Identification of Superior Improvement Trajectories for Production Lines via Simulation-Based Optimization with Reinforcement Learning

Günther Schuh, Andreas Gützlaff, Matthias Schmidhuber, Jan Maetschke, Max Barkhausen, Narendiran Sivanesan

Identification of superior improvement trajectories for production lines via simulation-based optimization with reinforcement learning

Günther Schuh[1], Andreas Gützlaff[1], Matthias Schmidhuber[1], Jan Maetschke[1,*], Max Barkhausen[2], Narendiran Sivanesan[2]

[1] Laboratory for Machine Tools and Production Engineering (WZL) at RWTH Aachen University, Campus-Boulevard 30, Aachen, Germany
[2] 654 Manhattan Avenue, 11222 Brooklyn, NY, United States of America
*j.maetschke@wzl.rwth-aachen.de

Abstract. An increasing variety of products contributes to the challenge of efficient manufacturing on production lines, e.g. in the Fast Moving Consumer Goods (FMCG) sector. Due to the complexity and multitude of adjustment levers, the identification of economic actions for improvement is challenging. Reinforcement learning offers a way to deal with such complex problems with little problem-specific adaptation. This paper presents a method for decision support for economic productivity improvement of production lines. A combination of discrete event simulation and reinforcement learning is used to identify efficient, sequential trajectories of improvements. The approach is validated with a fill-and-pack line of the FMCG industry.

Keywords: Manufacturing System, Production Line, Machine Learning, Simulation

1 Introduction

Manufacturing companies are facing customer demand for high quality products in a large number of variants. The resulting small batch sizes and frequent product changes lower the average Overall Equipment Effectiveness (OEE) [1] especially for companies that manufacture at high speed on production lines, such as the Fast Moving Consumer Good (FMCG) industry [2, 3]. Moreover, some companies allocate products in production networks back to western countries with high automation available due to a higher standard of digitalization [4]. This leads to consolidation and hence to increased planned utilization of production lines. As a result, the demands on the productivity and stability of production lines are increasing. Thus a growing focus on increasing the OEE can be observed within the industry sector leading to raising attention in research as well [5, 6].

The configuration of the production system, consisting of several machines, buffers, conveyors, etc., has a fundamental influence on its productivity and stability, and therefore on the OEE and ultimately on the production costs [7]. Improving such systems is a complex problem, the complexity of which increases dramatically with the number of machines. The buffer allocation subproblem alone is an NP-hard problem [8, 9].

To meet this challenge, Discrete Simulation-Based Optimization (DSBO) is widely used in the industry to improve the configuration of production lines because most real-world problems cannot be solved analytically due to their stochastic nature [8, 10]. However, studies show that companies need more help in conducting simulation studies, interpreting the results and deriving feasible step-by-step actions from them [11, 12].

Additionally, the identification of effective adjustment levers to improve brownfield production lines is challenging because the restraining element of the system shifts dynamically, due to the mutual dependencies of the systems elements. Because of this, and because different actions incur different costs, the identification and prioritization of efficient actions for improvement only make sense by considering the overall system behavior and costs, not only by focusing on the bottleneck-orientated OEE [5, 13, 14].

In the context of Cyber Production Management Systems (CPMS) [15], giving decision support in such complex but well-defined optimization problems, artificial intelligence (AI) methods, especially reinforcement learning (RL), are receiving more and more attention [16, 17]. The motivation for applying RL is that the RL agent learns to react efficiently to the dynamics of the environment, without any prior knowledge of the system dynamics. [18].

This paper presents a method for improving the productivity of production lines efficiently using RL. The aim is not to find an optimal configuration but to discover trajectories of sequential improvements, which could be interpreted and implemented step by step and thus create the basis for practical decision support.

This paper is structured as follows: In section 2, the advantages of using RL for optimizing production lines are presented. In Section 3, a comprehensive literature review on the state of the art is presented and the challenges of improving production lines are described. In section 4, the methodology and experimental setup considering a fill-and-pack line of a FMCG manufacturer are described. The findings are discussed in Section 5.

