TP Decoding

Yi Lu, Cyril Méasson and Andrea Montanari

***Abstract*— 'Tree pruning' (TP) is an algorithm for probabilistic inference on binary Markov random fields. It has been recently derived by Dror Weitz and used to construct the first fully polynomial approximation scheme for counting independent sets up to the 'tree uniqueness threshold.' It can be regarded as a clever method for pruning the belief propagation computation tree, in such a way to exactly account for the effect of loops.**

In this paper we generalize the original algorithm to make it suitable for decoding linear codes, and discuss various schemes for pruning the computation tree. Further, we present the outcomes of numerical simulations on several linear codes, showing that tree pruning allows to interpolate continuously between belief propagation and maximum a posteriori decoding. Finally, we discuss theoretical implications of the new method.

I. Introduction

Statistical inference is the task of computing marginals (or expectation values) of complex multi-variate distributions. *Belief propagation* (BP) is a generic method for accomplishing this task quickly but approximately, when the multivariate distribution factorizes according to a sparse graphical structure. The advent of sparse graph codes and iterative BP decoding [1] has naturally made decoding become an important case of this general problem. The present paper builds on this connection by 'importing' an algorithm that has been recently developed in the context of approximate counting and inference [2].

We will refer to the new algorithm as *tree pruning* (TP) *decoding*. For a number of reasons the application of this method to decoding is non-trivial. However, it is an interesting approach for the three following reasons. (i) It provides a sequence of decoding schemes that interpolates continuously between BP and the optimal maximum a posteriori (MAP) decoding. (ii) At each level of this sequence, the effect of loops of increasing length is taken into account. (iii) We expect that an appropriate truncation of this sequence might yield a polynomial algorithm for MAP decoding on general graphs of bounded degree, for low enough noise levels. Preliminary numerical results are encouraging.

A. Qualitative Features and Relation to BP Decoding

As for BP decoding, TP decoding aims at estimating the a posteriori marginal probabilities of the codeword bits. Unhappily, the relation between BP estimates and the actual marginals is in general poorly understood. In the case of random Low-Density Parity-Check (LDPC) codes and communication over memoryless channels, density evolution allows to show that, at small enough noise level, the BP bit error probability becomes arbitrarily small if the blocklength is large enough. This implies that the distance between BP estimates and the actual marginals vanishes as well. This result is not completely satisfactory, in that it relies in a crucial way on the locally tree-like structure of sparse random graphs. This property does not hold for structured graphs, and, even for large graphs, it kicks in only at very large blocklengths.

In contrast to this, the algorithm considered in this paper accounts systematically for short loops. It should therefore produce better performances, in particular in the error floor regime since this is dominated by small error events [3].

A convenient way of understanding the difference between BP and MAP decoding makes use of the so-called computation tree. Consider a code described by a factor graph $G = (V, F, E)$ whereby V represents the variable nodes, F the factor nodes, and E the edges. Let $i \in V$, then the corresponding computation tree denoted by $\mathsf{T}(i)$ is the tree of non-reversing walks in G that start at i. This gives a graph (tree) structure in a natural way: two nodes are neighbors if one is reached from the other adding a step.

BP uses the marginal at the root of $\mathsf{T}(i)$ as an estimate for the marginal distribution at i on the original graph G. If G contains short loops in the neighborhood of i, the computation tree differs from G in a neighborhood of the root and, as a consequence, the BP estimate can differ vastly from the actual marginal.

Weitz [2] made the surprising remark that there exists a simple way of pruning the computation tree (and fixing some of its variables) in such a way that the resulting root marginal coincides with the marginal on G. Unhappily the size of the pruned tree, which we call the *self-avoiding walk tree* and denote by $\mathsf{SAW}(i)$, is exponential in the size of the original graph G. Nevertheless, the tree can be truncated thus yielding a convergent sequence of approximations for the marginal at i. The complexity of the resulting algorithm is linear in the size of the truncated tree. Its efficiency depends on how sensitive is the root marginal to the truncation depth.

B. Contributions and Outline

Applying this approach to decoding linear codes poses several challenges:

(i) Weitz's construction is valid only for Markov random fields (MRFs) with pairwise interactions and binary variables. The decoding problem does not fit this framework.

Yi Lu is with the Department of Electrical Engineering, Stanford University, yi.lu@stanford.edu. Cyril Méasson is with the Math Center, Bell Labs, Alcatel-Lucent, Murray Hill, cyril.measson@gmail.com. Andrea Montanari is with Departments of Electrical Engineering and Statistics, Stanford University, montanari@stanford.edu.

(ii) The original justification for truncating the self-avoiding walk tree followed the so-called 'strong spatial mixing' or 'uniqueness' condition. This amounts to saying that the conditional marginal at the root given the variables at depth t, depends weakly on the values of the latter. This is (most of the times) false in decoding. For a 'good' code, the value of bit i in a codeword is completely determined by the values of bits outside a finite neighborhood around i.

(iii) Even worse, we found in numerical simulations that the original truncation procedure performs poorly in decoding.

The self-avoiding walk tree construction has already motivated several applications and generalizations in the past few months. Jung and Shah [5] discussed its relation with BP, and proposed a distributed implementation. Mossel and Sly [6] used it to estimate mixing times of Monte Carlo Markov chain algorithms. Finally, and most relevant to the problems listed above, Nair and Tetali [4] proposed a generalization to non-binary variables and multi-variable interactions. While this generalizations does in principle apply to decoding, its complexity grows polynomially in the tree size. This makes it somewhat unpractical in the present context.

In this paper we report progress on the three points above. Specifically, in Section II we use duality to rephrase decoding in terms of a generalized binary Markov random field. We then show how to generalize the self-avoiding walk tree construction to this context. In Section III we discuss the problems arising from the original truncation procedure, and describe two procedure that show better performances. Numerical simulations are presented in Section IV. Finally, one of the most interesting perspectives is to use TP as a tool for analyzing BP and, in particular, comparing it with MAP decoding. Some preliminary results in this direction are discussed in Section V.

We should stress that a good part of our simulations concerns the binary erasure channel (BEC). From a practical point of view, TP decoding is not an appealing algorithm in this case. In fact, MAP decoding can be implemented in polynomial time through, for instance, Gaussian elimination. The erasure channel is nevertheless a good starting point for several reasons. (i) Comparison with MAP decoding is accessible. (ii) We can find a particularly simple truncation scheme in the erasure case. (iii) Some subtle numerical issues that exist for general channels disappear for the BEC.

II. Decoding through the Self-Avoiding Walk Tree

Throughout this paper we consider binary linear codes of blocklength n used over a binary-input memoryless channel. Let $\text{BM}(\epsilon)$, where ϵ is a noise parameter, denote a generic channel. Assume that $\mathcal{Y}$ is the output alphabet and let $\{Q(y|x) : x \in \{0,1\}, y \in \mathcal{Y}\}$ denote its transition probability.

