Journal of Process Control 22 (2012) 738–750

Contents lists available at SciVerse ScienceDirect

Journal of Process Control

journal homepage: www.elsevier.com/locate/jprocont

Decentralized fault detection and diagnosis via sparse PCA based decomposition and Maximum Entropy decision fusion

Mihajlo Grbovic[a,*], Weichang Li[b], Peng Xu[b], Adam K. Usadi[b], Limin Song[b], Slobodan Vucetic[a]

[a] Department of Computer and Information Sciences, Temple University, Philadelphia, PA 19122, USA
[b] Corporate Strategic Research, ExxonMobil Research and Engineering Company, Annandale, NJ 08801, USA

ARTICLE INFO

Article history:
Received 2 April 2011
Received in revised form 6 February 2012
Accepted 7 February 2012
Available online 16 March 2012

Keywords:
Fault detection
Fault diagnosis
Decentralized process monitoring
Decision fusion

ABSTRACT

This paper proposes an approach for decentralized fault detection and diagnosis in process monitoring sensor networks. The sensor network is decomposed into multiple, potentially overlapping, blocks using the Sparse Principal Component Analysis algorithm. Local predictions are generated at each block using Support Vector Machine classifiers. The local predictions are then fused via a Maximum Entropy algorithm. Empirical studies on the benchmark Tennessee Eastman Process data demonstrated that the proposed decentralized approach achieves accuracy comparable to that of the fully centralized approach, while offering benefits in terms of fault tolerance, reusability, and scalability.

1. Introduction

Timely fault detection in complex manufacturing systems is critical to ensure safe and effective operation of plant equipment. The quantifiable objectives of fault detection include achieving low detection delay time, low false positive rate, and high detection rate. Early detection provides invaluable warning on emerging problems to avoid catastrophic consequences. Low false positive rate ensures the usability of the detection system. High detection probability is an essential requirement for successful detection and tracking of fault events. These performance metrics define an overall quality of any fault detection system, although tradeoff between detection speed and accuracy highly depends on specific applications. In general, data-driven fault detection algorithms can be categorized into unsupervised and supervised ones. In unsupervised algorithms, such as Principal Component Analysis (PCA) [4,11,14] and Independent Component Analysis (ICA) [13,15], normal operation is modeled and faults are detected as deviations from the normal behavior. In supervised algorithms, such as Support Vector Machines (SVMs) [12], AdaBoost [48] and Neural Networks (NN) [11], a binary classifier is trained on historical data containing both normal and faulty conditions and then used to predict faults.

Once a fault has been detected, diagnosis of its type is conducted for the purpose of fault mitigation and returning to normal operation. Fault diagnosis is effectively a multi-class classification problem of predicting a specific fault type given a pool of known alternatives. Various machine learning algorithms, such as Fisher Discriminant Analysis [3,18], PCA [30,31], SVM [18,17], Learning Vector Quantization (LVQ) [21] and Neural Networks [16] have been used for this purpose. In this paper, the term fault classification is used to describe supervised methods for joint fault detection and diagnosis.

An alternative to the data-driven fault detection and diagnosis approach is the model-driven approach [33] that requires accurate process modeling by semi-quantitative or qualitative models.

For small-scale sensor networks, it is reasonable to assume that all sensor measurements are sent to some central location where fault predictions are made. This is known as a centralized fault detection and diagnosis approach. For large scale networks, due to various constraints, such as communication bandwidth, the decentralized approach is often used [28], where network is decomposed into potentially overlapping blocks and each block provides local decisions that are fused at the central location. The appealing properties of the decentralized approach include fault tolerance, reusability and scalability. For example, when one or more blocks go offline due to maintenance of their sensors, the predictions can still be made using remaining sensors. In addition, when the physical facility is reconfigured, either by changing its components or sensors, it can be easier to modify part of the decentralized system impacted by the changes than to overhaul the whole centralized system. Finally, the scalability comes from reduced costs of system setup and update, communication, and decision making.

Decentralized process monitoring has been subject of active research in recent years [19,47,49]. Main challenges include correct

* Corresponding author.
E-mail address: mihajlo.grbovic@temple.edu (M. Grbovic).

0959-1524/$ – see front matter © 2012 Elsevier Ltd. All rights reserved.
doi:10.1016/j.jprocont.2012.02.003

Nomenclature

N	number of samples in data set
K	number of sensors in the system
B	number of blocks
$\mathbf{x}_i$	input row vector of all K sensors data at time i, $\mathbf{x}_i^k \in R^{1\times d}$
$\mathbf{X}$	input data matrix $\mathbf{X} \in R^{N\times d}$
$\mathbf{x}_i^b$	b-th block input row vector at time i, $\mathbf{x}_i^k \in R^{1\times d_b}$
$\mathbf{X}_b$	b-th block data matrix $\mathbf{X}_b = [(\mathbf{x}_1^b)^T, (\mathbf{x}_2^b)^T, \ldots, (\mathbf{x}_N^b)^T]^T$
$\mathbf{x}_i^k$	k-th sensor input data row vector at time i, $\mathbf{x}_i^k \in R^{1\times d_k}$
l	number of lagged variables
y_i	output (system state) at time i
$\hat{y}_i$	prediction of system state at time i
$\hat{\mathbf{y}}_i$	vector of B local predictions at time i, $\hat{\mathbf{y}}_i = [\hat{y}_i^1, \hat{y}_i^2, \ldots, \hat{y}_i^B]$
$\hat{\mathbf{y}}^b$	b-th block prediction at time i (hard, e.g. $\hat{\mathbf{y}}^b = 4$ or soft)
$\hat{y}_{i(c)}^b$	b-th block soft prediction for fault c
C	number of process faults
$\mathbf{C}$	set of all known faults $\mathbf{C} = \{1, \ldots, C\}$
$\mathbf{C}_b$	set of faults detectable by the b-th block
$\mathbf{S}$	covariance matrix of $\mathbf{X}$
$\mathbf{V}$	eigenvectors of $\mathbf{S}$, $\mathbf{V} = [\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_d]$
$\mathbf{\Lambda}$	eigenvalues of $\mathbf{S}$ corresponding to $\mathbf{V}$
a	number of most dominant eigenvectors
$\mathbf{V}^b$	b-th block eigenvectors from local data matrix $\mathbf{X}_b$
$\mathbf{\Lambda}^b$	local eigenvalues corresponding to $\mathbf{V}^b$
$\mathbf{U}$	sparse eigenvector matrix $\mathbf{U} = \mathbf{v}\mathbf{v}^T$
r	allowed number of non-zero eigenvector elements (Sparse PCA)
f_m	m-th feature function (rule)
ω_m	m-th feature function weight (confidence)
π	parameter controlling the strength of regularization

process decomposition and decision fusion. Two types of process decomposition have been considered: (1) a completely decentralized decomposition [37,39,41], in which each sensor is a separate decision maker, and (2) a multi-block decentralized decomposition, in which sensors are grouped into meaningful overlapping [35] or disjoint [19] blocks. For example, in several recent papers on multi-block decentralized models, the decomposition was based on the process topology [19,25,27–29,40,42] or prior knowledge [26,35].

Various local decision fusion strategies have been investigated. In some cases the system declares a fault if it is observed at any location [19,27,29,42], without requiring explicit decision fusion. In most cases of decentralized detection and diagnosis, however, this is done by combining the received binary [6,37,39,45], multi-class [43] or continuous [38,44,46] messages from local detectors using different fusion strategies. Most popular multi-class decision fusion strategies are described in a survey paper [36]. For instance, in voting-based fusion, the winner is the class with the sufficient and highest number of expert votes. In weighted voting-based fusion, each expert receives a class-specific weight proportional to its classification accuracy on the training set or a separate validation set and the class with the maximum total weight is considered the winner. The survey [36] also covers Bayesian-based fusion and Dempster–Shafer decision fusion.

In this paper, we propose a decentralized fault detection and diagnosis model based on Sparse PCA process decomposition, local SVM classifiers and Maximum Entropy (MaxEnt) [23,24] decision fusion. The sensors are partitioned into small and potential overlapping blocks based on the Sparse PCA [32] algorithm which preserves strong correlations among sensors. Given user generated rules for decision making, the MaxEnt algorithm allows efficient learning of their importance from data, which leads to more consistent fusion. The proposed model was compared to several other process decomposition and decision fusion strategies. The evaluation was performed using data from the benchmark Tennessee Eastman Process [2].

2. Preliminaries

2.1. Problem setup

We consider a plant monitored by a network of K sensors providing measurements synchronously in regular time intervals. At time i, state of the k-th sensor is represented by a row vector of variables $\mathbf{x}_i^k$. There are many ways to construct this vector. For example, one can use only the raw measurement at time i, the set of raw measurements at the most recent l time steps, or derive variables from the current and recent raw measurements. Combining all K sensors, we have row vector $\mathbf{x}_i = [\mathbf{x}_i^1, \mathbf{x}_i^2, \ldots, \mathbf{x}_i^k]$. We denote with d the length of the resulting vector. The process condition at time i is denoted with class label $y_i \in \{0, 1, \ldots, C\}$, where 0 represents normal condition, and values 1 to C present one of the C potential faults. We assume that N historical observations are collected in form of a data set $D = \{(\mathbf{x}_i, y_i), i = 1, \ldots, N\}$.

In fault detection, all faulty classes are treated as a single class and the resulting problem is analogous to binary classification. In fault diagnosis, the problem is multi-class, with $C+1$ classes. Therefore, fault detection can be considered as a special case of fault diagnosis. We denote by $\hat{y}_i$ the prediction of the process state at time i.

Centralized approach: All observations $\mathbf{x}_i = [\mathbf{x}_i^1, \mathbf{x}_i^2, \ldots, \mathbf{x}_i^k]$ are available at the central location where fault classification is performed. In this case, the objective is to train a single classification function h: $\mathbf{x}_i \rightarrow y_i$ from data set $D = \{(\mathbf{x}_i, y_i), i = 1, \ldots, N\}$.

Decentralized approach: The sensor network is partitioned into B potentially overlapping blocks, with the b-th block containing K_b sensors. Let us denote $\mathbf{x}_i^b$ as the input vector for the b-th block at time i. For example, in the completely decentralized scenario there are K blocks, each containing variables from a single sensor ($K_b = 1$). Let us denote by $\hat{y}_i^b$ the local classification for the b-th block at time i. Local fault classifier for the b-th block, h_b: $\mathbf{x}_i^b \rightarrow y_i$, is trained using local data set $D_b = \{(\mathbf{x}_i^b, y_i), i = 1, \ldots, N_b\}$ that is a subset of D. The local decisions at time i $\hat{y}_i^b$, $b = 1, \ldots, B$, are transmitted to the fusion center where they are combined into a single, final decision $\hat{y}_i$. The decision fusion model is trained using historical data of the form $F = \{(\hat{\mathbf{y}}_i, y_i), i = 1, \ldots, N\}$, where y_i is a true class label at time i and $\hat{\mathbf{y}}_i$ is a vector containing B local decisions, $\hat{\mathbf{y}}_i = [\hat{y}_i^1, \hat{y}_i^2, \ldots, \hat{y}_i^B]$, at time i.

