Condition-based Maintenance Policy Optimization Using Genetic Algorithms and Gaussian Markov Improvement Algorithm

Michael Hoffman[1], Eunhye Song[2], Michael Brundage[3], and Soundar Kumara[4]

[1,2,4] *The Pennsylvania State University, University Park, Pennsylvania, 16801, USA*
MichaelHoffman@psu.edu
eus358@psu.edu
u1o@engr.psu.edu

[3] *National Institute of Standards and Technology, Gaithersburg, Maryland, 20899, USA*
michael.brundage@nist.gov

Abstract

Condition-based maintenance involves monitoring the degrading health of machines in a manufacturing system and scheduling maintenance to avoid costly unplanned failures. As compared with preventive maintenance, which maintains machines on a set schedule based on time or run time of a machine, condition-based maintenance attempts to minimize the number of times maintenance is performed on a machine while still attaining a prescribed level of availability. Condition-based methods save on maintenance costs and reduce unwanted downtime over its lifetime. Finding an analytically-optimal condition-based maintenance policy is difficult when the target system has non-uniform machines, stochastic maintenance time and capacity constraints on maintenance resources. In this work, we find an optimal condition-based maintenance policy for a serial manufacturing line using a genetic algorithm and the Gaussian Markov Improvement Algorithm, an optimization via simulation method for a stochastic problem with a discrete solution space. The effectiveness of these two algorithms will be compared. When a maintenance job (i.e., machine) is scheduled, it is placed in a queue that is serviced with either a first-in-first-out discipline or based on a priority. In the latter, we apply the concept of opportunistic window to identify a machine that has the largest potential to disrupt the production of the system and assign a high priority to the machine. A test case is presented to demonstrate this method and its improvement over traditional maintenance methods.

1. Introduction

The importance of maintenance in manufacturing is often overlooked as it is considered a non-value added activity in the manufacturing process. However, it is critical for supporting the availability and productivity of machines in the system. A maintenance policy defines how decisions are made regarding when and where to perform maintenance. In this work, we focus on the development of a condition-based maintenance policy for continuously monitored deteriorating machines. Since it is assumed that the health of each machine is known at all times, we can use this information to decide when maintenance should be performed.

In the work presented here, we consider non-uniform machines, a capacity for maintenance resources (a maximum number of concurrent maintenance jobs), and non-instant repair times. Much of the previous related work makes simplifying assumptions and does not consider all of these factors in combination. A capacity for maintenance resources and non-instant repairs results in frequent occurrences of conflicting maintenance jobs. This makes the development of a maintenance policy more difficult because, in addition to deciding when to repair each machine, we must decide how to reconcile maintenance scheduling conflicts. The objective when optimizing the maintenance policy is minimizing the cost of maintenance activities over some time horizon.

The rest of the paper is organized as follows: Section 2 describes some of the previous work related to maintenance policy optimization. Section 3 presents the system being considered. This includes the notation used, the machine deterioration model, the maintenance job queue, and the cost measurement model. The algorithms used to optimize the maintenance policy are described in Section 4. In Section 5 the results from an example are shown. Lastly, conclusions and future work are given in Section 6.

2. BACKGROUND

Maintenance in manufacturing plays a pivotal role in ensuring efficiency in production. Maintenance includes all activities related to maintaining a specific level of availability of the system and components to perform at a certain level of quality and productivity (Al-Turki et al., 2014). One important aspect of maintenance is scheduling maintenance resources to ensure machine availability without sacrificing production throughput and quality. Scheduling maintenance resources requires determining when to send a technician, to what machine should the technician be sent, how often to schedule maintenance, and in what order, among other factors. For example, if two machines break down and a third machine is scheduled for preventive maintenance, which machine do you maintain first to avoid throughput loss and minimize cost?

Multiple maintenance scheduling policies exist that address this type of question. Jin et al. (2016) found that the majority of manufacturers still employ a mixture of reactive and preventive maintenance strategies. Reactive maintenance involves performing maintenance after an unanticipated failure of a piece of equipment. Preventive maintenance strategies involve performing maintenance after a set amount of time or after a piece of equipment is run for a certain number of cycles. These strategies are employed due to their low initial cost and low data requirements, however, they can lead to large productivity losses, equipment downtime, and can lead to over maintaining if the preventive maintenance is performed too frequently. To avoid over maintaining, condition-based maintenance (CBM) analyzes the current state of the machine and sets a predetermined level of health to maintain the machine (Jin et al., 2016). CBM requires large training data sets to build models for prediction, which requires considerable upfront cost and knowledge.

