Supporting information for: Application of Graph Theory and Filter Based Variable Selection Methods in the Design of a Distributed Data-Driven Monitoring System

Shaaz Khatib[a], Prodromos Daoutidis[a,*]

[a]*Department of Chemical Engineering and Materials Science, University of Minnesota, Minneapolis, MN 55455, USA*

1. Nomenclature

Table 1: Nomenclature for commonly used notations

Notation	Description
B	Number of subsystems
PDAC	Performance Driven Agglomerative Clustering
FSDPR	Forward Selection for Distributed Pattern Recognition
EFSDPR	Extended Forward Selection for Distributed Pattern Recognition
SASA	Subsystem and Sensor Allocation
ESASA	Extended Subsystem and Sensor Allocation
m	Number of sensors in the system
MCR	Misclassification Rate
V	Set of selected sensors for distributed pattern recognition
W	Vector containing the sensor weights in the EFSDPR method
$[1...i]$	Set of integers from 1 to i
A	Adjacency matrix encoding the cannot-link constraints
N_{lim}	Upper limit on the number of sensors allocated to a subsystem
G_S	Subgraph of the graph of cannot-link constraints induced by the set V
G'_S	Complement graph of G_S
MCCMC	Minimum Clique Covering with Maximum Cliques
S_j	Set of sensors in the j^{th} subsystem
Ω_j	Set of sensors not in S_j that have no cannot-link constraints with any of the sensors in S_j
RLF	Recursive Largest First
N_M	Number of monitoring locations
M_k^{sensor}	Set of sensors allocated to the k^{th} monitoring location
S_k^{store}	Sensors in the k^{th} subsystem of the decomposition selected for distributed data storage
GEA	Greedy Equal Allocation

*Corresponding author

Email addresses: khati013@umn.edu (Shaaz Khatib), daout001@umn.edu (Prodromos Daoutidis)

Table 2: Nomenclature for commonly used notations (cont.)

Notation	Description
J_{rem}	Set of subsystems that have been assigned to multiple monitoring locations in an iteration of the GEA algorithm
TEP	Tennessee Eastman Process
S_j^{diag}	Subsystem in the decomposition selected for distributed fault diagnosis
S_j^{det}	Subsystem in the decomposition selected for distributed fault detection
$\|S\|$	Number of elements in the set S
$\emptyset$	Empty set
$\setminus$	Set difference
D	Set representing a decomposition whose j^{th} element $D(j)$ is itself a set containing the sensors allocated to the j^{th} subsystem
n_f	Number of known faults
θ_k	Fault direction
μ	Mean vector
R	Covariance matrix
LFD	Local Fault Decision
GFD	Global Fault Decision

2. Distributed Pattern Recognition Method Used in the Case Study Involving the System with 1000 Sensors

Let us assume that the objective of the distributed pattern recognition method is to find which one of a set of known faults is most likely to have affected the system. Each of these fault is assumed to have a variable magnitude. Let n_f be the number of known faults with each fault being indexed by a number from 1 to n_f. The diagnosis that process operation is normal is given the index $n_f + 1$. Each of the known faults is characterized by a fault direction. Let the $m \times 1$ unit vector θ_k be the direction of the k^{th} fault (where $k \in [1...n_f]$) in the standardized measurement space whose i^{th} coordinate represents the standardized measurements from the i^{th} sensor. Let us assume that the decomposition selected to implement the distributed pattern recognition method has B subsystems with S_j representing the set containing the indices of the sensors in the j^{th} subsystem (where $j \in [1...B]$). Let m_j be the number of sensors in the j^{th} subsystem with $S_j(i)$ being i^{th} element of the set S_j. A new sample is generated after a fault is detected in the system. Each measurement in this sample is standardized using the normal mean value and standard deviation of the sensor. Let x be a $m \times 1$ column vector with $x(i)$ being a standardized measurement of the sample from the i^{th} sensor. The distributed pattern recognition method uses the sample x to find which one of the n_f faults is most likely. Let R be the correlation matrix of the sensors when process operation is normal.

2.1. Pattern Recognition Method Implemented in Each Subsystem

The pattern recognition method implemented in each subsystem comes to the diagnosis that a specific fault is most likely if the direction of the fault is most collinear with the new sample out of all the known faults. However, if the measurements of the sensors of a subsystem do not have a significant deviation from normal operation, then the local diagnosis of the subsystem will be that process operation is normal. Let the $m_j \times 1$ vector x_j contain measurements from sensors of the j^{th} subsystem where:

$$x_j(i) = x(S_j(i)) \qquad \forall i \in [1...m_j], \forall j \in [1...B] \tag{1}$$

Let the $m_j \times m_j$ matrix R_j represent the correlation matrix of the sensors of the j^{th} subsystem where:

$$R_j(i_1, i_2) = R(S_j(i_1), S_j(i_2)) \qquad \forall i_1, i_2 \in [1...m_j], \forall j \in [1...B] \tag{2}$$

Let θ_{jk} be a $m_j \times 1$ vector where:

$$\theta_{jk}(i) = \theta_k(S_j(i)) \qquad \forall i \in [1...m_j], \forall j \in [1...B] \tag{3}$$

The j^{th} subsystem comes to the diagnosis that the condition with index LFD_j is most likely based on the sample x:

$$LFD_j = \begin{cases} n_f + 1, & \text{if } x_j^T R_j^{-1} x_j \leq \chi^2_\alpha(m_j) \\ \underset{k \in K_j}{\arg\max} \ \frac{\theta_{jk}^T x_j}{\|\theta_{jk}\|_2}, & \text{otherwise} \end{cases} \tag{4}$$

where $K_j = \{k \mid \|\theta_{jk}\|_2 > 0, k \in [1...n_f]\}$, $\|\theta_{jk}\|_2$ is the euclidean norm of θ_{jk}, α is the user input level of significance for the T^2 test which determines if the subsystem comes to the diagnosis that process operation is normal and $\chi^2_\alpha(m_j)$ is the inverse of the χ^2 cumulative distribution function for a probability value of $1 - \alpha$ when the number of degrees of freedom is m_j.

Each subsystem assigns a value (called the weighted vote) to a possible diagnosis that quantifies the weight it gives to that diagnosis after it has reached a given local diagnosis. The weighted

votes are stored in a $n_f \times (n_f + 1)$ matrix WV_j for each subsystem $j \in [1...B]$. The element $WV_j(k, l)$ is the weighted vote given by subsystem j to the fault diagnosis k if it has reached the local diagnosis l. The matrix WV_j is given by:

$$WV_j(k,l) = \begin{cases} \frac{|\{i|\theta_{jl}(i)>0\}|}{|\{i|\theta_l(i)>0\}|}, & \text{if } l \neq n_f + 1 \ \& \ \|\theta_{jl}\|_2 > 0 \ \& \ \|\theta_{jk}\|_2 > 0 \ \& \ \frac{\theta_{jk}^T \theta_{jl}}{\|\theta_{jk}\|_2 \|\theta_{jl}\|_2} = 1 \\ 0, & \text{otherwise} \end{cases} \tag{5}$$

where:

$$\text{Number of sensors in subsystem } j \text{ affected by the fault } l = |\{i \mid \theta_{jl}(i) > 0\}| \tag{6}$$

$$\text{Total number of sensors affected by the fault } l = |\{i \mid \theta_l(i) > 0\}| \tag{7}$$

