Iterative Message Passing Algorithm for Bipartite Maximum Weighted Matching

Yuan-sheng Cheng
Department of Electrical Engineering
University of Southern California
Los Angeles, CA 90089-2565, USA
Email: yuanshec@usc.edu

Michael Neely
Department of Electrical Engineering
University of Southern California
Los Angeles, CA 90089-2565, USA
Email: mjneely@usc.edu

Keith M. Chugg
Department of Electrical Engineering
University of Southern California
Los Angeles, CA 90089-2565, USA
Email: chugg@usc.edu

***Abstract*— We derive iterative message passing update rules for solving the bipartite maximum weighted matching problem. It is shown that if the optimal matching solution is unique, the algorithm converges to this optimal solution at a rate comparable to the algorithm of Bayati et. al. It is shown that the two algorithms are both standard messages passing, but on dual graphs of each other. Also, the algorithm presented here requires less storage. We also provide a method to use the proposed algorithm to solve the integer Maximal Weighted Matching problem – i.e., where the optimal solution is generally not unique.**

I. Introduction

Message passing algorithms on various graphical models have been widely used and demonstrated in different areas, like modern coding theory, DNA sequences analysis, artificial intelligence, etc. It is well known [1] [2] that if the graphical models are cycle free, message passing algorithms converge to the optimal MPF (marginalized product of functions) solutions. However, if the graphical model has cycles, message passing algorithms experimentally perform well in many instances (like Turbo Decoding), while the optimality and convergence properties are still open problems.

The bipartite Maximum Weighted Matching (MWM) problem has been well studied in combinational optimization and network theory [11], as they provide several applications. In particular, it was shown in [12] [7] that scheduling according to an MWM for every timeslot will stabilize an $N \times N$ packet switch. Furthermore, the MWM problem has important applications to optimal routing, scheduling, and resource allocation in wireless ad-hoc networks, where distributed implementation is crucial [8] [9].

The Hungarian method [5] finds an optimal solution for the bipartite weighted matching problem in $O(n^3)$ operations. More recently, the Auction algorithm proposed by Bertsekas [6] solves the same problem as a suboptimal way. It finds a solution within $n\epsilon$ of being optimal in $O(\frac{n^2 \max\{|w_{ij}|\}}{\epsilon})$ operations, where ϵ is an arbitrary parameter.

Motivated by the analogies between the formulation of bipartite weighted matching and the decoding of modern codes, we derived the iterative message passing algorithm described in this paper. The final update rules are concise and can be implemented in a distributed way. Through simulation, we observed that the algorithm converged to the optimal MWM solution when the solution was unique. While we were attempting to prove this convergence property, Bayati, Shah, and Sharma demonstrated an iterative message-passing algorithm to solve the same problem and also proved the convergence properties that we observed empirically [3]. In this paper we apply the the proof technique of [3] to our algorithm to verify optimality and show the relationship between the algorithm of [3], which we will refer to as the BSS algorithm after its authors, and our algorithm.

In the next section, we describe the mathematical model we used and derive the message passing update rules in Section III. Convergence is proved in Section IV and the algorithm is compared to the BSS algorithm in Section VI. In Section V we introduce the modified algorithm for on integer MWM applications. Section VII contains discussion and suggestions for future work.

II. Mathematical model

Let $G = (T, B, E)$ be an symmetric complete bipartite graph. T and B are sets of n nodes. We label them as $T = \{T_1, T_2, T_3, ..., T_n\}$ and $B = \{B_1, B_2, B_3, ..., B_n\}$. We label all edge as $(t, b) \in E$, where $t \in T$ and $b \in B$. A subset $M \subseteq E$ is said to be a matching if no two edges in M are incident to the same node. If each edge (T_i, B_j) is associated with a real number w_{ij}, called a weight, finding a matching for which the sum of the weights of edges is maximum is called Maximum Weighted Matching.

For a particular matching M and for each edge (T_i, B_j), let x_{ij} represents an indicator variable that is equal to one if (T_i, B_j) is in M, and zero else. Let X be the n by n matrix which (i, j) component is x_{ij}. When considering only full matches, X corresponds to a permutation matrix. So Maximum Weighted Matching can be defined as:

$$X^* = \arg\max_{X \in P} \sum_{0 \le i,j \le n} w_{ij} x_{ij} \quad (1)$$

where P is the set of permutation matrices. We call X^* the maximum configuration of MWM.

