Contents lists available at ScienceDirect

Computers and Chemical Engineering

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

Application of graph theory and filter based variable selection methods in the design of a distributed data-driven monitoring system

Shaaz Khatib, Prodromos Daoutidis*

Department of Chemical Engineering and Materials Science, University of Minnesota, Minneapolis, MN 55455, USA

A R T I C L E I N F O

Article history:
Received 11 July 2020
Revised 31 August 2020
Accepted 15 September 2020
Available online 24 September 2020

Keywords:
Distributed monitoring
Large-scale system
System decomposition
Graph theory
Pattern recognition
Variable selection

A B S T R A C T

Two methods that represent extensions of our previously developed methods for distributed data-driven monitoring are proposed. The first, Extended Forward Selection for Distributed Pattern Recognition, selects a decomposition for distributed pattern recognition such that diagnostic performance is near optimal subject to constraints. It uses a filter method to select sensors and allocates them among a minimum number of subsystems using graph theoretic algorithms. Its advantage over the Forward Selection for Distributed Pattern Recognition method is that it scales to systems with sensors in the order of 1,000. The second method, Extended Subsystem and Sensor Allocation, uses graph theoretic algorithms to find the minimum number of locations for distributed monitoring, the sensors that should transmit to each location, and the monitoring tasks at each location. Its main advantage over the original Subsystem and Sensor Allocation method is that it is applicable even when data is not available before plant operation begins.

1. Introduction

Monitoring of a chemical plant is important to ensure that plant operation is safe, products of the required specifications are synthesized and there is no excessive control action. Process monitoring includes the objectives of determining whether the process is affected by an abnormality (i.e. fault detection), determining the reason for the abnormality (i.e. fault diagnosis) and bringing the process operation back to normal through plant maintenance (Chiang et al., 2001). A method used for fault detection or diagnosis will henceforth be referred to as a monitoring method. Some of the research done in the field of process monitoring is reviewed in (Chiang et al., 2001; Mason and Young, 2002; Ge et al., 2013; Qin, 2012; Blanke et al., 2016; Mhaskar et al., 2012). The use of data-driven methods for monitoring complex large-scale systems such as chemical plants is popular since these methods don't need a process model and are easy to implement (Venkatasubramanian et al., 2003a). Data-driven fault detection methods use statistical hypothesis testing to check if the sensor measurements are consistent with those expected when the process is operating normally. If a fault is detected, then a supervised pattern recognition method (Chiang et al., 2000; 2004; Heo and Lee, 2018) like discriminant analysis, support vector machine or an artificial neural network can be used to find the most likely diagnosis for a faulty sample out of a set of previously known faults.

The distributed configuration is an effective approach for implementing a monitoring method to detect and diagnose faults in a large-scale chemical process system (Daoutidis et al., 2019). In a distributed monitoring method (Fig. 1) the sensors must first be allocated to different potentially overlapping sets called subsystems. This allocation of sensors among different subsystems is called the system decomposition. The monitoring method is then applied to each subsystem in order to come to a local fault decision as to whether a fault is detected by the subsystem or what the subsystem's local diagnosis is. Finally a consensus strategy (Ghosh et al., 2011) is used to combine the local fault decisions of the subsystems so that the monitoring system as a whole can come to a fault decision (Khatib et al., 2018). An example of a distributed fault detection method is given in Khatib and Daoutidis (2020) and a distributed pattern recognition method is given in the supporting information. A key advantage of the distributed configuration is that it can satisfy some of the operational constraints associated with implementing a monitoring method for a large-scale system. For example, transmitting measurements from all the sensors of a large-scale system to a single location for implementing a monitoring method may be impractical or difficult (Blanke et al., 2016; Grbovic et al., 2012). This problem can be addressed by using computers installed at multiple locations to implement the distributed methods (Khatib and Daoutidis, 2019b).

* Corresponding author.
E-mail addresses: khati013@umn.edu (S. Khatib), daout001@umn.edu (P. Daoutidis).

https://doi.org/10.1016/j.compchemeng.2020.107098
0098-1354/© 2020 Elsevier Ltd. All rights reserved.

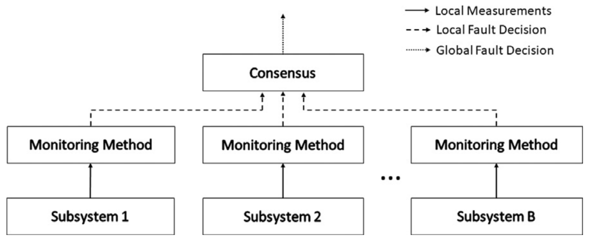

Fig. 1. Distributed monitoring method (Khatib et al., 2018).

The ability of the distributed configuration to satisfy the operational constraints and the performance of the distributed methods depend on the system decomposition (Khatib and Daoutidis, 2019a; 2019b). The sensors of a process unit are allocated to the same subsystem in the decomposition used in Georgakis et al. (1996), Qin et al. (2001), and Zhu et al. (2017). Sparse Principal Component Analysis is used to select the decomposition in Grbovic et al. (2012) where each subsystem contains sensors whose variance can be mostly explained along a different orthogonal direction and the sensors allocated to each subsystem satisfy certain operational constraints. Normal data is used to put sensors with a high correlation between each other in the same subsystem in the decompositions selected in Huang and Yan (2016), Xu et al. (2017), and Rong et al. (2019). Sensors whose measurements have a similar probability distribution are identified using normal data and are allocated to the same subsystem in the decomposition selected in Jiang et al. (2015) and Huang and Yan (2015). Yet, the optimal decomposition for a monitoring task depends on the faults that have to be detected and diagnosed and on the monitoring task itself (Khatib et al., 2018; Khatib and Daoutidis, 2019b). The limitation of the above methods is that they don't take these factors into account.

The above factors can be taken into account if the performance of a monitoring method in detecting or diagnosing faults is directly optimized for selecting a near optimal decomposition subject to user-imposed constraints. This approach is used in the decomposition methods for distributed fault detection that we have proposed in our previous work (Khatib et al., 2018; Khatib and Daoutidis, 2019a; 2020; Khatib, 2020). We also developed an optimization based decomposition method for distributed pattern recognition called Forward Selection for Distributed Pattern Recognition (FSDPR) (Khatib and Daoutidis, 2019b). An integer linear programming problem is first solved in FSDPR to allocate all the sensors to a minimum number of potentially overlapping subsystems with the subsystems having as many sensors as possible allocated to them without violating the imposed constraints. A variant of the forward selection algorithm is then used which runs simulations of a distributed pattern recognition method to identify and remove the set of sensors in each of the subsystems that do not improve the diagnostic performance of the distributed pattern recognition method. The limitation of FSDPR is that it is computationally expensive and applying it to select a decomposition of a system with more than 100 sensors may take a prohibitively large amount of time. This is because the number of constraints in the integer programming problem and the number of pattern recognition method simulations required in the forward selection algorithm can both become prohibitively large when the number of sensors in the system is large.

Large-scale chemical plants can have as many as 1500 sensors (Venkatasubramanian et al., 2003b). Therefore, one of the objectives of this work is to propose a decomposition method that can be applied to such large-scale systems and find a decomposition for which the performance of a distributed pattern recognition method is close to optimal subject to user-imposed constraints. The proposed method will henceforth be referred to as the Extended FSDPR (EFSDPR) method. A filter method (Jović et al., 2015) along with a variant of the forward selection algorithm are used in the first step of EFSDPR to select the sensors to implement the distributed pattern recognition method. Filter methods use a quick to compute metric to rank the variables based on their perceived importance in correctly assigning a new sample to its correct class. In each iteration of the forward selection algorithm, EFSDPR selects the highest ranked sensor that would improve the performance of the pattern recognition method when the sensor is added to the set of sensors that have already been selected. The second and final step of EFSDPR involves the use of scalable algorithms from graph theory (Lewis, 2015; Wu and Hao, 2015) to allocate the selected set of sensors to a minimum number of potentially overlapping subsystems with each subsystem containing a maximum number of sensors without violating the imposed constraints.

