Online Improvement of Condition-Based Maintenance Policy via Monte Carlo Tree Search

Michael Hoffman, Eunhye Song, Michael P. Brundage, and Soundar Kumara

***Abstract*— Often in manufacturing systems, scenarios arise where the demand for maintenance exceeds the capacity of maintenance resources. This results in the problem of allocating the limited resources among machines competing for them. This maintenance scheduling problem can be formulated as a Markov decision process (MDP) with the goal of finding the optimal dynamic maintenance action given the current system state. However, as the system becomes more complex, solving an MDP suffers from the curse of dimensionality. To overcome this issue, we propose a two-stage approach that first optimizes a static condition-based maintenance (CBM) policy using a genetic algorithm (GA) and then improves the policy online via Monte Carlo tree search (MCTS). The static policy significantly reduces the state space of the online problem by allowing us to ignore machines that are not sufficiently degraded. Furthermore, we formulate MCTS to seek a maintenance schedule that maximizes the long-term production volume of the system to reconcile the conflict between maintenance and production objectives. We demonstrate that the resulting online policy is an improvement over the static CBM policy found by GA.**

***Note to Practitioners*—This article proposes a method of scheduling maintenance in complex manufacturing systems in scenarios where there is frequent competition for maintenance resources. We use a condition-based maintenance policy that prescribes maintenance actions based on a machine's current health. However, when several machines are due for maintenance, a maintenance technician must choose between multiple competing jobs. While a common approach is to establish rules that dictate how maintenance jobs should be prioritized, such as the first-in, first-out rule, the goal of this work is to improve upon static policies in real time. We do this by strategically evaluating sequences of maintenance actions and playing out many "what–if" scenarios to see how the system will behave in the future. Implementation of the proposed method relies on the construction of a simulation model of the target system. This model is capable of retrieving the current state of the physical system, including the degradation state of machines, the availability of maintenance resources, and the distribution of parts throughout buffers in the system. We present several simulation experiments that demonstrate the improvement in system performance that our approach provides. Future work will aim to improve the efficiency of maintenance prioritization through online learning as well as more accurately identify manufacturing system configurations that will yield the greatest benefit of these methods.**

***Index Terms*— Condition-based maintenance (CBM), genetic algorithm (GA), Monte Carlo tree search (MCTS).**

Manuscript received March 15, 2021; revised May 25, 2021; accepted June 4, 2021. This article was recommended for publication by Editor F.-T. Cheng upon evaluation of the reviewers' comments. This work was supported by the National Institute of Standards and Technology through the Graduate Measurement Science and Engineering Fellowship. The work of Eunhye Song was supported by the National Science Foundation under Grant DMS-1854659. *(Corresponding author: Michael Hoffman.)*

Michael Hoffman, Eunhye Song, and Soundar Kumara are with the Department of Industrial and Manufacturing Engineering, Pennsylvania State University, University Park, PA 16802 USA (e-mail: hoffman@psu.edu; eus358@psu.edu; u1o@engr.psu.edu).

Michael P. Brundage is with the National Institute of Standards and Technology, Gaithersburg, MD 20899 USA (e-mail: michael.brundage@nist.gov).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TASE.2021.3088603.

Digital Object Identifier 10.1109/TASE.2021.3088603

I. Introduction

THE goal of maintenance in manufacturing is to support the availability and productivity of machines in the system. Implementation of maintenance is typically accomplished by using available system and machine data to develop a "policy" that dictates where and when maintenance should be scheduled. When downtime required for maintenance activities interferes with the established production plan [1], decision makers must evaluate the tradeoff between them. In 2016, the Annual Survey of Manufactures found that maintenance costs for American manufacturers exceeded U.S. \$50 billion annually [2]. This statistic accounts for costs attributed to labor, materials, and equipment used for maintenance but fails to consider the cost of lost production due to asset downtime. Significant downtime can result in failure to meet production objectives, particularly if a large portion of maintenance is unplanned.

In this work, we develop a maintenance framework that uses real-time system state information to schedule maintenance in support of system throughput by avoiding lost production through unnecessary idling of important machines. The framework provides a policy that aims to strategically schedule downtime for maintenance so that throughput disruption is avoided as much as possible. The proposed method can be applied to multicomponent systems of arbitrary configuration as well as to those with constrained maintenance capacity.

We classify maintenance jobs to be either corrective, indicating the "repair or replacement of components as a result of failure," or preventive, which includes the "repair or replacement of components at predetermined intervals/criteria" [3]. In general, corrective maintenance is more costly and time-consuming than preventive maintenance [4]. This cost and time increase is due to the severity of the damage as well as the uncertainty of a corrective maintenance task. For example, changing the oil on an engine every 3000 miles is a routine, generally low-cost maintenance task (preventive), and however, repairing or replacing the engine due to complete engine failure when not regularly changing oil (corrective) can be time-consuming and expensive in terms of the replacement cost and lost production. As such, preventive maintenance is desirable so that unplanned corrective maintenance is avoided.

By optimizing a maintenance policy, we attempt to balance the tradeoff of overmaintenance and undermaintenance. If maintenance jobs are scheduled too frequently, maintenance costs will be higher than necessary and downtime due to repair will be frequent. If jobs are not scheduled often enough, then there is a higher risk of costly machine failure between repairs. In order to optimize maintenance, it is useful to first classify the policy based on the set of decisions and available data (see the review in [5] for a thorough taxonomy of maintenance policies). In this work, we focus on a type of preventive maintenance known as condition-based maintenance (CBM), which relies on prescribing maintenance actions as a function of a machine's current state.

CBM has become popular with the proliferation of sensors and online data. CBM policies aim to identify the proper time to perform maintenance based on the real-time "health" condition of machines using information from internal or external sensors, product or process data, or routine inspections. Generally, a machine's health will deteriorate over time until it completely fails. An effective CBM policy will identify a point before this complete failure at which maintenance should be performed. Although this problem can be formulated as a Markov decision process (MDP) as reviewed in Section II, it becomes prohibitively expensive to solve using traditional methods as the problem scale increases and simplifying modeling assumptions are relaxed. Instead, another thread of research is finding a set of static health index thresholds at which to schedule maintenance for each machine. In optimizing these thresholds, the goal is to maximize the long-term average performance of the system; however, this problem is also challenging due to a combinatorially large solution space of health indices and lack of an analytical expression of the performance measure. The goal of this work is to find an optimal CBM policy for a general multicomponent manufacturing system that is subject to maintenance capacity constraints.

Our approach is to first find a static CBM policy that maximizes the throughput using a genetic algorithm (GA). We use a static queueing priority discipline whenever the number of machines due for maintenance exceeds the available maintenance capacity. The output of the GA is a health index threshold for each machine that determines when a machine will request maintenance.

Once a static CBM policy is found, it is adopted until a maintenance scheduling conflict occurs. We solve an MDP problem online whenever the number of machines requesting maintenance exceeds the available maintenance capacity to gain possible improvement upon the performance of the static policy. Here, what we mean by "online" is that the static policy is updated by incorporating the real-time information of the system. The state space for the MDP is reduced significantly compared to the original problem since we only consider taking a maintenance action for a machine that has exceeded its health index threshold as specified by the static CBM policy. Thus, for each machine, health states below the threshold can be merged together in the online scheduling problem. For the solution method, we adopt Monte Carlo tree search (MCTS) to efficiently navigate the state–action space under a given computation budget. The result of this search is the "best action" in the form of which machine to repair next given the current state of machines. This procedure is repeated each time a maintenance resource must choose between two or more pending maintenance jobs. Our proposed approach seeks optimal maintenance decisions in a dynamic state space with the goal of maximizing system performance.

As defined in [6], online scheduling problems are distinguished by incomplete information due to either a lack of knowledge of future jobs, unknown duration of scheduled jobs, or unknown times between machine failures. The problem formulated in this work has each of these characteristics and so we refer to our method of maintenance conflict resolution as an "online improvement" of the static policy. This is in contrast to use of the term "online" elsewhere in maintenance optimization literature to refer to online parameter estimation for nonstationary system processes. For example, Elwany *et al.* [7] and Chen *et al.* [8] updated the parameters of the degradation process for a single-component system as more degradation observations are gathered over time.

Our contribution can be summarized as follows:

1) a framework for optimizing CBM for systems of arbitrary structure;
2) online improvement of a static maintenance policy using real-time system state information;
3) consideration of a capacity on maintenance resources.

The rest of this article is organized as follows. Section II reviews the existing CBM literature. The problem statement is in Section III, and the methodology is outlined in Section IV. Section V shows experiment results, and finally, conclusions are given in Section VI.

