

Scheduling Individualized Products with Reinforcement Learning for Variable-Length Production Plans

Leon Vogel, Maylin Wartenberg

Suggested citation:

Vogel, Leon, and Maylin Wartenberg. 2025. "Scheduling Individualized Products with Reinforcement Learning for Variable-Length Production Plans." In *KI-Forum 2025 : KI in Forschung und Lehre an Hochschulen*, edited by Hanno Homann, Cedric Rohbani, and Jens Christian Will, 97–100. Hannover: HsH Applied Academics. <https://doi.org/10.25968/opus-3788>.

Abstract

This paper examines a reinforcement learning (RL) approach to solve a real-world, multi-objective scheduling problem in an automotive seat production line. The environment is dynamic and human-centered, aiming to optimize both tardiness and workload balance for workers. Building on prior work with fixed-length schedules, we introduce variable-sized production plans to better reflect real-world variability. A RL agent is trained using a reward based on relative performance against a genetic algorithm (GA) benchmark, capturing trade-offs between minimizing tardiness and maximizing workload variation. While the agent performs reliably on shorter plans, closely matching the GA baseline, performance in workload balancing deteriorates for larger plans. Results indicate that the agent tends to sacrifice workload balance to improve deadline adherence.

Terms of use

CC BY 4.0

This document is made available under these conditions:
Creative Commons - CC BY - Namensnennung 4.0 International
For more information see:
<https://creativecommons.org/licenses/by/4.0/deed.de>



Scheduling Individualized Products with Reinforcement Learning for Variable-Length Production Plans

Leon Vogel , Maylin Wartenberg 

Hochschule Hannover, Fakultät IV – Wirtschaft und Informatik, Abteilung Wirtschaftsinformatik

Abstract—This paper examines a reinforcement learning (RL) approach to solve a real-world, multi-objective scheduling problem in an automotive seat production line. The environment is dynamic and human-centered, aiming to optimize both tardiness and workload balance for workers. Building on prior work with fixed-length schedules, we introduce variable-sized production plans to better reflect real-world variability. A RL agent is trained using a reward based on relative performance against a genetic algorithm (GA) benchmark, capturing trade-offs between minimizing tardiness and maximizing workload variation. While the agent performs reliably on shorter plans, closely matching the GA baseline, performance in workload balancing deteriorates for larger plans. Results indicate that the agent tends to sacrifice workload balance to improve deadline adherence.

Index Terms—Reinforcement Learning, Permutation Flow Shop

I. INTRODUCTION

Effective production scheduling is essential in modern manufacturing, as it increases productivity, conserves resources, and reduces costs [1]. This study investigates a real-world scheduling task at a German automotive supplier that manufactures customized car seats. The company must continuously optimize the sequence of individualized products to meet customer demand and avoid bottlenecks. The production environment is highly dynamic: order specifications and due dates can change rapidly, and the product mix includes many custom variants. While automation is increasing, human workers remain critical for many assembly tasks. Highly repetitive or uneven work can cause fatigue or stress, highlighting the need for human-centered scheduling that balances workload [2].

The scheduling problem involves 12 sequential workstations arranged along a moving conveyor belt, where each station handles one product at a time. Processing times for each product at each station are known in advance, based on Methods-Time Measurement (MTM) [3], [4]. If a worker’s task exceeds the available time before the next product arrives, time pressure and stress increase. Conversely, consecutive tasks with minimal workload may lead to underutilization and boredom. Therefore, the two objectives are minimizing tardiness (delays past product deadlines) and reducing workload imbalances across workstations. The problem is modeled as a multi-objective permutation flow-shop scheduling problem.