With a slight abuse of terminology we shall identify a code with a particular parity-check matrix $\mathbb{H}$ that represents it,

$$\mathfrak{C} = \{\underline{x} \in \{0,1\}^n : \mathbb{H}\underline{x} = \underline{0} \mod 2\}.$$

Therefore, the code is further identified with a Tanner graph $G = (V, F, E)$ whose adjacency matrix is the parity-check matrix $\mathbb{H}$. We will denote by $\partial a \overset{\text{def}}{=} \{i \in V : (i,a) \in E\}$ the neighborhood of function (check) node a, and write $\partial a = (i_1(a), \dots, i_{k(a)}(a))$. Analogously, $\partial i \overset{\text{def}}{=} \{i \in V : (i,a) \in E\}$ indicates the neighborhood of the variable node i. The conditional distribution for the channel output $\underline{y}$ given the input $\underline{x}$ factorizes according to the graph G (also called factor graph). It follows immediately from Bayes rule that $\mathbb{P}\{\underline{X} = \underline{x} | \underline{Y} = \underline{y}\} = \mu^y(\underline{x})$, where

$$\mu^y(\underline{x}) = \frac{1}{Z(\underline{y})} \prod_{i \in V} Q(y_i|x_i) \prod_{a \in F} \mathbb{I}(\textstyle\sum_{j=1}^{k(a)} x_{i_j(a)} = 0 \mod 2). \tag{1}$$

We denote by $\mu_i^y(x_i) = \mathbb{P}\{X_i = x_i | \underline{Y} = \underline{y}\}$ the marginal distribution at bit i. *Symbol MAP decoding* amounts to the following prescription,

$$\hat{x}_i^{\text{MAP}}(\underline{y}) = \arg \max_{x_i \in \{0,1\}} \mu_i^y(x_i).$$

Both BP and TP decoders have the same structure, whereby the marginal $\mu_i^y(\,\cdot\,)$ is replaced by its approximation, respectively $\nu_i^{\text{BP}}(\,\cdot\,)$ or $\nu_i^{\text{TP}}(\,\cdot\,)$.

A. Duality and Generalized Markov Random Field

We call a *generalized Markov Random Field* (gMRF) over the finite alphabet $\mathcal{X}$ a couple $(\mathcal{G}, \psi)$, where $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is an ordinary graph over vertex set $\mathcal{V}$, and edge set $\mathcal{E}$. Further $\psi = \{\psi_{ij} : (i,j) \in \mathcal{E};\ \ \psi_i : i \in \mathcal{V}\}$ is a set of weights indexed by edges and vertices in $\mathcal{G}$, $\psi_{ij} : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$, $\psi_i : \mathcal{X} \to \mathbb{R}$. Notice that, unlike for ordinary MRFs, the edge weights in generalized MRFs are <u>not</u> required to be non-negative.

Given a subset $A \subseteq \mathcal{V}$, the *marginal* of the gMRF $(\mathcal{G}, \omega)$ on A, is defined as the function $\omega_A : \mathcal{X}^A \to \mathbb{R}$, with entries

$$\omega_A(\underline{x}_A) = \sum_{\{x_j : j \notin A\}} \prod_{(l,k) \in \mathcal{E}} \psi_{lk}(x_l, x_k) \prod_{l \in \mathcal{V}} \psi_l(x_l). \tag{2}$$

When $A = \mathcal{V}$, we shall omit the subscript and call $\omega(\underline{x})$ the *weight* of configuration $\underline{x}$. More generally, the *expectation* of a function $f : \mathcal{X}^{\mathcal{V}} \to \mathbb{R}$ can be defined as

$$\omega(f) = \sum_{\underline{x}} f(\underline{x}) \prod_{(l,k) \in \mathcal{E}} \psi_{lk}(x_l, x_k) \prod_{l \in \mathcal{V}} \psi_l(x_l). \tag{3}$$

Notice that $\omega(\,\cdot\,)$ is not (and in general cannot be) normalized. In the sequel, whenever the relevant MRF has non-negative weights and is normalizable, we shall use words 'expectation' and 'marginal' in the usual (normalized) sense.

Duality can be used to reformulate decoding (in particular, the posterior marginals $\mu_i^y(x_i)$) in terms of a gMRF. More precisely, given a code with Tanner graph $G = (V, F, E)$, we define a gMRF on graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ where $\mathcal{V} = (V, F)$ and $\mathcal{E} = E$, proceeding as follows. We let $\mathcal{X} = \{0,1\}$ and associate variables $x_i \in \{0,1\}$ to $i \in V$ and $n_a \in \{0,1\}$ to $a \in F$. We then introduce the weights

$$\forall i \in V,\ \psi_i(x_i) = Q(y_i|x_i), \quad \forall a \in F,\ \psi_a(n_a) = 1, \tag{4}$$

$$\forall (i,a) \in E,\ \psi_{ai}(n_a, x_i) = (-1)^{n_a x_i}. \tag{5}$$

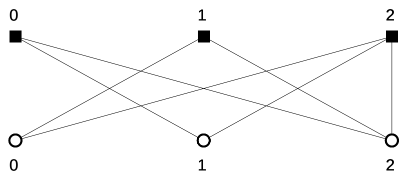


Fig. 1. Tanner graph for a repetition code of length 3.

Although next statement follows from general duality theory, it is convenient to spell it out explicitly.

Lemma 1. *The marginals of the a posteriori distribution defined in Eq. (1) are proportional to the ones of the gMRF defined in Eq. (4) and Eq. (5). More precisely, we get* $\mu_i^y(x_i) = \omega_i(x_i)/[\omega_i(0) + \omega_i(1)]$.

Proof. It is immediate to prove a stronger result, namely that the distribution $\mu^y(\underline{x})$ is proportional to $\sum_{\underline{n}} \omega(\underline{x}, \underline{n})$, where $\omega(\underline{x}, \underline{n})$ is defined using Eq. (4) and Eq. (5). We have

$$\begin{aligned}\sum_{\underline{n}} \omega(\underline{x}, \underline{n}) &= \sum_{\underline{n}} \prod_{i\in V} Q(y_i|x_i) \prod_{(i,a)\in E} (-1)^{x_i n_a} \\ &= \prod_{i\in V} Q(y_i|x_i) \prod_{a\in F} \sum_{n_a\in\{0,1\}} (-1)^{n_a \sum_{i\in\partial a} x_i} \\ &= \prod_{i\in V} Q(y_i|x_i) \prod_{a\in F} 2\, \mathbb{I}(\textstyle\sum_{i\in\partial a} x_i = 0 \mod 2)\,,\end{aligned}$$

which is proportional to the right-hand side of Eq. (1). □

This result, which derives from [9] and [8], motivates us to extend Weitz's construction to gMRFs.[1] This is the object of the next section.