II. Literature Review

The goal of CBM is to perform maintenance only when necessary by monitoring the machine status either continuously or periodically. Maintenance is scheduled when an abnormal signal, such as those from sensor readings or other sources, is identified [9]. This approach, when performed correctly, avoids unnecessary repairs on a healthy machine and prevents costly breakdowns. A CBM policy typically specifies maintenance actions as a function of a machine's current state. In addition to identifying the optimal state–action mapping, there is the challenge of prioritizing maintenance jobs when maintenance resources are limited. There has been substantial research in the area of condition monitoring and CBM in a variety of contexts. This section examines the literature most relevant to the problem that is addressed by this work.

The selection of a CBM policy is dependent on the assumed mode of deterioration; a discrete-time Markov degradation process is particularly well-suited for CBM. Under this model, machines begin in a perfect health state and move to states of increased degradation over time according to the specified degradation state transition matrix. The advantages of this model are that we only need to know the current state of a machine to estimate its future degradation and the discrete states provide a well-defined and intuitive domain for the CBM decision variables [10]. This model also allows for variations, such as sudden failures in the form of system shocks [11] or stochastic dependence of degradation among machines [12]. Examples of previous work that employ the continuously monitored discrete health state Markov model for maintenance optimization problems include [13]–[17].

In multicomponent manufacturing systems, the routing configuration of machines has a substantial impact on the formulation of the CBM policy optimization problem. Downtime events due to maintenance or failure of a machine can propagate through a system causing other machines to become starved or blocked [18]. Because of this interaction among machines, an independently optimized policy for a single machine may not be optimal when the machine is examined in the context of a larger manufacturing system [19]. Such dependence is designated in [20] as "performance dependence." Much of the previous work on maintenance optimization problems can be classified by the type of component configuration it considers, including series, parallel, series–parallel, k-out-of-N arrangements, and various combinations of these. A thorough description of each arrangement in the context of reliability is given in [21]. Often, an optimal policy is derived under the assumptions of the system configuration. However, many complex real-world systems cannot be characterized by one of these configurations, yet little previous work has allowed for arbitrary system structure. For instance, Gupta and Lawsirirat [22] and Nguyen *et al.* [23], [24] optimized CBM for general system configurations but relied on the assumption that maintenance activities are instantaneous and that there is no limit to the number of components that can be repaired simultaneously. Maintenance duration is imposed in [25], yet maintenance capacity is unlimited. A capacity on maintenance resources is a common real-world constraint that this work also aims to address.

Constraining maintenance capacity increases the complexity of a maintenance optimization problem because there will be cases where multiple machines are competing for the same maintenance resource. These situations impose the additional challenge of resolving such conflicts.

One class of maintenance policies that deal explicitly with maintenance capacity is selective maintenance. Often, under a selective maintenance policy, each job consumes some amount of constrained resources (such as time or labor) during a "maintenance break" for the entire system. Maintenance breaks occur between two consecutive production missions and the objective when scheduling maintenance is usually to minimize maintenance time or cost or to maximize asset reliability during the next mission. This class of policies was first introduced in [26] and a thorough review of recent work in selective maintenance is given in [27]. Although selective maintenance accounts for maintenance capacity, having predetermined maintenance breaks and fixed mission duration is not always applicable. In a realistic manufacturing setting, unplanned failures can occur at any point in time and need to be addressed immediately, whereas selective maintenance does not allow production to be interrupted until the next maintenance break.

Other recent work that has considered maintenance capacity includes [13], which deals with a series–parallel system (a serial line of subsystems each with multiple components in parallel) with a limited number of maintenance workers. They assume that n machines are in a system and there are at least $n - 1$ maintenance resources available, implying that there is no instance where a maintenance resource must decide among multiple machines to maintain. Meanwhile, de Smidt-Destombes *et al.* [28] and Moghaddass *et al.* [29] optimized the maintenance policy for a k-out-of-N system (a system of N identical components of which at least k must be functional for the system to be operational) with maintenance capacity. Since all machines are identical, the choice of which ones to repair has no impact on system performance.

A system consisting of nonuniform machines each with multiple levels of degradation as well as constrained maintenance capacity will have a significantly larger state space. In [14], the optimal CBM policy for stochastically and economically dependent components is a mapping of optimal maintenance actions to each possible system state. They use a factored MDP and approximate linear programming to overcome the exponential growth of the state space. Their formulation, however, relies on the assumption that the system contains only two-machine subsystems arranged in parallel and is not generalizable to more complex arrangements.

Maintenance priority assignment is used in [30] to schedule jobs under maintenance capacity. They seek the best priority arrangement (which is equivalent to a static sequence of jobs) for pending jobs in a system of a given state. Although this approach addresses the problem of scheduling maintenance on heterogeneous machines under a fixed capacity, it uses a predetermined maintenance schedule that does not adapt to the evolving state of the system. Even if a schedule performs well compared to common scheduling heuristics (first-in, first-out (FIFO), shortest processing time first (SPTF), and so on), it may not necessarily be the best policy in every scenario. In [31], FIFO is improved by using a priority heuristic based on the concept of the opportunity window for maintenance jobs that minimizes the disruption to system production. However, the priority measure considers each job in isolation and does not accurately evaluate the impact that a sequence of jobs has on system performance. There is a need for a method of dynamic maintenance scheduling that makes the best scheduling decisions under the current conditions.

The literature reviewed thus far has thoroughly studied the optimization of CBM policies under a variety of configurations. However, simplifying assumptions have been made in previous work that limits the effectiveness of these maintenance strategies in a generalized dynamic manufacturing setting with maintenance capacity. Our proposed approach improves upon this work by first seeking an optimal set of static maintenance thresholds that serve to reduce the size of the problem state space. We then update this policy online when needed by formulating and solving an MDP using MCTS.

III. System Description

See [32] for a classification and description of various types of production systems. We focus on discrete manufacturing systems with a finite maintenance capacity that may be greater than 1. In particular, our method can be applied not only to serial lines but also to complex manufacturing systems, assuming that a simulation model of the system can be constructed (see Fig. 4 for an example of such a system). In this section, we further elaborate on the target system.

A. Notation

We define the major notation used throughout this article in the following.

1) *n:* Number of machines in the system.
2) *M_i:* Label of the ith machine in the system. Machines M_i and M_{i+1} are not necessarily adjacent.
3) *t_i:* Cycle time of machine M_i.
4) *$\mathbf{P}_i$:* Degradation transition matrix of machine M_i.
5) *$H_i(t)$:* Health index of machine M_i at time t.
6) *h_i:* Integer-valued health index threshold at which CBM is scheduled for machine M_i.
7) *$h_{\max}$:* Health index at which a machine undergoes failure.
8) *$\mathbf{x} = \{h_1, h_2, \ldots, h_n\}$:* CBM maintenance policy of the system.
9) *S_j:* A station consisting of at least one machine in parallel. Multiple machines in a station are assumed to be identical.
10) *$\mathbf{x}_S = \{h_{S_1}, h_{S_2}, \ldots, h_{S_m}\}$:* Station-level CBM policy where maintenance threshold h_{S_j} is applied to all machines at station S_j.

B. Machine Degradation

Systems subject to a CBM strategy assume that the health condition of a machine can be monitored either periodically or continuously. Continuously monitored signals, such as vibration data, acoustic data, or temperature, are now commonplace in many manufacturing settings as sensing and data communication become increasingly cost-effective [33]. Furthermore, multiple disparate signals can be aggregated into a monotonic composite health index that reflects the state of machine degradation for the purpose of maintenance decision support [34]. We assume that this health index is continuously observable and model its behavior over time as a discrete-time Markov chain where higher states represent increased degradation and wear. Machines undergo time-based degradation, meaning that the degradation of a working machine continues even if it is idle due to blockage or starvation.

A machine M_i is considered to be in perfect health at time t when its health index is equal to 0, ($H_i(t) = 0$). Eventually, the machine will reach the maximum health state at future time t', indicating that the machine has failed ($H_i(t') = h_{\max}$). The degradation parameters, including $h_{\max}$, may be different across machines in the system, but for ease of exposition, we assume that it is the same in the following. Any time a machine receives maintenance, either preventive or corrective, its health index is restored to 0 once maintenance is finished, although this assumption can be relaxed easily.

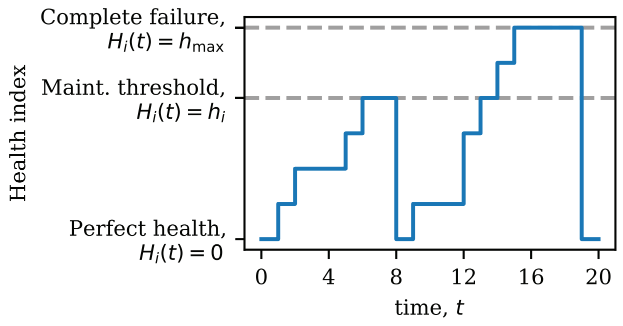


Fig. 1. Example of Markovian machine degradation.

