Proceedings of the ASME 2020 15th International Manufacturing Science and Engineering Conference
MSEC2020
June 22-26, 2020, Cincinnati, OH, USA

MSEC 2020-XXXX

UNDERSTANDING AND EVALUATING NAIVE DIAGNOSTICS ALGORITHMS APPLICABLE IN MULTISTAGE MANUFACTURING FROM A RISK MANAGEMENT PERSPECTIVE

Mehdi Dadfarnia, Michael Sharp, Timothy Sprock
National Institute of Standards and Technology
Gaithersburg, MD 20899

ABSTRACT

The world has entered a state of unprecedented access to machine intelligence algorithms, where the ease of deployment has created a scenario where nearly every facet of life and industry has been affected by AI. Especially within industry, where the options for enacting AI systems are wide and varied, the choice of which system will work best for a given application can be daunting. Understanding when, where, and why to apply a particular algorithm can provide competitive advantage on effectiveness as well as greater trust and justification when using the algorithms' outputs. This paper examines multistage manufacturing processes, where system complexity can greatly influence the burden of creating custom tailored monitoring solutions. Such barriers have encouraged many manufacturing small and medium enterprises (SME) to look towards generic 'black box' commercial software solutions, although they may lack the sufficient expertise to objectively determine which product best meets their requirements. Some of the considerations faced by SMEs are identifying tools that can successfully be deployed alongside a potential lack of sensor coverage and/or the desire for rapid system reconfiguration to accommodate smaller custom batch production sizes. In these environments, detailed analytics-based solutions are often not feasible for production equipment monitoring. This paper provides a procedure for assessing the suitability of various tools or algorithms used to evaluate production process performance based on product quality output. This paper also presents a preliminary comparative example study of several algorithms to demonstrate this process and evaluate the selected algorithms.

Keywords: Manufacturing simulation, problem diagnosis, fault isolation, evaluating algorithms.

1. INTRODUCTION

Manufacturing is a highly competitive industry where every decision should be qualified to ensure both effectiveness as well as a solid return on investment. A common choice faced by manufactures is the decision of if, when, and how to monitor both the quality of their product and the effectiveness of the machines used in the processes. Due to limits in resources, especially for SME manufacturers, the availability of information sources such as sensors to monitor individual machine effectiveness may be severely restricted or otherwise unsuitable for analysis. However, the two forms of information available to nearly any sized manufacturer are the end part quality and process path used to create each individual part.

Tracking part quality information and quickly identifying sources of problems has become particularly important in agile multistage manufacturing facilities where small batch sizes and rapid reconfiguration are vital to maintain a competitive edge. Rapidly changing system dynamics can exacerbate and propagate problems in machine performance across multiple product sets, costing thousands of dollars if not quickly identified and managed. Selecting proper tools and methods for monitoring system performance is a significant decision that can have long-term implications for management and factory operators who will have to interpret and interact with any deployed monitoring system. Being able to understand and justify decisions regarding the selection of a monitoring system can increase confidence in that system and help users understand any risk associated with its use on a factory floor.

Past research has focused on utilizing part quality data and part process path information to aid in the determination of problematic equipment or process links as the basis for maintenance activities [1]. Some of these activities are very specific, such as using historical part quality degradation to augment and improve linear system dynamics models to predict metrics such as tool wear [2]. The effort and effectiveness of each technique can vary widely between applications. This work seeks to provide a comparative analysis of several popular fault or problem isolation algorithms and to explore the general areas where they are most and least effective at identifying causes of part quality degradation. From this, a workflow is developed for testing the range of applicability of any algorithm

as well as a basic guide for determining suitability of tested algorithms for various system setups. This paper is not meant to be an exhaustive comparison of all popular methods of multistage manufacturing monitoring, instead it is directed at developing an environment which can be used to critically evaluate tools and methods in a manner which allows for objective comparative assessment.

1.1. Background and Motivation

Stream of variance modeling has been a staple of multistage manufacturing since it was introduced in the 1990s [3]. Developed because rigid body assumptions do not always hold through production, stream of variance modeling recognizes that multistage processes can add compounding errors through both parts handling and machining. Some work focused directly on diagnosing fixture variation in multistage manufacturing processes [4] and led to further investigations of machining errors through explicit system modeling, such as via linear state space models [5]. Later work in statistical process control with linear state space models was able to utilize probability and hypothesis testing to capture faulty elements within a process [6]. Intuitively, many of these techniques are sensitive to the measurements and recordings used as inputs to the monitoring algorithm [7], meaning that they require significant sensing capabilities throughout the system. Other barriers to correctly implementing stream of variance algorithms include creating an accurate representation of features, selecting an optimal parts sampling criteria for inspection, and developing an adequate level of detail in modeling of the possible process faults [8].

Although these and related methodologies can provide important and accurate information regarding the propagation of errors as well as their initial incident location, explicit system models tend to require an advanced level of expertise, both with the system dynamics and the algorithms themselves, to properly implement. Additionally, many monitoring strategies relying on explicit system modeling become impractical for increasingly complex systems. To combat this, Huang et al. (2002) suggest simplifications and substitutions to mitigate this problem of growing complexity [9]. Many of these simplifications are based around the notion that there are certain configurations which will obfuscate areas from pinpoint diagnosis by simple virtue of their design. Zhou et al. (2003) understood that areas of obfuscation within a system, while unavoidable in some situations, would be strictly undesirable from a monitoring standpoint and proposed ways to quantify them [10]. In general terms, areas within a system become indistinguishable from one another if no unique information is generated between any subset of elements (within that area of the system).