2.2. Performance measures

To evaluate performance of fault classification models trained using D, we assume that a set of labeled observations D_{test}, disjoint from D, is available consisting of both normal and faulty conditions. The data set D_{test} is assumed to contain J fault occurrences. Fig. 1 summarizes the measures used to evaluate performance of a fault classification model. The *true positive rate*, *TPR*, is defined as

$$\mathrm{TPR} = \frac{n_1}{N_1} \cdot 100, \tag{1}$$

where n_1 is the number of correctly classified faulty samples and N_1 is the total number of faulty samples in D_{test}.

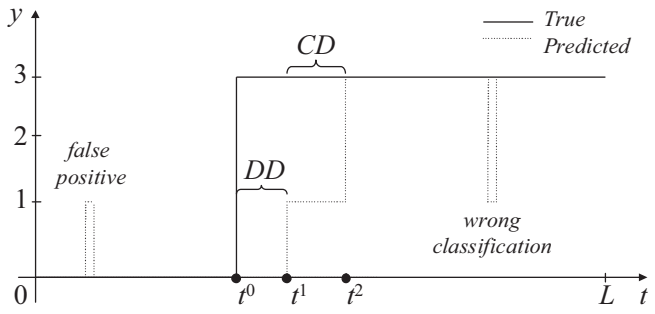

Fig. 1. Subsample of D_{test} that contains a faulty event j: true vs. predicted.

The *false positive rate*, FPR, is given as

$$\text{FPR} = \frac{n_0}{N_0} \cdot 100, \tag{2}$$

where n_0 is the number of misclassified normal data samples from D_{test} and N_0 is the total number of normal samples in D_{test}.

We define the *detection delay* DD_j for the j-th faulty occurrence from D_{test} as the delay between t_j^0, the introduction time of the fault, and its detection time t_j^1, i.e.

$$DD_j = t_j^1 - t_j^0. \tag{3}$$

The average detection delay DD for all fault occurrences in D_{test} is defined as

$$DD = \sum_{j=1}^{J} \frac{DD_j}{J}. \tag{4}$$

In addition to fault detection delay, we also consider the delay between fault detection and correct classification. We define the *classification delay* CD_j for the j-th faulty occurrence from D as the delay between its detection time t_j^1 and its correct classification time t_j^2,

$$CD_j = t_j^2 - t_j^1. \tag{5}$$

The average classification delay CD for all fault occurrences in D_{test} is

$$CD = \sum_{j=1}^{J} \frac{CD_j}{J}. \tag{6}$$

In general, $CD_j \neq 0$, as it takes a certain time for a fault to develop its unique pattern differentiable from those of other faults. These four metrics are all critical performance indicators of a fault detection and diagnosis model.

In Fig. 1, a subset of the training data set D_{test} used to evaluate the above performance metrics is shown. The flat line is the true label, and the dotted line is an example fault classification. The figure illustrates the detection delay and classification delay intervals, as well as examples of false positive and false negative predictions.

3. Centralized fault detection and diagnosis

This section reviews several typical approaches for centralized fault detection and diagnosis. They will be used for comparison with the proposed decentralized solutions. In addition, the techniques reviewed here will be used to construct the local classifiers in our decentralized model. A good review of the related work can be found in [1]. Fig. 2 shows a typical block diagram for a centralized fault detection and diagnosis model. In the following, we will overview both unsupervised and supervised centralized methods.

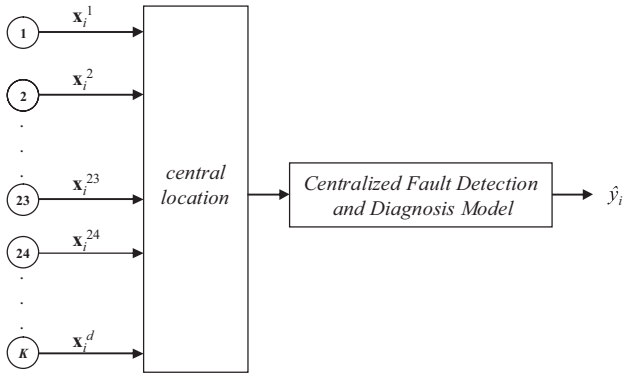

Fig. 2. Centralized fault classification approach.

3.1. Unsupervised fault detection methods

In the unsupervised approach, faults are detected as observations that deviate from the modeled normal behavior. Some of the popular unsupervised fault detection methods include PCA and ICA-based approaches. They are threshold-based approaches that input the observation vector and output a single number representing deviation from the normal behavior. The threshold is typically set such that the false positive rate is small (in the range between 1% and 5%).

The PCA-based approach [5] has been extensively studied and employed for fault detection. PCA projects the observation to a lower dimensional subspace, which describes most of the variance of the normal data. Given training data consisting of N normal condition observations, let us define $N \times d$ matrix $\mathbf{X} = [\mathbf{x}_1^T, \mathbf{x}_2^T, \ldots, \mathbf{x}_N^T]^T$, where $\mathbf{x}_i$ is the i-th observation d-dimensional row vector. Then, the principal components are found using eigenvalue decomposition of the covariance matrix $\mathbf{S}$,

$$\mathbf{S} = \frac{1}{N-1}\mathbf{X}^T\mathbf{X} = \mathbf{V}\mathbf{\Lambda}\mathbf{V}^T, \tag{7}$$

where the columns of $\mathbf{V} = [\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_d]$ are the eigenvectors of $\mathbf{S}$ and the diagonal elements of $\mathbf{\Lambda}$ are its eigenvalues. Two types of statistics were traditionally used in PCA-based process monitoring; namely, Hotelling's T^2 statistic [5] and the squared prediction error (SPE) statistic [5]. The T^2 statistic measures size of vector $\mathbf{x}_i$ in the subspace of typical behavior. More specifically,

$$T^2(i) = \mathbf{x}_i{}^T\mathbf{V}_a\mathbf{\Lambda}_a^{-1}\mathbf{V}_a^T\mathbf{x}_i, \tag{8}$$

where $\mathbf{V}_a$ is a set of the a ($a < d$) most dominant eigenvectors of $\mathbf{S}$ and $\mathbf{\Lambda}_a$ is the diagonal matrix of the corresponding eigenvalues. Parameter a is directly related to the percentage of variance retained in the normal condition data.

The SPE statistics is a measure of the amount of variation in the residual subspace, spanned by the $d - a$ smallest principal components, and is defined as

$$\text{SPE}(i) = \mathbf{x}_i^T\mathbf{V}_{d-a}\mathbf{V}_{d-a}^T\mathbf{x}_i. \tag{9}$$

In PCA-based fault detection, fault is reported if the value of T^2 statistic exceeds its threshold or the value of SPE statistic exceeds its threshold, both determined by desired FPR.

3.2. Supervised fault detection and diagnosis methods

Supervised methods, such as SVM, Neural Networks and LVQ that use historical normal and faulty conditions to train a classifier are appropriate for both fault detection [4,11,18] and fault diagnosis [12,16,17,21]. The advantage of the supervised methods is in their ability to explicitly utilize historical information about faulty

conditions, which increases their sensitivity to faults. Assuming C fault types, observed in training data D together with normal condition examples, the problem at hand is multi-class classification with $C+1$ classes. As our supervised model of choice, we describe a binary SVM classifier, followed by its extension to multi-class classification.

SVM (Support Vector Machine). Given a data set $D = \{(\mathbf{x}_i, y_i), i = 1, \ldots, N\}$, where $\mathbf{x}$ is d dimensional and y is binary, SVM classifier attempts to find the maximum-margin hyperplane that divides the examples of the opposite classes. SVM seeks the best classifier of type $g(\mathbf{x}) = w^T\Phi(\mathbf{x}) + \beta$, where Φ: $\Re^d \rightarrow H$ is a mapping from the original d-dimensional attribute space to a potentially high-dimensional attribute space H. It can be shown that maximizing the margin is equivalent to minimizing $||w||$. Thus, the problem is reduced to solving the following constrained optimization problem,

$$\begin{aligned} &\min_{w,b} \cdot \frac{1}{2}||w||^2 + \lambda\sum_{i=1}^{N}\xi_i \\ &s.t.\ \ y_i(w \cdot \Phi(x_i) + \beta) \geq 1 - \xi_i, \quad \xi_i \geq 0, \forall i \end{aligned}, \tag{10}$$

where ξ_i are the so-called slack variables, introduced to account for noise and non-separable data, and $\lambda > 0$ is a penalty parameter that trades-off model complexity and accuracy on training data. Problem (10) can be converted to dual form,

$$\begin{aligned} &\min_{0\leq\alpha_i\leq S} : \sum_i \alpha_i - \frac{1}{2}\sum_i\sum_j \alpha_i\alpha_j y_i y_j \Phi(x_i)^T\Phi(x_j) \\ &s.t.\ \ \sum_i \alpha_i y_i = 0, \quad 0 \leq \alpha_i \leq \lambda \forall i \end{aligned} \tag{11}$$

where α_i are the Lagrange multipliers associated with the constraints of the primal problem.

The resulting SVM classifier can be conveniently represented using the dual problem solution as

$$g(\mathbf{x}) = \sum_{i=1}^{N} y_i\alpha_i\Phi(\mathbf{x}_i)^T\Phi(\mathbf{x}) = \sum_{i=1}^{N} y_i\alpha_i\mathbf{K}(\mathbf{x}_i, \mathbf{x}), \tag{12}$$

where $\mathbf{K}(\cdot,\cdot)$ is the kernel function induced by certain classes of mapping functions Φ.

The conventional way to extend SVM to multi-class scenario is to decompose the $C+1$ class problem into a series of two-class problems, for which one-against-one [10] and one-versus-all [20] are the most widely used implementations. In one-versus-one approach, $C(C+1)/2$ binary SVM models are trained, one for each pair of classes, using data examples representing the opposing classes. In one-versus-all approach $C+1$ binary SVM models are trained using entire data, where the first SVM model interprets class 1 as the positive class and the remaining classes as the negative class. The process is repeated $C+1$ times, each time using different class as the positive class. In both cases, binary decisions are combined according to certain voting schema [10,20] to produce the final multi-class prediction $\hat{y}$.

An appealing feature of SVM is that it can provide different types of prediction outputs: soft values $g(\mathbf{x})$, hard decisions $sign(g(\mathbf{x}))$, or, after an appropriate output transformation [8], class probability estimates.