2.2. Combining the Local Diagnoses of All the Subsystems

The distributed pattern recognition method uses a weighted voting strategy that uses the local diagnoses of all the subsystems (LFD_j) along with the matrices WV_j to reach a final diagnosis. The j^{th} subsystem will give a weighted vote equal to $WV_j(k, LFD_j)$ for the diagnosis that fault k has occurred. The diagnosis reached by the distributed pattern recognition method is the one with the maximum sum of weight of votes from all the subsystems. The diagnosis that fault GFD has occurred is reached by the distributed pattern recognition method if:

$$GFD = \begin{cases} n_f + 1, & \text{if } LFD_j = n_f + 1 \ \ \forall j \in [1...B] \\ \arg\max\limits_k \ \sum_{j=1}^{B} WV_j(k, LFD_j), & \text{otherwise} \end{cases} \tag{8}$$

Remark 1: The statistical test employed in equation 4 is useful in the distributed pattern recognition method for diagnosing faults with greater accuracy if most of the faults affect only a small number of subsystems thereby leaving a large number of subsystems unaffected by the fault. A subsystem will give a weighted vote of zero if it reaches the diagnosis that process operation is normal after a fault has been detected by the monitoring system. Therefore, the statistical test can help prevent a subsystem whose sensors are unaffected by a fault from giving a vote to an incorrect diagnosis. The limitation of the statistical test is that it can prevent a subsystem from reaching a correct diagnosis if the fault has a small effect on the system's sensors. Therefore, the value of α should be selected based on the performance of the distributed pattern recognition method. If most of the subsystems are affected by most of the faults, then employing the statistical test may not be useful for fault diagnosis. The value of α should tend close to one in such a case so that the statistical test plays no role in the diagnosis. In the case study of the system with 10000 sensors, α is set equal to a value for which the diagnostic performance of the distributed pattern recognition method is close to optimal.

Remark 2: It is possible that the fault affecting the system may not be one of the n_f known faults. The distributed pattern recognition method only gives a likely diagnosis out of the previously known faults and the practitioner should always consider the possibility that an unknown fault could have occurred. The unknown fault scenario can also be incorporated into the distributed pattern recognition method. Let us assume that the unknown fault scenario is given the index $n_f + 2$. A subsystem is first allowed to reach its diagnosis using equation 4. Let us assume that subsystem j reaches the diagnosis that fault k has most likely occurred. If $k \neq n_f + 1$ and if the cosine similarity between θ_{jk} and x_j is below a user defined upper bound, then subsystem

j reaches the index $n_f + 2$ diagnosis that an unknown fault has occurred. An additional row and column also must be added to the end of the matrices WV_j with $WV_j(n_f + 2, n_f + 2) = 1$ and $WV_j(k, n_f + 2) = WV_j(n_f + 2, k) = 0 \ \forall k \in [1...(n_f + 1)]$. The consensus strategy can then be applied as described above to reach a final diagnosis. The limitation of adding the unknown fault scenario into the pattern recognition method is that it could degrade its performance in reaching the correct diagnosis when known faults occur. This is because it is more difficult to diagnose the correct scenario if larger number of possible scenarios are assumed. The unknown fault scenario is not incorporated into the distributed pattern recognition method in the case studies.

Remark 3: The unknown fault and normal process operation scenarios should not be incorporated into the pattern recognition method when it is simulated in the sensor selection stage of the EFSDPR method. This is because the pattern recognition method is simulated using data generated due to the previously known faults in the EFSDPR method and therefore the unknown fault and normal process operation scenarios are impossible in the simulations.

3. Pseudocode of the Approximate Greedy Algorithm

The pseudocode of the approximate greedy algorithm for solving the MCCMC problem is given in Algorithms 1, 2 and 3.

Algorithm 1 Approximate greedy algorithm

procedure ApproxGreedy(A, N_{lim}, W)
  $D_{s1} = RLF(A, N_{lim}, W)$ ▷ The pseudocode of the RLF algorithm is given in Algorithm 2
  ▷ The non-overlapping decomposition output by the RLF algorithm is represented by the set D_{s1} where the j^{th} element of D_{s1} (i.e. $D_{s1}(j)$) is itself a set containing the sensors in the j^{th} subsystem of the decomposition.
  $B = |D_{s1}|$ ▷ Number of subsystems
  $S_1 = D_{s1}(1)$ ▷ S_j is the set of sensors in the j^{th} subsystem of the selected decomposition.
  for $j = 2$ to B **do**
    $S_j = MAXCLIQUE(D_{s1}(j), A, N_{lim}, W)$ ▷ The pseudocode of the $MAXCLIQUE$ function is given in Algorithm 3
  end for
  return $D = \{S_1, S_2, \cdots, S_B\}$
end procedure

Algorithm 2 RLF algorithm

1: **procedure** RLF(A, N_{lim}, W)
2: m_A = No. of rows in A
3: $X = [1...m_A]$
4: $B = 0$ ▷ Number of Subsystems
5: **while** $X \neq \emptyset$ **do**
6: $B = B + 1$
7: $i_{add} = \arg\max_i \{W(i) \mid i \in X, \sum_{i_1 \in X}(A(i, i_1)) = \min_{i_2 \in X} \sum_{i_1 \in X}(A(i_2, i_1))\}$
8: $S_B = \{i_{add}\}$ ▷ In lines 7 and 8, the sensor in X which has the largest number of cannot-link constraints with other sensors in X is identified and is added to the new subsystem S_B. If multiple sensors meet this criterion, then the sensor with the largest weight is given preference.
9: **if** $|S_B| < N_{lim}$ **then**
10: $X = X \setminus \{i_{add}\}$
11: $Y = \{i \mid i \in X, A(i, i_{add}) = 0\}$ ▷ Y is the set of sensors that cannot be added to the subsystem S_B and have to be added to a subsystem S_j $j > B$ later in the algorithm.
12: $X = X \setminus Y$ ▷ X contains the sensors that can be added to S_B
13: **else**
14: $Y = X$
15: $X = \emptyset$
16: **end if**
17: **while** $X \neq \emptyset$ **do**
18: $i_{add} = \arg\max_i \{W(i) \mid i \in X, \sum_{i_1 \in Y}(A(i, i_1)) = \min_{i_2 \in X} \sum_{i_1 \in Y}(A(i_2, i_1))\}$
19: $S_B = S_B \cup \{i_{add}\}$
20: **if** $|S_B| < N_{lim}$ **then**
21: $X = X \setminus \{i_{add}\}$
22: $Y = Y \cup \{i \mid i \in X, A(i, i_{add}) = 0\}$
23: $X = X \setminus Y$
24: **else**
25: $Y = Y \cup X$
26: $X = \emptyset$
27: **end if**
28: **end while**
29: $X = Y$
30: **end while**
31: **return** $D = \{S_1, S_2, \cdots, S_B\}$
32: **end procedure**

Algorithm 3 Maximum clique algorithm

1: **procedure** MaxClique(S_{input}, A, N_{lim}, W)

2: ▷ S_{input} is a set containing the sensors that have already been permanently added to the subsystem to which the MaxClique algorithm is applied

3: m_A = No. of rows in A

4: $S = S_{input}$

5: **if** $|S| < N_{lim}$ **then**

6: $X = \{i \mid i \in [1...m_A] \setminus S, \sum_{i_1 \in S}(A(i, i_1)) = |S|\}$

7: **while** $X \neq \emptyset$ **do**

8: $ObjF_{max} = \max_{i_2 \in X} \sum_{i_1 \in X}(A(i_2, i_1)W(i_1))$

9: $i_{add} = \arg\max_{i}\{W(i) \mid i \in X, \sum_{i_1 \in X}(A(i, i_1)W(i_1)) = ObjF_{max}\}$

10: $S = S \cup \{i_{add}\}$ ▷ In lines 8-10, the sensor in X added to S is the one for which the sum of the total weight of its neighbors in X and its own weight is maximum. If multiple sensors meet this criterion, then the sensor with the largest weight is added to S.

