Proceedings of the 2022 Winter Simulation Conference
B. Feng, G. Pedrielli, Y. Peng, S. Shashaani, E. Song, C.G. Corlu, L.H. Lee, E.P. Chew, T. Roeder, and P. Lendermann, eds.

DYNAMIC SCHEDULING OF MAINTENANCE BY A REINFORCEMENT LEARNING APPROACH - A SEMICONDUCTOR SIMULATION STUDY

Michael Geurtsen

Industrial Technology and Engineering Centre
Nexperia
Jonkerbosplein 52
Nijmegen, 6534AB, THE NETHERLANDS

Ivo Adan
Zumbul Atan

Department of Industrial Engineering
Eindhoven University of Technology
PO Box 513
Eindhoven, 5600 MB, THE NETHERLANDS

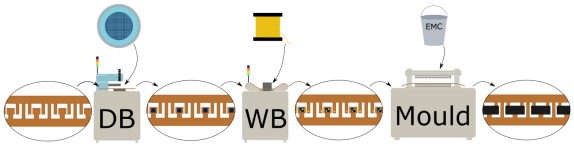
ABSTRACT

Scheduling in a semiconductor back-end factory is an extremely sophisticated and complex task. In semiconductor industry, more often than not, the scheduling of maintenance is underexposed to production scheduling. This is a missed opportunity as maintenance and production activities are deeply intertwined. This study considers the dynamic scheduling of maintenance activities on an assembly line. A policy is constructed to schedule a cleaning activity on the last machine of an assembly line such that the average production rate is maximized. The policy takes into account the given flexibility and the buffer content of the buffers in-between the machines in the assembly line. A Markov Decision Process is formulated for the problem and solved using Value Iteration and Reinforcement Learning Algorithms. In addition, for a real world case study, a simulation analysis is performed to evaluate the potential practical benefits.

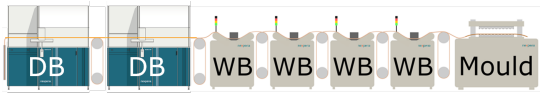
1 INTRODUCTION

This study is motivated by Nexperia's semiconductor back-end assembly facilities. At these facilities, Integrated Circuits (IC) are produced on assembly lines. The assembly process involves three steps in series: (1) die-bonding, (2) wire-bonding and (3) molding. Three different machines are connected by a support structure, referred to as the lead frame, that flows through the line. An illustration of this process is shown in Figure 1a. We consider an assembly line with multiple machines of each type and buffers in between each machine as depicted in Figure 1b. Note that the number of machines per type are unequal. This can be realized due to the difference in machine speed, i.e. the die-bonder produces twice as fast as the wire-bonder and the mold produces twice as fast as the die-bonder. In theory, if zero breakdowns occur and the machines operate exactly at their specified operating speed, a steady flow through the assembly line would be realised and buffers would be unnecessary.

The mold machine deteriorates quickly due to the residual dirt of the plastic material that is left behind after each injection molding step. Cleaning is performed to prevent the machine from an enormous breakdown and to realize quality products at all times. The facilities currently employ a time-based cleaning schedule, i.e. every N hours a cleaning activity is carried out. Performing cleaning in such manner is convenient and simple; the employees responsible for cleaning know exactly at what time they have to be at the machine. However, what happens during the N hours leading up to the cleaning activity differs every time due to the interdependence of all machines in the assembly line. For example, if one machine in the line breaks down and the buffer is not large enough to cover the down time, other machines in the line get affected. This effect can ripple through the line and can also be intensified by the behavior of other machines in the line. Due to this intricate process, the number of products produced during the N hour window is almost never constant. This is verified and shown in Figure 2. Data is gathered for six assembly lines and



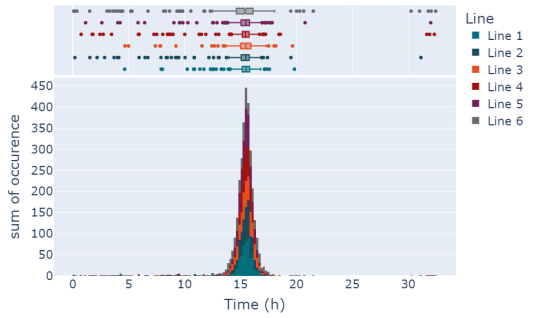
(a) Illustration of an assembly line with die-bond (DB) machine, wire bond (WB) machine and molding machine in series. On the DB the sawed wafer is loaded, on the WB a roll of wire is loaded (copper or gold) and the Mold uses EMC-shielding material to form an encapsulation for the IC. The lead frame flows through all machines and is shown in the figure between each machine.



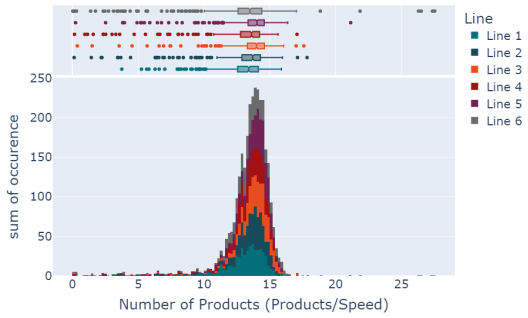
(b) Configuration of an assembly line considered in this study: two die-bonders, followed by four wire-bonders and ending with one mold machine. A buffer of varying size is positioned between each machine.

Figure 1: Assembly line.

Figure 2a shows the distribution of times between two consecutive cleaning activities. The figure shows that cleaning for these assembly lines is performed approximately every 16 hours. However, it is not always performed on time and outliers exist, most likely caused by excessive dirt and machine problems. Figure 2b shows the distribution of the number of produced products between two consecutive cleaning activities. One can observe that the distribution for the number of products is far wider than the distribution of time, which means that the variation in time cannot fully explain the variation in production. This supports the claim that the number of produced products during the N hour window is not constant. Consequently, the degree of deterioration varies. Therefore, it is more logical to transition to a cleaning schedule that is based on the number of produced products.