4. Decentralized fault detection and diagnosis

Unlike the centralized approach, in which all sensor measurements are sent to some central location and processed to train a single fault classification model, the idea behind the decentralized approach is that each sensor or a group of sensors has its own classifier. Instead of communicating the actual observations

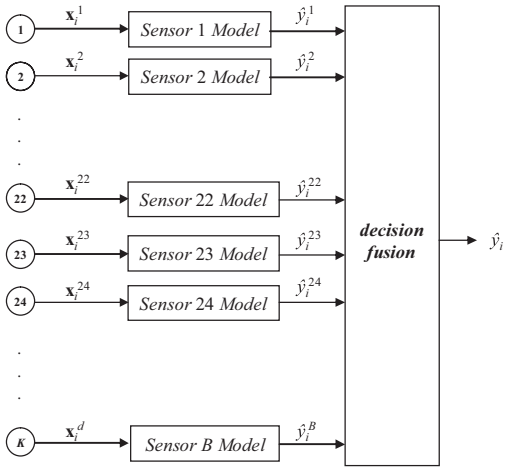

Fig. 3. Completely decentralized fault classification model.

$\mathbf{x}_i^b$, local predictions $\hat{y}_i^b$ are sent to the fusion center. Final predictions $\hat{y}_i$ are made by decision fusion of local classifier predictions $(\hat{y}_i^1, \hat{y}_i^2, \ldots, \hat{y}_i^B)$.

The objective of our study is to build a decentralized system whose performance is as close as possible to the centralized approach. Building a decentralized fault detection and diagnosis algorithm typically entails three phases:

1) Process decomposition
2) Building local classifiers
3) Fusion of local predictions

In the following, we describe each of the phases in more detail.

4.1. Process decomposition

We consider different process decomposition approaches, including completely decentralized strategy and two multi-block strategies as follows

A. *Completely decentralized process decomposition (CDD).* Each sensor is considered as a separate block $\mathbf{x}^b$, $b = 1, \ldots, K$ (Fig. 3). A model of this type is not likely to result in accurate fault classification because it makes it difficult to exploit correlations between sensors.
B. *Multi-block process decomposition.* In multi-block approach, sensors are grouped into B possibly overlapping blocks and a local model is trained at each block. Fig. 4 shows an example of a multi-block decentralized scheme.
   B.1. *Multi-block Process decomposition using domain knowledge (MB1).* Decomposition is typically based on domain knowledge, physical constraints, or process topology [19,25,27–29,40,42].
   B.2. *Multi-block Process decomposition using Sparse PCA (MB2).* We propose an alternative approach that partitions the process such that the strongest correlations among sensors are preserved. To accomplish this we use Sparse PCA [32] tool. Unlike regular PCA, that transforms the input space into a new space whose principal components are linear combinations of all variables, Sparse PCA produces components that are linear combinations of small number of variables.

There are several different formulations of Sparse PCA [9,32]. A Direct Sparse PCA (DSPCA) [32] algorithm that exploits convex optimization tools to solve a convex relaxation of the sparse PCA problem guarantees global convergence and has been shown to

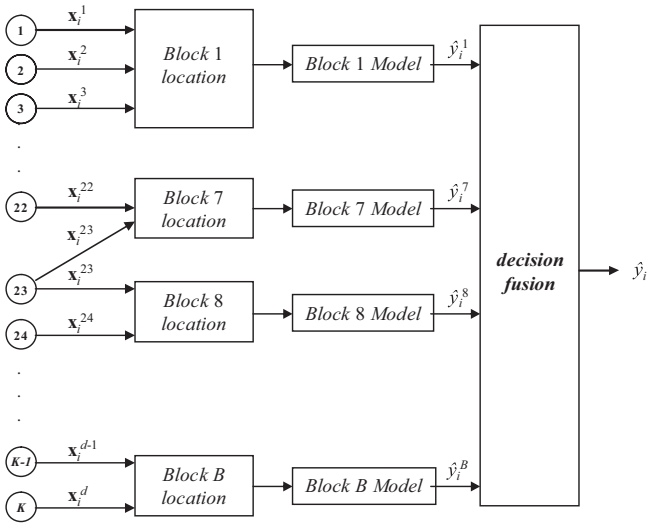

Fig. 4. Multi-block decentralized fault classification model.

provide better results than other algorithms, i.e. producing sparser vectors while explaining the same amount of variance. The DSPCA algorithm implementation in C and MATLAB is available online [22].

Given a data matrix $\mathbf{X} = [(\mathbf{x}_1^T), (\mathbf{x}_2^T), \ldots, (\mathbf{x}_N^T)]^T$ and its covariance matrix $\mathbf{S}$ from (7), the objective is to decompose $\mathbf{S}$ into sparse components $[\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_d]$, while constraining the number of non-zero elements (cardinality) of each $\mathbf{v}$ to r. The problem of maximizing the variance of $\mathbf{v} \in \Re^d$ with the constraint on cardinality is defined as

$$\begin{array}{ll} \text{maximize} & \mathbf{v}^T\mathbf{S}\mathbf{v} \\ \text{subject to} & ||\mathbf{v}||_2 = 1 \\ & Card(\mathbf{v}) \le r \end{array}. \tag{13}$$

Due to the cardinality constraint, this problem is NP-hard. Using semidefinite relaxation and replacement of the nonconvex constraints by weaker but convex ones DSPCA forms an optimization problem that is equivalent to (13),

$$\begin{array}{ll} \text{maximize} & Tr(\mathbf{S}\mathbf{U}) \\ \text{subject to} & Tr(\mathbf{U}) = 1 \\ & 1^T|\mathbf{U}|1 \le r \\ & \mathbf{U} \succcurlyeq 0 \end{array}, \tag{14}$$

where $\mathbf{U} = \mathbf{v}\mathbf{v}^T$ is a positive semidefinite matrix of size $d \times d$.

Let us denote the solution to (14) as $\mathbf{U}_1$. The first sparse component $\mathbf{v}_1$ is obtained as the most dominant eigenvector of $\mathbf{U}_1$. This process iterates by updating the covariance matrix

$$\mathbf{S}_2 = \mathbf{S}_1 - (\mathbf{v}_1^T\mathbf{S}_1\mathbf{v}_1)\mathbf{v}_1\mathbf{v}_1^T, \tag{15}$$

and solving (14) to obtain the second component, where $\mathbf{S}_1 = \mathbf{S}$ is the original covariance matrix. The procedure is repeated until all d sparse components are obtained.

In general, only the first B components that explain majority of variance in data are needed. In summary, DSPCA procedure is applied to $\mathbf{X}$ to obtain sparse components $\mathbf{V} = [\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_B]$. Each component defines a block, and non-zero indices of $\mathbf{v}_b$ define variables of b-th block.

4.2. Training local models

For each block, a separate classifier is trained using available data. As introduced in Section 3.1, a data set for training of b-th block local classifier can be expressed as $D_b = \{(\mathbf{x}_i^b, y_i), i = 1, \ldots, N_b\}$. Both unsupervised and supervised approaches explained in Section 4 can be used to train a local classifier. After training, at any time i, all B local models send their predictions $\hat{y}_i^b$, $b = 1, \ldots, B$, to a central location where the final prediction $\hat{y}_i$ is made based on a specific decision fusion rule.

There is an important property that distinguishes local classifiers from a centralized classifier. Since each local classifier has access to only a subset of all process measurements, it is likely that it would be able to detect only a subset of possible faults. Consequently, local classifiers should not be trained to recognize all faults. To address this issue, we propose to first analyze a given block to discover which faults can be recognized and then to proceed with training of the local classifier on the selected subset of faults.

To analyze local data, we developed a PCA-based method. Based on the assumption that not all sensor blocks are able to detect all faults, given a set $\mathbf{C} = \{1, \ldots, C\}$ of known faults detectable at the centralized level, the goal is to discover subsets $\mathbf{C}_b$ of faults detectable by the b-th block at the local level. The local eigenvectors $\mathbf{V}^b$ and eigenvalues $\mathbf{\Lambda}^b$ are found from the local data matrix $\mathbf{X}_b = [(\mathbf{x}_1^b)^T, (\mathbf{x}_2^b)^T, \ldots, (\mathbf{x}_N^b)^T]^T$ representing normal conditions at the b-th block. Each fault C from $\mathbf{C}$ is tested at each block by calculating local T^2 and SPE statistics for fault C data sequences and comparing to the block's local threshold, chosen such that FPR = 2%. Both statistics are used as they can detect different types of faults. Fault C is considered detectable at the b-th block, and added to $\mathbf{C}_b$, if its T^2 or SPE statistics exceed the corresponding b-th block thresholds more than 20% of the time. Finally, only sequences of faults from $\mathbf{C}_b$ and normal conditions are used in the b-th block training data set D_b.

4.3. Decision fusion using Maximum Entropy

Let us assume that a set of historical decision data $F = \{(\hat{\mathbf{y}}_i, y_i), i = 1, \ldots, N\}$, defined in Section 3.1, is available for training the decision fusion model. Depending on specific setup, the b-th component of the prediction vector $\hat{\mathbf{y}}_i = [\hat{y}_i^1, \hat{y}_i^2, \ldots, \hat{y}_i^B]$ can be a scalar $\hat{y}_i^b$ representing hard predictions (e.g. $\hat{y}_i^b = 14$ if b-th block classifier predicts fault 14) or a vector $\hat{\mathbf{y}}_i^b$ of soft predictions, $\hat{\mathbf{y}}^b = [\hat{y}_{i(0)}^b, \hat{y}_{i(1)}^b, \ldots, \hat{y}_{i(C)}^b]$, where $\hat{y}_{i(0)}^b$ represents the b-th block soft prediction for normal condition and $\hat{y}_{i(C)}^b$ represents the b-th block soft prediction for fault c. If necessary, soft predictions can be converted to approximate the posterior class probability.

