Fault-Tolerant Classification in Multisensor Networks Using Coding Theory

Tsang-Yi Wang  
EECS Department  
Syracuse University  
Syracuse, NY 13244, USA  
tswang@ecs.syr.edu

Yunghsiang S. Han  
Department of CSIE  
National Chi Nan Univ.  
Taiwan R.O.C.  
yshan@csie.ncnu.edu.tw

Pramod K. Varshney  
EECS Department  
Syracuse University  
Syracuse, NY 13244, USA  
varshney@syr.edu

Abstract – A fault-tolerant distributed classification system employing the fault-tolerant fusion rule approach was proposed in [1]. Code matrix design is essential for the design of such systems. Two efficient code matrix design algorithms are proposed in this paper. The relative merits of both algorithms are also studied. Performance evaluation of the DCFECC (distributed classification fusion using error correcting codes) approach with faults due to hardware/software damage or drained batteries in sensors are provided. These results have shown significant improvement of fault-tolerance capability as compared with conventional parallel fusion networks. A generalization of the above approach that may handle channel transition errors is also provided.

Keywords: Distributed classification, Decision fusion, Multisensor fusion, Error correcting codes, Fault-tolerance

1 Introduction

Distributed classification in sensor networks has been considered for a wide range of applications such as surveillance systems and remote sensing. An important issue in the design of classification systems, especially in harsh environments, is their fault-tolerance capability. In real world applications, sensors may fail due to physical hardware or software damage, drained batteries, and environmental interference. In the presence of one or more failed sensors the system should still perform a reasonably accurate detection/classification.

Several researchers have considered the design of fault-tolerant distributed detection systems given a priori fault probability [2, 3, 4, 5]. However, they only considered the binary detection problem. Therefore, a parallel classification fusion network with fault-tolerance ability is considered in this paper. Specifically, we consider the fault-tolerant design for situations when stuck-at faults may occur at local detectors, hardware/software may be damaged, batteries may run out, or channel errors may occur during data transmission. Very recently, we proposed a fault-tolerant distributed classification fusion approach using error correcting codes (DCF ECC) [1] that successfully coped with stuck-at faults present in sensor networks. Unlike the optimal fusion rule [6], the fault-tolerant fusion rule given in [1] provided enough distance among all the decision regions corresponding to their hypotheses. The observed local decision vectors could still fall into correct decision regions even when several failures of sensors occur. In addition to having good fault-tolerance ability, the DCFECC approach also reduced the memory requirement. The reduction in memory requirement is obtained due to the Hamming distance operations employed in the fault-tolerant fusion rule. Moreover, less memory results in less energy consumption and hardware cost. These potential advantages make the DCFECC approach quite suitable for use in modern micro-sensor networks [7].

The DCFECC approach was shown in [1] to have good performance when stuck-at faults are present. However, there are still two major design questions left in [1]. First, there is the lack of a systematic way to search for the code matrix incorporated at the fusion center. Second, in addition to stuck-at faults, there are several other types of sensor failures that are considered. In this paper, a gradient-based algorithm, which searches for a code matrix within the set of local optimal solutions, and a simulated annealing based algorithm, which is a computational heuristic for obtaining approximate solutions to combinatorial optimization problems, are proposed to search for an optimal code matrix. Furthermore, we investigate the performance of DCFECC in the presence of drained batteries, hardware or software damage, and channel...
transition errors. To examine the performance in the presence of channel transition errors, we extend the original DCFECC formulation by incorporating the a priori probability of channel transition errors.

The rest of this paper is organized as follows. System description and a brief introduction to the DCFECC approach are given in Section 2. In Section 3, we present the search algorithms for an optimal code matrix. In Section 4, the generalized DCFECC formulation with known a priori probabilities is derived. We also evaluate the performance of the DCFECC approach in the presence of several types of faults in this section. Concluding remarks are given in Section 5.