(a) Distribution of time.

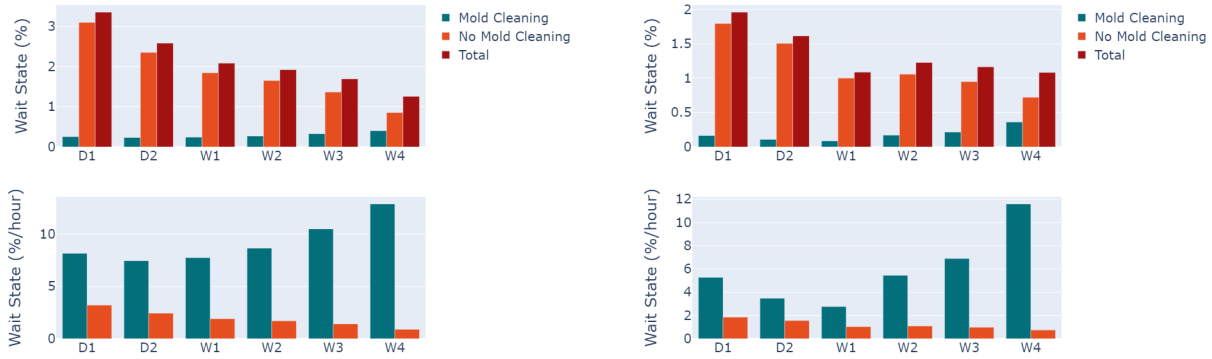


(b) Distribution of produced products.

Figure 2: Time (a) and produced products (b) between consecutive cleaning activities.

Surprisingly, Figure 2a shows that deviation from the time window is tolerated. It would be interesting to utilize this flexibility with the purpose of scheduling the cleaning activity to maximize throughput. Since, executing the cleaning activity exactly when the limit of the maximum produced products is reached can result in imperfect timing. For example, cleaning could be executed when the buffer before the mold is full. In such case, the machines upstream in the assembly line will have to wait, the extent of which obviously depends on the buffer contents upstream at the time of executing the cleaning. A cleaning activity takes on average 45 minutes to perform, which can lead to significant waiting times for the machines upstream. This is observed in the data extracted from two assembly lines, shown in Figure 3. The top histogram

shows the waiting time percentages during mold cleaning and during the period in between consecutive mold cleaning activities, i.e., normal production including breakdowns. In total, the machines are waiting more during the time outside mold cleaning time. However, the duration of mold cleaning is roughly 32 times smaller than the duration of the period of no mold cleaning. Therefore, the bottom histogram shows the normalized waiting percentages when taking into account the time. The impact of mold cleaning compared to normal production is unmistakable. The effect of mold cleaning also depends on how close the machine in the line is to the mold machine, i.e. the closer the machine, the more it is influenced. The lines on which this data analysis is performed have a buffer before the mold with a capacity of 1 hour and 20 minutes production. Clearly, if cleaning is executed with an (almost) empty buffer, the impact of cleaning would be negligible as the buffer has the ability to cover the down time. The machines upstream could continue their normal production without waiting on the cleaning. Nevertheless, the data obviously shows that this is almost never the case. As the goal of the buffers is to compensate for unavailability of the machines and thereby to increase the average throughput, it makes sense to further exploit the buffers by means of smart cleaning scheduling. This may lead to less waiting times during cleaning and thereby an increase in average throughput.



(a) Line 2.

(b) Line 3.

Figure 3: Waiting effects of the machines during mold cleaning and during normal production. The *D* represent the die-bonders, the *W* represents the wire-bonders.

Using buffers to dynamically schedule maintenance is also considered in (Gopalakrishnan et al. 2016). In this work, buffer levels are continuously monitored and serve as an indicator to determine which machine requires maintenance the most. Contrary to the cleaning problem described above, maintenance is initiated not based on number of produced products but instead reactively when a break-down occurs. In (Langer et al. 2010), in addition to reactive maintenance, preventive maintenance activities are considered as well and priority is assigned based on a bottleneck-approach. However, the preventive maintenance is time-based and can therefore not be scheduled. The studies by (Van der Duyn Schouten and Vanneste 1995) and (Kyriakidis and Dimitrakos 2006) share many similarities to the problem studied here. Both studies consider a two-machine single-buffer problem where inspections occur at discrete time epochs. Preventive maintenance can be initiated at any time epoch unless the machine is fully deteriorated, in which case corrective maintenance is required. While (Van der Duyn Schouten and Vanneste 1995) only consider a cost when maintenance is initiated while the buffer is empty, (Kyriakidis and Dimitrakos 2006) extend the cost function by including operating, maintenance and storage costs where operating costs depends on the age of the machine. (Kyriakidis and Dimitrakos 2006) prove the existence of an average-cost optimal policy and show that for a fixed-age machine and fixed buffer level, the optimal policy is of control-limit type. The problem presented in this study differs significantly from the previous studies as the machines are not expected to run production continuously throughout the entire planning horizon, but instead transition

continuously between *up* and *down* states. Therefore, the Markov Decision Process (MDP) of this problem is unique and suits the long-run average throughput objective very well. In view of the literature, the problem addressed in this study has not been studied before. For an extensive review on maintenance scheduling, refer to (Geurtsen et al. 2022). The remainder of the paper is organized as follows. In Section 2, a formal problem description is provided, the solution methods are proposed in Section 3 and in Section 4 we present our computational results. Finally, Section 5 concludes the paper with some recommendations.