In our previous work (Khatib and Daoutidis, 2019b) we also proposed a framework for implementing distributed monitoring methods using computers installed at multiple monitoring locations. The computer at each location has to collect data from a set of sensors and implement the monitoring method for each subsystem allocated to it. We proposed the Subsystem and Sensor Allocation (SASA) method in (Khatib and Daoutidis, 2019b) which solves a mixed integer linear programming problem to find the minimum number of locations required for distributed monitoring and to allocate each subsystem to a location at which its computations will be carried out subject to user-imposed constraints. The limitation of the SASA method is that it requires the user to input the decompositions selected for implementing the distributed methods before it can define the monitoring locations. However, most decomposition selection methods are data-driven and the data needed to select the decompositions may not be available before plant operation starts. Data-driven monitoring methods are typically trained using data collected during a phase just after plant operation commences in which the plant operators gain knowledge about the plant and analyze and classify the collected data (Mason and Young, 2002; Jones-Farmer et al., 2014). Therefore, the number of monitoring locations required and the set of sensors transmitting to each location may have to be determined before the decompositions are selected in order to collect data. The second aim of this work is to propose a method that addresses this limitation of the SASA method. We will refer to the proposed method as the Extended SASA (ESASA) method. The ESASA method first finds the minimum number of monitoring locations required and the set of sensors that each location should be equipped to receive measurements from subject to user-imposed constraints. This is achieved using the same algorithms employed in the second step of the EFSDPR method. Data is then collected at the different monitoring locations and used to select the decompositions needed to implement the distributed methods. Each subsystem will then be allocated to a monitoring location that is equipped to receive measurement from all of the subsystem's sensors. The software for the EFSDPR and ESASA methods can be accessed at z.umn.edu/PDAC. The different methods we've proposed for designing a distributed data-driven monitoring system are described in more detail and compared in Khatib (2020).

The sections of this manuscript are ordered as follows. The EFSDPR and ESASA methods are described in Section 2 and 3 respectively. EFSDPR and ESASA are applied to the benchmark Tennessee Eastman Process (Downs and Vogel, 1993) in Section 4. The framework we employ for implementing distributed monitoring methods using multiple locations is illustrated in the case study on the Tennessee Eastman Process in Section 4. Section 5 contains our final conclusions. Some definitions from graph theory that are needed to fully understand the proposed methods are added to the appendix. The supporting information of this manuscript contains a table describing the notations used. It also contains an example of a distributed pattern recognition method. Pseudocode illustrating the algorithms used in the proposed methods and some additional remarks on the methods are also part of the supporting information. EFSDPR and ESASA are also applied to a large-scale system with 1000 sensors and this second case study is described in the supporting information.

2. Extended forward selection for distributed pattern recognition

The objective of the EFSDPR method is to select a decomposition of a large-scale system so that the performance of a distributed pattern recognition method is near optimal. EFSDPR has two stages. The first stage involves the selection of sensors to implement the distributed pattern recognition method and is described in the first subsection. A feasible near optimal decomposition will then be generated in the second stage of EFSDPR in which the selected set of sensors will be allocated to a minimum number of subsystems. This stage will be described in the second subsection.

2.1. Stage 1: Sensor selection

Let us assume that a large-scale system has m sensors that are indexed from 1 to m. Some of these sensors may degrade the diagnostic performance of the pattern recognition method. In its first stage the EFSDPR method selects the set of sensors that will be used to implement the distributed pattern recognition method. The user must input a faulty dataset to implement this sensor selection stage. The faulty dataset contains samples generated due to the previously diagnosed faults and each sample is labeled based on its diagnosis. The faulty data may be a part of the historical dataset generated over the course of the system's operation or may be generated during the operation of a pilot plant. The faulty data can also be generated using a mathematical model of the system. The user must then specify the filter method that will be used in the sensor selection stage to rank the sensors based on their importance in distinguishing between the previously diagnosed fault classes, with the highest ranked sensor being the most important. Some of the popular filter methods are listed in (Jović et al., 2015). The user must also specify a pattern recognition method whose performance will be evaluated in the sensor selection stage using the faulty dataset for different subsets of the system's sensors. The misclassification rate (MCR) is used to quantify the diagnostic performance of the pattern recognition method. It is the fraction of samples that are incorrectly diagnosed by the pattern recognition method. The MCR of a set of sensors is the MCR calculated when the pattern recognition method is simulated using measurements from the set of sensors.

Let V be the set of selected sensors which is initially empty. Let U be the set of sensors that can be potentially added to V. All the sensors are initially a part of the set U. The first iteration of the sensor selection stage commences by simulating the pattern recognition method to calculate the MCR of V (which is 1 since V is empty). The sensors in the set U are then ranked using the filter method. Let i_{opt} be the index of the highest ranked sensor in U. The sensor i_{opt} is then removed from the set U. The MCR of the set $V \cup \{i_{opt}\}$ is then calculated by simulating the pattern recognition method. If the MCR of $V \cup \{i_{opt}\}$ is less than the MCR of V, then the sensor i_{opt} is added to the set V. This step helps correct any grievous ranking errors in the filter method in which high ranked sensors end up degrading the performance of the pattern recognition method when added to the selected set of sensors. This brings the first iteration to an end. The same procedure is repeated in subsequent iterations of the sensor selection stage. The first pass of the sensor selection stage comes to an end when the set U becomes empty. The second pass of the sensor selection stage can then start using the set V from the end of the first pass and setting $U = [1 \ldots m] \setminus V$ (where $[1 \ldots m]$ represents the set of integers from one to m). The final pass of the sensor selection stage is the one in which no sensors are added to V. The user can also specify a maximum limit on the number of passes employed in the sensor selection stage. If the MCR of the set of all sensors (i.e. $[1 \ldots m]$) is lower than the MCR of the set V generated at the end of final pass, then V is set equal to $[1 \ldots m]$. This brings the sensor selection stage of EFSDPR to an end. The main output of the sensor selection stage is the set of selected sensors V. Another output that can be generated in the sensor selection stage is an m element vector W where the weight $W(i)$ quantifies the relative importance of sensor i in distinguishing between the previously diagnosed fault classes. The weights in W can be set equal to values based on the metric used in the filter method for sensor ranking.

Remark 1. Filter methods can be classified into univariate filter methods and multivariate filter methods (Jović et al., 2015). Univariate filter methods rank each variable based on its individual importance in distinguishing between the different classes. If a univariate filter method is used in EFSDPR, then the sensors need to be ranked only once at the start of the sensor selection stage. Multivariate filter methods, on the other hand, use a metric to quantify the importance of a set of sensors in distinguishing between the fault classes. If a multivariate filter method is used in EFSDPR, then the sensors in the set U will have to be ranked in each iteration of the sensor selection stage. Each sensor $i \in U$ will be ranked based on the value of the multivariate filter metric for the set $V \cup \{i\}$.

Remark 2. It is possible that the pattern recognition method the user wishes to implement in the distributed configuration may be computationally expensive to train or implement online. If such a pattern recognition method is used in the sensor selection stage, then the computations involved in calculating the MCRs of multiple sensor sets may become prohibitively expensive. A workaround would be to use a computationally efficient pattern recognition method to evaluate the performance of the different sensor sets generated in the iterations of the sensor selection stage. The computationally expensive pattern recognition method can then be simulated to compare the MCRs of the set of sensors selected at the end of the final pass and the set of all sensors. Similarly if the number of samples in the faulty dataset is very large, then implementing the iterations of the sensor selection stage may become prohibitively expensive. In such a case the user can randomly sample an appropriate number of faulty samples from the large dataset for each faulty condition in order to create a smaller faulty dataset for implementing the sensor selection stage efficiently. If the user still perceives that the sensor selection stage of EFSDPR is computationally expensive, then the user may use a computationally more efficient variable selection method or may select all the system's sensors to implement the distributed pattern recognition method.