B. The Self-Avoiding Walk Tree for Generalized Markov Random Field

Assume we are given a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ and a node $i \in \mathcal{V}$. We have already described the computation tree rooted at i, which we denote by $\mathsf{T}(i)$.

An 'extended self-avoiding walk' (SAW) on a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, starting at $i \in \mathcal{V}$ is a non-reversing walk that never visits twice the same vertex, except, possibly, for its end-point. The 'self-avoiding walk tree' rooted at $i \in \mathcal{V}$ is the tree of all extended self-avoiding walks on $\mathcal{G}$ starting at i. It is straightforward to see that $\mathsf{SAW}(i)$ is in fact a finite sub-tree of $\mathsf{T}(i)$. Its size is bounded by $(\Delta - 1)^{|\mathcal{V}|}$, where Δ is the maximum degree of $\mathcal{G}$, and $|\mathcal{V}|$ the node number.

As an example, Figure 2 shows a SAW tree for the small graph $\mathcal{G}$ depicted in Figure 1. (In this case, $\mathcal{G}$ is the Tanner graph of a repetition code of length 3.) If we denote by $\mathcal{V}(i)$ the vertex set of $\mathsf{SAW}(i)$, there exists a natural projection $\pi : \mathcal{V}(i) \to \mathcal{V}$ that preserves edges. Formally, π maps a self-avoiding walk to its end-point.

Notice that $\mathsf{SAW}(i)$ has two types of leaf nodes: (i) Nodes that are leaves in the original graph $\mathcal{G}$. (ii) Nodes that are

[1]More explicitely, we can think of implementing a binary *Fourier* involution as proposed first in [9] and [8] on the graph edges (as later reported in Eq. (13)), while processing all graph vertices in a similar way.

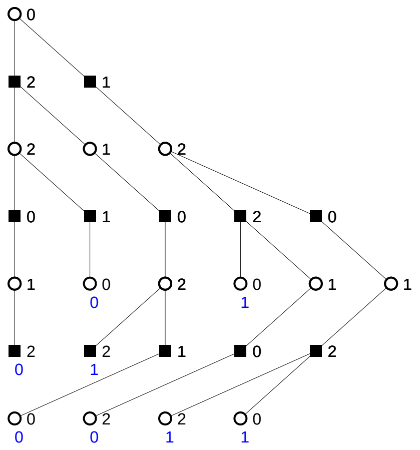


Fig. 2. Self-avoiding walk tree $\mathsf{SAW}(i)$ for the Tanner graph of Figure 1, rooted at variable node $i = 0$. In this picture, each node of the self-avoiding tree is labeled by its projection onto $\mathcal{V}$. At 'terminated' nodes we marked the value that the variable is forced to take.

not leaves in $\mathcal{G}$ but corresponds to extended self-avoiding walks that cannot be further continued. The latter case arises when the endpoint of the self-avoiding walk has already been visited (i.e., when a loop is closed). We shall refer to nodes of the second type as *terminated* nodes. Indeed, the self-avoiding walk tree $\mathsf{SAW}(i)$ can be obtained from $\mathsf{T}(i)$ by the following *termination* procedure. Imagine descending $\mathsf{T}(i)$ along one of its branches. When the same projection is encountered for the second time, terminate the branch. Formally, this means eliminating all the descendants of u whenever $\pi(u) = \pi(v)$ for some ancestor v of u.

Given a gMRF $(\mathcal{G}, \psi)$, we can define a gMRF on $\mathsf{SAW}(i)$ in the usual way. Namely, to any edge $(u, v) \in \mathsf{SAW}(i)$, we associate a weight coinciding with the one of the corresponding edge in $\mathcal{G}$: $\psi_{u,v}(x_u, x_v) = \psi_{\pi(u),\pi(v)}(x_u, x_v)$. The analogous definition is repeated for any non-terminated node: $\psi_u(x_u) = \psi_{\pi(u)}(x_u)$. Finally, the choice of weight on terminated nodes makes use of the hypothesis that $\mathcal{X} = \{0, 1\}$. Assume that the edges of $\mathcal{G}$ are labeled using a given order, e.g., a lexicographic order. Let u be a terminated node with $\pi(u) = j$. Then the self-avoiding walk corresponding to u contains a loop that starts and ends at j. We let $\psi_u(x_u) = \mathbb{I}(x_u = 0)$ (respectively, $\psi_u(x_u) = \mathbb{I}(x_u = 1)$) if this loop quits j along an edge of higher (respectively, lower) order than the one along which it enters j. The relevance of this construction is due to Weitz who considered[2] the case of *permissive* binary MRFs. By this we mean that $\psi_{kl}(x_k, x_l) \geq 0$, $\psi_k(x_k) \geq 0$, and, for any $k \in \mathcal{V}$, there exists $x_k^* \in \{0, 1\}$, such that $\psi_k(x_k^*) > 0, \psi_{kl}(x_k^*, x_l) > 0$ for any l with $(k, l) \in \mathcal{E}$ and $x_l \in \{0, 1\}$. (The latter is referred to as the 'permissivity' condition.)

[2]Weitz [2] considered the *independent set problem* but remarked that his construction generalized to a larger class of MRFs. Jung and Shah [5] studied this generalization. Nair and Tetali [4] discussed the case of 'hard-core' interactions (positively alignable) as well.

Proposition 1 (Weitz). *Given a* $\underline{\text{permissive}}$ *binary MRF* $(\mathcal{G}, \psi)$*, the marginal of* x_i *with respect to* $(\mathcal{G}, \psi)$ *is proportional to the root marginal on* $\mathsf{SAW}(i)$.

The problem with non-permissive MRFs and, *a fortiori*, with generalized MRFs, is that the tree model $\mathsf{SAW}(i)$ may not admit any assignment of the variables such that all the weights $\psi_l(x_l)$, $\psi_{kl}(x_k, x_l)$ are non-negative. As a consequence the MRF on $\mathsf{SAW}(i)$ does not define a probability distribution and this invalidates the derivation in [2] or [5]. Even worse, the procedure used in these papers was based in keeping track of ratios among marginals, of the form $R_i = \mu_i(x_i = 0)/\mu_i(x_i = 1)$. When the MRF does not define a distribution, ill-defined ratios such as $0/0$ can appear.

Let us stress that this problem is largely due to the 'termination' procedure described above. This in fact constrains the set of assignments with non-vanishing weight to be compatible with the values assigned at terminated nodes.

In order to apply the self-avoiding walk construction to gMRFs, we need to modify it in the two following ways.

