Imitation Learning Based on Deep Reinforcement Learning for Solving Scheduling Problems

Abdulrahman Nahhas
Faculty of Computer Science
Otto von Guericke University
Magdeburg, Germany
abdulrahman.nahhas@ovgu.de

Andrey Kharitonov
Faculty of Computer Science
Otto von Guericke University
Magdeburg, Germany
andrey.kharitonov@ovgu.de

Christian Haertel
Faculty of Computer Science
Otto von Guericke University
Magdeburg, Germany
christian.haertel@ovgu.de

Klaus Turowski
Faculty of Computer Science
Otto von Guericke University
Magdeburg, Germany
klaus.turowski@ovgu.de

Abstract

Scheduling problems are present in various industrial and service sectors and have a great deal of impact on the performance of these systems. The overwhelming majority of industrial problems exhibit a data-analytic or optimization nature, which can be reduced to known machine learning or optimization problems, respectively. This paper demonstrates the integration of optimization and Deep Reinforcement Learning (DRL) techniques to address scheduling problems. The study explores the potential advantages of Imitation Learning (IL) principles in achieving an optimization and machine learning pipeline for online scheduling. We employ an evolutionary optimization algorithm as an expert policy to generate high-quality solutions for solving scheduling problems. The obtained solutions are passed in the form of experiences to train a DRL-based IL technique. The presented approach is based on adopting the Nondominated Sorting Genetic Algorithm three (NSGA III) and the Monotonic Advantage Re-Weighted Imitation Learning (MARWIL). The presented approach is evaluated using real instances of a Hybrid Flow Shop (HFS) scheduling problem. The experimental analysis demonstrates that the presented DRL-based IL approach learns an appropriate scheduling policy, which is superior to training an agent without previous experiences. Additionally, the derived policy sustains a steady increase in performance when exposing the agent to different unknown problems in contrast to an established baseline from the literature for solving the same problems.

Keywords: Imitation Learning, Deep Reinforcement Learning (DRL), HFS Scheduling problems, Simulation, Optimization techniques, Parallelization

1. Introduction

Most problems in small and medium companies exhibit optimization or data-analytic nature, which can be reduced to known forms of optimization, classification, clustering, or prediction problems. Firstly, descriptive analytics is pursued to understand and characterize historical data to achieve a particular interpretation of the past. However, understanding the past is the first step for conducting predictive analytics, in which it is attempted to predict and obtain a rough estimation of the likely future. Finally, the final goal is to conclude this process by conducting prescriptive analytics, in which decision-making processes are supported based on adopting evolutionary optimization or machine learning techniques to improve the future (El Morr & Ali-Hassan, 2019). In this paper, we aim to investigate the utilization of machine learning and evolutionary optimization techniques to cover the mentioned stages for supporting real-time decision-making processes. We intend to investigate the design of a machine learning and optimization pipeline for addressing scheduling problems.

The majority of manufacturing problems are related to well-studied fields of research, such as resource allocation, scheduling, or controlling (Bakator et al., 2018). Scheduling problems are among the most challenging problems since their majority can be addressed only by adopting heuristic or metaheuristic techniques. Scheduling tasks involves assigning resources to complete a set of operations in a specific order. The common consensus in the related scheduling literature classifies scheduling solution techniques into two main categories: constructive and improvement techniques. These techniques are usually adopted when

URI: https://hdl.handle.net/10125/106585
978-0-9981331-7-1

the complexity of the investigated problem is known and/or proven to be not solvable in polynomial time using mathematical techniques, given the current computational limitations. Constructive heuristics are simple scheduling procedures that are overwhelmingly based on some ranking mechanisms. A production schedule is constructed based on the characteristics of jobs and a simple ranking property. Many Priority Dispatching Rules (PDRs), such as Shortest Processing Time (SPT) or Earliest Due Date (EDD), are simple constructive heuristics (Hunsucker & Shah, 1994).

Evolutionary and other metaheuristic algorithms are well-established improvement methods. They are adopted to solve combinatorial optimization problems (Glover & Kochenberger, 2003). Their popularity is explained by their high performance and robustness in solving complex industrial scheduling problems. As the name implies, this class of algorithms is inspired by evolution theory, in which evolution processes (natural selection, crossover, mate, and mutation) are applied systematically to an initial set of solution individuals to seek improvements (Kacem et al., 2002). Often, the optimization process starts with a set of randomly generated solutions that construct a set of production schedules. After that, improvements are pursued on the initial set through systematic modifications to minimize or maximize single or multi-objective optimization measures (Voudouris & Tsang, 2003). Other metaheuristic techniques are similarly designed to mimic some natural processes, such as Simulated Annealing (SA) (Kirkpatrick et al., 1983), which is based on imitating the metal annealing process or Swarm Intelligence (SI) (Liao et al., 2007), in which the behavior of a swarm in nature are reconstructed.

Despite the outstanding performance of the metaheuristics for solving various scheduling problems, they can be computationally expensive. To mitigate previous challenges, many research efforts on hybrid techniques are presented (Dey et al., 2018; Nahhas, Kharitonov, Alwadi, & Turowski, 2022; Ribas et al., 2010). Similarly, descriptive and predictive analytical approaches are well-investigated in the related literature on machine learning (Sharp et al., 2018). However, the adoption of machine learning techniques and their potential for solving complex scheduling problems is marginally investigated in the literature. Therefore, the presented approach aims to answer the following research questions (RQ):

- *RQ 1: How can we learn from previous high-quality solutions to accelerate the training process of Deep Reinforcement Learning (DRL) techniques?*
- *RQ 2: How can the principles of imitation learning be applied to train DRL techniques?*

The remainder of this paper is structured in six sections. After the introduction, we provide some necessary preliminaries for DRL and IL techniques. In the third section, an overview of related works is presented. The fourth section presents the DRL-based IL approach. In the fifth section, we discuss adopted algorithms and the implementation of the presented approach. The sixth section summarizes the collected results and discusses the performance of the approach. Finally, the last section concludes the research findings and suggests further research directions.