2 PROBLEM DESCRIPTION

Based on the findings and characteristics described in the previous section, a formal problem statement can be presented for a simplified version of the real-world setting. We consider an assembly line with multiple machines and buffers in series. Each machine in the line is subject to random break downs. The duration of up times and down times of the machine are derived from historical data and follow an exponential distribution. Each machine has a unique maximum speed and buffers have a finite capacity B. The last machine in the line is prone to deterioration and requires preventive maintenance in the form of cleaning. The cleaning duration is assumed to be deterministic. Cleaning may be performed according to three different policies: (1) cleaning based on time, (2) cleaning based on produced products and (3) cleaning based on produced products with a tolerance on the limit of produced products. The first two policies are fixed, while in the latter policy, cleaning can be scheduled. For this last policy, it is assumed that in the long-run, on average the exact same number of cleaning activities are performed as in the second policy. This is explained by Figure 4 where the horizontal axe represents the number of produced products, P is the fixed interval of cleaning and F corresponds to the size of the flexibility on the interval P. Cleaning can be performed anywhere in the window F. Then, the limit for the next cleaning event will not be P products after the execution, but instead remains as planned. In doing so, the window between two consecutive cleaning activities w is not fixed but varies throughout the scheduling horizon as is seen in Figure 4. However, since the flexibility windows F are bound to the periodic interval P, in the long-run, the number of cleaning activities is the same as in a policy without flexibility. The only difference is that the length of the window w varies, where the maximum length is $P+F$ and the minimum length is $P-F$. Within the window F, the decision variable X can be either 0 or 1, representing respectively the decisions to perform cleaning or to do nothing. The objective is to find a policy that uses the flexibility window F and the buffer with capacity B to maximize the long-run average reward. In the remainder of this study, cleaning will be referred to as maintenance.

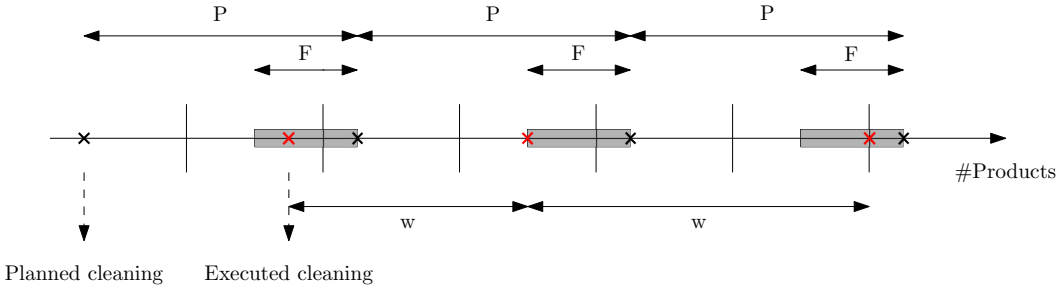


Figure 4: Schematic overview of when cleaning is performed based on produced products with flexibility. Black crosses mark the planned cleaning activities based on the fixed number of products, while red crosses mark actual executions of cleaning activities. Grey squares represent the flexibility window.

3 SOLUTION METHODS

In this section, several methods are described that not only establish promising methods to solve the problem close to optimal but also attempt to comprehend the impact of different maintenance strategies.

Accordingly, Section 3.1 proposes an MDP of the problem for a scaled-down setting of the real world. Then, Section 3.2 describes a method based on Value Iteration (VI) to solve the MDP close to optimal. Section 3.3 presents an algorithm related to Reinforcement Learning (RL) to determine its capabilities of solving the MDP with respect to the VI method. Finally, Section 3.4 describes a simulation model to capture the effect of multiple maintenance strategies for the real world setting.

3.1 Markov Decision Process

In a Markov Decision Process, a decision maker inhabits an environment which changes state randomly in response to action choices made by the decision maker. Formally, an MDP is defined by the tuple $\mathscr{M} = (\mathscr{S}, \mathscr{A}, \mathscr{R}, p)$, where $\mathscr{S}$ is a set of states, $\mathscr{A}$ is a set actions, $\mathscr{R}$ is a set of rewards, and $p : \mathscr{S} \times \mathscr{R} \times \mathscr{S} \times \mathscr{A} \longrightarrow [0,1]$ is the dynamics of the environment. At each discrete time step $t = 0, 1, 2, ...$ of a sequence of time steps, the decision maker receives an observation of a state from the environment $S_t \in \mathscr{S}$ and selects an action $A_t \in \mathscr{A}$, then receives from the environment a reward $R_{t+1} \in \mathscr{R}$ and the next state $S_{t+1} \in \mathscr{S}$, and so on. The transition dynamics can be described such that $p(s', r|s, a) = Pr(S_{t+1} = s', R_{t+1} = r|S_t = s, A_t = a)$ for all $s, s' \in \mathscr{S}, a \in \mathscr{A}$ and $r \in \mathscr{R}$.

The problem described in Section 2 can be transformed to an MDP. For the sake of simplicity, the problem is reduced to two machines and one buffer with finite capacity. The objective is to maximize the time that the second machine is producing output, i.e., to maximize the long-run average throughput. A state $s \in \mathscr{S}$ is composed of the production state of the first and the second machine, the buffer content of the buffer in-between the machines, the number of products to be produced until maintenance must be performed and the duration of maintenance when the second machine is in maintenance. Accordingly, the state $1-d-u-3-0$ implies that the buffer is filled with 1 product, the first machine is not producing (down), the second machine is producing (up), 3 more products need to be produced until maintenance must be executed and the duration of maintenance is still 0 as it has not yet started.

