Journal of Intelligent Manufacturing (2022) 33:911–927
https://doi.org/10.1007/s10845-022-01915-2

On reliability of reinforcement learning based production scheduling systems: a comparative survey

Constantin Waubert de Puiseau[1] · Richard Meyes[1] · Tobias Meisen[1]

Received: 7 April 2021 / Accepted: 20 January 2022 / Published online: 5 February 2022

Abstract

The deep reinforcement learning (DRL) community has published remarkable results on complex strategic planning problems, most famously in virtual scenarios for board and video games. However, the application to real-world scenarios such as production scheduling (PS) problems remains a challenge for current research. This is because real-world application fields typically show specific requirement profiles that are often not considered by state-of-the-art DRL research. This survey addresses questions raised in the domain of industrial engineering regarding the reliability of production schedules obtained through DRL-based scheduling approaches. We review definitions and evaluation measures of reliability both, in the classical numerical optimization domain with focus on PS problems and more broadly in the DRL domain. Furthermore, we define common ground and terminology and present a collection of quantifiable reliability definitions for use in this interdisciplinary domain. Concludingly, we identify promising directions of current DRL research as a basis for tackling different aspects of reliability in PS applications in the future.

Keywords Reinforcement learning · Production scheduling · Reliability · Robustness · Machine learning

Introduction

For many decades, finding an optimal production schedule has been a challenging and active field of research for its appealing combinatorial logic and application in real-world sociotechnical systems. In the past and present, production scheduling (PS) problems are solved using heuristics or numerical optimization methods. Both have their advantages and limitations: heuristics, such as Earliest Due Date First, often serve as good and easily-understandable guidelines for the decision-making process, but may not perform well on all types of tasks (Martí et al., 2018). Numerical optimization methods are more complex to apply, as the production environment and objectives need to be mathematically described in great detail and then solved numerically, but find provably more optimal solutions (Pinedo, 2016).

With remarkable results in board and video games (Badia et al., 2020; Vinyals et al., 2019), ongoing fast development and successful examples of the transfer of Reinforcement Learning (RL) algorithms, especially Deep Reinforcement Learning (DRL), from simulation to the real world (Nevena Bellemare et al., 2020; Lazic et al., 2018), RL has re-emerged as a promising third alternative for solving PS problems in the future. The motivating vision behind RL-based PS lies in the nature of PS problems as illustrated in Fig. 1. The left-hand side represents the environment of the PS system. In its entirety, the environment is a sociotechnical system with various components. These components by themselves exhibit stochastic behavior, such as fluctuations of productivity of a worker. Interconnectivity and mutual influence in obvious (machine—tool) and non-obvious ways (weather—process parameters) add even more complexity to the task of manually creating an accurate mathematical model of the system. With RL, however, hopes are that through interaction with the real-world environment, the RL agent can create an approximation of how the environment behaves. This rationale also extends to the right-hand side of Fig. 1 showing the evaluation criteria of the schedule. More abstract evaluation criteria, such as profit, are not easily correlated to ground-level components of the production system. The complexity of the evaluation is further increased with the addition of stability, robustness and risk criteria, often summarized in the term

✉ Constantin Waubert de Puiseau
waubert@uni-wuppertal.de

[1] Chair of Technologies and Management of Digital Transformation, University of Wuppertal, Wuppertal, Germany

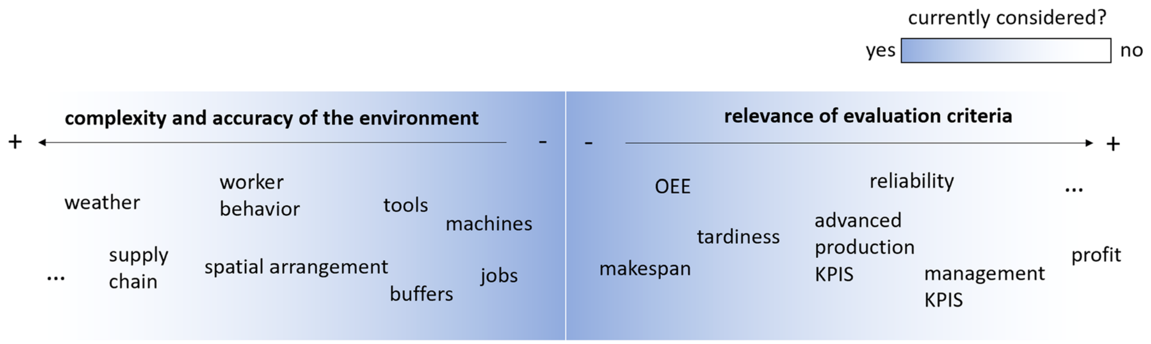


Fig. 1 The dimensions of the real-world PS problem

reliability. Consequently, the vision of end-to-end RL-based PS translates to providing the algorithm fine-grained information (left end in Fig. 1) and receiving actions maximizing high-level key performance indicators (KPIs) (right end in Fig. 1) in return.

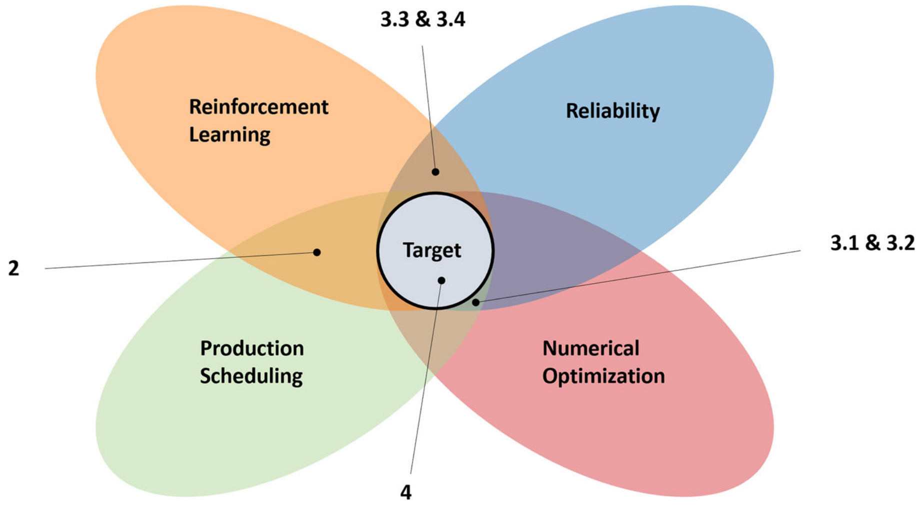
On the way to achieving this vision, a large and well-funded research community is making promising advancements in tackling large and continuous environment descriptions (Mnih et al., 2013; Vinyals et al., 2019). In the direction of capturing all relevant evaluation possibilities, however, we have identified a large gap in research on reliability in connection with RL. In this paper, we address this reliability gap by shedding light on measures of and techniques for reliable PS solutions.

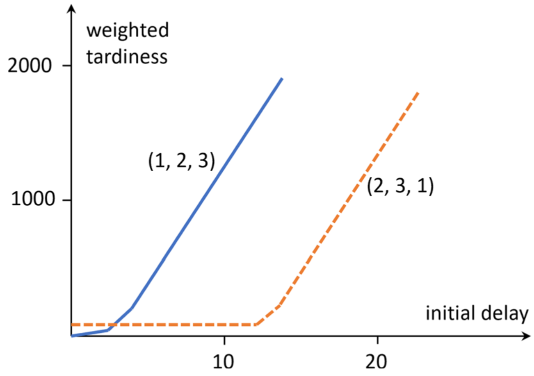
Contribution, scope and structure

This survey is aimed at understanding what reliability in PS means and how it can be addressed through RL-based solution approaches. To this end, we reviewed a large body of literature collected through a key-word search from SemanticScholar and GoogleScholar search engines. The key-words were tailored towards each section and are depicted in Fig. 2. The literature was first algorithmically filtered by relevance in the respective search engine and only the most relevant 500 papers considered. Subsequently, the body of literature was narrowed down through a manual search through titles and, later, abstracts.

The remainder of this article is structured as follows: we firstly review the state-of-the-art of RL-based PS and point out the lack of reliability awareness in (2). We then make use of literature from the mature field of classical optimization, narrowed down to the PS context, to collect a quantifiable definition of reliability (3.1) and common methods to achieve reliability (3.2). Next, we shift our view to the field of RL, where we examine reliability-related literature in a set framework of terminology and analyze measures (3.3) and methods (3.4) in this field. Having collected all aspects of reliability from both domains separately, we identify overlapping definitions and concepts. An overview of topics and respective sections in this survey are illustrated in Fig. 2. On that basis, we finally provide a structured list of recommended evaluation measures and most promising research directions for more reliable RL-based PS in future applications in (3.5).

Fig. 2 Illustration of the scope of evaluated literature and the target result





jobs	1	2	3
duration	10	10	10
due date	10	22	34
value	1	100	100

Fig. 3 Reliability example (adapted from (Pinedo, 2016))

Introduction to scheduling problems

According to Pinedo (2016), PS is a decision-making process that deals with the allocation of resources to tasks over a time period with the goal of optimizing one or more objectives in a production environment. In its most basic form, resources consist of raw materials and processing machines. More broadly, resources further include transport systems, tools, places for storage, human operating forces and more. Tasks, or jobs, are processing steps required to transform raw material into the desired output product and take up resources and time. Tasks may range from deformative manufacturing processes to transport and quality checks. Objectives of PS typically revolve around minimizing time, cost and tardiness in the production process. PS problems are combinatorial problems by nature and often subject to a variety of interdependencies and constraints of resources and tasks. The solution to PS problems, the schedule, is a plan of which job to start at which time step, often represented as Gantt chart (Pinedo, 2016).

In PS literature, abstracted and simplified combinatorial sub-problems are often the subject of investigation. One such abstraction is the Job-Floor Scheduling Problem (JFSP), in which it is assumed that machines are set up in series and every raw piece goes through the same tasks in the same order. A more complex abstraction is the Job-Shop Scheduling Problem (JSSP), in which the jobs (tasks) differ for different raw parts. Industrial applications often introduce very specific requirements regarding the problem and solution formalizations (Abdolrazzagh-Nezhad and Abdullah, 2017; Allahverdi, 2016; Fuchigami & Rangel, 2018).