2. Deep reinforcement learning and imitation learning: background

Many successful research and application efforts are presented in image analytics, video analytics, and language models (Liu et al., 2017). The adoption of Deep Neural Networks, including Deep Convolutional Neural Networks (CNNs) architectures, is dominant in literature since they deliver satisfactory performance. Recently, DRL techniques are considered cutting-edge machine learning techniques, initially based on the Markov Decision Process (Papadimitriou & Tsitsiklis, 1987). As a stand-alone ML paradigm, DRL combines the advantages of supervised and unsupervised learning strategies. Analogous to unsupervised learning, DRL methods do not require pre-learned training data, where each input's expected output is known in advance. However, DRL methods do not entirely dispense with training labels but instead generate them based on a well-defined reward function. The fundamental building blocks of a DRL technique are four folds:

- A DRL algorithm called agent is designed based on Deep Neural Networks (DNN).
- An environment that is exposed to the agent for interaction. Ideally, from a practical point of view, the environment is an abstract representation of a real system.
- Action and observation spaces, whereby the former allows the agent to manipulate the state of the environment and the latter informs that agent of the current state of the environment.
- A reward function is used to either positively reward or negatively punish the agent after taking a certain action.

The training process of a DRL algorithm starts with the agent's action, which is passed to the environment to transition to the next state and collect evaluation matrices. Throughout the training process, the agents are exposed to different environmental observations and assigned rewards for the actions taken. The agent's ultimate goal is to maximize the reward in dealing with

a particular task, which implies successfully learning an appropriate policy that forms a strategy to deal with this task (Henderson et al., 2018). Most DRL methods are investigated and primarily applied using game-like environments with little industrial context (Kanervisto et al., 2020). Improvement of the DRL agent's behavior, commonly denoted as policy, depends largely on the experiences used for training this policy. In the standard DRL settings, these experiences are acquired via direct agent interaction with the environment. Hence, an agent receives rewards given a specific environment state based on an environment-altering action. Typically, the agent starts without prior experience or knowledge of the environment. At this point, the agent explores the environment to generate initial experiences while attempting to establish a relationship between an environment state and an action. In essence, a new agent starts its existence by exploring the environment by taking random actions until it can discover a trajectory/ sequence of actions that improve the reward.

The DRL research community proposes multiple techniques to mitigate this challenge. One such technique is Imitation Learning (IL). In the scope of this work, the agent has access to a set of demonstrations generated by an expert (human or a system). These demonstrations are supplemented to the agent as initial experience used to batch-train the policies without the agent performing any interaction with the actual environment. Therefore, the agent uses these experiences to imitate the actions provided by the expert. Training of the agent's policies based on the experiences supplemented by an outside expert policy is technically possible in many DRL algorithms. However, additional information is often required for the agent to train on the experiences generated by an outside policy (e.g., action probability distribution), thus limiting the applicability of such algorithms where an unknown expert policy is used to generate the experiences.

Q. Wang et al., (2018) proposed a Monotonic Advantage Re-Weighted Imitation Learning (MARWIL) algorithm to mitigate issues typically associated with IL. This algorithm can be used in data-driven training of the agent's policies based on the batched historical data by applying monotonic advantage re-weighting. Therefore, it enables the agent to determine the impact of taking a certain action within a specific environment state, essentially estimating the advantage of taking this specific action. We selected MARWIL for the evaluation part of this work due to the relative simplicity of the algorithm, which enables rapid training of the agents. MARWIL allows deploying the agents whose policy was trained using expert-generated experiences. This capability is not always implicit within DRL algorithms trained or aimed at IL, but it is crucial for the concept discussed within this work.

3. Related works and state of the art

Our literature analysis reveals a few publications on applying imitation learning to handle scheduling problems. Therefore, we will discuss the identified related works in which RL, DRL, or IL techniques are adopted to investigate Hybrid Flow Shop (HFS) scheduling problems. The interpretation of the related works starts with the least complicated problem and progresses to the most complicated one. In an HFS scheduling problem, jobs undergo several processing stages. At every stage, parallel machines are available to process all types of jobs. Jobs are always processed in the same technological order. We rely on HFS scheduling problems for evaluation since some industrial problem instances are available. In addition, such operational specifications are widespread in many industrial environments.

3.1. Reinforcement Learning (RL)

Some attempt to deal with flow shop scheduling problems is presented by (Ren et al., 2021). The authors relied on RL for approximating the allocation of jobs to machines using Neural Networks (NN). It is worth noting that pure flow shop scheduling problems are way easier to solve than HFS problems. Another application of RL for solving HFS with three stages is proposed by (J. Wang et al., 2017). Each production stage contains two identical machines that can be used to process jobs. The authors evaluated the performance of the presented RL against known Priority Dispatching Rules (PDRs) such as the EDD, SPT, and First-In-First-Out (FIFO), taking into account the mean flow time, percentage of tardy jobs, and mean lateness.

Similarly, an application of RL combined with Boltzmann exploration policy for solving HFS scheduling problems is presented by (Han et al., 2019). The authors solved small HFS problem instances considering the minimization of the makespan compared to the previously discussed work. The makespan indicates the maximum completion time over the set of jobs that must be scheduled.