(i) We add further structure to $\mathsf{SAW}(i)$. For any $u \in \mathcal{V}(i)$, let $\mathsf{D}(u)$ be the set of its *children* (i.e., the set of extended self-avoiding walks that are obtained by adding one step to u). Then we partition $\mathsf{D}(u) = \mathsf{D}_1(u) \cup \cdots \cup \mathsf{D}_k(u)$ as follows. Let $v_1, v_2 \in \mathsf{D}(u)$ be two children of u, and write them as $v_1 = (u, j_1)$, $v_2 = (u, j_2)$. Further, let $j = \pi(u)$. Then we write $v_1 \sim v_2$ if there exists an extended self-avoiding walk of the form (u, j_1, u', j_2, j). Here we are regarding u, u' as walks on $\mathcal{G}$ (i.e., sequences of vertices) and we use $(u, v, w, \dots)$ to denote the concatenation of walks. It is not difficult to verify that $\sim$ is an equivalence relation. The partition $\{\mathsf{D}_1(u), \dots, \mathsf{D}_k(u)\}$ is defined to be the partition in equivalence classes under this relation.

(ii) We define the *generalized root marginal* of $\mathsf{SAW}(i)$ through a recursive procedure that makes it always well-defined. First notice that, if $\mathcal{G}$ is a tree rooted at i, then the marginal at i can be computed by a standard message passing (dynamic programming) procedure, starting from the leaves and moving up to the root. The update rules are, for $u \in \mathsf{D}(w)$,

$$\omega_{u\to w}(x_u) \;=\; \psi_u(x_u) \prod_{v \in \mathsf{D}(u)} \widehat{\omega}_{v\to u}(x_u)\,, \tag{6}$$

$$\widehat{\omega}_{v\to u}(x_u) \;=\; \sum_{x_v} \psi_{uv}(x_u, x_v)\, \omega_{v\to u}(x_v)\,, \tag{7}$$

where edges are understood to be directed towards the root. The marginal at the root is obtained by evaluating the right hand side of Eq. (6) with $u = i$.

The generalized root marginal is defined by the same procedure but changing Eq. (6) as follows. Given the partition $\mathsf{D}(u) = \mathsf{D}_1(u) \cup \cdots \cup \mathsf{D}_k(u)$ described above, we let

$$\omega_{u\to w}(x_u) \;=\; \psi_u(x_u) \prod_{l=1}^{k} \widehat{\omega}_{\mathsf{D}_l(u)}(x_u)\,, \tag{8}$$

where we define $\widehat{\omega}_{\mathsf{D}_l(u)}(x_u)$ through a *concatenation* procedure. Let $(\widehat{\omega}^{(1)}(\,\cdot\,), \dots, \widehat{\omega}^{(k)}(\,\cdot\,))$ be the set of messages $\{\widehat{\omega}_{v\to u}(\,\cdot\,) \,:\, v \in \mathsf{D}(u)\}$ ordered according to the order of edges $(\pi(v), \pi(u))$ in $\mathcal{G}$. Then we let

$$(\widehat{\omega}_{\mathsf{D}_l(u)}(0), \widehat{\omega}_{\mathsf{D}_l(u)}(1)) \overset{\text{def}}{=} (\widehat{\omega}^{(1)}(0), \widehat{\omega}^{(k)}(1))\,. \tag{9}$$

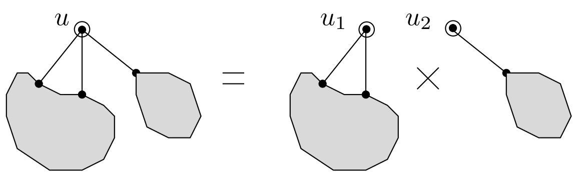
The reason for calling this a 'concatenation' follows from the remark that, with the notations above, we have $\widehat{\omega}^{(1)}(1) = \widehat{\omega}^{(2)}(0)$, $\widehat{\omega}^{(2)}(1) = \widehat{\omega}^{(3)}(0)$, etc. We refer to the discussion (and proof) below for a justification of this claim. As a consequence, the procedure in Eq. (9) can be described as follows: write the components of $\widehat{\omega}^{(1)}(\,\cdot\,), \widehat{\omega}^{(2)}(\,\cdot\,) \cdots, \widehat{\omega}^{(k)}(\,\cdot\,)$ in sequence, and eliminate repeated entries.

With this groundwork, we obtain the following generalization of Weitz's result.

Proposition 2. *Given a gMRF* $(\mathcal{G}, \psi)$*, the marginal at* $i \in \mathcal{V}$ *with respect to* $(\mathcal{G}, \psi)$ *is equal to the generalized root marginal on* $\mathsf{SAW}(i)$.

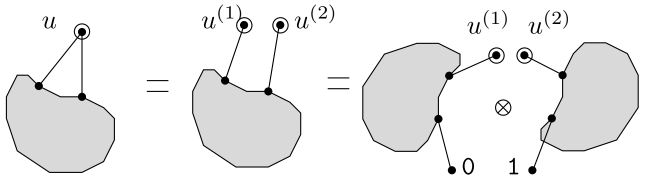
Proof. The proof is very similar to Weitz's original proof in [2]; the difference is that special care must be paid to avoid ill-defined expressions. The argument consists in progressively simplifying the graph $\mathcal{G}$ (rooted at i) until $\mathsf{SAW}(i)$ is obtained. We shall represent these simplifications graphically.

Consider the first step, corresponding to Eq. (8), with $u = i$. The partition of $\mathsf{D}(u)$ in $\{\mathsf{D}_1(u), \dots \mathsf{D}_k(u)\}$, corresponds to a partition of of the subgraph $\mathcal{G} \setminus i$ (obtained by eliminating from $\mathcal{G}$, i as well as its adjacent vertices) into connected components. This correspondence is depicted below (whereby gray blobs correspond to connected subgraphs). After factoring out the term $\psi_u(x_u)$, the definition of



marginal in Eq. (2) factorizes naturally on such components, leading to Eq. (8)

Consider now one of such components, call it $\mathcal{G}_1$, such as the one depicted below. The corresponding generalized root marginal is computed using the concatenation rule, specified in Eq. (9). In order to derive this rule, first consider the graph



$\mathcal{G}_1'$ obtained from $\mathcal{G}_1$ by replacing its root u by $k = \deg(u)$ copies $u^{(1)}, \dots, u^{(k)}$, each of degree 1 (here $\deg(v)$ denotes the degree of vertex v). Each of the newly introduced vertices is adjacent to one of the edges incident on the root in $\mathcal{G}_1$. Further $u^{(1)}, \dots, u^{(k)}$ are labeled according to the ordering (chosen at the beginning of the reduction procedure) on the adjacent edges. These k nodes will be referred to as 'split nodes' in the sequel.

From the definition of marginal in Eq. (2), and using the notation ω' for the gMRF on $\mathcal{G}_1'$, we have

$$\omega_u(x) = \omega'_{u^{(1)}\dots u^{(k)}}(\underbrace{x\cdots\cdots x}_{k}), \tag{10}$$

for $x \in \{0,1\}$. This identity is represented as the first equality in the figure above.

