Wu, C.-H., Yao, Y.-C., Dauzère-Pérès, S., & Yu, C.-J. (2020). Dynamic dispatching and preventive maintenance for parallel machines with dispatching-dependent deterioration. *Computers & Operations Research*, *113*, 104779. doi:https://doi.org/10.1016/j.cor.2019.104779

Dynamic Dispatching and Preventive Maintenance for Parallel Machines with Dispatching-dependent Deterioration

Cheng-Hung Wu, Yi-Chun Yao, Stéphane Dauzère-Pérès, and Cheng-Juei Yu

Abstract

A dynamic decision model that coordinates dispatching and preventive maintenance decisions for failure-prone parallel machines in make-to-order (MTO) production environments is developed in this research. The primary objective is to minimize the weighted long-run average waiting costs of MTO systems. Two common but seldom studied stochastic factors, namely, the dispatching-dependent deterioration of machines and machine-health-dependent production rates, are explicitly modeled in the proposed dynamic dispatching and preventive maintenance (DDPM) model. Although the DDPM model is developed using Markov decision processes, it is equally effective in non-Markovian production environments. The performance of the DDPM model is validated in Markovian and non-Markovian production environments. Compared with several methods from the literature, simulation results show an improvement of at least 45.2% in average job waiting times and a minimum reduction of 48.9% in average machine downtimes. The comparison results between the optimal dynamic dispatching policies with and without coordinated preventive maintenance show that performance improvement can be mostly attributed to the coordination between preventive maintenance and dispatching decisions.

Introduction and Problem Formulation

This research investigates joint dispatching and preventive maintenance optimization problems for parallel machines with dispatching-dependent deterioration. In the literature, unrelated parallel machines are flexible machines that can process the same group of jobs but have different processing time distributions. Parallel machines that can perform various operations or produce different products are widely used to satisfy the requirements of product diversification. In the current research, the varying production rates of parallel machines can result from machine deterioration. In practice, the deterioration of machines frequently depends on dispatching decisions (Kazaz et al., 2013). For example, a computer numerically controlled (CNC) machine may be used to perform machining operations on different materials and products. However, various materials may stress the cutting tools of CNC machines in different ways and lead to varying equipment health deterioration processes, which are defined as *dispatching-dependent machine health deterioration* in this research.

When machine health deteriorates over time, the accumulated wear on machines or tools may impact product quality, cause additional rework, reduce production efficiency, and eventually lead to machine failure (Kaufman and Lewis, 2007; Zhang and Daigle, 2012; Kao et al., 2018). For example, when an operation with high quality requirements is assigned to a deteriorated machine, the high rework rate of the deteriorated machine may lead to a low throughput rate and high production costs. By contrast, the same deteriorated machine may still perform operations with low quality requirements and not result in high rework rates. Consequently, different production rates can be observed in the same group of machines due to varying equipment health, which is defined as *machine-health-dependent production rates*.

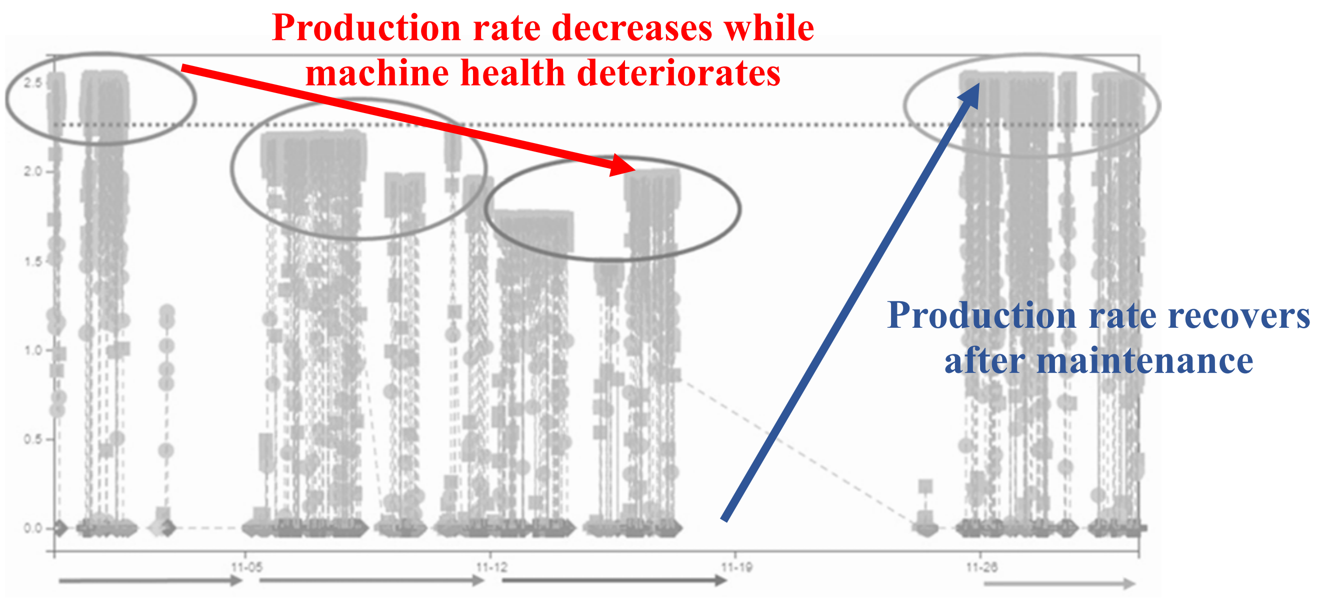


Figure 1 Production rate decreases with machine health in semiconductor testing and assembly processes

Dispatching-dependent deterioration and health-dependent production rates are common stochastic factors in manufacturing systems. For example, diverse materials stress the tool bits of CNC machines in different ways and cause varying deterioration rates in machining processes. When deteriorated tools are used, a low yield quality, such as a rough surface, is expected. Consequently, high rework/scrap rates hinder production efficiency. Similar machine–product fitness examples can be found in the high-tech manufacturing industry.

Yu et al. (2002) discovered this problem in the printed circuit board (PCB) manufacturing line of the Japanese electronics industry. As shown in Figure 1, one of our industry partners from the semiconductor testing and assembly industry also found similar deterioration and production rate changes in their production line. In the figure, the throughput rates and health of a machine deteriorate over time, and throughput rate recovery is observed after machine maintenance.

Most existing scheduling methods use metaheuristic algorithms to find the optimal schedule for a finite set of jobs on a fixed number of unrelated machines. The production time of a job is frequently assumed to be a machine-dependent constant, and production time uncertainties are disregarded. Yu et al. (2002) developed Lagrangian relaxation methods for scheduling unrelated parallel machines in the printed wire board industry. Chen and Wu (2007) and Chen (2015) presented heuristic scheduling algorithms to minimize the total weighted completion time of unrelated parallel machines. Rezaeian (2003), Liaw et al. (2003), Afzalirad and Rezaeian (2015), and Rezaeian (2016) developed metaheuristic algorithms to optimize the machine loads of unrelated parallel machines. Lin and Ying (2014) developed an artificial bee colony algorithm to minimize the maximum completion time of jobs, i.e. the makespan, when the setup time between job switching is sequence-dependent. To schedule jobs on unrelated parallel batching machines, Kao et al. (2018) propose two mixed integer linear programs that balance between productivity and quality risk.

The dynamic dispatching and dynamic preventive maintenance of failure-prone machines are closely related to our work. Stochastic optimization methods are frequently used in dynamic production control problems because machine deterioration and failure are uncertain events in these environments. Kaufman and Lewis (2007) asserted that optimal maintenance policies depend on workload and machine health. Cai et al. (2013) studied work-dependent deterioration in an M/G/1 queueing system with two types of job and a single deteriorating machine. Celen and Djurdjanovic (2015) developed heuristic algorithms for coordinating product sequencing and maintenance under operation-dependent deterioration. Wu et al. (2008) proposed heuristic algorithms to minimize the weighted cycle time in a serial production line. Zhou et al. (2009) studied preventive maintenance scheduling problems to minimize short-term maintenance costs using dynamic programming. Cui et al. (2014) proposed a robust three-phase dynamic production scheduling and maintenance algorithm for a single flexible machine. Borrero et al. (2013) develop Markov decision process

(MDP) models for systems with state-dependent processing rates. Zhang and Zeng (2015) and Zhou et al. (2015) proposed different heuristic methods for minimizing the opportunity costs of maintenance in multi-unit series systems. The literature in this area shows that dynamic scheduling and preventive maintenance are challenging research problems even for a single machine.

Thus, the contribution of the current research is the joint optimization of dispatching and preventive maintenance decisions for parallel machines when dispatching-dependent deterioration and health-dependent production rates are considered. Coordination between dispatching and preventive maintenance decisions is critical for such systems, and both factors are explicitly modeled in our dynamic dispatching and preventive maintenance (DDPM) model. To our knowledge, no previous research has yet considered both stochastic factors for parallel machines.

DDPM Model

This research considers a make-to-order (MTO) production system that produces I products on J multifunctional machines. Let $i \in \{1,2,\dots,I\}$ represent a type of product. Type i demands join queue i and wait to be manufactured (as shown in Figure 2) by one of the J machines. To consider the random deterioration of the machines explicitly, stochastic optimization methods must be adopted. In the literature, two stochastic optimization methods, namely stochastic programming (SP) and MDP, are widely used in stochastic production control problems. In the current research, given that dispatching and preventive maintenance decisions must be taken many times a day, a long planning horizon with many decision epochs is required. In Feinberg and Shwartz (2002), Wallace and Ziemba (2005), and Lee and Meng (2014), the computational complexity of multiple recourse stochastic programming grows exponentially with the planning horizon and is computationally intractable in our problem. Thus, to consider a planning horizon with many decision epochs, we develop our DDPM model using MDP, and the demand arrival, production, and machine failure processes are assumed to follow Poisson or Markov processes to satisfy the required Markov assumptions of the MDP models. Although the model is developed and solved under Markov assumptions, the optimal control policies generated by the DPPM model are validated in Markovian and non-Markovian simulation environments. In non-Markovian environments that do not comply with the Markov assumptions, numerical results indicate a similar performance advantage over other methods.

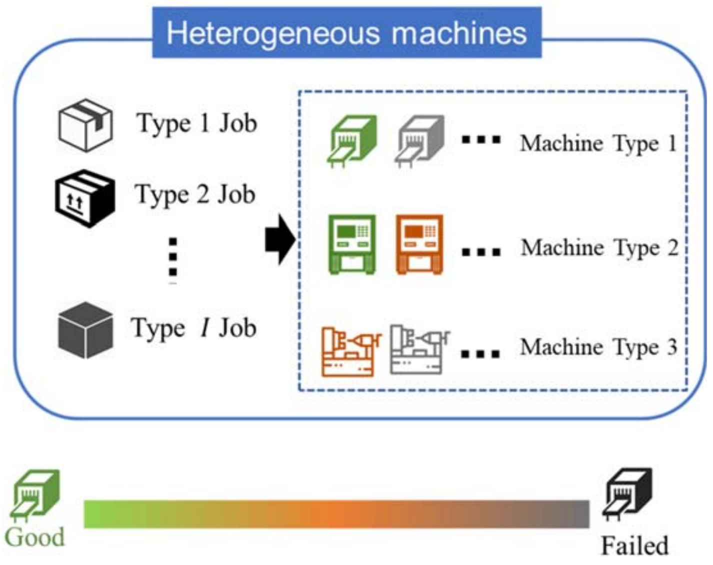


Figure 2 Production system with failure-prone parallel machines

Thus, the demand arrivals are assumed to follow Poisson arrival processes with product-specific arrival rates λ_i. Each machine has K different machine health states, and production rates depend on the product and machine health state. Let $k \in \{0,1,\dots,K-1\}$ be the real-time health state of a machine. A machine in health state k is healthier than a machine in health state k' when $k > k'$. Moreover, $k = K-1$ indicates the best health state, whereas $k = 0$ indicates the failure state of a machine. We define μ_i^k as the production rate for type i when a machine is under health state k. To model the influence of machine deterioration, we assume that $\mu_i^k \geq \mu_i^{k'}$ when $k > k'$ (i.e., the production rate does not increase when machine health deteriorates).