2.2. Stage 2: Sensor allocation

The objective of the second and final stage of EFSDPR is to find a feasible decomposition of the selected sensors for which the diagnostic performance of the distributed pattern recognition method is close to optimal. The performance of a distributed pattern recognition method tends to degrade as the number of subsystems in the selected decomposition increases because it is difficult to combine the different local diagnoses that a large number of subsystems can come to and find the correct diagnosis (Khatib and Daoutidis, 2019b). We would also like to maximize the sum of the weights of sensors in each subsystem so that maximum amount of information is available to the subsystem to make a local diagnosis. Therefore, the ideal decomposition we would like to select is one in which all the selected sensors are allocated to a minimum number of potentially overlapping subsystems with the sum of the weights of sensors in each subsystem being maximum subject to the user imposed constraints.

Cannot-link constraints are used in EFSDPR to represent the user imposed constraints that prohibit two sensors from being allocated to the same subsystem. For example, if two sensors cannot transmit to the same location due to the system's layout, then a cannot-link constraint can be imposed between the two sensors so that the sensors are allocated to different subsystems and the pattern recognition method is applied to these subsystems at different locations. If two sensors have different sampling rates, then a cannot-link constraint can be imposed to allocate the sensors to different subsystems and these subsystems can then implement the pattern recognition method using different sampling rates. If different pattern recognition methods need to be applied to different groups of sensors, then cannot-link constraints can be used to force sensors from different groups into different subsystems. A binary symmetric adjacency matrix (A) is used to encode the cannot-link constraints where:

$$A(i_1, i_2) = \begin{cases} 0, & \text{if sensor } i_1 \text{ and sensor } i_2 \text{ can't be} \\ & \text{allocated to the same subsystem} \\ 1, & \text{otherwise} \end{cases} \tag{1}$$

The diagonal elements of A are set equal to one. An upper limit (N_{lim}) can also be placed on the number of sensors allocated to each subsystem if the pattern recognition method is computationally expensive. Therefore, to summarize the inputs of the second stage of EFSDPR are the set of selected sensors V, the adjacency matrix A, the upper limit N_{lim} and the weights in W.

The cannot-link constraints can also be represented using a vertex weighted undirected graph in which the vertices represent the sensors, the i^{th} vertex is given the weight $W(i)$ and the absence of an edge between two vertices represents a cannot-link constraint between the corresponding sensors. Let G_S be the subgraph of the graph of cannot-link constraints induced by the set of selected sensors V and let G'_S be the complement of G_S. A decomposition satisfies the cannot-link constraints if the sensors in each of its subsystems form a clique in G_S or an independent set in G'_S. Therefore, a feasible decomposition must be a clique covering in G_S. If the upper limit N_{lim} isn't imposed, then the ideal decomposition would be a minimum clique covering of G_S in which the sum of weights of the vertices in each clique is maximum. Therefore, we will henceforth refer to the problem of finding the ideal decomposition as the Minimum Clique Covering with Maximum Cliques (MCCMC) problem.

The problem of finding a near optimal solution to the MCCMC problem is simplified by dividing it into two subproblems in EFSDPR. The objective of the first subproblem is to find a feasible non-overlapping decomposition (i.e. a decomposition in which each of the selected sensors is allocated to one of the subsystems) with minimum number of subsystems. If the upper limit N_{lim} isn't imposed, then an optimal solution to the first subproblem would be an optimal graph coloring of G'_S. Therefore, a near optimal solution to the first subproblem can be found by applying a graph coloring algorithm (Lewis, 2015) to G'_S after the algorithm is modified to place an upper limit N_{lim} on the number of vertices that are assigned the same color. The sensors (or vertices) that are assigned the same color by the graph coloring algorithm will then be a part of the same subsystem in the non-overlapping decomposition. The objective of the second subproblem is to find the additional set of selected sensors that should be added to each subsystem generated after solving the first subproblem so that the sum of weights of the additional set of sensors is maximum and all the constraints are satisfied. Let S_j be the set of sensors in the j^{th} subsystem of the non-overlapping decomposition generated after solving the first subproblem. Let Ω_j be the set of sensors not in S_j that have no cannot-link constraints with any of the sensors in S_j. If no upper limit N_{lim} is imposed, then an optimal solution to the second subproblem for the j^{th} subsystem is a maximum weight clique of the subgraph of G_S induced by vertices in the set Ω_j. Therefore, a near optimal solution to the second subproblem can be found by applying a maximum weight clique algorithm (Bomze et al., 1999; Wu and Hao, 2015) after the algorithm is modified to include an upper limit on the number of vertices in the clique. The decomposition selected by EFSDPR is the one generated after the additional sensors are added to each subsystem of the non-overlapping decomposition. A large number of graph coloring and maximum weight clique algorithms have been proposed and some of these algorithms are reviewed in (Lewis, 2015) and (Bomze et al., 1999; Wu and Hao, 2015) respectively. A simple approximate greedy algorithm for solving the MCCMC problem is described below.

2.2.1. *Approximate greedy algorithm*

The approximate greedy algorithm uses slightly modified versions of the Recursive Largest First (RLF) algorithm (Leighton, 1979) and the maximum clique algorithm of (Johnson, 1974) to find near optimal solutions to the first and second subproblems of the MCCMC problem respectively. The RLF algorithm generates a subsystem and allocates sensors to the subsystem in each of its iterations. A sensor that is more constrained and that has a larger weight is allocated to a subsystem that is generated in an earlier iteration. The RLF algorithm ends when all the selected sensors have been allocated to one of the subsystems.

The maximum clique algorithm of (Leighton, 1979) is applied to add more sensors to each subsystem of the non-overlapping decomposition generated by the RLF algorithm. Let us consider that the maximum clique algorithm is applied to j^{th} subsystem of this decomposition. The set of selected sensors that can be added to the subsystem without violating the constraints is first identified. A sensor from this set is selected and added to the subsystem. The sensor with smaller number of cannot-link constraints and larger weight is given priority for early addition into the subsystem. The same procedure is repeated and the maximum clique algorithm comes to an end when no more sensors can be added to the subsystem. The resulting subsystem is then the j^{th} subsystem of the decomposition selected by the approximate greedy algorithm. Pseudocode of the approximate greedy algorithm describes the RLF and maximum clique algorithms in greater detail and is part of the supporting information.

Remark 3. A more advanced algorithm to solve the MCCMC problem based on the Tabu search heuristic is described in the supporting information. The algorithm first partitions G_S into its connected components (Hopcroft and Tarjan, 1973). The MCCMC problem is then solved for each connected component by using slightly modified versions of the TABUCOL algorithm (Lewis, 2015) and the Multi Neighborhood Tabu Search algorithm (Wu et al., 2012) to solve the first and second subproblems respectively. Both the approximate greedy algorithm and the algorithm based on the Tabu search heuristic can be applied to systems with a large number of sensors. The advantage of the algorithm based on the Tabu search heuristic over the approximate greedy algorithm is that it should find a better solution to the MCCMC problem if the approximate greedy algorithm doesn't find the optimal solution. The limitation of the algorithm based on the Tabu search heuristic is that it is more difficult to implement than the approximate greedy algorithm.

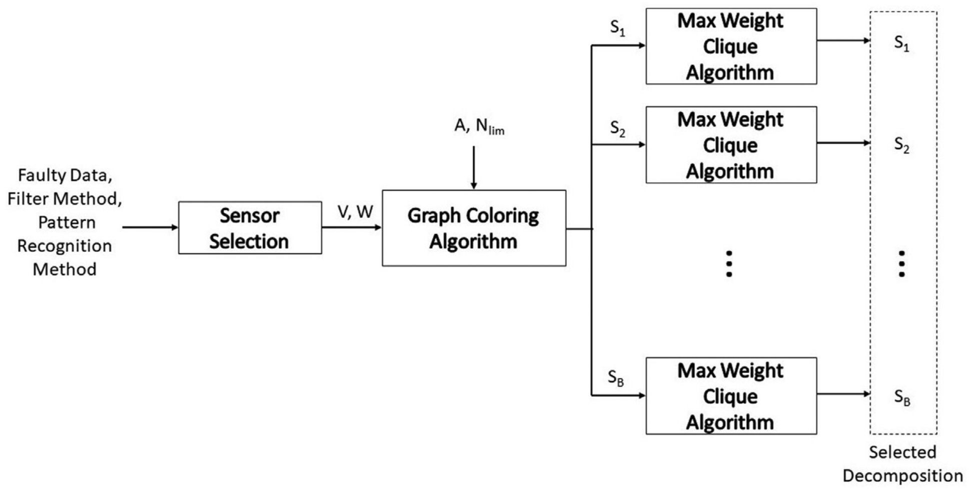

Fig. 2. Schematic of the EFSDPR method.

