Fault Identification via Non-parametric Belief Propagation

Danny Bickson, *Member, IEEE*, Dror Baron, *Senior Member, IEEE*, Alexander Ihler, *Member, IEEE*, Harel Avissar, *Student Member, IEEE*, Danny Dolev, *Member, IEEE*

***Abstract*—We consider the problem of identifying a pattern of faults from a set of noisy linear measurements. Unfortunately, maximum a posteriori probability estimation of the fault pattern is computationally intractable. To solve the fault identification problem, we propose a non-parametric belief propagation approach. We show empirically that our belief propagation solver is more accurate than recent state-of-the-art algorithms including interior point methods and semidefinite programming. Our superior performance is explained by the fact that we take into account both the binary nature of the individual faults and the sparsity of the fault pattern arising from their rarity.**

***Index Terms*—compressed sensing, fault identification, message passing, non-parametric belief propagation, stochastic approximation.**

I. Introduction

FAULT identification is the task of determining which of a set of possible failures (faults) have occurred in a system given observations of its behavior. Such problems arise in a variety of applications, including aerospace [1, 2], industrial process control [3], and automotive systems [4].

A common approach to fault identification is to exploit a mathematical model of the system to construct *residuals* (deviations from the system's expected behavior), which can be used to detect and diagnose atypical behavior. A wide variety of approaches have been proposed for constructing residual measurements. For example, state space models of the system may be used to generate parity checks, or the magnitude of errors in a Kalman filter may be used as an observer to detect changes in behavior. The occurrence of a particular fault results in a characteristic change, or signature, which can be used to identify which fault or set of faults has occurred. For an overview and survey of residual methods, see Frank [5].

D. Bickson is with the Machine Learning Department, Carnegie Mellon University, Pittsburgh PA 15213, USA. (email: bickson@cs.cmu.edu) Parts of this work were performed when D. Bickson was a researcher at IBM Haifa Research Labs, Israel.

D. Baron is with the Department of Electrical and Computer Engineering at North Carolina State University, Raleigh, NC 27695, USA. (email: barondror@ncsu.edu) Parts of this work were performed when D. Baron was with the Department of Electrical Engineering at the Technion, Haifa 32000, Israel.

A. Ihler is with the Bren School of Information and Computer Science, University of California, Irvine CA 92697, USA. (email: ihler@ics.uci.edu)

H. Avissar and D. Dolev are with the School of Computer Science and Engineering, The Hebrew University, Jerusalem 91904, Israel. (email: harela01@cs.huji.ac.il, dolev@cs.huji.ac.il)

A. Binary faults and sparse signature matrix

A typical model describes the presence of each possible fault using a binary variable, and assumes that each fault affects the observed system measurements in a linear additive way. Systems of binary faults have been extensively studied in the computer science community, for example posing the problem as constraint satisfaction and using heuristic search techniques [6, 7]. An alternative approach is to use a convex relaxation of the original combinatorial problem, for example using the interior point method of Zymnis et al. [8].

In such systems there are often more potential faults than measurements, resulting in an under-determined set of linear equations. By assuming that faults are relatively rare, one can construct a combinatorial problem to identify the most likely pattern of faults given the system measurements.

Another common assumption is that the fault signature matrix, which is the linear transformation matrix applied to the fault vector, is sparse. This is the case in many applications, where each fault may affect only a small part (such as a single subsystem) of the whole [9–12]. The assumption that the signature matrix is sparse is especially valid in large scale systems where different faults affect different parts of the system. Although our algorithm is designed for sparse signature matrices, we find that it performs well even for moderately dense systems, as shown in the numerical results section below.

B. Related work

Fault identification is closely related to several coding and signal reconstruction problems. For example, in multi-user detection for code division multiple access (CDMA) systems [13–17], the system measurements are given by the received noisy wireless signal and the goal is to estimate the transmitted bit pattern, which plays the role of the fault pattern. One important difference is that in multi-user detection, each bit typically has an equal probability of being 0 or 1, whereas in fault identification the prior probability that a bit is 1 (indicating that a fault is present) is typically much lower.

Compressed sensing (CS) [18–26] is also closely related to fault identification. Informally, CS reconstructs a sparse signal from a set of (possibly noisy) linear measurements of the signal. As in fault identification, the system of linear equations is ill conditioned, and the assumption of sparsity (corresponding to the rarity of faults) is critical in the reconstruction. Because of this similarity between CS and fault identification,

in our numerical results section we compare the performance of several CS algorithms with previously proposed methods for fault identification.

C. Contributions

In this work, we develop a novel approach for fault identification based on a variant of the belief propagation (BP) algorithm called non-parametric belief propagation (NBP). BP approaches have been applied to solve many similar discrete, combinatorial problems in coding. NBP allows the algorithm to reason about real-valued variables. Variants of NBP have been used in CS [20] and low-density lattice codes (codes defined over real-valued alphabets) [27–29]. Our method constructs a relaxation of the fault pattern prior using a mixture of Gaussians, which takes into account both the binary nature of the problem as well as the sparsity of the fault pattern.

Using an experimental study, we show that our approach provides the best performance in identifying the correct fault patterns when compared to recent state-of-the-art algorithms, including interior point methods and semidefinite programming. To demonstrate the importance of each component of our model, we compare both to existing approaches for fault identification as well as to CS algorithms and a discrete BP formulation. We explain how to implement an efficient quantized version of NBP, and provide the source code used in our experiments [30]. Our favorable results can be explained in the context of recent theoretic results in the related domains of multi-user detection [13, 14] and CS [24–26]. In the large system limit, the posteriors estimated by BP converge in distribution to the true posteriors, and so NBP is asymptotically optimal.

The structure of this paper is as follows. Section II introduces the fault identification problem in terms of maximum a posteriori (MAP) estimation, and describes graphical models and the BP algorithm. Section III presents our solution, based on NBP. Section IV explains implementation details and optimizations. Section V compares the accuracy of several state-of-the-art methods for fault identification, and shows that our proposed method has the highest accuracy. We shed light on the favorable performance of NBP from an information theoretic perspective in Section VI, and conclude with a discussion in Section VII.

II. Fault Identification Problem

In this section we describe the fault model in detail, and the basic maximum a posteriori (MAP) approach for estimating the fault pattern. We then briefly review probabilistic graphical models and the belief propagation algorithm within this context. Our goal is to infer the MAP fault value, the fault pattern most likely to have occurred given a set of observations.

A. Fault model and prior distribution

We consider a system in which there are n potential faults, any combination of which (2^n in total) can occur. A fault pattern, i.e., a set of faults, is represented by a vector $x \in \{0,1\}^n$, where $x_s = 1$ means that fault s, $s \in \{1, \cdots, n\}$ has occurred. We assume that faults are independent and identically distributed (i.i.d.), and that fault s occurs with known probability p_s. Thus, the (prior) probability of fault pattern x occurring is

$$p(x) = \prod_{s=1}^{n} p_s^{x_s} \left(1 - p_s\right)^{1-x_s} .$$

The fault pattern $x = 0$ corresponds to the null hypothesis, the situation in which no faults have occurred, with probability $p(0) = \prod_{s=1}^{n}(1 - p_s)$. The expected number of faults is $\sum_{s=1}^{n} p_s$.

We assume that m scalar real measurements, denoted by y, $y \in \mathbb{R}^m$, are available. These measurements depend on the fault pattern $x \in \{0,1\}^n$ linearly:

$$y = Ax + v\,,$$

where $A \in \mathbb{R}^{m \times n}$ is the *fault signature matrix*, and the measurement noise $v \in \mathbb{R}^m$ is random, with v_i independent of each other and x, each with $\mathcal{N}(x; 0, \sigma^2)$ distribution. Typically the system of linear equations is under-determined ($n > m$), which means the number of potential faults is greater than the number of measurements.

The fault signature matrix A is assumed to be known. Its s^{th} column $a_s \in \mathbb{R}^m$ corresponds to the measurements, when only fault s has occurred and assuming no noise. For this reason a_s is called the s^{th} *fault signature*. We denote $\tilde{a}_j \in \mathbb{R}^n$ as the j^{th} row of the matrix A. Since x is a Boolean vector, Ax is the sum of the fault signatures corresponding to the faults that have occurred. We further assume A to be sparse, i.e., the percentage q of the non-zero values of A is much smaller than one. Indeed, the matrix A is sparse in many applications, where each fault may affect only a small part (such as a single subsystem) of the whole [9–12].