For a more complex two-stage HFS scheduling problem with identical machines, (Lang et al., 2021) presented a hybrid approach using a combination of Reinforcement Learning (RL) and Genetic Algorithms (GA). The authors relied on GA to optimize the structure of the underlying neural networks. The optimized NN is employed to approximate a sequence based on which a set of jobs must be scheduled. The presented NeuroEvolution of augmenting topologies is applied to solve the problem considering the minimization of the total tardiness.

3.2. Deep Reinforcement Learning (DRL)

The adoption of DRL techniques for scheduling is even more limited than RL. For instance, an application of DQN (Mnih et al., 2015) is presented by (Lang et al., 2020) to address job shop scheduling problems. The authors presented computational results superior to related works for solving four problem instances. The problems contained five to twenty jobs that must be mapped to mostly five available machines. For solving a complex HFS scheduling problem, (Zhu et al., 2020) presented one of the earliest adoptions of Proximal Policy Optimization (PPO) (Schulman et al., 2017) to minimize the makespan. The evaluation is conducted on randomly generated problem instances in which the agent acts as a dispatcher to set the sequence of the jobs. The results are compared to priority dispatching rules.

An application of the rival to the PPO DRL technique, namely, Advantage-Actor-Critic (A2C), is presented in (Gerpott et al., 2022). The authors evaluated the performance of A2C in solving a two-stage HFS scheduling problem to minimize total tardiness. The results suggest that the presented approach can support real-time scheduling. However, scheduling problems are overwhelmingly characterized by many objective measures in manufacturing environments. Obviously, many objective measures are conflicting, such as the makespan, total tardiness, or average flow time.

Consequently, modern solution techniques must satisfy tradeoffs and support multi-objective optimization. Recently, (Nahhas, Kharitonov, & Turowski, 2022) adopted two DRL techniques to address complex HFS scheduling problems considering the minimization of multiple optimization measures. PPO and Asynchronous Advantage Actor-Critic (A3C) are adopted to solve the problem based on an industrial use case. The presented approach comprises two main components: a DRL component that includes PPO and A3C implementations and a discrete event simulation model, which serves as an environment for the agents. The presented approaches are evaluated for solving four instances of multi-stage scheduling problems. The computational results show that both applied algorithms with various levels of success derive appropriate policies for solving the problems based on the encoding.

The empirical results show that the agents can achieve high-quality solutions that minimize the makespan, total tardiness, and major setup times over multiple production stages. However, experiments suggest that the agents' performance suffers when exposed to entirely different problems, requiring additional training to recover performance loss. Google Brain and DeepMind similarly highlight such generalization issues in (Chiyuan Zhang et al., 2018).

3.3 Imitation learning (IL)

After searching and scanning in Google Scholar and ScienceDirect and to the best of our knowledge, we found no application of IL techniques for solving HFS scheduling problems. After broadening our search, we identified very few publications that applied IL to solving Job shop scheduling problems (Cong Zhang et al., 2020). The authors proposed an approach based on PPO, in which the critic network is pre-trained with previous experiences obtained using different priority dispatching rules for solving the problems. The presented concept is based on graph embedding using a graph isomorphism network. The presented results prove the superiority of the presented concept against known Priority Dispatching Rules (PDRs). In addition, the authors argue that the approach sustains a certain level of generalization for solving large problem instances (Cong Zhang et al., 2020).

4. DRL-based Imitation Learning

To address the drawbacks that are pointed out in the previous works and to answer the research questions, we present a DRL-based Imitation Learning (IL) approach. We combine the use of state-of-the-art optimization and deep reinforcement learning techniques. The conceptual representation of the discussed concept is presented in Figure 1. The concept consists of four main components in addition to the parallelization component, which we discuss in the implementation section :

- An optimization component based on the Nondominated Sorting Genetic Algorithms three (NSGA III) (Jain & Deb, 2014).
- A machine learning component based on the novel Monotonic Advantage Re-Weighted Imitation Leaning (MARWIL) algorithm (Q. Wang et al., 2018).
- A simulation component based on the open-source Salabim simulation package (van der Ham, 2018).
- A repository of a set of constructive heuristics for allocation and sequencing. This component comprises the source code of known PDRs and other advanced heuristic algorithms from previous works (Nahhas et al., 2021).

Optimization component - This component acts as a stand-alone expert policy from the DRL point of view in conjunction with a simulation model. From a practical perspective, adopting a multi-objective evolutionary optimization technique is sufficient for solving the majority of scheduling problems. Certainly, the encoding of the problem and the formulation of the objective functions must be carefully conducted.