Machine health deterioration is dispatching-dependent and is assumed to follow a continuous-time Markov chain. Different job assignments result in various machine deterioration rates. That is, when different types of job are assigned to a machine, the machine health deterioration rate changes accordingly. Let β_i^k denote the machine health deterioration rate between health states k and $k-1$ when a machine is assigned to process type i products.

A decision maker can choose to conduct optional preventive maintenance when machine deterioration is observed. Maintenance decisions immediately place a machine in maintenance state. Once maintenance begins, the process takes an exponentially distributed amount of time with rate γ_k to complete. The preventive maintenance rate is assumed to depend on the health of a machine at the time when the preventive maintenance decision was made. If no preventive maintenance is conducted, then a machine may eventually fail (i.e., $k = 0$). When machine failure is observed, an exponentially distributed repair time with rate γ_0 is required to mend the machine.

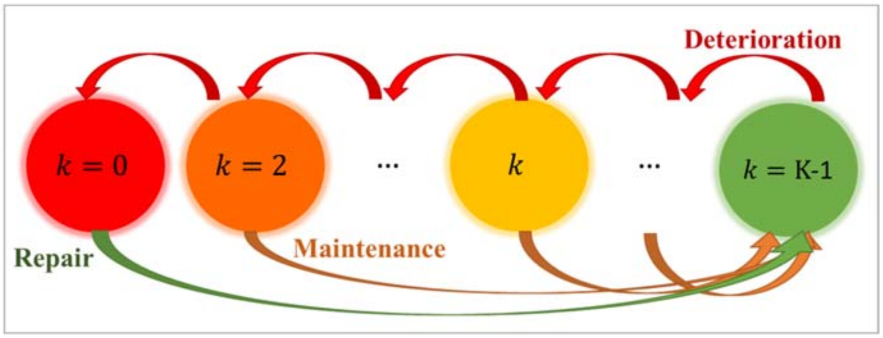


Figure 3 Transition diagram between machine health states

In this section, the DDPM model is developed to coordinate dispatching and preventive maintenance decisions in systems with multiple product types and machines. Optimal dispatching and maintenance decisions depend on the real-time queue and machine health states due to the dynamic nature of a system (Kaufman and Lewis, 2007). Thus, we define the state variable of the dynamic optimization model at time t as $s_t = (q_t, x_t, y_t)$, where $q_t = (q_1, q_2, \dots, q_I)$, $x_t = (x_0, x_1, \dots, x_{K-1})$, and $y_t = (y_0, y_1, \dots, y_{K-1})$. In these equations, q_i denotes the real-time queue length of product i, y_k denotes the number of available machines with health state k, and x_k represents the number of machines undergoing preventive maintenance or repair under health state k. Moreover, $\sum_k (x_k + y_k) = J$ because the system has exactly J machines.

Coordination between maintenance and dispatching decisions is critical for the current research because manufacturing and machine health deterioration processes are dispatching-dependent. Thus, the following decision variables are defined and considered for all states s_t.

(1) Dispatching decision: a_t

Let $a_t = (a_0, a_1, \dots, a_{K-1})$ be the dispatching decision at time t, where $a_k = i \in \{1, 2, \dots, I\}$ represents the dispatching decision for machines with health state k. If $a_k = i$, then the highest priority is assigned to product i on health state k machines.

(2) Preventive maintenance decision: b_t

Let b_t =$(b_0, b_2, \dots, b_{K-1})$ be the preventive maintenance decision at time t, where $0 \le b_k \le y_k$ is a non-negative integer decision variable that represents the state-dependent preventive maintenance decision on health state k machines. When $b_k > 0$, preventive maintenance begins on b_k of y_k available machines that are currently in health state k. Let $b_{K-1} = 0$ to prevent unnecessary calculation because machines with perfect health state require no maintenance. We also constantly assume $b_0 = y_0$ because corrective maintenance begins immediately on all failed machines.

The objective of the DDPM model is to minimize the weighted long-run average waiting costs of an MTO production system. Let h_i be the holding cost rate of product i (i.e., waiting cost h_i is accrued for every unfulfilled type i demand per unit time). Notably, minimizing average cycle time by setting $h_i = 1$ for all i is a special case of the general DDPM model.

Uniformization is applied to transform the original continuous-time Markov decision problem into an equivalent discrete-time problem (Serfozo, 1978). For a system with I products/J machines/K machine health states, the uniformization rate φ can be defined as

$$\varphi = \sum_{i=1}^{I} \lambda_i + J \times \left\{ \sum_{k=0}^{K-1} \left[\sum_{i=1}^{I} \left(\mu_i^k + \beta_i^k + \gamma_k \right) \right] \right\}.$$

Table 1 provides the summary of all the variables and notations used in the DDPM model.

Table 1 Notations and definitions used in the DDPM model

Notation	Definition
i	Product type, $i \in \{1, 2, \dots, I\}$
I	Number of product types
j	Machine, $j \in \{1, 2, \dots, J\}$
J	Number of machines
k	Machine health state, $k \in \{0, 1, \dots, K-1\}$ $K-1$: perfect health, 0: failed
K	Number of different machine health states
q_i	Queue length of product type i, $q_i \in \mathbb{N} \cup \{0\}$
x_k	Number of health state k machines under maintenance/repair
y_k	Number of available health state k machines
s_t	State variable, $s_t = (q_t, x_t, y_t) \in \mathrm{S}$, where $q_t = (q_1, \dots, q_I)$, $x_t = (x_0, \dots, x_{K-1})$, and $y_t = (y_0, \dots, y_{K-1})$

S	State space, set of all possible system states
h_i	Holding cost rate of product i
λ_i	Demand arrival rate of product i
μ_i^k	Production rate of product i on a health state k machine
β_i^k	Machine health deterioration rate from health state k to $k-1$ when a machine is assigned to process product i
γ_k	Maintenance/repair rate of a health state k machine
φ	Uniformization rate

After applying uniformization and transforming the continuous-time problem into a discrete-time equivalent model, we can focus on the discrete time points where state transitions are observed. The expected cost between two consecutive decision epochs can be defined as

$$\mathrm{C}(s_t) = \frac{1}{\varphi} \times \sum\nolimits_{i=1}^{I} (h_i q_i).$$

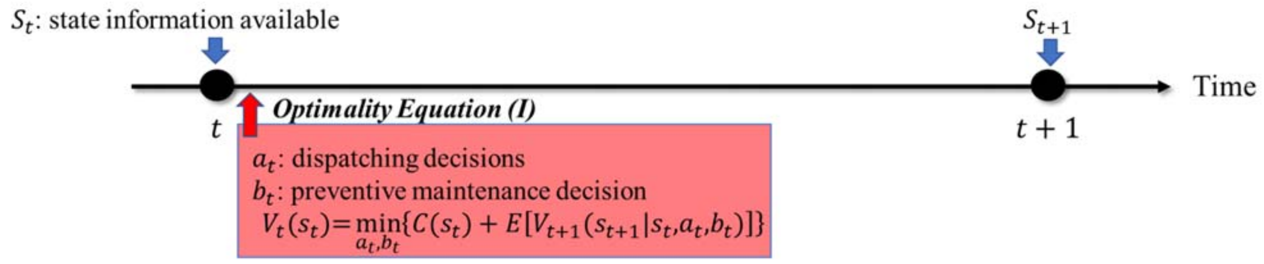
If we disregard preventive maintenance decision b_t and focus on the one-step transition probability function under s_t and dispatching decision a_t, then the one-step transition probability without preventive maintenance can be defined as $P(s_{t+1} = (x', y', q') | (q_t, x_t, y_t), a_t)$ for each of the following five transition types (b_t and its impact on the system state are discussed in a later section).

Table 2 Transition probabilities $P(s_{t+1} | s_t, a_t)$

Event/Transition Type	**Notation**	**Probability**
Type i demand arrival	$P_{1,i}$	$\frac{\lambda_i}{\varphi}$
Type i service completion on health state k machines	$P_{2,i,k}$	$\frac{y_k \times \mu_i^k}{\varphi}$
Deterioration of a health state k machine when producing product i	$P_{3,i,k}$	$\frac{y_k \times \beta_i^k}{\varphi}$
Maintenance/repair completion of a health state k machine	$P_{4,k}$	$\frac{x_k \times \gamma_k}{\varphi}$
Dummy transition (to make the probability functions summed up to 1)	P_5	$1 - \sum\nolimits_{i=1}^{I} P_{1,i} - \sum\nolimits_{k=0}^{K-1} (\sum\nolimits_{i=1}^{I} (P_{2,i,k} + P_{3,i,k}) + P_{4,k})$

Value Iteration Algorithm for Solving the DDPM Model

Let $V_t(s_t)$ be the optimal value function defined in the backward induction dynamic program. As shown in Figure 4 and in ***Optimality Equation (I)***, the optimal value function is iteratively defined by the optimality equation in the backward value iteration algorithm. Every feasible action in the current case is evaluated for all states using the backward induction value iteration algorithm.



S_t: state information available
S_{t+1}
Time
t
$t+1$
Optimality Equation (I)
a_t: dispatching decisions
b_t: preventive maintenance decision
$V_t(s_t)=\min_{a_t,b_t}\{C(s_t)+E[V_{t+1}(s_{t+1}|s_t,a_t,b_t)]\}$

Figure 4 Time chart of the value iteration algorithm

Machine maintenance is preemptive and assumed to begin immediately after decision b_t is made. Hence, system state changes from $s_t = (q_t, x_t, y_t)$ to

$\bar{s}_t = (\bar{q}_t, \bar{x}_t, \bar{y}_t) = (q_t, x_t + b_t, y_t - b_t)$.

