Non-parametric Belief Propagation Applications

Thesis submitted in fulfillment of the requirement for the degree of
Master of Science

by

Harel Avissar

School of Engineering and Computer Science
The Hebrew University of Jerusalem

This work was carried out under the supervision of
Prof. Danny Dolev and Dr. Danny Bickson

September 2009

Acknowledgements

I would Like to thank Prof. Danny Dolev for advising and supervising throughout the work presented and for offering valuable input as well as criticism along the way. Danny Dolev accommodated a very unorthodox schedule for me as an M.Sc student and allowed me to contribute when I could with respect to my special situation.

I would like to thank Dr. Danny Bickson, for his close and personal supervision, for countless tips, ideas and advice and for his help in explaining, re-explaining and simplifying a new and unfamiliar field for me. I was lucky to get to work with Danny Bickson who had great influence both on my productivity and the good choice of right directions to pursue and who managed it all with a friendly and enjoyable guidance.

To the entire DANSS session crowd for allowing me to show this work and for helping to uncover strengths and weaknesses that needed addressing.

Last but not least to my family for supporting, taking interest and at times even proof reading my work as a first line of supporting critics.

Abstract

The canonical problem of solving a system of linear equations arises in numerous contexts in information theory, communication theory, and related fields. Gaussian belief propagation (GaBP) has been shown to solve this problem in a manner that does not involve direct matrix inversion. The iterative nature of GaBP allows for a distributed message-passing implementation of the solution algorithm. Non-parametric Belief Propagation (NBP) is an extension of GaBP that allows the prior distributions to be Gaussian mixtures instead of single Gaussian, which enables approximating many complex distributions. The importance of these prior distribution is in solving systems of linear equations with constraints on the values of x, the prior approximation is a very intuitive yet accurate approximation.

We first address the problem of Low Density Lattice Codes (LDLC) decoding algorithm, with bipolar constraints on x, and equal probability for $x = 1, -1$. In general the LDLC problem has integer contraints on x, and the NBP solver supports this as well. We show that an NBP based solver allows for low Symbol Error Rate (SER), especially when working close to channel capacity, compared to present solvers. We also give some general convergence conditions borrowed from the world of GaBP, and while no proof of convergence has been found to date we report preliminary results which show good experimental convergence.

The Fault Detection problem further expands our formulation to allow for different probability of $x = 1, x = -1$, resulting in a closely related problem. We again show the NBP based solver has better results than current state of the art algorithms from several related fields. We also compare a solution via NBP based on 3 different prior distributions, and show that best results are achieved when we incorporate our full knowledge of the constraints on the solution into the prior distribution.

Contents

Chapter 1

Introduction

1.1 Introduction

Solving a linear system of equations $\mathbf{Ax} = \mathbf{b}$ is one of the most fundamental problems in algebra, with countless applications in the mathematical sciences and engineering as well as social sciences. Non-parametric Belief Propagation (NBP) is an iterative message passing algorithm that we propose to link to the solution of this problem under constraints on x, without direct matrix inversion. In this thesis, we show how NBP can be used with two different prior distributions to solve two different problems with similar formulation and establish that the more prior knowledge is incorporated into the NBP prior the better it will approximate the true x solution.

1.2 Material Covered in this Thesis

This thesis is divided into two parts. The first part discusses the solution of the LDLC decoding problem via NBP covers the following paper: [1]. The second part discusses the solution of the Fault Detection problem via NBP, and the importance of the different priors, and was covered in the following paper: [2].

Before we go on with the applications, we will survey some of the related works, focusing on factor graphs and GaBP as the base and NBP as a relaxation of the original GaBP algorithm. We will also survey recent work related to our two problems LDLC and FD, and current solutions to these problem.

1.3 Related work

1.3.1 Gaussian Belief Propagation

Factor graphs

Factor graphs provide a convenient mechanism for representing structure among random variables. Suppose a function or distribution $p(x)$ defined on a large set of variables $x = [x_1, \ldots, x_n]$ factors into a collection of smaller functions $p(x) = \prod_s f_s(x_s)$, where each x_s is a vector composed of a smaller subset of the x_i. We represent this factorization as a bipartite graph with "factor nodes" f_s and "variable nodes" x_i, where the neighbors Γ_s of f_s are the variables in x_s, and the neighbors of x_i are the factor nodes which have x_i as an argument (f_s such that x_i in x_s). For compactness, we use subscripts s, t to indicate factor nodes and i, j to indicate variable nodes, and will use x and x_s to indicate sets of variables, typically formed into a vector whose entries are the variables x_i which are in the set.

The belief propagation (BP) or sum-product algorithm [3] is a popular technique for estimating the marginal probabilities of each of the variables x_i. BP follows a message-passing formulation, in which at each iteration τ, every variable passes a message (denoted M_{is}^{τ}) to its neighboring factors, and factors to their neighboring variables. These messages are given by the general form,

$$M_{is}^{\tau+1}(x_i) = f_i(x_i) \prod_{t \in \Gamma_i \backslash s} M_{ti}^{\tau}(x_i)\,, \qquad M_{si}^{\tau+1}(x_i) = \int_{x_s \backslash x_i} f_s(x_s) \prod_{j \in \Gamma_s \backslash i} M_{js}^{\tau}(x_j)\,. \tag{1.1}$$

Here we have included a "local factor" $f_i(x_i)$ for each variable, to better parallel our development in the sequel. When the variables x_i take on only a finite number of values, the messages may be represented as vectors; the resulting algorithm has proven effective in many coding applications including low-density parity check (LDPC) codes [4]. In keeping with our focus on continuous-alphabet codes, however, we will focus on implementations for continuous-valued random variables.

Gaussian Belief Propagation

When the joint distribution $p(x)$ is Gaussian, $p(x) \propto \exp\{-\frac{1}{2}x^T J x + h^T x\}$, the BP messages may also be compactly represented in the same form. Here we use the "information form" of the Gaussian distribution, $\mathcal{N}(x; \mu, \Sigma) = \mathcal{N}^{-1}(h, J)$ where $J = \Sigma^{-1}$ and $h = J\mu$. In this case, the distribution's factors can always be written in a pairwise form, so that each function involves at most two variables x_i, x_j, with $f_{ij}(x_i, x_j) = \exp\{-J_{ij}x_i x_j\}$, $j \neq i$, and $f_i(x_i) = \exp\{-\frac{1}{2}J_{ii}x_i^2 + h_i x_i\}$.

Gaussian BP (GaBP) then has messages that are also conveniently represented as information-form Gaussian distributions. If s refers to factor f_{ij}, we have

$$M_{is}^{\tau+1}(x_i) = \mathcal{N}^{-1}(\beta_{i\backslash j}, \alpha_{i\backslash j})\,, \qquad \alpha_{i\backslash j} = J_{ii} + \sum_{k \in \Gamma_i \backslash j} \alpha_{ki}\,, \qquad \beta_{i\backslash j} = h_i + \sum_{k \in \Gamma_i \backslash j} \beta_{ki}\,, \tag{1.2}$$

$$M_{sj}^{\tau+1}(x_j) = \mathcal{N}^{-1}(\beta_{ij}, \alpha_{ij})\,, \qquad \alpha_{ij} = -J_{ij}^2 \alpha_{i\backslash j}^{-1}\,, \qquad \beta_{ij} = -J_{ij}\alpha_{i\backslash j}^{-1}\beta_{i\backslash j}\,. \tag{1.3}$$

From the α and β values we can compute the estimated marginal distributions, which are Gaussian with mean $\hat{\mu}_i = \hat{K}_i(h_i + \sum_{k\in\Gamma_i} \beta_{ki})$ and variance $\hat{K}_i = (J_{ii} + \sum_{k\in\Gamma_i} \alpha_{ki})^{-1}$. It is known that if GaBP converges, it results in the exact MAP estimate x^*, although the variance estimates $\hat{K}_i$ computed by GaBP are only approximations to the correct variances [5].

The GaBP algorithm is developed in full, thoroughly discussed and convergence is proven in the following papers: [6, 7, 8, 9, 10, 11].

Nonparametric belief propagation

In more general continuous-valued systems, the messages do not have a simple closed form and must be approximated. Nonparametric belief propagation, or NBP, extends the popular class of particle filtering algorithms, which assume variables are related by a Markov chain, to general graphs. In NBP, messages are represented by collections of weighted samples, smoothed by a Gaussian shape – in other words, Gaussian mixtures.

NBP follows the same message update structure of (1.1). Notably, when the factors are all either Gaussian or mixtures of Gaussians, the messages will remain mixtures of Gaussians as well, since the product or marginalization of any mixture of Gaussians is also a mixture of Gaussians [12]. However, the product of d Gaussian mixtures, each with N components, produces a mixture of N^d components; thus every message product creates an exponential increase in the size of the mixture. For this reason, one must approximate the mixture in some way. NBP typically relies on a stochastic sampling process to preserve only high-likelihood components, and a number of sampling algorithms have been designed to ensure that this process is as efficient as possible [13, 14, 15]. One may also apply various deterministic algorithms to reduce the number of Gaussian mixture components [16]; for example, in [17, 18], an $O(N)$ greedy algorithm (where N is the number of components before reduction) is used to trade off representation size with approximation error under various measures.

1.3.2 Low Density Lattice Codes

Lattices and low-density lattice codes

An n-dimensional lattice Λ is defined by a generator matrix G of size $n \times n$. The lattice consists of the discrete set of points $x = (x_1, x_2, ..., x_n) \in R^n$ with $x = Gb$, where $b \in Z^n$ is the set of all possible integer vectors.

A low-density lattice code (LDLC) is a lattice with a non-singular generator matrix G, for which $H = G^{-1}$ is sparse. It is convenient to assume that $det(H) = 1/det(G) = 1$. An (n, d) regular LDLC code has an H matrix with constant row and column degree d. In a latin square LDLC, the values of the d non-zero coefficients in each row and each column are some permutation of the values $h_1, h_2, \cdots, h_d$.

We assume a linear channel with additive white Gaussian noise (AWGN). For a vector of integer-valued information b the transmitted codeword is $x = Gb$, where G is the LDLC encoding matrix, and the received observation is $y = x + w$ where w is a vector of i.i.d. AWGN with

diagonal covariance $\sigma^2 I$. The decoding problem is then to estimate b given the observation vector y; for the AWGN channel, the MMSE estimator is

$$b^* = \arg\min_{b \in \mathbb{Z}^n} ||y - Gb||^2 . \tag{1.4}$$

Lattice codes provide a continuous-alphabet encoding procedure, in which integer-valued information bits are converted to positions in Euclidean space. Motivated by the success of low-density parity check (LDPC) codes [19], recent work by Sommer *et al.* [20] presented low density lattice codes (LDLC). Like LDPC codes, a LDLC code has a sparse decoding matrix which can be decoded efficiently using an iterative message-passing algorithm defined over a factor graph. In the original paper, the lattice codes were limited to Latin squares, and some theoretical results were proven for this special case.

LDLC decoder

The LDLC decoding algorithm is also described as a message-passing algorithm defined on a factor graph [3], whose factors represent the information and constraints on x arising from our knowledge of y and the fact that b is integer-valued. Here, we rewrite the LDLC decoder update rules in the more standard graphical models notation. The factor graph used is a bipartite graph with variables nodes $\{x_i\}$, representing each element of the vector x, and factor nodes $\{f_i, g_s\}$ corresponding to functions

$$f_i(x_i) = \mathcal{N}(x_i; y_i, \sigma^2) , \qquad g_s(x_s) = \begin{cases} 1 & H_s x \in \mathbb{Z} \\ 0 & \text{otherwise} \end{cases} ,$$

where H_s is the s^{th} row of the decoding matrix H. Each variable node x_i is connected to those factors for which it is an argument; since H is sparse, H_s has few non-zero entries, making the resulting factor graph sparse as well. Notice that unlike the construction of [20], this formulation does not require that H be square, and it may have arbitrary entries, rather than being restricted to a Latin square construction. Sparsity is preferred, both for computational efficiency and because belief propagation is typically more well behaved on sparse systems with sufficiently long cycles [3]. We can now directly derive the belief propagation update equations as Gaussian mixture distributions, corresponding to an instance of the NBP algorithm. We suppress the iteration number τ to reduce clutter.