Maximum Entropy (MaxEnt) model. We propose a decision fusion algorithm based on the discriminative probabilistic model called the Maximum Entropy (MaxEnt) model [23,24]. MaxEnt represents the conditional probability $p(y_i|\hat{y}_i^1, \hat{y}_i^2, \ldots, \hat{y}_i^B)$ as

$$P(y_i|\hat{y}_i^1, \hat{y}_i^2, \ldots, \hat{y}_i^B) = \frac{\exp(\sum_{m=1}^M \omega_m f_m(\hat{y}_i^1, \hat{y}_i^2, \ldots, \hat{y}_i^B, y_i))}{\sum_{c=0}^C \exp(\sum_{m=1}^M \omega_m f_m(\hat{y}_i^1, \hat{y}_i^2, \ldots, \hat{y}_i^B, y_i = c))}. \tag{16}$$

where $f_m(\hat{y}_i^1, \hat{y}_i^2, \ldots, \hat{y}_i^B, y)$ is the m-th feature function and ω_m is the parameter of the m-th feature function. A large modeling flexibility is allowed in design of feature functions. For the purposes of illustration, we give the following example of feature function f_1,

$$f_1(\hat{y}_i^1, \hat{y}_i^2, \ldots, \hat{y}_i^B, y_i) = \begin{cases} 1, & \hat{y}^{10} \ \textit{exists} \ \wedge \hat{y}_i^{10} = 2 \wedge y_i = 2 \\ 0, & \text{otherwise} \end{cases}. \tag{17}$$

This feature function equals 1 when the 10th block local prediction is available ($\hat{y}^{10}$ *exists*) and it equals $\hat{y}_i^{10} = 2$ and the true label is $y_i = 2$, and equals 0 otherwise. The term $\hat{y}^{10}$ *exists* is a fault tolerance term which ensures a fast and valid f_1 output even when local prediction $\hat{y}^{10}$ is not available. Note that f_1 is evaluated using prediction from the 10th block only, which can change over time i.

Given: B number of sensor blocks, B sets $\mathbf{C}_b$ of faults recognizable at the b-th block with hard predictions $\hat{y}_i^b, j=1$
FOR $b=1$ **TO** B // for each sensor block b
  FOR $c \in \{0, \mathbf{C}_b\}$ // for normal conditions and C_b faults

$$f_j(\hat{y}_i^1, \hat{y}_i^2, ..., \hat{y}_i^B, y_i) = \begin{cases} 1, & y_i = c \wedge \hat{y}_{ic}^b = c \wedge \hat{y}_i^b \ exists \\ 0, & otherwise \end{cases}, \ j = j+1$$

  END
END

Fig. 5. Hard feature function construction algorithm.

The issue of construction of feature functions will be addressed in the following subsection. Once the feature functions are known, the remaining issue is how to learn parameters ω_m from decision data F. MaxEnt achieves this by maximizing likelihood defined as

$$L = \sum_{i=1}^{N} \log \ P(y_i | \hat{y}_i^1, \hat{y}_i^2, \ldots, \hat{y}_i^B), \tag{18}$$

Depending on the process at hand and number of faults to diagnose, there might be a need for large number of feature functions. This adds a risk of overfitting that can be avoided by weighted l_2 regularization. Instead of (18) the regularized likelihood is maximized,

$$L^* = L - \pi \sum_{m=1}^{M} \omega_m^2, \tag{19}$$

where π is a hyperparameter controlling the strength of regularization. Since L^* is a concave function, there exists a global optimum solution that can be found using standard convex optimization algorithms. In this paper, we use the gradient ascent method that is an iterative procedure for updating current ω_m^{old} estimates as

$$\omega_m^{new} = \omega_m^{old} + \eta \frac{\partial L^*}{\partial \omega_m}. \tag{20}$$

Gradient ascent iterations are performed until L^* converges. When the optimal parameters ω_m, $m=1, \ldots, M$, are learned the final prediction $\hat{y}$ is made from the local predictions $\hat{y}^1, \hat{y}^2, \ldots, \hat{y}^B$ as the one that maximizes the conditional probability,

$$\hat{y}_i = arg\max_y P(y | \hat{y}_i^1, \hat{y}_i^2, \ldots, \hat{y}_i^B). \tag{21}$$

In summary, the Maximum Entropy approach allows using a large number of feature functions that are thought to be useful for fault classification. The procedure starts by proposing M feature functions f_m. Then, parameters ω_m are initially set to $\omega_m = 1/M$, $m=1, \ldots, M$ and updated using data F to find the optimal values.

Note that the structure of the feature functions f_m has no influence on the optimization procedure. A feature function can be any given rule and even rules of different type (structure) can be used.

Parameters ω_m reveal two important properties of feature functions and MaxEnt model. First, feature functions with larger parameters are more influential in final prediction. Second, MaxEnt models with large parameters also reveal large confidence in prediction. Therefore, ω_m could also be treated as prediction uncertainty parameters. Weights of unimportant rules will be near zero and ω_m could even become negative during training if f_m is a poor decision maker. To speed-up learning, we remove a feature function from the ensemble once its parameter becomes negative.

Constructing Feature Functions. Importance of feature functions in MaxEnt is similar to importance of input attributes in classification. The more informative the feature functions are, the higher accuracy of decision fusion. There are several ways to construct feature functions. One is to do it manually using domain knowledge. However, this can be very labor intensive. In this paper, we propose an automatic feature generator that uses predictions from a single block for any given feature function. The functional feature's form is the same as (17). Fig. 5 summarizes feature generator using hard local classifier predictions.

The algorithm produces a total of $B \cdot sum_b\ (\mathbf{C}_b + 1)$ feature functions when the PCA approach explained in Section 4.2 is used. If the PCA fault selection procedure is not used, a total of $9 \cdot B \cdot (C+1)$ feature functions are constructed, resulting in increased computational cost and increased risk of overfiting.

Following a similar strategy, an algorithm for feature function construction based on soft local predictions is shown in Fig. 6. A prototype soft feature function is of the form

$$f_2(\hat{y}_i^1, \hat{y}_i^2, \ldots, \hat{y}_i^B, y_i) = \begin{cases} 1, & \hat{y}^{10} \ exists \wedge \hat{y}_{i(2)}^{10} > 0.5 \wedge y_i = 2 \\ 0, & \text{otherwise} \end{cases}, \tag{22}$$

where f_2 is activated if the true label is $y_i = 2$ and the 10th block soft prediction for fault 2, $\hat{y}_{i(2)}^{10}$, is greater than 0.5. This algorithm results in $9 \cdot B \cdot sum_b\ (\mathbf{C}_b + 1)$ feature functions, which is 9 times more than the result of algorithm from Fig. 5.

Note that this decision fusion strategy can be applied in case of a single fault type ($C=1$) as well. The number of hard feature functions is then at most $B \cdot 2$ and soft feature functions at most $9 \cdot B \cdot 2$.

Given: number of sensor blocks B, and B sets $\mathbf{C}_b$ of faults recognizable at the b-th block with soft predictions $\hat{y}_{i(c)}{}^b, j=1$
FOR $b=1$ **TO** B // for each sensor block b
  FOR $c \in \{0, \mathbf{C}_b\}$ // for normal conditions and C_b faults
    FOR $p = 0.1$ **TO** 0.9 // in steps of 0.1

$$f_j(\hat{\mathbf{y}}_i^1, \hat{\mathbf{y}}_i^2, ..., \hat{\mathbf{y}}_i^B, y_i) = \begin{cases} 1, & y_i = c \wedge \hat{y}_{i(c)}^b > p \wedge \hat{\mathbf{y}}_i^b \ exists \\ 0, & otherwise \end{cases}, j = j+1$$

    END
  END
END

Fig. 6. Soft feature function construction algorithm.

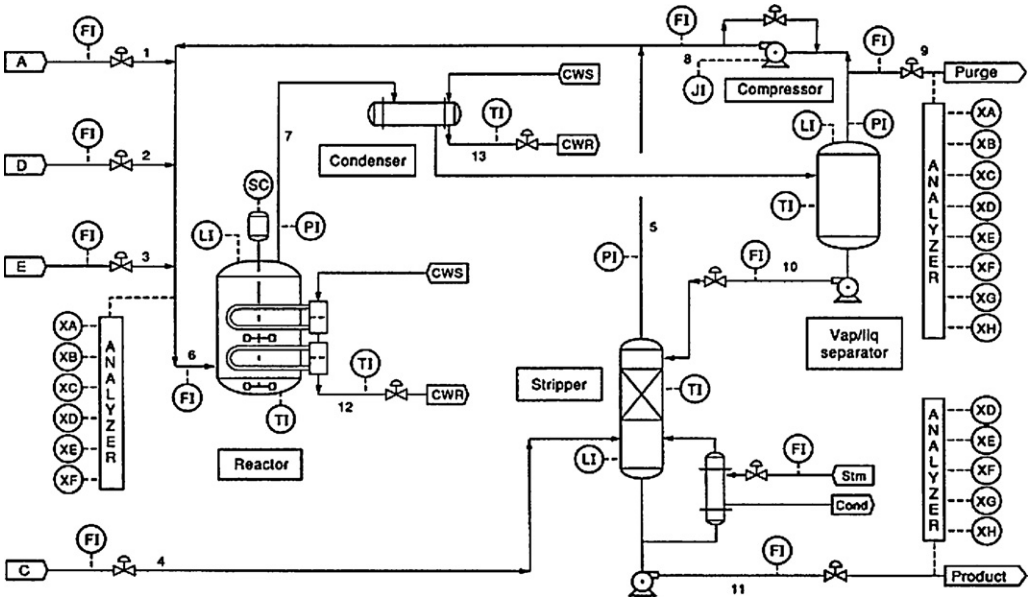

Fig. 7. Tennessee Eastman Process scheme.

5. Experiments

The proposed decentralized fault detection and diagnosis model was evaluated on the benchmark Tennessee Eastman Process (TEP) data set [2]. This section presents results from using different process decomposition methods, including complete decomposition, location based block partition and sparse PCA based partition, as well as two different decision fusion strategies, including the baseline voting scheme and the proposed MaxEnt algorithm.

The Tennessee Eastman Process data set is a well-known simulated industrial problem for process monitoring and control. Over years, it has become a benchmark data for a large number of fault detection [11–14] and fault diagnosis [16–18] approaches. The model and the simulator were described in detail in [2]. The TEP model is a chemical process with five major operation units: a reactor, a condenser, a compressor, a stripper, and a separator (Fig. 7). The plant represents an open-loop unstable plant that produces two liquid products G and H from gaseous feeds A, C, D, E, and the inert component B.

The process has 53 variables, including 22 process measurements, 19 analyzer measurements, and 12 manipulated variables. Our evaluation results presented here focus on all the process measurements and 11 manipulated variables, where the remaining constant manipulated variable was ignored. We did not consider the analyzer measurements.

The modified closed-loop version [50] of the original open-loop Fortran TEP implementation [2] was used to simulate data for our experiments. The source code uses a discrete control algorithm to stabilize the process. A total of 20 types of faults can be simulated, ranging from faults that are easy to detect with no delay (e.g., faults 1 and 4), to faults that are detectable only after a certain amount of propagating time (e.g., faults 17 and 18), to very subtle faults that are hard to detect (faults 3, 9, 15) even with centralized methods. These properties of the TEP faults have been observed by previous studies [10–18]. As a result, our evaluation did not consider faults 3, 9 and 15.

5.1. Experimental setup

Here we describe the details involving data generation process, preprocessing and parameter selection.

