

# Convergent Tree-reweighted Message Passing for Energy Minimization

Vladimir Kolmogorov

vnk@microsoft.com

June 2005

Technical Report

MSR-TR-2005-38

## Abstract

Algorithms for discrete energy minimization have a fundamental importance in computer vision. In this paper we focus on the recent technique proposed by Wainwright et al. [18] - tree-reweighted max-product message passing (TRW). It was inspired by the problem of maximizing a lower bound on the energy. However, the algorithm is not guaranteed to increase this bound - it may actually go down. In addition, TRW does not always converge. We develop a modification of this algorithm which we call *sequential tree-reweighted message passing*. Its main property is that the bound is guaranteed not to decrease. We also give a *weak tree agreement* condition which characterizes local maxima of the bound with respect to TRW algorithms. We prove that our algorithm has a limit point that achieves weak tree agreement. Finally, we show that our algorithm requires half as much memory as traditional message passing approaches. Experimental results demonstrate that on certain synthetic and real problems our algorithm outperforms both the ordinary belief propagation and tree-reweighted algorithm in [18]. In addition, on stereo problem with Potts interactions we obtain lower energy than graph cuts.

**Index terms:** Energy minimization, graph algorithms, message passing, belief propagation, early vision, Markov Random Fields, stereo.

Microsoft Research  
Microsoft Corporation  
One Microsoft Way  
Redmond, WA 98052  
<http://www.research.microsoft.com>

# 1 Introduction

Many early vision problems can be naturally formulated in terms of energy minimization where the energy function has the following form:

$$E(\mathbf{x} | \theta) = \theta_{const} + \sum_{s \in \mathcal{V}} \theta_s(x_s) + \sum_{(s,t) \in \mathcal{E}} \theta_{st}(x_s, x_t) \quad (1)$$

Set  $\mathcal{V}$  usually corresponds to pixels;  $x_s$  denotes the label of pixel  $s \in \mathcal{V}$  which must belong to some finite set. For motion or stereo, the labels are disparities, while for image restoration they represent intensities.  $\theta$  defines parameters of the energy:  $\theta_s(\cdot)$  is a unary data penalty function, and  $\theta_{st}(\cdot, \cdot)$  is a pairwise interaction potential. This energy is often derived in the context of Markov Random Fields [5]: a minimum of  $E$  corresponds to a *maximum a-posteriori* (MAP) labeling  $\mathbf{x}$ .

In general, minimizing  $E$  is an NP-hard problem, so researchers have focused on approximate minimization algorithms. The two most well-known techniques are graph cuts and belief propagation. The former one was introduced in the 90's [2, 6, 7, 11] and showed a major improvement over previously used simulated annealing [5]. To our knowledge, graph cuts are currently considered to be the most accurate minimization algorithm for energy functions arising in many vision applications, e.g. stereo [2, 10], image restoration [2], image segmentation [1], texture synthesis [12]. In fact, for some functions it finds a global minimum.

However, graph cuts can be applied only to a limited class of energy functions [2, 7, 11]. If a function falls outside this class then one has to use other techniques such as max-product belief propagation (BP) [4, 13, 16, 22]. BP can be applied to any function of the form as in eq. 1, but it has some drawbacks. First, it usually finds a solution with higher energy than graph cuts (in cases when graph cuts can be applied) [17]. Second, BP does not always converge - it often goes into a loop.

**Tree-reweighted message passing** Recently, Wainwright et al. [18] introduced a new algorithm for energy minimization called *max-product tree-reweighted message passing* (TRW). We believe that it will become a serious rival to both graph cuts and BP. As BP, TRW can be applied to any function of the form as in eq. 1. In addition, in this paper we show that for stereo problem with Potts interactions TRW obtains lower energy than graph cuts and BP. The improvement over graph cuts is marginal and probably would not be a factor for the stereo problem - both algorithms find solutions whose energy is very close to the global minimum. However, it shows the potential of TRW; the difference may become significant for more difficult functions.

Wainwright et al. [18] give two versions of TRW algorithm which differ in the schedule of passing messages. They are inspired by the problem of maximizing a concave lower bound on the energy. These algorithms have the following property: if their fixed point satisfies a certain condition ("tree agreement") then it is guaranteed to give a MAP solution (i.e. a global minimum of the energy).

However, TRW algorithms in [18] cannot be viewed as algorithms for direct maximization of the bound. Indeed, in our experiments we observed that sometimes they decrease it. Also, the algorithms do not always converge; when it happens, the value of the bound often goes into a loop.

Our main contribution is as follows: we show how to modify TRW algorithms so that the value of the bound is guaranteed not to decrease. Thus, we are guaranteed to find at least a "local" maximum of the bound. The word "local" is in quotes since for concave functions all local maxima are global, if the standard metric space topology is used. Here we use a weaker topology: our maxima are local with respect to the TRW algorithms. We formulate the *weak*

*tree agreement* condition (WTA) which gives a precise characterization of such maxima. We prove that our algorithm has a subsequence converging to a vector satisfying WTA.

An interesting question is whether WTA always gives a global maximum of the bound. We show that this is not the case by providing a counterexample. Note that this is different from *sum-product* tree-reweighted message passing [20]: in the latter case a fixed point of TRW is guaranteed to be the global maximum of the lower bound on the negative log partition function.

TRW algorithms require some choice of trees covering the graph. If the trees have a special structure (namely, chains which are *monotonic* with respect to some ordering on the graph) then our algorithm reduces to TRW message-passing algorithm of Wainwright et al. [18], but with a significant distinction: we update messages in a specific *sequential* order rather than in parallel. In the context of ordinary BP it was observed experimentally that sequential updates are superior to parallel updates [17] although convergence is still not guaranteed. We give a theoretical justification of sequential updates in the case of tree-reweighted algorithms.

Our sequential schedule has an additional advantage: we show that it can be implemented using half as much space as traditional message passing approaches. Similar observation was made in [4] for bipartite graphs and parallel schedule of updating messages. Our technique can be applied to any graph and is a strict generalization of that in [4].

**Outline** The paper is organized as follows. In section 2 we introduce our notation and review some results from [18], in particular the lower bound on the energy function via convex combination of trees and duality result. Our new tree-reweighted algorithm and its analysis are given in section 3. Experimental results are described in section 4. Finally, we give conclusions in section 5.

## 2 Notation and background

In this paper we closely follow the notation used in [18]. However, instead of maximizing posterior probability we minimize an energy function. Therefore, we replace “min” with “max”, “inf” with “sup” and vice versa.

Let  $\mathcal{G} = (\mathcal{V}, \mathcal{E})$  be an undirected graph with the set of vertices  $\mathcal{V}$  and the set of edges  $\mathcal{E}$ . For each  $s \in \mathcal{V}$ , let  $x_s$  be a variable taking values in some discrete space  $\mathcal{X}_s$ . By concatenating the variables at each node, we obtain a vector  $\mathbf{x}$  with  $n = |\mathcal{V}|$  elements. This vector takes values in the space  $\mathcal{X} = \mathcal{X}_1 \times \mathcal{X}_2 \times \dots \times \mathcal{X}_n$ . Unless noted otherwise, symbols  $s$  and  $t$  will denote nodes in  $\mathcal{V}$ ,  $(s, t)$  - edge in  $\mathcal{E}$ ,  $j$  and  $k$  - variables in  $\mathcal{X}_s$  and  $\mathcal{X}_t$ , respectively.

As stated in the previous section, our goal is minimizing function  $E(\mathbf{x} | \theta)$  (eq. 1) defined by parameter  $\theta$ . This parameter is specified by constant term  $\theta_{const}$ , unary terms  $\theta_s(j)$  and pairwise terms  $\theta_{st}(j, k)$ . It is convenient to denote the last two terms as  $\theta_{s;j}$  and  $\theta_{st;jk}$ , respectively. Then  $\theta$  can be viewed as a vector  $\theta = \{\theta_\alpha | \alpha \in \mathcal{I}\} \in \mathbb{R}^d$  where the index set  $\mathcal{I}$  is

$$\mathcal{I} = \{const\} \cup \{(s; j)\} \cup \{(st; jk)\}$$

Note that  $(st; jk) \equiv (ts; kj)$ , so  $\theta_{st;jk}$  and  $\theta_{ts;kj}$  are the same element. We will use notation  $\theta_s$  to denote a vector of size  $|\mathcal{X}_s|$  and  $\theta_{st}$  to denote a vector of size  $|\mathcal{X}_s \times \mathcal{X}_t|$ .

It is convenient to express  $E(\mathbf{x} | \theta)$  as the Euclidean product of two vectors depending on  $\theta$  and  $\mathbf{x}$ . Of course, we cannot write it as  $\langle \theta, \mathbf{x} \rangle$  since  $\theta$  and  $\mathbf{x}$  belong to different spaces. However, we can introduce mapping  $\phi: \mathcal{X} \rightarrow \mathbb{R}^d$  so that

$$E(\mathbf{x} | \theta) = \langle \theta, \phi(\mathbf{x}) \rangle = \sum_{\alpha \in \mathcal{I}} \theta_\alpha \phi_\alpha(\mathbf{x})$$

Mapping  $\phi$  called the *canonical overcomplete representation* [19, 21] consists of the following functions  $\phi_\alpha : \mathcal{X} \rightarrow \mathbb{R}$ :

$$\{\phi_{const}(\mathbf{x})\} \cup \{\delta_j(x_s)\} \cup \{\delta_j(x_s)\delta_k(x_t)\}$$

where  $\delta_j(x_s)$  is an *indicator function* - it is equal to one if  $x_s = j$ , and zero otherwise. Function  $\phi_{const}(\mathbf{x})$  equals to 1 for all arguments.

Let us introduce another notation which we will use extensively throughout the paper. Functions  $\Phi, \Phi_{s;j}, \Phi_{st;jk} : \mathbb{R}^d \rightarrow \mathbb{R}$  give information about the minimum values of the energy under different constraints:

$$\begin{aligned} \Phi(\theta) &= \min_{\mathbf{x} \in \mathcal{X}} E(\mathbf{x} | \theta) \\ \Phi_{s;j}(\theta) &= \min_{\mathbf{x} \in \mathcal{X}, x_s=j} E(\mathbf{x} | \theta) \\ \Phi_{st;jk}(\theta) &= \min_{\mathbf{x} \in \mathcal{X}, x_s=j, x_t=k} E(\mathbf{x} | \theta) \end{aligned}$$

Values  $\Phi_{s;j}(\theta)$  and  $\Phi_{st;jk}(\theta)$  are called *min-marginals* for node  $s$  and edge  $(s, t)$ , respectively.

## 2.1 Max-product belief propagation

The key subroutine of tree-reweighted message passing algorithms is *max-product belief propagation* (BP) of Pearl [13]. BP is an algorithm for approximate minimization of energy  $E(\mathbf{x} | \bar{\theta})$  as in eq. 1; it is exact if the graph is a tree. Let us review how BP works. It maintains a *message*  $M_{st} = \{M_{st;k} | k \in \mathcal{X}_t\}$  for each directed edge  $(s \rightarrow t) \in \mathcal{E}$ , which is a vector with  $|\mathcal{X}_t|$  components. We denote  $M = \{M_{st}\}$  to be the vector of all messages.

The basic operation of BP is *passing a message* from node  $s$  to node  $t$  for directed edge  $(s \rightarrow t) \in \mathcal{E}$ . It consists of updating vector  $M_{st}$  as follows:

$$M_{st;k} := \min_{j \in \mathcal{X}_s} \{(\bar{\theta}_{s;j} + \sum_{(u \rightarrow s) \in \mathcal{E}, u \neq t} M_{us;j}) + \bar{\theta}_{st;jk}\} + const_t$$

where  $const_t$  is a constant independent of  $k$ <sup>1</sup>. We will say that a message for directed edge  $(s \rightarrow t)$  is *valid* if this update does not change  $M_{st}$  (or change it by a constant independent of  $k$ ). BP algorithm keeps passing messages for edges in some order until convergence, i.e. until all messages are valid.

If the graph contains loops then in general convergence is not guaranteed. In this paper, however, BP is applied only to tree-structured subgraphs. In this case only two passes are needed to achieve convergence: forward (sending messages from leaves to a root) and backward (sending messages from the root to leaves). Note that any node can serve as a root.