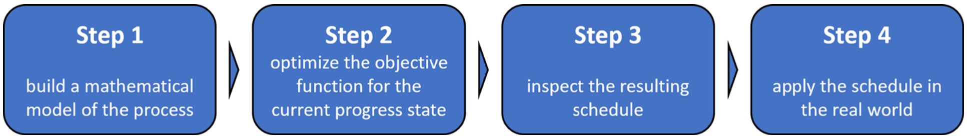
Even simplified production scenarios are not trivial to solve and evaluate in presence of uncertainties (Tarek Chaari et al., 2014). For illustration purposes, consider the following JFSP from Pinedo (2016). Three jobs are sequentially scheduled on a single machine. The jobs are independent of each other and characterized by a processing duration, a set due date and a weighting factor for how much they are worth if finished on time. The objective is to minimize the weighted tardiness. All parameters are shown in Fig. 1, right. Without disruptions, the optimal solution is to schedule the jobs in order (1, 2, 3), because all jobs can be finished before their respective due date and the weighted tardiness is zero. If we now consider random downtimes of the machine before the first job, the optimal schedule becomes a function of the downtime. For a delay of one time-unit, (1, 2, 3) is still optimal. For a delay of ten time-units, however, neither job 1 nor job 2 are finished on time. In this case, a scheduling order of (2, 3, 1) would have been better. The weighted tardiness for both schedules as a function of early downtime is plotted in Fig. 3, right. Hence, (2, 3, 1) can be considered a more robust, or risk-aware schedule.

Classical methods for solving PS problems follow the four-step approach shown in Fig. 4. A mathematical model of the production environment builds the foundation. Then, initial conditions featuring the job description are passed into the model and the resulting equations are optimized for a defined objective function.

Scheduling can take place in an online or offline fashion. Figure 4 describes an offline scheduling system, in which a schedule is prepared before the process starts and then remains unmodified. Online scheduling systems monitor the process and decide which action to take in the next step.

Introduction to deep reinforcement learning

Scheduling and other discrete time-dependent problems can often be formulated as Markov Decision Processes MDPs.


Step 1 build a mathematical model of the process

Step 2 optimize the objective function for the current progress state

Step 3 inspect the resulting schedule

Step 4 apply the schedule in the real world

Fig. 4 Four-step approach for scheduling with classical numerical optimization

In such, the system, or environment, is initially described through a state $s_0 \in S$ at time$t = 0$, at which an action $a_0 \in A$ is taken, leading to a new state$s_{0+1} = 0$, based on the underlying transition model$T : StxAtxSt + 1 \rightarrow [0, 1]$. For every discrete state-action-state transition, the reward function $R : StxAtxSt + 1 \rightarrow \mathbb{R}$ quantifies how good the transition is for fulfilling the underlying task. RL algorithms exploit this structure through interaction with the environment over time, finding optimal actions for each observed state to maximize the cumulative reward in a set time period. The interacting algorithms are also referred to as RL agents (Sutton & Barto, 2018).

For example, in the context of a JSSP, the state may include all status information of machines and jobs. The actions may be the decision which job to schedule next and the reward may be a positive value based on the number of parts finished in the last time-step. Inherently, RL is used in online scheduling schemes, taking one action in each time step.

RL can roughly be divided into value-based and policy-based methods. Value-based methods assign Q-values to all possible actions given state s, representing the expected discounted cumulative reward:

$$Q^{\pi}(s, a) = \mathbb{E}_{\pi}\left[\sum_{k=0}^{\infty} \gamma^{k} R(s_t|a_t)|s_0 = s, a_0 = a\right]$$

The RL agent's policy $\pi(s|a)$ is then determined by choosing the action with the maximum corresponding Q-value. For sufficiently small problems, Q-values for each discrete state-action-pair may be stored in a lookup table. This becomes infeasible for very large or even continuous state and action spaces. In Deep Reinforcement Learning DRL, the Q-values are represented through a parameterized function, a deep neural network, which maps the state observation to Q-values. Such functions are called Deep-Q-Networks DQNs (Mnih et al., 2013). In contrast to lookup tables, DQNs will return Q-values even for previously unseen states. The ability to form a useful representation of the underlying problem for unseen states is called generalization (Witty et al., 2018). Policy-based methods optimize the policy π more directly, affectively mapping a state to an optimal action. The policy function is also represented through a deep neural network, in which updates follow the gradient

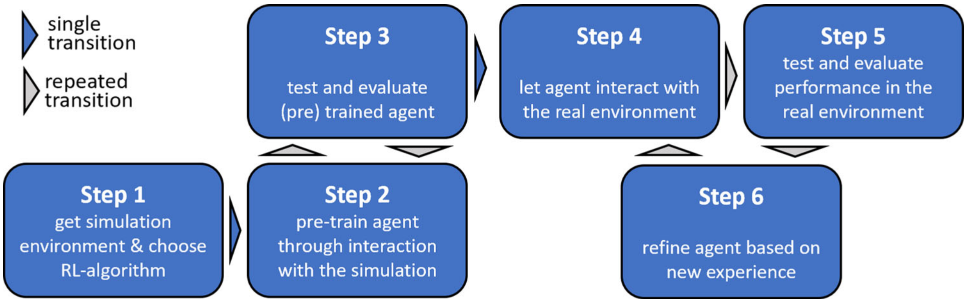
$$\nabla_{\theta} J(\theta|s) = \mathbb{E}_{\pi_{\theta}}\left[Q^{\pi}(s, a) \bullet \nabla_{\theta}\log\pi_{\theta}(s, a)\right]$$

first proposed in Sutton et al. (1999). In both cases, the parameters are iteratively updated through observations of the environment, collected from experienced trajectories in the environment, also called episodes.

Figure 5 illustrates the typical stepwise application process of RL-Algorithms to real-world problems (Dulac-Arnold et al., 2019; Tobin et al., 2017). The first step consists of building a simulation environment and choosing a suitable RL algorithm based on specified properties of the state-, observation-, action- and reward-space. In step two, the algorithm is pre-trained through interaction with the simulation. The next step is intensive testing and an evaluation of the pretrained agent in many scenarios. If the results fulfill the expectations, the trained agent is used to control the real environment in step 4, typically under close human supervision. Around the transition from one system to another, e.g., simulation to reality, a whole research branch called transfer learning has developed (Da Silva & Costa, 2019). Optionally, the agent then continues to learn from newly experienced interactions in the real environment to refine its policy and close the gap between simulation and reality, as depicted in steps 4 through 6.

Since the control action of the RL agent is one suggested action per output for each new observation, the RL agent's behavior can be seen as strictly reactive. Therefore, one does not obtain a whole schedule at the first timestep, but only in retrospect, after looking at the actions taken over time. This is a fundamental difference between any online-scheduling routine and offline optimization methods: an a priori analysis and evaluation of a strategy in the real environment is not possible. The only two options for evaluating a schedule are: (1) A run-through of the whole simulated production environment before any actions are taken in the real environment. (2) A posteriori evaluation.

Fig. 5 Illustration of the six-step approach used for training and deployment of RL-agents


single transition
repeated transition
Step 3
test and evaluate (pre) trained agent
Step 4
let agent interact with the real environment
Step 5
test and evaluate performance in the real environment
Step 1
get simulation environment & choose RL-algorithm
Step 2
pre-train agent through interaction with the simulation
Step 6
refine agent based on new experience

State-of-the-art of reinforcement learning based production scheduling

This section provides a summary of state-of-the-art attempts at solving PS problems using RL. We discuss recent successes and limitations of RL-based approaches to deal with real-world production scenarios.

Arviv et al. (2016) examined a Q-learning approach to a PS problem, controlling the dispatching of raw material to machines by transport robots. Based on a state representation of whether buffers and machines were occupied or not, the robots learned to transport material between buffers. The focus lay on the collaboration of robots and reward systems, aiming to optimize the completion time. The results show that the algorithms improve with training but are not benchmarked against an existing approach.

Luo (2020) used a DQN to map a generic representation of the production state to action values, which represent six different set dispatching rules, aiming to minimize the total tardiness. The problem is described as a flexible job shop problem with new insertions, meaning that new jobs can be added during the execution of a schedule. The generic observed state representation consists of high-level features, such as the utilization rate of machines or the average completion rate of jobs at each time step, so that the state representation is independent of the job sizes, number of machines and other scaling factors. This choice is a trade-off for observation accuracy since job and task-specific features are neglected. Their results show a better average performance of the DQN-algorithm in many, but not all test scenarios, when compared to each of the six dispatching rules separately.

Inspired by AlphaZero's (Silver et al., 2018) success in board games, (Rinciog et al., 2020) applied an adapted algorithm to offline scheduling (obtained in a simulation) in a sheet metal production scenario to optimize tardiness and material waste. The situational state space contains information about whether a machine is used at the moment, the job value and status as well as remaining slack times. An action determining which job is scheduled next is requested whenever a machine is idle. The RL agent is pretrained to approximate an earliest due date (EDD) heuristic and then further refined, eventually outperforming the EDD heuristic and a shallow random Monte Carlo Tree Search (MCTS) benchmark on average. The results present a meaningful proof-of-concept, but given that AlphaZero was applied to a very simplified and static scenario, the question of real-world applicability remains largely undiscussed.

Waschneck et al. (2018) applied DQNs to control workstations in a production environment, observing the availability of machines and job characteristics and mapping the current state to the position of a job in the planning schedule. The DQNs are trained to jointly maximize the throughput of the factory. Comparing the average throughput over all tests, the trained agents achieve results comparable to human experts but worse than a heuristic dispatching rule based on the job due dates and first-in-first-out principles.

Kuhnle et al. (2020) studied different design choices, e.g., reward formulations and definitions of the episode length in the continual production process, for building a PS system using the TRPO (Schulman et al., 2015) RL algorithm. The state-space consists of machine and job, including current failure states, remaining processing times, buffer spaces and waiting times and is mapped to actions corresponding to the transport of pieces from A to B. They optimize a multi-objective function and evaluate the machine utilization and inventory levels. Depending on the evaluation metric, the RL agent shows a similar or slightly better performance than heuristics. The study empirically showed that more information in the state space leads to better overall results, since a balance between common heuristics can be found.