The degradation transition matrices are assumed to be upper triangular. This follows the monotonicity assumption of the health index measure by ensuring that a machine cannot transition to a healthier state without the intervention of a maintenance action. An example of machine degradation over time is shown in Fig. 1. In this example, the machine reaches its threshold for maintenance at time $t = 6$ and is restored to perfect health at $t = 8$. The machine reaches its maintenance threshold again at $t = 14$, but is not repaired immediately, perhaps because maintenance resources are occupied elsewhere in the system. The machine reaches failure at $t = 16$, at which point it can only be restored by corrective repair.

Although we do not demonstrate them in this work, general Markov degradation transition matrices that forego the assumption of an upper triangular structure are compatible with our method. Such a model implies that a machine may transition to a state of lesser degradation without the intervention of a maintenance action. In these instances, a machine should request maintenance once its degradation level reaches the threshold for maintenance. If the machine later transitions to a degradation state below the threshold before receiving maintenance, the maintenance request should persist and the machine should continue to be considered for maintenance.

In this work, we specify the machine degradation transition probabilities, though several methods exist for estimating a Markov degradation transition matrix from observed data (see [35], [36]).

C. Maintenance Queue

If the number of machines in the system exceeds the capacity for maintenance, it is possible that a scenario will eventually arise where we must choose which machine to repair first and which jobs to defer. To assist with these scheduling decisions, machines waiting for maintenance enter a virtual maintenance queue. The maintenance queue contains machines that are failed and require corrective action as well as functional machines that are awaiting preventive repairs. Under CBM, a machine enters the maintenance queue when its health reaches the CBM threshold prescribed by the policy. When a maintenance resource becomes available, it chooses a machine to repair from among those in the queue.

D. Assumptions

The following assumptions are made regarding the behavior of the system examined in this work.

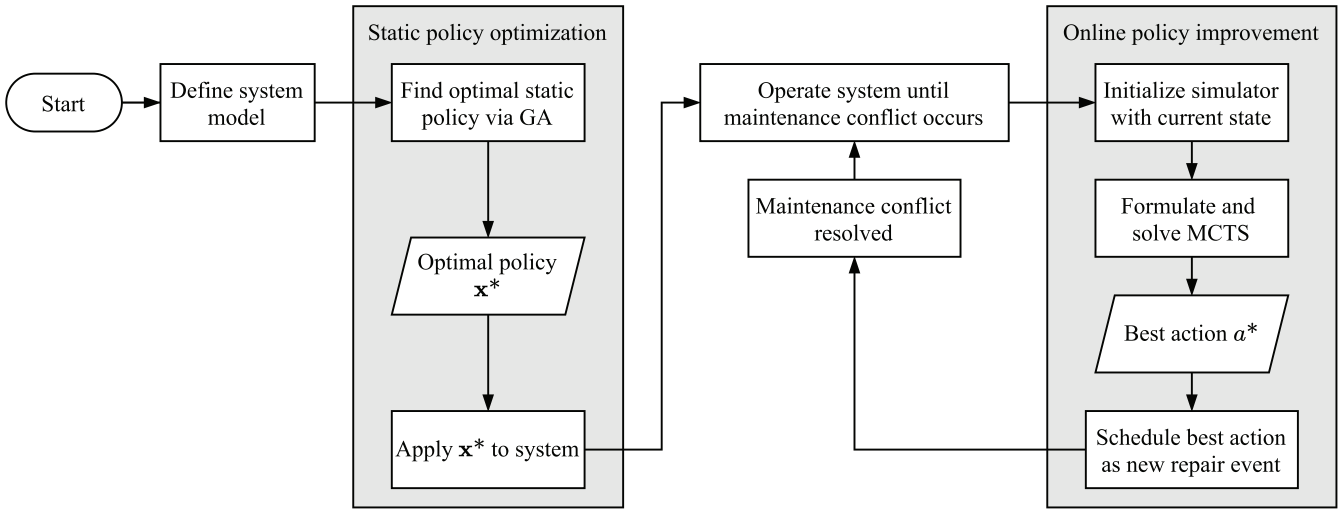

Static policy optimization
Online policy improvement
Start
Define system model
Find optimal static policy via GA
Optimal policy $\mathbf{x}^*$
Apply $\mathbf{x}^*$ to system
Operate system until maintenance conflict occurs
Maintenance conflict resolved
Initialize simulator with current state
Formulate and solve MCTS
Best action a^*
Schedule best action as new repair event

Fig. 2. Procedure for static policy optimization and online improvement. Once the optimal static policy is found, the system operates as usual until a maintenance conflict occurs (the number of machines in the queue exceeds the available maintenance capacity). We then use MCTS to find the best action, begin the associated repair, and continue operating the system.

1) Each machine operates at its full capacity unless it is in the failed state.
2) Machines produce identical discrete parts that are processed for the duration of its cycle time and then placed downstream.
3) When a machine fails, any part in process is discarded.
4) Any maintenance action on a machine will restore that machine to perfect health.

IV. Methodology

We formulate the CBM optimization problem in two parts: static CBM threshold optimization and its online improvement whenever there are competing maintenance jobs. The two parts of this formulation are summarized in Fig. 2. The static CBM policy reduces the state space by excluding relatively healthy machines from consideration for maintenance. This state-space reduction allows MCTS to avoid allocating its search budget to evaluating the possible action of repairing a healthy machine, which is not likely to be optimal if there are other machines in more advanced degradation states.

A. Static Policy Optimization

The optimization objective is to find a CBM policy that maximizes the expected production volume of the system (in number of units) over a fixed time horizon. The decision variables of the policy are the health index thresholds for machines at which maintenance is scheduled. Recall that a policy is denoted by $\mathbf{x} = \{h_1, h_2, \ldots, h_n\}$. For n machines with a maximum health state of $h_{\max}$, there are $(h_{\max})^n$ possible solutions. Since we consider a constrained maintenance capacity, the threshold for each machine reflects the time at which maintenance is requested, which is not necessarily the time at which maintenance is executed. If the health of a machine reaches its threshold for maintenance when no maintenance resources are available, the machine will wait until an ongoing maintenance action is completed. The optimal maintenance policy will account for this potential waiting time and assign maintenance thresholds accordingly.

As the complexity of a manufacturing system increases, it can quickly become difficult to analytically model its performance without imposing unrealistic simplifying assumptions [37]. For this reason, we use simulation to model the behavior of the system and estimate the expected production volume under a given policy. The solution space of policies grows exponentially as the number of machines increases, so we employ a GA to seek an optimal policy.

GAs are a metaheuristic method of problem-solving that attempts to replicate evolutionary behavior observed in nature. A population of individuals (or maintenance policies) "evolves" over time by selecting the best candidates, as determined by a fitness function, to produce the succeeding generation. Starting with an initial set of random individuals, a new population is generated by reproduction and added to the total population. The best individuals from this group are then chosen to produce the next generation, and the process repeats until some termination criteria are met. The problem of optimizing a CBM policy is well-suited for GAs as this approach is robust and effective for large, complex manufacturing systems [38].

We apply GA as it is described in [39]. For the CBM policy optimization problem, we use a candidate policy $\mathbf{x} = \{h_1, h_2, \ldots, h_n\}$ as a value-encoded individual. The fitness of each individual is determined by the production volume that is obtained by the system over a fixed time horizon when simulated under the policy it represents. Individuals are selected for reproduction with likelihood proportional to their estimated fitness so that better solutions are more likely to be chosen to aid in the creation of the next generation. Once two individuals are selected for reproduction, an offspring is produced using uniform single-point crossover, that is, for two parent policies $\mathbf{x}_1 = \{h_1^1, h_2^1, \ldots, h_n^1\}$ and $\mathbf{x}_2 = \{h_1^2, h_2^2, \ldots, h_n^2\}$, we choose a position uniformly between 0 and n (inclusive) to serve as the crossover point. Elements to the left of this point in $\mathbf{x}_1$ are combined with elements to the right of this point in $\mathbf{x}_2$ to form a complete policy and create a new individual. For example, if the crossover point is m, the offspring of $\mathbf{x}_1$ and $\mathbf{x}_2$ would be $\{h_1^1, \ldots h_m^1, h_{m+1}^2, \ldots h_n^2\}$.

To maintain diversity in the population, there is a small chance of mutation in each element of a newly created individual. If mutation occurs, the element changed to a random value on its domain. We also employ an elitist strategy in which the best subset of individuals from the current generation is carried over into the new generation. This approach improves the performance of the GA by retaining good solutions once they are identified [40]. For our implementation, we use a population size of 30 and a mutation probability of 0.01.

B. Online Improvement of Static Policy

To improve upon the static policy that applies FIFO to resolve maintenance scheduling conflicts, we use Monte Carlo tree search (MCTS) to seek the best machine to maintain when a conflict occurs in online fashion. We formulate MCTS online each time a maintenance resource becomes available in the system and there is more than one machine in the maintenance queue. MCTS has seen success in its application to artificial intelligence in games by using a single-player stochastic game formulation or an MDP, where there is one decision maker and the state transition is random.

