Towards Optimal Assembly Line Order Sequencing with Reinforcement Learning: A Case Study

1st Saad Shafiq
Johannes Kepler University
Linz, Austria
saad.shafiq@jku.at

2nd Christoph Mayr-Dorn
Johannes Kepler University
Linz, Austria
christoph.mayr-dorn@jku.at

3rd Atif Mashkoor
SCCH GmbH &
Johannes Kepler University
Linz, Austria
atif.mashkoor@{scch | jku}.at

4th Alexander Egyed
Johannes Kepler University
Linz, Austria
alexander.egyed@jku.at

Abstract—**The new era of Industry 4.0 is leading towards self-learning and adaptable production systems requiring efficient and intelligent decision making. Achieving high production rate in a short span of time, continuous improvement, and better utilization of resources is crucial for such systems. This paper discusses an approach to achieve production optimization by finding optimal sequences of orders, which yield high throughput using reinforcement learning. The feasibility of our approach is evaluated by simulating a plant modelled on a higher level of abstraction taken from a real assembly line. The applicability of the proposed approach is demonstrated in the form of code utilizing the simulation model. The obtained results show promising accuracy of sequences against corresponding throughput during the simulation process.**

Index Terms—**Order sequencing, Optimization, Reinforcement learning**

I. Introduction

Production sequencing, in general, deals with movement of different order types (e.g., a specific vehicle configuration) in a particular sequence. Each order type in an assembly line comes with varying complexity in terms of tasks required to complete the order. Knowing the constraints of limited resources, orders must be sequenced in order to maximize productivity. An arbitrary sequence of order types could potentially yield sub-optimal throughput, thus creating a bottleneck in the process. The manufacturing industry transitioning towards Industry 4.0 faces the challenge of implementing adaptive, self-learning, and efficient systems [1], [2]. In order to maximize the efficiency of these systems, new paradigms to facilitate product scheduling and sequencing need to be explored.

The recent advancements in machine learning (ML) have made its algorithms more powerful and efficient in order to solve modern day problems. Some of its subsets among many include deep learning (DL) and reinforcement learning (RL). DL [3] has progressed significantly over recent years and proven to show promising results in big industrial areas such as automotive [4], image recognition [5], and text classification [6]. RL, on the other hand, is a huge shift in the algorithmic paradigm and contains a range of existing algorithms. RL has proven to be effective, specially in simulated environments but has yet to be explored in real world scenarios [7]–[9]. RL models train on policies specifically designed for tasks focusing on solving state space problems. The policy adjudicates what action to select at a given state. A reward is given after each action the model performs. In short, the models aim to identify a policy, which lets them achieve maximum expected outcome. The increasing popularity of RL is due to the fact that it is capable of solving complex and challenging sequential decision-oriented problems.

While the evaluation in current RL research papers is limited to virtual actions made by the agent, in the real world, all the scenarios (including worst) should be taken into account in order to ensure safety [10]. Several studies already exist showing the utilization of RL in order to achieve production scheduling. Schneider et al. [11] introduced a value function to generate optimal scheduling. However, there could exist numerous variants of this formulated problem, especially, considering the fact that these are NP-complete and cannot be solved with algorithms completing in polynomial computation time. Our approach provides an alternative solution to this problem by reducing time considerably. Our approach applies to the assembly context that have the following two main constraints:

1) The order types in an assembly line have different complexity in terms of varying task durations.
2) An assembly line has a fixed tact time.

The main contribution of this study is a novel approach to provide optimal order type sequences, which could help the production managers to make cost effective decisions ahead of time. We believe that this study could be a starting point for researchers to look at the factors, which influence order sequencing, thus affecting the overall productivity of an assembly line.

Our approach introduces RL in identifying the optimal sequence given a set of orders in a production plant. In our evaluation, the modeled production plant is a real assembly line for manufacturing of vehicles, which comprises of multiple stations, tasks associated with each worker, and a given set of orders having specified features. Our approach consists of the RL variant known as Q-Learning, which helps in identifying the optimal sequences that will potentially yield high throughput by exploring the plant simulation environment. Results show that our approach is able to demonstrate optimal sequences yielding desired throughput keeping in view the constraints and complexity of the order types.

The rest of the paper is organized as follows: Section II describes the motivation of this study. Section III presents the background of RL in production optimization. Sections IV, V, and VI describe the approach, algorithm, and implementation of the proposed approach demonstrated through a case study, respectively. The study evaluation and discussion is addressed in Section VII and VIII, respectively. Threats to validity of the study are described in Section IX. Section X discusses the related work. The paper is concluded in Section XI.

II. Problem Statement (Motivation)

There is a plethora of different possible sequences of order types in an assembly line and small changes in a sequence could effect the overall throughput. Hence, brute force trying to find the optimum is not an efficient option. As the simulation of an assembly line can be represented as discrete events occurring at discrete times, RL aims to capture this by constituting the Markov decision process (MDP) and deemed to be befitting for this sort of problem, hence we have opted for RL.

Although there are many studies focusing on the applications of RL in virtual environments, there exist only a few studies that are addressing its applications in production and manufacturing systems in terms of order sequencing. In the real world, the execution of the entire scenario is an ongoing continuous process and determining the sequence keeping in mind the order constraints is crucial at the time of initialization. We aim to leverage RL in order to assist production managers to decide earlier which sequence of orders fits best under the given circumstances. For this purpose, we have formulated following research questions.

