Max-Product for Maximum Weight Matching: Convergence, Correctness, and LP Duality

Mohsen Bayati, Devavrat Shah, and Mayank Sharma

***Abstract*—Max-product "belief propagation" (BP) is an iterative, 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 combinatorial optimization, which involve graphs with many cycles, theoretical results about both the correctness and convergence of the algorithm are known in only a few cases (see Section I for references).**

In this paper, we will prove the correctness and convergence of max-product for finding the maximum weight matching (MWM) in bipartite graphs. Even though the underlying graph of the MWM problem has many cycles, somewhat surprisingly we show that the max-product algorithm converges to the correct MWM as long as the MWM is unique. We provide a bound on the number of iterations required and show that for a graph of size n, the computational cost of the algorithm scales as $O(n^3)$, which is the same as the computational cost of the best known algorithms for finding the MWM.

We also provide an interesting relation between the dynamics of the max-product algorithm and the *auction* algorithm, which is a well-known distributed algorithm for solving the MWM problem.

***Index Terms*—Auction algorithm, belief propagation (BP), distributed optimization, linear programming, Markov random fields, maximum weight matching (MWM), max-product algorithm, message-passing algorithms, min-sum algorithm.**

I. Introduction

GRAPHICAL models (GMs) 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 [16], statistics [12], error-correcting codes [8], [11], [17], and neural networks. Two central problems in probabilistic inference over GMs 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 algorithms.

The belief propagation (BP) and max-product algorithms [16] 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 [2], [24], [18]. BP-based message-passing algorithms have been very successful in the context of, for example, iterative decoding for turbo codes, computer vision, and finding satisfying assignments for random satisfiability problems. The simplicity, wide scope of application, and experimental success of BP has attracted a lot of attention recently [2], [11], [15], [17], [25].

BP (or max-product) is known to converge to the correct marginal (or MAP) probabilities on graphs with no cycles [16]. For graphs with a single cycle, the convergence and correctness of BP are rigorously analyzed in [1], [20]. For GMs with arbitrary underlying graphs, little is known about the correctness of BP. Partial progress consists of: the correctness of BP for Gaussian GMs was proved in [22], an attenuated modification of BP is shown to work [10], the iterative turbo decoding algorithm based on BP is shown to work in the asymptotic regime with probabilistic guarantees in [17], and fixed points of BP are shown to be locally optimal in [23], [9]. To the best of our knowledge, limited theoretical progress has been made in understanding when BP works on graphs with cycles?

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

Manuscript received November 30, 2005; revised August 8, 2007. The work of D. Shah was supported by the National Science Foundation under CAREER Grant CNS-0546590. The material in this paper was presented at the IEEE International Symposium on Information Theory, Adelaide, Australia, September 2005. This work was performed while M. Bayati was with the Department of Electrical Engineering, Stanford University, Stanford, CA.

M. Bayati is with Microsoft Research, Redmond, WA 98052 USA (e-mail: mohsenb@microsoft.com).

D. Shah is with the Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, Cambridge, MA 02139 USA (e-mail: devavrat@mit.edu).

M. Sharma is with the IBM T. J. Watson Research Center, Yorktown Heights, NY 10598 USA (e-mail: mxsharma@us.ibm.com).

Communicated by P. L. Bartlett, Associate Editor for Pattern Recognition, Statistical Learning and Inference.

Color version of Figure 2 in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TIT.2007.915695

extremely well studied algorithmically. Attempts to find better MWM algorithms contributed to the development of the rich theory of network flow algorithms [9], [13]. The assignment problem has been studied in various contexts such as job assignment in manufacturing systems [9], switch scheduling algorithms [14], and auction algorithms [7]. Recently, we used the max-product algorithm effectively in high-speed switch scheduling and wireless scheduling where the distributed nature of the algorithm and its simplicity are very attractive for implementation purposes [5].

A. Our Results

The main result of this paper is to show that the max-product algorithm for MWM always finds the correct solution, as long as the solution is unique. Our proof is purely combinatorial and uses only bipartite nature of the graph. We think that this result and in particular our methods may lead to further insights in understanding how BP algorithms work when applied to a more general class of optimization problems.

We show that the complexity of this algorithm scales as $O(n^3 w^*/\varepsilon)$, where n is the size of the graph, ε is the difference between weight of the unique MWM and the second MWM, and w^* is the maximal value of edge weight. Thus, the running time of max-product for MWM is essentially the same as the running time of both the best centralized algorithm (assuming w^*, ε constant), and the auction algorithm proposed by Bertsekas.

Somewhat interestingly, we find that the dynamics of the auction algorithm and the max-product algorithm are essentially the same and this observation leads to a precise relation between these two algorithms. The auction algorithm with a relaxation method can find the MWM (as well as a good approximate solution) even in the absence of a unique solution. The above connection between auction and max-product suggests a modified version of the max-product algorithm. We show that the fixed point of this modified max-product algorithm coincides with a good approximate solution and can lead to an MWM when the parameters are chosen properly. In general, this suggests a method to obtain a (deterministic) modification of max-product which can converge to a good approximate solution even when the problem has multiple solutions. We believe that this heuristic should also be of interest for other optimization problems.

B. Organization

The rest of the paper is organized as follows. In Section II, we provide the setup, define the MWM problem (or assignment problem) and describe a version of the max-product algorithm (or the min-sum algorithm) for finding the MWM. In this paper, we will use the term max-product and min-sum interchangeably for the same algorithm. Essentially, the min-sum algorithm is obtained from the max-product algorithm by replacing each variable with its logarithm.

Section III states and proves the main result of this paper. Section IV presents a simplification of the max-product algorithm and evaluates its computational cost. Section V discusses the relation between the max-product algorithm and the celebrated auction algorithm. The auction algorithm solves the dual of a linear programming (LP) relaxation for the MWM problem. Our result suggests the possibility of a deeper connection between max-product and dual algorithms for optimization problems. Finally, we discuss some implications of our results in Section VI.

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 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 $1 \le i, j \le n$. Let each edge (α_i, β_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 π. The weight of matching π, denoted by W_π, is defined as

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

Then, the MWM π^* 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. Otherwise, one may make the MWM unique by adding sufficiently small independent random *noise* to each of the edge weights.

Next, we model the problem of finding MWM as finding a MAP assignment in a GM 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 [12]. 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

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

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

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

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

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

and Z is the normalization constant. We note that the pairwise potential ensures that the following two constraints are satisfied for any $(\overline{X}, \overline{Y})$ with positive probability: a) If node α_i is matched to node β_j (i.e., $X_i = j$), then node β_j must be match to node α_i (i.e., $Y_j = i$). b) If node α_i is not matched to β_j (i.e., $X_i \neq j$), then node β_j must not be matched to node α_i (i.e., $Y_j \neq i$). These two constraints encode the property that the support of the above defined probability distribution is restricted to matchings.

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 $\frac{1}{Z} e^{2\sum_i w_{ix_i}}$.

When, $p(\overline{X}, \overline{Y}) > 0$, then the product of $\phi_\cdot(\cdot)$'s makes the probability a monotone function of the sum of the edge weights that are part of the corresponding matching. Formally, we state the following claim.