While the above strategies determine the frequency of maintenance, they do not address the priority rules for the order in which to perform the maintenance. This priority determination is especially challenging when maintenance resources are limited and several maintenance jobs are scheduled concurrently. The decision must be made as to which jobs should be carried out first. Opportunistic maintenance addresses this issue by analyzing trade-offs between production and maintenance to reduce production losses (Zhou, Xi, & Lee, 2009; Chang, Ni, Bandyopadhyay, Biller, & Xiao, 2007). This paper describes a CBM schedule with opportunistic priority rules minimizing both cost and production loss.

CBM optimization policies can be classified in several ways. Such classifications are based on maintenance policy parameters, system configuration, deterioration model, maintenance resource configuration, and optimization objectives (Khazraei & Deuse, 2011). This section provides examples of previous work in each classification scheme.

Two main categories of decision variables are considered when defining a CBM policy. One is the interval between inspections of components in the system. When continuous monitoring is not available, the health of components can only be known by performing an inspection that typically incurs some fixed cost. Upon inspection, the decision must be made as to whether or not maintenance should be performed. Such models are thoroughly described by Kallen & van Noortwijk (2006). The alternative, and the approach that is used here, is a continuously monitored system where the health of components is known at each discrete time step. In this case, the decision is related to at what health level should maintenance be scheduled.

The configuration of the system of interest will influence the optimal policy as well. Yang, Ma, & Zhao (2017) consider CBM of a single-unit system with multiple failure modes and determine the optimal policy given the state of the component. Maintenance policy optimization for multiple machines in series is studied by Bartholomew-Biggs, Zuo, & Li (2009). A series-parallel system is considered by Marseguerra, Zio, & Podofillini (2002), in which serial subsystems are comprised of identical machines in parallel. A policy is developed for a single machine type and then used for all machines within a subsystem.

A discrete-time Markov model is often used to represent the health of deteriorating machines. Each machine is assumed to be at a discrete level of health, known as the health index, at any point in time. The machine then transitions to another health index at the next time step with some probability. One variation of this model is the addition of random shocks, where a machine may transition into a complete failure state at any time, as examined by Yang et al. (2017). Some work has considered dependence among multiple machines, which can have a significant impact on the optimal maintenance policy. Rasmekomen & Parlikad (2016) develop a CBM policy for a system of components with stochastic dependence in which the degradation rate of a machine depends on that of others in the system.

While the minimization of cost is a typical objective in the design of maintenance policies, other competing objectives are also considered. In addition to minimizing cost, Marseguerra et al. (2002) aim to maximize the availability of the system. Lei, Liu, Ni, & Lee (2010) attempt also to maximize the throughput of the system by scheduling maintenance so that downtime does not hinder production. Many of these objectives can be combined into a single cost objective. For example, production that is lost due to downtime for maintenance can be assigned a cost that worsens the objective function value. By including such measures in the cost objective function, only one objective needs be considered.

3. SYSTEM DESCRIPTION

In this section, we define the notation that is used throughout the remainder of the paper and the underlying assumptions of the system of interest. The system considered is a serial production line with M machines each with buffer size B and deterministic processing time t_m as depicted in Figure 1. Each machine will produce at its maximum rate while it is functional, so long as it is not starved or blocked. The first machine in the series is never starved and the last machine is never blocked.

3.1. Notation

- T - observed time horizon
- M - number of machines in series
- B - buffer capacity
- $b_m(t)$ - input buffer level of machine m at time t
- t_m - process time of a single part for machine m
- t_{m^*} - bottleneck process time (process time of bottleneck machine m^*)
- q_m - degradation rate of machine m
- $H_m(t)$ - health index of machine m at time t. $H_m(t) = 0$ indicates a machine is in perfect health, and the health index increases over time as the machine health deteriorates.
- h_m - health index threshold at which condition-based maintenance is scheduled for machine m
- $H_m(t) = h_{\max}$ - the health index at which a machine experiences total failure
- $\mathbf{P} = \{h_1, h_2, \cdots, h_M\}$ - CBM maintenance policy of the system
- r - maintenance capacity
- C_P - cost of a planned maintenance job
- C_U - cost of an unplanned maintenance job
- C_{LP} - cost per unit of lost production
- x_P - the number of planned maintenance jobs performed over the time horizon
- x_U - the number of unplanned maintenance jobs performed over the time horizon
- u - the actual number of units produced over the time horizon
- C_T - total policy cost