Consider the function $f(X) : \mathbb{R}^{n \times n} \to \mathbb{R}$:

$$f(X) = \prod_{i,j}^{n} f_{ij}(x_{ij}) \prod_{k=1}^{n} \chi_{T_k}(\{x_{kj}\}_{j=1}^{n}) \prod_{l=1}^{n} \chi_{B_l}(\{x_{il}\}_{i=1}^{n}) \quad (2)$$

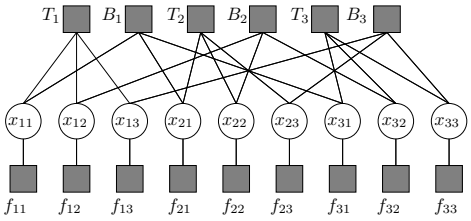

Fig. 1. Factor Graph for $f(X)$ with $n = 3$

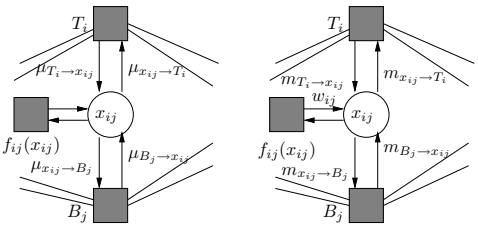

Fig. 2. A detailed view of the messages of the iMPA for MWM

where

$$\chi_{T_k}(\cdot) = \begin{cases} 1 & \text{, if exactly one of } \{x_{kj}\}_{j=1}^n \text{ variables is 1} \\ 0 & \text{, otherwise} \end{cases}$$

$$\chi_{B_l}(\cdot) = \begin{cases} 1 & \text{, if exactly one of } \{x_{il}\}_{i=1}^n \text{ variables is 1} \\ 0 & \text{, otherwise} \end{cases}$$

$$f_{ij}(x_{ij}) = \exp(w_{ij}x_{ij}) \tag{3}$$

By the definition of indicator functions $\chi_{T_k}(\cdot)$s and $\chi_{B_l}(\cdot)$s, $f(X)$ is non-zero if and only if X is a permutation matrix. Since $\max$ and $\log$ functions are interchangeable, the maximum configuration of $f(X)$ is equal to the X^*. The final expression become:

$$X^* = \arg\max_{X\in P} \sum_{0\le i,j\le n} w_{ij}x_{ij} = \arg\max_{X\in P} f(X) \tag{4}$$

III. Max-sum update rules

While there are many equivalent notational conventions for graphical models, we use factor graphs in this paper [1]. The factor graph associated with (2) is shown in Fig. 1. The graph is very similar to Tanner graph representing a parity check code with the difference being that the constraint nodes associate with $\chi_{T_i}(\cdot)$s and $\chi_{B_j}(\cdot)$s enforce a "exactly one 1" rule instead of even parity. Fig. 1 shows the example for the case $n = 3$.

Our goal may be viewed as being equivalent to the Maximum Likelihood Sequence Detection (MLSD) with the associated constraints on $\{x_{ij}\}$. Thus, we may apply message passing update rules on max-product semiring [1], [4]. We consider a schedule where messages start from the variable nodes and are passed toward constraint nodes and upwards and downwards iteratively. By symmetry, for each variable x_{ij}, we define the corresponding messages $\mu^{(k)}_{x_{ij}\to T_i}(x_{ij})$, $\mu^{(k)}_{x_{ij}\to B_j}(x_{ij})$, $\mu^{(k)}_{B_j\to x_{ij}}(x_{ij})$ and $\mu^{(k)}_{B_j\to x_{ij}}(x_{ij})$ (See Fig. 2), where the superscripts k are the iteration number of the messages and all messages are real functions defined on $\{0, 1\}$ and may be viewed as tables or vectors of two values. By convention, the initial messages are set to 1:

$$\mu^{(-1)}_{T_i\to x_{ij}}(x_{ij}) = \mu^{(-1)}_{B_j\to x_{ij}}(x_{ij}) = 1, \qquad x_{i,j} = 0, 1$$