A schematic of the EFSDPR method can be seen in Fig. 2. The pseudocode of EFSDPR is added to the supporting information.

Remark 4. If new faults are observed after a distributed pattern recognition method is trained to diagnose a specific set of faults and if the practitioner wishes to retrain the distributed pattern recognition method to diagnose these new faults, then the EFSDPR method should be reapplied to select the decomposition. This is because some of the sensors that were originally excluded for implementing the distributed pattern recognition method may become essential for diagnosing the new faults. The EFSDPR method can be applied online if new faults are periodically observed and if the decomposition has to therefore be periodically updated.

Remark 5. The advantage of the proposed EFSDPR method over the existing FSDPR method is that it can scale to systems with larger number of sensors. The scalable graph theoretic algorithms used in EFSDPR solve the MCCMC problem within a few seconds even for systems with a large number of sensors. FSDPR, on the other hand, allocates sensors among a minimum number of subsystems by solving an integer linear optimization problem which may have a prohibitively large number of constraints if the number of sensors is large and if a large number of cannot-link constraints are imposed. EFSDPR also requires smaller number of pattern recognition method simulations when compared to FSDPR because a filter method is incorporated into the sensor selection stage of EFSDPR. The forward selection algorithm of FSDPR uses a greedy approach in which the best addition of a sensor to a subsystem in an iteration is identified by using multiple simulations in which every possible sensor addition to a subsystem is considered. Therefore, the number of simulations required in FSDPR increases polynomially with an increase in the number of sensors. On the other hand, only a single simulation is required in each iteration of the sensor selection stage of EFSDPR because the filter method identifies a single sensor that should be considered for selection in each iteration. Therefore, if the maximum number of passes in the sensor selection stage of EFSDPR is set equal to a small value, then the number of simulations required in EFSDPR would only increase linearly with an increase in the number of sensors.

3. Extended subsystem and sensor allocation

Distributed monitoring methods are capable of being implemented using multiple interacting computers that can be installed at multiple locations. This is useful if transmitting measurements from all the sensors of a large-scale system to a single location is impractical or difficult. Each monitoring location is fitted with a computer to implement the monitoring methods and to display results. If multiple monitoring locations are employed for distributed monitoring, then each subsystem of the decompositions selected for distributed monitoring must be allocated to one of the monitoring locations. The computer at each monitoring location will implement the monitoring methods for the subsystems allocated to the location. Each monitoring location will also be equipped with some storage capacity to store the measurements from a subset of the system's sensors. The objectives of the ESASA method are to determine the number of monitoring locations required, to allocate the subsystems among the monitoring locations, to find the set of sensors that must transmit to each monitoring location, and to determine the monitoring location at which each sensor will store its data. The framework for implementing the distributed methods using multiple monitoring locations is described in greater detail in the case study.

Let us assume that plant operation has not started and we need to determine the number of monitoring locations required for distributed monitoring. We also need to determine the set of sensors that the computer at each monitoring location should be equipped to receive measurement from. This set of sensors will henceforth be referred to as the set of sensors allocated to the monitoring location. Every sensor of the system must have the capability to transmit to at least one monitoring location. The primary objective of the ESASA method at this stage is to minimize the number of monitoring locations and the secondary objective is to maximize the number of sensors allocated to each monitoring location subject to user imposed constraints. Smaller number of monitoring locations is preferred because the costs associated with purchasing and installing hardware will be smaller if the number of monitoring locations is smaller. The number of sensors allocated to each monitoring location is maximized so that the computer at each monitoring location will have access to maximum amount of information.

Cannot-link constraints are used to define the pairs of sensors that cannot transmit to the same location due to the system layout. The adjacency matrix A_M encodes these constraints:

$$A_M(i_1, i_2) = \begin{cases} 0, & \text{if sensor } i_1 \text{ and sensor } i_2 \text{ can't} \\ & \text{transmit to the same location} \\ 1, & \text{otherwise} \end{cases} \tag{2}$$

The diagonal elements of A_M are set equal to one. The user can also place an upper bound N^M_{lim} on the number of sensors allocated to a monitoring location if the cost associated with equipping a computer to receive measurements from a large number of sensors is significant. If the user has process knowledge on the importance of each sensor for fault detection and diagnosis, then a weight can be assigned to each sensor in the vector W_M that quantifies its importance for monitoring. The sum of the weights of sensors allocated to a monitoring location can then be maximized instead of the number of sensors. However, it is unlikely that this knowledge will be available before the system starts operation and therefore W_M is usually a vector of ones. The ESASA method allocates a close to maximum number of sensors to a close to minimum number of monitoring locations by applying a MCCMC algorithm using the inputs A_M, N_{lim}^M, W_M and considering that all the sensors have to be allocated to at least one monitoring location. The MCCMC algorithm will be allocating all the system's sensors to monitoring locations in the ESASA method as opposed to the sensors selected for fault diagnosis to subsystems in the EFSDPR method. Let N_M be the number of monitoring locations selected and M_k^{sensor} be the set of sensors allocated to the k^{th} monitoring location using the ESASA method.

The next objective of the ESASA method is to determine the monitoring location at which each sensor will store its data. The task of storing data collected from all the system's sensors at multiple locations will be referred to as distributed data storage. The system decomposition for distributed data storage has N_M subsystems with S_k^{store} being the set of sensors allocated to the k^{th} subsystem of this decomposition. The k^{th} monitoring location will store data collected from sensors in the set S_k^{store}. The set S_k^{store} must be a subset of M_k^{sensor} because the k^{th} monitoring location can only store data acquired from sensors it is equipped to receive measurements from. Therefore, one possible approach for selecting the decomposition for distributed data storage would be to set $S_k^{store} = M_k^{sensor}$ i.e. each monitoring location stores data from the all the sensors allocated to it. However, it is possible that many of the sensors will be allocated to multiple monitoring locations. Therefore, if data collected from the sensors allocated to each monitoring location are permanently stored at the location and if some of the sensors are allocated to multiple locations, then there will be identical data stored at different locations. Ideally we would like to store the data from each sensor at one of the monitoring locations. Therefore, one of the objectives of the ESASA method is to select a feasible non-overlapping decomposition for distributed data storage in which the number of senors in each subsystem is as close to equal as feasible. Equal partition of the sensors into subsystems is favored for distributed data storage so that the computational load of storing data is distributed equally among the monitoring locations. The ESASA method uses a greedy algorithm to select the decomposition which will be referred to as the Greedy Equal Allocation (GEA) algorithm and is described in the only subsection of this section.