3.2. Deterioration Model

As described in Section 2, many condition-based maintenance processes assume that component deterioration can be modeled as a discrete Markov process. In this work, we assume that a machine m is in perfect working condition when its health index $H_m(t) = 0$ and that it degrades at each time step with a known probability. As the machine degrades, its health index increases until it is repaired or experiences a complete failure. The degradation rate of a machine can depend on many factors including age of the machine, stress on the machine, utilization, or the degradation state of other components in the system (Nicolai & Dekker, 2008). When a random failure occurs the machine stops functioning completely until it is repaired. When maintenance is completed on a component, whether preventive or in response to a complete failure, its health index is restored to zero and degradation resumes. The transition matrix $\mathbf{Q}_m$ of the degradation process of machine m is:

$$\mathbf{Q}_m = \begin{bmatrix} 1-q_m & q_m & & & \\ & 1-q_m & q_m & & \\ & & \ddots & & \\ & & & 1-q_m & q_m \\ & & & & 1 \end{bmatrix} \tag{1}$$

Note that $\mathbf{Q}_m$ is upper bidiagonal.

Figure 2 shows the health index of a single machine over time. At time $t = 3$, the health index reaches the threshold for maintenance and is repaired at time $t = 4$. The machine is then restored to perfect health and degradation resumes. At time $t = 10$, the machine incurs a complete failure.

3.3. Maintenance Queue

Since we impose maintenance capacity on the system considered in this work, multiple maintenance jobs will simultaneously compete for limited maintenance resources. To handle these situations, we queue arriving maintenance jobs. When a maintenance job is placed in the queue (i.e. when the health index of a machine exceeds the threshold for CBM), it is serviced if there are sufficient maintenance resources available. Otherwise, it must wait for some time to be serviced. While waiting in the queue for maintenance, machines continue to degrade until their health index reaches $h_{\max}$.

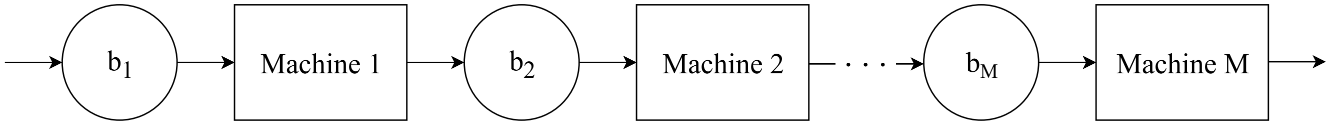

Figure 1. Series production system

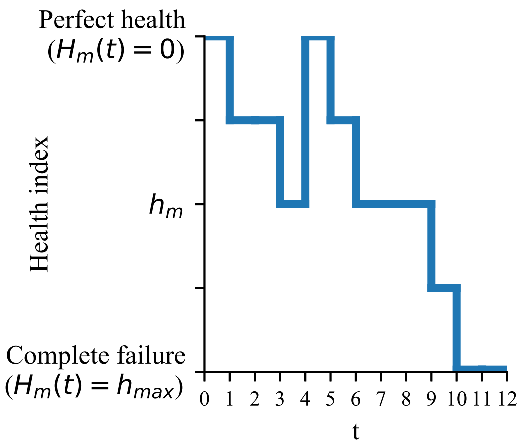

Figure 2. Degradation over time of a single machine, a repair is completed at time $t = 4$, and a complete failure occurs at time $t = 10$

We consider two queueing disciplines: first in, first out (FIFO) and priority queues. Under the FIFO rule, maintenance jobs are serviced in the order that they arrive in the queue by available maintenance resources. While this policy is simple to implement, it ignores the fact that high-risk machines with a greater potential to disrupt system throughput (e.g., the bottleneck machine) will be ignored if they are not in the front of the queue.

An alternative approach is to assign each maintenance job a priority measure and always service the job in the queue with the highest priority. To minimize lost production due to machine down time, maintenance jobs are assigned a priority that is related to the size of each machine's *maintenance opportunity window*. This concept is explained further in the following section.

3.3.1. Maintenance Opportunity Window

The maintenance opportunity window is the length of time a machine can stop production without hindering the overall system throughput. Throughput loss is avoided through the use of buffers in the system and by making sure the bottleneck machine is not blocked or starved. If a machine m in a serial line is upstream from the bottleneck machine m^* ($m < m^*$), the opportunity window for machine m is the time it takes for all buffers between m and m^* to become empty. At this point, the bottleneck machine is starved and throughput is hindered. If m is downstream from m^* ($m > m^*$), the opportunity window for m is the time for all buffers between m^* and m to become full. The bottleneck machine will then be blocked. This concept is described thoroughly by (Chang, Xiao, Biller, & Li, 2013) and summarized by Eq. (2):

$$W_m(t) = \begin{cases} t_{m^*} \sum_{k=m+1}^{m^*} b_k(t), & m < m^* \\ 0, & m = m^* \\ t_{m^*} \sum_{k=m^*+1}^{m} (B - b_k(t)), & m > m^*, \end{cases} \quad (2)$$

where $W_m(t)$ is the duration of the opportunity window for machine m at time t. This equation assumes that machine m is the only machine that is broken down over the duration of the opportunity window; however, work has shown how this equation also provides a rough estimate of opportunity window of each machine with simultaneous failures (Brundage, Chang, Li, Arinez, & Xiao, 2016). Future work will further refine the opportunity window equation for this purpose.