RQ-1: Does RL help in finding optimal sequences?
Rationale: This question refers to the applicability of RL in the domain of order sequencing. It would help explore the potential of using RL in detecting optimal sequences in an assembly line. The optimality in production optimization is highly subjective and with the exact goal can not be guaranteed with ML every time. However, the aim is to get as close to the desired goal as possible. Therefore, to address this, we have defined a minimum desired threshold for our RL approach based on our observation of company's processes.

RQ-2: Is a machine learning-based model able to replace a simulation?
Rationale: This question refers to the possibility of adopting a machine learning-based model instead of having a simulation. It would let us understand the factors playing a vital role in differentiating an ML model from the simulation environment. An ML model would provide the possibility to substitute long running simulations for less accurate but quicker responding models to evaluate the effect of an RL agent's decisions.

III. Background of RL in Production Optimization

The applicability of RL in production optimization has been extensively explored by industrial researchers but there are no studies found that are addressing the order sequencing problem. We have modelled a vehicle production assembly line while addressing product and feature constrains in our plant simulation. We have used SARSA algorithm - an improved version of Q-learning - in our approach. It was proposed by Sutton et al. [12]. As observed in [13], it comprises of a Q-matrix that stores information regarding states (s_t), actions (a_t) and reward (r). An action is applied on a state by the agent yielding a reward. The value of $Q(s_t, a_t)$ is updated when a transition takes place from one state-action pair to the succeeding one. As an equation, it is written as:

$$Q(s_t, a_t) \longleftarrow Q(s_t, a_t) + \alpha(r_{t+1} + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t))$$

where α denotes the rate of learning and γ represents the discount factor.

The problem of sequence optimization can be deemed as Markov decision process (MDP): The state of the system (the simulation environment) s_t can be described as current input sequence of orders, an action a_t (decision) performed by the RL agent can be described as a changing in a single order type in a sequence. The goal of the optimization is to develop a policy function π that is able to minimize cost, i.e., achieving positive reward $R(s, a, t)$ over time. It can be mathematically denoted as:

$$min_\pi = \sum_{t=1}^{t'} R(s, a, t)$$

where *R*, *s*, *a*, t' denotes the reward function, state of the simulation environment, action taken by the RL agent, and time taken to finish the last iteration, respectively.

IV. Approach

We have abstracted out the essential attributes of the production line in order to model it in the plant simulation software[1]. The attributes include maximum number of orders, their complexity based on task duration and features constraints, number of stations, and team of workers.

The steps involved in the approach are shown in Fig. 1 and are as follows:

1. The documents underwent file parser where the documents are parsed, transformed to entities, and saved to the database.

2. The information in database is then imported in the plant simulation software and used to set the initial values for the components of the simulation. The components of the simulation model are addressed in Section VII-A.

3. The initial sequence of orders (state) is updated in the Q-Matrix, then the simulation ran on that sequence of orders, getting the throughput in return.

[1]https://www.dex.siemens.com/plm/tecnomatix/plant-simulation

4. The throughput is assessed and a positive reward of +1 is assigned to the state with corresponding action if the desired throughput is achieved otherwise a negative reward of -1 is assigned to the state with the corresponding action. In our context, the states represent the order places and the agent is able to perform action by changing order at a single place in a training step. The information regarding states, action and reward is updated in Q-Matrix, which lets the agent select actions that potentially yield positive reward.

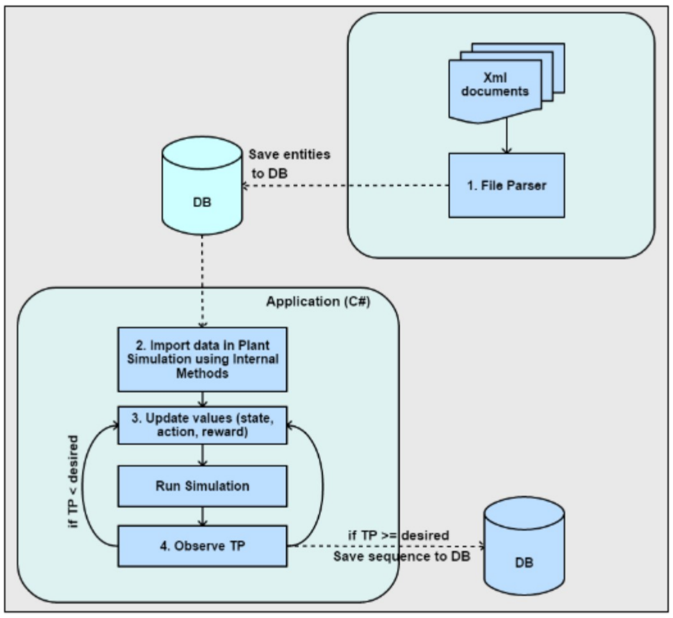


Fig. 1: Architecture of the RL approach

V. Algorithm

The RL approach takes the initial sequence of orders as the first initial action, which is determined by allocating order types of varying complexity on multiple *n* places. Then, Q-Matrix is initialized with random or specific number of states and actions. The terms *s*, *a*, and *TP* used below represent state, action, and throughput, respectively. As an output, the RL approach returns the set of optimal sequences of orders. We have represented our RL approach as a pseudo code in Algorithm 1, which is described in detail below.