2 Chances of optimizing production lines using RL

Discrete-event simulations (DES) are suitable for the evaluation of complex, stochastic systems, where a closed-form mathematical model is hard to find. Although simulation is not an optimization technique on its own, it can be combined with complementary optimization to improve real-world systems effectively by integrating in metaheuristics [19]. Instead of using the simulation as a black box solution, it is advisable to closely integrate optimization with simulation, statistically extract information from existing simulation runs to guide the parameter-search [10, 20]. Thus, metaheuristics may need to be adapted to the specific characteristics of the problem [20, 21]. The same applies to specific mathematical models.

For this reason, more and more approaches use AI for optimization in combination with simulation [12]. What makes RL an attractive solution candidate is that it does not require holistic a-prioiri knowledge of the problem or a dedicated mathematical model

of the target production setup. RL is model-free in the sense that the RL agent learns about its environment simply by interacting with it [19].

RL can be understood as learning from a sequence of interactions between an *agent* and its *environment*, where the agent learns how to behave in order to achieve a goal [18]. The default formal framework to model RL problems is the Markov Decision Processes (MDP), a sequential decision process modeled by a state space, an action space, and transition probabilities between states and rewards [18, 19].

In an MDP, the agent acts based on observations of the *states* of the environment – in our case, these are the observations returned by the DES. The *rewards* received by the agent are the basis for evaluating these choices. The agent learns a *policy*, which may be understood as a function from state observations to actions. The agent's objective is to maximize the future cumulative discounted reward received over a sequence of actions [18].

Below, we will explain how improving a production line sequentially can be modeled as an MDP. This said, several challenges for the simulation-based optimization of production lines exist, which will be discussed in the following section.

3 State of the art

The improvement of production lines has been the subject of research for decades. Due to the large number of publications, only the most relevant and recent approaches are listed here. For more detailed references, the reader is advised to refer to [8, 12, 22, 23]. The approaches can be roughly divided into analytical or mathematical models and simulation-based approaches [23]. Even though analytical approaches usually cannot cover the complexity of real use cases [8, 10, 19], there are a variety of specific analytical models for subproblems [24, 25]. Especially the optimization of buffer allocation has received much attention from researchers [22], such as in [9, 26]. There are further approaches, which solve other specific subproblems. For example, [25] designs an algorithm for the optimization of split and merge production and [27, 28] focus on downtime reductions (affected by Mean-Time-to-Repair).

However, since the combination of several small actions on different machines is expected to yield higher efficiency gains than major improvement on single machines [29], an isolated consideration of subproblems is therefore of limited benefit. [8, 13, 14] also argue, that due to the complex dependencies, optimization is only possible by considering the entire system and not by focusing on improvement actions on the bottleneck. At the same time, after a certain point, optimizing cycle times is more economical than further improving the availability of all machines [14].

[2, 30] explicitly consider fill-and-pack lines in the FMCG industry. However, they do not present an optimization approach, but rather simulation case studies. On the other hand, they underline the potential of optimizing such lines and show the need for a combined consideration of improvement costs and increased productivity.

[13, 23] show that without considering the overall system, prioritizing improvement activities such as maintenance activities is not advisable and that this is not adequately

addressed in the literature. None of the approaches listed systematically considers improvement trajectories, i.e. a sequence of independently realizable actions to improve a production system. Rather, they focuses on finding an (near-) optimal overall solution rather than looking at the path to get there, i.e. the improvement trajectories.

[12] gives an overview of DSBO approaches in manufacturing in general and shows that machine learning approaches for optimizing production systems are getting more and more attention in research. [10] sees the need for further research combining statistical learning in combination with DSBO. [16] predicts a vast increase in the importance of automated decisions based on AI in production management.

4 Methodology and experimental setup

The methodology to identify superior improvement trajectories for production lines via DSBO with RL consists of three steps. First, we describe the used DES. Second, we introduce the MDP formulation to apply RL in order to optimize the simulation. Finally, we address the algorithm used to learn this MDP.