Variable to factor messages. Suppose that our factor to variable messages $M_{si}(x_i)$ are each described by a Gaussian mixture distribution, which we will write in both the moment and information form:

$$M_{si}(x_i) = \sum_l w_{si}^l \mathcal{N}(x_i \,;\, m_{si}^l, \nu_{si}^l) = \sum_l w_{si}^l \mathcal{N}^{-1}(x_i \,;\, \beta_{si}^l, \alpha_{si}^l) . \tag{1.5}$$

Then, the variable to factor message $M_{is}(x_s)$ is given by

$$M_{is}(x_s) = \sum_l w_{is}^l \mathcal{N}(x_s \,;\, m_{is}^l, \nu_{is}^l) = \sum_l w_{is}^{\mathbf{1}} \mathcal{N}^{-1}(x_s \,;\, \beta_{is}^l, \alpha_{is}^l) , \tag{1.6}$$

where $\mathbf{l}$ refers to a vector of indices $[l_s]$ for each neighbor s,

$$\alpha_{is}^{\mathbf{l}} = \sigma^{-2} + \sum_{t \in \Gamma_i \setminus s} \alpha_{ti}^{l_t}, \quad \beta_{it}^{\mathbf{l}} = y_i \sigma^{-2} + \sum_{t \in \Gamma_i \setminus s} \beta_{ti}^{l_s}, \quad w_{it}^{\mathbf{l}} = \frac{\mathcal{N}(x^*; y_i, \sigma^2) \prod w_{si}^{l_s} \mathcal{N}^{-1}(x^*; \beta_{si}^{l_s}, \alpha_{si}^{l_s})}{\mathcal{N}^{-1}(x^*; \beta_{it}^{\mathbf{l}}, \alpha_{it}^{\mathbf{l}})}. \tag{1.7}$$

The moment parameters are then given by $\nu_{it}^{\mathbf{l}} = (\alpha_{it}^{\mathbf{l}})^{-1}$, $m_{it}^{\mathbf{l}} = \beta_{it}^{\mathbf{l}} (\alpha_{it}^{\mathbf{l}})^{-1}$. The value x^* is any arbitrarily chosen point, often taken to be the mean $m_{it}^{\mathbf{l}}$ for numerical reasons.

Factor to variable messages. Assume that the incoming messages are of the form (1.6), and note that the factor $g_s(\cdot)$ can be rewritten in a summation form, $g_s(x_s) = \sum_{b_s} \delta(H_s x = b_s)$, which includes all possible integer values b_s. If we condition on the value of both the integer b_s and the indices of the incoming messages, again formed into a vector $\mathbf{l} = [l_j]$ with an element for each variable j, we can see that g_s enforces the linear equality $H_{si} x_i = b_s - \sum H_{sj} x_j$. Using standard Gaussian identities in the moment parameterization and summing over all possible $b_s \in \mathbb{Z}$ and $\mathbf{l}$, we obtain

$$M_{si}(x_i) = \sum_{b_s} \sum_{\mathbf{l}} w_{si}^{\mathbf{l}} \mathcal{N}(x_i \,;\, m_{si}^{\mathbf{l}}, \nu_{si}^{\mathbf{l}}) = \sum_{b_s} \sum_{\mathbf{l}} w_{si}^{\mathbf{l}} \mathcal{N}^{-1}(x_i \,;\, \beta_{si}^{\mathbf{l}}, \alpha_{si}^{\mathbf{l}}), \tag{1.8}$$

where

$$\nu_{si}^{\mathbf{l}} = H_{si}^{-2} \left(\sum_{j \in \Gamma_s \setminus i} H_{js}^2 \nu_{js}^{l_j} \right), \quad m_{si}^{\mathbf{l}} = H_{si}^{-1} \left(-b_s + \sum_{j \in \Gamma_s \setminus i} H_{js} m_{js}^{l_j} \right), \quad w_{si}^{\mathbf{l}} = \prod_{j \in \Gamma_s \setminus i} w_{js}^{l_j}, \tag{1.9}$$

and the information parameters are given by $\alpha_{si}^{\mathbf{l}} = (\nu_{si}^{\mathbf{l}})^{-1}$ and $\beta_{si}^{\mathbf{l}} = m_{si}^{\mathbf{l}} (\nu_{si}^{\mathbf{l}})^{-1}$.

Notice that (1.8) matches the initial assumption of a Gaussian mixture given in (1.5). At each iteration, the exact messages remain mixtures of Gaussians, and the algorithm iteslf corresponds to an instance of NBP. As in any NBP implementation, we also see that the number of components is increasing at each iteration and must eventually approximate the messages using some finite number of components. To date the work on LDLC decoders has focused on deterministic approximations [20, 21, 22, 23], often greedy in nature. However, the existing literature on NBP contains a large number of deterministic and stochastic approximation algorithms [13, 17, 16, 14, 15]. These algorithms can use spatial data structures such as KD-Trees to improve efficiency and avoid the pitfalls that come with greedy optimization.

The vector x_s is defined to include only those entries of x for which H_s is non-zero. The LDLC message update rules are then given as:

Variable nodes Each message has the form of a mixture of n Gaussians,

$$M_{is}^{\tau}(x_i) = \sum_{k=1}^{n} c_k \mathcal{N}(x_i \,;\, m_{is}^{\tau,k}, v_{is}^{\tau,k}),$$

where $c_k > 0$ are the relative weights, with $\sum_k c_k = 1$, and $m_{is}^{\tau,k}$ and $v_{is}^{\tau,k}$ are the mean and variance of the k^{th} mixture component. The variable-to-factor messages are initialized to be Gaussian, with

$$M_{is}^0(x_i) = \mathcal{N}(x_i; y_i, \sigma^2)\,. \tag{1.10}$$

In subsequent iterations, these messages will also be Gaussian mixtures, given by

$$\frac{1}{v_{is}^{\tau,k}} = \frac{1}{\sigma^2} + \sum_{t\in\Gamma_i\setminus s} \frac{1}{v_{ti}^{\tau-1,k}}\,, \qquad \frac{m_{is}^{\tau,k}}{v_{is}^{\tau,k}} = \frac{y_i}{\sigma^2} + \sum_{t\in\Gamma_i\setminus s} \frac{m_{ti}^{\tau-1,k}}{v_{ti}^{\tau-1,k}}\,. \tag{1.11}$$

Factor nodes The factor-to-variable message

$$M_{si}^{\tau}(x_i) = \sum_{b=-\infty}^{\infty} Q(x_i) \tag{1.12}$$

$$Q(x_i) = \sum_k c_k \mathcal{N}(x_i\,;\, \mu_{si}(b), v_{si}^{\tau,k}) \tag{1.13}$$

where

$$v_{si} = H_{si}^2(\sum_{k\in\Gamma_s\setminus i} H_{ks}^2 v_{ks})^{-1}\,. \qquad \mu_{si}(b)v_{si} = H_{si}^{-1}(b - \sum_{k\in\Gamma_s\setminus i} H_{sk}m_{ks})\,, \tag{1.14}$$

Estimating the codewords. The original codeword x can be estimated using its belief, an approximation to its marginal distribution given the constraints and observations:

$$B_i(x_i) = f_i(x_i)\prod_{s\in\Gamma_i} M_{si}(x_i)\,. \tag{1.15}$$

The value of each x_i can then be estimated as either the mean or mode of the belief, e.g., $x_i^* = \arg\max B_i(x_i)$, and the integer-valued information vector estimated as $b^* = \text{round}(Hx^*)$.

1.3.3 Fault Detection

Fault detection problems arise in most computer based engineering systems. Examples include aerospace (*e.g.*, jet engine health monitoring [24, 25]), industrial process control [26], automotive [27], telecommunications and networking [28, 29], computer systems [30], circuit fault identification [31], and many others.

In most papers in the literature it is assumed that faults affect measurements in an additive way. In fact there is a whole body of research devoted to optimal signature matrix design [32, 33]. A number of publications are concerned with the problem of fault identification in linear dynamical systems for both parametric [25] and binary faults [34, 35]. A number of heuristics have been proposed to tackle this problem, including variations of least-squares [25] and methods based on Kalman filtering [34, 24]. For some general references on this kind of work see [36, 37, 38].

The problem of fault identification with binary measurements has been extensively studied by the computer science community. The main references in this work are [39, 40], in which this problem is posed as a logical constraint satisfaction problem and a number of heuristics are proposed for solving it. More recent references on this type of problem are [41, 42], in which the authors formulate this problem as a minimum set cover problem on a graph, which they approximately solve using a method based on Lagrangian relaxation.

The idea of using convex relaxation as the basis for a heuristic for solving a combinatorial problem is quite old. Some recent examples include compressed sensing [43], sparse regressor selection [44], sparse signal detection [45], and sparse decoding [46]. The fault estimation problem is (mathematically) closely related to several detection problems arising in communications. In multi-user detection in code division multiple access (CDMA) systems, [47, 48, 6] the received signal plays the role of the measurements, and the transmitted bit pattern plays the role of the fault pattern; the goal is to estimate the transmitted bit pattern. (One important difference is that in the multi-user detection problem, each bit typically has an equal probability of being 0 or 1, whereas in fault detection, the prior probabilities that a bit is 1 is typically much lower). As has been pointed out in the literature, a good approach here is to solve a relaxed version of the resulting combinatorial problem. For example in [49, 50] the authors propose a semi definite programming (SDP) relaxation of the resulting mixed integer quadratic program. The performance of this method is theoretically analyzed in [51], while the authors of [52] perform an extensive experimental comparison of this SDP relaxation with various other heuristics. The most related paper to this work is the work of Zymnis *et al.* presented in [53]. Zymnis proposes an efficient Newton's method [54, §9.5] for solving the fault identification problem. The method can scale to large problems, and produces fault pattern estimation results of equal quality.

We note that several other works have proposed to use belief propagation for computing inference in related problems [20, 21, 55]. Those algorithms use a factor graph formulation where the messages sent are mixtures of Gaussians (using either parametric form or quantization). [20, 21] performs decoding of Low Density Lattice Codes (LDLC), [55] proposes to utilize belief propagation in the compressive sensing domain.

Chapter 2

LDLC

The non-parametric belief propagation (NBP) algorithm is an efficient method for approximated inference on continuous graphical models. The NBP algorithm was originally introduced in [12], but has recently been rediscovered independently in several domains, among them compressive sensing [55, 56] and low density lattice decoding [20], demonstrating very good empirical performance in these systems.

In this chapter, we investigate the theoretical relations between the LDLC decoder and belief propagation, and show it is an instance of the NBP algorithm. This understanding has both theoretical and practical consequences. From the theory point of view we provide a cleaner and more standard derivation of the LDLC update rules, from the graphical models perspective. From the practical side we propose to use the considerable body of research that exists in the NBP domain to allow construction of efficient decoders.

We further propose a new family of LDLC codes as well as a new LDLC decoder based on the NBP algorithm . By utilizing sparse generator matrices rather than the sparse parity check matrices used in the original LDLC work, we can obtain a more efficient encoder and decoder. We introduce the theoretical foundations which are the basis of our new decoder and give preliminary experimental results which show our decoder has comparable performance to the LDLC decoder.

The structure of this chapter is as follows. Section 2.1 rederive the original LDLC algorithm using standard graphical models terminology, and shows it is an instance of the NBP algorithm. Section 2.2 presents a new family of LDLC codes as well as our novel decoder. We further discuss the relation to the GaBP algorithm. In Section 2.3 we discuss convergence and give more general sufficient conditions for convergence, under the same assumptions used in the original LDLC work [20]. Section 2.4 brings preliminary experimental results of evaluating our NBP decoder vs. the LDLC decoder. We conclude in Section **??**.