B. Analog circuit example

As an example, consider linear analog circuits, in which it is common to use *nodal analysis* to describe the behavior of the system; non-linear circuits may be approximately linearized to apply a similar approach. Each internal node of the circuit is assigned a variable representing its voltage v_i, and voltage and current sources are given variables indicating their induced current or voltage, respectively. Applying Kirchoff's current law, we know that the total current flowing from any internal node i must be zero; when these currents are written in terms of the nodal voltages they yield a set of linear equations $Av = w$ that describe the system, where $w_i = 0$ for internal nodes, and equals the known voltage/current of a fixed voltage/current source.

For example, for a sequence of three nodes i, j, k connected to their neighbors with resistances R_{ij}, R_{jk}, Kirchoff's law applied to v_j yields $\frac{v_j - v_i}{R_{ij}} + \frac{v_j - v_k}{R_{jk}} = 0$. One can solve for the internal voltages easily as $v = A^{-1}w$; we assume that some subset of these voltages are measurable for testing, $v_M = MA^{-1}w$, where M is a measurement matrix.

A fault, including an incorrect component value, short, or open-circuit, then corresponds to a localized change in the

matrix A associated with those two nodes. For example, a change $R_{jk} \to R'_{jk}$ adds the term $(v_j - v_k)(\frac{1}{R'_{jk}} - \frac{1}{R_{jk}})$ to row j of A, and subtracts it from row k. If the resulting new, faulty circuit matrix is A', the fault signature is then $v'_M - v_M = M(A'^{-1} - A^{-1})w$. Note that the underlying assumption is that the number of actual faults in a single circuit is small, which explains the imbalance in the fault prior probability $p \ll 1$. For more information about detecting faults in linear analog circuits, see e.g. [31].

C. Posterior probability

Let $p(x|y)$ denote the (posterior) probability of pattern x given the measurements y. By Bayes rule we have

$$\begin{aligned} p(x|y) &\propto p(y|x)p(x) = p(x,y) \\ &= \mathcal{N}(y\ ;\ Ax, \sigma^2 I) \prod_s p_s^{x_s}(1-p_s)^{1-x_s}\,. \end{aligned} \tag{1}$$

Letting C and C' indicate constants (values independent of x), we can define the log-loss function as the negative log probability $l_y(x) = -\log p(x|y)$ and write

$$\begin{aligned} l_y(x) &= \lambda^T x + \tfrac{1}{2\sigma^2}\sum_{i=1}^{m}(y-Ax)^T(y-Ax) + C \\ &= \tfrac{1}{2\sigma^2}x^T A^T A x + (\lambda - \sigma^{-2}A^T y)^T x + C'\,, \end{aligned} \tag{2}$$

where $\lambda_s = \log((1-p_s)/p_s)$ denotes the log-odds ratio. Note that (2) is a convex quadratic function of x (a binary-valued vector), and MAP estimation is thus a convex integer quadratic program.

D. Graphical Models

Graphical models are used to represent and exploit the *structure* of the cost function (2), or its associated probability distribution (1), to develop efficient estimation algorithms. Specifically, we factor $p(x|y)$ into a product of smaller functions, called factors, each of which is defined using only a few variables. This collection of smaller functions is then represented as a factor graph G, in which each variable is associated with a *variable node* and each factor with a *factor node*. Factor nodes are connected to variable nodes that represent their arguments. Because (2) can be represented as the sum of terms involving at most two variables,

$$l_y(x) = \sum_{s,t>s} J_{st}x_s x_t + \sum_s h_s x_s\,, \tag{3}$$

where

$$\begin{aligned} J_{st} &= \tfrac{1}{\sigma^2}(A^T A)_{st}\,, \\ h_s &= (\lambda - \tfrac{1}{\sigma^2}A^T y)_s + \tfrac{1}{2\sigma^2}(A^T A)_{ss}\,, \end{aligned}$$

and we can represent $p(x|y)$ or $l_y(x)$ as a pairwise graph with an edge between x_s and x_t if and only if J_{st} is non-zero. This corresponds to factoring $p(x|y)$ as

$$\begin{aligned} p(x|y) &\propto \prod_{i=(s,t>s)} f_i(x_s,x_t)\prod_s g_s(x_s) \\ &= \prod_{(s,t>s)} \exp(-J_{st}x_s x_t)\prod_s \exp(-h_s x_s)\,, \end{aligned}$$

where $f_i(\cdot)$ and $g_s(\cdot)$ are factors; we use the convention that functions g and indices s,t refer to local (single-variable) factors while f and indices i,j refer to higher-order (in this case, pairwise) factors. For compactness, we generically write $f(x)$ to indicate a function f over some subset of the x_s.

The graph structure is used to define efficient inference algorithms, including MAP estimation or marginalization. Both problems are potentially difficult, as they require optimizing or summing over a large space of possible configurations. However, structure in the graph may induce an efficient method of performing these operations. For example, in sequential problems the Viterbi algorithm [32] (and more generally, dynamic programming) provides efficient optimization, and can be represented as a message-passing algorithm on the graph G.

In graphs with more complicated structure such as cycles, exact inference is often difficult; however, similar message-passing algorithms such as loopy belief propagation perform approximate inference [33]. The max-product algorithm is a form of BP that generalizes dynamic programming to an approximate algorithm. One computes messages m^f_{is} from factors to variables and m^v_{si} from variables to factors,

$$m^f_{is}(x_s) \propto \max_{x\setminus x_s} f_i(x) \prod_{t\in\Gamma_i\setminus s} m^v_{ti}(x_t)\,, \tag{4}$$

$$m^v_{si}(x_s) \propto \quad g_s(x_s) \prod_{j\in\Gamma_s\setminus i} m^f_{js}(x_s)\,, \tag{5}$$

where messages are normalized for numerical stability, and Γ_s is the neighborhood of node s in the graph (all nodes for which $\tilde{a}_i$ is non-zero, excluding s). One can also compute a "belief" $b_s(x_s)$ about variable x_s,

$$b_s(x_s) = g_s(x_s)\prod_{i\in\Gamma_s} m^f_{is}(x_s)\,, \tag{6}$$

which can be used to select the configuration of x_s by choosing its maximizing value. If the graph G is singly-connected (no cycles), then the max-product algorithm is equivalent to dynamic programming. However, the algorithm performs well even in graphs with cycles, and has been shown to be highly successful in many problems, most notably the decoding of low-density parity check (LDPC) codes [34]. Max-product and its so-called reweighted variants are closely related to linear programming relaxation techniques [35–37], but by exploiting the problem structure can be more efficient than generic linear programming packages [38].

The sum-product formulation of BP is intended for approximate marginalization, rather than optimization. Despite this, the sum-product algorithm has been frequently applied to MAP estimation problems, as it often exhibits better convergence behavior than max-product [39]. It has an almost identical message-passing formulation,

$$m^f_{is}(x_s) \quad \propto \textstyle\sum_{x\setminus x_s} f_i(x)\prod_{t\in\Gamma_i\setminus s} m^v_{ti}(x_t)\,, \tag{7}$$

where the product step is computed as in (5). Again, one can estimate each x_s by choosing the value that maximizes the belief $b_s(x_s)$, in this case corresponding to the maximum posterior marginal estimator. Although the sum-product beliefs

are intended to approximate the marginal distributions of each x_s, if the most likely joint configurations all share a particular value for x_s, then this will be reflected in the marginal probability as well. Finally, a connection between non-asymptotic block length sum-product belief propagation and joint maximum likelihood or MAP detection was described in [40].

Variants of BP typically rely on graph sparsity (few edges) to ensure both efficiency and accuracy. When the graph has no cycles, these algorithms are exact; for graphs with cycles, they are typically only approximate but are often accurate in systems with long, weak, or irregular cycles [41]. Unfortunately, although the fault signature matrix A may be sparse, the same is typically not true of the matrix $J = A^T A$, especially for large m and n. In the dense case, a direct application of BP to (3) may fail. For example, in experiments with A sized 50×100 and approximately 10% non-zero values ± 1, $A^T A$ is approximately 50% non-zero, and BP algorithms defined using (3) did less well than the current state of the art methods. As the dimension sizes m and n are increased, $A^T A$ becomes dominated by non-zeros. This motivates us to define an alternative graphical model that relies on the sparsity of A itself.