## 2.2 Reparameterization

If two parameter vectors  $\theta$  and  $\bar{\theta}$  define the same energy function (i.e.  $E(\mathbf{x} | \theta) = E(\mathbf{x} | \bar{\theta})$  for all  $\mathbf{x} \in \mathcal{X}$ ) then  $\theta$  is called a *reparameterization* of  $\bar{\theta}$  [19, 21]. We will write this as  $\theta \equiv \bar{\theta}$ . Note that this condition does not necessarily imply that  $\theta = \bar{\theta}$  since there are various linear relations among potential functions  $\phi_\alpha$ . Indeed, any message vector  $M = \{M_{st}\}$  defines reparameterization  $\theta = \bar{\theta}[M]$  of the original parameter vector  $\bar{\theta}$  as follows:

$$\begin{aligned} \theta_t &= \bar{\theta}_t + \sum_{(s \rightarrow t) \in \mathcal{E}} M_{st} \\ \theta_{st;jk} &= \bar{\theta}_{st;jk} - M_{st;k} - M_{ts;j} \\ \theta_{const} &= \bar{\theta}_{const} \end{aligned}$$

---

<sup>1</sup>Throughout the paper we use notation  $const_t$  or  $const_{st}$  to denote constants independent of  $k$  or  $j$ . These constants may be different in different equations.

We prove in Appendix B that the opposite is also true: any reparameterization  $\theta$  of vector  $\bar{\theta}$  can be expressed via messages, up to a *constant parameter vector*.

Reparameterization provides an alternative way of implementing BP algorithm. As discussed in section 2.1, standard BP stores original parameter vector  $\bar{\theta}$  and messages  $M$ . Alternatively, we can maintain reparameterization  $\theta = \bar{\theta}[M]$ ; then vectors  $\bar{\theta}$  and  $M$  are not needed. It turns out that sending a message from node  $s$  to node  $t$  is equivalent to reparameterizing vectors  $\theta_t$  and  $\theta_{st}$  for node  $t$  and edge  $(s, t)$ , respectively [21]. It can also be shown [21] that a message for directed edge  $(s \rightarrow t)$  is valid iff

$$\min_{j \in \mathcal{X}_s} \{\theta_{s;j} + \theta_{st;jk}\} = \text{const}_{st} \quad \forall k \in \mathcal{X}_t \quad (2)$$

If this condition holds, then sending a message from  $s$  to  $t$  does not change  $\theta_{st}$  and  $\theta_t$  (or change them by a constant independent of  $j$  or  $k$ ). We say that  $\theta$  is in a *normal form* if all messages are valid.

For a tree-structured graph values  $\theta_{s;j}$  and  $\theta_{st;jk}$  for vector  $\theta$  in a normal form have a particularly simple interpretation [21] - they correspond to min-marginals (up to a constant):

$$\bar{\Phi}_{s;j}(\theta) = \theta_{s;j} + \text{const}_s \quad (3a)$$

$$\bar{\Phi}_{st;jk}(\theta) = \{\theta_{s;j} + \theta_{st;jk} + \theta_{t;k}\} + \text{const}_{st} \quad (3b)$$

**Canonical normal form** We will say that vector  $\theta$  is in a *canonical normal form* if in addition to equations 2 it satisfies the following conditions for all nodes and edges:

$$\begin{aligned} \min_j \theta_{s;j} &= 0 \\ \min_{j,k} \{\theta_{s;j} + \theta_{st;jk} + \theta_{t;k}\} &= 0 \end{aligned}$$

Any vector  $\theta$  in a normal form can be reparameterized into a canonical normal form by subtracting a constant from vectors  $\theta_s$  and  $\theta_{st}$  and adding the same constant to  $\theta_{const}$ . Canonical normal form is quite useful because constants in equations 2 and 3 can be written down explicitly (see Appendix A).

### 2.3 Lower bounds on the energy function

**Linear programming relaxation** In general the problem of minimizing energy of the form 1 is NP-hard. Therefore, researchers have focused on approximation algorithms. One of the approaches [3, 18] uses a linear programming (LP) relaxation technique. Let us define the following constraint set:

$$LOCAL(\mathcal{G}) = \left\{ \tau \in \mathbb{R}_+^d \left| \begin{array}{ll} \tau_{const} = 1 & \\ \sum_{j \in \mathcal{X}_s} \tau_{s;j} = 1 & \forall s \in \mathcal{V} \\ \sum_{j \in \mathcal{X}_s} \tau_{st;jk} = \tau_{t;k} & \forall (s \rightarrow t) \in \mathcal{E}, k \in \mathcal{X}_t \end{array} \right. \right\}$$

It is easy to check that for any configuration  $\mathbf{x} \in \mathcal{X}$  solution vector  $\phi(\mathbf{x})$  belongs to this set. Therefore, the following minimization problem yields a lower bound on  $\Phi(\bar{\theta})$ :

$$\min_{\tau \in LOCAL(\mathcal{G})} \langle \bar{\theta}, \tau \rangle \quad (4)$$

As showed in [3], this relaxation has several interesting properties. Here we mention just one. Suppose that energy function  $E(\cdot | \theta)$  has *Potts* interaction terms and parameter vector  $\theta$  is non-negative. Then by solving this minimization problem and by using randomized technique

proposed in [8] we obtain configuration  $\mathbf{x}$  such that the expected value of  $E(\mathbf{x} | \theta)$  is at most twice the optimal value of the energy.

**Convex combination of trees** Unfortunately, general linear programming algorithms are rather slow, and currently solving problem 4 is not computationally feasible for large problems such as those that occur in computer vision [2]. As an alternative, we could try to solve the Lagrangian dual of this problem. This is the approach that was taken in [18]. In order to describe it, we need to introduce some notation.

Let  $\mathcal{T}$  be a collection of trees in graph  $\mathcal{G}$  and  $\rho^T$ ,  $T \in \mathcal{T}$  be some distribution on  $\mathcal{T}$ . Throughout the paper we assume that each tree has a non-zero probability and each edge in  $\mathcal{E}$  is covered by at least one tree.

For a given tree  $T = (\mathcal{V}^T, \mathcal{E}^T)$  we define a set

$$\mathcal{I}^T = \{const\} \cup \{(s; j) \mid s \in \mathcal{V}^T\} \cup \{(st; jk) \mid (s, t) \in \mathcal{E}^T\}$$

corresponding to those indexes associated with vertices and edges in the tree.

To each tree  $T \in \mathcal{T}$ , we associate an energy parameter  $\theta^T$  that must respect the structure of  $T$ . More precisely, the parameter  $\theta^T$  must belong to the following linear constraint set:

$$\mathcal{A}^T = \{\theta^T \in \mathbb{R}^d \mid \theta_\alpha^T = 0 \quad \forall \alpha \in \mathcal{I} \setminus \mathcal{I}^T\}$$

By concatenating all of the tree vectors, we form a larger vector  $\boldsymbol{\theta} = \{\theta^T \mid T \in \mathcal{T}\}$ , which is an element of  $\mathbb{R}^{d \times |\mathcal{T}|}$ . Vector  $\boldsymbol{\theta}$  must belong to the constraint set

$$\mathcal{A} = \{\boldsymbol{\theta} \in \mathbb{R}^{d \times |\mathcal{T}|} \mid \theta^T \in \mathcal{A}^T \text{ for all } T \in \mathcal{T}\}$$

Consider function  $\Phi_\rho : \mathcal{A} \rightarrow \mathbb{R}$  defined as follows:

$$\Phi_\rho(\boldsymbol{\theta}) = \sum_T \rho^T \Phi(\theta^T) = \sum_T \rho^T \min_{\mathbf{x} \in \mathcal{X}} \langle \theta^T, \phi(\mathbf{x}) \rangle$$

[18] shows that if  $\sum_T \rho^T \theta^T = \bar{\theta}$  then  $\Phi_\rho(\boldsymbol{\theta})$  is a lower bound on the optimal value of the energy for vector  $\bar{\theta}$  (this follows from Jensen's inequality). To get the tightest bound we can consider the following maximization problem:

$$\max_{\boldsymbol{\theta} \in \mathcal{A}, \sum_T \rho^T \theta^T = \bar{\theta}} \Phi_\rho(\boldsymbol{\theta}) \tag{5}$$

$\Phi_\rho$  is a concave function of  $\boldsymbol{\theta}$ ; moreover, the constraints of problem 5 are linear in  $\boldsymbol{\theta}$ . The following theorem proved in [18] characterizes the dual to problem 5<sup>2</sup>.

**Theorem 2.1.** *Minimization problem 4 is the Lagrangian dual to maximization problem 5. Strong duality holds, so their optimal values coincide.*

A surprising consequence of this theorem is that the optimal value of problem 5 does not depend on the choice of trees and their probabilities (as long as each edge is covered with non-zero probability).

### 3 New tree-reweighted message passing algorithm

In the previous section we introduced our notation and described previous work. Now we can concentrate on our contributions.

---

<sup>2</sup> [18] formulated this theorem for the case when trees in  $\mathcal{T}$  are *spanning*. However, their proof never uses this assumption. In this paper we do not assume that trees are spanning.

We start by modifying maximization problem 5. This problem inspired two algorithms in [18] - tree-reweighted message passing with edge based updates (TRW-E) and with tree based updates (TRW-T). However, neither algorithm maintains constraint  $\sum_T \rho^T \theta^T = \bar{\theta}$  of problem 5. Indeed, they perform reparameterizations of the original parameter vector, so this equality may become violated<sup>3</sup>. Let us replace it with the constraint  $\sum_T \rho^T \theta^T \equiv \bar{\theta}$ . Thus, we are now interested in the following maximization problem:

$$\max_{\theta \in \mathcal{A}, \sum_T \rho^T \theta^T \equiv \bar{\theta}} \Phi_\rho(\theta) \quad (6)$$

The following lemma justifies this formulation.

**Lemma 3.1.** *The optimal value of problem 6 equals to the optimal value of problem 5.*

*Proof.* See Appendix C. The proof involves showing that any reparameterization can be expressed via messages.  $\square$

As shown in [18], TRW-E and TRW-T algorithms maintain the constraint of problem 6. Unfortunately, they do not guarantee that the objective function  $\Phi_\rho$  monotonically increases - in our experiments we have observed that sometimes it goes down. In fact, when the algorithms failed to converge the value of  $\Phi_\rho(\theta)$  often had gone into a loop. Next we design a new algorithm with the property that  $\Phi_\rho$  never decreases.

Our algorithm is shown in Fig. 1. It iterates between two steps: (a) reparameterizing vectors  $\theta^T$ , and (b) averaging element  $\omega \in \mathcal{V} \cup \mathcal{E}$ . The goal of the reparameterization step is to make sure that the algorithm satisfies the *min-marginal property*: in the beginning of the averaging operation for element  $\omega$  components of vectors  $\theta^T$  corresponding to element  $\omega$  give correct min-marginals for this element (for trees  $T \in \mathcal{T}_\omega$ ). This property will turn out to be crucial in the analysis of the algorithm.

Note that TRW-E and TRW-T algorithms can also be formulated as combinations of reparameterization and averaging operations<sup>4</sup>. However, they update vectors  $\theta^T$  in parallel, while we do it sequentially. Therefore, we call our algorithm “sequential tree-reweighted message passing” (TRW-S).

Reparameterization step 1(a) can be implemented in many different ways. One possibility is to convert vectors  $\theta^T$  to normal forms by running the ordinary max-product BP. However, this would be very expensive if trees are large. A more efficient technique is discussed in section 3.4.

### 3.1 Weak tree agreement

The algorithm in Fig. 1 does not specify what the stopping criterion is. In this section we address this issue by giving *weak tree agreement* condition (WTA). Later we will show that it characterizes local maxima of the algorithm with respect to function  $\Phi_\rho$ . More precisely, we will prove that the algorithm has a subsequence converging to a vector satisfying WTA condition. Moreover, if a vector satisfies this condition, then the algorithm will not make any progress, i.e. it will not increase function  $\Phi_\rho$ .

<sup>3</sup>Note that lemmas 5 and 11 in [18] seem to contain a mistake. Lemma 11, for example, says that TRW-T algorithm maintains the property  $\sum_T \rho^T \theta^T = \bar{\theta}$ . However, the proof does not take into account reparameterization step.

<sup>4</sup>TRW-T algorithm iterates between two phases: (a) running max-product BP for *all* trees, and (b) performing averaging operation for *all* nodes and edges. TRW-E is similar, except that in phase (a) it performs one parallel message update operation for all trees. In general, TRW-E and TRW-T algorithm do not satisfy the min-marginal property.