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 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. Before proceeding further, we provide an illustrative example of the above defined GM.

Example 1: Consider a complete bipartite graph with $n = 2$. The random variables X_i, $i = 1, 2$ correspond to the index of the β node to which α_i is connected under the GM. Similarly, the random variables Y_i, $i = 1, 2$ correspond to the index of the α node to which β_i is connected. For example, $X_1 = 1$ means that α_1 is connected to β_1. The pairwise potential function $\psi_{\cdot\cdot}$ encodes the matching constraints. For example, $(X_1, X_2; Y_1, Y_2) = (1, 2; 1, 2)$ corresponds to the matching where α_1 is connected to β_1 and α_2 is connected to β_2. This is encoded (and allowed) by $\psi_{\cdot\cdot}$: in this example, $\psi_{\alpha_1\beta_2}(X_1, Y_2) = \psi_{\alpha_1\beta_2}(1, 2) = 1$, etc. On the other hand, $(X_1, X_2; Y_1, Y_2) = (1, 2; 2, 1)$ is not a matching as α_1 connects to β_1 while β_1 connects to α_2. This is imposed by the following: $\psi_{\alpha_1\beta_1}(X_1, Y_1) = \psi_{\alpha_1\beta_1}(1, 2) = 0$. We recommend that the reader study this example in further detail in order to gain familiarity with the above defined GM.

B. Min-Sum Algorithm for $K_{n,n}$

The max-product and min-sum algorithms can be seen to be equivalent. In this paper, we will look at the min-sum version for the GM defined above. The max-product version and its equivalence to min-sum algorithm are given in [3]. Now, the min-sum algorithm is described as follows.

Min-sum algorithm.

(1) Let

$$M^k_{\alpha_i \to \beta_j} = \left[m^k_{\alpha_i \to \beta_j}(1), m^k_{\alpha_i \to \beta_j}(2), \ldots, m^k_{\alpha_i \to \beta_j}(n)\right]^t \in \mathbb{R}^{n \times 1}$$

denote the messages passed from α_i to β_j in the iteration $k \geq 0$, for $1 \leq i, j \leq n$. Similarly, $M^k_{\beta_j \to \alpha_i}$ is the message vector passed from β_j to α_i in the iteration k.

(2) Initially $k = 0$ and set the messages as follows. Let

$$M^0_{\alpha_i \to \beta_j} = \left[m^0_{\alpha_i \to \beta_j}(1) \ldots m^0_{\alpha_i \to \beta_j}(n)\right]^t$$

and

$$M^0_{\beta_j \to \alpha_i} = \left[m^0_{\beta_j \to \alpha_i}(1) \ldots m^0_{\beta_j \to \alpha_i}(n)\right]^t$$

where

$$m^0_{\alpha_i \to \beta_j}(r) = \begin{cases} w_{ij} & \text{if } r = i \\ 0 & \text{otherwise} \end{cases} \quad (2)$$

$$m^0_{\beta_i \to \alpha_j}(r) = \begin{cases} w_{ji} & \text{if } r = i \\ 0 & \text{otherwise.} \end{cases} \quad (3)$$

(3) For $k \geq 1$, messages in iteration k are obtained from messages of iteration $k-1$ recursively as follows: for all α_i, β_j, and all $1 \leq q, p \leq n$

$$m^k_{\alpha_i \to \beta_j}(q) = \max_{1 \leq p \leq n} \psi_{\alpha_i\beta_j}(p, q) \left[\sum_{\ell \neq j} m^{k-1}_{\beta_\ell \to \alpha_i}(p) + w_{ip}\right]$$

$$m^k_{\beta_j \to \alpha_i}(p) = \max_{1 \leq q \leq n} \psi_{\alpha_i\beta_j}(p, q) \left[\sum_{\ell \neq i} m^{k-1}_{\alpha_\ell \to \beta_j}(q) + w_{qj}\right]. \quad (4)$$

(4) Define the beliefs ($n \times 1$ vectors) at nodes α_i and β_j, $1 \leq i, j \leq n$, in iteration k as follows: $1 \leq r \leq n$

$$b^k_{\alpha_i}(r) = w_{ir} + \sum_\ell m^k_{\beta_\ell \to \alpha_i}(r),$$

$$b^k_{\beta_j}(r) = w_{rj} + \sum_\ell m^k_{\alpha_\ell \to \beta_j}(r). \quad (5)$$

(5) The estimated[1] MWM at the end of iteration k is π^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 π^k converges.

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.

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

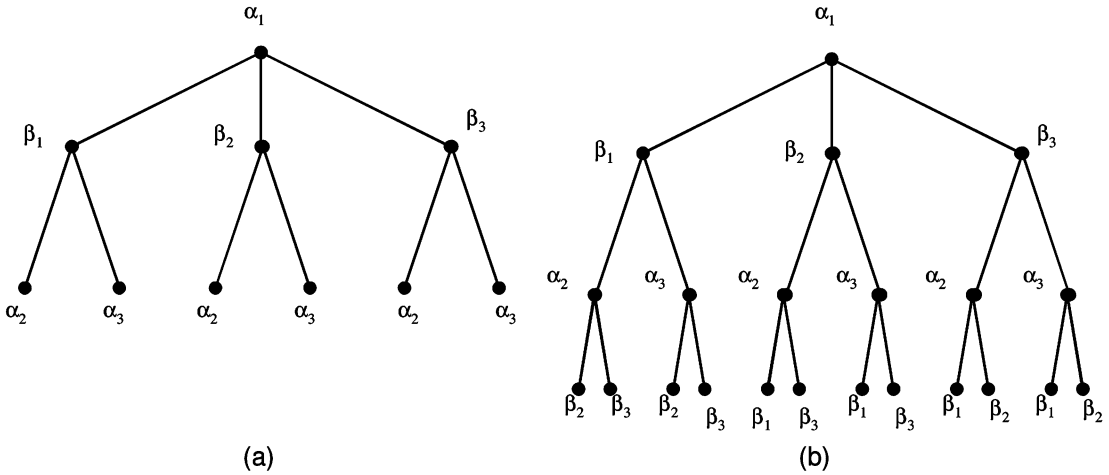

Fig. 1. When $n = 3$ (a) is $T^1_{\alpha_i}$ and (b) is $T^2_{\alpha_i}$.

Definition 1: Let ε be the difference between the weights of the MWM and the second MWM; i.e.,

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

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

Theorem 1: For any weighted complete bipartite graph $K_{n,n}$ with unique MWM, 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 $\lceil \frac{2nw^*}{\varepsilon} \rceil$ iterations.

A. Proof of Theorem 1

We first present some useful notation and definitions. Consider α_i, $1 \leq i \leq n$. Let $T^k_{\alpha_i}$ be the level-k unrolled tree corresponding to α_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) the root has label α_i; b) the n children of the root α_i have labels $\beta_1, \ldots, \beta_n$; and c) the children of each non-leaf node whose parent has label α_r (or β_r) have labels $\beta_1, \ldots, \beta_{r-1}, \beta_{r+1}, \ldots, \beta_n$ (or $\alpha_1, \ldots, \alpha_{r-1}, \alpha_{r+1}, \ldots, \alpha_n$). The edge between nodes labeled α_i, β_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 Fig. 1.