Even with methods for simplifying the system model, often smaller enterprises do not develop monitoring tools that require explicit system models because the system dynamics change too rapidly to make any explicit modeling a practicality. This has driven investigations into less analytically-explicit monitoring algorithms. Further, due to the comparatively low amount of "stable data" produced by reconfigurable systems, practitioners seek algorithms that can operate with minimum input. One approach relies on using historical data to develop patterns of expected behavior from the machines based on post-production (or intermittent production) product quality reports, then comparing current behavior to develop diagnostic information. For example, methods for comparing expected distribution curves [11] and data- data-driven techniques for variation reduction [12] are gaining acceptance in multistage manufacturing.

Ultimately increasing data integration in automation and manufacturing depends on strong algorithms supporting human decision-making [13]. The strength of these algorithms depends on their performance evaluation and applicability. Evaluating the wide variety of algorithms in development and in practice has been inconsistent and traditionally reliant on the expertise levels of the developers or practitioners involved in the algorithm deployment. Most evaluations will stop if there exists a comparative analysis against one other algorithm on a limited set of system scenarios or data. This is due in part to the large time investment required to set up a large comparative study, but also due to a lack of understanding about how to set up such a study. In order to best evaluate an algorithm's suitability for a system, a majority of potentially disruptive scenarios must be considered. Regardless of the method or selection and availability of input data, the procedure for qualifying an algorithm or tool on a system must consider a majority of relevant edge cases as well as the nominal expected scenarios.

This work explores evaluating process fault or problem diagnostic tools with a very limited set of input data, specifically process path and part quality. That does not preclude extension of these procedures to more extensive algorithm testing. The choice to focus on algorithms that utilize limited information was motivated to both highlight the procedure on a simple comparative case with standard input parameters and to help shed light on real decision-making problems faced by SMEs. This case study will examine some of the 'black-box' solutions that are being applied to this problem. These include probabilistic statistical algorithms, gradient descent, genetic algorithms, and neural networks. These solutions will be framed to evaluate unsophisticated applications similar to that as may be identified by an SME seeking low-cost solutions to integrating new monitoring programs.

2. METHODOLOGY

This paper utilizes a general workflow for testing various fault or problem isolation algorithms. Defined or obtained series of system configurations will serve as the validation environment for obtaining exemplar data to investigate representations of a wide range of normal and off-normal operations. Special emphasis should be placed on investigating scenarios for both configurations and conditions that are reasonably expected to exhibit themselves in practice and would be antagonistic towards the algorithms being evaluated. In most multistage manufacturing systems, it is unreasonable to attempt to investigate all possible scenarios; part of this work is meant to help guide and highlight the process of determining high-risk edge cases that will provide the most pertinent information regarding the algorithm being evaluated. The final steps of the evaluation procedure are to apply the selected algorithms and measure them via metrics most suited to the end goals of the production line and, if necessary, iterate through

the antagonistic scenarios to develop confidence in the coverage and performance of the algorithms.

2.1. Selecting Testing Scenarios

To show the viability and effectiveness of this workflow, a simulator was developed that can represent a variety of multistage manufacturing systems and conditions. For an actual manufacturing facility, it may be sufficient to simulate the actual or possible configurations for that specific production floor, possibly augmented with available historical data. This paper focuses on cases where the particular system that an algorithm will be tested on is not known beforehand. This additionally addresses the broader research perspective on which types of systems an algorithm is useful for. Testing various configurations follows from understanding that system configuration can profoundly impact the performance of the manufacturing process [14], and thus any monitoring tools deployed on them.

Each testing scenario consists of two major aspects: the system configuration and the condition. The configuration relates to how the equipment is connected and the process flow. The condition conveys the health of machines, production rates, etc. In many systems, the large number of possible system configurations and conditions makes it impractical to test every single one. Therefore, the testing methodology focuses on three primary classes of scenarios: nominal, high-risk, and those that are antagonistic towards the algorithm being tested. Some scenarios may fall into multiple classes, but a proper set of testing scenarios should provide sufficient coverage representing these classes as evenly as is practical. Finally, each scenario has a likelihood that gives a relative expectation of how often that scenario would be expected to occur in the lifetime of the system.

A major driver in selecting scenarios is the potential consequences of that scenario, both good and bad. Understanding the potential outcomes of combinations of missed fault alarms, false fault alarms, as well as correct alarms all can help drive the selection of scenarios. Coupled with the associated frequency, a potential risk for each scenario can be developed and used to identify important test cases. The risk metric (frequency * consequences) should be evaluated in terms most suited to the application. In a manufacturing setting, for example, risk can be measured as production loss per hour. Once acceptable levels of risk are established, any scenario which falls below that can be safely ignored.

2.1.1. Low-Risk High-Frequency and High-Risk Scenarios

The scenarios that should be developed first are those capturing the most common configurations and expected conditions of the factory environment. Even if the risk of these scenarios is fairly low, the fact that they are the most common scenarios necessitates their evaluation for any potential monitoring algorithm. These include scenarios such as a single faulty machine that does not stop operations. One occasionally overlooked scenario that should always be included is the most common configuration(s) with fault-free, nominal conditions. This is critical to help characterize the false positive rates of any diagnostic algorithms and establish a risk based on false alarms.

