Proceedings of the 2024 Winter Simulation Conference
H. Lam, E. Azar, D. Batur, S. Gao, W. Xie, S. R. Hunter, and M. D. Rossetti, eds.

MAINTENANCE PLANNING WITH DETERIORATION BY A REINFORCEMENT LEARNING APPROACH - A SEMICONDUCTOR SIMULATION STUDY

Cas Leenen[1,2], Michael Geurtsen[1,2], Ivo Adan[1], and Zumbul Atan[1]

[1]Industrial Technology and Eng. Centre, Nexperia, Nijmegen, THE NETHERLANDS
[2]Dept. of Industrial Eng., Eindhoven University of Technology, Eindhoven, THE NETHERLANDS

ABSTRACT

Manufacturing companies are often faced with deteriorating production systems, which can greatly impact their overall performance. Scheduling the preventive maintenance activities optimally is vital for these companies. This study utilizes historical data on product quality deterioration in the last machine of a real-world serial, buffered production line to refine maintenance decisions. Currently, maintenance intervals for the last machine are fixed. The objective is to improve this policy by increasing the production system's long-term throughput. A simplified two-machine-one-buffer (2M1B) system is modeled to devise and assess policies. Various optimization techniques are applied, with the average reward Reinforcement Learning (RL) technique showing the most promising results in numerical experiments. The RL-optimized policy exhibits significant potential by considering product quality deterioration, buffer levels ahead of the last machine, and the machine states of the last two machines.

1 INTRODUCTION

In modern manufacturing, striking for the right balance between high production quality and minimal production loss due to shutdowns for preventive maintenance (PM) is a constant challenge. Finding an optimized PM policy that decides when to do maintenance is crucial in maximizing the overall throughput of a production system.

This study addresses a specific problem at Nexperia, a semiconductor manufacturer of Integrated Circuits (ICs). The focus is on the last machine of a serial buffered production line. A representation of the entire production line is provided in Figure 1. In between each machine, there is a finite buffer for storing products. The focus of this study is on the PM activity of the last machine in the line, the injection molding machine. The PM activity considered is a cleaning activity, which ensures that the production quality is back to normal. The current time-based mold cleaning (MC) policy, performed every 12 hours, lacks consideration for production quality and the production line's state. Not considering these characteristics leads to more production losses due to frequent and wrongly timed MC. Activities performed on the last machine in the line can cause stoppage of preceding machines, potentially resulting in lower system throughput. However, defining production quality deterioration is complex due to multiple data sources that could indicate mold quality performance. Knowledge about quality deterioration is crucial for an optimal trade-off between production loss and interruption due to cleaning. This research explores dynamic programming (DP) and reinforcement learning (RL) techniques to develop an optimized MC policy considering the challenges of defining production quality deterioration, creating an accurate mathematical framework for a two-machine-on-buffer production-line, and training RL techniques effectively.

The remainder of the paper is organized as follows. Sections 2 provides a brief overview of the literature. In Section 3, a formal problem description is provided. The solution methods are proposed in Section 5. In Section 6 we present our computational results. Finally, Section 7 concludes the paper with recommendations for future research.

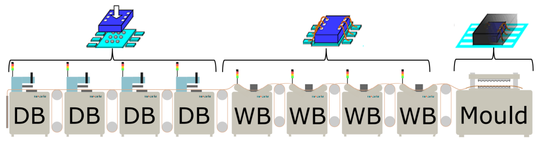


Figure 1: An illustration of a production line at Nexperia, showing multiple machines with varying processes.

2 LITERATURE

Past and current research extensively examines preventive maintenance (PM) policies, ranging from single-component systems to entire production lines. (Wang 2002) provides an overview of various maintenance policy categories. The primary objective of PM is to prevent component breakage, which typically incurs higher costs associated with corrective maintenance (CM). Therefore, much research aims to minimize maintenance costs, as demonstrated in (Alrabghi and Tiwari 2015). Alternatively, some approaches focus on maximizing overall throughput by addressing product output quality deterioration, as in (Cui et al. 2022).

Detecting deterioration presents challenges. When sensors indicate machine deterioration, determining the extent is straightforward. However, without direct sensor data, estimation becomes necessary. Literature employs various techniques to model deterioration realistically. Distinctions between discrete and continuous time modeling are crucial (Alaswad and Xiang 2017). Continuous time degradation models include the Wiener, Gamma, and Inverse Gaussian processes. The choice depends on whether the degradation process is monotonic or non-monotonic. For instance, (Do et al. 2015) use a Gamma stochastic process for a monotonically decreasing degradation process involving incremental shocks. Deterioration modeling often employs discrete-time methods when precise measurements are challenging. (Alaswad and Xiang 2017) note instances where real-world deterioration is continuous, yet discrete-time is used for modeling. A common approach is the discrete-state Markov process, depicting a sequence of possible states where transitions depend on the current state. (Kang and Ju 2019) and (Wei et al. 2022) exemplify this with discretized deterioration states, ranging from optimal to threshold states requiring maintenance.