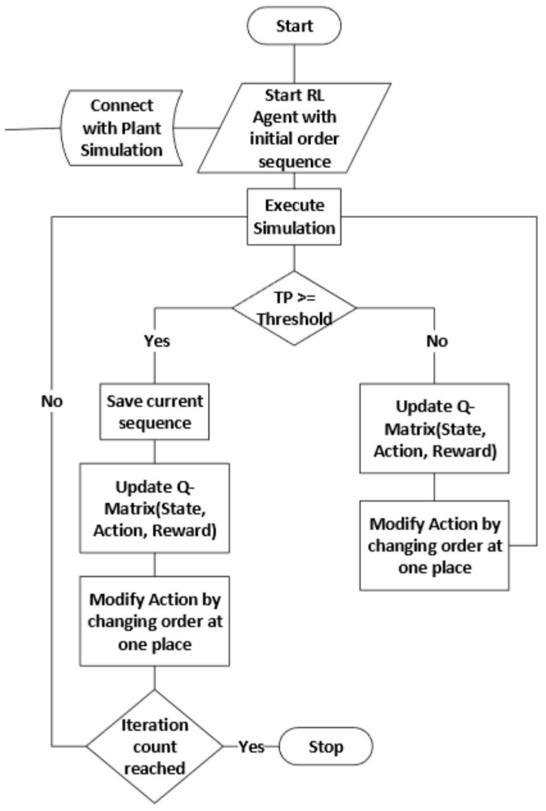
- Line 1 - Initialize the Connection with plant simulation and load the model
- Line 2 - Set the Iteration variable count to 0
- Line [3-6] - While Iteration is less than the specified max number of iterations: increment the Iteration variable, execute simulation with the initial action
- Line 7 - Set the Step variable count to 0
- Line [8-11] - While the obtained TP is less than the threshold: increment the Step variable, update Q-Matrix with current s, a, and reward. Execute simulation again with new action
- Line 12 End while loop
- Line 13 - Save the current sequence in the database
- Line 14 - Update Q-Matrix with current s, a, and reward
- Line 15 - Execute simulation with new action selected from Q-Matrix
- Line 16 End while loop.

Algorithm 1 The RL approach

```
Input
seq = {O_1....O_n}
Initial sequence of orders of length n
state = {P_1....P_n}
Define number of states
action = {A_1....A_n}
Define number of actions
Output Set of Optimal Sequences
Begin
 1: InitializeConnection(ComName)
        Connect to Plant Simulation
 2: Iteration = 0
 3: while Iteration<Max do
 4:     Iteration++
 5:     Set RL Agent with initial values
          Setting plant simulation model with initial order sequence
 6:     Execute Simulation
 7:     Step = 0
 8:     while TP<Threshold do
 9:         Step++
10:         Update Q-Matrix with current state action reward
11:         Execute Simulation with new action
12:     end while
13:     Save current sequence
14:     Update Q-Matrix with current state action reward
15:     Execute Simulation with new action
16: end while
End
```

VI. Implementation

In order to perform the experiment, we have treated the plant simulation as our agent's interacting environment. The information regarding stations, team, task duration and feature constraints are provided by the assembly company in the form of XML documents, which were later employed in building the plant simulation model. Furthermore, we have implemented the algorithm of the approach as a C# application in order to execute the training process of our RL model. The plant simulation software provides a *COM* interface to interact with the software using external applications. First, we developed the code to configure the port and loaded the plant simulation model from our application. Then, we set the initial values of the state, i.e., initial sequence of orders, minimum/desired throughput (goal to achieve), and number of iterations. A reward function R is developed which continuously updates the reward for a particular state-action pair at each step. Once the agent reaches the goal, Q-matrix is updated. The agent gets the next state, selects action (based on the Q value) to perform, and continues the same process for the next iteration until the specified limit of number of iterations is reached. The implementation of the approach's algorithm is graphically represented in Fig. 2.



Start
Connect with Plant Simulation
Start RL Agent with initial order sequence
Execute Simulation
TP >= Threshold
Yes
No
No
Save current sequence
Update Q-Matrix(State, Action, Reward)
Modify Action by changing order at one place
Update Q-Matrix(State, Action, Reward)
Modify Action by changing order at one place
Iteration count reached
Yes
Stop

Fig. 2: Workflow of the implemented algorithm

VII. Evaluation

A. Study Design

Initially, a replica model of the assembly line was built in the plant simulation software. Plant simulation is a generic software provided by Siemens which is designed to model and simulate production plant ranging from simple to complex ones. We replicated the vehicle plant using the plant simulation software covering the high level components of the plant. The model consists of single assembly line with five stations. Each station comprises of a set of tasks with feature constraints corresponding to the incoming orders. An order type can be considered as a move-able unit passing through each station and worked on by the worker assigned to each station. An order type identifies what product type (with respect to its configuration) to produce. The number of these order types is set to 10 with varying complexity with respect to the task duration. The complexity (different task times) of an order type can vary from simple (lower task times) to complex (higher task times). This is modeled by setting a fixed tact time (20 minutes) of the assembly line and keeping the task times of order types varying within the range of tact time but not exceeding it more than 2 minutes. For instance, a sequence of length 10 consists of 3 order types (simple) with task times below 20 minutes and 7 order types with task times varying between 20-22 minutes in one line a day. Similar to the fixed tact times, if an order exceeds the tact time at a given station while the prior station has completed the processing of its order, the exceeding time is recorded in a variable and propagated to the task times of the incoming order at the same station. An illustration of this scenario is demonstrated in Fig. 3.