The second priority scenarios are those that may or may not be common but present a high-risk factor if the algorithm is unable to correctly identify the source of incipient defects or problems. High risk can come from both false positives and false negatives in terms of identification and must be evaluated for different combinations of both.

2.1.2. Antagonistic Scenarios

Antagonistic scenarios are less intuitive because they are typically more dependent on the algorithm than the physical process, but they are important to understand when interpreting evaluation testing. Antagonistic scenarios are configurations and conditions for which an algorithm is expected to perform poorly based on the assumptions and capabilities of that particular algorithm. These types of scenarios may not be known prior to the evaluation of the algorithm, but if a set of poor performing scenarios begins to develop, any identifiable commonalities can be used to group the full set of test scenarios and add extra examples to those groups if needed.

Once a set of antagonistic scenarios is developed, the existing suite of test scenarios can be checked for how commonly those antagonistic qualities occur. If they are prevalent, especially in high risk scenarios, there may not be a need to continue evaluating that algorithm, because it would be expected to perform poorly in these high-risk scenarios. However, if there are not many scenarios with these common traits, it is prudent to construct more such scenarios and evaluate if these have a high enough cumulative risk to affect the decision of accepting the algorithm.

2.2. Obtain Evaluation Data

Simulations and/or historic data will be required to perform these tests. Where available, simulators would generally be desirable to augment the available scenarios that can be evaluated. Real systems with huge backlogs of data can be used as an initial set of testing scenarios, but it is unlikely that real systems have substantial amounts of high-risk data under the array of faulted conditions that would be desired to fully evaluate a potential scenario. The flexibility of a simulator, augmenting any available real scenario data, allows for better exploration of high risk and antagonistic scenarios. Existing data or logs of activities can also be used to help develop the associated frequencies and consequences of any scenario. Maintenance and production logs may be a good source of this information.

2.3. Evaluate Algorithms

Once the test scenarios and data has been obtained, the algorithms should be streamed onto the data as they would receive it in an actual production environment. For example, if there is not a live stream of the product quality available in the plant, then the algorithm should be provided batch style updates with corresponding time stamps and other relevant meta-data. The processing time of the algorithm under evaluation should also be noted at this time. If the routine takes longer to process than the update rate for the system, special considerations and accommodations must be made, such as artificially slowing the input rate. If this cannot be done, or the accommodations are

too severe to work at scale, the algorithm may be deemed unusable without further testing.

2.4. Representing Key Algorithm Performance Indicators

When evaluating algorithms, it is important to select performance indicators that not only reflect the performance of the algorithms but do it in a way that is relatable to the end goals of the monitoring algorithms. For most diagnostic isolation problems, the most important metrics are those that directly relate to negative consequences: false positives - identifying a fault where there isn't one, false negatives - failing to identify a fault, and circumstances that produce both. Because the repercussions of these three outcomes are generally significantly different and may even relate to specific equipment within the system, it is most appropriate and convenient to record the performance of the algorithms as a series of confusion matrices, one for each testing scenario. The important aspect for a risk-based evaluation is that each testing scenario have a numeric representation of the rate of negative consequences that can be translated into a probability during the final evaluation.

2.5. Summary

Below is a summary of the algorithm evaluation process presented in this section. The methods and selection criteria presented in this section are not intended to be interpreted as the only, or best possible criteria for every case. Instead they are described as a possible set that would be applicable in most cases and are the methods used in this work. The next section will describe the specifics of this work as applied to the developed test cases. The algorithms chosen for evaluation are also described, followed by the general outcomes of that investigation.

1. Define Testing Environment
2. Select Testing Scenarios
   a. Nominal and High-Risk Scenarios
   b. Antagonistic Scenarios
3. Obtain Data Covering Selected Scenario
4. Evaluate Selected Algorithm(s)
   a. Check Usability / Operational Concerns
   b. Evaluate Performance
5. Repeat as needed to discover edge cases with poor performance

3. PROOF OF CONCEPT CASE STUDY

This case study highlights the methodology for evaluating a specific class of monitoring algorithms that may be used to determine potential locations of induced damage or defects in products output from multistage manufacturing processes. The scenarios and configurations presented are selected to allow broad level performance investigation of algorithms on a potentially unknown system. A simulator was created as part of this work to rapidly create arbitrary scenarios generating the required end-of-line part quality and part process production path data. This data is then used to evaluate multistage manufacturing system diagnostic algorithms. The setup of the simulator includes the manufacturing system configuration (section 3.1), test case scenarios that characterize nominal and edge cases faced by the system (section 3.2), and diagnostic algorithms applied to analyze the scenarios (section 3.3). This evaluation is presented as preliminary work and additional work is needed to completely characterize the algorithms under evaluation.

3.1. Define Testing Environment

Figure 1 shows an example of different multistage manufacturing paths to produce a part, given a limited number of paths and available machines. Machines can be used across multiple paths (process plans), subject to timing restrictions, resulting in a directed graph representation of product production. Here we assume that it is possible for different production paths to use the same machine, but not possible for any path to use a particular machine more than once (reentrant flows). The subset of machines within each process path is assigned, either randomly or manually, at the beginning of each scenario and held static for the duration of the test.