A schematic representation of the MDP is partly depicted in Figure 5. In this example, the buffer capacity is 2, the fixed interval of maintenance P (as described in Section 2) is 3, the flexibility window F is 1, the duration of maintenance is set to 1 and both machines have the same speed equal to 1 product per time step. In Figure 5, we start in state $0-u-u-2-0$. Since we have 2 products left until maintenance must be executed, we are not yet in the flexibility window. Therefore, the only action that can be taken is to continue, for which a reward of 1 is gained since the second machine is in production and products flow out of the assembly line. The action leads to four different states each with a specific probability p, depending on whether a machine breaks down or not. In state $0-u-u-1-0$, there are two actions that can be taken as the flexibility window is reached. When the action is chosen to perform maintenance, the next state becomes $0-x-pm-4-0$ where x can be either a down (d) or an up (u) state. The 4 in this state originates from the addition of P (which is 3 in this example) to the number of products we had left until the limit is reached (which was 1). When the action is chosen to be continue, one arrives at four different states again. In state $0-u-d-0-0$, the limit is reached and the only action available is to perform maintenance. States s and b represent a starved (buffer prior to the machine is empty) and a blocked (buffer after the machine is full) state, respectively. The proposed MDP has discrete steps that are formed by time as well as produced products. In a maintenance mode, state transitions are based on time, while in normal production mode, state transitions are represented by produced products. In the latter case, the step size is defined by the time step t multiplied by the machine speed. The machine speed is constant and equal for both machines. Accordingly, the time step t defines the size of the state space.

3.2 Value Iteration

A recursive solution approach from dynamic programming, the *value-iteration* algorithm (VI) is well suited for large-scale Markov decision problems. To estimate the best policy for this problem, a method appropriate for the long run average reward problem is adopted (Tijms 1986). This method recursively

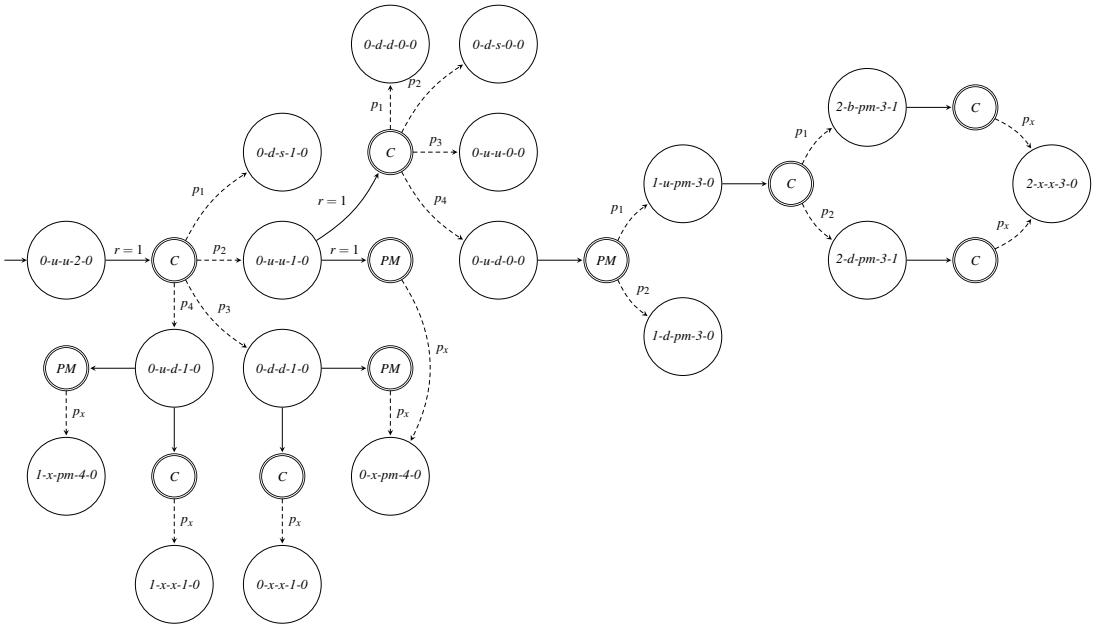


Figure 5: Schematic representation on a sub-part of the MDP. *C* represents a *continue* action, *PM* describes the *preventive maintenance* action and *x* refers to "multiple possible outcomes".

computes a sequence of value functions approximating the maximum average reward per unit time. The value functions provide lower and upper bounds on the maximum average reward and these bounds closely approximate the maximum reward rate, under a certain aperiodicity condition. The algorithm is shown in Algorithm 1. Here, V represents the value function and M_n and m_n the maximum and minimum bound, respectively, required for the termination condition.

Algorithm 1: Value Iteration for long run average reward

Algorithm parameters: tolerance number ε

Initialize V(s) $\forall s \in S$ arbitrarily (e.g., to zero, but not higher than the maximum reward)

while $0 \leq M_n - m_n \leq \varepsilon \cdot m_n$ **do**

$$V_n(s) = \min_{a \in A(s)} \left\{ R_s(a) + \sum_{s' \in S} p_{s,s'}(a) V_{n-1}(s') \right\}$$

$$m_n = \min_{s' \in S} \left\{ V_n(s') - V_{n-1}(s') \right\}$$

$$M_n = \max_{s' \in S} \left\{ V_n(s') - V_{n-1}(s') \right\}$$

end

The main procedure involves recursively computing the value function for all states and its actions, computing a lower and an upper bound with respect to the previous iteration and checking whether the improvement falls within the specified tolerance.