5.1.1. Data sets, preprocessing and parameter selection

To create input attributes $\mathbf{x}_i$, we used the dynamic approach from [12,13]. Observations from all K sensors at time i were augmented with observations from the previous l time moments and stacked into a variable vector $\mathbf{x}_i$ with $(l+1)\cdot K$ variables. We experimented with different values of l. Cross-validation was used to select the value of l that resulted in the best trade-off between all performance measures. Larger values of l resulted in improved performance with respect to FPR and TPR but increase in the Detection Delay. As low Detection Delay was preferred, we did not find $l > 5$ a good choice for this application.

The training data for the centralized approach $D_{tr} = \{(\mathbf{x}_i, y_i), i = 1, \ldots, N, y_i \in \{0, 1, \ldots, C\}\}$ consists of $N = 68{,}000$ examples, 34,000 examples describing normal process operation and 34,000 examples that describe process faults. To make evaluation in terms of Detection Delay (DD) and Classification Delay (CD) possible, data D_{tr} represents a collection of time-series in which normal and faulty conditions interchange. Each time-series contains $L = 1000$ examples, where first 500 examples represent normal conditions and the remaining examples represent fault behavior. For each of the 17 considered faults (faults 3, 9 and 15 were excluded) four such time-series were simulated.

In the decentralized approach, the plant was first decomposed into B blocks, which yields B training data sets $D_{tr}^b = \{(\mathbf{x}_i^b, y_i), \quad i = 1, \ldots, N\}$, $b = 1, \ldots, B$. After applying the PCA fault selection procedure (Section 4.2), each local training set was reduced to $D_{tr}^b = \{(\mathbf{x}_i^b, y_i), \quad y_i \in \{0, \mathbf{C}_b\}, i = 1, \ldots, N_b\}$, where $\mathbf{C}_b$ represent faults detectable by b-th local classifier and N_b are training samples representing normal conditions and detectable faults. Parameter a was set using parallel analysis technique proposed in [14].

For purposes of evaluation a test data set D_{test} of size $N_{test} = 340{,}000$ was used. Similarly to D_{tr}, test data set D_{test} is a collection of time-series in which normal and faulty conditions interchange. A total of 10 time-series for each of the 17 faults were simulated. The time-series were of same length, $L = 2000$, where the fault was introduced after the 1001st example.

The number of allowed non-zero eigenvector elements in Sparse PCA decomposition algorithm was set to $r = 8$. It was selected by cross-validation as the parameter that results in the best trade-off between the performance measures. The resulting eigenvectors were sorted by their eigenvalues, such that the first eigenvector

Table 1
Sensor-by-sensor fault detection ability.

Sensor (b)	Sets $\mathbf{C}_b$ of faults detectable by sensor b	Sensor (b)	Sets $\mathbf{C}_b$ of faults detectable by sensor b
*xmeas*1	1, 5, 6, 7, 8, 12, 13, 18	*xmeas*18	1, 2, 5, 6, 7, 8, 10, 12, 13, 16, 18
*xmeas*2	2, 6, 7, 12, 13, 18	*xmeas*19	1, 2, 5, 6, 7, 8, 12, 13, 16, 18, 19
*xmeas*3	1, 2, 5, 6, 7, 8, 12, 13, 18	*xmeas*20	1, 5, 6, 7, 8, 12, 13, 18, 19, 20
*xmeas*4	1, 2, 5, 6, 7, 8, 12, 13, 18	*xmeas*21	1, 5, 6, 7, 8, 12, 13, 14, 17, 18
*xmeas*5	19	*xmeas*22	1, 2, 5, 6, 7, 8, 12, 13, 18, 20
*xmeas*6	1, 2, 5, 6, 7, 8, 12, 13, 18	*xmv*1	18
*xmeas*7	1, 2, 5, 6, 7, 8, 12, 13, 14, 18, 19, 20	*xmv*2	1, 2, 5, 6, 7, 8, 12, 13, 18
*xmeas*8	1, 5, 6, 7, 8, 12, 13, 18	*xmv*3	1, 5, 6, 7, 8, 12, 13, 18
*xmeas*9	4, 7, 11, 12, 14, 17, 18	*xmv*4	1, 2, 5, 7, 8, 12, 13, 18
*xmeas*10	1, 2, 5, 6, 7, 8, 12, 13, 18	*xmv*5	1, 2, 5, 6, 7, 8, 12, 13, 18, 19, 20
*xmeas*11	1, 2, 5, 6, 7, 8, 12, 13, 17, 18, 20	*xmv*6	1, 2, 5, 6, 7, 8, 12, 13, 18
*xmeas*12	–	*xmv*7	–
*xmeas*13	1, 2, 5, 6, 7, 8, 12, 13, 18, 19, 20	*xmv*8	–
*xmeas*14	18	*xmv*9	1, 2, 5, 6, 7, 8, 10, 12, 13, 16, 18
*xmeas*15	–	*xmv*10	1, 4, 5, 6, 7, 8, 11, 12, 13, 14, 17, 18
*xmeas*16	1, 2, 5, 6, 7, 8, 12, 13, 18, 19, 20	*xmv*11	5, 6, 12, 18
*xmeas*17	12, 18		

retains the most variance. The leading B components that explain 95% of variance in normal behavior data are preserved as resulting blocks. Note that r can be selected based on process constraints as well. Small values of r result in a large number of smaller blocks, which makes the system more decentralized. Larger values of r allow for, but not necessarily result in, bigger size blocks.

For training of MaxEnt decision fusion model, a data set $F = \{(\hat{\mathbf{y}}_i, y_i),\ i = 1, \ldots, N\}$ was formed from local model predictions. Regularization parameter π was chosen by cross-validation ($\pi = 0.5$) to maximize the decision fusion generalization power. The gradient ascent procedure was terminated when likelihood did not improve by more than 10^{-4}.

5.1.2. Plant decomposition

Completely decentralized process decomposition (CDD). Each sensor was considered separately, with $K_b = 1$ and $B = K = 33$. Table 1 gives a detailed TEP sensor-by-sensor fault detection ability, obtained using PCA fault selection procedure from Section 4.2. We can observe that some faults affect the entire process and are visible by almost all sensors (e.g. fault 18 can be detected by 28 out of 33 sensors) or majority of sensors (faults 1, 2, 6, 8, 12, 13 and 20). Other faults are more "local" and can be detected by only a few sensors (faults 4, 10, 11, 14, 16, 17 and 19). It is interesting to observe that sensors *xmeas*12, *xmeas*15, *xmv*7 and *xmv*8 alone could not detect any fault.

Plant decomposition based on location (MB1). Grouping the TEP sensors into blocks around various equipment and streams in the plant resulted in 9 blocks. Sensor distribution is depicted in Fig. 8 and listed in the second column of Table 2. The first block groups separate input feed sensors, the second block represents the total input feed, blocks 3–8 consist of sensors concentrated around the Reactor, Condenser, Compressor, Purge, Separator and Striper, respectively, and block 9 groups product stream sensors.

The resulting abilities of blocks to detect faults, as obtained by PCA fault selection procedure from Section 4.2 are reported in the third column of Table 2. We can observe that faults 12 and 18 are

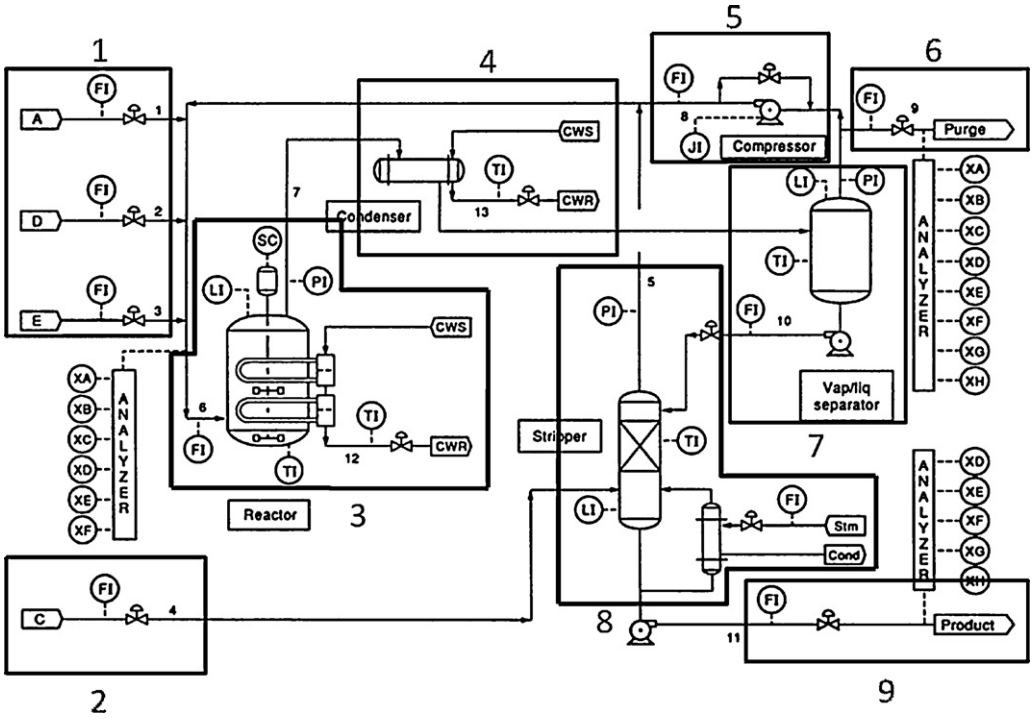

Fig. 8. Tennessee Eastman Process decomposition by location.

Table 2
MB1 sensor distribution among blocks and block-by-block fault detection ability.

Block b	Sensors	Sets $\mathbf{C}_b$ of faults detectable by block b
1	*xmeas*1, *xmv*1, *xmeas*2, *xmv2*, *xmeas*3, *xmv*3	1, 2, 5, 6, 7, 8, 12, 13, 18
2	*xmeas*4, *xmv*4	1, 2, 5, 6, 7, 8, 12, 13, 18
3	*xmeas6, xmeas7, xmeas8, xmeas9, xmeas21, xmv*10	1, 2, 4, 5, 6, 7, 8, 10, 11, 12, 13, 14, 17, 18, 19, 20
4	*xmeas*22, *xmv*11	1, 2, 5, 6, 7, 8, 12, 13, 18, 20
5	*xmeas*5, *xmeas*20, *xmv*5	1, 2, 5, 6, 7, 8, 12, 13, 18, 19, 20
6	*xmeas*10, *xmv*6	1, 2, 5, 6, 7, 8, 12, 13, 18
7	*xmeas*11, *xmeas*12, *xmeas*13, *xmeas*14, *xmv*7	1, 2, 5, 6, 7, 8, 12, 13, 14, 17, 18, 19, 20
8	*xmeas*15, *xmeas*16, *xmeas*18, *xmeas*19, *xmv*9	1, 2, 5, 6, 7, 8, 10, 12, 13, 16, 17, 18, 19, 20
9	*xmeas*17, *xmv*8	12, 18

Table 3
MB2 sensor distribution among blocks and block-by-block fault detection ability.