Machines with the smallest maintenance opportunity window will be assigned the highest priority. By minimizing the downtime of high-risk machines, we can reduce the throughput impact of performing maintenance. A comparison of the performance of the FIFO and priority queue policies is described in Section 5.

3.4. Cost Model

In general, minimizing the cost of the maintenance policy as given by Eq. (3) will be the primary objective. The cost of a policy over some time horizon consists of three components: planned maintenance activities ($C_P x_P$), unplanned maintenance activities ($C_U x_U$), and lost production due to downtime in the system (C_{LP}). The cost of a planned maintenance job is incurred when a machine is repaired before reaching the complete failure state. The total cost of planned maintenance is the product of the number of jobs that occur and the cost of planned maintenance activities. Similarly, the total cost of unplanned maintenance is the number of repairs completed on a machine in a failed state multiplied by the cost of the activity. The number of maintenance events over the observed time horizon is represented by a random variable.

Cost is also measured in terms of lost production due to machine downtime. Lost production is defined as the difference between the production requirement in units over the time horizon and the actual number of units produced by the system. The production requirement can be a fixed number of units, or a fraction of the “ideal production” that is obtained from a perfect system with no downtime events. The maintenance policy cost function is

$$C_T = C_P x_P + C_U x_U + C_{LP}\left(\frac{T}{t_{m^*}} - u\right). \quad (3)$$

Generally, planned maintenance is less costly than unplanned maintenance because the machine avoids a complete failure and downtime that results in system throughput disruption (Chitra, 2003). Unplanned maintenance occurs when a machine's health index reaches the total failure state, $h_{\max}$, and it is forced to stop production until repaired. Unplanned failures can occur when a machine that is waiting for planned maintenance continues to degrade to the point of total failure while waiting for maintenance resources to become available. Since we consider the duration of maintenance activities in the model, maintenance on a machine will disrupt the machine's production and possibly the overall throughput of the system. Lost production is defined as the difference between the production volume of the system if there was no degradation of machines and the actual production volume observed. Each unit of lost production incurs some cost that contributes to the cost of the maintenance policy.

4. METHODOLOGY

The goal of this work is to find an optimal CBM policy for a serial manufacturing system. As described in Section 2, a CBM policy is defined by the health index thresholds at which CBM is scheduled for each machine. Since there are M machines in the system, a solution, or policy, $\mathbf{P}$ can be value encoded by a set of M thresholds $\mathbf{P} = \{h_1, h_2, \cdots, h_M\}$. The minimum value of a threshold is 1, which would indicate maintenance is scheduled on a machine as soon its health index deteriorates by one unit. The maximum value is $h_{\max}$, the health index that indicates a machine has experienced a complete failure. A threshold at its maximum value is equivalent to a corrective maintenance policy.

For the problem presented here, the primary objective is to find the maintenance policy that minimizes expected cost per unit time. Due to the stochasticity and complex interactions that occur in the system under consideration, it is difficult to analytically determine the cost of the policy as a function of a set of CBM thresholds for each machine. Analytically determining cost often requires simplifying assumptions that reduce the accuracy of the cost measurement (Alrabghi & Tiwari, 2015). For this reason simulation will be used to evaluate the solutions by estimating the expected cost of a policy. We will compare the effectiveness of a genetic algorithm and the Gaussian Markov Improvement Algorithm in finding a solution to this problem In Section 4.3, details of the simulation model used in the experiments are given.

4.1. Genetic Algorithm

Genetic algorithms (GAs) are a metaheuristic method of problem solving that attempt to replicate evolutionary behavior found in nature. A population of solutions (or individuals) evolves over time by selecting the best members of the population to produce the succeeding generation. As in nature, a population benefits from diversity, so the algorithm begins with an initial set of random individuals. Starting with this initial population, offspring solutions are generated and added to the population. The best individuals from this group are then chosen to produce the next generation, and the process repeats until some termination criteria is met. There are four main considerations when using a genetic algorithm: the representation or encoding of the solutions within the context of the problem, the fitness function by which candidate solutions will be evaluated, the method of selecting individuals for reproduction, and the method of reproduction used (Deb et al., 2002). For this problem, a policy defined as $\mathbf{P} = \{h_1, h_2, \cdots, h_M\}$ represents an individual in the genetic algorithm.