Each sequence of orders consists of N number order types organized in a sequential manner.

$$sequence = O_1...O_n$$

Each order needs to be entertained by single worker at a time from the worker pool W. Once the worker completes the given task in a specified duration, the order then moves to the next allocated station. The time taken by each task at a particular station is maintained in a time table. Each station has a designated set of tasks along with its feature constraints. Each order is processed at each station but the time needed depends on the order type (recall that the type reflects the configuration of the product and, thus determines the detailed steps needed for mounting the respective parts). Furthermore, the fixed tact time at each station is modelled programmatically in a PlantSim *Method* component provided within the plant simulation software. The total execution time of the simulation model is 24 hours. The components of the simulation model are illustrated in Fig. 4.

B. Input Data

The input data consists of multiple XML documents. The documents are divided into two levels for a single order. A level 0 document contains designated stations, tasks and their duration, and the team assigned to stations. A level 1 document contains feature constraints associated with these tasks at each station. The information is then utilized to set the tasks times and feature constraints associated with tasks for each order in the plant simulation model.

We initially set the total number of orders types *(n = 10)*. The complexity level ranges from 1-10 with 10 being the most complex. The arrangement of orders is represented as places such as place 1 denotes the first position. There are total of 100 places in the simulation model. The order types are set randomly at each place as the initial sequence.

C. Results

Once the model is trained, it starts yielding the sequences that suffice our minimum threshold for throughput. In order to select the minimum threshold, we first simulated the model on randomly generated data of 1000 observations, which showed throughput ranging from 18-46 inferring that the change in the sequence of orders have a significant impact on the overall throughput. Moreover, the uni-variate distribution of TP in