2.1 A Pairwise Construction of the LDLC decoder

Before introducing our novel lattice code construction, we demonstrate that the LDLC decoder can be equivalently constructed using a *pairwise* graphical model. This construction will have

important consequences when relating the LDLC decoder to Gaussian belief propagation (Section 2.2.2) and understanding convergence properties (Section 2.3).

Theorem 1. *The LDLC decoder algorithm is an instance of the NBP algorithm executed on the following pairwise graphical model. Denote the number LDLC variable nodes as n and the number of check nodes as k[1]. We construct a new graphical model with $n+k$ variables, $X=(x_1,\cdots,x_{n+k})$ as follows. To match the LDLC notation we use the index letters $i,j,..$ to denote variables $1,...,n$ and the letters $s,t,...$ to denote new variables $n+1,...,n+k$ which will take the place of the check node factors in the original formulation. We further define the self and edge potentials:*

$$\psi_i(x_i) \propto \mathcal{N}(x_i; y_i, \sigma^2)\,, \quad \psi_s(x_s) \triangleq \sum_{b_s=-\infty}^{\infty} \mathcal{N}^{-1}(x_s; b_s, 0)\,, \quad \psi_{i,s}(x_i, x_s) \triangleq \exp(-x_i H_{is} x_s)\,. \tag{2.1}$$

Proof. The proof is constructed by substituting the edge and self potentials (2.7) into the belief propagation update rules. Since we are using a pairwise graphical model, we do not have two update rules from variable to factors and from factors to variables. However, to recover the LDLC update rules, we make the artificial distinction between the variable and factor nodes, where the nodes x_i will be shown to be related to the variable nodes in the LDLC decoder, and the nodes x_s will be shown to be related to the factor nodes in the LDLC decoder.

LDLC variable to factor nodes We start with the integral-product rule computed in the x_i nodes:

$$M_{is}(x_s) = \int_{x_i} \psi(x_i, x_s)\psi_i(x_i) \prod_{t\in\Gamma_i\backslash s} M_{ti}(x_i)dx_i\,.$$

The product of a mixture of Gaussians $\prod_{t\in\Gamma_i\backslash s} M_{ti}(x_i)$ is itself a mixture of Gaussians, where each component in the output mixture is the product of a single Gaussians selected from each input mixture $M_{ti}(x_i)$.

Lemma 2 (Gaussian product)**.** *[11, Claim 10], [20, Claim 2] Given p Gaussians $\mathcal{N}(m_1,v_1),\cdots,\mathcal{N}(m_p,v_p)$ their product is proportional to a Gaussian $\mathcal{N}(\bar{m},\bar{v})$ with*

$$\bar{v}^{-1} = \sum_{i=1}^{p} \frac{1}{v_i} = \sum_{i=1}^{p} \alpha_i\,, \qquad \bar{m} = (\sum_{i=1}^{p} m_i/v_i)\bar{v} = \sum_{i=1}^{p} \beta_i \bar{v}.$$

Proof. Is given in [11, Claim 10]. □

Using the Gaussian product lemma the l_s mixture component in the message from variable node i to factor node s is a single Gaussian given by

$$M_{is}^{l_s}(x_s) = \int_{x_i} \psi_{is}(x_i, x_s)\Big(\psi_i(x_i) \prod_{t\in\Gamma_i\backslash s} M_{ti}^{\tau}(x_i)\Big)dx_i =$$

[1] Our construction extends the square parity check matrix assumption to the general case.

$$\int_{x_i} \psi_{is}(x_i,x_s)\Big(\psi_i(x_i)\exp\{-\tfrac{1}{2}x_i^2(\sum_{t\in\Gamma_i\setminus s}\alpha_{ti}^{l_s})+x_i(\sum_{t\in\Gamma_i\setminus s}\beta_{ti}^{l_s})\}\Big)dx_i =$$

$$\int_{x_i} \psi_{is}(x_i,x_s)\Big(\exp(-\tfrac{1}{2}x^2\sigma^{-2}+x_iy_i\sigma^{-2})\exp\{-\tfrac{1}{2}x_i^2(\sum_{t\in\Gamma_i\setminus s}\alpha_{ti}^{l_s})+x_i(\sum_{t\in\Gamma_i\setminus s}\beta_{ti}^{l_s})\}\Big)dx_i =$$

$$\int_{x_i} \psi_{is}(x_i,x_s)\Big(\exp\{-\tfrac{1}{2}x_i^2(\sigma^{-2}+\sum_{t\in\Gamma_i\setminus s}\alpha_{ti}^{l_s})+x_i(y_i\sigma^{-2}+\sum_{t\in\Gamma_i\setminus s}\beta_{ti}^{l_s})\}\Big)dx_i\,.$$

We got a formulation which is equivalent to LDLC variable nodes update rule given in (1.7). Now we use the following lemma for computing the integral:

Lemma 3 (Gaussian integral)**.** *Given a (1D) Gaussian* $\phi_i(x_i) \propto \mathcal{N}(x_i;m,v)$*, the integral* $\int_{x_i}\psi_{i,s}(x_i,x_s)\phi_i(x_i)dx_i$ *, where is a (2D) Gaussian* $\psi_{i,s}(x_i,x_s)\triangleq\exp(-x_iH_{is}x_s)$ *is proportional to a (1D) Gaussian* $\mathcal{N}^{-1}(H_{is}m,H_{is}^2v)$*.*

Proof.

$$\int_{x_i}\psi_{ij}(x_i,x_j)\phi_i(x_i)dx_i \quad\propto\quad \int_{x_i}\exp\left(-x_iH_{is}x_s\right)\exp\{-\tfrac{1}{2}(x_i-m)^2/v\}dx_i =$$

$$=\int_{x_i}\exp\Big((-\tfrac{1}{2}x_i^2/v)+(m/v-H_{is}x_s)x_i\Big)dx_i \quad\propto\quad \exp\left((m/v-H_{is}x_s)^2/(-\tfrac{2}{v})\right),$$

where the last transition was obtained by using the Gaussian integral:

$$\int_{-\infty}^{\infty}\exp\left(-ax^2+bx\right)dx=\sqrt{\pi/a}\exp\left(b^2/4a\right).$$

$$\exp\left((m/v-H_{is}x_s)^2/(-\tfrac{2}{v})\right)=\exp\{-\tfrac{1}{2}(v(m/v-H_{is}x_s)^2)\}=$$
$$=\exp\{-\tfrac{1}{2}(H_{is}^2v)x_s^2+(H_{is}m)x_s-\tfrac{1}{2}v(m/v)^2\}\propto\exp\{-\tfrac{1}{2}(H_{is}^2v)x_s^2+(H_{is}m)x_s\}\,.$$

□

Using the results of Lemma 3 we get that the sent message between variable node to a factor node is a mixture of Gaussians, where each Gaussian component k is given by

$$M_{is}^{\mathbf{l}}(x_s)=\mathcal{N}^{-1}(x_s;H_{is}m_{is}^{l_s},H_{is}^2v_{is}^{l_s})\,.$$

Note that in the LDLC terminology the integral operation as defined in Lemma 3 is called stretching. In the LDLC algorithm, the stretching is computed by the factor node as it receives the message from the variable node. In NBP, the integral operation is computed at the variable nodes.

LDLC Factors to variable nodes We start again with the BP integral-product rule and handle the x_s variables computed at the factor nodes.

$$M_{si}(x_i) = \int_{x_s} \psi_{is}(x_i, x_s)\psi_s(x_s) \prod_{j\in\Gamma_s\setminus i} M_{js}(x_j)\, dx_s.$$

Note that the product $\prod_{j\in\Gamma_s\setminus i} M^{\tau}_{js}(x_j)$ is a mixture of Gaussians, where the k-th component is computed by selecting a single Gaussian from each message M^{τ}_{js} from the set $j \in \Gamma_s \setminus i$ and applying the product lemma (Lemma 2). We get

$$\int_{x_s} \psi_{is}(x_i, x_s)\Big(\psi_s(x_s)\exp\{-\tfrac{1}{2}x_s^2(\sum_{k\in\Gamma_s\setminus i} H_{ks}^2 v_{ks}^{l_i}) + x_s(\sum_{k\in\Gamma_s\setminus i} H_{ks} m_{ks}^{l_i})\}\Big)dx_s. \tag{2.2}$$

We continue by computing the product with the self potential $\psi_s(x_s)$ to get

$$= \int_{x_s} \psi_{is}(x_i, x_s)\Big(\sum_{b_s=-\infty}^{\infty} \exp(b_s x_s)\exp\{-\tfrac{1}{2}x_s^2(\sum_{k\in\Gamma_s\setminus i} H_{ks}^2 v_{ks}^{l_i}) + x_s(\sum_{k\in\Gamma_s\setminus i} H_{ks} m_{ks}^{l_i})\}\Big)dx_s =$$

$$= \sum_{b_s=-\infty}^{\infty}\int_{x_s} \psi_{is}(x_i, x_s)\Big(\exp(b_s x_s)\exp\{-\tfrac{1}{2}x_s^2(\sum_{k\in\Gamma_s\setminus i} H_{ks}^2 v_{ks}^{l_i}) + x_s(\sum_{k\in\Gamma_s\setminus i} H_{ks} m_{ks}^{l_i})\}\Big)dx_s =$$

$$= \sum_{b_s=-\infty}^{\infty}\int_{x_s} \psi_{is}(x_i, x_s)\Big(\exp\{-\tfrac{1}{2}x_s^2(\sum_{k\in\Gamma_s\setminus i} H_{ks}^2 v_{ks}^{l_i}) + x_s(b_s + \sum_{k\in\Gamma_s\setminus i} H_{ks} m_{ks}^{l_i})\}\Big)dx_s =$$

$$= \sum_{b_s=\infty}^{-\infty}\int_{x_s} \psi_{is}(x_i, x_s)\Big(\exp\{-\tfrac{1}{2}x_s^2(\sum_{k\in\Gamma_s\setminus i} H_{ks}^2 v_{ks}^{l_i}) + x_s(-b_s + \sum_{k\in\Gamma_s\setminus i} H_{ks} m_{ks}^{l_i})\}\Big)dx_s\,.$$

Finally we use Lemma 3 to compute the integral and get

$$= \sum_{b_s=\infty}^{-\infty} \exp\{-\tfrac{1}{2}x_s^2 H_{si}^2(\sum_{k\in\Gamma_s\setminus i} H_{ks}^2 v_{ks}^{l_i})^{-1} + x_s H_{si}(\sum_{k\in\Gamma_s\setminus i} H_{ks}^2 v_{ks}^{l_i})^{-1}(-b_s + \sum_{k\in\Gamma_s\setminus i} H_{ks} m_{ks}^{l_i})\}dx_s\,.$$

It is easy to verify this formulation is identical to the LDLC update rules (1.9). □

2.2 Using Sparse Generator Matrices

We propose a new family of LDLC codes where the generator matrix G is sparse, in contrast to the original LDLC codes where the parity check matrix H is sparse. Table 2.1 outlines the properties of our proposed decoder. Our decoder is designed to be more efficient than the original LDLC decoder, since as we will soon show, both encoding, decoding and initialization operations are more efficient in the NBP decoder. We are currently in the process of fully evaluating our decoder performance relative to the LDLC decoder. Initial results are reported in Section VI. Our decoder is available for download on [57].

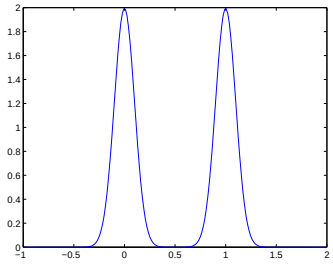

Figure 2.1: The approximating function $g_s^{relax}(x)$ for the binary case.