III. Belief Propagation for Fault Identification

Since the structure of the matrix A is sparse, let us define an alternative graphical model that uses A explicitly. In particular, we can write the probability distribution

$$p(x, y) = \prod_i f_i(y_i, x) \prod_s g_s(x_s) \,,$$

in terms of the factors,

$$f_i(y_i, x) = \mathcal{N}(y_i; \tilde{a}_i x,\, \sigma^2), \quad g_s(x_s) = p_s^{x_s}(1-p_s)^{1-x_s}. \quad (8)$$

Notably, if A is sparse (specifically, if each row $\tilde{a}_i$ is sparse), then the factors f_i will depend only on the few x_s for which $\tilde{a}_i$ is non-zero and the graph representing this factorization will be sparse as well. An example factor graph is shown in Fig. 1(a).

Using either max-product (4) or sum-product (7) on the resulting factor graph is computationally difficult, as it involves eliminating (maximizing or summing over) all exponentially many configurations of the neighboring variables of f_i (2^d evaluations for factors over d variables). This can quickly become intractable for even moderate neighborhood sizes. Although the relationships among the x_s and y_i are simple and linear, our model defines a hybrid distribution over both continuous valued and discrete valued random variables. Observing y_i creates a combinatorial dependence among the neighboring discrete-valued x_s. Again, in our experiments with $n = 100$, $m = 50$, these computations became extremely slow for even modest values of q; for example, when $q \approx 0.15$, even the average factor has $d = 15$ and the largest factor is typically significantly higher.

Although it may seem counter-intuitive, we can avoid some of these difficulties by converting the graphical model to a fully continuous model. We abuse notation slightly to define continuous variables x_s in place of their discrete counterparts, with corresponding factors

$$g_s(x_s) = p_s \delta(x_s = 1) + (1 - p_s)\delta(x_s = 0) \,.$$

It will also prove convenient to relax the prior slightly into a Gaussian form, using the approximation

$$\hat{g}_s(x_s) = p_s \mathcal{N}(x_s; 1, \nu) + (1 - p_s)\mathcal{N}(x_s; 0, \nu) \,, \quad (9)$$

where the variance ν controls the quality of the approximation; as $\nu \to 0$, we recover the original prior on x_s. The factors $f_i(\cdot)$ and $g_s(\cdot)$ are illustrated in Fig. 1(b).

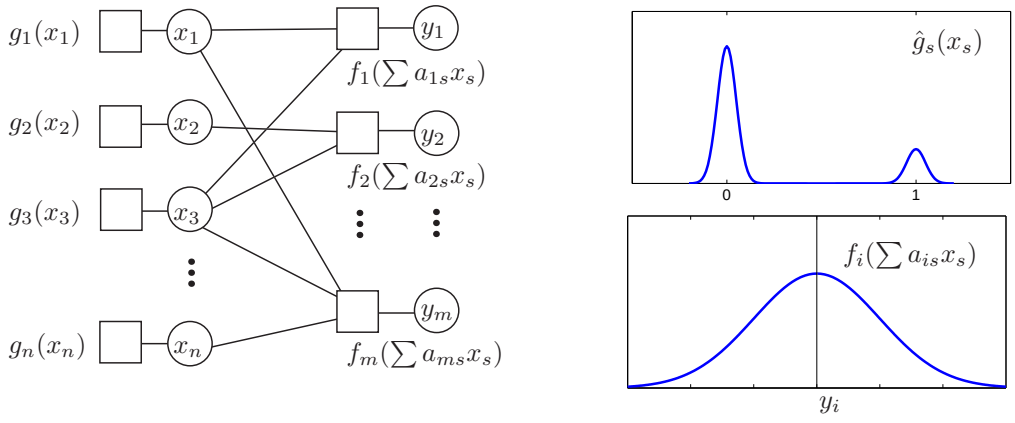
The BP message-passing algorithm remains applicable to models defined over continuous random variables. However, message representation is more difficult as we must represent continuous functions over those variables. For non-Gaussian models, message representation typically requires some form of approximation.

In our continuous model, both sets of factors f_i and $\hat{g}_s$ consist of mixtures of Gaussians; thus their product, and the messages computed during BP, will also be representable as mixtures of Gaussians [27, 33]. Unfortunately, at each step of the algorithm, the number of mixture components required to represent the messages will increase at an exponential rate, and must be approximated by a smaller mixture. To handle the exponential growth of mixture components, approximations to BP over continuous variables were proposed in NBP [33] and variants in a broad array of problem domains [20, 27, 28, 42]. Those approximations use sampling to limit the number of components in each mixture. A number of sampling algorithms have been designed to ensure that sampling is efficient [43–45]. Such sample based representations are particularly useful in high dimensional spaces, where discretization becomes computationally difficult. Various authors have also proposed message approximation methods based on dynamic quantization or discretization techniques [46, 47].

For the fault identification problem our variables x_s are one dimensional and can be reasonably restricted to a finite domain (Section IV-B). Thus, it is both computationally efficient and sufficiently accurate to use a simple uniform discretization over possible values, allowing our functions over x_s to be represented by fixed-length vectors [20, 27]. Although typically the term "non-parametric BP" refers to an algorithm using stochastic samples and Gaussian mixture approximations to the messages, rather than a fixed discrete quantization, here we use it more generically to distinguish BP in our fully continuous model from a standard discrete BP performed directly on (3).

IV. Implementation

In this section we discuss some details of our implementation of NBP and our overall fault identification algorithm. These include several techniques that make our algorithm more efficient, and two local optimization heuristics that may improve solution quality.



(a) (b)

Fig. 1: Graphical model for the fault identification problem. (a) Factor graphs represent dependence among variables, including faults $\{x_s\}$ and measurements $\{y_i\}$. Graph edges indicate non-zero values of $\tilde{a}_i$. (b) Potential functions $\hat{g}_s$ capture the binary and sparse nature of the x_s, and f_i represent the linear measurements with Gaussian noise.

A. Computation of messages

We apply the BP algorithm presented in (7) using the factors (8) and the self potentials defined by the Gaussian approximation (9). The variable-to-factor messages $m^v_{si}(x_s)$ are given as in (7), and we compute the message product more efficiently by noting that

$$g_s(x_s)\prod_{j\in\Gamma_s\setminus i} m^f_{js}(x_s) = \frac{g_s(x_s)\prod_{j\in\Gamma_s} m^f_{js}(x_s)}{m^f_{is}(x_s)}\,. \qquad (10)$$

The product on the r.h.s of (10) can be computed once and reused for each outgoing message [42, 48].

The factor-to-variable messages require a more detailed analysis. It is important to note that our factors $f_i(x)$ are functions of several variables, say $\Gamma_i = \{x_{s_1}\dots x_{s_d}\}$. A message computation, for example from factor i to variable s_1, can be explicitly written as

$$m^f_{is_1} \propto \int_{x_{s_2}}\dots\int_{x_{s_d}}\Big[f_i\Big(\sum_{j=1}^{d} a_{is_j}x_{s_j}\Big)\prod_{j=2}^{d} m^v_{s_ji}(x_i)\Big]dx_{\Gamma_i}\,. \qquad (11)$$