Lang et al. (2020) trained two separate DQN based RL agents in sequence on the allocation of jobs to machines and operation sequence selection, respectively. Each PS problem was characterized by the specific number of jobs, with different deterministic processing times and due dates. The resulting performance was evaluated based on the total time to finish all jobs and the tardiness (sum of the difference between due dates and finishing times of each job) and benchmarked against a metaheuristic approach. Their algorithms showed comparable or better performance on different problem sizes.

The solutions above give the promising impression that RL is generally applicable for PS problems. However, regarding the state-space complexity (cf Fig. 1, left), current research is still mainly in the proof-of-concept stage. The other axis for improvement is the basis on which production schedules are evaluated (cf. Figure 1, right). Recent literature mainly focuses on maximizing the average value of a KPI. Luo (2020) report the standard deviation of the obtained results over multiple runs, showing that their results often deviate by more than 100%. Random schedules and those obtained by dispatching rules showed significantly smaller relative standard deviations. A smaller deviation of the results can be a desirable property to build trust in the reliability of a system. Overall, the existing literature does not investigate reliability criteria, which are equally important as a large average return (Birolini, 2004).

One reason why these characteristics are not investigated may lay in the hope that by maximizing the average reward, the RL agent creates an implicit probabilistic model of the environment, including occasional disruptions, and might learn to choose the appropriate order accordingly. However, for management and workers to accept a new scheduling system, the evaluation of schedules for both KPIs and reliability is crucial. In some cases, it may be beneficial to be able to deliberately trade in some of the optimality of KPIs for reliability, as described in 1.2.

Review on reliability

Reliability definitions in production scheduling

In the broader engineering context, reliability of a system is defined as "the probability that the […] [system] is able to perform as required for a given time interval" (Birolini, 2004). It follows that reliability measures are based on system- and problem-specific requirements. For this investigation, we assume that some more fundamental requirements for a PS system, such as availability of the planning algorithm (Birolini, 2004; Policella et al., 2007), are given. According to Leusin et al. (2018), an optimal reliable dynamic scheduling system:

(1) anticipates common disturbances in the production scenario,
(2) can be adjusted to fit new circumstances with minimal change to the original plan, and
(3) quickly reacts to such new circumstances by suggesting the best adjustments.

The latter criterion is a property of the scheduling algorithm and its implementation and integration rather than that of a schedule itself. Instead, we focus on criteria (1) and (2), which are requirements of the product of said algorithm: the schedule.

Classically, the reliability of a schedule is described in terms of "robustness" or "stability" of the production process being executed according to that schedule (Goren & Sabuncuoglu, 2008). Definitions of these terms may differ depending on the specific context of the analyzed production system. Generally, robustness describes that performance measures are only minimally affected by the occurrence of a disruption, when following the schedule. Large stability, on the other hand, describes the small deviation of the timing of discrete events due to a disruption (Goren & Sabuncuoglu, 2008). Whenever a time shift of events impacts performance, it is evident that robustness and stability are no longer independent features. Yet, depending on the setup and performance measures, it has also been observed that optimizing a schedule for robustness yields less stable results and vice versa (Goren & Sabuncuoglu, 2008; Leon et al., 1994), suggesting a tradeoff between those two aspects of reliability. Robustness and stability both have their implications and importance, varying strongly depending on the production scenario and the subjective perspective on the optimality of a schedule. Aiming to present a comprehensive collection of commonly used reliability measures, we do not strictly discriminate robustness and stability in this work but treat all measures as independent facets to reliability instead.

Perhaps the most intuitive option to get a numerical value for reliability is to compare the original plan, or expected plan, to one or multiple test runs in the real system after the execution of that plan. By that measure, a larger difference indicates a smaller reliability of the original plan. Some used the sum of all deviations in completion times CTs of jobs. CT is defined as the time from start to finish of a job or the processing time plus the waiting time (Al-Hinai & ElMekkawy, 2011; Goren & Sabuncuoglu, 2008; Rahmani & Heydari, 2014). Other options to use the CT are to track the average deviation of CTs or the average deviation of CTs only of jobs, which are directly affected by a disturbance (Al-Hinai & ElMekkawy, 2011). Some researchers compared the expected and realized makespan, which is the time required to complete all jobs (Rahmani & Heydari, 2014; Shen et al., 2017), or the workload, which is the maximum running time of any machine until all jobs are finished (Shen et al., 2017). Note that all of these measures rely on an a priori calculation of the expected value and can only be used to assert the reliability of a schedule in retrospect, after its application. Since the expectation of the whole original schedule is needed, these measures cannot be applied to online scheduling systems.

Another option is to evaluate a fixed schedule for multiple estimated or simulated test runs with different initializations or parameterizations of the production environment and use minimum values of the objective function (performance measure) as an indication of how risky the produced schedules were. In the job shop scheduling literature, the maximum tardiness of production times is often the evaluated performance measure (Goren & Sabuncuoglu, 2008; Hall & Posner, 2004; Pinedo, 2016), representing the difference between a set due date and the completion or delivery time. Tardiness is a popular objective, since delayed deliveries often cause penalty payments and have a direct influence on profit margins. Tardiness is also linked to the concept of just-in-time production (Cheng & Podolsky, 1996). Other monitored measures for the multiple-test setup include the sum over all completion times (Wu et al., 2009), which is closely related to the sum of all finishing times, or the maximum realized makespan within all tests (Leusin et al., 2018; Luo, 2020; Zhu & Wang, 2017). The obvious advantage is that a reliability measure is given before the deployment in the real world. On the other hand, the accuracy of the results depends on how well the environment model can be parameterized.

Both approaches directly evaluate common production KPIs, making them easily interpretable and easily adjustable to any other production KPI of interest. Some examples may be found in Amrina and Yusof (2011). It can also make sense to modify the KPIs for a reliability measure. Pinedo (2016), for example, considers a weighting factor for tardiness, multiplying the importance or monetary value of a product with its tardiness. Another consideration is the difference in any objective value divided by the duration of the causing disturbance (Pinedo, 2016) as a measure for reliability. Hence, a large value suggests that small disturbances cause large harm. Another indicator for reliability is the sum of all differences in the objective value caused by a disruption of magnitude d, multiplied by the probability of the disruption having that magnitude (Pinedo, 2016). The indication relies on the knowledge about the disruption magnitude probabilities. This measure is very intuitive, since a human planner would also consider the probability of a disruption, based on experience or available data. Sotskov et al. (1997) took a different approach, defining a stability radius as the quantity of change in a single processing time, within which the found schedule is still optimal when tested. The stability radius can be determined with respect to any KPI, as long as a parameterizable production environment is available. Recent studies on reliability are borrowing from risk-measures in finance (Gleißner, 2011), using the value at risk VaR for reliability, which is the value of a worst defined quantile of the objective value distribution over all processes, or the average or conditional value at risk AVaR/CVaR (Göttlich & Knapp, 2020), which is the average of all objective values within a worst defined quantile. The VaR-related measures can be used whenever a statistical evaluation of the performance over multiple test runs is available. It is therefore applicable to both the statistical evaluation of differences between expectation and test and the evaluations of test runs.

All measures discussed so far are based on analyses of KPIs across real, estimated or simulated runs. Comparatively few measures directly characterize a set schedule before running and before testing it. One such measure is flex, which counts the number of pairwise jobs which are unrelated by precedence constraints. Another is fluidity, which is an estimate for the probability of localized changes absorbing a temporal variation instead of propagating it through the rest of the schedule. Unfortunately, flex and fluidity are only measurable if the problem is formulated as a partial order schedule, which is not always possible (N. Policella et al., 2007). In contrast to all negative measures, quantifying unreliability rather than reliability, (Pinedo, 2016) suggests that there are also positive indicators for reliability, such as the utilization of the bottleneck of a production scenario. Inherently, this measure requires the identification of all bottlenecks. Shen et al. (2017) further consider the presence of idle times between jobs or tasks as a characteristic which increases reliability. For this measure, it is particularly evident that classical optimality, such as a small makespan, and reliability measures may present themselves as counteracting trade-offs.

Noteworthily, criterion (2), though highly relevant since frequent rescheduling is a necessity in most real-world applications (Pinedo, 2016; Vieira et al., 2003), is not specifically investigated or even quantified in a large part of the screened literature. We believe that the main challenge for measuring this criterion is that the easy adjustability of a schedule depends not only on the schedule itself but also on the rescheduling routine. Rescheduling routines can roughly be divided into three categories: Firstly, common in the academic setting, “right-shift rescheduling” is applied, in which the production process is effectively put on hold for the duration of a disruption and then continued according to the original plan once the disruption is resolved. Secondly, “partial rescheduling” adjusts the original schedule, where it has been most affected by a disruption. The last category comprises complete rescheduling routines. Those may either take the original schedule into account or optimize the allocation of all remaining jobs fully anew (Leusin et al., 2018).

It follows that evaluating a schedule for criterion (2) depends on:

1. The timing of the disruption, since a disruption towards the end of a schedule does not influence already completed jobs.
2. The ability and focus of the algorithm to trade-off between an optimal new schedule and small adjustments to the original schedule.

One attempt to capture criterion (2) was made by Kouvelis and Yu (1997), who calculated the *deviation of positions* occupied by jobs after right-shifting following a disruption. In other words, the impact of a disruption on the order of the schedule was quantified. Another attempt was made by Bean et al. (1991), who counted the *number of reassigned jobs* after complete rescheduling following a disruption. Those two measures quantify the induced change by rescheduling and do not discriminate, whether the reliability stems from the original schedule being easily adjustable or if it stems from a reliable rescheduling routine. Though not specifically designed to measure criterion (2), the previously discussed measures flex, fluidity and the presence of idle times also indicate a positive characteristic of the original schedule with regard to criterion (2).

Approaches to achieve reliable schedules through classical optimization

In this chapter we briefly categorize approaches to achieve reliable schedules through classical optimization schemes along with selected examples, highlighting those elements and ideas, which may be adapted by RL-practitioners and clarifying the limits of the state-of-the-art.