2.2.1 The NBP decoder

We use an undirected bipartite graph, with variables nodes $\{b_i\}$, representing each element of the vector b, and observation nodes $\{z_i\}$ for each element of the observation vector y. We define the self potentials $\psi_i(z_i)$ and $\psi_s(b_s)$ as follows:

$$\psi_i(z_i) \propto \mathcal{N}(z_i; y_i, \sigma^2)\,, \qquad \psi_s(b_s) = \begin{cases} 1 & b_s \in \mathbb{Z} \\ 0 & \text{otherwise} \end{cases}\,, \tag{2.3}$$

and the edge potentials:

$$\psi_{i,s}(z_i, b_s) \triangleq \exp(-\tfrac{1}{2} z_i G_{is} x_s)\,.$$

Each variable node b_s is connected to the observation nodes as defined by the encoding matrix G. Since G is sparse, the resulting bipartite graph sparse as well. As with LDPC decoders [4], the belief propagation or sum-product algorithm [58, 3] provides a powerful approximate decoding scheme.

For computing the MAP assignment of the transmitted vector b using non-parametric belief propagation we perform the following relaxation, which is one of the main novel contributions of this paper. Recall that in the original problem, b are only allowed to be integers. We relax the function $\psi_s(x_s)$ from a delta function to a mixture of Gaussians centered around integers.

$$\psi_s^{relax}(b_s) \propto \sum_{i \in \mathbb{Z}} \mathcal{N}(i, v)\,.$$

The variance parameter v controls the approximation quality, as $v \to 0$ the approximation quality is higher. Figure 2 plots an example relaxation of $\psi_i(b_s)$ in the binary case. We have defined the self and edge potentials which are the input to the NBP algorithm. Now it is possible to run the NBP algorithm using (1.1) and get an approximate MAP solution to (1.4). The derivation of the NBP decoder update rules is similar to the one done for the LDLC decoder, thus omitted. However, there are several important differences that should be addressed. We start by analyzing the algorithm efficiency.

We assume that the input to our decoder is the sparse matrix G, there is no need in computing the encoding matrix $G = H^{-1}$ as done in the LDLC decoder. Naively this initialization takes

Algorithm	LDLC	NBP
Initialization operation	$G = H^{-1}$	None
Initialization cost	$O(n^3)$	-
Encoding operation	Gb	Gb
Encoding cost	$O(n^2)$	$O(nd), \quad d \ll n$
Cost per node per iteration	$O(q \log(q) d)$	$O(qd)$
Post run operation	Hx	None
Post run cost	$O(nd)$	-

Table 2.1: Comparison of LDLC decoder vs. NBP decoder

Algorithm	LDLC decoder	NBP decoder
Update rules	Two: factor to variables and vice versa	One
Sparsity assumption	Decoding matrix H	Encoding matrix G
Algorithm derivation	Custom	Standard NBP
Graphical model	Factor graph	Pairwise potentials
Related Operations	Stretch/Unstretch Convolution periodic extension	Integral product product

Table 2.2: Inheret differences between LDLC and NBP decoders

$O(n^3)$ cost. The encoding in our scheme is done as in LDLC by computing the multiplication Gb. However, since G is sparse in our case, encoding cost is $O(nd)$ where $d << n$ is the average number of non-zeros entries on each row. Encoding in the LDLC method is done in $O(n^2)$ since even if H is sparse, G is typically dense. After convergence, the LDLC decoder multiplies by the matrix H and rounds the result to get b. This operation costs $O(nd)$ where d is the average number of non-zero entries in H. In contrast, in the NBP decoder, b is computed directly in the variable nodes.

In addition to efficiency, there are several inherent differences between the two algorithms. Summary of the differences is given in Table 2.2. First, we use a standard formulation of BP using pairwise potentials form, which means there is a single update rule, and not two update rules from left to right and right to left. This can potentially allow more efficient implementations, since the LDLC decoder can be implemented using quantization, such that the convolution is computed in the fft domain. In our framework, there is no convolution, and the product using quantization is computed using elementwise product. Second, our formulation is different since we use different self potentials. The self potentials are incorporated into the computation via the product rule of the integral-product algorithm. Our self potentials perform operations which are equivalent in the LDLC terminology to the convolution and periodic extension operation. The integral computed in the NBP decoder is related to the stretch/unstretch operation of the LDLC decoder.

2.2.2 The relation of the NBP decoder to GaBP

In this section we show that simplified version of the NBP decoder coincides with the GaBP algorithm. The simplified version is obtained, when instead of using our proposed Gaussian mixture prior, we initialize the NBP algorithm with a prior composed of a single Gaussian.

Theorem 4. *By initializing $\psi_s(b_s) \sim \mathcal{N}(0,1)$ to be a (single) Gaussian the NBP decoder update rules are identical to update rules of the GaBP algorithm.*

Lemma 5. *By initializing $\psi_s(x_s)$ to be a (single) Gaussian the messages of the NBP decoder are single Gaussians.*

Proof. Assume both the self potentials $\psi_s(b_s), \psi_i(z_i)$ are initialized to a single Gaussian, every message of the NBP decoder algorithm will remain a Gaussian. This is because the product (1.2) of single Gaussians is a single Gaussian, the integral and (1.3) of single Gaussians produce a single Gaussian as well. □

Now we are able to prove Theorem 4:

Proof. We start writing the update rules of the variable nodes. We initialize the self potentials of the variable nodes $\psi_i(z_i) = \mathcal{N}(z_i; y_i, \sigma^2)$, Now we substitute, using the product lemma and Lemma 3.

$$M_{is}(b_s) = \int_{z_i} \psi_{i,s}(z_i, b_s)\Big(\psi_i(z_i) \prod_{t\in\Gamma_i\setminus s} M_{ti}(z_i)\Big)dz_i =$$

$$\int_{z_i} \psi_{i,s}(z_i, b_s)\Big(\exp(-\tfrac{1}{2}z_i^2\sigma^{-2} + y_i z_i \sigma^{-2}) \prod_{t\in\Gamma_i\setminus s} \exp(-\tfrac{1}{2}z_i^2\alpha_{ti} + z_i\beta_{ti})\Big)dz_i$$

$$\int_{z_i} \psi_{i,s}(z_i, b_s)\Big(\exp(-\tfrac{1}{2}z_i^2(\sigma^{-2} + \sum_{t\in\Gamma_i\setminus s} \alpha_{ti}) + z_i(\sigma^{-2}y_i + \sum_{t\in\Gamma_i\setminus s} \beta_{ti})\Big)dz_i =$$

$$\propto \exp\Big(-\tfrac{1}{2}z_i^2 G_{is}^2(\sigma^{-2} + \sum_{t\in\Gamma_i\setminus s} \alpha_{ti})^{-1} + z_i G_{is}(\sigma^{-2} + \sum_{t\in\Gamma_i\setminus s} \alpha_{ti})^{-1}(\sigma^{-2}y_i + \sum_{t\in\Gamma_i\setminus s} \beta_{ti})\Big)$$

. Now we get GaBP update rules by substituting $J_{ii} \triangleq \sigma^{-2}, J_{is} \triangleq G_{is}, h_s \triangleq \sigma^{-2}y_i$:

$$\alpha_{is} = -J_{is}^2\alpha_{i\setminus s}^{-1} = -J_{is}^2(J_{ii} + \sum_{t\in\Gamma_i\setminus s} \alpha_{ti})^{-1}, \quad \beta_{is} = -J_{is}\alpha_{i\setminus s}^{-1}\beta_{i\setminus s} = -J_{is}\Big(\alpha_{i\setminus s}^{-1}(h_i + \sum_{t\in\Gamma_i\setminus s} \beta_{ti})\Big)\,. \tag{2.4}$$

We continue expanding

$$M_{si}(z_i) = \int_{b_s} \psi_{i,s}(z_i, b_s)\Big(\psi_s(b_s) \prod_{k\in\Gamma_s\setminus i} M_{ks}^{\tau}(b_s)\Big)db_s$$

Similarly using the initializations $\psi_s(b_s) = \exp\{-\frac{1}{2}b_s^2\},\ \psi_{i,s}(z_i, b_s) \triangleq \exp(-z_i G_{is} b_s)$.

$$\int_{b_s} \psi_{i,s}(z_i, b_s)\Big(\exp\{-\tfrac{1}{2}b_s^2\} \prod_{k\in\Gamma_s\setminus i} \exp(-\tfrac{1}{2}b_s^2\alpha_{is} + b_s\beta_{ks})\Big)db_s =$$

$$\int_{b_s} \psi_{i,s}(z_i, b_s)\Big(\exp\{-\tfrac{1}{2}b_s^2(1 + \sum_{k\in\Gamma_s\setminus i} \alpha_{is}) + b_s(\sum_{k\in\Gamma_s\setminus i} \beta_{ks})\}\Big)db_s =$$

$$\exp\{-\tfrac{1}{2}b_s^2 G_{is}^2(1 + \sum_{k\in\Gamma_s\setminus i} \alpha_{is})^{-1} + b_s G_{is}(1 + \sum_{k\in\Gamma_s\setminus i} \alpha_{is})^{-1}(\sum_{k\in\Gamma_s\setminus i} \beta_{ks})\}$$

. Now we get GaBP update rules by substituting $J_{ii} \triangleq 1, J_{si} \triangleq G_{is}, h_i \triangleq 0$:

$$\alpha_{si} = -J_{si}^2\alpha_{s\setminus i}^{-1} = -J_{si}^2(J_{ii} + \sum_{k\in\Gamma s\setminus i} \alpha_{is})^{-1}, \quad \beta_{si} = -J_{si}\alpha_{s\setminus i}^{-1}\beta_{s\setminus i} = -J_{si}\Big(\alpha_{s\setminus i}^{-1}(h_i + \sum_{k\in\Gamma s\setminus i} \beta_{ks})\Big). \tag{2.5}$$

□

Tying together the results, in the case of a single Gaussian self potential, the NBP decoder is initialized using the following inverse covariance matrix:

$$J \triangleq \begin{pmatrix} I & G \\ G^T & \mathbf{diag}(\sigma^{-2}) \end{pmatrix}$$

. We have shown that a simpler version of the NBP decoder, when the self potentials are initialized to be single Gaussians boils down to GaBP algorithm. It is known [7] that the GaBP algorithm solves the following least square problem $\min_{b\in\mathbb{R}^n} \|Gb - y\|$ assuming a Gaussian prior on b, $p(b) \sim \mathcal{N}(0, 1)$, we get the MMSE solution $b^* = (G^TG)^{-1}G^Ty$. Note the relation to (1.4). The difference is that we relax the LDLC decoder assumption that $b \in \mathbb{Z}^n$, with $b \in \mathbb{R}^n$.

Getting back to the NBP decoder, Figure 2 compares the two different priors used, in the NBP decoder and in the GaBP algorithm, for the bipolar case where $b \in \{-1, 1\}$. It is clear that the Gaussian prior assumption on b is not accurate enough. In the NBP decoder, we relax the delta function (2.3) to a Gaussian mixture prior composed of mixtures centered around Integers. Overall, the NBP decoder algorithm can be thought of as an extension of the GaBP algorithm with more accurate priors.

2.3 Convergence analysis

The behavior of the belief propagation algorithm has been extensively studied in the literature, resulting in sufficient conditions for convergence in the discrete case [59] and in jointly Gaussian models [60]. However, little is known about the behavior of BP in more general continuous systems. The original LDLC paper [20] gives some characterization of its convergence properties under several simplifying assumptions. Relaxing some of these assumptions and using our pairwise

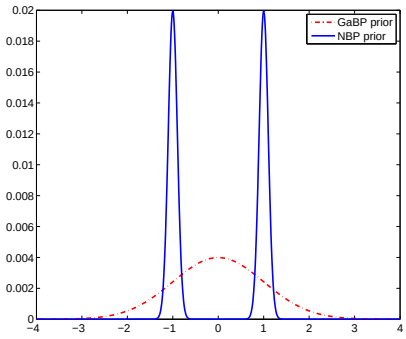

Figure 2.2: Comparing GaBP prior to the prior we use in the NBP decoder for the bipolar case ($b \in \{-1, 1\}$).

factor formulation, we show that the conditions for GaBP convergence can also be applied to yield new convergence properties for the LDLC decoder.