Note 2. $T^k_{\alpha_i}$ is often called the level-k *computation tree* at node α_i corresponding to the GM under consideration. The computation tree 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 original graph. They are constructed so that the messages received by the node α_i after k iterations in the actual graph are equivalent to those that would be received by the root α_i in the computation tree, if the messages are passed up along the tree from the leaves to the root. The computation tree has been used in most of the previous work on analyzing the BP algorithm, e.g., [8], [10], [20], [22], [23].

A collection Λ of edges in the computation tree is called a T-*matching* if no two edges of Λ are adjacent in the tree (Λ is a matching in the computation tree) and each non-leaf node is the endpoint of exactly one edge from Λ. Let $t^k_{\alpha_i}(r)$ be the weight of a maximum weight T-matching in $T^k_{\alpha_i}$ which uses the edge (α_i, β_r) at the root.

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

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

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

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

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

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

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

Proof of Lemma 1: It is known [21] that under the min-sum (or max-product) algorithm, the vector $b^k_{\alpha_i}$ corresponds to the correct max-marginals for the root α_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 T-matching. Now, each edge has two endpoints and hence its weight is counted twice in the weight of the T-matching.

Next, consider the jth entry of $b^k_{\alpha_i}$, $b^k_{\alpha_i}(j)$. By definition, it corresponds to the MAP assignment with the value of α_i at the root being j. That is, the edge (α_i, β_j) is chosen at the root in

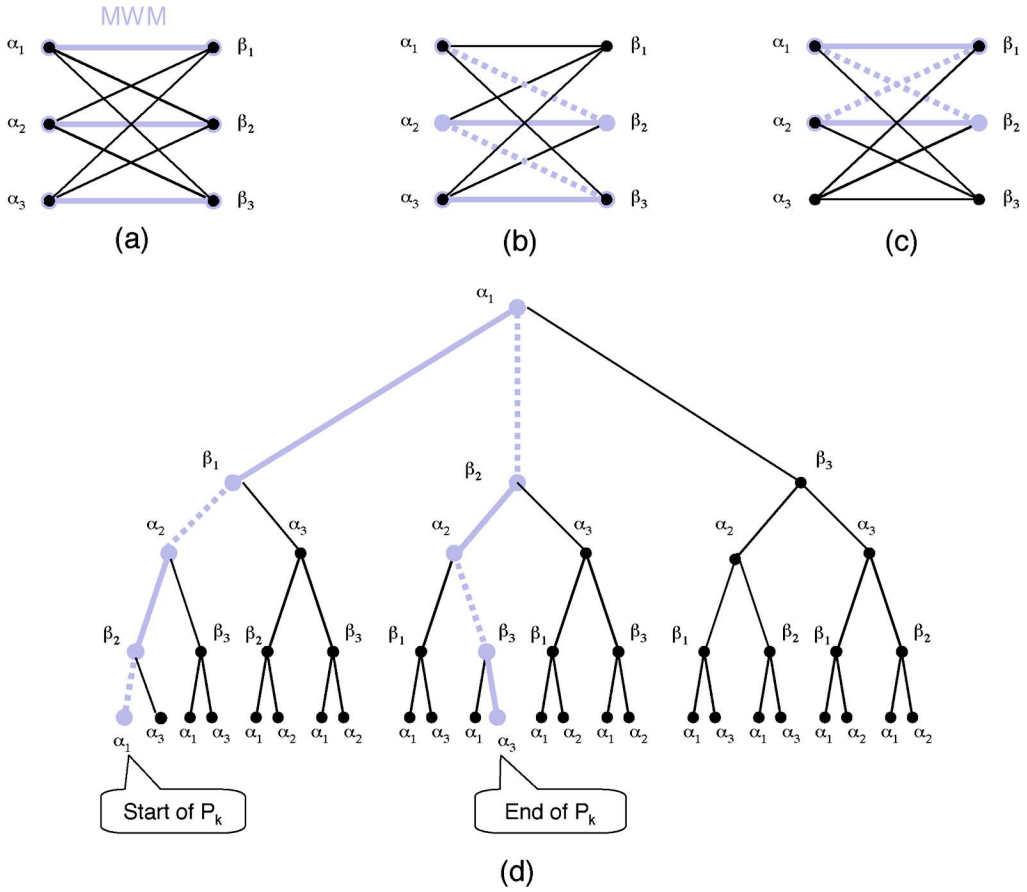

Fig. 2. Consider a graph with MWM shown in (a). Projection of the path P_k for $k = 4$ as shown in (d) is decomposed to (b): path Q of length 4 and (c): cycle C_1 of length 4. The *dashed* edges belong to Λ while *bold* edges belong to Π^*.

the tree. From the above discussion, $b^k_{\alpha_i}(j)$ must be equal to $2t^k_{\alpha_i}(j)$. □

Lemma 2 is the main step in proving Theorem 1 and its proof covers more than one page. Before going into the details of proof, let us give a high level description of it. Consider the computation tree $(T^k_{\alpha_i})$ rooted at vertex (α_i) and look at maximum weight T-matching on it. We assume that at the root, maximum weight T-matching of $T^k_{\alpha_i}$ does not choose the correct edge $(\alpha_i, \beta_{\pi*(i)})$. Then we use the property of T-matchings that each vertex is connected to exactly one of its neighbors to construct a new T-matching on computation tree. This new matching is going to have larger total weight if depth of the computation tree is large enough. This last step uses an *augmenting path* based argument for this matching problem. The above will contradict the assumption that decision at the root is incorrect, and proves Lemma 2.

Proof of Lemma 2: Assume the contrary that for some $k > \frac{2nw^*}{\varepsilon}$,

$$\pi^*(i) \neq \arg\max_r t^k_{\alpha_i}(r) \triangleq \hat{i}, \quad \text{for some } i. \tag{6}$$

Then, let $\hat{i} = \pi^*(i_1)$ for $i_1 \neq i$. Let Λ be the T-matching on $T^k_{\alpha_i}$ whose weight is $t^k_{\alpha_i}(\hat{i})$. We will modify Λ and find Λ' whose weight is more than Λ and which connects $(\alpha_i, \beta_{\pi^*(i)})$ at the root instead of $(\alpha_i, \beta_{\pi^*(i_1)})$, thus contradicting (6).

First note that the set of all edges of $T^k_{\alpha_i}$ whose projection in $K_{n,n}$ belongs to π^* is a T-matching which we denote by Π^*. Now consider paths P_ℓ, $\ell \geq 0$ in $T^k_{\alpha_i}$, that contain edges from Π^* and Λ alternatively defined as follows. Let $\alpha_{i_0} = \text{root } \alpha_i$, $i_0 = i$, and $P_0 = (\alpha_{i_0})$ be a single vertex path. Let $P_1 = \big(\beta_{\pi^*(i_0)}, \alpha_{i_0}, \beta_{\pi^*(i_1)}\big)$, where i_1 is such that $\alpha_{i_0} = \alpha_i$ is connected to $\beta_{\pi^*(i_1)}$ under Λ. For $r \geq 1$, define P_{2r} and P_{2r+1} recursively as follows:

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

where $\alpha_{i_{-r}}$ is the node at level $2r$ to which the endpoint node $\beta_{\pi^*(i_{-r+1})}$ of path P_{2r-1} is connected to under Λ, and i_{r+1} is such that α_{i_r} at level $2r$ (part of P_{2r}) is connected to $\beta_{\pi^*(i_{r+1})}$ under Λ. Note that, by definition, such paths P_ℓ for $0 \leq \ell \leq 2k$ 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.