An MDP is defined by a tuple $M = \langle S, A, P, R, \gamma \rangle$, where S is the finite state space, A is the finite action space, P is the transition function such that $P(s, a, s')$ is the probability of transitioning to state s' by taking action $a \in A$ in state s, R is the reward function that gives the expected immediate reward of transitioning to state s' after taking action a in s, $R(s, a, s')$, and $\gamma \in [0, 1]$ is the discount factor. Since we consider the infinite horizon case, we set $\gamma < 1$ so that the cumulative rewards over time converge to a finite sum [41]. In general, a discount factor closer to 0 favors obtaining earlier rewards. A larger discount factor, on the other hand, gives more preference to deferred rewards that occur further from the initial state. The return in the initial state s_0 is given by

$$\begin{aligned} G^{\pi}(s_0) &= R(s_0, a_0, s1) + \gamma R(s_1, a_1, s_2) + \cdots \\ &= \sum_{k=0}^{\infty} \gamma^{t(s_k)} R(s_k, a_k, s_{k+1}) \end{aligned} \tag{1}$$

where $s_0, s_1, s_2, \ldots$ is the series of states encountered following a sequence of actions determined by policy π, which maps each state to an action, and $t(s_k)$ is the integer-valued simulation clock when it enters s_k. The goal is to identify the policy π that maximizes the expected return.

For the maintenance scheduling problem, the state of the system is defined by the health index for each machine, remaining processing time for machines with a part in progress, and elapsed repair time for machines under repair as well as the current level of each buffer. In practice, this method requires that this information is readily available in real time. As discussed previously, we assume that the health of each machine is known continuously through state-of-the-art condition monitoring techniques. In most modern manufacturing environments, information regarding the work in progress at each machine and the current buffer levels are typically available through the use of manufacturing execution system (MES) software or similar technologies [42]. For machines currently under repair, we need to estimate the distribution of the remaining repair time in order to simulate the future behavior from the current point in time. Since the time to repair distribution is known, we can find the appropriate conditional probability distribution of the remaining repair time, given the time that has already elapsed. This initial current state is denoted s_0.

The state of a system at time t can be represented as the set of state variables for each component in the system. For a machine M_i, the health state variable is -1 if it is under repair and 0 if $H_i(t) < h_i$ and $H_i(t) \in [h_i, h_{\max}]$ otherwise. We use a health state of 0 to indicate a general "healthy" state for machines whose health does not exceed their threshold for maintenance. There are therefore $h_{\max} - h_i + 3$ possible encoded health states for M_i.

The remaining process time is encoded as -1 if M_i is failed, under repair, or starved (it is in an operational state and does not have a part), 0 if it is blocked (it is in an operational state and holding a completed part and cannot place it downstream), or $t_i - u_i$ if it is holding a part that has been processed for duration $u_i \in [0, t_i - 1]$. The remaining process time can take on one of $t_i + 2$ values for each machine.

The elapsed repair time of a machine is encoded -1 if it is not under repair or r_i if it began repair at time $t - r_i$. Therefore, for a general time to repair distribution $\mathcal{G}$, the remaining repair time on M_i (q) is distributed $\mathcal{G}(q | q \geq r_i)$. The repair state can be simplified in the case of geometrically distributed repair times; the repair state can be represented as a binary variable with 1 indicating that a machine is currently under repair and 0 indicating otherwise.

Finally, the level of each buffer is integer-valued between 0 and its maximum capacity, b_j. Combining each of these state variables, we obtain a numerical state vector, which allows for convenient representation and comparison of states. In particular, the total size of the state space for a system consisting of n machines subject to geometric repair times and m buffers can be bounded above by

$$|S| \leq \left[\prod_{i=1}^{n} (h_{\max} - h_i + 3) \cdot (t_i + 2) \cdot 2\right] \cdot \left[\prod_{j=1}^{m} (b_j + 1)\right]. \tag{2}$$

Notice that the upper bound increases exponentially with each additional machine or buffer. However, the upper bound tends to be loose since some included states are not valid. For instance, states where the number of machines under repair exceeds the capacity for maintenance will never be encountered.

From each state s, the available set of actions $A(s)$ is to repair a machine currently in the maintenance queue or, if the queue is empty, do nothing and wait until a machine enters the queue. The maximum number of possible actions in a particular state is therefore equal to the number of machines currently in the maintenance queue, which is again upper bounded by n. When executing an action, the transition to the new state is sampled using simulation since the transition function P is unknown. The reward of a state s is the ratio of observed production in units to the ideal production in that state since the initial state, where the ideal production is the number of units that would be produced if machines in the system experienced no downtime. For example, $R(s_0, a_0, s_1)$ is equal to the proportion of ideal production obtained between time $t(s_0)$ and time $t(s_1)$. Since the ideal production is the maximum that is achievable and the minimum possible production count is 0, $R(s, a, s') \in [0, 1]$.

Given this MDP formulation, each node of the search tree represents a unique state–action sequence that starts from the current state, s_0. Choosing action a_0 from this state results in traversing an edge of the tree from s_0 to s_1. Thus, a node of depth d represents a state–action sequence $s_0, a_0, s_1, a_1, \ldots, a_{d-1}, s_d$.

The MCTS procedure, as described in [43], includes four basic steps that are repeated until a specified search budget is expended: selection, expansion, simulation, and backpropagation. MCTS is an anytime algorithm, meaning that it can be terminated after any duration of time or any number of iterations. In practice, the run time of the algorithm can be terminated depending on the amount of time available for making maintenance decisions. Each of these steps is described in detail within the context of the maintenance scheduling problem in the following.

1) Selection: The tree is initialized with the root node, which represents the real system in its current state s_0. Since the tree contains only one node at the first iteration, the selection step is trivial. For the node selection policy beyond the first iteration, we must choose an action from each node and sample the future state that results from taking that action in the current state. We use the action selection strategy described in [44] and the upper confidence bound for trees (UCT) algorithm proposed in [45] to evaluate candidate actions. Given the set of possible actions in state s, $A(s)$, an action a^* is chosen such that

$$a^* = \max_{a \in A(s)} \left(\bar{R}_a + 2C_p \sqrt{\frac{2 \ln n}{n_a}} \right) \tag{3}$$

where $\bar{R}_a$ is the average reward from taking action a from the current node, C_p is the exploration constant, n is the number of times that the current node has been visited, and n_a is the number of times that action a has been chosen so far. Larger values of C_p will favor the search of less visited actions, whereas smaller values will cause the search to spend more time evaluating promising actions. Because $R_a \in [0, 1]$, we choose $C_p = 1/\sqrt{2}$, which has been shown to be ideal for rewards in this range [45].

Once an action is selected, the resulting future state is sampled by simulating the execution of that action. The node selection process is repeated recursively until we reach either an unexpanded node (as defined under the expansion step) or a terminal node (as defined under the simulation step).

2) Expansion: A node is considered "expanded" if every available action from that node has been sampled at least once. Otherwise, the node is unexpanded. If an unexpanded node is encountered during the node selection procedure, we randomly select a valid action that has not yet been chosen and execute that action to sample a future state. A new node representing this future state is then added to the current search tree. We simulate from the new node to obtain an estimate of its reward.

3) Simulation: The rollout policy used in the simulation step determines how we obtain an outcome from an intermediate nonterminal node. The default policy chooses random children nodes (i.e., selecting random available actions) until finding a leaf node of sufficient depth. The reward at each node is the production volume that is observed over the elapsed time horizon divided by the ideal production over that horizon. We use a discount factor of $\gamma = 0.9624$, which results in 99% of the cumulative reward at the root node being obtained within 2 h of the initial time.

4) Backpropagation: After the simulation result is obtained, the statistics are updated for nodes along the path from the simulated leaf node to the root. These statistics include the number of visits to each node and their cumulative reward. The number of visits is incremented only for nodes that were chosen by the node selection policy. Nodes visited during the rollout of the simulation step are not considered visited.

5) Termination: Once the search budget is expended, we have an estimation of the expected reward for each action available in the current state s_0. This reward represents the expected proportion of ideal production that is achieved over the specified 6-h evaluation horizon. Typically, in MCTS, the action with the highest average observed reward is chosen as the best. However, with a fixed search budget, we may not be able to conclude that a single action is statistically significantly better than the others. We therefore use statistical testing to find the best candidate actions and apply knowledge of the system to resolve ties among such actions.

To find the best set of actions, we first use a one-way analysis of variance to test the hypothesis that each sample of rewards shares an equal mean. If this hypothesis is not rejected, we cannot conclude that the mean rewards for each action are different, and thus, we include all available actions in the best set. Otherwise, we use Tukey's honestly significant difference (HSD) test, a multiple comparison test for determining whether the mean of several samples is significantly different. This test overcomes the inflated type I error that may occur when conducting pairwise statistical tests independently.