11: **if** $|S| < N_{lim}$ **then**

12: $X = X \setminus \{i_{add}\}$

13: $X_{old} = X$

14: $X = \{i \mid i \in X_{old}, A(i, i_{add}) = 1\}$

15: **else**

16: $X = \emptyset$

17: **end if**

18: **end while**

19: **end if**

20: **return** S

21: **end procedure**

4. Tabu Search Based Algorithm

The algorithm based on the tabu search heuristic can be used to find a better solution to the MCCMC problem than the approximate greedy algorithm if the approximate greedy algorithm doesn't find the optimal solution. The tabu search based algorithm uses slightly modified versions of the TABUCOL algorithm [1] and the Multi Neighborhood Tabu Search (MN/TS) algorithm [2] to find close to optimal solutions to the first and second subproblems of the MCCMC problem respectively. The algorithm starts by first partitioning the graph G_S into its connected components. If the graph G_S can be partitioned into multiple connected components, then solving the MCCMC problem for the entire graph G_S is decomposed into solving the MCCMC problem for each of the connected components. This is because there is a cannot-link constraint between every pair of sensors from different connected components and therefore a feasible subsystem can only contain sensors from one of the connected components. The depth first search algorithm [3] can be used to identify the connected components of a graph.

Let us now consider that the tabu search based algorithm has to find a near optimal solution to the MCCMC problem for a connected component. The TABUCOL algorithm starts by applying the RLF algorithm to generate a feasible non-overlapping decomposition of the selected sensors in the connected component. The TABUCOL algorithm then merges two subsystems of the feasible decomposition to generate an infeasible decomposition with smaller number of subsystems. The TABUCOL algorithm then applies a fine tuning procedure to this infeasible decomposition in which sensors are shifted from one subsystem of the decomposition to a different subsystem with the aim of reducing the magnitude of constraint violation. After a sensor is shifted to a different subsystem, then it can no longer be shifted back to the same subsystem for a certain number of iterations unless shifting the sensor generates a decomposition with lower constraint violation. The fine tuning procedure ends when it generates a feasible decomposition or if the magnitude of constraint violation cannot be reduced for a certain number of iterations. Two subsystems of this feasible decomposition are then again merged to generate an infeasible decomposition and the same process is repeated until a feasible decomposition cannot be generated using the fine tuning procedure. The feasible decomposition generated in TABUCOL with the smallest number of subsystems is then output by TABUCOL.

The additional set of selected sensors that are to be added to each subsystem of the decomposition output by TABUCOL are determined using the MN/TS algorithm. Let us consider that MN/TS is applied to one of the subsystems output by TABUCOL. The MN/TS algorithm first uses a procedure to randomly add sensors to the subsystem without violating the imposed constraints. Three operations are then used in each iteration of the algorithm to modify the subsystem without violating the constraints. The first operation involves the addition of a sensor to the subsystem, the second operation involves the removal of a sensor from the subsystem and the third operation involves the swapping of a sensor from the subsystem with a different sensor. The modification of the subsystem that would result in the greatest increase or the least decrease in the sum of its sensor's weights without constraint violation is then carried out at the end of an iteration. If a senor is removed or swapped out of the subsystem in an iteration, then the addition of the sensor back to the subsystem will be prohibited for a certain number of iterations. Also, the sensors that are allocated to the subsystem by TABUCOL cannot be removed throughout the algorithm. If the sum of sensor weights in the subsystem cannot be improved for a certain number of iterations, then the algorithm restarts by again randomly adding sensors to the initial set of sensors in the subsystem. MN/TS thus generates a number of possible sensor sets that can be added to the subsystem without

constraint violation and the sensor set with the largest sum of weights is then permanently added to the subsystem. The resulting subsystem is then one of the subsystem of the selected decomposition.

The pseudocode of the tabu search based algorithm is given in Algorithms 4-11.

Algorithm 4 Tabu search based algorithm

1: **procedure** TABUSEARCH($A, N_{lim}, W, \rho_1, \rho_2, \rho_3$)

2: ▷ The inputs ρ_1, ρ_2 and ρ_3 are parameters that define the stopping criteria of the TABUCOL and MN/TS algorithms

3: Find the connected components in the undirected graph defined by the adjacency matrix A. Let N_{cc} be the number of connected components and let V_k ($k \in [1...N_{cc}]$) be the set of sensors in the k^{th} connected component.

4: **for** $k = 1$ to N_{cc} **do**

5: Define a $|V_k| \times |V_k|$ matrix A_k with $A_k(i_1, i_2) = A(V_k(i_1), V_k(i_2))\ \forall i_1, i_2 \in [1...|V_k|]$

6: Define a $|V_k| \times 1$ vector W_k with $W_k(i_1) = W(V_k(i_1))\ \forall i_1 \in [1...|V_k|]$

7: $D_F = RLF(A_k, N_{lim}, W_k)$ ▷ The pseudocode of the RLF algorithm is given in Algorithm 2. D_F is a set representing a feasible decomposition with $D_F(j)$ being the set of sensors in its j^{th} subsystem.

8: $D_{s1} = D_F$ ▷ The set D_{s1} represents the non-overlapping decomposition output after solving the first subproblem.

9: $B_k = |D_F|$ ▷ Number of subsystems in the decomposition of sensors of the k^{th} connected component.

10: **while** $D_F \neq \emptyset$ AND $B_k > 2$ **do**

11: **for** $j_1 = 1$ to $(B_k - 1)$ **do**

12: **for** $j_2 = (j_1 + 1)$ to B_k **do**

13: $N_{12} = |D_F(j_1)| + |D_F(j_2)|$

14: $MCV_{j_1 j_2} = 0.5(N_{12}^2 - \sum_{i_1, i_2 \in D_F(j_1) \cup D_F(j_2)}(A_k(i_1, i_2))) + \max\{\frac{N_{12}}{N_{lim}} - 1, 0\}$

15: **end for**

16: **end for**

17: $(j_1^{opt}, j_2^{opt}) = \arg\min_{j_1, j_2} \{MCV_{j_1 j_2} \mid j_1 \in [1...(B_k - 1)], j_2 \in [(j_1 + 1)...B_k]\}$

18: $D_I = D_F \setminus \{D_F(j_1^{opt}), D_F(j_2^{opt})\}$

19: $D_I = D_I \cup \{D_F(j_1^{opt}) \cup D_F(j_2^{opt})\}$ ▷ In lines 11-19, two subsystems of D_F are merged to form an infeasible decomposition D_I such that the magnitude of constraint violation of the infeasible decomposition is as small as possible.

20: ▷ Algorithm continues on next page

Algorithm 5 Tabu search based algorithm (cont.)

