17th European Symposium on Computer Aided Process Engineesing – ESCAPE17
V. Plesu and P.S. Agachi (Editors)
© 2007 Elsevier B.V. All rights reserved.

Multi-agent Framework for Fault Detection & Diagnosis in Transient Operations

Ng Yew Seng[a] and Rajagopalan Srinivasan[a,b]

[a]Department of Chemical and Biomolecular Engineering, National University of Singapore, 10 Kent Ridge Crescent, Singapore 119260.
[b]Process Sciences and Modeling, Institute of Chemical & Engineering Sciences, 1 Pesek Road, Jurong Island, Singapore 627833. E-mail: {chenys, chergs}@nus.edu.sg

Abstract
With the increasing emphasis on agile operations, the process industries have begun to focus on effectively managing transient operations such as transitions or batch/fed-batch processes. In this paper, we propose a multi-agent based decision fusion framework for monitoring and diagnosing faults during transitions. The proposed method integrates three fault diagnosis methodologies into a uniform and coordinated manner where collaboration among heterogeneous methods is enabled to achieve optimality in speed and accuracy of fault detection. We illustrate the efficacy of the proposed approach through a pilot-scale distillation unit startup case study.

Keywords: Transitions, Bayesian fusion, fault diagnosis, hybrid framework, software agent.

1. Introduction to Transient Operations

Increasingly, manufacturing facilities operate at a multitude of states and frequently switch between them. The switch from one state to another is termed as a process transition. Plant startups and shutdowns are common examples of transitions in the process and allied industries including refining, petrochemicals, paper & pulp, steel, and cement manufacture. Other transitions occur due to feedstock, throughput, or product slate changes as well as maintenance operations such as furnace decoking or absorber regeneration.

Transitions are also common in high-value added specialty and pharmaceutical plants which commonly operate in batch and fed-batch phases.
From a monitoring perspective, transitions correspond to discontinuities in the plant operation such as change of setpoints, change of equipment configuration, turning on or idling of equipments, etc. Hybrid discrete-continuous behavior of the process therefore has to be considered when monitoring transitions. Multi-time scale effects also become important where some variables change quickly (order of seconds) and others respond over hours. Though there exist some approaches in the literature for fault detection and diagnosis during transient mode of operations, i.e., see Qin (2003) [1], Chen and Liu (2002) [2], Ng and Srinivasan (2004) [3], the performance of most of these approaches is process dependant and inadequate in many instances. In this paper, using examples from a distillation-unit startup, we demonstrate a practically implement-able multi-agent framework that performs effectively in a broad range of applications.

2. Multi-agent Approach for Collaborative FDI

Since each FDI method exhibits strengths and shortcomings that are process dependant, collaboration among heterogeneous methods is needed to bring forth the benefits of each FDI method, so that monitoring resolution and robustness of the FDI system can be brought to higher ground. Towards this end, each FDI method can be represented as an agent and the diagnostic results from multiple agents combined. Through this, the strengths of the different methods can be integrated while the drawbacks of the individual methods diminished through *collaboration*. Another key benefit of the multi-agent framework is that it enables efficient computational integration of high fidelity and complex FDI methods. These two aspects are the focus of this paper here

2.1. Bayesian Fusion for Collaboration among FDI agents

When multiple FDI agents are used in parallel, a conflict resolution strategy is needed to arbitrate among the contradictory decisions generated by the various FDI methods so that one consolidated solution can be presented to the plant personnel. In this paper, a Bayesian approach is used for this purpose. In general, the predictions from all diagnostic agents, A_r, can be recorded in a confusion matrix as follows [4]:

$$CM^r = \begin{pmatrix} n_{11}^r & n_{12}^r & \cdots & n_{1(J+1)}^r \\ n_{21}^r & n_{22}^r & \cdots & n_{2(J+1)}^r \\ \vdots & \vdots & \ddots & \vdots \\ n_{J1}^r & n_{J2}^r & \cdots & n_{J(J+1)}^r \end{pmatrix} \qquad \text{(Eqn. 1)}$$