The type of manufactured product is unspecified during the simulation, but it is assumed that quality metrics of any produced parts are scalable to some equivalent metric. Each part is given a single score (*Q_part***)** to indicate its build quality at the end of its sequence (end-of-line part quality). End quality assessments (*Q_part*) are scaled to a percentage of acceptability, where 100 % is perfect and anything below 0 % is considered lost product. This score relies on the added value from each machine that the part interacts with in its production path. Added quality value to the part is uniform across all machines unless the part interacts with a machine that underwent a sudden degradation or failure. In this case, the degraded machine will subtract from the part's accumulated value. Currently, part quality inspection is limited to the end of each production path.

Although simplified and abstract, this system configuration simulator exhibits key behaviors of real-world production settings, including resource constraints, a limited number of production path setups, and the limited number of available machines for production. Further development of the system configuration in future studies will include constraints on machine-ordering in a path (i.e., having rules such as "machine #6 may never be used before machine #3") and available

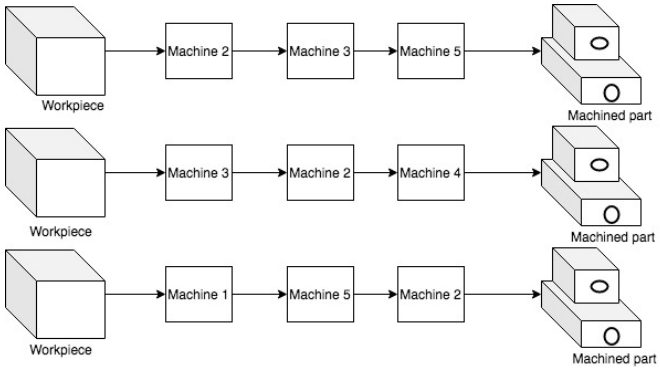

Figure 1. An example of three, unique three-stage machining paths to produce a part given five available machines.

machines categorized by different types, with constraints on the number of each machine type required to build a part. The machines may also have a more expansive feature space that determines each machine's added value to a part's quality.

3.2. Select Testing Scenarios

The above setup provides a framework to simulate different scenarios that could be used to evaluate multistage manufacturing system diagnostic algorithms that attempt to identify causes for part quality degradation. The focus in this paper is on part quality degradation due to sudden machine degradation or failure. The simulator receives specifications that after a certain number of products, a subset of the total set of available machines will degrade their added manufacturing values to the quality of parts that interact with it. As the quality for each part relies on the accumulation of the added values from all machines that are in the part's production path, the paths that have one or more of these degraded machines will output parts with lower end-of-line part quality values. Figure 2 illustrates a temporal view of the part quality outputs for an example test case scenario. Note that although the scalar part quality values are arbitrary, there is a drop-in value for parts that have been processed by some of the paths (that include the machine degradation at t=200 s).

The inputs into the configuration of the simulator are the number of machines on the "production floor", the number of processing steps each item needs to be created, the number of production paths or lines that a part can be created on, and the rate at which those parts are made. These inputs define a system of machines and paths involved in the multistage manufacturing process to produce each part. These key manufacturing parameters that go into the system configuration are summarized in Table 1. Presented in this paper are two preliminary scenarios defined by the values given in Table 1.

Test case scenarios are built to evaluate how well diagnostic algorithms identify degraded machines. In particular, it is interesting to evaluate the response of these diagnostic algorithms to different numbers of machines degraded at a time. Using the number of degraded machines as the main experimental factor in test case scenarios provides insight into the usability and accuracy of each tested algorithm

Table 1. System Configuration Parameters

Description	System #1	System #2
Total number of available machines	10	12
Machines in a production path	4	6
Number of unique production paths	13	20
Part production rate (part per second)	1	1

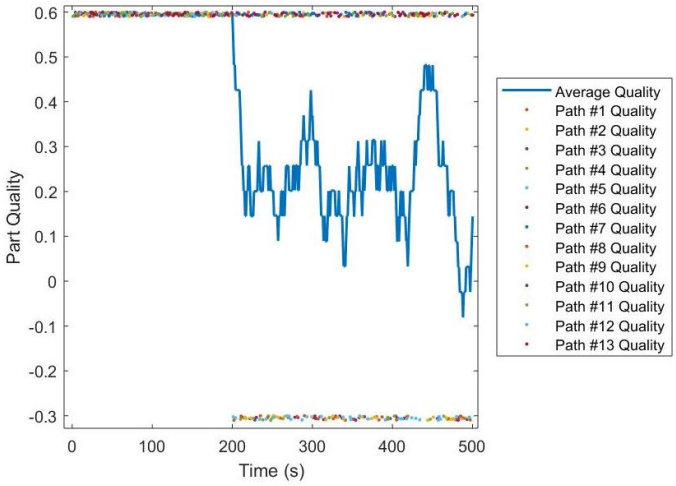

Figure 2. Example of a temporal view of part quality production. Machine degradation begins at t=200 s. Subsequently, paths that include the degraded machine produce lower-quality products.

for different numbers of degraded machines. These test cases are implemented on the two system configurations in Table 1.