Although an arbitrary function over d variables would require $O(n^d)$ computation, where n is the number of discretization bins, the messages can be computed more efficiently, because f_i is a function over a linear combination of the x_s [27, 49]. In essence, one uses a change of variables to separate each integrand, defining variables for the cumulative sum, $\bar{x}_{s_j} = a_{is_j}x_{s_j}+\bar{x}_{s_{j+1}}$, and scaled messages $\bar{m}_{s_ji}(x) = m_{s_ji}(x/a_{is_j})$. We re-express (11),

$$m^f_{is_1}(x_{s_1}) \propto \int_{\bar{x}_{s_2}} f_i(a_{is_1}x_{s_1}+\bar{x}_{s_2})\int_{\bar{x}_{s_3}}\bar{m}_{s_2i}(\bar{x}_{s_2}-\bar{x}_{s_3})\int_{\bar{x}_{s_4}}\dots\bar{m}_{s_di}(\bar{x}_{s_d})d\bar{x}_{\Gamma_i},$$

in which each integral (approximated by a discrete sum) requires $O(n^2)$ computation. Note also that, computing from the right-hand side, each step takes the form $m_j(\bar{x}_j) = \int \bar{m}(\bar{x}_j-\bar{x}_{j+1})m_{j+1}(\bar{x}_{j+1})dx_{j+1}$, and can thus be thought of as a convolution operator, $m(\bar{x}_j)\otimes m(\bar{x}_{j+1})$ [50]. For convenience, we write this convolution as

$$m^f_{is}(x_s) \propto f_i(a_{is}x_s)\otimes\bigotimes_{t\in\Gamma_i\setminus s} m^v_{ti}(x_t/a_{it})\,. \qquad (12)$$

Moreover, since convolution can be computed by an element-wise product in the Fourier domain, the factor-to-variable messages can be evaluated even more efficiently by first rescaling each incoming message, transforming into the Fourier domain, taking a product, transforming back, and unscaling, resulting in the update rule [50],

$$m^f_{is}(x_s/a_{is}) \propto \mathcal{F}^{-1}\Big(\mathcal{F}(f_i)\prod_{t\in\Gamma_i\setminus s}\mathcal{F}(m^v_{ti}(x_t/a_{it}))\Big)\,, \qquad (13)$$

where $\mathcal{F}$ and $\mathcal{F}^{-1}$ are the discrete Fourier and inverse Fourier transforms, respectively. Again, the products are computed more efficiently using all terms and dividing as in (10).

This highlights the basic advantage over a formulation in which the x_s are explicitly discrete-valued. Although the exact calculations are exponential over the degree of the factor f_i in both cases, the continuous-valued formulation provides us the opportunity to approximate the intermediate quantities (in our implementation, using a discretization) and separates the computation into a simple and efficiently computed form (13). We note that the rescaling step is equivalent to the stretch/unstretch operations proposed in LDLC [27]. Additionally, if the scale factors a_{it} are bipolar (±1), then rescaling becomes trivial (e.g., for -1, the vector is reversed), and the algorithm simplifies further.

We proceed by computing all messages from variables to factors according to (10), then computing all messages from factors to variables (13). The algorithm is run for a predetermined number of iterations, or until convergence is detected locally. To detect convergence, we use the ℓ_2 norm of the product of all incoming messages in the current iteration, relative to the product of all incoming messages in the previous

iteration. Finally, each variable node x_s computes its belief, and we estimate its value by rounding to the closest fault value (either zero or one),

$$x_s = \text{round}\left(\arg\max_{x_s}\left\{g_s(x_s)\prod_{i\in\Gamma_s} m_{is}^f(x_s)\right\}\right). \quad (14)$$

B. Discretization of the Gaussian mixture

Recall that messages are Gaussian mixtures representing the posterior and are discretized for efficiency. To store each message, we allocate a vector of fixed length b. Entries in this vector are real positive values. Typically we use values of b in the range $512-1024$. A higher value of b makes the algorithm more accurate but slows execution time. We evaluate the messages at fixed intervals Δ within the range of interest, identical for each variable. This range should include, for example, both 0 and 1 when the fault pattern is binary and the actual real-valued measurements; because of noise, the range of discretization is further increased to include several standard deviations of uncertainty. We used the following heuristic formula to determine the scope of discretization:

$$R = 1.2\max_i \max\{|y_i|, \sum_s |a_{is}| + 3\sigma\}, \quad (15)$$

and centered around zero, i.e., the range $[-R, R]$. For Bernoulli matrices A, $|a_{is}| = 1$ and when $qn \gg \sigma$, we can further simplify the right-hand term to qn. This range ensures that the range is great enough to include the partial sums of each message, plus some noise; the symmetry condition is useful for computing the scaling operation on negative-valued edges, by first computing the scale then reversing the output around zero.

Note that multiplication of two FFTs followed by an inverse FFT (13) is not equivalent to linear convolution, but rather circular convolution, and therefore zero padding must be carried out. We selected the bins for quantization (15) to be large enough such that no significant probability is near the edges, and so any aliasing is minimal.

C. Local optimization procedures

We outline two useful heuristics proposed by Zymnis et al. [8] that are used for improving the quality of our solutions: variable threshold rounding and a local search procedure. The purpose of these heuristics is to obtain an integer solution out of the fractional solution obtained using our BP solver.

a) Variable threshold rounding: Let $x^\star$ denote a *soft decision*, a vector of the same length as x whose entries are real-valued (rather than binary); $x^\star$ may correspond, for example, to the BP belief or to the optimal point of a convex relaxation of the original problem. Our task is to round the soft decision $x^\star$ to obtain a valid Boolean fault pattern (or *hard decision*). Let $\theta \in (0, 1)$ and set

$$\hat{x} = \lceil (x^\star - \theta) \rceil.$$

To create $\hat{x}$, we simply round all entries of $x^\star$ smaller than the threshold θ to zero. Thus θ is a threshold for guessing that a fault has occurred, based on the relaxed MAP solution $x^\star$. As θ varies from 0 to 1, this method generates up to n different estimates $\hat{x}$, as each entry in $x^\star$ falls below the threshold. We can efficiently find them all by sorting the entries of $x^\star$, and setting the values of each $\hat{x}_s$ to one in the order of increasing $x_s^\star$. We evaluate the loss for each of these patterns (2), and take as our best fault estimate the one that has least loss, which we denote by x^+.

b) Local search: Further improvement of the estimate can be obtained by performing a local search around x^+. Initializing $\hat{x}$ to x^+, we cycle through indices $s = 1, \ldots, n$, where at step s we replace $\hat{x}_s$ with $1 - \hat{x}_s$. When this change decreases the loss function, the change is accepted; we then move on to the next index. We continue in this manner until we have rejected changes in all entries of $\hat{x}$. At the end of this search, $\hat{x}$ is at least 1-OPT, which means that no change in any one entry will improve the loss function.

V. Numerical Results

A. Algorithms compared

We have implemented our NBP solver using Matlab; our implementation is available online [30]. Table I lists the different algorithms we evaluated. We compared NBP to several groups of competing state-of-art algorithms. First, we considered the interior point method (IP) for solving the fault identification problem [8]. Second, we evaluated two other variants of NBP: (*i*) CSBP [20]; and (*ii*) the low density lattice decoder (LDLC) [27]. Third, we ran several non-Bayesian CS algorithms: (*i*) CoSaMP [21]; (*ii*) GPSR [22]; and (*iii*) iterative hard thresholding (HardIO) [23]. Fourth, we implemented a semidefinite programming relaxation [51, 52]. Finally, for a MAP problem over binary variables, it is natural to employ the discrete BP max-product algorithm defined directly over the model (3), and so we implemented this algorithm as well. In practice, max-product BP performed significantly worse than other algorithms. We also evaluated a factor graph representation with binary-valued variables, implemented in C++ using the libDAI toolbox [53]. For small values of q the factor graph BP was as effective as NBP, but as discussed in Section III it requires time exponential in the size of the largest factor; for $q = 0.15$ it required on average 3 minutes per execution, compared to 1.6 seconds for NBP.

A technical subtlety that arises when handling the various algorithms is that they use one of two equivalent formulations of the problem: either a Boolean or bipolar representation. Table II outlines the two models and the transformation needed to shift between them; we use the notation $\mathbf{1}$ for the all-ones vector of appropriate size. Without loss of generality we use the binary form when describing the algorithms.

Within the BP group, max-product BP fails to exploit the sparsity of the measurement matrix, and CSBP assumes a sparse (not necessarily binary) signal. Our NBP uses the same framework as CSBP, but with the correct fault prior. As we show shortly, using the correct prior results in a significant improvement in identifying the correct fault pattern. LDLC assumes a binary prior, but assigns faults and non-faults equal probability, which degrades performance.