Transitioning from modeling single machines to complete production systems involves discrete Markov processes or simulation-based methods. In (Wei et al. 2023), a two-machine-one-buffer (2M1B) system is modeled as a Markov process. Decomposition into stand-alone 2M1B systems is explored in (Kang and Ju 2019), who simplify a longer serial production line using an aggregation-based approach, representing it as a single virtual machine with two states. Simulation, such as Discrete Event Simulation (DES), is another modeling option. (Geurtsen et al. 2023) utilize a DES to model a production line which is then used as an environment for deep reinforcement learning to generate maintenance policies. (Yang et al. 2008) compare maintenance schedules through discrete event simulation on a parallel production lines without buffers, while (Arab et al. 2013) employ a metaheuristic policy tested through simulation iterations. The study by (Kim and Morrison 2019) examines a 2-server-1-buffer system with machines that deteriorate until a threshold, requiring PM. They model an MDP and aim to find the optimal maintenance policy to minimize costs.

The choice of maintenance optimization method hinges on system modeling. Dynamic Programming (DP), exemplified in (Geurtsen et al. 2022), employ Markov Decision Processes (MDP) with known transition probabilities. (Kang and Ju 2019) also use a DP algorithm, Value Iteration (VI), for optimizing the maintenance policy in a 2M1B system. Reinforcement Learning (RL), versatile in both MDP and simulation settings, is a prevalent optimization technique, particularly in stochastic problems without prior system knowledge. (Huang et al. 2019) apply Q-learning, a common RL algorithm, to minimize maintenance costs in a serial buffered system over a short one-week horizon.

A summary of the most related studies are presented in Table 1. It can be seen that no specific research aligns precisely with the problem addressed in this paper. Current studies primarily center around machine deterioration and minimizing maintenance costs. Studies that consider product-related deterioration typically examine only a small set of discrete deterioration states, rather than the continuous-like deterioration used in this study. This motivates our research, focusing on optimizing a preventive maintenance (PM) policy that considers the states of a buffered serial production system, including product quality deterioration in the last machine, with the goal to maximize long-term net throughput.

Table 1: Summary of the relevant literature.

Paper	Buffer(s)	Deterioration type	Deterioration mode	Environment	Method	Objective
(Yang et al. 2008)	✓	Machine	Discrete	Serial/Parallel	MH	Max. profit
(Arab et al. 2013)	✓	Machine	Continuous	Serial	MH	Max. throughput
(Huang et al. 2019)	x	Machine	Discrete	Serial	RL	Min. cost
(Kang and Ju 2019)	✓	Machine	Discrete	Serial	DP	Max. throughput
(Cui et al. 2022)	✓	Product	Discrete	Serial	DRL	Min. cost
This study	✓	Product	Continuous	Serial	VI/RL	Max. throughput

3 PROBLEM DESCRIPTION

We consider a production system where the product quality declines with increased production due to residual molding material. Therefore, mold cleaning is crucial on the mold machine to restore optimal production quality. Currently, MC occurs at fixed 12-hour intervals across multiple production lines, each dedicated to a specific product type. Historical data reveals significant variation in the number of strokes between mold cleanings, despite the fixed time-based policy. Each stroke involves mold closing, material insertion, and mold opening resulting in the production of multiple products. Figure 2 illustrates the strokes' distribution between consecutive mold cleanings over the past year for various mold machines.

The production line distribution is derived from Equipment State Monitoring (ESM), capturing machine states during production or maintenance. Despite an average of around 1900 strokes between mold cleanings, significant variability exists, indicating diverse effective production times. Moreover, the imprecise timing of MCs suggests that the current policy is suboptimal, as each cleaning occurs at a different production quality level. Considering additional production line characteristics could enhance efficiency, ultimately aiming to maximize long-term net throughput while minimizing production quality loss.

The production line, as shown in Figure 1, is a serial buffered production line comprising N machines (defined by the set M) and $N-1$ buffers (defined by the set B). Each machine n can be in either one of the following four production states: up, down, blocked or starved. To mimic reality as close as possible, the following assumptions are made regarding the production system:

1. During production, the machine state is "up".
2. "Down" state occurs when a machine cannot produce due to factors such as mechanical breakdowns.
3. Machines with an upstream buffer may be "blocked" when the buffer is full, halting production.
4. The "starved" state occurs when a machine's buffer is empty, except for the first machine in line.
5. Machine speed (v_n) is measured in products per second, reduced to zero in down, starved, or blocked states.
6. Buffer level (b_n) of the buffer preceding machine n is determined by machine states and speeds upstream and downstream.
7. Only the last machine, the mold, undergoes deterioration affecting production quality.
8. Deterioration doesn't influence up and down behavior.
9. Mold cleaning is possible regardless of machine states.
10. MC duration spans from production interruption to machine reactivation.
11. Resources are always available for mold cleaning activities.