For the first system configuration example, where there are 13 unique production paths and 4 machines (selected from an available 10) in each path, there are three levels to the experiment factor: one, two, or three machines are degraded. A maximum of three degraded machines is selected because it reaches sufficient coverage over scenarios where most of the machines in some production paths (3 of 4) are producing undesirable results.

Ten experiments, or test case scenarios, are performed for each of these three machine degradation levels. Each of these experiments is performed with different randomly-selected permutations of degraded machines. Ten is the maximum number of permutations when only one machine degradation occurs out of ten available machines. The number of possible permutations for two or three machine degradations is much higher. For the sake of brevity and producing initial results, the other machine degradation levels were limited to ten test case scenarios as well.

Table 2. Test Case Scenario Setup

	System #1	System #2
Number of Factor Levels	3	4
Number of Experiments Per Level	10	12
Total Number of Test Case Scenarios	30	48
Degradation Start Time (s)	200	200
Number of parts (stopping time)	500	800

Test case scenarios are similarly designed on the second system configuration example, where there are 20 unique production paths and 6 machines (out of an available 12) in each path. Since the scope of this example's multistage manufacturing system is larger, there are instead four levels to the experiment factor (a maximum of four degraded machines) and 12 experiments with different permutations at each level (12 is the maximum number of permutations when only one of 12 available machines degrade). Table 2 summarizes the test case scenario setup for each of the system configurations in Table 1.

The simulator generates the scenarios and stores the machines that degrade in each test case. The sequence of machines in the paths of each system configuration is held constant against the different machine degradation schemes from the test cases. This allows for applying diagnostic algorithms on each of these test case scenarios to compare their prediction of degraded machines against the actual machine failures as well as to evaluate their applicability to different machine degradation scenarios (i.e. different numbers of machines that degrade at a time).

The system configuration and test case scenario setups also enable the derivation of different properties from the system configurations. These properties may include the frequency of degraded machines within all the production paths or the amount of information that can be distinguished from the machines traversed in each path. This gives the opportunity to manipulate the properties and observe the prediction response of diagnostic algorithms to changes in different properties in the system configuration. This provides insight into the applicability and limitations of the diagnostic algorithms when there are changes in the system configuration. Discovering these insights requires changing the test case scenario setup to use the derived properties as experimental factors in a more extensive, robustly-designed experimental methods [15]. Deriving these properties and designing experiments around them are topics intended to be explored in future studies. Also, this paper has only been looking into the case of sudden machine failures. Future studies will expand to look at different types of machine failure modes, such as observing when diagnostic algorithms identify more gradual machine degradations.

3.3. Diagnostic Algorithms Application Setup

Five different diagnostic algorithms have been selected for evaluation and applied on each of the test scenarios for this work. Each algorithm makes use of only two sources of information: the end-of-production-line part quality, and the part process production path. Testing different algorithms allows to identify strengths and weaknesses of each one with respect to the different system configurations, amounts of machine induced degradation, and locations of induced damage. The selected algorithms are only a small random sampling of potentially useful methods for isolating induced faults on a multistage production line. The following describe the five algorithms tested.

3.3.1. Probabilistic Statistical Algorithms

One of the more intuitive mechanisms for performing diagnostic isolation is to evaluate the probability or likelihood that a given machine is producing a defective part based directly on the observed outputs corresponding to that process element. This could be framed under the guise of temporal difference reinforcement learning, a relative to Q learning. Presented below are two algorithms that utilize the intuitive nature of the process to create logical evaluations of the relative probability any given machine (process element) is inducing damage on the final observed product.

Part Quality Contribution Indicator (PQCI):

The first algorithm uses a running log of parts produced by each machine. It checks the ratio of the quality of parts operated on by a particular machine versus the quality of parts that were not. It uses this comparison to determine an estimate of the average induced damage at that machine. For this study, the quality of the last ten parts to be operated on by each machine is kept and measured. One of the obvious shortcomings of this method is that if a machine does operate on a significant number of parts as compared to the total number produced, it will be slow to identify the problem machine. However, with even levels of part flow or if the problem element is not a high-risk machine, then this algorithm may be expected to perform suitably for many scenarios.

Estimated Part-Path Contribution Indicator (EPPCI):

The second algorithm investigated also makes use of a moving window of part quality to produce diagnostic indicators. However, this algorithm keeps running logs of each process path instead of one for each individual processing element. In scenarios with relatively few process paths, this may improve scalability. Conversely this could also hinder scalability in highly complex system configurations with a large number of paths.

By performing a simple calculation using the system configuration and path quality logs, this algorithm calculates the average product quality of each machine per path to identify both the best and the average part quality produced by each machine based on the paths that include it. This method attempts to circumvent the potential problem of having certain process links, instead of individual machines, be the source of induced errors. For this paper, isolation of links was strictly excluded as it is intended for future papers, but the setup of this algorithm begins that process. Regardless, this algorithm would also be expected to perform well in isolating individual problem machines and so is evaluated on such here.