The problem of optimizing a CBM policy is well-suited for GAs as this approach is robust and effective for large, complex manufacturing systems (Kobbacy, 2008).

4.2. Gaussian Markov Improvement Algorithm

Discrete optimization via simulation (DOvS) refers to finding an optimal solution of a problem with discrete decision variables whose objective function does not have a closed-form expression, but can be evaluated by stochastic simulation. Because of its flexibility, DOvS is a popular method for solving a complex stochastic problem. For the problems with small-to-medium feasible solution spaces where one can afford to simulate all feasible solutions, ranking and selection (R&S) has been successfully applied (Kim & Nelson, 2006); however, when the feasible solution space is large, it is practically impossible and inefficient to simulate all solutions. For the latter category of DOvS problems, several adaptive random search (ARS) algorithms have been developed. In general, an ARS algorithm initially simulates a small number of solutions and iteratively selects the next solution to simulate based on the simulation history. Since these initially sampled solutions are often used to estimate the necessary parameters of the algorithm, they are referred to as initial design points. A good ARS algorithm uses statistical inference based on simulated solutions to choose the next solution to simulate balancing exploration and exploitation.

The Gaussian Markov Improvement Algorithm (GMIA) finds the globally optimal solution of a DOvS problem with probability 1 when the simulation budget increases without a bound (Salemi, Song, Nelson, & Staum, 2018). GMIA is an ARS that draws statistical inference on the performance of feasible solutions by fitting a metamodel of the objective function at all feasible solutions based on the simulated solutions. The particular metamodel GMIA employs is a Gaussian Markov random field (GMRF), which models the unknown objective function values at the solution as Gaussian random variables with positive spatial correlations among solutions. From the simulation results of the initial design points, parameters of the GMRF model are estimated. Then, at each iteration, the distribution of the GMRF is updated conditional on the cumulative simulation results up to that iteration.

GMIA imposes the correlation such that nearby solutions have stronger positive correlation in the objective function

Table 1. Test case 1 description.

Machine	1	2	3
t_m	3	4	5
q_m	0.02	0.06	0.01

values. This works for DOvS as solutions that are close in the feasible solution space often have similar objective function values. Therefore, even if a solution is not simulated yet, we can infer the objective function value at the solution based on simulated solutions and guide the search towards a more promising region of the feasible solution space. We defer the implementation details of GMIA in this paper; see (Salemi et al., 2018).

4.3. Simulation

Simulation is used to evaluate the quality of a maintenance policy solution. The system is simulated in its steady state for some period of time and then the cost of the defined maintenance policy is calculated using Eq. (3) described in Section 3.4. The system is considered to be in its steady state when the production rate of each machine is relatively constant over time. Once the steady state is achieved, the system is observed for the specified time horizon, T.

5. NUMERICAL RESULTS

A three-machine serial production line is used to demonstrate the methodology presented in the previous section. The system will be evaluated under both FIFO and priority queue disciplines. The machines in the system are described in Table 1. Table 2 describes additional parameters of the system.

GA and GMIA can be compared by evaluating the performance of each for a defined simulation budget. The simulation budget will be a maximum number of fitness function evaluations (NFE) that will occur. NFE for GMIA is given by

$$(2 \cdot \text{max iterations} + k) \cdot r, \tag{4}$$

where k is the initial number of design points and r is the number of simulation replications of each sampled solution. The NFE for GA depends on the population size, maximum number of generations, and the number of replications. NFE is given by

$$(2n + 1) \cdot \text{population size} \cdot \text{replications}, \tag{5}$$

where n is the number of generations. The parameters of both algorithms will be defined such that NFE is the same for both.

For the GMIA example shown here, the maximum number of iterations is 200, the number of initial design points is $k = 10$, and the number of simulation replications for each sampled solution is $r = 10$. This results in a total of 4100 fitness function evaluations at the termination of the GMIA. The parameters for the GA are a population size of 20, a maximum of 10 generations, and 10 simulation replications per solution. 4200 fitness function evaluations are used for GA.

For this system, the solution space is small enough that all solutions can be exhausted in order to find the global optimum. At 10,000 replications, the largest standard error observed was 3.23. The algorithms can also be compared to see if they converge to this solution.