That is, s_t and $\bar{s}_t$ are the system states before and after the machine maintenance decision in period t. The expected one-step cost remains unchanged after the maintenance decision because preventive maintenance has no immediate effect on queue length. Thus,

$$\mathrm{C}(\bar{s}_t) = \mathrm{C}(s_t) = \frac{1}{\varphi} \times \sum_{i=1}^{I} (h_i q_i).$$

Subsequently, we define an indicator variable $I_{(a_k=i)}$ as

$$I_{(a_k=i)} = \begin{cases} 1, & \text{if } a_k = i \\ 0, & \text{otherwise} \end{cases}.$$

Let $\bar{s}_t$ be the real-time system state at time t after maintenance decision b_t. The optimal finite horizon policies can be defined as

Optimality Equation (I):

$$V_t(s_t)$$

$$= \min_{a_t, b_t} \left\{ \mathrm{C}(s_t) + \sum_{s_{t+1}} \Pr(s_{t+1} | s_t, a_t, b_t) \cdot V_{t+1}(s_{t+1}) \right\}$$

$$= \mathrm{C}(s_t) + \min_{a_t, b_t} \left\{ \sum_{s_{t+1}} \Pr(s_{t+1} | (q_t, x_t + b_t, y_t - b_t), a_t) \cdot V_{t+1}(s_{t+1}) \right\}$$

$$= \mathrm{C}(s_t) + \min_{b_t}\left\{\min_{a_t}\left\{\sum\nolimits_{s_{t+1}} \Pr(s_{t+1}|(q_t, x_t + b_t, y_t - b_t), a_t) \cdot V_{t+1}(s_{t+1})\right\}\right\}$$

$$= \mathrm{C}(s_t) + \min_{b_t}\left\{\min_{a_t}\left\{\sum\nolimits_{s_{t+1}} \Pr(s_{t+1}|\bar{s}_t, a_t) \cdot V_{t+1}(s_{t+1})\}\right\}\right\}$$

$$= \frac{1}{\varphi} \times \left(\sum\nolimits_i h_i q_i\right) + \min_{b_t}\left\{\begin{array}{c} \sum_{i=1}^{I} P_{1,i} \times V_{t+1}\big((q_1, \dots, q_i + 1, \dots, q_I), x_t + b_t, y_t - b_t\big) \\ + \sum_{k=0}^{K-1} W_k(\bar{s}_t) \\ + \sum_{k=1}^{K} P_{4,k} \times V_{t+1}\big(q_t, (x_1 + b_1, \dots, x_k + b_k - 1, \dots, x_K + b_K), (y_1 - b_1, \dots, y_{K-1} + 1)\big) \\ + P_5 \times V_{t+1}(\bar{s}_t) \end{array}\right\},$$

where $W_k(\bar{s}_t)$ represents all the terms that are dependent on dispatching and preventive maintenance decisions. $W_k(\bar{s}_t)$ is defined as

$W_k(\bar{s}_t) = \min_{a_k \in (1,2,\dots I)} \{\mathrm{W}_k(\bar{s}_t, a_k)\}$, where

$$\mathrm{W}_k(\bar{s}_t, a_k)$$

$$= \sum_i \left\{\begin{array}{c} I_{(a_k=i)} \times P_{2,i,k} \times V_{t+1}\big((q_1, \dots, q_i - 1, \dots, q_I), x_t + b_t, y_t - b_t\big) \\ + I_{(a_k=i)} \times P_{3,i,k} \times V_{t+1}\big(q_t, x_t + b_t, (y_0 - b_0, \dots, y_{k-1} - b_{k-1} + 1, y_k - b_k - 1, \dots, y_{K-1})\big) \\ + \big(1 - I_{(a_k=i)}\big) \times P_{2,i,k} \times V_{t+1}(q_t, x_t + b_t, y_t - b_t) \\ + \big(1 - I_{(a_k=i)}\big) \times P_{3,i,k} \times V_{t+1}(q_t, x_t + b_t, y_t - b_t) \end{array}\right\}$$

$$= \sum\nolimits_i \left\{\begin{array}{c} \frac{I_{(a_k=i)} \times y_k \times \mu_i^k}{\varphi} \times V_{t+1}\big((q_1, \dots, q_i - 1, \dots, q_I), x_t + b_t, y_t - b_t\big) \\ + \frac{I_{(a_k=i)} \times y_k \times \beta_i^k}{\varphi} \times V_{t+1}\big(q_t, x_t + b_t, (y_0 - b_0, \dots, y_{k-1} - b_{k-1} + 1, y_k - b_k - 1, \dots, y_{K-1})\big) \\ + \frac{(1 - I_{(a_k=i)}) \times y_k \times \mu_i^k}{\varphi} \times V_{t+1}(q_t, x_t + b_t, y_t - b_t) \\ + \frac{(1 - I_{(a_k=i)}) \times y_k \times \beta_i^k}{\varphi} \times V_{t+1}(q_t, x_t + b_t, y_t - b_t) \end{array}\right\}.$$

Notably, finding the solution for the optimality equation requires iterative calculation. We need to find the optimal a_t and $\mathrm{W}_k(\bar{s}_t, a_k)$ for every preventive maintenance decision b_t, which requires three layers of for-loops on a_t and b_t for every s_t.

Table 3 Pseudocode for the original value iteration algorithm

```
For all $t$
  {
    For all $s_t$
      {
        For all $b_t$
            {
            For all $a_t$
                {
                Find $W_k(\bar{s}_t, a_t)$
                $W_k(\bar{s}_t) = \min_{a_k \in (1,2,\ldots I)} \{W_k(\bar{s}_t, a_k)\}$ that optimizes the dispatching decisions
                }
            Given $W_k(\bar{s}_t)$,
            Find $\min_{a_t}\{C(s_t) + \sum_{s_{t+1}} \Pr(s_{t+1}|\bar{s}_t, a_t) \cdot V_{t+1}(s_{t+1})\}$ for all $b_t$
            }
        $V_t(s_t) = \min_{b_t} \left\{ \min_{a_t}\{C(s_t) + \sum_{s_{t+1}} \Pr(s_{t+1}|\bar{s}_t, a_t) \cdot V_{t+1}(s_{t+1})\} \right\}$
      }
  }
```

The $c\mu$-rule (Cμ) is widely recognized as the optimal dispatching rule for queuing systems with multiple customer classes when the system does not exhibit dispatching-dependent deterioration (van Mieghem, 1995; Baras et al., 1985; Iravani and Kolfal, 2005). The Cμ dispatching policy assigns high priorities to products with large $c_i\mu_i$ values, where c_i is the waiting cost rate of product i and μ_i is the production rate. In **Proposition 1**, given ***Optimality Equation (I)***, we can demonstrate that Cμ can be sub-optimal in systems with dispatching-dependent deterioration.

Proposition 1: For queuing systems with machines that exhibit dispatching-dependent deterioration, Cμ that assigns highest priority to products with large $c_i\mu_i$ values can be sub-optimal.

Proof: We present the proposition using a counter example. For a queuing system with one machine, two products, and two machine health states (i.e., good or failed), the system parameters are set as follows:

Parameters	**Holding cost rate**	**Demand arrival rate**	**Production rate (good health)**	**Deterioration rate**	**Repair rate**
Values	$\begin{cases} h_1 = 1 \\ h_2 = 1 \end{cases}$	$\begin{cases} \lambda_1 = 0 \\ \lambda_2 = 0 \end{cases}$	$\begin{cases} \mu_1^1 = 3 \\ \mu_2^1 = 2 \end{cases}$	$\begin{cases} \beta_1^1 = 2 \\ \beta_2^1 = 0 \end{cases}$	$\gamma_0 = 1$

In this single-machine system, the state variable can be defined as $(q_1, q_2), (x_0, x_1), (y_0, y_1)$. Considering that the best health machine requires no maintenance and any failed machine immediately begins the repair process, x_1 and y_0 are always 0, and we can simplify the state definition to consider only non-zero variables q_1, q_2, x_0, and y_1. Hence, given a state (q_1, q_2, x_0, y_1), let $\widehat{\mathrm{V}}(q_1, q_2, x_0, y_1)$ be the optimal total expected costs before clearing all the jobs in the queues.

We can easily find the optimal expected costs before clearing all the queues for any given system state because the demand arrival rates are 0. When only one job is present in the queues, finding $\widehat{\mathrm{V}}(q_1, q_2, x_0, y_1)$ is easy because optimal dispatching will not make the server idle. Thus, by finding the expected costs of the non-idling policy, we can obtain

$$\begin{cases}\widehat{\mathrm{V}}(1,0,0,1) = 1 \\ \widehat{\mathrm{V}}(1,0,1,0) = 2\end{cases} \text{and} \begin{cases}\widehat{\mathrm{V}}(0,1,0,1) = \frac{1}{2} \\ \widehat{\mathrm{V}}(0,1,1,0) = \frac{3}{2}\end{cases}.$$

Then, consider states $(1,1,0,1)$ and $(1,1,1,0)$ that have exactly one job in both queues. Under the $\mathrm{C\mu}$ and non-$\mathrm{C\mu}$ policies, the expected total costs before clearing the queues can be determined by using the previously mentioned job $\widehat{\mathrm{V}}(q_1, q_2, x_0, y_1)$ and ***Optimality Equation (I)***.

- Expected total costs under the **$\mathrm{C\mu}$ policy**: $\begin{cases}V_{\mathrm{C\mu}}(1,1,0,1) = 5/2 \\ V_{\mathrm{C\mu}}(1,1,1,0) = 9/2\end{cases}$
- Expected total costs under the **non-$\mathrm{C\mu}$ policy:** $\begin{cases}V_{non-\mathrm{C\mu}}(1,1,0,1) = 2 \\ V_{non-\mathrm{C\mu}}(1,1,1,0) = 4\end{cases}$

Apparently, the expected costs of the following $\mathrm{C\mu}$ policy is higher than those of the non-$\mathrm{C\mu}$ policy. Thus, we conclude that the $\mathrm{C\mu}$ policy can be sub-optimal under the dispatching-dependent deterioration setting.

▩

The $\mathrm{C\mu}$ policy can be demonstrated to be optimal without dispatching-dependent deterioration and health-dependent production rate (Nain, 1989). However, when either dispatching-dependent deterioration or health-dependent production rate is present, the $\mathrm{C\mu}$ policy becomes sub-optimal, unstable, and will fail to achieve throughput optimality under a single-machine setting (Huang et al., 2018). Given that an unstable policy will lead to an unbounded cycle time and queue length, applying a simple heuristic policy, such as the $\mathrm{C\mu}$ policy, can be sub-optimal. Consequently, the optimal policy should be found by using the DPPM model.

Nested Induction Algorithm for Solving the DDPM Model

In accordance with **Proposition 1**, the Cμ dispatching rule is sub-optimal for systems with dispatching-dependent deterioration. Evaluating all possible combinations of states and action variables in ***Optimality Equation (I)*** is required in each iteration of the backward induction algorithm because the simple and widely used Cμ rule may be nonoptimal. However, the backward induction algorithm is known to be inefficient when the number of states or feasible actions is large.

Thus, we present a nested induction algorithm to simplify the computation process and facilitate the implementation of the algorithm. $v_t(s_t)$ is defined as the value function under the optimal one-step dispatching decision while the current period's preventive maintenance is ignored. $v_t(s_t)$ and $V_t(s_t)$ assume that the decision maker will proceed optimally beginning from the next decision epoch $t+1$ with regard to dispatching and maintenance decisions.

Therefore, we introduce the nested induction algorithm to $v_t(s_t)$ and $V_t(s_t)$. As shown in Figure 5, nested induction begins from the iterative definition of $v_t(s_t)$, ***Optimality Equation (II-A).*** Then, the optimal value function $V_t(s_t)$ can be iteratively determined by using $v_t(s_t)$ in ***Optimality Equation (II-B)***. Nested induction can considerably reduce computation time.