Tukey's HSD test provides a pairwise comparison of the mean reward for each action. From this result, we choose the actions for which there is no statistically significantly better action as the best set. If there is more than one potential action in this set, we select the action of repairing the machine with the earliest request for maintenance. This method of breaking ties among the best actions ensures that the scheduling procedure will not perform worse than FIFO in the cases where MCTS cannot clearly identify a single best action.

C. Alternative Scheduling Disciplines

In addition to FIFO and MCTS, we compare several other commonly used methods of maintenance conflict resolution: SPTF, longest processing time first (LPTF), and Birnbaum importance.

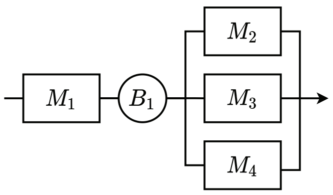


Fig. 3. Four-machine production line.

Under an SPTF discipline, the machine in the maintenance queue with the shortest expected repair time is chosen first. Since we consider two types of maintenance jobs, preventive and corrective, SPTF results in preferring preventive jobs over corrective ones. If there are multiple machines awaiting preventive maintenance, we break ties according to FIFO.

Similar to SPTF, under LPTF, we choose maintenance jobs with longer expected repair times. This strategy results in a preference for completing corrective jobs earlier. Ties in this case are also broken with FIFO.

The Birnbaum importance aims to quantify the structural importance of each component in a system [46]. It has been used effectively for maintenance prioritization in complex manufacturing systems by identifying machines with a greater potential to disrupt system throughput and selecting these critical machines for prioritized maintenance [23], [47]. Consider, for example, the four-machine system shown in Fig. 3. If machine M_1 has failed while M_2, M_3, and M_4 are functioning, it blocks the throughput of the entire system since each part produced must pass through M_1. However, if M_2 has failed while the rest of the machines are working, the system throughput is not disrupted since parts can traverse through M_3 or M_4 as alternatives to M_2 due to the parallel arrangement of these machines.

When calculating the Birnbaum importance, the operational state of each machine M_i is represented by a binary variable y_i, where

$$y_i = \begin{cases} 1, & \text{if } M_i \text{ is operational} \\ 0, & \text{if } M_i \text{ is failed} \end{cases} \tag{4}$$

and the state vector of the system is $\mathbf{y} = (y_1, y_2, \ldots, y_n)$. The system state is described by the binary function $\phi(\mathbf{y})$ where

$$\phi(\mathbf{y}) = \begin{cases} 1, & \text{if the system is operational} \\ 0, & \text{otherwise.} \end{cases} \tag{5}$$

For the purposes of this calculation, we consider the system to be operational if there is at least one path of working machines that allows parts to completely traverse through the system. The Birnibaum importance of machine M_i is then given by

$$IB_i = \frac{\sum_{(\cdot_i,\mathbf{y})}(\phi(1_i,\mathbf{y}) - \phi(0_i,\mathbf{y}))}{2^{n-1}} \tag{6}$$

where $(\cdot_i, \mathbf{y})$ is the set of all 2^{n-1} state vectors with $y_i \in \{0, 1\}$, $\phi(1_i, \mathbf{y})$ is the system state with $y_i = 1$, and $\phi(0_i, \mathbf{y})$ is the system state with $y_i = 0$. Therefore, the term $\phi(1_i, \mathbf{y}) - \phi(0_i, \mathbf{y})$ is equal to 1 if a failure on M_i would change the system state from operational to nonoperational. In these cases, M_i is considered to be critical to the functionality of the system.

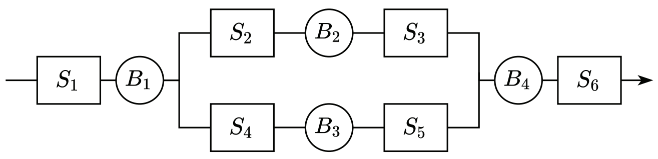


Fig. 4. Six-station complex production line.

TABLE I
STATIONS FOR SYSTEM CONFIGURATIONS A AND B

	Configuration	
	A	B
S_1	$\{M_1\}$	
S_2	$\{M_2\}$	
S_3	$\{M_3, M_4, M_5\}$	
S_4	$\{M_6, M_7\}$	$\{M_6, \ldots, M_{13}\}$
S_5	$\{M_8\}$	$\{M_{14}\}$
S_6	$\{M_9\}$	$\{M_{15}\}$

Machines that are deemed critical in a higher proportion of system states are assigned a higher measure of importance and should be prioritized for maintenance. If multiple machines due for maintenance are tied for the highest importance, then one is selected randomly.

Each of these scheduling rules serves as a baseline rule for obtaining a static policy to which our proposed online scheduling method is compared.

V. RESULTS

In this section, we compare the performance of the online scheduling improvement to static CBM policies for the complex production line shown in Fig. 4 under the two machine configurations outlined in Table I. For each arrangement, we consider machines in a parallel station (such as machines M_3, M_4, and M_5 in this example) to be identical with a common cycle time and degradation rate.

We first optimize the static CBM policy using GA under the aforementioned static scheduling rules and then improve each policy using MCTS. We use the same maintenance threshold for each machine at a station when applying the CBM policy.

For each example shown, machines at stations S_1–S_6 have constant cycle times of 10, 60, 180, 40, 20, and 10 min, respectively. Each buffer has a maximum capacity of ten parts. Also, in all scenarios, the duration of maintenance follows the geometric distribution, which is the distribution of X number of independent Bernoulli trials with a specified probability of success that are needed until one success is observed. For success probability p, the probability mass function is $P(X = k) = (1 - p)^{k-1}p$ and the expected value of X is $1/p$. We choose $p = 1/20$ for all preventive maintenance jobs and consider several success probabilities for corrective maintenance in the following experiments.

We use the matrix $\mathbf{P}_i$ for the degradation transition matrix of machine M_i of the form shown in (7), as shown at the bottom of the next page. In this matrix, p_i is the probability of degrading by one unit at each time step or the degradation rate of M_i. We also allow for the possibility of sudden failures from any health state as indicated by the probability f_j.

TABLE II
STATION-LEVEL BIRNBAUM IMPORTANCE FOR NINE-MACHINE LINE CONFIGURATIONS A AND B

		Birnbaum importance					
		S_1	S_2	S_3	S_4	S_5	S_6
Config.	A	0.324	0.137	0.020	0.035	0.105	0.324
	B	0.359	0.110	0.016	0.001	0.140	0.359

TABLE III
EXPERIMENTAL SYSTEM CONFIGURATION SETTINGS

	Level		
Factor	1	2	3
1. Degradation rate, p_i	0.03	0.07	-
2. Maintenance capacity	1	2	3
3. Corrective time to repair	$Geom(1/60)$	$Geom(1/80)$	-
4. Machine configuration	A	B	-

For each of the following results, we set $f_j = 0.005 \cdot (j + 1)$ for all machines. This results in an increasing probability of sudden failure at higher states of degradation. Bear in mind that our method allows much more complex upper triangular degradation matrices than (7).

A. Maintenance Policy Optimization

We demonstrate the optimization of the CBM policy for the system in Fig. 4 where each machine is subject to a degradation rate of $p_i = 0.03$, the maintenance capacity is 3, and corrective maintenance duration is geometrically distributed with success probability 1/60 (configuration 10 in Table IV). We use a warm-up period of one week to achieve the steady-state performance and a time horizon of one week for evaluation. Since the dynamics of the system are stationary, the duration of the evaluation horizon will not affect optimization as long as the system is observed in its steady state. To justify this warm-up period, we consider the average throughput (in parts per day) over time of this system, as shown in Fig. 5. System throughput is measured as the rate at which finished parts leave the system, which is equivalent to the observed rate of production of station S_6 in this arrangement of machines. At time $t = 0$, the system is empty and all machines begin in a perfectly healthy state. As the buffers become populated, the throughput of the system eventually converges to an approximately constant level. Beyond this point, the throughput is stable, and therefore, we can conclude that the system is in its steady state.

Fig. 6 shows the convergence of the solution under the Birnbaum priority scheduling averaged over 30 runs of the GA. Across these runs, the best station-level policy found was $\mathbf{x}_S = \{7, 8, 10, 10, 7, 10\}$ (recall that $h_{\max} = 10$) with an average objective function value of 419.01 units produced over the one week evaluation period.

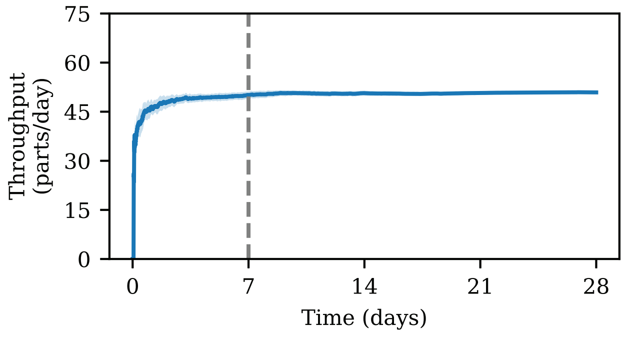


Fig. 5. Average throughput of the system over 28 days. The vertical line at a time of one week marks the end of the chosen warm-up period.