In classical optimization for problem setups featuring uncertainties, there are two major paradigms: robust optimization and stochastic optimization. Robust optimization aims to find the worst possible parameter configuration and optimizes the schedule for this fixed setting. Examples of such scheduling problems and solution approaches may be found in Zhu and Wang (2017), Wiesemann et al. (2013), Takayuki Osogami (2012). Since robust optimization solves worst-case scenarios, resulting schedules are often over-conservative. Stochastic optimization aims to utilize knowledge about the uncertainty distributions to find a solution, which is optimal with a certain probability. Examples of stochastic optimization for scheduling problems are Bäuerle and Ott (2011), Bäuerle and Rieder (2014), Yoshida (2019), Prashanth (2014), Ruszczyński (2010), Chow, Ghavamzadeh, et al. (2018), Wu et al. (2009), Daniels and Carrillo (1997), Goldpîra and Tirkolaee (2019).

Others approached uncertainties by examining many solutions under different fixed parameterizations empirically, which could easily be adapted for RL-approaches, when suitable simulated environments already exist. Göttlich and Knapp (2020) simulated multiple solutions in a parameter range of disturbances and manually chose a trade-off between the optimality of the estimated objective function and risk measures. Similarly, Al-Hinai and ElMekkawy (2011) applied their found solution to many simulated test cases with generated breakdowns, subtracted reliability measures based on completion time deviations from the objective function and used a genetic algorithm to refine their solution accordingly. However, computational cost scales quadratically with the number of parameters, hence the computation of all parameter combinations and manual selection may not be feasible in practice, when the parameter space becomes larger or even continuous.

To the best of our knowledge, all of these solution concepts have not been empirically proven to work well on problems with large state and action spaces, but were tested on comparatively small and simplified problems. This raises the question, if the approaches will be transferrable to real-world problems with large and continuous state and action spaces as well as complex non-deterministic environment behavior.

Reliability definitions in deep reinforcement learning

Terminology, definitions and measures of reliability in DRL are not congruent with those in PS. This is because the application of RL to real-world problems introduces additional challenges on which the research community has recently concentrated (Dulac-Arnold et al., 2019). This section provides a summary of relevant challenges, relates them to consequences for the reliable application of RL to PS problems and introduces common evaluation metrics aiming to measure how well the challenge is overcome.

Some reliability measures in RL are noticeably different from those seen in classical optimization approaches. Since RL agents are usually used for online decision-making, meaning that the control actions are taken at each time step of the process, one can usually not evaluate the reliability of the whole schedule a priori. Instead, one must resort to a complete run-through on a simulation model or in the real world to obtain the whole schedule. Doing that, one must keep in mind that once a scheduling system has been transferred to the real-world environment, playing the schedule through in the simulation in advance to deployment may paint an inaccurate picture of the agent's performance because there will likely be a difference between the two environments.

Reliability considerations in DRL can roughly be divided into robustness, safety, performance consistency and stability, which are described in the following sections.

Robustness

Robustness can refer to a limited sensitivity of the performance of the RL agent towards.

1. noise in the observation (Ferdowsi et al., 2018),
2. inaccuracies in the control action (Tessler et al., 2019), or
3. non-deterministic transition behavior of the environment. (Abdullah et al., 2019; Mankowitz et al., 2018)

The goal of robustness is therefore to observe similar behavior and success in a similar setting. Robustness, by this definition, also relates to successful agent behavior in unknown but similar problem settings, often referred to as generalization ability (Kenton et al., 2019). The real system always differs from the simulation, because of inaccurate training models (Hiraoka et al., 2019). The real world is also inevitably subject to more noise and non-deterministic transition behavior, technical or human imprecision and errors in the real world. Hence, robustness is of utmost importance for the reliable transferal of the agent from simulation to reality. As one measure for robustness of the RL agent, one can log the performance during test-runs in the training and testing phase. Typically, the average cumulative reward and standard deviation of the reward in multiple runs are evaluated (Abdullah et al., 2019). Others have additionally tracked the CVaR during training runs (Hiraoka et al., 2019).

Safety

Safety in RL refers to the fulfillment of constraints in the

1. action- space or
2. state-space (Dalal et al., 2012; Cheng, Orosz, et al., 2019; Felix Berkenkamp et al., 2017)

during training and testing. Examples of constraints in the PS domain are physically impossible actions (action-space constraints), such as the scheduling of two jobs on a machine at the same time, or undesirable states, such as the large delay of the finishing time of a product. Hence, for PS, the term safety is not necessarily related to physical harm of human beings as much as it might be in autonomous driving. However, if certain actions or states are impossible, decisions on how to deal with the situation will be left to human decision makers, effectively introducing more randomness and noise to state transition behavior observed by the agent. This, in turn, can make a schedule unreliable. Hence, inexecutable actions and dangerous or impossible states should be avoided in a reliable schedule. Moreover, the schedule should account for small mistakes in the execution leading to such constraint violations. Inevitably, the line between robustness and safety can be blurry, since a lack of robustness might be the cause of constraint violations. Safety is typically not a problem in purely simulated problems such as the Atari environments (Bellemare et al., 2013), but highly relevant in real-world applications. The terminology about the concept of safety in RL is sometimes ambiguous. Consequently, considerations of safety may also appear in the literature as risk-sensitive behavior (Dabney et al., 2018) or cautious behavior (Zhang et al., 2020) by the RL agent.

In RL literature, engineers usually do not measure how safe a single episode is or was, but statistically evaluate a policy over multiple episodes in retrospect. Constraints are often indirectly incentivized through a cost or penalty term in the reward function. Therefore, most researchers log the *average and standard deviation of the cost shaped return* (Achiam et al., 2017; Bohez et al., 2019; Boutilier & Lu, 2016; Cheng, Orosz, et al., 2019; Cheng, Verma, et al., 2019; Chow, Nachum, et al., 2018; Derman et al., 2018). Some also separately *measure the average penalty* (Bohez et al., 2019) or the *average and variance of constraint values* (Cheng, Orosz, et al., 2019; Cheng, Verma, et al., 2019; Chow, Nachum, et al., 2018) or the *cumulative constraint violation values* (Dalal et al., 2012; Yang et al., 2020). Another option is to *count all constraint violations* (Pinedo, 2016) and *calculate the percentage of tests ending in pre-defined catastrophes* (Kenton et al., 2019) to evaluate safety.

Steady performance

In academic problems it is often sufficient to track the average test score or average regret during training and evaluation, to determine how well the algorithm fulfills a particular task (Badia et al., 2020; Duan et al., 2016; Henderson et al., 2017; Osband et al., 2020). However, RL algorithms often exhibit high variance in performance in different runs, diminishing their reliability in terms of performance consistency (Cheng, Verma, et al., 2019). Note that the range in performance can have its root in the trained agent (inherently only successful on a subset of tasks (Yehuda et al., 2020)), in the faced problem (e.g., more or less complex setups of the problem), or in the agent's ability to compensate for non-deterministic transition behavior in the environment. Note that the latter is strongly tied to robustness.

In some use cases, a minimum performance threshold may be necessary at all times. Accordingly, Chan et al. (2020) recently defined reliability measures for the RL community: Firstly, in terms of dispersion of the obtained reward during training and testing, measured by the inter-quartile-range (IQR). Secondly in terms of risk, measured by the CVaR during training and testing. Though originally targeted at measuring reproducibility of the results, we argue that these measures, especially during testing, are also valuable for the definition of reliability in the context of a steady performance of a scheduling system.

Stability

We define the term stability of a RL algorithm, though it is often used interchangeably with robustness in RL literature (Chan et al., 2020; Chollet, 2019), as the successful, reproducible and smooth learning ability of an algorithm during training (Henderson et al., 2017). This ability can be hindered by

1. algorithm implementation parameters (Teh et al., 2017; Logan Engstrom et al., 2020),
2. random seeds in the initialization of parts of the algorithm or the environment (Chan et al., 2020) or
3. noisy reward signals (Fu et al., 2017; Henderson et al., 2017).

Importantly, stability is desired in the original training phase in the simulation but becomes a strong requirement when learning from the real environment, because the policy should not get worse through training. In this work, we will not consider the learning ability of algorithms but instead, focus on measures for the reliability of schedules after learning is complete.

Approaches to achieve reliability in DRL solutions

The previous section highlighted the main aspects to consider when aiming for more reliability in RL, namely safety, robustness and steady performance. This section summarizes recent work on these reliability aspects. Here, we do not limit this review to PS problems, but open the scope to all approaches covering relevant reliability aspects (cf. Figure 2.).

Fulfilling constraints (safety)

A large body of research focuses on safety, i.e., the fulfillment of constraints in the state and action space. At the latest, the fulfillment of constraints becomes relevant when the RL agent interacts with the real environment. Though agents can theoretically learn how impossible and undesired actions and states are treated through carefully hand-crafted negative rewards, in some cases, it is more convenient and precise to specify constraints (Achiam et al., 2017).

An active direction of research has emerged around (Risk-) Constrained MDPs (CMDPs). It involves the reformulation of an MDP to one with constraints on an expected cumulative cost or penalty. Note that the definitions of robust MDPs (cf. Section 3.2) and CMDPs are not mutually exclusive: The cost term in CMDPs can contain expressions for constraints not only of states and action but also rewards. Within this research direction, a lot of focus lies on safe (in the sense of remaining within constraints) exploration during training, which is particularly relevant when fine-tuning an agent in a real-world environment (Dalal et al., 2012; Wabersich & Zeilinger, 2018).

Achiam et al. (2017) employ a trust-region-optimization related approach called Constrained Policy Optimization (CPO). It is based on policy improvement steps which guarantee an increase in reward while satisfying constraints. Using this method, they guarantee constraint satisfaction both during training and testing. On the downside, their method is very conservative, since it ensures the satisfaction of all constraints in every time step and not just the result, which may not be needed in every case. Yang et al. (2020) extend the trust-region approach of Achiam et al. (2017) with a projection of the policy onto the closest constraint satisfying policy, so as to provide a lower bound on reward improvement and an upper bound on constraint violation for each policy update. Their algorithm performs well on several tasks with constrained state-spaces. If the approach performs well on environments with non-deterministic transition behavior, as present in many PS scenarios, is still subject of future research.