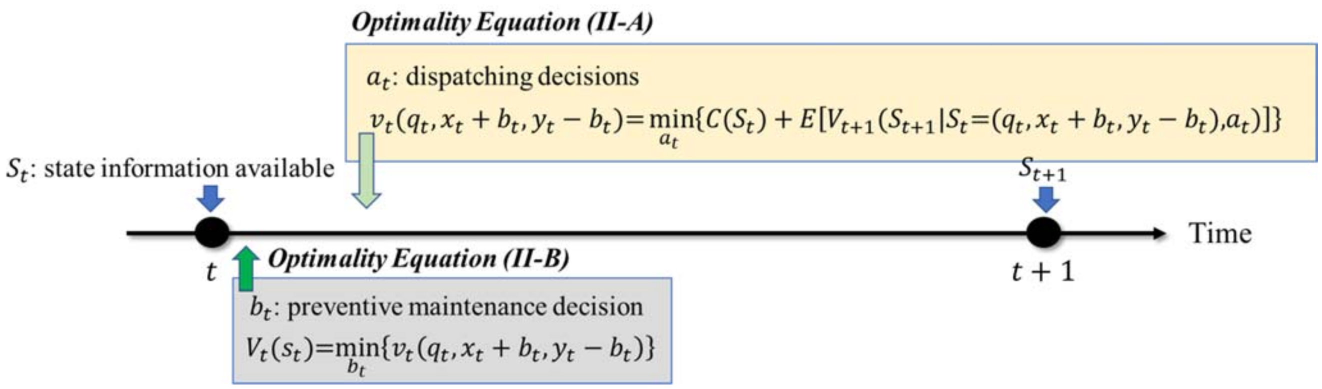

Optimality Equation (II-A)
a_t: dispatching decisions
$v_t(q_t, x_t+b_t, y_t-b_t)=\min_{a_t}\{C(S_t)+E[V_{t+1}(S_{t+1}|S_t=(q_t, x_t+b_t, y_t-b_t), a_t)]\}$
S_t: state information available
S_{t+1}
Time
t
$t+1$
Optimality Equation (II-B)
b_t: preventive maintenance decision
$V_t(s_t)=\min_{b_t}\{v_t(q_t, x_t+b_t, y_t-b_t)\}$

Figure 5 Time chart of the nested induction algorithm

Given that $v_t(s_t)$ considers only one-step optimal dispatching, we can write $v_t(s_t)$ as follows:

Optimality Equation (II-A):

$$v_t(s_t)$$
$$= \mathrm{C}(s_t) + \min_{a_t}\{\sum\nolimits_{s_{t+1}} \Pr(s_{t+1}|s_t, a_t)\cdot v_{t+1}(s_{t+1})\}$$

$$= \frac{1}{\varphi} \times \left(\sum_i h_i q_i\right) + \sum_{i=1}^{I} P_{1,i} \times v_{t+1}(q_{t+1} = (q_1, \dots, q_i + 1, \dots, q_I), x_{t+1} = x_t, y_{t+1} = y_t)$$
$$+ \sum_{k=0}^{K-1} W_k(s_t)$$
$$+ \sum_{k=0}^{K-1} P_{4,k} \times V_{t+1}\big(q_{t+1} = q_t, x_{t+1} = (x_0, \dots, x_k - 1, \dots, x_{K-1}), y_{t+1} = (y_0, \dots, y_{K-1} + 1)\big)$$
$$+ P_5 \times v_{t+1}(q_{t+1} = q_t, x_{t+1} = x_t, y_{t+1} = y_t),$$

where $\mathrm{W}_k(s_t)$ represents the dispatching-dependent terms for machines with health state k. Once more, $\mathrm{W}_k(s_t)$ can be defined as

$\mathrm{W}_k(s_t) = \min_{a_k \in (1,2,\dots I)} \{\mathrm{W}_k(s_t, a_k)\}$, where $\mathrm{W}_k(s_t, a_k)$ is defined as

$$\mathrm{W}_k(s_t, a_k)$$
$$= \sum_i \left\{ \begin{array}{c} I_{(a_k=i)} \times P_{2,i,k} \times V_{t+1}\big((q_1, \dots, q_i - 1, \dots, q_I), x_t, y_t\big) \\ + I_{(a_k=i)} \times P_{3,i,k} \times V_{t+1}\big(q_t, x_t, (y_0, \dots, y_{k-1} + 1, y_k - 1, \dots, y_{K-1})\big) \\ + (1 - I_{(a_k=i)}) \times P_{2,i,k} \times V_{t+1}(q_t, x_t, y_t) \\ + (1 - I_{(a_k=i)}) \times P_{3,i,k} \times V_{t+1}(q_t, x_t, y_t) \end{array} \right\}$$
$$= \sum_i \left\{ \begin{array}{c} \frac{I_{(a_k=i)} \times y_k \times \mu_i^k}{\varphi} \times V_{t+1}\big((q_1, \dots, q_i - 1, \dots, q_I), x_t, y_t\big) \\ + \frac{I_{(a_k=i)} \times y_k \times \beta_i^k}{\varphi} \times V_{t+1}\big(q_t, x_t, (y_0, \dots, y_{k-1} + 1, y_k - 1, \dots, y_{K-1})\big) \\ + \frac{(1 - I_{(a_k=i)}) \times y_k \times \mu_i^k}{\varphi} \times V_{t+1}(q_t, x_t, y_t) \\ + \frac{(1 - I_{(a_k=i)}) \times y_k \times \beta_i^k}{\varphi} \times V_{t+1}(q_t, x_t, y_t) \end{array} \right\}.$$

After finding $v_t(s_t)$ iteratively for all $s_t \in \mathrm{S}$, the optimality equation, including preventive maintenance decisions, can now be defined using $v_t(s_t)$.

Optimality Equation (II-B):

$$V_t(s_t) = \min_{b_t} \{v_t(q_t, x_t + b_t, y_t - b_t)\}$$

Table 4 Pseudocode for the nested induction algorithm

```
For all $t$
  {
    For all $s_t$
      {
```

```
      For all $a_t$
            {
                  Find $W_k(s_t, a_t)$
                  $W_k(s_t) = \min_{a_k \in (1,2,\ldots I)} \{W_k(s_t, a_k)\}$ that optimizes the dispatching decisions
            }
      Given $W_k(s_t)$, $v_t(s_t) = \min_{a_t} \{C(s_t) + \sum_{s_{t+1}} \Pr(s_{t+1}|s_t, a_t) \cdot V_{t+1}(s_{t+1})\}$
   }

   For all $s_t$
   {
      For all $b_t$
            {
            Find $\min_{b_t}\{v_t(q_t, x_t + b_t, y_t - b_t)\}$
            }
      $V_t(s_t) = \min_{b_t}\{v_t(q_t, x_t + b_t, y_t - b_t)\}$
   }
}
```

Proposition 2: The nested induction algorithm converges to the optimal value function $V_t(s_t)$ of the original backward induction algorithm.

Proof: System state $s_t = (q_t, x_t, y_t)$ immediately transits to $\bar{s}_t = (\bar{q}_t, \bar{x}_t, \bar{y}_t) = (q_t, x_t + b_t, y_t - b_t)$ when a preventive maintenance decision b_t is made because preventive maintenance is instantly initiated. Thus, $v_t(\bar{s}_t) = v_t(q_t, x_t + b_t, y_t - b_t)$ defined in ***Optimality Equation (II-A)*** represents the sum of the current and optimal future expected costs after preventive maintenance decision b_t**.** Therefore, finding $V_t(s_t)$ via $V_t(s_t) = \min_{b_t}\{v_t(q_t, x_t + b_t, y_t - b_t)\}$ in ***Optimality Equation (II-B)*** is equivalent to finding $V_t(s_t)$ in ***Optimality Equation (I).***

▓

Subsequently, we show that the computational complexity of the nested induction algorithm is less than that of the original value iteration algorithm. $\|S\|$ is defined as the number of states in the state space, $\|A\|$ as the average number of distinct dispatching decisions per state, and $\|B\|$ as the average number of distinct preventive maintenance decisions per state (i.e., $\|A\| \times \|B\|$ represents the average size of the action space per state).

Let $\boldsymbol{B}^{VIA}$ be the time complexity of solving the original backward induction algorithm and $\boldsymbol{B}^{NIA}$ be the complexity of the nested induction algorithm. We then present **Proposition 3** to characterize the computational complexity of both algorithms.

Proposition 3: The computational complexity of $\boldsymbol{B}^{NIA}$ of the nested induction algorithm is less than that of $\boldsymbol{B}^{VIA}$ of the original value iteration algorithm, and the difference in computational complexity can be estimated by

$$\frac{\boldsymbol{B}^{NIA}}{\boldsymbol{B}^{VIA}} \leq \frac{\|S\| \times (\|A\| + \|B\|)}{\|S\| \times \|B\| \times \|A\|}.$$

Proof:

The comparison of the optimality equations and pseudocodes of both algorithms indicates that the complexity of finding $W_k(\bar{s}_t, a_k)$ and $W_k(s_t, a_k)$ is the same for both algorithms. Let τ be the required computational effort for finding $W_k(s_t, a_k)$ or $W_k(\bar{s}_t, a_k)$, and the following observations are made.

- Through the pseudocode of the original value iteration algorithm, $W_k(\bar{s}_t, a_k)$ is evaluated within the three-layered for-loops on s_t, a_t, and b_t with a complexity of $\|S\| \times \|B\| \times \|A\| \times \tau$. Meanwhile, $W_k(s_t, a_k)$ is evaluated within the two-layered for-loops on s_t and b_t by the nested induction algorithm with a complexity of $\|S\| \times \|A\| \times \tau$.
- In the pseudocode of the nested induction algorithm, another two-layered for-loops on s_t and b_t are used to determine $V_t(s_t)$ by comparing $v_t(q_t, x_t + b_t, y_t - b_t)$ under all feasible b_t. This two-layered for-loops is not required in the original value iteration algorithm. The computational complexity of the additional loops is less than $\|S\| \times \|B\| \times \tau$ in the nested induction algorithm because the required computational effort for evaluating $v_t(q_t, x_t + b_t, y_t - b_t)$ in the additional loops is considerably less than that for $W_k(s_t, a_k)$.

We can conclude that the estimated time complexity $\boldsymbol{B}^{VIA}$ of the original algorithm is $\|S\| \times \|B\| \times \|A\| \times \tau$, and the estimated time complexity $\boldsymbol{B}^{NIA}$ of the nested induction algorithm is less than the sum of $\|S\| \times \|A\| \times \tau$ and $\|S\| \times \|B\| \times \tau$. Thus,

$$\frac{\boldsymbol{B}^{NIA}}{\boldsymbol{B}^{VIA}} \le \frac{\|S\| \times \|A\| \times \tau + \|S\| \times \|B\| \times \tau}{\|S\| \times \|B\| \times \|A\| \times \tau} = \frac{\|S\| \times (\|A\| + \|B\|)}{|S| \times |B| \times \|A\|}.$$

▩

The number of available dispatching decisions $\|A\|$ and $\|B\|$ is generally large. For example, $\|A\|$ can be estimated by $\|A\| = I^K$, and $\|B\|$ depends on the number of distinct combinations of available machines under all health states. The nested induction algorithm is considerably more efficient than the original value iteration algorithm because $\|A\|$ and $\|B\|$ are large in a reasonably sized system.

Implementation of the Nested Induction Algorithm for the DDPM Model

A software tool is developed using Microsoft Visual C# to solve the DDPM model. The software generates optimal dispatching and preventive maintenance policies. In the following numerical example, we illustrate the solution to the DDPM model for a problem with two machines, two products, and three machine health states (I = 2, J = 2, and K = 3). The parameters listed in Table 5 are used to construct the illustrative example.

