Antonis Kokossis, Michael C. Georgiadis, Efstratios N. Pistikopoulos (Eds.)
PROCEEDINGS OF THE 33rd European Symposium on Computer Aided Process Engineering (ESCAPE33), June 18-21, 2023, Athens, Greece
© 2023 Elsevier B.V. All rights reserved. http://dx.doi.org/10.1016/B978-0-443-15274-0.50251-1

Modular Development of Condition Monitoring Systems for the Tennessee Eastman Process

Rexonni B. Lagare,[a] Marcial Gonzalez,[b] Zoltan K. Nagy,[a] Gintaras V. Reklaitis[a]

[a]*Davidson School of Chemical Engineering, Purdue University, West Lafayette, IN 47907, USA*
[b]*School of Mechanical Engineering, Purdue University, West Lafayette, IN 47907, USA*

Abstract

This paper presents a condition monitoring system development framework that incorporates process knowledge to enhance the performance of machine learning models. Essentially, the framework uses information about the process to create a representation of the process condition, which can be broken down into modules using concepts borrowed from probabilistic graphical modeling. These modules represent simpler problems for fault detection and diagnosis, which allows traditional machine learning (ML) models to perform better without the need for a larger set of training data. Using the Tennessee Eastman Process (TEP) as a case study, the framework was shown to improve detection and diagnosis of all fault types in the TEP fault library under relevant metrics for evaluating condition monitoring systems.

Keywords: Tennessee Eastman Process, Machine Learning, Condition Monitoring, Fault Detection and Diagnosis, Condition-based Maintenance

1. Introduction

Continuous manufacturing processes often require a condition-based maintenance approach in order to fully realize its benefits. This requires the effective implementation of a condition monitoring (CM) system that can holistically oversee the condition of a process.(Schenkendorf, 2016) It is often challenging to obtain a mechanistic model of the condition of a process, so most of the work in literature focused on data-driven methods such as machine learning (ML).(Yin et al., 2012) The workflow for this development can be visualized in Figure 1, where data from the equipment and sensors of a continuous tableting line are used to train a ML model that can detect and diagnose faults. Given the right dataset, this workflow can be very effective in creating high performing fault detection and diagnosis algorithms. The problem is real scenarios rarely have the right dataset available. Hence, it is often the case that traditional ML models developed in this manner underperform, and the course of action is to acquire a better dataset, which might be prohibitively expensive, and/or to use a better ML model.

Another course of action that one could take is to build on the existing knowledge about the process to develop a mechanistic model. However, level of process knowledge that is required to do this could be even more expensive than acquiring a better dataset for training a machine learning model. Hence, any available process knowledge is often neglected because it is not enough for mechanistic modeling, and modeling efforts would be directly to a purely data-driven approach. An innovative solution would be to find a way to apply this knowledge in enhancing the performance of the ML model development workflow in Figure 1. Such a framework was developed and found to be effective for a continuous pharmaceutical tableting pilot plant. (Lagare et al., 2022a)

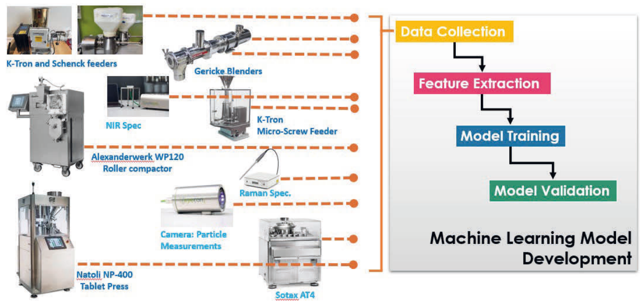

K-Tron and Schenck feeders
Gericke Blenders
NIR Spec
K-Tron Micro-Screw Feeder
Alexanderwerk WP120 Roller compactor
Raman Spec.
Camera: Particle Measurements
Natoli NP-400 Tablet Press
Sotax AT4
Data Collection
Feature Extraction
Model Training
Model Validation
Machine Learning Model Development

Figure 1. Machine Learning Model Development Workflow Applied to the Continuous Tableting Pilot Plant at Purdue University

It is now interesting to question the limitations of the framework, especially in its applicability towards larger processes with a larger fault library. For this purpose, the Tennessee Eastman Process (TEP) will be considered,(Downs and Vogel, 1993) which is a very different process than a continuous tableting line. First, the material streams are no longer solids, but either liquids or gases. The TEP also has more chemical components and chemical transformations are involved. There are more unit operations in the TEP and to add to its complexity, includes a recycle stream, which the pharmaceutical solids processing system does not have. Finally, the fault library of the TEP is much larger, with 22 conditions that need to be determined from 53 input variables taken from sensors and equipment across the process. (Chiang et al., 2000)

As expected, the condition monitoring system for the TEP, developed using workflow in Figure 1, showed poor performance using metrics relevant to condition monitoring. This study shows that the condition monitoring system development framework, which will be discussed in Section 2.1, was effective in improving the performance of the machine learning models in monitoring the condition of the process.