Applying max-product message update rules [1], we get:

$$\mu^{(k)}_{x_{ij}\to T_i}(x_{ij}) = f_{ij}(x_{ij}) \cdot \mu^{(k-1)}_{B_j\to x_{ij}}(x_{ij})$$

$$\mu^{(k)}_{x_{ij}\to B_j}(x_{ij}) = f_{ij}(x_{ij}) \cdot \mu^{(k-1)}_{T_i\to x_{ij}}(x_{ij})$$

$$\mu^{(k)}_{B_j\to x_{ij}}(x_{ij}) = \max_{x_{ij}} \left[\chi_{B_j}(\{x_{ij}\}_{i=1}^n) \prod_{m\neq i} \mu^{(k-1)}_{x_{mj}\to B_j}(x_{mj}) \right]$$

$$\mu^{(k)}_{T_i\to x_{ij}}(x_{ij}) = \max_{x_{ij}} \left[\chi_{T_i}(\{x_{ij}\}_{j=1}^n) \prod_{l\neq j} \mu^{(k-1)}_{x_{il}\to T_i}(x_{il}) \right]$$

where each of the above holds for $x_{ij} = 0, 1$. At each iteration k, we compute

$$D^{(k)}_{(ij)}(x_{ij}) = f_{ij}(x_{ij}) \cdot \mu^{(k)}_{T_i\to x_{ij}}(x_{ij}) \cdot \mu^{(k)}_{B_j\to x_{ij}}(x_{ij})$$

Our estimation of x_{ij} based on kth iteration is: We choose $x^{(k)}_{ij} = 1$ if $D^{(k)}_{(ij)}(1) > D^{(k)}_{(ij)}(0)$, otherwise $x^{(k)}_{ij} = 0$.

The above is standard max-product message-passing [1], [4]. We now provide some simplifications based on standard methods used in the iterative message-passing literature (cf. [4]). First, since the $\log(\cdot)$ function is monotonically increasing, we can take $\log(\cdot)$ on both sides of update rules without changing the results. This places the algorithm into max-sum form. Furthermore, message normalization [4] can be used to reduce the required memory by a factor of 2; i.e., storing the difference between the messages for 0 and 1 values at each step of update. This yields call the normalized (scalar) messages $m^{(k)}_{T_i\to x_{ij}}$, $m^{(k)}_{B_j\to x_{ij}}$, $m^{(k)}_{x_{ij}\to T_i}$ and $m^{(k)}_{x_{ij}\to B_j}$ (See Fig. 2) – e.g., $m^{(k)}_{B_j\to x_{ij}} = \log(\mu^{(k)}_{B_j\to x_{ij}}(1)/\mu^{(k)}_{B_j\to x_{ij}}(0))$. The result is **Algorithm 1**:

---

1) Initialization:

$$m^{(0)}_{T_i\to x_{ij}} = -\max_{m\neq i} w_{mj} \tag{5}$$

$$m^{(0)}_{B_j\to x_{ij}} = -\max_{l\neq j} w_{il} \tag{6}$$

2) At kth iteration:

$$m^{(k)}_{T_i\to x_{ij}} = -\max_{m\neq j}\{m^{(k-1)}_{B_m\to x_{im}} + w_{im}\} \tag{7}$$

$$m^{(k)}_{B_j\to x_{ij}} = -\max_{l\neq i}\{m^{(k-1)}_{T_l\to x_{lj}} + w_{lj}\} \tag{8}$$

3) At kth iteration, we computes:

$$M^{(k)}_{ij} = m^{(k)}_{T_i\to x_{ij}} + m^{(k)}_{B_j\to x_{ij}} + w_{ij} \tag{9}$$

Estimation of x_{ij} based on kth iteration is: $\hat{x}^{(k)}_{ij} = 1$ if $M^{(k)}_{ij} > 0$ and $\hat{x}^{(k)}_{ij} = 1$ otherwise.

Note that when a message passes through a variable node x_{ij}, it increases it value by w_{ij} and the message that leaves the constraint nodes (any top or bottom node) is simply the negative value of the maximum of other incoming messages at the same constraint node. Also note that **Algorithm 1** could be implemented in a distributed manner.