Example 2: Fig. 2(d) provides an example of such a path. The corresponding bipartite graph has $n = 3$ with its MWM shown in Fig. 2(a). Fig. 2(d) shows $T^3_{\alpha_1}$, the computation tree for node α_1, till depth $k+1 = 4$. A path, P_4 is highlighted by thick edges alternatively complete and bold (edges from Π^*) and *dashed*

(edges from Λ). In the figure, $P_0 = (\alpha_1)$; $P_1 = (\beta_1, \alpha_1, \beta_2)$; $P_2 = (\alpha_2, \beta_1, \alpha_1, \beta_2, \alpha_2) = (\alpha_3, P_2, \alpha_2)$; and so on. Finally

$$P_4 = (\alpha_1, \beta_2, \alpha_2, \beta_1, \alpha_1, \beta_2, \alpha_2, \beta_3, \alpha_3) = C_1 \cup Q$$

where $C_1 = (\alpha_1, \beta_1, \alpha_2, \beta_2, \alpha_1)$ is a cycle of length 4 (see Fig. 2(c)) and $Q = (\alpha_1, \beta_2, \alpha_2, \beta_3, \alpha_3)$ is a path of length 4 (see Fig. 2(b)).

Now consider the path P_k of length $2k$. Its edges are alternately partitioned into edges from Λ and edges Π^*. Let us refer to the edges of Λ as the Λ-edges of P_k. Replacing the Λ-edges of P_k with their complement in P_k (all Π^* edges of P_k) produces a new matching Λ' in $T^k_{\alpha_i}$; this follows from the way the paths are constructed. Note that Λ' is exactly equal to Λ on $T^k_{\alpha_i}$ except along the path P_k where it uses edges from Π^*.

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

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

Now, we provide the proof of Lemma 3.

Prooof of Lemma 3: It suffices to show that the total weight of the Λ-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 $\{C_1, C_2, \ldots, C_m\}$ 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 $2k$

$$m \geq \frac{2k}{2n} = \frac{k}{n}. \tag{7}$$

Consider one of these simple cycles, say C_s. Construct the matching π' in $K_{n,n}$ as follows: i) For $\alpha_l \in C_s$, select edges incident on α_l that belong to Λ. 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 π^*, that is, add the edge $(\alpha_l, \beta_{\pi^*(l)})$.

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

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

However, $W_{\pi^*} - W_{\pi'}$ is exactly equal to the total weight of the Π^*-edges of C_s, denoted by $W_{\Pi^*(C_s)}$, minus the total weight of the Λ-edges of C_s, denoted by $W_{\Lambda(C_s)}$. Thus

$$\begin{aligned} W_{\Lambda(C_s)} - W_{\Pi^*(C_s)} &= -(W_{\pi^*} - W_{\pi'}) \\ &\leq -\varepsilon. \end{aligned} \tag{8}$$

Since the path Q is of even length, either the first edge or the last edge is an Λ-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 MWM π^*. Hence, using the same argument as above, we obtain

$$\begin{aligned} W_{\Lambda(Q)} - W_{\Pi^*(Q)} &= \sum_{1 \leq r \leq l} w_{i_{j_r} \pi^*(i_{j_{r+1}})} - \sum_{1 \leq r \leq l} w_{i_{j_r} \pi^*(i_{j_r})} \\ &\leq -\varepsilon + |w_{i_{j_l} \pi^*(i_{j_1})}| + |w_{i_{j_l} \pi^*(i_{j_{l+1}})}| \\ &\leq -\varepsilon + 2w^*. \end{aligned} \tag{9}$$

From (7)–(9), we obtain that for T-matchings Λ' and Λ in $T^k_{\alpha_i}$

$$\begin{aligned} \text{weight of } \Lambda - \text{weight of } \Lambda' &\leq -(m+1)(\varepsilon) + 2w^* \\ &\leq -\frac{k}{n}\varepsilon + 2w^* \\ &< 0. \end{aligned} \tag{10}$$

This completes the proof of Lemma 3. □

IV. Complexity

In this section, we will analyze the complexity of the min-sum algorithm described in Section II-B. Theorem 1 suggests that the number of iterations required to find the MWM is $O\left(\frac{nw^*}{\varepsilon}\right)$. Now, in each iteration of the min-sum algorithm each node sends a vector of size n (i.e., n numbers) to each of the n nodes in the other partition. Thus, the total number of messages exchanged in each iteration are $O(n^2)$ with each message of length n. Now, each node performs $O(n)$ basic computational operations (comparison, addition) to compute each element in a message vector of size n. That is, each node performs $O(n^2)$ operations to compute a message vector in each iteration. Since each node sends n message vectors, the total cost is $O(n^3)$ per node or $O(n^4)$ per iteration for all nodes. Thus, the total cost for $O(nw^*/\varepsilon)$ iterations is $O(n^5 w^*/\varepsilon)$.

Thus, for fixed w^* and ε, the running time of the algorithm scales as $O(n^5)$. Standard algorithms such as the Edmond–Karp algorithm [9] or the auction algorithm [7] have a complexity of $O(n^3)$. In what follows, we simplify the min-sum algorithm so that the overall running time of the algorithm becomes $O(n^3)$ for fixed w^* and ε. We make a note here that the Edmond–Karp algorithm is strongly polynomial (i.e., does not depend on w^* and ε) while the auction algorithm's complexity is $O(n^3 w^*/\varepsilon)$.

A. Simplified Min-Sum Algorithm for $K_{n,n}$

We first present the algorithm and show that it is exactly the same as the min-sum algorithm. Later, we analyze the complexity of the algorithm.

Simplified min-sum algorithm.

(1) Unlike min-sum algorithm, now each α_i sends a number to β_j and *vice versa*. Let the message from α_i to β_j in iteration k be denoted as

$$\hat{m}^k_{\alpha_i\to\beta_j}.$$

Similarly, let the messages from β_j to α_i in iteration k be denoted as

$$\hat{m}^k_{\beta_j\to\alpha_i}$$

(2) Initially $k=0$ and set the messages as follows:

$$\hat{m}^0_{\alpha_i\to\beta_j}=w_{ij}.$$

Similarly

$$\hat{m}^0_{\beta_j\to\alpha_i}=w_{ij}$$

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

$$\begin{aligned}\hat{m}^k_{\alpha_i\to\beta_j}&=w_{ij}-\max_{\ell\neq j}\hat{m}^{k-1}_{\beta_\ell\to\alpha_i}\\ \hat{m}^k_{\beta_j\to\alpha_i}&=w_{ij}-\max_{\ell\neq i}\hat{m}^{k-1}_{\alpha_\ell\to\beta_j}.\end{aligned}\tag{11}$$

(4) The estimated MWM at the end of iteration k is π^k, where $\pi^k(i)=\arg\max_{1\leq j\leq n}\{\hat{m}^k_{\beta_j\to\alpha_i}\}$, for $1\leq i\leq n$.