Block	Sensors	Sets $\mathbf{C}_b$ of faults detectable by block b
1	*xmeas*7, *xmeas*11, *xmeas*13, *xmeas*16, *xmeas*20, *xmv*5	1, 2, 5, 6, 7, 8, 12, 13, 14, 17, 18, 19, 20
2	*xmeas*18, *xmeas*19, *xmv*9	1, 2, 5, 6, 7, 8, 10, 12, 13, 16, **17**, 18, 19, 20
3	*xmeas*12, *xmv*7	–
4	*xmeas*15, *xmv*8	–
5	*xmeas*17, *xmv*11	1, 2, 5, 6, 7, 8, 12, 13, 18, 20
6	*xmeas*1, *xmv*3	1, 5, 6, 7, 8, 12, 13, 18
7	*xmeas*10, *xmv*6	1, 2, 5, 6, 7, 8, 12, 13, 18
8	*xmeas*9, *xmv*10	1, 4, 5, 6, 7, 8, 11, 12, 13, 14, 17, 18
9	*xmeas*2, *xmeas*21	1, 5, 6, 7, 8, 12, 13, 14, 17, 18, **20**
10	*xmeas*3, *xmeas*4, *xmv*2	1, 2, 5, 6, 7, 8, 12, 13, 18
11	*xmv*1, *xmv*4	1, 2, 5, 6, 7, 8, 12, 13, 18
12	*xmeas*5, *xmeas*6	1, 2, 5, 6, 7, 8, 12, 13, 18, 19
13	*xmeas*8, *xmeas*22	1, 2, 5, 6, 7, 8, 12, 13, 18, 20
14	*xmeas*14	12, 18
15	*xmeas*8, *xmv*1	1, 5, 6, 7, 8, 12, 13, 18
16	*xmv*2	1, 2, 5, 6, 7, 8, 12, 13, 18
17	*xmeas*3	1, 2, 5, 6, 7, 8, 12, 13, 18
18	*xmeas*6	1, 2, 5, 6, 7, 8, 12, 13, 18
19	*xmv*4	1, 2, 5, 6, 7, 8, 12, 13, 18
20	*xmeas*11	1, 2, 5, 6, 7, 8, 12, 13, 17, 18, 20

visible by each block, while faults 1, 2, 5, 6, 7, 8, and 13 are visible by almost all blocks. Faults 4, 11, 16 are visible by only a single block and 10 and 14 by only two blocks.

Plant decomposition using Sparse PCA (MB2). A total of 33 sparse eigenvectors were obtained by decomposition of TEP sensors using *Sparse PCA* with $r = 8$ (Section 4.1). First $B = 20$ components are retained as they explained over 95% of variance in normal behavior data. The non-zero indices in each retained eigenvector define the block structure. The second column of Table 3 shows the resulting sensor distribution among blocks. As it can be observed, the first block (corresponding to the largest eigenvector) contains 6 sensors, while the remaining ones are rather small, typically containing

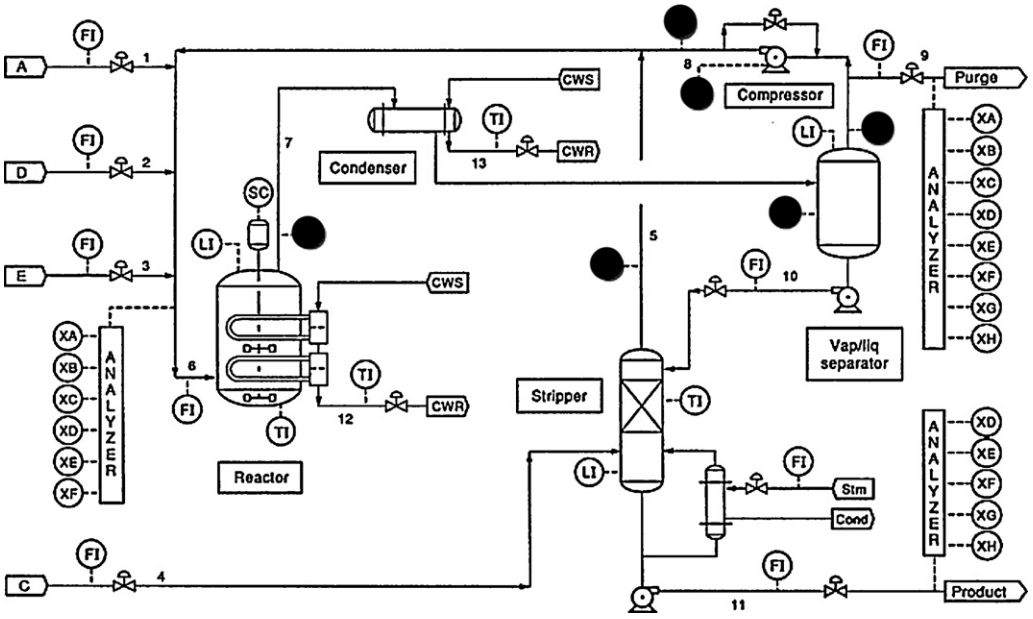

Fig. 9. Sensors which form block 1: obtained with DSPCA.

Table 4
Centralized SVM fault classification results.

Fault	TPR (%)	DD	CD
1	99.7	1.8	1.1
2	98.7	11.5	0.6
4	99.4	0	3.6
5	99.5	0	0.2
6	100	0	0
7	100	0	0
8	84.8	18.1	14.6
10	65.5	20.3	0.1
11	88.7	5.1	0
12	82.6	6.8	6
13	82.9	51.4	18.5
14	99.8	0	1.2
16	77.4	13.4	0
17	97.4	24.1	0.1
18	88.5	46.1	1.3
19	90.5	3.1	0.1
20	86.5	39.9	0
Avg	90.7	14.2	2.78
FPR	1.02%		

only one or two sensors. In this sense, the MB2 decomposition is more similar to CDD than to MB1. Fig. 9 shows locations of sensors forming the 1st block, which appears to be consistent with the process flow leading to the compressor and separator, indicating a certain degree of process correlation captured by this partition technique even though it is entirely data driven.

Table 3 also reports fault detection abilities of MB2 blocks as obtained by the PCA fault selection procedure. Blocks 3 and 4 appear insensitive to all faults. It can be noticed that the 2nd block consisting of sensors *xmeas*18, *xmeas*19 and *xmv*9 can detect fault 17 even though sensors *xmeas*18, *xmeas*19 and *xmv*9 alone cannot (Table 1). This is also the case with the 9th block which can detect faults 17 and 20 (bold in Table 3) while its two sensors *xmeas*2 and *xmeas*21 cannot.

5.1.3. Decision fusion

Majority vote decision fusion technique. The majority vote of the local classifiers was used as the baseline approach. In this approach, all sensor blocks have the same importance when voting. Because some faults are observable by only a subset of blocks, the majority vote rule was modified such that the only time normal condition decision is declared is when it is predicted by all blocks. In case of a tie, the final prediction was made by randomly picking one of the faults with the largest number of votes. This voting strategy reduces to a simple "or" rule in case of a single fault type ($C = 1$).

MaxEnt decision fusion. For conciseness, only results using MaxEnt soft feature functions (Fig. 6) are shown and discussed. We just note that our experiments (not shown here) indicate that hard feature functions (Fig. 5) result in slightly decreased accuracy as compared to soft feature functions. The algorithm from Fig. 6 resulted in 2390 CDD feature functions, 837 MB1 feature functions and 1548 MB2 feature functions. For comparison, if the PCA procedure for selection of fault sets $\mathbf{C}_b$ were not used, the algorithm from Fig. 6 would produce 6237 CDD feature functions. Thus, the PCA procedure results in significant computational savings.

A large number of feature functions are necessary for two reasons: (1) a large number of fault types and blocks, (2) different blocks are good at classifying different faults and poor or marginal at classifying other faults. As large number of parameters can lead to poor generalization, a regularization term was added to the original optimization (19) to avoid over-fitting. The evaluation was performed using a large test set to ensure there were no generalization issues.

5.2. Experimental results

Both centralized and decentralized fault classification models were multi-class SVM classifiers trained using data sets described in VI.*A*. *LibSVM* software [7] one-versus-one multi-class implementation with probability outputs [8] was used. SVM classifiers used a Gaussian RBF kernel, where the kernel width γ was set to inverse median of squared distances $\left\| x_i - x_j \right\|^2$ between normal condition data points in D (or D_b in case of local models) and slack parameter was set to $\lambda = 10$. The dynamic variable approach with $l = 2$ lagged variables resulted in the best performance. Only these results are shown.

Table 4 shows centralized SVM method fault-by-fault performance on the test set D_{test} based on the multi-criterion.

It can be seen that faults are detected with detection delay (DD) which ranges from zero (faults 4, 5, 6, 7 and 14), very low (faults 1, 11, 12 and 19), to relatively high (faults 8, 10, 13, 16, 17, 18 and 20). Classification delay (CD) ranged from none (faults 6, 7, 11 and 16), very low (faults 1, 2, 4, 5, 10, 14, 17, 18 and 19) to relatively high (faults 8, 12 and 13). Interestingly, faults in different DD and CD delay categories overlap. The achieved FPR of 1.02% is satisfactory. The true positive rates (TPR) show that some faults are classified almost perfectly (faults 1, 2, 4, 5, 6, 7, 14, and 17), while some achieve fair TPR (faults 8, 11, 12, 13, 18–20). The only two faults with TPR lower than 80% are faults 10 and 16.

Table 5 shows the completely decentralized (CDD) model performance. Notable performance changes, when compared to centralized model, are shown in bold. Symbol ∞ is used if a fault cannot be detected or classified correctly. Its delay is counted as 500 in the average value calculation in the last row. It can be observed that the overall TPR for the completely decentralized approach using majority vote and MaxEnt compared poorly to 90.7% TPR in the centralized approach. The difference in TPR was mostly due to faults 10, 16 and 18 (large drops in accuracy) and 5, 13 and 20 (moderate drops in accuracy). Faults 1, 2, 4 and 14 were classified with about the same accuracy by both the centralized and CDD decentralized model. DD increased only slightly, while CD increased largely for faults 5, 7, 10 and 18 and moderately for faults 1, 2, 6 and 13. The proposed MaxEnt algorithm outperformed majority vote in all performance measures.