IV. Proof of Optimality

Our approach follows that of Bayati et. al. [3]. Let $w^* = \max\{|w_{ij}|\}$ and ϵ be the difference between maximum weight matching and the second largest weight matching. Then we have the following results:

Theorem 1: For a symmetric complete bipartite graph, if the maximum weighted matching is unique, Algorithm 1 converges to the optimal solution after $k > \frac{3nw^*}{\epsilon}$. In other words, the decision $\hat{x}_{ij}^{(k)}$ is the same as $[X^*]_{ij}$ after k-th iterations.

To prove theorem 1, we use the computation tree first introduced by Wiberg [10]. The computation tree $\overline{G}_{ij}$ for variable x_{ij} in factor graph G is a factor graph constructed by creating a root node $\overline{x}_{ij}$ corresponding to x_{ij} in G and then recursively adding edges and nodes to $\overline{G}_{ij}$ that correspond to the messages passed in the message passing algorithm. Each vertex in $\overline{G}_{ij}$ corresponding a unique vertex in G. The kth level computation tree, $\overline{G}_{ij}^{(k)}$, represents the computation for k iterations. i.e., applying the algorithm on the computation tree from leaves to root, the message obtained by x_{ij} on the root is exactly the messages obtained by x_{ij} on kth iteration. Fig. 3 shows $\overline{G}_{11}^{(2)}$ for example.

To simplify the proof, in the following parts of this section, "nodes" on the computation tree represent only the constraint nodes($\{T_i\}$,$\{B_j\}$), not including variable nodes($\{x_{ij}\}$). Also, an edge in $\overline{G}_{ij}^{(k)}$ will comprise the two edges between nodes and the corresponding vertex. For example, the edge e shown in Fig. 3 comprises the connection between B_2 and T_1 and includes x_{12}. Thus, the edge corresponding to x_{ml}, has an associated weight $\mathrm{w}(e) = w_{ml}$. We also define the subtree $H_{ij}^{(k)}(e)$ as the combination of e and its descendants part of tree (see the area encloded by the dashed box in Fig. 3 for example). In particular, if e is the rooted edge, $H_{ij}^{(k)}(e)$ is just the $\overline{G}_{ij}^{(k)}$. Furthermore, we define a *regular tree matching* on $H_{ij}^{(k)}(e)$ to be a matching in which all the nodes on $H_{ij}^{(k)}(e)$ are incident to exactly one link of matching, except possible the leaf nodes. In other words, in a regular tree matching the constraints at each node, except possibly the leaf nodes, are satisfied – i.e., exactly only one connected variable takes the 1 value. Finally, we define the function $\mathrm{mwm}(e)$, mapping edges to real numbers, as the largest weight of all regular tree matchings on $H_{ij}^{(k)}(e)$ that including e as part of the matching. Similarly, $\mathrm{mwm}'(e)$ refers to the largest weight of regular tree matchings on $H_{ij}^{(k)}(e)$ that do not include e.

Now consider how **Algorithm 1** operates on $\overline{G}_{ij}^{(k)}$. In general, the messages comes from leaves (bottom nodes) towards the root (top edge), and there are only three types of updates.

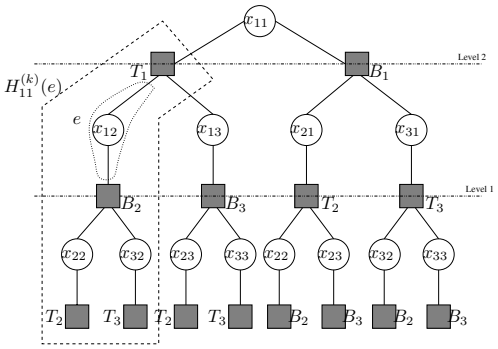

Fig. 3. Computation Tree $\overline{G}_{11}^{(k)}$

When messages pass through a node, the upwards output message is the negative of the maximum of input messages. When messages pass through an edge, it simply increased by the weight of the edge. At the root edge of the tree, $M_{ij}^{(k)}$ is simply the sum of w_{ij} and the two upward messages from T_i and B_j. Then we have the following lemma:

Lemma 1: At each node in the $\overline{G}_{ij}^{(k)}$, with output edge o and input edges $i_1, i_2, ...i_l$,: (i) the message at input i_m is equal to $\mathrm{mwm}(i_m) - \mathrm{mwm}'(i_m)$ and (ii) the output message is equal to $\mathrm{mwm}(o) - \mathrm{mwm}'(o) - \mathrm{w}(o)$..

