Inference in Binary Pair-wise Markov Random Fields through Self-Avoiding Walks

Kyomin Jung and Devavrat Shah

Abstract

In a recent result, Weitz [31] established equivalence between the marginal distribution of a node, say v, in any binary pair-wise Markov Random Field (MRF), say G, with the marginal distribution of the root node in the self-avoid walk tree of the G starting at v. In this paper, we exploit this remarkable connection to obtain insights in the performance of the widely popular Belief Propagation heuristic for computing marginal distribution (sum-product) and max-marginal distribution (max-product).

We obtain a tight characterization of the size of self-avoiding walk tree for any connected graph as a function of number of edges. This may be of interest in its own right.

I. Introduction

Markov Random Fields (MRFs) [21] have been extremely useful in modeling across various disciplines. A pair-wise MRF is an n-vector of random variables $X = \{X_1, X_2, \ldots, X_n\}$ whose dependency structure is described by a graph $G = (V, E)$ with vertices $V = \{1, \ldots, n\}$ and edge set E. Here vertex $i \in V$ corresponds to random variable X_i. Let each $X_i \in \Sigma$. Then the joint distribution of $X = (X_1, \ldots, X_n)$ is given by

$$\mathbb{P}[X = x] \quad \propto \quad \prod_{i \in V} \phi_i(x_i) \prod_{(i,j) \in E} \psi_{ij}(x_i, x_j), \tag{1}$$

for $x \in \Sigma^n$. Here $\phi_i : \Sigma \to \mathbb{R}^+ \stackrel{\triangle}{=} \{x \in \mathbb{R} : x \geq 0\}$, and $\psi_{ij} : \Sigma^2 \to \mathbb{R}^+$ are some non-negative (real-valued) functions. We will use notation that $\psi_{ij}(a, b) = \psi_{ji}(b, a)$. For the distribution of X to be well-defined, it should be the case that $\prod_{i \in V} \phi_i(x_i) \prod_{(i,j) \in E} \psi_{ij}(x_i, x_j) \neq 0$ for at least one $x \in \Sigma^n$. In this setting, the followings are two questions of interest:

1. Compute a Maximum a-posteriori (MAP) assignment x^*, where
$$x^* = \arg\max_{\sigma \in \Sigma^n} \mathbb{P}[X = \sigma].$$
2. Compute marginal distributions of variables, i.e.
$$\mathbb{P}[X_v = \sigma_v]; \quad \text{for } \sigma_v \in \Sigma \text{ and } v \in V.$$

Computing MAP is equivalent to computing a minimal energy assignment (or ground state) where energy, $\mathcal{E}(x)$, of state $x \in \Sigma^n$ is defined as

$$\mathcal{E}(x) = -\sum_{i \in V} \log \phi_i(x_i) - \sum_{(i,j) \in E} \log \psi_{ij}(x_i, x_j) + \mathsf{Constant}.$$

Similarly, algorithm for computing marginal distribution can lead to (with self-reduction technique cf. [7]) computation of log-partition function defined as

$$\log \mathcal{Z} = \log \left(\sum_{x \in \Sigma^n} \prod_{i \in V} \phi_i(x_i) \prod_{(i,j) \in E} \psi_{ij}(x_i, x_j) \right).$$

The question of finding MAP (or ground state) comes up in many important application areas such as coding theory [4], [18], [22], discrete optimization [11], [25], image denoising [6]. Similarly, computing marginal distribution as

K. Jung is with Applied Math, MIT and D. Shah is with EECS, MIT. Email : {kmjung, devavrat}@mit.edu.

well as log-partition function have varied applications including counting combinatorial objects as well as loss-probability in computer networks (see for example, [9]). It is well known that the problem of finding MAP in general is NP-hard and hard to approximate even within constant factor (for example, see implication of results in [15]). Hence, it is important to identify useful heuristics for MAP and marginal probability computation. It is also important to identify graph structures that yield simple algorithms for these problems, since in many cases (such as codes) a designer can opt for simple structures (such as LDPC codes).

A. *Previous Work*

The above goals have motivated a lot of interesting research across communities. Here we discuss a few notable results that are most relevant to our work.

Most work has focussed on the design of simple, efficient heuristics for finding MAP and marginal distributions. The most prominent attempts have been simulated annealing [10], graph cuts (see for example, [3]), generalized belief propagation (BP) [32] and tree-reweighted algorithm (TRW) [26]. Among these the MP, BP and TRW have received a lot of recent interest. The MP algorithm is essentially distributed iterative implementation of dynamic programming on a graph assuming it to have tree structure. Similarly, BP is exact for trees and provides an approximation for other graphs. Their success in many practical situations have fueled a lot of interest in understanding when these algorithms work (convergence and correctness), the error they induce when they do not work and whether there are natural corrections to improve their performance.

As a first step towards understanding these algorithms, researchers have found characterizations of their fixed points [12], [13], [27]–[30], [32]. However, this does not provide explicit error bounds. While general set of sufficient conditions for convergence have been derived [24], in general, with exception of few cases (e.g. [1], [2], [5], [8], [16], [17], [20], [23]) the convergence and correctness properties of these algorithms are not well understood.

It is well understood that if MRF graph structure is locally tree-like and there is some form of *correlation decay*, then algorithms MP and BP find almost correct estimates (implication of [24]). However, cycles or loops become a big issue. To improve upon this status one of the most interesting approach was presented by Wainwright, Jaakkola and Willsky [26], [27] in terms of a tree re-weighted algorithm. Intuitively, this algorithm attempts to run MP or BP on multiple trees of a given graph simultaneously with the hope that the algorithm will converge to a good approximation. This is supported by an insight based on the linear programming relaxation of the energy minimization problem by using the spanning trees of the MRF graph for the MAP problem (similar is true for computation of log partition function). Thus, if the algorithm solved this linear program exactly then it will provide a maximum lower bound on minimum energy based on this specific relaxation. They established that the fixed point of the algorithm is an exact MAP solution if the problem satisfies certain *tree agreement* conditions. Recently, Kolmogorov [12] proposed a modification of this algorithm and established that under a certain weaker form of *tree agreement* conditions it achieves “local” maxima of the desired lower bound. Finally, Kolmogorov and Wainwright [13] improved the guarantees for the goodness of fixed point of this algorithm by showing that for any pair-wise binary MRF the algorithm fixed point is a global maxima for linear programming relaxation under weak *tree agreement* conditions. The question of sufficient conditions for TRW (and its variant) to converge remains unresolved. All the above tree re-weighted algorithms have provided better answers compared to MP for represenstative experimental results that are presented collectively in the above papers.

B. *Relevant Previous Work*

The starting point of this paper is a recent result by Weitz [31] that establishes the following non-intuitive and remarkable connection: the marginal distribution for any node, say v in MRF G, is equal to the marginal distribution of v in the MRF induced on the self-avoiding walk tree obtained on G starting from v. We recall the formal statement of this result later in the paper. The important insight here is the possibility of computing the marginal of a node in G by means of constructing self-avoiding walk tree. As we shall see, this self-avoiding walk tree, by definition is a sub-tree of the breadth-first search tree and this fact leads to ”message-passing” implementation of the algorithm.

C. *Contribution*

The result of Weitz [31] about the surprising equality between marginal distribution of node v in MRF G and that in self-avoiding walk tree carries over in the context of max-marginal for computing MAP as well. Essentially, this is due to the fact that the summation and product operations commute in the same way as the maximum and product.

In this paper, our aim is to build upon these results to make an attempt at answering the following questions: (1) When the MRF G has loops, is it possible to identify precise error (possibly hard to compute) in the computation of BP (and MP) in terms of the structure of MRF ? Does this provide some simple error bounds ? (2) In case of error, are there possibly simple corrections for BP and MP in presence of *few* cycles ? If so, how much do they cost in terms of operations ? (3) More generally, is it possible to come up with *better* heuristics compared to TRW for graphs with many loops.