In the next set of experiments we evaluated the performance of location-based (MB1) and correlation-based (MB2) multi-block decentralized models. MB1 fault-by-fault test set performance is shown in Table 6. The results confirm that *MB*1 model was much closer to the centralized model than the CDD model. The overall TPR improved over the CDD approach by approximately 15%, while the overall CD was considerably reduced. The FPR improved as well. The biggest overall improvement can be observed on faults 10 and 16 (bold in Table 6). The proposed MaxEnt algorithm outperformed majority vote on all faults.

Table 7 summarizes the performance of MB2 model. MB2 was more successful than MB1 even though MB2 decomposition was more decentralized, as it contained 20 blocks compared to the 9 MB1 blocks. The overall TPR improved over the completely decentralized approach by approximately 20%, and by approximately 5% over the MB1 approach. In addition, the overall DD and CD were noticeably improved. The biggest improvement was achieved on faults 5, 8, 10, 11, 16 and 20 (bold in Table 7). MaxEnt fusion was superior to majority vote fusion as it achieved much lower FPR and higher TPR. It can be concluded that the *MB*2 model with MaxEnt decision fusion came quite close to the centralized model, considering both accuracy and delay measures.

While intuitively, the accuracy is expected to grow with the size of blocks, a comparison of Tables 6 and 7 reveals that smaller MB2 blocks result in higher accuracy than larger MB1 blocks. This

Table 5
Results of the CDD model: majority vote vs. MaxEntr.

Fault	TPR (%)		DD		CD	
	Majority	MaxEnt	Majority	MaxEnt	Majority	MaxEnt
1	84.9	93.3	3.3	2.1	**31.3**	**20.8**
2	95.5	97.2	15.8	13.7	**28.1**	**21**
4	97.7	98.2	0	0	4.1	4.1
5	**43.8**	**51.7**	0	0	**183.6**	**161.2**
6	56.4	63.2	0	0	**16.1**	**14.2**
7	69.8	75.2	0	0	**260.1**	**211.2**
8	68.6	71.8	18.2	18.2	16.5	15.2
10	**0.79**	**19.7**	49.4	32.5	∞	50.1
11	66.1	75.1	6.8	5.7	4.9	2.7
12	60.6	66.2	7.6	6.8	9.1	7.1
13	**45.5**	**52.1**	54.8	51.7	**58.3**	**42.6**
14	99.3	99.3	0.5	0.3	1.2	1.4
16	**19.3**	**31.2**	36.6	34.2	9.3	9.3
17	75.3	82.6	25.2	24.8	5.3	5.7
18	**11**	**23.6**	46.7	46.7	**78.9**	**73.6**
19	82.6	86.8	3.5	2.6	1.5	2.4
20	**44.5**	**48.1**	46.6	46.8	3.7	3.5
Avg	60.1	66.4	18.53	16.83	71.3	38
FPR	2.18%	1.68%				

Table 6
Performance of the MB1 model: majority vote vs. MaxEntr.

Fault	TPR (%)		DD		CD	
	Majority	MaxEnt	Majority	MaxEnt	Majority	MaxEnt
1	95.9	97.8	2.7	2.4	13.8	8.7
2	96.3	97.5	14.3	13.1	19.8	12.8
4	99.4	98.9	0	0	4.9	4.2
5	44.9	75.8	0	0	18.5	15.5
6	82.5	90.1	0	0	6	5.5
7	89.1	94.1	0	0	3.4	2
8	72.2	78.4	19.7	18.9	14.9	14.3
10	**57.5**	**58.8**	**20.3**	**20.7**	**0.6**	**0.2**
11	88.5	88.5	5.6	5.6	1.3	1.3
12	76.3	78.2	7.2	7	6	5.6
13	57.3	69.5	54.3	52.1	29.7	28.6
14	97.8	99.4	0.5	0.5	5.1	2.1
16	**69.7**	**72.4**	**18.6**	**15.2**	**0.9**	**1.3**
17	90.1	94	25.1	25.1	2.5	2
18	22.6	60.1	46.7	46.5	33.1	16.2
19	90.2	90.2	1.4	1.4	2.3	2.3
20	47.2	65.8	46.9	44.3	4.3	4
Avg	75.1	82.9	15.48	14.88	9.83	7.44
FPR	1.65%	1.38%				

Table 7
Performance of the MB2 model: majority vote vs. MaxEntr.

Fault	TPR (%)		DD		CD	
	Majority	MaxEnt	Majority	MaxEnt	Majority	MaxEnt
1	95.9	99.1	2.1	2.1	11.8	1.6
2	96.8	97.9	14.2	12.4	14.6	10.1
4	98.9	99.2	0	0	4.2	3.9
5	**86.5**	**98.2**	**0**	**0**	**18.3**	**2.5**
6	84.2	95.3	0	0	7.6	5.1
7	91.3	96.2	0	0	3.1	1.4
8	**72.8**	**81.3**	**18.7**	**18.4**	**14.2**	**14.3**
10	**57.2**	**61.9**	**24.3**	**21.8**	**0.3**	**0.1**
11	**88**	**88.2**	**7.1**	**6.3**	**5.9**	**1.4**
12	77.1	79.2	6.9	6.9	4.3	5.1
13	57.3	76.3	51.7	51.7	31.9	21.5
14	98.9	99.7	0.3	0	1.4	1.4
16	**76.1**	**77**	**13.4**	**13.4**	**2.1**	**0.8**
17	90.1	95.7	24.5	24.1	3.2	1.7
18	21.2	73.3	46.4	46.1	36.1	3.5
19	89.4	90.2	1.3	1.4	1.4	0.8
20	**73.2**	**81.7**	**40.1**	**40**	**0**	**0**
Avg	79.7	87.6	14.78	14.39	9.4	4.42
FPR	1.57%	1.02%				

Table 8
MB2 model robustness to irrelevant feature functions.

Fault	TPR (%)		DD		CD	
	MB2	*MB2**	*MB2*	*MB2**	*MB2*	*MB2**
1	99.1	99	2.1	2.1	1.6	1.6
2	97.9	97.9	12.4	12.4	10.1	10.1
4	99.2	99	0	0	3.9	3.9
5	98.2	98.2	0	0	2.5	2.5
6	95.3	95.2	0	0	5.1	5.1
7	96.2	96.2	0	0	1.4	1.4
8	81.3	79.9	18.4	18.5	14.3	14.2
10	61.9	60.8	21.8	22	0.1	0.6
11	88.2	88	6.3	6.3	1.4	1.4
12	79.2	79.2	6.9	6.9	5.1	5.1
13	76.3	76.3	51.7	51.7	21.5	21.5
14	99.7	99.7	0	0	1.4	1.4
16	77	76.2	13.4	13.4	0.8	0.8
17	95.7	95.4	24.1	24.1	1.7	1.7
18	73.3	70.8	46.1	48	3.5	4.2
19	90.2	89.9	1.4	1.4	0.8	0.8
20	81.7	79.6	40	41	0	0
Avg	87.6	87.2	14.39	14.57	4.42	4.48
FPR	1.02%	1.12%				

Table 9
Results of the MB2 model: block failure scenario.

Fault	TPR (%)		DD		CD	
	Centralized	*MB2*	Centralized	*MB2*	Centralized	*MB2*
1	99.7	99.1	1.8	2.1	1.4	1.6
2	98.5	98.2	11.5	12.6	1.1	9.8
4	98.4	99.2	0	0	3.6	3.9
5	**18.3**	**18.2**	**0**	**0**	**3.9**	**2.5**
6	99.2	94.9	0	0	0	5.8
7	99.9	96.1	0	0	0	1.4
8	82.5	80.2	18.1	19	14.7	13.8
10	**6.9**	**0**	**145**	∞	**247**	∞
11	89.5	88.7	5.8	6.3	1.3	1.4
12	81.1	69.8	7.4	6.9	6	5.1
13	74.3	70.1	52.1	52.1	18.5	21
14	99.8	99.7	0	0	1.3	1.4
16	**0**	**0**	∞	∞	∞	∞
17	97.4	96.2	24.1	24.1	0.8	0.8
18	76.9	68.7	46.1	46.1	4.2	4.2
19	92.2	91.4	3.1	1.4	0.4	0.8
20	84.8	80.5	39.9	38.5	0	0
Avg	76.3	73.6	50.3	71.1	47.3	63.1
FPR	1.52%	1.67%				

indicates that the strategy of grouping sensors into blocks is more important than the actual number of sensors per block. The results confirm that Sparse PCA variance-maximization approach used in MB2 is more reasonable than distance-based partitioning used in MB1. To gain insight into the differences, we can observe that MB1 fails to capture strong correlations between 20 sensor pairs which are recognized by MB2, but belong to different blocks in MB1.

To evaluate MaxEnt algorithm sensitivity to feature function selection and fault tolerance capabilities of the proposed decentralized model (*MB2* with MaxEnt), two additional experiments were performed.

MaxEnt decision fusion sensitivity to feature function selection was tested by constructing 200 irrelevant feature functions (23) and adding them to the existing ones (used in the previous experiment). An example of an irrelevant feature function is

$$f(\hat{\mathbf{y}}_i^1, \hat{\mathbf{y}}_i^2, \ldots, \hat{\mathbf{y}}_i^B, y_i) = \begin{cases} 1, & \hat{\mathbf{y}}_i^5 \ \ exists \wedge \hat{y}_{i(19)}^5 > 0.8 \wedge y_i = 2 \\ 0, & \text{otherwise} \end{cases}, \quad (23)$$

which is activated when $y_i = 2$ and the 5th block detected fault 19 with probability higher than 80%. Robustness to irrelevant feature functions is a useful property, since the constructor of feature functions, human or machine, is prone to mistakes.

Table 8 compares *MB2* MaxEnt from Table 7 and *MB2** MaxEnt with the additional irrelevant feature functions. The results show that MaxEnt fusion is robust to irrelevant feature functions as the performance degraded only slightly. During training, the irrelevant feature functions received negative or very low weights and were thus removed or insignificantly participated in decision making.

To test the *MB2* MaxEnt fusion robustness to sensor failures, a scenario in which blocks 2, 5 and 10 were offline was simulated. Table 9 compares *MB2* MaxEnt fusion model in the block failure scenario with the centralized model which was retrained without sensor variables from blocks 2, 5 and 10. Symbol ∞ is used if a fault cannot be detected or classifier correctly, and it accounted for penalty 500 in the average value calculation in the last row.

The results show that *MB2* model with MaxEnt decision fusion achieved similar results to the retrained centralized model, without having to be retrained. For example, MaxEnt performance degraded only on faults 5, 10, and 16 (bold in Table 9). This is understandable because the disabled 2nd block was the only one that could observe faults 10 and 16. We note that the performance also slightly changed for faults 13 and 18 (underlined in Table 9) and that the overall increase of 0.5% in FPR was observed.