Group	Algorithm	Abbreviation	Prior on x
BP	**NBP Solver** (current work)	NBP	binary and sparse
	Max-product BP (current work)	Max-prod	binary and sparse
	Compressed sensing Belief Propagation [20]	CSBP	sparse
	Low density lattice decoder [27]	LDLC	binary
LP	Interior point (Newton method) [8]	IP	$x \in [0,1]$
	Semidefinite programming[51, 52]	SDP	$x \in [0,1]$
CS	Iterative signal recovery [21]	CoSaMP	sparse
	Gradient Projection for Sparse Reconstruction [22]	GPSR	sparse
	Iterative hard thresholding [23]	hardIO	sparse
Other	All zero hypothesis	Null	x is constant

TABLE I: Algorithms used for comparison, grouped into general categories: belief propagation methods (BP), linear programming (LP), and compressed sensing (CS).

Bipolar	Binary	Transformation
$x \in \{-1,1\}^n$	$\overline{x} \in \{0,1\}^n$	$\overline{x} = (x+1)/2$
$y = Ax + v$	$\overline{y} = (2A)\overline{x} + v$	$\overline{y} = y + A\mathbf{1}$
$\min_x \|Ax - y\|$	$\min_{\overline{x}} \|(2A)\overline{x} - \overline{y}\|$	
s.t. $x \in \{-1,1\}^n$	s.t. $\overline{x} \in \{0,1\}^n$	

TABLE II: Transformation between bipolar and binary representations.

Within the linear programming group, linear programming and semidefinite programming relax the binary fault prior into the continuous domain, returning fractional results, and sparsity of the fault pattern is not assumed or exploited. Within the CS group, the binary nature of the signal is not exploited.

B. Experimental settings

We consider an example with $m = 50$ sensors, $n = 100$ possible faults, and linear measurements. For simplicity, we assume that all faults are equiprobable, i.e., $p_s = p$ for all s. The matrix A is taken to be sparse with a fixed percentage q of non-zero entries, whose values are drawn from a bipolar Bernoulli distribution (non-zero elements of A are chosen randomly and independently to be either $+1$ or -1). Bipolar matrices often occur, for example, in fault identification of linear analog circuits [54, 55] (Section II-B). We fix the measurement noise standard deviation to $\sigma = 1$.

We use word error rate (WER) to measure the fraction of problems on which we recover the correct fault pattern exactly. We also compare the methods using the standard precision/recall measure, where precision is the number of true positives (correctly identified faults) divided by the total number of identified faults (including the false positives), and recall is the number of true positives divided by the total number of true positive and false negatives.

C. Discussion of results

We show two sets of results, varying the sparsity p of the fault patterns (rarity of faults) in Fig. 2 and the sparsity q of fault signatures (the matrix A) in Fig. 3. In both cases, we compare all ten algorithms listed in Table I with and without local optimization heuristics. At each sparsity level, the plots show performance averaged over 1000 experiments; vertical error bars indicate the corresponding confidence intervals. Unless otherwise stated, we use default parameters for fault sparsity $p = 0.12$ and fault signature sparsity $q = 0.2$.

We evaluate the effect of fault pattern sparsity on solution quality. Fig. 2 shows the performance of each algorithm, where the sparsity level q of A is fixed to be 0.2, and the fault prior p varies between 3% and 15%. When p increases, the problem becomes harder, because there are an increasing number of *a priori* likely fault combinations. For example, with a prior of $1/n$ we have on average only one fault, with n possible locations; when the prior is $2/n$ there are on average two faults and $n(n-1)/2$ possible locations, and so on. Increasing the prior fault probability shows a clear separation between the performance of the different algorithms, in which NBP outperforms IP in all cases (both with and without local optimization).

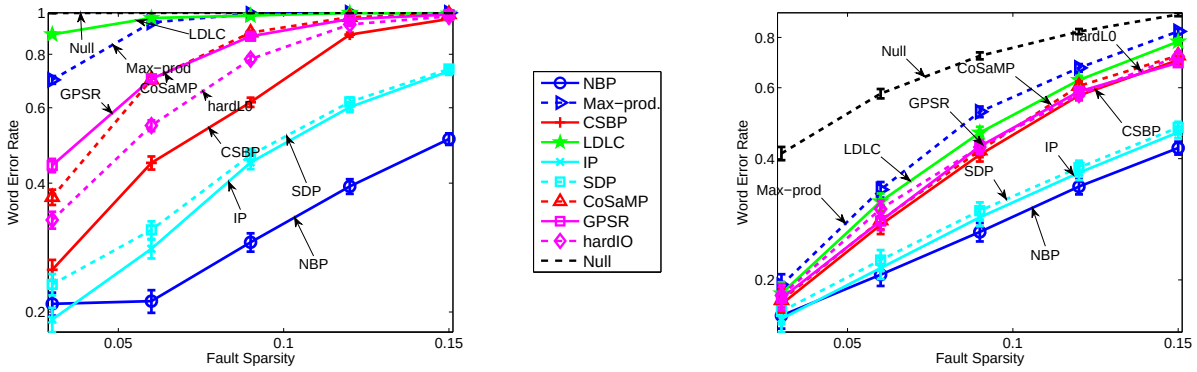
Fig. 3 corresponds to fault probability 0.12 across a range of fault signature (the matrix A) sparsities. It is clear that when the fault signature matrix is less sparse, the problem becomes easier, and the performance of all the algorithms improves. Again, NBP has the lowest WER in all tested scenarios (without local optimization). When using local optimization, IP performs better on dense matrices (when 45% of the matrix entries are non-zeros).

We note the effect of the local optimization heuristics on the quality of the solutions. In both Figs. 2,3 the optimization heuristics have a positive effect, and this effect is particularly powerful when the probability of a fault is low.

Regarding running time, shown in Fig. 4(a),(b), NBP is comparable to other BP algorithms and requires a few seconds. However, linear programming algorithms such as IP and SDP are an order of magnitude faster. This suggests that when accuracy is the priority, NBP is better, but when a fast approximation is sufficient, IP should be used instead. A similar conclusion was offered in [20].

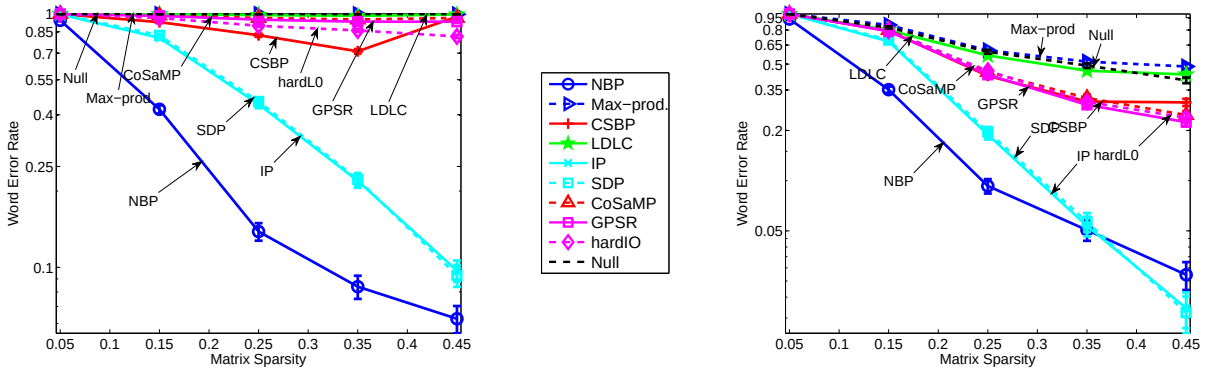
Fig. 4(c) shows a standard precision/recall curve, averaged over 1000 experiments. NBP clearly dominates other algorithms in all ranges. The closest performance to NBP is obtained using both IP and SDP.