3.3 Reinforcement Learning

In reinforcement learning, an agent's interaction with its environment can be formalized by a finite MDP as described in Section 3.1. The agent selects an action $A_t \in \mathscr{A}$ using a certain behavior policy $b : \mathscr{A} \times \mathscr{S}$. One of the best known algorithms for reinforcement learning is the Q-learning algorithm (Watkins and

Dayan 1992). An action-value function, the Q-function, is learned which directly approximates the optimal action-value function q_*. The Q-function is represented by a table of state-action pairs. The best action A_t for a given state S_t can then be derived by taking the action that has the highest value in the table. Q-values in the table are incrementally updated by the following equation:

$$Q(S_t,A_t) \leftarrow Q(S_t,At) + \alpha\left[R_{t+1} + \gamma \max_a Q(S_{t+1},a) - Q(S_t,A_t)\right] \tag{1}$$

Here, α is the learning rate and γ is the discount factor, which determines to what extent future rewards are more or less important than immediate rewards. In this problem, large periods of not being able to schedule maintenance are alternated by short periods in which maintenance may be scheduled, i.e., when the agent is in the flexibility window. Thus, most of the time, no actions can be taken. Since rewards are given whenever the second machine is in a production state, i.e., when products flow out of the assembly line, taking an action does not result in an immediate reward. In addition, during the time from one action to the next action, a lot of uncontrollable activity has occurred due to the stochastic processes within the environment. Using Equation 1 to update the Q-function is therefore not suitable. Instead, it is better to focus on the reward rate, maximizing the average reward per step. The novel differential Q-learning algorithm presented in (Wan et al. 2021), represented in Algorithm 2, is used to let the agent learn to maximize the reward rate.

Algorithm 2: Differential Q-learning (one-step off-policy control)

Input: The policy b to be used (e.g., ε-greedy)
Algorithm parameters: step-size parameters α, η
Initialize $Q(s,a)\ \forall s,a;\bar{R}$ arbitrarily (e.g., to zero)
Obtain initial S
while *still time to train* **do**
  A $\leftarrow$ action given by b for S
  Take action A, observe R, S'
  $\delta = R - \bar{R} + \max_a Q(S',a) - Q(S,A)$
  $Q(S,A) = Q(S,A) + \alpha\delta$
  $\bar{R} = \bar{R} + \eta\alpha\delta$
  $S = S'$
end
return Q

Here $\bar{R}$ represents the scalar estimate of the optimal reward rate and η is a positive constant. For the behavior policy, the standard ε-greedy policy is used as a basis. In ε-greedy, a uniform random number between 0 and 1 is drawn and if the number is lower than the threshold ε, a random action is taken, otherwise the optimal (greedy) action is chosen. In this problem, to explore all the states in the flexibility window, multiple consecutive *continue* actions should take place to reach the end of the flexibility window. For this reason, the ε-greedy mechanism is extended such that whenever random actions are taken, the *continue* action is taken n times in a row and alternated by taking the *maintenance* action n times in a row.

3.4 Discrete-Event Simulation

A *Discrete-Event Simulation* (DES) model of the assembly line is developed to simulate the behavior of the assembly line. This method is used to understand the impact of different maintenance strategies on the throughput of the assembly line. The simulation model described by (Wilschut et al. 2014) and (Adan et al. 2018) is adopted. The assembly line comprises several machines with buffers in between, as depicted in Figure 1b. In the work of (Wilschut et al. 2014) the DES model is used with the goal of finding process

errors that are most critical to the line throughput, while the study by (Adan et al. 2018) focuses on using the DES model for optimizing the buffer capacities. In this study, the DES model will be used to analyze the effects of different maintenance strategies. For details on the DES model, readers are referred to the aforementioned studies. The basics are as follows: a machine has two main states, *up* and *down*. A buffer has a fill rate which is determined by the difference in speed of the direct neighbouring machines. When the buffer is full, the machine prior to the buffer changes its state to *blocked* and adjusts its speed to the speed of the machine after the buffer. Similarly, when the buffer is empty, the machine after the buffer changes its state to *starved* and adjusts its speed to the machine prior to the buffer. Finally, as described in Section 2, the last machine requires mandatory maintenance from time to time, which is denoted by the state *maintenance*.

As an input, each machine in the DES model has a maximum speed and a unique machine behavior. Ideally, the DES model should mimic the real world as close as possible. Therefore, machine behavior is derived by acquiring distributions of *up* times and *down* times from real world data. Similarly, maximum machine speeds can be derived from real world data by extracting the production times and the produced products. Furthermore, each buffer in the DES model demands a maximum buffer capacity, which can simply be derived from the real world setting.

4 RESULTS

In this section, multiple experiments are performed. First, experiments are carried out to explore whether RL can be a promising method for the real world problem. Then, a case-study on real world production lines is conducted to mark the potential benefits of adopting different maintenance strategies for the studied problem. The MDP and VI described in Section 3 are coded in python 3.8 and the DES model for the real-world case study is coded in C# 10.0. All experiments are run on a computer with an Intel Core i5-8600K processor running at 3.60 GHz and 16 GB of RAM memory.

4.1 Simplified Case Study For Reinforcement Learning

Before the real world problem is examined, it is sensible to get a feeling for the problem and its properties. For this reason, the real world problem is reduced in order to generate near optimal solutions. By reducing the problem scale, the MDP formulation described in Section 3.1 can be applied. To solve the MDP and generate near optimal solutions, the VI algorithm proposed in Section 3.2 is implemented. For this toy problem, the near-optimal policy derived from the VI algorithm is compared to a fixed method where maintenance is performed only when the limit of the number of produced products is reached. For the experiment, an analysis is performed over multiple instances. The instances are generated such that a wide variety of problem sizes can be examined. A total of 863 unique instances are created, formed by combining the values in the ranges of parameters shown in Table 1. The speed of the machines is set to 20 products per second and a time step of 200 seconds is chosen. The time step is chosen such that a moderate-sized state space is generated. Then, the step size is equal to $20 \cdot 200 = 4000$ products. Historical data is used to generate realistic transition probabilities. As exponential up and down times are assumed, the probability of ending an up or down state for a time step of 200 seconds can be determined for each machine type using the historical data. These are 0.26 (up), 0.74 (down) and 0.15 (up), 0.85 (down) for the first and the second machine respectively. With these base values, all other transition combinations can be derived.