(5) Repeat (3)–(4) till π^k converges

Now, we state and prove the claim that relates the above modified algorithm to the original min-sum algorithm.

Lemma 4: In min-sum algorithm adding an equal amount to all coordinates of any message vector $M^k_{\alpha_i\to\beta_j}$ (similarly $M^k_{\beta_j\to\alpha_i}$) at any time does not change the resulting estimated matching π^m for all k, m.

Proof: If a number is added to all coordinates of $M^k_{\alpha_i\to\beta_j}$ it is not hard to see from (4) and structure of $\psi_{\alpha_i\beta_j}(\cdot,\cdot)$ that other message and belief vectors will change only up to an additive constant to their coordinates. Hence, these changes do not affect $\pi^m(i)=\arg\max_{1\leq j\leq n}\{b^m_{\alpha_i}(j)\}$, for $1\leq i\leq n$. □

Lemma 5: The algorithms min-sum and simplified min-sum produce identical estimated matchings π^m at the end of every iteration m.

Proof: Consider the min-sum algorithm. In particular, consider a message vector $M^k_{\alpha_i\to\beta_j}$ in iteration k. First, we claim that all for any given $k\geq 0$, $m^k_{\alpha_i\to\beta_j}(r)$, $r\neq i$ are the same. That is, for $r_1\neq r_2$ and $r_1,r_2\neq i$

$$m^k_{\alpha_i\to\beta_j}(r_1)=m^k_{\alpha_i\to\beta_j}(r_2).$$

For $k=0$, this claim holds by definition. For $k\geq 1$, consider the definition of $m^k_{\alpha_i\to\beta_j}(r)$, $r\neq i$

$$\begin{aligned}m^k_{\alpha_i\to\beta_j}(r)&=\max_{1\leq q\leq n}\psi_{\alpha_i\beta_j}(q,r)\left[w_{iq}+\sum_{\ell\neq j}m^{k-1}_{\beta_\ell\alpha_i}(q)\right]\\&=\max_{q\neq j}\left[w_{iq}+\sum_{\ell\neq j}m^{k-1}_{\beta_\ell\alpha_i}(q)\right].\end{aligned}\tag{12}$$

The first equality follows from definition in min-sum algorithm while second equality follows from property of $\psi_{\alpha_i\beta_j}(\cdot,\cdot)$. Equation (12) is independent of $r(\neq i)$. This proves the desired claim.

The above stated property of min-sum algorithm immediately implies that the vector $M^k_{\alpha_i\to\beta_j}$ has only two distinct values, one corresponding to $m^k_{\alpha_i\to\beta_j}(i)$ and the other corresponding to $m^k_{\alpha_i\to\beta_j}(r)$, $r\neq i$. Now subtract $m^k_{\alpha_i\to\beta_j}(r)$, $r\neq i$ from all coordinates of $M^k_{\alpha_i\to\beta_j}$. Lemma 4 guarantees the resulting matching π^m for all m does not change. Performing the same modification to all message vectors yields a *modified min-sum* algorithm with the same outcome as min-sum. But each message vector $M^k_{\alpha_i\to\beta_j}$ in this modified min-sum has all coordinates equal to zero except the ith coordinate. Denote these ith coordinates by $\tilde{m}^k_{\alpha_i\to\beta_j}$. Now (4) shows these for all i,j,k numbers $\tilde{m}^k_{\alpha_i\to\beta_j}$ satisfy the following recursive equations:

$$\begin{aligned}\tilde{m}^k_{\alpha_i\to\beta_j}&=w_{ij}-\max_{\ell\neq j}(\tilde{m}^{k-1}_{\beta_\ell\to\alpha_i}+w_{i\ell}),\\ \tilde{m}^k_{\beta_j\to\alpha_i}&=w_{ij}-\max_{\ell\neq i}(\tilde{m}^{k-1}_{\alpha_\ell\to\beta_j}+w_{\ell j}).\end{aligned}\tag{13}$$

Similarly, for new beliefs we have

$$\begin{aligned}\tilde{b}^k_{\alpha_i}(r)&=\tilde{m}^k_{\beta_r\to\alpha_i}+w_{ir}\\ \tilde{b}^k_{\beta_j}(s)&=\tilde{m}^k_{\alpha_s\to\beta_j}+w_{sj}.\end{aligned}\tag{14}$$

Now by adding w_{ij} to each side of (13) and dividing them by 2 it can be seen from (11) that numbers $\frac{\tilde{m}^k_{\alpha_i\to\beta_j}+w_{ij}}{2}$ and $\hat{m}^k_{\alpha_i\to\beta_j}$ satisfy the same recursive equations. They also satisfy the same initial conditions. As a result for all i,j,k we have

$$\hat{m}^k_{\alpha_i\to\beta_j}=\frac{\tilde{m}^k_{\alpha_i\to\beta_j}+w_{ij}}{2}=\tilde{b}_{\alpha_i}(j)\tag{15}$$

and

$$\hat{m}^k_{\beta_j\to\alpha_i}=\frac{\tilde{m}^k_{\beta_j\to\alpha_i}+w_{ij}}{2}=\tilde{b}_{\beta_j}(i).\tag{16}$$

This shows that the estimated matching computed at nodes in modified min-sum and simplified min-sum algorithms are exactly the same at each iteration which completes the proof of Lemma 5. □

Note 3. The simplified min-sum equations can also be derived in a direct way by looking at the interpretation of the messages

$\left\{\hat{m}^k_{\alpha_i\to\beta_j}\right\}_{i,j,k}$ in the computation tree. More specifically, consider the level-$(k+1)$ computation tree rooted at α_i, $T^{k+1}_{\alpha_i}$. Also consider its subtree $T^k_{\alpha_i,\beta_j}$, built by adding the edge (α_i,β_j) at the root of $T^{k+1}_{\alpha_i}$ to graph of all descendants of β_j. One can show that the message $\hat{m}^k_{\beta_j\to\alpha_i}$ is equal to the difference between weight of maximum weight T-matching in $T^k_{\alpha_i,\beta_j}$ that uses the edge (α_i,β_j) at the root and weight of the maximum weight T-matching in $T^k_{\alpha_i,\beta_j}$ that does not use that edge. Now a simple induction gives us the update (11).

B. Complexity of Simplified Min-Sum

Lemma 5 and Theorem 1 immediately imply that the simplified min-sum, like min-sum, converges after $O\left(\frac{nw^*}{\varepsilon}\right)$ iterations. As described above, the simplified min-sum algorithm requires a total of $O(n^2)$ messages per iteration. Thus, for fixed w^* and ε, the algorithm requires a total of $O(n^3)$ messages to be exchanged.