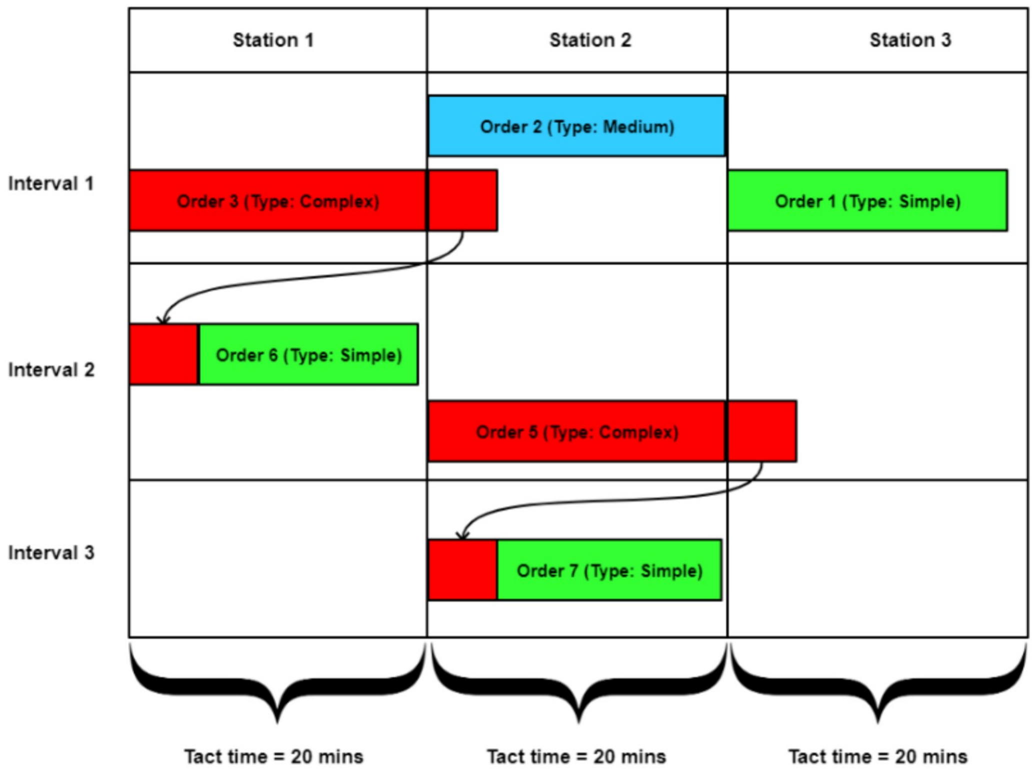


Fig. 3: Order execution scenario

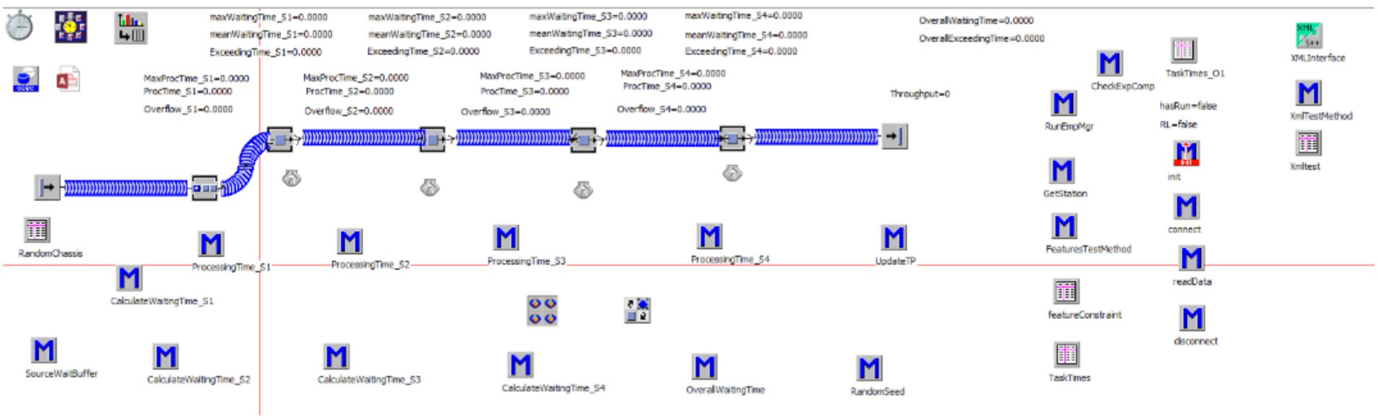

maxWaitingTime_S1=0.0000
meanWaitingTime_S1=0.0000
ExceedingTime_S1=0.0000
maxWaitingTime_S2=0.0000
meanWaitingTime_S2=0.0000
ExceedingTime_S2=0.0000
maxWaitingTime_S3=0.0000
meanWaitingTime_S3=0.0000
ExceedingTime_S3=0.0000
maxWaitingTime_S4=0.0000
meanWaitingTime_S4=0.0000
ExceedingTime_S4=0.0000
OverallWatingTime=0.0000
OverallExceedingTime=0.0000
MaxProcTime_S1=0.0000
ProcTime_S1=0.0000
Overflow_S1=0.0000
MaxProcTime_S2=0.0000
ProcTime_S2=0.0000
Overflow_S2=0.0000
MaxProcTime_S3=0.0000
ProcTime_S3=0.0000
Overflow_S3=0.0000
MaxProcTime_S4=0.0000
ProcTime_S4=0.0000
Overflow_S4=0.0000
Throughput=0
hasRun=false
RL=false

Fig. 4: Plant model

1000 observations is shown in Fig. 5 with average/mean of 29.86 and standard deviation of 4.52. We then decided to have a mean TP (30) as minimum/desired threshold in our experiment.

Evidently, the RL approach successfully identified the optimal sequences (meeting the minimum/desired output) of order types. The number of steps taken to reach to the goal in each iteration of the training process and a sample generated sequence is shown in Fig. 6. The RL agent took 66 steps (actions) in order to reach the desired throughput in the first iteration and 3 steps in the third iteration. The rest of the iterations required only one step (action) to reach the minimum/desired throughput implying high learning capability of the model. After completing 10 iterations (specified limit) the model has provided a sequence of orders yielding desired throughput. The place in the sample generated sequence represent the position of the order in an assembly line while the alphanumeric characters corresponding to each place represent the order types. As can be seen in Fig. 7, there are 10 sequences yielding throughput (31) surpassing our minimum/desired throughput (30) along with the frequency of orders (expressed as Len_Order#) in each sequence.

VIII. Discussion

A. *Answers to RQs*

RQ-1: Does RL help in finding optimal sequences?

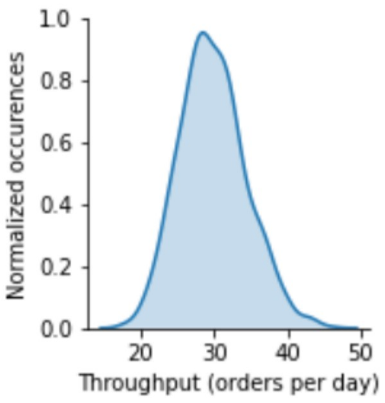
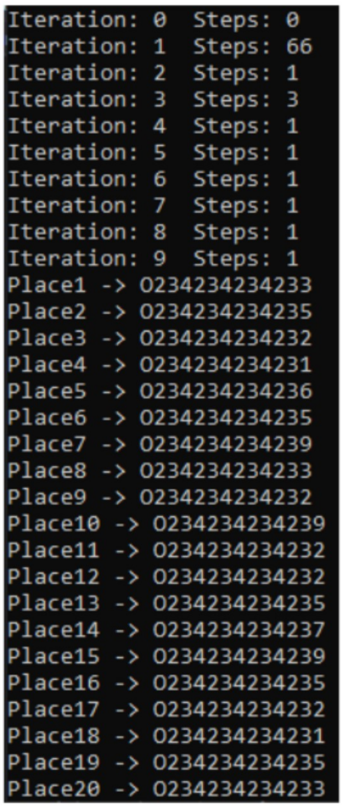