Another proposition on how to solve CMDPs was made by Tessler et al. (2018), who use the Lagrangian formulation of constraints for the reward function and for converging towards fulfilling these constraints during optimization steps. Relatedly, Bohez et al. (2019) regularize the optimization problem with the use of the Lagrangian formulation of state constraints, minimizing the penalty while respecting a lower bound on the performance on the task. Both latter approaches allow working with mean valued constraints, constraining the expected sum of a cost term over the whole episode, but do not strictly guarantee the satisfaction of constraints in every time-step. Stooke et al. (2020) related updates of the Lagrangian variable into integral control of classical control-theory in PID-controllers. Arguing that the integral control part oscillates without the proportional and derivative control parts, they added these to the Lagrangian update term for policy gradients, achieving a more stable learning process. The approach is especially interesting when the RL agent learns from direct interaction with the real environment and should not deviate much from its previous behavior.

Another control-theoretic approach is given by Fisac et al. (2019), who use a discounted safety formulation based on Hamilton–Jacobi safety analysis, learned as a separate value function in the Bellman equation. By manually defining failure states, the algorithm is kept from getting too close to those.

Zhang et al. (2020) derived a primal–dual policy gradient method with a risk-term, based on the policy's long-term state-action occupancy distribution, in the cost function. Despite the effectiveness of the algorithm in the exemplary grid-world setting, its extension to large and continuous state and action spaces remains unsolved.

Cheng, Verma, et al. (2019) suggest a regularization technique to stay close to a hand-crafted controller policy (prior). They show that given a prior with control-theoretic constraint guarantees, the prior can be approximated and improved through policy gradients, while experiencing smaller variance and restricting the action space both during training and testing compared to learning from scratch. Unfortunately, the hand-crafted prior is typically unattainable in the PS problem.

A supervised learning based approach, called intrinsic fear, was suggested by Lipton et al. (2016) to guide the policy away from undesired states. In it, states are first categorized as avoidable catastrophes, if they should not be visited by an optimal policy, and as danger states, which are close to catastrophic states. A neural network then learns to classify these states and penalizes the Q-learning target according to the observed state. The idea is interesting, since a neural network could also be used to learn undesirable states from experience. However, due to the black-box character of neural networks, it is unclear if the accuracy of the classification of safe and unsafe states is sufficient.

Another popular way to ensure constrain compliance is to modify or override an action if it would otherwise lead to a constraint dissatisfaction. One such method is shielding (Mohammed Alshiekh et al., 2018; Osbert Bastani, 2019), where a safe backup policy is known and applied if needed. This only works, if such a backup policy exists. Others have suggested ways to change the action slightly to stay on a safe policy parameter set, if the next step is predicted to be unsafe by a critic-like structure (Chow, Nachum, et al., 2018; Dalal et al., 2012; Wabersich & Zeilinger, 2018). However, these methods rely on expert knowledge of unstable equilibrium points, which are more obvious in mechanical systems than for scheduling tasks.

Robustness

The ability to perform well in new situations and under noisy observation, action or transition behavior is not necessarily improved through increased training volume, model capacity or exploration alone (Witty et al., 2018). Robust MDPs were introduced to tackle uncertainty in environment parameters and find strategies which assume worst-case constellations. Tamar, Glassner, et al. (2015) form a CVaR-gradient to guide policy updates in a robust MDP setting, showing that their approach results in conservative policies, which trade off peak performance for less risky strategies, i.e., better CVaRs. Refining this approach in Tamar, Chow, et al. (2015), both the VaR of the total discounted return and assessments of the risk in intermediate steps were included to capture static and dynamic risk. The assessment is performed through a sampling-based critic. Both approaches were found to be overly-conservative in some scenarios because they assume worst-case conditions. These assumptions were relaxed for so-called soft-robust policy gradients, in which worst-case scenarios and average scenarios are weighted by an importance parameter (Derman et al., 2018; Hiraoka et al., 2019). We believe this to be a promising direction of research for PS regarding the stochasticity of real-world environments.

Another option is to adjust the training process so that the space of unknown scenarios becomes smaller. One common challenge is that not every action is executed exactly according to the policy output, but may differ in timing, intensity or be entirely different. This challenge is addressed in action robust reinforcement learning. Tessler et al. (2019) trained an RL agent, for which the output action was altered with a set probability or altered by an adversary algorithm. That way, they obtained a policy, which was less sensitive to action perturbations. A similar approach was taken in Pinto et al. (2017), who used an adversarial agent in a zero-sum game to add perturbations in the environment behavior to learn more robust policies for the simulation to reality transfer of an agent. If challenging environment setups are known, one can simulate manually defined worst-case transition models of the environment and train the agent to keep a minimum distance to these scenarios (Abdullah et al., 2019). Tobin et al. (2017) introduced domain randomization, which introduces synthetic variance in the observation space during training and has shown to improve generalization capabilities for the transfer from simulation to reality. Carefully chosen and random perturbations introduced in training can help to evaluate and enhance the robustness of the trained RL agent, but could also make the combinatorial scheduling problem harder to solve.

Robustness as the lack of generalization was also addressed by Kenton et al. (2019), who suggested training an ensemble of agents and averaging over their action values. The standard deviation within the different action prediction was leveraged to quantify the uncertainty of the policy at a particular state, enabling to set a threshold on uncertainty and "call for help", i.e., manual interference. If no backup policy is attainable in practice, the approach is useless during execution. However, the uncertainty of the ensemble might help to evaluate how robust the algorithm is during training.

Steady performance

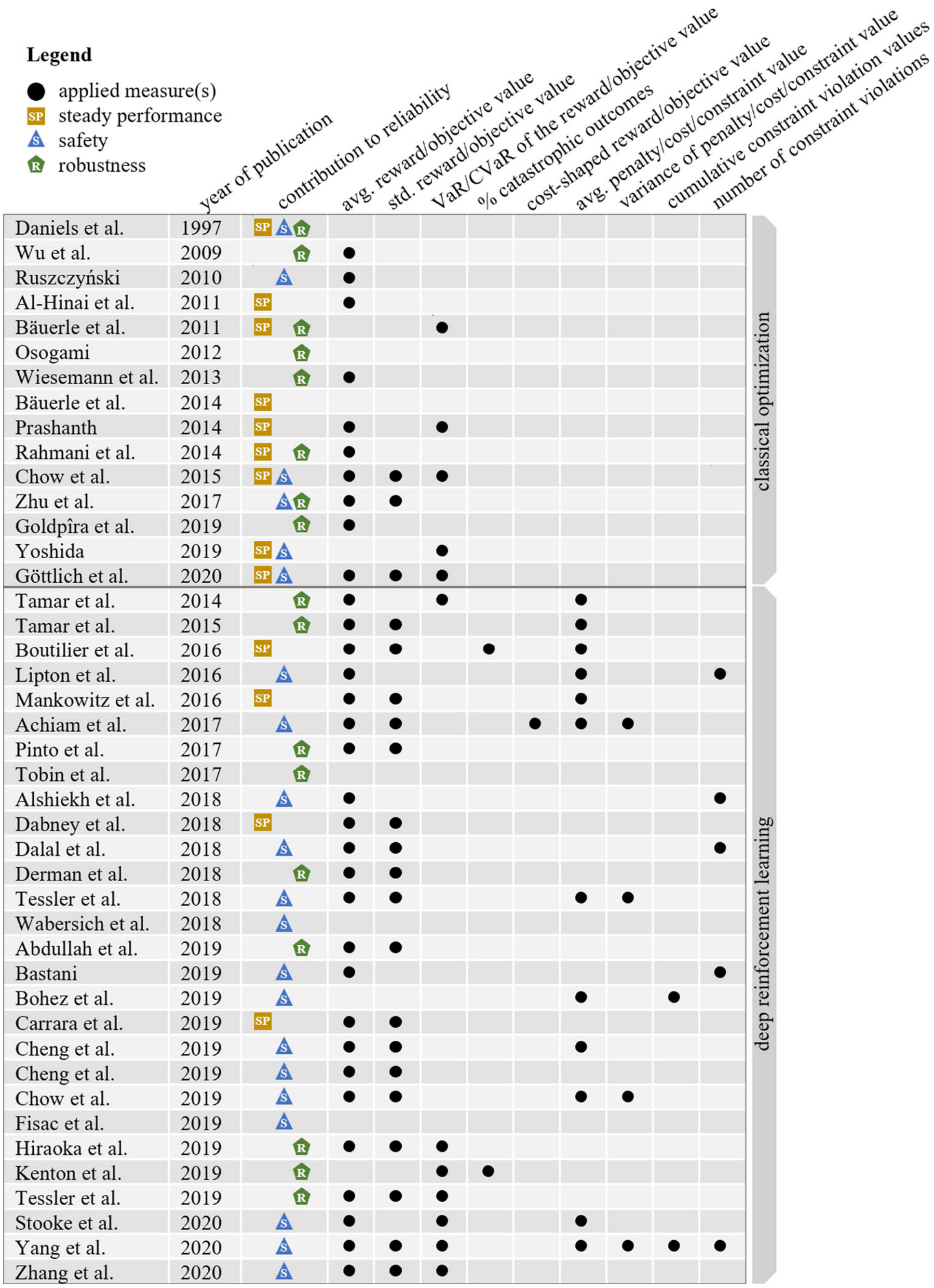
Some work on steady performance was already described in the section on "fulfilling constraints" above, when rewards were constrained to a certain window. Another approach for steadier performance is the Probabilistic Goal Semi-MDP (Mankowitz et al., 2016). In this formulation of MDPs, the objective is reformulated to consider the probability of the expectation to surpass a certain threshold. However, the transfer of that formulation to DRL has not been completed. Boutilier and Lu (2016) propose another variation to MDPs called Budgeted MDPs (BDMP). Their approach introduces the notion of a budgeted resource, e.g., consumed energy in a motion control example, which is limited within a period of time. The agents learn a policy including the tradeoff between resource consumption and reward, in which the policy is a function of the maximum expected consumption. Such an approach could be useful to budget the reliance of a strategy on risky states, if those states can be quantified. Carrara et al. (2019) further developed the approach to be applicable to continuous action spaces and unknown system dynamics.

