# Maximum Weight Matching via Max-Product Belief Propagation

Mohsen Bayati
Department of EE
Stanford University
Stanford, CA 94305
Email: bayati@stanford.edu

Devavrat Shah
Departments of EECS & ESD
MIT
Boston, MA 02139
Email: devavrat@mit.edu

Mayank Sharma
Department of EE
Stanford University
Stanford, CA 94305
Email: msharma@stanford.edu

***Abstract*— The max-product "belief propagation" algorithm is an iterative, local, message passing algorithm for finding the maximum a posteriori (MAP) assignment of a discrete probability distribution specified by a graphical model. Despite the spectacular success of the algorithm in many application areas such as iterative decoding and computer vision which involve graphs with many cycles, theoretical convergence results are only known for graphs which are tree-like or have a single cycle.**

**In this paper, we consider a weighted complete bipartite graph and define a probability distribution on it whose MAP assignment corresponds to the maximum weight matching (MWM) in that graph. We analyze the fixed points of the max-product algorithm when run on this graph and prove the surprising result that even though the underlying graph has many short cycles, the max-product assignment converges to the correct MAP assignment. We also provide a bound on the number of iterations required by the algorithm.**

## I. INTRODUCTION

Graphical models (GM) are a powerful method for representing and manipulating joint probability distributions. They have found major applications in several different research communities such as artificial intelligence [11], statistics [8], error-control coding [6] and neural networks. Two central problems in probabilistic inference over graphical models are those of evaluating the *marginal* and *maximum a posteriori* (MAP) probabilities, respectively. In general, calculating the marginal or MAP probabilities for an ensemble of random variables would require a complete specification of the joint probability distribution. Further, the complexity of a brute force calculation would be exponential in the size of the ensemble. GMs assist in exploiting the dependency structure between the random variables, allowing for the design of efficient inference algorithms.

The belief propagation (BP) and max-product algorithms [11] were proposed in order to compute, respectively, the marginal and MAP probabilities efficiently. Comprehensive surveys of various formulations of BP and its generalization, the junction tree algorithm, can be found in [1], [20], [14]. BP-based message-passing algorithms have been very successful in the context of, for example, iterative decoding for turbo codes and in computer vision. The simplicity, wide scope of application and experimental success of belief propagation has attracted a lot of attention recently [1], [7], [12], [19].

BP is known to converge to the correct marginal/MAP probabilities on tree-like graphs [11] or graphs with a single loop [2], [16]. For graphical models with arbitrary underlying graphs, little is known about the correctness of BP. Partial progress consists of [17] where correctness of BP for Gaussian GMs is proved, [5] where an attenuated modification of BP is shown to work, and [12] where the iterative turbo decoding algorithm based on BP is shown to work in the asymptotic regime with probabilistic guarantees. To the best of our knowledge, little theoretical progress has been in resolving the question: Why does BP work on arbitrary graphs?

Motivated by the objective of providing justification for the success of BP on arbitrary graphs, we focus on the application of BP to the well-known combinatorial optimization problem of finding the Maximum Weight Matching (MWM) in a bipartite graph, also known as the "Assignment Problem". It is standard to represent combinatorial optimization problems, like finding the MWM, as calculating the MAP probability on a suitably defined GM which encodes the data and constraints of the optimization problem. Thus, the max-product algorithm can be viewed at least as a heuristic for solving the problem. In this paper, we study the performance of the max-product algorithm as a method for finding the MWM on a weighted complete bipartite graph.

Additionally, using the max-product algorithm for problems like finding the MWM has the potential of being an exciting application of BP in its own right. The assignment problem is extremely well-studied algorithmically. Attempts to find better MWM algorithms contributed to the development of the rich theory of network flow algorithms [4], [9]. The assignment problem has been studied in various contexts such as job-assignment in manufacturing systems [4], switch scheduling algorithms [10] and auction algorithms [3]. We believe that the max-product algorithm can be effectively used in high-speed switch scheduling where the distributed nature of the algorithm and its simplicity can be very attractive.

The main result of this paper is to show that the max-product algorithm for finding the MWM always finds the correct solution, as long as the solution is unique. Our proof is purely combinatorial and depends on the graph structure. We think that this result may lead to further insights in understanding how BP algorithms work when applied to other

optimization problems. The rest of the paper is organized as follows: In Section II, we provide the setup, define the assignment problem and describe the max-product algorithm for finding the MWM. Section III states and proves the main result of this paper. Finally, we discuss some implications of our results in Section IV.