Next we replace the graph $\mathcal{G}_1'$ by k copies of it, $\mathcal{H}_1,\dots,\mathcal{H}_k$. With a slight abuse of notation, we re-name $u^{(1)}$ the first of the k 'split nodes' in $\mathcal{H}_1$, $u^{(2)}$ the second in $\mathcal{H}_2$, and so on. Further we add node weights to the other 'split nodes,' (i.e., the ones that remained un-named), either of the form $\psi_v(x_v) = \mathbb{I}(x_v = 0)$ (forcing x_v to take value 0) or of the form $\psi_v(x_v) = \mathbb{I}(x_v = 1)$ (forcing x_v to take value 1). More precisely, for any $j \in \{1,\dots,k\}$ on $\mathcal{H}_j$ we force to 0 those split nodes that come before $u^{(j)}$, and to 1 the ones that come after.

As a consequence, if we use $\omega^{(j)}$ for the gMRF $\mathcal{H}^{(j)}$, we have

$$\omega^{(j)}_{u^{(j)}}(x) = \omega'_{u^{(1)}\dots u^{(k)}}(\underbrace{0\cdots 0}_{j-1}x\underbrace{1\cdots 1}_{k-j}). \tag{11}$$

In particular, for any $j \in \{1,\dots,k-1\}$, $\omega^{(j)}_{u^{(j)}}(1) = \omega^{(j+1)}_{u^{(j+1)}}(0)$. As a consequence of this fact and of Eq. (10), we get

$$(\omega_u(0), \omega_u(1)) = (\omega^{(1)}_{u^{(1)}}(0), \omega^{(k)}_{u^{(k)}}(1)). \tag{12}$$

This proves Eq. (9) with $\omega^{(1)}_{u^{(1)}}(x) \overset{\text{def}}{=} \widehat{\omega}^{(1)}(x)$ (second equality in the last figure above).

Finally, Eq. (7) follows by considering the marginal of a node of degree 1, as $u^{(1)},\dots,u^{(k)}$ in graphs $\mathcal{H}_1,\dots,\mathcal{H}_k$, and expressing it in terms of the marginal of its only neighbor.

This completes one full step of the procedure that breaks the loops through node i. By recursively repeating the same steps, the graph is completely unfolded giving rise to $\mathsf{SAW}(i)$. □

The self-avoiding walk tree $\mathsf{SAW}(i)$ appears as a convenient way to organize the calculation of the marginal at i in the general case. In the case of permissive MRFs this calculation coincides with a standard marginal calculation on the tree $\mathsf{SAW}(i)$. It is instructive to check this explicitly.

Fact 1. *Proposition 1 is a special case of Proposition 2 for permissive MRFs.*

Proof. First notice that, for permissive MRFs, the self-avoiding walk tree construction yields a MRF on $\mathsf{SAW}(i)$ that defines a probability distribution (non-negative and normalizable), whose marginals will be denoted as ω as well. We have to prove that, in this case, the generalized root marginal is proportional to the ordinary marginal at the root of $\mathsf{SAW}(i)$. The crucial remark is that, because of permissivity, the messages are non-negative and, in particular, $\omega_{u\to v}(x_u^*) > 0$ and $\widehat{\omega}_{u\to v}(x_v^*) > 0$.

Assume, without loss of generality, that $x_u^* = 0$. We define the likelihood ratios on the $\mathsf{SAW}(i)$ tree $R_{u\to v} = \omega_{u\to v}(1)/\omega_{u\to v}(0)$, $\widehat{R}_{u\to v} = \hat{\omega}_{u\to v}(1)/\hat{\omega}_{u\to v}(0)$ and $R_i = \omega_i(1)/\omega_i(1)$. The ratio $\widehat{R}_{\mathsf{D}_l(u)}$ is defined analogously in terms of $\omega_{\mathsf{D}_l(u)}(\,\cdot\,)$. Equation (7) then implies

$$\widehat{R}_{u\to v} = \frac{\psi_{uv}(0,1) + \psi_{uv}(1,1)\,R_{u\to v}}{\psi_{uv}(0,0) + \psi_{uv}(1,0)\,R_{u\to v}}. \tag{13}$$

Eq. (8) yields on the other hand

$$R_{u\to w} = \frac{\psi_u(1)}{\psi_u(0)}\prod_{l=1}^{k}\widehat{R}_{\mathsf{D}_l(u)}. \tag{14}$$

Finally, using the remark that $\widehat{\omega}^{(l)}(1) = \widehat{\omega}^{(l+1)}(0)$ for $l = 1,\dots,k-1$, we get from Eq. (9)

$$\widehat{R}_{D_l(u)} = \widehat{R}^{(1)}\cdots\widehat{R}^{(k)} = \prod_{v\in\mathsf{D}_l(u)}\widehat{R}_{v\to u}. \tag{15}$$

Putting the last two equations together

$$R_{u\to w} = \frac{\psi_u(1)}{\psi_u(0)}\prod_{v\in\mathsf{D}(u)}\widehat{R}_{v\to u}. \tag{16}$$

It is now easy to check that, Eq. (16) and Eq. (13) coincide with the appropriate recursive definition of probability marginal ratios on $\mathsf{SAW}(i)$. □

Proposition 2 does not yield an efficient way of computing marginals of gMRF. The conundrum is that the resulting complexity is linear in the size of $\mathsf{SAW}(i)$ which is in turn exponential in the size of the original graph $\mathcal{G}$. On the other hand, it provides a systematic way to define and study algorithms for computing efficiently such a marginal. The idea, proposed first in [2], is to deform $\mathsf{SAW}(i)$ in such a way that its generalized root marginal does not change too much, but computing it is much easier.

III. Truncating the Tree

BP can be seen as an example of the approach mentioned at the end of the previous section. In this case $\mathsf{SAW}(i)$ is replaced by the first t generations of the computation tree, to be denoted by $\mathsf{T}(i;t)$. In this case the complexity of evaluating the generalized root marginal scales as t rather than as $|\mathsf{T}(i;t)|$.

A different idea is to cut some of the branches of $\mathsf{SAW}(i)$ in such a way to reduce drastically its size. We will call *truncation* the procedure of cutting branches of $\mathsf{SAW}(i)$. It is important to keep in mind that truncation is different from the *termination* of branches when a loop is closed in $\mathcal{G}$. While termination is completely defined, we are free to define truncation to get as good an algorithm as we want. In the following we shall define truncation schemes parametrized by an integer t, and denoted as $\mathsf{SAW}(i;t)$. We will have $\mathsf{SAW}(i;t) = \mathsf{SAW}(i)$ for $t \geq n$, thus recovering the exact marginal by Proposition 2.

In order for the algorithm to be efficient, we need to ensure the following constraints. (i) $\mathsf{SAW}(i;t)$ is 'small enough' (as the complexity of computing its generalized root marginal is at most linear in its size). (ii) $\mathsf{SAW}(i;t)$ is 'easy to construct.' For coding applications, this second constraint is somewhat less restrictive because the tree(s) $\mathsf{SAW}(i;t)$ can be constructed in a preprocessing stage and not recomputed at each use of the code.