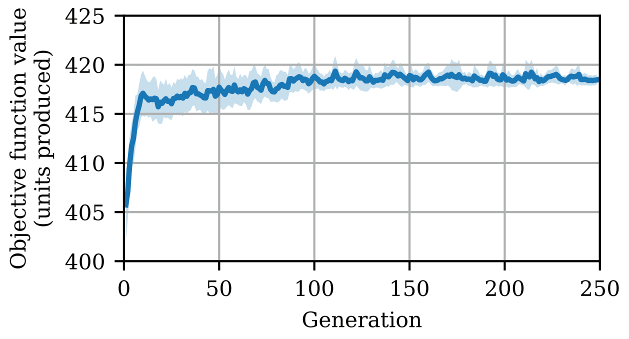


Fig. 6. Estimated objective function value over 250 generations averaged across 30 runs with a shaded 95% confidence interval of the mean value.

Fig. 7 shows the improvement in production throughput under the optimal policy when using MCTS scheduling. MCTS was applied each time a maintenance scheduling conflict occurred. The improved policy provides an average throughput increase of 3.37 parts per day or 23.60 parts per week. This is a 6.19% improvement for this configuration.

B. Scheduling Problem Instance

We demonstrate the online policy improvement by generating an instance of maintenance scheduling conflict in the system described in Section V-A by simulating the system until four machines are waiting in the maintenance queue. We formulate the online scheduling problem at the point in time when the maintenance resource is released and must now decide which machine to repair next.

At simulation time $t = 366$ min of the replication, a maintenance resource completes a repair on machine M_{13} and

$$\mathbf{P}_i = \begin{bmatrix} 1-(p_i+f_0) & p_i & & & & f_0 \\ & 1-(p_i+f_1) & p_i & & & f_1 \\ & & & \ddots & & \vdots \\ & & & & 1-(p_i+f_{h_{\max}-1}) & p_i + f_{h_{\max}-1} \\ & & & & & 1 \end{bmatrix} \qquad (7)$$

TABLE IV

SYSTEM CONFIGURATION EXPERIMENT RESULTS

Config.	Factor Levels				FIFO				SPTF				LPTF				Birnbaum			
	1	2	3	4	Baseline Prod.	MCTS Prod.	Impr.	p	Baseline Prod.	MCTS Prod.	Impr.	p	Baseline Prod.	MCTS Prod.	Impr.	p	Baseline Prod.	MCTS Prod.	Impr.	p
1	1	1	1	1	386.93	422.50	9.19%	*	429.03	460.10	7.24%	*	371.27	446.63	20.30%	*	417.67	443.37	6.15%	*
2	1	1	1	2	255.03	494.97	94.08%	*	311.30	496.23	59.41%	*	248.10	484.10	95.12%	*	95.10	488.50	413.67%	*
3	1	1	2	1	274.07	334.27	21.97%	*	339.53	328.97	-3.11%		266.67	324.90	21.84%	*	298.77	337.67	13.02%	*
4	1	1	2	2	183.93	360.10	95.78%	*	220.53	364.80	65.42%	*	181.20	356.90	96.96%	*	70.03	363.77	419.42%	*
5	1	2	1	1	690.87	702.03	1.62%	*	715.97	694.70	-2.97%		700.97	710.57	1.37%	*	693.70	699.93	0.90%	
6	1	2	1	2	601.67	744.90	23.81%	*	697.97	756.47	8.38%	*	572.40	748.77	30.81%	*	657.97	746.17	13.40%	*
7	1	2	2	1	647.03	677.27	4.67%	*	682.53	676.67	-0.86%		634.03	656.93	3.61%	*	672.73	679.13	0.95%	
8	1	2	2	2	440.50	635.13	44.18%	*	605.90	632.43	4.38%	*	427.00	631.10	47.80%	*	513.13	629.20	22.62%	*
9	1	3	1	1	749.60	752.97	0.45%	*	759.50	758.37	-0.15%		752.10	754.80	0.36%		754.03	756.00	0.26%	
10	1	3	1	2	721.07	784.87	8.85%	*	735.23	795.13	8.15%	*	700.17	790.60	12.92%	*	757.07	807.20	6.62%	*
11	1	3	2	1	728.77	742.73	1.92%	*	744.13	738.57	-0.75%		741.00	742.63	0.22%		730.70	749.33	2.55%	*
12	1	3	2	2	645.70	735.10	13.85%	*	697.30	743.00	6.55%	*	586.77	730.03	24.42%	*	691.00	733.37	6.13%	*
13	2	1	1	1	162.10	166.50	2.71%		173.87	174.63	0.44%		160.70	183.70	14.31%	*	104.90	170.70	62.73%	*
14	2	1	1	2	114.33	202.50	77.11%	*	111.10	206.93	86.26%	*	120.40	208.03	72.79%	*	5.47	209.40	3730.49%	*
15	2	1	2	1	121.17	121.17	0.00%		129.10	122.33	-5.24%		121.67	119.50	-1.78%		58.47	123.23	110.78%	*
16	2	1	2	2	82.53	154.23	86.87%	*	87.67	146.73	67.38%	*	84.63	151.60	79.13%	*	5.93	136.23	2196.07%	*
17	2	2	1	1	313.90	344.80	9.84%	*	378.43	359.57	-4.99%		311.20	309.53	-0.54%		349.40	358.13	2.50%	
18	2	2	1	2	262.57	378.00	43.96%	*	322.63	390.90	21.16%	*	256.77	386.60	50.56%	*	104.53	393.03	275.99%	*
19	2	2	2	1	243.63	230.77	-5.28%		298.57	263.13	-11.87%		232.90	238.13	2.25%		267.63	254.33	-4.97%	
20	2	2	2	2	188.47	296.47	57.30%	*	225.07	294.73	30.95%	*	184.20	291.83	58.43%	*	42.80	297.37	594.78%	*
21	2	3	1	1	482.73	492.93	2.11%	*	485.57	494.50	1.84%	*	482.03	494.13	2.51%	*	485.57	496.47	2.24%	*
22	2	3	1	2	409.53	496.83	21.32%	*	446.47	503.97	12.88%	*	391.60	493.03	25.90%	*	370.37	496.20	33.98%	*
23	2	3	2	1	448.43	470.10	4.83%	*	469.60	464.47	-1.09%		442.73	468.40	5.80%	*	465.50	464.70	-0.17%	
24	2	3	2	2	279.93	394.57	40.95%	*	405.23	407.87	0.65%		285.07	401.40	40.81%	*	206.03	396.67	92.53%	*

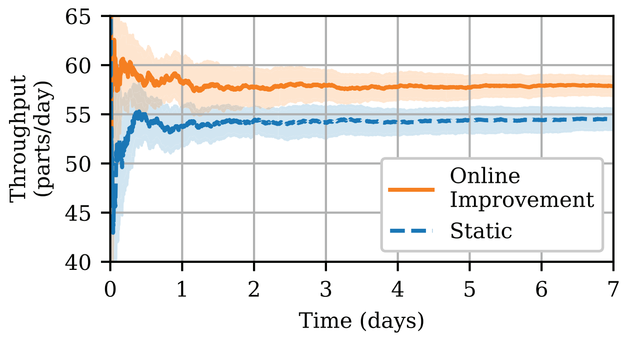


Fig. 7. Average throughput comparison of FIFO and MCTS scheduling over one week of production. The shaded region shows a 95% confidence interval for each average throughput.

observes machines M_1, M_4, M_{11}, and M_{14} with respective health states 6, 10, 10, and 4 in the maintenance queue. Machines M_8 and M_{12} are currently under repair, while the remaining machines are in use. We must allocate the available capacity to conducting a repair on a machine in the queue.

According to the default Birnbaum priority scheduling policy as outlined in Table II, machine M_1 should be repaired as it has the highest Birnbaum importance in this system. Instead, we apply the online improvement procedure described in Section IV-B to attempt to find a better maintenance action.

In addition to the health states of each machine, we first gather the necessary additional information about the current system state, including the remaining cycle time of each machine and the current level of each buffer. In the current state, we also indicate that machines M_8 and M_{12} are undergoing corrective repair. This information forms the initial state from which we conduct MCTS to seek the optimal action.

With the initial state specified in the simulator, we run the MCTS for 1000 iterations. The resulting best action is to repair machine M_{14}, which is awaiting preventive maintenance. Maintaining M_{14} is identified as the best action because it is estimated to maximize the reward function stated in (1), indicating that this action provides the greatest expected discounted production volume in the current state. The action is carried out in the real-world system and it continues to operate until another maintenance scheduling conflict occurs.

