University of Applied Sciences and Arts Northwestern Switzerland
School of Engineering

Solving the Job-Shop Scheduling Problem with Reinforcement Learning

University of Applied Sciences and Arts FHNW

School of Engineering

September 2020

Keywords: Reinforcement Learning,
Optimization,
Scheduling,
Shop-Floor Scheduling,
Job-Shop Scheduling Problem

Author: David Schlebusch, david.schlebusch@students.fhnw.ch

Professors: Adrian Specker, adrian.specker@fhnw.ch
Raoul Waldburger, raoul.waldburger@fhnw.ch

Editor: Roger Siegenthaler, roger.siegenthaler@students.fhnw.ch

Date: Thursday, September 24, 2020

Version: 1

Abstract:
This literature research explores the research done into solving the job-shop scheduling problem with linear optimization and reinforcement learning methods. It looks at a timeline of the problem and how methods to solve it have changed over time. The research should give an understanding of the problem and explore possible solutions. For that, an extensive search for papers was done on Scopus, a research paper database. 27 promising papers were selected, rated, and categorized to facilitate a quick understanding of the problem and show potential research gaps. Two such gaps were found; Firstly, little research has been done on how reinforcement learning can be improved by implementing data or process mining strategies to further improve accuracy. Secondly, no research was found connecting reinforcement learning with a takt schedule, like the one proposed by the SRS project. The gathered papers give an extensive overview of the problem and demonstrate a multitude of solutions to the job-shop scheduling problem, which are discussed in detail in the results of this report. This should provide all the necessary information to be able to implement one's own version of reinforcement learning for the job-shop scheduling problem.

Table of Contents

1 Introduction

This literature research is done to answer a problem created by the novel solution to the job-shop scheduling problem (JSSP) with a takt proposed by Walburger as a smart scheduling recommender system (SRS). SRS has the goal to reduce the makespan of jobs by introducing a takt so that each step of the jobs can be done in one shift (time-unit) and the next step of the job in the following shift and so on. This should reduce the makespan of the job to exactly the number of steps in shifts, which simplifies planning and helps keep the shop-floor footprint low since no large temporary stores should be needed. Furthermore, it should also guarantee delivery on time, since the makespan per product is now fixed to a certain number of shifts.
This approach prompts the question of how job-shop scheduling is solved at the moment and what research has been done previously, in particular with a focus on a production takt.

1.1 Structure of this Document

This document is structured as follows. After this introduction, the JSSP is described and the problem is defined. In the following two chapters different optimization and reinforcement learning (RL) methods, relevant to the problem, are elaborated to lay a foundation for the methods analyzed in the literature research. The fourth chapter goes into the literature stating the research question, the methodological approach, and the found papers categorized into two lists, closing with a discussion of the found results in chapter 6. Lastly, this report concludes with a summary of the work done and a future lookout.

2 The Job-Shop Scheduling Problem

Job-Shop scheduling can get very complex with the growth of the production facility. Many factors play into the prioritization of a job to guarantee the delivery of a product on time. A prioritization based purely on the due date is an easy and naïve approach, which can be managed by most companies quite well. However, the utilization of the machinery and the processing time leaves room for optimization.
While in theory the JSSP could be solved by a linear algorithm, the many factors playing into the successful optimization of the problem makes the processing time on a normal computer infeasible [1], since linear programming optimization belongs to the NP-hard problems [2]. In practice, a heuristic approach is most commonly used.

2.1 Model of the Job-Shop Scheduling Problem

In general, the job shop scheduling problem follows the set of rules below:

(1) there exists a set of orders
(2) an order has an ordered set of jobs
(3) an order has a due date
(4) there exists a set of machines
(5) a job can be performed on exactly one machine
(6) one machine can only perform one job at a time
(7) the order of jobs within an order must be kept
(8) the time for each job to process is known

The manufacturing works from a backlog of orders (1), where each order has a set of ordered jobs (2) to perform in order (7). Each task must be done before the due date (3). A job can usually only be performed on one machine (5), if this is not the case and this possibility should be modeled, the model can be adapted to the manufacturing process. In the regular case a machine can do only one job at the time (6) should that not be the case it might be needed to split the machine into virtual ones for the model. To allow the algorithm to find a solution to the scheduling problem, the machining time for each job needs to be known (8).

2.2 The complexity of the JSSP

The JSSP is highly scalable in complexity and is directly dependent on the structure of the factory and the number of jobs it can process and the variations of the jobs themselves. The solvability of the problem becomes quickly very hard, the 10 x 10 problem state by Muth et al. [3] remained unsolvable for 20 years, though it was stated in 1963. At the time researchers tried to solve the problem with linear

programming and branch and bound algorithms. As of 2016, it was still impossible to solve the problem of size 20 x 15 as a mixed-integer program in under an hour [4].

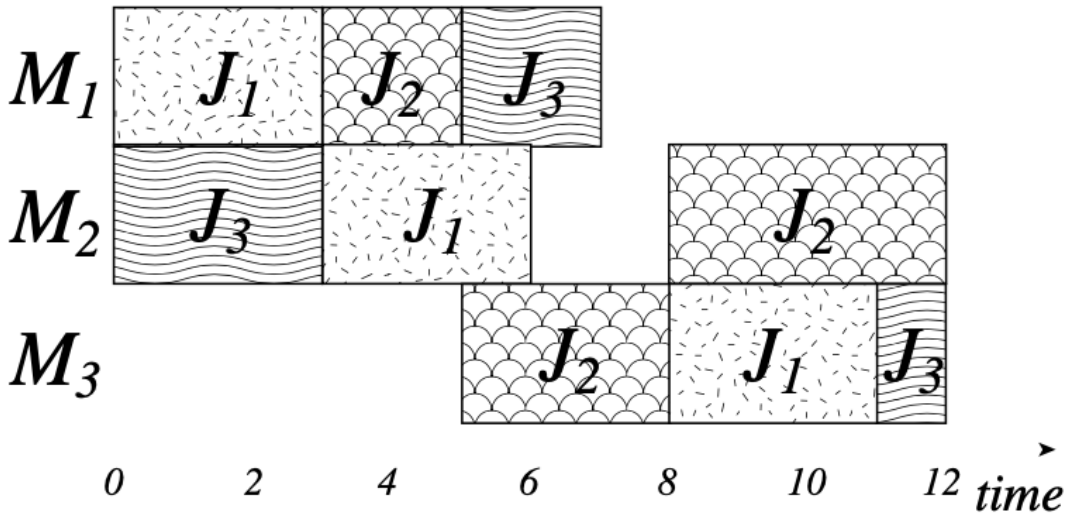

Figure 1: A Gantt-Chart representation of a solution for a 3 × 3 problem by Yamada et al. [5].

3 Optimization

Mathematical optimization or mathematical programming belongs to a highly studied field with a lot of disciplines taking interest in it from computer sciences to engineering and economics. Optimization has its goal to find the optimal solution to a mathematically stated problem which is often referred to as minimization or maximization of the problem. Optimization problems which are NP-complete often must use heuristic optimization methods to find solutions since they normally perform better than brute force, since brute force search can take a lot of time. In general optimization strategies can be classified into three categories. Finite optimization techniques, like the simplex algorithm, perform computation until the global maximum of the function is found. These calculations are highly computation intensive and take a lot of time. Iterative methods only try to find a good enough or approximated solution to the problem. One example is gradient descent which is used in machine learning (ML). The problem of these iterative methods is that they choose a starting point at random and then explore it into the next minima. In complex environments, they seldom explore the whole solution space. For iterative methods to work a classification problem is normally needed. With heuristic methods, intelligent guessing based on stochastics is used. Compared to the other two methods, heuristics do not guarantee a mathematical solution to the problem, but in practice, the found solution or state is normally a good enough approximation.

3.1 Finite Methods

Finite optimization methods like linear programming are guaranteed to find the mathematical correct solution to the given problem function. The problem with them is that it can take a lot of time to calculate and can quickly become infeasible to solve on average consumer machines or even supercomputers. Problems that are solved mathematical should not have any time pressing matter and should not be done too often since renting powerful hardware can be quite expensive. To help solve linear programming problems very sophisticated software exists, which can speed up calculations significantly compared to a naïve approach learned in a calculus class. Using such software to solve the problem should be considered every time. One of these programs called Gurobi is available as closed-source and is known as one of the market leaders in the field.