Finding optimal permutations is NP-hard, so practitioners often use heuristics or metaheuristics [2]. Genetic algorithms (GAs) and other population-based methods can identify trade-

offs between conflicting objectives but are computationally expensive for large-scale or real-time problems [5]. Simple dispatching rules (e.g., earliest due date) are computationally efficient but usually result in low-quality schedules [6]. Recently, deep reinforcement learning (RL) has shown promise in scheduling and control tasks because agents can learn from experience and adapt to dynamic environments [7]. However, most RL-based scheduling approaches focus on a single objective (e.g., minimizing makespan or idle time), while human-centric objectives, such as workload balancing are often neglected [2], [8], [9]. A key challenge in multi-objective RL is balancing competing goals. Many approaches combine objectives into a single weighted sum, but this requires manual tuning of weights and may not reflect the actual trade-offs [10]. To overcome this, we leverage a GA baseline. For each instance, the RL agent’s solution is compared to the GA’s, and a reward is calculated based in relative performance. This approach captures multi-objective trade-offs without needing manual weight tuning and has shown superior results over fixed-weight strategies in prior studies [11].

In earlier work, products were grouped using clustering, and the product with the earliest deadline from each cluster defined the RL action space. This paper extends that approach by generalizing to variable-sized production schedules. Whereas previous work used fixed-length plans (20 jobs), we now sample plan lengths from a normal distribution to better reflect real-world variability. This enables us to examine whether the RL agent maintains its performance advantage under more realistic conditions [11].

II. METHOD

A. Permutation Flow Shop Problem

The assembly line is modeled as a permutation flow shop with 12 sequential workstations. Products move at constant speed on a conveyor belt, creating fixed time windows for task completion. When a worker needs longer to finish a task than is available at the workstation, the next task will need to be finished sooner. High workload in consecutive products can lead to worker stress, while low workload sequences may cause boredom. To balance the workload, the product sequence should maximize workload variation per worker. In addition, each product has a delivery deadline, requiring a balance between ergonomic criteria and timely completion.

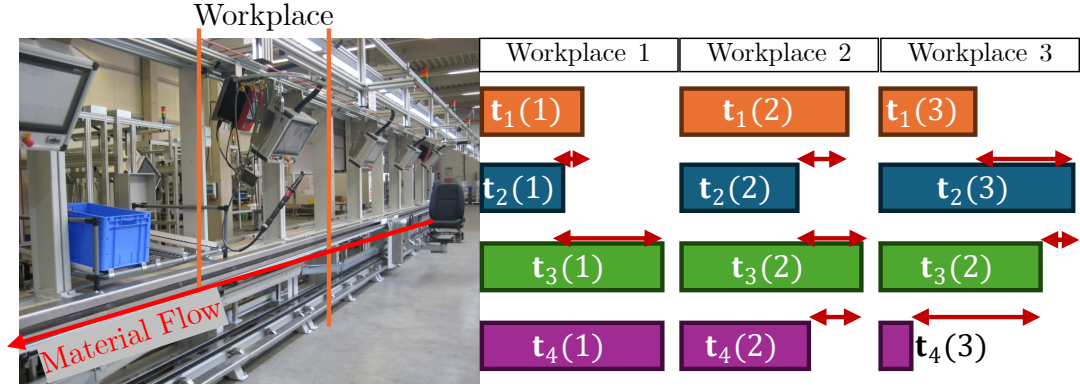


Figure 1. Left: Image of the line scheduled in this paper. Right: Example production plan with three machines and four products. The red arrows indicate the absolute difference in workload between the products (adapted from [11]).

Fig. 1 illustrates the production line setup and a simplified example with three machines and four products. Products are processed in the same order at each station, and the difference in workload between consecutive products, indicated by red arrows, is used as a key metric to improve sequence diversity and reduce stress. The schedule should also provide sufficient buffer time before product deadlines to minimize tardiness [11].

The problem is formulated as a permutation flow-shop scheduling problem, with the key distinction that product movement is synchronized, meaning all products have the same cycle time [2]. The following is the description of the problem as a multi-objective optimization problem from the previous work.