where n_{ij}^r, $i = [1\ J]$ and $j = [1\ J+1]$, indicates the number of samples belonging to class C_i that have been assigned to fault class j by Agent A_r. The

diagonal elements of CM^r are then the correct predictions from A_r. The confusion matrix is computed for each FDI during the offline training stage.
The information in CM^r is utilized in real-time as follows: the conditional probability that implies $\boldsymbol{x} \in C_j$ given the evidence $A_r(\boldsymbol{x}) = j$ can be computed as [4]:

$$P(\boldsymbol{x} \in C_j \mid \kappa_r(\boldsymbol{x}) = j) = \frac{n^r_{ij}}{n^r_{\cdot j}} = \frac{n^r_{ij}}{\sum_{i=1}^{M} n^r_{ij}}, \; i \in [1 \;\; J]. \qquad \text{(Eqn. 2)}$$

$P(\boldsymbol{x} \in C_j \mid \kappa_r(\boldsymbol{x}) = j)$ can be considered the confidence of a classifier regarding the assignment of sample $\boldsymbol{x}$ to class C_j. When R FDI agents are involved, each has its own confusion matrix, CM^r, $r = [1 \;\; R]$, and R evidences are produced in real-time, $e = e_1(x), ..., e_R(x)$. Each FDI agent, A_r, expresses its predictions supporting the proposition that $\boldsymbol{x} \in C_j$ in the form of conditional probability. The combined probability P_E that supports $\boldsymbol{x} \in C_j$ from the collaboration among the agents can then be written as:

$$P_E(\boldsymbol{x} \in C_j \mid e_r(\boldsymbol{x}) = j_r, EN) = P(\boldsymbol{x} \in C_j \mid e_1(\boldsymbol{x}) \cap ... \cap e_R(\boldsymbol{x}) = j)$$
$$= \frac{\prod_{r=1}^{R} P(\boldsymbol{x} \in C_j \mid e_r(\boldsymbol{x}) = j)}{\sum_{i=1}^{J} \prod_{r=1}^{R} P(\boldsymbol{x} \in C_j \mid e_r(\boldsymbol{x}) = j)} \qquad \text{(Eqn. 3)}$$

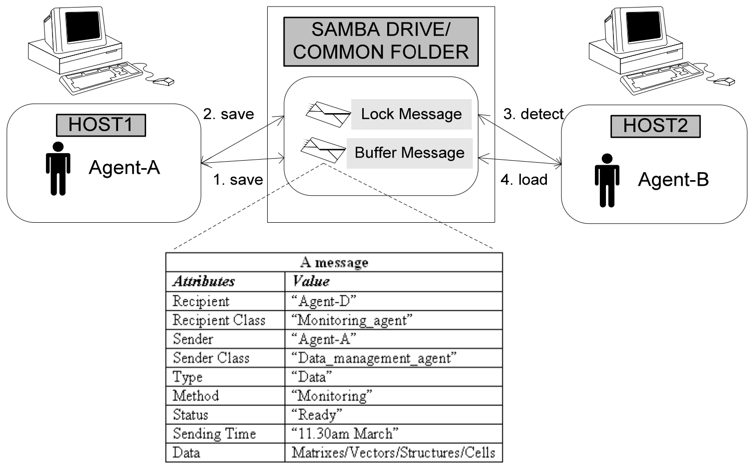
Based on Eqn. 3, a sample $\boldsymbol{x}$ is classified into class j depending on the combined conditional probability. The class C_j with the highest P_{E_j} can be selected as the optimal combined prediction:

$$E(\boldsymbol{x}) = \begin{cases} j, & \textit{if } j = \arg\max\limits_{j \in \Lambda}(P_{E_j}) \\ J+1, & \textit{otherwise} \end{cases} . \qquad \text{(Eqn. 4)}$$

Here, class *J+1* flags a novel fault.