0. Initialize  $\boldsymbol{\theta}$  so that  $\boldsymbol{\theta} \in \mathcal{A}$  and  $\sum_T \rho^T \boldsymbol{\theta}^T \equiv \bar{\boldsymbol{\theta}}$ .
1. Select some order for nodes and edges in  $\mathcal{V} \cup \mathcal{E}$ . For each element  $\omega \in \mathcal{V} \cup \mathcal{E}$  find all trees  $\mathcal{T}_\omega \subset \mathcal{T}$  containing  $\omega$ . If there is more than one tree, then do the following:
  - (a) For all trees  $T \in \mathcal{T}_\omega$  reparameterize  $\boldsymbol{\theta}^T$  such that values  $\theta_{s;j}^T$  (if  $\omega = s$  is a node) or  $\theta_{s;j}^T + \theta_{st;jk}^T + \theta_{t;k}^T$  (if  $\omega = (s, t)$  is an edge) give correct min-marginals for tree  $T$  as in formulas 3.
  - (b) “Averaging” operation:
    - If  $\omega = s$  is a node in  $\mathcal{V}$  then
      - Compute  $\tilde{\theta}_s = \frac{1}{\rho_s} \sum_{T \in \mathcal{T}_s} \rho^T \theta_s^T$
      - Set  $\theta_s^T := \tilde{\theta}_s$  for trees  $T \in \mathcal{T}_s$
    - If  $\omega = (s, t)$  is an edge in  $\mathcal{E}$  then
      - Compute  $\tilde{\nu}_{st;jk} = \frac{1}{\rho_{st}} \sum_{T \in \mathcal{T}_{st}} \rho^T (\theta_{s;j}^T + \theta_{st;jk}^T + \theta_{t;k}^T)$
      - Set  $\theta_s^T, \theta_{st}^T, \theta_t^T$  for trees  $T \in \mathcal{T}_{st}$  so that  $\theta_{s;j}^T + \theta_{st;jk}^T + \theta_{t;k}^T = \tilde{\nu}_{st;jk}$
2. Check whether a stopping criterion is satisfied; if yes, terminate, otherwise go to step 1.

Figure 1: **Sequential tree-reweighted algorithm (TRW-S).**

In order to define WTA condition, it is convenient to introduce some notation. Let  $\text{OPT}^T(\boldsymbol{\theta}^T)$  be the set of optimal configurations for parameter  $\boldsymbol{\theta}^T$ . Let  $\text{OPT}(\boldsymbol{\theta})$  be the collection  $\{\text{OPT}^T(\boldsymbol{\theta}^T) \mid T \in \mathcal{T}\}$  of the sets of optimal configurations for vectors  $\boldsymbol{\theta}^T$ . It belongs to the set  $(2^{\mathcal{X}})^{|\mathcal{T}|} = 2^{\mathcal{X}} \times \dots \times 2^{\mathcal{X}}$  ( $|\mathcal{T}|$  times). For two collections  $\mathbb{S}, \tilde{\mathbb{S}} \in (2^{\mathcal{X}})^{|\mathcal{T}|}$  we will write  $\mathbb{S} \subset \tilde{\mathbb{S}}$  if  $\mathbb{S}^T \subset \tilde{\mathbb{S}}^T$  for any tree  $T$ .

Consider some collection of sets of configurations  $\mathbb{S} = \{\mathbb{S}^T\} \in (2^{\mathcal{X}})^{|\mathcal{T}|}$ . We say that  $\mathbb{S}$  is *consistent* if it satisfies the following three conditions:

- (a) For any tree  $T$  set  $\mathbb{S}^T$  is non-empty.
- (b) If node  $s$  is contained in trees  $T$  and  $T'$ , then for any configuration  $\mathbf{x} \in \mathbb{S}^T$  there exists configuration  $\mathbf{x}' \in \mathbb{S}^{T'}$  which agrees with  $\mathbf{x}$  on node  $s$ , i.e.  $x_s = x'_s$ .
- (c) If edge  $(s, t)$  is contained in trees  $T$  and  $T'$ , then for any configuration  $\mathbf{x} \in \mathbb{S}^T$  there exists configuration  $\mathbf{x}' \in \mathbb{S}^{T'}$  which agrees with  $\mathbf{x}$  on nodes  $s$  and  $t$ , i.e.  $x_s = x'_s, x_t = x'_t$ .

Now we can define WTA condition.

**Definition 3.2.** Vector  $\boldsymbol{\theta} = \{\boldsymbol{\theta}^T\} \in \mathcal{A}$  is said to satisfy the weak tree agreement condition if there exists collection  $\mathbb{S} \subset \text{OPT}(\boldsymbol{\theta})$  which is consistent.

Note that it can be viewed as a generalization of the *tree agreement* condition introduced in [18]: vectors satisfying tree agreement also satisfy WTA condition.

Also note that WTA condition is different from the *fixed point* condition of TRW algorithms. The latter means that any step of the algorithm does not change vector  $\boldsymbol{\theta}$ . This in turn implies that all vectors  $\boldsymbol{\theta}^T$  are in a normal form and  $\theta_\omega^T = \theta_\omega^{T'}$  for every element  $\omega \in \mathcal{V} \cup \mathcal{E}$  and for every pair of trees  $T, T' \in \mathcal{T}_\omega$ . It is easy to see that every fixed point of TRW satisfies WTA condition, but not the other way around.

### 3.2 Analysis of the algorithm: Main theorems

First we show that similarly to TRW-T algorithm, TRW-S maintains the constraint of problem 6.

**Lemma 3.3.** TRW-S algorithm performs reparameterization of the original parameter vector  $\bar{\boldsymbol{\theta}}$ , i.e. it maintains the property  $\sum_T \rho^T \boldsymbol{\theta}^T \equiv \bar{\boldsymbol{\theta}}$ .



*Proof.* By definition step 1(a) performs reparameterizations of tree vectors  $\theta^T$  and, thus, reparameterizations of the sum  $\sum_T \rho^T \theta^T$ . Checking that step 1(b) is a reparameterization is simple algebra.  $\square$

Next we analyze the behaviour of objective function  $\Phi_\rho$  during the algorithm. To be specific, we assume for the rest of this section that after reparameterization step 1(a) we have  $\min_j \{\theta_{s;j}^T\} = 0$  (if  $\omega = s$  is a node) or  $\min_{j,k} \{\theta_{s;j}^T + \theta_{st;jk}^T + \theta_{t;k}^T\} = 0$  (if  $\omega = (s, t)$  is an edge). This assumption is not essential, however; the behaviour of the algorithm will be the same for any other normalization.

- Theorem 3.4.** (a) *After any number of steps functions  $\Phi$  and  $\Phi_\rho$  do not decrease.*
- (b) *If vector  $\theta$  does not satisfy WTA condition then after a finite number of iterations function  $\Phi_\rho$  will increase.*
- (c) *If vector  $\theta$  satisfies WTA condition with collection  $\mathbb{S}$  then after any number of steps it will still satisfy WTA with the same collection  $\mathbb{S}$ . Function  $\Phi_\rho$  will not change.*

A proof of this theorem is given in the next section.

As an immediate consequence of theorem 3.4(b) we get the following result:

**Corollary 3.5.** *If vector  $\theta$  maximizes problem 6 then  $\theta$  satisfies WTA condition.*

Unfortunately, the converse is not necessarily true as example in Appendix D demonstrates.

Finally, we give the convergence theorem. A proof is given in the next section.

**Theorem 3.6.** *Let  $\{\theta^{(i)}\}_i$  be an infinite sequence of vectors obtained by applying step 1 of TRW-S algorithm. Let  $\{\tilde{\theta}^{(i)}\}_i$  be the reparameterization of this sequence obtained by reparameterizing vectors  $\theta^{(i)}$  into the canonical normal form. Then there is a subsequence  $\{\tilde{\theta}^{(i(m))}\}_m$  such that*

- (a) *It converges to some vector  $\theta^* \in \mathcal{A}$ .*
- (b) *Sequence  $\{\Phi_\rho(\tilde{\theta}^{(i)})\}_i$  converges to  $\Phi_\rho(\theta^*)$ .*
- (c) *Vector  $\theta^*$  satisfies WTA condition.*

Note that since the algorithm does not specify exactly how we reparameterize vectors  $\theta^{(i)}$ , we cannot claim that sequence  $\{\theta^{(i)}\}_i$  always contains a converging subsequence. As a counterexample, we could choose  $\{\theta^{(i)}\}_i$  so that it is not bounded. Reparameterization  $\{\tilde{\theta}^{(i)}\}_i$  in the theorem was chosen only to make sure that it is bounded.

### 3.3 Analysis of the algorithm: Proofs

In this section we prove theorems 3.4 and 3.6. One technical part, however, is deferred to Appendix E.

#### 3.3.1 Proof of theorem 3.4

All quantities used in the theorem are defined using functions  $E(\cdot|\theta^T)$ . By definition, reparameterization step 1(a) does not change these functions. Therefore, we do not need to consider these steps, so we only need to analyze averaging step 1(b). The following lemma shows the effect of the averaging operation on functions  $\Phi$ .

**Lemma 3.7.** *Suppose that the averaging operation for element  $\omega$  transforms vectors  $\theta^T$  to vectors  $\tilde{\theta}^T$ .*

- *If  $\omega = s$  is a node, then  $\Phi_{s;j}(\tilde{\theta}^T) = \Phi(\theta^T) + \tilde{\theta}_{s;j}$  for any tree  $T \in \mathcal{T}_s$ .*
- *If  $\omega = (s, t)$  is an edge, then  $\Phi_{st;jk}(\tilde{\theta}^T) = \Phi(\theta^T) + \tilde{\theta}_{s;j} + \tilde{\theta}_{st;jk} + \tilde{\theta}_{t;k}$  for any tree  $T \in \mathcal{T}_{st}$ .*

*Proof.* We consider only the case when  $\omega = s$  is a node. The second case is very similar.

Because of our normalization assumption we have

$$\Phi(\theta^T) = \min_{j \in \mathcal{X}_s} \Phi_{s;j}(\theta^T) = \min_{j \in \mathcal{X}_s} \{\theta_{s;j}^T\} + \text{const}_s = \text{const}_s$$

where  $\text{const}_s$  is the constant in formula 3a. Plugging this into formula 3a we get  $\Phi_{s;j}(\theta^T) = \Phi(\theta^T) + \theta_{s;j}^T$ .

Vectors  $\theta^T$  and  $\tilde{\theta}^T$  agree on all nodes and edges other than node  $s$ . Thus, they have the same optimal configuration  $\mathbf{x}^j$  under the constraint  $x_s^j = j$ . We can write

$$\Phi_{s;j}(\tilde{\theta}^T) = E(\mathbf{x}^j | \tilde{\theta}^T) = E(\mathbf{x}^j | \theta^T) - \theta_{s;j}^T + \tilde{\theta}_{s;j}^T = \Phi_{s;j}(\theta^T) - \theta_{s;j}^T + \tilde{\theta}_{s;j}^T = \Phi(\theta^T) + \tilde{\theta}_{s;j}^T$$

□

Parts (a) and (c) of theorem 3.4 are immediate consequences of this lemma. Indeed, for the node averaging operation, for example, we can write

$$\Phi(\tilde{\theta}^T) = \min_{j \in \mathcal{X}_s} \Phi_{s;j}(\tilde{\theta}^T) = \Phi(\theta^T) + \min_{j \in \mathcal{X}_s} \{\tilde{\theta}_{s;j}^T\} \quad (7)$$

We have  $\theta_s^T \geq 0$ ; inspecting update rule of step 1(b) we conclude that  $\tilde{\theta}_s^T \geq 0$  as well. Therefore, the minimum on the RHS is non-negative, so  $\Phi(\tilde{\theta}^T) \geq \Phi(\theta^T)$  and  $\sum_T \rho^T \Phi(\tilde{\theta}^T) \geq \sum_T \rho^T \Phi(\theta^T)$ . This proves part (a).

Let us prove part (c). Suppose that  $\theta$  satisfies WTA with collection  $\mathbb{S} = \{\mathbb{S}^T\}$ . We need to prove that  $\mathbb{S} \subset \text{OPT}(\tilde{\theta})$ . We can ignore trees  $T'$  that do not contain node  $s$  since vector  $\theta^{T'}$  and set  $\text{OPT}(\theta^{T'})$  do not change. Let us pick a tree  $T \in \mathcal{T}_s$ . Consider a configuration  $\mathbf{x} \in \mathbb{S}$ , and let  $j = x_s$ . Since  $\mathbb{S}$  is consistent, for any tree  $T' \in \mathcal{T}_s$  there exists configuration  $\mathbf{x}'$  with  $x_s' = j$  which is optimal for  $\theta^{T'}$ . This implies that  $\theta_{s;j}^{T'} = 0$  for any  $T' \in \mathcal{T}_s$ . Therefore, we have  $\tilde{\theta}_{s;j}^T = 0$ .

One consequence of this fact is that the minimum in equation 7 is zero, so  $\Phi(\tilde{\theta}^T) = \Phi(\theta^T)$ . Also, it means that  $E(\mathbf{x} | \tilde{\theta}) = \Phi(\tilde{\theta}^T)$ , i.e.  $\mathbf{x}$  is an optimal configuration for vector  $\tilde{\theta}^T$ :  $\mathbf{x} \in \text{OPT}^T(\tilde{\theta}^T)$ . Part (c) is proved.

To prove part (b), we need the following lemma.