In order to achieve the second goal, we must define the partition $\mathsf{D}(u) = \mathsf{D}_{1,t}(u) \cup \cdots \cup \mathsf{D}_{k,t}(u)$ of children of u according to the subtree $\mathsf{SAW}(i;t)$ used in the computation. Consider two children of u, which we denote by $v_1, v_2 \in \mathsf{D}(u)$. In a similar way as for the $\mathsf{SAW}(i)$ in the complete tree case, we write them as $v_1 = (u, j_1)$, $v_2 = (u, j_2)$, and define $v_1 \sim_t v_2$ if there exists a descendant v_1' of v_1 in $\mathsf{SAW}(i;t)$ such that $\pi(v_1') = j_2$ or a descendant v_2' of v_2 such that $\pi(v_2') = j_1$. The construction of the partition $\{\mathsf{D}_{1,t}(u), \dots, \mathsf{D}_{k,t}(u)\}$ will be different whether communication takes place over erasure or general channels.

A. Weitz's Fixed Depth Scheme and its Problems

The truncation procedure proposed in [2] amounts to truncating all branches of $\mathsf{SAW}(i)$ at the same depth t, unless they are already terminated at smaller depth. Variables at depth t (boundary) are forced to take arbitrary values.

The rationale for this scheme comes from the 'strong spatial mixing' property that holds for the system studied in [2]. Namely, if we denote by $\omega_{i|t}(x_i|\underline{x}_t)$ the normalized marginal distribution at the root i given variable assignment at depth t, we have

$$||\omega_{i|t}(\,\cdot\,|\underline{x}_t) - \omega_{i|t}(\,\cdot\,|\underline{x}_t')||_{\rm TV} \le A\,\lambda^t\,, \tag{17}$$

uniformly in the boundary conditions $\underline{x}_t$, $\underline{x}_t'$ for some constants $A > 0$, $\lambda \in [0,1)$.

It is easy to realize that the condition in Eq. (17) generically does not hold for 'good' sparse graph codes. The reason is that fixing the codeword values at the boundary of a large tree, normally determines their values inside the same tree. In other words the TP estimates strongly depend on this boundary condition.

One can still hope that some simple boundary condition might yield empirically good estimates. An appealing choice is to leave 'free' the nodes at which the tree is truncated. This means that no node potential is added on these boundary vertices. We performed numerical simulations with this scheme on the same examples considered in the next section. The results are rather poor: unless the truncation level t is very large (which is feasible only for small codes in practice) the bit error rate is typically worse than under BP decoding.

B. Improved Truncation Schemes: Erasure Channel

For decoding over the BEC, a simple trick improves remarkably the performances of TP decoding. First, construct a subtree of $\mathsf{SAW}(i)$ of depth at most t by truncating at the deepest variable nodes whose depth does not exceed t. The partition $\{\mathsf{D}_{1,t}(u), \dots, \mathsf{D}_{k,t}(u)\}$ is constructed using the equivalence class of the transitive closure of $\sim_t$. Then run ordinary BP on this graph, upwards from the leaves towards the root, and determine all messages in this direction. If a node u is decoded in this way, fix it to the corresponding value and further truncate the tree $\mathsf{SAW}(i)$ at this node.

The resulting tree $\mathsf{SAW}(i;t)$ is not larger than the one resulting from fixed-depth truncation. For low erasure probabilities it is in fact much smaller than the latter.

C. Improved Truncation Schemes: General channel

The above trick cannot be applied to general BM channels. We therefore resort to the following two constructions.

(i) Construction $\mathsf{MAP}(i;t)$: Define the distance $d(i,j)$ between two variable nodes i, j to be the number of check nodes encountered along the shortest path from i to j. Let $\mathsf{B}(i;t)$ be the subgraph induced[3] by variable nodes whose distance from i is at most t. Then we let $\mathsf{SAW}(i;t) \overset{\rm def}{=} \mathsf{MAP}(i;t)$ be the *complete* self-avoiding walk tree for the subgraph $\mathsf{B}(i;t)$. This corresponds to truncating $\mathsf{SAW}(i)$ as soon as the corresponding self-avoiding walk exits $\mathsf{B}(i;t)$. No forcing self-potential is added on the boundary.

A nice property of this scheme is that it returns the *a posteriori* estimate of transmitted bit X_i given the channel outputs within $\mathsf{B}(i;t)$, call it $\underline{Y}_{\mathsf{B}(i;t)}$. As a consequence, many reasonable performance measures (bit error probability, conditional entropy, etc.) are monotone in t [11].

On the negative side, the size of the tree $\mathsf{MAP}(i;t)$ grows very rapidly (doubly exponentially) with t at small t. This prevented us from using $t \ge 3$.

(ii) Construction $\mathsf{MAP}(i;t) - \mathsf{BP}(\ell)$: The tree $\mathsf{MAP}(i;t)$ constructed as in the previous approach is augmented by adding some descendants to those nodes that are terminated in $\mathsf{MAP}(i;t)$. More precisely, below any such node u in the mentioned tree, we add the first ℓ generations of the computation tree.

(iii) Construction $\mathsf{SAW}(i;t)$: We can implement a finer scheme for the general BM case. This scheme operates on $\mathsf{SAW}(i;t)$ obtained by truncating all branches of $\mathsf{SAW}(i)$ at the same depth t. The description of this method is slightly lengthy. We omit the details here and choose to present the numerical results in Fig. 7 of Section IV.

IV. Numerical simulations

For communication over the BEC, the implementation of TP decoding as described in Section III-B is satisfying. While it is simple enough for practical purpose, it permits us to depict performance curves that interpolate successfully between BP and MAP decoding. The binary erasure channel, which we denote by BEC(ϵ) if the erasure probability is ϵ, is appealing for a first study for the following reasons. (i) The accessibility of performance curves under MAP decoding allows for a careful study of the new algorithm. (ii) The TP decoder turns out to be 'robust' with respect to changes in the truncation method, hence simpler to study.

As an example for a generic BM channel, we shall consider the binary-input additive white Gaussian noise channel with standard deviation σ, which we denote by BAWGN(σ^2).

[3]The subgraph induced by a subset U of variable nodes is the one including all those check nodes that only involve variables in U.

Let us stress that the TP decoder is not (in general) symmetric with respect to codewords. This complicates a little the analysis (and simulations) which has to be performed for uniformly random transmitted codewords.

A. *Binary Erasure Channel*

The erasure case is illustrated by three examples: a tail-biting convolutional code, the $(23, 12)$ Golay code and a sparse graph code. Here the comparison is done with BP after convergence ('infinite' number of iterations) and MAP as implemented through Gaussian elimination.