The underlying research hypothesis is that by making available observations $X(s)$ on the status of the production line, and by supplying the agent with the profit of an action (reward R_a), the agent is able to learn policies which lead to efficient improvement trajectories for production lines.

The overall goal is to discover *interpretable and implementable* trajectories of parameter changes, including those affecting availability and cycle time, starting from the status quo of the production line and leading to increased productivity efficiently. This is measured by the profit resulting from the additional products produced and the costs of the necessary improvement actions. Hence, the goal is not only finding (near-)optimal parameter sets, which could be harder to interpret and unrealistic to implement in a real-world setting.

4.1 Discrete Event Simulation (DES)

We consider a simplified fill-and-pack-line of the FMCG industry, consisting of four machines (blower, filler, case packer, shrinker) linearly linked by accumulative conveyors. The third machine, the case packer, combines six of the produced goods with one tray (packaging material) from a case erector (see Fig. 1).

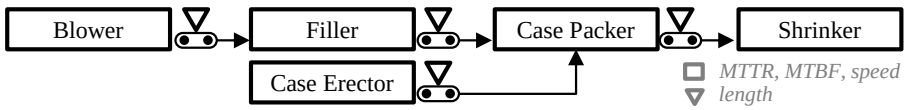

Fig. 1. Modelled production line with five machines and four conveyors

The failure probabilities of the machines are described with Mean-Time-To-Repair (MTTR) and Mean-Time-Between-Failures (MTBF), using an Erlang distribution [31]. The cycle times of the machines are given in pieces/min. The second machine, the filler, represents the bottleneck in terms of cycle time. The speed of the conveyors is not

changed, but it is always higher than the surrounding machines. Due to the constant size of the products, the buffer size is determined by the length of the conveyor. Therefore, 19 parameters of four different types (5x MMTR, MTBF, speed and 4x length, see Fig.1) result in the state $s \in S$ (see section 4.2).

The simulation logs the time that each machine stands still due to its own failure (failure time), as well as the time that a machine is blocked due to a downstream failure (blockage time). In addition, the blockage time of the conveyors is logged. These 14 times are added to the observation vector $X(s)$ and are thus seen by the agent (see section 4.2). Each run of the simulation is 500 minutes.

For each adjustment of a parameter resp. each improvement measure, machine-specific costs are incurred. These costs are based on assumptions, which depend on the application-specific context, as in [14]. To calculate the earnings from an improvement measure, the quantity of additional goods produced is set in relation to the speed of the bottleneck machine, as proposed by [1]. An increase of this relation is equated with correspondingly positive earnings. The profit for the company is the difference between the earnings from the increased output and the costs for the corresponding improvement action. This profit forms the reward R_a (see section 4.2). For a reinforcement learning agent, this reward function incentivizes profit maximizing behavior.

The simulation environment used is a customized version of the DES ManPy based on SimPy [32].

4.2 Framework for reinforcement learning: Formulation of MDP

The standard formal framework for reinforcement learning problems is a MDP, as mentioned above [18]. An MDP is a 4-tuple (S, A_s, P_a, R_a), where S is a set of states called the *state space*, $A_s$is a set of actions in the state $s \in S$ called the *action space*, $P_a$is the *probability* that at time t action $a \in A_s$ in state $s \in S$ will lead to state s' at time $t + 1$, R_a is the immediate or expected *reward* received for transitioning from state $s \in S$ to state s' by taking the action $a \in A_s$. Importantly, an MDP is characterized by the Markov property, meaning that transition probabilities between states and states, and between state-action pairs and rewards, depend only on the present state, but not on past states.

