical point of view, the artifacts serve as a level-1 design science contribution to research (Gregor and Hevner 2013) and present the impact of information systems research on operations research.

6 References

Aytug, H., Bhattacharyya, S., and Koehler, G. J. 1998. "Genetic learning through simulation: An investigation in shop floor scheduling," *Annals of Operations Research* (78), pp. 1-29 (doi: 10.1023/A:1018989730961).

Banks, J. 1996. "Output Analysis Capabilities of Simulation Software," *SIMULATION* (66:1), pp. 23-30 (doi: 10.1177/003754979606600103).

Baykasoğlu, A., and Ozsoydan, F. B. 2018. "Dynamic scheduling of parallel heat treatment furnaces: A case study at a manufacturing system," *Journal of Manufacturing Systems* (46), pp. 152-162 (doi: 10.1016/j.jmsy.2017.12.005).

Cândido, G., Di Orio, G., and Barata, J. 2013. "Self-Learning Production Systems: Adapter Reference Architecture," in *Advances in Sustainable and Competitive Manufacturing Systems*, A. Azevedo (ed.), Heidelberg: Springer International Publishing, pp. 681-693 (doi: 10.1007/978-3-319-00557-7_56).

Chen, P. P.-S. 1976. "The entity-relationship model—toward a unified view of data," *ACM Transactions on Database Systems* (1:1), pp. 9-36 (doi: 10.1145/320434.320440).

Cunha, B., Madureira, A. M., Fonseca, B., and Coelho, D. 2020. "Deep Reinforcement Learning as a Job Shop Scheduling Solver: A Literature Review," in *Hybrid Intelligent Systems*, A. M. Madureira, A. Abraham, N. Gandhi and M. L. Varela (eds.), Springer International Publishing, pp. 350-359 (doi: 10.1007/978-3-030-14347-3_34).

Da Silveira, G., Borenstein, D., and Fogliatto, F. S. 2001. "Mass customization: Literature review and research directions," *International Journal of Production Economics* (72:1), pp. 1-13 (doi: 10.1016/S0925-5273(00)00079-7).

Framinan, J. M., and Ruiz, R. 2010. "Architecture of manufacturing scheduling systems: Literature review and an integrated proposal," *European Journal of Operational Research* (205:2), pp. 237-246 (doi: 10.1016/j.ejor.2009.09.026).

Freier, P. 2020. *Empirische Erkenntnisse und Gestaltungsansätze für Entscheidungsunterstützungssysteme in der Ablaufplanung im Kontext von Cyber-Physischen Systemen*, Göttingen: Cuvillier Verlag.

Freier, P., and Schumann, M. 2020. "Design and Implementation of a Decision Support System for Production Scheduling in the Context of Cyber-Physical Systems," in *WI2020 Zentrale Tracks*, N. Gronau, M. Heine, K. Poustcchi and H. Krasnova (eds.), GITO Verlag, pp. 757-773 (doi: 10.30844/wi_2020_g5-freier).

Garey, M. R., and Johnson, D. S. 1979. *Computers and intractability*, freeman San Francisco.

Graham, R. L., Lawler, E. L., Lenstra, J. K., and Kan, A. 1979. "Optimization and Approximation in Deterministic Sequencing and Scheduling: a Survey," in *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, Elsevier, pp. 287-326 (doi: 10.1016/S0167-5060(08)70356-X).

Gregor, S., and Hevner, A. R. 2013. "Positioning and Presenting Design Science Research for Maximum Impact," *MIS Quarterly* (37:2), pp. 337-355.

Groth, M., Freier, P., and Schumann, M. 2021. "Using Self-Play within Deep Q Learning to improve real-time Production Scheduling," in *AMCIS 2021 Proceedings*, pp. 1-10.

Groth, M., Schumann, M., and 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 *Flexible Automation and Intelligent Manufacturing: Establishing Bridges for More Sustainable Manufacturing Systems*, F. J. G. Silva, L. P. Ferreira, J. C. Sá, M. T. Pereira and C. M. A. Pinto (eds.), Cham: Springer Nature Switzerland, pp. 119-127 (doi: 10.1007/978-3-031-38165-2_15).

Howard, A., Kochhar, A., and Dilworth, J. 1999. "Application of a generic manufacturing planning and control system reference architecture to different manufacturing environments," *Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture* (213:4), pp. 381-396 (doi: 10.1243/0954405991516859).