”An assembly line consists of a set of workplaces m_1, \dots, m_M with $M \in \mathbb{N}$. Each workplace has a fixed time window of $\tau \in \mathbb{R}^+$ to complete its respective task. Jobs enter the production line at the first workplace with a time interval of $\gamma \in \mathbb{R}^+$. Thus every γ seconds a job enters the first workplace. If $\gamma < \tau$ then the worker might still be occupied with the predecessor product. Thus, two products with high workload following each other will cause stress for the workers. The goal is to find a permutation P of a set of jobs $\{j_1, \dots, j_J\}$, $J \in \mathbb{N}$ such that the objectives are optimized. Each job has a corresponding time vector $\mathbf{t}_j \in \mathbb{R}^M$ where $\mathbf{t}_j(m)$ contains the workload of job j at workstation m . Additionally, a deadline d_j is given for each job. The objectives are the workload variety maximization and the tardiness minimization. Solutions to the problem are permutations P of all jobs, where P_n denotes the job at position n of the permutation.” [11]

The workload objective of a given permutation P is calculated as the sum of the absolute differences in workload

between consecutive jobs:

$$w(P) = \sum_{n=2}^J \|\mathbf{t}_{P_n} - \mathbf{t}_{P_{n-1}}\|_1 \quad (1)$$

The completion time $\sigma(n)$ of a job at position n in a given permutation P is determined by two components. The time proportional to its position in the sequence until it is being processed, γn , and a fixed time it takes to pass through all M workstations, $\sigma(n) = \gamma n + \tau M$. Based on this, the tardiness objective is calculated using an exponential penalty on late completion:

$$b(P) = \sum_{n=1}^J e^{\sigma(n) - d_{P_n}} \quad (2)$$

The exponential penalty function disproportionately penalizes late completions, while early finishes benefit from the time buffer through a decreased contribution to the objective.

B. Genetic Algorithm Solutions

The dataset used in this study builds on the one from prior work, which contains realistic production orders collected from an automotive supplier. Initially, the 20 most time-critical jobs were extracted every hour during working hours over one month. Each job included the configuration-specific workload, derived from Methods-Time-Measurement (MTM) standards, and a deadline in seconds [3], [4].

In contrast to the previous work with only 20 jobs per plan, this study introduces variability. Training and test plans were generated such that the number of products follows a normal distribution with a mean of 20 and a standard deviation of 5. A total of 400 training and 100 test instances were generated. To compare with a fixed-length agent, the original dataset was reused for the baseline scenario. To enable a direct comparison with an agent trained on fixed-size instances, the original dataset from the previous study was reused.

Each instance was solved using the same genetic algorithm (GA) from the earlier study. GAs are well-suited for permutation problems and operate through crossover, mutation,

and selection. A steady-state strategy retains top-performing solutions across generations. The NSGA-II implementation from pymoo was used for optimization [12].

C. Reinforcement Learning

In this work, we exclusively use the SparseGA reward introduced in our previous study, as it achieved the best performance [11]. The number of clusters used to group products by workload ranges from 6 to 10, based on the best-performing configurations identified in the previous study. The reward is calculated at the end of each episode by comparing the agent’s solution to a reference solution obtained via a genetic algorithm. Specifically, it reflects the relative improvement or degradation in both objectives—workload variation and tardiness—compared to the GA baseline. No intermediate rewards are provided during the episode. The formulation allows the agent to implicitly learn the trade-offs encoded in the GA’s multi-objective optimization, without the need for manual reward weighting or tuning. The observation space remains unchanged and includes information about the most recent scheduled products, the most urgent product in each cluster, the number of remaining products per cluster, and the episode progress.

III. RESULTS

To evaluate the method, the RL agent was trained on a dataset consisting of 400 problem instances and tested on 100 previously unseen cases. Training was performed for 1.5 million steps using Proximal Policy Optimization from the Stable Baselines 3 framework [13]. The hyperparameters included a batch size of 128, a linearly decaying learning rate of 0.0005, a discount factor of 0.9999, and training updates every 1024 steps.

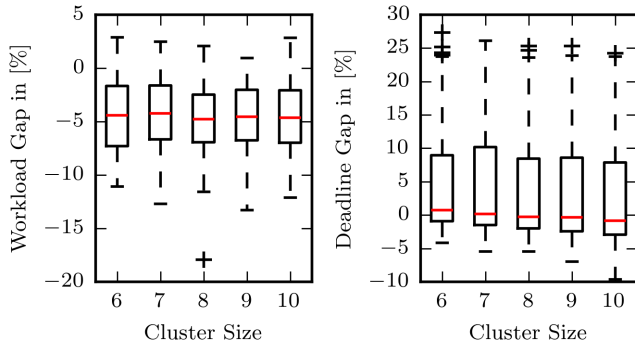


Figure 2. Test results with mixed plan sizes for different number of clusters. Left: Deadline gap. Right: Workload gap.