In this case, the set of states is a subset of $\mathbb{R}^{20}$ where the first 19 entries are observations from the line simulation (see Section 4.1) and the last entry is the time step given by an integer. Each state s yields an observation vector $X(s)$ composed of the 19 parameters, the failure and blockage time for each element, the costs of the last improvement action, and an integer entry, which counts the steps of the optimization procedure. The action space is parametrized by a discrete and a continuous parameter, where the first represents which machine and which parameter to manipulate next, and the latter represents the delta to the current parameter of the machine currently being optimized. State transitions are governed by the stochastic simulation and the deterministic cost function for the improvement actions resp. parameter changes. The reward is the change in profit output by the simulation (as defined in section 4.1) in going from state $s \in S$ to state $s' \in S$ via action $a \in A_s$. The resulting MDP has the Markov property for the obvious reason that the results of a simulation run, and hence the reward and next state,

are only affected by properties of the simulation and the present configuration, but not on any configurations of previous simulation runs.

An episode, i.e. a possible improvement trajectory, consists of five improvement opportunities. That is, the agent is allowed to change five parameters in sequence, choosing both the parameter to change next, and the delta.

4.3 Training the agent

There are multiple ways to learn a successful policy, i.e. a probability distribution over action given state observations; see [18] for an introduction. An important class of methods, which proved to be very successful by using *deep neural networks*, are so-called *policy gradients* [33]. The goal of policy gradient methods is to learn a probability distribution over actions, given a state observation $X(S)$, such that future discounted rewards are maximized. In order to choose an action given a state observation (in our case, an action changing the configuration of the production line) at a step in time t, the agent samples from a probability distribution parametrized by the outputs of the neural network. Policy gradient methods all have in common that the agent learns to optimize future expected rewards at each step, in formulae, $\mathrm{E}[G_t|s,\pi] = \mathrm{E}[R_t + \gamma * G_{t+1}|s,\pi]$, where G_t is the total future reward received by the agent from time t on, and $\gamma \epsilon [0;1]$ is a discount factor for future rewards. We set $\gamma = 0.9$. In practice, this means that at $t = 0$, present and future rewards $r_0 .. r_4$ are discounted at rates $< 1, 0.9, 0.81, 0.72, 0.65 >$. Unlike in a continuous MPD, in a finite-horizon MDP like ours, the expected value of total future rewards for the agent is finite even if the $\gamma = 1$, i.e. even if no discount is applied to future rewards. But as is often the case, we have found that, by applying a significant discount to future rewards and hence by incentivizing more greedy behavior, convergence is achieved much more quickly.

In this setup, we use *Proximal Policy Optimization (PPO)*, a class of algorithms developed at OpenAI, which yields state-of-the-art results for a wide range of problems. The main motivation for using PPO as opposed to classical Policy Gradient methods lies in the fact that the latter are very sensitive to choices of step size, whereas PPO adapts the step size by taking into account the Kullback-Leibler divergence of the previous policy and the updated policy. [34]

5 Application and key findings

Performing the described methodology with the set-up outlined, it can be observed, that the reward over the training episodes increases on average, i.e. the agent learned optimizing the reward and thus the production line considering the complex dependencies and the assumed costs without any prior knowledge of the system (s. Fig. 2a). It can be further stated that the presentation of the observations of the line (blockage times) to the agent accelerates the learning and improves the achieved reward.

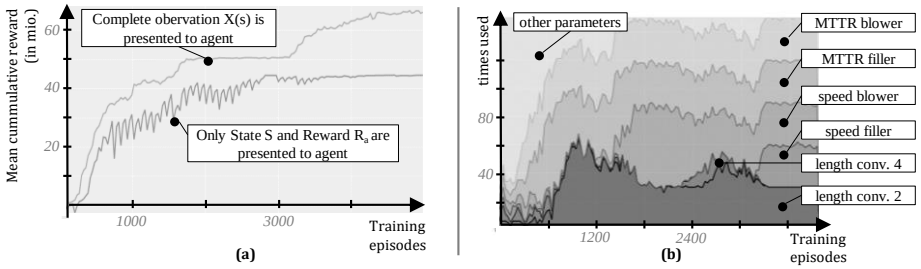

Fig. 2. Cumulative Reward (left) and changed parameters (right) over training episodes

Fig. 2b shows that in the beginning, parameters are changed randomly (note that parameters are combined as *other*), but as the training progresses, a few parameters (shown on the very right of Fig.2b) are selected significantly more often, as these are the most successful in terms of increasing productivity for the modelled production line. The most successful trajectory found consists of adjustments to reduce the MTTR of the filler, the reduction of MTTR of the blower, the increase of the speed of the filler, the adjustment of the conveyor 2 and the speed of the blower. This trajectory earns a reward of 64.3 Mio. €.