## II. SETUP AND PROBLEM STATEMENT

In this section, we first define the problem of finding the MWM in a weighted complete bipartite graph and then describe the max-product BP algorithm for solving it.

### A. MAXIMUM WEIGHT MATCHING

Consider an undirected weighted complete bipartite graph $K_{n,n} = (V_1, V_2, E)$, where $V_1 = \{\alpha_1, \ldots, \alpha_n\}$, $V_2 = \{\beta_1, \ldots, \beta_n\}$ and $(\alpha_i, \beta_j) \in E$ for $1 \leq i, j \leq n$. Let each edge $(\alpha_i, \beta_j)$ have weight $w_{ij} \in \mathbb{R}$.

If $\pi = \{\pi(1), \ldots, \pi(n)\}$ is a permutation of $\{1, \ldots, n\}$ then the collection of $n$ edges $\{(\alpha_1, \beta_{\pi(1)}), \ldots, (\alpha_n, \beta_{\pi(n)})\}$ is called a *matching* of $K_{n,n}$. We denote both the permutation and the corresponding matching by $\pi$. The weight of matching $\pi$, denoted by $W_\pi$, is defined as

$$W_\pi = \sum_{1 \leq i \leq n} w_{i\pi(i)}.$$

Then, the Maximum Weight Matching (MWM), $\pi^*$, is the matching such that

$$\pi^* = \text{argmax}_\pi \; W_\pi.$$

**Note 1.** In this paper, we always assume that the weights are such that the MWM is unique. In particular, if the weights of the edges are independent, continuous random variables, then with probability 1, the MWM is unique.

Next, we model the problem of finding MWM as finding a MAP assignment in a graphical model where the joint probability distribution can be completely specified in terms of the product of functions that depend on at most two variables (nodes). For details about GMs, we urge the reader to see [8]. Now, consider the following GM defined on $K_{n,n}$: Let $X_1, \ldots, X_n, Y_1, \ldots, Y_n$ be random variables corresponding to the vertices of $K_{n,n}$ and taking values from $\{1, 2, \ldots, n\}$. Let their joint probability distribution, $p\left(\overline{X} = (x_1, \ldots, x_n); \overline{Y} = (y_1, \ldots, y_n)\right)$, be of the form:

$$p\left(\overline{X}, \overline{Y}\right) = \frac{1}{Z} \prod_{i,j} \psi_{\alpha_i \beta_j}(x_i, y_j) \prod_i \phi_{\alpha_i}(x_i) \phi_{\beta_i}(y_i), \quad (1)$$

where the pairwise compatibility functions, $\psi_{..}(\cdot, \cdot)$, are defined as

$$\psi_{\alpha_i \beta_j}(r, s) = \begin{cases} 0 & r = j \text{ and } s \neq i \\ 0 & r \neq j \text{ and } s = i \\ 1 & \text{Otherwise} \end{cases}$$

and the potentials at the nodes, $\phi_{.}(\cdot)$, are defined as

$$\phi_{\alpha_i}(r) = e^{w_{ir}}, \quad \phi_{\beta_j}(r) = e^{w_{rj}}, \quad \forall \;\; 1 \leq \; i, j, r, s \; \leq \; n.$$

The following claims are a direct consequence of these definitions.

*Claim 1:* For the GM as defined above, the joint density $p\left(\overline{X} = (x_1, \ldots, x_n), \overline{Y} = (y_1, \ldots, y_n)\right)$ is nonzero if and only if $\pi_\alpha(\overline{X}) = \{(\alpha_1, \beta_{x_1}), (\alpha_2, \beta_{x_2}), \ldots, (\alpha_n, \beta_{x_n})\}$ and $\pi_\beta(\overline{Y}) = \{(\alpha_{y_1}, \beta_1), (\alpha_{y_2}, \beta_2), \ldots, (\alpha_{y_n}, \beta_n)\}$ are both matchings and $\pi_\alpha(\overline{X}) = \pi_\beta(\overline{Y})$. Further, when nonzero, they are equal to $\frac{1}{Z} e^{2 \sum_i w_{ix_i}}$.

*Claim 2:* Let $(\overline{X}^*, \overline{Y}^*)$ be such that

$$(\overline{X}^*, \overline{Y}^*) = \arg\max\{p\left(\overline{X}, \overline{Y}\right)\}.$$

Then, the corresponding $\pi_\alpha(\overline{X}^*) = \pi_\beta(\overline{Y}^*)$ is the MWM in $K_{n,n}$.