Jeffrey, P., and Seaton, R. 1995. "The Use of Operational Research Tools: A Survey of Operational Research Practitioners in the UK," *Journal of the Operational Research Society* (46:7), pp. 797-808 (doi: 10.1057/jors.1995.113).

Jian, W., Xue, Y., and 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),* The Hague, Netherlands. 10-13 Oct. 2004, IEEE, pp. 4284-4289 (doi: 10.1109/ICSMC.2004.1401204).

Kardos, C., Laflamme, C., Gallina, V., and Sihn, W. 2021. "Dynamic scheduling in a job-shop production system with reinforcement learning," *Procedia CIRP* (97), pp. 104-109 (doi: 10.1016/j.procir.2020.05.210).

Kistner, K.-P., and Steven, M. 2001. *Produktionsplanung*, Heidelberg: Physica-Verlag HD.

Kück, M., Ehm, J., Hildebrandt, T., Freitag, M., and Frazzon, E. M. 2016. "Potential of data-driven simulation-based optimization for adaptive scheduling and control of dynamic manufacturing systems," in *Winter Simulation Conference*, T. M. K. Roeder, P. I. Frazier, R. Szechtman, E. Zhou, T. Huschka and S. E. Chick (eds.), Washington, DC, USA, IEEE, pp. 2820-2831 (doi: 10.1109/WSC.2016.7822318).

Kuhnle, A., Röhrig, N., and Lanza, G. 2019. "Autonomous order dispatching in the semiconductor industry using reinforcement learning," *Procedia CIRP* (79), pp. 391-396 (doi: 10.1016/j.procir.2019.02.101).

Lasi, H., Fettke, P., Kemper, H.-G., Feld, T., and Hoffmann, M. 2014. "Industry 4.0," *WIRTSCHAFTSINFORMATIK* (56:4), pp. 239-242 (doi: 10.1007/s11576-014-0424-4).

Li, W., Han, D., Gao, L., Li, X., and Li, Y. 2022. "Integrated Production and Transportation Scheduling Method in Hybrid Flow Shop," *Chinese Journal of Mechanical Engineering* (35:1) (doi: 10.1186/s10033-022-00683-7).

Li, X., and Olafsson, S. 2005. "Discovering Dispatching Rules Using Data Mining," *Journal of Scheduling* (8:6), pp. 515-527 (doi: 10.1007/s10951-005-4781-0).

Liang, Y., Sun, Z., Song, T., Chou, Q., Fan, W., Fan, J., Rui, Y., Zhou, Q., Bai, J., Yang, C., and Bai, P. 2022. "Lenovo Schedules Laptop Manufacturing Using Deep Reinforcement Learning," *INFORMS Journal on Applied Analytics* (52:1), pp. 56-68 (doi: 10.1287/inte.2021.1109).

Maravelias, C. T. 2012. "General framework and modeling approach classification for chemical production scheduling," *AIChE Journal* (58:6), pp. 1812-1828 (doi: 10.1002/aic.13801).

Masse, M. 2011. *REST API Design Rulebook*, O'Reilly Media, Inc.

Monostori, L. 2014. "Cyber-physical Production Systems: Roots, Expectations and R&D Challenges," *Procedia CIRP* (17), pp. 9-13 (doi: 10.1016/j.procir.2014.03.115).

Nguyen, S., Mei, Y., Xue, B., and Zhang, M. 2019. "A Hybrid Genetic Programming Algorithm for Automated Design of Dispatching Rules," *Evolutionary computation* (27:3), pp. 467-496 (doi: 10.1162/evco_a_00230).

Nie, L., Shao, X., Gao, L., and Li, W. 2010. "Evolving scheduling rules with gene expression programming for dynamic single-machine scheduling problems," *The International Journal of Advanced Manufacturing Technology* (50:5-8), pp. 729-747 (doi: 10.1007/s00170-010-2518-5).

Niemeyer, G., and Shiroma, P. 1996. "Production scheduling with genetic algorithms and simulation," in *Parallel Problem Solving from Nature — PPSN IV*, G. Goos, J. Hartmanis, J. van Leeuwen, H.-M. Voigt, W. Ebeling, I. Rechenberg and H.-P. Schwefel (eds.), Springer, pp. 930-939 (doi: 10.1007/3-540-61723-X_1056).

npm. 2023. "npm," available at https://www.npmjs.com/, accessed on Mar 6 2023.

OASIS. 2006. "Reference Model for Service Oriented Architecture," available at https://www.oasis-open.org/committees/download.php/16587/wd-soa-rm-cd1ED.pdf.