We prove Lemma 1 by induction. For all nodes in the level 1 (refer to Fig. 3, the nodes B_2, B_3, T_2, T_3), the two statements are clearly true by inspection. Then suppose the two statements are true for all nodes at level l. For nodes at level $l+1$, note that every input message in this level comes from the output message of certain node at level l. By the induction hypothesis, for any input edge i_m, the input message is: $\mathrm{mwm}(i_m) - \mathrm{mwm}'(i_m) - \mathrm{w}(i_m) + \mathrm{w}(i_m) = \mathrm{mwm}(i_m) - \mathrm{mwm}'(i_m)$. Also, for a node at level $l+1$, with output edge o and input edges $i_1, i_2, ...i_p$, the following equations can be derived by the properties of regular tree matching:

$$\mathrm{mwm}(o) = w(o) + \sum_{m=1}^{p} \mathrm{mwm}'(i_m)$$

$$\mathrm{mwm}'(o) = \max_{1\le m\le p} \{\mathrm{mwm}(i_m) + \sum_{r\neq m} \mathrm{mwm}'(i_r)\}$$

From the update rules operated at nodes:

$$\begin{aligned}\text{output message} &= -\max_{1\le m\le p} \{\mathrm{mwm}(i_m) - \mathrm{mwm}'(i_m)\} \\ &= \sum_{m=1}^{p} \mathrm{mwm}'(i_m) \\ &\quad - \max_{1\le m\le p} \{\mathrm{mwm}(i_m) + \sum_{r\neq m} \mathrm{mwm}'(i_r)\} \\ &= \mathrm{mwm}(o) - \mathrm{mwm}'(o) - \mathrm{w}(o)\end{aligned}$$

This completes the proof of Lemma 1 and it is straightforward to reach the following Lemma.

Lemma 2: Let $e_r = (T_i, B_j)$ be the root edge of $\overline{G}_{ij}^{(k)}$. Then $M_{ij}^{(k)}(x_{ij}) = \mathrm{mwm}(e_r) - \mathrm{mwm}'(e_r)$.

Note the rooted edge e_r separates $\overline{G}_{ij}^{(k)}$ into two branches. Let $H_{ij}^{(k)}(R)$ be the combination of edge e_r and the right branch of $\overline{G}_{ij}^{(k)}$, and $H_{ij}^{(k)}(L)$ be the combination of edge e_r and the right branch of $\overline{G}_{ij}^{(k)}$. Then $\mathrm{mwm_R}(e_r)$ refers to the largest weight over all regular tree matching of $\overline{H}_{ij}^{(k)}(R)$ that including e_r as part of a matching. Furthermore, $\mathrm{mwm'_R}(e_r)$, $\mathrm{mwm'_L}(e_r)$ and $\mathrm{mwm_L}(e_r)$ are defined in similar ways. Then:

$$\begin{aligned} M_{x_{ij}}^{(k)}(x_{ij}) =& w_{ij} + (\mathrm{mwm_L}(e_r) - \mathrm{mwm'_L}(e_r) - w_{ij}) \\ &+ (\mathrm{mwm_R}(e_r) - \mathrm{mwm'_R}(e_r) - w_{ij}) \\ =& (\mathrm{mwm_L}(e_r) + \mathrm{mwm_R}(e_r)) - w_{ij}) \\ &- (\mathrm{mwm'_L}(e_r) + \mathrm{mwm'_R}(e_r)) \\ =& \mathrm{mwm}(e_r) - \mathrm{mwm'}(e_r) \end{aligned}$$

This completes the proof of Lemma 2.