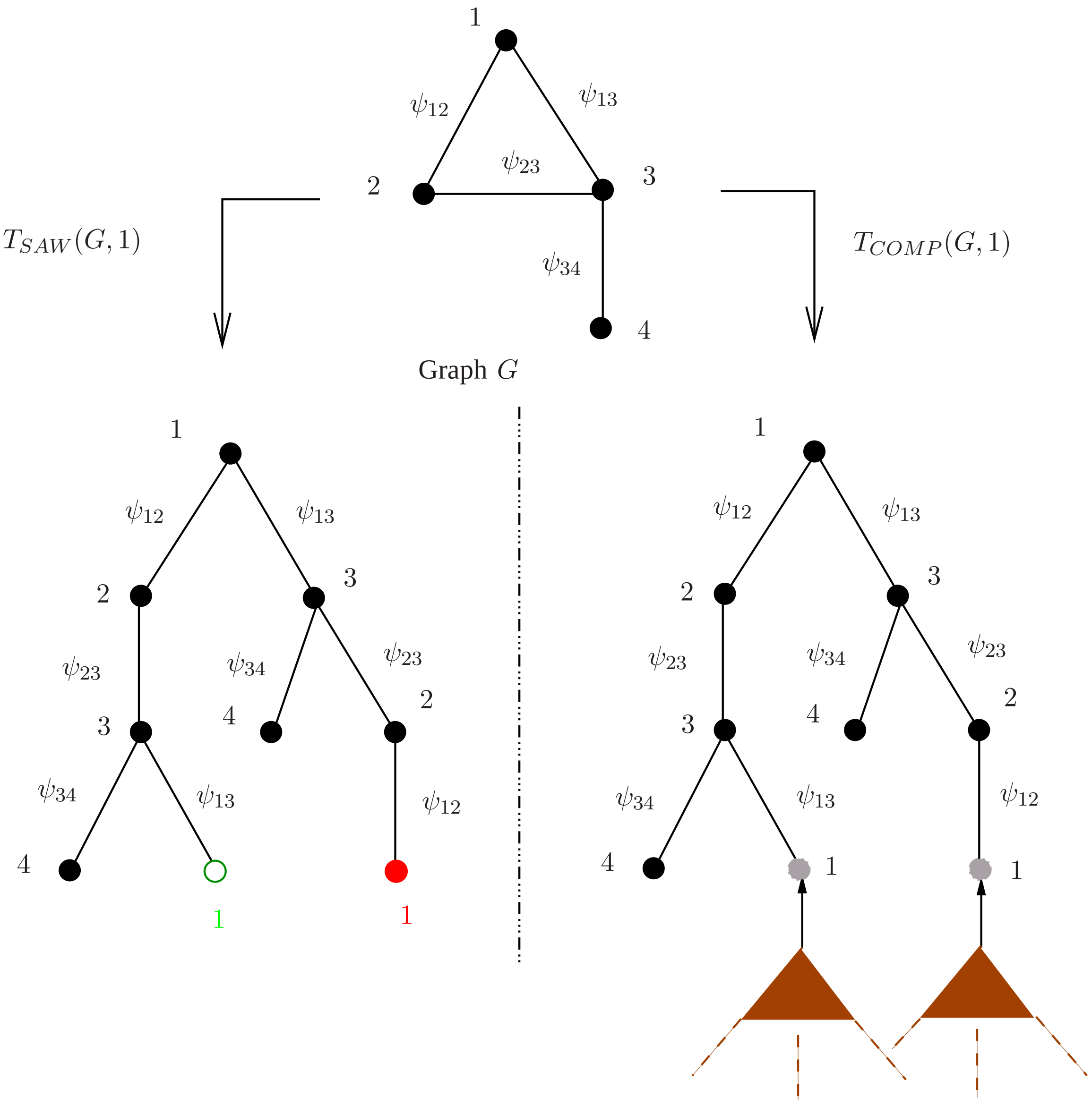

Fig. 1. A graph G of 4 nodes with one loop is given. On left, we have the self-avoiding walk tree of G for node 1, i.e. $T_{SAW}(G,1)$ with green and red being special nodes. On right, we have computation tree $T_{COMP}(G,1)$ for node 1's computation under Belief Propagation (or Max-Product) algorithm. The grey nodes of $T_{COMP}(G,1)$ correspond to green and red node of $T_{SAW}(G,1)$ on the left.

In what follows, we provide an example in which we identify error in BP (or MP) building upon result of [31]. The precise formal statements will follow later in the paper. Consider a pair-wise MRF G of 4 nodes as shown in the Figure 1. This G has one loop. Now consider node 1 of G. The self-avoiding walk tree T_{SAW} of node 1 (with corresponding MRF) is shown on the bottom left while the computation tree T_{COMP} (for MP and BP) of node 1 is shown on bottom right. The self-avoiding walk tree is essentially the breadth-first search walk of G starting from 1 while avoiding excursion in a direction beyond cycle completion.

The important thing to notice is that T_{SAW} can be obtained by chopping off T_{COMP} at the two leaf nodes (grey leaf nodes) which are replica of node 1 as shown in the figure. The T_{SAW} fixes the values of these special nodes as follows: green node to value 1 (i.e. modify ϕ for this node so that $\phi(0) = 0, \phi(1) = 1$) and red node to value 0 (i.e. modify ϕ for this node so that $\phi(0) = 1, \phi(1) = 0$) respectively. (More precise definition of green node and red node is given in Definition 1.) In contrast, in T_{COMP} we will have *messages* coming from bottom trees to these grey nodes when BP is executed on the original MRF G (thus modifying their node potentials).

Weitz's result implies that marginal probability of node 1 in G is exactly the same as that of the root node of T_{SAW} (Theorem 1). Direct adaptation of Theorem 1 for max-marginal implies that the max-marginal of node 1 in G is exactly the same as max-marginal of root node in T_{SAW} (Theorem 3).

The following are important implications of these results : (a) If we run BP (resp. MP) on T_{SAW} then at the root of T_{SAW} it will converge to the correct marginal (resp. max-marginal) of node 1 in G. Thus, BP or MP algorithm can be corrected by setting messages to 0 or 1 at grey nodes (corresponding to green or red nodes) in a deterministic manner (see algorithms CMP and CBP in Sections III-A and IV-A). In our example, the original BP can be *corrected* in the following simple manner: node 1 should always send message to node 2 as if it has $\phi(1) = 1, \phi(0) = 0$ and always send message to node 3 as if it has $\phi(1) = 0, \phi(0) = 1$; everything else should be the same as original BP. (b) To evaluate complexity of these algorithms, we need to evaluate size of the self-avoiding walk tree for a given graph. While study of self-avoiding walks have been of great interest (see book by Madras and Slade [19]), to the best of our knowledge, we have not seen characterization of size of self-avoiding walk tree for arbitrary connected graph. We show that the size of self-avoiding walk tree for connected graph with $n + k$ edges is essentially $n2^k$ (Theorem 5). (c) The algorithms for exact computation naturally give rise to simple, distributed heuristic that allow for many corrections in MP and BP. These are described in Sections III-C and IV-C. The experimental results validate the excellent performance of our heuristic algorithm based on CMP as explained in Section V. It outperforms the known Tree re-weighted algorithm on many of the interesting setup. These heuristics provide a single tunable parameter that allows for trading off computation power with accuracy in answers. (d) Finally, connection between computation tree of BP and self-avoiding walk tree allow for identification of error in the BP heuristic in terms of structural property of G. For example, in the Figure 1 is essentially due to the effect of messages coming to grey nodes of T_{COMP} onto the marginal probability (resp. max-marginal) of root. Based on this, conditions for correctness and error bound for BP can be obtained (Section IV-D).

II. Equivalence: MRF and Self-Avoiding Walk Tree

In this section, we describe the equality relation between the marginal probability or max-marginal of a node v in MRF G and the marginal probability or max-marginal of root node in self-avoiding walk tree MRF obtained for v in G.

A. Preliminaries

We are interested in pair-wise binary MRF given by graph of n nodes $G = (V, E)$ with given edge-compatibility functions $\psi_{uv}(\cdot, \cdot), (u, v) \in E$ and node-potentials $\phi_v(\cdot), v \in V$. Let $\mathbb{P}_G(\cdot)$ denote the probability distribution induced by this MRF on boolean cube $\{0,1\}^n$ as per pair-wise Markvoian relation given by (1). We will be primarily interested in the following two values: for each node $v \in V$,

(1) The marginal distribution of v, i.e.

$$p_v(1) = \sum_{\sigma \in \{0,1\}^n : \sigma_v = 1} \mathbb{P}_G(\sigma), \text{ and } p_v(0) = \sum_{\sigma \in \{0,1\}^n : \sigma_v = 0} \mathbb{P}_G(\sigma).$$

(2) The max-marginal for node v or equivalently

$$p^*_v(1) = \max_{\sigma \in \{0,1\}^n : \sigma_v = 1} \mathbb{P}_G(\sigma), \text{ and } p^*_v(0) = \max_{\sigma \in \{0,1\}^n : \sigma_v = 0} \mathbb{P}_G(\sigma).$$

Definition 1 (Self-Avoiding Walk Tree): Consider graph $G = (V, E)$ of pair-wise binary MRF. For $v \in V$, we define the self avoiding walk tree $T_{SAW}(G, v)$ as follows. First, for each $u \in V$, give an ordering of its neighbors $N(u)$. This ordering can be arbitrary but remains fixed forever. Given this, $T_{SAW}(G, v)$ is constructed by the breadth first search of nodes of G starting from v without backtracking. Then stop the bread-first search along a direction when an already visited vertex is encountered (but include it in $T_{SAW}(G,v)$ as a leaf). Say one such leaf be $\hat{w}$ of $T_{SAW}(G,v)$ and let it be a copy of a node w in G. We call such a leaf node of $T_{SAW}(G,v)$ as *Marked*. A marked leaf node is assigned color *Red* or *Green* according to the following condition: The leaf $\hat{w}$ is marked since we encountered node w of G twice along our bread-first search excursion. Let the (directed) path between these two encounters of w in G be given by $(w, v_1, \dots, v_k, w)$. Naturally, $v_1, v_k \in N(w)$ in G. We mark the leaf node $\hat{w}$ as *Green* if according to the ordering done by node w in G of its neighbors, if v_k is given smaller number than that of v_1. Else, we mark it as *Red*. Let $\mathbf{V}_v$ and $\mathbf{E}_v$ denote the set of nodes and vertices of tree $T_{SAW}(G,v)$. With little abuse of notation, we will call root of $T_{SAW}(G,v)$ as v.

Given a $T_{SAW}(G,v)$ for a node $v \in V$ in G, an MRF is naturally induced on it as follows: all edges inherit the pair-wise compatibility function (i.e. $\psi_{\cdot\cdot}(\cdot,\cdot)$) and all nodes inherit node-potentials (i.e. $\phi_{\cdot}(\cdot)$) from those of MRF G in a natural manner. The only distinction is the modification of the node-potential of *marked* leaf nodes of $T_{SAW}(G,v)$ as follows. A marked leaf node, say $\hat{w}$ of $T_{SAW}(G,v)$ modifies its potentials as follows: if it is *Green* than it sets $\phi_{\hat{w}}(1) = \phi_w(1), \phi_{\hat{w}}(0) = 0$ but if it is *Red* leaf node then it sets $\phi_{\hat{w}}(0) = \phi_w(0), \phi_{\hat{w}}(1) = 0$.