The most important assumption made in the LDLC convergence analysis [20] is that the system converges to a set of "consistent" Gaussians; specifically, that at all iterations τ beyond some number τ_0, only a *single* integer b_s contributes to the Gaussian mixture. Notionally, this corresponds to the idea that the decoded information values themselves are well resolved, and the convergence being analyzed is with respect to the transmitted bits x_i. Under this (potentially strong) assumption, sufficient conditions are given for the decoder's convergence. The authors also assume that H consists of a Latin square in which each row and column contain some permutation of the scalar values $h_1 \geq \ldots \geq h_d$, up to an arbitrary sign.

Four conditions are given which should all hold to ensure convergence:

- LDLC-I: $\det(H) = \det(G) = 1$.
- LDLC-II: $\alpha \leq 1$, where $\alpha \triangleq \frac{\sum_{i=2}^{d} h_i^2}{h_1^2}$.
- LDLC-III: The spectral radius of $\rho(F) < 1$ where F is a $n \times n$ matrix defined by:

$$F_{k,l} = \begin{cases} \frac{h_{rk}}{h_{rl}} & \text{if } k \neq l \text{ and there exist a row } r \text{ of } H \text{ for which } |H_{rl}| = h_1 \text{ and } H_{rk} \neq 0 \\ 0 & \text{otherwise} \end{cases}$$

- LDLC-IV: The spectral radius of $\rho(\tilde{H}) < 1$ where $\tilde{H}$ is derived from H by permuting the rows such that the h_1 elements will be placed on the diagonal, dividing each row by the appropriate diagonal element ($+h_1$ or $-h_1$), and then nullifying the diagonal.

Using our new results we are now able to provide new convergence conditions for the LDLC decoder.

Corollary 6. *The convergence of the LDLC decoder depends on the properties of the following matrix:*

$$J \triangleq \begin{pmatrix} \mathbf{0} & H \\ H^T & \mathbf{diag}(1/\sigma^2) \end{pmatrix}. \tag{2.6}$$

Proof. In Theorem 1 we have shown an equivalence between the LDLC algorithm to NBP initialized with the following potentials:

$$\psi_i(x_i) \propto \mathcal{N}(x_i; y_i, \sigma^2), \quad \psi_s(x_s) \triangleq \sum_{b_s=-\infty}^{\infty} \mathcal{N}^{-1}(x_s; b_s, 0), \quad \psi_{i,s}(x_i, x_s) \triangleq \exp(x_i H_{is} x_s). \tag{2.7}$$

We have further discussed the relation between the self potential $\psi_s(x_s)$ and the periodic extension operation. We have also shown in Theorem 4 that if $\psi_s(x_s)$ is a *single* Gaussian (equivalent to the assumption of "consistent" behavior), the distribution is jointly Gaussian and rather than NBP (with Gaussian mixture messages), we obtain GaBP (with Gaussian messages). Convergence of the GaBP algorithm is dependent on the inverse covariance matrix J and not on the shift vector h.

Now we are able to construct the appropriate inverse covariance matrix J based on the pairwise factors given in Theorem 1. The matrix J is a 2×2 block matrix, where the check variables x_s are assigned the upper rows and the original variables are assigned the lower rows. The entries can be read out from the quadratic terms of the potentials (2.7), with the only non-zero entries corresponding to the pairs (x_i, x_s) and self potentials (x_i, x_i).

□

Based on Corollary 6 we can characterize the convergence of the LDLC decoder, using the sufficient conditions for convergence of GaBP. Either one of the following two conditions are sufficient for convergence:

[GaBP-I] (*walk-summability* [60]) $\rho(I - |D^{-1/2} J D^{-1/2}|) < 1$ where $D \triangleq \mathbf{diag}(J)$.

[GaBP-II] (*diagonal dominance* [5]) J is diagonally dominant (i.e. $|J_{ii}| >= \sum_{j \neq i} |J_{ij}|, \forall i$).

A further difficulty arises from the fact that the upper diagonal of (2.6) is zero, which means that both [GaBP-I,II] fail to hold. There are three possible ways to overcome this.

1. Create an approximation to the original problem by setting the upper left block matrix of (2.6) to $\mathbf{diag}(\epsilon)$ where $\epsilon > 0$ is a small constant. The accuracy of the approximation grows as ϵ is smaller. In case either of [GaBP-I,II] holds on the fixed matrix the "consistent Gaussians" converge into an approximated solution.

2. In case a permutation on J (2.6) exists where either [GaBPI,II] hold for permuted matrix, then the "consistent Gaussians" convergence to the correct solution.

3. Use preconditioning to create a new graphical model where the edge potentials are determined by the information matrix HH^T, $\psi_{i,s}(x_i, x_s) \triangleq \exp(x_i \{HH^T\}_{is} x_s)$ and the self

potentials of the x_i nodes are $\psi_i(x_i) \triangleq \exp\{-\frac{1}{2}x_i^2\sigma^{-2} + x_i\{Hy\}_i\}$. The proof of the correctness of the above construction is given in [47]. The benefit of this preconditioning is that the main diagonal of HH^T is surely non zero. If either [GaBP-I,II] holds on HH^T then "consistent Gaussians" convergence to the correct solution. However, the matrix HH^T may not be sparse anymore, thus we pay in decoder efficiency.

Overall, we have given two sufficient conditions for convergence, under the "consistent Gaussian" assumption for the means and variances of the LDLC decoder. Our conditions are more general because of two reasons. First, we present a single sufficient condition instead of four that have to hold concurrently in the original LDLC work. Second, our convergence analysis does not assume Latin squares, not even square matrices and does not assume anything about the sparsity of H. This extends the applicability of the LDLC decoder to other types of codes. Note that our convergence analysis relates to the mean and variances of the Gaussian mixture messages. A remaining open problem is the convergence of the amplitudes – the relative heights of the different consistent Gaussians.

2.4 Experimental results

In this section we report preliminary experimental results of our NBP-based decoder. Our implementation is general and not restricted to the LDLC domain. Specifically, recent work by Baron *et al.* [56] had extensively tested our NBP implementation in the context of the related compressive sensing domain. Our Matlab code is available on the web on [57].

We have used a code lengths of $n = 100, n = 1000$, where the number of non zeros in each row and each column is $d = 3$. The non-zeros entries of the sparse encoding matrix G were selected randomly out of $\{-1, 1\}$. We have used bipolar signaling, $b \in \{-1, 1\}$. We have calculated the maximal noise level σ^2_{max} using Poltyrov generalized definition for channel capacity using unrestricted power assumption [61]. For bipolar signaling $\sigma^2_{max} = 4\sqrt[n]{\det(G)^2}/2\pi e$. When applied to lattices, the generalized capacity implies that there exists a lattice G of high enough dimension n that enables transmission with arbitrary small error probability, if and only if $\sigma^2 < \sigma^2_{max}$. Figure 2.3 plots SER (symbol error rate) of the NBP decoder vs. the LDLC decoder for code length $n = 100, n = 1000$. The x-axis represent the distance from capacity in dB as calculated using Poltyrov equation. As can be seen, our novel NBP decoder has better SER for $n = 100$ for all noise levels. For $n = 1000$ we have better performance for high noise level, and comparable performance up to 0.3dB from LDLC for low noise levels. We are currently in the process of extending our implementation to support code lengths of up $n = 100,000$. Initial performance results are very promising.

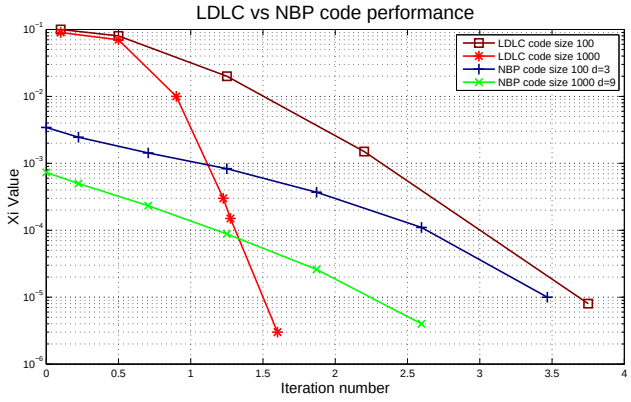

Figure 2.3: NBP vs. LDLC decoder performance

Chapter 3

Fault Detection

In this chapter we present a novel distributed method for identifying a pattern of faults, given a set of noisy measurements of a system. The goal is to estimate the fault pattern most likely to have occurred. An important secondary goal is to identify a (possibly empty) set of other fault patterns, called the *ambiguity group*, that explain the measurements almost as well as the most likely one.

In this work we propose a novel approach, based on the non-parametric belief propagation algorithm. We propose a novel relaxation of the fault pattern prior using a mixture of Gaussians. Our solution takes into account both the binary nature of the problem as well as the sparsity (the prior knowledge about fault probability). As far as we know, this is the first time both considerations are taken into account in this context. Using an extensive experimental study we show that our approach has the best performance in detecting correct fault patterns relative to recent state-of-the-art algorithms. Furthermore, unlike many of the reference algorithms, our algorithm is distributed and can be applied in sensor networks for performing online fault identification.

The structure of this Chapter is as follows. Section 3.1 introduces the fault identification problem in terms of MAP estimation. Section 3.2 presents our novel solution which is based on the non-parametric belief propagation algorithm. Section 3.3 details some local optimization procedures for improving the quality of the computed solution. Section 3.4 compares the accuracy of eight different state-of-the-art methods for solving the fault identification problem, and shows that our proposed method has the highest accuracy.

3.1 Fault identification problem

In this section we describe the model in detail, and the basic MAP method for estimating the fault pattern.

3.1.1 Fault model and prior distribution

We consider a system in which any of the 2^n combinations of n potential faults can occur. We encode a fault pattern, *i.e.*, a set of faults, as a vector $x \in \{0,1\}^n$, where $x_j = 1$ means that fault j has occurred. We assume that faults occur independently, and that fault j occurs with known probability p_j. Thus, the (prior) probability of fault pattern x occurring is

$$p(x) = \prod_{j=1}^{n} p_j^{x_j} \left(1 - p_j\right)^{1-x_j} .$$

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

3.1.2 Measurement model

We assume that m scalar real measurements, denoted $y \in \mathbf{R}^m$, are available. These measurements depend on the fault pattern x in the following way:

$$y = Ax + v,$$

where $A \in \mathbf{R}^{m\times n}$ is the *fault signature matrix* and $v \in \mathbf{R}^m$ is a vector of measurement noises, this formulation assumes a linear addition of faults.

We assume the fault signature matrix A is known. Its jth column $a_j \in \mathbf{R}^m$ gives the measurement, if the measurement were linear and there were no noise, when only fault j has occurred. For this reason a_j is called the jth *fault signature*. Since x is a Boolean vector, Ax is just the sum of the fault signatures corresponding to the faults that have occurred. So our real assumption here is only that, without measurement noise and nonlinearity, the measurements would be additive in the faults.

We assume the measurement noise $v \in \mathbf{R}^m$ is random, with v_i independent of each other and x, each with $\mathcal{N}(0, \sigma^2)$ distribution.

3.1.3 Posterior probability

Let $p(x|y)$ denote the (posterior) probability of fault pattern x, given the measurement y. We define the loss of x, given the measurement y, as the log of the ratio of the posterior probability of the null hypothesis to the posterior probability of x, *i.e.*,

$$\begin{aligned} l_y(x) &= \log \frac{p(0|y)}{p(x|y)} = \log\left(\frac{p(0)p(y|0)}{p(x)p(y|x)}\right) = \log p(0) - \log p(x) + \log p(y|0) - \log p(y|x) \\ &= \lambda^T x + \sum_{i=1}^{m} \left(\log p(y_i|0) - \log p(y_i|x)\right), \end{aligned}$$

where $\lambda_j = \log((1-p_j)/p_j)$. In these expressions we have to interpret $p(y_i|x)$ and $p(y_i|0)$ carefully. When y is a linear measurement, these are conditional *densities*; when y_i takes on only a finite number of values, as occurs with quantized measurements, these are actual *probabilities*.