C. System Configuration Experiments

In this section, we examine the performance of the online maintenance policy improvement on a variety of systems under different parameters. The system configuration factors and the corresponding levels are given in Table III. A full-factorial design of these four factors results in 24 system configurations. For each configuration, we find the optimal static CBM policy for each of the baseline scheduling heuristics described in Section IV-C using the GA, as described in Section IV.

We then simulate each system under its optimal policy using each baseline heuristic for maintenance conflict resolution and again using our online policy improvement via MCTS for comparison. We use a one-week warm-up period and a one-week evaluation horizon to measure the production observed under each method. The results are summarized in Table IV.

In Table IV, for each static scheduling heuristic, we give the baseline production volume and the production obtained under MCTS scheduling as well as the relative improvement MCTS provides. An asterisk in the p column signifies a p-value less than 0.05 for the one-sided two-sample t-test for equal mean production volume, indicating that MCTS provides a significant improvement over the corresponding static heuristic. In 90 of the 96 cases studied, MCTS either improves or preserves the performance of the static policy.

The throughput improvement obtained from MCTS scheduling strongly depends on the system configuration parameters and the resulting behavior of the system. For example, if maintenance conflicts occur very rarely in a system (such as in the case of a high maintenance capacity or machines with low degradation rate), then there will be few opportunities to apply MCTS to make scheduling decisions. This will result in approximately equal performance to any static scheduling heuristic. On the other hand, if machines degrade more quickly and maintenance resources are scarce, we expect to see more maintenance conflicts and instances of online scheduling via MCTS. Among the system configurations studied in this work, MCTS scheduling provided a 10% or greater increase in throughput only in cases where maintenance resource utilization was at least 90%.

Under certain system configurations, it may also be possible to derive an optimal or near-optimal heuristic for maintenance scheduling by inspection of the system. Consider, for example, a system with n machines arranged in parallel where the cycle time of machine M_i is i minutes. A simple and effective heuristic would be to repair the machine with the shortest cycle time first since it offers the greatest contribution to the overall system throughput. In this scenario, it is unlikely that MCTS scheduling would result in a significantly higher productivity over this static heuristic.

Based on these results, we expect our proposed method of online static policy improvement to offer the greatest throughput increase for systems with a high rate of maintenance resource utilization as well as a sufficiently complex configuration for which an effective static heuristic cannot easily be derived.

VI. Conclusion

We have demonstrated the performance of GA in static CBM policy optimization and MCTS for improving the policy online. MCTS is effective, especially when the maintenance resource utilization is high. It overcomes the limitations of simple scheduling heuristics by looking ahead to the future to evaluate the outcome of sequential maintenance decisions. Although MCTS has performed well in the examples shown in this work, for very large problems, it may require significant run time to converge to a good solution. This can be burdensome in instances where there is very little time to make maintenance decisions, such as the sudden occurrence of a critical failure.

Future work includes examining the effectiveness of MCTS over heuristic scheduling for arbitrary manufacturing systems. For example, if a system consists of very reliable machines that require infrequent maintenance, then it may be rare for scheduling conflicts to occur. A simple priority policy, such as FIFO, would be effective if the average maintenance utilization level is very low as seen from our experiments. Under these conditions, the effort required to implement an MCTS scheduling policy may not be worthwhile. We identified several such cases for a system of particular arrangement under various parameter settings, but doing the same for generalized system structure is also of interest. Furthermore, the application of MCTS to the maintenance scheduling problem can be improved by learning from the results of each search to influence future decisions. If the MCTS problem is formulated and solved for a particular system state, the information can likely be used again if a scheduling conflict arises in the future, while the system is in a similar state.

Acknowledgment

Contribution from Penn State was supported in part by Allen E. Pearce and Allen M. Pearce endowment.

References

[1] A. Berrichi, L. Amodeo, F. Yalaoui, E. Châtelet, and M. Mezghiche, “Bi-objective optimization algorithms for joint production and maintenance scheduling: Application to the parallel machine problem,” *J. Intell. Manuf.*, vol. 20, no. 4, p. 389, 2009.

[2] U.S. Census Bureau. (2016). *2016 Annual Survey of Manufactures*. [Online]. Available: https://www.census.gov/programs-surveys/asm.html

[3] *Railway Applications—Heating, Ventilation and Air Conditioning Systems for Rolling Stock—Part 1: Terms and Definitions*, Standard ISO 19659-1:2017, International Organization for Standardization, Aug. 2017.

[4] T. Wireman, “Maintenance organizations,” in *Benchmarking Best Practices in Maintenance Management*. New York, NY, USA: Industrial Press Inc., 2004, ch. 3, pp. 55–84.

[5] K. Khazraei and J. Deuse, “A strategic standpoint on maintenance taxonomy,” *J. Facilities Manage.*, vol. 9, no. 2, pp. 96–113, May 2011.

[6] S. Albers, “Online scheduling,” in *Introduction to Scheduling*, Y. Robert and F. Vivien, Eds. Boca Raton, FL, USA: CRC Press, 2009, ch. 3.

[7] A. H. Elwany, N. Z. Gebraeel, and L. M. Maillart, “Structured replacement policies for components with complex degradation processes and dedicated sensors,” *Oper. Res.*, vol. 59, no. 3, pp. 684–695, Jun. 2011.

[8] N. Chen, Z.-S. Ye, Y. Xiang, and L. Zhang, “Condition-based maintenance using the inverse Gaussian degradation model,” *Eur. J. Oper. Res.*, vol. 243, no. 1, pp. 190–199, May 2015.

[9] X. Jin *et al.*, “The present status and future growth of maintenance in US manufacturing: Results from a pilot survey,” *Manuf. Rev.*, vol. 3, p. 10, Jan. 2016.

[10] X. S. Si, W. Wang, C. H. Hu, and D. H. Zhou, “Remaining useful life estimation—A review on the statistical data driven approaches,” *Eur. J. Oper. Res.*, vol. 213, no. 1, pp. 1–14, 2011.

[11] L. Yang, X. Ma, and Y. Zhao, “A condition-based maintenance model for a three-state system subject to degradation and environmental shocks,” *Comput. Ind. Eng.*, vol. 105, pp. 210–222, Mar. 2017.

[12] N. Rasmekomen and A. K. Parlikad, “Condition-based maintenance of multi-component systems with degradation state-rate interactions,” *Rel. Eng. Syst. Saf.*, vol. 148, pp. 1–10, Apr. 2016.

[13] M. Marseguerra, E. Zio, and L. Podofillini, “Condition-based maintenance optimization by means of genetic algorithms and Monte Carlo simulation,” *Rel. Eng. Syst. Saf.*, vol. 77, no. 2, pp. 151–165, Aug. 2002.

[14] Y. Zhou, T. R. Lin, Y. Sun, and L. Ma, “Maintenance optimisation of a parallel-series system with stochastic and economic dependence under limited maintenance capacity,” *Rel. Eng. Syst. Saf.*, vol. 155, pp. 137–146, Nov. 2016.

[15] C. D. Dao and M. J. Zuo, “Optimal selective maintenance for multi-state systems in variable loading conditions,” *Rel. Eng. Syst. Saf.*, vol. 166, pp. 171–180, Oct. 2017.

[16] C. D. Dao and M. J. Zuo, “Selective maintenance of multi-state systems with structural dependence,” *Rel. Eng. Syst. Saf.*, vol. 159, pp. 184–195, Mar. 2017.

[17] Y. Zhou, Y. Guo, T. R. Lin, and L. Ma, “Maintenance optimisation of a series production system with intermediate buffers using a multi-agent FMDP,” *Rel. Eng. Syst. Saf.*, vol. 180, pp. 39–48, Dec. 2018.

[18] X. Gu, S. Lee, X. Liang, M. Garcellano, M. Diederichs, and J. Ni, “Hidden maintenance opportunities in discrete and complex production lines,” *Expert Syst. Appl.*, vol. 40, no. 11, pp. 4353–4361, Sep. 2013.

[19] D. I. Cho and M. Parlar, “A survey of maintenance models for multi-unit systems,” *Eur. J. Oper. Res.*, vol. 51, no. 1, pp. 1–23, Mar. 1991.

[20] M. C. A. O. Keizer, S. D. P. Flapper, and R. H. Teunter, “Condition-based maintenance policies for systems with multiple dependent components: A review,” *Eur. J. Oper. Res.*, vol. 261, no. 2, pp. 405–420, Sep. 2017.

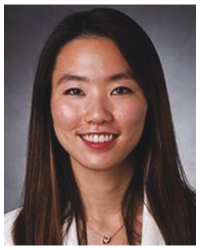
[21] C. E. Ebeling, *An Introduction to Reliability and Maintainability Engineering*. New York, NY, USA: McGraw-Hill, 2004.