2. Methods

2.1. Condition Monitoring System Development Framework

The proposed framework adds several steps to the ML model development workflow in Figure 1, which is now depicted as a node (white) in Figure 2. The two additional steps, i.e., representation and modularization, is mainly responsible for improving the ensuing ML model development step and will be the main subject of this paper.

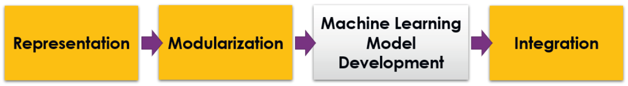


Figure 2. Proposed Condition Monitoring System Development Workflow

The final step of the proposed framework—integration—is critical in holistically interpreting the predictions of the modules. The result is a more robust condition monitoring system that can still function amid sensor maintenance repairs, reducing the need for product diversions and/or process shutdowns.(Lagare et al., 2022b)

2.1.1. Process Representation

Process representation is the first step in the CM system development framework. This step is responsible for incorporating available process knowledge into the workflow, and its key components are shown in Figure 3.

Based on these components, the minimum process knowledge requirement to be able to perform process representation is a process flow diagram (PFD). The PFD shows the material transformations that taking place, the unit operations responsible for the material transformations, the locations of the measured and manipulated variables, and the locations of the faults in the fault library. Based on the PFD of the TEP (Chiang et al., 2000), the process condition may be represented as in Figure 4.

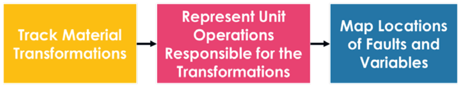

Track Material Transformations
Represent Unit Operations Responsible for the Transformations
Map Locations of Faults and Variables

Figure 3. Process Representation Workflow

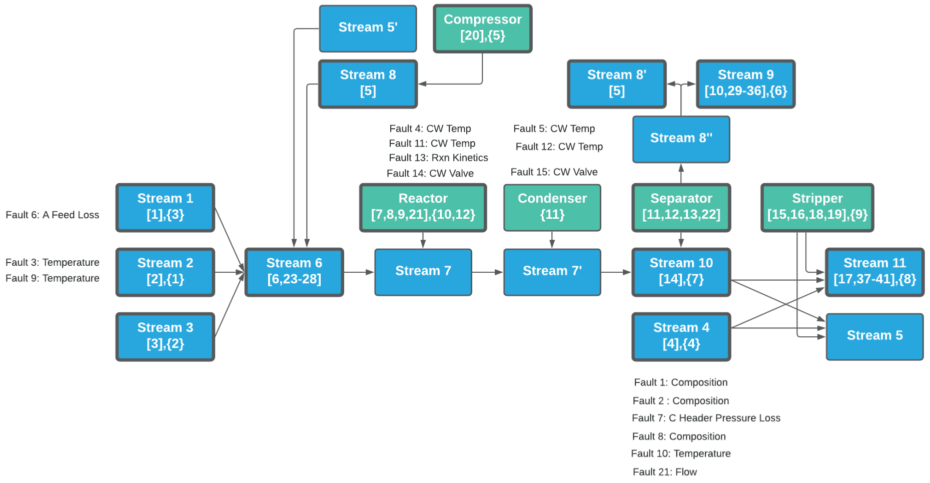


Figure 4. Process Condition Representation of the TEP depicting material and unit operation condition as blue and green nodes respectively. (The numbers in straight brackets are the designated numbers of the measured variables, and the numbers in curly brackets are the designated numbers of the manipulated variables)

2.1.2. Process Modularization

Once the process representation is available, it is now possible to modularize the process—i.e., break it down into smaller sections. Although there would be countless ways to do this, there are logical limitations to this combinatorics problem. First, a module must contain at least one fault. Without a fault, there is no condition monitoring job. It is thus interesting to see in Figure 4 that the faults are not dispersed throughout the process but are concentrated on certain nodes. With this first constraint, one can see that the condition representation of the TEP can be broken down into five modules, where the nodes that contain the fault can be considered the central node of the module.