Plant operation can begin after we've determined the number of monitoring locations, the allocation of sensors among the monitoring locations and the location at which each sensor's data will be stored. Data-driven monitoring methods are typically implemented in two phases (Mason and Young, 2002; Jones-Farmer et al., 2014). Phase 1 is a period that begins just after plant operation commences during which the operators gain knowledge about the plant, collect data and classify each sample in the collected data as being representative of normal operation or as having been caused due to a particular fault. The operator must then decide which monitoring methods should be used for fault detection and diagnosis and calculate the parameters needed to implement these methods online. Therefore, after the data is collected we need to select the decompositions for distributed fault detection and distributed pattern recognition subject to the imposed constraints. The methods of Khatib and Daoutidis (2019a, 2020), and Khatib (2020) can be applied using the collected data to select a feasible decomposition for distributed fault detection for which the detection performance is near optimal. The FSDPR (Khatib and Daoutidis, 2019b) and EFSDPR methods, on the other hand, can be used to select a feasible decomposition for distributed pattern recognition for which the diagnostic performance is close to optimal. The sensors in each subsystem of the selected decompositions must be a subset of the sensors allocated to at least one of the monitoring locations. This is because the computer at a monitoring location must be equipped to receive measurements from all the sensors of a subsystem if it is to implement the monitoring method for the subsystem. This constraint is imposed by placing a cannot-link constraint between each pair of sensors that are not allocated to the same monitoring location. Let the adjacency matrix A_M^* encode these cannot-link constraints. The matrix A_M^* is:

$$A_M^*(i_1, i_2) = \begin{cases} 1, & \text{if sensor } i_1 \in M_k^{sensor} \text{ and sensor } i_2 \in M_k^{sensor} \text{ for} \\ & \text{some } k \in [1 \dots N_M] \\ 0, & \text{otherwise} \end{cases} \tag{3}$$

The cannot-link constraints encoded in A_M^* must be a subset of the cannot-link constraints imposed in selecting the decompositions (i.e. $A_M^* - A$ must be non-negative).

After the decompositions for distributed monitoring have been selected, then each subsystem has to be allocated to one of the monitoring locations whose computer will implement the monitoring method for the subsystem. Each subsystem will be allocated to a monitoring location which is equipped to receive measurements from the subsystem's sensors. The ESASA method uses the GEA algorithm to allocate subsystems to monitoring locations with the objective of equally allocating subsystems and sensors among the monitoring locations so that the computational load of implementing the distributed methods is divided equally among the monitoring locations. It is possible that some of the sensors allocated to a monitoring location may not be part of any of the subsystems allocated to it. Therefore, after the subsystems have been allocated to the monitoring locations, then each set M_k^{sensor} has to be updated to only include the sensors of the subsystems allocated to the k^{th} monitoring location. If S_j^k $(j \in [1 \dots B_k])$ is the set of senors in one of the subsystems allocated to the k^{th} monitoring location, then M_k^{sensor} is updated using:

$$M_k^{sensor} = \bigcup_{j=1}^{B_k} S_j^k \tag{4}$$

The set M_k^{sensor} will now represent the set of sensors that will transmit to the k^{th} monitoring location after the set has been updated. After the subsystems are allocated to the monitoring locations and the parameters needed to implement the distributed methods are calculated using the collected data, then phase 2 of monitoring can begin. In phase 2 newly generated samples from the plant will be used by the distributed fault detection and diagnosis methods to determine whether process operation is faulty and what the cause of a faulty sample is.

A schematic description of the ESASA method is given in Fig. 3. The GEA algorithm is described in the subsection below.

3.1. Greedy equal allocation algorithm

The GEA algorithm has two applications in the ESASA method. First, it selects the decomposition for distributed data storage by assigning each sensor to a monitoring location which will store the sensor's data. Second, it is used to assign each subsystem to a monitoring location. This includes the subsystems in the decompositions selected for distributed data storage, fault detection and fault diagnosis. The algorithm minimizes an objective function to

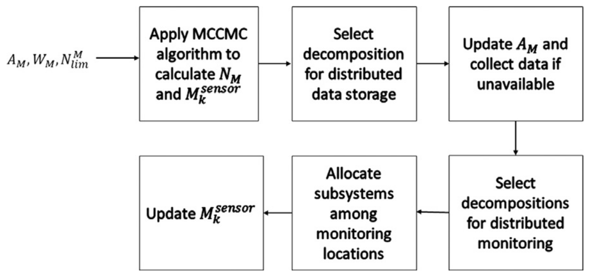
A_M, W_M, N_{lim}^M
Apply MCCMC algorithm to calculate N_M and M_k^{sensor}
Select decomposition for distributed data storage
Update A_M and collect data if unavailable
Select decompositions for distributed monitoring
Allocate subsystems among monitoring locations
Update M_k^{sensor}

Fig. 3. Schematic of the ESASA method.

find the optimal assignment of sensors/subsystems that increases if the number of sensors/subsystems assigned to a monitoring location exceeds the number of sensors/subsystems per monitoring location. This is done so that the computational load is divided equally among the monitoring locations. For an assignment to be feasible each sensor/subsystem must be assigned to only one of the monitoring locations. The algorithm starts by generating an initial assignment in which each sensor/subsystem is assigned to every possible monitoring location it can be assigned to. A sensor/subsystem can be assigned to a monitoring location if the monitoring location is equipped to receive its measurements. The algorithm then finds the set of sensors/subsystems (denoted by J_{rem}) that have been assigned to multiple monitoring locations in the initial assignment. In the first iteration, the algorithm generates candidate assignments by considering every possible unassignment of a sensor/subsystem in J_{rem} from one of the monitoring locations it is assigned to in the initial assignment. The algorithm then calculates the objective function for each of these candidate assignments. The candidate assignment with the lowest objective function value then becomes the initial assignment for the next iteration and the set J_{rem} is updated based on this new initial assignment. The same procedure is repeated in subsequent iterations and the algorithm stops when the initial assignment becomes feasible (i.e. when $J_{rem} = \emptyset$). The feasible initial assignment is then output by the algorithm. A more detailed description of the algorithm is given in the pseudocode of the algorithm which is added to the supporting information.

Remark 6. If the cannot-link constraints encoded in A_M^* are the only constraints imposed in the EFSDPR method, then the second stage of EFSDPR does not have to be applied. The decomposition selected by EFSDPR in this scenario will have N_M subsystems. If S_j^{diag} represents the set of sensors in the j^{th} subsystem of the decomposition, then:

$$S_j^{diag} = M_j^{sensor} \cap V \quad \forall j \in [1 \ldots N_M] \tag{5}$$

4. Case studies

The EFSDPR and ESASA methods are first applied to the benchmark Tennessee Eastman Process (TEP) (Downs and Vogel, 1993) in this section. The proposed methods are also applied to an artificial large-scale system with 1000 sensors in the supporting information. The calculations involved in the case studies are carried out in MATLAB using a 2.7 GHz Intel Core i7-7500U processor.

4.1. Tennessee eastman process

TEP has five major process units which include a reactor, a condenser, a separator, a compressor and a stripper. The process is described in Downs and Vogel (1993). The datasets generated in Khatib and Daoutidis (2019b) using the TEP simulator of Bathelt et al. (2015) are used in this case study. The datasets consist of measurements from 82 sensors of TEP that are indexed from 1 to 82. Two of the datasets (dataset 1 and dataset 2) were generated by simulating 28 different faults with each dataset containing 960 samples per fault class. The decomposition in the EFSDPR method is selected using dataset 1 and the diagnostic performance of different decompositions is tested and compared using dataset 2. The sensors (along with their indexes) and faults of TEP are listed in the supporting information of Khatib and Daoutidis (2019b) and a process flow diagram can be found in Bathelt et al. (2015).