Example 1 (Self-avoiding walk tree): Consider 4 node binary pair-wise MRF G in Figure 1. Let node 1 gives number a to node 2, number b to node 3 so that $a > b$. Given this numbering, the bottom left of Figure 1 represents $T_{SAW}(G,1)$. The Green leaf node essentially means that we set its value permanently to 1.

With above description, $T_{SAW}(G,v)$ gives rise to a pair-wise binary MRF. Let $\mathbb{Q}_{G,v}$ denote the probability distribution induced by this MRF on boolean cube $\{0,1\}^{|\mathbf{V}_v|}$. Our interest will be primarily in two values:

(1) The marginal distribution of root v, i.e.

$$q_v(1) = \sum_{\sigma \in \{0,1\}^{|\mathbf{V}_v|}:\sigma_v=1} \mathbb{Q}_{G,v}(\sigma), \text{ and } q_v(0) = \sum_{\sigma \in \{0,1\}^{|\mathbf{V}_v|}:\sigma_v=0} \mathbb{Q}_{G,v}(\sigma).$$

(2) The max-marginal for root v or equivalently

$$q_v^*(1) = \max_{\sigma \in \{0,1\}^{|\mathbf{V}_v|}:\sigma_v=1} \mathbb{Q}_{G,v}(\sigma), \text{ and } q_v^*(0) = \max_{\sigma \in \{0,1\}^{|\mathbf{V}_v|}:\sigma_v=0} \mathbb{Q}_{G,v}(\sigma).$$

B. Equivalence I: Marginal Distribution

The following is a result of Weitz [31]. We present the proof from [31] for completeness.

Theorem 1: Consider any binary pair-wise MRF $G = (V,E)$. For any $v \in V$, let $p_v(\cdot)$ be its marginal probability under distribution $\mathbb{P}_G$. Let $T_{SAW}(G,v)$ be the self-avoiding walk tree MRF and let $q_v(\cdot)$ marginal probability of root node of $T_{SAW}(G,v)$ with respect to $\mathbb{Q}_{G,v}$. Then,

$$p_v(1) = q_v(1) \quad \text{and} \quad p_v(0) = q_v(0). \tag{2}$$

Proof: The proof of Theorem 1 follows by mathematical induction over the number of *unfixed* nodes of the graph G.

Initial condition. Trivially the desired statement holds for any graph with exactly one *unfixed* vertex, by definition of MRF, i.e. (1). The reason is that for such a graph, due to all but one node being fixed, the marginal probability of each node is purely determined by its immediate neighbors due to Markovian nature of MRF and the immediate neighborhood of v in $T_{SAW}(G,v)$ and G is the same.

Hypothesis. Assume that the statement is true for any graph with less than or equal to $m \in \mathbb{N}$ *unfixed* nodes.

Induction step. Without loss of generality, suppose that our graph of interest, G, has $m+1$ *unfixed* vertices. If v is *fixed* vertex, then (2) holds trivially. Let $v \in V$ be a unfixed vertex of G. Then we will show via inductive hypothesis that

$$\frac{q_v(1)}{q_v(0)} = \frac{p_v(1)}{p_v(0)}.$$

Let d be the degree of v; $v_1, v_2, \dots, v_d$ be the neighbors of v where the order of neighbors is the same as that used in definition of $T_{SAW}(G,v)$. Let T_ℓ be the ℓth subtree of $T_{SAW}(G,i)$ having v_ℓ as its root and $Y(\ell)$ be the binary pair-wise MRF induced on T_ℓ by restriction of $T_{SAW}(G,v)$. Let $q_\ell(\sigma)$ be the marginal probability of vertex v_ℓ taking value $\sigma \in \Sigma = \{0,1\}$ with respect to $Y(\ell)$. Note that when T_ℓ consists of a single vertex, then $q_\ell(\sigma) \propto \phi_{v_\ell}(\sigma)$. Let $\lambda_v = \frac{\phi_v(1)}{\phi_v(0)}$. Then from definition of pair-wise MRF and tree-structure,

$$\frac{q_v(1)}{q_v(0)} \quad = \quad \lambda_v \prod_{\ell=1}^{d} \frac{\sum_{\sigma\in\Sigma} \psi_{v_\ell,v}(\sigma,1) q_\ell(\sigma)}{\sum_{\sigma\in\Sigma} \psi_{v_\ell,v}(\sigma,0) q_\ell(\sigma)}. \tag{3}$$

Now to calculate $\frac{p_v(1)}{p_v(0)}$, we define a new graph G' and the corresponding pair-wise MRF X' as follows. Let G' be the same as G except that v is replaced by d vertices $v'_1, v'_2, \dots, v'_d$; each v'_ℓ is connected only to v_ℓ, $1 \le \ell \le d$. The X' is defined same as X except that $\phi_{v'_\ell}(1) = \lambda_v^{1/d}\phi_v(1)$, $\phi_{v'_\ell}(0) = \phi_v(0)$ and $\psi_{v_\ell v'_\ell} = \psi_{v_\ell v}$. Then,

$$\frac{p_v(1)}{p_v(0)} \quad = \quad \frac{\sum_{\left\{X' : X'_{v'_1}=1, X'_{v'_2}=1,\dots,X'_{v'_d}=1\right\}} \mathbb{P}_{G'}(X')}{\sum_{\left\{X' : X'_{v'_1}=0, X'_{v'_2}=0,\dots,X'_{v'_d}=0\right\}} \mathbb{P}_{G'}(X')} \quad = \quad \prod_{\ell=1}^{d} \frac{\mu_\ell(1)}{\mu_\ell(0)}, \tag{4}$$

where define $\mu_\ell(\sigma) = \sum_{\{X'_{v'_\ell}=\sigma\}} \mathbb{P}[X' |\ X'_{v'_1} = 0, \dots, X'_{v'_{(\ell-1)}} = 0, X'_{v'_{(\ell+1)}} = 1, \dots, X'_{v'_d} = 1]$. The second equality in (4) follows by standard trick of Telescoping multiplication and Lemma 2.

Now for $1 \le \ell \le d$, consider MRF $X'(\ell)$ induced on $G'(\ell) = G' - \{v'_\ell\}$ by fixing $\{v'_1, \dots v'_d\} - \{v'_\ell\}$ as follows: let $(\phi_{v'_1}(0) = 1, \phi_{v'_1}(1) = 0); \dots; (\phi_{v'_{\ell-1}}(0) = 1, \phi_{v'_{\ell-1}}(1) = 0); (\phi_{v'_{\ell+1}}(0) = 0, \phi_{v'_{\ell+1}}(1) = 1); \dots; (\phi_{v'_d}(0) = 0, \phi_{v'_d}(1) = 1)$. Then let $\nu_\ell(\sigma), \sigma \in \Sigma$ denote the max-marginal of v_ℓ for taking value σ with respect to $X'(\ell)$. Given this, by definition of MRF X' as well $X'(\ell)$ and noting that v'_ℓ is a leaf (only connected to v_ℓ) with respect to graph G', we have

$$\frac{\mu_\ell(1)}{\mu_\ell(0)} \quad = \quad \lambda_v^{1/d} \frac{\sum_{\sigma\in\Sigma} \psi_{v_\ell,v'_\ell}(\sigma,1)\nu_\ell(\sigma)}{\sum_{\sigma\in\Sigma} \psi_{v_\ell,v'_\ell}(\sigma,0)\nu_\ell(\sigma)}. \tag{5}$$

From (3), (4) and (5) it is sufficient to show that

$$\frac{\nu_\ell(1)}{\nu_\ell(0)} \quad = \quad \frac{q_\ell(1)}{q_\ell(0)}, \quad 1 \le \ell \le d. \tag{6}$$

Now, note that T_ℓ is the same as $T_{SAW}(G(\ell))$ with respect to $X'(\ell)$. Because for each $\ell = 1, \dots d$, $G'(\ell)$ has one less *unfixed* node than G, the desired result (11) follows by induction hypothesis. ■

Lemma 2: Consider a distribution on $X = (X_1, \dots, X_n)$ where X_i are binary variables. Let $p_s = \mathbb{P}[X = s], s \in \Sigma^n$. Let $p_{s|a_2,\dots,a_d} = \mathbb{P}[X = s | X_2 = a_2, \dots, X_d = a_d]$ for any $d \ge 1$. Let $S(a_1, \dots, a_d) = \{s = (s_1, \dots, s_n) \in \Sigma^n : s_1 = a_1, \dots, s_d = a_d\}$. Then,

$$\frac{\sum_{s\in S(a_1,a_2\dots,a_d)} p_s}{\sum_{s\in S(\hat{a}_1,a_2,\dots,a_d)} p_s} = \frac{\sum_{s\in S(a_1,a_2\dots,a_d)} p_{s|a_2,\dots,a_d}}{\sum_{s\in S(\hat{a}_1,a_2,\dots,a_d)} p_{s|a_2,\dots,a_d}}.$$

Proof: Let $q = \mathbb{P}(X_2 = a_2, \dots, X_d = a_d)$. Then, by definition of conditional probability for $s \in S(a_1, a_2, \dots, a_d) \cup S(\hat{a}_1, a_2, \dots, a_d)$, $p_s = p_{s|a_2,\dots,a_d} q$. From this, Lemma follows immediately. ■

C. Equivalence II: MAP

Here we present similar equivalence result about max-marginal. While this is not stated in [31], the Theorem 3 follows from arguments exactly the same as those used in Theorem 1 with ”summation” replaced by ”maximum”. We will present proof for completeness.

Theorem 3: Consider any binary pair-wise MRF $G = (V, E)$. For any $v \in V$, let $p^*_v(\cdot)$ be as defined above with respect to $\mathbb{P}_G$. Let $T_{SAW}(G, v)$ be the self-avoiding walk tree MRF and let $q^*_v(\cdot)$ be as defined above for root node of $T_{SAW}(G, v)$ with respect to $\mathbb{Q}_{G,v}$. Then,

$$\frac{p^*_v(1)}{p^*_v(0)} \quad = \quad \frac{q^*_v(1)}{q^*_v(0)}. \tag{7}$$

Here we allow ratio to be $0, \infty$.