Claim 2 implies that finding the MWM is equivalent to finding the maximum a posteriori (MAP) assignment on the GM defined above. Thus, the standard max-product algorithm can be used as an iterative strategy for finding the MWM. In fact we show that this strategy yields the correct answer. Next we describe the max-product algorithm (and the equivalent min-sum algorithm) for the GM defined above.

### B. MAX-PRODUCT ALGORITHM FOR $K_{n,n}$

We need some definitions and notations before we can describe the max-product algorithm. Consider the following.

*Definition 1:* Let $D \in \mathbb{R}^{n \times n}$ and $X, Y, Z \in \mathbb{R}^{n \times 1}$. Then the operations $*, \odot$ are defined as follows:

$$D * X = Z \iff z_i = \max_j d_{ij} x_j, \; \forall i, \quad (2)$$

$$X \odot Y = Z \iff z_i = x_i y_i, \; \forall i. \quad (3)$$

For $X_1, \ldots, X_m \in \mathbb{R}^{n \times 1}$,

$$\bigodot_{i=1}^{m} X_i = X_1 \odot X_2 \odot \ldots \odot X_n. \quad (4)$$

Define the compatibility matrix $\Psi_{\alpha_i \beta_j} \in \mathbb{R}^{n \times n}$ such that its $(r, s)$ entry is $\psi_{\alpha_i \beta_j}(r, s)$, for $1 \leq i, j \leq n$. Also, let $\Phi_{\alpha_i}, \Phi_{\beta_j} \in \mathbb{R}^{n \times 1}$ be the following:

$$\Phi_{\alpha_i} = [\phi_{\alpha_i}(1), \ldots, \phi_{\alpha_i}(n)]^t, \quad \Phi_{\beta_j} = [\phi_{\beta_j}(1), \ldots, \phi_{\beta_j}(n)]^t.$$

**Max-Product Algorithm.**

(1) Let $M^k_{\alpha_i \to \beta_j} = [m^k_{\alpha_i \to \beta_j}(1), m^k_{\alpha_i \to \beta_j}(2), \ldots, m^k_{\alpha_i \to \beta_j}(n)]^t \in \mathbb{R}^{n \times 1}$ denote the messages passed from $\alpha_i$ to $\beta_j$ in the iteration $k \geq 0$, for $1 \leq i, j \leq n$. Similarly, $M^k_{\beta_j \alpha_i}$ is the message vector passed from $\beta_j$ to $\alpha_i$ in the iteration $k$.

(2) Initially $k = 0$ and set the messages as follows. Let $M^0_{\alpha_i \to \beta_j} = [m^0_{\alpha_i \to \beta_j}(1) \ldots m^0_{\alpha_i \to \beta_j}(n)]^t$ and $M^0_{\beta_j \to \alpha_i} = [m^0_{\beta_j \to \alpha_i}(1) \ldots m^0_{\beta_j \to \alpha_i}(n)]^t$ where

$$m^0_{\alpha_i \to \beta_j}(r) = \begin{cases} e^{w_{ij}} & \text{if } r = i \\ 1 & \text{otherwise} \end{cases} \quad (5)$$

$$m^0_{\beta_i \to \alpha_j}(r) = \begin{cases} e^{w_{ji}} & \text{if } r = i \\ 1 & \text{otherwise} \end{cases} \quad (6)$$

(3) For $k \geq 1$, messages in iteration $k$ are obtained from messages of iteration $k-1$ recursively as follows:

$$M^k_{\alpha_i \to \beta_j} = \Psi^t_{\alpha_i \beta_j} * \left( (\bigodot_{l \neq j} M^{k-1}_{\beta_l \to \alpha_i}) \odot \Phi_{\alpha_i} \right)$$
$$M^k_{\beta_i \to \alpha_j} = \Psi^t_{\alpha_i \beta_j} * \left( (\bigodot_{l \neq j} M^{k-1}_{\alpha_l \to \beta_i}) \odot \Phi_{\beta_i} \right) \quad (7)$$

(4) Define the beliefs ($n \times 1$ vectors) at nodes $\alpha_i$ and $\beta_j$, $1 \leq i, j \leq n$, in iteration $k$ as follows.

$$b^k_{\alpha_i} = \left( \bigodot_l M^k_{b_l \to \alpha_i} \right) \odot \Phi_{\alpha_i}$$
$$b^k_{\beta_j} = \left( \bigodot_l M^k_{\alpha_l \to \beta_i} \right) \odot \Phi_{\beta_i} \quad (8)$$

(5) The estimated[1] MWM at the end of iteration $k$ is $\pi^k$, where $\pi^k(i) = \arg\max_{1 \leq j \leq n} \{b^k_{\alpha_i}(j)\}$, for $1 \leq i \leq n$.