The loss tells us how improbable it is that fault x has occurred, given the measurement y, compared to the null hypothesis $x = 0$. If $l_y(x) = 0$, the fault pattern x is just as probable as the null hypothesis $x = 0$. If $l_y(x) = -1$, the fault pattern x is e times more probable than the null hypothesis.

We now work out the loss function more explicitly. Using linear measurements, the conditional density of y_i given x is

$$p(y_i|x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-(y_i - \tilde{a}_i^T x)^2/2\sigma^2\right),$$

where $\tilde{a}_i$ is the ith row of A. Therefore we have

$$\begin{aligned} l_y(x) &= \lambda^T x + (1/2\sigma^2)\sum_{i=1}^{m}(-y_i^2 + (y_i - \tilde{a}_i^T x)^2) = \lambda^T x + (1/2\sigma^2)(-\|y\|^2 + \|Ax - y\|^2) \\ &= (1/2\sigma^2)x^T A^T A x + (\lambda - (1/\sigma^2)A^T y)^T x, \end{aligned}$$

which is a convex quadratic function of x.

3.1.4 MAP estimation

To find the fault pattern x with maximum posterior probability (or equivalently, minimum loss $l_y(x)$) we solve the problem

$$\begin{array}{ll} \text{minimize} & l_y(x) \\ \text{subject to} & x_j \in \{0,1\}, \quad j = 1, \ldots, n, \end{array} \tag{3.1}$$

with variable x. We have already seen that with linear or quantized measurements, the objective function is convex. The constraint is that x is a Boolean vector, so a problem like this is sometimes called a Boolean-convex program or a mixed-integer convex program. When the measurements are linear, the MAP problem (3.1) is a convex mixed-integer quadratic program (MIQP).

Any solution of the MAP problem (3.1) is a MAP estimate of the fault pattern, *i.e.*, a fault pattern that is most probable, given the measurement. It is also very useful to obtain the ambiguity group, *i.e.*, the set of fault patterns with loss that is near to the loss of a MAP estimate. If all other fault patterns have a loss much larger than the MAP estimate (*i.e.*, the ambiguity group is empty), we can have high confidence in our estimate. On the other hand, if several other fault patterns have a loss similar to the MAP loss, they explain the measurement almost as well as the MAP estimate, and so must be considered possible values of the true fault. One way to determine the ambiguity group is to find the K fault patterns with least loss, *i.e.*, highest posterior probability. From these ambiguity group candidates, we can form the ambiguity group by taking only the patterns with loss near to the MAP loss.

The method we present in this paper is, like these methods, a heuristic for solving the MAP problem (3.1) approximately. Our approach is based on using the Non-parametric belief propagation algorithm for computing an approximate MAP solution.

3.2 Solution via NBP

In this section we present our main novel contribution. We propose to utilize the non-parametric belief propagation for solving the fault detection problem. We shift the problem from linear algebra domain to the probabilistic domain and find the MAP assignment using probabilistic tools. As far as we know, it is the first time this algorithm is linked to the solution of interior-point methods.

3.2.1 Non-parametric BP overview

Non-parametric belief propagation (NBP) is an inference algorithm for graphical models containing continuous, possibly non-Gaussian random variables [12]. NBP extends the popular class of particle filtering algorithms, which assume variables are related by a Markov chain, to general graphs. Such sample-based representations are particularly useful in high-dimensional spaces, where discretization becomes computationally difficult. In NBP, messages are represented by collections of weighted samples, smoothed by a Gaussian shape – in other words, Gaussian mixtures.

In the current work, we propose to utilize the NBP algorithm for solving the fault identification problem. NBP can be easily formulated using a factor graph. NBP proceeds by sending messages between the nodes of the factor graph, where the message at iteration $\tau+1$ is a function of the incoming messages at iteration τ:

$$M_{is}^{\tau+1}(x_i) = f_i(x_i) \prod_{t\in\Gamma_i\setminus s} M_{ti}^{\tau}(x_i)\,, \tag{3.2}$$

$$M_{si}^{\tau+1}(x_i) = \int_{x_s} g_s(x_s) \prod_{j\in\Gamma_s\setminus i} M_{js}^{\tau}(x_j)\,. \tag{3.3}$$

where Γ_i indicates the neighbors of node x_i. It is reasonably straightforward to see that, if initialized to Gaussians or Gaussian mixtures, the messages M will maintain the form of Gaussian mixture distributions. For compactness, we have abused notation to use indices i, j as shorthand for variable nodes (x_i, x_j) and s, t as shorthand for factor nodes (g_s, g_t). in which case it computes messages as in (3.2)-(3.3). Note that the number of Gaussian mixtures which result from the product procedure grows exponentially. For this reason, one must approximate the mixture product in some way. NBP, relies on a stochastic sampling process to preserve only high-likelihood components. A number of sampling algorithms have been designed to ensure that this process is as efficient as possible [13, 14, 15]. These methods avoid incremental products, which have the potential to discard important components in the early steps, before seeing all the incoming messages [13].

Deterministic reduction of the Gaussian mixture components has also been applied to NBP. In [17,18], an $O(n)$ greedy algorithm is employed to trade off the complexity of the representation (measured in terms of communication cost) with the resulting error under various metrics, where n is the number of samples drawn before approximation. Other density approximation methods could just as easily be applied [16].

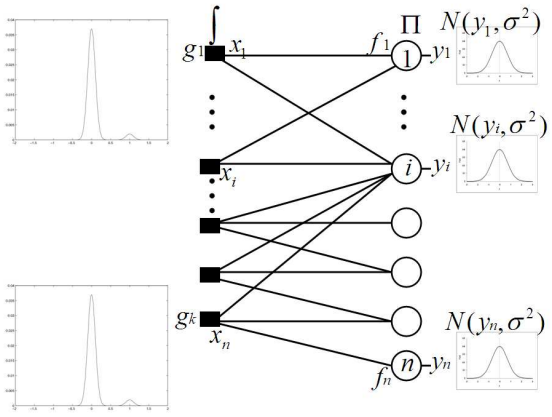

Figure 3.1: Factor graph representation of the NBP algorithm. To the right are the variable nodes which are initialized with the observations as self-potentials (function f). To the left are the factor nodes (function g) which represent the hidden variables x.

3.2.2 Custom NBP algorithm for solving the FD problem

The non-parametric belief propagation algorithm for solving the fault detection problem is best described in terms of a factor graph [3], representing the information and constraints on x arising from our knowledge of y and the fact that $x \in \{0, 1\}$. Specifically, the factor graph is a bipartite graph with variables nodes $\{x_i\}$, representing each element of the hidden vector x, and factor nodes $\{f_i, g_s\}$ corresponding to functions

$$f_i(x_i) = \mathcal{N}(z_i; y_i, \sigma^2)\,, \qquad g_s(x_s) = \begin{cases} 1 & x_s \in \{0, 1\} \\ 0 & \text{otherwise} \end{cases}\,,$$

where A_s is the s^{th} row of the fault signature matrix A. Each variable node x_i is connected to those factors for which it is an argument. We assume that A is sparse, A_s has few non-zero entries, making the resulting factor graph sparse as well.

Since $f_i(x_i)$ is Gaussian, corresponding to the observation likelihood (the channel model), this product (3.2) is also a Gaussian mixture. The integration step is equivalent to a convolution operation with a train of delta functions, which also preserves the form of a Gaussian mixture model [20]. Typically, the algorithm is initialized using the local observations, so that $M_{is}^0(x_i) = f_i(x_i)$.

3.2.3 Problem relaxation

The main contribution of this work, is in proposing a new type of relaxation for the boolean constraints function $g_x(x_s)$. Unlike the convex relaxations described in the introduction which

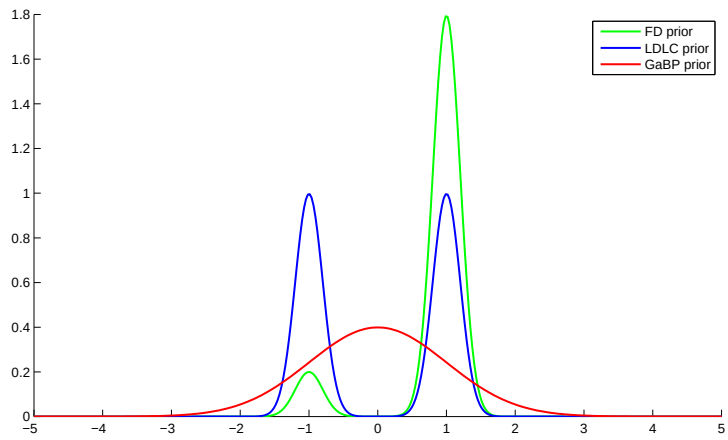

Figure 3.2: The approximating function $g_s^{relax}(x)$, Compared against the GaBP prior and the LDLC prior

typically limit x_s to the domain $[0, 1]$, we relax the function $g_s(x)$ to be a mixture of two Gaussians

$$g_s^{relax}(x_s) \propto w\mathcal{N}(0, v) + (1 - w)\mathcal{N}(1, v)\,.$$

and then we apply the NBP algorithm for computing the MAP assignment. The gaussian mixture components are centered around the values $\{0, 1\}$. The precision parameter v controls the approximation quality, as $v \to 0$ the approximation quality is higher. w is a weight parameter which weights the importance of the mixture components. In our setting, it captures our prior knowledge of fault probabilities. Figure 2 plots an example relaxation of $g_s(x)$ where $v = 0.1, w = 0.9$.

Now, we use the self potentials $g_s^{relax}(x), f_i(x)$ as an input to the NBP algorithm. Intuitively, the benefit in allowing weighted combination in the Gaussian mixtures which form the prior, is to use the weights for incorporating the prior fault occurrence into the computation. As reported in Section 3.4, this construction indeed improves the accuracy of the computation.

3.3 Local optimization procedures

Next we describe two useful heuristics proposed in [53] that are used for improving the quality of the solution: variable threshold rounding and local optimization procedure. Those mechanisms are deployed using the output of our NBP algorithm. The heuristic purpose is to obtain an integer solution out of the fractional solution.

3.3.1 Variable threshold rounding

Let $z^\star$ denote the optimal point in one of the convex relaxations. We refer to $z^\star$ as a *soft decision*, since its components can be strictly between 0 and 1. The next step is to round the soft decision

$z^\star$ to obtain a valid Boolean fault pattern (or *hard decision*). Let $\theta \in (0,1)$ and set

$$\hat{x} = \mathbf{sgn}(z^\star - \theta).$$

To create $\hat{x}$, we simply round all entries of $z_i^\star$ 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 $z^\star$. As θ varies from 0 to 1, this method generates up to n different estimates $\hat{x}$, as each entry in z falls below the threshold. We can efficiently find them all by sorting the entries of $z^\star$, and setting the values of $\hat{x}_i$ to one in the order of increasing $z_i^\star$.

We evaluate the loss for each of these, and can take as our relaxed MAP fault estimate the one that has least loss, which we denote by x^{rmap}. We can also take the K fault patterns with least loss as candidates for the ambiguity group.

3.3.2 Local optimization

Further improvement in our estimate can sometimes be obtained by a local optimization method, starting from x^{rmap}. We describe here the simplest possible local optimization method. We initialize $\hat{x}$ as x^{rmap}. We then cycle through $j = 1, \ldots, n$, at step j replacing $\hat{x}_j$ with $1 - \hat{x}_j$. If this leads to a reduction in the loss function, we accept the change, and continue. If (as usually is the case) flipping the jth bit results in an increase in l_y, we go on to the next index. We continue until we have rejected changes in all entries in $\hat{x}$. (At this point we can be sure that $\hat{x}$ is 1-OPT, which means that no change in one entry will improve the loss function.) Numerical experiments show that this local optimization method often has no effect, which means that x^{rmap} is often 1-OPT. In some cases, however, it can lead to modest reduction of loss, compared to x^{rmap}.