![Figure 1: Distributed classification architecture](image)

2 Preliminaries

The DCFECC approach is based on the concept of error-correcting codes. The fusion rule at the fusion center is designed as follows. For the fusion center, we first design an error-correcting code matrix. Each codeword forms a row in the code matrix and corresponds to one of the classes to be distinguished. Each local detector (sensor) employs a decision rule based on the corresponding column in the code matrix. The fusion center decides on the class based on the binary inputs received (the received vector) from the local detectors. To provide fault-tolerance ability, the fusion center performs minimum distance decoding, i.e., it decides on the codeword that is closest in Hamming distance to the received vector, where the Hamming distance between two binary vectors is defined as the number of distinct positions between these vectors. This decision on a codeword is equivalent to making the M-ary decision regarding the classes, i.e., to making a classification decision. The fault-tolerance or error correction capability of the system is determined by the minimum Hamming distance of the code employed. A general distributed classification architecture is given in Fig. 1.

Let $H_\ell$, where $\ell = 0, 1, \ldots, M - 1$ and $M > 2$, denote the $M$ hypotheses to be distinguished at each of the $N$ sensors. Furthermore, the a priori probabilities of these $M$ hypotheses are denoted by $P(H_\ell) = P_\ell$ respectively. The observation at each local sensor or detector is represented by $y_i$, where $i = 1, \ldots, N$. Assume that the distribution function of $y_i$ under each hypothesis is known. Thus, the conditional probability density functions of these observations, when $H_\ell$ is given, can be represented by $P(y_i|H_\ell)$.

According to the DCFECC approach, a code matrix $C$ to perform distributed classification fusion is selected in advance. The code matrix is an $M \times N$ matrix with elements $c_{\ell i} \in \{0, 1\}$, $\ell = 0, \ldots, M - 1, i = 1, \ldots, N$. Each hypothesis $H_\ell \in \Omega = \{H_0, H_1, \ldots, H_{M-1}\}$ is associated with a row in the code matrix $C$. Each column of the matrix $C$ stands for a binary classification rule at each corresponding sensor. That is, each sensor employs a decision rule $g_i(y_i)$ to make a binary decision $u_i^*$, where

$$u_i^* = \begin{cases} 0, & \text{if } H_\ell^1 \text{ is declared;} \\ 1, & \text{if } H_\ell^0 \text{ is declared.} \end{cases}$$

$H_\ell^1 = \{H_\ell | c_{\ell i} = 1, H_\ell \in \Omega\}$ and $H_\ell^0 = \{H_\ell | c_{\ell i} = 0, H_\ell \in \Omega\}$ denote the two sets of classes with the properties $H_\ell^1 \cap H_\ell^0 = \emptyset$ and $H_\ell^1 \cup H_\ell^0 = \Omega$ which are determined according to the $i$th column of $C$. Note that each local sensor makes its decision by itself and is independent of the other sensors. After processing the observations locally, the local decisions $u_i^*$ are transmitted to the fusion center. The fusion center then receives a word consisting of the binary decisions made by the local sensors, which can be expressed as $u = (u_1, u_2, \ldots, u_n)$, where $u_i, i = 1, \ldots, n$, is the received local decision of sensor $i$ at the fusion center. In [1], authors assumed that $u_i = u_i^*$. However, in general, $u_i$ is not equal to $u_i^*$ when a channel transition error is present. Fault-tolerant fusion processing at each fusion center is carried out to obtain the multiclass decision $k$ based on the minimum Hamming distance decoding rule. That is, the multiclass decision is $H_k$ if $k = \arg \min \{d_H(x, y), \ell = 0, \ldots, M - 1\}$, where $d_H(x, y)$ is the Hamming distance between $x$ and $y$, and $c_\ell = (c_{\ell 1}, c_{\ell 2}, \ldots, c_{\ell N})$ is the row of $C$ corresponding to the hypothesis $H_\ell$. The tie-break rule is to randomly pick a codeword from those with the same smallest Hamming distance to the received vector.