**Lemma 3.8.** *Suppose that a complete pass of step 1 transforms vector  $\theta$  to vector  $\tilde{\theta}$  and  $\Phi_\rho(\theta) = \Phi_\rho(\tilde{\theta})$ . Then  $\text{OPT}(\tilde{\theta})$  does not acquire new configurations, i.e.  $\text{OPT}(\tilde{\theta}) \subset \text{OPT}(\theta)$ . If in addition  $\text{OPT}(\theta)$  is not consistent then it will shrink, i.e. the inclusion  $\text{OPT}(\tilde{\theta}) \subset \text{OPT}(\theta)$  is strict.*

*Proof.* First we show that if  $\Phi_\rho$  stays the same during a single step then  $\text{OPT}(\theta)$  cannot acquire any new configurations, i.e.  $\text{OPT}(\tilde{\theta}) \subset \text{OPT}(\theta)$ . Again, consider the averaging operation for node  $s$ , and consider a tree  $T$  containing this node. Suppose that  $\mathbf{x} \in \text{OPT}(\tilde{\theta}^T)$ , and let  $j = x_s$ . Condition  $\Phi_\rho(\tilde{\theta}) = \Phi_\rho(\theta)$  implies that  $\Phi(\tilde{\theta}^T) = \Phi(\theta^T)$  since all values in the sum  $\Phi_\rho(\theta) = \sum_T \rho^T \Phi(\theta^T)$  do not decrease. Therefore, the minimum in formula 7 is zero, so  $\tilde{\theta}_{s;j}^T = 0$ . This could only have happened if  $\theta_{s;j}^{T'} = 0$  for all trees  $T' \in \mathcal{T}_s$ , which means that  $\mathbf{x}$  was an optimal configuration for vector  $\theta^T$ . We proved that  $\text{OPT}(\tilde{\theta}) \subset \text{OPT}(\theta)$ .

Now suppose that  $OPT(\boldsymbol{\theta})$  is not consistent and a complete pass of step 1 does not change the value of function  $\Phi_\rho$ . It means that in the beginning of step 1 consistency condition for collection  $OPT(\boldsymbol{\theta})$  is violated for some element (either a node or an edge). Consider the moment when the algorithm reaches this element. If collection  $OPT(\boldsymbol{\theta})$  has already shrunk by that time, then we do not have to prove anything - the lemma holds. Let us assume that  $OPT(\boldsymbol{\theta})$  stays the same. Thus, the consistency condition is still violated for this element.

Below we consider only the case when this element is a node  $s$ ; the situation for an edge can be analyzed similarly. Thus, we assume that there exist trees  $T$  and  $T'$  containing the node and configuration  $\mathbf{x}$  with  $x_s = j$  such that  $\mathbf{x}$  is an optimal configuration for vector  $\boldsymbol{\theta}^T$  (so  $\theta_{s;j}^T = 0$ ) but there exists no configuration  $\mathbf{x}'$  with  $x'_s = j$  which is optimal for  $\boldsymbol{\theta}^{T'}$  (so  $\theta_{s;j}^{T'} > 0$ ). This implies that  $\tilde{\theta}_s^T > 0$ , therefore configuration  $\mathbf{x}$  is no longer optimal for  $\tilde{\boldsymbol{\theta}}^T$ . Thus, collection  $OPT(\tilde{\boldsymbol{\theta}})$  has shrunk. The lemma is proved.  $\square$

We now proceed with the proof of theorem 3.4(b). Let  $\boldsymbol{\theta}^0, \boldsymbol{\theta}^1, \boldsymbol{\theta}^2, \dots$  be the sequence of vectors obtained from one another by applying a complete pass of step 1 (where  $\boldsymbol{\theta}^0 = \boldsymbol{\theta}$ ). Suppose that function  $\Phi_\rho$  stays the same after any number of steps:  $\Phi_\rho(\boldsymbol{\theta}^0) = \Phi_\rho(\boldsymbol{\theta}^1) = \Phi_\rho(\boldsymbol{\theta}^2) = \dots$ . Let us show that  $\boldsymbol{\theta}$  satisfies WTA condition.

By lemma 3.8 we have  $OPT(\boldsymbol{\theta}) \supset OPT(\boldsymbol{\theta}^1) \supset OPT(\boldsymbol{\theta}^2) \supset \dots$ . Since  $OPT(\boldsymbol{\theta})$  is finite it cannot shrink indefinitely, therefore after a finite number of iterations (let us say,  $n$ )  $OPT(\boldsymbol{\theta}^n)$  will become consistent. We have  $OPT(\boldsymbol{\theta}^n) \subset OPT(\boldsymbol{\theta})$  so by definition vector  $\boldsymbol{\theta}$  satisfies WTA condition. Theorem 3.4 is proved.

### 3.3.2 Proof of theorem 3.6

To simplify notation, we assume that sequence  $\{\boldsymbol{\theta}^{(i)}\}_i$  is already in the canonical normal form, so  $\tilde{\boldsymbol{\theta}}^{(i)} = \boldsymbol{\theta}^{(i)}$ . Appendix E proves that sequence  $\{\boldsymbol{\theta}^{(i)}\}_i$  is bounded, therefore the existence of converging subsequence follows from Bolzano-Weierstrass theorem. Part (b) follows from the facts that sequence  $\{\Phi_\rho(\boldsymbol{\theta}^{(i)})\}_i$  is non-decreasing and  $\Phi_\rho$  is continuous. We now prove that for any converging subsequence  $\{\boldsymbol{\theta}^{(i(m))}\}_m$  the limit  $\boldsymbol{\theta}^*$  satisfies WTA condition.

Suppose that this is not true. Let us apply TRW-S algorithm to vector  $\boldsymbol{\theta}^*$ . Theorem 3.4(b) says that after a finite number of steps (let us say,  $n$ ) we obtain a configuration  $\tilde{\boldsymbol{\theta}}^*$  such that  $\Phi_\rho(\tilde{\boldsymbol{\theta}}^*) > \Phi_\rho(\boldsymbol{\theta}^*)$ .

Let  $\pi : \mathcal{A} \rightarrow \mathcal{A}$  be a mapping defined as follows: we take vector  $\boldsymbol{\theta}$  and apply  $n$  steps of TRW-S algorithm. Each step is a continuous mapping  $\mathcal{A} \rightarrow \mathcal{A}$ , therefore  $\pi$  is continuous as well. Thus,

$$\{\pi(\boldsymbol{\theta}^{(i(m))})\} \xrightarrow{m \rightarrow \infty} \pi(\boldsymbol{\theta}^*) = \tilde{\boldsymbol{\theta}}^*$$

Function  $\Phi_\rho$  is also continuous, so

$$\{\Phi_\rho(\pi(\boldsymbol{\theta}^{(i(m))}))\} \xrightarrow{m \rightarrow \infty} \Phi_\rho(\tilde{\boldsymbol{\theta}}^*)$$

Thus, there exists index  $m$  such that  $\Phi_\rho(\pi(\boldsymbol{\theta}^{(i(m))})) \geq \Phi_\rho(\boldsymbol{\theta}^*) + \epsilon$  where  $\epsilon = \frac{1}{2}(\Phi_\rho(\tilde{\boldsymbol{\theta}}^*) - \Phi_\rho(\boldsymbol{\theta}^*)) > 0$ . Note that  $\pi(\boldsymbol{\theta}^{(i(m))}) = \boldsymbol{\theta}^{(n+i(m))}$ . Using theorem 3.4(a), we get that  $\Phi_\rho(\boldsymbol{\theta}^{(i)}) \geq \Phi_\rho(\boldsymbol{\theta}^*) + \epsilon$  for any index  $i \geq n+i(m)$ , which contradicts to part (b) of the convergence theorem.

## 3.4 TRW-S algorithm for a graph with monotonic chains

In this section we focus on step 1(a) - reparameterizing vector  $\boldsymbol{\theta}^T$ . Recall that its goal is to make sure that the algorithm satisfies the min-marginal property.

For simplicity, consider the case when  $\omega = s$  is a node. In general, a complete forward pass of the ordinary max-product BP is needed for trees  $T \in \mathcal{T}_s$  - sending messages from leaves to

0. Initialize  $\theta$  so that  $\theta \in \mathcal{A}$  and  $\sum_T \rho^T \theta^T \equiv \bar{\theta}$ .
1. For nodes  $s \in \mathcal{V}$  do the following operations in the order of increasing  $i(s)$ :
  - (a) Perform the averaging operation for node  $s$ .
  - (b) For every edge  $(s, t) \in \mathcal{E}$  with  $i(t) > i(s)$  do the following:
    - If  $\mathcal{T}_{st}$  contains more than one chain then perform the averaging operation for edge  $(s, t)$  so that vectors  $\theta_s^T$  do not change.
    - For chains in  $\mathcal{T}_{st}$  pass a message from  $s$  to  $t$ .
2. Reverse the ordering: set  $i(u) := |\mathcal{V}| + 1 - i(u)$ .
3. Check whether a stopping criterion is satisfied; if yes, terminate, otherwise go to step 1.

Figure 2: **TRW-S algorithm for a graph with monotonic chains.**

node  $s$  which we treat as a root<sup>5</sup>. However, this would make the algorithm very inefficient if trees are large. Fortunately, a complete forward pass is not always necessary.

The key idea is that the averaging operation does not invalidate certain messages in trees  $T$ , as proposition 3.9 below shows. In other words, if a message was valid before the operation, then it remains valid after the operation<sup>6</sup>. Therefore, we can “reuse” some of the messages passed in previous steps, i.e. not pass them again.

**Proposition 3.9.** *The averaging operation for element  $\omega \in \mathcal{V} \cup \mathcal{E}$  does not invalidate messages in trees  $T \in \mathcal{T}_\omega$  oriented towards  $\omega$ .*

*Proof.* Consider edge  $(s \rightarrow t)$  oriented towards  $\omega$  in tree  $T$ . The averaging operation can affect only the endpoint vector  $\theta_t^T$ . However, condition 2 of a valid message involves vectors  $\theta_s^T$  and  $\theta_{st}^T$  but not  $\theta_t^T$ .  $\square$

To exploit this property fully, we need to choose trees and the order of averaging operations in a particular way. Specifically, we require trees to be chains which are *monotonic* with respect to some ordering on the graph:

**Definition 3.10.** *Graph  $\mathcal{G}$  and chains  $T \in \mathcal{T}$  are said to be monotonic if there exists an ordering of nodes  $i(u), u \in \mathcal{V}$  such that each chain  $T$  satisfies the following property: if  $u_1^T, \dots, u_{n(T)}^T$  are the consecutive nodes in the chain, then the sequence  $i(u_1^T), \dots, i(u_{n(T)}^T)$  is monotonic.*

As an example, we could choose  $\mathcal{T}$  to be the set of edges; it is easy to see that they are monotonic for any ordering of nodes. However, it might be advantageous to choose longer trees since the information might propagate faster through the graph.

The algorithm for a graph with monotonic chains is shown in Fig. 2. Its properties are summarized by the following lemma.

**Lemma 3.11.** *Starting with the second pass, the following properties hold during step 1 for node  $s$ :*

- (a) *For each edge  $(u, v) \in \mathcal{E}$  with  $i(u) < i(v)$  and  $i(u) < i(s)$  messages  $(u \rightarrow v)$  in trees  $T \in \mathcal{T}_{uv}$  are valid. This property also holds for node  $u = s$  in the end of step 1(b).*

<sup>5</sup>Note that the backward pass (sending messages from the root to leaves) is not needed. It would convert vectors  $\theta^T$  to normal forms but would not change vectors  $\theta_s^T$ .

<sup>6</sup>Recall that our algorithm is message-free. As discussed in section 2.2, the phrase “message is valid for directed edge  $(s \rightarrow t)$  in tree  $T$ ” means that equation 2 holds (or that sending a message from node  $s$  to node  $t$  would not modify  $\theta_{st,jk}^T$  and  $\theta_{t,k}^T$  except for a constant independent of  $j$  or  $k$ ).

0. Set all messages to zero.
1. For nodes  $s \in \mathcal{V}$  do the following operation in the order of increasing  $i(s)$ :
  - Compute  $\widehat{\theta}_s = \bar{\theta}_s + \sum_{(u,s) \in \mathcal{E}} M_{us}$
  - For every edge  $(s, t) \in \mathcal{E}$  with  $i(s) < i(t)$  update message  $M_{st}$  as follows:

$$M_{st;k} := \min_j \left\{ \left( \frac{\rho_{st}}{\rho_s} \widehat{\theta}_{s;j} - M_{ts;j} \right) + \bar{\theta}_{st;jk} \right\}$$

2. Reverse the ordering: set  $i(u) := |\mathcal{V}| + 1 - i(u)$ .
3. Check whether a stopping criterion is satisfied; if yes, terminate, otherwise go to step 1.

Figure 3: **Efficient implementation of the algorithm in Fig. 2 using messages.**

(b) For each edge  $(u, v) \in \mathcal{E}$  with  $i(s) < i(u) < i(v)$  messages  $(v \rightarrow u)$  in trees  $T \in \mathcal{T}_{uv}$  are valid. This property also holds for node  $u = s$  in the beginning and in the end of step 1(a).