3.3.2. Prediction Error Minimization Algorithms

There are a class of low entry barrier machine learning tools and algorithms that attempt to minimize some predictive function designed to describe the observed output of the system. The commonality of this class is the need for that descriptive function. In this case, a simple minimization between the observed product damage (1 – Quality) for each machine and the predicted total amount of added damage from the processing machines. Assuming each machine linearly adds a calculable amount of damage to the product during processing, this class

of algorithms can estimate the average amount of that damage that is added by each machine.

$$EQ(1) \quad Error = abs(\,(1 - sum(IDam\_machs\_part)) - (1 - Q\_part)), \qquad for\ all\ parts$$

The selected predictive equation of error for a system, shown in Eq(1), is utilized for three different diagnostic algorithms. Based on simple Gradient Descent (GD), one on Genetic Algorithms (GA), and one on applied Neural Networks (NN). Regardless of the mechanical implementation, these tress routines are all essentially trying to minimize this error function by producing induced damage estimations (IDam) for each machine.

Gradient Descent (GD): The Gradient Descent based method utilizes a standard off-the-shelf product (Matlab's *fmincon*) to search the possibility space for the combination of each machining element between sensible limits set by the user. For simplicity in this test, the choice was made to limit the possible damage induced by each machine to be between 0 (no damage) and 1 (complete loss of product acceptance). While other implementations of this algorithm may find other bounds more suitable, these are simple enough to cover a broad range of cases and limit processing time to a functionally usable amount.

Genetic Algorithms (GA): Considered a generally more robust optimization method, genetic algorithms have been used extensively on optimization problems due to their broad applicability, ease of use, and global perspective [16]. Like the gradient descent method above, a genetic algorithm is applied to test case scenarios to minimize the objective function in Eq. (1). The optimization process also outputs each machine's contribution to part quality degradation and a prediction of the machines that have degraded. To ensure the feasibility of the predictions of machine degradations, the algorithm was iterated 5 times over each test case scenario and its predictions were averaged. The genetic algorithm implemented in Matlab's Global Optimization Toolbox was evaluated using its default options (i.e., population size = 50, maximum number of generations = 100).

Neural Networks (LSTM): The final prediction error minimization technique applied in this work is a Long Short Term Memory (LSTM) based neural network. These are a special form of recurrent neural networks, which as explained by [17], are better able to process time series information due to having storage potential creating an effective memory of local trends. Neural networks have become standard machine learning tools due to their ease of use, effectiveness as classifiers, and practicality for obtaining features out of a dataset. The LSTMs created for this work serve to provide not only preliminary insights on their applicability, but also to justify further testing on additional architectures and configurations in future studies.

The selected network configuration under evaluation has a sequence input layer, a series of two hidden LSTM layers (each followed by a 20 % dropout layer), one fully connected layer, and a final regression layer. The input vector to the LSTM network includes the part quality and a corresponding binary vector representing production path elements of that part. Creation and execution of the LSTM network was performed using Matlab's Deep Learning Toolbox. The specifics of hyperparameter selection and architecture development are beyond the scope of this work. The process used here to compare various algorithms could be equally suited for comparing various neural network architectures.

4. TEST RESULTS

The results of evaluating the algorithms discussed in Section 3.3 compare their ability to identify damage-inducing machines in different scenarios and begin to establish their expected performance for classes of scenarios. The five diagnostic algorithms are evaluated and compared by applying them to the test scenarios (see Table 2) developed for each of the two multistage manufacturing system configurations (see Table 1). That is 30 test case scenarios for system configuration #1, evenly divided into 3 different categories of test cases, and 48 test case scenarios for system configuration #2, evenly divided into 4 different categories.

Each algorithm is evaluated using metrics chosen to measure the algorithm's performance at diagnosing machine degradation. To relate to real world based effects, metrics that can be directly combined with negative scenario consequences were selected: the false negative rate (FNR), the false positive rate (FPR), and the combination of the two. The false positive rate is calculated as the number of good machines incorrectly predicted to be inducing damage divided by the total number of machines that are operating in the system (Type-I error). Likewise, the false negative rate is calculated as the number of machines inducing damage not isolated by the algorithm divided by the total number of machines that are indeed producing damaged units (Type-II error). Not identifying a machine as a source of damage incurs costs associated with manufacturing defective parts and goods. While incorrectly identifying a machine as a source of damage incurs costs from misspent maintenance, inspection, and production downtime.

Figures 3A and 3B summarize the evaluation results from applying each of the five diagnostic algorithms to the selected test scenarios. In system configuration #1 there are three classes of test case scenarios corresponding to the number of machines that are producing degraded products: one, two, or three machines. Each diagnostic algorithm, except the LSTM neural network, is applied over the ten scenarios in each category. The resulting FNR and FPR evaluations are averaged for each category. This is similarly done for system configuration #2, but with four different level categories of test case scenarios and twelve test cases in each category.

Due to the requirements for development of a neural network, the LSTM network testing was performed differently. Rather than evaluate its performance on each category of test scenarios individually, the LSTM was trained, tested, and evaluated on all the test case scenarios available for each system configuration (30 test cases for system #1 and 48 for system #2). The model randomly selected 67 % of each configuration's test case scenarios as training data and the rest of the scenarios as testing data. This ensured that the network had training of various conditions on each configuration, allowing it to produce

predictions of both nominal and problem status of machines. This also means that the LSTM's evaluated FNR and FPR metrics in Figures 3A and 3B cannot be directly compared to the corresponding evaluations of the other algorithms, but it still points towards preliminary insights into the applicability of neural networks for machine failure predictions. The LSTM results are included here for completeness but will be readdressed in future work.