To assess how production quality changes with an increasing number of strokes after the last mold cleaning, the study examines multiple MC periods over a year of production. Each MC period is subdivided into intervals, each covering a range of strokes from zero to the period's end, in 500-stroke intervals. For every 500-stroke region within an MC period, the quality is calculated. The weighted average quality ($\bar{y}$) for a specific region is determined by (1).

$$\bar{y} = \frac{\sum_{i=1}^{b} w_i \cdot y_i}{\sum_{i=1}^{b} w_i} \tag{1}$$

where b is the number of batches produced in that region of 500 strokes, and y_i is the quality level of batch i produced in the region. Here, w_i denotes the batch i's contribution percentage to the 500 strokes, calculated by dividing the strokes produced by batch i within the region by the total 500 strokes. If a batch spans the entire region, the weighted average quality is the batch's corresponding quality. Otherwise, when multiple batches are produced within 500 strokes, the weighted average formula determines the region's quality accurately.

To provide a comprehensive view of quality levels, the weighted averages for the same region across different MC periods are aggregated and averaged. Figure 3 illustrates the relationship between production quality and mold cleaning across various production lines, showcasing the averaged quality levels for a 500-stroke region after mold cleaning. It reveals varying deterioration patterns among production lines, with some exhibiting more significant quality reduction as strokes increase, while others show less deterioration.

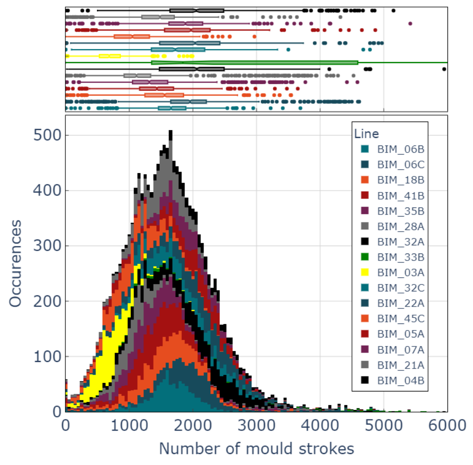


Figure 2: Overview of the distribution of the number of strokes between two consecutive mold cleanings for the mold machines of different production lines.

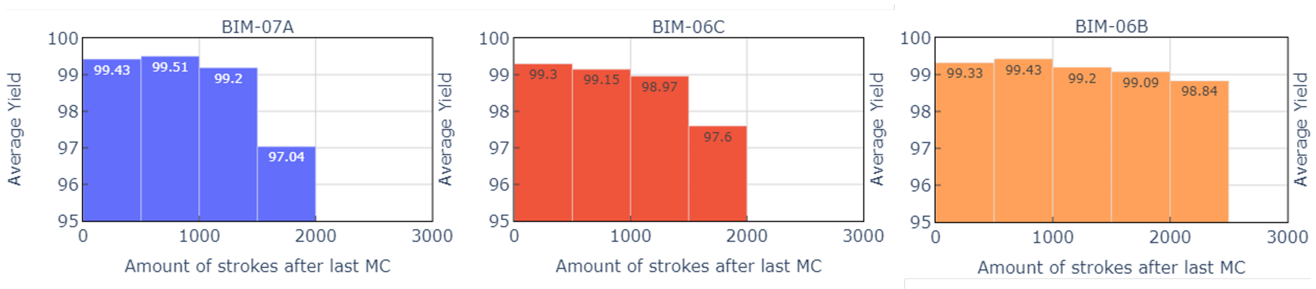


Figure 3: Average quality level per bin of 500 strokes production for multiple production lines.

4 SYSTEM MODELING

The optimization techniques outlined in Section 2 show promise but present a challenge. To facilitate policy training and evaluation, this section adopts a simplified two-machine-one-buffer (2M1B) system, modeled as a Markov Decision Process (MDP), representing the complete production line.

4.1 Simplified 2M1B Model

A Markov Decision Process (MDP) models decision-making in scenarios with random events and agent actions, defining states, actions, transition probabilities, and rewards. Each time step involves taking an action, transitioning to a new state, and receiving a reward. The goal is to find optimal policies based on accumulated rewards. Constructing a 2M1B environment requires complete system knowledge, though real-world stochastic behavior is imprecise. This MDP aligns with reality by using relevant parameters, detailed in the subsequent subsections addressing the specific mold cleaning policy problem.

4.1.1 State Space

The MDP's state space encompasses all possible combinations of the 2M1B system components: the machine preceding the mold, the mold itself, and the buffer in-between. This choice is rooted in the assumption that the machine preceding the mold has the most substantial impact on its performance. The state space includes all possible combinations of these four components:

- **First Machine State (M_1)**, indicates if the machine is in production (U) or down (D). The down-state probability is determined from historical data, potentially due to mechanical, electrical, or software breakdowns. ($M_1 = U, D$)
- **Buffer Level (b_1)**, represents the remaining mold strokes in the buffer. The buffer content depends on surrounding machine states. A full buffer blocks the first machine, while an empty buffer starves the mold, causing zero production. Buffer values are discretized based on the step size, S. ($b_1 = 0, S, 2S, \cdots, MS$, where MS is the maximum buffer size)
- **Mold Machine State (M_2)**, determines if the mold is in production (U), down (D), or undergoing mold cleaning (MC). The mold enters the cleaning state with a probability of 1 when cleaning is decided, while the down-state probability is determined similarly as the first machine. ($M_2 = U, D, MC$)
- **Strokes After Last Mold Cleaning (n)**, describes the number of strokes produced since the last mold cleaning, influencing production quality ($n = 0, S, 2S, \cdots,$ max number of strokes). This MDP is finite; thus, a limit on postponed cleaning is set based on the lowest acceptable quality level.