This local optimization method can also be used to improve our candidate ambiguity group. When we evaluate the loss of a candidate, we insert it in our list of K least loss fault patterns, whenever it is better than the worst fault pattern in the list.

We refer to the procedure of convex relaxation, followed by variable threshold rounding, and, possibly, local optimization, as *relaxed MAP (RMAP) estimation*. It is clearly not necessary to solve the RMAP to high accuracy, since we round the entries to form our fault pattern estimate.

3.4 Numerical examples

3.4.1 Algorithms for comparison

We have implemented our NBP solver using Matlab, our implementation is available on [57]. Table I lists the different algorithms implemented. Our main algorithm for comparison is the interior point method (IP) for solving the fault identification problem [53]. We have implemented two algorithms which are variants of the NBP algorithm, D. Baron *et al.* compressive sensing BP algorithm [55, 56] and the low density lattice decoder (LDLC) algorithm [20]. From the related compressive sensing domain, we have implemented three algorithms: CoSaMP [62], GPSR [63] and iterative hard thresholding (HardIO) [64]. We have further implemented the semidefinite programming relaxation algorithm of [49, 50].

Algorithm	Short name	Prior on x
NBP Solver	NBP	binary and sparsity
Newton method [53]	IP	$x \in [0,1]$
Compressive sensing Belief Propagation [55, 56]	CSBP	sparsity
Low density lattice decoder [20]	LDLC	binary
Iterative signal recovery [53]	CoSaMP	sparsity
Gradient Projection for Sparse Reconstruction [53]	GPSR	lasso
Iterative hard thresholding [53]	hardIO	sparsity
Semidefinite programming [49, 50]	SDP	$x \in [0,1]$
All zero hypothesis	NON	x is constant

Table 3.1: Algorithms for comparison

These algorithms run on one of two different yet equivalent formulations of the problem as either boolean or bipolar representation. Following, we show that both formulations are equivalent and therefore all algorithms are comparable:

$$\begin{aligned} x \in \{-1,1\}^n \quad &\Leftrightarrow \quad \overline{x} = (x+\mathbf{1})/2 \in \{0,1\}^n\,, \\ y = Ax + v \quad &\Leftrightarrow \quad \overline{y} = (2A)\overline{x} + v = y + A\mathbf{1}\,, \\ \min_x \|Ax - y\| \quad \text{s.t. } x \in \{-1,1\}^n \quad &\Leftrightarrow \quad \min_{\overline{x}} \|(2A)\overline{x} - \overline{y}\| \quad \text{s.t. } \overline{x} \in \{0,1\}^n\,. \end{aligned}$$

While NBP has the built in flexibility to work with either formulation, the linear approximation or sparsity assumptions force the other algorithms to use the boolean one. We aim to show that by incorporating both sparse knowledge and bipolar knowledge of x's nature into our solution we achieve better results than by incorporating only sparse knowledge (as done in compressive sensing) or only bipolar knowledge (as done in low density lattice decoding). The LDLC decoder utilizes the knowledge of the bipolar nature of x but gives equal probability for $x = 1, x = -1$. We will see how this assumption infers a different and indeed wrong prior probability over the x's which causes the algorithm to converge to the wrong value of x. The compressive sensing algorithms (CoSaMP, hardIO) get twice the expected number of faults $2np$ as input, resulting in always returning the largest $2np$ entries of x. However the lack of knowledge on the boolean nature leads to a wrong solution space, which may lead to a wrong solution being returned, as we will see in the results.

3.4.2 Experimental results

We consider an example with $m = 50$ sensors, $n = 100$ possible faults, and linear measurements. The elements of A are chosen randomly and independently to be either -1 or 1 with A set to be sparse with certain non-zero percentage q. We note that entries of A can be chosen to be normally distributed however this would force us to employ extrapolation techniques which, for the time being, may affect the runtime and accuracy of the algorithm. In the future, extrapolation

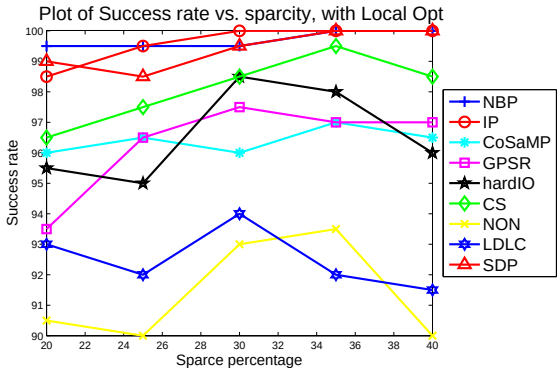

Figure 3.3: Percentage of success of the different methods used, applying local optimization heuristics

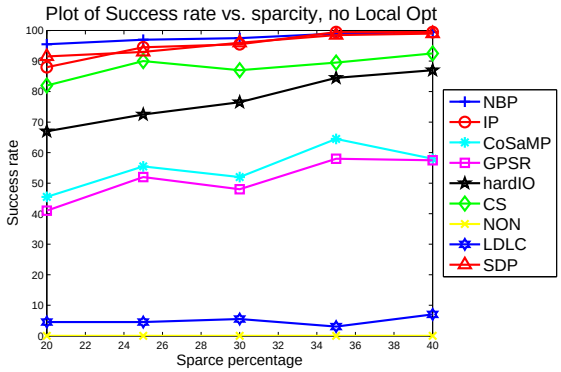

Figure 3.4: Percentage of success of the different methods used, without local optimization heuristics

techniques targeted at Gaussian extrapolation may help expand this solution with very small effect on behavior. We set the noise standard deviation to $\sigma = 1$. The fault probability is $p_j = 0.05$ for all j, which means that the expected number of faults is 5. The problem would seem to be quite challenging, since we have only 50 sparse measurements to estimate a fault pattern with 100 possible faults.

We define as a successful reconstruction a reconstruction run of an algorithm resulting in a solution with equal or better likelihood than the true solution. Empirically, the possibility of such a solution not being the true solution is experimentally very small for sparsity levels $q > 0.2$. As all solutions are based on the principle of returning the most likely candidate as x judging differently would create bias against good approximation in cases when the most Likely candidate is not the true solution. The statistical chance of the true solution not being the most likely candidate can be computed and these cases are not considered as errors by the algorithm.

3.4.3 Results discussion

Fig. 3.3,3.4 shows the success rates of the different methods tested with and without the local optimization step. The x-axis represent the sparsity level of the fault pattern matrix A, y-axis present the percentage of successful reconstruction of the correct random faults. For each sparsity level, the graphs are an average over 200 runs. As shown in both graphs, our NBP based solver gives the highest accuracy in detecting the faults, especially without the local optimization, and at very sparse levels of A. The second best algorithm is the interior point method.

Fig. 3.3shows that the success rate of all algorithms with local optimization is very high, this is an artificial result as local optimization is a powerful tool - as is shown by the "NON" line, showing success rate of the all zero hypothesis after local optimization. The local optimization also introduces a not-increasing affect into the graphs, which results in arbitrary behavior especially

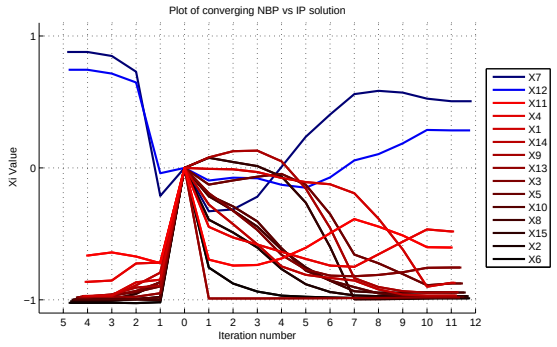

Figure 3.5: Convergence behavior of NBP vs. IP. NBP is shown to the left.

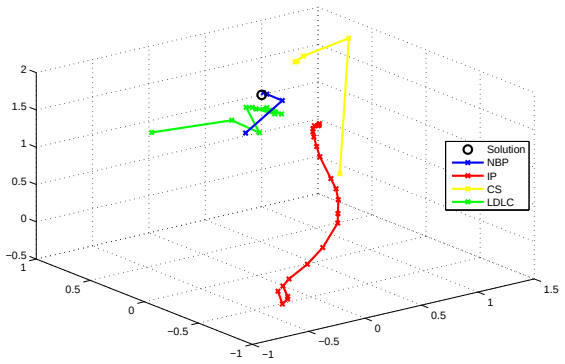

Figure 3.6: 3D plot depicting convergence of 3 faulty bits using different algorithms.

for algorithms who's success rate depends on local optimization. For this reason the rest of our discussion will focus on the no local optimization graph.

In Fig. 3.4 We can distinguish 3 groups of algorithms. First the LDLC algorithm, which has poor (< 10 percent) success rate for solution. The reason for this is that the prior distribution that the LDLC prior forces on x values is biased towards an even number of $x = 1, x = -1$ values, and therefore the solution is a MAP under a different set of probabilities. The second group is of CS algorithms, and while GPSR and CoSaMP have worse performance none of the CS algorithms cross over the 90% success rate. This behavior is caused by the fact that these algorithms also operate with partial knowledge as they are in no way aware of the boolean nature of x. We can see that CSBP has the best success rate in this group, this is due to the fact that the CSBP does hold some knowledge over the expected non-zero values. The prior consists of two Gaussian mixture components centered at 0, and the standard deviation of the wider one does in fact hold information which is more consistent with boolean solution, A wider Gaussian would result in worse performance by this CS algorithm as well.

The third group is of NBP, IP and SDP - all targeted at this specific problem and built to do well for problems both sparse and boolean in nature. The clear out performance of NBP with the fault detection prior over the other two implementations using NBP offer evidence that by including both sparse and binary knowledge we get a better performance. We can therefore state that the closer the NBP prior is to our real knowledge of the nature of x, the MAP approximation of NBP will also be closer to the real MAP. Within this group we can see note that NBP has better performance for sparse A as the solutions of NBP are consistently more likely. To better understand the differences between NBP and IP we need to compare their convergence.

Fig. 3.5 outlines convergence of our NBP solver vs. IP algorithm, in a system with $m = 10$, $n = 15$ faults. In this example two faults occurred. The x-axis represent the iteration number, where positive numbers are IP iterations and negative number are NBP iterations. y-

axis represent the intermediate value of x_i^t, the ith value in round t. Blue colors indicate faults and red colors indicate non-faulty locations. As clearly shown, after the first iteration NBP solver has a very good distinction between the faulty and non faulty indices, while the IP method is quite fluctuating until the 7th iteration. This advantage towards NBP is especially significant where communications rounds are costly since it requires fewer rounds to converge to a good solution, and even if it is stopped before it converges we see that through variable threshold rounding we would have had the correct solution.

Fig. 3.6 uses a 3D plot for demonstrating convergence of the following algorithms: NBP, CSBP, LDLC and IP in the same setting of $m = 10, n = 15$, where three faults occurred. Each of the axis indicates a different fault of the three. It can be seen that in a few iterations the NBP algorithm converges into the correct solution, while the IP method converges using many more iterations to an approximate solution. However, the computational cost per iteration of the NBP algorithm is much higher. NBP converges accurately to the vicinity of the real value of x, because the prior distribution forces it to converge to the proximity of the fault. CSBP converges to some positive value, while LDLC is again bound by the binary prior and is therefore the closer than CSBP.

Chapter 4

Conclusion and Future Work

We have shown that the LDLC decoder is a variant of the NBP algorithm. This allowed us to use current research results from the non-parametric belief propagation domain, to extend the decoder applicability in several directions. First, we have extended algorithm applicability from Latin squares to full column rank matrices (possibly non-square). Second, We have extended the LDLC convergence analysis, by discovering simpler conditions for convergence. Third, we have presented a new family of LDLC which are based on sparse encoding matrices.