Now we are ready to prove Theorem 1. The proof is by contradiction. Assume $e_r = (T_i, B_j)$ is not part of the MWM in the original bipartite graph but $\hat{x}_{ij}^{(k)} = 1$ – i.e., at the kth iteration, $\mathrm{mwm}(e_r) > \mathrm{mwm'}(e_r)$. We will show this is impossible when $k > \frac{3nw^*}{\epsilon}$. Let the set Ω represent all edges in $\overline{G}_{ij}^{(k)}$ that are used to construct the matching with weight $\mathrm{mwm}(e_r)$ (so Ω is a regular tree matching). By the definition of $\mathrm{mwm}(e_r)$, $e_r \in \Omega$. Let X^* be the MWM in the original bipartite graph. Let Ω^* be the mapping of X^* on the $\overline{G}_{ij}^{(k)}$, then clearly Ω^* is a regular tree matching on $\overline{G}_{ij}^{(k)}$ (this follows the assumption the bipartite graph is complete and symmetric) and $e_r \notin \Omega^*$. We then construct a path P in the $\overline{G}_{ij}^{(k)}$ in the following way: Let $P^{(0)} = e = \{T^{(0)}, B^{(0)}\}$, where $T^{(0)} = T_i$ and $B^{(0)} = B_j$ and for $l \geq 1$, we augment P by:

- If l is odd, let $P^{(l)}$ be $\{B^{(l)}, P^{(l-1)}, T^{(l)}\}$, where $(T^{(l-1)}, B^{(l)})$, $(T^{(l)}, B^{(l-1)}) \in \Omega^*$.
- If l is even, let $P^{(l)}$ be $\{T^{(l)}, P^{(l-1)}, B^{(l)}\}$, where $(T^{(l-1)}, B^{(l)})$, $(T^{(l)}, B^{(l-1)}) \in \Omega$.

Continue augmenting the path P until it reaches the leaves of $\overline{G}_{ij}^{(k)}$, then edges of P belong to Ω and Ω^* alternatively. (Since both Ω^* and Ω are regular tree matchings, we ensure such a path exists). We modify Ω to Ω' by adding edges belonging to $P \bigcap \Omega^*$ and removing those from $P \bigcap \Omega$. Note Ω' is also a regular tree matching of $\overline{G}_{ij}^{(k)}$ and $e_r \notin \Omega'$. If we can show $\mathrm{weight}(\Omega') > \mathrm{weight}(\Omega)$, then:

$$\mathrm{mwm'}(e_r) \geq \mathrm{weight}(\Omega') > \mathrm{weight}(\Omega) = \mathrm{mwm}(e_r)$$

and a contradiction is obtained.

Since the only difference between Ω' and Ω is edges in P, the only thing left to show is $\mathrm{weight}(P \bigcap \Omega^*) > \mathrm{weight}(P \bigcap \Omega)$. This follows from the following Lemma.

Lemma 3: If P is constructed by the way mentioned above, $\mathrm{weight}(P \bigcap \Omega^*) > \mathrm{weight}(P \bigcap \Omega)$ for $k > \frac{3nw^*}{\epsilon}$.

Note that P has length $2k+1$. The trick is we can map P onto the original bipartite graph G, and call the new path (on G) P'. The corresponding edges on P and P' have the same weights. In general, P' composes a series of cycles C_1, C_2, ..., C_l, and a path $Q' = (e_1, e_2, ..., e_m)$, where $e_1, e_2, ...$ are a series of edges. Since the length of cycle on original graph is at most $2n$, we have $l \geq \lfloor \frac{2k+1}{2n} \rfloor$. Note edges in each cycle alternatively belong to Ω^* and Ω. By the arguments in [3], if the MWM is unique, in each cycle C_i, the weight of edges $\in \Omega^*$ minus the weight of edges $\in \Omega$ is greater that or equal to ϵ.

For odd k, where m is odd (since each cycle has even length) and $e_1, e_m \in \Omega^*$, we can add a new edge e_{m+1} to connect the two ends of Q' to form a cycle. Let Q be the mapping of Q' from G to $\overline{G}_{ij}^{(k)}$, then by the same cycle arguments above:

$$\begin{aligned} \mathrm{weight}(Q \bigcap \Omega^*) - \mathrm{weight}(Q \bigcap \Omega) &\geq \epsilon + \mathrm{weight}(e_{m+1}) \\ &\geq \epsilon - w^* \end{aligned}$$

Considering the whole path P:

$$\begin{aligned} \mathrm{weight}(P \bigcap \Omega^*) - \mathrm{weight}(P \bigcap \Omega) &\geq \lfloor \frac{2k+1}{2n} \rfloor \epsilon + \epsilon - w^* \\ &> \frac{k}{n}\epsilon - w^* \\ &> 0 \text{ for } k > \frac{nw^*}{\epsilon} \end{aligned}$$

For even k, we can get a similar results for $k > \frac{3nw^*}{\epsilon}$. Considering both cases, we complete proof of Lemma 3.