$D_F = TABUCOL(D_I, A_k, N_{lim}, \rho_1)$ ▷ The function $TABUCOL$ implements the fine tuning procedure of the TABUCOL algorithm and its pseudocode is given in Algorithms 6-8. If D_F is an empty set, then a feasible decomposition cannot be generated by modifying the infeasible decomposition D_I in the fine tuning procedure.

if $D_F \neq \emptyset$ **then**

$D_{s1} = D_F$

$B_k = B_k - 1$

end if

end while

for $j = 1$ to B_k **do**

$\Omega_j = \{i \mid i \in [1...|V_k|] \setminus D_{s1}(j), \sum_{i_1 \in D_{s1}(j)} A_k(i, i_1) = |D_{s1}(j)|\}$

if $\Omega_j \neq \emptyset$ **then**

Define a $|\Omega_j| \times |\Omega_j|$ matrix A_Ω with $A_\Omega(i_1, i_2) = A_k(\Omega_j(i_1), \Omega_j(i_2))$ $\forall i_1, i_2 \in [1...|\Omega_j|]$

Define a $|\Omega_j| \times 1$ vector W_Ω with $W_\Omega(i_1) = W_k(\Omega_j(i_1))$ $\forall i_1 \in [1...|\Omega_j|]$

$S_{mnts} = MN/TS(A_\Omega, N_{lim} - |D_{s1}(j)|, W_\Omega, \rho_2, \rho_3)$ ▷ The pseudocode of the MN/TS function is given in Algorithms 9-11

Define a set S with $|S| = |S_{mnts}|$ and $S(i_1) = \Omega_j(S_{mnts}(i_1))$ $\forall i_1 \in [1...|S_{mnts}|]$

else

$S = \emptyset$

end if

$S = S \cup D_{s1}(j)$

Define a set S_{jk} with number of elements equal to $|S|$ where $S_{jk}(i_1) = V_k(S(i_1))$ $\forall i_1 \in [1...|S|]$

end for

end for

return $D = \{S_{11}, S_{21}, \cdots, S_{B_1 1}, S_{12}, S_{22}, \cdots, S_{B_2 2}, \cdots, S_{1N_{cc}}, S_{2N_{cc}}, \cdots, S_{B_{N_{cc}} N_{cc}}\}$

end procedure

Algorithm 6 TABUCOL algorithm

procedure TABUCOL(D_I, A, N_{lim}, ρ_1)

▷ ρ_1 is the number of iterations for which no reduction in the magnitude of constraint violation will be tolerated in the fine tuning procedure before it is brought to an end

m_A = No. of rows in A

$A_C = 1_{m_A \times m_A} - A$ ▷ $1_{m_A \times m_A}$ is a $m_A \times m_A$ matrix of ones.

$B = |D_I|$ ▷ Number of subsystems

$D_F = \emptyset$ ▷ The set D_F represents the feasible decomposition output by this function. Its j^{th} element is the set of sensors in its j^{th} subsystem. If D_F is an empty set, then this means that no feasible decomposition can be generated by the function.

for $i = 1$ to m_A **do**

$NBRS_i = \{i_1 \mid i_1 \in [1...m_A], A_C(i, i_1) = 1\}$ ▷ $NBRS_i$ is the set of neighbors of sensor i in the complement graph

end for

Define a $m_A \times 1$ vector SUB whose i^{th} element is the index of the subsystem to which sensor i belongs in D_I i.e. $SUB(i) = j$ if $i \in D_I(j)$

$N_1^{it} = 1$ ▷ Iteration number

$N_2^{it} = 0$ ▷ Number of iterations for which there has been no reduction in the magnitude of constraint violation

Define a $m_A \times B$ matrix T where $T(i, j)$ is the iteration number until which sensor i cannot be moved to subsystem j in the algorithm. T is initially a matrix of zeros.

Define a $m_A \times B$ matrix C where $C(i, j)$ is the number of cannot-link constraints between sensor i and the sensors of the j^{th} subsystem of D_I i.e. $C(i, j) = \sum_{i_1 \in D_I(j)} A_C(i, i_1)$ $\forall i \in [1...m_A]$, $\forall j \in [1...B]$.

$MCV = 0$ ▷ Magnitude of constraint violation of D_I

for $j = 1$ to B **do**

$\Delta N(j) = \max\{\frac{|D_I(j)|}{N_{lim}} - 1, 0\}$

$MCV = MCV + \Delta N(j) + 0.5 \sum_{i_1 \in D_I(j)} \sum_{i_2 \in D_I(j)} A_C(i_1, i_2)$

end for

$MCV_{min} = MCV$ ▷ Algorithm continues on next page

Algorithm 7 TABUCOL algorithm (cont.)

21: **while** $N_2^{it} < \rho_1$ **do**
22: $MCV_{min}^{cand} = \infty$
23: **for** $i = 1$ to m_A **do**
24: $s = SUB(i)$
25: **if** $C(i,s) \neq 0$ or $\Delta N(s) > 0$ **then** ▷ A sensor will be shifted out of its subsystem only if it is responsible for a constraint violation in its original subsystem
26: **for** $j \in [1...B] \setminus s$ **do**
27: $MCV^{cand} = MCV + C(i,j) - C(i,s) + \max\{\frac{|D_I(j)|+1}{N_{lim}} - 1, 0\} + \max\{\frac{|D_I(s)|-1}{N_{lim}} - 1, 0\} - \Delta N(s) - \Delta N(j)$ ▷ MCV^{cand} is the magnitude of constraint violation of the decomposition generated when sensor i is shifted from subsystem s of D_I to subsystem j
28: **if** $MCV^{cand} = 0$ **then**
29: $MCV_{min}^{cand} = 0$
30: $i_{opt} = i$
31: $j_{opt} = j$
32: **break**
33: **end if**
34: **if** $MCV^{cand} < MCV_{min}^{cand}$ AND ($MCV^{cand} < MCV_{min}$ OR $T(i,j) < N_1^{it}$) **then**
35: $MCV_{min}^{cand} = MCV^{cand}$
36: $i_{opt} = i$
37: $j_{opt} = j$
38: **end if**
39: **end for**
40: **end if**
41: **if** $MCV_{min}^{cand} = 0$ **then**
42: **break**
43: **end if**
44: **end for**
45: **if** $MCV_{min}^{cand} = \infty$ **then**
46: **return** $\emptyset$
47: **end if**
48: $D_I(j_{opt}) = D_I(j_{opt}) \cup \{i_{opt}\}$
49: $D_I(SUB(i_{opt})) = D_I(SUB(i_{opt})) \setminus \{i_{opt}\}$
50: $MCV = MCV_{min}^{cand}$ ▷ Algorithm continues on next page

Algorithm 8 TABUCOL algorithm (cont.)

if $MCV = 0$ **then**
  $D_F = D_I$
  return D_F
end if
if $MCV < MCV_{min}$ **then**
  $MCV_{min} = MCV$
  $N_2^{it} = 0$
else
  $N_2^{it} = N_2^{it} + 1$
end if
for $i \in NBRS_{i_{opt}}$ **do**
  $C(i, SUB(i_{opt})) = C(i, SUB(i_{opt})) - 1$
  $C(i, j_{opt}) = C(i, j_{opt}) + 1$
end for
$\Delta N(j_{opt}) = \max\{\frac{|D_I(j_{opt})|}{N_{lim}} - 1, 0\}$
$\Delta N(SUB(i_{opt})) = \max\{\frac{|D_I(SUB(i_{opt}))|}{N_{lim}} - 1, 0\}$
Generate a random integer r from 1 to 10
$T(i_{opt}, SUB(i_{opt})) = N_1^{it} + round(0.6MCV - 1 + r)$
$SUB(i_{opt}) = j_{opt}$
$N_1^{it} = N_1^{it} + 1$
end while
end procedure