In addition, property (a) holds during the first pass of the algorithm.

*Proof.* We will use induction. The base of induction is straightforward - right after initialization the set of messages considered in the lemma is empty.

We need to consider the following cases:

- Assuming that the lemma holds in the beginning of step 1(a), prove that it holds in the end of step 1(a).
- Assuming that the lemma holds in the beginning of step 1(b), prove that it holds in the end of step 1(b).
- Assuming that the lemma holds in the end of step 1(b) for node  $s$ , prove that it holds in the beginning of step 1(a) for the next node  $s'$  with  $i(s') = i(s) + 1$ .
- Assuming that the lemma holds in the end of step 1(b) for the last node, prove that it holds in the beginning of step 1(a) if the order of nodes is reversed.

The last two cases are straightforward - they do not involve any reparameterization, and the set of messages considered in the postcondition is the same or smaller than the set of messages in the precondition. The first two cases follow from proposition 3.9 and the fact that after passing a message from node  $s$  to node  $t$  in tree  $T \in \mathcal{T}_{st}$  message  $(s \rightarrow t)$  in this tree becomes valid.  $\square$

The lemma implies that starting with the second pass, all messages in trees  $T \in \mathcal{T}_\omega$  oriented towards element  $\omega \in \mathcal{V} \cup \mathcal{E}$  are valid in the beginning of the averaging operation for element  $\omega$ . Therefore, passing messages from leaves to  $\omega$  would not change parameters  $\theta^T$  (except for constants), so the algorithm satisfies the min-marginal property. Note that this property may not hold during the first pass of the algorithm; however, we can treat this pass as a part of initialization. Then the algorithm in Fig. 2 becomes a special case of the algorithm in Fig. 1.

**Efficient implementation** The algorithm in Fig. 2 requires  $O(|\mathcal{T}_s| \cdot |\mathcal{X}_s|)$  storage for node  $s$  and  $O(|\mathcal{T}_{st}| \cdot |\mathcal{X}_s| \cdot |\mathcal{X}_t|)$  storage for edge  $(s, t)$ . However, we can reduce it to  $O(|\mathcal{X}_s|)$  and  $O(|\mathcal{X}_s| + |\mathcal{X}_t|)$ , respectively, using two ideas<sup>7</sup>. First, it can be seen that the algorithm maintains

<sup>7</sup>We assume that storage required for vectors  $\bar{\theta}_{st}$  is negligible. This holds for many energy functions used in practice, e.g. for functions with Potts terms.

the following equalities:  $\theta_s^T = \theta_s^{T'}$  for  $T, T' \in \mathcal{T}_s$  and  $\theta_{st}^T = \theta_{st}^{T'}$  for  $T, T' \in \mathcal{T}_{st}$  (assuming that they hold after initialization). Second, vectors  $\theta^T$  can be stored using messages  $M_{st} = \{M_{st;k} \mid k \in \mathcal{X}_t\}$  for directed edges  $(s \rightarrow t) \in \mathcal{E}$  according to the following formulas:<sup>8</sup>

$$\begin{aligned}\theta_t^T &= \frac{1}{\rho_t}(\bar{\theta}_t + \sum_{(s,t) \in \mathcal{E}} M_{st}) \\ \theta_{st;jk}^T &= \frac{1}{\rho_{st}}(\bar{\theta}_{st;jk} - M_{st;k} - M_{ts;j})\end{aligned}$$

The resulting algorithm is shown in Fig. 3. Note that for many important choices of terms  $\bar{\theta}_{st}$  message update in step 1 can be done very efficiently using distance transforms [4].

An important property of our algorithm is that it requires half as much memory as traditional BP. Indeed, the latter needs to store messages in both direction for each edge, while we can store only messages oriented *towards* current node  $s$  (or, more precisely, messages that are valid according to lemma 3.11). The reverse messages are not needed since we update them before they are used. The same space in memory can be used for storing either message  $M_{st}$  or  $M_{ts}$ . The exact moment when  $M_{ts}$  gets replaced with  $M_{st}$  is when edge  $(s, t)$  is processed during step 1 for node  $s$ .

This observation can also be applied to traditional BP with the same schedule of passing messages - we just need to set  $\rho_{st}/\rho_s = 1$ . The fact that memory requirements of BP can be reduced by half was first observed in [4]. However, they considered only bipartite graphs, and their goal was to simulate the parallel schedule of updating messages. We show that the amount of memory needed can be reduced for any graph. In fact, we give a strict generalization of the technique in [4]. Indeed, the schedule used in [4] is a special case of our sequential schedule if we choose the ordering of nodes such that any node in the first set of a bipartite graph is before any node in the second set.

The complexity of one pass of our algorithm is  $O(|\mathcal{E}| \cdot K)$  where  $K = \max_s |\mathcal{X}_s|$  (assuming that distance transforms can be done in  $O(K)$  time). This also compares favorably with traditional BP. The latter needs  $O(d_s \cdot K)$  time to pass a message from  $s$  to  $t$  where  $d_s$  is the degree of node  $s$ , which results in  $O(\max_s d_s \cdot |\mathcal{E}| \cdot K)$  complexity of one pass.

## 4 Experimental results

We have compared four algorithms: ordinary max-product BP and three tree-reweighted algorithms (TRW-E, TRW-T, TRW-S). Their settings are summarized below.

**Choice of trees** In all experiments we constructed trees in a greedy manner as follows. Given a graph, we select a monotonic chain such that it is not possible to extend it, i.e. for the first node  $s$  there are no edges  $(u, s)$  with  $i(u) < i(s)$  and for the last node  $t$  there are no edges  $(t, v)$  with  $i(v) > i(t)$ . After that we remove corresponding edges from the graph and repeat the procedure until no edges are left. Thus, we ensure that each edge is covered by exactly one tree. All trees are assigned the uniform probability.

Clearly, this method can produce different sets of trees depending on what chains we choose. However, the behaviour of TRW-S algorithm is specified uniquely (assuming that the order of nodes is fixed). Indeed, the algorithm depends only on ratios  $\rho_{st}/\rho_s$  for edges  $(s \rightarrow t)$ . It can be seen that the number of trees containing node  $s$  is

$$n_s = \max\{ |(u, s) \in \mathcal{E} : i(u) < i(s)|, |(s, v) \in \mathcal{E} : i(v) > i(s)| \}.$$

---

<sup>8</sup>Note that messages  $M_{st}$  that we use here are slightly different from messages  $M_{st}^{[9]}$  used in [9, 18]. The relationship between the two is as follows:  $M_{st} = \rho_{st} M_{st}^{[9]}$ . We decided to scale the messages because it simplified some of the equations.

Therefore, we have  $\rho_{st}/\rho_s = 1/n_s$ .

**Choosing solution** An important question is how to construct solution  $\mathbf{x}$  given reparameterization  $\hat{\theta} = \sum_T \rho^T \theta^T$ . A possible approach is to choose label  $x_s$  for node  $s$  that minimizes  $\hat{\theta}_s(x_s)$ . However, it is often the case that the minimum is not unique within the precision of floating point numbers. This is not surprising. Indeed, if we all nodes have unique minimum then it means that we found the optimal solution, as shown in [18]. In general we cannot expect this since minimizing energy 1 is NP-hard.

Thus, it is essential how we treat nodes with multiple minima. We used the following technique. We assign labels to nodes in some order  $i(s)$  (which in fact was the same as in TRW-S algorithm). For node  $s$  we choose label  $x_s$  that minimizes  $\hat{\theta}_s(x_s) + \sum_{i(u) < i(s)} \hat{\theta}_{us}(x_u, x_s)$  where the sum is over edges  $(u, s) \in \mathcal{E}$ . In terms of messages, this is equivalent to minimizing  $\bar{\theta}_s(x_s) + \sum_{i(u) < i(s)} \bar{\theta}_{us}(x_u, x_s) + \sum_{i(v) > i(s)} M_{vs}(x_s)$ . This scheme alleviates the problem of multiple minima, but does not solve it completely. Many nodes are still assigned essentially at random (more precisely, the solution is determined by numerical errors).

**Damping** For TRW-E and TRW-T algorithms we also used damping parameter  $\gamma \in (0, 1]$ ; as reported in [18], the algorithms converge if sufficiently damped. For each problem described below we chose  $\gamma$  as follows. We tried values 0.1, 0.2, ..., 0.9, 1 and determined the average energy after 30 iterations. (The averaging was performed over the different instances of the same problem). Then we picked the value with the smallest energy. If the optimal value was  $\gamma = 0.1$ , we kept halving it until the energy started increasing. Note that damping was necessary for TRW-T algorithm, otherwise it always diverged. As for TRW-E algorithm, this was necessary only in some of the cases.

**BP algorithm** For ordinary max-product BP algorithm we implemented the same sequential schedule of updating messages as for TRW-S algorithm. We experimented with the parallel update scheme and found that it was much slower.

## 4.1 Synthetic problems

We tested the algorithms on two types of graphs: grids 30x30 with 4-neighborhood system and complete graphs with 50 nodes. Moreover, in each case we tested the Ising model with attractive potentials and with mixed potentials. Single-node potentials were generated as independent gaussians:  $\bar{\theta}_{s;0}, \bar{\theta}_{s;1} \sim \mathcal{N}(0, 1)$ . Pairwise potentials were set as follows:  $\bar{\theta}_{st;00} = \bar{\theta}_{st;11} = 0$ ,  $\bar{\theta}_{st;01} = \bar{\theta}_{st;10} = \lambda_{st}$  where  $\lambda_{st}$  was generated as  $|\mathcal{N}(0, \sigma^2)|$  for attractive potentials and as  $\mathcal{N}(0, \sigma^2)$  for mixed potentials. Parameter  $\sigma$  determines the strength of potentials; we used values  $\sigma \in \{\frac{1}{\sqrt{d}}, \frac{2}{\sqrt{d}}, \frac{3}{\sqrt{d}}\}$  where  $d$  is the degree of nodes ( $d = 4$  for the grid and  $d = 49$  for the complete graph). We tested the algorithms on 100 sample problems.

As a measure of performance we used two quantities: the value of the lower bound and the energy of current solution (except for BP algorithm where we can determine only one of these quantities). Note that the former is a lower bound on the optimal value of the energy, and the latter is an upper bound. We plot these quantities as functions of the number of iterations (i.e. the number of passed messages divided by the number of directed edges in the graph). We report only the average values over 100 samples.

**Attractive potentials** The results for attractive potentials are shown in Fig. 4. Note that in this case the global minimum can be found in polynomial time using the maxflow algorithm. In all cases TRW-S outperforms TRW-E and TRW-T. It also outperforms BP since it converges to the global minimum of the energy, while BP does not. However, in the case of grid graph TRW-S algorithm is slower in the beginning than BP.

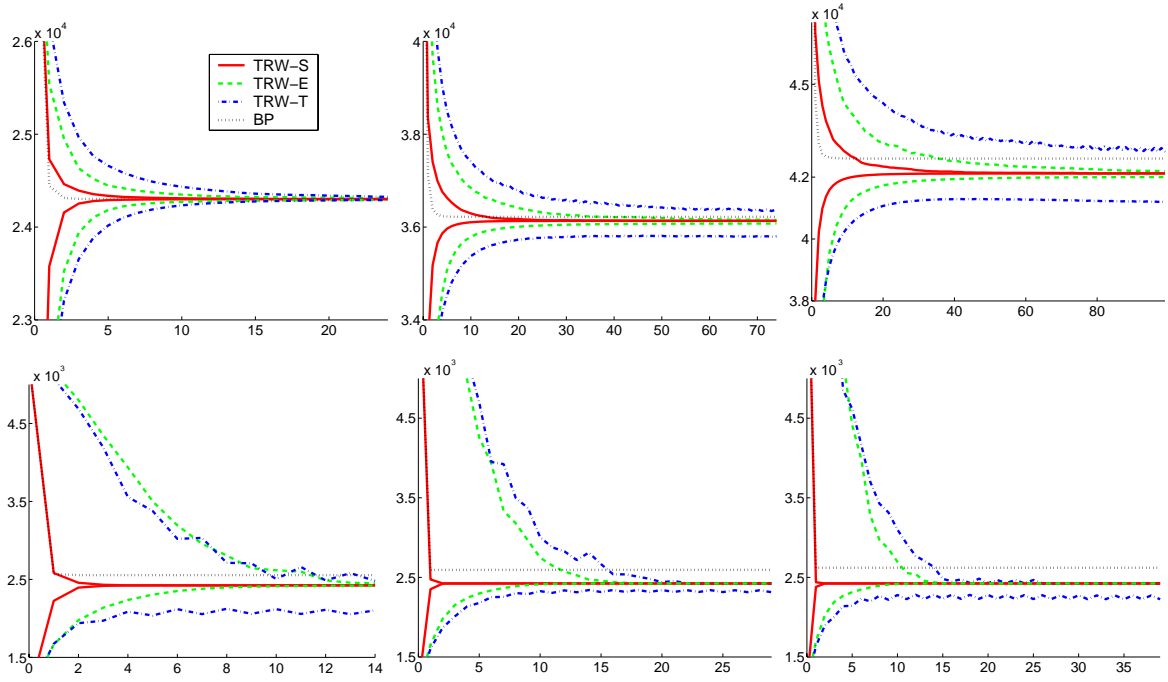


Figure 4: Synthetic problems, attractive potentials. Horizontal axis: number of iterations. Vertical axis, upper curves: average value of the energy  $E(\mathbf{x}|\bar{\theta})$ . Vertical axis, lower curves for TRW algorithms: average value of  $\Phi_\rho(\theta)$ . Columns 1-3: different strengths of the interaction potential ( $\sigma = \frac{1}{\sqrt{d}}, \frac{2}{\sqrt{d}}, \frac{3}{\sqrt{d}}$ ). First row: grid graph 30x30. Second row: complete graph with 50 nodes.