The optimal local decision rules for the DCFECC approach must be determined after the code matrix is chosen. Let us define $C_{i_1, i_2, \ldots, i_N, \ell}$, where $i_1, i_2, \ldots, i_N \in \{0, 1\}$, as the cost that the received
word at the local fusion center, \( u = (u_1, u_2, \ldots, u_N) \),
equals \((i_1, i_2, \ldots, i_N)\) and the true hypothesis is \( H_t \).
These costs \( C_{i_1, i_2, \ldots, i_N, \ell} \) can be determined by the decision regions of codewords. According to the fault-
tolerant fusion rule at the fusion center, the decision region \( D \) of a codeword \( c \in C_w \) is given as follows:
\[
D(c) = \{ u | d_H(u, c) \leq d_H(u, c') \text{ for all } c' \in C_w \},
\]
where \( C_w = \{ c \ell | \ell = 0, \ldots, M - 1 \} \) is the set of all codewords, i.e., all rows of the code matrix. In order to minimize the probability of misclassification, set \( C_{i_1, \ldots, i_N, \ell} = 0 \) if \((i_1, \ldots, i_N)\) is in the decision region of \( c \ell \) that is the row of \( C \) corresponding to the hypothesis \( H_{\ell} \); otherwise set \( C_{i_1, \ldots, i_N, \ell} = 1 \). Whenever a received vector \((i_1, \ldots, i_N)\) simultaneously belongs to decision regions of \( c_0, c_1, \ldots, c_{q-1} \), where \( q > 1 \), for all \( \ell = 0, \ldots, q - 1 \), set \( C_{i_1, \ldots, i_N, \ell} = (1 - 1/q) \).

The probability of misclassification can then be expressed as
\[
P_e = \sum_{i_1, \ldots, i_N, \ell} \int_{y_1, \ldots, y_N} P(u_1 = i_1, \ldots, u_N = i_N, y_1, \ldots, y_N, H_{\ell}) 
\times C_{i_1, \ldots, i_N, \ell},
\]
(1)

Without considering channel transition errors, the optimal local decision rule at sensor \( k \) is obtained as [1]
\[
\sum_{\ell} P(y_k|H_{\ell}) K_{k\ell} > 0, \quad u_k^* = 1
\]
(2)
where
\[
K_{k\ell} = \sum_{i_1, \ldots, i_{k-1}, i_{k+1}, \ldots, i_N} \int_{y_1, \ldots, y_{k-1}, y_{k+1}, \ldots, y_N} \nu_{\ell} P(u_1|y_1) \times \cdots \times P(u_{k-1}|y_{k-1}) \times 
\times P(u_{k+1}|y_{k+1}) \cdots \times P(u_N|y_N) \times P(y_1, \ldots, y_{k-1}, y_{k+1}, \ldots, y_N|y_k, H_{\ell}) 
\times |C_{i_1, \ldots, i_{k-1}, 0, i_{k+1}, \ldots, i_N, \ell} - 
C_{i_1, \ldots, i_{k-1}, 1, i_{k+1}, \ldots, i_N, \ell}|,
\]
and \( u_k = u_k^*, k = 1, \ldots, N \). \( K_{k\ell} \) can be further simplified into
\[
K_{k\ell} = \sum_{i_1, \ldots, i_{k-1}, i_{k+1}, \ldots, i_N} \nu_{\ell} P(u_1|H_{\ell}) \times \cdots \times 
\times P(u_{k-1}|H_{\ell}) P(u_{k+1}|H_{\ell}) \cdots \times P(u_N|H_{\ell}) |C_{i_1, \ldots, i_{k-1}, 0, i_{k+1}, \ldots, i_N, \ell} - 
C_{i_1, \ldots, i_{k-1}, 1, i_{k+1}, \ldots, i_N, \ell}|
\]
(3)
if the observations \( y_1, y_2, \ldots, y_N \) are conditionally independent given their hypotheses.

### 3 Code design methodology

The objective of designing a good code matrix is that the fusion system has good performance in both fault-free and faulty situations. In general, the minimum Hamming distance in a code matrix should be as large as possible since larger Hamming distance between codewords can tolerate more faults. However, for the code matrix used in the DCFECC, larger Hamming distance does not always ensure good performance in both fault-free and faulty situations. System performance also depends on the patterns of columns in the code matrix, which determine the performance of local binary classifiers (detectors). If a code matrix has larger Hamming distance but with poor binary classifiers, then the overall system performance degrades. Moreover, when some local detectors fail to perform normally, the operation of the system will rely only on the other normal local detectors. Obviously, the fault-tolerance ability will also be degraded when poor binary local classifiers dominate the system performance. Therefore, a good code matrix should have a large minimum Hamming distance and simultaneously result in good local binary classifiers.