Algorithm 9 MN/TS algorithm

procedure MN/TS($A, N_{lim}, W, \rho_2, \rho_3$)
  if $N_{lim} = 0$ **then**
    return $\emptyset$
  end if
  $m_A =$ No. of rows in A
  $ObjF_{max} = 0$
  $N_1^{it} = 0$
  while $N_1^{it} < \rho_2$ **do**
    $ObjF_{max}^{old} = ObjF_{max}$
    $S = \emptyset$ ▷ Set of sensors in the clique S
    $ADD = [1...m_A]$
    while $ADD \neq \emptyset$ **do**
      Generate a random integer r from 1 to $|ADD|$
      $S = S \cup \{ADD(r)\}$
      $ADD = ADD \setminus \{ADD(r)\}$
      if $|S| = N_{lim}$ **then**
        $ADD = \emptyset$
      else
        $ADD = \{i \mid i \in ADD, A(i, ADD(r)) = 1\}$
      end if
    end while
    if $\sum_{i \in S} W(i) > ObjF_{max}$ **then**
      $S_{opt} = S$ ▷ Set of sensors in the optimal clique S_{opt}
      $ObjF_{max} = \sum_{i \in S} W(i)$
    end if
    $S' = [1...m_A] \setminus S$
    Define a $m_A \times 1$ vector T whose i^{th} element is the iteration number until which the i^{th} sensor cannot be added to the clique S. T is a vector of zeros at this step
    $SWAP = \{(i_1, i_2) \mid i_1 \in S', \sum_{i_3 \in S} A(i_1, i_3) = |S| - 1, i_2 \in S, A(i_1, i_2) = 0\}$
    $N_3^{it} = N_2^{it} = 0$ ▷ Algorithm continues on next page

Algorithm 10 MN/TS algorithm (cont.)

30: **while** $N_2^{it} < \rho_3$ **do**
31: $N_3^{it} = N_3^{it} + 1$
32: **if** $SWAP \neq \emptyset$ **then**
33: $(i_1^{swap}, i_2^{swap}) = \arg\max_{(i_1,i_2)} \{W(i_1) - W(i_2) \mid (i_1, i_2) \in SWAP\}$
34: $\Delta W_{swap} = W(i_1^{swap}) - W(i_2^{swap})$
35: **else**
36: $\Delta W_{swap} = -\infty$
37: **end if**
38: **if** $ADD \neq \emptyset$ **then**
39: $i^{add} = \arg\max_{i} \{W(i) \mid i \in ADD\}$
40: **if** $W(i^{add}) > \Delta W_{swap}$ **then**
41: $S = S \cup \{i^{add}\}$
42: $S' = S' \setminus \{i^{add}\}$
43: **else**
44: $S = S \cup \{i_1^{swap}\}$
45: $S = S \setminus \{i_2^{swap}\}$
46: $S' = S' \cup \{i_2^{swap}\}$
47: $S' = S' \setminus \{i_1^{swap}\}$
48: $T(i_2^{swap}) = q + 7 + N_3^{it}$ where q is a random integer from 1 to $|SWAP|$
49: **end if**
50: **else**
51: $i^{rem} = \arg\max_{i} \{-W(i) \mid i \in S\}$
52: **if** $\Delta W_{swap} > -W(i^{rem})$ **then**
53: Repeat lines 44-48
54: **else**
55: $S = S \setminus \{i^{rem}\}$
56: $S' = S' \cup \{i^{rem}\}$
57: $T(i^{rem}) = 7 + N_3^{it}$
58: **end if**
59: **end if**

▷ Algorithm continues on next page

Algorithm 11 MN/TS algorithm (cont.)

60: **if** $\sum_{i \in S} W(i) > ObjF_{max}$ **then**
61: $S_{opt} = S$ ▷ Set of sensors in the optimal clique S_{opt}
62: $ObjF_{max} = \sum_{i \in S} W(i)$
63: $N_2^{it} = 0$
64: **else**
65: $N_2^{it} = N_2^{it} + 1$
66: **end if**
67: **if** $|S| < N_{lim}$ **then**
68: $ADD = \{i \mid i \in S', T(i) < N_3^{it}, \sum_{i_1 \in S} A(i, i_1) = |S|\}$
69: **else**
70: $ADD = \emptyset$
71: **end if**
72: **if** $S = \emptyset$ and $ADD = \emptyset$ **then**
73: **break**
74: **end if**
75: $SWAP = \{(i_1, i_2) | i_1 \in S', T(i_1) < N_3^{it}, \sum_{i_3 \in S} A(i_1, i_3) = |S| - 1, i_2 \in S, A(i_1, i_2) = 0\}$
76: **end while**
77: **if** $ObjF_{max} > ObjF_{max}^{old}$ **then**
78: $N_1^{it} = 0$
79: **else**
80: $N_1^{it} = N_1^{it} + 1$
81: **end if**
82: **end while**
83: **end procedure**

Remark 4: The constraint that places an upper limit on the number of sensors in each subsystem is a specific instance of a class of constraints that limit the sum of some property of each sensor in a subsystem to be less than equal to a user defined threshold. This class of constraints is incorporated into the MCCMC algorithms in the software of the EFSDPR and ESASA methods.

Remark 5: If the user wishes to impose a must-link constraint between two sensors in order to force the two sensors to be allocated to the same subsystem, then these two sensors should be considered as a single variable when a MCCMC algorithm is applied for sensor allocation.

Remark 6: The input parameters ρ_1, ρ_2 and ρ_3 define the stopping criteria at different stages of the tabu search based algorithm. The parameter ρ_1 is the number of iterations for which no reduction in the magnitude of constraint violation will be tolerated in the fine tuning procedure of TABUCOL before it is brought to an end. The parameter ρ_2 is the number of passes for which no increase in the objective function will be tolerated in the MN/TS algorithm before it is brought to an end. The parameter ρ_3 is the number of iterations for which no increase in the objective

function will be tolerated in a pass of the MN/TS algorithm before the pass is brought to an end and a new clique is generated randomly for the next pass of the algorithm. If larger values of ρ_1, ρ_2 and ρ_3 are input, then the probability that the TABUCOL and MN/TS algorithms will reach a suboptimal solution will be smaller, but their computation times will be larger. Let m_A be the number of variables in the graph to which either the TABUCOL or the MN/TS algorithm is applied. The default values of ρ_1, ρ_2 and ρ_3 are $10m_A$, 5 and $\max\{m_A, 100\}$ respectively in the EFSDPR/ESASA software.

5. Pseudocode of the EFSDPR Method

The pseudocode of the EFSDPR method is given in Algorithms 12 and 13.

Algorithm 12 EFSDPR

procedure EFSDPR(Inputs)

The inputs of EFSDPR include the adjacency matrix A encoding the cannot-link constraints and the upper limit N_{lim} on the number of sensors in each subsystem. The input X_F is a faulty dataset. The input Y_F contains the fault that each sample in X_F is generated due to. $PRALG$ is a user input function that simulates the pattern recognition method using different subsets of the system's sensors and the input $PRInputs$ includes any additional inputs needed to run $PRALG$ besides the set of sensors, X_F and Y_F. The input function $FILTER$ implements the filter method and the input $FilterInputs$ contains its inputs. The input $MCCMCALG$ is the algorithm used to solve the MCCMC problem and $MCCMCInputs$ contains any of its additional inputs besides A,W and N_{lim}. The input W is the vector of sensor weights. If it is unspecified, then the value of W is calculated using the $FILTER$ function. The input NP_{max} is the maximum number of passes of the first stage of EFSDPR.