Peffers, K., Tuunanen, T., Rothenberger, M. A., and Chatterjee, S. 2007. "A Design Science Research Methodology for Information Systems Research," *Journal of Management Information Systems* (24:3), pp. 45-77 (doi: 10.2753/MIS0742-1222240302).

Pinedo, M. L. 2016. *Scheduling*, Cham: Springer International Publishing.

Plant Simulation. 2023. "Plant Simulation," available at https://plant-simulation.de/, accessed on Jan 6 2023.

Qu, S., Wang, J., and Shivani, G. 2016. "Learning adaptive dispatching rules for a manufacturing process system by using reinforcement learning approach," in *2016 IEEE 21st International Conference on Emerging Technologies and Factory Automation (ETFA),* Berlin, Germany, IEEE, pp. 1-8 (doi: 10.1109/ETFA.2016.7733712).

Rajpathak, D., Motta, E., and Roy, R. 2001. "A generic task ontology for scheduling applications,"

Riley, M., Mei, Y., and Zhang, M. 2016. "Improving job shop dispatching rules via terminal weighting and adaptive mutation in genetic programming," in *2016 IEEE Congress on Evolutionary Computation (CEC),* Vancouver, BC, Canada. 24.07.2016 - 29.07.2016, IEEE, pp. 3362-3369 (doi: 10.1109/CEC.2016.7744215).

Rolf, B., Reggelin, T., Nahhas, A., Lang, S., and Müller, M. 2020. "Assigning dispatching rules using a genetic algorithm to solve a hybrid flow shop scheduling problem," *Procedia Manufacturing* (42), pp. 442-449 (doi: 10.1016/j.promfg.2020.02.051).

Sabuncuoglu, I., and Goren, S. 2009. "Hedging production schedules against uncertainty in manufacturing environment with a review of robustness and stability research," *International Journal of Computer Integrated Manufacturing* (22:2), pp. 138-157 (doi: 10.1080/09511920802209033).

Sanko, J., and Kotkas, V. 2016. "Ontology-Driven Scheduling System for Manufacturing," *Baltic Journal of Modern Computing* (4:3), pp. 508-522.

Saqlain, M., Ali, S., and Lee, J. Y. 2022. "A Monte-Carlo tree search algorithm for the flexible job-shop scheduling in manufacturing systems," *Flexible Services and Manufacturing Journal* (doi: 10.1007/s10696-021-09437-4).

Schmidt, G. 1996. "Modelling production scheduling systems," *International Journal of Production Economics* (46-47), pp. 109-118 (doi: 10.1016/0925-5273(95)00019-4).

Schneeweiß, C. 1999. *Einführung in die Produktionswirtschaft*, Berlin, Heidelberg, s.l.: Springer Berlin Heidelberg.

Schuh, G. 2007. *Produktionsplanung und -steuerung*: *Grundlagen, Gestaltung und Konzepte*, Berlin, Heidelberg: Springer.

Shalev-Shwartz, S., and Ben-David, S. 2014. *Understanding Machine Learning: From Theory to Algorithms*, Cambridge University Press.

Shukla, O. J., Soni, G., Kumar, R., and A, S. 2018. "An agent-based architecture for production scheduling in dynamic job-shop manufacturing system," *at - Automatisierungstechnik* (66:6), pp. 492-502 (doi: 10.1515/auto-2017-0119).

Smith, S. F., and Becker, M. A. 1997. *An Ontology for Constructing Scheduling Systems*.

Stack Overflow. 2022. "Stack Overflow Developer Survey 2022," available at https://survey.stackoverflow.co/2022, accessed on Mar 6 2023.

Taillard, E. 1993. "Benchmarks for basic scheduling problems," *European Journal of Operational Research* (64:2), pp. 278-285 (doi: 10.1016/0377-2217(93)90182-M).

Tamaki, H., Kryssanov, V. V., and Kitamura, S. 1999. "A simulation engine to support production scheduling using genetics-based machine learning," in *Global Production Management*, K. Mertins, O. Krause and B. Schallock (eds.), Boston, MA: Springer US, pp. 482-489 (doi: 10.1007/978-0-387-35569-6_59).

Tao, F., Cheng, Y., Zhang, L., and Nee, A. Y. C. 2017. "Advanced manufacturing systems: socialization characteristics and trends," *Journal of Intelligent Manufacturing* (28:5), pp. 1079-1094 (doi: 10.1007/s10845-015-1042-8).