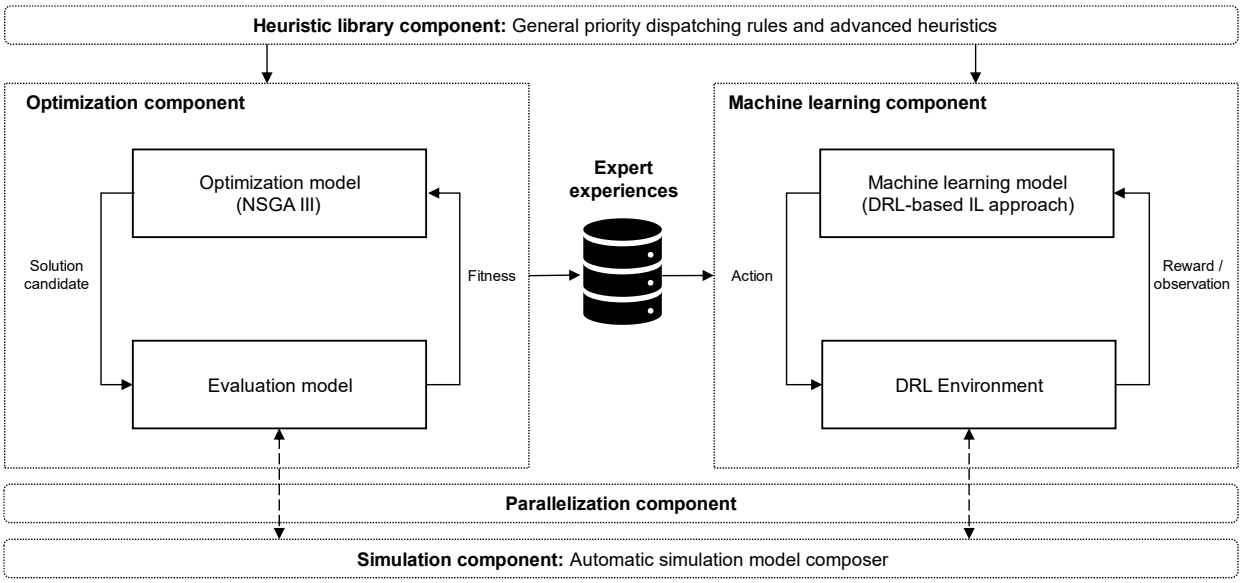

Figure 1. A DRL-based imitation learning approach.

However, as mentioned earlier, computational effort becomes a major drawback. Therefore, we suggest training a machine learning algorithm that imitates the behavior of an *expert* for solving the problems. The superiority of evolutionary optimization techniques for solving scheduling problems is beyond denial. Therefore, we employ the NSGA III as the *expert* in the presented concept. The optimization component can be used in the first stage of the solution deployment to suggest scheduling policies while collecting high-quality solutions *(previous experiences).* These experiences can be fed to a machine-learning technique directly or in batches.

Machine learning component - Previous studies proved the applicability of the Deep Reinforcement Learning (DRL) technique for solving HFS scheduling problems (Lang et al., 2020; Nahhas, Kharitonov, & Turowski, 2022; Zhu et al., 2020). However, these papers partially answer the first research question, while the use of high-quality experiences is yet to be explored. Although some of the known reference implementations of DRL techniques are described to support imitation learning, no investigation of their use for solving HFS scheduling problems is found in the literature. Therefore, after empirical analysis of the viability of implementation and performance of PPO, A3C, and MARWIL, we decided to rely on MARWIL as the core of the machine learning component. The machine learning component is designed to learn from previous experiences that an expert generates. These experiences can be historical data or on-the-fly generated solutions through optimization. The experiences of the optimization component can be either passed directly to the machine learning component or collected in batches before updating the learning policy of a DRL-based IM agent. In the online case, after an optimization run is triggered, high-quality solutions individuals are passed to the machine learning component with appropriate observations from a simulation model.

Simulation component – This component builds up on a discrete-event simulation core engine to automatically compose simulation models using structural data of a considered system. As demonstrated in Figure 1, the optimization and machine learning components heavily rely on the simulation component to evaluate. A simulation model acts as an evaluation method to investigate the fitness of generated solutions by the optimization component. As for the machine learning component, an abstraction of a real system is required to couple a DRL / IL agent with an appropriate environment. This process is automated by the simulation component, which requires structural data of the considered system to build a corresponding simulation model.

Heuristic library component - In this paper, the suggested encoding of scheduling problems allows an optimization strategy (NSGA III / MARWIL) to control a set of allocation and sequencing algorithms used during the simulation to construct a production schedule. The heuristic library is enriched with general and field-specific heuristic algorithms such as Priority-Dispatching-Rules (PDRs). It is designed and plugged into the optimization and machine learning components in a modular fashion within the implemented framework, which easily allows the extension of the library by including further algorithms.

5. Prototypical implementation of the concept for scheduling problems

5.1. Problem formulation and objective values

To evaluate the presented Imitation Learning (IL) approach, we relied on problem instances addressed in previous works to establish a baseline for comparison (Nahhas, Kharitonov, & Turowski, 2022). We solve four-stage Hybrid Flow Shop (HFS) problem instances with major and minor family setup times. The first, second, and third processing stages contain five parallel machines of different speeds. Two parallel machines are available to process all jobs in the fourth processing stage. The following attributes characterize a job j∈ J:

- The processing time in the first stage $p_{j,\ s1}$
- The processing time in the second stage $p_{j,\ s2}$
- The processing time in the third stage $p_{j,\ s3}$
- The processing time in the fourth stage $p_{j,\ s4}$
- The due date of a job d_j
- The completion time C_j
- The tardiness of job j is T_j if the completion C_j is bigger than its due date d_j.