The adjustment of local decision rules in distributed classification makes the code design even more complicated. To achieve system-wide optimization, local sensors often use different decision strategies from the case when they are not in collaboration. Thus, the code matrix design can not be viewed as the independent design of individual column vectors (binary classifiers). Instead of analytically designing the code matrix with these interwoven rules, we propose two heuristic algorithms to efficiently solve the code design problem.

### 3.1 Code design by the gradient approach

For the gradient approach, we first set a minimum Hamming distance constraint for the required fault-tolerance capability. We then initialize the algorithm by picking a code matrix \( C^{(0)} = \{ c_1^{(0)}, c_2^{(0)}, \ldots, c_n^{(0)} \} \), where \( C^{(0)} \) must satisfy the minimum Hamming distance constraint. The first superscript \( i \) of each column in this matrix indicates that this column is the result after \( i \)th iteration. The second superscript indicates the column index of the code matrix. During the first iteration, the column vector \( c_1^{(0),1} \) is first replaced by \( c_1^{(1),1} \) so as to minimize \( P_e(c_1^{(1),1}, c_2^{(0),2}, \ldots, c_n^{(0),n}) \) and ensure that the code matrix \( \{ c_1^{(1),1}, c_2^{(0),2}, \ldots, c_n^{(0),n} \} \) fulfills the minimum Hamming distance requirement while other column vectors \( c_2^{(0),2}, \ldots, c_n^{(0),n} \) remain the same, where \( P_e(\{ c_1^{(1),1}, c_2^{(0),2}, \ldots, c_n^{(0),n} \}) \) denotes the probability of misclassification when \( \{ c_1^{(1),1}, c_2^{(0),2}, \ldots, c_n^{(0),n} \} \) is the code matrix in Equation (1). Second, the second
column $c^{(0),2}$ is replaced by $c^{(1),2}$ so as to minimize $Pe(\{c^{(1),1}, c^2, \ldots, c^{(0),n}\})$ and at the same time meet the minimum Hamming distance requirement. This procedure is continued for all the columns. The first iteration is complete after all the column vectors have been updated. Once the first iteration is complete, the next iterations are run in a similar way. The iterative algorithm terminates when the code matrix remains the same after an iteration.

It is surprising that this gradient-based algorithm converges very fast in all the experiments we have conducted. The algorithm took only about one or two iterations to converge. This indicates that there are lots of local optimal solutions in this code search procedure. In order to avoid convergence to a local optimal solution, we propose to use simulated annealing to search for a global optimal solution.

### 3.2 Code design by simulated annealing

Simulated annealing is a stochastic algorithm for obtaining approximate solutions to combinatorial optimization problems. It has been successful in many diverse applications. This algorithm is designed to search for the global optimum and is usually robust. Gamal et al. [8] have constructed good source codes, error-correcting codes, and spherical codes by using simulated annealing. For constructing the optimal code matrix here, the energy function is set to the probability of misclassification as shown in Equation (1). The minimum Hamming distance is also set to meet the fault-tolerance requirement.

The annealing schedule is decided as follows. We first generate some random code matrices and use them to determine the possible encountered range of values of $\Delta E = Pe(C) - Pe(C')$, where $C$ and $C'$ are potential candidates of code matrices. Then a starting value for the temperature $T = 1$ is chosen such that it is considerably larger than the largest $\Delta E$ normally encountered. The cooling control parameter $\alpha = 0.9$ is chosen for lowering the temperature using $T \leftarrow \alpha T$. This chosen value of $\alpha$ ensures slow cooling in the annealing process, and is essential for achieving a low energy state.

The random changes in the code configuration are achieved by perturbation of the code matrix. A codeword of the current code matrix is randomly selected and then two randomly selected bits of it are flipped. When the size of the code matrix is small, only one bit is flipped during the perturbation process to save computational time. Each time a new code matrix is generated we have to check whether or not the new code matrix meets the minimum Hamming distance requirement. If not, a new code matrix is generated according to the above code matrix perturbation until the minimum Hamming distance constraint is satisfied for the new code matrix. Specifically, the algorithm is given below:

**Step 1:** Set the minimum Hamming distance requirement, and initialize the algorithm by selecting a random code matrix $C$ satisfying the minimum Hamming distance constraint. Set the temperature $T = 1$. Compute the energy $Pe(C)$ according to equation (1).