5.1. FIFO Maintenance Queue

For this case, the overall best policy is $\mathbf{P} = \{8, 7, 8\}$ which was found to have a cost of 382.07 when simulated for 10,000 replications. Under this policy, machines 1, 2, and 3 are scheduled for repair when their health index reaches 8, 7 and 8, respectively. Figure 3 shows the convergence of each algorithm to the optimal solution for the system under a FIFO maintenance queueing discipline. The objective function value at each stage is the average of ten simulation replications. The cost shown on the vertical axis is the true cost of the best solution, as found by exhausting the solution space. When evaluating a solution, the algorithms are not likely to obtain an estimate that is equal to the true expected cost of the policy due to the high variance in the simulation. This results in the selection of "worse" solutions at some steps of each algorithm. Both algorithms are able to improve the solution over time, but on an average GMIA finds a policy with a lower cost.

In many cases, the true cost of a policy is different than that determined by the GA. As shown in Figure 4, the cost of the best solution at each generation as predicted by the GA (referred to as the observed cost) is much lower than the true cost of that solution. In fact, the observed cost of the best solution at termination is lower than the true minimum cost. This is

Table 2. Parameters for both test cases.

Parameter	Value
Buffer capacity (B)	2
Planned maintenance time to repair	Uniform$(5, 15)$
Planned maintenance cost (C_P)	50
Unplanned maintenance time to repair	Uniform(10, 20)
Unplanned maintenance cost (C_U)	300
Unit lost production cost (C_{LP})	10

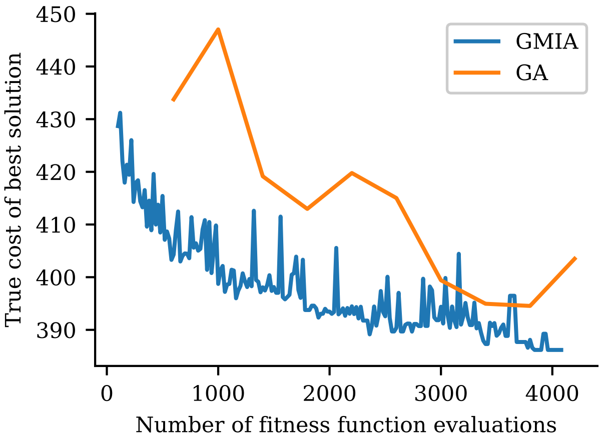

Figure 3. True cost of best solution versus simulation replications for a FIFO maintenance queue averaged over ten runs of each algorithm.

due to the small number of simulation replications that are made when the GA evaluates a solution. The high degree of replication variability makes it difficult to accurately measure the fitness of a solution with only a few replications.

5.2. Priority Maintenance Queue

Similar results can be examined for the system under a priority queue discipline for maintenance jobs. The true minimum cost is obtained for the policy $\mathbf{P} = \{8, 7, 8\}$ which has an average objective function value of 383.21 after 10,000 simulation replications, so the cost is not improved by a priority queue. Both algorithms are again compared using a prescribed maximum number of fitness function evaluations. In Figure 5 the convergence of each algorithm is shown. Again it appears that on average the GMIA obtains a better solution for a given NFE.

Just as in the FIFO maintenance queue case, the GA tends to underestimate the cost of the best solution, as shown in Figure 6. This is again a result of the small number of replications that are used to evaluate the candidate solutions. There is a trade off between the accuracy of solution evaluations and the number of unique solutions evaluated. Conversely, another disadvantage of the GA is that favorable solutions may be overlooked due to the variability in their evaluation. Just as the fitness of some solutions is overestimated, it is likely that fitness is frequently underestimated as well. This could result in better solutions not being selected for reproduction, resulting in a non-optimal population of solutions.

6. CONCLUSIONS

For the problem of optimizing a condition-based maintenance policy for a series manufacturing system, both GA and GMIA have shown to be effective search techniques. For a given

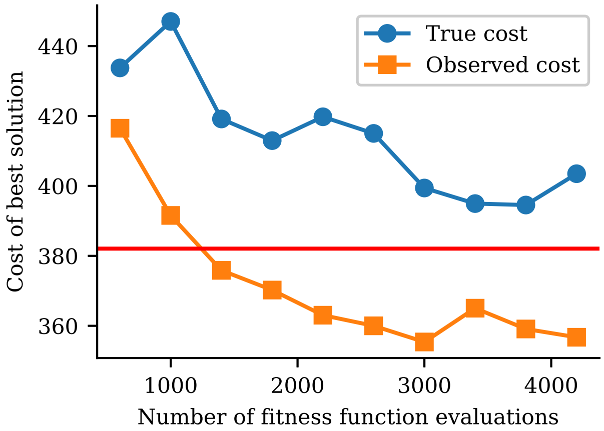

Figure 4. True and observed cost versus simulation replications for GA in a FIFO maintenance queue (horizontal line represents the true minimum cost)