$m =$ No. of rows of A ▷ Stage 1 of EFSDPR starts here

if $FILTER$ is a univariate filter method **then**

$RANK = FILTER(FilterInputs)$ ▷ $RANK$ is a $m \times 1$ vector whose i^{th} element is the rank given to sensor i by the filter method

end if

If W is not input by the user, then it is calculated using the input filter method.

$V = \emptyset$ ▷ Set of selected sensors

$MCR_V = \infty$

$NP = 0$

while $NP < NP_{max}$ **do**

$ADD = [1...m] \setminus V$

if $FILTER$ is a multivariate filter method **then**

Use $FILTER$ to rank each sensor $i \in ADD$ based on the value of the multivariate filter metric for the set $V \cup i$. Store the sensor ranks in the variable $RANK$. If a sensor i is not in ADD, then the value of $RANK(i)$ is set equal to an arbitrary value.

end if

$V_{old} = V$ ▷ Algorithm continues on next page

Algorithm 13 EFSDPR (cont.)

17: **while** $ADD \neq \emptyset$ **do**
18: $i_{add} = \arg\min_i \{RANK(i) \mid i \in ADD\}$
19: $ADD = ADD \setminus \{i_{add}\}$
20: $MCR = PRALG(V \cup \{i_{add}\}, X_F, Y_F, PRInputs)$
21: **if** $MCR < MCR_V$ **then**
22: $V = V \cup \{i_{add}\}$
23: $MCR_V = MCR$
24: Repeat steps on lines 13-15
25: **end if**
26: **end while**
27: **if** $V = V_{old}$ **then**
28: **break**
29: **end if**
30: $NP = NP + 1$
31: **end while**
32: $MCR_{all} = PRALG([1...m], X_F, Y_F, PRInputs)$
33: **if** $MCR_{all} < MCR_V$ **then**
34: $V = [1...m]$
35: **end if** ▷ Stage 1 of EFSDPR ends here
36: Define a $|V| \times |V|$ matrix A_V with $A_V(i_1, i_2) = A(V(i_1), V(i_2))\ \forall i_1, i_2 \in [1...|V|]$
37: Define a $|V| \times 1$ vector W_V with $W_V(i_1) = W(V(i_1))\ \forall i_1 \in [1...|V|]$
38: $D_V = MCCMCALG(A_V, N_{lim}, W_V, MCCMCInputs)$ ▷ The set D_V represents the selected decomposition. $D_V(j)$ is the set of sensors in the j^{th} subsystem. The indexing of sensors in D_V may not be the indexing input by the user at this step and may have to be updated
39: $B = |D_V|$
40: **for** $j = 1$ to B **do**
41: $S = D_V(j)$
42: Define a set S_j (with $|S_j| = |S|$) containing the indices of the sensors in the j^{th} subsystem of the selected decomposition. It is given by $S_j(i) = V(S(i))\ \forall i \in [1...|S|]$.
43: **end for**
44: **return** $D = \{S_1, S_2, \cdots, S_B\}$
45: **end procedure**

Remark 7: The input parameter NP_{max} of EFSDPR is the maximum number of passes in the sensor selection stage of EFSDPR. If the value of NP_{max} is unspecified, then the sensor selection stage ends only when there is no change in the set of selected sensors from one pass to the next. The user can instead input a small value of NP_{max} to ensure that the computation time of the sensor selection stage is reduced as it terminates more quickly. Inputing a small value of NP_{max} is more important if the number of sensors in the system is large and if the computation time of a pattern recognition method simulation is large. If the input filter method is expected to work well in ranking the sensors correctly in terms of their importance for fault diagnosis, then it is recommended that the value of NP_{max} should be set equal to two.

Remark 8: The user should consider two factors before choosing a filter method to input into EFSDPR. First is its performance in correctly ranking the sensors based on their importance for fault diagnosis and second is the computation time it requires to rank the sensors. If the number of sensors in the system is very large, then the computation time of the filter method becomes a more important factor. The use of pattern recognition method simulations in the sensor selection stage ensures that the performance of the set of sensors it selects is less sensitive to the input filter method. If the filter method incorrectly gives a sensor a high rank when it is irrelevant for fault diagnosis, then a simulation of the pattern recognition method will identify that the sensor should not be selected in the sensor selection stage. However, using a better performing filter method will ensure that a near optimal set of sensors is selected after a smaller number of passes of the sensor selection stage.

6. Pseudocode of the GEA Algorithm

The pseudocode of the GEA algorithm is given in Algorithms 14 and 15.

Algorithm 14 Greedy Equal Allocation Algorithm

1: **procedure** GEA(D_M, M, ML)

2: The input D_M is a set whose j^{th} element is a set containing the indices of the sensors of one of the subsystems that needs to be allocated to a monitoring location. If the GEA algorithm is applied to select the decomposition for distributed data storage, then input D_M is a m element cell with $D_M(i) = \{i\}$ i.e. each element of D_M represent a single sensor. The input M is a set whose k^{th} element is a set containing the indices of the sensors that the k^{th} monitoring location is equipped to receive measurements from. The input ML is a $|D_M| \times 1$ vector. If $ML(j) = k$, then this means that a constraint is placed that subsystem $D_M(j)$ must be allocated to the k^{th} monitoring location ($D_M(j) \subset M(k)$ for this constraint to be consistent). If $ML(j) = 0$, then $D_M(j)$ can be allocated to any monitoring location that is equipped to receive measurements from all of its sensors.

3: $B_M = |D_M|$ ▷ Total no. of subsystems

4: $N_M = |M|$ ▷ No. of monitoring locations

5: $m = \max_i\{i \mid i = \max M(k), k \in [1...N_M]\}$ ▷ No. of sensors in the system

6: Define a set MON with B_M elements whose j^{th} element is the set of monitoring locations that subsystem j is allocated to.

7: **for** $j = 1$ to B_M **do**

8: **if** $ML(j) > 0$ **then**

9: $MON(j) = \{ML(j)\}$

10: **else**

11: $MON(j) = \{k \mid k \in [1...N_M], D_M(j) \subset M(k)\}$

12: **end if**

13: **end for**

14: Define a set M^{sub} with N_M elements whose k^{th} element is the set of subsystems allocated to the k^{th} monitoring location. $M^{sub}(k) = \{j \mid j \in [1...B_M], k \in MON(j)\}\ \forall k \in [1...N_M]$

15: ▷ Algorithm continues on the next page

Algorithm 15 Greedy Equal Allocation Algorithm (cont.)

16: **for** $k = 1$ to N_M **do**

17: $M^{sensor} = \bigcup_{j \in M^{sub}(k)} D_M(j)$

18: $ObjF(k) = (\max\{|M^{sensor}| - \frac{m}{N_M}, 0\}) \times B_M \times N_M + (\max\{|M^{sub}(k)| - \frac{B_M}{N_M}, 0\})$ ▷ $ObjF(k)$ is the contribution to the objective function by the k^{th} monitoring location

19: **end for**

20: $J_{rem} = \{j \mid j \in [1...B_M], |MON(j)| > 1\}$ ▷ J_{rem} is the set of subsystems that have been allocated to more than one monitoring location. The algorithm proceeds by deallocating subsystems in J_{rem} from monitoring locations until each subsystem has been allocated to one of the monitoring locations.