A promising research branch is distributional RL, in which a distribution over the expected returns is learned. These distributions could naturally be used as trade-off between the maximum average reward and the worst possible outcome of an action, since they incorporate the uncertainty over possible returns (Dabney et al., 2018).

In conclusion of Sect. 3.4, we have identified different directions for improvement along the dimensions of constraint satisfaction, robustness and steadiness of performance in this section. A summary of the mentioned literature is given in Fig. 6. We have indicated, which aspect of reliability from the RL-perspective is primarily addressed by each publication and how this aspect has been quantified in the results for each publication. Regarding the used measures it should be noted that some measures were either supplemented or fully based on human expert inspections of the algorithm output behavior. None of the approaches have been applied to PS yet and it remains unclear, which approaches are applicable to scheduling problems in real-world environments.

Suggestions on how to deal with reliability in DRL-based scheduling solutions

Below, we summarize our findings and articulate recommendations for future research on RL-based PS. As stated



Legend

● applied measure(s)
SP steady performance
S safety
R robustness

		year of publication	contribution to reliability	avg. reward/objective value	std. reward/objective value	VaR/CVaR of the reward/objective value	% catastrophic outcomes	cost-shaped reward/objective value	avg. penalty/cost/constraint value	variance of penalty/cost/constraint value	cumulative constraint violation values	number of constraint violations
classical optimization	Daniels et al.	1997	SP S R									
	Wu et al.	2009	R	●								
	Ruszczyński	2010	S	●								
	Al-Hinai et al.	2011	SP	●								
	Bäuerle et al.	2011	SP R			●						
	Osogami	2012	R									
	Wiesemann et al.	2013	R	●								
	Bäuerle et al.	2014	SP									
	Prashanth	2014	SP	●		●						
	Rahmani et al.	2014	SP R	●								
	Chow et al.	2015	SP S	●	●	●						
	Zhu et al.	2017	S R	●	●							
	Goldpîra et al.	2019	R	●								
	Yoshida	2019	SP S			●						
	Göttlich et al.	2020	SP S	●	●	●						
deep reinforcement learning	Tamar et al.	2014	R	●		●			●			
	Tamar et al.	2015	R	●	●				●			
	Boutilier et al.	2016	SP	●	●		●		●			
	Lipton et al.	2016	S	●					●			●
	Mankowitz et al.	2016	SP	●	●				●			
	Achiam et al.	2017	S	●	●			●	●	●		
	Pinto et al.	2017	R	●	●							
	Tobin et al.	2017	R									
	Alshiekh et al.	2018	S	●								●
	Dabney et al.	2018	SP	●	●							
	Dalal et al.	2018	S	●	●							●
	Derman et al.	2018	R	●	●							
	Tessler et al.	2018	S	●	●				●	●		
	Wabersich et al.	2018	S									
	Abdullah et al.	2019	R	●	●							
	Bastani	2019	S	●								●
	Bohez et al.	2019	S						●		●	
	Carrara et al.	2019	SP	●	●							
	Cheng et al.	2019	S	●	●				●			
	Cheng et al.	2019	S	●	●							
	Chow et al.	2019	S	●	●				●	●		
	Fisac et al.	2019	S									
	Hiraoka et al.	2019	R	●	●	●						
	Kenton et al.	2019	R			●	●					
	Tessler et al.	2019	R	●	●	●						
	Stooke et al.	2020	S	●		●			●			
	Yang et al.	2020	S	●	●	●			●	●	●	●
	Zhang et al.	2020	S	●	●	●						

Fig. 6 Summary of literature concerning the reliability of scheduling algorithms

before, not all methods of classical numerical optimization may apply to RL, because RL-based approaches are online scheduling systems. We aim to answer the question, which measures and approaches to use given a particular problem setting.

Borrowing from classical numerical optimization approaches, we divide the considerations of reliability in RL-based PS into three main categories which build upon each other: the aspects of reliability of one fixed schedule, the aspects of reliability of a policy in terms of performance across multiple test runs and the reliability in terms of the ability to reschedule well.

The reliability of a single schedule

Obtaining one schedule through RL involves logging all actions taken by the RL agent over a certain period of time. The schedule can either be obtained from the simulated or the real environment. This single schedule can then be evaluated through any KPI of choice (e.g., makespan or tardiness). If the schedule stems from the real environment, it can only be benchmarked against another schedule for the same problem, if the environment is sufficiently simple. Otherwise, the interaction with the real environment is likely to affect the environment differently than the simulation, hindering the comparability. In that case, the stand-alone reliability measures can only be based on expert knowledge and opinion or may follow general principles. Such measures include:

1. M1. The utilization of bottleneck machines
2. M2. The presence of idle times

If the schedule stems from a simulation, one can evaluate the fixed schedule for several test runs in that simulation with different parameterizations (e.g., of machine breakdowns). During the test, parametrizations and KPIs of interest should be logged. Useful reliability measures then include the ones mentioned above and:

1. M1. Average and standard deviation of performance
2. M2. The worst performance
3. M3. The best performance
4. M4. The VaR/CVaR of the performance
5. M5. The stability radius
6. M6. The difference in performance divided by the change in parameterizations
7. M7. The difference in performance after a change in parameterizations times the probability of that change

If the problem is subject to constraints, it should also be logged if any constraint was violated and by how much. The above list can then be extended by:

1. M1. The percentage of tests within constraint violations
2. M2. The cumulative constraint violation values
3. M3. The average and variance of constraint violation values

The progression of reliability aspects for a single schedule goes hand in hand with methods to achieve them. The measures of a single, stand-alone schedule may be improved through reward shaping, exploiting expert knowledge to incentivize the right characteristics of the schedules. Testing the schedule in different parameterized environments hoping to measure similar performance is considered in robustness research in RL and robust and risk-constrained MDP formulations. Here, we have identified the following promising approaches:

1. A1. Domain randomization (Tobin et al., 2017)
2. A2. Adversarial agents (Pinto et al., 2017) and
3. A3. Soft-robust policy gradients (Derman et al., 2018; Hiraoka et al., 2019)

Dealing with constrained problems, one should resort to the safe RL literature. For this case, we believe.

1. A1. Projection Based Constrained Policy Optimization (Yang et al., 2020)
2. A2. PID Lagrangian Methods (Stooke et al., 2020) to be the most promising research directions for PS.

Reliability of a policy

To measure the reliability of a trained policy, it makes sense to consider more than one problem during testing. However, all measures and methods of the previous section also apply. The measures can be extended by a statistical analysis of performance and reliability measures of a single schedule. Thus, they include:

1. M3. Average and standard deviation of performances
2. M6. VaR/CVaR of performances
3. M13. Average and standard deviation of reliability measures of a single schedule (M1-M12)
4. M14. VaR/CVaR of reliability measures of a single schedule (M1-M12)

The methods are extended by those considering steady performance. Particularly.

1. A1. Distributional RL (Dabney et al., 2018)

Reliability of rescheduling algorithms

For any consideration of rescheduling, one needs to compare an original planned schedule to a newly planned schedule. Since RL-based PS is an online scheduling scheme, rescheduling is only of interest, if the original schedule is formed as a sequence of actions taken in a simulation. To the best of our knowledge, rescheduling has not yet been explicitly studied in RL-based PS. Our best guess is that little changes to an original schedule would be implemented through penalizing or constraining changes to the original schedule. Although we believe the applicable use-cases to be limited and found measures to be rather vague (cf. Section 3.2), we name them here for completeness:

1. M15. The deviation of task positions
2. M16. The number of re-ordered jobs
3. M17. Flex
4. M18. Fluidity
5. M19. Presence of idle times

More research on existing problems and the demand for reliability in rescheduling should be conducted to address this challenge more clearly.

Conclusion and future work

We have identified a lack of considerations of reliability in current RL-based PS attempts. This lack is a fundamental obstacle to the confident application of DRL to PS problems. Since this research field is still in its infancy today, there exists no ready-to-use schema or set of methods for the design of an RL-based PS solution to achieve competitive and reliable production schedules. A critical step towards such a schema or set of methods is the creation of a common understanding, i.e., terminology, definitions, and measures of reliability. This work contributes a documented and common starting point for the interdisciplinary discussion of reliability in the online PS and DRL community. The measures and definitions are meant to aid the community with translating real-world PS requirements to suitable DRL problem formulations, algorithms, solutions and evaluation metrics. To this end, we have mapped common problem settings of PS systems to applicable measures (M1-M19) and have suggested promising solution approaches (A1-6), conclusively detailed in Sect. 3.5. Most measures can easily be tracked and reported to facilitate the community to compare reliability aspects of found solutions, rooted in a common understanding of what these measures imply.

The next step towards reliable DRL-based PS lies in the empirical validation of the applicability of the found reliability aspects, i.e., safety, steady performance, and robustness. Future studies need to address the following questions in close collaboration of production planning experts and DRL experts:

1. Are the found reliability aspects sufficient for the quantification of reliability criteria for production schedules in the eyes of production planners and workers, or does the list need to be extended?
2. How can requirements for the performance and reliability expressed by production planners be quantitatively and unambiguously be translated to DRL training procedures and testing results?
3. How well are the found DRL-based approaches addressing aspects of reliability applicable to complex, non-deterministic production scenarios?

To answer these questions, we are currently creating simulation environments for PS scenarios of different complexity and with selected realistic constraints and stochastic qualities. We are planning on implementing the suggested solution approaches to test and validate their applicability. During this validation, we are planning on reporting all reliability measures to evaluate their expressivity in cooperation with industrial partners.

We hope that this work helps fellow researchers choose appropriate evaluation metrics for their respective scheduling algorithms and scenarios. We further hope to encourage the adaption of modern RL algorithms to more reliability-aware DRL-based PS.

Funding Open Access funding enabled and organized by Projekt DEAL. This work was funded by the German Federal Ministry for Economics and Energy (BMWi) within the research project AlphaMES.

References

Abdolrazzagh-Nezhad, M., & Abdullah, S. (2017). Job shop scheduling: Classification, constraints and objective functions. *International Journal of Computer and Information Engineering, 11*, 429–434.