4.1.1. EFSDPR

A distributed pattern recognition method is used for fault diagnosis of TEP which uses discriminant analysis for fault diagnosis in each subsystem and combines the local diagnoses of the subsystems using a weighted voting consensus strategy (Ghosh et al., 2011). This distributed pattern recognition method is described in the supporting information of Khatib and Daoutidis (2019b). Let us assume that all the sensors have the same sampling rate. Let us also assume that the layout of TEP makes it difficult to transmit measurements from a sensor of the reactor and a sensor of the stripper to the same location. Therefore, cannot-link constraints are imposed between any sensor from the set $C_R = \{7-9, 21, 22, 46-49, 81, 82\}$ (that are related with the reactor) and any sensor from the set $C_S = \{15-19, 37-41, 80\}$ (that are related with the stripper). EFSDPR is applied to select the decomposition of TEP for distributed pattern recognition subject to these cannot-link constraints. The maximum number of passes in the first stage of EFSDPR is set equal to two. The Laplacian Score (He et al., 2006) is the filter method used in the EFSDPR method for this case study. It is implemented using the *fsulaplacian* function in MATLAB. The distance metric used in the *fsulaplacian* function is the Mahalanobis distance. All other optional input parameters of the *fsulaplacian* function are set equal to their default values. The approximate greedy algorithm is used to solve the MCCMC problem in this case study. EFSDPR is applied using dataset 1 which is divided into two equally sized datasets (dataset 1a and dataset 1b) with samples generated at even sampling instants being part of dataset 1a and samples generated at odd sampling instants being part of dataset 1b. This is done to make sure that the trends of the time series in dataset 1 are there in dataset 1a and dataset 1b (Khatib and Daoutidis, 2019b). Dataset 1a is used to implement the filter method and to train the discriminant analysis method in the simulations of the sensor selection stage of EFSDPR. Dataset 1b, on the other hand, is used to test the discriminant analysis method to calculate the MCRs of the different sensor sets generated in the sensor selection stage of EFSDPR. The decomposition selected using EFSDPR has two subsystems with 69 sensors being allocated to the two subsystems S_1^{diag} and S_2^{diag}. The specific form of this decomposition is given in the supporting information.

The performance of the EFSDPR method is compared with the FSDPR method. The two methods will be compared based on their computation time and on the MCR of the decomposition they select. The MCR of a decomposition is calculated as being the fraction of samples in dataset 2 that are incorrectly diagnosed when the distributed pattern recognition method is simulated using the decomposition. The computation time of EFSDPR is 142 seconds whereas the computation time of FSDPR is 3059 seconds. The EFSDPR method has a smaller computation time in this case because the sensor selection stage of EFSDPR requires much smaller number of simulations of the pattern recognition method than the variant of the forward selection algorithm employed by the FSDPR method to select sensors in each subsystem. The use of the filter method in the forward selection algorithm of EFSDPR helps keep the number of pattern recognition method simulations low in EFSDPR when compared to the FSDPR method which does not use a filter method. The MCR of the decomposition selected using EFS-

DPR is 0.0561 whereas the MCR of the decomposition selected using FSDPR is 0.0552 (Khatib and Daoutidis, 2019b). Therefore, the EFSDPR and FSDPR methods are comparable in terms of diagnostic performance in this case. If the sensor selection stage wasn't applied in the EFSDPR method and instead all 82 sensors were selected in the first stage, then the EFSDPR method would select a decomposition whose MCR is 0.0640. Therefore, sensor selection helps improve the diagnostic performance of the distributed pattern recognition method in this case besides reducing the number of sensors required to implement the method.

4.1.2. ESASA

This case study will now be used as an example to describe the ESASA method and the framework for implementing distributed monitoring methods using computers installed at multiple locations. Let us assume that process operation has not begun and no process data is available in this case study. The SASA method cannot be applied in this scenario because it requires data before process operation begins to first select decompositions for distributed monitoring and then determine the number of monitoring locations required and the subsystem and sensor allocation. The ESASA method, on the other hand, is tailored to deal with the scenario in which no data is available before process operation begins. The MCCMC problem is first solved subject to cannot-link constraints that prevent any sensor of C_S from transmitting to the same monitoring location as any sensor of C_R. The approximate greedy algorithm and the Tabu search based algorithm find the same solution to the MCCMC problem in this case. Two monitoring locations are required for distributed monitoring with $M_1^{sensor} = [1 \dots 82] \setminus C_S$ and $M_2^{sensor} = [1 \dots 82] \setminus C_R$. A computer is installed at each monitoring location. Wired connections are used to transmit measurements from the sensors in M_k^{sensor} to the computer at the k^{th} monitoring location. The computers also have the ability to communicate monitoring results with each other over a wireless network. The matrix A_M encoding the cannot-link constraints is updated using Eq. 3 ($A_M^* = A_M$ in this case). The GEA algorithm is also applied to select the decomposition for distributed data storage and the specific form of this decomposition is given in the supporting information. The selected non-overlapping decomposition has two subsystems (S_1^{store} and S_2^{store}), each with 41 sensors. The first and second monitoring locations will store data collected from sensors of S_1^{store} and S_2^{store} respectively.

Process operation is now allowed to begin and data is collected in phase 1. After sufficient amount of data has been collected, the collected data is taken offline for analysis. The samples in the collected data are classified as being normal or faulty. The faulty samples are further divided into different classes with the samples in each class being generated by the process due to a specific fault. Outlier detection methods, change-point methods and clustering methods can be used to classify the samples (Jones-Farmer et al., 2014; Jain et al., 1999). After data analysis, decompositions for distributed fault detection and distributed pattern recognition are selected by applying the Performance Driven Agglomerative Clustering (PDAC) and EFSDPR methods respectively using the collected data subject to cannot-link constraints encoded in A_M^*. PDAC (Khatib and Daoutidis, 2019a) is a method that directly optimized the performance of a distributed fault detection method in detecting a set of faulty samples in order to find a feasible near optimal decomposition for distributed fault detection. The decomposition selected for distributed fault detection has nine subsystems ($S_1^{det}, S_2^{det}, \cdots, S_9^{det}$) and the specific form of this decomposition is given in the supporting information of (Khatib and Daoutidis, 2019a). The distributed fault detection method (Khatib and Daoutidis, 2019a) used in this case applies Principal Component Analysis for fault detection in each of the nine subsystems. The veto-voting consensus strategy (Khatib and Daoutidis, 2019a) is then used to combine the local detection results of the nine subsystems where the detection of a sample as being faulty by at least one of the nine subsystems results in a fault being detected by the monitoring system as a whole. The parameters needed to implement the monitoring method in each subsystem are calculated using the collected data after the decompositions are selected. The subsystems are then allocated among the two monitoring locations using the GEA algorithm and the sets M_1^{sensor} and M_2^{sensor} are updated using Eq. 4.

Fig. 4 shows the allocation of subsystems and sensors between the two monitoring location and is a schematic of the distributed configuration used to monitor TEP in this case study. In phase 2 the computer at each monitoring location will receive a new sample of measurements from the sensors allocated to it. Each fault detection designated subsystem in both monitoring locations will apply statistical hypothesis tests using its sensors' measurements in the new sample to check if the new sample is generated due to a fault. If a fault is detected by at least one of the subsystems of a monitoring location, then this information will be communicated to the other monitoring location so that the process of diagnosing the fault can commence at both monitoring locations. After the fault is detected, each subsystem designated to implement the pattern recognition method will use its sensors' measurements in the new faulty sample to reach a diagnosis as to which of the 28 faults is most likely or if an unknown fault is expected. The local diagnosis of each subsystem will be communicated to the other monitoring location. The weighted voting consensus strategy will be implemented in both monitoring locations using the local diagnoses of the two subsystems so that a final diagnosis is reached. Fault identification methods (Chiang et al., 2001; Bersimis et al., 2017) (represented by the Fault ID block in Fig. 4) will also be implemented at each monitoring location after the fault is detected to find the sensors that are out of control and to rank the sensors based on their deviation from normal operation. This information can be used by the operators to reach a diagnosis if the fault is not one of the previously diagnosed 28 faults. For example, out of control sensors in each monitoring location can be identified using univariate statistical tests and the out of control sensors can then be ranked based on the magnitude of the Z-score of their measurements (i.e. the magnitude of mean deviation scaled by the standard deviation).

One of the advantages of the distributed monitoring framework is that it offers some tolerance to hardware or transmission failures. For example, if the computer at one of the monitoring locations fails, the computers at the other monitoring locations will still have the capacity to detect and diagnose faults. If the computers are unable to communicate information due to the failure of the wireless network, then the operators can still use the local monitoring results at each location to detect and diagnose faults.