2.2. Inter-Agent Communication in a multiprocessor environment

Since most FDI algorithms are computationally expensive, a parallel implementation of the framework is essential for decision support in real-time. To realize this in practice, our agents use Message Passing Interface (MPI) for inter-agent communication [5]. So, the agents can be executed from various distributed processors (either distributed-memory or shared-memory systems). Agents communicate with each other by exchanging messages based on a purpose-designed ontology. A knowledge-base approach based on string recognition is used for ontology synchronization, where each class of agent has a vocabulary (list of strings) containing the queries the agent is capable of responding to. An agent can thus communicate with another by sending it appropriate queries within the latter's vocabulary. An example of inter-agent communication is shown in Figure 1.



A message	
Attributes	***Value***
Recipient	"Agent-D"
Recipient Class	"Monitoring_agent"
Sender	"Agent-A"
Sender Class	"Data_management_agent"
Type	"Data"
Method	"Monitoring"
Status	"Ready"
Sending Time	"11.30am March"
Data	Matrixes/Vectors/Structures/Cells

Figure 1: Inter-host message passing among heterogeneous FDI agents

Suppose there are two agents, A^a and A^b, where A^a intends to request A^b to perform a FDI task with task related information in dataset $\boldsymbol{x}_i$. The MPI message passing between A^a and A^b functions using a common communications folder. A^a first writes the message $\Psi^{A,buffer}$, which contains data (information) for the task to be executed by A^b, to the common folder. After the buffering of $\Psi^{A,buffer}$ is completed, it creates a semaphore message, $\Psi^{A,lock}$. The agent A^b continually checks the common folder for existence of semaphore $\Psi^{A,lock}$. Once $\Psi^{A,lock}$ is detected, A^b loads the message $\Psi^{A,buffer}$ and executes the requested tasks based on the data in $\Psi^{A,buffer}$. With this message passing architecture, the agents can communicate freely across different hosts in a distributed agent-environment. One key benefit of distributed multi-agent framework is speedup of computationally demanding tasks by exploiting multiple processors. Two performance measurement indicators are used in this work to measure the benefit of speedup, namely, *overall speed enhancement index*, S_p, and the *overall system efficiency index*, E_p.

$$S_p = \frac{\text{Average time required for single machine}}{\text{Average time required for p machines}} \qquad \text{(Eqn. 5)}$$

$$E_p = \frac{\text{Speed up with } p \text{ processors}}{p} \qquad \text{(Eqn. 6)}$$

3. Case study

In this section, the proposed multi-agent framework is tested using a pilot-scale distillation unit startup case study. We illustrate the benefits through integration of three FDI methods – self-organizing map, SOM [6], nonparametric kernel density estimation, KDE approaches [7], and neural-networks, NN [8].

The distillation-column considered in this case study is of 2 meters height and 30 cm width. It is integrated with a control console and a data acquisition system. Cold startup of the distillation column with ethanol-water at 30% v/v mixture is performed based on a predefined standard operating procedures. All data used for training and testing of the MAS can be obtained from the distillation-unit case study homepage [9]. The multi-agent system was tested on a Linux cluster with 16 nodes (each containing 2 Intel® Pentium^{TM} Xeon 3.06GHz processors with 2GB memory).

3.1. Results & Discussions

DST05 corresponds to a sensor fault. The fault was introduced at *t*=4250s when the process approaches steady-state. Throughout this fault, all variables remain normal except for the affected sensor (Tray 6 Temperature Sensor). In this case study, the KDE agent, A^m_{KDE}, fails to detect DST05. All faults are successfully detected and isolated based on the proposed multi-agent system with an average detection and diagnostic delay of 57.9 samples and 58.4 samples respectively. Average diagnostic delay based on single FDI approach is 110.6 samples (SOM), 86.9 samples (KDE), and 85.7 samples (NN). A minimum improvement of at least 31.86% is achieved in fault identification time by combining these three FDI methods compared to any one FDI method. The S_p and E_p observed using multiple processors are shown in Figure 2. The proposed MAS has been able to achieve a speedup of ~4.4x. Average processing time has been reduced from ~12s per sample to ~2.7s per sample during abnormal events (when high level of CPU flops are required).