We are currently working on an open source implementation of the NBP based decoder, using an undirected graphical model, including a complete comparison of performance to the LDLC decoder. Another area of future work is to examine the practical performance of the efficient Gaussian mixture product sampling algorithms developed in the NBP domain to be applied for LDLC decoder. As little is known about the convergence of the NBP algorithm, we plan to continue examining its convergence in different settings. Finally, we plan to investigate the applicability of the recent convergence fix algorithm [10] for supporting decoding matrices where the sufficient conditions for convergence do not hold.

We have also shown that by using the non-parametric belief propagation algorithm with the right priors we are able to get the most accurate solution for the fault identification problem. The algorithm is distributed and works well when the matrix A is sparse. In a communication network where the communication is costly our algorithm is preferred since it involves a significantly lower number of communication rounds.

An area of future work is to investigate the relation between NBP and interior point methods on different problems.

Bibliography

[1] D. Bickson, A. T. Ihler, H. Avissar, and D. Dolev, "A low-density lattice decoder via non-parametric belief propagation," Tech. Rep., 2009, http://arxiv.org/abs/0901.3197.

[2] D. Bickson, H. Avissar, D. Dolev, A. Zymnis, S. P. Boyd, and A. T. Ihler, "Distributed fault identification via non-parametric belief propagation," *, to appear*, 2009.

[3] F. Kschischang, B. Frey, and H. A. Loeliger, "Factor graphs and the sum-product algorithm," vol. 47, pp. 498–519, Feb. 2001.

[4] R. J. McEliece, D. J. C. MacKay, and J. F. Cheng, "Turbo decoding as an instance of Pearl's 'belief propagation' algorithm," vol. 16, pp. 140–152, Feb. 1998.

[5] Y. Weiss and W. T. Freeman, "Correctness of belief propagation in Gaussian graphical models of arbitrary topology," *Neural Computation*, vol. 13, no. 10, pp. 2173–2200, 2001.

[6] D. Bickson, O. Shental, P. H. Siegel, J. K. Wolf, and D. Dolev, "Linear detection via belief propagation," in *Proc. 45th Allerton Conf. on Communications, Control and Computing*, Monticello, IL, USA, Sept. 2007.

[7] O. Shental, D. Bickson, P. H. Siegel, J. K. Wolf, and D. Dolev, "Gaussian belief propagation solver for systems of linear equations," in *IEEE International Symposium on Information Theory (ISIT)*, Toronto, Canada, July 2008.

[8] ——, "Gaussian belief propagation for solving systems of linear equations: Theory and application," in *IEEE Transactions on Information Theory, submitted for publication*, June 2008.

[9] ——, "A message-passing solver for linear systems," in *Information Theory and Applications (ITA) Workshop*, San Diego, CA, USA, January 2008.

[10] J. K. Johnson, D. Bickson, and D. Dolev, "Fixing convergence of Gaussian belief propagation," in *IEEE International Symposium on Information Theory (ISIT)*, Seoul, South Korea, 2009.

[11] D. Bickson, "Gaussian belief propagation: Theory and application," Ph.D. dissertation, The Hebrew University of Jerusalem, October 2008.

[12] E. Sudderth, A. Ihler, W. Freeman, and A. Willsky, "Nonparametric belief propagation," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2003.

[13] A. Ihler, E. Sudderth, W. Freeman, and A. Willsky, "Efficient multiscale sampling from products of gaussian mixtures," in *Neural Information Processing Systems (NIPS)*, Dec. 2003.

[14] M. Briers, A. Doucet, and S. S. Singh, "Sequential auxiliary particle belief propagation," in *International Conference on Information Fusion*, 2005, pp. 705–711.

[15] D. Rudoy and P. J. Wolf, "Multi-scale MCMC methods for sampling from products of Gaussian mixtures," in *IEEE International Conference on Acoustics, Speech and Signal Processing*, vol. 3, 2007, pp. III–1201–III–1204.

[16] A. T. Ihler. Kernel Density Estimation Toolbox for MATLAB [online] `http://www.ics.uci.edu/~ihler/code/`.

[17] A. T. Ihler, Fisher, R. L. Moses, and A. S. Willsky, "Nonparametric belief propagation for self-localization of sensor networks," *Selected Areas in Communications, IEEE Journal on*, vol. 23, no. 4, pp. 809–819, 2005.

[18] A. T. Ihler, J. W. Fisher, and A. S. Willsky, "Particle filtering under communications constraints," in *Statistical Signal Processing, 2005 IEEE/SP 13th Workshop on*, 2005, pp. 89–94.

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

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

[21] B. Kurkoski and J. Dauwels, "Message-passing decoding of lattices using Gaussian mixtures," in *IEEE Int. Symp. on Inform. Theory (ISIT)*, Toronto, Canada, July 2008.

[22] Y. Yona and M. Feder, "Efficient parametric decoder of low density lattice codes," in *IEEE International Symposium on Information Theory (ISIT)*, Seoul, S. Korea, July 2009.

[23] B. M. Kurkoski, K. Yamaguchi, and K. Kobayashi, "Single-Gaussian messages and noise thresholds for decoding low-density lattice codes," in *IEEE International Symposium on Information Theory (ISIT)*, Seoul, S. Korea, July 2009.

[24] D. Viassolo, S. Adibhatla, B. Brunell, J. Down, N. Gibson, A. Kumar, H. Mathews, and L. Holcomb, "Advanced estimation for aircraft engines," in *Proceedings of the American Control Conference*, 2007, pp. 2807–2821.

[25] S. Ganguli, S. Deo, and D. Gorinevsky, "Parametric fault modeling and diagnostics of a turbofan engine," in *Proceedings of the IEEE International Conference on Control Applications*, 2004, pp. 223–228.

[26] 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, 1991.

[27] 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, 1993.

[28] C. Hood and C. Ji, "Proactive network-fault detection," *IEEE Transactions on Reliability*, vol. 46, no. 3, pp. 333–341, 1997.

[29] F. Feather, D. Siewiorek, and R. Maxion, "Fault detection in an ethernet network using anomaly signature matching," *Applications, Technologies, Architectures, and Protocols for Computer Communication*, pp. 279–288, 1993.

[30] 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.

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

[32] 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, 1990.

[33] A. Saberi, A. Stoorvogel, and P. Sannuti, "Fundamental problems in fault detection and identification," *International Journal of Robust Nonlinear Control*, vol. 10, pp. 1209–1236, 2000.

[34] U. Lerner, R. Parr, D. Koller, and G. Biswas, "Bayesian fault detection and diagnosis in dynamic systems," in *Proceedings of the American Association of Artificial Intelligence*, 2000, pp. 531–537.

[35] M. Morari, A. Bemporad, and D. Mignone, "A framework for control, state estimation, fault detection, and verification of hybrid systems," *Scientific Computing in Chemical Engineering II*, vol. 2, pp. 46–61, 1999.

[36] R. Isermann, "Process fault detection based on modeling and estimation methods," *Automatica*, vol. 20, no. 4, pp. 387–404, 1984.

[37] J. Gertler, *Fault Detection and Diagnosis in Engineering Systems*. Marcel Dekker, Inc., 1998.

[38] G. Goodwin, M. Seron, and J. D. Doná, *Constrained Control and Estimation: An Optimisation Approach*. Springer, 2004.

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

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

[41] F. Tu, K. Pattipati, S. Deb, and V. Malepati, "Multiple fault diagnosis in graph-based systems," in *Proceedings of SPIE, Component and Systems Diagnostics, Prognostics, and Health Management II*, 2002, pp. 168–179.

[42] ——, "Computationally efficient algorithms for multiple fault diagnosis in large graph-based systems," *IEEE Transactions on Systems, Man and Cybernetics, Part A*, vol. 33, no. 1, pp. 73–85, 2003.

[43] D. Donoho, "Compressed sensing," *IEEE Transactions on Information Theory*, vol. 52, no. 4, pp. 1289–1306, 2006.

[44] R. Tibshirani, "Regression shrinkage and selection via the lasso," *Journal of the Royal Statistical Society. Series B (Methodological)*, vol. 58, no. 1, pp. 267–288, 1996.

[45] J. Tropp, "Just relax: convex programming methods for identifying sparse signals in noise," *IEEE Transactions on Information Theory*, vol. 52, no. 3, pp. 1030–1051, 2006.

[46] J. Feldman, D. Karger, and M. Wainwright, "LP decoding," in *Proceedings of the 41st Allerton Conference on Communications, Control, and Computing*, Monticello, Illinois, USA, October 2003, pp. 1–3.

[47] D. Bickson, O. Shental, P. H. Siegel, J. K. Wolf, and D. Dolev, "Gaussian belief propagation based multiuser detection," in *IEEE International Symposium on Information Theory (ISIT)*, Toronto, Canada, July 2008.

[48] A. Montanari, B. Prabhakar, and D. Tse, "Belief propagation based multi-user detection," in *Proc. 43th Allerton Conf. on Communications, Control and Computing*, Monticello, IL, USA, Sept. 2005.

[49] P. Tan and L. Rasmussen, "The application of semidefinite programming for detection in CDMA," *IEEE Journal on Selected Areas in Communications*, vol. 19, no. 8, pp. 1442–1449, 2001.

[50] 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 Processing Letters*, vol. 9, no. 6, pp. 165–167, 2002.

[51] M. Kisialiou and Z. Q. Luo, "Performance analysis of quasi-maximum-likelihood detector based on semi-definite programming," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, vol. 3, Philadelphia, PA, 2005.

[52] P. H. Tan and L. K. Rasmussen, "Multiuser detection in CDMA – a comparison of relaxations, exact, and heuristic search methods," *IEEE Transactions on Wireless Communications*, vol. 3, no. 5, pp. 1802–1809, 2004.

[53] S. Joshi and S. Boyd, "Sensor selection via convex optimization," *IEEE Transactions on Signal Processing*, vol. 57, no. 2, pp. 451–462, February 2009.

[54] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge University Press, March 2004.

[55] S. Sarvotham, D. Baron, and R. G. Baraniuk, "Compressed sensing reconstruction via belief propagation," Rice University, Houston, TX, Tech. Rep. TREE0601, July 2006.

[56] D. Baron, S. Sarvotham, and R. G. Baraniuk, "Bayesian compressive sensing via belief propagation," *IEEE Trans. Signal Processing, to appear*, 2009.

[57] Gaussian Belief Propagation implementation in matlab [online] `http://www.cs.huji.ac.il/labs/danss/p2p/gabp/`.

[58] J. Pearl, *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. San Francisco: Morgan Kaufmann, 1988.

[59] A. T. Ihler, J. W. F. III, and A. S. Willsky, "Loopy belief propagation: Convergence and effects of message errors," *Journal of Machine Learning Research*, vol. 6, pp. 905–936, May 2005.

[60] D. M. Malioutov, J. K. Johnson, and A. S. Willsky, "Walk-sums and belief propagation in Gaussian graphical models," *Journal of Machine Learning Research*, vol. 7, Oct. 2006.

[61] G. Poltyrev, "On coding without restrictions for the AWGN channel," in *IEEE Trans. Inform. Theory*, vol. 40, Mar. 1994, pp. 409–417.

[62] D. Needell and J. A. Tropp, "Cosamp: Iterative signal recovery from incomplete and inaccurate samples," Apr 2008. [Online]. Available: http://arxiv.org/abs/0803.2392

[63] M. Figueiredo, R. Nowak, and S. Wright, "Gradient projection for sparse reconstruction: application to compressed sensing and other inverse problems," in *IEEE Journal of Selected Topics in Signal Processing: Special Issue on Convex Optimization Methods for Signal Processing*, vol. 1, no. 4, 2007, pp. 586–598.

[64] T. Blumensath and M. E. Davies, "Iterative hard thresholding for compressed sensing," *ArXiv e-prints*, May 2008. [Online]. Available: http://arxiv.org/abs/0805.0510