3.2 Iterative Methods

Iterative methods pick a starting point in the solution space at random and then start exploring the surrounding, generally speaking, the algorithm looks at its surrounding

and picks the direction which brings it closer to the solution. The problem with these algorithms is that if they land at a local minimum they will stay there since no move can give them a better solution. The calculations to find the next direction step can be very expensive to calculate which is why they normally only performed once per problem and the iterations stop when one minimum is found. When working on classification problems these methods are a good match. In ML the problem of finding only a shallow minimum for a single problem is averaged out by the sheer number of problems given to the model.

3.3 Heuristic Methods

Compared to finite algorithms and convergent iterative methods, heuristic methods are not guaranteed to find a mathematical solution. Still, heuristics can be a very powerful tool to get approximate solutions to a problem. Some of the better known are genetic algorithms (GA), particle swarm optimization (PSO), tabu search (TS), and variable neighborhood search (VNS).

3.3.1 Genetic Algorithms

GAs take their inspiration from nature trying to project evolution on to an agent. It works by creating a set of agents with random parameters (analogous to DNA made from chromosomes and gens) and tests their performance on the value function. When all calculations are done, the ones who performed best are taken for a new iteration (generation). In the new generation, mutations are introduced into the parameters and the size is restocked to its original and a new life cycle begins. To further improve the algorithm a lot of different methods of DNA splicing and transferring is done to refine the best genetic code for the given problem. Research trying to combine RL with GAs has been done [6].

3.3.2 Particle Swarm Optimization

PSO copies its strategy from nature as well, trying to simulate a flock of birds or school of fish. The idea being that every individual particle knows the optimum in its immediate neighborhood and is drawn to it but at the same time, it is communicating with all other particles in the swarm, which are trying to pull the particle to a better optimum. This method guarantees that in theory the complete solution environment is explored.

3.3.3 Tabu Search

TS is a local or neighborhood search algorithm that uses a memory list to avoid cyclic behavior and in comparison, to basic local search a worsening the target value is allowed as well. Figure 2 shows the implementation of TS in pseudocode.

```
1 sBest ← s0
2 bestCandidate ← s0
3 tabuList ← []
4 tabuList.push(s0)
5 while (not stoppingCondition())
6     sNeighborhood ← getNeighbors(bestCandidate)
7     bestCandidate ← sNeighborhood[0]
8     for (sCandidate in sNeighborhood)
9         if ( (not tabuList.contains(sCandidate)) and (fitness(sCandidate) > fitness(bestCandidate)) )
10            bestCandidate ← sCandidate
11        end
12    end
13    if (fitness(bestCandidate) > fitness(sBest))
14        sBest ← bestCandidate
15    end
16    tabuList.push(bestCandidate)
17    if (tabuList.size > maxTabuSize)
18        tabuList.removeFirst()
19    end
20 end
21 return sBest
```

Figure 2: Pseudocode for Tabu Search form Wikipedia [7].

3.3.4 Variable Neighborhood Search

Similar to TS the basis of VNS uses local search to find a local optimum. When one is found a transfer of the search is made to a new neighborhood. A neighborhood is defined from intuition and informed guessing by the implementer of the algorithm. There are various extensions of the VNS algorithm and its application reaches from scheduling to routing and location problems.

4 Reinforcement Learning

Reinforcement Learning (RL) is an agent-based optimization strategy using probabilities in Markov decision chains to decide the next step of the agent. In the learning process, the agent has to build a balance between the exploration of new solutions while not forgetting already found good solutions, for that normally the ϵ-Greedy algorithm is used. Unlike other ML algorithms, no labeled data is need for RL, since only a good enough solution is needed and a trade-off between exploitation and exploration is made. With that, RL belongs to the unsupervised learning category. The general goal of an RL agent is to traverse an environment as efficiently as possible while conforming to a set of rules given by or added to the environment.

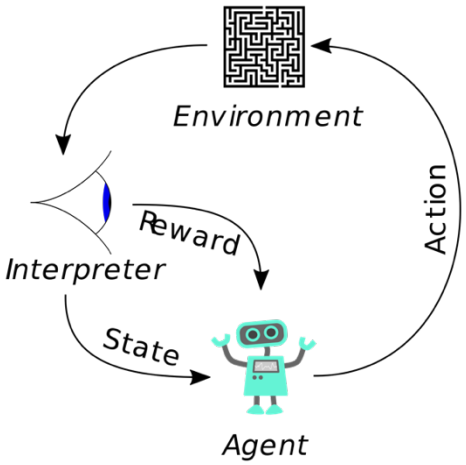

Figure 3: The typical framing of an RL scenario by Megajuice [48].

In comparison to greedy optimization strategies, RL agents are allowed to do steps with negative rewards, if at the end of an episode a net gain is achieved. With that RL works very well for games like Go where planning ahead can give great rewards [8] as well as many more applications mentioned by Li [9].

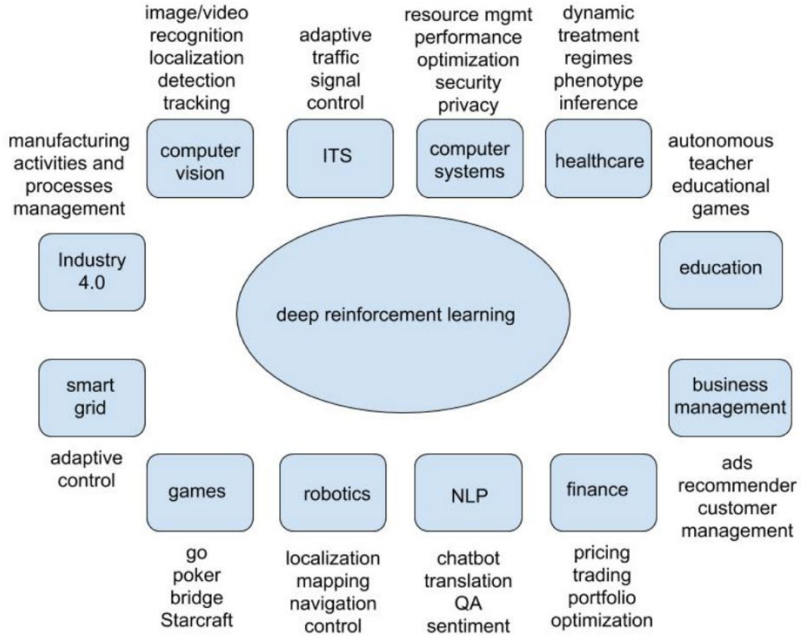

Figure 4: Deep RL Applications by Li [9].

4.1 Fundamentals

4.1.1 Markov Decision Process

The MDP is an extension of Markov chains. MDP provides a framework for modeling decision making and is very useful for optimization problems. MDPs lay the ground for most RL algorithms as well.
An MDP is described as a 4-tuple (S, A, P_a, R_a) [10] [10], where

- S is a set of state space,
- A is a set of actions called the action space (alternatively, $A_s$is the set of actions available from state s),
- $P_a(s, s') = \Pr(s_{t+1} = s' | s_t = s, a_t = a)$ is the probability that action a in state time t will lead to a state s' at time $t + 1$,
- $R_a(s, s')$ is the immediate reward (or expected immediate reward) received after transitioning from state s to state s', due to action a

4.1.2 Model-Based versus Model-Free Learning

When building an RL algorithm for a problem the question comes if a model-based or model-free learning algorithm can or should be used. In general if the model of the environment, in other words, the "rules", are known a model-based approach can be used to train the algorithms with known state transitions, for example, the chess computer AlphaZero [11]. In some cases, the RL algorithm can learn the model as well like in World Models [12]. Model-free algorithms like DQN are optimally used when state transitions are neither known nor can be learned directly other than by exploring the environment.