This leads to the four-component state s_t at time step, t, of:

$$s_t = [M_1, b_1, M_2, n] \tag{2}$$

4.1.2 Action Space

The action space includes two options at a given state: "Continue" and "Execute Mold Cleaning." These actions are exclusive to the mold, with no direct actions for the first machine or buffer. Any state allows choosing between these two actions except during ongoing mold cleaning, where "Continue" is the only option. Additionally, when reaching the maximum mold strokes, the MDP must take the "Execute Mold Cleaning" action. After completing the mold cleaning process, the MDP can immediately opt for the mold cleaning action again.

4.1.3 Transition Probabilities

The machines in the MDP transition from an up state to a down state and vice versa according to a geometric distribution. Each time step, an independent trial is conducted with two outcomes; for example, in case a line is up, it either remains up with a certain probability or it goes down with a certain probability. These probabilities remain the same for each time step. Machine 1 and Machine 2 operate at similar rates, producing a fixed number of products per time step only when operational. The buffer fills when Machine 1 is up and Machine 2 is down, and it empties when the roles are reversed. If the buffer is full, Machine 1 is shut down, and vice versa for Machine 2. For more details, we refer to (Geurtsen et al. 2022)

4.1.4 Reward Structure

Constructing the reward function is crucial as it directly ties to the throughput of the 2M1B system. In many state space combinations, the throughput is zero, qualifying a reward of zero. A reward is given only when the mold's machine state is up, and the buffer is not empty. In addition to machine and buffer states, the number of strokes produced after the last mold cleaning (n) influences the reward function. The immediate reward function for a state-action pair is denoted as $r_{imm}(s,a)$, with s defined in (2). The action a can be either "Continue" or "Execute MC". At time step t, the immediate reward is calculated as follows:

$$r_{imm}(s,a) = Z(n) \times TP \tag{3}$$

Here, $Z(n)$ represents the deterioration factor corresponding to the quality at that specific number of strokes n (for example, if at $n = 1000$ the quality is 98%, then $Z(1000) = 0.98$). TP indicates the production output in case the mold machine is in production and has just finished maintenance, i.e. the production output in case of producing 100% good quality products. Figure 4 illustrates the deterioration factor derived from historical quality data for production line 07A, as seen in Figure 3. The bars show the deterioration profile of the mold in production line 07A in relation to the stroke count.

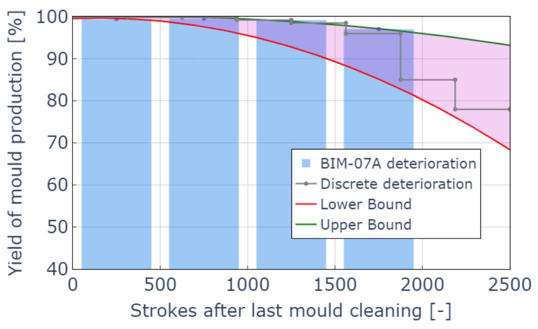


Figure 4: Quality level used in the reward function for the mold of production line 07A.

Assumptions are necessary to model deterioration progression, leading to the use of upper and lower bounds, as illustrated in Figure 4. The uncertainty grows at higher stroke levels due to limited test data. Upper bounds follow original data points, while lower bounds are determined by available data accuracy. Note that deterioration might be higher as not all tests that affect production quality are included. Numerical experiments in Section 6 considers this uncertainty region, testing discrete profiles to observe their impact in an MDP environment.

5 SOLUTION METHODS

This section describes two techniques to train and optimize the mold cleaning policy. First, an exact solution method, Dynamic Programming (DP), is used to find a (near)-optimal solution in the MDP environment.

Then, reinforcement learning is applied to test its potential for more complex systems which cannot be solved by DP.

5.1 Dynamic Programming

The dynamic programming technique is model-based, meaning it needs to have knowledge about the whole environment to find an optimal solution. Consequently, this technique can only be used on small problems such as the 2M1B system considered in this study.

5.1.1 Value Function

The main component within dynamic programming is the value function, which examines the performance of taking actions at certain states in the MDP. The value function assigns a value to each state, indicating its importance. It is a fundamental concept in dynamic programming but also reinforcement learning techniques. The value function, denoted as $V(s)$, represents the expected return that can be achieved from being in a particular state, see (4).

$$V(s) = max_{a \in A} \left[r_{imm}(s,a) + \gamma \sum \left[P(s' \mid s,a) \cdot V(s') \right] \right] \tag{4}$$

where $V(s)$ represents the value of state s, $r_{imm}(s,a)$ denotes the immediate reward obtained when taking action a in state s, γ (gamma) is a discount factor that determines the importance of future rewards, $P(s' \mid s,a)$ represents the transition probability moving from state s to s' when action a is taken, and the summation is taken over all possible next states s' that are reachable within one step from the current state.