4.2. System with 1000 sensors

The proposed EFSDPR and ESASA methods are also applied to an artificial system with 1000 sensors. A detailed description of this case study and its results and discussions can be found in the supporting information. The main result of this case study is that the computation time of the proposed methods is within reasonable limits. The computation time of EFSDPR is 6697 seconds and the algorithms used in the ESASA method are able to generate solutions within a few seconds. Therefore, the proposed methods are shown to be applicable to systems with a large number of sensors. The FSDPR method, on the other hand, is not able to select a decomposition within 24 hours in this case study. This is because the number of constraints in the integer optimization problem solved in FSDPR for sensor allocation (i.e. in the order of magnitude of 10^6) is prohibitively large and the number pattern recog-

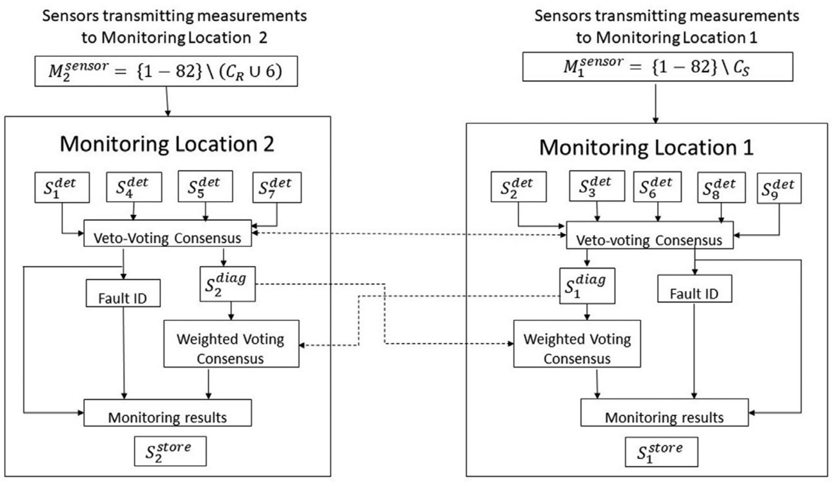

Fig. 4. Schematic of the distributed configuration used for monitoring TEP.

nition method simulations (i.e. in the order of magnitude of 10^4) required in the sensor selection step of FSDPR is also large. The SASA method and the proposed ESASA method are also applied and compared in this case study. The advantage of the proposed ESASA method over the SASA method in this case study is that it selects a smaller number of monitoring locations. On the other hand, the advantage of the SASA method over the ESASA method in this case study is that it imposes smaller number of constraints in selecting the decompositions.

5. Conclusions

This paper describes extensions of our previously developed distributed data-driven monitoring methods aiming to make them scalable to large-scale systems. The proposed EFSDPR method is able to select the decomposition of a large-scale system for which the diagnostic performance of a distributed pattern recognition method is close to optimal. The proposed ESASA method finds the number of locations required for distributed monitoring of a large-scale system, the set of sensors that need to transmit to each monitoring location, the monitoring location at which data from each sensor will be stored and the monitoring location at which the computations of each subsystem will be implemented. The proposed methods are fully automated. Two case studies are used to demonstrate their performance and computational efficiency.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

CRediT authorship contribution statement

Shaaz Khatib: Conceptualization, Methodology, Software, Formal analysis, Validation, Visualization, Investigation, Writing - original draft. **Prodromos Daoutidis:** Supervision, Conceptualization, Writing - review & editing, Funding acquisition, Resources.

Acknowledgments

Partial financial support from NSF-CBET is gratefully acknowledged.

Appendix A. Some Definitions from Graph Theory

- **Undirected Graph:** An undirected graph G is a set of vertices linked by a set of edges with none of the edges having any specific orientation. A vertex weighted undirected graph is one in which each vertex is assigned a weight. An example of a vertex weighted undirected graph is given in Fig. A.5. The set of vertices is {1, 2, 3, 4, 5} and the set of edges is {(1, 4), (2, 3), (2, 4), (3, 4)} in the example. The weight assigned to vertex i is w_i in Fig. A.5.
- **Neighborhood:** A vertex is a neighbor of another vertex if there is an edge directly linking the two vertices. The set of neighbors of a vertex is its neighborhood. For example, vertex 3 is a neighbor of vertex 2 in Fig. A.5 and the neighborhood of vertex 2 is {3, 4}.
- **Degree:** The degree of a vertex is the cardinality of its neighborhood. The degree of vertex 2 in the example graph is 2.
- **Adjacency Matrix:** An undirected graph can be encoded in a square binary symmetric matrix called the adjacency matrix. The number of rows of the adjacency matrix is the number of vertices in the graph. A vertex directly links vertices i and j if and only if $A(i, j) = 1$.
- **Complement of a Graph:** The undirected graph G' is a complement of G if G' has the same vertex set as G and an edge directly links two vertices in G' if and only if an edge does not directly link the same two vertices in G. The complement of the undirected graph of Fig. A.5 is shown in Fig. A.6.
- **Induced Subgraph:** G_S is an induced subgraph of G if the vertex set of G_S is a non-empty subset of the vertex set of G and if an edge links any two vertices in G_S if and only if an edge links

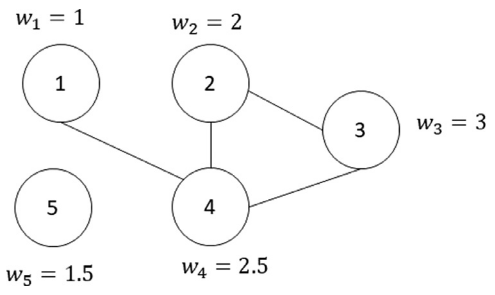

Fig. A.5. Example of a vertex weighted undirected graph.

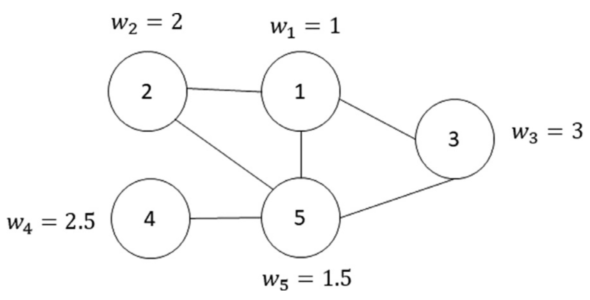

Fig. A.6. Complement of the graph of Fig. A.5.

the same two vertices in G. The subgraph of G induced by a vertex set V_S is the induced subgraph with vertex set V_S.

- **Path:** It is a distinct sequence of edges which link a distinct sequence of vertices in a graph.
- **Connected Component:** The connected component of an undirected graph G is an induced subgraph of G in which there is a path between every two vertices of the subgraph and there is no path in G between any one vertex of the subgraph and any one of the remaining vertices of G. The Fig. A.5 graph can be partitioned into 2 connected components with vertex sets {1, 2, 3, 4} and {5} respectively.
- **Clique:** A clique is a subset of the vertices of a graph such that an edge directly links every pair of vertices in the subset. Vertices 2, 3 and 4 form a clique in the example graph of Fig. A.5.
- **Independent Set:** An independent set is a subset of the vertices of a graph such that there is no edge linking any of the vertices in the subset. Vertices 2, 3 and 4 form an independent set in complement graph of Fig. A.6. A set of vertices form a clique in a graph if and only if the same set of vertices form an independent set in the complement of the graph.
- **Maximum Clique:** The maximum clique of a graph is a clique with the largest number of vertices. The maximum weight clique of a graph is a clique whose vertices have the maximum sum of vertex weights. The clique {2, 3, 4} is the maximum weight clique of the example graph of Fig. A.5.
- **Clique Covering:** It is a set of cliques of G such that each vertex of G is a member of at least one of the cliques. The cliques {1, 4}, {2, 3, 4} and {5} form a clique covering in the example graph of Fig. A.5.
- **Clique Covering Number:** The clique covering number of a graph is the minimum number of cliques that a graph can be partitioned into. The clique covering number of the example graph of Fig. A.5 is 3.
- **Minimum Clique Covering:** A minimum clique covering is a clique covering in which the number of cliques is equal to the clique covering number. The cliques {1, 4}, {2, 3, 4} and {5} also form a minimum clique covering in the example graph of Fig. A.5.
- **Graph Coloring:** A graph coloring is a partition of the vertices of a graph into independent sets. The vertices in the same independent set of a graph coloring are given the same label or 'color'. The independent sets {1}, {2, 3, 4} and {5} form a graph coloring in the complement graph of Fig. A.6.
- **Chromatic Number:** The chromatic number of a graph is the minimum number of independent sets that the graph can be partitioned into. The clique covering number of a graph is equal to the chromatic number of its complement graph. An optimal graph coloring of a graph is a graph coloring in which the number of independent sets is equal to the chromatic number.