These results are promising as they indicate that it could be possible to prune sensors and blocks (e.g., in order to minimize the communication costs), without significantly degrading the fault classification performance.

6. Conclusion

In this paper we proposed a Sparse PCA method for process decomposition and a Maximum Entropy method for decision fusion to be used in decentralized fault detection and diagnosis. We compared the Sparse PCA decomposition method with two baseline decomposition approaches and the Maximum Entropy decision fusion method to the baseline majority vote decision fusion in several different decentralized fault detection and diagnosis scenarios. The results of our study are quite promising as they show that Maximum Entropy is much more effective in decision fusion than the baseline method and that Sparse PCA-based process decomposition leads to better fault classification results than the topology-based approach. Our results also indicate that Sparse PCA process decomposition with Maximum Entropy decision fusion lead to decentralized model whose performance approaches that of a centralized model, while providing increased flexibility and fault tolerance.

There are several open questions that are worth addressing. One is focusing on explicit process decomposition algorithms that would create blocks that are robust (by grouping the sensors such that the probability of block failure is minimized), maintenance friendly (by grouping the sensors with similar maintenance needs), and cost effective (by grouping the sensors such that the overall communication cost is minimized). To achieve this goal, we are studying ways to impose additional constraints to the original Sparse PCA optimization problem. Another question is how to reduce a need to have abundant training data for model development. The typical scenario for most of the related work in fault diagnosis is that we have abundant historical data prior to training where all types of faults have already happened at least once. This assumption is likely to be violated in practical settings because new fault types can emerge over time. To address this issue, we are exploring ways of combining the strengths of unsupervised and supervised methods to build a model capable of doing well on both previously seen and new types of faults.

References

[1] L.H. Chiang, E. Russell, R.D. Braatz, Fault Detection and Diagnosis in Industrial Systems, Springer, 2001.
[2] D.D. Downs, E.F. Vogel, A plant-wide industrial process control problem, Computers & Chemical Engineering 17 (3) (1993) 245–255.
[3] R.O. Duda, P.E. Hart, Pattern Classification and Scene Analysis, John Wiley & Sons, New York, 1973.
[4] J.F. MacGregor, Statistical process control of multivariate process, in: IFAC Int. Symp. on Advanced Control of Chemical Processes, 1994, pp. 427–435.
[5] I.T. Jolliffe, Principal Component Analysis, Springer, 2002.
[6] J.-F. Chamberland, V.V. Veeravalli, Decentralized detection in sensor networks, IEEE Transactions on Signal Processing 51 (2003) 407–416.
[7] C.C. Chang, C. Lin, LIBSVM: A Library for Support Vector Machines (Version 2.3), 2001.
[8] J. Platt, Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods, in: Advances in Large Margin Classifiers, 1999.
[9] H. Zou, T. Hastie, R. Tibshirani, Sparse principal component analysis, Journal of Computational and Graphical Statistics 15 (2) (1996) 265–286.
[10] R. Debnath, N. Takahide, H. Takahashi, A decision based one-against-one method for multi-class support vector machine, Pattern Analysis & Applications 7 (2004) 164–175.
[11] J. Chen, C.M. Liao, Dynamic process fault monitoring based on neural network and PCA, Journal of Process Control 27 (2002) 277–289.
[12] A. Kulkarni, V.K. Jayaraman, B.D. Kulkarni, Knowledge incorporated support vector machines to detect faults in Tennessee Eastman Process, Computers and Chemical Engineering 29 (2005) 2128–2133.
[13] J.M. Leea, C.K. Yoob, I.-B. Leea, Statistical monitoring of dynamic processes based on dynamic independent component analysis, Chemical Engineering Science 59 (2004) 2995–3006.
[14] W. Ku, R.H. Storer, C. Georgakis, Disturbance detection and isolation by dynamic principal component analysis, Chemometrics and Intelligent Laboratory Systems 30 (1995) 179–196.
[15] A. Hyvärinen, J. Karhunen, E. Oja, Independent Component Analysis, John Wiley & Sons, 2001.
[16] B.S. Yang, T. Han, J.L. An, ART-KOHONEN neural network for fault diagnosis of rotating machinery, Mechanical Systems and Signal Processing 18 (3) (2004) 645–657.
[17] M. Ge, R. Du, G. Zhang, Fault diagnosis using support vector machine with an application in sheet metal stamping operations, Mechanical Systems and Signal Processing 18 (1) (2004) 143–159.
[18] L.H. Chiang, M.E. Kotanchek, A.K. Kordon, Fault diagnosis based on Fisher discriminant analysis and support vector machines, Computers and Chemical Engineering 28 (8) (2003) 1389–1401.
[19] Y. Zhang, H. Zhou, S.J. Qin, T. Chai, Decentralized fault diagnosis of large-scale processes using multiblock kernel partial least squares, IEEE Transactions on Industrial Informatics 6 (1) (2010) 3–10.
[20] Y. Liu, Y.F. Zheng, One-against-all multi-class SVM classification using reliability measures, in: International Joint Conference on Neural Networks, 2005.
[21] H. Marzi, S.F. Xavier, Real-time fault detection and isolation in industrial machines using learning vector quantization, Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture 218 (2004) 949–959.
[22] DSPCA MATLAB package [Online]. Available: http://www.princeton.edu/~aspremon//DSPCA.htm.
[23] A. Ratnaparkhi, A maximum entropy model for part-of-speech tagging, in: Proc. EMNLP, Association for Computational Linguistics, New Brunswick, NJ, 1996.
[24] A.L. Berger, S.A. Della Pietra, V.J. Della Pietra, A maximum entropy approach to natural language processing, Computational Linguistics 22 (1996).
[25] S.J. Qin, S. Valle, M.J. Piovoso, On unifying multiblock analysis with application to decentralized process monitoring, Journal of Chemometrics 15 (2001) 715–742.
[26] J.T. Spooner, K.M. Passino, Decentralized adaptive control of nonlinear systems using radial basis neural networks, IEEE Transactions on Automatic Control 44 (1999) 2050–2057.
[27] A. Smilde, J. Westerhuis, R. Boque, Multiway multiblock component and covariates regression models, Journal of Chemometrics 14 (2000) 301–331.
[28] R. Isermann, Fault-Diagnosis Systems: An Introduction from Fault Detection to Fault Tolerance, Springer, Berlin, 2006.
[29] G.A. Cherry, S.J. Qin, Multiblock principal component analysis based on a combined index for semiconductor fault detection and diagnosis, IEEE Transactions on Semiconductor Manufacturing 19 (2006) 159–172.
[30] B.M. Wise, N.L. Ricker, D.F. Veltkamp, Upset and sensor failure detection in multivariate process, Technical Report, Eigenvector Research, Manson, WA, 1989.
[31] A.C. Raich, A. Cinar, Multivariate statistical methods for monitoring continuous process: Assessment of discriminatory power disturbance models and diagnosis of multiple disturbances, Chemometrics and Intelligent Laboratory Systems 30 (1995) 37–48.
[32] A. D'Aspremont, L.E. Ghaoui, M.I. Jordan, G.R.G. Lanckriet, A direct formulation for sparse PCA using semidefinite programming, Advances in Neural Information Processing Systems 17 (2004) 41–48.
[33] V. Venkatasubramanian, R. Rengaswamy, K. Yin, S.N. Kavuri, A review of process fault detection and diagnosis. Part I: quantitative model-based methods, Computers and Chemical Engineering 27 (2003) 293–311.
[35] R. Ferrari, T. Parisini, M.M. Polycarpou, Distributed fault diagnosis with overlapping decompositions: an adaptive approximation approach, IEEE Transactions on Automatic Control 54 (2009) 794–799.
[36] K. Ghosh, Y.S. Ng, R. Srinivasan, Evaluation of decision fusion strategies for effective collaboration among heterogeneous fault diagnostic methods, Computers and Chemical Engineering 35 (2) (2011) 342–355.
[37] Q. Cheng, P.K. Varshney, J.H. Michels, C.M. Belcastro, Distributed fault detection with correlated decision fusion, Transactions on Aerospace and Electronic Systems 45 (2009).
[38] T. Clouqueur, P. Ramanathan, K.K. Saluja, K.-C. Wang, Value-fusion versus decision-fusion for fault-tolerance in collaborative target detection in sensor networks, in: International Conference on Information Fusion, 2001.
[39] Q. Cheng, P.K. Varshney, J. Michels, C.M. Belcastro, Distributed fault detection via particle filtering and decision fusion, in: International Conference on Information Fusion, vol. 2, 2005.
[40] W. Li, W.H. Gui, Y.F. Xie, S.X. Ding, Decentralised fault detection of large-scale systems with limited network communications, IET Control Theory and Applications 4 (2010) 1867–1876.
[41] S.M. Magrabi, P.W. Gibbens, Decentralised fault detection and diagnosis in navigation systems for unmanned aerial vehicles, Position Location and Navigation Symposium (2000) 363–370.
[42] X. Zhang, M.M. Polycarpou, T. Parisini, Decentralized fault detection in a class of large-scale nonlinear uncertain systems, IEEE Conference on Decision and Control (2009) 6988–6993.
[43] B.V. Dasarathy, Decision fusion strategies in multisensor environments, IEEE Transactions on Systems, Man and Cybernetics 21 (1991) 1140–1154.
[44] A. D'Costa, A.M. Sayeed, Data versus decision fusion for distributed classification in sensor networks, Military Communications Conference (2003) 585–590.
[45] J.-J. Xiao, Z.-Q. Luo, Universal decentralized detection in a bandwidth-constrained sensor network, IEEE Transactions on Signal Processing 53 (2005) 2617–2624.
[46] V. Saligrama, M. Alanyali, O. Savas, Distributed detection in sensor networks with packet losses and finite capacity links, IEEE Transactions on Signal Processing 54 (2006) 4118–4132.
[47] M. Grbovic, S. Vucetic, Decentralized estimation using learning vector quantization, in: Data Compression Conference (DCC), Snowbird, UT, 2009, 446 pp.
[48] M. Grbovic, M.S. Vucetic, L. Weichang, X. Peng, A.K. Usadi, A boosting method for process fault detection with detection delay reduction and label denoising, in: KDD workshop Data Mining for Service and Maintenance, The 17th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (ACM SIGKDD), 2011.
[49] J. Zhou, A. Lazarevic, K.-W. Hsu, J. Srivastava, Y. Fu, Y. Wu, Unsupervised learning based distributed detection of global anomalies, International Journal of Information Technology and Decision Making (2011) 935–957.
[50] E.L. Russell, L.H. Chiang, R.D. Braatz, Data-driven Techniques for Fault Detection and Diagnosis in Chemical Processes, Springer-Verlag, London, 2000.