[22] A. Gupta and C. Lawsirirat, “Strategically optimum maintenance of monitoring-enabled multi-component systems using continuous-time jump deterioration models,” *J. Qual. Maintenance Eng.*, vol. 12, no. 3, pp. 306–329, Jul. 2006.

[23] K.-A. Nguyen, P. Do, and A. Grall, "Condition-based maintenance for multi-component systems using importance measure and predictive information," *Int. J. Syst. Science, Oper. Logistics*, vol. 1, no. 4, pp. 228–245, Oct. 2014.
[24] K.-A. Nguyen, P. Do, and A. Grall, "Multi-level predictive maintenance for multi-component systems," *Rel. Eng. Syst. Saf.*, vol. 144, pp. 83–94, Dec. 2015.
[25] Y. Liu and H.-Z. Huang, "Optimal replacement policy for multi-state system under imperfect maintenance," *IEEE Trans. Rel.*, vol. 59, no. 3, pp. 483–495, Sep. 2010.
[26] W. Rice, C. Cassady, and J. Nachlas, "Optimal maintenance plans under limited maintenance time," in *Proc. 7th Ind. Eng. Res. Conf.*, 1998, pp. 1–3.
[27] W. Cao, X. Jia, Q. Hu, J. Zhao, and Y. Wu, "A literature review on selective maintenance for multi-unit systems," *Qual. Rel. Eng. Int.*, vol. 34, no. 5, pp. 824–845, Jul. 2018.
[28] K. S. de Smidt-Destombes, M. C. van der Heijden, and A. van Harten, "On the availability of a k-out-of-N system given limited spares and repair capacity under a condition based maintenance strategy," *Rel. Eng. Syst. Saf.*, vol. 83, no. 3, pp. 287–300, Mar. 2004.
[29] R. Moghaddass, M. J. Zuo, and M. Pandey, "Optimal design and maintenance of a repairable multi-state system with standby components," *J. Stat. Planning Inference*, vol. 142, no. 8, pp. 2409–2420, Aug. 2012.
[30] Z. Yang, Q. Chang, D. Djurdjanovic, J. Ni, and J. Lee, "Maintenance priority assignment utilizing on-line production information," *J. Manuf. Sci. Eng.*, vol. 129, no. 2, pp. 435–446, Apr. 2007.
[31] M. Hoffman, E. Song, M. Brundage, and S. Kumara, "Condition-based maintenance policy optimization using genetic algorithms and Gaussian Markov improvement algorithm," in *Proc. PHM Soc. Conf.*, vol. 10, no. 1, 2018, pp. 1–9.
[32] J. Li and S. M. Meerkov, *Production Systems Engineering*. New York, NY, USA: Springer, 2008.
[33] A. K. S. Jardine, D. Lin, and D. Banjevic, "A review on machinery diagnostics and prognostics implementing condition-based maintenance," *Mech. Syst. Signal Process.*, vol. 20, no. 7, pp. 1483–1510, Oct. 2006.
[34] K. Liu, A. Chehade, and C. Song, "Optimize the signal quality of the composite health index via data fusion for degradation modeling and prognostic analysis," *IEEE Trans. Autom. Sci. Eng.*, vol. 14, no. 3, pp. 1504–1514, Jul. 2017.
[35] H.-S. Baik, H. S. Jeong, and D. M. Abraham, "Estimating transition probabilities in Markov chain-based deterioration models for management of wastewater systems," *J. Water Resour. Planning Manage.*, vol. 132, no. 1, pp. 15–24, Jan. 2006.
[36] Y. Tsuda, K. Kaito, K. Aoki, and K. Kobayashi, "Estimating Markovian transition probabilities for bridge deterioration forecasting," *Struct. Eng./Earthquake Eng.*, vol. 23, no. 2, pp. 241s–256s, 2006.
[37] A. Alrabghi and A. Tiwari, "State of the art in simulation-based optimisation for maintenance systems," *Comput. Ind. Eng.*, vol. 82, pp. 167–182, Apr. 2015.
[38] K. A. H. Kobbacy, *Artificial Intelligence in Maintenance*. London, U.K.: Springer, 2008, pp. 209–231.
[39] S. J. Russell and P. Norvig, "Reinforcement learning," in *Artificial Intelligence: A Modern Approach*, 3rd ed. Upper Saddle River, NJ, USA: Prentice-Hall, 2010, ch. 20.
[40] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Trans. Evol. Comput.*, vol. 6, no. 2, pp. 182–197, Apr. 2002.
[41] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: MIT Press, 2018.
[42] B. S. D. Ugarte, A. Artiba, and R. Pellerin, "Manufacturing execution system—A literature review," *Prod. Planning Control*, vol. 20, no. 6, pp. 525–539, Sep. 2009.
[43] G. Chaslot, S. Bakkes, I. Szita, and P. Spronck, "Monte-Carlo tree search: A new framework for game Ai," in *Proc. AIIDE*, 2008, pp. 1–2.
[44] H. S. Chang, M. C. Fu, J. Hu, and S. I. Marcus, "An adaptive sampling algorithm for solving Markov decision processes," *Oper. Res.*, vol. 53, no. 1, pp. 126–139, Feb. 2005.
[45] L. Kocsis, C. Szepesvári, and J. Willemson, "Improved Monte-Carlo search," Univ. Tartu, Tartu, Estonia, Tech. Rep. 1, 2006, vol. 1.
[46] Z. W. Birnbaum, "On the importance of different components in a multicomponent system," Washington Univ. Seattle Lab Stat. Res., Seattle, WA, USA, Tech. Rep. 54, 1968.
[47] S. Si, M. Liu, Z. Jiang, T. Jin, and Z. Cai, "System reliability allocation and optimization based on generalized Birnbaum importance measure," *IEEE Trans. Rel.*, vol. 68, no. 3, pp. 831–843, Sep. 2019.

Michael Hoffman received the Ph.D. degree in industrial engineering and operations research from The Pennsylvania State University, State College, PA, USA, in 2021.

He is currently a Graduate Measurement Science and Engineering Fellow of the National Institute of Standards and Technology (NIST), Gaithersburg, MD, USA, where he is a member of the Knowledge Extraction and Application for Manufacturing Operations Project. Previously, he was a Walker Graduate Assistant with the Materials and Manufacturing Division, Applied Research Laboratory, The Pennsylvania State University. His research interests include intelligent manufacturing systems and simulation for industrial maintenance decision support.



Eunhye Song received the Ph.D. degree in industrial engineering and management sciences from Northwestern University, Evanston, IL, USA, in 2017.

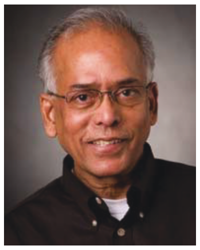
She is currently the Harold and Inge Marcus Early Career Assistant Professor with the Department of Industrial and Manufacturing Engineering and an Associate of the Institute for Computational and Data Sciences, The Pennsylvania State University, State College, PA, USA. Her research interests include simulation design of experiments, uncertainty and risk quantification, and simulation optimization.

Dr. Song is an Active Member of the INFORMS Simulation Society and has served on the Society's Underrepresented Minorities and Women Committee. She was a recipient of the National Science Foundation CAREER Award and was a Finalist of the 2020 INFORMS Junior Faculty Interest Group Paper Competition.

Michael P. Brundage received the Ph.D. degree in mechanical engineering from Stony Brook University, Stony Brook, NY, USA, in 2015.

He is currently an Industrial Engineer with the Informational Modeling and Testing Group at the National Institute of Standards and Technology (NIST), Gaithersburg, MD, USA. He serves as the Project Leader for the Knowledge Extraction and Application for Manufacturing Operations Project in the Model-Based Enterprise Program. His work contributes to guidelines for intelligent maintenance. He is part of a task group for creating an ASME Prognostics Health Management (PHM) Standards Committee. He worked closely with ASTM International E60.13 in the development of a guideline for sustainable manufacturing performance indicators (ASTM E3096-17). He has authored over 20 peer-reviewed publications. His research interests include smart manufacturing diagnostics for intelligent maintenance, sustainable manufacturing performance measurement, smart manufacturing capability assessment, and manufacturing knowledge visualization.

Dr. Brundage has chaired multiple ASME MSEC Symposia and industry forums/workshops at NIST.



Soundar Kumara is currently the Allen, E., and Allen, M., Pearce Professor of Industrial Engineering at Pennsylvania State University, University Park, PA, USA. He is also with the School of Information Sciences and Technology, Pennsylvania State University. His research interests are in smart manufacturing, large-scale networks, sensing and control, the Industrial Internet of Things (IIoT), and machine learning in manufacturing and healthcare.

Prof. Kumara is a fellow of the Institute of Industrial Engineers (IIE), the International Academy of Production Engineering (CIRP), the American Association for Advancement of Science (AAAS), and the American Association of Mechanical Engineers (ASME). He has guided 60 Ph.D. and 75 M.S. students.