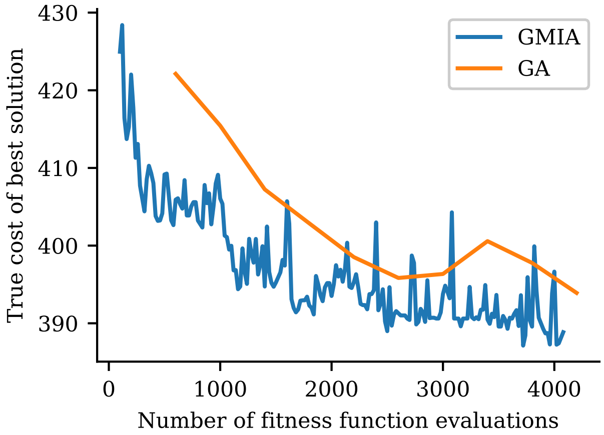

Figure 5. True cost of best solution versus simulation replications for a priority maintenance queue averaged over ten runs of each algorithm.

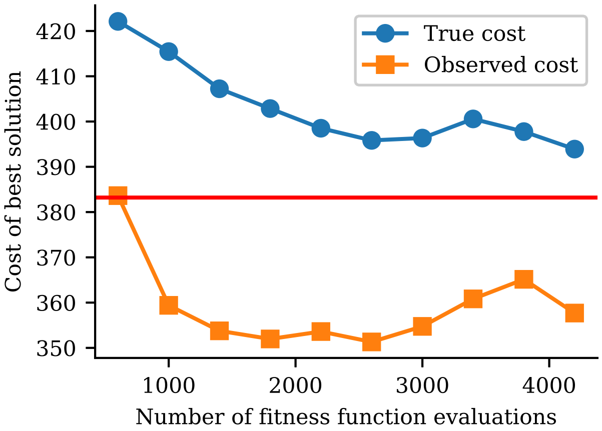

Figure 6. True and observed cost versus simulation replications for GA in a priority maintenance queue (horizontal line represents the true minimum cost)

simulation budget, GMIA is able to find a better optimal solution on average. This is an important consideration as the simulation of complex systems can be time-consuming and computationally expensive.

The maintenance of the example system presented here did not benefit from a priority maintenance job queueing discipline. This could be due to fact that failures of other machines are ignored when finding the opportunity window of a machine. Improving upon the opportunity window priority measure is among the next steps of this work. It may also be the case that there are not many instances of conflicting maintenance jobs, and so there is little need to decide the order in which jobs should be performed. A priority queue would likely be more effective for a system with a greater number of machines or machines with higher rates of degradation. In both cases, more maintenance jobs would occur over a given time horizon, thus increasing the occurrence of scheduling conflicts. This will be examined further in future work.

REFERENCES

Alrabghi, A., & Tiwari, A. (2015). State of the art in simulation-based optimisation for maintenance systems. *Computers Industrial Engineering*, *82*, 167.

Al-Turki, U. M., Ayar, T., Yilbas, B. S., & Sahin, A. Z. (2014). *Integrated maintenance planning in manufacturing systems*. Springer.

Bartholomew-Biggs, M., Zuo, M. J., & Li, X. (2009). Modelling and optimizing sequential imperfect preventive maintenance. *Reliability Engineering & System Safety*, *94*(1), 53–62.

Brundage, M. P., Chang, Q., Li, Y., Arinez, J., & Xiao, G. (2016). Implementing a real-time, energy-efficient control methodology to maximize manufacturing profits. *IEEE transactions on systems, man, and cybernetics: systems*, *46*(6), 855–866.

Chang, Q., Ni, J., Bandyopadhyay, P., Biller, S., & Xiao, G. (2007). Maintenance opportunity planning system. *Journal of Manufacturing Science and Engineering*, *129*(3), 661–668.

Chang, Q., Xiao, G., Biller, S., & Li, L. (2013). Energy saving opportunity analysis of automotive serial production systems (march 2012). *IEEE Transactions on Automation Science and Engineering*, *10*(2), 334-342.

Chitra, T. (2003). Life based maintenance policy for minimum cost. In (p. 470-474). IEEE.

Deb, K., Pratap, A., Agarwal, S., & Meyarivan, T. (2002). A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE transactions on evolutionary computation*, *6*(2), 182–197.