The tolerance in the VI algorithm described in Algorithm 1 is set to 0.01%. An example of a policy derived with the VI algorithm is shown in Figure 6a. It can be seen that the policy suggests to never perform maintenance when both machines are up and running, always do maintenance if the buffer is empty and the second machine is starved, always do maintenance when the first machine is down, the second machine is up, and the buffer has 1 step left until being empty, and to perform maintenance if the number of products until reaching the limit falls below $12k$ in case the first machine is blocked and the second machine is down. Interestingly, stepping curves are observed for the case where the second

Table 1: Toy problem instances. Here k represents 1000 products.

Parameter		**Values**
Fixed maintenance interval	P	{ 100k, 120k, 140k, 160k, 180k, 200k }
Flexibility window	F	{ 40k, 60k, 80k }
Buffer capacity	b	{ 12k, 16k, 20k, 24k, 28k, 32k, 36k, 40k }
Maintenance duration	d	{ 600, 1200, 1800, 2400, 3000, 3600 }

machine is down and the first machine is either up or down. Such policies are derived for all the 863 instances and, using Monte Carlo simulation, compared to the policy where maintenance is performed based on the number of produced product without flexibility, i.e. fixed production based maintenance. Results are shown in Table 2. An average improvement of 1.13% is achieved with a smart policy derived from VI, where improvements up to 3.32% can be realized in some cases. Interestingly, the average buffer content increases slightly which contradicts common reasoning that increased throughput leads to less mean buffer content. However, the buffer content at the start of executing maintenance and the average content during maintenance is significantly lower with VI. This can explain the increase in throughput despite an increase in average buffer content. The policies derived with VI are able to utilize the given flexibility in combination with buffer behavior such that throughput increases are possible. Improvements are highest for the problem instances where the difference between the period P and flexibility window F is small and the buffer capacity b is small. The maintenance duration d does not seem to have a large effect.

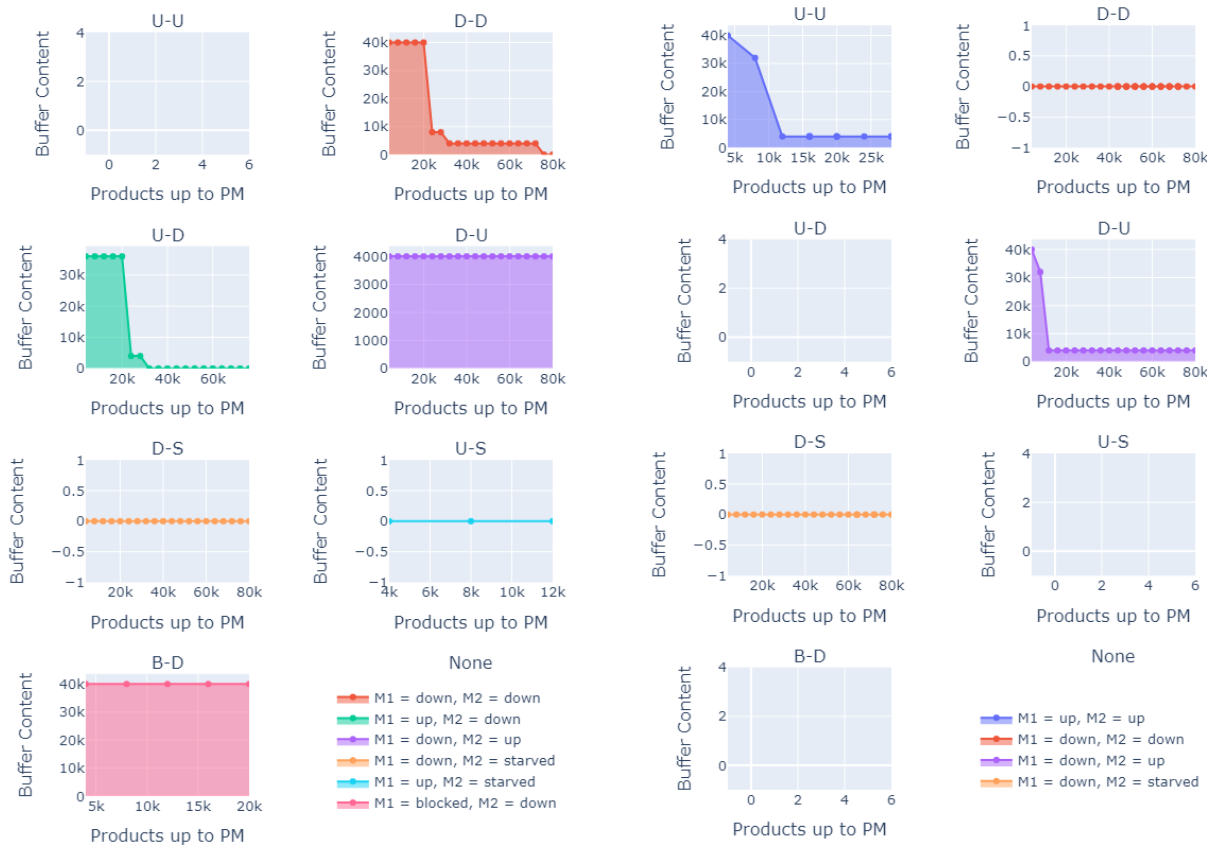
Table 2: Results of flexible maintenance versus fixed maintenance for the toy problem instances. *Throughput* indicates the improvement in average production rate, *Buffer* represents the average buffer content while *Buffer start* represents the average buffer content at the moment of starting maintenance and *Buffer during* indicates the buffer content during the entire maintenance duration. *Window* represents how much of the flexibility window is used on average, compared to no flexibility.