The problem is to be solved by finding a production schedule in which all operations ($O_{j,\ s1} \rightarrow O_{j,\ s4}$) are completed. The optimization is conducted to minimize many objective measures: makespan C_{max}, the total number of the required major setup times on the first and fourth processing stages $MS_{first,\ fourth}$, the total tardiness *T* and the total number of unit penalties *U*. The objective values are formalized in (1), (2), (3), and (4), respectively. The major setup times $MS_{first,\ fourth}$ are lost configuration time of the machines when switched to process jobs from a different family.

$$C_{max},\ max\, C_j : \forall J_j\ (j \in \{1, \ldots, n\} \quad (1)$$

$$MS_{first,\ fourth} \in \{0, \ldots, n-1\} \quad (2)$$

$$T = \sum_{j=1}^{n} T_j\ :\ \forall J_j\ (j \in \{1, \ldots, n\}) \quad (3)$$

$$U = \sum_{j=1}^{n} U_j\ :\ \forall J_j\ (j \in \{1, \ldots, n\}) \quad (4)$$

5.2. Encoding the scheduling problem in the presented concept

This section presents the rationale for our suggested IL technique. We must rely on a unified encoding to collect high-quality experiences from the optimization component and pass them further to the machine learning component. In the presented IL approach, the optimization technique seeks to select the best combination of allocation and sequencing heuristics from the heuristic library component during the scheduling period. An example of adopting a metaheuristic to select different heuristics for solving HFS scheduling problems can be found in (Rolf et al., 2020). Based on the selection, jobs are allocated to the available machine in the first processing stage. Then, jobs are dispatched using the selected sequencing algorithms on every machine. In essence, a solution individual represents an arbitrary selection of allocation and sequencing heuristics maintained in two vectors of integer values. These values point to the indexes of heuristics.

5.3. Optimization component

Such encoding is crucial for successfully applying the DRL-based IL approach since the problem must be exposed to the agent as a game. The optimization starts with an initial set of randomly generated solution individuals. Then, the suggested solutions are evaluated using a discrete event simulation model, representing the investigated system's operational procedures and constraints. Based on the obtained fitness, we follow the natural selection strategy of the NSGA III using uniformly distributed reference points to select the individuals for generating the offspring population. We rely on two-point crossover mechanisms to construct offspring genomes based on the selection. As for the mutation, we adopt a simple shuffle mutation function in which two genes are swapped. After generating the offspring of the next generation, individuals are again evaluated using the simulation component. This process is repeated until some breaking criterion is met. In our setup, we use the number of maximum generations to break the optimization process. All generated solutions are stored in a database. A stored solution includes characteristics, objective values, and detailed production schedules. These solutions are then fed to machine learning in the form of previous experiences.

5.4. Machine learning component

Actions space – As mentioned earlier, we relied on a unified encoding identical to the optimization and used multi-discrete action space to formulate the problem for the DRL-based IL (Delalleau et al., 2019). Such formulation of the action space allows presenting the problem to the agent in a game-like manner. The agent maintains, in essence, two controllers (sequencing and allocation). Through these controllers, it interacts with the respective environment using the simulation component. Multi-discrete action is typically adopted

when the agent must produce a set of independent discrete actions to interact with an environment (Delalleau et al., 2019). Further elaboration of the definition of multi-discrete action space can be found in (Delalleau et al., 2019; Kanervisto et al., 2020). The agent decides which algorithms are used for allocation and sequencing at predefined points during a scheduling period. Based on the selection, jobs are allocated, and a selected sequencing algorithm is used for dispatching jobs.

Observation space and reward function – The design of the observation space of the DRL-based IL approach is threefold: The entire production schedule, system-level Key Performance Indicators (KPIs), and job-level KPIs. This encoding is based on previous work presented by (Nahhas, Kharitonov, & Turowski, 2022). The detailed production schedule provides the agent with a profound view of the system states resulting from the set of taken actions. It exposes the agent to the life cycle of every job, which includes details of the starting time and finishing times of all operations in the system on all machines. The second and third parts of the observation space dispense the agent with KPIs well-studied in the related literature. For instance, we supply the agent with the average waiting time, average flow time, the makespan, the total number of major setup times on processing stages, the total tardiness, and the total number of penalties (Ribas et al., 2010; Ruiz & Vázquez-Rodríguez, 2010). Other relevant objective measures are supplemented in the last part of the observation space on a job level, such as the total pure processing time, the overall waiting time, the tardiness, and the lateness of a job. Finally, the formulation of the reward function is based on the aforementioned objective optimization measures, which are minimized in our optimization and presented in (1), (2), (3), and (4). We further formulate these objective values in (5) in a simple maximization function for the agent and weight the unit penalties with ten to increase their importance for the agent.

$$R = -\left(C_{max} + MS_{first,\ fourth} + T + U*10\right) \quad (5)$$

5.5. Implementation overview

The implementation of the presented approach is purely developed in Python and supported by the parallelization component, which relies on the parallelization capabilities of Python multiprocessing, RabbitMQ (Roy, 2017) messaging system, and Ray RLlib distribution framework (Eric Liang et al., 2018). The objective of the parallelization component is to support an efficient distribution of optimization and machine learning experiments. This, in turn, ensures an efficient application of imitation learning. As discussed in the previous section, we rely on an evolutionary optimization technique to collect expert experiences for solving scheduling problems. The obtained results are then used to train the DRL-based IL agent. The development of the optimization component is relatively less complicated than the machine learning component. The implementation of the Nondominated Genetic Algorithms (NSGA III) is straightforward based on the original paper (Jain & Deb, 2014). The evaluation component is based on a Salabim simulation package, which is an open-source Python simulator. As for the parallelization, we relied on the Python multiprocessing package and RabbitMQ to accelerate the optimization process. Solution individuals are evaluated in parallel on all available CPU cores.