Proof: The proof follows by induction. As a part of the proof, we will come across graphs with some *fixed* vertices, where a vertex u is said to be fixed to 0 (resp. 1) if $\phi_u(0) > 0$, $\phi_u(1) = 0$ (resp. $\phi_u(1) > 0$,

$\phi_u(0) = 0$). The induction is on the number of *unfixed* vertices of G. We essentially prove the following, which implies the statement of Lemma: given any pair-wise MRF on a graph G (with possibly some *fixed* vertices), construct corresponding $T_{SAW}(G, v)$ MRF for some node v. If the number of *unfixed* vertex of G is at most m, then the (7) holds. Next, inductive proof.

Initial condition. Trivially the desired statement holds for any graph with exactly one *unfixed* vertex, by definition of MRF, i.e. (1). The reason is that for such a graph, due to all but one node being fixed, the max-marginal of each node is purely determined by its immediate neighbors due to Markovian nature of MRF. The immediate neighborhood of v in $T_{SAW}(G, v)$ and G is the same.

Hypothesis. Assume that the statement is true for any graph with less than or equal to $m \in \mathbb{N}$ *unfixed* nodes.

Induction step. Without loss of generality, suppose that our graph of interest, G, has $m + 1$ *unfixed* vertices. If v is a *fixed* vertex, then (7) holds trivially. Let $v \in V$ be an unfixed vertex of G. Then we will show via inductive hypothesis that

$$\frac{q_v^*(1)}{q_v^*(0)} = \frac{p_v^*(1)}{p_v^*(0)}.$$

Let d be the degree of v; $v_1, v_2, \dots, v_d$ be the neighbors of v where the order of neighbors is the same as that used in definition of $T_{SAW}(G, v)$. Let T_ℓ be the ℓth subtree of $T_{SAW}(G, i)$ having v_ℓ as its root and $Y(\ell)$ be the binary pair-wise MRF induced on T_ℓ by restriction of $T_{SAW}(G, v)$. Let $q_\ell^*(\sigma)$ be the max-marginal of vertex v_ℓ taking value $\sigma \in \Sigma = \{0, 1\}$ with respect to $Y(\ell)$. Note that when T_ℓ consists of a single vertex, then $q_\ell^*(\sigma) \propto \phi_{v_\ell}(\sigma)$. Let $\lambda_v = \frac{\phi_v(1)}{\phi_v(0)}$. Then from definition of pair-wise MRF and tree-structure,

$$\frac{q_v^*(1)}{q_v^*(0)} = \lambda_v \prod_{\ell=1}^{d} \frac{\max_{\sigma \in \Sigma} \psi_{v_\ell, v}(\sigma, 1) q_\ell^*(\sigma)}{\max_{\sigma \in \Sigma} \psi_{v_\ell, v}(\sigma, 0) q_\ell^*(\sigma)}. \tag{8}$$

Now to calculate $\frac{p_v^*(1)}{p_v^*(0)}$, we define a new graph G' and the corresponding pair-wise MRF X' as follows. Let G' be the same as G except that v is replaced by d vertices $v'_1, v'_2, \dots, v'_d$; each v'_ℓ is connected only to v_ℓ, $1 \le \ell \le d$. The X' is defined same as X except that $\phi_{v'_\ell}(1) = \lambda_v^{1/d} \phi_v(1)$, $\phi_{v'_\ell}(0) = \phi_v(0)$ and $\psi_{v_\ell v'_\ell} = \psi_{v_\ell v}$. Then,

$$\frac{p_v^*(1)}{p_v^*(0)} = \frac{\max_{\left\{X' : X'_{v'_1} = 1, X'_{v'_2} = 1, \dots, X'_{v'_d} = 1\right\}} \mathbb{P}_{G'}(X')}{\max_{\left\{X' : X'_{v'_1} = 0, X'_{v'_2} = 0, \dots, X'_{v'_d} = 0\right\}} \mathbb{P}_{G'}(X')} = \prod_{\ell=1}^{d} \frac{\mu_\ell(1)}{\mu_\ell(0)}, \tag{9}$$

where define $\mu_\ell(\sigma) = \max_{\{X' : X'_{v'_\ell} = \sigma\}} \mathbb{P}[X' |\ X'_{v'_1} = 0, \dots, X'_{v'_{(\ell-1)}} = 0, X'_{v'_{(\ell+1)}} = 1, \dots, X'_{v'_d} = 1]$. The second equality in (9) follows by standard trick of Telescoping multiplication and Lemma 4.

Now for $1 \le \ell \le d$, consider MRF $X'(\ell)$ induced on $G'(\ell) = G' - \{v'_\ell\}$ by fixing $\{v'_1, \dots v'_d\} - \{v'_\ell\}$ as follows: let $(\phi_{v'_1}(0) = 1, \phi_{v'_1}(1) = 0); \dots; (\phi_{v'_{\ell-1}}(0) = 1, \phi_{v'_{\ell-1}}(1) = 0); (\phi_{v'_{\ell+1}}(0) = 0, \phi_{v'_{\ell+1}}(1) = 1); \dots; (\phi_{v'_d}(0) = 0, \phi_{v'_d}(1) = 1)$. Then let $\nu_\ell(\sigma), \sigma \in \Sigma$ denote the max-marginal of v_ℓ for taking value σ with respect to $X'(\ell)$. Given this, by definition of MRF X' as well $X'(\ell)$ and noting that v'_ℓ is a leaf (only connected to v_ℓ) with respect to graph G', we have

$$\frac{\mu_\ell(1)}{\mu_\ell(0)} = \lambda_v^{1/d} \frac{\max_{\sigma \in \Sigma} \psi_{v_\ell, v'_\ell}(\sigma, 1) \nu_\ell(\sigma)}{\max_{\sigma \in \Sigma} \psi_{v_\ell, v'_\ell}(\sigma, 0) \nu_\ell(\sigma)}. \tag{10}$$

From (8), (9) and (10) it is sufficient to show that

$$\frac{\nu_\ell(1)}{\nu_\ell(0)} = \frac{q_\ell^*(1)}{q_\ell^*(0)}, \quad 1 \le \ell \le d. \tag{11}$$

Now, note that T_ℓ is the same as $T_{SAW}(G(\ell))$ with respect to $X'(\ell)$. Because for each $\ell = 1, \dots d$, $G'(\ell)$ has one less *unfixed* node than G, the desired result (11) follows by induction hypothesis. ■

Lemma 4: Consider a distribution on $X = (X_1, \dots, X_n)$ where X_i are binary variables. Let $p_s = \mathbb{P}[X = s], s \in \Sigma^n$. Let $p_{s|a_2,\dots,a_d} = \mathbb{P}[X = s | X_2 = a_2, \dots, X_d = a_d]$ for any $d \ge 1$. Let $S(a_1, \dots, a_d) = \{s = (s_1, \dots, s_n) \in \Sigma^n : s_1 = a_1, \dots, s_d = a_d\}$. Then,

$$\frac{\max_{s \in S(a_1, a_2 \dots, a_d)} p_s}{\max_{s \in S(\hat{a}_1, a_2, \dots, a_d)} p_s} = \frac{\max_{s \in S(a_1, a_2 \dots, a_d)} p_{s|a_2,\dots,a_d}}{\max_{s \in S(\hat{a}_1, a_2, \dots, a_d)} p_{s|a_2,\dots,a_d}}.$$

Proof: Let $q = \mathbb{P}(X_2 = a_2, \ldots, X_d = a_d)$. Then, by definition of conditional probability for $s \in S(a_1, a_2, \ldots, a_d) \cup S(\hat{a}_1, a_2, \ldots, a_d)$, $p_s = p_{s|a_2,\ldots,a_d} q$. From this, Lemma follows immediately. ■

III. Algorithm: MAP

A. Exact MAP

Theorem 3 suggests the following algorithm *Correction of Max-Product* (CMP) for exact MAP estimate in binary pair-wise MRF. The distributed message-passing implementation of this algorithm is presented in Section III-D. But here we present the basic idea behind it for ease of the reading.

CMP

(1) Let nodes V be numbered $1, \ldots, n$. Initially, none of the nodes is set to have its assignment value. Starting from $v = 1$, iteratively set values of nodes as follows:

(2) Given G with values of nodes $1, \ldots, v-1$ are set, obtain the corresponding self-avoiding walk tree $T_{SAW}(G, v)$.

(3) Compute $\frac{q_v^*(1)}{q_v^*(0)}$ by running standard max-product algorithm on $T_{SAW}(G, v)$. Set value of v to 1 if the above ration is ≥ 1, else set value of v to 0.

(4) Increment $v \leftarrow v + 1$ and repeat (2)-(3) till $v = n$ (i.e. values of all nodes are set).

(5) The resulting assignment of all nodes is a MAP assignment from Theorem 3.

B. *Complexity of* CMP

Consider the self-avoiding walk tree $T_{SAW}(G, v) = (\mathbf{V}_v, \mathbf{E}_v)$. Since $T_{SAW}(G, v)$ is a connected tree, we have $|\mathbf{V}_v| = |\mathbf{E}_v| + 1$. We will denote by size of $T_{SAW}(G, v)$ as $|\mathbf{E}_v|$. Now, above described algorithm CMP (in its distributed implementation described in Section III-D) takes total $O(|T_{SAW}(G, v)|)$ distributed operations to find the $T_{SAW}(G, v)$ as well as $O(|T_{SAW}(G, v)|)$ for fixing value of node v using max-product. Thus, the total complexity of algorithm is $O(n \max_{v \in V} |T_{SAW}(G, v)|)$ distributed operations (or message exchanges). When graph G is tree, this is essentially the same as the complexity of max-product algorithm. For graphs with loops, we know that max-product may not find exact solution and not even terminate. Our algorithm will always terminate but may take a lot longer than the running time for tree graph. Next, we quantify how long it takes for a graph with cycles.