Now, we consider the number of computational operations done by each node in an iteration. From the description of simplified min-sum algorithm, it may seem that each node will require to do $O(n)$ work for sending each message and thus $O(n^2)$ work overall at one node. But, we present a simple method that shows each node can compute message for all of its n neighbors with $O(n)$ computational operation (comparison, addition/subtraction). This will result in $O(n^2)$ overall computation per iteration. Thus, it will take $O\left(\frac{n^3w^*}{\varepsilon}\right)$ computation in $O\left(\frac{nw^*}{\varepsilon}\right)$ iterations. This will result in total complexity of $O\left(\frac{n^3w^*}{\varepsilon}\right)$ in terms of overall messages as well as computation operations.

Here we describe an algorithm to compute messages $\hat{m}^k_{\alpha_1\to\beta_j}$, $1\le j\le n$ using received messages $\hat{m}^{k-1}_{\beta_j\to\alpha_1}$, $1\le j\le n$. This is the same algorithm that all α_i, $1\le i\le n$, and β_j, $1\le j\le n$, need to employ. Now, define

$$\begin{aligned} i_1 &= \operatorname{argmax}_{1\le j\le n}\hat{m}^{k-1}_{\beta_j\to\alpha_1}\\ i_2 &= \operatorname{argmax}_{1\le j\le n, j\ne i_1}\hat{m}^{k-1}_{\beta_j\to\alpha_1}\\ \mathrm{Mx}_1 &= \hat{m}^{k-1}_{\beta_{i_1}\to\alpha_1}\\ \mathrm{Mx}_2 &= \hat{m}^{k-1}_{\beta_{i_2}\to\alpha_1}. \end{aligned}$$

Then, from (11) we obtain

$$\begin{aligned} \hat{m}^k_{\alpha_1\to\beta_{i_1}} &= w_{1i_1}-\mathrm{Mx}_2\\ \hat{m}^k_{\alpha_1\to\beta_j} &= w_{1j}-\mathrm{Mx}_1,\quad \text{for } j\ne i_1. \end{aligned} \tag{17}$$

We see that computing all messages $\hat{m}^k_{\alpha_1\to\beta_j}$ takes $O(n)$ operations. From (17), it takes node α_1 $O(n)$ computations to find i_1, i_2, Mx_1, Mx_2, then it takes $O(1)$ computation to compute each of the $\hat{m}^k_{\alpha_1\to\beta_j}$, $1\le j\le n$. That is, it takes $O(n)$ operations for computing all messages $\hat{m}^k_{\alpha_1\to\beta_j}$, $1\le j\le n$.

Thus, we have established that each node α_i, $1\le i\le n$, and β_j, $1\le j\le n$, need to perform $O(n)$ computations to compute all of its messages in a given iteration. That is, the total computation cost per iteration is $O(n^2)$. In summary, Theorem 1, Lemma 5, and discussion of this Section IV-B immediately yield the following result.

Theorem 2: The simplified min-sum algorithm finds the MWM in $O\left(\frac{nw^*}{\varepsilon}\right)$ iterations with total computation cost of $O\left(\frac{n^3w^*}{\varepsilon}\right)$ and $O\left(\frac{n^3w^*}{\varepsilon}\right)$ total number of message exchanges.

V. Auction and Min-Sum Algorithms

In this section, we will first recall the auction algorithm [7] and then describe its relation to the min-sum algorithm.

A. Auction Algorithm for MWM

The auction algorithm finds the MWM via an "auction": all α_i become buyers and all β_j become objects. Let p_j denote the price of β_j and w_{ij} be the value of object β_j for buyer α_i. The net benefit of an assignment or matching π is defined as

$$\sum_{i=1}^{n}\left(w_{i\pi(i)}-p_{\pi(i)}\right).$$

The goal is to find π^* that maximizes this net benefit. It is clear that for any set of prices $p_1,\ldots,p_n$, the MWM maximizes the net benefit. The auction algorithm is an iterative method for finding the optimal prices and an assignment that maximizes the net benefit (and is therefore the MWM).

Auction algorithm.

- Initialize the assignment $S=\emptyset$, the set of unassigned buyers $I=\{\alpha_1,\ldots,\alpha_n\}$, and prices $p_j=0$ for all j.
- The algorithm runs in two phases, which are repeated until S is a complete matching.
- Phase 1: Bidding.
  For all $\alpha_i\in I$
  (1) Find benefit maximizing β_j. Let

$$j_i=\operatorname{argmax}_j\{w_{ij}-p_j\},\quad v_i=\max_j\{w_{ij}-p_j\}$$
$$\text{and}\quad u_i=\max_{j\ne j_i}\{w_{ij}-p_j\}. \tag{18}$$

  (2) Compute the "bid" of buyer α_i, denoted by $b_{\alpha_i\to\beta_{j_i}}$ as follows: given a fixed positive constant δ,

$$b_{\alpha_i\to\beta_{j_i}}=w_{ij_i}-u_i+\delta.$$

- Phase 2: Assignment.
  For each object β_j,

  (3) Let $P(j)$ be the set of buyers from which β_j received a bid. If $P(j) \neq \emptyset$, increase p_j to the highest bid

  $$p_j = \max_{\alpha_i \in P(j)} b_{\alpha_i \to \beta_j}.$$

  (4) Remove the maximum bidder α_{i_j} from I and add (α_{i_j}, β_j) to S. If $(\alpha_k, \beta_j) \in S$, $k \neq i_j$, then put α_k back in I.

Theorem 3 [6]: If $0 < \delta < \varepsilon/n$, then the assignment S converges to the MWM in $O(nw^*/\varepsilon)$ iterations with running time $O(n^3 w^*/\varepsilon)$ (where ε and w^* are as defined earlier).

B. Connecting Min-Sum and Auction

The similarity between (17) and (18) suggests a connection between the min-sum and auction algorithms. In the auction algorithm, the equations for calculating the bids are exactly the same as those for updating messages in the simplified min-sum algorithm. But when updating the prices, the maximum is taken over all incoming bids which is different from the dynamics of the simplified min-sum equations. Moreover, in the auction algorithm, bidders do not bid at every iteration and do not bid to every object but in the simplified min-sum algorithm each vertex sends a message to *all* of its neighbors at *every* iteration. Based on these similarities and the difference we made modifications to both the simplified min-sum and auction algorithms which we called min-sum auction I and min-sum auction II, respectively. We will show that these versions are equivalent and derive some of their key properties. Here we consider the naïve auction algorithm (when $\delta = 0$) and deal with the case $\delta > 0$ in Section V-B-I.

Min-sum auction I.

(1) Each α_i sends a number to β_j and *vice versa*. Let the messages in iteration k be denoted as $\tilde{m}^k_{\alpha_i \to \beta_j}$, $\tilde{m}^k_{\beta_j \to \alpha_i} \in \mathbb{R}$.

(2) Initialize $k = 0$ and set $\tilde{m}^0_{\beta_j \to \alpha_i} = 0$.

(3) For $k \geq 1$, update messages as follows:

$$\tilde{m}^k_{\alpha_i \to \beta_j} = w_{ij} - \max_{\ell \neq j} \left\{ w_{i\ell} - \tilde{m}^{k-1}_{\beta_\ell \to \alpha_i} \right\}$$
$$\tilde{m}^k_{\beta_j \to \alpha_i} = \max_{\ell=1}^{n} \tilde{m}^k_{\alpha_\ell \to \beta_j} \tag{19}$$

(4) The estimated MWM at the end of iteration k is the set of edges $\pi^k = \{(\alpha_{i_j}, \beta_j)\}$ where

$$i_j = \arg \max_{1 \leq \ell \leq n} \left(\tilde{m}^k_{\alpha_\ell \to \beta_j} \right)$$

and

$$\tilde{m}^k_{\alpha_{i_j} \to \beta_j} \geq \tilde{m}^{k-1}_{\beta_j \to \alpha_i}.$$

(5) Repeat (3)–(4) till π^k is a complete matching.