Fig. 2 illustrates the effect of different cluster sizes on agent performance for mixed plan sizes. The workload gap (left) and deadline gap (right) are expressed as relative deviations from the GA benchmark.

Across all cluster configurations, the workload objective was on average approximately 5% worse than the GA baseline, with variations ranging from -12% to +3%. This indicates that

the RL agent was unable to consistently match the workload balancing of the GA.

For the deadline objective, smaller cluster sizes yielded better performance. Configurations with 6 or 7 clusters outperformed the GA on average. Some test instances achieved deadline gaps better than 25%, although in rare cases performance dropped to 10% below the GA. The wider spread in results reflects higher variability for this objective.

Overall, these findings suggest that the RL agent prioritizes deadline adherence at the expense of workload balancing. This contrasts with the RL agent from previous work, which achieved a more balanced trade-off when trained on fixed-size plans. Thus, no clear advantage of using variable-length training was observed in this setting.

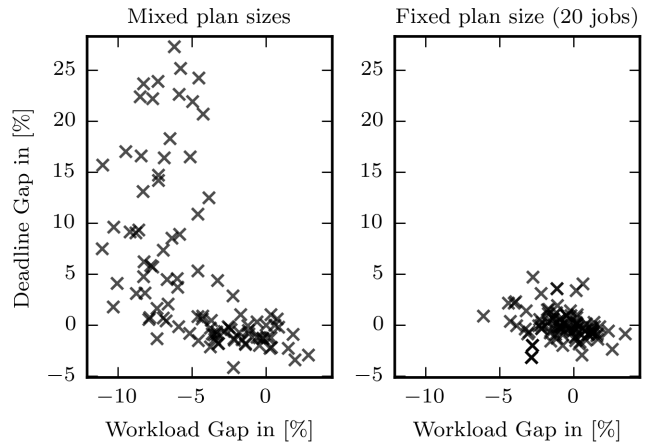


Figure 3. Test results of deadline gap and workload gap with 6 clusters. Left: Agent scheduling mixed plan sizes. Right: Agent scheduling plans of 20 jobs.

Fig. 3 compares agents trained on mixed-length plans (left) and fixed-size plans (right), both using six clusters. Each point represents a test instance, with workload gap on the x-axis and deadline gap on the y-axis.

For the fixed-size agent, most test instances cluster tightly around 0% deviation for both objectives. The mixed-size agent shows similar performance for many cases but also displays a substantial number of instances where deadline performance significantly improves—by up to 25% at the cost of much worse workload balancing (up to -10%). This reinforces the observation that the mixed-size agent tends to favor timeliness over ergonomic balance.

Fig. 4 analyzes the relationship between plan size and objective performance. The x-axis indicates the number of jobs in a test instance, while the y-axis shows the deadline gap. Data points are colored by workload gap.

For plans with 20 or fewer products, both objectives remain close to the GA baseline. Deadline gaps range from -4% to +2%, and workload gaps from -2% to +2%. However, as plan sizes exceed 20, performance deteriorates: deadline gaps increase, and workload gaps drop between -6% and -10%.

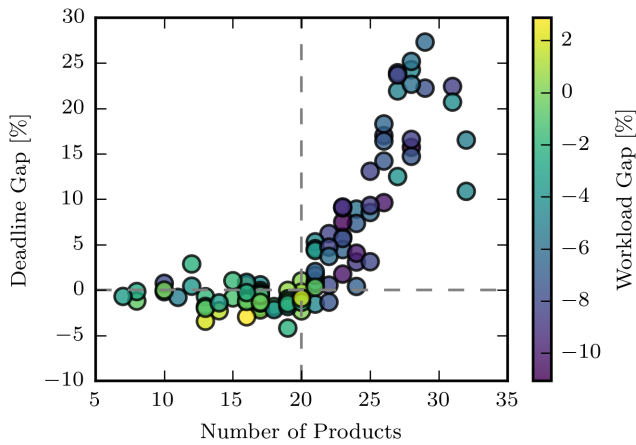


Figure 4. Deadline gap for different number of jobs from agent using cluster size 6. Workload colorized.