Abdullah, M. A., Ren, H., Ammar, H. B., Milenkovic, V., Luo, R., Zhang, M., et al. (2019). Wasserstein robust reinforcement learning. https://arxiv.org/pdf/1907.13196.

Achiam, J., Held, D., Tamar, A., & Abbeel, P. (2017). Constrained policy optimization. In: ICML'17: Proceedings of the 34th international conference on machine learning (70), 22–31.

Al-Hinai, N., & ElMekkawy, T. Y. (2011). Robust and stable flexible job shop scheduling with random machine breakdowns using a hybrid genetic algorithm. *International Journal of Production Economics, 132*, 279–291. https://doi.org/10.1016/j.ijpe.2011.04.020

Allahverdi, A. (2016). A survey of scheduling problems with no-wait in process. *European Journal of Operational Research, 255*, 665–686. https://doi.org/10.1016/j.ejor.2016.05.036

Alshiekh, M., Bloem, R., Ehlers, R., Könighofer, B., Niekum, S., & Topcu, U. (2018). Safe reinforcement learning via shielding. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 32

Amrina, E., & Yusof, S. M. (2011). Key performance indicators for sustainable manufacturing evaluation in automotive companies. In: Proceedings of the 2011 IEEE international conference on industrial engineering and engineering management (IEEM), Singapore, Singapore, 12/6/2011–12/9/2011 (pp. 1093–1097). [Piscataway, NJ]: IEEE. doi:https://doi.org/10.1109/IEEM.2011.6118084.

Arviv, K., Stern, H., & Edan, Y. (2016). Collaborative reinforcement learning for a two-robot job transfer flow-shop scheduling problem. IEEE SMC 2013 Conference, 54(4), 1196–1209.

Badia, A. P., Piot, B., Kapturowski, S., Sprechmann, P., Vitvitskyi, A., Guo, D., et al. (2020). Agent57: Outperforming the Atari human benchmark. Proceedings of the 37th International Conference on Machine Learning, 37(119), 507–5017.

Bastani, O. (2019). Safe reinforcement learning with nonlinear dynamics via model predictive shielding. https://arxiv.org/abs/1905.10691.

Bäuerle, N., & Ott, J. (2011). Markov decision processes with average-value-at-risk criteria. *Mathematical Methods of Operations Research, 74*, 361–379. https://doi.org/10.1007/s00186-011-0367-0

Bäuerle, N., & Rieder, U. (2014). More risk-sensitive markov decision processes. *Mathematics of Operations Research, 39*, 105–120.

Bean, J. C., Birge, J. R., Mittenthal, J., & Noon, C. E. (1991). Matchup scheduling with multiple resources, release dates and disruptions. *Operations Research, 39*(3), 470–483.

Bellemare, M. G., Candido, S., Castro, P. S., Gong, J., Machado, M. C., Moitra, S., et al. (2020). Autonomous navigation of stratospheric balloons using reinforcement learning. *Nature, 588*, 77–82. https://doi.org/10.1038/s41586-020-2939-8

Bellemare, M. G., Naddaf, Y., Veness, J., & Bowling, M. (2013). The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research, 47*, 253–279. https://doi.org/10.1613/jair.3912

Berkenkamp, F., Turchetta, M., Schoellig, A., & Krause, A. (2017). Safe model-based reinforcement learning with stability guarantees. In: NIPS'17: Proceedings of the 31st International Conference on Neural Information Processing Systems, pp. 908–918.

Birolini, A. (2004). *Reliability engineering: Theory and practice*. Springer.

Bohez, S., Abdolmaleki, A., Neunert, M., Buchli, J., Heess, N., & Hadsell, R. (2019). Value constrained model-free continuous control. https://arxiv.org/pdf/1902.04623.

Boutilier, C., & Lu, T. (2016). Budget allocation using weakly coupled, constrained Markov decision processes. In: UAI'16: Proceedings of the Thirty-Second Conference on Uncertainty in Artificial Intelligence, 52–61.

Carrara, N., Leurent, E., Laroche, R., Urvoy, T., Maillard, O.-A., & Pietquin, O. (2019). Budgeted reinforcement learning in continuous state space. NeurIPS 2019: Advances in neural information processing systems, 32.

Chaari, T., Chaabane, S., Aissani, N., & Trentesaux, D. (2014). Scheduling under uncertainty: Survey and research directions. In: Proceedings of the 3rd international conference on advanced logistics and transport, 2014, IEEE. doi:https://doi.org/10.1109/ICAdLT.2014.6866316.

Chan, S. C. Y., Fishman, S., Canny, J., Korattikara, A., & Guadarrama, S. (2020). Measuring the reliability of reinforcement learning algorithms. International Conference on Learning Representations.

Cheng, R., Orosz, G., Murray, R. M., & Burdick, J. W. (2019). End-to-end safe reinforcement learning through barrier functions for safety-critical continuous control tasks. *Proceedings of the AAAI Conference on Artificial Intelligence, 33*, 3387–3395. https://doi.org/10.1609/aaai.v33i01.33013387

Cheng, R., Verma, A., Orosz, G., Chaudhuri, S., Yue, Y., & Burdick, J. W. (2019b). Control regularization for reduced variance reinforcement learning. In: Proceedings of the 36th international conference on machine learning (7).

Cheng, T. C. E., & Podolsky, S. (1996). *Just-in-time manufacturing: An introduction/T. C. E. Cheng and S. Podolsky* (2nd ed.). Chapman & Hall.

Chollet, F. (2019). On the Measure of Intelligence. https://arxiv.org/pdf/1911.01547.

Chow, Y., Ghavamzadeh, M., Janson, L., & Pavone, M. (2018). Risk-constrained reinforcement learning with percentile risk criteria. *Journal of Machine Learning Research, 18*, 1–51.

Chow, Y., Nachum, O., Faust, A., Duenez-Guzman, E., & Ghavamzadeh, M. (2018b). Lyapunov-based safe policy optimization for continuous control. In: NIPS'18: Proceedings of the 32nd International Conference on Neural Information Processing Systems, 8103–8112.

Da Silva, F. L., & Costa, A. H. R. (2019). A survey on transfer learning for multiagent reinforcement learning systems. *Journal of Artificial Intelligence Research, 64*, 645–703. https://doi.org/10.1613/jair.1.11396

Dabney, W., Ostrovski, G., Silver, D., & Munos, R. (2018). Implicit quantile networks for distributional reinforcement learning. In: Proceedings of the 35th International Conference on Machine Learning, 1096–1105.

Dalal, G., Dvijotham, K., Vecerik, M., Hester, T., Paduraru, C., & Tassa, Y. (2012). Safe exploration in continuous action spaces. *Journal of Artificial Intelligence Research, 45*, 1.

Daniels, R. L., & Carrillo, J. E. (1997). Beta-robust scheduling for single-machine systems with uncertain processing times. *IIE Transactions, 29*, 977–985. https://doi.org/10.1023/A:1018500319345

Derman, E., Mankowitz, D. J., Mann, T. A., & Mannor, S. (2018). Soft-robust actor-critic policy-gradient. In: Proceedings of the 35th International Conference on Machine Learning, vol. 80

Duan, Y., Chen, X., Houthooft, R., Schulman, J., & Abbeel, P. (2016). Benchmarking deep reinforcement learning for continuous control. In: Proceedings of the 33th international conference on machine learning, vol. 48, pp. 1329–1338.

Dulac-Arnold, G., Mankowitz, D., & Hester, T. (2019). Challenges of real-world reinforcement learning. ICML Workshop on Real-Life Reinforcement Learning.

Engstrom, L., Ilyas, A., Santurkar, S., Tsipras, D., Janoos, F., Rudolph, L., et al. (2020). Implementation matters in deep RL: a case study on PPO and TRPO. In: Eighth international conference on learning representations.

Ferdowsi, A., Challita, U., Saad, W., & Mandayam, N. B. (2018). Robust deep reinforcement learning for security and safety in autonomous vehicle systems. In: Proceedings of the 21st international conference on intelligent transportation systems (ITSC).

Fisac, J. F., Lugovoy, N. F., Rubies-Royo, V., Ghosh, S., & C. J. Tomlin. (2019). Bridging Hamilton-Jacobi safety analysis and reinforcement learning. In: Proceedings of the 2019 international conference on robotics and automation (ICRA) (pp. 8550–8556). doi:https://doi.org/10.1109/ICRA.2019.8794107.

Fu, J., Luo, K., & Levine, S. (2017). Learning robust rewards with adversarial inverse reinforcement learning. https://arxiv.org/pdf/1710.11248.

Fuchigami, H. Y., & Rangel, S. (2018). A survey of case studies in production scheduling: Analysis and perspectives. *Journal of Computational Science, 25*, 425–436. https://doi.org/10.1016/j.jocs.2017.06.004

Gleißner, W. (2011). Quantitative Verfahren im Risikomanagement: Risikoaggregation, Risikomaße und Performancemaße. Der Controlling-Berater, vol. 16

Golpîra, H., & Tirkolaee, E. B. (2019). Stable maintenance tasks scheduling: A bi-objective robust optimization model. *Computers and Industrial Engineering*. https://doi.org/10.1016/j.cie.2019.106007

Goren, S., & Sabuncuoglu, I. (2008). Robustness and stability measures for scheduling: Single-machine environment. *IIE Transactions, 40*, 66–83. https://doi.org/10.1080/07408170701283198

Göttlich, S., & Knapp, S. (2020). Uncertainty quantification with risk measures in production planning. *Journal of Mathematics in Industry*. https://doi.org/10.1186/s13362-020-00074-4

Hall, N. G., & Posner, M. E. (2004). Sensitivity analysis for scheduling problems. *Journal of Scheduling, 7*, 49–83. https://doi.org/10.1023/B:JOSH.0000013055.31639.f6

Henderson, P., Islam, R., Bachman, P., Pineau, J., Precup, D., & Meger, D. (2017). Deep reinforcement learning that matters. http://arxiv.org/pdf/1709.06560v3.