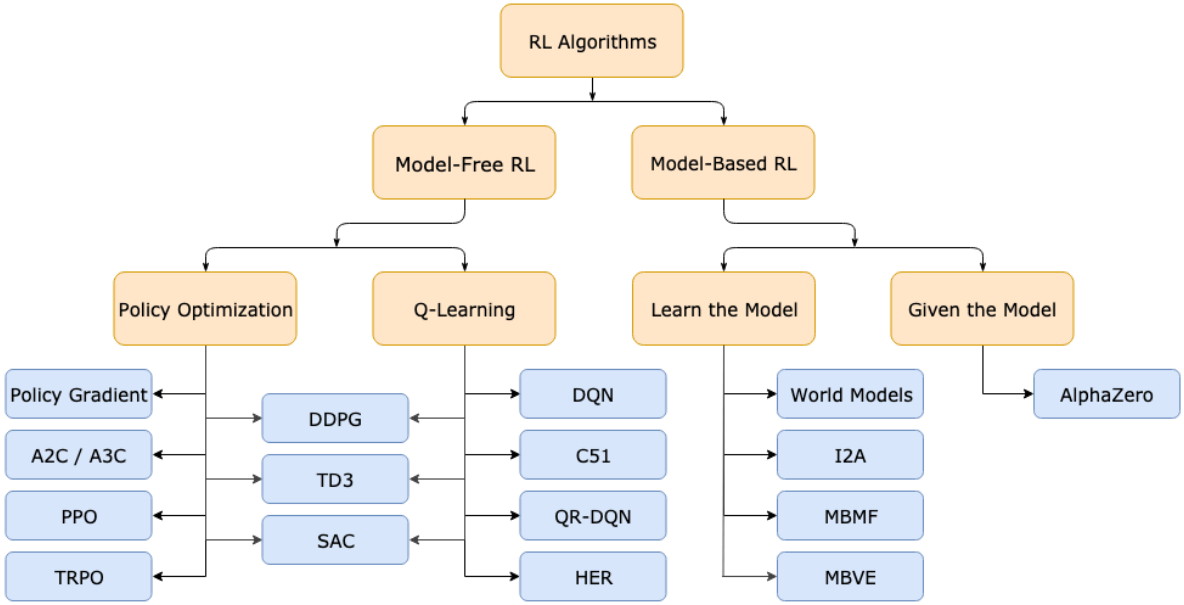

Figure 5: A non-exhaustive, but useful taxonomy of algorithms in modern RL by OpenAi [13].

4.1.3 Online and Offline Learning

Offline learning is the approach of pretraining a model with a fixed data set and environment and then moving it over to production. With its fixed policies and decisions, this reduces the expenses needed to be made to implement the model in production, since the results of the trained model can be exported as a decision matrix and applied to new incoming environments. The behavior of the model stays consistent. With this approach, the model cannot react to environmental changes, which were not in the training set previously.
Online learning means a continuous updating of policies while the model is in production. This makes the model highly adaptive and able to react to changes in the environment, which were not observed previously. Since the environment changes cannot be predicted the changes to the model might not be desired by the creator. Thus online learning can be prone to seasonal overtraining since no shuffling of the data is done, it learns a certain sequence of events, instead of handling every event uniquely [14].

4.2 ϵ-Greedy

The epsilon greedy algorithm offers a solution to the multi-armed bandit problem [15]. The problem with learning algorithms is that they tend to prefer to choose the actions which are known to offer the greatest rewards, thus strongly limits the exploration of the solution space. To solve the exploitation versus exploration problem ϵ-Greedy was developed. It states that a certain percentage of actions need to be completely random, to allow the exploration of new solutions. If fast learning is desired an epsilon value of 0.1 is recommended, but 0.01 will give a higher accuracy overall, another technique is to change epsilon value over time [16]. Though there are other exploitation-exploration strategies when it comes to RL, ϵ-Greedy is one of the most commonly found and very essential for many algorithms to work.

4.3 Temporal Difference Learning

Temporal difference learning (TD) is an RL method that regularly updates the value function by predicting the final outcome iteratively, in comparison to Monte Carlo methods, which wait for the whole process to finish to update the function. Instead of waiting until the end state is reached, each step updates the model to get a better prediction for the end state. TD-lambda (TD(λ)) adds the possibility of a decay in the reward, so more or less reward can be given for more distant states. TD(λ) was first introduced as TD-Gammon, learning the game backgammon in 1992 and was nearly as good as top human players [17].
Below describes the algorithm for TD-Gammon [18]:

where:

$w_{t+1} - w_t$ is the amount to change the weight from its value on the previous turn.

$Y_{t+1} - Y_t$ is the difference between the current and previous turn's board evaluations.

α is the "learning rate" parameter

λ is a parameter that affects how much the present difference in board evaluations should feedback to previous estimates. λ = 0 makes the program correct only the previous turn's estimate; λ = 1 makes the program attempt to correct the estimates on all previous turns; values of λ between 0 and 1 specify different rates at which the importance of older estimates should "decay" with time.

$\nabla_w Y_k$ is the gradient of neural-network output with respect to weights: that is, how much changing the weight affects the output.

4.4 Q-Learning

The Q-Learning algorithm was developed to find the best solution for a finite MDP given enough computational time. The Q in Q-Learning stands for the quality of the found solution. Q-Learning works with a table listing all state combinations. After each action, the Q is updated. Q-Learning belongs to the model-free algorithms. The state transitions are dependent on the chosen action and the previous state. Q-Learning was first introduced by Watkins et al. [19] in 1989.

4.5 Deep Q Networks

Deep Q Networks (DQN) was created by combining convolutional networks, known from ML, with Q-tables. This enables the algorithm to have a kind of receptive field, by enabling the model to give weights to the perceived inputs. The first DQN was developed by the researchers at Deepmind in 2013 [20]. DQN became famous when a trained model was able to beat the best-known human player in Go in 2015 [8]. With its success, DQN are now considered one of the best solutions when a self-learning algorithm is needed. Its ability to look ahead for future rewards and take previous actions into account in a single computation cycle makes DQN very fast in learning, while still keeping the benefits from evolution-based algorithms because it does not have to throw away failed mutations.

4.5.1 Extensions of the DQN

While the benefits of only having one actor running at a time increase learning speed and lowers computation time, it makes overtraining of the model easy. This is why an

extension of the DQN model was made called asynchronous advantage actor-critic (A3C) [21]. The benefits of the new approach are that that multiple agents train in their own environment separately and critique the actions taken by their colleagues compared to what they learned.

5 Literature Review

Research into the job-shop or shop-floor scheduling problem dates back into the seventies [22]. The proposed solutions to the problem vary strongly in complexity and approach. Besides, reality often differs heavily from the research.
In general, most shop-floor scheduling is based on a prioritization of incoming jobs into a stack, which will be worked through from the top, while new jobs are added to the bottom. The ordering is often split into at least two levels, a rough overlooking level, and a finer more detailed level. Once a job is on the stack no rescheduling is normally done. In practice, the top level of the job pool is selected by the due date, while the fine level is selected by computer-aided calculations and the intuition of the shop-floor manager. The manager often uses the help of graphical tools, mostly in the form of Gant charts, to order the selected jobs, to maximize the utilization of his resources.
This literature research looks into what RL methods are used for job-shop prioritization, how they are implemented what the benefits and challenges are, as well as how they could be implemented with a takt based scheduling system proposed by the SRS project.

5.1 Approach

The preliminary literature search was done with keywords in the Scopus paper database. The search terms that were used can be found in Table 1. To decide if the paper was on the topic of this literature research a quick readthrough of the abstract and in some cases the conclusion was done. If deemed appropriate the paper was saved for further reading. When selecting papers, a focus was put on uniqueness and publishing date to gather a wide variety of research. In addition to papers found on Scopus, interesting citations of the found papers as well as recommendations from fellow scholars were added to the pool of relevant research.

5.2 Research Questions

The following research question was formulated during the project:

What research has been done solving the job-shop scheduling problem with reinforcement learning?

5.3 Search Terms

Table 1: Number of search results for used keywords.