TP decoder permits us to plot a sequence of performance curves, indexed by the truncation parameter t. In all of the cases considered, TP improves over BP already at small values of t. As t increases, TP eventually comes very close to MAP. The gain is particularly clear in codes with many short loops, and at low noise. This confirms the expectation that, when truncated, TP effectively takes care of small 'pseudocodewords.'

The first example is a memory two and rate $1/2$ convolutional code in tailbiting form with blocklength $n = 100$. The performance curves are shown in Fig. 3. The TP and BP decoders are based on a periodic Tanner graph associated with the tailbiting code with generator pair $(1+D^2, 1+D+D^2)$. More precisely, they are based on the graph representing the parity-check matrix with circulant horizontal pattern `110111`.

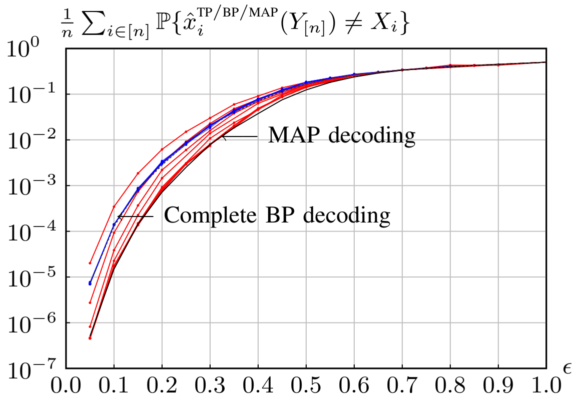


Fig. 3. Tailbiting convolutional code with generator pair $(1 + D^2, 1 + D + D^2)$ and blocklength $n = 100$. Black curve: BP decoding with $t = \infty$. Red curve: MAP decoding (BP and Gaussian elimination). Blue curves: BP decoding with $t = 3, 4, 5, 6, 8, 10, 12, 14$ (almost indistinguishable). Red curves: TP decoding with $t = 3, 4, 5, 6, 8, 10, 12, 14$ (truncated tree).

The second example is the standard (perfect) Golay code with blocklength $n = 23$. It is shown in Fig. 4.

The third example, an LDPC code with blocklength $n = 50$, is depicted in Fig. 5.

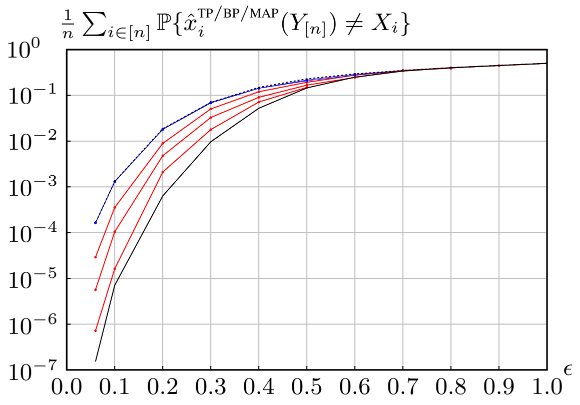


Fig. 4. $(23, 12)$ Golay code with blocklength $n = 23$. Blue curve: BP decoding with $t = \infty$. Black curve: MAP decoding (BP and Gaussian elimination). Blue curves: BP decoding with $t = 4, 5, 6$. Red curves: TP decoding with $t = 4, 5, 6$ (truncated tree).

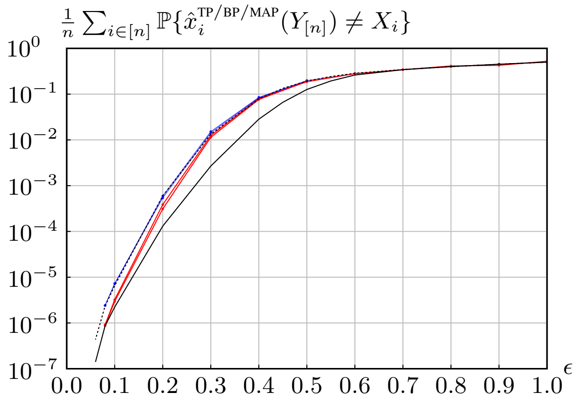


Fig. 5. A regular $(3, 6)$ LDPC code with blocklength $n = 50$. Blue curve: BP decoding with $t = \infty$. Black curve: MAP decoding (BP and Gaussian elimination). Blue curves: BP decoding with $t = 7, 8$. Red curves: TP decoding with $t = 7, 8$ (truncated tree).

B. *Binary-Input Additive White Gaussian Noise Channel*

In the case of the BAWGN channel, we consider a single example of code, the tail-biting convolutional code used above, and two truncation schemes, the constructions (ii) and (iii) described in Section III-C.

Our results are shown in Fig. 6 and Fig. 7. The TP and BP decoders are based on the natural periodic Tanner graph associated with the tailbiting code. We run BP a large number of iterations and check the error probability to be roughly independent of the iterations number. The MAP decoder is performed using BP on the single-cycle tailbiting trellis (i.e., BCJR on a ring [7], [9], [10]).

We observe that the two schemes $\mathsf{MAP}(i;t) - \mathsf{BP}(\ell)$ and $\mathsf{SAW}(i;t)$ with $t = 8$ outperform BP. Unhappily, due to complexity constraints we were limited to small values of t and therefore could not approach the actual MAP performances.

V. Theoretical Implications

One interesting direction is to use the self-avoiding walk tree construction for analysis purposes. We think in particular of two types of developments: (i) a better understanding of the relation between BP and MAP decoding, and (ii) a study of the 'inherent hardness' of decoding sparse graph codes.

While the first point is self-explanatory, it might be useful to spend a few words on the second. The most important

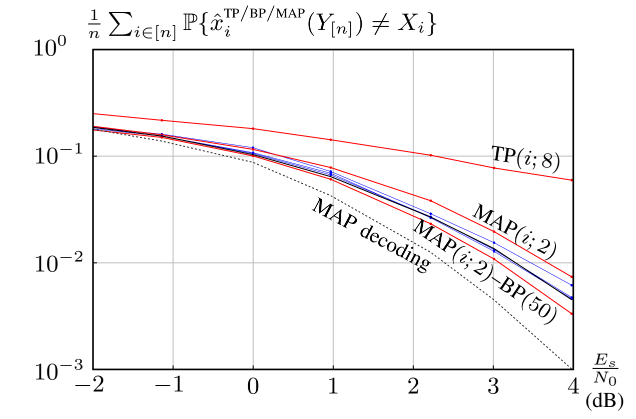


Fig. 6. Tailbiting convolutional code with generator pair $(1+D^2, 1+D+D^2)$ and blocklength $n=50$. Dashed black curve: BP decoding with $t=400$. Black curve: MAP decoding (wrap-around BCJR). Blue curves: BP decoding with $t=8,50$. Red curves: TP decoding with $t=8$ (truncated tree, denoted by TP$(i;8)$), TP decoding on a ball of radius 2 (scheme (ii) with no BP processing, denoted by MAP(2)) and, TP decoding according to scheme (ii) (with parameters as indicated by MAP$(i;2)-$BP(50)).