By iteratively updating the value of each state using the above equation, algorithms like value iteration converge to the optimal value function. The value function is used to assess the significance or quality of different states in the environment. It helps in decision-making to prioritize actions that lead to states with higher values.

5.1.2 Value Iteration Algorithm

There are mainly two DP techniques widely used: Policy iteration (PI) and Value iteration (VI). VI is preferred over policy iteration in the context of a large MDP with sparse transitions. This choice is motivated by the computational efficiency of value iteration in comparison with policy iteration. The VI algorithm initiates the value function for each state. The value of each state is then updated iteratively using the value function (4) until convergence is reached. The exact steps of the algorithm can be found in Algorithm 1.

Algorithm 1: Value iteration

```
  Data: MDP, $\gamma$, tolerance $\varepsilon$
  Result: $V(s)$
1 Initialize $V_0(s)$ to all zero;
2 while $M_t - m_t \geq \varepsilon \cdot m_t$ do
3     for all $s \in S$ do
4         $V_{t+1}(s) = \max_{a \in A} [r_{imm}(s,a) + \gamma \sum [P(s' \mid s,a) \cdot V_t(s')]]$;
5     end
6     $m_t = min_{s \in S}[V_{t+1}(s) - V_t(s)]$;
7     $M_t = max_{s \in S}[V_{t+1}(s) - V_t(s)]$;
8     $V_t(s) \leftarrow V_{t+1}(s)$;
9 end
```

The variable t represents the time step in each iteration, starting at zero and continuing until the stopping condition is met. This method utilizes the span semi-norm, to examine the difference between the upper and lower bounds. The upper- and lower-bound are compared to ε multiplied by the minimum difference in value during the step. As a result, convergence is achieved when there is enough similarity between the minimum and maximum updates in the value function. In this study, the γ is set to 1 to ensure that the algorithm optimizes the average step over an infinite time horizon. This method is also called average reward value iteration, see (Tijms 1986).

5.2 Reinforcement Learning

In model-free reinforcement learning, the agent interacts with an environment without any prior knowledge about the transition probabilities or the available states. The agent learns from receiving rewards when taking actions in the environment. Many reinforcement learning techniques rely on a value table that stores an agent's past experiences. This table is updated at every time step, with each row representing a state and each column representing an available action. The values in the table, called action values, show the rewards the agent can expect from taking a particular action in a given state. Q-learning is a widely used technique that involves using a Q-table as the value table. At each step, the Q-table is updated using the Bellman equation (5), which calculates the optimal Q-value for a state-action pair by adding the immediate reward to the maximum Q-value of the next state-action pair, discounted by a factor (gamma). The discount factor, which ranges from 0 to 1, determines the importance of future rewards.

$$Q_{t+1}(s,a) \leftarrow Q_t(s,a) + \alpha * [r_{imm}(s,a) + \gamma * max_{a' \in A}(Q_t(s',a')) - Q_t(s,a)] \tag{5}$$

Where:

- $Q_t(s,a)$ represents the Q-value of state s and action a, at time step t.
- Learning rate α is a constant that controls the rate at which the agent learns and typically ranges between 0 and 1.
- $r_{imm}(s,a)$ is the immediate reward received after taking action a, in state s.
- $max_{a' \in A} Q_t(s',a')$ is the Q-value of the maximum next state-action pair off all possible actions in A.

By iterative updating the Q-table, the agent gradually learns which actions lead to higher rewards in different states. Through this learning process, the agent can eventually converge to an optimized policy for maximizing rewards in the environment. During this learning process, there is an essential trade-off between exploration and exploitation. The exploration part is needed to explore the whole environment and not get stuck in a local optimum. On the other hand, exploitation is needed to achieve convergence to an optimal policy at some point in the learning process.

5.2.1 R-learning Algorithm

For this problem, traditional discounted RL algorithms like Q-learning may not quality desired outcomes due to convergence challenges over long-term horizons. Instead, we adopt an average reward RL technique, named R-learning. Details on this particular technique can be found in (Schwartz 1993). R-learning optimizes average rewards for long-term performance. Like Q-learning, R-learning employs a tabular format, to store state-action pairs. However, in R-learning, Q-table values are adjusted during training based on the average reward, denoted as ρ_t. This adjustment alters the Q-value update formula to (6).

$$Q_{t+1}(s,a) \leftarrow Q_t(s,a)(1-\beta) + \beta[r_{imm}(s,a) - \rho_t + max_{a' \in A} Q_t(s',a')] \tag{6}$$

where β is the learning rate that decides how quickly errors in the estimated Q-values are corrected. Here, subtracting the average reward from the immediate reward $r_{imm}(s,a)$ effectively scales down the impact of the immediate reward on the Q-value update. In other words, the average reward in R-learning helps in estimating the expected average reward per time step and influences the agent's decision-making process. By incorporating the average reward, the algorithm aims to optimize the agent's actions towards maximizing the long-term cumulative reward rather than just the immediate rewards at each time step. The update function of the average reward is described in (7).