This suggests that the agent performs reliably for smaller instances but fails to generalize to larger schedules. In practical applications, such an agent may be suitable for environments with low to moderate job volumes. Future research should investigate whether agents trained solely on fixed-size plans can generalize better to smaller instances than agents trained on mixed plan sizes.

IV. CONCLUSION

This study explored a reinforcement learning approach to schedule customized car seat production in a dynamic, human-centered manufacturing environment. While extending prior work to variable-length plans increases realism, the RL agent showed mixed results. Although it achieved comparable results to GA for plans with 20 and less products, it underperformed in workload balancing when plans include more products.

Our findings indicate that the RL agent, when trained on variable plan sizes, prioritizes meeting deadlines at the expense of ergonomic workload distribution. This contrasts with prior

results from fixed-length schedules, where a more balanced trade-off was achieved.

In practice, the agent could still be beneficial in environments with smaller and more predictable workloads. Future work should examine transferability of agents trained on fixed-length schedules to variable-length schedules.

REFERENCES

- [1] C. Shyalika, T. Silva, and A. Karunananda, "Reinforcement learning in dynamic task scheduling: A review," *SN Computer Science*, vol. 1, p. 306, 2020. doi:10.1007/s42979-020-00326-5.
- [2] L. Vollenkemper, F. Grumbach, M. Kohlhasse, and P. Reusch, "Humanzentrierte Ablaufplanung von Montagelinien," *wt Werkstattstechnik online*, vol. 113, no. 4, pp. 158–163, 2023.
- [3] A. Syska, *Methods-Time-Measurement (MTM)*. Wiesbaden, Germany: Gabler, 2006. doi:10.1007/978-3-8349-9091-4_37.
- [4] American Society of Mechanical Engineers, "Automated navigation of method time measurement tables for automotive assembly line planning," in *Proc. Int. Design Eng. Tech. Conf. Comput. Inf. Eng. Conf.*, vol. 4, 2013. doi:10.1115/DETC2013-13325.
- [5] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," 2002.
- [6] A. Boes and A. Ziegler, "Forschungsreport – Umbruch in der Automobilindustrie," 2021. doi:10.36194/IDGUZDA_Forschungsbericht_Auto.
- [7] B. M. Kayhan and G. Yildiz, "Reinforcement learning applications to machine scheduling problems: A comprehensive literature review," *J. Intell. Manuf.*, vol. 34, pp. 905–929, 2023. doi:10.1007/s10845-021-01847-3.
- [8] W. Han, F. Guo, and X. Su, "A reinforcement learning method for a hybrid flow-shop scheduling problem," *Algorithms*, vol. 12, 2019. doi:10.3390/a12110222.
- [9] J. Ren, C. Ye, and F. Yang, "Solving flow-shop scheduling problem with a reinforcement learning algorithm that generalizes the value function with neural network," *Alexandria Eng. J.*, vol. 60, pp. 2787–2800, 2021. doi:10.1016/j.aej.2021.01.030.
- [10] Z. Pan, L. Wang, C. Dong, and J. F. Chen, "A knowledge-guided end-to-end optimization framework based on reinforcement learning for flow shop scheduling," *IEEE Trans. Ind. Inf.*, vol. 20, pp. 1853, 2024. doi:10.1109/TII.2023.3282313.
- [11] L. Vogel, L. Vollenkemper, A. Müller, and M. Kohlhasse, "Scheduling Individualized Products with Reinforcement Learning to reduce Worker Stress," *ICIEA-EU 2025*, accepted and expected to be published in 2025/2026.
- [12] J. Blank and K. Deb, "pymoo: Multi-objective optimization in Python," *IEEE Access*, vol. 8, pp. 89497–89509, 2020.
- [13] A. Raffin et al., "Stable-baselines3: Reliable reinforcement learning implementations," *J. Mach. Learn. Res.*, vol. 22, no. 268, pp. 1–8, 2021. Available: <http://jmlr.org/papers/v22/20-1364.html>.