In the case e_r belongs to X^* but $\hat{x}_{ij}^{(k)} = 0$, the proof is similar and omitted fro brevity. Thus, this completes the proof of Theorem 1.

V. Modified Algorithm for Integer Weights

The above optimality result applies only for weighted bipartite graphs that have unique maximum (e.g., the algorithm doesn't work if all edges have the same weight). In applications like $N \times N$ packet switches [7], the optimal solution is usually not unique. However, using the fact that queueing backlogs are always integers, one possible way to solve this is to modify w_{ij} before applying the iMPA algorithm. For example, for integer w_{ij}, we consider

$$w'_{ij} = w_{ij} + \frac{2^{(i-1)n+j}}{2^{n^2+1}}$$

Then apply iMPA on the new weights w'_{ij}. Since for any permutation $\pi(\cdot)$, we have:

$$\begin{aligned} \sum_{i=1}^{n} w'_{i\pi(i)} &= \sum_{i=1}^{n} w_{i\pi(i)} + \sum_{i=1}^{n} \frac{2^{(i-1)n+\pi(i)}}{2^{n^2+1}} \\ &< \sum_{i=1}^{n} w_{i\pi(i)} + \sum_{i=1}^{\infty} \frac{1}{2^n} = \sum_{i=1}^{n} w_{i\pi(i)} + 1 \end{aligned}$$

Because the original weights are integers, two weighted matchings that differ must differ by an integer and the modification doesn't change the relative ordering of their total weights. Furthermore, since $(i-1)n + j < n^2 + 1$, we can represent $\frac{2^{(i-1)n+j}}{2^{n^2+1}}$ as 2^{-l}, where l is a positive integer. Since every different pair of (i, j) corresponds to a unique l, it is clear that all combinations of these terms will yield different values

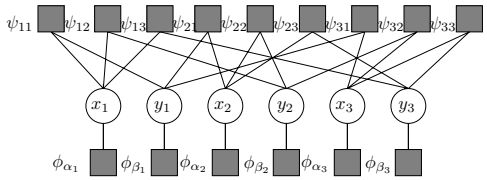

Fig. 4. Factor Graph for the BSS model with $n = 3$

– i.e., weighted matchings originally having the same weights will have different weights after modification. The above arguments show the correctness of this modification and with this modification, the algorithm finds an optimal solution for arbitrary integer weighted matching problems.

VI. Comparison with the BSS Algorithm

In this section we compare our algorithm with the one proposed by Bayati et. al. [3]. The BSS algorithm may also be viewed as running standard iterative message passing (max-product or max-sum) on a graph that differs from that of Fig. 1. The factor graph shown in Fig. 4[1] corresponds to the BSS model. Note that in Fig. 1, the variables are binary and represent the possible edge connections being active while the constraints are that only one edges be active per row or column. The BSS model in Fig. 4 is the dual graph where the variables are n-ary, corresponding to the row (column) location of the active edge and the constraints are that no two of these variables corresponding to rows (columns) take the same value. The algorithms can be compared in terms of computation and memory requirements:

Computation per Iteration: In each iteration, **Algorithm 1** needs one $\max$ operation and n additions per node. Considering $\max(\cdot)$ an $O(n)$ operation, **Algorithm 1** has totally $2n \cdot O(n) + 2n^2 = O(n^2)$ complexity per iteration. The BSS algorithm presented in [3] requires $O(n^2)$ operations per node, but it is also claimed that this can be reduced to $O(n)$, yielding $O(n^2)$ complexity per iteration. Thus, the computational complexity of the two algorithms is roughly the same.

Memory Requirements: By inspecting equations (9) and (10), at each node, if we express the n outgoing messages as a vector, the elements of the vector have only two values and exactly one of the element has different value from others. So we need 3 units of memory per node (two values and the location of the different one) and totally $6n$ units for **Algorithm 1**. By comparison, all the messages in BSS algorithm are $n \times 1$ vectors (each has two values in their elements). So it appears to require $3n$ units of memory per node and $O(n^2)$ total memory is required[2]. Thus, the algorithm presented herein appears to be more memory efficient.