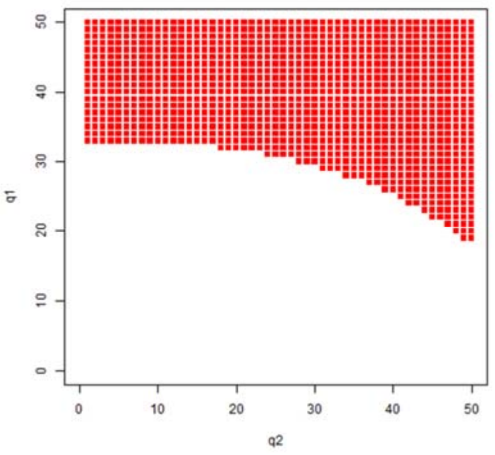
Table 5 Parameters of the example with two machines, two products, and three machine health states

Parameter	**Numerical example**
Holding cost of product i: h_i	$\begin{cases} h_1 = 1 \\ h_2 = 1 \end{cases}$
Arrival rate of product i: λ_i	$\begin{cases} \lambda_1 = 3 \\ \lambda_2 = 3 \end{cases}$
Production rate of product i on a perfectly conditioned machine (i.e., k=2): μ_i	$\begin{cases} \mu_1^2 = 5 \\ \mu_2^2 = 4 \end{cases}$
Production rate of product i on a fairly conditioned machine (i.e., k=1): $\mu_i^1 = r_i \mu_i$	$\begin{cases} \mu_1^1 = 3.5 \\ \mu_2^1 = 3.6 \end{cases}$
Deterioration rate of the machine while processing product i	$\begin{cases} \beta_1^2 = 0.03 \\ \beta_2^2 = 0.04 \end{cases}$
Failure rate of the machine while processing product i	$\begin{cases} \beta_1^1 = 0.02 \\ \beta_2^1 = 0.03 \end{cases}$
Maintenance rate	$\gamma_1 = 0.5$
Repair rate	$\gamma_0 = 0.4$

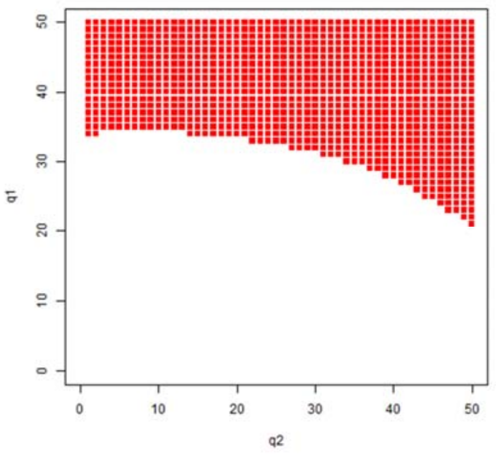
Under the proposed DPPM method, dispatching decisions are made in real time for each individual machine as soon as the machine becomes idle. Therefore, the DPPM model only handles one dispatching problem for a specific machine at a time. Given that a machine processes only one product at a time, the idle machine will immediately be assigned one job. Thus, if the highest priority queue is non-empty, only the highest priority product can be assigned to the idle machine. Meanwhile, in accordance with optimality equation (I) or (II-A), the highest priority will not be assigned to a product with empty queue to avoid machine idling, which is costly. As a result, the optimal dispatching rule generated by the DPPM model always dynamically assigns the highest priority to a product with non-empty queue depending on the real-time queue lengths. For example, in this illustrative dispatching problem of two product types, a non-idling policy that allocates product 1 to all machines whenever the queue for product 2 is empty is optimal. On the contrary, when the queue for product 1 is empty, all idle machines are assigned to process product 2.

In this two-machine example, the system state can be defined using eight variables, i.e., $s_t = (q_1, q_2, x_0, x_1, x_2, y_0, y_1, y_2)$, where $x_k, y_k \in \{0,1,2\}$ and $\sum_k (x_k + y_k) = 2$. Given that corrective maintenance immediately begins on failed machines and machines with the best health require no maintenance, y_0 and x_2 are constantly 0. We omit the two variables in the remainder of this paper to simplify the notation. We fix four variables ($x_0, x_1, y_1,$ and y_2) and plot the optimal control policy in the 2-dimension (2D) subsets (q_1, q_2) of the state space because we cannot easily visualize the optimal policy in the 8-dimension state space (six dimensions after omitting y_0 and x_2).

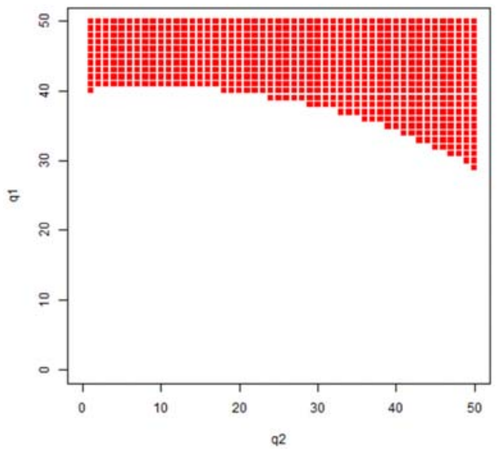
Figure 6 plots the optimal preventive maintenance decisions on fairly healthy machines (i.e., $k = 1$) when at least one fairly conditioned machine is available (i.e., $y_1 > 0$). Factory managers frequently delay preventive maintenance policy b_1 when systems are heavily loaded to prevent capacity loss during the process. In this numerical example, however, the optimal policy tends to begin when queues are long. The preventive maintenance decision depends on the health of not only an individual machine but also of other machines because such activity differs in subpanels (a), (b), (c), and (d) of Figure 6. Figure 6 also illustrates the need for the joint optimization of dispatching and preventive maintenance processes because the common practice of delaying such activity under a heavy load generally fails to achieve optimality.



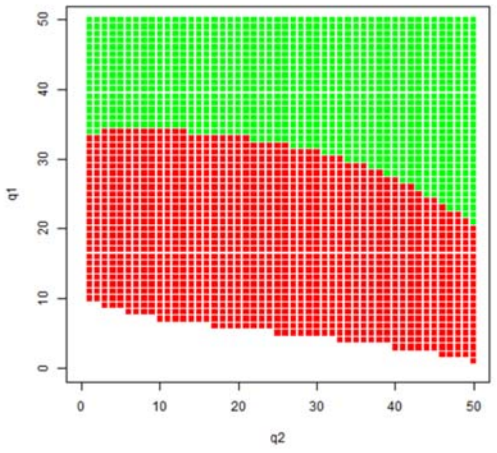
(a) $x_0 = 1, x_1 = 0,\ y_1 = 1, y_2 = 0$



(b) $x_0 = 0, x_1 = 1, y_1 = 1, y_2 = 0$



(c) $x_0 = 0, x_1 = 0, y_1 = 1, y_2 = 1$



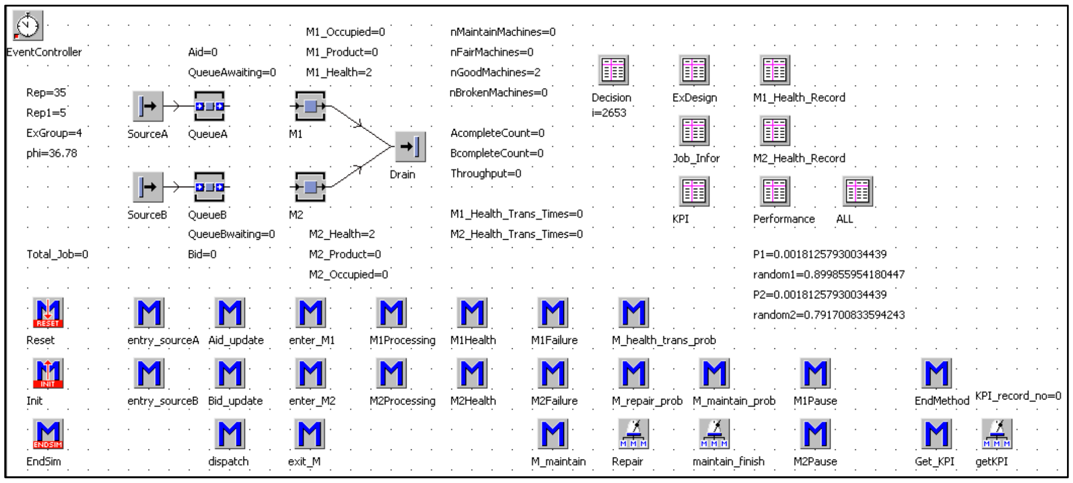
(d) $x_0 = 0, x_1 = 0, y_1 = 2, y_2 = 0$

⬜: will not maintain any fairly conditioned machine ($b_1 = 0$)
🟥: will maintain one fairly conditioned machine ($b_1 = 1$)
🟩: will maintain two fairly conditioned machines ($b_1 = 2$)

Figure 6 Optimal preventive maintenance decisions on machines with fair health

Model Validation and Numerical Analysis

In this research, the discrete event simulation software eM-Plant is used to build a simulation model to compare DDPM with several dispatching rules from the literature. We use systems with two machines, two products, and three machine health states ($I = 2, J = 2$, and $K = 3$) for model validation. The layout of the simulation system is shown in Figure 7. For all the simulation studies, the simulation periods are 365 days with an additional 10-day warm-up period.



Rep=35
Rep1=5
ExGroup=4
phi=36.78
Total_Job=0
Aid=0
QueueAwaiting=0
QueueBwaiting=0
Bid=0
M1_Occupied=0
M1_Product=0
M1_Health=2
M2_Health=2
M2_Product=0
M2_Occupied=0
nMaintainMachines=0
nFairMachines=0
nGoodMachines=2
nBrokenMachines=0
AcompleteCount=0
BcompleteCount=0
Throughput=0
M1_Health_Trans_Times=0
M2_Health_Trans_Times=0
i=2653
P1=0.00181257930034439
random1=0.899855954180447
P2=0.00181257930034439
random2=0.791700833594243
KPI_record_no=0

Figure 7 Simulation model for the numerical study

Dispatching Rules from the Literature

DDPM is compared with several widely used dispatching rules, namely, dynamic dispatching without preventive maintenance (DD), dynamic dispatching with condition-based preventive maintenance (DDCPM), first-come, first-served (FCFS), round-robin (RR), and $C\mu$.

1. DD

DD rules adopt the same logic as that in DDPM but do not consider preventive maintenance. That is, let $b_k = 0$ at all times. Notably, DD is the optimal dispatching policy when preventive maintenance decisions are not considered. The performance difference between DD and DDPM enables us to analyze the impact and value of coordinating preventive maintenance and dispatching decisions.

2. DDCPM

DDCPM adopts the condition-based preventive maintenance (CPM) policy that is widely used in practice. Under CPM, preventive maintenance begins on deteriorated machines as soon as deterioration is observed instead of using workload-dependent optimal preventive maintenance policies. A dynamic dispatching policy is then solved under the CPM rule. In summary, DDCPM differs from DDPM only in terms of maintenance policy and any performance improvement observed under DDPM results from a better preventive maintenance policy.

3. FCFS

FCFS is a common, simple, and easy-to-implement dispatching rule. Bernier et al. (2004) and Wein (1988) indicated that FCFS is used in semiconductor production systems. Under FCFS, products are manufactured in the order that they enter the system.

4. RR

RR policies are known to achieve throughput optimality in production systems and computer networks with unlimited buffer capacity (Andradottir et al., 2002). In our simulation study, we allow machines to process different products in circular order in pre-assigned time slices that aim to maximize overall throughput under a given product mix.

5. Cμ

Cμ is known to achieve minimal weighted cycle time in systems without dispatching-dependent deterioration (van Mieghem, 1995). Under the Cμ policy, higher priority will be given to a product with the largest $c_i\mu_i -$ value, where c_i is the waiting cost of product i and μ_i is the production rate for producing product i. Baras et al. (1985) and Iravani and Kolfal (2005) proposed that Cμ is suitable for controlling multiproduct manufacturing systems.

Data Collection and Key Performance Indicators (KPIs) for the Numerical Analysis

In the simulation study, the following performance indicators are collected and analyzed:

1. Throughput:

The throughput indicator simply counts the total output during the simulation period.

2. Average processing time:

The average processing time of jobs will possibly depend on dispatching rules because production slows down with machine deterioration. Therefore, we collect the average processing time of jobs in our simulation study.

3. Average waiting time:

The average waiting time of jobs in queues will also depend on dispatching and maintenance policies. Notably, the sum of the average processing time and the average waiting time will become the average cycle time. Given that the average processing and waiting times are provided in the numerical results, the average cycle time will not be separately provided to avoid redundancy.