The advantage of this approach is that several trajectories with different combinations of parameters but comparable rewards were found. These interpretable improvement trajectories can thus be used for step-by-step decision support in the optimization process of production lines to prioritize alternative improvement actions and their combination. In this way, the use of DSBO becomes more easy for the user to interpret and thus more practical.

6 Conclusion and further research

In this paper, a methodology for identifying alternative improvement trajectories for production lines with RL has been presented. An RL agent with policy gradient method was able to learn policies from a DES and generate alternative trajectories without a-priori knowledge. Thus, a practical and interpretable assistance for the prioritization of improvement actions is presented. The promising results motivate further research. A fixed budget for an optimization could be specified and given to the agent as another constraint. Additionally production knowledge in the form of heuristics could be added to the RL agent to further improve the quality of trajectories or reduce computational effort. An extension of the validation to a larger industrial use case in combination with the comparison with other optimization methods like [8] is intended.

Acknowledgment

The authors would like to thank the German Research Foundation DFG for funding this work within the Cluster of Excellence “Internet of Production” (Project ID: 390621612).

References

1. Nakajima, S.: Introduction to TPM. Total productive maintenance. Productivity Press, Cambridge, Mass. (1988)
2. Bartkowiak, T., Ciszak, O., Jablonski, P., Myszkowski, A., Wisniewski, M.: A Simulative Study Approach for Improving the Efficiency of Production Process of Floorboard Middle Layer. Springer International Publishing, Cham (2018)
3. Bech, S., Brunoe, T.D., Nielsen, K., Andersen, A.-L.: Product and Process Variety Management: Case study in the Food Industry. Procedia CIRP (2019)
4. Butollo, F.: Digitalization and the geographies of production: Towards reshoring or global fragmentation? Competition & Change (2020)
5. Andersson, C., Bellgran, M.: On the complexity of using performance measures: Enhancing sustained production improvement capability by combining OEE and productivity. Journal of Manufacturing Systems (2015)
6. Corrales, Lisbeth Del Carmen Ng, Lambán, M.P., Hernandez Korner, M.E., Royo, J.: Overall Equipment Effectiveness: Systematic Literature Review and Overview of Different Approaches. Applied Sciences (2020)
7. Koren, Y., Hu, S.J., Weber, T.W.: Impact of Manufacturing System Configuration on Performance. CIRP Annals (1998)
8. Yegul, M.F., Erenay, F.S., Striepe, S., Yavuz, M.: Improving configuration of complex production lines via simulation-based optimization. Computers & Industrial Engineering (2017)
9. Xi, S., Chen, Q., MacGregor Smith, J., Mao, N., Yu, A., Zhang, H.: A new method for solving buffer allocation problem in large unbalanced production lines. International Journal of Production Research (2020)
10. Rabe, M., Deininger, M., Juan, A.A.: Speeding up computational times in simheuristics combining genetic algorithms with discrete-Event simulation. Simulation Modelling Practice and Theory (2020)
11. Karlsson, I.: An interactive decision support system using simulation-based optimization and knowledge extraction. Dissertation Series. Doctoral thesis, monograph, University of Skövde. http://urn.kb.se/resolve?urn=urn:nbn:se:his:diva-16369 (2018)
12. Trigueiro de Sousa Junior, W., Barra Montevechi, J.A., Carvalho Miranda, R. de, Teberga Campos, A.: Discrete simulation-based optimization methods for industrial engineering problems: A systematic literature review. Computers & Industrial Engineering (2019)
13. Ylipää, T., Skoogh, A., Bokrantz, J., Gopalakrishnan, M.: Identification of maintenance improvement potential using OEE assessment. Int J Productivity & Perf Mgmt (2017)
14. Wu, K., Zheng, M., Shen, Y.: A generalization of the Theory of Constraints: Choosing the optimal improvement option with consideration of variability and costs. IISE Transactions (2020)
15. Burggraf, P., Wagner, J., Koke, B.: Artificial intelligence in production management: A review of the current state of affairs and research trends in academia. IEEE, Piscataway, NJ (2018)
16. Burggräf, P., Wagner, J., Koke, B., Bamberg, M.: Performance assessment methodology for AI-supported decision-making in production management. Procedia CIRP (2020)
17. Gosavi, A.: Solving Markov Decision Processes via Simulation. Handbook of simulation optimization. In: vol. 216
18. Sutton, R.S., Barto, A.: Reinforcement learning. An introduction. Adaptive computation and machine learning. The MIT Press, Cambridge, MA, London (2018)
19. Gosavi, A.: Simulation-Based Optimization. Parametric Optimization Techniques and Reinforcement Learning. Springer, New York **55** (2015)