Fig. 5: TP uni-variate distribution (1000 observations)


```
Iteration: 0  Steps: 0
Iteration: 1  Steps: 66
Iteration: 2  Steps: 1
Iteration: 3  Steps: 3
Iteration: 4  Steps: 1
Iteration: 5  Steps: 1
Iteration: 6  Steps: 1
Iteration: 7  Steps: 1
Iteration: 8  Steps: 1
Iteration: 9  Steps: 1
Place1 -> 0234234234233
Place2 -> 0234234234235
Place3 -> 0234234234232
Place4 -> 0234234234231
Place5 -> 0234234234236
Place6 -> 0234234234235
Place7 -> 0234234234239
Place8 -> 0234234234233
Place9 -> 0234234234232
Place10 -> 0234234234239
Place11 -> 0234234234232
Place12 -> 0234234234232
Place13 -> 0234234234235
Place14 -> 0234234234237
Place15 -> 0234234234239
Place16 -> 0234234234235
Place17 -> 0234234234232
Place18 -> 0234234234231
Place19 -> 0234234234235
Place20 -> 0234234234233
```
Fig. 6: Sample generated sequence over 10 iterations

Results of our experiment show that our RL approach worked well in exploring the simulation environment and found sequences surpassed our threshold in a considerably short span of time. Although, the approach might still needs to be evaluated on very large and complex production lines, this study can be a building block to further research on employing RL in production optimization.

RQ-2: Is a machine-learning based model able to replace a simulation?

In order to address this RQ, we aimed to leverage ML to predict the overall throughput of the plant for a given sequence of orders. We have generated the simulation data from the plant simulation. The data represents the sequence of orders being processed along with the throughput produced over the course of one day. We ran the experiments with order types of length N=100 with randomly generated sequences, which ultimately provided the throughput against each corresponding sequence. We then used this information to train our Keras MLP (multi layer perceptron) regressor model[2]. The model is used to forecast values based on the trained data.

We have trained the model on the dataset considering the sequence of orders as input and throughput value as the label (output) in the training process. The sequence of orders (categorical) are converted and mapped to integer values 0,1 using one-hot encoding as it helps model to make better predictions. We have considered this problem as a regressing problem due to the fact that we want the model to forecast predictions based on the given input sequence.

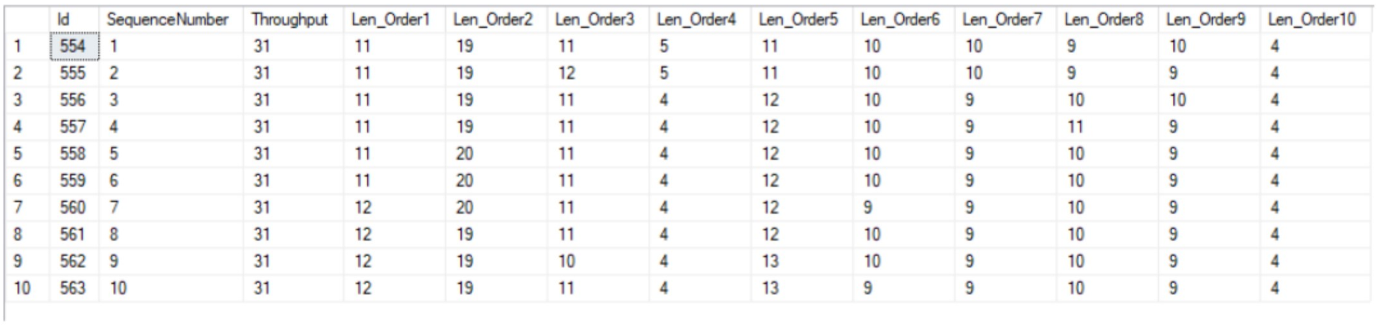
To evaluate the model, we have used MAE (mean absolute error) as an evaluation metric. MAE denotes the amount of errors in a set of predictions made by the model. Moreover, we have used K-fold cross-validation to evaluate the performance of the model. The K-fold cross-validation process is a mean to evaluation model's performance by splitting the dataset into k folds of train and test sets. This process is adopted in order to avoid over-fitting of the model. The MAE for 5 folds turns out to be 8.73 (mean) and 0.28 (standard deviation). After the selection of best model from the k-fold cross-validation process, the model was then validated on a held-out validation set. The validation results showed minimal loss (10.24) inferring a moderate prediction capability of the model. However, the model prediction duration is similar to the plant simulation executing the actual sequence of orders (products) but has an error rate of 8.73 when compared to simulation results implying less accuracy as compared to the simulation itself. Hence, evidently, the ML model used in this study is not yet capable of replacing the simulation itself, but the results may vary depending on the complexity of the simulation model for the assembly line. A more complex simulation model could require to load external data, which could lead to large execution times thus creating a need to replace it with a machine learning model with less prediction time.

B. Practical Implications

Our approach could be of benefit to the plant managers in making decisions concerning the order (product) sequence or the dynamic generation of efficient sequences of orders (products). Especially, where the desired throughput is of high priority, this approach can help yield the most suitable sequences. Moreover, the plant simulation could help in identifying the potential bottlenecks in the model providing plant managers information to better orient their assembly components.