Hiraoka, T., Imagawa, T., Mori, T., Onishi, T., & Tsuruoka, Y. (2019). Learning robust options by conditional value at risk optimization. In: NeurIPS 2019: Advances in neural information processing systems, vol. 33

Kenton, Z., Filos, A., Evans, O., & Gal, Y. (2019). Generalizing from a few environments in safety-critical reinforcement learning. In: SafeML ICLR 2019 Workshop.

Kouvelis, P., & Yu, G. (1997). Robust discrete optimization and its applications (Nonconvex optimization and its applications, Vol. 14). Boston, MA: Springer.

Kuhnle, A., Kaiser, J.-P., Theiß, F., Stricker, N., & Lanza, G. (2020). Designing an adaptive production control system using reinforcement learning. *Journal of Intelligent Manufacturing*. https://doi.org/10.1007/s10845-020-01612-y

Lang, S., Lanzerath, N., Reggelin, T., Behrendt, F., & Müller, M. (2020). Integration of deep reinforcement learning and discrete-event simulation for real-time scheduling of a flexible job shop production. In: Proceedings Winter Simulation Conference 2020.

Lazic, N., Lu, T., Boutilier, C., Ryu, M.K., Wong, E.J., Roy, B., et al. (2018). Data center cooling using model-predictive control.

Leon, V. J., Wu, S. D., & Storer, R. H. (1994). Robustness measures and robust scheduling for job shops. *IIE Transactions, 26*(5), 32–43.

Leusin, M., Frazzon, E., Uriona Maldonado, M., Kück, M., & Freitag, M. (2018). Solving the job-shop scheduling problem in the industry 4.0 era. Technologies, 6, 107. doi:https://doi.org/10.3390/technologies6040107.

Lipton, Z. C., Azizzadenesheli, K., Kumar, A., Li, L., Gao, J., & Deng, L. (2016). Combating reinforcement learning's sisyphean curse with intrinsic fear. https://arxiv.org/pdf/1611.01211.

Luo, S. (2020). Dynamic scheduling for flexible job shop with new job insertions by deep reinforcement learning. *Applied Soft Computing, 91*, 106208. https://doi.org/10.1016/j.asoc.2020.106208

Mankowitz, D. J., Tamar, A., & Mannor, S. (2016). Situational awareness by risk-conscious skills. https://arxiv.org/pdf/1610.02847.

Mankowitz, D. J., Mann, T. A., Bacon, P., Precup, D., & Mannor, S. (2018). Learning robust options (wasserste). https://arxiv.org/abs/1802.03236.

Martí, R., Pardalos, P. M., & Resende, M. G. C. (2018). *Handbook of heuristics (Springer reference)*. Springer.

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., et al. (2013). Playing Atari with deep reinforcement learning. https://arxiv.org/pdf/1312.5602.

Osband, I., Doron, Y., Hessel, M., Aslanides, J., Sezener, E., Saraiva, A., et al. (2020). Behaviour suite for reinforcement learning. In; International Conference on Learning Representations.

Osogami, T. (2012). Robustness and risk-sensitivity in Markov decision processes. In: NIPS'12: Proceedings of the 25th international conference on neural information processing systems, vol. 1, pp. 233–241.

Pinedo, M. (2016). Scheduling: Theory, algorithms, and systems/by Michael L. Pinedo. Cham: Springer.

Pinto, L., Davidson, J., Sukthankar, R., & Gupta, A. (2017). Robust adversarial reinforcement learning. In: International Conference on Machine Learning, pp. 2817–2826.

Policella, N., Cesta, A., Oddi, A., & Smith, S. (2007). From precedence constraint posting to partial order schedules: A CSP approach to Robust Scheduling. *AI Communications, 20*, 163–180.

Prashanth, L. A. (2014). Policy gradients for CVaR-constrained MDPs. In: P. Auer (Ed.), Cham, 2014 (pp. 155–169, LNCS sublibrary. SL 7, Artificial intelligence, Vol. 8776). Cham: Springer.

Rahmani, D., & Heydari, M. (2014). Robust and stable flow shop scheduling with unexpected arrivals of new jobs and uncertain processing times. *Journal of Manufacturing Systems, 33*, 84–92. https://doi.org/10.1016/j.jmsy.2013.03.004

Rinciog, A., Mieth, C., Scheikl, P. M., & Meyer, A. (2020). Sheet-metal production scheduling using AlphaGo zero. doi:https://doi.org/10.15488/9676.

Ruszczyński, A. (2010). Risk-averse dynamic programming for Markov decision processes. *Mathematical Programming, 125*, 235–261. https://doi.org/10.1007/s10107-010-0393-3

Schulman, J., Levine, S., Moritz, P., Jordan, M. I., & Abbeel, P. (2015). Trust region policy optimization. In: ICML'15: Proceedings of the 32nd international conference on international conference on machine learning, vol. 37, pp. 1887–1897.

Shen, X.-N., Han, Y., & Fu, J.-Z. (2017). Robustness measures and robust scheduling for multi-objective stochastic flexible job shop scheduling problems. *Soft Computing, 21*, 6531–6554. https://doi.org/10.1007/s00500-016-2245-4

Silver, D., Hubert, T., Schrittwieser, J., Antonoglou, I., Lai, M., Guez, A., et al. (2018). Mastering Chess and Shogi by self-play with a general reinforcement learning algorithm. *Science, 6419*, 1140–1144.

Sotskov, Y., Sotskova, N. Y., & Werner, F. (1997). Stability of an optimal schedule in a job shop. *Omega, 25*, 397–414. https://doi.org/10.1016/S0305-0483(97)00012-1

Stooke, A., Achiam, J., & Abbeel, P. (2020). Responsive safety in reinforcement learning by PID lagrangian methods. In: International Conference on Machine Learning, pp. 9133–9143.

Sutton, R. S., McAllester, D., Singh, S., & Mansour, Y. (1999). Policy gradient methods for reinforcement learning with function approximation. Advances in Neural Information Processing Systems, p. 12.

Sutton, R. S., & Barto, A. G. (2018). *Reinforcement learning: An introduction*. MIT Press.

Tamar, A., Chow, Y., Ghavamzadeh, M., & Mannor, S. (2015a). Policy gradient for coherent risk measures. In: NIPS'15: Proceedings of the 28th international conference on neural processing systems, pp. 1468–1476.

Tamar, A., Glassner, Y., & Mannor, S. (2015b). Optimizing the CVaR via sampling. In: AAAI'15: Proceedings of the twenty-ninth AAAI conference on artificial intelligence, pp. 2993–2999.

Tessler, C., Mankowitz, D. J., & Mannor, S. (2018). Reward constrained policy optimization. https://arxiv.org/pdf/1805.11074.

Tessler, C., Efroni, Y., & Mannor, S. (2019). Action robust reinforcement learning and applications in continuous control. In: Proceedings of the 36th international conference on machine learning, vol. 97, pp. 6215–6224.

The, Y., Bapst, V., Czarnecki, W.M., Quan, J., Kirkpatrick, J., Hadsell, R., et al. (2017). Distral: Robust multitask reinforcement learning. In: NIPS'17: Proceedings of the 31st international conference on neural information processing systems, pp. 4496–4506.

Tobin, J., Fong, R., Ray, A., Schneider, J., Zaremba, W., & Abbeel, P. (2017). Domain randomization for transferring deep neural networks from simulation to the real world. In: Proceedings of the 2017 IEEE/RSJ international conference on intelligent robots and systems (IROS).

Vieira, G. E., Herrmann, J. W., & Lin, E. (2003). Rescheduling manufacturing systems: A framework of strategies, policies, and methods. *Journal of Scheduling, 6*, 39–62. https://doi.org/10.1023/A:1022235519958

Vinyals, O., Babuschkin, I., Czarnecki, W. M., Mathieu, M., Dudzik, A., Chung, J., et al. (2019). Grandmaster level in StarCraft II using multi-agent reinforcement learning. *Nature, 575*, 350–354. https://doi.org/10.1038/s41586-019-1724-z

Wabersich, K. P., & Zeilinger, M. N. (2018). Safe exploration of nonlinear dynamical systems: A predictive safety filter for reinforcement learning. https://arxiv.org/pdf/1812.05506.

Waschneck, B., Reichstaller, A., Belzner, L., Altenmüller, T., Bauernhansl, T., Knapp, A., et al. (2018). Optimization of global production scheduling with deep reinforcement learning. *Procedia CIRP, 72*, 1264–1269. https://doi.org/10.1016/j.procir.2018.03.212

Wiesemann, W., Kuhn, D., & Rustem, B. (2013). Robust Markov decision processes. *Mathematics of Operations Research, 38*, 153–183. https://doi.org/10.1287/moor.1120.0566

Witty, S., Lee, J. K., Tosch, E., Atrey, A., Littman, M., & Jensen, D. (2018). Measuring and characterizing generalization in deep reinforcement learning. https://arxiv.org/pdf/1812.02868.

Wu, C. W., Brown, K. N., & Beck, J. C. (2009). Scheduling with uncertain durations: Modeling -robust scheduling with constraints. *Computers and Operations Research, 36*, 2348–2356. https://doi.org/10.1016/j.cor.2008.08.008

Yang, T.-Y., Rosca, J., Narasimhan, K., & Ramadge, P. J. (2020). Projection-based constrained policy optimization. https://arxiv.org/pdf/2010.03152.

Yehuda, G., Gabel, M., & Schuster, A. (2020). It's not what machines can learn, it's what we cannot teach. In: International conference on machine learning, pp. 10831–10841.

Yoshida, Y. (2019). Risk-sensitive markov decision under risk constraints with coherent risk measures. In V. Torra, Y. Narukawa, G. Pasi, & M. Viviani (Eds.), Cham, 2019 (pp. 29–40, LNCS SublibraryL SL7 - Artificial Intelligence, Vol. 11676). Cham, Switzerland: Springer.

Zhang, J., Bedi, A. S., Wang, M., & Koppel, A. (2020). Cautious reinforcement learning via distributional risk in the dual domain. https://arxiv.org/pdf/2002.12475.

Zhu, W., & Wang, B. (2017). New robust single machine scheduling to hedge against processing time uncertainty. In: Proceedings of the 2017 29th Chinese Control And Decision Conference (CCDC) (pp. 2418–2423). doi:https://doi.org/10.1109/CCDC.2017.7978920.