Key Words	Database	Articles
reinforcement AND learning	Scopus	50017
machine AND learning AND scheduling	Scopus	3767
reinforcement AND learning AND planning	Scopus	2381
reinforcement AND learning AND scheduling	Scopus	1347
takt	Scopus	563
reinforcement AND learning AND data AND mining	Scopus	553
deep AND reinforcement AND learning AND scheduling	Scopus	287
reinforcement AND learning AND job AND scheduling	Scopus	215
reinforcement AND learning AND job-shop	Scopus	113
reinforcement AND learning AND job-shop AND scheduling	Scopus	109
scheduling AND takt	Scopus	57
q[1] AND job-shop AND scheduling	Scopus	45
process AND mining AND job-shop	Scopus	37
flexible AND manufacturing AND systems AND reinforcement AND learning	Scopus	34
reinforcement AND learning AND data AND mining AND scheduling	Scopus	25

[1] The search term "Q" comes from "Deep Q Learning" which comes from "Q Learning" which is a reinforcement learning algorithm introduced by Watkins 1989 [19] and later refined into "Deep Q Learning" (DQN) by people at Deepmind [20].

reinforcement AND learning AND shop-floor	Scopus	17
learning AND takt	Scopus	13
reinforcement AND learning AND process AND mining AND scheduling	Scopus	12
job-shop AND takt	Scopus	10
reinforcement AND learning AND data AND mining AND job-shop	Scopus	2
machine AND learning AND takt	Scopus	1
reinforcement AND learning AND process AND mining AND shop-floor	Scopus	1
reinforcement AND learning AND process AND mining AND job-shop	Scopus	0
reinforcement AND learning AND takt	Scopus	0

The keyword search on Scopus resulted in a lot of hits when searching for reinforcement and learning even in combination with scheduling there are quite a few results. RL in the combination of job scheduling, job-shop, or shop-floor starts to thin out the results and brings the results in line with the research question. The table above shows that there is some research done into takt, but as of today not in combination with RL algorithms. The same can be said for process mining in combination with RL which yields only a few results.

5.4 Classification

The found papers were rated for relevance and categorized.

5.4.1 Relevance

To better navigate the number of selected papers a relevance matrix was created, which can be found in Table 2. The following criteria were marked from 0 to 5:

- Reinforcement Learning
- Scheduling
- Job-Shop
- Machine Learning
- Heuristic Algorithms
- Data / Process Mining
- Takt
- Relevance

Reinforcement Learning:
Describes the papers' relevance for RL using TD as a basis and the variants thereof. This category helps to distinguish between RL and genetic or swarm algorithms as well as papers that only applied other ML methods.
Scheduling:
To rate the paper according to scheduling problems, since there are scheduling problems, which are not part of the JSSP but more in general or software focused. Some papers have very low relevance to a scheduling problem in general since they only use it as an initial position.
Job-Shop:
As mentioned above some papers focus on scheduling without the JSSP in focus. This criterion tries to show the significance of the paper for the JSSP.
Machine Learning:
Indicates the significance of ML in the paper. Though RL is a subcategory of ML it was not counted in this column, so a paper is either focused on RL or other ML methods.
Heuristic Algorithms:
All papers which proposed heuristic methods like GA and PSO are rated in this category. Since heuristics are an ample part of the JSSP and show how the problem was solved before the introduction of RL algorithms.
Data / Process Mining:
This category shows if it was used for the approach in the paper.
Takt:
The relevance of it in the paper.
Relevance:
To indicate the overall significance of the work in relation to the stated research question a relevance score was given, on a subjective basis by the author.

5.4.2 Categorization

Next to the matrix, each paper was categorized into one of the following main topics:

- Data Mining
- Reinforcement Scheduling
- Swarm Scheduling
- Takt Planning
- Predictive Scheduling
- Genetic Scheduling
- Simulation Scheduling
- Repair Scheduling
- Scheduling Graph Modelling
- Linear Scheduling

This should help gain a fast understanding of the topic of the paper and help the readers quickly find the papers they are looking for.

5.5 Relevance

Table 2: Paper relevance matrix.

Title	Reinforcement Learning	Scheduling	Job-Shop	Machine Learning	Heuristic Algorithms	Data / Process Mining	Takt	Relevance
A Branch-and-Bound Algorithm for the Continuous- Process Job-Shop Scheduling Problem	0	5	5	0	3	0	0	1
A data mining approach for population-based methods to solve the JSSP	0	2	4	0	3	5	0	2
A Reinforcement Learning Approach to Job-shop Scheduling	5	4	4	0	0	0	0	5

A reinforcement learning approach to parameter estimation in dynamic job shop scheduling	5	4	4	0	0	0	0	4
A review of machine learning in dynamic scheduling of flexible manufacturing systems	0	1	1	1	1	0	0	1
A Review of Machine Learning in Scheduling	2	2	2	2	0	0	0	2
An improved particle swarm optimization with decline disturbance index (DDPSO) for multi-objective job-shop scheduling problem	0	2	2	0	5	0	0	1
Can a takt plan ever survive beyond the first contact with the trades on-site?	0	0	0	0	0	0	4	1
Classical Planning in MDP Heuristics: with a Little Help from Generalization	0	4	0	0	5	0	0	0
Data Centers Job Scheduling with Deep Reinforcement Learning	5	3	0	0	0	0	0	1
Deep Reinforcement Learning for Semiconductor Production Scheduling	5	4	4	0	0	0	0	5
Design, Implementation and Evaluation of Reinforcement Learning for an Adaptive Order Dispatching in Job Shop Manufacturing Systems	5	5	5	0	0	0	0	5
Dynamic scheduling for flexible job shop with new job insertions by deep reinforcement learning	5	4	4	0	0	0	0	5
Genetic algorithm in flexible work shop scheduling based on multi-objective optimization	0	1	1	0	3	0	0	0

Hybrid Deep Neural Network Scheduler for Job-Shop Problem Based on Convolution Two-Dimensional Transformation	0	3	3	4	0	0	0	2
Local Search Genetic Algorithms for the Job Shop Scheduling Problem	0	4	4	0	5	0	0	3
MINERVA: A Reinforcement Learning-based Technique for Optimal Scheduling and Bottleneck Detection in Distributed Factory Operations	5	4	4	0	0	0	0	5
Minimizing total energy cost and tardiness penalty for a scheduling-layout problem in a flexible job shop system, A comparison of four metaheuristic algorithms	0	4	2	0	4	0	0	1
Multiple Resource Management and Burst Time Prediction using Deep Reinforcement Learning	4	3	0	0	0	0	0	2
Optimization of global production scheduling with deep reinforcement learning	5	5	5	0	0	0	0	5
Optimization of setup times in the furniture industry	0	3	3	0	0	0	0	0
Petri-net-based dynamic scheduling of flexible manufacturing system via deep reinforcement learning with graph convolutional network	5	4	5	0	0	0	0	5
Real-time scheduling for a smart factory using a reinforcement learning approach	5	4	4	0	2	0	0	5
Research on Open-pit Mine Vehicle Scheduling Problem with Approximate Dynamic Programming	4	4	0	0	0	0	0	0
Scheduling and Rescheduling with Iterative Repair	2	5	0	0	1	0	0	3

Solving batch process scheduling/planning tasks using reinforcement learning	3	4	0	0	0	0	0	2
The disjunctive graph machine representation of the job shop scheduling problem	0	5	3	0	0	0	0	1

Table 2 shows the relevance of the papers to the above-defined categories. Overall, quite a few papers have been found focusing on the JSSP and RL, some of them strongly satisfying the research question. A few papers were found showing the evolution of solutions to the JSSP from linear programming to heuristic and give valuable insight into how the problem should be modeled and what the problems and pitfalls are. A few papers were on a more precise level no longer relevant to the research question, these have a low mark in overall relevance. The matrix shows as well that the fields of process and data mining in combination with RL for the JSSP has barely been explored. No linkage of RL and a takt schedule could be found in the research, which leaves the topic open for further exploration.

5.6 Overview of Found Papers

Table 3: Detailed overview of 27 found papers with a summary.