On the contrary, developing the machine learning component necessitates empirical and technical analysis. Many DRL techniques are not designed to leverage previous experiences or imitate other solution techniques. We empirically and technically investigated the adoption of Proximal Policy Optimization (PPO), Advantage-Actor-Critic (A3C), and Monotonic Advantage Re-Weighted Imitation Learning (MARWIL). We can adopt A3C and PPO as stand-alone algorithms while feeding them with previous experiences requires further technical investigations. The major challenge we faced was merging experiences generated by unknown policies (i.e., the experiences of NSGA III) with experiences generated by A3C or PPO. Both algorithms are eventually not intended DRL algorithms for imitation learning. We are aware of an implementation of PPO, which can be fed with previous experiences as presented by (Cong Zhang et al., 2020). Our computational results show that A3C and PPO learn appropriate policies for solving HFS scheduling problems without exploiting the potential of imitation learning, which aligns with the reported results in (Nahhas et al., 2022). We successfully adopted MARWIL for solving the scheduling problem based on expert experiences from the optimization component. To achieve an optimization and machine learning pipeline, we recommend relying on MARWIL, which also yields learning an appropriate policy for solving scheduling problems.

6. Evaluation and computational results

To solve the HFS scheduling problem, we configured the NSGA III to run an optimization for 200 generations. The optimization is conducted to solve three problem instances used for evaluation (Nahhas, Kharitonov, & Turowski, 2022). The population size is set at 75 individuals. The optimization is conducted to minimize the makespan, the total number of major setup times on the first and fourth processing stages, the total

tardiness, and the total number of penalties as previously discussed in (1), (2), (3), and (4) respectively. We applied a 0.8 crossover rate and 0.6 mutation rate to maintain population diversity. The parameterization is obtained empirically after conducting preliminary hyperparameter tuning. The optimization component generated fifteen thousand solution individuals for solving each problem instance. These solutions are then fed to the machine learning component to investigate imitation learning principles and evaluate the performance of the presented approach.

As for the machine learning component, we designed the experimental setup to highlight the functionality of imitating expert experience and to answer the first research question. The computational results are presented in Figure 2. We conducted multiple optimization and machine learning experiments to statistically observe their behavior, which is demonstrated as a dispersion in the figure. The agent is trained to solve three problem instances to achieve comparability to a published baseline. The DRL-based IL agent is trained with 450,000 previous experiences before exposing it to online mode for further training. The agent samples new solutions based on experiences and a built-in sampler with a predefined advantage factor-beta during the online mode. This factor controls the weighting of the quality of previous experiences, favoring high-quality ones (Q. Wang et al., 2018). We performed hyperparameter tuning through rapid experimentations. The parameters are set based on our analysis and to the best of our knowledge from related literature (Q. Wang et al., 2018). The DRL-based IL agent learning rate is set to be 1e-4. The *beta* parameter is set to 1.0 to motivate the agent to use weights of previous experiences for generating new solutions. The size of the reply buffer is set at 15,000. The reply buffer size controls the number of previous experiences that are available for the agent during the sampling. The computational results suggest that the DRL-based IL agent successfully derives an appropriate scheduling policy to deal with the HFS scheduling problems. As shown in Figure 2, the achieved mean reward based on the formulated reward function steadily increases during the training processes. This implies that the agent is successfully learning to solve the problems based on the encoding that we discussed in the previous section.

The orange line (*MARWIL Online*) in the presented figure represents the results of training MARWIL to solve the problems based on our encoding without exposing it to previous experiences. The blue line (*MARWIL IL*) depicts the result of the IL agent, which is fed with previous expert experiences. Over multiple optimization and machine learning runs, we observe consistent outperformance of the DRL-based IL technique for solving the HFS scheduling problems in terms of the achieved mean reward presented in equation (6). It is, on average, between nine and ten percent. The presented results strongly suggest that we can apply imitation learning principles based on the MARWIL algorithm for solving scheduling problems. The computational results provide initial evidence articulating that relying on high-quality experiences accelerates the training process of the DRL technique since MARWIL IL achieves a 10 percent higher reward on average, which answers the first research question - *RQ 1: How can we learn from previous high-quality solutions to accelerate the training process of Deep Reinforcement Learning (DRL) techniques?*. By integrating optimization and machine learning services along with a unified encoding of the problem, efficient application of imitation learning can be achieved. This would conclude our findings to answer the second research question - *RQ 2: How can the principles of imitation learning be applied to train DRL techniques?*

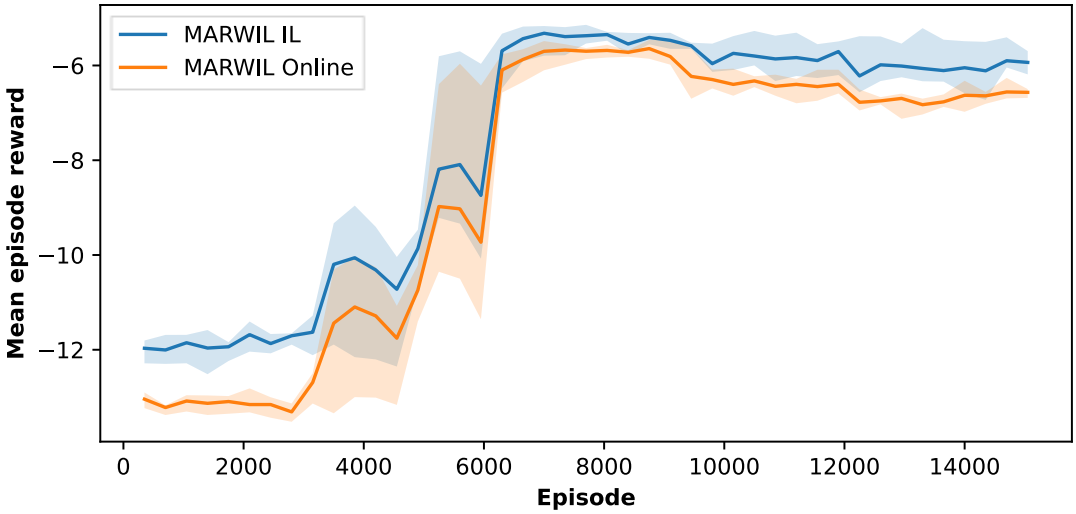

Figure 2. A comparison between MAWIL-IL and pure MAWIL without previous experiences