$$\rho_{t+1} \leftarrow \rho_t(1-\alpha) + \alpha[r_{imm}(s,a) + max_{a' \in A} Q_t(s',a') - max_{a \in A} Q_t(s,a)] \tag{7}$$

A complete description of the R-learning technique and the updates of the formulas above are described in Algorithm 2. The average reward ρ, is only updated when a non-exploratory action is selected. In other words, when the best possible action based on the current Q-values is selected.

```
Algorithm 2: R-learning
   Data: MDP, $\alpha$, $\beta$
   Result: $Q$
1  Initialize $Q_0(s,a)$ arbitrarily;
2  Initialize $\rho_0$ (e.g. $\rho = 0$);
3  Initialize $s$ randomly;
4  while Not done do
5      Select action $a$, with some exploration strategy(Section 5.2.2);
6      Take step;
7      $Q_{t+1}(s,a) \leftarrow Q_t(s,a)(1-\beta) + \beta[r_{imm}(s,a) - \rho_t + max_{a' \in A} Q_t(s',a')]$;
8      if non-exploratory action is selected then
9          $\rho_{t+1} \leftarrow \rho_t(1-\alpha) + \alpha[r_{imm}(s,a) + max_{a' \in A} Q_t(s',a') - max_{a \in A} Q_t(s,a)]$;
10     end
11     $s \leftarrow s'$;
12 end
```

5.2.2 Exploration Strategy

This study utilizes the Uncertainty Estimation (UE) exploration strategy proposed by (Mahadevan 1996). In this strategy, the agent selects the action, a, from all possible actions, A, with a probability p according to (8).

$$a = max_{a^* \in A}[Q(s,a^*) + \frac{c}{N_f(s,a^*)}] \tag{8}$$

Here, c represents a constant, and $N_f(s,a^*)$ denotes the number of times the agent has previously taken that particular state-action pair during the learning process. This approach incentivizes the agent to explore regions with low $N_f(s,a^*)$ values.

6 EXPERIMENTS

This section analyzes the performance of the VI algorithm and the R-learning method. Results are compared against two benchmark policies.

6.1 Parameter Setting

The simulation setting is detailed in Table 5a. It is chosen such to closely resemble real production conditions. The experiment utilizes historical data from production line 07A from Figure Figure 4 to assess policy effectiveness. Each experiment is run for 10 years and ten replications for each method. Simulations are conducted on a computer equipped with an Intel Core i7-10610U processor and 32.0 GB of RAM.

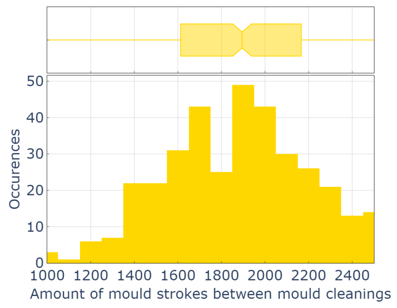
6.2 Benchmark Policies

6.2.1 Time-based Policy

The first benchmark policy is the time-based policy which is currently adopted by the factory. Under this policy, the mold cleaning is performed every 12 hours. To ensure a realistic benchmark policy, data on the number of strokes between two consecutive mold cleaning activities is used. This is based on real historical data. The distribution is shown in Figure 5b. This policy operates as follows: it assumes that the distribution of the number of strokes between cleaning activities follows a normal distribution. Consequently, in the MDP, each cleaning period is determined using a normal distribution characterized by a mean and a standard deviation, derived from historical data. In other words, for every interval between two cleaning activities, the normal distribution determines when the next cleaning will be performed based on the number of strokes. This approach guarantees that the policy accurately reflects the practice, establishing it as a benchmark policy.

Parameters	Values
Machine speeds	10 pcs/sec
Up/down probabilities of mould	Up: 85%, Down: 15%
Up/down probabilities of wirebond	Up: 85%, Down: 15%
Maximum buffer content of last buffer	250 strokes
Maximum to minimum yield level	99.51%-78.0%
Mould cleaning duration	75 min
Maximum number of strokes between MCs	2500 strokes
Number of products per mould stroke	100
VI tolerance ε	0.001
RL learning rate α (min-max)	0.02 - 0.12
RL learning rate β (min-max)	0.005 - 0.02
RL UE exploration constant c	50
RL UE probability p	0.9

(a) Simulation parameters.



(b) Time based mold-stroke distribution.

Figure 5: Simulation settings.

6.2.2 Stroke-based Policy

While the time-based policy relies on a distribution for the number of strokes between mold cleanings, the stroke-based policy operates with a fixed number of strokes. Mold cleaning occurs consistently after reaching a specific stroke count, without variation. This straightforward approach disregards machine and buffer states. Nevertheless, this policy offers valuable insights when compared with policies that consider machine and buffer states for decision-making.