20. Juan, A.A., Faulin, J., Grasman, S.E., Rabe, M., Figueira, G.: A review of simheuristics: Extending metaheuristics to deal with stochastic combinatorial optimization problems. Operations Research Perspectives (2015)
21. Hubscher-Younger, T., Mosterman, P.J., DeLand, S., Orqueda, O., Eastman, D.: Integrating discrete-event and time-based models with optimization for resource allocation. IEEE, [Place of publication not identified] (2012)
22. Tempelmeier, H.: Practical considerations in the optimization of flow production systems. International Journal of Production Research (2003)
23. Bergeron, D., Jamali, M.A., Yamamoto, H.: Modelling and analysis of manufacturing systems: a review of existing models. IJPD (2010)
24. Nourelfath, M., Nahas, N., Ait-Kadi, D.: Optimal design of series production lines with unreliable machines and finite buffers. J of Qual in Maintenance Eng (2005)
25. Liu, Y., Li, J.: Split and merge production systems: performance analysis and structural properties. IIE Transactions (2010)
26. Spinellis, D.D., Papadopoulos, C.T.: A simulated annealing approach for buffer allocation in reliable production lines. Annals of Operations Research (2000)
27. Zhang, M., Matta, A.: Models and algorithms for throughput improvement problem of serial production lines via downtime reduction. IISE Transactions (2020)
28. Filho, M.G., Utiyama, M.H.R.: Comparing the effect of different strategies of continuous improvement programmes on repair time to reduce lead time. Int J Adv Manuf Technol (2016)
29. Godinho Filho, M., Utiyama, M.H.R.: Comparing different strategies for the allocation of improvement programmes in a flow shop environment. Int J Adv Manuf Technol (2015)
30. Jasiulewicz-Kaczmarek, M., Bartkowiak, T.: Improving the performance of a filling line based on simulation. IOP Conf. Ser.: Mater. Sci. Eng. (2016)
31. Harrington, B.:Breakdowns Happen. How to Factor Downtime into your Simulation. Whitepaper. (2014)
32. Dagkakis, G., Heavey, C., Robin, S., Perrin, J.: ManPy: An Open-Source Layer of DES Manufacturing Objects Implemented in SimPy. IEEE, Piscataway, NJ (2013)
33. Sutton, R.S., McAllester, D., Singh, S., Mansour, Y.: Policy Gradient Methods for Reinforcement Learning with Function Approximation. MIT Press, Cambridge, MA, USA (1999)
34. Schulman, J., Wolski, F., Dhariwal, P., Radford, A., Klimov, O.: Proximal Policy Optimization Algorithms. https://arxiv.org/pdf/1707.06347 (2017)