Category	Title	Author	Year	Keywords	Summary
Linear Scheduling	A Branch-and-Bound Algorithm for the Continuous- Process Job-Shop Scheduling Problem	George Bozoki, Jean-Paul Richard	1970	– -	One of the earliest papers found on the JSSP. Solving the problem with a branch and bound algorithm. [22]
Data Mining	A data mining approach for population-based methods to solve the JSSP	Mohammad Mahdi Nasiri, Sadegh Salesi, Ali Rahbari, Navid Salmanzadeh Meydani, Mojtaba Abdollai	2018	– Scheduling – Job shop – Data mining – Population generation – Particle swarm optimization – Genetic algorithm	Data mining-based approach to generate an improved initial population for population-based heuristics/meta- heuristics solving the JSSP. Using genetic and swarm algorithms to validate the rules gathered from the mined data. The initial mined population outperformed random populations. The paper suggests applying the mining technique to other scheduling problems. [23]

Reinforcement Scheduling	A Reinforcement Learning Approach to Job-shop Scheduling	Wei Zhang, Thomas G. Dietterich	1995	– -	RL for the job shop scheduling problem replacing Zweben's iterative repair method with a TD algorithm. The results showed that in speed TD(λ) outperforms Zweben's iterative repair method but given enough time Zweben's iterative repair method will find the optimal solution and it will be similar to the TD(λ) solution. [24]
Reinforcement Scheduling	A reinforcement learning approach to parameter estimation in dynamic job shop scheduling	Jamal Shahrabi, Mohammad Amin Adibi, Masoud Mahootchi	2017	– Reinforcement learning – Q–factor – Dynamic job shop scheduling – Variable neighborhood search	Recommends an online Q-factor algorithm with VNS for DJSS to be able to react to breakdowns and new jobs dynamically. ε-greedy is adjusted to the state of the shop floor. [25]
Predictive Scheduling	A review of machine learning in dynamic scheduling of flexible manufacturing systems	Paolo Priore, David de la Fuente, Alberto Gomez, Javier Puente	2001	– Discrete Simulation – Dispatching Rules – Dynamic Scheduling – Flexible Manufacturing Systems – Machine Learning	A review of literature on dynamic scheduling of FMSs using ML and classification thereof. [26]
Predictive Scheduling	A Review of Machine Learning in Scheduling	Haldun Aytug, Siddhartha Bhattacharyya, Gary J. Koehler, Jane L. Snowdon	1994	– -	Gives a clear definition of the scheduling problem. Discusses existing methods and explains the need for ML for the JSSP. [27]

Swarm Scheduling	An improved particle swarm optimization with decline disturbance index (DDPSO) for multi-objective job-shop scheduling problem	Fuqing Zhao, Jianxin Tang, Junbiao Wang, Jonrinaldi	2013	– particle swarm optimization – expanded job shop scheduling problem – multi-objective job shop scheduling problem – decline disturbance – adaptive meta-Lamarckian strategy	Discussion of an improved PSO method. A very good definition of the JSSP with all constraints. [28]
Takt Planning	Can a takt plan ever survive beyond the first contact with the trades on-site?	Otto Alhava, Vili Rinne, Enni Laine, Lauri Koskela	2019	– takt planning – takt control (TPTC) – job sequencing – work in progress – making do/task diminishment – tolerance management	In theory, a takt plan can significantly reduce throughput. In reality, the shop prioritized keeping up the schedule instead of producing quality goods, which lead to delays. A new method for problem management was needed. [29]
Predictive Scheduling	Classical Planning in MDP Heuristics: with a Little Help from Generalization	Andrey Kolobov, Mausam, Daniel S. Weld	2010	– -	The paper proposes a novel heuristic approach to solve an MDP scheduling problem. [30]

Reinforcement Scheduling	Data Centers Job Scheduling with Deep Reinforcement Learning	Sisheng Liang, Zhou Yang, Fang Jin, Yong Chen	2020	– job scheduling – cluster scheduling – deep reinforcement learning – actor critic	RL applied to the job scheduling problem of processor resources. Their model showed better results when learning the rules itself instead of the fixed rule models previously used. [31]
Reinforcement Scheduling	Deep Reinforcement Learning for Semiconductor Production Scheduling	Bernd Waschneck, Andre´ Reichstaller, Lenz Belzner, Thomas Altenmüller, Thomas Bauernhansl, Alexander Knapp, Andreas Kyek	2018	– Production Scheduling – Reinforcement – Learning – Machine Learning – Semiconductor Manufacturing	Discusses the benefits and problems of an RL solution for the JSSP. To solve the problem, they suggest using multiple agents working in cooperation to achieve a balance of optimization goals. A reduction of lot delay was achieved. The paper further ratifies the benefits of flexibility coming from RL. [32]
Reinforcement Scheduling	Design, Implementation and Evaluation of Reinforcement Learning for an Adaptive Order Dispatching in Job Shop Manufacturing Systems	Andreas Kuhnle, Louis Schäfer, Nicole Stricker, Gisela Lanza	2019	– Reinforcement Learning, – Production Scheduling – Order Dispatching – Methodical Approach	A very comprehensive guide for creating RL models for the JSSP. It covers design, evaluation, and implementation on a methodical level. It states as well that a “Digital Twin” is one of the required prerequisites for a successful application of RL [33]

Reinforcement Scheduling	Dynamic scheduling for flexible job shop with new job insertions by deep reinforcement learning	Shu Luo	2020	– Flexible job shop scheduling – New job insertion – Dispatching rules – Deep reinforcement learning – Deep Q network	A very comprehensive paper of a concrete implementation of a DQN for the JSSP. Literature research compares other methods to the proposed one in the paper. The results show a significant performance increase in tardiness when comparing to naive scheduling methods like FIFO. [34]
Swarm Scheduling	Genetic algorithm in flexible work shop scheduling based on multi-objective optimization	Yahui Wang, Liuqiang Fu, Yongqiang Su, Qian Yang & Linfeng Wu	2018	– Multi-objective – optimized genetic algorithm – flexible work shop – scheduling	Experimental proof of a performance increase in flexible workshop scheduling using a GA. This paper is very sparse with information. It states it study was done to legitimize further research. [35]
Predictive Scheduling	Hybrid Deep Neural Network Scheduler for Job-Shop Problem Based on Convolution Two-Dimensional Transformation	Zelin Zang, Wanliang Wang, Yuhang Song, Linyan Lu, Weikun Li, Yule Wang, and Yanwei Zhao	2019	– -	ML used to extract scheduling knowledge from historical data. [36]
Genetic Scheduling	Local Search Genetic Algorithms for the Job Shop Scheduling Problem	Beatrice M. Ombuki, Mario Ventresca	2004	– genetic algorithms – local search – tabu search – job shop scheduling – combinatorial optimization	A great introduction to the JSSP. Solved the JSSP with local search GA. [37]

Reinforcement Scheduling	MINERVA: A Reinforcement Learning-based Technique for Optimal Scheduling and Bottleneck Detection in Distributed Factory Operations	Tara Elizabeth Thomas, Jinkyu Koo, Somali Chaterji, Saurabh Bagchi	2018	– -	The paper tackles tow problems at the same time. First, it solves the JSSP by introducing a reinforcement algorithm based on Q-learning. Second, it proposes a way to find the bottleneck resources on the schedule. The proposed model performs significantly better than naïve ones like FIFO and the bottleneck indication is superior to average waiting time and utilization. [38]
Reinforcement Scheduling	Minimizing total energy cost and tardiness penalty for a scheduling-layout problem in a flexible job shop system, A comparison of four metaheuristic algorithms	Ahmad Ebrahimi, Hyun Woo Jeon, Seokgi Lee, Chao Wang	2020	– Scheduling – Layout – Energy consumption – Transportation time – Hybrid metaheuristic – Flexible job shop	A comparison of different optimization models to optimize the energy consumption of a shop-floor layout. [39]
Reinforcement Scheduling	Multiple Resource Management and Burst Time Prediction using Deep Reinforcement Learning	Vaibhav Kumar, Siddhant Bhambri, Prashant Giridhar Shambharkar	2019	– reinforcement learning – job scheduling – Deep-Q Network	Developed DQN to solve job burst time detection and scheduling thereof. [40]