6.3 State-dependent Policies

6.3.1 VI Policy

Figure 6 showcases the policy generated using the VI algorithm. It comprises four graphs, each illustrating a combination of up and down states for both the mold machine and the machine preceding it. The x-axis represents the number of strokes produced on the mold, while the grey line, paired with the left y-axis, denotes the corresponding quality level. Additionally, the right y-axis indicates the remaining strokes in the buffer. A filled or colored area in any graph indicates the policy's decision to execute mold cleaning. It can be seen that is is favourable to initiate maintenance when the machine prior to the mold is in a down state, as the optimal area of initiation is larger.

6.3.2 R-learning Policy

The RL agent is trained until convergence in the average throughput is achieved. The resulting policy shows some notable differences in behavior compared to the VI policy. Maintenance is initiated earlier in the RL policy in case the mold machine is down, while is postponed compared to the VI policy in case

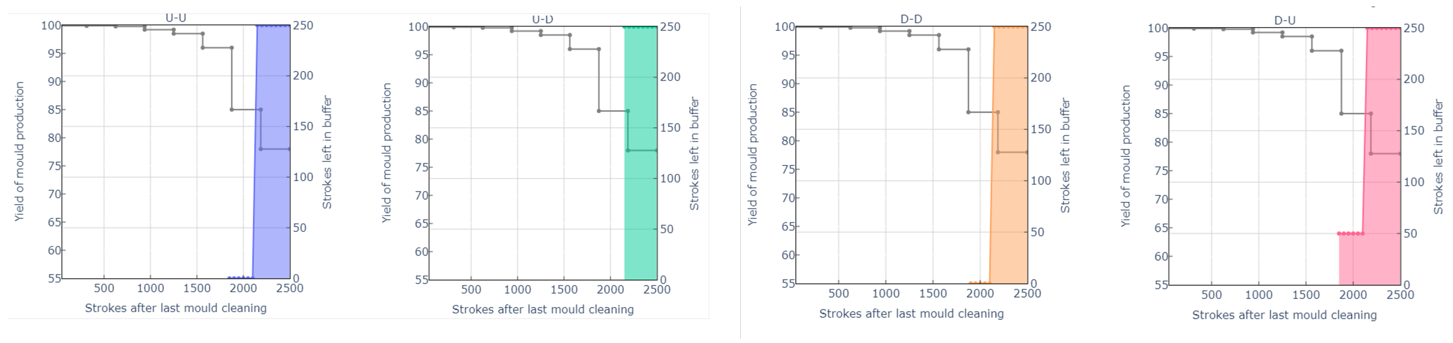


Figure 6: VI policy. The grey line indicates the quality level, and the filled areas, the moment cleaning is initiated for different combinations of machine production states.

Policy	Time-based	1000-strokes	1300-strokes	1600-strokes	1900-strokes	2200-strokes	2500-strokes	VI	R-learning
Throughput	6.831	6.118	6.523	6.795	6.955	6.978	6.942	6.988	6.967
95% confidence interval	1.76E-05	1.22E-05	1.74E-05	1.64E-05	1.77E-05	1.63E-05	1.47E-05	1.70E-05	2.42E-05
% improvement	-	-10.43%	-4.51%	-0.52%	1.82%	2.16%	1.63%	2.31%	2.00%

Table 2: Results of the different policies simulated in the 2M1B environment.

the mold machine is up.

6.4 Results

The results of the benchmark and state-dependent policies are presented in Figure 7 and Table 2. As expected, the time-based policy underperforms due to its lack of optimization. Notably, the optimal stroke value for the stroke-based policy is around 2200. Lower values lead to excessive maintenance, while higher values result in significant quality loss and production disruption. Remarkably, the stroke-based policy's 2200-stroke value closely aligns with the near-optimal policy obtained with VI. The VI policy, as observed in Figure 6, shows minimal variation across different buffer levels and machine states, except when the buffer is almost empty and the molding machine is active. This slight divergence contributes to a modest 0.15% improvement over the stroke-based policy. The R-learning policy performs admirably, trailing the VI policy by a mere 0.31%. This suggests promise for larger environments beyond the 2M1B MDP.

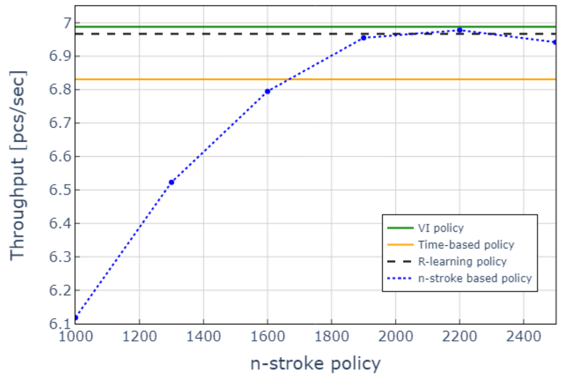


Figure 7: Policy performances. The x-axis represents the number of strokes for the stroke-based policy.