Tavakkoli-Moghaddam, R., Rahimi-Vahed, A. R., and Mirzaei, A. H. 2008. "Solving a multi-objective no-wait flow shop scheduling problem with an immune algorithm," *The International Journal of Advanced Manufacturing Technology* (36:9-10), pp. 969-981 (doi: 10.1007/s00170-006-0906-7).

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), pp. 1-6.

TypeScript. 2023. "TypeScript: JavaScript with syntax for types.," available at https://www.typescriptlang.org/, accessed on Mar 6 2023.

Varela, L., Aparício, J., and Silva, C. 2005. *Production Scheduling Concepts Modeling through XML*.

Vela, C. R., Varela, R., and 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), pp. 139-165 (doi: 10.1007/s10732-008-9094-y).

Vidoni, M., and Vecchietti, A. 2016. "Towards a Reference Architecture for Advanced Planning Systems," in *Proceedings of the 18th International Conference on Enterprise Information Systems*, SCITEPRESS - Science and and Technology Publications (doi: 10.5220/0005785804330440).

vom Brocke, J., Simons, A., Niehaves, B., Riemer, K., Plattfaut, R., and Cleven, A. 2009. "Reconstructing the giant: On the importance of rigour in documenting the literature search process," in *ECIS 2009 Proceedings*.

Wang, K., Choi, S. H., Qin, H., and Huang, Y. 2013. "A cluster-based scheduling model using SPT and SA for dynamic hybrid flow shop problems," *The International Journal of Advanced Manufacturing Technology* (67:9-12), pp. 2243-2258 (doi: 10.1007/s00170-012-4645-7).

Waschneck, B., Reichstaller, A., Belzner, L., Altenmuller, T., Bauernhansl, T., Knapp, A., and Kyek, A. 2018. "Deep reinforcement learning for semiconductor production scheduling," in *2018 29th Annual SEMI Advanced Semiconductor Manufacturing Conference (ASMC),* Saratoga Springs, NY, USA, IEEE, pp. 301-306 (doi: 10.1109/ASMC.2018.8373191).

Webster, J., and Watson, R. T. 2002. "Analyzing the Past to Prepare for the Future: Writing a Literature Review," *MIS Quarterly* (26:2), pp. xiii-xxiii.

Xue, Y., Yang, Q., and 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,* Busan, South Korea. 2-6 Nov. 2004, IEEE, pp. 1279-1283 (doi: 10.1109/IECON.2004.1431760).

Yang, W., and Takakuwa, S. 2017. "Simulation-based dynamic shop floor scheduling for a flexible manufacturing system in the industry 4.0 environment," in *Winter Simulation Conference*, W. K. V. Chan (ed.), Las Vegas, NV, IEEE, pp. 3908-3916 (doi: 10.1109/WSC.2017.8248101).

Yingzi, W., Xinli, J., Pingbo, H., and Kanfeng, G. 2009. "Multi-agent Co-evolutionary Scheduling Approach Based on Genetic Reinforcement Learning," in *2009 Fifth International Conference on Natural Computation,* Tianjian, China. 8/14/2009 - 8/16/2009, IEEE, pp. 573-577 (doi: 10.1109/ICNC.2009.475).

Zhang, R., Song, S., and Wu, C. 2020. "Robust Scheduling of Hot Rolling Production by Local Search Enhanced Ant Colony Optimization Algorithm," *IEEE Transactions on Industrial Informatics* (16:4), pp. 2809-2819 (doi: 10.1109/TII.2019.2944247).

Zhao, F., Hong, Y., Yu, D., Chen, X., and Yang, Y. 2005. "Integration of Artificial Neural Networks and Genetic Algorithm for Job-Shop Scheduling Problem," in *Advances in Neural Networks — ISNN 2005*, J. Wang, X. Liao and Z. Yi (eds.), Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 770-775 (doi: 10.1007/11427391_123).

Zheng, Y.-L., Li, Y.-X., Lei, D.-M., and Ma, C.-X. 2010. "Scheduling fuzzy job shop using random key genetic algorithm," in *2010 International Conference on Machine Learning and Cybernetics*, Qingdao, China. 7/11/2010 - 7/14/2010, IEEE, pp. 1887-1892 (doi: 10.1109/ICMLC.2010.5580535).

Zhou, Y., Yang, J.-J., and Zheng, L.-Y. 2019. "Multi-Agent Based Hyper-Heuristics for Multi-Objective Flexible Job Shop Scheduling: A Case Study in an Aero-Engine Blade Manufacturing Plant," *IEEE Access* (7), pp. 21147-21176 (doi: 10.1109/ACCESS.2019.2897603).