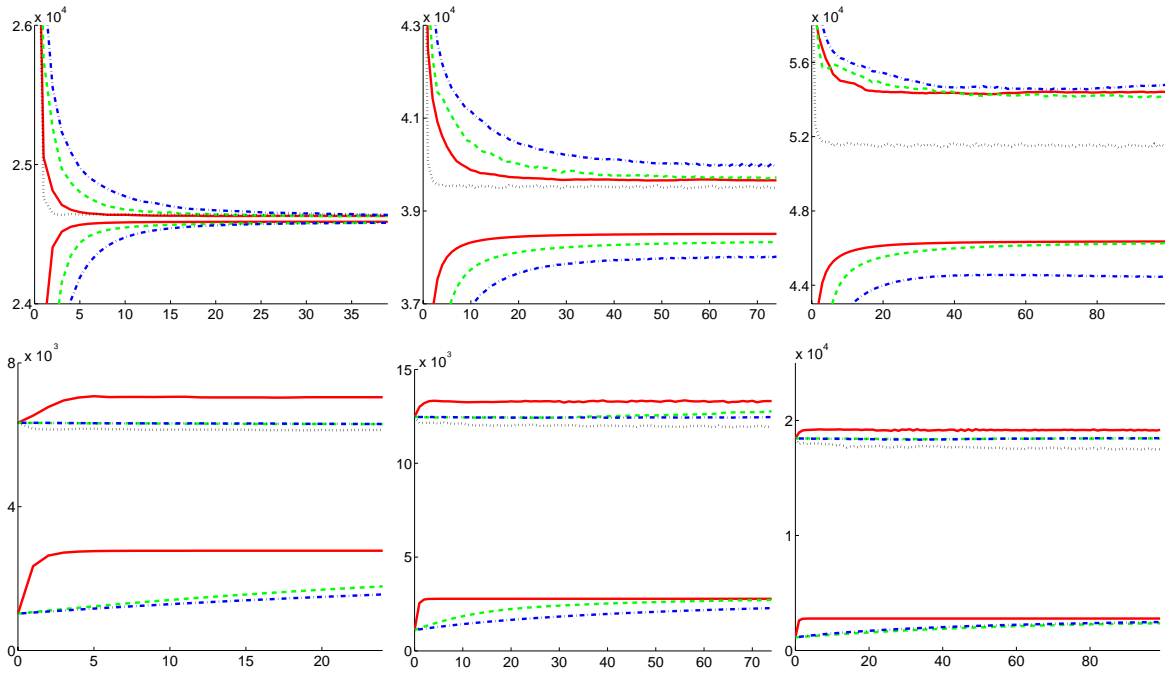


Figure 5: Synthetic problems, mixed potentials. For description, see Fig. 4.



**Mixed potentials** The results for this case are shown in Fig. 5. The situation is very different from the case of attractive potentials. In all cases BP outperforms TRW algorithms. We believe that this happens because BP yields a smaller number of nodes  $s$  with multiple minima of  $\hat{\theta}_s(j)$ . To improve results, it is essential to change the way how we choose solution  $\mathbf{x}$  from reparameterization  $\hat{\theta}$ .

## 4.2 Real problems

**Binary segmentation** First we tested the algorithms on the energy function in *grabcut* [14]. This is a function with binary labels where unary data terms come from user-provided hard constraints and learned background and foreground color models, and pairwise terms come from image gradients. A regular grid graph with 8-neighborhood system is used. Hard (background) constraints may exist only at the grid boundary; they correspond to a rectangle dragged around the object. Note that the function has attractive potentials, so a global minimum can be computed via the maxflow algorithm.

We used 49 different instances with the average size  $1.18 \times 10^5$  nodes. Energy plots are shown in Fig. 6. The average running time for TRW-S algorithm was 0.15 secs per iteration on Pentium IV 3.2 GHz processor.

**Stereo matching** We have tested the algorithms on the energy function arising in the stereo matching problem with Potts interactions [2]. The input is two images taken from different viewpoints, and the goal is to find a disparity for every pixel in the left image. We used the four datasets from [15] - Tsukuba, Map, Sawtooth and Venus. Fig. 6 shows the energy plots, and Fig. 7 shows disparity maps for one of the datasets (Tsukuba).

In addition to message passing techniques, we included the results of the expansion move method of Boykov, Veskler and Zabih [2], which is referred to as BVZ. Note that the horizontal axis for BVZ denotes time; to match it to the number of messages used for other techniques, we used the timing of TRW-S algorithm. Thus, the correspondence between BVZ and other message passing algorithms (TRW-E, TRW-T, BP) is only approximate<sup>9</sup>. Each point in Fig. 6 corresponds to one iteration of BVZ algorithm, and the last point indicates that BVZ has converged.

TRW-S clearly outperforms other message passing techniques. Compared to BVZ, it is slower in the beginning, but eventually finds lower energy as shown below:

| dataset  | size<br>(W x H x D) | TRW-S<br>time | TRW-S<br>accuracy | BVZ<br>accuracy |
|----------|---------------------|---------------|-------------------|-----------------|
| Tsukuba  | 384 x 288 x 16      | 0.348         | 0.0037%           | 1.0%            |
| Map      | 284 x 216 x 30      | 0.374         | 0.055%            | 0.42%           |
| Sawtooth | 434 x 380 x 20      | 0.707         | 0.096%            | 0.12%           |
| Venus    | 434 x 383 x 22      | 0.822         | 0.014%            | 0.061%          |

Column “TRW-S time” shows the time per one iteration on Pentium IV 3.2 GHz processor. The column “accuracy” reflects the value of the minimum energy found. For TRW-S it is defined as  $\frac{E_{min} - E_{bound}}{E_{bound}} \cdot 100\%$  where  $E_{min}$  and  $E_{bound}$  are the best values of energy and lower bound, respectively, found during 512 iterations. For BVZ the accuracy is defined similarly, only  $E_{min}$  is now the value of the energy at convergence.

<sup>9</sup>In our implementation one iteration of TRW-E and TRW-T is much slower than TRW-S. One reason is that we have not optimized TRW-E and TRW-T, but there are also objective reasons; for example, TRW-E and TRW-T require at least twice as much memory as TRW-S. On the other hand, one iteration of BP can be implemented slightly faster and with less memory than TRW-S since BP can work with integers while it is preferable to use double precision numbers for TRW-S.

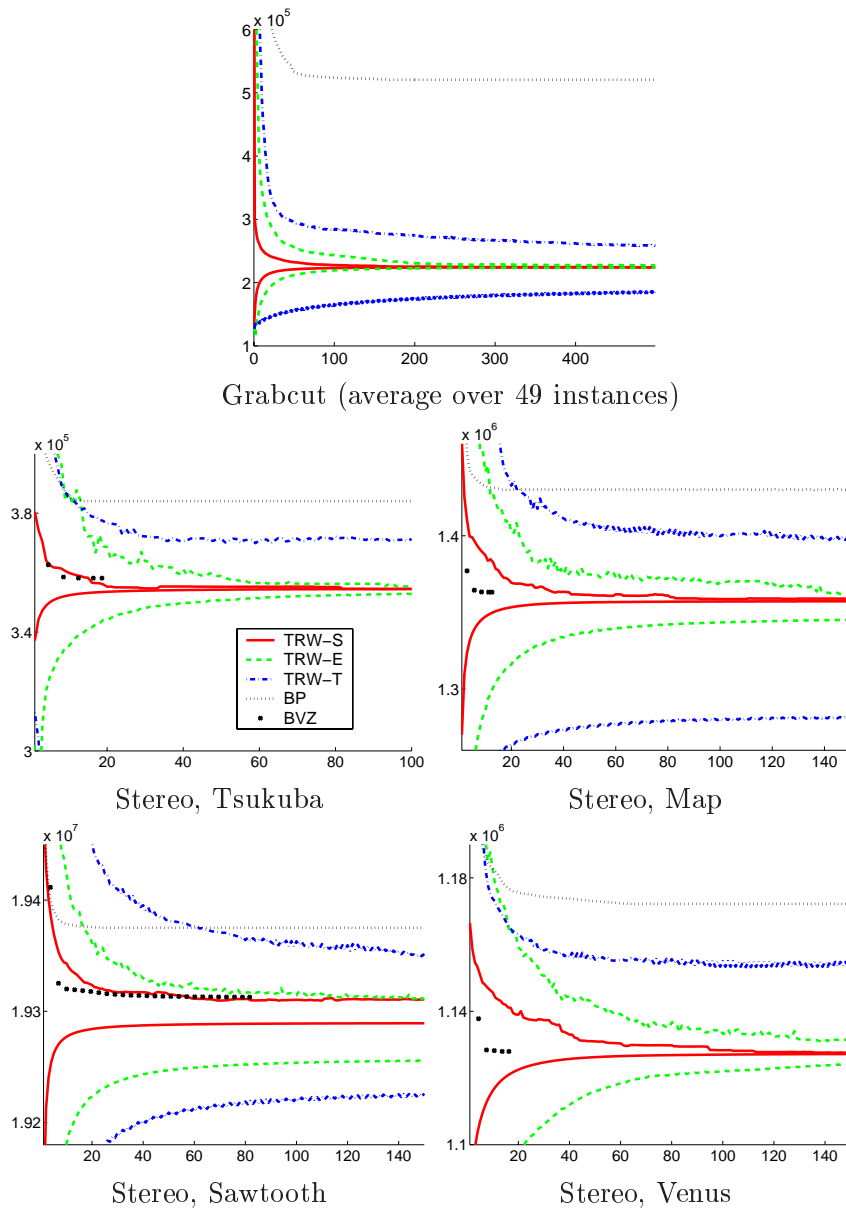


Figure 6: Results on real problems. For description of BVZ, see text.

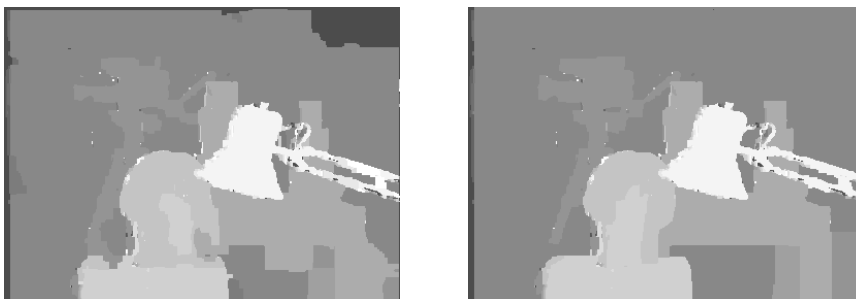


Figure 7: Left: result of BP after convergence. Right: result of TRW-S after 100 iterations. The result of BVZ is visually indistinguishable from the latter.

## 5 Discussion and conclusions

We have developed a new algorithm which can be viewed as a method for direct maximization of objective function  $\Phi_\rho$  subject to the constraint of problem 6. We gave a precise characterization of local maxima of this function with respect to TRW-S algorithm. We showed that the algorithm is guaranteed to have a subsequence converging to such a maximum.

As all tree-reweighted algorithms, our method is not guaranteed to find a *global* maximum. Nevertheless, experimental results suggest that this is not an issue for certain synthetic and real problems. For the stereo matching problem we were able to obtain slightly lower energy than the expansion move algorithm [2] which is considered to be the most accurate energy minimization technique for such problems. TRW-S algorithm outperforms both TRW algorithms in [18] and ordinary max-product BP.

In our experiments we noticed that TRW-S algorithm would always converge to a fixed point of TRW, although such convergence would usually take much longer than achieving a weak tree agreement. However, we have not been able to prove this general convergence. On the other hand, in practice it does not make much sense to run the algorithm after WTA has been achieved since function  $\Phi_\rho$  will not change.

### Acknowledgements

I would like to thank Thomas Minka for helpful discussions.