	Throughput (%)	**Buffer (%)**	**Buffer start (%)**	**Buffer during (%)**	**Window (%)**
Minimum	0.002	5.701	2.620	2.274	-23.065
Mean	1.129	0.937	-31.875	-7.928	-56.158
Maximum	3.318	-3.112	-82.143	-37.238	-90.225

The *Differential Q-learning* algorithm presented in Section 3.3 is applied on the same problem instances. For a fair comparison, for each instance the learning time for the agent is set to the time the VI method took to reach its solution for that particular instance. The values for the learning rate α and the positive constant η are tuned manually and set to 0.01 and 0.2 respectively. Results from the experiments showed a 0.06% deviation in throughput performance with respect to the results obtained with the VI algorithm. This demonstrates the potential benefit of RL for this problem. The policy for the same instance as shown in Figure 6a is depicted in Figure 6b. Interestingly, a slightly higher throughput of 0.09% is realized with RL for this particular instance with a policy which is significantly different and less intuitive. Since, the RL agent in this instance generally suggests that it is never good to performance maintenance when the second machine is not producing, except when the first machine is also not producing in which case maintenance can be executed only if the buffer is empty. A stepping curve is observed only when the second machine is in production, which is the exact opposite of the policy derived with VI. This shows multiple good policies exist for this problem and that RL serves as a promising solution approach with good exploration abilities.

4.2 Real World Case Study

As explained in Section 2, the current practice within the semiconductor manufacturing facility is to perform maintenance in a fixed time-based manner. This approach will be compared to a method where maintenance



(a) Policy derived with the VI algorithm.

(b) Policy derived using RL.

Figure 6: Policies for the case: $p = 200k$, $f = 80k$, $b = 40k$, $d = 3000$. Here U represents a production state, D represents a break-down state, B represents a blocked state and S represents a starved state.

is performed after producing a fixed number of products. In addition, this method is extended by allowing some flexibility on the number of produced products and acting on the buffer content with a simple heuristic.

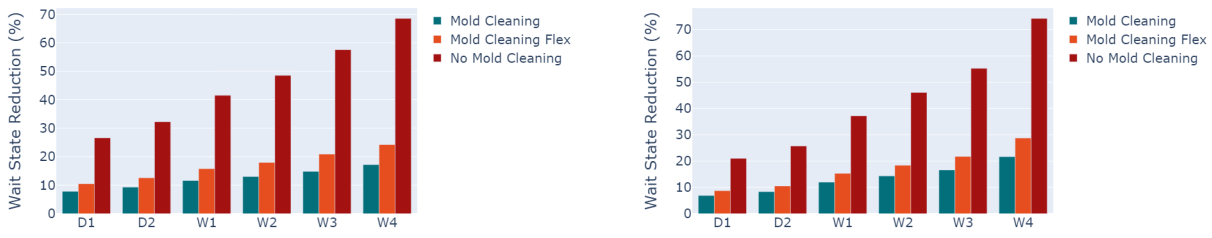
The experiments are conducted by using the DES model described in Section 3.4. Real world data of 11 assembly lines from Nexperia is used to extract the machine speeds and the unique *up* and *down* behavior. All 11 lines have a similar configuration, i.e. they are comprised by 7 machines, of which the 1st and 2nd are die-bonders, the 3rd till the 6th are wire-bonders and the last machine is an injection-molding machine. As described in Section 1, the numbers of the different machine types that cover the complete assembly line are calibrated such that a steady flow through the line is realized. Buffer contents are the same for each line and equal to $25k$, $9k$, $25k$, $25k$, $25k$, $120k$, in sequential order, where $120k$ refers to the buffer before the last machine. Machines in the DES model sample *up* and *down* times empirically from historical data. In addition, settings regarding the maintenance activity on the last machine are as follows: (1) time-based maintenance occurs every 12 hours, (2) production-based maintenance occurs after 1.15 million products are produced (this corresponds to 12 hours as the theoretical speed of the injection molding machine is 96k products per second) and (3) buffer-based maintenance occurs after the production of 1.15 million produced products with a varying flexibility of 100k to 600k products, see Section 2 for more details on this method. The maintenance duration is assumed to be constant and equal to 45 minutes.

For the buffer-based maintenance policy, a simple heuristic is tested to determine the effect of scheduling maintenance in relation to fixed production based maintenance. The heuristic is based on a threshold of the buffer content. Whenever maintenance is allowed to be executed, i.e. we have arrived in the flexibility

window, the buffer content of the latest buffer in the line is monitored. Maintenance is only performed in case the buffer content falls below the specified limit, otherwise maintenance is postponed. For the experiment, two different buffer thresholds are examined. The first threshold corresponds to the scenario in which the buffer is nearly empty, which is arbitrarily chosen to be 2000. The second threshold is based on the theoretical flow of throughput in the line which is equal to the speed of the mold machine, i.e. $96k$ products per hour. During a maintenance, if there was an infinite buffer before the mold machine and the other machines would run perfectly, $78k$ products could be produced during the cleaning activity. As the buffer capacity is actually $120k$, setting the threshold at $48k$ products would suffice to cover the cleaning activity. Table 3 summarizes the results of the experiments. In addition to the cases described previously, a comparison between time based maintenance and doing no maintenance at all, is carried out. The result of this comparison denotes the improvement that is maximally achievable. Table 3 shows that the maximum realisable improvement equals 3.40% whereas moving from time based maintenance to fixed production based maintenance yields an improvement of 1.01%. With a simple heuristic, this improvement can be further extended to 1.27% for a $600k$ flexibility window and a $2k$ buffer threshold. The table clearly shows that the more flexibility is provided, the higher the improvement. Furthermore, a $2k$ buffer threshold appears to perform better than an $18k$ threshold. There are differences among the assembly lines, with a maximum and minimum improvement of 1.18% and 0.59%, respectively. Figure 7 shows, in a similar style to Figure 3, the improvement in waiting state of all the machines for the cases no maintenance, production-based maintenance and production-based maintenance with a flexibility (case $600k$ window and $2k$ buffer threshold). Reductions are determined with respect to time-based maintenance. The simulations show that significant improvements in machine waiting state can be realized.