(6) Repeat (3)-(5) till $\pi^k$ converges.

**Note 2.** For computational stability, it is often recommended that messages be normalized at every iteration. However, such normalization does not change the output of the algorithm. Since we are only interested in theoretically analyzing the algorithm, we will ignore the normalization step. Also, the messages are usually all initialized to one. Although the result doesn't depend on the initial values, setting them as defined above makes the analysis and formulas nicer at the end.

### C. MIN-SUM ALGORITHM FOR $K_{n,n}$

The max-product and min-sum algorithms can be seen to be equivalent by observing that the logarithm function is monotone and hence $\max_i \log(\alpha_i) = \log(\max_i \alpha_i)$. In order to describe the min-sum algorithm, we need to redefine $\Phi_{\alpha_i}, \Phi_{\beta_j}$, $1 \leq i, j \leq n$, as follows:

$$\Phi_{\alpha_i} = [w_{i1}, \ldots, w_{in}]^t, \quad \Phi_{\beta_j} = [w_{1j}, \ldots, w_{nj}]^t.$$

Now, the min-sum algorithm is exactly the same as max-product with the equations (6), (7) and (8) replaced by:

(a) Replace (6) by the following.

$$m^0_{\alpha_i \to \beta_j}(r) = \begin{cases} w_{ij} & \text{if } r = i \\ 0 & \text{otherwise} \end{cases} \quad (9)$$

$$m^0_{\beta_i \to \alpha_j}(r) = \begin{cases} w_{ji} & \text{if } r = i \\ 0 & \text{otherwise} \end{cases} \quad (10)$$

(b) Replace (7) by the following.

$$M^k_{\alpha_i \to \beta_j} = \Psi^t_{\alpha_i \beta_j} * \left( (\sum_{l \neq j} M^{k-1}_{\beta_l \to \alpha_i}) + \Phi_{\alpha_i} \right)$$
$$M^k_{\beta_i \to \alpha_j} = \Psi^t_{\alpha_i \beta_j} * \left( (\sum_{l \neq j} M^{k-1}_{\alpha_l \to \beta_i}) + \Phi_{\beta_i} \right) \quad (11)$$

[1]Note that, as defined, $\pi^k$ need not be a matching. Theorem 1 shows that for large enough $k$, $\pi^k$ is a matching and corresponds to the MWM.

(c) Replace (8) by the following.

$$b^k_{\alpha_i} = (\sum_l M^k_{\beta_l \to \alpha_i}) + \Phi_{\alpha_i}$$
$$b^k_{\beta_j} = (\sum_l M^k_{\alpha_l \to \beta_i}) + \Phi_{\beta_i} \quad (12)$$

**Note 3.** The min-sum algorithm involves only summations and subtractions compared to max-product which involves multiplications and divisions. Computationally, this makes the min-sum algorithm more efficient and hence very attractive.

## III. MAIN RESULT

Now we state and prove Theorem 1, which is the main contribution of this paper. Before proceeding further, we need the following definitions.

*Definition 2:* Let $\epsilon$ be the difference between the weights of the MWM and the second maximum weight matching; i.e.

$$\epsilon = W_{\pi^*} - \max_{\pi \neq \pi^*}(W_\pi).$$

Due to the uniqueness of the MWM, $\epsilon > 0$. Also, define $w^* = \max_{i,j}(|w_{ij}|)$.

*Theorem 1:* For any weighted complete bipartite graph $K_{n,n}$ with unique maximum weight matching, the max-product or min-sum algorithm when applied to the corresponding GM as defined above, converges to the correct MAP assignment or the MWM within $\lceil \frac{2nw^*}{\epsilon} \rceil$ iterations.