**Step 2:** (Main iteration)

1. **Step 2.1:** Obtain $C'$ by perturbing $C$ until the minimum Hamming distance constraint is satisfied.

2. **Step 2.2:** Compute $\Delta E = Pe(C) - Pe(C')$, and replace $C$ by $C'$ if $\Delta E < 0$; otherwise replace $C$ by $C'$ with probability $e^{-\Delta E/T}$.

3. **Step 2.3:** Repeat Step 2.1 and Step 2.2 until several energy drop or too many iterations.

4. **Step 2.4:** Lower temperature by $T \leftarrow \alpha T$, and return to Step 2.1.

**Step 3:** Terminate the algorithm when a stable code matrix configuration is observed.

### 3.3 Numerical examples for code design

A fusion center and ten independent local sensors are considered to identify four equally likely hypotheses $H_0, H_1, H_2, H_3$ in the example presented here. Furthermore, we assume that the sensors are identical, that is, all sensor observations have the same characteristics. The observations are statistically independent. The probability density function for each hypothesis is assumed to be a Gaussian distribution with the same variance ($\sigma^2 = 1$) but with different means 0, $s$, 2$s$, and 3$s$, respectively. The signal-to-noise power ratio (SNR) is now 20$\log_{10}$dB since the variance of the noise is equal to one. The algorithms based on the gradient approach and simulated annealing were started with two different initial code matrices given in Table 1 and Table 2 respectively. It is easy to see that the minimum Hamming distance between any pair of the codewords in both matrices is 5. This value is set as the minimum Hamming distance requirement in both algorithms. We then searched the optimal code matrix when SNR= 0 dB. The optimal local decision rules were computed via the Gauss-Seidel cyclic coordinate descent algorithm [9]. Probability of misclassification as shown in Equation (1) was computed by numerical integration. The code matrices obtained from the gradient approach with these two initial code matrices are
shown in Table 3 and Table 4, respectively. Since the code matrix obtained from simulated annealing with both initial code matrices is the same, we only present one result in Table 5.

Table 1: The first initial code matrix
\[
\begin{array}{cccccccccc}
H_0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\
H_1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 \\
H_2 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\
H_3 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\
\end{array}
\]

Table 2: The second initial code matrix
\[
\begin{array}{cccccccccc}
H_0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 1 & 1 \\
H_1 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\
H_2 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 \\
H_3 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 1 \\
\end{array}
\]

Table 3: The code matrix obtained by the gradient approach with the first initial code matrix
\[
\begin{array}{cccccccccc}
H_0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \\
H_1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 \\
H_2 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
H_3 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 \\
\end{array}
\]

Table 4: The code matrix obtained by the gradient approach with the second initial code matrix
\[
\begin{array}{cccccccccc}
H_0 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\
H_1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\
H_2 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 \\
H_3 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
\end{array}
\]

Table 5: The code matrix obtained by the simulated annealing algorithm with both initial code matrices
\[
\begin{array}{cccccccccc}
H_0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 1 \\
H_1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 \\
H_2 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 \\
H_3 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\
\end{array}
\]

The probability of misclassification for each matrix is computed and plotted in Figure 2. One can observe that the performance of the gradient approach is dependent on its initial code matrix. The performance of the code matrix obtained by simulated annealing is better than that obtained by the gradient approach at all SNR values considered even though the code matrices are obtained at SNR = 0 dB. However, this is not true in general. Figure 3 shows the results, obtained from these two algorithms, for 5 \times 11 code matrices when eleven sensors are used for identifying five equally likely hypotheses. The observations are drawn from one of five Gaussian distributions with the same variance ($\sigma^2 = 1$) but with different means $-2s$, $-s$, $0$, $s$, and $2s$, respectively. The code matrices are obtained at 0 dB SNR for both the gradient approach and simulated annealing. One can see that the performance of the code matrix obtained by simulated annealing is better than that by gradient approach only at SNR values close to 0 dB. In order to investigate this situation we also obtained the code matrices when both algorithms operated at 5 dB SNR and illustrate their performances in Figure 4. This result shows that the performance of the code matrix obtained by simulated annealing is better than that obtained by the gradient approach at all SNR values considered. From Figures 3 and 4, we can also observe the following results. When the performance is evaluated at 0 dB, the code matrix obtained when the algorithms operate at 0 dB is better than that at 5 dB. Similarly, when the performance is evaluated at 5 dB, the code matrix obtained when the algorithms operate at 5 dB is better than that at 0 dB.