Reinforcement Scheduling	Optimization of global production scheduling with deep reinforcement learning	Bernd Waschneck, André Reichstaller, Lenz Belzner, Thomas Altenmüller, Thomas Bauernhansl, Alexander Knapp, Andreas Kyek	2018	– Production Scheduling – Reinforcement Learning – Machine Learning in Manufacturing	The paper discusses the use of a DQN in Industry 4.0 over multiple production facilities optimizing for different goals. The results showed that after 2 days of training the DQN was on par with previous heuristics without prior knowledge of the environment. They argue that a DQN is a superior solution to JSSP due to its flexibility and training speed. [41]
Simulation Scheduling	Optimization of setup times in the furniture industry	Tomasz Gawroński	2012	– Dispatching rule – Dynamic scheduling – Furniture – Sequence-dependent setup	Simulation-based priority matrix for the JSSP. [42]
Reinforcement Scheduling	Petri-net-based dynamic scheduling of flexible manufacturing system via deep reinforcement learning with graph convolutional network	Liang Hua, Zhenyu Liu, Weifei Hua, Yueyang Wang, Jianrong Tana, Fei Wu	2020	– Dynamic scheduling – Petri nets – Deep reinforcement learning Graph – convolutional networks – Digital twin – Flexible Manufacturing Systems	The paper solves the JSSP with an extension of a DQN with a Petri-net convolution layer. It analyses the benefits of being a smaller model size. The paper gives a programmatic overview of their used algorithms, too. [43]
Reinforcement Scheduling	Real-time scheduling for a smart factory using a reinforcement learning approach	Yeou-Ren Shiue, Ken-Chuan Lee, Chao-Ton Su		– Machine learning – Q-learning – Real-time scheduling – Reinforcement learning – Shop floor control	The paper proves the superiority of an RL algorithm compared to a GA for the JSSP of an FMS. [44]

Reinforcement Learning	Research on Open-pit Mine Vehicle Scheduling Problem with Approximate Dynamic Programming	Te Xu, Fengyuan Shi, Wenbo Liu	2019	– Open-pit mine – Vehicle scheduling – Approximate Dynamic Programming	The paper proposes an ADP learning algorithm based on Q-Learning to solve the scheduling of query tuck routes of an open-pit mine. They propose a dynamic model to schedule in real-time depended on the weather and other conditions. [45]
Repair Scheduling	Scheduling and Rescheduling with Iterative Repair	Monte Zweben, Eugene Davis, Brian Daun, Michael J. Deale	1993	– -	A scheduling system that uses iterative repair. Iteratively repairing a schedule with better options till a good solution is found. The paper compares different repair heuristics to balance accuracy and computational time. [46]
Repair Scheduling	Solving batch process scheduling/planning tasks using reinforcement learning	E. C. Martinez	1999	– Batch Process Management – Scheduling – Learning – Combinatorial Optimization	A combination of RL and iterative repair to find a good schedule. [47]
Scheduling Graph Modelling	The disjunctive graph machine representation of the job shop scheduling problem	Jacek Błażewicz, Erwin Pesch, Małgorzata Sterna	2000	– Disjunctive graph – Graph representations – Graph matrix – Scheduling theory	The paper discusses how to model a schedule on a graph and from a matrix from it, on which one can perform optimizations. [1]

Table 3 is the collection of 27 papers found during the literature research. Each paper was classified into a category and summarized. The summary is kept short to quickly give an overview and should be used together with the relevance matrix to determine the importance of the paper.

6 Summary

The research found agrees that the JSSP is NP-hard and thus with higher complexities it remains unsolvable in a useful time frame on current generation hardware [37] [41]. While other methods of optimization like GA and PSO have been explored and studied elaborately, it has been found that, especially when it comes to a highly flexible and adaptive work floor environment, an RL approach is superior [34] [36] [44].

Scheduling strategy	TP		MCT (minutes)		NT	
	Mean	SD	Mean	SD	Mean	SD
RL	13122.66	33.47	947.98	206.81	1442.80	338.22
SOM	13085.57	25.64	1155.95	270.37	1450.13	499.89
GA + DT	13067.90	40.71	1327.98	314.22	1601.77	649.02
GA + SVM	13078.56	37.51	1417.50	273.40	1588.77	668.74
DS	13048.60	78.36	1569.73	395.94	3189.90	1855.20
EDD	12978.17	91.22	2065.82	437.31	8021.40	2226.72
SPT	13061.00	49.00	1440.84	297.91	1759.10	713.56
SIO	13017.37	80.55	1673.51	336.97	3076.73	1575.75
SRPT	13056.77	43.30	1536.91	346.79	1630.60	750.74

Figure 6: This table shows results obtained by Shiue et al. [44] comparing different JSSP optimizers, indicating that RL is superior.

6.1 The Job-Shop Scheduling Problem

The JSSP is extensively researched with papers discussing the topic found on Scopus dating back till 1970 [22], where the researchers made use of a branch and bound algorithm to minimize completion time (makespan) of jobs. With increasing computational power researchers in 1993 at NASA started experimenting with iterative methods and created the iterative repair method to find an optimal schedule for their space shuttle launches [46]. In 1995 one of the first RL methods for the JSSP was introduced expanding on the previous work done by NASA [24]. Combining the iterative repair method with TD they managed to bring down the calculation time while maintaining accuracy. With more computing time researchers explored different optimization strategies mostly genetic [37] and particle swarm algorithms [28]. With a lot of new research done in ML in recent years, it opened the field for new ideas and techniques in RL as well. Researchers at Deepmind managed to extend the Q-Learning algorithm and develop an agent, who can play Go, who won against the human world champion, which was previously thought to be impossible for a computer [8]. The

algorithm found by Deepmind was quickly adapted to JSSP and opened a large field of studies on how to perfect it for the JSSP [25] [31] [38] [32].
The idea of the problem is very consistent over all papers, defining the problem without any huge deviations, mostly in a linear programming problem type of style. The research agrees as well on the topic that the problem is NP-hard and thus not solvable in realistic time as a linear programming problem [37] [41].
The research further agrees that the JSSP can be modeled as an MDP, where RL is a powerful method to solve it. As a prerequisite to making RL work the factory needs to be of the so-called Industry 4.0 ("Smart Factory") type and have a digital twin of their resources [44].

6.2 Solutions

There are various ways to solve the JSSP with different benefits and deficits and approaches have changed over time as well. The need for RL for job-shop scheduling was discussed as early as 1994 [27]. In general, it can be said that as long as the production goals of the factory are met, the algorithm and heuristic used to achieve the goal have their merit. Taking a closer look at the different strategies it quickly becomes apparent what the pros and cons of each method are. While nearly any JSSP can be stated as a linear programming problem, the solving of one by mathematical means is rarely discussed and quickly dismissed since the computation time exceeds the benefits of finding a perfect solution. Since close to perfect solutions are generally accepted as good enough, this makes iterative and heuristic methods perfect candidates to solve the problem.
To solve the problem various variants of GAs, VNS, TS, and PSO where explored often in combination with TD. Since the success of deep Q-Learning, the focus of research has changed strongly to versions of it. Researched showed that given enough time nearly all proposed optimization algorithms will find a good enough solution which will not significantly differ to their competitors [24], but it was found that deep Q-Learning algorithms offer a huge advantage in processing time and flexibility in comparison to the other approaches [44]. DQNs are highly and quickly adaptable to new environments and need only a short amount of training time in comparison. When it comes to complexity, DQNs are also scalable over multiple factories and allow multi-target optimization, by training multiple agents in cooperation with different priorities. All these points make deep Q-Learning the superior solution to JSSPs.

6.2.1 Concrete solutions to JSSP

Shahrabi et al. [25] proposed a Q-Learning approach using VNS to find the close to best solution for JSSP. The proposed algorithm starts over with the old state if the planning horizon is violated ending in zero rewards. If the planning horizon is met, a reward is calculated, and the new schedule is used for the next iteration.