As structured in Section 3.3, the algorithms listed each produce relative values for problem likelihood. In order to translate this into a classification of 'problem', 'no problem', a series of thresholds must be established. A threshold of -0.1 likely induced damage is selected as a discriminator for evaluating PQCI and EPPCI. GD and GA have been framed to operate on a different scale which translates their cutoff to a threshold of 0.6 is selected as the discriminator for evaluating both GA and GD.

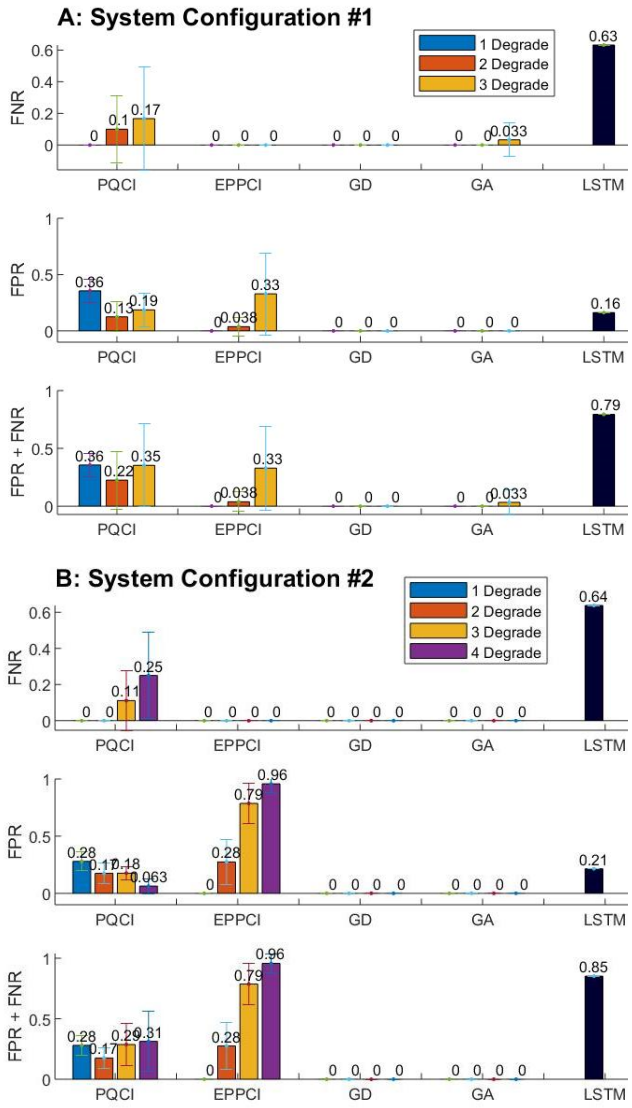

Figure 3. Evaluation results for applying the five diagnostic algorithms to predict machine degradation for three different test case scenario categories of: (A) system #1 and (B) system #2.

Like the GD and GA, the LSTM neural network also tries to predict which machines have contributed to part quality degradation. This means the threshold must also be between 0 and 1 to quantify the degree of a machine's contribution to part degradation. However, training the neural network on all the test case scenarios of each system configuration means that it tried learning from a wider range of antagonistic test case scenarios (rather than make a prediction from one test case at a time like the other four algorithms). This decreases the confidence in the predictions and so the discriminating threshold was lowered to 0.3.

At first glance, the results in Figures 3A and 3B show that the GD and GA algorithms provide superior performance to all the other algorithms. GD results in FNR and FPR values of 0, meaning it never identifies a failed machine as operational or an operational machine as degraded. This result is similar for GA, except a small false negative rate is produced when there are 3 machine degradations in system configuration #1. This supports a hypothesis that GD and GA will not make perfect predictions when there are too many machine failures in a production line. More testing is needed to see if this is indeed the case and to identify antagonistic test scenarios that may hamper the performance of GD and GA.

The PQCI and EPPCI algorithms exhibit trade-offs with each other. EPPCI produced no false negatives – no predictions of failed machines as operational - in any of the test case scenario levels for either system configuration. However, EPPCI produces a lot of false positives, especially for test cases where there are more machines that simultaneously degrade. It especially showed bad performance for multiple machine degradations in the second system configuration. On the other hand, PQCI produced false negatives that increased with more machine degradations, but it showed consistent levels of false positive rates that did not increase (drastically) with more machine degradations. The strength of one of these two algorithms over the other depends on the costs of having false negatives versus false positives in a manufacturing setting.

The predictive performance of both PQCI and EPPCI does not show to be as good as GD and GA, but their potential for scalability of more complex systems and to search for more indirect link based problems shows more promise. GD and GA attempt to search a full error space which grows exponentially with system complexity, while the evaluation space of the Q-learning space of PQCI and EPPCI grows linearly.

These insights are invaluable for creating antagonistic scenarios for further testing and evaluation of the algorithms. When identifying antagonistic test scenarios, an algorithm may have associated ranges of FNR and FPR that could help to further generalize the evaluation outcome. By properly combining the likelihood of these categories of test cases, their FNR and FPR values, an aggregated weighted evaluation of each diagnostic algorithm could be made to augment the existing test scenarios. This additional 'virtual testing' can save time and resources, while still valuably expand the coverage of the algorithm evaluation.