Table 3: Improvement in percentages for the simulation of different maintenance policies with respect to time based maintenance. Here, *PB* represents production based cleaning.

Line	**No PM**	**PB**	**PB Flex 100k**		**PB Flex 200k**		**PB Flex 300k**		**PB Flex 400k**		**PB Flex 500k**		**PB Flex 600k**	
			2k	48k	2k	48k	2k	48k	2k	48k	2k	48k	2k	48k
1	3.45	1.00	1.05	1.04	1.08	1.09	1.15	1.12	1.19	1.12	1.23	1.18	1.28	1.24
2	3.69	1.08	1.11	1.13	1.20	1.20	1.27	1.27	1.36	1.31	1.43	1.36	1.42	1.39
3	2.98	0.95	0.98	0.98	1.08	1.02	1.12	1.04	1.12	1.11	1.12	1.17	1.22	1.16
4	4.52	1.14	1.18	1.17	1.23	1.22	1.25	1.27	1.33	1.31	1.36	1.36	1.39	1.39
5	4.02	1.12	1.11	1.05	1.14	1.19	1.21	1.16	1.23	1.26	1.26	1.30	1.29	1.30
6	3.11	0.95	1.00	0.93	1.02	1.00	1.04	1.04	1.15	1.07	1.20	1.10	1.26	1.21
7	3.42	1.05	1.00	1.09	1.10	1.09	1.10	1.14	1.22	1.15	1.19	1.20	1.25	1.25
8	1.47	0.59	0.59	0.54	0.64	0.56	0.66	0.65	0.71	0.64	0.77	0.71	0.74	0.72
9	3.78	1.04	1.08	1.10	1.08	1.13	1.18	1.20	1.21	1.19	1.19	1.22	1.25	1.28
10	3.74	1.18	1.23	1.23	1.32	1.27	1.39	1.39	1.37	1.38	1.46	1.41	1.57	1.50
11	3.23	1.03	1.04	1.02	1.10	1.10	1.15	1.13	1.19	1.19	1.24	1.24	1.28	1.32
Mean	3.40	1.01	1.03	1.03	1.09	1.08	1.14	1.13	1.19	1.16	1.22	1.20	1.27	1.25



(a) Line 2.

(b) Line 3.

Figure 7: Improvement in the waiting state with respect to maintenance based on fixed time.

5 CONCLUSIONS AND FUTURE WORK

This study considers a problem where maintenance must be executed on the last machine of an assembly line. Multiple analyses are performed in which different maintenance strategies are compared, i.e., time-based, fixed interval production-based and flexible production-based. For the simplified problem of two machines and one buffer, near-optimal policies are derived for the flexible production-based method using Value Iteration (VI). Compared to fixed-interval maintenance, throughput improvements can be realized. A Reinforcement Learning (RL) approach applied on the same problem showed that the RL agent achieves similar results as the VI method. Simulations on the real world problem demonstrated that significant throughput improvements can also be realized in practice. In future work, RL and deep RL can be applied on the real world problem to evaluate whether further increases are possible. In addition, the problem can be extended such that multiple assembly lines, which share a common limited resource, can be considered.

REFERENCES

Adan, J., S. Sneijders, A. Akcay, and I. J. Adan. 2018. "Re-enactment Simulation for Buffer Size Optimization in Semiconductor Back-end Production". In *2018 Winter Simulation Conference*, edited by M. Rabe, A. A. Juan, N. Mustafee, A. Skoogh, S. Jain, and B. Johansson, 3432–3440. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Geurtsen, M., J. B. Didden, J. Adan, Z. Atan, and I. Adan. 2022. "Production, Maintenance and Resource Scheduling: A Review". *European Journal of Operational Research*. https://www.sciencedirect.com/science/article/pii/S0377221722002673, accessed 22 March 2022.

Gopalakrishnan, M., A. Skoogh, and C. Laroque. 2016. "Buffer Utilization based Scheduling of Maintenance Activities by a Shifting Priority Approach - a Simulation Study". In *2016 Winter Simulation Conference*, edited by T. M. K. Roeder, P. I. Frazier, R. Szechtman, E. Zhou, T. Huschka, and S. E. Chick, 2797–2808. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Kyriakidis, E., and T. Dimitrakos. 2006. "Optimal Preventive Maintenance of a Production System with an Intermediate Buffer". *European Journal of Operational Research* 168(1):86–99.

Langer, R., J. Li, S. Biller, Q. Chang, N. Huang, and G. Xiao. 2010. "Simulation Study of a Bottleneck-based Dispatching Policy for a Maintenance Workforce". *International Journal of Production Research* 48(6):1745–1763.

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

Van der Duyn Schouten, F., and S. Vanneste. 1995. "Maintenance Optimization of a Production System with Buffer Capacity". *European Journal of Operational Research* 82(2):323–338.

Wan, Y., A. Naik, and R. S. Sutton. 2021. "Learning and Planning in Average-Reward Markov Decision Processes". In *38th International Conference on Machine Learning*. July 18th-24th, Proceedings of Machine Learning Research, 10653–10662.

Watkins, C. J. C. H., and P. Dayan. 1992. "Q-learning". *Machine Learning* 8(3):279–292.

Wilschut, T., I. J. Adan, and J. Stokkermans. 2014. "Big Data in Daily Manufacturing Operations". In *2014 Winter Simulation Conference*, edited by A. Tolk, S. Y. Diallo, I. O. Ryzhov, L. Yilmaz, S. J. Buckley, and J. A. Miller, 2364–2375. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

AUTHOR BIOGRAPHIES

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.