Min-sum auction II.

- Initialize the assignment $S = \emptyset$ and prices $p_j = 0$ for all j.
- The algorithm runs in two phases, which are repeated until S is a complete matching.
- Phase 1: Bidding.
  For all α_i,

  (1) Find β_j that maximizes the benefit. Let

  $$j_i = \text{argmax}_j \{w_{ij} - p_j\}, \quad v_i = \max_j \{w_{ij} - p_j\}$$
  $$\text{and } u_i = \max_{j \neq j_i} \{w_{ij} - p_j\}. \tag{20}$$

  (2) Compute the "bid" of buyer α_i, denoted by $b_{\alpha_i \to \beta_j}$

  $$b_{\alpha_i \to \beta_{j_i}} = w_{ij_i} - u_i$$
  $$\text{and} \quad b_{\alpha_i \to \beta_j} = w_{ij} - v_i, \; j \neq j_i.$$

- Phase 2: Assignment.
  For each object β_j

  (3) Set price p_j to the highest bid, $p_j = \max_{\alpha_i} b_{\alpha_i \to \beta_j}$.

  (4) Reset $S = \emptyset$. Then, for each j add the pair (α_{i_j}, β_j) to S if $b_{\alpha_{j_i} \to \beta_j} \geq p_j$, where α_{i_j} is a buyer attaining the maximum in step (3)

Theorem 4: The algorithms min-sum auction I and II are equivalent.

Proof: Let $b^k_{\alpha_i \to \beta_j}$ and p^k_j denote the bids and prices at the end of iteration k in algorithm min-sum auction II. Now, identify $b^k_{\alpha_i \to \beta_j}$ with $\tilde{m}^k_{\alpha_i \to \beta_j}$ and p^k_j with $\tilde{m}^k_{\beta_j \to \alpha_i}$. Then it is immediate that min-sum auction II becomes identical to min-sum auction I. This completes the proof of Theorem 4. □

Next we will prove that if the min-sum auction algorithm terminates (we omit reference to I or II), it finds the correct MWM. As we will see, the proof uses standard arguments (see [7] for example).

Theorem 5: Let σ be the termination matching of the min-sum auction I (or II). Then it is the MWM, i.e., $\sigma = \pi^*$.

Proof: The proof follows by establishing that at termination, the messages of min-sum auction form the optimal solution for the dual of the MWM problem and σ is the corresponding optimal solution to the primal, i.e., MWM. To do so, we first state the dual of the MWM problem

$$\min \quad \sum_{i=1}^{n} r_i + \sum_{j=1}^{n} p_j$$
$$\text{subject to} \quad r_i + p_j \geq w_{ij}. \tag{21}$$

Let (r^*, p^*) be the optimal solution to the above stated dual problem and let π^* solve the primal MWM problem. Then, the standard complimentary slackness conditions are

$$r^*_i + p^*_{\pi^*(i)} = w_{i\pi^*(i)}. \tag{22}$$

Thus, (r^*, p^*, π^*) are the optimal dual-primal solution for the MWM problem if and only if a) π^* is a matching, b) (r^*, p^*) satisfy (21), and c) the triple satisfies (22). To complete the proof, we will prove the existence of r^*, p^* such that (r^*, p^*, σ) satisfy conditions a)–c).

To this end, first note that σ is a matching by the termination condition of the algorithm; thus, condition a) is satisfied. We shall consider the min-sum auction II algorithm for the purpose of the proof. Suppose the algorithm terminates at some iteration k. Let p_j^{k-1} and p_j^k be the prices of β_j in iterations $k-1$ and k, respectively. Since all β_j's are matched at the termination, from step (4) of the min-sum auction II, we obtain

$$p_j^k \geq p_j^{k-1}, \quad \forall\, j. \tag{23}$$

At termination (iteration k), α_i is matched with $\beta_{\sigma(i)}$ or β_j is matched with $\alpha_{\sigma^{-1}(j)}$. By the definition of the min-sum auction II algorithm

$$p_j^k = w_{\sigma^{-1}(j)j} - \max_{\ell \neq j} \left[w_{\sigma^{-1}(j)\ell} - p_\ell^{k-1} \right]. \tag{24}$$

From (23) and (24), we obtain that

$$w_{\sigma^{-1}(j)j} - p_j^k \geq \max_{\ell \neq j} \left[w_{\sigma^{-1}(j)\ell} - p_\ell^k \right]. \tag{25}$$

Define $r_i^* = w_{i\sigma(i)} - p_{\sigma(i)}^k$ and $p_j^* = p_j^k$. Then, from (25), (r^*, p^*) satisfy the dual feasibility, that is, (21). Further, by definition they satisfy the complimentary slackness condition (22). Thus, the triple (r^*, p^*, σ) satisfies conditions a)–c) as required. Hence, the algorithm min-sum auction II produces the MWM, i.e., $\sigma = \pi^*$. □

The min-sum auction II algorithm looks very similar to the auction algorithm and inherits some of its properties. However, it also inherits some properties of the min-sum algorithm. This causes it to behave differently from the auction algorithm. The proof of convergence of the auction algorithm relies on two properties of the auctioning mechanism: a) the prices are always nondecreasing and b) the number of matched objects is always nondecreasing. By design, a) and b) can be shown to hold for the auction algorithm. However, it is not clear if a) and b) are true for min-sum auction. In what follows, we state the result that prices are eventually nondecreasing in the min-sum auction algorithm; however, it seems difficult to establish a statement similar to b) for the min-sum algorithm as of now.

Theorem 6: If π^* is unique then in the min-sum auction II algorithm prices eventually increase. That is, $\forall k \in \mathbb{Z}^+; \exists\, T > k$ s.t. $\forall t \geq T; p_j^t > p_j^k, 1 \leq j \leq n$.

Proof: Proof of Theorem 6 is essentially based on i) the equivalence between the min-sum auction algorithms I and II, and ii) arguments very similar to the ones used in the proof of Lemma 2, where we relate prices with the computation tree. □

Our simulations suggests that in the absence of the condition "$\tilde{m}^k_{\alpha_{i_j} \to \beta_j} \geq \tilde{m}^{k-1}_{\beta_j \to \alpha_i}$" from step (4) of min-sum auction I, the algorithm always terminates and finds the MWM as long as it is unique. This along with Theorem 6 leads us to the following conjecture.

Conjecture 1: If π^* is unique then the min-sum auction I terminates in a finite number of iterations if condition "$\tilde{m}^k_{\alpha_{i_j} \to \beta_j} \geq \tilde{m}^{k-1}_{\beta_j \to \alpha_i}$" is removed from step (4).