## Appendix A: Properties of a canonical normal form

**Lemma 5.1.** *Suppose that vector  $\theta$  is in a canonical normal form. Then*

- *Constant  $const_{st}$  in equation 2 is zero:*

$$\min_{j \in \mathcal{X}_s} \{\theta_{s;j} + \theta_{st;jk}\} = 0 \quad \forall k \in \mathcal{X}_t \quad (8)$$

- *For a tree-structured graph, formulas 3 for min-marginals can be rewritten as*

$$\Phi(\theta) = \theta_{const} \quad (9a)$$

$$\Phi_{s;j}(\theta) = \theta_{const} + \theta_{s;j} \quad (9b)$$

$$\Phi_{st;jk}(\theta) = \theta_{const} + \{\theta_{s;j} + \theta_{st;jk} + \theta_{t;k}\} \quad (9c)$$

*Proof.* The fact that constant  $const_{st}$  in equation 2 is zero can be derived as follows. Let us add  $\theta_{t;k}$  to this equation:

$$\min_{j \in \mathcal{X}_s} \{\theta_{s;j} + \theta_{st;jk} + \theta_{t;k}\} = \theta_{t;k} + const_{st} \quad \forall k \in \mathcal{X}_t$$

Now take the minimum over  $k \in \mathcal{X}_t$ . By the definition of a canonical normal form we get zero on the LHS and  $const_{st}$  on the RHS, i.e.  $0 = const_{st}$ . The first property is proved.

Now assume that the graph is a tree. Let  $\mathbf{x}^*$  be an optimal configuration for vector  $\theta$  (then  $E(\mathbf{x}^* | \theta) = \Phi(\theta)$ ). For a fixed node  $s \in \mathcal{V}$  the minimum of  $\theta_s(j)$  among  $j \in \mathcal{X}_s$  is achieved at  $j = x_s^*$  (due to formula 3a). By the definition of a canonical normal form the value of this minimum is zero. Thus,  $\theta_s(x_s^*) = 0$ . Using the same reasoning for edge  $(s, t) \in \mathcal{E}$ , we get that  $\theta_s(x_s^*) + \theta_{st}(x_s^*, x_t^*) + \theta_t(x_t^*) = 0$ , which implies that  $\theta_{st}(x_s^*, x_t^*) = 0$ . Therefore,

$$\Phi(\theta) = \theta_{const} + \sum_{s \in \mathcal{V}} \theta_s(x_s^*) + \sum_{(s,t) \in \mathcal{E}} \theta_{st}(x_s^*, x_t^*) = \theta_{const}$$

Now for node  $s \in \mathcal{V}$  let us plug  $j = x_s^*$  into equation 3a. The LHS is  $\Phi(\theta)$  and the RHS is  $const_s$ . Therefore,  $const_s = \Phi(\theta) = \theta_{const}$ . Similarly we can derive that  $const_{st} = \Phi(\theta) = \theta_{const}$  in equation 3b. The second property is proved.  $\square$

## Appendix B: Reparameterization and messages

As discussed in section 2.2, any message vector defines a reparameterization. We now show that the converse is also true: any reparameterization can be described via messages, up to a constant parameter vector.

**Definition 5.2.** Vector  $\tilde{\theta} \in \mathbb{R}^d$  is called a constant parameter vector if for each node  $s \in \mathcal{V}$ ,  $j \in \mathcal{X}_s$  and for each edge  $(s, t) \in \mathcal{E}$ ,  $(j, k) \in \mathcal{X}_s \times \mathcal{X}_t$  values  $\tilde{\theta}_{s;j}$  and  $\tilde{\theta}_{st;jk}$  are constants independent of  $j, k$ .

**Lemma 5.3.** Suppose that  $\theta^2 \equiv \theta^1$  (i.e.  $\theta^2$  is a reparameterization of  $\theta^1$ ). Then there exists message vector  $M$  such that  $\theta^2 = \theta^1[M] + \tilde{\theta}$  where  $\tilde{\theta}$  is a constant parameter vector such that  $\tilde{\theta} \equiv \mathbf{0}$ .

*Proof.* The following properties will be quite useful in the proof:

$$(P1) \text{ For any vectors } \theta, \theta' \text{ and } M \text{ we have } (\theta + \theta')[M] = \theta[M] + \theta' = \theta + \theta'[M]$$

$$(P2) \text{ For any vectors } \theta, M \text{ and } M' \text{ we have } \theta[M + M'] = \theta[M][M']$$

In addition, we will need the following proposition.

**Proposition 5.4.** For any edge  $(s, t) \in \mathcal{E}$  and variables  $j, j' \in \mathcal{X}_s$ ,  $k, k' \in \mathcal{X}_t$

$$\theta_{st;jk}^1 + \theta_{st;j'k'}^1 - \theta_{st;jk'}^1 - \theta_{st;j'k}^1 = \theta_{st;jk}^2 + \theta_{st;j'k'}^2 - \theta_{st;jk'}^2 - \theta_{st;j'k}^2$$

*Proof.* Let  $\mathbf{x}^{jk}, \mathbf{x}^{jk'}, \mathbf{x}^{j'k}, \mathbf{x}^{j'k'} \in \mathcal{X}$  be four configurations such that they agree on all nodes other than  $s$  and  $t$ , and

$$\begin{aligned} x_s^{jk} &= j, x_t^{jk} = k & x_s^{jk'} &= j, x_t^{jk'} = k' \\ x_s^{j'k} &= j', x_t^{j'k} = k & x_s^{j'k'} &= j', x_t^{j'k'} = k' \end{aligned}$$

We can write

$$E(\mathbf{x}^{jk} | \theta^1) + E(\mathbf{x}^{j'k'} | \theta^1) - E(\mathbf{x}^{j'k} | \theta^1) - E(\mathbf{x}^{jk'} | \theta^1) = \theta_{st;jk}^1 + \theta_{st;j'k'}^1 - \theta_{st;jk'}^1 - \theta_{st;j'k}^1$$

- it is easy to see that all other terms cancel each other. We can write similar expression for vector  $\theta^2$ . Since  $\theta^1 \equiv \theta^2$ , the expressions on the LHS are equal. Therefore, the expressions on the RHS are equal as well, which gives us the desired result.  $\square$

We now proceed with the proof of the lemma. Let  $\theta = \theta^1 - \theta^2$ . As showed in [21], there exists message vector  $M$  such that  $\tilde{\theta} = \theta[M]$  is a fixed point of max-product BP. We have

$$\theta^1[M] = \theta^2 + (\theta^1 - \theta^2)[M] = \theta^2 + \theta[M] = \theta^2 - (-\tilde{\theta})$$

Also,  $\theta^1 \equiv \theta^2$  implies that  $\tilde{\theta} \equiv \theta \equiv \mathbf{0}$ . Thus, we can prove lemma 5.3 by proving that  $\tilde{\theta}$  is a constant parameter vector.

We can convert  $\tilde{\theta}$  to a canonical normal form  $\hat{\theta}$  by adding a constant parameter vector. We will prove that  $\hat{\theta} = \mathbf{0}$ .

Consider an edge  $(s, t) \in \mathcal{E}$ . Since  $\hat{\theta}$  is a reparameterization of a constant function, by proposition 5.4 we have

$$\hat{\theta}_{st;jk} + \hat{\theta}_{st;j'k'} - \hat{\theta}_{st;jk'} - \hat{\theta}_{st;j'k} = 0 \quad (10)$$

for any variables  $j, j' \in \mathcal{X}_s$ ,  $k, k' \in \mathcal{X}_t$ . Condition 8 for a canonical normal form yields the following formulas:

$$\min_{j \in \mathcal{X}_s} \{\hat{\theta}_{s;j} + \hat{\theta}_{st;jk}\} = 0 \quad \forall k \in \mathcal{X}_t \quad (11a)$$

$$\min_{k \in \mathcal{X}_t} \{\hat{\theta}_{t;k} + \hat{\theta}_{st;jk}\} = 0 \quad \forall j \in \mathcal{X}_s \quad (11b)$$

Let us prove that  $\hat{\theta}_{st;jk} \leq 0$  for any  $(j, k) \in \mathcal{X}_s \times \mathcal{X}_t$ . Let  $j'$  and  $k'$  be variables that minimize 11a and 11b, respectively; then  $\hat{\theta}_{s;j'} + \hat{\theta}_{st;j'k} = \hat{\theta}_{t;k'} + \hat{\theta}_{st;jk'} = 0$ . Using equations 10 we get

$$\hat{\theta}_{st;jk} = \hat{\theta}_{st;j'k} + \hat{\theta}_{st;jk'} - \hat{\theta}_{st;j'k'} = -\hat{\theta}_{s;j'} - \hat{\theta}_{t;k'} - \hat{\theta}_{st;j'k'}$$

The expression on the RHS is non-positive by the definition of a canonical normal form.

Now let us prove that  $\hat{\theta}_{st;jk} = 0$  for any  $(j, k) \in \mathcal{X}_s \times \mathcal{X}_t$ . Let  $j^*$  and  $k^*$  be variables that minimize  $\hat{\theta}_{s;j'}$  over  $j'$  and  $\hat{\theta}_{t;k'}$  over  $k'$ , respectively; then  $\hat{\theta}_{s;j^*} = \hat{\theta}_{t;k^*} = 0$ . Using equation 11a for  $k$  and  $k^*$  and equation 11b for  $j$  we conclude that  $\hat{\theta}_{st;j^*k} \geq 0$ ,  $\hat{\theta}_{st;jk^*} \geq 0$  and  $\hat{\theta}_{st;jk^*} \geq 0$ . Therefore,  $\hat{\theta}_{st;j^*k} = \hat{\theta}_{st;j^*k^*} = \hat{\theta}_{st;jk^*} = 0$ . Finally,

$$\hat{\theta}_{st;jk} = \hat{\theta}_{st;j^*k} + \hat{\theta}_{st;jk^*} - \hat{\theta}_{st;j^*k^*} = 0$$

The fact that  $\hat{\theta}_{st} = \mathbf{0}$  for all edges tells us that nodes are independent. We also know that  $E(\mathbf{x} | \hat{\theta})$  is zero for any configuration  $\mathbf{x}$ . By considering configurations which agree on all nodes except node  $s$  we conclude that  $\hat{\theta}_{s;j} = \text{const}_s$  for any  $j \in \mathcal{X}_s$ . This constant must be zero since  $\hat{\theta}$  is in a canonical form.

We have showed that  $\hat{\theta}_s = \mathbf{0}$  for any nodes  $s$  and that  $\hat{\theta}_{st} = \mathbf{0}$  for any edge  $(s, t)$ , i.e. that  $\hat{\theta}_s$  is a constant parameter vector. Lemma 5.3 is proved.  $\square$

## Appendix C: Proof of lemma 3.1

Let us prove the following lemma first.

**Lemma 5.5.** *If  $\theta^1 \equiv \theta^2$  then for any vector  $\tau \in LOCAL(\mathcal{G})$*

$$\langle \theta^1, \tau \rangle = \langle \theta^2, \tau \rangle$$

*Proof.* By lemma 5.3 in Appendix B there exists message vector  $M$  and constant parameter vector  $\tilde{\theta}$  such that  $\theta^2 = \theta^1[M] + \tilde{\theta}$ . We will show that  $\langle \theta^1[M], \tau \rangle = \langle \theta^1, \tau \rangle$  and  $\langle \tilde{\theta}, \tau \rangle = 0$  for any  $\tau \in LOCAL(\mathcal{G})$ .

(1) Since  $\theta^1[M] = \theta^1 + \mathbf{0}[M]$  it suffices to show that  $\langle \mathbf{0}[M], \tau \rangle = 0$ . Moreover, due to property P2 in Appendix B it is enough to prove this for vectors  $M$  with a single non-zero element. Let  $M_{st;k}$  be this element. Then the only non-zero elements of  $\theta = \mathbf{0}[M]$  are  $\theta_{t,k} = M_{st;k}$  and  $\theta_{st;jk} = -M_{st;k}$  for all  $j \in \mathcal{X}_s$ . We can write

$$\langle \mathbf{0}[M], \tau \rangle = \theta_{t,k} \cdot \tau_{t,k} + \sum_j \theta_{st;jk} \cdot \tau_{st;jk} = (\tau_{t,k} - \sum_j \tau_{st;jk}) \cdot M_{st;k} = 0$$

since  $\tau \in LOCAL(\mathcal{G})$ .