VII. Conclusion and Discussion

The algorithm presented and the BSS algorithm have complexity of $O(\frac{n^3 \max |w_{ij}|}{\epsilon})$. The Auction algorithm [6] finds a solution within $n\epsilon'$ with complexity $O(\frac{n^2 \max |w_{ij}|}{\epsilon'})$. Letting $\epsilon' = \epsilon/n$, where ϵ is the same definition in this paper, then the complexity becomes $O(\frac{n^3 \max |w_{ij}|}{\epsilon})$. Thus, our algorithm, the BSS, and the Auction algorithm have the same complexity. Further research is required to better understand of the relation between the Auction algorithm and standard message-passing on graphs.

It is also interesting that the traditional [11] way to solve the weighted matching problem is to apply primal-dual algorithm to a linear program (LP) yielding the optimal "Hungarian method". This optimality is obtained despite the fact that, in general, a LP has fractional optimal solutions since the constraint $x_{ij} \in \{0, 1\}$ is relaxed. However, for the LP corresponding to the MWM problem, these fractional optimal solutions will not be the basic feasible solutions. The relation of integer linear optimization and the max-product algorithm has been recently studied [13] and the optimality of the algorithm described in this paper provides a good example. An interesting future direction is the characterization of such LP problems and the applicability of message-passing algorithms.

[1] We use the notation in [3] and express their model as a factor graph.

[2] Memory requirements are not discussed in [3].

References

[1] Frank R. Kschischang, Brendan J. Frey, and Hans-Andrea Loeliger,"Factor Graphs and Sum-Product Algorithm,"*IEEE Transactions on Information Theory,* Vol.47, pp.498-519, Feb 2001.

[2] Srinivas M. Aji and Robert J. McEliece,"The Generalized Distributive Law,"*IEEE Transactions on Information Theory,* Vol.46, pp.325-343, Mar 2000.

[3] M. Bayati, D. Shah, M. Sharma, "Maximum Weight Matching via Max-Product Belief Propagation," *IEEE International Symposium on Information Theory,* Sept 2005.

[4] Keith Chugg, Achilleas Anastasopoulos, Xiaopeng Chen,"Iterative Detection," Kluwer Academic Publishers, 2001.

[5] H.W. Kuhn,"The hungarian method for the assignment problem," *Naval Research Logistics Quarterly,* 1995, 83-87.

[6] Bertsekas, D.P.,"A Distributed Asynchronous Relaxation Algorithm for the Assignment Problem," *Proc. 24th IEEE Conf. Dec. & Contr.,* 1985.

[7] N.McKeown and V. Anantharam and J. Walrand,"Achieving 100% Throughput in an Input-Queued Switch," *Proc. INFOCOM* Vol 1, pp 296-302, March 1996.

[8] M. J. Neely and E. Modiano and C. E Rohrs, "Dynamic Power Allocation and Routing for Time Varying Wireless Networks," *IEEE Journal on Selected Areas in Communications,* Vol 23, Issue 1, pp.93-109, Jan 2005.

[9] X. Wu and R. Srikant, "Regulated Maximal Matching: A Distributed Scheduling Algorithm for Multi-Hop Wireless Networks with Node-Exclusive Spectrum Sharing," Submitted to *IEEE Conference on Decision and Control,* 2005.

[10] N.Wiberg,"Codes and decoding on general graphs," Ph.D. dessertation, Linköping Univ., Linköping, Sweden,1996.

[11] Eugene Lawler, "Combinatorial Optimization,"Dover Publications,2001.

[12] L.Tassiulas and A.Ephremides, "Stability properties of contrained queing systems an scheduling policies for maximum throughput in multihop radio networks.," *IEEE Transactions on Automatic Control,* Vol.37, no.12, Dec. 1992.

[13] Jon Feldman, Martin J. Wainwright and David R. Karger, "Using Linear Programming to Decode Binary Linear Codes,"*IEEE Transactions on Information Theory,* Vol 51, Issue 3, pp 954-972, March 2005.