Next we evaluate how changes in the algorithm setup affect performance. Fig. 5(a),(b) investigates how the number of quantization points affect both performance and accuracy of the solution. There is clearly a tradeoff between quantization level and efficiency. When fewer quantization points are used, the algorithm runs faster but with reduced accuracy. Fig. 5(c)



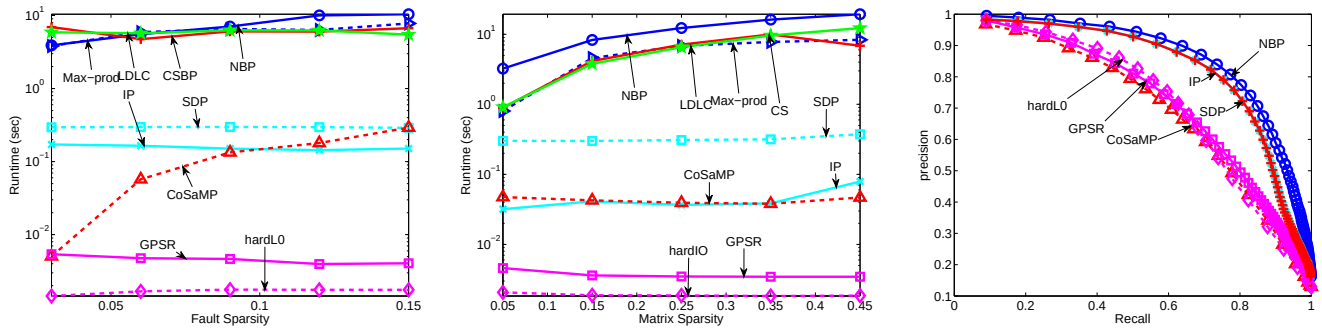
(a) WER without local optimization (b) WER with local optimization

Fig. 2: The effect of changing the probability p of faults on the reconstruction success of the different methods.



(a) WER without local optimization (b) WER with local optimization

Fig. 3: The effect of changing the fault signature matrix sparsity q on the reconstruction success of the different methods.



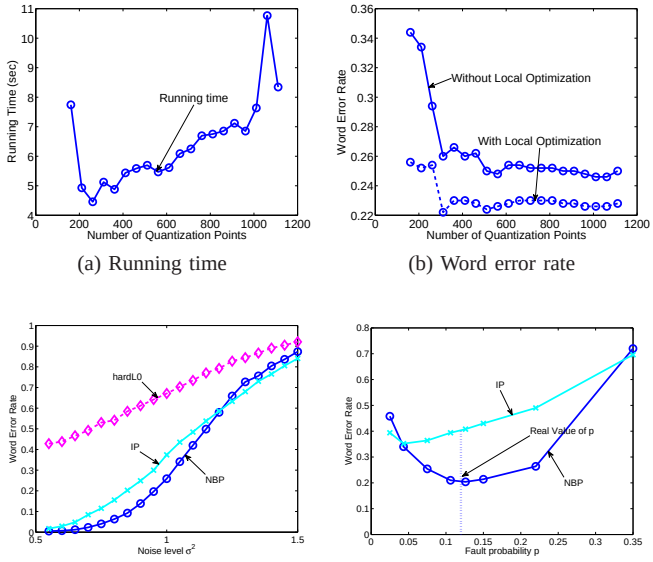
(a) Running time vs. fault sparsity (b) Running time vs. matrix sparsity (c) Precision/Recall curve

Fig. 4: (a),(b) Running times of the different algorithms across experimental conditions; (c) precision/recall curve for each method, at $p = 0.12$ and $q = 0.2$, as the fault decision threshold is varied.

investigates the effect of the noise level σ^2 on performance. As expected, as the noise increases the problem becomes more difficult and the error in identifying the correct fault patterns is larger.

Another issue of interest is that NBP requires us to know the statistics of the fault patterns, and the performance of NBP may deteriorate if the estimate of the fault sparsity p is wrong. To demonstrate that the deterioration is not severe, Fig 5(d) shows the effect of an incorrect estimate of p on the algorithms' performance. The true fault sparsity was taken to be $p = 0.12$, but a false value of p is reported to the algorithm (x-axis). NBP remained more accurate than the other methods even with moderate model mismatch.

Finally, we illustrate the convergence of NBP, CSBP, LDLC, and IP in Fig. 6, where $m = 10, n = 15$, and two faults occurred. Each axis indicates the belief or soft decision around one of the two (correct) faults. Note that all algorithms converge to solutions greater than 0.5, and will thus round to



(a) Running time (b) Word error rate

(c) Noise WER (d) Robust WER

Fig. 5: Top row: The effect of quantization on the (a) running time of NBP, and (b) accuracy of NBP. Increasing quantization slows NBP, but below a certain point can significantly degrade performance. Bottom row: the effect of (c) changing observation noise variance σ^2 on WER, and (d) using the wrong fault prior p on WER (the true fault probability was $p = 0.12$).

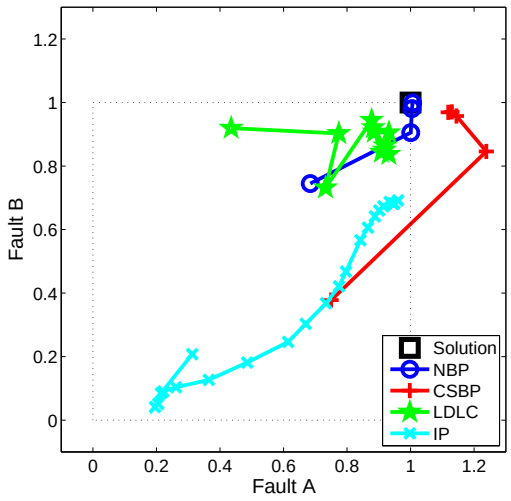


Fig. 6: Convergence of the soft decision values for two faulty bits using different algorithms.

the correct solution in post-processing. The NBP algorithm converges in two iterations to the correct solution while IP requires more iterations for converging to an approximate solution, although we should be cautious in giving this conclusion too much weight, since the computational cost per iteration of the NBP algorithm is higher. NBP converges accurately, because the prior distribution encourages it to converge to zero or one. CSBP converges to some positive but non-integral value, while LDLC also encourages binary values and is therefore the closest after NBP.

Overall, NBP has better performance for fault identification in most of the cases tested. The best competitor is IP, which performed better when the fault signature matrix was dense and local optimization heuristics were applied. The main benefits of NBP are a) it is a distributed algorithm that can scale to large problems (see for example [27]), whereas other algorithms are centralized; b) it performs better when the fault signature matrix is sparse or the fault vector is dense; and c) local optimization heuristics, which require more computation, are less important than for other algorithms. Finally, IP is subject to numerical errors (see discussion of numerical errors in [56]), which we encountered when running on larger problem sizes (e.g., $m = 400, n = 200$). In contrast, we did not encounter numerical errors when using NBP.

The drawbacks of NBP are a) it is slower than IP; and b) it has several configurable parameters that need to be fine-tuned, including the quantization level, quantization bounds, and the variance of the local potential approximation, (9).

VI. Information Theoretic Characterization

Why does NBP perform so well relative to state-of-the-art algorithms? A partial answer can be obtained by considering recent information theoretic results in the related domains of multi-user detection [13, 14] and CS [24–26]. Consider the signal and measurement models as before, where $y = Ax + v$, x_s is i.i.d. Bernoulli with $\Pr(x_s = 1) = p$, and v is i.i.d. zero mean unit norm Gaussian. Suppose also that the measurement matrix A is assumed to be i.i.d. where the probability of non-zero entries is q, and that non-zero matrix entries follow some unit norm distribution. We note in passing that the non-zero probability q must scale to zero as m and n grow; see details in Guo and Wang [13].

Following the conventions of Guo et al. [13, 24, 57], the signal to noise ratio (SNR) γ can be computed as,

$$\gamma = mq.$$

We also define a distortion metric D that evaluates the approximation error between x and $\widehat{x}$ by averaging over per-element distortions,

$$D(x, \widehat{x}) = \frac{1}{n} \sum_{s=1}^{n} d(x_s, \widehat{x}_s),$$

where $d(\cdot, \cdot)$ is a distortion metric such as square error or Hamming distortion. For this problem where a sparse measurement matrix A is used, Guo and Wang [13] provided the fundamental information theoretical characterization of the optimal performance that can be achieved, namely the minimal D that can be achieved as m and n scale to infinity with fixed aspect ratio δ, i.e., $\lim_{n\to\infty} \frac{m}{n} = \delta$.