4. Average machine downtime:

Preventive maintenance and corrective maintenance frequently require different amounts of time to complete. Although preventive maintenance helps reduce unexpected machine failures, conducting frequent preventive maintenance may increase failure/maintenance counts. Thus, overall machine downtime, which is the sum of the corrective maintenance and preventive maintenance times, may or may not decrease with optimal preventive maintenance decisions. We collect the average machine downtime to observe changes in machine availability.

Design of Experiments

The objective of our numerical study is to validate the performance of the DDPM model within a wide range of production environments. In the numerical study, the system parameters are selected on the basis of an actual industry problem in semiconductor manufacturing. To conduct sensitivity analysis and validate the proposed method in general production environments, parameters are scaled up and down in the factorial experiments.

To maximize the diversity of our numerical cases, experimental design techniques are used to generate test systems. Instead of randomly generating system parameters, we use the two-level Plackett–Burman experimental design technique to generate 20 sets of system parameters. The high–low levels of the model parameters are listed in Table 6. The complete Plackett–Burman designs of the 20 sets of parameters are summarized in Table 7.

Table 6 High–low levels of the experimental factors

Experimental Factors	**{Low, High}**
$\beta_i^1, \ i \in \{1,2\}$	$\{\frac{1}{10}, \frac{1}{15}\}$
$\beta_i^2, \ i \in \{1,2\}$	$\{\frac{1}{8}, \frac{1}{12}\}$
γ_0	$\{\frac{1}{2}, \frac{1}{3}\}$
$\mu_i^2, \ i \in \{1,2\}$	$\{4, 6\}$
$r_i, \ i \in \{1,2\}$ $(\mu_i^1 = r_i\mu_i^2)$	$\{0.5, 0.8\}$

ρ, system utilization	$\{0.75, 0.9\}$
α_i, $i \in \{1,2\}$, product mix	$\{2, 3\}$
τ ($\gamma_1 = \tau\gamma_0$)	$\{2, 3\}$

Table 7 Twenty sets of system parameters using the Plackett–Burman design

	$\boldsymbol{\beta_1^2}$	$\boldsymbol{\beta_2^2}$	$\boldsymbol{\beta_1^1}$	$\boldsymbol{\beta_2^1}$	$\boldsymbol{\gamma_0}$	$\boldsymbol{\mu_1^2}$	$\boldsymbol{\mu_2^2}$	$\boldsymbol{r_1}$	$\boldsymbol{r_2}$	$\underline{\boldsymbol{\rho}}$	$\boldsymbol{\alpha_1}$	$\boldsymbol{\alpha_2}$	$\boldsymbol{\tau}$
1	1/15	1/15	1/8	1/8	1/2	4	6	0.5	0.8	0.75	3	3	3
2	1/15	1/15	1/12	1/8	1/2	6	6	0.5	0.8	0.9	2	2	2
3	1/10	1/10	1/12	1/12	1/2	6	6	0.5	0.5	0.75	2	3	2
4	1/10	1/15	1/12	1/8	1/2	4	4	0.8	0.5	0.9	2	3	3
5	1/10	1/15	1/8	1/12	1/3	6	6	0.5	0.5	0.9	3	2	3
6	1/15	1/10	1/12	1/8	1/3	6	6	0.8	0.5	0.75	3	3	2
7	1/10	1/10	1/8	1/8	1/3	4	6	0.5	0.8	0.9	3	3	2
8	1/10	1/15	1/8	1/12	1/2	6	6	0.8	0.8	0.75	2	3	3
9	1/15	1/15	1/12	1/12	1/2	4	6	0.8	0.5	0.9	3	2	2
10	1/10	1/15	1/12	1/8	1/3	6	4	0.5	0.5	0.75	3	2	3
11	1/15	1/10	1/12	1/12	1/2	4	4	0.5	0.8	0.75	3	2	3
12	1/15	1/15	1/8	1/8	1/3	6	4	0.8	0.8	0.75	2	2	2
13	1/15	1/15	1/8	1/12	1/3	4	4	0.5	0.5	0.9	2	3	2
14	1/10	1/10	1/12	1/8	1/3	4	6	0.8	0.8	0.9	2	2	3
15	1/10	1/10	1/8	1/12	1/2	6	4	0.8	0.8	0.9	3	2	2
16	1/15	1/10	1/8	1/8	1/2	6	4	0.8	0.5	0.9	3	3	3
17	1/15	1/10	1/8	1/12	1/3	4	6	0.8	0.5	0.75	2	2	3
18	1/10	1/10	1/8	1/8	1/2	4	4	0.5	0.5	0.75	2	2	2
19	1/15	1/10	1/12	1/12	1/3	6	4	0.5	0.8	0.9	2	3	3
20	1/10	1/15	1/12	1/12	1/3	4	4	0.8	0.8	0.75	3	3	2

Numerical Results and Analysis: Exponential Processing Time Distribution

We run the simulation for all the 20 systems under each control policy. Tables 8 and 9 summarize the simulation results under exponential processing times (the detailed simulation results of each of the 20 systems can be found in the Appendix). Notably, DDPM outperforms all the other methods for all the KPIs.

In Table 8, all the results are normalized to the result of DDPM to facilitate reading and comparison. For example, the throughput of DDCPM is 0.9982 times the throughput of DDPM, and the average waiting time of DDCPM is 1.4523 times the average waiting time of DDPM. In addition to average performance, the confidence intervals for all the 20 systems under each KPI are listed in Tables A1–A4.

In Table 9, the pairwise *t*-test results are summarized to verify the significance of performance improvement between DDPM and all the other policies. For example, compared with that under DD, the performance improvement under DDPM is significant on all the 20 systems in terms of average processing time, average waiting time, and average machine downtime.

Table 8 Average normalized simulation results under exponential processing time distribution

	Processing Time Distribution: Exponential					
KPI	DDPM	DD	DDCPM	FCFS	RR	Cμ
Throughput	1	1.0011	0.9982	0.9973	0.9857	1.0002
Average processing time	1	1.1778	1.0036	1.1978	1.1949	1.1861
Average waiting time	1	5.2162	1.4523	7.5410	23.0186	5.7680
Average machine downtime	1	1.6290	1.4885	1.6519	1.6325	1.6449

Throughput: A higher value is better.

Processing time/waiting time/downtime: A lower value is better.

Table 9 Pairwise *t*-test results between DDPM and the other policies

	DDPM vs. DD			**DDPM vs. DDCPM**			**DDPM vs. FCFS**			**DDPM vs. RR**			**DDPM vs. Cμ**		
	B	**I**	**W**	**B**	**I**	**W**	**B**	**I**	**W**	**B**	**I**	**W**	**B**	**I**	**W**
Throughput	0	20	0	2	18	0	1	19	0	7	13	0	0	20	0
Average processing time	20	0	0	0	15	5	20	0	0	20	0	0	20	0	0
Average waiting time	20	0	0	11	9	0	20	0	0	20	0	0	20	0	0
Average machine downtime	20	0	0	10	10	0	20	0	0	20	0	0	20	0	0

B: Number of systems, DDPM is significantly better.

I: Number of systems, difference is insignificant.

W: Number of systems, DDPM is significantly worse

The findings and observations from the simulation results are summarized as follows.

● Throughput performance of DDPM:

Redundant capacity exists in our numerical study because capacity utilization levels are set between 0.75 and 0.9. Thus, a manufacturing system can process all demand and achieve similar throughput performance. From the first row of Table 9, the throughput of DDPM still considerably outperforms those of RR, FCFS, and DDCPM in seven systems, one system, and two systems, respectively, whereas the difference in the other systems is insignificant.

Considering that the simulation model assumes limited buffer capacity, demand is lost when the queue is full and losing demand may occurs if production control is inefficient. Thus, DDPM can still achieve significant throughput improvement in several systems. Significant throughput improvements are observed among the systems, and the capacity utilization levels are all 0.9. This result implies that DDPM can benefit from high system utilization, and significant throughput improvement can be observed under high capacity utilization.

● Average processing time:

The second row of Table 9 indicates that DDPM significantly outperforms all the other policies in terms of average processing time in all the systems. As shown in Table 8, the average savings in processing time ranges from 17.8% to 19.8%, except when compared with DDCPM. Given that DDCPM always performs preventive maintenance when machine deterioration is observed, the machine is always operating under perfect condition, which makes the processing time similar to that of DDPM. However, DDCPM suffers from an aggressive maintenance policy and has a longer preventive maintenance time and longer job waiting time in queue.

● Average waiting time:

DDPM significantly outperforms DD, FCFS, RR, and $C\mu$ in terms of average waiting time in all 20 systems. As shown in Table 8, the average waiting times of DD, FCFS, RR, and $C\mu$ are 5.2, 7.5, 23.0, and 5.7 times that of DDPM, respectively. DDPM significantly outperforms DDCPM in terms of average waiting time in 11 systems, whereas the performance difference is insignificant in the nine other systems. A considerable reduction in waiting time and queue lengths are achieved by simply coordinating preventive maintenance decisions with dispatching decisions. Meanwhile, DD and DDCPM are the optimal dispatching policies without preventive maintenance and with condition-based preventive maintenance, respectively. This finding suggests that coordinating preventive maintenance decisions with dispatching decisions is important for production systems with dispatching-dependent deterioration.

In addition, queue length directly reflects the work-in-process (WIP) level of a production system. A reduction in average waiting time implies that coordinating scheduling and preventive maintenance can reduce average WIP levels by an average of approximately 80%. Although lean manufacturing attempts to eliminate WIP and inventory in production systems, optimal preventive maintenance is key to achieving leaner production.

● Average machine downtime:

Over 50% reduction in machine downtime is observed when optimal preventive maintenance is adopted. Overall machine downtime may not decrease with better preventive maintenance decisions because such activity frequently requires different periods to complete. Our simulation results suggest that conducting optimal preventive maintenance may not only reduce unexpected machine failures but also average machine downtime.

To evaluate the performance of DDPM in systems with more than two products, we randomly generate 20 systems with three and four products using the parameters in Table 6. For each of the 20 randomly generated systems, DDPM outperforms all other policies by at least 0.6% and 16.5% in throughput and cycle time, respectively. However, although the simulation results suggest the superiority of DPPM in systems with more products, the required computational effort for solving large-scale MDP problems becomes a challenge. To overcome those computational challenges, we plan to combine artificial intelligence algorithms with MDP to control the required computational effort in the future.

Model Validation in Non-Markovian Systems: Constant and Uniformly Distributed Processing Times

To respond to this well-known weakness of MDP models, i.e., assuming that the processing times of jobs follow an exponential distribution, we conduct a simulation study with uniformly distributed and constant processing times.

- Uniform processing time distribution: Processing time is assumed to follow uniform random variables $U(\frac{0.8}{\mu_i^k}, \frac{1.2}{\mu_i^k})$, where $\frac{1}{\mu_i^k}$ represents the average processing time of a job i on a health state k machine, and

$U(\frac{0.8}{\mu_i^k}, \frac{1.2}{\mu_i^k})$ suggests that processing times are uniformly distributed between 80% and 120% of the average processing time.

- Constant processing time: Constant processing time $\frac{1}{\mu_i^k}$ is assumed for job i on a health state k machine. Under the constant processing time setting, no processing time uncertainty exists in the simulation.

We conduct simulations using uniform and constant processing time distributions for all 20 systems, and the simulation results are summarized in Tables 10 and 11. Notably, DDPM performs equally well in systems with a non-Markovian processing time distribution.