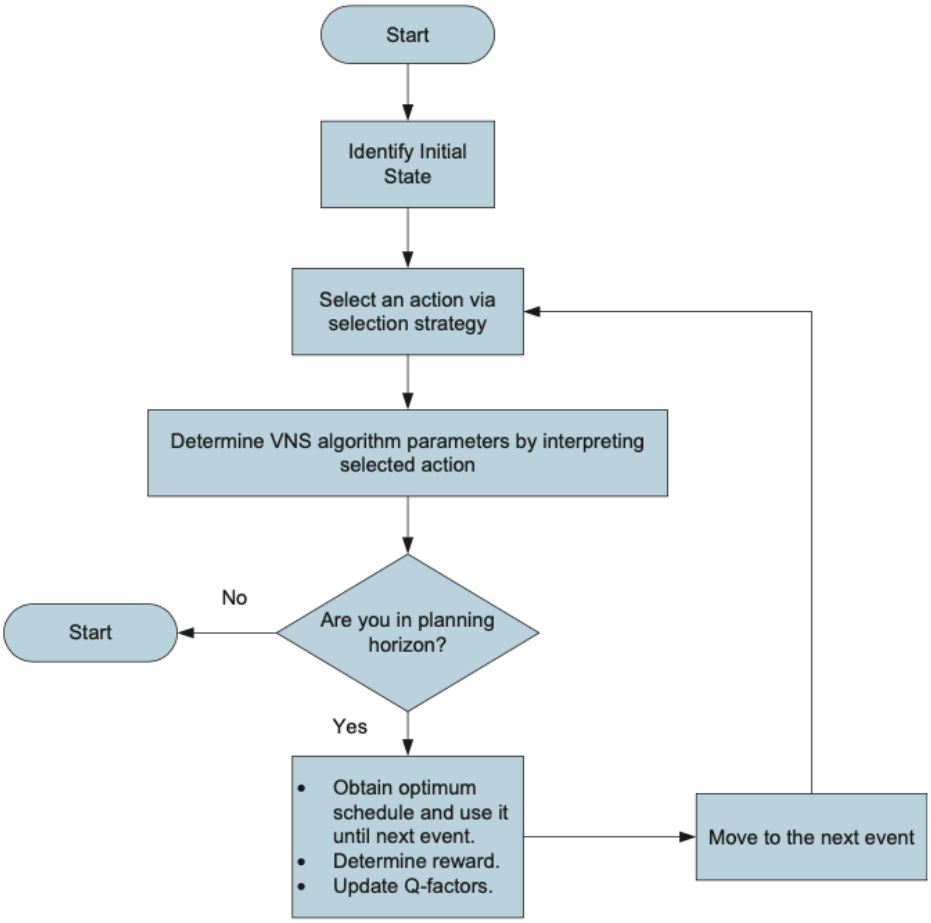

Figure 7: The proposed flow of the solution for the JSSP by Shahrabi et al. [25].

The work done by Priore et al. [26] suggests using online learning for scheduling since the model is faster to react and adapt to environmental changes and manual overwrites, such as breakdowns and rushed jobs.
Research done by Thomas et al. [38] suggests a domain-specific language (DSL) to simplify the description of the shop-floor and the production jobs for orders to enable easier development of optimization solutions for the JSSP.

6.3 Results

The literature research yields a large number of papers trying to solve the JSSP in various ways. The found research indicates that the future of solving the JSSP is heading into the direction of self-learning agents like DQNs. The need for RL was already discussed in 1994 [27]. And as of today, a complete guide to how to design, implement, and validate a DQN model for the JSSP [33], is given over diverse papers. Open remains the question of how deep Q-Learning can be applied to other job-shop management styles, in particular the one of a takt, where no research was found, referring to the results shown in Table 1.

In addition, while a lot of comparisons between models were made, till today there is no standard job database or creation ruleset to allow a deeper and more accurate comparison between models. Some models were only validated versus naïve methods like FIFO, which are barely used in commercial production scheduling systems.
Further, most of the research only used labor simulations and real-world validation of the methods is missing.

7 Conclusion

The JSSP is a highly researched topic with a lot of improvements happening over the years. Starting from a simple branch and bound algorithms to heuristics like GA and PSO, with the current state of research focusing strongly on RL in particular DQNs. Finding a niche to further improve and extend existing methods is difficult. The results of the literature show that so far no-one has touched the subject of RL in a takt production environment, like the one proposed by the SRS project, which should be further explored. From the solutions other researchers proposed, applying a DQN to the SRS project seems the most reasonable.

7.1 Future Lookout

The next steps for the SRS project will be to create a value function and define a data set to enable training. After that, a model of the SRS should be created in the OpenAi gym environment to allow the testing of different reinforcement algorithms. The OpenAi gym is the research standard when it comes to testing RL algorithms [20]. The results of that would be integrated into the final system.

8 Glossary

Table 4: Abbreviations and synonyms

Abbreviation	Word
A3C	Asynchronous Advantage Actor-Critic
DQN	Deep Q Network
GA	Genetic Algorithm
JSSP	Job-Shop Scheduling Problem
ML	Machine Learning
MDP	Markov Decision Process
PSO	Particle Swarm Optimization
RL	Reinforcement Learning
TS	Tabu Search
TD	Temporal Difference Learning
VNS	Variable Neighborhood Search

9 Bibliography

[1] J. Błażewicz, E. Pesch and M. Sterna, "The disjunctive graph machine representation of the job shop scheduling problem," *European Journal of Operational Research,* vol. 127, no. 2, pp. 317-331, 1 12 2000.

[2] M. R. Garey and D. S. Johnson, Computers and Intractability: A Guide to the Theory of NP-Completeness, W. H. Freeman and Company, 1979.

[3] J. F. Muth and G. L. Thompson, Industrial scheduling, Englewood Cliffs, N.J., Prentice-Hall, 1963.

[4] W.-Y. Ku and J. C. Beck, "Mixed Integer Programming Models for Job Shop Scheduling: A Computational Analysis," *Computers & Operations Research,* vol. 73, pp. 165-173, 2016.

[5] T. Yamada and R. Nakano, "Chapter 7: Job-shop scheduling," in *Genetic algorithms in engineering systems*, The Institution of Electrical Engineers, 1997, p. 134–160.

[6] M. M. Drugan, "Reinforcement learning versus evolutionary computation: A survey on hybrid algorithms," *Swarm and Evolutionary Computation,* vol. 44, pp. 228-246, 2019.

[7] Wikipedia, "Tabu search," 11 8 2020. [Online]. Available: https://en.wikipedia.org/wiki/Tabu_search. [Accessed 28 8 2020].

[8] D. Silver, A. Huang, A. Guez, C. J. Maddison, L. Sifre, G. van den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner and Sutsk, "Mastering the game of Go with deep neural networks and tree search," *Nature,* vol. 529, no. 7567, pp. 484-489, 2016.

[9] Y. Li, "DEEP REINFORCEMENT LEARNING: AN OVERVIEW," 15 10 2018. [Online]. Available: https://arxiv.org/abs/1810.06339. [Accessed 20 8 2020].

[10] Wikipedia, "Markov decision process," 14 8 2020. [Online]. Available: https://en.wikipedia.org/wiki/Markov_decision_process. [Accessed 29 8 2020].

[11] D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel, T. Lillicrap, K. Simonyan and D. Hassabis, "A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play," Science, 2018.

[12] D. Ha and J. Schmidhuber, "World Models," 27 3 2018. [Online]. Available: https://arxiv.org/pdf/1803.10122.pdf. [Accessed 29 8 2020].

[13] OpenAi, "Part 2: Kinds of RL Algorithms," OpenAi, 2018. [Online]. Available: https://spinningup.openai.com/en/latest/spinningup/rl_intro2.html#citations-below. [Accessed 29 8 2020].

[14] A. Clemmer, "What are the pros and cons of offline vs. online learning? In what scenarios are each useful?," Quora, 12 11 2012. [Online]. Available: https://www.quora.com/What-are-the-pros-and-cons-of-offline-vs-online-learning-In-what-scenarios-are-each-useful. [Accessed 30 8 2020].

[15] V. Kuleshov and D. Precup, "Algorithms for multi-armed bandit problems," *Journal of Machine Learning Research ,* vol. 1, pp. 1-48, 2000.