7. Conclusions and future research directions

In this research, we presented a DRL-based IL approach to investigate adopting the principles of imitation learning for addressing scheduling problems. The DRL-based IL approach is evaluated for solving HFS scheduling problems from related literature to establish a baseline comparison. To answer the second research question, *RQ 2*, we demonstrate that efficient integration of optimization machine learning services facilitates leveraging the principles of imitation learning. Our conceptual presentation of the integration does not roll out the use of historical high-quality schedules that might be collected in a company. However, we stress a unified encoding of the DRL problem and supplemented experiences to train an agent successfully. This recommendation is based on dealing with tedious technical problems due to slight deviations in the shape of the fed solutions. The DRL-based IL outperforms an agent without expert experiences, demonstrating a notable application of IL principles for addressing industrial problems, which answers the first research question, *RQ 1*.

The obtained results may slightly mitigate the reported generalization issues (Nahhas, Kharitonov, & Turowski, 2022). As demonstrated in Figure 2, exposing the agent to different problems does not lead to significant degradation in their performance in contrast to previous work. However, these findings are subject to certain limitations due to the computationally expensive nature of DRL experiments. The generalization and stability of DRL and DRL-based IL remain challenging and require further investigation for industrial use. Nonetheless, we performed further preliminary analysis to confirm this behavior in a dedicated experiment. We train the agent with previous experiences from the optimization component for solving three problems and then expose it to completely unknown other three problems. The results show that the DRL-based IL approach avoids significant performance degradation and sustains a steady increase in the mean reward. From a practical point of view, we present a two-phase approach that can be employed for addressing HFS scheduling concerns in real environments. In the first phase, an evolutionary optimization technique is used for scheduling, while DRL-based IL is trained with previous experiences. The second deployment phase is finally conducted once the agent proves stability and robustness for real-time scheduling.

References

Bakator, M., Ćoćkalo, D., & Nikolić, M. (2018). A model for manufacturing optimization and achieving higher productivity in small and medium-sized enterprises. In *Proceedings of VIII International Conference Industrial Engineering and Environmental Protection, Zrenjanin, Serbia*.

Delalleau, O., Peter, M., Alonso, E., & Logut, A. (2019, December 23). *Discrete and Continuous Action Representation for Practical RL in Video Games*.

Dey, S., De, S., & Bhattacharyya, S. (2018). Introduction to Hybrid Metaheuristics. In S. Bhattacharyya (Ed.), *Series in machine perception and artificial intelligence, 1793-0839: Vol. 84. Hybrid metaheuristics: Research and applications / Siddhartha Bhattacharyya* (Vol. 84, pp. 1–38). World Scientific Publishing.

El Morr, C., & Ali-Hassan, H. (2019). Descriptive, Predictive, and Prescriptive Analytics. In C. El Morr & H. a. Ali-Hassan (Eds.), *SpringerBriefs in Health Care Management and Economics, 2193-1704. Analytics in healthcare: A practical introduction / Christo El Morr, Hossam Ali-Hassan* (pp. 31–55). Springer.

Eric Liang, Richard Liaw, Robert Nishihara, Philipp Moritz, Roy Fox, Ken Goldberg, Joseph Gonzalez, Michael Jordan, & Ion Stoica (2018). Rllib: Abstractions for Distributed Reinforcement Learning. *International Conference on Machine Learning*, 3053–3062.

Gerpott, F. T., Lang, S., Reggelin, T., Zadek, H., Chaopaisarn, P., & Ramingwong, S. (2022). Integration of the A2C Algorithm for Production Scheduling in a Two-Stage Hybrid Flow Shop Environment. *Procedia Computer Science*, *200*, 585–594.

Glover, F., & Kochenberger, G. A. (Eds.). (2003). *Handbook of Metaheuristics*. Springer Science & Business Media.

Han, Guo, & Su (2019). A Reinforcement Learning Method for a Hybrid Flow-Shop Scheduling Problem. *Algorithms*, *12*(11), 222.

Henderson, P., Islam, R., Bachman, P., Pineau, J., Precup, D., & Meger, D. (2018). Deep Reinforcement Learning That Matters. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Hunsucker, J. L., & Shah, J. R. (1994). Comparative performance analysis of priority rules in a constrained flow shop with multiple processors environment. *European Journal of Operational Research*, *72*(1), 102–114.

Jain, H., & Deb, K. (2014). An Evolutionary Many-Objective Optimization Algorithm Using Reference-Point Based Nondominated Sorting Approach, Part II: Handling Constraints and Extending to an Adaptive Approach. *IEEE Transactions on Evolutionary Computation*, *18*(4), 602–622.

Kacem, I., Hammadi, S., & Borne, P. (2002). Approach by localization and multi-objective evolutionary optimization for flexible job-shop scheduling problems. *IEEE Transactions on Systems, Man and Cybernetics, Part C (Applications and Reviews)*, *32*(1), 1–13.

Kanervisto, A., Scheller, C., & Hautamäki, V. (2020, April 2). *Action Space Shaping in Deep Reinforcement Learning*.