Supplementary material

Supplementary material associated with this article can be found, in the online version, at doi:10.1016/j.compchemeng.2020.107098

References

Bathelt, A., Ricker, N.L., Jelali, M., 2015. Revision of the tennessee eastman process model. IFAC-PapersOnLine 48 (8), 309–314.
Bersimis, S., Sgora, A., Psarakis, S., 2017. Methods for interpreting the out-of-control signal of multivariate control charts: a comparison study. Qual. Reliab. Eng. Int. 33 (8), 2295–2326.
Blanke, M., Kinnaert, M., Lunze, J., Staroswiecki, M., 2016. Diagnosis and Fault-Tolerant Control. Springer, Berlin.
Bomze, I.M., Budinich, M., Pardalos, P.M., Pelillo, M., 1999. The maximum clique problem. In: Handbook of Combinatorial Optimization. Springer, Boston, pp. 1–74.
Chiang, L.H., Kotanchek, M.E., Kordon, A.K., 2004. Fault diagnosis based on fisher discriminant analysis and support vector machines. Comput. Chem. Eng. 28 (8), 1389–1401.
Chiang, L.H., Russell, E.L., Braatz, R.D., 2000. Fault diagnosis in chemical processes using fisher discriminant analysis, discriminant partial least squares, and principal component analysis. Chemom. Intell. Lab. Syst. 50 (2), 243–252.
Chiang, L.H., Russell, E.L., Braatz, R.D., 2001. Fault Detection and Diagnosis in Industrial Systems. Springer, London; New York.
Daoutidis, P., Allman, A., Khatib, S., Moharir, M.A., Palys, M.J., Pourkargar, D.B., Tang, W., 2019. Distributed decision making for intensified process systems. Curr. Opin. Chem. Eng. 25, 75–81.
Downs, J.J., Vogel, E.F., 1993. A plant-wide industrial process control problem. Comput. Chem. Eng. 17 (3), 245–255.
Ge, Z., Song, Z., Gao, F., 2013. Review of recent research on data-based process monitoring. Ind. Eng. Chem. Res. 52 (10), 3543–3562.
Georgakis, C., Steadman, B., Liotta, V., 1996. Decentralized PCA charts for performance assessment of plant-wide control structures. IFAC Proc. Vol. 29 (1), 5894–5898.
Ghosh, K., Ng, Y.S., Srinivasan, R., 2011. Evaluation of decision fusion strategies for effective collaboration among heterogeneous fault diagnostic methods. Comput. Chem. Eng. 35 (2), 342–355.
Grbovic, M., Dance, C.R., Vucetic, S., 2012. Sparse principal component analysis with constraints. In Proc. Twenty-Sixth AAAI Conf. Artific. Intell. 935–941.
He, X., Cai, D., Niyogi, P., 2006. Laplacian score for feature selection. In: Advances in neural information processing systems, pp. 507–514.
Heo, S., Lee, J.H., 2018. Fault detection and classification using artificial neural networks. IFAC-PapersOnLine 51 (18), 470–475.
Hopcroft, J., Tarjan, R., 1973. Algorithm 447: efficient algorithms for graph manipulation. Commun. ACM 16 (6), 372–378.
Huang, J., Yan, X., 2015. Double-step block division plant-wide fault detection and diagnosis based on variable distributions and relevant features. J. Chemom. 29 (11), 587–605.
Huang, J., Yan, X., 2016. Angle-based multiblock independent component analysis method with a new block dissimilarity statistic for non-gaussian process monitoring. Ind. Eng. Chem. Res. 55 (17), 4997–5005.
Jain, A.K., Murty, M.N., Flynn, P.J., 1999. Data clustering: a review. ACM Comput. Surv. (CSUR) 31 (3), 264–323.
Jiang, Q., Wang, B., Yan, X., 2015. Multiblock independent component analysis integrated with hellinger distance and bayesian inference for non-gaussian plant-wide process monitoring. Ind. Eng. Chem. Res. 54 (9), 2497–2508.
Johnson, D.S., 1974. Fast algorithms for bin packing. J. Comput. Syst. Sci. 8 (3), 272–314.
Jones-Farmer, L.A., Woodall, W.H., Steiner, S.H., Champ, C.W., 2014. An overview of phase i analysis for process improvement and monitoring. J. Qual. Technol. 46 (3), 265–280.
Jović, A., Brkić, K., Bogunović, N., 2015. A review of feature selection methods with applications. In: 2015 38th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO), pp. 1200–1205.
Khatib, S., 2020. System Decomposition for Distributed Data-Driven Process Monitoring. University of Minnesota Ph.D. thesis.
Khatib, S., Daoutidis, P., 2019. Generating optimal overlapping subsystems for distributed statistical fault detection subject to constraints. J. Process. Contr. 80, 143–151.
Khatib, S., Daoutidis, P., 2019. Optimal feature selection for distributed data-driven process monitoring. Ind. Eng. Chem. Res. 59 (6), 2307–2317.
Khatib, S., Daoutidis, P., 2020. Multiple hotelling's T^2 tests for distributed fault detection of large-scale systems. Comput. Chem. Eng. 136, 106807.
Khatib, S., Daoutidis, P., Almansoori, A., 2018. System decomposition for distributed multivariate statistical process monitoring by performance driven agglomerative clustering. Ind. Eng. Chem. Res. 57 (24), 8283–8298.
Leighton, F.T., 1979. A graph coloring algorithm for large scheduling problems. J. Res. Natl. Bur. Stand. 84 (6), 489–506.
Lewis, R., 2015. A Guide to Graph Colouring. Springer, Berlin.

Mason, R.L., Young, J.C., 2002. Multivariate statistical process control with industrial applications, 9. SIAM.
Mhaskar, P., Liu, J., Christofides, P.D., 2012. Fault-tolerant process control: Methods and applications. Springer-Verlag, London.
Qin, S.J., 2012. Survey on data-driven industrial process monitoring and diagnosis. Annu. Rev. Control. 36 (2), 220–234.
Qin, S.J., Valle, S., Piovoso, M.J., 2001. On unifying multiblock analysis with application to decentralized process monitoring. J. Chemom. 15 (9), 715–742.
Rong, M., Shi, H., Tan, S., 2019. Large-scale supervised process monitoring based on distributed modified principal component regression. Ind. Eng. Chem. Res. 58 (39), 18223–18240.
Venkatasubramanian, V., Rengaswamy, R., Kavuri, S.N., Yin, K., 2003. A review of process fault detection and diagnosis part III: process history based methods. Comput. Chem. Eng. 27 (3), 327–346.
Venkatasubramanian, V., Rengaswamy, R., Yin, K., Kavuri, S.N., 2003. A review of process fault detection and diagnosis part I: quantitative model-based methods. Comput. Chem. Eng. 27 (3), 293–311.
Wu, Q., Hao, J.-K., 2015. A review on algorithms for maximum clique problems. Eur. J. Oper. Res. 242 (3), 693–709.
Wu, Q., Hao, J.-K., Glover, F., 2012. Multi-neighborhood tabu search for the maximum weight clique problem. Ann. Oper. Res. 196 (1), 611–634.
Xu, C., Zhao, S., Liu, F., 2017. Distributed plant-wide process monitoring based on PCA with minimal redundancy maximal relevance. Chemom. Intell. Lab. Syst. 169, 53–63.
Zhu, J., Ge, Z., Song, Z., 2017. Distributed parallel PCA for modeling and monitoring of large-scale plant-wide processes with big data. IEEE Trans. Ind. Informat. 13 (4), 1877–1885.