The notion of number of cycles in a graph is hard to define. Instead, we look at connected graphs with edges $n - 1 + k$ for $k \geq 0$. We state the precise statement as follows.

Theorem 5: Consider a connected graph $G = (V, E)$ with $|V| = n$ nodes and $|E| = n - 1 + k$ edges, $k \geq 0$. Then for any $v \in V$,

$$|T_{SAW}(G, v)| \leq (n + k - 1)2^{k+1}.$$

Further, there exists a graph with $n - 1 + k$ edges with $k < n/2$ so that for any node $v \in V$,

$$|T_{SAW}(G, v)| \geq n2^{k-2}.$$

Proof: The proof is divided into two parts. We first provide the proof of lower bound. Consider a line graph of n nodes (with $n - 1$ edges). Now add $k < n/2$ edges as follows. Add an edge between 1 and n. Remaining $k - 1$ edges are added between node pairs: $(2, 4), (4, 6), \ldots, (2(k-2), 2(k-1)), (2(k-1), 2k)$. Consider any node, say v. It is easy to see that there are at least 2^{k-2} different ways in which one can start walking on the graph from node v towards node 1, cross from 1 to n via edge $(1, n)$ and then come back to node v. Each of these different loops, starting from v and ending at v creates 2 distinct paths in the self-avoiding walk tree of length at least$\frac{n}{2}$. Thus, the size of self-avoiding walk tree of each node is at least $n2^{k-2}$ for each node. This completes the proof of lower bound.

Now, we prove the upper bound of $n2^{k+1}$ on the size of self-avoiding walk tree for each node $v \in V$. Given that G is connected, we can divide the edge set $E = E_T \cup E_k$ where $E_k = \{e_1, \ldots, e_k\}$ and $T = (V, E_T)$ forms a spanning tree of G. Let $\mathcal{S}$ be the set of all subsets of $E_k = \{e_1, \ldots, e_k\}$ (there are 2^k of them including empty set). Now fix a vertex $v \in V$ and we will concentrate on $T_{SAW}(G, v)$. Consider any $u \in V$ (can be v) and $S \in \mathcal{S}$. Next, we wish to count number of paths in $T_{SAW}(G, v)$ that end at (a copy of) u (however, u need not be a leaf), contain all edges in S but none from $E_k \backslash S$. We claim the following.

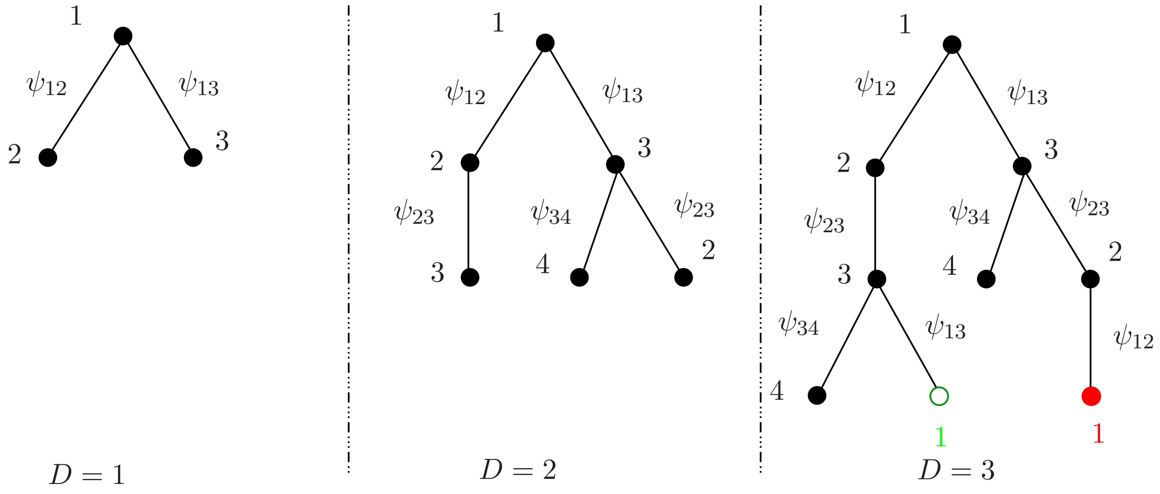

Fig. 2. Given graph G of 4 in Figure 1, the above describes $T^{D}_{SAW}(G, 1)$ for $D = 1, 2$ and 3.

Claim. There can be at most one path of $T_{SAW}(G, v)$ from v to (a copy of) u and containing all edges from S but none from $E_k \backslash S$.

Proof: To prove the above claim, suppose it is not true. Then there are at least two distinct paths from v to u that contain all edges in S (but none from $E_k \backslash S$). Consider the symmetric difference of these two paths (in terms of edges). This symmetric difference must be a non-empty subset of E_T and also contain a loop (as the two paths have same starting and ending point). But this is not possible as $T = (V, E_T)$ is a tree and it does not contain a loop. This contradicts our assumption and proves the claim. ■

Given the above claim, for any node u, clearly the number of distinct paths from node v to (a copy of) u in $T_{SAW}(G, v)$ are at most 2^k. Now each edge has two end points. For each appearance of an edge of G in $T_{SAW}(G, v)$, a distinct path from v to one of its end point must appear in $T_{SAW}(G, v)$. From above claim, this can happen at most $2 \times 2^k = 2^{k+1}$. There are $n + k - 1$ edges of G in total. Thus, net number of edges that can appear in $T_{SAW}(G, v)$ is at most $(n + k - 1)2^{k+1}$. This completes the proof of upper bound for Theorem 5. ■

Remark. We note that for any connected graph, the size of $T_{SAW}(G, v)$ can not be larger than the number of different permutations of $n = |V|$ since each distinct path in $T_{SAW}(G, v)$ can be identified with a distinct permutation of n numbers. That is, for any connected graph G (by Stirling's approximation)

$$|T_{SAW}(G, v)| \;\le\; n! \;=\; O\left(2^{n \log_2 n}\right).$$

C. Heuristic

The algorithm CMP obtains exact MAP. However, it may take too long for graphs with many loops. In such situations, we will need heuristic. The primary reason for CMP to take too long is the size of $T_{SAW}(G, v)$ for $v \in V$. Hence a natural way to obtain heuristic is to reduce the size of $T_{SAW}(G, v)$ for $v \in V$. To this end, consider the following definition.

Definition 2 (D-truncated Self-Avoiding Walk Tree): Given graph $G = (V, E)$ of pair-wise binary MRF, the D-truncated self-avoiding walk tree $T^{D}_{SAW}(G, v)$ is obtained by removing all nodes (and edges incident on them) that are at distance more than D from root v of $T_{SAW}(G, v)$ in $T_{SAW}(G, v)$.

The Figure 2 presents $T^{D}_{SAW}(G, v)$ for 4 node graph considered earlier in Figure 1. Naturally, for any graph G, by taking $D = n$ (or any D larger or equal to the graph diameter), we obtain that $T^{n}_{SAW}(G, v) = T_{SAW}(G, v)$ for all $v \in V$. Based on $T^{D}_{SAW}(G, v)$, we obtain the following heuristic which is a direct adaption of CMP .

CMP **(D)**

(1) Let nodes V be numbered $1, \ldots, n$. Initially, none of the nodes is set to have its assignment value. Starting from $v = 1$, iteratively set values of nodes as follows:
(2) Given G with values of nodes $1, \ldots, v-1$ are set, obtain the corresponding self-avoiding walk tree $T^{D}_{SAW}(G, v)$.

```
(3) Compute $\frac{q_v^*(1)}{q_v^*(0)}$ by running standard max-product algorithm on $T_{SAW}^D(G, v)$. Set value of $v$ to 1 if the above
    ration is $\geq 1$, else set value of $v$ to $0$.
(4) Increment $v \leftarrow v + 1$ and repeat (2)-(3) till $v = n$ (i.e. values of all nodes are set).
(5) The resulting assignment of all nodes is an estimate of MAP assignment.
```

D. *Distributed Message-Passing Implementation*

The following is a pseudo-code of a distributed message passing algorithm for CMP . The CMP finds exact MAP, by Theorem 3. This section is of interest primarily for two reasons: (1) It proves the possibility of distributed message-passing algorithm for MAP, and (2) Guide-lines for implementation for an interested reader. However, a reader may skip this section on the first read.

To describe the pseudo-code, we need some notation. Each node $v \in V$, let $N(v)$ denote the set of all its neighbors, i.e. $N(v) = \{u \in V : (u, v) \in E\}$. Node v assigns an arbitrary fixed order to all nodes in $N(v)$. For example, if v has neighbors u, w and z then it can number u as the first neighbor, w as second neighbor and z as third neighbor. The ordering chosen by each node is independent of choices of all other nodes. The algorithm operates in two phases. In the first phase, algorithm explores local topology for each node via sending "path sequences". By "path sequence" we mean a finite sequence of vertices $(v_1, v_2, \ldots, v_k)$, where $(v_\ell, v_{\ell+1}) \in E$ for $1 \leq \ell \leq k - 1$. In the second phase, algorithm uses the path sequences to recursively calculate "computation sequence" which in turn leads to calculation of $q_v^*(\cdot)$ at nodes. A "computation sequence" is of the form $(v_1, v_2, \ldots, v_k, m_{v_k}(0), m_{v_k}(1))$, where $m_{v_k}(\cdot)$ are certain real-numbers (which have interpretation of message). As we shall see, the structure of recursive calculation to obtain "computation sequence" is the same as that of max-product algorithm. Thus, there is very strong connection between MP and CMP . For ease of exposition, the algorithm is described to compute the ratio $q_v^*(1)/q_v^*(0)$ for all $v \in V$.