21: **while** $J_{rem} \neq \emptyset$ **do**

22: **for** $j \in J_{rem}$ **do**

23: **for** $k \in MON(j)$ **do**

24: $M^{sensor} = \bigcup_{j_1 \in M^{sub}(k), j_1 \neq j} D_M(j_1)$

25: $\Delta ObjF_{jk} = (\max\{|M^{sensor}| - \frac{m}{N_M}, 0\}) \times B_M \times N_M + (\max\{|M^{sub}(k)| - 1 - \frac{B_M}{N_M}, 0\}) - ObjF(k)$

26: **end for**

27: **end for**

28: $(j_{opt}, k_{opt}) = \arg\min_{(j,k)} \{\Delta ObjF_{jk} \mid j \in J_{rem}, k \in MON(j)\}$

29: $M^{sub}(k_{opt}) = M^{sub}(k_{opt}) \setminus \{j_{opt}\}$

30: $MON(j_{opt}) = MON(j_{opt}) \setminus \{k_{opt}\}$

31: **if** $|MON(j_{opt})| = 1$ **then**

32: $J_{rem} = J_{rem} \setminus \{j_{opt}\}$

33: **end if**

34: Setting $k = k_{opt}$, update $ObjF(k_{opt})$ using lines 17 and 18

35: **end while**

36: **return** M^{sub}

37: **end procedure**

Remark 9: The primary objective of the GEA algorithm is to allocate the sensors among the monitoring locations as close to equally as feasible and to minimize the overlap between the sets of sensors transmitting to the different monitoring locations. The secondary objective of the GEA algorithm is to allocate the subsystems among the monitoring locations as close to equally as feasible. The GEA algorithm has these objectives in order to minimize the number of monitoring locations that each sensor has to transmit to and to equally distribute the computational load of implementing the monitoring methods among the monitoring locations. Let the set S_j^M ($j \in [1...B_M]$) be the set of sensors in a subsystem (indexed by the integer j) where B_M is the total number of subsystems. This includes the subsystems in the decompositions selected for distributed data storage, fault detection and fault diagnosis. Let M_k^{sub} ($k \in [1...N_M]$) be the set of subsystems allocated to the k^{th} monitoring location. The objective of the GEA algorithm is to select M_k^{sub} to

minimize the objective function:

$$\sum_{k=1}^{N_M} \left(\left(\max\{|M_k^{sensor}| - \frac{m}{N_M}, 0\} \right) \times B_M \times N_M + \max\{|M_k^{sub}| - \frac{B_M}{N_M}, 0\} \right) \tag{9}$$

where:

$$M_k^{sensor} = \bigcup_{j \in M_k^{sub}} S_j^M \qquad \forall k \in [1...N_M] \tag{10}$$

Each subsystem must be allocated to one of the monitoring locations. If a subsystem is allocated to a monitoring location, then the monitoring location must be originally equipped to receive measurements from all of the subsystem's sensors. If the GEA algorithm is applied to select the decomposition for distributed data storage, then set $S_i^M = \{i\}$ $\forall i \in [1...m]$.

Remark 10: The input vector ML encodes the user-imposed must-link constraints that force a particular subsystem to be allocated to a specific monitoring location in the GEA algorithm. These constraints are useful if a set of subsystems need to be allocated to monitoring locations using the GEA algorithm after a different set of subsystems have already been allocated to monitoring locations. For example, let us consider a scenario in which the normal data required to implement a distributed fault detection method is initially available, but the faulty data data required to implement a distributed pattern recognition method is initially unavailable. Therefore, a decomposition is selected for distributed fault detection and its subsystems are allocated to monitoring locations using the GEA algorithm. The data required to implement the distributed pattern recognition method becomes available after a certain period of time and its decomposition is selected. If the GEA algorithm is now applied a second time to allocate all the subsystems to monitoring locations, then the user can impose a must-link constraint between each subsystem of the decomposition selected for distributed fault detection and the monitoring location it was initially assigned to (in the earlier application of the GEA algorithm) for ensuring that the subsystem is not allocated to a different monitoring location in the GEA algorithm.

7. Case Study: System with 1000 Sensors

The proposed EFSDPR and ESASA methods are applied to an artificial system with 1000 sensors to show that they can be applied to large-scale systems. The mean value of all the sensors is zero if process operation is normal and the covariance matrix (R) of the sensors is:

$$R(i_1, i_2) = 0.9^{|i_1 - i_2|} \qquad \forall i_1, i_2 \in [1...1000] \tag{11}$$

The system is known to have been affected by 300 different faults. Each of these faults causes a step change in the mean value of the measurements of a subset of the sensors. The magnitude of each of these faults is variable. Therefore, each fault is characterized by a specific direction in the measurement space. The direction of each of the 300 faults is generated using a randomized algorithm. Let the unit vector θ_j represent the direction of the j^{th} fault. θ_j is generated by first randomly selecting the number of sensors affected by the fault and then randomly selecting the sensors that are affected by the fault. The components of θ_j corresponding to the sensors affected by the fault are set equal to random values between -1 and 1. The component of θ_j corresponding to sensors unaffected by the fault are set equal to zero. Finally, the resulting vector θ_j is normalized.

7.1. EFSDPR

The EFSDPR method is applied to select a decomposition of the system with 1000 sensors in order to implement a distributed pattern recognition method which is described in detail in an earlier section of the supporting information. The pattern recognition method implemented in each subsystem comes to the diagnosis that a new sample is most likely generated due to a specific fault if the sample is most collinear with the specific fault's direction when compared to other fault directions. This pattern recognition method is used in the case study since the faults are assumed to not have a fixed magnitude. The local diagnoses of the subsystems are then combined using a voting based consensus strategy. The decomposition for distributed pattern recognition is selected subject to cannot-link constraints. The matrix A encoding these cannot-link constraints is generated using a randomized algorithm in which the probability of a cannot link constraint between two sensors is 0.1. The filter method used to implement the EFSDPR method ranks the sensors based on the number of faults that each sensor is affected by. The sensor affected by a larger number of faults is given a higher rank. The maximum number of passes in stage 1 of EFSDPR is set equal to two. The tabu search based algorithm is used to solve the MCCMC problem and the weights in W are set equal to one. A faulty training dataset is generated to implement the EFSDPR method and it contains 100 samples for each of the 300 faults. A sample contains a measurement from each of the 1000 sensors. The fault magnitude is set between $3-4$ to generate the samples in the dataset. The samples of the same fault are generated using the same fault magnitude. Different fault magnitudes are set to generate samples of different faults in the dataset.

The EFSDPR method selects a feasible decomposition with 23 subsystems. All 1000 sensors are selected by EFSDPR to implement the distributed pattern recognition method. The computation time of the EFSDPR method is 6697 seconds in this case study. Almost all of this time is spent on implementing the sensor selection stage of EFSPDR in which the number of pattern recognition method simulations is in the order of magnitude of 10^3 for this case. The sensor allocation stage takes a few seconds to complete in this case study. The FSDPR method is not able to select a decomposition of this system for distributed pattern recognition within 24 hours due to its size. In the first stage of the FSDPR method, the MCCMC problem is formulated as an integer linear optimization problem and solved to find the number of subsystems and the set of sensors that

can be allocated to each subsystem without violating the cannot-link constraints. The number of constraints in the integer optimization problem for this case study is in the order of 10^6 which is prohibitively large. In the second stage of FSDPR, a variant of the forward selection algorithm is used to generate a large number of feasible candidate decompositions and the decomposition with the lowest MCR is selected. This stage of FSDPR requires simulations of the pattern recognition method and the consensus strategy in order to calculate the MCRs of the candidate decompositions. The number of pattern recognition method simulations (i.e. in the order of magnitude of 10^4) required in FSDPR for this case is in the order of 10 times larger when compared to EFSDPR. The number of consensus strategy simulations required is in the order of magnitude of 10^5 in FSDPR for this case and these simulations are not required in EFSDPR. Therefore, the computation time required for running simulations in FSDPR is also prohibitively large in this case. On the other hand, the use of the filter method in EFSDPR helps keep its computation time low when compared to FSDPR and allows EFSDPR to scale to systems with larger number of sensors.