Figure 2: Performance comparison of 4 \times 10 code matrices designed at 0 dB by the gradient approach (both initial strategies) and simulated annealing

4 Performance evaluation with different types of faults

In this section we consider the faults caused by hardware or software damage, drained batteries at sensors,
and channel transition errors.

For modelling channel transition errors, we use a probabilistic fault model similar to that proposed in [4, 5] to generalize the DCFECC formulas. Let $p_{ii} = P(u_i = 1|u_i^* = 0)$ be the probability that the fusion center receives $u_i = 1$ when the local decision output in sensor $i$, $u_i^*$, is 0. The probability $p_{0i} = P(u_i = 0|u_i^* = 1)$ is similarly defined. The conditional probabilities of observed local decision $u_i$ at the fusion center given the observations at sensor $i$ can be expressed as

$$P(u_i|y_i) = P(u_i|u_i^* = 0)P(u_i^* = 0|y_i) + P(u_i|u_i^* = 1)P(u_i^* = 1|y_i).$$

By minimizing the probability of misclassification and employing the fault-tolerant fusion rule at the fusion center, the decision rule at local sensor $k$ can be modified as

$$u_k^* = 1$$

$$\sum_{\ell} P(y_k|H_\ell)K_{k\ell} > 0,$$  \hspace{1cm} (4)

$$u_k = 0$$

where

$$K_{k\ell} = \sum_{i_1, \ldots, i_{k-1}, i_{k+1}, \ldots, i_N} \int y_1 \ldots y_{k-1} y_{k+1} \ldots y_N P_i(u_1|y_1) \times \cdots \times P(u_{k-1}|y_{k-1}) \times P(u_{k+1}|y_{k+1}) \times \cdots \times P(u_N|y_N) \times P(y_1, \ldots, y_{k-1}, y_{k+1}, \ldots, y_N|y_k, H_\ell) \times [C_i \times \cdots \times 1, i_{k+1}, \ldots, i_N, \ell] \times C_i \times \cdots \times 1, i_{k-1}, i_{k+1}, \ldots, i_N, \ell] \times (1 - p_{ik} - p_{0k}).$$

Similarly, $K_{k\ell}$ can be simplify as

$$K_{k\ell} = \sum_{i_1, \ldots, i_{k-1}, i_{k+1}, \ldots, i_N} P_i(u_1|H_\ell) \times \cdots \times P(u_{k-1}|H_\ell)P(u_{k+1}|H_\ell) \times \cdots \times P(u_N|H_\ell)[C_i \times \cdots \times 1, i_{k+1}, \ldots, i_N, \ell] \times [C_i \times \cdots \times 1, i_{k-1}, i_{k+1}, \ldots, i_N, \ell] \times (1 - p_{ik} - p_{0k}),$$  \hspace{1cm} (5)

if the observations $y_1, y_2, \ldots, y_N$ are conditionally independent given their hypotheses. We can easily see that the Equations (4) and (5) reduce to Equations (2) and (3) when the communication channel is perfect.

Although the generalized DCFECC scheme given above includes the effect of channel transition errors, we found by numerical examples that its performance does not have significant improvement unless these a priori failure probabilities of channel are fairly large. This indicates that employing the fault-tolerant fusion rule has already captured most of the effect of channel transition errors.