Kirkpatrick, S., Gelatt, C. D., & Vecchi, M. P. (1983). Optimization by simulated annealing. *Science (New York, N.Y.)*, *220*(4598), 671–680.

Lang, S., Lanzerath, N., Reggelin, T., Müller, M., & Behrendt, F. (2020). Integration of Deep Reinforcement Learning and Discrete-Event Simulation for Real-Time Scheduling of a Flexible Job-Shop Production. In K.-H. G. Bae, B. Feng, S. Kim, S. Lazarova-Molnar, Z. Zheng, T. Roeder, & R. Thiesing (Eds.), *Proceedings of the 2020 Winter Simulation Conference (WSC'20)*. IEEE.

Lang, S., Reggelin, T., Schmidt, J., Müller, M., & Nahhas, A. (2021). NeuroEvolution of augmenting topologies for solving a two-stage hybrid flow shop scheduling problem: A comparison of different solution strategies. *Expert Systems with Applications*, *172*, 114666.

Liao, C.-J., Tseng, C.-T., & Luarn, P. (2007). A discrete version of particle swarm optimization for flowshop scheduling problems. *Computers & Operations Research*, *34*(10), 3099–3111.

Liu, W., Wang, Z., Liu, X., Zeng, N., Liu, Y., & Alsaadi, F. E. (2017). A survey of deep neural network architectures and their applications. *Neurocomputing*, *234*, 11–26.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., & Hassabis, D. (2015). Human-level control through deep reinforcement learning. *Nature*, *518*(7540), 529–533.

Nahhas, A., Kharitonov, A., Alwadi, A., & Turowski, K. (2022). Hybrid Approach for Solving Multi-Objective Hybrid Flow Shop Scheduling Problems with Family Setup Times. *Procedia Computer Science*, *200*, 1685–1694.

Nahhas, A., Kharitonov, A., & Turowski, K. (2022). Deep Reinforcement Learning Techniques For Solving Hybrid Flow Shop Scheduling Problems: Proximal Policy Optimization (PPO) and Asynchronous Advantage Actor-Critic (A3C). In T. Bui (Ed.), *Proceedings of the 55th Hawaii International Conference on System Sciences*. Hawaii International Conference on System Sciences.

Nahhas, A., Krist, M., & Turowski, K. (2021). An adaptive scheduling framework for solving multi-objective hybrid flow shop scheduling problems. In T. Bui (Chair), *HICSS*.

Papadimitriou, C. H., & Tsitsiklis, J. N. (1987). The Complexity of Markov Decision Processes. *Mathematics of Operations Research*, *12*(3), 441–450.

Ren, J., Ye, C., & Yang, F. (2021). Solving flow-shop scheduling problem with a reinforcement learning algorithm that generalizes the value function with neural network. *Alexandria Engineering Journal*, *60*(3), 2787–2800.

Ribas, I., Leisten, R., & Framiñan, J. M. (2010). Review and classification of hybrid flow shop scheduling problems from a production system and a solutions procedure perspective. *Computers & Operations Research*, *37*(8), 1439–1454.

Rolf, B., Reggelin, T., Nahhas, A., Lang, S., & Müller, M. (2020). Assigning dispatching rules using a genetic algorithm to solve a hybrid flow shop scheduling problem. *Procedia Manufacturing*, *42*, 442–449.

Roy, G. M. (2017). *Rabbitmq in Depth*. Manning Publications.

Ruiz, R., & Vázquez-Rodríguez, J. A. (2010). The hybrid flow shop scheduling problem. *European Journal of Operational Research*, *205*(1), 1–18.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., & Klimov, O. (2017, July 20). *Proximal Policy Optimization Algorithms*.

Sharp, M., Ak, R., & Hedberg, T. (2018). A survey of the advancing use and development of machine learning in smart manufacturing. *Journal of Manufacturing Systems*. Advance online publication.

van der Ham, R. (2018). Salabim: open source discrete event simulation and animation in Python. In *Proceedings of the 2018 Winter Simulation Conference*.

Voudouris, C., & Tsang, E. (2003). Guided Local Search. In F. Glover & G. A. Kochenberger (Eds.), *Handbook of Metaheuristics* (Vol. 57, pp. 185–218). Springer Science & Business Media.

Wang, J [Jinzhi], Qu, S., Wang, J [Jie], Leckie, J. O., & Xu, R. (2017). Real-Time Decision Support with Reinforcement Learning for Dynamic Flowshop Scheduling. In *Smart SysTech 2017; European Conference on Smart Objects, Systems and Technologies*.

Wang, Q., Xiong, J., Han, L., sun, p., Liu, H., & Zhang, T [Tong] (2018). Exponentially Weighted Imitation Learning for Batched Historical Data. *Advances in Neural Information Processing Systems*, *31*.

Zhang, C [Chiyuan], Vinyals, O., Munos, R., & Bengio, S. (2018). *A Study on Overfitting in Deep Reinforcement Learning*.

Zhang, C [Cong], Song, W., Cao, Z., Zhang, J., Tan, P. S., & Chi, X. (2020). Learning to Dispatch for Job Shop Scheduling via Deep Reinforcement Learning. *Advances in Neural Information Processing Systems*, *33*, 1621–1632.

Zhu, J., Wang, H., & Zhang, T [Tao] (2020). A Deep Reinforcement Learning Approach to the Flexible Flowshop Scheduling Problem with Makespan Minimization. In *2020 IEEE 9th Data Driven Control and Learning Systems Conference (DDCLS)* (pp. 1220–1225). IEEE.