Our RL approach still needs to be evaluated on large and complex production plants where the number of states

[2]https://keras.io



	Id	SequenceNumber	Throughput	Len_Order1	Len_Order2	Len_Order3	Len_Order4	Len_Order5	Len_Order6	Len_Order7	Len_Order8	Len_Order9	Len_Order10
1	554	1	31	11	19	11	5	11	10	10	9	10	4
2	555	2	31	11	19	12	5	11	10	10	9	9	4
3	556	3	31	11	19	11	4	12	10	9	10	10	4
4	557	4	31	11	19	11	4	12	10	9	11	9	4
5	558	5	31	11	20	11	4	12	10	9	10	9	4
6	559	6	31	11	20	11	4	12	10	9	10	9	4
7	560	7	31	12	20	11	4	12	9	9	10	9	4
8	561	8	31	12	19	11	4	12	10	9	10	9	4
9	562	9	31	12	19	10	4	13	10	9	10	9	4
10	563	10	31	12	19	11	4	13	9	9	10	9	4

Fig. 7: Top 10 sequences

and actions can be quite large and could result in potential state explosion problem. Due to the limitation of libraries available in C# for deep RL and integration with our plant simulation model, we were unable to use some of the powerful deep RL techniques available such as Google's DeepMind Deep Q Network[3] (DQN). However, we believe, that modern variants of RL such as Google's DeepMind Deep Q Network (DQN) could potentially address this problem and can provide scalability.

IX. Threats to validity

A. External validity

To reduce the external validity threat, we have used a significant sized real case study and a simulation environment in order to train our reinforcement learning model. Although the domain specific constraints of plant do limit this study to be applicable in other domains, the approach with slight tailoring can still be applicable in similar production plants.

B. Internal validity

The case study we have employed and the data generated by the simulation is based on the documents provided by the actual manufacturer of the plant, which eliminates the threat of internal validity.

C. Construct validity

The experiment performed in this study is aimed at solving a specific problem of sequence optimization for an assembly line. The factor may vary depending on the type and complexity of the assembly line. However, our approach tends to generate optimal sequences that are applicable to all assembly lines sharing similar environment as in our case study.

D. Conclusion validity

While the results are still debatable due to the limitation of benchmark studies, we believe that this study could embark a new direction in the area of industrial research in production sequence optimization.

[3]https://deepmind.com/blog/article/deep-reinforcement-learning

X. Related work

Many studies have been conducted on production scheduling, which address the optimization problem using RL such as [11], [14]–[16]. The results of these studies have showed great success of RL agents for maintenance scheduling of machines in a transfer line. Zhang et al. [17] use RL in a combination with neural network in order to optimize a job shop scheduling, whereas a recent study [2] has employed new variants of RL, such as Deep Q learning by Google, to solve a complex job production and scheduling problem.

Bierwirth et al. [18] used a genetic algorithm (GA) to address job shop scheduling problem. In job shop scheduling, all jobs have certain release times, which ultimately determines their arrival time at the shop floor. The release time is non-deterministic and may require re-scheduling of jobs after regular interval. Results show that the proposed approach reduced the mean-flow time of jobs at a reasonable time.

Dulac-Arnold et el. [10] discussed the challenges in applying RL in various domains such as multi dimensional state spaces, constraints regarding safety and security [19], [20], and generating real time inference in production systems. The study also demonstrated the real world challenges using an example environment and highlighted different ways to address these challenges.

Wang et al. [21] investigated the capability of a RL agent for the selection of rules for single machines. The results show that the RL agent implemented using Q-learning - a variant of RL - is able to select best suitable rules which ultimately meets single machine objectives.

Akyol et al. [22] explored different studies proposing artificial neural network (ANN) based approaches aimed at solving production scheduling problems. The study results reveal that hybrid approaches - used in conjunction with one another - were mostly aimed at solving the job shop scheduling problem, which is considered to be one of the hardest optimization problems experienced in real scheduling environments.

As stated in previous studies, these problems are considered to be NP-complete, thus a generalized solution is difficult to produce. Therefore, we have explored the problem specific to our case study, which can be applied to similar order sequencing problems. To the best of our knowledge, this study is first of its kind that deals with order sequence optimization using RL. This study further explores the applicability of the approach in sequence optimization through interacting with the production-grade plant simulation environment.

XI. Conclusion

In this study, we have presented an approach for sequence optimization of production plants. We have introduced RL to identify the best possible sequences of orders types in a production line. We have demonstrated the applicability of our approach by modeling a real vehicle manufacturing assembly line. The results of the study show that the approach was able to successfully identify the best possible sequences of the product (ranging from simple to complex) in a short span of time. Moreover, we also explored and compared the proposed approach with a machine learning approach of predicting the potential throughput for a given sequence of orders types. However, the predictions made by the model were with significantly low MAE. The time taken by the model to generate predictions was similar to the plant simulation software with MAE of 0 in ideal case. To summarize, our RL approach is able to spot sequences yielding minimum specified throughput in considerably less time.

In the future, we intend to put this approach into practice to further evaluate our RL model. We also intend to explore the impact of task switching on the productivity. We believe that optimizing tasks using machine learning algorithms could potentially enhance the performance and efficiency of a plant or assembly line. We also plan to extend our approach using an improved RL variant, such as Deep Q Networks, in order to scale it to complex or large scale production plants.

Acknowledgment

This work is partially supported by the Linz Institute of Technology (The LIT Artificial Intelligence Lab and the LIT Secure and Correct Systems Lab) and the FFG, Contract No. 854184. Pro^{2}Future and SCCH are funded within the Austrian COMET Program — Competence Centers for Excellent Technologies — under the auspices of the Austrian Federal Ministry of Transport, Innovation and Technology, the Austrian Federal Ministry for Digital and Economic Affairs and of the Provinces of Upper Austria and Styria. COMET is managed by the Austrian Research Promotion Agency FFG.