The LSTM neural network does provide many false negatives but produced false positive rates that are comparable

to PQCI and EPCCI. A portion of the unfavorable results is due to training the LSTM on all test case scenarios for each system configuration. This points to the conclusion that learning on too many different antagonistic scenarios may not be as helpful for making machine failure predictions – there is a trade-off between the number of different antagonistic scenarios and the quality of the machine failure predictions. It will be helpful to see how the LSTM network performs when learning from and being applied to only one test case category level at a time. Furthermore, the LSTM network presented here is supposed to showcase preliminary work in the area, and more work will be done to fine-tune LSTM-related hyperparameters or evaluate other neural network structures.

5. CONCLUSIONS AND FUTURE WORK

This study showcases a preliminary understanding, applicability, and comparison of diagnostic algorithms (Section 3.3) when applied to predict machine failures in different simulated scenarios. The goal is to develop procedure recommendations for applying diagnostic monitoring algorithms across different scenarios, ranging from nominal expected scenarios to edge cases, faced by generic system configurations or particular configurations.

Further work needs to be done to extend the selection space of testing scenarios that are covered by the evaluation methodology, and to increase diagnostic algorithm development and evaluation. Two areas of particular interest right now include, 1) further development of risk assessment as an algorithm evaluation criteria, and 2) extending quality inspection to include mid-production process inspections. Algorithm evaluation can more precisely assess risk by incorporating costs and consequences associated with false negative and false positive machine degradation predictions, as well as relative likelihoods of different scenarios faced by the system configuration. This risk metric will represent more-practical effects and trade-offs of algorithms in relatable real-world measures beyond abstract classification metrics. In this preliminary study, the part quality inspection was conducted only at the end of the production path. This inspection data can be used to determine points in the production line that may require additional inspection or maintenance effort. These additional inspection points may provide diagnostic algorithms with more data collected throughout the production path and more context to make better predictions.

DISCLAIMER

The use of any products described in this paper does not imply recommendation or endorsement by the National Institute of Standards and Technology, nor does it imply that products are necessarily the best available for the purpose.

REFERENCES

[1] Lu, Biao & Zhou, Xiaojun. (2019). Quality and reliability oriented maintenance for multistage manufacturing systems subject to condition monitoring. Journal of Manufacturing Systems. 52. 76-85.

[2] Hao, Li & Bian, Linkan & Gebraeel, Nagi & Shi, Jianjun. (2016). Residual Life Prediction of Multistage Manufacturing Processes With Interaction Between Tool Wear and Product Quality Degradation. IEEE Transactions on Automation Science and Engineering. 14. 1-14. 10.1109/TASE.2015.2513208.

[3] Hu, S.J., Yoram Koren (1997). ”Stream-of-Variation Theory for Automotive Body Assembly”, CIRP Annals, Volume 46, Issue 1,1997, Pages 1-6.

[4] Ding, Yu & Ceglarek, Dariusz & Shi, Jianjun. (2000). Modeling and diagnosis of multistage manufacturing processes: Part I state space model.

[5] Djurdjanovic,D., J. Ni, (2001). "Linear state space modeling of dimensional machining errors", Trans. NAMRI/SME, vol. XXIX, pp. 541-548.

[6] Li, Yanting, & Fugee Tsung (2009) False Discovery Rate-Adjusted Charting Schemes for Multistage Process Monitoring and Fault Identification, Technometrics, 51:2, 186-205.

[7] Zeng, Li, & Shiyu Zhou (2007) Variability monitoring of multistage manufacturing processes using regression adjustment methods, IIE Transactions, 40:2, 109-121.

[8] Zhang, Min & Djurdjanovic, Dragan & Ni, Jun. (2007). Diagnosibility and sensitivity analysis for multi-station machining processes. International Journal of Machine Tools and Manufacture. 47. 646-657.

[9] Huang, Q., Zhou, S., and Shi, J. (2002). “Diagnosibility of Multi-Operational Machining Processes Through Variation Propagation Analysis,” Robotics and CIM Journal, 18, 233–239.

[10] Zhou, Shiyu, Yu Ding, Yong Chen & Jianjun Shi (2003) Diagnosability Study of Multistage Manufacturing Processes Based on Linear Mixed-Effects Models, Technometrics, 45:4, 312-325.

[11] Davari-Ardakani, H., & Lee, J. (2018). A Minimal-Sensing Framework for Monitoring Multistage Manufacturing Processes Using Product Quality Measurements.

[12] Liu, Y., Sun, R. and Jin, S. (2019), "A survey on data-driven process monitoring and diagnostic methods for variation reduction in multi-station assembly systems", Assembly Automation.

[13] Thoben, K. D., Wiesner, S., & Wuest, T. (2017). “Industrie 4.0” and smart manufacturing-a review of research issues and application examples. International Journal of Automation Technology, 11(1), 4-16.

[14] Koren, Yoram , S. Jack Hu, Thomas W. Weber, (1998) Impact of Manufacturing System Configuration on Performance, CIRP Annals, Volume 47, Issue 1.

[15] Drain, D. C. (1997). Handbook of experimental methods for process improvement. CRC Press.

[16] Goldberg, D. E. (1989). Genetic Algorithms for Search, Optimization, and Machine Learning. Reading, MA: Addison-Wesley.

[17] Cady, F. (2017). *The Data Science Handbook*. John Wiley & Sons.