Nodes directly adjacent to the central node may be included in the module, especially if the central node does not have input variables for ML model development. If these secondary nodes, do not have any measured variables, then other nodes that are directly adjacent to the secondary nodes may be added to the module. This process may be repeated until the added node has a measured variable or if there are no more nodes to add. This methodology is represented in Figure 5, and is consistent with the concept of d-separation in probabilistic graphical modeling, where observations in one probabilistic variable/node removes the probabilistic relationship between the parent and the child of the observed node.(Bishop and Nasrabadi, 2006) Hence, if a measured variable is involved in an added node, adding a node that is conditionally dependent on that measured variable becomes unnecessary since it gives you no further information about the central node that involves the faults for the module.

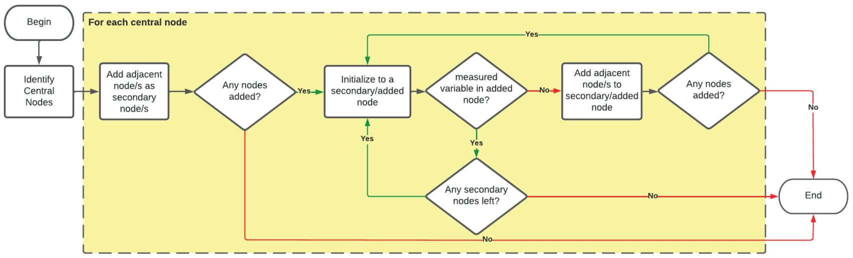
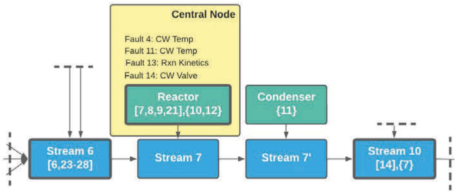

Begin
Identify Central Nodes
For each central node
Add adjacent node/s as secondary node/s
Any nodes added?
Yes
Initialize to a secondary/added node
measured variable in added node?
No
Add adjacent node/s to secondary/added node
Any nodes added?
Yes
No
Yes
Yes
Any secondary nodes left?
No
End
No

Figure 5. Methodology for modularization of process condition representation



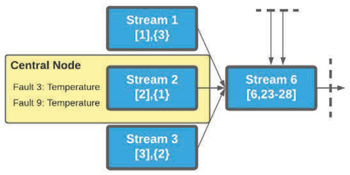


Figure 6. Reactor Condition Module Figure 7. Stream 2 Condition Module

Another noteworthy implication of the workflow in Figure 5 is that only the central nodes are mutually exclusive among the modules, and the measured/manipulated variables in the secondary nodes may be shared. An example of a module based on the reactor condition as the central node is shown in Figure 6 and a module based on the feed stream condition is shown in Figure 7. Notice that both modules have “Stream 6” condition node as a secondary node. The dashed lines indicate the conditional independence of the central node to the rest of the condition nodes in the process condition representation.

2.1.3. Machine Learning Model Development Workflow

After the modularization step, ML classifiers are developed using the Model Builder feature by the ML.NET application. Model Builder can take the labelled input variables from each condition module to explore different traditional machine algorithms (i.e., not neural networks) and recommend the best one based appropriate classification metrics.(Microsoft, 2022)

2.2. Performance Metrics for Condition Monitoring

In this study, condition monitoring performance will be determined using five metrics: fault detection rate, false alarm rate, accuracy, normal condition certainty index (NCCI), and the overall prediction certainty index (OPCI). Fault detection rate, false alarm rate, and accuracy are standard metrics used for evaluating classifiers(Fawcett, 2006) and fault detection algorithms for the TEP.(Yin et al., 2012)

However, the two certainty indices are new metrics that is a unique contribution of this paper. These indices assume a threshold and work with classifiers (machine learning classification algorithms) that produces a probability for each possible condition in a module. For such classifiers, the condition with the highest assigned probability is considered the prediction condition and its assigned probability can be considered its prediction certainty. If the OPCI for a classifier is 0.95 for a threshold of 0.90, this means

95 out of 100 predictions made had a prediction probability/certainty higher than 0.90. The NCCI works similarly, but only considers the normal condition predictions. These certainty indices reflect the confidence of the predictions by the machine learning algorithm, which the operator can use to evaluate classifier performance. Interestingly, the NCCI has implications for novel fault detection capabilities. If the NCCI is close to 1.0, then future classification predictions that are lower than the threshold could be labelled as novel faults that would require further action by a human operator. Formulas for the OPCI and NCCI are shown below.

$$OPCI = \frac{\sum Predictions|_{Prediction\ Probability > Threshold}}{\sum Predictions} \qquad \text{Equation 1}$$

$$NCCI = \frac{\sum Normal\ Condition\ Predictions|_{Prediction\ Probability > Threshold}}{\sum Normal\ Condition\ Predictions} \qquad \text{Equation 2}$$

3. Results and Discussion

To evaluate the effectiveness of the framework, a base case is considered where a machine learning model is developed to use all 53 of the input variables in the TEP for predicting all 22 possible conditions (i.e., normal condition plus 21 fault types). With the framework applied, the ML development task is much simpler after modularization; for the case of the reactor module in Figure 6, the classifier only needs to classify 5 conditions (i.e., normal condition plus 4 fault types) from 16 input variables.