### A. PROOF OF THEOREM 1

We first present some useful notation and definitions. Consider $\alpha_i$, $1 \leq i \leq n$. Let $T^k_{\alpha_i}$ be the level-$k$ unrolled tree corresponding to $\alpha_i$, defined as follows: $T^k_{\alpha_i}$ is a weighted regular rooted tree of height $k+1$ with every non-leaf having degree $n$. All nodes have labels from the set $\{\alpha_1, \ldots, \alpha_n, \beta_1, \ldots, \beta_n\}$ according to the following recursive rule: (a) root has label $\alpha_i$; (b) the $n$ children of the root $\alpha_i$ have labels $\beta_1, \ldots, \beta_n$; and (c) the children of each non-leaf node whose parent has label $\alpha_r$ (or $\beta_r$) have labels $\alpha_1, \ldots, \alpha_{r-1}, \alpha_{r+1}, \ldots, \alpha_n$ (or $\beta_1, \ldots, \beta_{r-1}, \beta_{r+1}, \ldots, \beta_n$). The edge between nodes labeled $\alpha_i, \beta_j$ in the tree is assigned weight $w_{ij}$ for $1 \leq i, j \leq n$. Examples of such a tree for $n = 3$ are shown in the Figure 1.

**Note 4.** $T^k_{\alpha_i}$ is often called the level-$k$ *unwrapped graph* at node $\alpha_i$ corresponding to the GM under consideration. The unwrapped graph in general is constructed by replicating the pairwise compatibility functions $\psi_{\alpha_i \beta_j}(r, s)$ and potentials $\phi_{\alpha_i}(r), \phi_{\beta_j}(s)$, while preserving the local connectivity of the (possibly loopy) graph. They are constructed so that the messages received by node $\alpha_i$ after $k$ iterations in the actual graph are equivalent to those that would be received by the root $\alpha_i$ in the unwrapped graph, if the messages are passed up along the tree from the leaves to the root. Let $t^k_{\alpha_i}(r)$ be the weight of maximum weight matching in $T^k_{\alpha_i}$ which uses the edge $(\alpha_i, \beta_r)$ at the root. Here, we consider only the matchings on the tree under which all non-leaf nodes of $T^k_{\alpha_i}$ are the endpoints of exactly one edge.

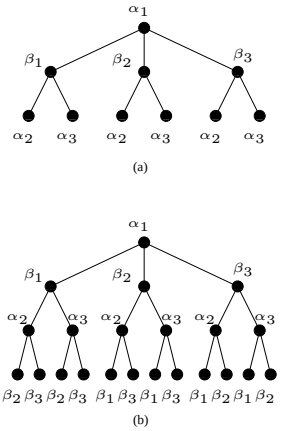


Fig. 1. When $n = 3$ (a) is $T^1_{\alpha_i}$ and (b) is $T^2_{\alpha_i}$.

Now, we state two important lemmas that will lead to the proof of Theorem 1. The first presents an important characterization of the min-sum algorithm while the second lemma relates the maximum weight matching over the unwrapped tree-graph and the MWM in $K_{n,n}$.

*Lemma 1:* At the end of the $k^{th}$ iteration of the min-sum algorithm, the belief at node $\alpha_i$ of $K_{n,n}$ is precisely $b^k_{\alpha_i} = [2t^k_{\alpha_i}(1)\dots 2t^k_{\alpha_i}(n)]^t$.

*Lemma 2:* If $\pi^*$ is the MWM of graph $K_{n,n}$ then for $k > \frac{2nw^*}{\epsilon}$ we have

$$\pi^*(i) = \arg\max_r \{t^k_{\alpha_i}(r)\}.$$

That is, for $k$ large enough, the maximum weight matching in $T^k_{\alpha_i}$ chooses the edge $(\alpha_i, \beta_{\pi^*(i)})$ at the root.

*Proof:* [Theorem 1] Consider the min-sum algorithm. Let $b^k_{\alpha_i} = [b^k_{\alpha_i}(1), \dots, b^k_{\alpha_i}(n)]^t$. Recall that $\pi^k = (\pi^k(i))$ where $\pi^k(i) = \arg\max_r\{b^k_{\alpha_i}(r)\}$. Then, by Lemmas 1 and 2, for $k > \frac{2nw^*}{\epsilon}$, $\pi^k = \pi^*$. ∎

Next, we present the proofs of Lemmas 1 and 2 in that order.

*Proof:* [Lemma 1] It is known [15] that under the min-sum (or max-product) algorithm, the vector $b^k_{\alpha_i}$ corresponds to the correct marginals for the root $\alpha_i$ of the MAP assignment on the GM corresponding to $T^k_{\alpha_i}$. The pairwise compatibility functions force the MAP assignment on this tree to be a matching. Now, each edge has two endpoints and hence its weight is counted twice in the weight of matching.