[16] R. S. Sutton and A. G. Barto, Reinforcement Learning: An Introduction, Massachusetts: The MIT Press Cambridge, 2014.

[17] G. Tesauro, "TD-Gammon: A Self-Teaching Backgammon Program," in *Applications of Neural Networks*, Boston, Springer, 1995, pp. 267-285.

[18] Wikipedia, "TD-Gammon," 17 2 2020. [Online]. Available: https://en.wikipedia.org/wiki/TD-Gammon. [Accessed 28 8 2020].

[19] C. J. C. H. Watkins, "Learning from Delayed Rewards," 5 1989. [Online]. Available: http://www.cs.rhul.ac.uk/~chrisw/new_thesis.pdf. [Accessed 20 8 2020].

[20] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra and M. Riedmiller, "Playing Atari with Deep Reinforcement Learning," 19 12 2013. [Online]. Available: https://arxiv.org/pdf/1312.5602.pdf. [Accessed 20 8 2020].

[21] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Harley, T. P. Lillicrap, D. Silver and K. Kavukcuoglu, "Asynchronous Methods for Deep Reinforcement Learning," 4 2 2016. [Online]. Available: https://arxiv.org/pdf/1602.01783.pdf. [Accessed 29 8 2020].

[22] G. Bozoki and J.-P. Richard, "A branch-and-bound algorithm for the continuous-process job-shop scheduling problem," *A I I E Transactions ,* vol. 2, no. 3, pp. 246-252, 1970.

[23] M. M. Nasiri, S. Salesi, A. Rahbari, N. S. Meydani and M. Abdollai, "A data mining approach for population-based methods to solve the JSSP," *Soft Computing,* vol. 23, p. 11107–11122, 2019.

[24] W. Zhang and T. G. Dietterich, "A reinforcement learning approach to job-shop scheduling," *IJCAI'95: Proceedings of the 14th international joint conference on Artificial intelligence,* vol. 2, pp. 1114-1120, 1995.

[25] J. Shahrabi, M. A. Adibi and M. Mahootchi, "A reinforcement learning approach to parameter estimation in dynamic job shop scheduling," *Computers & Industrial Engineering,* vol. 110, pp. 75-82, 2017.

[26] P. Priore, D. d. l. Fuente, A. Gomez and J. Puente, "A review of machine learning in dynamic scheduling of flexible manufacturing systems," *Artificial Intelligence for Engineering Design, Analysis and Manufacturing,* vol. 15, pp. 251-263, 2001.

[27] H. Aytug, S. Bhattacharyya, G. J. Koehler and J. L. Snowdon, "A Review of Machine Learning in Scheduling," *Transactions on Engineering Management,* vol. 41, no. 2, pp. 165-171, 1994.

[28] F. Zhao, J. Tang, J. Wang and Jonrinaldi, "An improved particle swarm optimization with decline disturbance index (DDPSO) for multi-objective job-shop scheduling problem," *Computers & Operations Research,* vol. 45, pp. 38-50, 2014.

[29] O. Alhava, V. Rinne, E. Laine and L. Koskela, "Can a Takt Plan Ever Survive Beyond the First Contact With the Trades On-Site?," in *Proceedings of the 27th Annual Conference of the International Group for Lean Construction*, 2019.

[30] A. Kolobov, Mausam and D. S. Weld, "Classical Planning in MDP Heuristics: with a Little Help from Generalization," in *Proceedings of the 20th International Conference on Automated Planning and Scheduling*, Toronto, 2010.

[31] S. Liang, Z. Yang, F. Jin and Y. Chen, "Data Centers Job Scheduling with Deep Reinforcement Learning," in *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, 2020.

[32] B. Waschneck, A. Reichstaller, L. Belzner, T. Altenmüller, T. Bauernhansl, A. Knapp and A. Kyek, "Deep Reinforcement Learning for Semiconductor Production Scheduling," in *29th Annual SEMI Advanced Semiconductor Manufacturing Conference (ASMC)*, 2018.

[33] A. Kuhnle, L. Schäfer, N. Stricker and G. Lanza, "Design, Implementation and Evaluation of Reinforcement Learning for an Adaptive Order Dispatching in Job Shop Manufacturing Systems," in *52nd CIRP Conference on Manufacturing Systems*, 2019.

[34] S. Luo, "Dynamic scheduling for flexible job shop with new job insertions by deep reinforcement learning," *Applied Soft Computing Journal,* vol. 91, 2020.

[35] Y. Wang, L. Fu, Y. Su, Q. Yang and L. Wu, "Genetic algorithm in flexible work shop scheduling based on multi-objective optimization," *Journal of Interdisciplinary Mathematics,* vol. 21, no. 5, pp. 1249-1254, 2018.

[36] Z. Zang, W. Wang, Y. Song, L. Lu, W. Li, Y. Wang and Y. Zhao, "Hybrid Deep Neural Network Scheduler for Job-Shop Problem Based on Convolution Two-Dimensional Transformation," *Computational Intelligence and Neuroscience,* vol. 2019, no. 2, pp. 1-19, 2019.

[37] B. M. Ombuki and M. Ventresca, "Local Search Genetic Algorithms for the Job Shop Scheduling Problem," *Applied Intelligence,* vol. 21, pp. 99-109, 2004.

[38] T. E. Thomas, J. Koo, S. Chaterji and S. Bagchi, "MINERVA: A Reinforcement Learning-based Technique for Optimal Scheduling and Bottleneck Detection in Distributed Factory Operations," in *10th International Conference on Communication Systems & Networks (COMSNETS)*, 2018.

[39] A. Ebrahimi, H. W. Jeon, S. Lee and C. Wang, "Minimizing total energy cost and tardiness penalty for a scheduling-layout problem in a flexible job shop system: A comparison of four metaheuristic algorithms," *Computers & Industrial Engineering,* vol. 141, 2020.

[40] V. Kumar, S. Bhambri and P. G. Shambharkar, "Multiple Resource Management and Burst Time Prediction using Deep Reinforcement Learning," in *Eighth International Conference on Advances in Computing, Communication and Information Technology CCIT*, 2019.

[41] B. Waschneck, A. Reichstaller, L. Belzner, T. Altenmüller, T. Bauernhansl, A. Knapp and A. Kyek, "Optimization of global production scheduling with deep reinforcement Optimization of global production scheduling with deep reinforcement learning," in *51st CIRP Conference on Manufacturing Systems*, 2018.

[42] T. Gawroński, "Optimization of setup times in the furniture industry," *Annals of Operations Research ,* vol. 201, pp. 169-182, 2012.

[43] L. Hua, Z. Liu, W. Hua, Y. Wang, J. Tana and F. Wu, "Petri-net-based dynamic scheduling of flexible manufacturing system via deep reinforcement learning with graph convolutional network," *Journal of Manufacturing Systems,* vol. 55, pp. 1-14, 2020.

[44] Y.-R. Shiue, K.-C. Lee and C.-T. Su, "Real-time scheduling for a smart factory using a reinforcement learning approach," *Computers & Industrial Engineering,* vol. 125, pp. 604-614, 2018.

[45] T. Xu, F. Shi and W. Liu, "Research on Open-pit Mine Vehicle Scheduling Problem with Approximate Dynamic Programming," in *2019 IEEE International Conference on Industrial Cyber Physical Systems (ICPS)*, Taipei, 2019.

[46] M. Zweben, E. Davis, B. Daun and M. J. Deale, "Scheduling and Rescheduling with Iterative Repair," *IEEE Transactions on Systems Man and Cybernetics ,* vol. 23, no. 6, pp. 1588-1596, 1993.

[47] E. C. Martinez, "Solving batch process scheduling/planning tasks using reinforcement learning," *Computers & Chemical Engineering,* vol. 23, pp. 527-530, 1999.

[48] Wikipedia, "Reinforcement learning," Wikipedia, 22 7 2020. [Online]. Available: https://en.wikipedia.org/wiki/Reinforcement_learning. [Accessed 20 8 2020].

10Table of Figures

11 Table of Tables