When the faults are caused by hardware or software damage, there are two possible scenarios. First, local sensors still send incorrect decisions to the fusion center. In this case, the fusion center is unable to know that these decisions are wrong. These incorrect decisions could be present in many forms. In this paper, we assume that the faulty sensors randomly send 1 or 0 to the fusion center with equal probability. Second, the fusion center does not receive any decisions from local sensors. Unlike the first scenario, the fusion center is able to know that some damage occurred at local sensors. Drained batteries can also be treated as the second scenario of damaged sensors.
4.1 Numerical examples for performance evaluation with faults

In the following, we evaluate the performance of the generalized DCFECC with channel errors, and the DCFECC when two out of ten sensors have hardware/software damage. As in the previous example, we try to classify the objects coming from four equally likely hypotheses \( H_0, H_1, H_2, H_3 \). The probability density function for each hypothesis is the same as the previous example. The code matrix in Table 5 is used in this example. For performance comparison purposes, we design the fault-tolerant system with binary local decisions and employing the optimal fusion rule obtained by extending from the design of binary detection systems with channel errors [5]. We call this one bit fault-tolerant conventional approach (FCA). We also design a distributed classification system using binary local decisions without considering probabilistic fault model by extending from the theory of general binary fusion networks [10], and name this system one bit conventional approach (CA). We do not provide the formulas here due to space limitations. In this example, the optimal local decision rules and the optimal fusion rule for the FCA and CA approaches are computed via the Gauss-Seidel iterative algorithm. The computation of the optimal local decision rules for the DCFECC approach also employs the Gauss-Seidel iterative algorithm.

The performance comparison of generalized DCFECC and FCA when communication channels are not perfect is provided in Figure 5. In this example, we assume that \( p_{i1} = p_{2i} = p_i \), and design both the DCFECC and FCA at 5dB SNR while assuming that the probabilities of channel transition errors, \( p_i \), are 0.01, 0.05, and 0.1 for \( i = 1, \ldots, N \). The local decision rules of the generalized DCFECC are computed by Equations (4) and (5). Since the performance for the DCFECC approach under these three channel transition error probabilities cannot be distinguished, we only plot one result in Figure 5. In order to investigate the robustness of the two approaches, we evaluate their performances when the channel transition error probability, \( p_{ce} \), is varied from 0 to 0.3. The results illustrate that the generalized DCFECC is the most robust compared with the FCA. Its performance is only slightly worse than those of the FCA designed at \( p_i = 0.01 \), and \( p_i = 0.05 \) when \( p_{ce} \leq 0.04 \). We can also observe that performance curves for FCA become closer to that of the DCFECC approach as the channel transition error probability becomes larger.

In Figure 6, the performance comparison of DCFECC and CA when two sensors are with first type of hardware/software damage is provided. The results show that the performance of the DCFECC approach is much better than CA at all the SNR values considered.

When the second type of sensor damage is considered, the system is operated in a different way. For the DCFECC approach, the fusion center treats the observed local decision \( u_i \) as 0, if \( P(u_i = 0) > P(u_i = 1) \); it sets \( u_i = 1 \) if \( P(u_i = 0) < P(u_i = 1) \). The tie-break rule is to randomly make the decision in favor of 1 or 0 with equal probability. For the CA, the fusion center simply ignores the damaged sensors. Note that, for the CA, the performance with this strategy (ignoring the damaged sensors) is better than that of guessing the most likely local decisions. It is because fusing incorrect information results in a much worse performance than simply ignoring it. The performance comparison of DCFECC and CA with the second type of sensor damage is illustrated in Figure 7. The results show that the performance of CA is better than that of DCFECC. However, the performance of CA could be much worse than that of DCFECC when other faults are also present such as stuck-at faults.

5 Conclusions

The DCFECC approach, the distributed classification system employing the fault-tolerant fusion rule, has shown great fault-tolerance capability in [1]. In this paper, we have developed two efficient algorithms to search for good code matrices for implementing the DCFECC approach. The gradient approach is usually fast but may converge to a local optimum depending on the chosen initial code matrix. On the other hand, simulated annealing approach is robust to the selection.
of the initial code matrix, and has better performance even though it takes more time to converge.

The performance of DCFECC with different types of faults has also been evaluated. In addition, we generalized the DCFECC approach by modelling the channel transition error with a known a priori probability. It is shown that DCFECC has great fault-tolerance capability under different types of faults, except the second type of sensor damage. Thus, in order to tolerate most types of faults, the system should be designed using the DCFECC approach.

Acknowledgment

This work was supported in part by the W.M. Keck Foundation under grant No. CK #1329, by the SUPRIA program of the CASE Center at Syracuse University, and by the National Science Council of Taiwan, R.O.C., under grants NSC 90-2213-E-260-007 and NSC 91-2213-E-260-021.

References