Table 2: Performance evaluation of each FDI agent and the proposed multi-agent approach (with best performance highlighted)

FDI methods	DSTs Identified	Recognition Rate	Avg Detection Delay	Avg Diagnosis Delay
Self Organizing Map	10	56.21	68.3	110.6
Kernel Density Est.	9	66.32	67.9	86.9
Neural Network	10	87.15	58.0	85.7
Bayesian Combination	10	89.80	57.9	58.4

4. Conclusions

Fault diagnosis during transitions is challenging due to various complexities. By combining all three FDI agents (SOM, KDE and NN), the multi-agent approach is able to diagnose all ten disturbances with significant improvement in all criteria measured, i.e., recognition rate (↑3.04%), speed of fault detection

(↑3.27%) and diagnostic delay (↑31.9%), compared to a solitary FDI method (SOM, KDE or NN). The agent architecture can be naturally deployed on multiple processor clusters for computational performance improvements.

References

1. Qin, S. J., (2003). Statistical process monitoring: basics and beyond, Journal of Chemometrics 17, pp. 480-502.
2. Chen, J., and Liu, K.C., (2002). On-line batch process monitoring using dynamic PCA and dynamic PLS models, Chemical Engineering Science 57, pp. 63-75.
3. Ng, Y.S, and Srinivasan, R., (2004). Transitions in the process industries: Opportunities and prospective solution, IEEE International Symposium on Intelligent Control, Taipei, Taiwan, pp. 246-251.
4. Xu, L., Krzyżak, A., Suen, C.Y., (1992). Methods of combining multiple classifiers and their applications to handwriting recognition, IEEE Transactions on Systems, Man, and Cybernatics 22(3), pp. 418-435.
5. Kepner, J., and Ahalt, S., (2004). MatlabMPI, Journal of Parallel and Distributed Computing, 8(64), pp. 997-1005.
6. Ng, Y. S. and R. Srinivasan, (2004). Monitoring of distillation-column operation through Self-organizing Map, 7th International Symposium on Dynamics and Control of Process System (DYCOPS), Massachusetts, USA, July 5 – 7, 2004.
7. Martin, E.B., and Morris, A.J., (1996). Non-parametric confidence bounds for process performance monitoring charts, Journal of Process Control 6, pp. 349-358.
8. Srinivasan, R., Wang. C., Ho, W.K., and Lim, K.W., (2005). Neural network systems for multi-dimensional temporal pattern classification, Computers & Chemical Engineering 29, pp. 965-981.
9. Ng, Y.S., and Srinivasan, R., (2005). Distillation unit case study homepage, iACE-Laboratory, Singapore, http://www.iace.eng.nus.edu.sg/research/Distillation_column/index.htm, National University of Singapore.

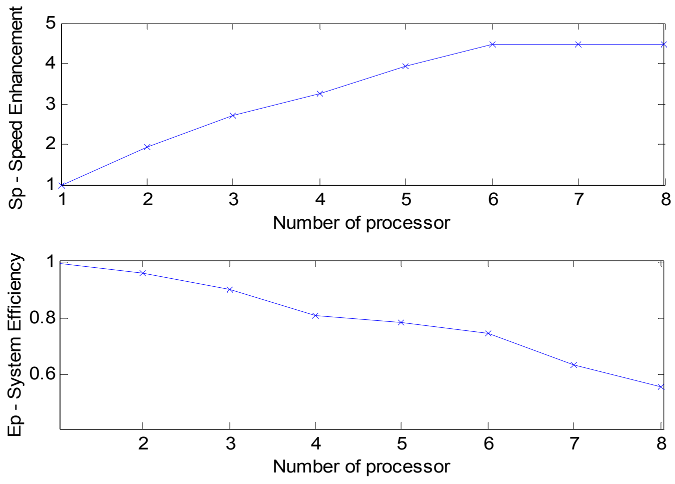


Figure 2: Speed enhancement and system efficiency measured on a Linux cluster