CMP

```
(0) Initially, each vertex $v$ sends a path sequence $(v)$ to each of its neighbors.
(1) When node $u$ receives a path sequence $(v_1, v_2, \ldots, v_k)$ from its neighbor $v$, (note that, by construction given
    later, $v_k = v$) it does the following:
    ○ If $u$ is a leaf (i.e. $u$ is connected only to $v$), $u$ sends back a computation sequence $(v_1, v_2, \ldots, v_k, u, m_u(0), m_u(1))$
      to $v$, where
      $$m_u(\sigma) \propto \max_{\sigma_u \in \Sigma} \psi_{u,v}(\sigma_u, \sigma)\phi_u(\sigma_u) \quad \text{and} \quad \sum_{\sigma_u \in \Sigma} m_u(\sigma_u) = 1.$$
    ○ If $u$ is not a leaf, check whether $u$ appears among $v_\ell$, $1 \leq \ell \leq k$:
      * (x[1]) If NO, $u$ sends a path sequence $(v_1, \ldots, v_k, u)$ to each of $u$'s neighbors but $v$.
      * If YES, then let $v_\ell = u, 1 \leq \ell < k$.
        – If, with respect to the ordering given by node $u$ to its neighbors, the rank (order) of node $v_{\ell+1}$ is
          larger then $v$, then $u$ sends back (to $v$) a computation sequence $(v_1, v_2, \ldots, v_k, u, m_u(0), m_u(1))$,
          where $m_u(1) = 1$ and $m_u(0) = 0$.
        – Otherwise (i.e. the rank of node $v_{\ell+1}$ is smaller than $v$), $u$ sends back (to $v$) computation sequence
          $(v_1, v_2, \ldots, v_k, u, m_u(0), m_u(1))$, where $m_u(0) = 1$ and $m_u(1) = 0$.
(2) Once a node $u$ receives a computation sequence $(v_1, \ldots, v_k, m_{v_k}(0), m_{v_k}(1))$ from its neighbor $v$, (note that,
    by construction $v_k = v$ and $v_{k-1} = u$). Store this computation sequence in $u$'s memory and do the following:
    ○ If $k > 2$, check whether $u$ has stored computation sequences of the form $(v_1, \ldots, v_{k-1}, w, m_w(0), m_w(1))$
      for all $w \in N(u) - \{v_{k-2}\}$. If so, $u$ sends a computation sequence $(v_1, \ldots, v_{k-1}(= u), m_u(0), m_u(1))$ to
      $v_{k-2}$ where
      $$m_u(\sigma) \propto \left[\max_{\sigma_u \in \Sigma} \psi_{u,v_{k-2}}(\sigma_u, \sigma)\phi_u(\sigma_u) \prod_{w \in N(u) - \{v_{k-2}\}} m_w(\sigma_u)\right], \quad \text{and} \quad \sum_{\sigma_u \in \Sigma} m_u(\sigma_u) = 1.$$
```

[1] The symbol **x** is a marker. It is to indicated that it is the only clause of algorithm that will be changed for CMP (D).

```
        Delete computation sequences $(v_1, \ldots, v_{k-1}, w, m_w(0), m_w(1))$ for all $w \in N(j) - \{i_{k-2}\}$ from $u$'s memory.
      ○ If $k = 2$, then check whether for all $w \in N(j)$, $u$ has stored computation sequences $(v_1, w, m_w(0), m_w(1))$.
        If so, compute the (estimate of) max-belief of $u$ as
```

$$q_u^*(\sigma) \propto \phi_u(\sigma) \prod_{w \in N(u)} m_w(\sigma), \quad \text{and} \quad \sum_{\sigma \in \Sigma} q_u^*(\sigma) = 1.$$

```
(3) When all nodes have computed their max-beliefs, declare $q_v^*(1)/q_v^*(0)$ as an estimate of $p_v^*(1)/p_v^*(0)$ $\forall$ $v \in V$.
```

We make a note here that the pseudo-code for CMP given above can be modified easily by concentrating on path-sequences of length at most D to obtain CMP (D). The modification need to be applied to the part marked by (**x**) in the above code.

IV. Algorithm: Marginal Distribution

A. *Exact Marginal Distribution*

Theorem 1 suggests, just like exact MAP computation, we can design algorithm for computing exact marginal distribution based on the self-avoiding walk tree of graph. We describe the algorithm, *Correction of BP*, CBP as follows.

CBP

```
(1) Let nodes $V$ be numbered $1, \ldots, n$. Initially, none of the nodes is set to have its value.
(2) For each node $v \in V$, compute the self-avoiding walk tree $T_{SAW}(G, v)$.
(3) Compute the marginal probabilities $q_v(0), q_v(1)$ for the root node $v$ of $T_{SAW}(G, v)$ by running standard sum-
    product (belief propagation) algorithm on $T_{SAW}(G, v)$.
(4) The resulting marginal probability estimates $q_v(0), q_v(1)$ are exact for all nodes $v \in V$ from Theorem 1.
```

The algorithm CBP , like CMP can be implemented in a distributed manner. The pseudo-code CMP described in Section III-D suggests how CBP can be implemented in a distributed message-passing manner. We do not describe this algorithm as it essentially follows from the description of CMP by replacing "max" operation by "summation".

B. *Complexity of* CBP

Like CMP , the above described algorithm CBP (in its distributed implemenation) takes total $O(n \max_{v \in V} |T_{SAW}(G, v)|)$ distributed operations. As stated in Theorem 5, the algorithm takes at most $n(n+k)2^{k+2}$ distributed operations for any connected graph G with $n+k$ edges. Thus, for small k our algorithm is essentially as efficient as the standard sum-product algorithm and always finds the correct solution.

C. *Heuristic*

As explained for algorithm CMP , a heuristic can be designed for computing marginal distribution based on CBP . This is done by computing marginal distribution based on D-truncated self-avoiding walk tree just as in Section III-C. For completeness, we describe the algorithm (similar to CMP (D)) as follows.

CBP **(**D**)**

```
(1) Let nodes $V$ be numbered $1, \ldots, n$.
(2) For each node $v \in V$, obtain the self-avoiding walk tree $T_{SAW}^D(G, v)$.
(3) Compute $\frac{q_v(1)}{q_v(0)}$ by running standard sum-product algorithm on $T_{SAW}^D(G, v)$.
(4) The resulting estimates are exact if $D = n$.
```

D. Error in BP

In this section, we use relation between the self-avoiding walk tree and computation tree to obtain a handle on error incurred in the estimation of BP depending on the underlying graph topology. Specifically, we will attempt to quantify the error in BP as well as condition under which BP will be correct even in presence of loops. To avoid cumbersome notations and formulas, we consider example in Figure 1 to exemplify our approach. It will be clear to any (may be familiar enough) reader how this can be generalized for any graph with loops (with the help of cumbersome extra notations).

Example 2 (Graph with Loops: Correctness of BP): Consider the specific 4 node graph G shown in Figure 1 with cycle. The figure also shows its self-avoiding walk tree $T_{SAW}(G,1)$ for node 1 and the computation tree $T_{COMP}(G,1)$ for node 1. Clearly, the $T_{SAW}(G,1)$ can be obtained from $T_{COMP}(G,1)$ by first truncating it at *grey* nodes and then making left *grey* node *green* (i.e. set to value 1) and right *grey* node *red* (i.e. set to value 0). With abuse of notation, we will call the left grey node in $T_{COMP}(G,1)$ as well as green node in $T_{SAW}(G,1)$ as 1_o; we will call the right grey node in $T_{COMP}(G,1)$ as well as red node in $T_{SAW}(G,1)$ as 1_c.

From Theorem 1 and property of BP being correct for any tree graph, when we run BP on $T_{SAW}(G,1)$ we get the exact answer for marginal probability of the root node 1. On the other hand, while running BP on G the answer corresponds to exact answer[2] on $T_{COMP}(G,1)$. Clearly, the only difference is due to the difference in effect of the *grey* nodes and *green*, *red* nodes on the computation of probability at the root. Thus, error in BP on G can be quantified by concentrating on this difference caused by these special leaf nodes. We will do this next. Note that, such will be the case for *any* graph.

To this end, first consider BP running on G. Consider algorithm at the end of a large enough iteration (here ≥ 4). The computation tree corresponding to this will have all the nodes up to depth 3 including *grey* nodes. Suppose the normalized message coming to the left *grey* node 1_o (corresponding to *green* node of $T_{SAW}(G,1)$) be α and $1-\alpha$, $\alpha \in [0,1]$ (i.e. α is message for node taking value 1 and $1-\alpha$ for node taking value 0). Similarly, for the right-hand side *grey* node 1_c, the message be β and $1-\beta$, $\beta \in [0,1]$. For the BP operating on $T_{SAW}(G,1)$, we have set 1_o to 1 while 1_c to 0. That is, effectively we can think of normalized message $1,0$ to node 1_o and $0,1$ to node 1_c. Now, to bound the difference between computation of the probabilities at roots in $T_{COMP}(G,1)$ and $T_{SAW}(G,1)$ we need to quantify the effect of messages at nodes $1_o, 1_c$ on the root.