C. Relation to δ-Relaxation

In the previous section, we established a relation between the min-sum and auction (with $\delta = 0$) algorithms. In [7], [6] the author extends the auction algorithm to obtain guaranteed convergence in a finite number of iterations via a δ-relaxation for some $\delta > 0$. At termination, the δ-relaxed algorithm produces a triple (r^*, p^*, π^*) such that (a1) π^* is a matching, (b1) (r^*, p^*) satisfy (21), and (c1) the following modified complimentary slackness conditions are satisfied:

$$r_i^* + p_{\pi^*(i)}^* \leq w_{i\pi^*(i)} + \delta. \tag{26}$$

The conditions (c1) are referred to as δ-CS conditions in [7]. This modification is reflected in the description of the auction algorithm where we have added δ to each bid in step (2). We established the relation between min-sum and auction for $\delta = 0$ in the previous section. Here we make a note that for every $\delta > 0$, a similar relation holds. To see this, we consider min-sum auction I and II where the bid computation is modified as follows: modify step (3) of min-sum auction I as

$$\tilde{m}^k_{\alpha_i \to \beta_j} = w_{ij} - \max_{\ell \neq j} \left\{ w_{i\ell} - \tilde{m}^{k-1}_{\beta_\ell \to \alpha_i} \right\} + \delta$$

and modify step (2) of min-sum auction II as $b_{\alpha_i \to \beta_{j_i}} = w_{ij_i} - u_i + \delta$ and $b_{\alpha_i \to \beta_j} = w_{ij} - v_i + \delta$, $j \neq j_i$. For these modified algorithms, we obtain the following result using arguments very similar to the ones used in Theorem 5.

Theorem 7: For $\delta > 0$, let σ be the matching obtained from the modified min-sum auction algorithm I (or II). Then, $w_\sigma \geq w_{\pi^*} - n\delta$ (i.e., σ is within $n\delta$ of the MWM).

D. Implications

The relation between min-sum and auction algorithms resulted in equivalent algorithms min-sum auction I and II. The further modification of the min-sum auction I (or II) based on the δ-relaxation method allows for designing (deterministic) distributed algorithm that works even in the presence of nonunique MWM (Theorem 7). This suggests a method for designing modification of min-sum or max-product for general optimization problem so as to work in the presence of a nonunique solution. Further, the min-sum auction I algorithm by design is dual unlike the auction being primal-dual. This may be of interest in optimization methods on its own.

VI. Discussion and Conclusion

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

Using the regularity of the structure of the problem, we managed to simplify the max-product algorithm. 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 ε, 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.

The relation that we established between the auction algorithm and the min-sum algorithm is tantalizing. It suggests a method to design modification of max-product algorithm for general optimization problem that may work even in the presence of nonunique solutions.

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.[2] 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.

ACKNOWLEDGMENT

The authors would like to thank the anonymous referees for their helpful comments.

REFERENCES

[1] S. M. Aji, G. B. Horn, and R. J. McEliece, "On the convergence of iterative decoding on graphs with a single cycle," in *Proc. IEEE Int. Symp. Information Theory*, Cambridge, MA, Aug. 1998, p. 276.

[2] S. M. Aji and R. J. McEliece, "The generalized distributive law," *IEEE Trans. Inf. Theory*, vol. 46, no. 2, pp. 325–343, Mar. 2000.

[3] M. Bayati, D. Shah, and M. Sharma, "Maximum weight matching via max-product belief propagation," in *Proc. IEEE Int. Symp. Information Theory*, Adelaide, Australia, Sep. 2005, pp. 1763–1767.

[4] M. Bayati, D. Shah, and M. Sharma, "A simpler max-product maximum weight matching algorithm and the auction algorithm," in *Proc. IEEE Int. Symp. Information Theory*, Seattle, WA, Jul. 2006, pp. 557–561.

[2]A key fact in the proof of Lemma 3 was the property that bipartite graphs do not have odd cycles.

[5] M. Bayati, B. Prabhakar, D. Shah, and M. Sharma, "Iterative scheduling algorithm," in *Proc. IEEE Infocom*, Anchorage, AK, May 2007, pp. 445–453.

[6] D. P. Bertsekas, "Auction algorithms for network flow problems: A tutorial introduction," *Comput. Optimiz. Applic.*, vol. 1, pp. 7–66, 1992.

[7] D. Bertsekas and J. Tsitsiklis, *Parallel and Distributed Computation: Numerical Methods*. Englewood Cliffs, NJ: Prentice-Hall, 1989.

[8] R. G. Gallager, *Low Density Parity Check Codes*. Cambridge, MA: MIT Press, 1963.

[9] J. Edmonds and R. Karp, "Theoretical improvements in algorithmic efficiency for network flow problems," *J. ACM*, vol. 19, pp. 248–264, 1972.

[10] B. J. Frey and R. Koetter, "Exact inference using the attenuated max-product algorithm," in *Advanced Mean Field Methods: Theory and Practice*, M. Opper and D. Saad, Eds. Cambridge, MA: MIT Press, 2000.

[11] G. B. Horn, "Iterative Decoding and Pseudocodewords," Ph.D. dissertation, Dep. Elec. Eng., California Insst. Technol., Pasadena, CA, 1999.

[12] S. Lauritzen, *Graphical models*. Oxford, U.K.: Oxford Univ. Press, 1996.

[13] E. Lawler, *Combinatorial Optimization: Networks and Matroids*. New York: Holt, Rinehart and Winston, 1976.

[14] N. McKeown, V. Anantharam, and J. Walrand, "Achieving 100% throughput in an input-queued switch," in *Proc. IEEE Inforcom*, San Francisco, CA, Mar. 1996, vol. 1, pp. 296–302.

[15] M. Mezard, G. Parisi, and R. Zecchina, "Analytic and algorithmic solution of random satisfiability problems," *Science*, vol. 297, p. 812, 2002.

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

[17] T. Richardson and R. Urbanke, "The capacity of low-density parity check codes under message-passing decoding," *IEEE Trans. Inf. Theory*, vol. 47, no. 2, pp. 599–618, Feb. 2001.

[18] M. Wainwright and M. Jordan, Graphical Models, Exponential Families, and Variational Inference Dep. Statist., Univ. California, Berkeley, CA, 2003.

[19] M. J. Wainwright, T. S. Jaakkola, and A. S. Willsky, "Tree consistency and bounds on the performance of the max-product algorithm and its generalizations," *Statistics and Computing*, vol. 14, pp. 143–166, Apr. 2004.

[20] Y. Weiss, "Correctness of local probability propagation in graphical models with loops," *Neural Comput.*, vol. 12, pp. 1–42, 2000.

[21] Y. Weiss, Belief Propagation and Revision in Networks With Loops MIT AI Lab., 1997, Tech. Rep. 1616.

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

[23] Y. Weiss and W. T. Freeman, "On the optimality of solutions of the max-product belief-propagation algorithm in arbitrary graphs," *IEEE Trans. Inf. Theory*, vol. 47, no. 2, pp. 736–744, Feb. 2001.

[24] J. Yedidia, W. Freeman, and Y. Weiss, Understanding Belief Propagation and Its Generalizations Mitsubishi Elect. Res. Lab., 2000, TR-2001-22.

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