(2)  $\tilde{\theta}$  is a constant parameter vector, so  $\tilde{\theta}_{s;j} = c_s$  and  $\tilde{\theta}_{st;jk} = c_{st}$  where  $c_s$  and  $c_{st}$  are constants independent of  $j, k$ . We can write

$$\begin{aligned} \langle \tilde{\theta}, \tau \rangle &= \tilde{\theta}_{const} + \sum_{s \in \mathcal{V}} \sum_j \tilde{\theta}_{s;j} \cdot \tau_{s;j} + \sum_{(s,t) \in \mathcal{E}} \sum_{j,k} \tilde{\theta}_{st;jk} \cdot \tau_{st;jk} = \\ &= \tilde{\theta}_{const} + \sum_{s \in \mathcal{V}} c_s \left( \sum_j \tau_{s;j} \right) + \sum_{(s,t) \in \mathcal{E}} c_{st} \left( \sum_{j,k} \tau_{st;jk} \right) = \tilde{\theta}_{const} + \sum_{s \in \mathcal{V}} c_s + \sum_{(s,t) \in \mathcal{E}} c_{st} \end{aligned}$$

so it is a constant independent of  $\tau$ . Plugging  $\tau = \phi(\mathbf{x})$  for some configuration  $\mathbf{x} \in \mathcal{X}$  we conclude that this constant is zero.  $\square$

Now we can prove lemma 3.1. Let  $\theta^*$  be a vector maximizing problem 5. Clearly, it satisfies constraints of problem 6. Now consider another vector  $\theta \in \mathcal{A}$  such that  $\hat{\theta} = \sum_T \rho^T \theta^T$  is a reparameterization of  $\tilde{\theta}$ . We can write

$$\Phi_\rho(\theta) \leq \max_{\theta \in \mathcal{A}, \sum_T \rho^T \theta^T = \hat{\theta}} \Phi_\rho(\theta) = \min_{\tau \in LOCAL(\mathcal{G})} \langle \hat{\theta}, \tau \rangle = \min_{\tau \in LOCAL(\mathcal{G})} \langle \tilde{\theta}, \tau \rangle = \Phi_\rho(\theta^*)$$

where the first and the third equality follow from theorem 2.1 and the second equality follows from lemma 3.1. Therefore,  $\theta^*$  maximizes problem 6.

## Appendix D: Local maxima of TRW-S algorithm

As we proved, any vector  $\theta$  that maximizes problem 6 must satisfy WTA condition. Here we show the converse is not necessarily true. We give an example of vectors  $\theta$  and  $\tilde{\theta}$  such that they both satisfy WTA condition and  $\sum_T \rho^T \tilde{\theta}^T \equiv \sum_T \rho^T \theta^T$ , but  $\Phi_\rho(\tilde{\theta}) \neq \Phi_\rho(\theta)$ .

The graph with two trees is shown in Figure 8. Vectors  $\theta = \{\theta^1, \theta^2\}$  and  $\tilde{\theta} = \{\tilde{\theta}^1, \tilde{\theta}^2\}$  are given as follows. All single-node potentials are zero:  $\theta_{s;j}^T = \tilde{\theta}_{s;j}^T = 0$ . Other elements of vectors  $\theta^1$  and  $\theta^2$  are given by

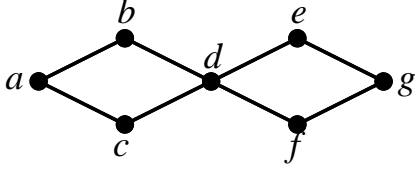
$$\begin{array}{ccccc} \theta_{ab}^1 = \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix} & \theta_{bd}^1 = \begin{bmatrix} 0 & 2 & 2 \\ 2 & 0 & 0 \end{bmatrix} & \theta_{de}^1 = \begin{bmatrix} 2 & 0 \\ 0 & 2 \\ 2 & 0 \end{bmatrix} & \theta_{eg}^1 = \begin{bmatrix} 0 & 2 \\ 2 & 0 \end{bmatrix} & \theta_{const}^1 = 0 \\ \theta_{ac}^2 = \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix} & \theta_{cd}^2 = \begin{bmatrix} 2 & 0 & 0 \\ 0 & 2 & 2 \end{bmatrix} & \theta_{df}^2 = \begin{bmatrix} 2 & 0 \\ 2 & 0 \\ 0 & 2 \end{bmatrix} & \theta_{fg}^2 = \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix} & \theta_{const}^2 = 0 \end{array}$$

and elements of vectors  $\tilde{\theta}^1$  and  $\tilde{\theta}^2$  - by

$$\begin{array}{ccccc} \tilde{\theta}_{ab}^1 = \begin{bmatrix} 3 & 0 \\ 0 & 1 \end{bmatrix} & \tilde{\theta}_{bd}^1 = \begin{bmatrix} 0 & 1 & 1 \\ 3 & 0 & 0 \end{bmatrix} & \tilde{\theta}_{de}^1 = \begin{bmatrix} 0 & 0 \\ 0 & 4 \\ 0 & 0 \end{bmatrix} & \tilde{\theta}_{eg}^1 = \begin{bmatrix} 0 & 4 \\ 0 & 0 \end{bmatrix} & \tilde{\theta}_{const}^1 = 0.5 \\ \tilde{\theta}_{ac}^2 = \begin{bmatrix} 1 & 0 \\ 0 & 3 \end{bmatrix} & \tilde{\theta}_{cd}^2 = \begin{bmatrix} 3 & 0 & 0 \\ 0 & 1 & 1 \end{bmatrix} & \tilde{\theta}_{df}^2 = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 4 \end{bmatrix} & \tilde{\theta}_{fg}^2 = \begin{bmatrix} 4 & 0 \\ 0 & 0 \end{bmatrix} & \tilde{\theta}_{const}^2 = 0.5 \end{array}$$

Condition  $\sum_T \rho^T \tilde{\theta}^T \equiv \sum_T \rho^T \theta^T$  can be checked using the definition of reparameterization. It is easy to see that all tree vectors are in a canonical normal form, and WTA condition is satisfied for both  $\theta$  and  $\tilde{\theta}$ . However, we have  $\Phi_\rho(\theta) = 0$  and  $\Phi_\rho(\tilde{\theta}) = 0.5$ .

This example applies not only to TRW-S, but also to TRW-E and TRW-T algorithms. Indeed, it is not difficult to see that vector  $\theta$  is a fixed point of TRW algorithms.



$$\begin{aligned}\mathcal{X}_a = \mathcal{X}_b = \mathcal{X}_c = \mathcal{X}_e = \mathcal{X}_f = \mathcal{X}_g &= \{1, 2\} \\ \mathcal{X}_d &= \{1, 2, 3\}\end{aligned}$$

Figure 8: Graph for the example in Appendix D. Nodes  $a, b, c, e, f, g$  have two states and node  $d$  has three states. There are two trees with equal probabilities:  $T_1 = (a, b, d, e, g)$  and  $T_2 = (a, c, d, f, g)$ .

## Appendix E: Proof of theorem 3.6

We now show that vectors  $\theta \in \{\theta^{(i)}\}_i$  are bounded (assuming that they are in the canonical normal form). We will do it in three steps: first we show that elements  $\theta_{const}^T$  are bounded, then we show that for each node  $s \in \mathcal{V}^T$  vectors  $\theta_s^T$  are bounded, and finally we show that for each edge  $(s, t) \in \mathcal{E}^T$  vectors  $\theta_{st}^T$  are bounded.

(1) According to theorem 3.4(a) terms  $\Phi(\theta^T)$  are non-decreasing. By lemma 5.1 we have  $\theta_{const}^T = \Phi(\theta^T)$ . This shows that  $\theta_{const}^T$  are bounded from below. The following inequality then implies that they are also bounded from above:

$$\sum_T \rho^T \theta_{const}^T = \sum_T \rho^T \Phi(\theta^T) \leq \Phi(\bar{\theta})$$

(2) Consider node  $s \in \mathcal{V}$  and variable  $j \in \mathcal{X}_s$ . By definition of the canonical normal form, elements  $\theta_{s;j}^T$  are bounded from below. To prove that they are bounded from above, we derive an inequality similar to the previous one.

We use function  $\Phi_{s;j}$  for this purpose. It is a minimum of a finite number of linear functions and, thus, is concave. Applying Jensen's inequality yields

$$\sum_T \rho^T (\theta_{const}^T + \theta_{s;j}^T) = \sum_T \rho^T \Phi_{s;j}(\theta^T) \leq \Phi_{s;j}(\sum_T \rho^T \theta^T) = \Phi_{s;j}(\bar{\theta})$$

Since we already showed that  $\theta_{const}^T$  are bounded, this implies that  $\theta_{s;j}^T$  are bounded from above.

(3) Consider an edge  $(s, t) \in \mathcal{V}$  and variables  $(j, k) \in \mathcal{X}_s \times \mathcal{X}_t$ . By definition of the canonical normal form,  $\theta_{s;j}^T + \theta_{st;jk}^T + \theta_{t;k}^T \geq 0$ . It implies that  $\theta_{st;jk}^T$  is bounded from below since  $\theta_{s;j}^T$  and  $\theta_{t;k}^T$  are bounded. The proof that it is bounded from above is completely analogous to the previous case; we just need to use function  $\Phi_{st;jk}$ .

## References

- [1] Yuri Boykov and Marie-Pierre Jolly. Interactive graph cuts for optimal boundary and region segmentation of objects in n-d images. In *Proc. Int. Conf. Computer Vision*, 2001.
- [2] Yuri Boykov, Olga Veksler, and Ramin Zabih. Fast approximate energy minimization via graph cuts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(11), November 2001.
- [3] Chandra Chekuri, Sanjeev Khanna, Joseph Naor, and Leonid Zosin. Approximation algorithms for the metric labeling problem via a new linear programming formulation. *Symposium on Discrete Algorithms*, pages 109–118, 2000.
- [4] Pedro F. Felzenszwalb and Daniel P. Huttenlocher. Efficient belief propagation for early vision. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2004.
- [5] S. Geman and D. Geman. Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images. *IEEE Trans. Pattern Anal. Machine Intell.*, 6:721–741, 1984.

- [6] D. Greig, B. Porteous, and A. Seheult. Exact maximum a posteriori estimation for binary images. *Journal of the Royal Statistical Society, Series B*, 51(2):271–279, 1989.
- [7] Hiroshi Ishikawa. Exact optimization for Markov Random Fields with convex priors. *IEEE Trans. Pattern Anal. Machine Intell.*, 25(10):1333–1336, October 2003.
- [8] Jon Kleinberg and Eva Tardos. Approximation algorithms for classification problems with pairwise relationships: Metric labeling and Markov Random Fields. *IEEE Symposium on Foundations of Computer Science*, pages 14–23, 1999.
- [9] Vladimir Kolmogorov. Convergent tree-reweighted message passing for energy minimization. In *Artificial Intelligence and Statistics*, January 2005.
- [10] Vladimir Kolmogorov and Ramin Zabih. Multi-camera scene reconstruction via graph cuts. In *Proc. European Conf. Computer Vision*, volume 3, pages 82–96, 2002.
- [11] Vladimir Kolmogorov and Ramin Zabih. What energy functions can be minimized via graph cuts? *IEEE Trans. Pattern Anal. Machine Intell.*, 26(2):147–159, February 2004. An earlier version of this paper appeared in *ECCV*.
- [12] Vivek Kwatra, Arno Schödl, Irfan Essa, Greg Turk, and Aaron Bobick. Graphcut textures: Image and video synthesis using graph cuts. *ACM Transactions on Graphics, SIGGRAPH 2003*, July 2003.
- [13] Judea Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann Publishers Inc., 1988.
- [14] Carsten Rother, Vladimir Kolmogorov, and Andrew Blake. Grabcut - interactive foreground extraction using iterated graph cuts. *ACM Transactions on Graphics, SIGGRAPH*, August 2004.
- [15] Daniel Scharstein and Richard Szeliski. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *Int. J. Computer Vision*, 47:7–42, April 2002.
- [16] J. Sun, N. Zheng, and H. Shum. Stereo matching using belief propagation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(7):787–800, 2003.
- [17] M. F. Tappen and W. T. Freeman. Comparison of graph cuts with belief propagation for stereo, using identical MRF parameters. In *Proc. Int. Conf. Computer Vision*, October 2003.
- [18] M.J. Wainwright, T.S. Jaakkola, and A.S. Willsky. MAP estimation via agreement on (hyper)trees: Message-passing and linear-programming approaches. Technical Report UCB/CSD-03-1269, UC Berkeley CS Division, August 2003.
- [19] M.J. Wainwright, T.S. Jaakkola, and A.S. Willsky. Tree-based reparameterization framework for analysis of sum-product and related algorithms. *IEEE Transactions on Information Theory*, 45(9):1120–1146, May 2003.
- [20] M.J. Wainwright, T.S. Jaakkola, and A.S. Willsky. Tree-reweighted belief propagation and approximate ML estimation by pseudo-moment matching. *Proceedings of the Ninth International Workshop on Artificial Intelligence and Statistics*, January 2003.
- [21] 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*, 14(2):143–166, April 2004.
- [22] Jonathan S. Yedidia, William T. Freeman, and Yair Weiss. Generalized belief propagation. In *Advances in Neural Information Processing Systems*, pages 689–695, 2000.