Next consider the $j^{th}$ entry of $b^k_{\alpha_i}$, $b^k_{\alpha_i}(j)$. By definition, it corresponds to the MAP assignment with the value of $\alpha_i$ at the root being $j$. That is, $(\alpha_i, \beta_j)$ edge is chosen in the tree at the root. From the above discussion, $b^k_{\alpha_i}(j)$ must be equal to $2t^k_{\alpha_i}(j)$. ∎

*Proof:* [Lemma 2] We prove the lemma by contradiction. Assume to contrary that for some $k > \frac{2nw^*}{\epsilon}$,

$$\pi^*(i) \neq \arg\max_r t^k_{\alpha_i}(r) \stackrel{\triangle}{=} \hat{i}, \quad \text{for some } i. \tag{13}$$

Then, let $\hat{i} = \pi^*(i_1)$ for $i_1 \neq i$. Let $\Lambda$ be the matching on $T^k_{\alpha_i}$ whose weight is $t^k_{\alpha_i}(\hat{i})$. We will modify $\Lambda$ and find $\Lambda'$ whose weight is more than $\Lambda$ and which connects $(\alpha_i, \beta_{\pi^*(i)})$ at the root instead of $(\alpha_i, \beta_{\pi^*(i_1)})$, thus contradicting with (13).

Consider paths $P_\ell$, $\ell \geq 0$, that contain edges from matchings[2] $\pi^*$ and $\Lambda$ alternatively on the tree $T^k_{\alpha_i}$ defined as follows. Let $\alpha_0 =$ root $\alpha_i$, $i_0 = i$ and $P_1 = (\alpha_0)$ be a single vertex path. Let $P_2 = (\beta_{\pi^*(i_0)}, \alpha_0, \beta_{\pi^*(i_1)})$, where $i_1$ is such that $\alpha_0 = \alpha_i$ is connected to $\beta_{\pi^* i_1}$ under $\Lambda$. For $r \geq 1$, define $P_{2r+1}$ and $P_{2r+2}$ recursively as follows:

$$P_{2r+1} = (\alpha_{i_{-r}}, P_{2r}, \alpha_{i_r}),$$
$$P_{2r+2} = (\beta_{\pi^*(i_{-r})}, P_{2r+1}, \beta_{\pi^*(i_{r+1})})$$

where $\alpha_{i_{-r}}$ is the node at level $2r$ to which the endpoint node $\beta_{\pi^*(i_{-r+1})}$ of path $P_{2r}$ is connected to under $\Lambda$, and $i_{r+1}$ is such that $\alpha_{i_r}$ at level $2r$ (part of $P_{2r+1}$) is connected to $\beta_{\pi^*(i_{r+1})}$ under $\Lambda$. Note that, by definition, such paths $P_\ell$ for $\ell \leq k$ exist since the tree $T^k_{\alpha_i}$ has $k+1$ levels and can support a path of length at most $2k$ as defined above.

Now consider the path $P_k$ of length $2k$. It's edges are alternately from $\Lambda$ and $\pi^*$. Let us refer to the edges of $\Lambda$ as the $\Lambda$-edges of $P_k$. Replacing the $\Lambda$-edges of $P_k$ with their complement in $P_k$ produces a new matching $\Lambda'$ in $T^k_{\alpha_i}$; this follows from the way the paths are constructed.

*Lemma 3:* The weight of matching $\Lambda'$ is strictly higher than that of $\Lambda$ on tree $T^k_{\alpha_i}$.

This completes the proof of Lemma 2 since Lemma 3 shows that $\Lambda$ is not the maximum weight matching on $T^k_{\alpha_i}$, leading to a contradiction. ∎

Now, we provide the proof of Lemma 3.

*Proof:* [Lemma 3] It suffices to show that the total weight of the $\Lambda$-edges is less than the total weight of their complement in $P_k$. Consider the projection $P'_k$ of $P_k$ in the graph $K_{n,n}$. $P'_k$ can be decomposed into a union of a set of simple cycles $\{C_1, C_2, \dots, C_m\}$ and at most one even length path $Q$ of length at most $2n$. Since each simple cycle has at most $2n$ vertices and the length of $P_k$ is $2k$,

$$m \geq \frac{2k}{2n} = \frac{k}{n}. \tag{14}$$

Consider one of these simple cycles, say $C_s$. Construct the matching $\pi'$ in $K_{n,n}$ as follows: (i) For $\alpha_l \in C_s$, select edges incident on $\alpha_l$ that belong to $\Lambda$. Such edges exist by the property of the path $P_k$ that contains $C_s$. (ii) For $\alpha_l \notin C_s$, connect it according to $\pi^*$, that is, add the edge $(\alpha_l, \beta_{\pi^*(l)})$.