Jin, X., Siegel, D., Weiss, B. A., Gamel, E., Wang, W., Lee, J., & Ni, J. (2016). The present status and future growth of maintenance in US manufacturing: results from a pilot survey. *Manufacturing Rev.*, *3*, 10. doi: 10.1051/mfreview/2016005

Kallen, M. J., & van Noortwijk, J. M. (2006). Optimal periodic inspection of a deterioration process with sequential condition states. *International Journal of Pressure Vessels and Piping*, *83*(4), 249-255.

Khazraei, K., & Deuse, J. (2011). A strategic standpoint on maintenance taxonomy. *Journal of Facilities Management*, *9*(2), 96-113.

Kim, S.-H., & Nelson, B. L. (2006). Chapter 17 selecting the best system. In S. G. Henderson & B. L. Nelson (Eds.), *Simulation* (Vol. 13, p. 501 - 534). Elsevier.

Kobbacy, K. A. H. (2008). Artificial intelligence in maintenance. In *Complex system maintenance handbook* (pp. 209–231). London: Springer London.

Lei, Y., Liu, J., Ni, J., & Lee, J. (2010). Production line simulation using STPN for maintenance scheduling. *Journal of Intelligent Manufacturing*, *21*(2), 213-221.

Marseguerra, M., Zio, E., & Podofillini, L. (2002). Condition-based maintenance optimization by means of genetic algorithms and monte carlo simulation. *Reliability Engineering and System Safety*, *77*(2), 151-165.

Nicolai, R. P., & Dekker, R. (2008). Optimal maintenance of multi-component systems: A review. In (p. 263-286). London: Springer London.

Rasmekomen, N., & Parlikad, A. K. (2016). Condition-based maintenance of multi-component systems with degradation state-rate interactions. *Reliability Engineering System Safety*, *148*, 1-10.

Salemi, P., Song, E., Nelson, B. L., & Staum, J. (2018). Gaussian markov random fields for discrete optimization via simulation: Framework and algorithms. *Operations Research*.

Yang, L., Ma, X., & Zhao, Y. (2017). A condition-based maintenance model for a three-state system subject to degradation and environmental shocks. *Computers & Industrial Engineering*, *105*, 210-222.

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.

BIOGRAPHIES

Michael Hoffman is a Ph.D. candidate in Industrial Engineering and Operations Research in the Department of Industrial and Manufacturing Engineering at the Penn State University. He is a Graduate Student Measurement Science and Engineering (GMSE) fellow at NIST. Previously, he was a Walker Graduate Assistant with the Applied Research Lab at Penn State. He received his B.S. in Industrial Engineering from Penn State. His research interests include intelligent manufacturing systems and big data in manufacturing.

Eunhye Song is Harold and Inge Marcus Early Career Assistant Professor in the Department of Industrial and Manufacturing Engineering at the Penn State University. She completed her B.S. and M.S. in Industrial and Systems Engineering from KAIST in Daejeon, South Korea and PhD in Industrial Engineering and Management Sciences from Northwestern University in Evanston, IL, USA. Her research interests include simulation design of experiments, simulation uncertainty and risk quantification, optimization via simulation under input model risk and large-scale discrete optimization via simulation.

Michael Brundage, Ph.D. is an Industrial Engineer in the Informational Modeling and Testing Group at the National Institute of Standards and Technology (NIST). Dr. Brundage's interests include Smart Manufacturing Diagnostics for Intelligent Maintenance, Sustainable Manufacturing Performance Measurement, Smart Manufacturing Capability Assessment, and Manufacturing Knowledge Visualization. His work contributes to guidelines for intelligent maintenance and he is part of a task group for creating an ASME Prognostics Health Management (PHM) standards committee. He also worked closely with ASTM International E60.13 in the development of a guideline for sustainable manufacturing performance indicators (ASTM E3096-17). He authored over 20 peer reviewed publications and has chaired multiple ASME MSEC Symposia and industry forums/workshops at NIST. Dr. Brundage is the recipient of the 2018 ASME Old Guard Early Career Award and was selected as one of SME's 2018 Class of 30 Under 30.

Soundar Kumara is the Allen, E., and Allen, M., Pearce Professor of Industrial Engineering at Penn State and has an affiliate appointment with the school of Information Sciences and Technology. His research interests are in smart manufacturing, large-scale networks, sensing and control, IIOT and Machine Learning in Manufacturing and Health Analytics. He is a Fellow of Institute of Industrial Engineers (IIE), International Academy of Production Engineering (CIRP), American Association for Advancement of Science (AAAS), and American Association of Mechanical Engineers (ASME). 54 Ph.D., and 64 MS students graduated under his tutelage. His Google citations is around 7700 and his Erdős number is 3.