Consider the following definitions. To quantify the effect of node 1_o on the root (in $T_{SAW}(G,1)$ as well as in $T_{COMP}(G,1)$), consider the following: for $x,y \in \{0,1\}$,

$$A(x,y) = \sum_{(b_2,b_3,b_4)\in\{0,1\}^3} \left(\psi_{12}(y,b_2)\psi_{23}(b_2,b_3)\psi_{34}(b_3,b_4)\psi_{31}(b_3,x)\right)\phi_2(b_2)\phi_3(b_3)\phi_4(b_4). \tag{12}$$

Similarly, consider the following for quantifying effect of 1_c: for $x,y \in \{0,1\}$,

$$B(x,y) = \sum_{(b_2,b_3,b_4)\in\{0,1\}^3} \left(\psi_{13}(y,b_3)\psi_{23}(b_2,b_3)\psi_{34}(b_3,b_4)\psi_{21}(b_2,x)\right)\phi_2(b_2)\phi_3(b_3)\phi_4(b_4). \tag{13}$$

Recall that by definition $\psi_{ij}(a,b) = \psi_{ji}(b,a)$. It is easy to check, that the definition implies for any $(x,y) \in \{0,1\}^2$,

$$A(x,y) = B(y,x). \tag{14}$$

Now, let $p_1(b), b \in \{0,1\}$ be correct marginal probability of node 1 according to MRF G and $p_1^{BP}(b), b \in \{0,1\}$ be estimate of marginal probability of node 1 obtained by BP algorithm at the end of iteration of our interest. By Theorem 1, we have

$$\frac{p_1(1)}{p_1(0)} = \lambda_1 \frac{A(1,1)B(0,1)}{A(1,0)B(0,0)}, \tag{15}$$

$$\frac{p_1^{BP}(1)}{p_1^{BP}(0)} = \lambda_1 \frac{(\alpha A(1,1) + (1-\alpha)A(0,1))(\beta B(1,1) + (1-\beta)B(0,1))}{(\alpha A(1,0) + (1-\alpha)A(0,0))(\beta B(1,0) + (1-\beta)B(0,0))}, \tag{16}$$

[2] For a reader not familiar with BP, we suggest reading literature [21], [25], [32] to understand relation between BP and computation tree.

where $\lambda_1 = \frac{\phi_1(1)}{\phi_1(0)}$. Now, it is not hard to argue, based on the fact that the functions $\psi_{\cdot\cdot}(\cdot,\cdot)$, $\phi_{\cdot}(\cdot)$ are non-negative, (14) and (16), that the following bounds hold:

$$\begin{aligned} \frac{p_1^{BP}(1)}{p_1^{BP}(0)} &\leq \max\left\{\frac{A(1,1)}{A(1,0)}, \frac{A(0,1)}{A(0,0)}\right\} \times \max\left\{\frac{A(1,1)}{A(0,1)}, \frac{A(1,0)}{A(0,0)}\right\}, \qquad (17)\\ \frac{p_1^{BP}(1)}{p_1^{BP}(0)} &\geq \min\left\{\frac{A(1,1)}{A(1,0)}, \frac{A(0,1)}{A(0,0)}\right\} \times \min\left\{\frac{A(1,1)}{A(0,1)}, \frac{A(1,0)}{A(0,0)}\right\}. \qquad (18)\end{aligned}$$

Then (14), (15), (17) and (18) imply that the BP is always correct if the following condition hold:

$$A(0,0)A(1,1) \quad = \quad A(1,0)A(0,1). \qquad (19)$$

More generally, when the above condition does not hold, (14)-(18) imply the error bounds on computation of BP as follows:

$$\begin{aligned} \frac{\frac{p_1^{BP}(1)}{p_1^{BP}(0)}}{\frac{p_1(1)}{p_1(0)}} &\leq \max\left\{\frac{A(0,0)}{A(1,0)}, \frac{A(0,1)}{A(1,1)}\right\} \times \max\left\{\frac{A(0,0)}{A(0,1)}, \frac{A(1,0)}{A(1,1)}\right\}\\ \frac{\frac{p_1^{BP}(1)}{p_1^{BP}(0)}}{\frac{p_1(1)}{p_1(0)}} &\geq \min\left\{\frac{A(0,0)}{A(1,0)}, \frac{A(0,1)}{A(1,1)}\right\} \times \min\left\{\frac{A(0,0)}{A(0,1)}, \frac{A(1,0)}{A(1,1)}\right\}. \qquad (20)\end{aligned}$$

The results of above example, specifically condition (19) for correctness of BP and (20) for error in BP, can be easily generalized to graphs with more loops in terms of *effect* of the *special cycle breaking* leaf nodes as part of computation tree. This can also help in providing error bound on computation of BP in graphs with more loops. We make a note of the fact that the existence of *correlation decay* (such as satisfaction of generalized Dobrushin's condition) will imply that the above errors are small if graph has locally tree-like structure. We finally note that the above approach can be naturally extended for obtaining error bound on computation of CBP (D) algorithm as well. In summary, we believe that this frame-work to quantify error in BP can be extremely useful in general.

V. Experiments

In this section we present experimental results of CMP (D) on two models. The first model is a random Ising model and the second is a Gaussian model on which we compare our algorithm with TRW [13].

Random Ising Model. The MRF is defined on 2-dimension $N \times N$ regular grid of $n = N^2$ nodes with V, E be its vertex and edge set respectively. Then, probability distribution is given by

$$\mathbb{P}[X = x] \quad \propto \quad \exp\left[\sum_{i \in V} \Theta_i x_i + \sum_{(i,j) \in E} \Theta_{ij} x_i x_j\right],$$

for $x \in \{-1,1\}^n$. In each trial of our experiment the single node potentials were chosen at random with uniform distribution over $[-1,1]$ denoted as $\Theta_i \sim \mathcal{U}[-1,1]$, whereas the edge potentials Θ_{ij} were chosen randomly as $\Theta_{ij} \sim \mathcal{U}[-\frac{r}{2}, \frac{r}{2}]$, where r is the edge strength. In the experiment, for each fixed N and r we generated random MRFs according to this random model for 20 times and for each obtained MRFs we executed CMP (D) for each depth $D = 1, \ldots, 9$. Then we reported the average value of $\mathsf{Error}(D)$ for each setup, where $\mathsf{Error}(D)$ is defined as follows: Let x^D be the assignment that is obtained by CMP (D). Then $\mathsf{Error}(D)$ is the percentage of vertices having different values in the assignment of CMP (D) from that of CMP (9), i.e.

$$\mathsf{Error}(D) = \frac{d_H(x^D, x^9)}{N^2} \times 100,$$

where $d_H(\cdot,\cdot)$ is the Hamming distance. Panels (a-1) to (a-3) of Figure 3 are plots showing Error versus D for the random Ising model with $N = 10, 20, 30$ resp. Different curves in each panel correspond to edge strength $r \in \{0.5, 1, 1.5, 2\}$.

Interestingly the result suggests that x^D converges rapidly as D increases, especially when edge strength $r \leq 1$. When $r \leq 1$, in most of the samples we generated, x^D became the same for $D \geq 6$. Hence this result suggests that when edge strength is small, the truncated CMP even with small constant depth finds a MAP assignment or an

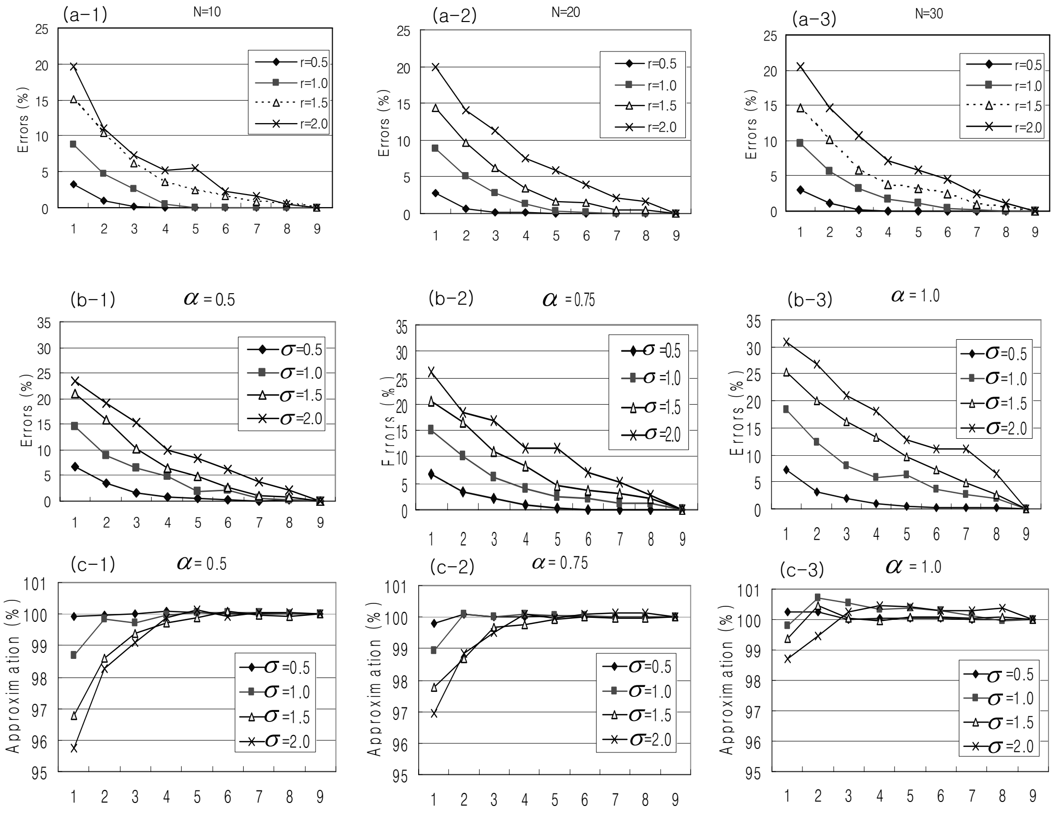

Fig. 3. (a-1) to a(a-3): Plots showing $\mathsf{Error}(D)$ versus depth D for random Ising model on $N \times N$ Grids with $N = 10, 20, 30$ resp. Different curves in each panel correspond to edge strength $r \in \{0.5, 1, 1.5, 2\}$. (b-1) to (b-3): Similar plots for Gaussian model on 32×32 grids with mixing parameter $\alpha = 0.5, 0.75, 1$ resp. Different curves in each panel correspond to potential strength $\sigma \in \{0.5, 1, 1.5, 2\}$. (c-1) to (c-3): Same setup for Gaussian model as (b-1) to (b-3), and y axis represents the ratio of $\log[\mathbb{P}(x^D)]$ to $\log[\mathbb{P}(x^9)]$.

assignment that is very close to a MAP assignment. In summary, for $r \leq 1$ the algorithm finds correct assignment for all nodes within small D, while for $r \geq 1$ it finds an assignment which is close to MAP in terms of ”energy”. Thus, our algorithm is good overall approximation.