Table 10 Simulation results with uniformly distributed processing times

	Processing Time Distribution: Uniform $U(\frac{0.8}{\mu_i^k}, \frac{1.2}{\mu_i^k})$					
KPI	DDPM	DD	DDCPM	FCFS	RR	Cμ
Throughput	1	1.0009	0.9884	0.9999	0.9855	1.00002
Average processing time	1	1.1843	1.0012	1.2039	1.1932	1.1901
Average waiting time	1	5.8329	1.5916	9.0932	30.8172	6.6703
Average machine downtime	1	1.6173	1.4922	1.6442	1.6214	1.6325

Throughput: A higher value is better.

Processing time/waiting time/downtime: A lower value is better.

Table 11 Simulation results with a constant processing time

	Processing Time Distribution: Constant					
KPI	DDPM	DD	DDCPM	FCFS	RR	Cμ
Throughput	1	0.9994	0.9992	0.9901	0.9851	0.9990
Average processing time	1	1.1834	1.0029	1.2001	1.1928	1.1923
Average waiting time	1	6.3279	1.6202	9.3582	34.3034	7.2445
Average machine downtime	1	1.6226	1.5211	1.6521	1.6249	1.6451

Throughput: A higher value is better.

Processing time/waiting time/downtime: A lower value is better.

Compared with the simulation results using exponentially distributed processing times, considerable improvements in average processing time/waiting time/machine downtime can still be observed, and throughput performance remains similar under all processing time settings. The simulation results suggest the robustness of the DDPM model in general systems that do not comply with the Markov assumption.

Conclusion

This research develops a dynamic decision model that coordinates dispatching and preventive maintenance decisions for failure-prone parallel machines. The proposed DDPM model simultaneously considers two important features of production systems: dispatching-dependent deterioration of machines and machine-health-dependent production rates. Although these features are common in production systems, our research is among the first to consider both features in dispatching and preventive maintenance decision problems.

The proposed DDPM model is validated through a thorough numerical study. Compared with those of other widely used dispatching methods in the literature, the average waiting time and machine downtime can be reduced by at least 45.2% and 48.9%, respectively. Moreover, the proposed method is equally robust in non-Markovian systems with constant and uniformly distributed processing times. Our numerical results suggest that coordination between dispatching and preventive maintenance is critical. Compared with the optimal dispatching policy without controlled preventive maintenance, waiting and production times are significantly reduced.

Given that this study is limited to a single workstation, we plan to extend our research to multiple-stage serial production systems in the future. In addition, although the proposed DDPM model is a general model that allows any number of products, machines, and machine health states, the required computational effort for solving large-scale problems remains a challenge. Considering the recent developments in artificial intelligence (AI), particularly in reinforcement learning, we plan to develop efficient AI-based algorithms for solving large DDPM models in the future.

ACKNOWLEDGMENT

We thank the editor and the anonymous reviewers for their insightful comments and suggestions. This research is supported in part by the Ministry of Science and Technology of Taiwan under Grant No. 105-2923-E-002-009-MY3/ 107-2628-E-002-006-MY3 and the *Agence Nationale de la Recherche* of France under Grant No. ANR-15-CE10-0003.

REFERENCES

[1] Afzalirad, M., & Rezaeian, J. (2015). *Design of high-performing hybrid meta-heuristics for unrelated parallel machine scheduling with machine eligibility and precedence constraints.* Engineering Optimization, 1-21.

[2] Andradottir, S., Ayhan, H. and Down D.G. (2001). *Server assignment policies for maximizing the steady-state throughput of finite queueing systems*, Management Science, 47(10), 1421-1439.

[3] Baras, J. S., Dorsey, A. J., & Makowski, A. M. (1985). *Two competing queues with linear costs and geometric service requirements: the μc-rule is often optimal.* Advances in Applied Probability, 17(01), 186-209.

[4] Bernier, V., & Frein, Y. (2004). *Local scheduling problems submitted to global FIFO processing constraints.* International journal of production research, 42(8), 1483-1503.

[5] Borrero, J. S., & Akhavan-Tabatabaei, R. (2013). *Time and inventory dependent optimal maintenance policies for single machine workstations: An MDP approach.* European Journal of Operational Research, 228(3), 545-555.

[6] Cai, Y., Hasenbein, J. J., Kutanoglu, E., & Liao, M. (2013). *Single-machine multiple recipe predictive maintenance*. Probability in the Engineering and Informational Sciences, 27(2), 209-235.

[7] Celen, M., & Djurdjanovic, D. (2015). *Integrated maintenance decision-making and product sequencing in flexible manufacturing systems.* Journal of Manufacturing Science and Engineering, 137(4):041006–041006–15.

[8] Cui, W.W., Lu, Z., & Pan, E. (2014). *Integrated production scheduling and maintenance policy for robustness in a single machine*. Computers & Operations Research, 47, 81-91.

[9] Chen, A., & Wu, G. S. (2007). *Real-time health prognosis and dynamic preventive maintenance policy for equipment under aging Markovian deterioration.* International Journal of Production Research, 45(15), 3351-3379.

[10] Chen, J. F. (2015). *Unrelated parallel-machine scheduling to minimize total weighted completion time.* Journal of Intelligent Manufacturing, 1099-1112.

[11] Feinberg, E.A. & Shwartz, A. (2002). Handbook of Markov Decision Processes: Methods and Applications. Kluwer, Berlin.

[12] Huang, J., Down, D.G., Lewis, M. E., & Wu, C.-H. (2018) *Dynamic scheduling and maintaining a flexible server*, preprint.

[13] Iravani, S. M., & Kolfal, B. (2005). *When does the cμ rule apply to finite-population queueing systems?*. Operations Research Letters, 33(3), 301-304.

[14] Kaufman, D. L., & Lewis, M. E. (2007). *Machine maintenance with workload considerations.* Naval Research Logistics (NRL), 54(7), 750-766.

[15] Kazaz, B., & Sloan, T. W. (2013). *The impact of process deterioration on production and maintenance policies.* European Journal of Operational Research, 227(1), 88-100.

[16] Kao, Y. T., Dauzère-Pérès, S., Blue, J., & Chang, S. C. (2018). *Impact of integrating equipment health in production scheduling for semiconductor fabrication*. Computers & Industrial Engineering, 120, 450-459.

[17] Lee, C.-Y., & Meng, Q. (2014). *Handbook of Ocean Container Transport Logistics: Making Global Supply Chains Effective.* Springer, Berlin.

[18] Lee, S., & Ni, J. (2013). *Joint decision making for maintenance and production scheduling of production*

systems. The International Journal of Advanced Manufacturing Technology, 66(5-8), 1135-1146.

[19] Liaw, C-F., Lin, Y-K., Cheng, C-Y., & Chen, M. (2003). *Scheduling unrelated parallel machines to minimize total weighted tardiness.* Computers & Operations Research, vol.30, no.12, pp.1777-1789.

[20] Lin, S. W., & Ying, K. C. (2014). *ABC-based manufacturing scheduling for unrelated parallel machines with machine-dependent and job sequence-dependent setup times.* Computers & Operations Research, 51, 172-181.

[21] Nain., P. (1989) *Interchange arguments for classical scheduling problems in queues.* Systems & Control Letters, 12(2):177–184.

[22] Rezaeian, J. (2016). *A robust hybrid approach based on particle swarm optimization and genetic algorithm to minimize the total machine load on unrelated parallel machines.* Applied Soft Computing.

[23] Samhouri, M. S. (2009). *An intelligent opportunistic maintenance (OM) system: a genetic algorithm approach.* In Science and Technology for Humanity (TIC-STH), 2009 IEEE Toronto International Conference (pp. 60-65). IEEE.

[24] Serfozo, R. (1978) *An equivalence between continuous and discrete time Markov decision processes.* Operations Research, 27(3), 616–620.

[25] van Mieghem, J. A. (1995). *Dynamic Scheduling with Convex Delay Costs: The Generalized c|mu Rule*. The Annals of Applied Probability, 5(3), 809-833

[26] Wallace, S.W. & Ziemba, W.T. (2005) *Applications of Stochastic Programming*. SIAM Publication, Philadelphia.

[27] Wu, C. H., Down, D. G., Lewis, M. E. (2008). *Heuristics for allocation of reconfigurable resources in a serial line with reliability considerations.* IIE Transactions, 40(6), 595-611.

[28] Yao, X., Xie, X., Fu, M. C., & Marcus, S. I. (2005). *Optimal joint preventive maintenance and production policies.* Naval Research Logistics (NRL), 668-681.

[29] Yu, L., Shih, H. M., Pfund, M., Carlyle, W. M., & Fowler, J. W. (2002). *Scheduling of unrelated parallel machines: an application to PWB manufacturing.* IIE transactions, 34(11), 921-931.

[30] Zhang, X., & Zeng, J. (2015). *Deterioration state space partitioning method for opportunistic maintenance modelling of identical multi-unit systems.* International Journal of Production Research, 53(7), 2100-2118.

[31] Zhou, B., Yu, J., Shao, J., & Trentesaux, D. (2015). *Bottleneck-based opportunistic maintenance model for series production systems.* Journal of Quality in Maintenance Engineering.

[32] Zhou, X., Xi, L., & Lee, J. (2009). *Opportunistic preventive maintenance scheduling for a multi-unit series system based on dynamic programming.* International Journal of Production Economics, 118(2), 361-366.

[33] Z.-J Zhang, & J. Daigle.(2012) *Analysis of job assignment with batch arrivals among heterogeneous servers.* European Journal of Operational Research 217.1: 149-161.

Appendix A: Detailed Results of the Numerical Experiments

- Average Throughput

Table A1. Normalized average throughput and confidence interval of 30 simulation replications

Throughput							
System	DDPM	DD	DDCPM	FCFS	RR	Cμ	DDPM Rank
1	1	1.0005 ±0.0023	0.9996 ±0.018	1.0009 ±0.0021	1.002 ±0.0022	1.0013 ±0.002	5
2	1	0.9989 ±0.0017	0.9974 ±0.0017	0.9853 ±0.0268	0.9977 ±0.0018	1.0003 ±0.0018	2
3	1	1.0046 ±0.0074	1.0049 ±0.0078	1.0041 ±0.008	1.0038 ±0.0076	1.0042 ±0.0067	6
4	1	0.9999 ±0.0025	0.9808 ±0.0387	1.0005 ±0.0023	0.9364 ±0.0029	1 ±0.0023	3
5	1	0.9992 ±0.0015	1.0005 ±0.0018	0.998 ±0.0014	0.9786 ±0.003	0.9986 ±0.0024	2
6	1	1.0005 ±0.002	0.9989 ±0.002	0.9989 ±0.0026	0.9762 ±0.0471	0.9998 ±0.0018	2
7	1	1.0209 ±0.0441	1.0073 ±0.0516	1.0211 ±0.0448	1.0069 ±0.0531	1.0217 ±0.0446	6
8	1	1.0003 ±0.0019	1.0001 ±0.002	0.9992 ±0.0014	0.9998 ±0.0016	1.0009 ±0.0016	4
9	1	0.9991 ±0.0019	0.9987 ±0.0018	0.9799 ±0.0398	0.973 ±0.0021	0.9994 ±0.0023	1
10	1	0.9996 ±0.0019	0.999 ±0.0022	0.9994 ±0.0022	0.987 ±0.0272	1.0001 ±0.0022	2
11	1	1.0005 ±0.003	1.0026 ±0.0027	0.9882 ±0.027	0.9821 ±0.0385	0.9998 ±0.003	3
12	1	0.9982 ±0.0026	0.9979 ±0.002	0.998 ±0.0028	0.9994 ±0.0023	0.9997 ±0.0026	5
13	1	1.0005 ±0.0026	1.0001 ±0.0023	1.0003 ±0.0021	0.98 ±0.0031	0.9983 ±0.0022	4
14	1	0.9998 ±0.0018	0.9999 ±0.0024	0.978 ±0.0399	1.0001 ±0.0018	1.001 ±0.0024	3
15	1	1.0011 ±0.0015	0.9987 ±0.0021	0.9995 ±0.0022	0.9219 ±0.0024	0.9966 ±0.0067	2
16	1	0.9996 ±0.0025	1.0003 ±0.0026	0.9999 ±0.0024	0.9979 ±0.0025	1.0001 ±0.0022	3
17	1	1.0001 ±0.0017	0.9987 ±0.0023	0.9997 ±0.0024	0.9988 ±0.0022	0.9815 ±0.0371	2
18	1	1.001 ±0.0024	0.999 ±0.0023	0.9984 ±0.003	0.9992 ±0.0029	0.9994 ±0.0024	2
19	1	0.9986 ±0.0023	0.9816 ±0.0361	0.9992 ±0.0022	0.9872 ±0.0027	1.0016 ±0.0019	2
20	1	0.9996 ±0.0023	1.0006 ±0.0026	0.9818 ±0.0383	0.9599 ±0.0597	0.9996 ±0.002	2
Avg	1	1.0011	0.9982	0.9973	0.9857	1.0002	3