References

[1] A. I. Orhean, F. Pop, and I. Raicu, "New scheduling approach using reinforcement learning for heterogeneous distributed systems," *Journal of Parallel and Distributed Computing*, vol. 117, pp. 292–302, 2018.

[2] B. Waschneck, A. Reichstaller, L. Belzner, T. Altenmüller, T. Bauernhansl, A. Knapp, and A. Kyek, "Optimization of global production scheduling with deep reinforcement learning," *Procedia CIRP*, vol. 72, pp. 1264–1269, 2018.

[3] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, may 2015. [Online]. Available: http://www.nature.com/articles/nature14539

[4] F. Falcini, G. Lami, I. Science, A. M. Costanza, and F. C. Automobiles, "Deep Learning in Automotive Software," *IEEE Software*, 2017.

[5] C. Chen, O. Li, C. Tao, A. J. Barnett, J. Su, and C. Rudin, "This Looks Like That: Deep Learning for Interpretable Image Recognition," *33rd Conference on Neural Information Processing Systems (NeurIPS 2019)*, no. NeurIPS, pp. 1–12, 2018. [Online]. Available: http://arxiv.org/abs/1806.10574

[6] M. Choetkiertikul, H. K. Dam, T. Tran, T. Pham, A. Ghose, and T. Menzies, "A Deep Learning Model for Estimating Story Points," *IEEE Transactions on Software Engineering*, vol. 45, no. 7, pp. 637–656, 2019.

[7] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, "Human-level control through deep reinforcement learning," *Nature*, vol. 518.7540, pp. 529–533, 2015.

[8] D. Hafner, T. Lillicrap, I. Fischer, R. Villegas, D. Ha, H. Lee, and J. Davidson, "Learning Latent Dynamics for Planning from Pixels," in *36th International Conference on Machine Learning*, 2019.

[9] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, and D. Hassabis, "Mastering the game of Go with deep neural networks and tree search," *Nature*, vol. 529, no. 7587, pp. 484–489, 2016. [Online]. Available: http://dx.doi.org/10.1038/nature16961

[10] G. Dulac-Arnold, D. Mankowitz, and T. Hester, "Challenges of Real-World Reinforcement Learning," in *36th International Conference on Machine Learning*, 2019. [Online]. Available: http://arxiv.org/abs/1904.12901

[11] J. G. S. J. A. B. A. W. Moore and Jeff G. Schneider; Justin A. Boyan; Andrew W. Moore, "Value Function Based Production Scheduling," in *ICML '98: Proceedings of the Fifteenth International Conference on Machine Learning*, 1998, pp. 522–530.

[12] R. S. Sutton and A. G. Barto, *An introduction to reinforcement learning*, 2nd ed. London: Cambridge: MIT Press, 1998.

[13] Y. H. Wang, T. H. S. Li, and C. J. Lin, "Backward Q-learning: The combination of Sarsa algorithm and Q-learning," *Engineering Applications of Artificial Intelligence*, vol. 26, no. 9, pp. 2184–2193, 2013. [Online]. Available: http://dx.doi.org/10.1016/j.engappai.2013.06.016

[14] S. Mahadevan, N. Marchalleck, T. K. Das, and A. Gosavi, "Self-improving factory simulation using continuous-time average-reward reinforcement learning," in *Proc 14th International Conference on Machine Learning*, no. August, 1997, pp. 202–210. [Online]. Available: http://www-anw.cs.umass.edu/rlr/domains.html

[15] S. Mahadevan and G. Theocharous, "Optimizing Production Manufacturing using Reinforcement Learning," *Proceedings of the Eleventh International Florida Artificial Intelligence Research Society Conference*, no. Gershwin 1994, pp. 372–377, 1998.

[16] S. Riedmiller and M. Riedmiller, "A neural reinforcement learning approach to learn local dispatching policies in production scheduling," *IJCAI International Joint Conference on Artificial Intelligence*, vol. 2, pp. 764–769, 1999.

[17] W. Zhang and T. G. Dietterich, "A Reinforcement Learning Approach to Job-shop Scheduling," *1995 International Joint Conference on Artificial Intelligence*, pp. 1114–1120, 1995.

[18] C. Bierwirth and D. C. Mattfeld, "Production Scheduling and Rescheduling with Genetic Algorithms," *Evolutionary Computation*, vol. 7, no. 1, pp. 1–17, 1999.

[19] M. Biro, A. Mashkoor, J. Sametinger, and R. Seker, "Software Safety and Security Risk Mitigation in Cyber-physical Systems," *IEEE Software*, vol. 35, no. 1, pp. 24–29, 2017.

[20] A. Mashkoor, J. Sametinger, M. Biro, and A. Egyed, "Security- and safety-critical cyber-physical systems," *Journal of Software: Evolution and Process*, vol. 32, no. 2, pp. 1–2, 2020.

[21] Y. C. Wang and J. M. Usher, "Application of reinforcement learning for agent-based production scheduling," *Engineering Applications of Artificial Intelligence*, vol. 18, no. 1, pp. 73–82, 2005.

[22] D. E. Akyol and G. M. Bayhan, "A review on evolution of production scheduling with neural networks," *Computers and Industrial Engineering*, vol. 53, no. 1, pp. 95–122, 2007.