Now $\pi' \neq \pi^*$ by construction. Since the MWM is unique, the definition of $\epsilon$ gives us

$$W_{\pi'} \leq W_{\pi^*} - \epsilon.$$

But, $W_{\pi^*} - W_{\pi'}$ is exactly equal to the total weight of the non-$\Lambda$-edges of $C_s$ minus the total weight of the $\Lambda$-edges of $C_s$. Thus,

$$\text{weight of } \Lambda\text{-edges of } C_s - \text{weight of rest of } C_s = -(W_{\pi^*} - W_{\pi'}) \quad \leq \quad -\epsilon. \tag{15}$$

[2]The matching $\pi^*$ is defined on $K_{n,n}$ but can be naturally projected to the tree $T^k_{\alpha_i}$. Hence, when we refer to 'edges of matching $\pi^*$', we mean edges in $K_{n,n}$ or the tree $T^k_{\alpha_i}$ depending on the context.

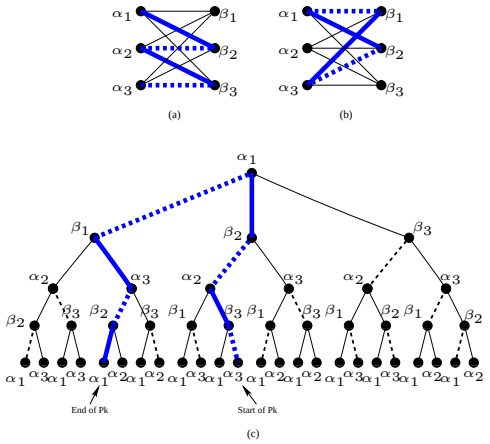


Fig. 2. Projection of the path $P_k$ is decomposed to (a): path Q of length 4 and (b) cycle $C_1$ of length 4.

Since the path $Q$ is of even length, either the first edge or the last edge is an $\Lambda$-edge. Without loss of generality, assume it is the last edge. Then, let

$$Q = (\beta_{\pi^*(i_{j_1})}, \alpha_{i_{j_1}}, \beta_{\pi^*(i_{j_2})}, \ldots, \beta_{\pi^*(i_{j_l})}, \alpha_{i_{j_l}}, \beta_{\pi^*(i_{j_{l+1}})}).$$

Now consider the cycle

$$C = (\beta_{\pi^*(i_{j_1})}, \alpha_{i_{j_1}}, \beta_{\pi^*(i_{j_2})}, \ldots, \beta_{\pi^*(i_{j_l})}, \alpha_{i_{j_l}}, \beta_{\pi^*(i_{j_1})}).$$

Alternate edges of $C$ are from the maximum weight matching $\pi^*$. Hence, using the same argument as above, we obtain

$$\begin{aligned} & \text{weight of } \Lambda\text{-edges of } Q - \text{weight of rest of } Q \\ = & \sum_{1\le r\le l} w_{i_{j_r}\pi^*(i_{j_{r+1}})} - \sum_{1\le r\le l} w_{i_{j_r}\pi^*(i_{j_r})} \\ \le & -\epsilon + |w_{i_{j_l}\pi^*(i_{j_1})}| + |w_{i_{j_l}\pi^*(i_{j_{l+1}})}| \\ \le & -\epsilon + 2w^*. \end{aligned} \quad (16)$$

From (14)-(16), we obtain that for matchings $\Lambda'$ and $\Lambda$ in $T^k_{\alpha_i}$:

$$\begin{aligned} \text{weight of } \Lambda - \text{weight of } \Lambda' & \le -(m+1)(\epsilon) + 2w^* \\ & \le -\frac{k}{n}\epsilon + 2w^* < 0. \end{aligned} \quad (17)$$

This completes the proof of Lemma 3. ■

## IV. DISCUSSION AND CONCLUSION

In this paper, we proved that the max-product algorithm converges to the desirable fixed point in the context of MWM for bipartite graph, even in the presence of loops. This result has a twofold impact. First, it will possibly open avenues for demystification of the max-product algorithm. Second, the same approach may provably work for other combinatorial optimization problems and possibly lead to better algorithms.