There are several noteworthy aspects in the analysis by Guo and Wang. First, Theorem 1 [13] proves that the problem behaves as if each individual input element x_s were estimated individually from a corrupted version z_s, with $z_s = x_s + w_s$ where w_s is Gaussian noise. That is, the vector estimation problem is related to an estimation problem defined over scalar

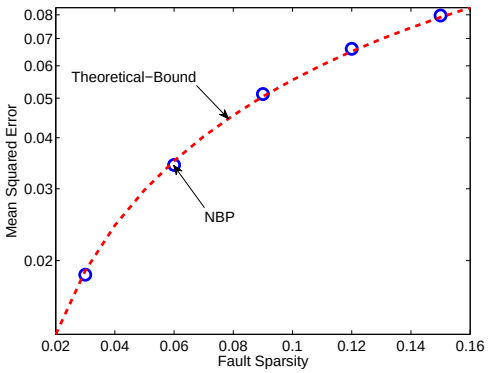


Fig. 7: Mean squared error results for NBP as a function of fault sparsity, compared to the theoretical lower bound on MSE performance. NBP provides nearly optimal estimates even on modestly sized problems.

Gaussian channels. Second, Theorem 2 [13] demonstrates that the amount of scalar Gaussian noise in the random variable W_s can be computed by finding the fixed point for η of the following expression:

$$\eta^{-1} = 1 + \frac{\gamma}{\delta}\text{mmse}(p, \eta\gamma), \tag{16}$$

where $\eta \in (0, 1)$ is the degradation in SNR, i.e., the SNR of the scalar channel is $\eta\gamma$ instead of γ, and $\text{mmse}(p, \eta\gamma)$ is the minimum mean square error (MMSE) for estimating the random variable $X_s \sim \text{Ber}(p)$ from the output Z_s of the dual scalar channel whose SNR is $\eta\gamma$. That is, the scalar Gaussian channels are degraded. Combining these insights, the fundamental limiting performance can be computed for any distortion metric $d(\cdot, \cdot)$ by examining the output of the degraded scalar channel with SNR $\eta\gamma$. Finally, Guo and Wang demonstrate that in the large system limit (when m and n scale to infinity) with fixed aspect ratio δ, the posteriors estimated by BP converge in distribution to the true posteriors. Consequently, it should be no surprise that numerical results in Section V indicate that NBP outperforms other techniques for fault identification in practice.

To verify that our NBP algorithm indeed offers MSE performance that is comparable to the information theoretic lower bound (16), we simulated a setting similar to that of Section V where $m = 250$, $n = 500$, $q = 0.1$, $\sigma = 4$, and varying p from 0.03 to 0.15. Figure 7 illustrates that the mean square error achieved by NBP is almost indistinguishable from the MMSE bound (16).

Although the fundamental performance of fault identification is well understood when A is sparse and i.i.d., when A is not sparse the analysis becomes more involved. In the non-sparse case, replica method analyses have been used to derive information theoretic characterizations in a less rigorous fashion; see Guo and Verdú [14] or Guo and Tanaka [57].

VII. Discussion

In this paper, we propose a novel approach for solving the fault identification problem using non-parametric belief propagation. By exploiting prior information about faults, we are able to improve significantly our ability to correctly identify fault patterns. Our main improvement arises from accounting for both the binary nature and prior probability of the faults.

Our approach elects to quantize a continuous relaxation of the true discrete model before solving it using belief propagation (BP). Although this step introduces additional approximations, it enables us to exploit the fact that each likelihood function f_i is a function of a linear combination of variables x_s. This makes the algorithm efficient even when q is not small, so that the f_i are defined on a large number of variables.

A possible extension is the inclusion of non-i.i.d. noise. In principle, non-i.i.d. noise can be dealt with by augmenting the model in Fig. 1(a) with a graph structure over measurements y that reflects the inverse covariance matrix of the noise. We expect our approach to continue to do well for relatively sparse inverse covariance matrices; an exact characterization is left for future study.

Another extension would be to consider non-binary faults. As long as the fault pattern is i.i.d., the recent information theoretic results of Guo et al. [13, 24, 57] indicate that BP should continue to perform well.

As a final note, we mention that our proposed algorithm is distributed, since the underlying BP algorithm can be distributed over multiple nodes, and works well when the matrix A is sparse. In a network where communication is costly, our algorithm offers the additional advantage of requiring less communication.

Acknowledgements

Danny Bickson was partially supported by grants ARO MURI W911NF0710287, ARO MURI W911NF0810242, NSF Mundo IIS-0803333 and NSF Nets-NBD CNS-0721591. Parts of this work were performed while Danny Bickson was a research staff member at IBM Haifa Labs. Dror Baron thanks the Department of Electrical Engineering at the Technion for generous hospitality while parts of the work were performed, and in particular the support of Tsachy Weissman. Danny Dolev is Incumbent of the Berthold Badler Chair in Computer Science, and was supported in part by the Israel Science Foundation (ISF) Grant 0397373.

References

[1] D. Viassolo, S. Adibhatla, B. Brunell, J. Down, N. Gibson, A. Kumar, H. Mathews, and L. Holcomb, "Advanced estimation for aircraft engines," in *Amer. Control Conf.*, 2007, pp. 2807–2821.

[2] S. Ganguli, S. Deo, and D. Gorinevsky, "Parametric fault modeling and diagnostics of a turbofan engine," in *Int. Conf. Control Appl.*, 2004, pp. 223–228.

[3] J. Hoskins, K. Kaliyur, and D. Himmelblau, "Fault diagnosis in complex chemical plants using artificial neural networks," *AIChE Journal*, vol. 37, no. 1, pp. 137–141, Jan. 1991.

[4] J. Gertler, M. Costin, X. Fang, R. Hira, Z. Kowalczuk, and Q. Luo, "Model-based on-board fault detection and diagnosis for automotive engines," *Control Engineering Practice*, vol. 1, no. 1, pp. 3–17, Jan. 1993.

[5] P. Frank, "Fault diagnosis in dynamic systems using analytical and knowledge-based redundancy — A survey and some new results," *Automatica*, vol. 26, no. 3, pp. 459–474, May 1990.

[6] R. Reiter, "A theory of diagnosis from first principles," *Artificial Intelligence*, vol. 32, no. 1, pp. 57–95, Apr. 1987.

[7] J. de Kleer and B. Williams, "Diagnosing multiple faults," *Artificial Intelligence*, vol. 32, no. 1, pp. 97–130, Apr. 1987.

[8] A. Zymnis, S. Boyd, and D. Gorinevsky, "Relaxed maximum a posteriori fault identification," *Signal Processing*, vol. 89, no. 6, pp. 989–999, June 2009.

[9] C. Hood and C. Ji, "Proactive network-fault detection," *IEEE Trans. Rel.*, vol. 46, no. 3, pp. 333–341, Sept. 1997.

[10] F. Feather, D. Siewiorek, and R. Maxion, "Fault detection in an ethernet network using anomaly signature matching," in *ACM SIGCOMM*, 1993, pp. 279–288.

[11] P. Stelling, C. DeMatteis, I. Foster, C. Kesselman, C. Lee, and G. von Laszewski, "A fault detection service for wide area distributed computations," *Cluster Computing*, vol. 2, no. 2, pp. 117–128, 1999.

[12] J. Bandler and A. Salama, "Fault diagnosis of analog circuits," *Proc. IEEE*, vol. 73, no. 8, pp. 1279–1325, 1985.

[13] G. Guo and C.-C. Wang, "Multiuser detection of sparsely spread CDMA," *IEEE J. Sel. Areas Commun.*, vol. 26, no. 3, pp. 421–431, 2008.

[14] D. Guo and S. Verdú, "Randomly Spread CDMA: Asymptotics Via Statistical Physics," *IEEE Trans. Inf. Theory*, vol. 51, no. 6, p. 1983, 2005.

[15] A. Montanari, B. Prabhakar, and D. Tse, "Belief propagation based multi-user detection," in *Allerton Conf. Comm. Control Comput.*, Sept. 2005.

[16] D. Bickson, O. Shental, P. H. Siegel, J. K. Wolf, and D. Dolev, "Gaussian belief propagation based multiuser detection," in *Int. Symp. Info. Theory*, July 2008, pp. 1878 – 1882.

[17] ——, "Linear detection via belief propagation," in *Allerton Conf. Comm. Control Comput.*, Sept. 2007, pp. 1207–1213.