Since the performance of the decompositions selected by the EFSDPR and FSDPR methods cannot be compared in this case study, the performance of the EFSDPR decomposition will be compared with the centralized configuration and the univariate configuration. In the centralized configuration, the decomposition is a single subsystem containing all the selected sensors. The centralized configuration violates all the cannot-link constraints and it is the decomposition that EFSDPR would select if no constraints were imposed. In the univariate configuration, the decomposition has 1000 subsystems with each subsystem containing one of the selected sensors. The univariate configuration satisfies all the imposed cannot-link constraints. A testing dataset is generated to compare the performance of the different decompositions and calculate their MCRs. The testing dataset also contains 100 samples per fault. However, each sample in the testing dataset is generated by taking the average of 10 measurements of each sensor. This is done to increase the signal to noise ratio in the sample. The fault magnitude is again set between $3 - 4$ to generate samples for each fault. The fault magnitudes used to generate the testing dataset are different from the fault magnitudes used to generate the training dataset. The MCRs of the centralized configuration, the EFSDPR decomposition and the univariate configuration are 0, 0.0233 and 0.6876 respectively. The centralized configuration with the smallest number of subsystems has the smallest MCR of the three decompositions, whereas the univariate configuration with the largest number of subsystems has the largest MCR. This suggests that the performance of the distributed pattern recognition method has a tendency to degrade as the number of subsystems increases. The higher MCR of the EFSDPR decomposition when compared to the centralized configuration shows that the operational benefit of satisfying the imposed cannot-link constraints comes at the cost of poorer diagnostic performance in this case. The MCR of the EFSDPR decomposition is however reduced to zero if the samples in the testing dataset are generated by averaging 20 measurements instead of 10.

7.2. ESASA

The ESASA method is also applied in this case study. Let us assume that $A_M = A$ i.e. the cannot-link constraints that define the pairs of sensors that cannot transmit to the same location are the same cannot-link constraints imposed in the previous subsection to select the decomposition for distributed pattern recognition. The MCCMC problem is first solved for the system using the tabu search based algorithm. The algorithm is able to find the number of monitoring locations required and the allocation of sensors among the monitoring locations for this case study within a few seconds. A total of 23 monitoring locations are required for distributed monitoring. If the approximate greed algorithm is applied, then the number of monitoring locations it selects would be 25. Therefore, the tabu search based algorithm is able to find a better solution to the MCCMC

problem than the approximate greedy algorithm in this case study. The GEA algorithm is then applied in this case study to select the decomposition for distributed data storage and it is able to select the decomposition within a few seconds. The matrix A_M is then updated based on the sensor allocation among the monitoring locations using equation 3 of the manuscript. It should be noted that $A_M^* \neq A_M$ in this case study implying that additional cannot-link constraints have to be imposed to ensure that the subsystems can be allocated among the 23 monitoring locations and additional monitoring locations are not required. Let us assume that a distributed implementation of Hotelling's T^2 test [4] is used for fault detection. The decomposition for distributed fault detection is selected using the Performance Driven Agglomerative Clustering T^2 (PDACT2) method [4] subject to the cannot-link constraints encoded in A_M^*. PDACT2 is a modified version of PDAC (i.e. the decomposition method for distributed fault detection used in the TEP case study) that can scale to systems with larger number of sensors than PDAC because it uses quick to compute analytic expressions for evaluating the detection performance of different subsystems and decompositions. The decomposition selected for distributed fault detection has 53 subsystems. The decomposition for distributed pattern recognition is selected using the EFSDPR method. After the decompositions are selected, the GEA algorithm is applied to allocate the subsystems among the monitoring locations and the sets M_k^{sensor} are then updated using equation 4 of the manuscript.

The ESASA method and the SASA method are also compared in this case study. Let us assume that the covariance matrix of the sensors and the direction of the faults is known through process knowledge before the system's operation is started. Therefore, the SASA method can be applied in this case study to find the number of monitoring locations and the subsystem and sensor allocation. In the SASA method, the decompositions for distributed fault detection and distributed pattern recognition are first selected subject to the cannot-link constraints encoded in A_M using the PDACT2 and EFSDPR methods respectively. The decomposition selected for distributed pattern recognition is the same as the one selected in the ESASA method. However, the decomposition selected for distributed fault detection is different and has 136 subsystems. A mixed integer linear optimization problem is then solved in the SASA method to find the minimum number of monitoring locations and to allocate the sensors and subsystems among the monitoring locations subject to the imposed constraints. In the SASA method, the decomposition for distributed data storage can be selected after the mixed integer linear optimization problem is solved to find the number of monitoring locations and the subsystem and sensor allocation. The number of monitoring locations calculated using the SASA method is 150 in this case study which is much higher than the ESASA method. This is because the minimum number of monitoring locations in the SASA method is calculated subject to the cannot-link constraints and constraints that require each subsystem's senors to be allocated to the monitoring location that the subsystem itself is allocated to. The minimum number of monitoring locations in the ESASA method, on the other hand, is determined only subject to the cannot-link constraints. Additional cannot-link constraints are then imposed in the ESASA method to select the decompositions to ensure that each subsystem can be allocated to one of the 23 monitoring locations that is equipped to receive the measurements from all of the subsystem's sensors. The decompositions in the SASA method, on the other hand, are selected subject to the original set of cannot-link constraints encoded in A_M. Therefore, the advantage of the SASA method over the ESASA method in this case is that it imposes smaller number of cannot-link constraints in selecting the decompositions and therefore the performance of the decomposition it selects for distributed fault detection is better than the decomposition selected in the ESASA method. The advantage of the ESASA method over the SASA method is that it selects a smaller number of monitoring locations. It should be noted that the large disparity between the number of monitoring locations selected using the ESASA and SASA methods in this case study

is because the imposed cannot-link constraints are set randomly and because the decomposition selected for distributed fault detection in the SASA method has a large number of subsystems. It should also be noted that if $A_M^* = A_M$, then the ESASA method and the SASA method should select the same number of monitoring locations.

8. Decompositions Selected in the TEP Case Study

Table 3: Decomposition selected for distributed pattern recognition in the TEP case study

Subsystem Index	Sensors in the Subsystem
1	1,2,9,11,20-36,42-79,81
2	1,2,11,15,16,18,20,23-45,50-80

Table 4: Decomposition selected for distributed data storage in the TEP case study

Subsystem Index	Sensors in the Subsystem
1	1-14,20-36,42-49,81,82
2	15-19,37-41,50-80

References

[1] R. Lewis, A Guide to Graph Colouring, Springer, Berlin, 2015.

[2] Q. Wu, J.-K. Hao, F. Glover, Multi-neighborhood tabu search for the maximum weight clique problem, Ann. Oper. Res. 196 (1) (2012) 611–634.

[3] J. Hopcroft, R. Tarjan, Algorithm 447: Efficient algorithms for graph manipulation, Commun. ACM 16 (6) (1973) 372–378.

[4] S. Khatib, P. Daoutidis, Multiple hotellings T^2 tests for distributed fault detection of large-scale systems, Comput. Chem. Eng. 136 (2020) 106807.