Gaussian Model. Here graph is the same 2-dimensional grid of $n = N^2$ nodes with

$$\mathbb{P}[X = x] \quad \propto \quad \exp\left[\sum_{i \in V} \hat{\phi}_i(x_i) + \sum_{(i,j) \in E} \hat{\psi}_{ij}(x_i, x_j)\right],$$

where node potentials are given as Gaussian $\hat{\phi}_i(0), \hat{\phi}_i(1) \sim \mathcal{N}(0,1)$ and edge potentials $\hat{\psi}$'s are defined by $\hat{\psi}_{ij}(0,1) = \psi_{ij}(1,0) = 0$ and $\hat{\psi}_{ij}(0,0) = \psi_{ij}(1,1) = \lambda_{ij}$, where $\lambda_{ij} \sim |\mathcal{N}(0,\sigma^2)|$ with probability α and $\lambda_{ij} \sim -|\mathcal{N}(0,\sigma^2)|$ with probability $1-\alpha$. Here $\alpha \in [0,1]$ is a mixing parameter. We generated this model to compare performance of CMP (D) with that of TRW algorithms described in [13] for $N = 32$.

For each fixed α and σ, we generated random MRFs according to this model for 20 times and for each obtained MRFs we executed CMP (D) for each depth $D = 1, \ldots, 9$. Then we reported the average value of $\mathsf{Error}(D)$ for each setup. Panels (b-1) to (b-3) of Figure 3 represent changes of $\mathsf{Error}(D)$ according to D for $\alpha = 0.5, 0.75, 1$ respectively. Notice that when $\alpha = 0.5$, if $\sigma \leq 1$ then CMP (D) converges for $D \geq 7$. For $\alpha = 0.75$ or $\alpha = 1$, if $\sigma \leq 0.5$ CMP (D) converges for $D \geq 6$. Hence for these cases, the result suggest that the assignment obtained by CMP (D) for $D \geq 7$ would be equal to a MAP assignment. This is in contrast with poor performance of TRW in this range [13].

Now, Panels (c-1) to (c-3) of Figure 3 shows the ratio of $\log[\mathbb{P}(x^D)]$ to $\log[\mathbb{P}(x^9)]$ for $\alpha = 0.5, 0.75, 1$ respectively. Surprisingly in most of the cases, the $\log[\mathbb{P}(x^D)]$ value converges very rapidly from around $D = 6$. Hence it implies that even when σ is large, $\log[\mathbb{P}(x^D)]$ value would be very close to the maximum value of $\log[\text{MAP}]$ for small constant D even though exact assignment may be very different from MAP for many nodes.

Now, compare the above stated our results to the following report that is taken verbatim from [13] for TRW for Gaussian model: for $0.5 \leq \alpha \leq 0.75$ and $\sigma = 1$, TRW algorithm output less than 85% (empirical observation) of the variable assignment values (which are guaranteed to be correct), but give no information about the other variable values. And when $0.5 \leq \alpha \leq 0.75$ and $\sigma \geq 1.5$, the output percentage of TRW becomes less than 50% [13].

In summary, the above comparison shows that there is a large range where our algorithm outperforms TRW algorithm. The primary reason is the correction of errors done by our algorithms for small cycles.

VI. Future Work

For future work, we plan to combine ideas behind tree re-weighted algorithm and the above described heuristic to obtain a better inference algorithm.

Acknowledgements. We thank Andrea Montanari, Chandra Nair and Dror Weitz for pointing out discrepancies in the earlier draft regarding the documentation of results of [31]. We are grateful to Shivani Agarwal for her help in improving the readability of the paper to a great extent.

References

[1] M. Bayati, D. Shah and M. Sharma, “Maximum Weight Matching via Max-Product Belief Propagation,” *IEEE ISIT*, 2005.

[2] M. Bayati, D. Shah and M. Sharma, “A simple Max-Product Maximum Weight Matching Algorithm and The Auction Algorithm,” *IEEE ISIT*, 2006.

[3] Boykov, Veksler and Zabih, “Fast Approximate Energy Minimization via Graph Cuts,” *IEE Transactions on Pattern Analysis and Machine Intelligence*, 2001.

[4] R. Gallager, “Low-Density Parity Check Codes,” PhD Thesis, MIT, 1963.

[5] D. Gamarnik, T. Nowicki and G. Swirscsz, “Maximum Weight Independent Sets and Matchings in Sparse Random Graphs. Exact Results using the Local Weak Convergence Method,” *Random Structures and Algorithms*, 2006.

[6] D. Greig , B. Porteous and A. Seheult, “Exaact maximum a posteriori estimation for binary images,” *Journal of the Royal Statistical Society*, 51(2), 1989.

[7] M. Jerrum and A. Sinclair, “ Polynomial-time Approximation Algorithms for the Ising Model,” *SIAM Journal of Computing*, 22, 1993.

[8] J. Johnson, D. Malioutov and A. Willsky, “Walk-Sum Interpretation and Analysis of Gaussian Belief Propagation,” *NIPS*, 2005.

[9] F. P. Kelly, “Stochastic models of computer communication systems,” *Journal of the Royal Statistical Society (Series B)*, 47, 1985.

[10] S. Kirkpatrick and C. D. Gelatt and M. P. Vecchi, “Optimization by Simulated Annealing,” *Science*, 220:4598, 1983.

[11] R. Kleinberg and E. Tardos, “Approximation Algorithms for classification problems with pair-wise relationships: Metric labeling and Markov Random Fields,” *IEEE FOCS*, 1998.

[12] V. Kolmogorov, “Convergent Tree-reweighted Message Passing for Energy Minimization,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2006.

[13] V. Kolmogorov and M. Wainwright, “On optimality of tree-reweighted max-product message-passing,” *Uncertainty in Artificial Intelligence*, 2005.

[14] F. R. Kschischang, B. J. Frey and H.-A. Loeligeer, “Factor graphs and the sum-product algorithm,” *IEEE Transaction on Information Theory*, 47, 2001.

[15] C. Lund and M. Yannakakis, “On the hardness of approximating minimization problems,” *ACM STOC*, 1993.

[16] M. Luby, M. Mitzenmacher, M Shokrollahi and D. Spielman, “Analysis of Low Density Codes and Improved Designs using Irregular Graphs,” *ACM STOC*, 1998.

[17] M. Luby, M. Mitzenmacher, M Shokrollahi and D. Spielman,“Improved Low-Density Parity-Check Codes Using Irregular Graphs and Belief Propagation,” *IEEE ISIT*, 1998.

[18] D. MacKay, “Information Theory, Inference, and Learning Algorithms,” *Cambridge University Press*, 2003.

[19] N. Madras and G. Slade, ”The Self-Avoiding Walk,” *Birkhauser, Boston*, 1993.

[20] C. Moallemi and B. Van Roy, “Convergence of the min-sum message passing algorithm for quadratic optimization. *Stanford University Technical report*, 2006.

[21] J. Pearl, “Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference,” San Francisco, CA: Morgan Kaufmann, 1988.

[22] T. Richardson and R. Urbanke, “Modern Coding Theory,” *Book homepage http://lthcwww.epfl.ch/mct/index.php*, 2006.

[23] T. Richardson and R. Urbanke, “The capacity of low-density parity check codes under message-passing decoding,” *IEEE Trans. Inform. Theory*, 2001.

[24] S. C. Tatikonda and M. I. Jordan, “Loopy Belief Propagation and Gibbs Measure,” *Uncertainty in Artificial Intelligence*, 2002.

[25] M. Wainwright and M. Jordan, “ Graphical models, exponential families, and variational inference,” UC Berkeley, Dept. of Statistics, Technical Report 649, 2003.

[26] M. J. Wainwright, T. Jaakkola and A. S. Willsky, “Tree-based reparameterization framework for analysis of sum-product and related algorithms,” *IEEE Transactions on Information Theory*, 2003.

[27] M. J. Wainwright, T. S. Jaakkola and A. S. Willsky, “MAP estimation via agreement on (hyper)trees: Message-passing and linear-programming approaches,” *IEEE Transactions on Information Theory*, 51(11), 2005.

[28] Y. Weiss, “Correctness of local probability propagation in graphical models with loops,” *Neural Comput.*, vol. 12, pp. 1-42, 2000.

[29] Y. Weiss and W. Freeman, “Correctness of belief propagation in Gaussian graphical models of arbitrary topology,” *Neural Comput.*, Vol. 13, Issue 10, pp 2173-2200, 2001.

[30] Y. Weiss W. Freeman, “On the optimality of solutions of the max-product belief propagation algorithm in arbitrary graphs.,” *IEEE Trans. Info. Theory*, Vol. 47, pp 736-744, 2001.

[31] D. Weitz, “Counting independent sets up to the tree threshold,” *ACM STOC*, 2006.

[32] J. Yedidia, W. Freeman and Y. Weiss, “Generalized Belief Propagation,” *Mitsubishi Elect. Res. Lab.*, TR-2000-26, 2000.