7 CONCLUSIONS AND FUTURE WORK

This study integrates historical product quality data into the preventive maintenance policy for the last machine in a serial buffered production line, aiming to maximize long-term throughput. Various policy optimization techniques are examined using a 2M1B MDP model and compared to benchmark policies. Results reveal that the stroke-based policy closely approaches the optimal VI policy. R-learning also shows promise in achieving results close to the VI policy. The work can be extended to a more realistic, complex environment with multiple machines and buffers, modeled using, for example, discrete event simulation.

REFERENCES

Alaswad, S. and Y. Xiang. 2017. "A review on condition-based maintenance optimization models for stochastically deteriorating system". *Reliability engineering & system safety* 157:54–63.

Alrabghi, A. and A. Tiwari. 2015. "State of the art in simulation-based optimisation for maintenance systems". *Computers & Industrial Engineering* 82:167–182.

Arab, A., N. Ismail, and L. S. Lee. 2013. "Maintenance scheduling incorporating dynamics of production system and real-time information from workstations". *Journal of Intelligent Manufacturing* 24:695–705.

Cui, P.-H., J.-Q. Wang, and Y. Li. 2022. "Data-driven modelling, analysis and improvement of multistage production systems with predictive maintenance and product quality". *International Journal of Production Research* 60(22):6848–6865.

Do, P., A. Voisin, E. Levrat, and B. Iung. 2015. "A proactive condition-based maintenance strategy with both perfect and imperfect maintenance actions". *Reliability Engineering & System Safety* 133:22–32.

Geurtsen, M., I. Adan, and Z. Atan. 2022. "Dynamic Scheduling of Maintenance by a Reinforcement Learning Approach-A Semiconductor Simulation Study". In *2022 Winter Simulation Conference (WSC)*, 3110–3121. IEEE.

Geurtsen, M., I. Adan, and Z. Atan. 2023. "Deep reinforcement learning for optimal planning of assembly line maintenance". *Journal of Manufacturing Systems* https://doi.org/https://doi.org/10.1016/j.jmsy.2023.05.011.

Huang, J., Q. Chang, and N. Chakraborty. 2019. "Machine preventive replacement policy for serial production lines based on reinforcement learning". In *2019 IEEE 15th International Conference on Automation Science and Engineering*, 523–528.

Kang, Y. and F. Ju. 2019. "Flexible preventative maintenance for serial production lines with multi-stage degrading machines and finite buffers". *IISE Transactions* 51(7):777–791.

Kim, T. and J. R. Morrison. 2019. "A Numerical Study on the Structure of Optimal Preventive Maintenance Policies in Prototype Tandem Queues". In *2019 Winter Simulation Conference (WSC)*, 2281–2291.

Mahadevan, S. 1996. "Average reward reinforcement learning: Foundations, algorithms, and empirical results". *Machine learning* 22:159–195.

Schwartz, A. 1993. "A reinforcement learning method for maximizing undiscounted rewards". In *Proceedings of the tenth international conference on machine learning*, Volume 298, 298–305.

Tijms, H. C. 1986. *Stochastic Modelling and Analysis: A Computational Approach*. USA: John Wiley Sons, Inc.

Wang, H. 2002. "A survey of maintenance policies of deteriorating systems". *European Journal of Operational Research* 139(3):469–489 https://doi.org/https://doi.org/10.1016/S0377-2217(01)00197-7.

Wei, S., M. Nourelfath, and N. Nahas. 2022. "Condition-based Maintenance Optimization of Degradable Systems". *International Journal of Mathematical, Engineering and Management Sciences* 7(1):1.

Wei, S., M. Nourelfath, and N. Nahas. 2023. "Analysis of a production line subject to degradation and preventive maintenance". *Reliability Engineering System Safety* 230:108906 https://doi.org/https://doi.org/10.1016/j.ress.2022.108906.

Yang, Z., D. Djurdjanovic, and J. Ni. 2008. "Maintenance scheduling in manufacturing systems based on predicted machine degradation". *Journal of intelligent manufacturing* 19:87–98.

AUTHOR BIOGRAPHIES

CAS LEENEN is a graduated student with a Master's degree from the Department of Mechanical Engineering at Eindhoven University of Technology. He is currently employed as an engineer for Joloda Hydraroll. His email address is casleenen@hotmail.com.

MICHAEL GEURTSEN is a doctoral candidate in the Department of Industrial Engineering of the Eindhoven University of Technology and a Sr. Business Process Analyst at Nexperia. His current research interests are in the area of modeling and optimization of maintenance in manufacturing systems. His email address is michaelgeurtsen@protonmail.com.

IVO ADAN is a Professor at the Department of Industrial Engineering of the Eindhoven University of Technology. His current research interests are in the modeling and design of manufacturing systems, warehousing systems and transportation systems, and more specifically, in the analysis of multi-dimensional Markov processes and queueing models. His email address is i.adan@tue.nl.

ZUMBUL ATAN is an Associate Professor at the Department of Industrial Engineering of the Eindhoven University of Technology. Her current research interests are in the interface of supply chain and revenue management with applications in retail industry. Her email address is z.atan@tue.nl.