● Average processing time of jobs

Table A2. Normalized average processing time of jobs and confidence interval

Average processing time of jobs							
System	DDPM	DD	DDCPM	FCFS	RR	Cμ	DDPM Rank
1	1	1.1457 ±0.0038	0.9987 ±0.0021	1.1647 ±0.0035	1.1606 ±0.0043	1.1627 ±0.0038	2
2	1	1.1299 ±0.0031	1.0012 ±0.0015	1.1836 ±0.0033	1.1718 ±0.0032	1.1298 ±0.0034	1
3	1	1.3655 ±0.0065	0.9983 ±0.0021	1.3765 ±0.0045	1.3716 ±0.0055	1.3737 ±0.0057	2
4	1	1.1646 ±0.0034	0.9996 ±0.0025	1.2065 ±0.0042	1.195 ±0.003	1.1978 ±0.0044	2
5	1	1.2569 ±0.0045	0.9989 ±0.0022	1.2812 ±0.0056	1.284 ±0.0042	1.2764 ±0.005	2
6	1	1.1856 ±0.0035	1.0001 ±0.0020	1.1985 ±0.0042	1.1981 ±0.0043	1.1961 ±0.004	1
7	1	1.1795 ±0.0033	0.9995 ±0.0023	1.2269 ±0.004	1.2158 ±0.0048	1.2118 ±0.0047	2
8	1	1.0948 ±0.003	1.0147 ±0.0025	1.0995 ±0.0029	1.0977 ±0.0023	1.1001 ±0.0039	1
9	1	1.13 ±0.0024	0.9997 ±0.0022	1.1664 ±0.0034	1.1615 ±0.0032	1.1278 ±0.0029	2
10	1	1.2937 ±0.0049	0.9997 ±0.0026	1.2805 ±0.0061	1.2854 ±0.0059	1.2845 ±0.0056	2
11	1	1.2189 ±0.0047	1.0095 ±0.0028	1.2501 ±0.0054	1.2357 ±0.0055	1.2211 ±0.0051	1
12	1	1.0738 ±0.0033	0.9987 ±0.0025	1.0727 ±0.0021	1.0753 ±0.003	1.0763 ±0.0029	2
13	1	1.2254 ±0.0043	0.9998 ±0.0023	1.2499 ±0.0051	1.2488 ±0.0052	1.2467 ±0.0042	2
14	1	1.1112 ±0.0028	1.0213 ±0.0026	1.1101 ±0.003	1.1084 ±0.003	1.1089 ±0.0029	1
15	1	1.1128 ±0.003	1.0014 ±0.0022	1.1098 ±0.0034	1.1293 ±0.0036	1.1101 ±0.003	1
16	1	1.169 ±0.004	1.0008 ±0.0024	1.2029 ±0.0037	1.1977 ±0.0038	1.1981 ±0.0032	1
17	1	1.1577 ±0.005	1.0096 ±0.003	1.1855 ±0.0051	1.1721 ±0.0032	1.1554 ±0.0055	1
18	1	1.2818 ±0.0043	1 ±0.0027	1.2815 ±0.0055	1.2878 ±0.0051	1.2827 ±0.0048	1
19	1	1.1512 ±0.0032	1.0212 ±0.0067	1.1987 ±0.004	1.1916 ±0.0034	1.1505 ±0.0038	1
20	1	1.1086 ±0.0027	1.0002 ±0.0019	1.1108 ±0.003	1.1107 ±0.0034	1.1116 ±0.0024	1
Avg	1	1.1778	1.0036	1.1978	1.1949	1.1861	1

● Average waiting time of jobs in queues

Table A3. Normalized average waiting time of jobs in queues and confidence interval

Average waiting time of jobs							
System	DDPM	DD	DDCPM	FCFS	RR	Cμ	DDPM Rank
1	1	3.7505 ±0.1168	1.4190 ±0.0266	5.2056 ±0.1895	18.4688 ±0.3537	4.1909 ±0.1993	1
2	1	3.649 ±0.1771	0.9795 ±0.0304	7.7497 ±0.3819	14.4849 ±0.4596	3.787 ±0.197	2
3	1	7.4867 ±0.3685	1.0126 ±0.0268	7.7539 ±0.262	37.5004 ±1.2775	7.737 ±0.3292	1
4	1	5.2791 ±0.2302	1.7312 ±0.2094	12.1064 ±0.7163	30.7202 ±0.5379	9.1211 ±0.5062	1
5	1	13.5231 ±0.5595	2.2757 ±0.0807	18.4701 ±1.5422	42.8866 ±0.9762	15.0898 ±0.6094	1
6	1	3.3589 ±0.1472	0.9791 ±0.0428	4.3531 ±0.2292	13.9026 ±0.4455	3.6689 ±0.18	2
7	1	4.8017 ±0.2263	0.9812 ±0.033	10.0969 ±0.6601	16.2172 ±0.7444	6.2883 ±0.3316	2
8	1	2.7109 ±0.09	1.5932 ±0.0465	2.7853 ±0.085	20.1713 ±0.4306	2.8832 ±0.1224	1
9	1	3.5351 ±0.1507	0.989 ±0.0284	6.3814 ±0.4039	23.0993 ±0.4014	3.5015 ±0.1545	2
10	1	8.6658 ±0.3587	2.0031 ±0.082	9.6552 ±0.3916	38.7173 ±1.5326	8.4015 ±0.4684	1
11	1	4.2629 ±0.1701	1.4029 ±0.0338	5.8926 ±0.231	33.5862 ±1.4611	4.2518 ±0.1602	1
12	1	2.341 ±0.1278	0.9925 ±0.0391	2.6197 ±0.1257	9.6622 ±0.2759	2.2700 ±0.0876	2
13	1	9.2207 ±0.5352	1.0125 ±0.0371	11.7081 ±0.6212	30.9423 ±0.8233	9.7687 ±0.4136	1
14	1	4.9124 ±0.2754	2.7608 ±0.1029	6.0297 ±0.3171	12.8941 ±0.5844	4.9782 ±0.2761	1
15	1	3.0507 ±0.1608	0.9904 ±0.0271	3.532 ±0.2172	17.0661 ±0.3992	2.9431 ±0.1558	2
16	1	6.4099 ±0.2745	1.6855 ±0.0569	11.9509 ±0.6781	26.7849 ±0.9397	8.2778 ±0.3445	1
17	1	4.4485 ±0.1579	1.8949 ±0.0783	5.5042 ±0.2945	18.3166 ±0.4805	5.1133 ±1.1468	1
18	1	5.8697 ±0.2016	1.0004 ±0.0315	5.5823 ±0.2434	18.2243 ±0.6251	5.7903 ±0.1791	1
19	1	5.0525 ±0.222	2.3306 ±0.1526	10.9574 ±0.6314	28.6026 ±0.7308	5.0522 ±0.2551	1
20	1	1.9943 ±0.068	1.0122 ±0.0321	2.4858 ±0.5624	8.0891 ±0.3170	2.2462 ±0.1053	1
Avg	1	5.2162	1.4523	7.5410	23.0168	5.7680	1

● Average machine downtime

Table A4. Normalized average machine downtime and confidence interval

Average downtime of machines (maintenance + repair time)							
System	DDPM	DD	DDCPM	FCFS	RR	Cμ	DDPM Rank
1	1	2.2374 ±0.0364	1.9860 ±0.0329	2.2332 ±0.0367	2.2544 ±0.0403	2.2514 ±0.0381	1
2	1	1.3966 ±0.0268	0.9941 ±0.0216	1.3913 ±0.0496	1.4156 ±0.0265	1.4046 ±0.0266	2
3	1	1.2487 ±0.0279	0.9935 ±0.0209	1.2490 ±0.0207	1.2393 ±0.0234	1.2417 ±0.0192	2
4	1	1.8502 ±0.0281	1.9262 ±0.0832	2.0927 ±0.0262	1.9024 ±0.0245	2.0469 ±0.3300	1
5	1	2.1412 ±0.0326	1.9705 ±0.0253	2.1228 ±0.0266	2.0712 ±0.0223	2.1113 ±0.0305	1
6	1	1.3271 ±0.0270	1.0020 ±0.0197	1.3657 ±0.026	1.3244 ±0.0673	1.3575 ±0.0299	1
7	1	1.3354 ±0.0535	0.9979 ±0.0470	1.3896 ±0.0603	1.3498 ±0.0751	1.3666 ±0.0552	2
8	1	1.8377 ±0.0370	2.0003 ±0.0318	1.8362 ±0.0319	1.8395 ±0.0278	1.8461 ±0.0274	1
9	1	1.2508 ±0.0206	1.0026 ±0.0174	1.2788 ±0.0539	1.2707 ±0.0234	1.2599 ±0.0205	1
10	1	2.0956 ±0.0385	2.0319 ±0.0355	2.1482 ±0.0447	2.1434 ±0.0774	2.1202 ±0.0334	1
11	1	1.8645 ±0.0355	2.0074 ±0.0345	1.9221 ±0.0628	1.8707 ±0.0786	1.8876 ±0.0332	1
12	1	1.4447 ±0.0855	1.0253 ±0.0551	1.4331 ±0.0717	1.4451 ±0.0824	1.4469 ±0.0799	1
13	1	1.5420 ±0.0258	0.9986 ±0.0188	1.5072 ±0.0275	1.4709 ±0.0249	1.5209 ±0.0251	2
14	1	1.6250 ±0.023	1.9499 ±0.0222	1.6434 ±0.0686	1.6623 ±0.0235	1.6711 ±0.0246	1
15	1	1.1211 ±0.0152	0.9968 ±0.0142	1.1204 ±0.016	1.0490 ±0.0162	1.1214 ±0.0206	2
16	1	2.1184 ±0.0409	1.9889 ±0.0377	2.1161 ±0.0354	2.1212 ±0.0363	2.1065 ±0.0305	1
17	1	2.0111 ±0.0321	2.0063 ±0.0335	1.9675 ±0.0375	2.0213 ±0.0328	1.9807 ±0.0818	1
18	1	1.4255 ±0.0213	1.0047 ±0.0174	1.4061 ±0.0196	1.4403 ±0.0178	1.4177 ±0.0223	1
19	1	1.6291 ±0.0274	1.8812 ±0.079	1.7185 ±0.0231	1.6989 ±0.0288	1.6255 ±0.0296	1
20	1	1.0785 ±0.0226	1.0054 ±0.023	1.0955 ±0.0471	1.0591 ±0.0674	1.1129 ±0.0201	1
Avg	1	1.6290	1.4885	1.6519	1.6325	1.6449	1