Though, the algorithm described in the paper may seem complicated, we have managed to simplify[3] it using the regularity of the structure of the problem. In the simplified algorithm, each node needs to perform $O(n)$ addition-subtraction operations in each iteration. Since $O(n)$ iterations are required in the worst case, for finite $w^*$ and $\epsilon$, the algorithm requires $O(n^3)$ operations at the most. This is comparable with the best known MWM algorithm. Furthermore, the distributed nature of the max-product algorithm makes it particularly suitable for networking applications like switch scheduling where scalability is a necessary property.

Future work will consist of trying to extend our result to finding the MWM in a general graph, as our current arguments do not carry over[4]. Also, we would like to obtain tighter bounds on the running time of the algorithm since simulation studies show that the algorithm runs much faster on average than the worst case bound obtained in this paper.

[3]More details will appear in a technical report

## ACKNOWLEDGMENT

While working on this paper the first and the last author were supported by grant AF F49620-01-1-0365.

## REFERENCES

[1] S. M. Aji and R. J. McEliece, "The Generalized Distributive Law," *IEEE Trans. Inform. Theory*, Vol. 46, pp. 325-343, 2000.
[2] S. M. Aji, G. B. Horn and R. J. McEliece, "On the Convergence of Iterative Decoding on Graphs with a Single Cycle," *Proc. 1998 IEEE Int. Symp. Information Theory*, Cambridge, MA, p. 276, 1998.
[3] D. Bertsekas and J. Tsitsiklis, "Parallel and Distributed Computation: Numerical Methods," *Prentice Hall*, Englewood Cliffs, N. J., 1989.
[4] J. Edmonds and R. Karp, "Theoretical Improvements in Algorithmic Efficiency for Network Flow Problems," *Jour. of the ACM*, Vol. 19, pp 248-264, 1972.
[5] B.J. Frey, R. Koetter, "Exact inference using the attenuated max-product algorithm", in *Advanced Mean Field Methods: Theory and Practice*, ed. Manfred Opper and David Saad, MIT Press, 2000.
[6] R. G. Gallager, "Low Density Parity Check Codes," *MIT Press*, Cambridge, MA, 1963.
[7] G. B. Horn, "Iterative Decoding and Pseudocodewords," *Ph.D. dissertation*, Dept. elect. Eng., Calif. Inst. Technol., Pasadena, CA, 1999.
[8] S. Lauritzen, "Graphical models," *Oxford University Press*, 1996.
[9] E. Lawler, "Combinatorial Optimization: Networks and Matroids", *Holt, Rinehart and Winston*, New York, 1976.
[10] N. McKeown, V. Anantharam and J. Walrand, "Achieving 100 % Throughput in an Input-Queued Switch," *Infocom*, Vol. 1, pp 296-302, 1996.
[11] J. Pearl, "Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference," San Francisco, CA: Morgan Kaufmann, 1988.
[12] T. Richardson and R. Urbanke, "The Capacity of Low-Density Parity Check Codes under Message-Passing Decoding," *IEEE Trans. Info. Theory*, Vol. 47, pp 599-618, 2001.
[13] M. Wainwright, T. Jaakkola and A. Willsky, "Tree Consistency and Bounds on the Performance of the Max-Product Algorithm and its Generalizations," *Statistics and Computing*, Vol. 14, pp 143-166, 2004.
[14] M. Wainwright, M. Jordan, "Graphical models, exponential families, and variational inference," *Tech. Report*, Dept. of Stat.,University of Cal., Berkeley, 2003.
[15] Y. Weiss, "Belief propagation and revision in networks with loops," *MIT AI Lab.*, Tech. Rep. 1616, 1997.
[16] Y. Weiss, "Correctness of local probability propagation in graphical models with loops," *Neural Comput.*, Vol. 12, pp. 1-42, 2000.
[17] 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
[18] 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.
[19] J. Yedidia, W. Freeman and Y. Weiss, "Generalized Belief Propagation," *Mitsubishi Elect. Res. Lab.*, TR-2000-26, 2000.
[20] J. Yedidia, W. Freeman and Y. Weiss, "Understanding Belief Propagation and its Generalizations," *Mitsubishi Elect. Res. Lab.*, TR-2001-22, 2000.

[4]A key fact in the proof of lemma was the property that bipartite graphs do not have odd cycles.