[18] D. Donoho, "Compressed sensing," in *IEEE Trans. Inf. Theory*, vol. 52, no. 4, Apr. 2006, pp. 1289–1306.

[19] E. J. Candes, J. Romberg, and T. Tao, "Robust uncertainty principles: exact signal reconstruction from highly incomplete frequency information," *IEEE Trans. Inf. Theory*, vol. 52, no. 2, pp. 489–509, Feb. 2006.

[20] D. Baron, S. Sarvotham, and R. G. Baraniuk, "Bayesian compressive sensing via belief propagation," *IEEE Trans. Signal Process.*, vol. 58, pp. 269–280, Jan. 2010.

[21] D. Needell and J. A. Tropp, "CoSaMP: Iterative signal recovery from incomplete and inaccurate samples," *Appl. Comp. Harmonic Anal.*, vol. 26, pp. 301–321, 2008.

[22] M. Figueiredo, R. Nowak, and S. Wright, "Gradient projection for sparse reconstruction: Application to compressed sensing and other inverse problems," *IEEE J. Sel. Topics Signal Process.*, vol. 1, no. 4, pp. 586–598, 2007.

[23] T. Blumensath and M. E. Davies, "Iterative hard thresholding for compressed sensing," *App. Comput. Harmonic Anal.*, vol. 27, no. 3, pp. 265–274, May 2009.

[24] D. Guo, D. Baron, and S. Shamai, "A single-letter characterization of optimal noisy compressed sensing," in *Allerton Conf. Comm. Control Comput.*, Sep. 2009.

[25] S. Rangan, A. K. Fletcher, and V. K. Goyal, "Asymptotic analysis of MAP estimation via the replica method and applications to compressed sensing," CoRR, Tech. Rep. abs/0906.3234, June 2009.

[26] W. Wang, M. J. Wainwright, and K. Ramchandran, "Information-theoretic limits on sparse signal recovery: dense versus sparse measurement matrices," *IEEE Trans. Inf. Theory*, vol. 56, no. 6, pp. 2967–2979, June 2010.

[27] N. Sommer, M. Feder, and O. Shalvi, "Low-density lattice codes," in *IEEE Trans. Inf. Theory*, vol. 54, no. 4, 2008, pp. 1561–1585.

[28] B. Kurkoski and J. Dauwels, "Message-passing decoding of lattices using Gaussian mixtures," in *Int. Symp. Info. Theory*, July 2008.

[29] D. Bickson, A. Ihler, H. Avissar, and D. Dolev, "A low density lattice decoder via non-parametric belief propagation," in *Allerton Conf. Comm. Control Comput.*, Sept. 2009.

[30] D. Bickson, "Gaussian belief propagation Matlab toolbox," http://www.cs.cmu.edu/~bickson/gabp/.

[31] M. Bushnell and V. Agrawal, *Essentials of Electronic Testing for Digital, Memory, and Mixed-Signal VLSI Circuits*, 2nd ed. Springer, 2000, vol. 17.

[32] A. Viterbi, "Error bounds for convolutional codes and an asymptotically optimum decoding algorithm," *IEEE Trans. Inf. Theory*, vol. 13, no. 2, pp. 260–269, 1967.

[33] E. Sudderth, A. Ihler, W. Freeman, and A. Willsky, "Nonparametric belief propagation," in *Comp. Vision Pattern Recog. (CVPR)*, June 2003.

[34] R. G. Gallager, "Low density parity check codes," *IRE Trans. Inf. Theory*, vol. 8, pp. 21–28, 1962.

[35] M. Wainwright, T. Jaakkola, and A. Willsky, "Tree-reweighted belief propagation and approximate ML estimation by pseudo-moment matching," in *AI and Statistics*, Jan. 2003.

[36] ——, "MAP estimation via agreement on (hyper)trees: Message-passing and linear-programming approaches," in *IEEE Trans. Inf. Theory*, vol. 51, no. 11, Nov. 2005, pp. 3697–3717.

[37] ——, "A new class of upper bounds on the log partition function," *IEEE Trans. Inf. Theory*, vol. 51, no. 7, pp. 2313–2335, July 2005.

[38] C. Yanover, T. Meltzer, and Y. Weiss, "Linear programming relaxations and belief propagation – an empirical study," *J. Machine Learning Research*, vol. 7, pp. 1887–1907, 2006.

[39] S. Joshi and S. Boyd, "Sensor selection via convex optimization," *IEEE Trans. Signal Process.*, vol. 57, no. 2, pp. 451–462, February 2009.

[40] J. M. Walsh and P. A. Regalia, "On the relationship between belief propagation decoding and joint maximum likelihood detection," *IEEE Trans. Commun.*, vol. 58, pp. 2753–2758, Oct. 2010.

[41] A. Ihler, "Accuracy bounds for belief propagation," in *Uncertainty Artif. Intell. (UAI)*, July 2007.

[42] A. Ihler, J. Fisher, R. Moses, and A. Willsky, "Nonparametric belief propagation for self-localization of sensor networks," *IEEE J. Sel. Areas Commun.*, vol. 23, no. 4, pp. 809–819, 2005.

[43] A. Ihler, E. Sudderth, W. Freeman, and A. Willsky, "Efficient multiscale sampling from products of Gaussian mixtures," in *Neural Info. Proc. Systems (NIPS)*, Dec. 2003.

[44] M. Briers, A. Doucet, and S. S. Singh, "Sequential auxiliary particle belief propagation," in *Int. Conf. Information Fusion*, 2005, pp. 705–711.

[45] D. Rudoy and P. J. Wolf, "Multi-scale MCMC methods for sampling from products of Gaussian mixtures," in *ICASSP*, vol. 3, 2007, pp. III–1201–III–1204.

[46] J. Coughlan and H. Shen, "Dynamic quantization for belief propagation in sparse spaces," *Comput. Vis. Image Underst.*, vol. 106, no. 1, pp. 47–58, 2007.

[47] M. Isard, J. MacCormick, and K. Achan, "Continuously-adaptive discretization for message-passing algorithms," in *Neural Info. Proc. Systems (NIPS)*, 2009, pp. 737–744.

[48] D. Bickson, D. Dolev, and Y. Weiss, "Modified belief propagation for energy saving wireless and sensor networks," School of Computer Science and Engineering, The Hebrew University, Tech. Rep. Leibniz Center TR-2005-85, 2005.

[49] B. Potetz and T. S. Lee, "Efficient belief propagation for higher-order cliques using linear constraint nodes," *Comput. Vis. Image Underst.*, vol. 112, no. 1, pp. 39–54, 2008.

[50] P. Felzenszwalb and D. Huttenlocher, "Efficient belief propagation for early vision," *Int. J. Comp. Vision*, vol. 70, no. 1, Oct. 2006.

[51] P. H. Tan and L. K. Rasmussen, "The application of semidefinite programming for detection in CDMA," *IEEE J. Sel. Areas Commun.*, vol. 19, no. 8, pp. 1442–1449, Aug. 2001.
[52] M. Abdi, H. Nahas, A. Jard, and E. Moulines, "Semidefinite positive relaxation of the maximum-likelihood criterion applied to multiuser detection in a CDMA context," *IEEE Signal Process. Lett.*, vol. 9, no. 6, pp. 165–167, June 2002.
[53] J. M. Mooij, "libDAI: A free and open source C++ library for discrete approximate inference in graphical models," *J. Machine Learning Research*, vol. 11, pp. 2169–2173, Aug. 2010.
[54] J. Hou and A. Chatterjee, "Concurrent transient fault simulation for analog circuits," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 22, no. 10, pp. 1385–1398, September 2003.
[55] A. Johnson, "Efficient fault analysis in linear analog circuits," *IEEE Trans. Circuits Syst.*, vol. 26, no. 7, pp. 475–484, Jan. 2003.
[56] D. Bickson, Y. Tock, A. Zymnis, S. Boyd, and D. Dolev., "Distributed large scale network utility maximization," in *Int. Symp. Info. Theory*, vol. 2, June 2009, pp. 829–833.
[57] D. Guo and T. Tanaka, "Generic multiuser detection and statistical physics," in *Advances in Multiuser Detection*, M. Honig, Ed., 2009, pp. 251–310.