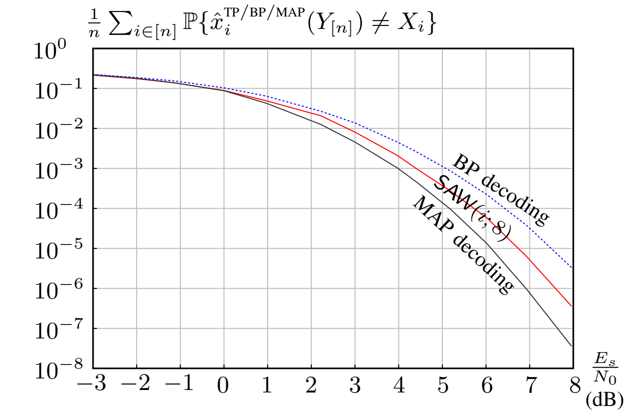


Fig. 7. Tailbiting convolutional code with generator pair $(1+D^2, 1+D+D^2)$ and blocklength $n=50$. Blue curve: BP decoding with $t=400$. Black curve: MAP decoding (wrap-around BCJR). Red curve: TP decoding according to scheme (iii) (with parameter as indicated by SAW$(i;t=8)$ using a suitable truncated tree).

outcome of the theory of iterative coding systems can be phrased as follows.

> There exist families of graphs (expanders [12], random [13]) with diverging size and bounded degree, below a certain noise level, MAP decoding can be achieved in linear time up to a 'small error.'

We think (a formal version of) the same statement to be true for *any family of graphs* with bounded degree. This can be proved for the erasure channel.

Proposition 3. *Let $\{G_n\}$ be a family of Tanner graphs of diverging blocklength n, with maximum variable degree $\mathtt{l}$ and check degree $\mathtt{r}$. Consider communication over $\text{BEC}(\epsilon)$ with $\epsilon < 1/(\mathtt{l}-1)(\mathtt{r}-1)$. Then, for any $\delta > 0$ there exists a decoder whose complexity is of order $n\text{Poly}(1/\delta)$ and returning estimates $\{\hat{x}_1(\underline{y}), \hat{x}_2(\underline{y}), \dots, \hat{x}_n(\underline{y})\}$ such that $\mathbb{P}\{\hat{x}_i(\underline{y}) \neq \hat{x}_i^{\text{MAP}}(\underline{y})\} \leq \delta$.*

Proof. The decoder consists in returning the MAP estimate of i given the subgraph $\mathsf{B}(i;t)$ and the values received therein. Consider the subgraph $G_n(\underline{y})$ of G_n obtained by removing non-erased bits. The proof consists in an elementary percolation estimate on this graph, see [14].

It is easy to see that $\mathbb{P}\{\hat{x}_i(\underline{y}) \neq \hat{x}_i^{\text{MAP}}(\underline{y})\}$ is upper bounded by the probability that the connected component of $G_n(\underline{y})$ that contains i is not-contained in $\mathsf{B}(i;t)$. This is in turn upper bounded by the number of paths between i and a vertex at distance $t+1$ (which is at most $\mathtt{l}(\mathtt{l}-1)^t(\mathtt{r}-1)^t$) times the probability that one such path is completely erased (which is ϵ^{t+1}). Therefore, for $A = \mathtt{l}\epsilon > 0$ and $\lambda = (\mathtt{l}-1)(\mathtt{r}-1)\epsilon < 1$, we get $\mathbb{P}\{\hat{x}_i(\underline{y}) \neq \hat{x}_i^{\text{MAP}}(\underline{y})\} \leq A\lambda^t$. The proof is completed by taking $t = \log(A/\delta)/\log(1/\lambda)$, and noticing that $\mathsf{B}(i;t)$ can be decoded in time polynomial in its size, that is polynomial in $1/\delta$. The computation is repeated for each $i \in \{1,\dots,n\}$ whence the factor n. □

We think that a strengthening (better dependence on the precision δ) and generalization (to other channel models) of this result can be obtained using the self-avoiding walk tree construction.

VI. Acknoledgments

Yi Lu is supported by the Cisco Stanford Graduate Fellowship. Cyril Méasson is supported by the Swiss National Fund Postdoctoral Fellowship.

References

[1] T. Richardson and R. Urbanke, *Modern Coding Theory*, draft available at `http://lthcwww.epfl.ch/index.php`

[2] D. Weitz, "Counting independent sets up to the tree threshold," Proc. 38th ACM Symp. on Theory of Comp., Seattle, May 2006

[3] T. Richardson, "Error floors of LDPC codes," Proc. 41st Allerton Conf. on Comm., Contr., and Computing. Monticello, IL, 2003

[4] C. Nair and P. Tetali, "The correlation decay (CD) tree and strong spatial mixing in multi-spin systems," `arXiv:math/0701494v3`, 2007

[5] K. Jung and D. Shah, "Inference in Binary Pair-wise Markov Random Fields through Self-Avoiding Walks," `arXiv:cs/0610111v2`, 2006

[6] E. Mossel and A. Sly, "Rapid Mixing of Gibbs Sampling on Graphs that are Sparse on Average," `arXiv: 0704.3603`, 2007

[7] R. J. McEliece, "On the BCJR trellis for linear block codes," *IEEE Trans. Inform. Theory,* vol. 42, pp. 1072–1092, July 1996

[8] G. Battail, M. Decouvelaere, and P. Godlewski, "Replication decoding," *IEEE Trans. Inform. Theory,* vol. 25, pp. 332–345, May 1979

[9] C. Hartmann and L. Rudolph, "An optimum symbol-by-symbol decoding rule for linear codes," *IEEE Trans. Inform. Theory,* vol. 22, pp. 514–517, Sept. 1976

[10] J. B. Anderson and S. M. Hladik, "Tailbiting MAP decoders," *IEEE Journal on selected areas in communications,* vol. 16, no. 2, pp. 297–302, Feb. 1998

[11] A. Montanari, "Two Lectures on Coding Theory and Statistical Mechanics," in *Mathematical Statistical Physics: Lecture Notes of the Les Houches Summer School 2005*, A. Bovier editor, Elsevier 2006

[12] M. Sipser and D. Spielman, "Expander codes," *IEEE Trans. Inform. Theory,* vol. 42, pp. 1710–1722, Nov. 1996

[13] T. J. Richardson and R. Urbanke, "The capacity of low-density parity-check codes under message-passing decoding," *IEEE Trans. Inform. Theory,* vol. 49, pp. 1611–1635, July 2003

[14] G. Grimmett, *Percolation*, Springer, New York, 1999