To determine the impact of the d-separation concept in the modularization workflow, a modified case of the framework was considered so that for each module, all the TEP variables would be used as input to determine the faults that are local to the central node of the module. For the reactor module in Figure 6, this would be using all 53 input variables to classify the 5 conditions considered in the module. This results in three cases that can be considered when evaluating the CM performance of each module. The details of these three cases when evaluating the reactor module is summarized in Table 1.

Table 1. Reactor Module Comparison Cases

	Base Case	**Modular Case**	**Modified Modular Case**
No. of Faults	21	4	4
No. of Input Variables	53	4	53

Table 2. Performance Summary for Reactor Module Faults

	Base Case	**Modular Case**	**Modified Modular Case**
Fault Detection Rate (%)	99-100	88-99	80-100
False Alarm Rate (%)	19-67	0-1	0-2
Accuracy (%)	33-99	89-100	94-99
NCCI	0.57-0.70	0.97-1.00	0.98-1.00
OPCI	0.33-0.89	0.95-1.00	0.92-1.00

For the faults local to the reactor module, the base case performance can be summarized in Table 2. While the fault detection rate is perfect, the other metrics, particularly the high rates of false alarms, make it unusable for condition monitoring. On the contrary, the modular cases show better performance across all metrics, with negligible false alarm rates and high certainty indices which indicate novel fault detection capabilities. For brevity, the performance summaries of the other modules would not be explicitly shown in this paper, albeit the aforementioned trends still hold for other modules. Overall, there seem to be no significant difference between the two modular cases, except for some faults like fault 5 which saw a lower false alarm rate by 14% compared to the modified modular case.

4. Conclusions

The condition monitoring system development framework that was originally developed for a continuous pharmaceutical tableting line proved to be effective for the Tennessee Eastman Process. The process representation workflow allowed the incorporation of process knowledge into machine learning model development. The ensuing modularization of the process representation simplified the machine learning model development task, which resulted in high performance classifiers that have novel fault detection potential.

For the impact of applying d-separation in the modularization workflow, it was not significant across all faults, although it yielded major improvements for some. Ultimately, this case study validates the applicability of the development framework across two different kinds of continuous manufacturing processes and suggests that it could be effective for other kinds of continuous processes.

5. Acknowledgements

The authors acknowledge Dr. Ziyan Sheriff for sharing valuable insights and knowledge about the Tennessee Eastman Process, which proved valuable in shaping the direction of this research.

This work was supported by the NSF under grant #2140452.

References

Bishop, C.M., Nasrabadi, N.M., 2006. Pattern recognition and machine learning. Springer.
Chiang, L.H., Russell, E.L., Braatz, R.D., 2000. Fault detection and diagnosis in industrial systems. Springer Science & Business Media.
Downs, J.J., Vogel, E.F., 1993. A plant-wide industrial process control problem. Comput Chem Eng 17, 245–255.
Fawcett, T., 2006. An introduction to ROC analysis. Pattern Recognit Lett 27, 861–874.
Lagare, R.B., Nagy, Z., Reklaitis, G. v., 2022a. Applying process knowledge for more powerful data-driven condition monitoring systems. Under preparation.
Lagare, R.B., Sheriff, M.Z., Gonzalez, M., Nagy, Z., Reklaitis, G. v., 2022b. A Comprehensive Framework for the Modular Development of Condition Monitoring Systems for a Continuous Dry Granulation Line. Computer Aided Chemical Engineering 49, 1543–1548.
Microsoft, 2022. What is ML.NET and how does it work? [WWW Document]. URL https://learn.microsoft.com/en-us/dotnet/machine-learning/how-does-mldotnet-work?WT.mc_id=dotnet-35129-website (accessed 12.1.22).
Schenkendorf, R., 2016. Supporting the shift towards continuous pharmaceutical manufacturing by condition monitoring. Conference on Control and Fault-Tolerant Systems, SysTol 2016-November, 593–598.
Yin, S., Ding, S.X., Haghani, A., Hao, H., Zhang, P., 2012. A comparison study of basic data-driven fault diagnosis and process monitoring methods on the benchmark Tennessee Eastman process. J Process Control 22, 1567–1581.