Accepted to *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 2006.

Convergent Tree-reweighted Message Passing for Energy Minimization

Vladimir Kolmogorov
University College London, UK

Abstract

Algorithms for discrete energy minimization are of fundamental importance in computer vision. In this paper we focus on the recent technique proposed by Wainwright *et al.* [33] - 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 [33]. In addition, on stereo problems with Potts interactions we obtain a lower energy than graph cuts.

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

1 Introduction

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

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

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. θ 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 [8]: 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 [3,9,10,15] and showed a major improvement over previously used simulated annealing [8]. 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 [3,14], image restoration [3], image segmentaion [2], texture synthesis [19]. In fact, for some functions it finds a global minimum.

However, graph cuts can be applied only to a limited class of energy functions [3,10,15]. If a function falls outside this class then one has to use other techniques such as max-product belief propagation (BP) [6,22,27,35]. BP can be applied to any function of the form as in eqn. (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) [28]. Second, BP does not always converge - it often goes into a loop.

Tree-reweighted message passing Recently, Wainwright *et al.* [33] introduced a new algorithm for energy minimization called *max-product tree-reweighted message passing* (TRW). We believe that it may become a serious rival to both graph cuts and BP. As BP, TRW can be applied to any function of the form as in eqn. (1). In addition, in this paper we show that for stereo problems with Potts interactions TRW obtains a lower energy than both 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.* [33] gave two versions of the TRW algorithm which differ in the schedule of passing messages. These algorithms were inspired by the idea of maximizing a concave lower bound on the energy, and 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 [33] 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 [31]: 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 - see section 3.4) then our algorithm reduces to the message-passing algorithm of Wainwright *et al.* [33], 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 [28] 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 [6] 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 [6].

Other related work Tree-reweighted message passing is closely related to a certain *linear programming (LP) relaxation* of the energy function (described in more detail in section 2.3). This relaxation has been widely studied in the literature in different contexts. Schlesinger [24] applied it to energy functions of the form (1) whose pairwise terms encode hard constraints: $\theta_{st}(\cdot,\cdot) \in \{0, +\infty\}$. Koster *et al.* [17] formulated this LP relaxation for arbitrary functions $E(\mathbf{x} \mid \theta)$ (in their terminology the problem is called *partial constraint satisfaction*). Chekuri *et al.* [4] used the formulation for functions with *metric* pairwise terms; they proved certain performance guarantees, e.g. 2-approximation bound for the *Potts* energy. (Their work extended the approach of Kleinberg *et al.* [11]). Wainwright *et al.* [33] studied this LP formulation in the context of TRW algorithm for general energy functions of the form (1). Recently, Komodakis *et al.* [16] showed that graph cuts (or, more precisely, the expansion move method in [3]) has close links with this LP. Thus, the LP relaxation in [4, 17, 24, 33] plays a very important role in the theory of MRF optimization algorithms.

Several authors developed specialized techniques that try to solve this linear program. Koval and Schlesinger [18, 25] showed how to find a subgradient direction assuming that a certain *arc consistency* condition is violated. (A description of their *augmenting DAG* algorithm can be found in [34]). Another algorithm with the same stopping criterion (namely, arc consistency) is *max-sum diffusion*[1]. Note that after some transformations arc consistency can be shown to be equivalent to the WTA condition formulated in this paper. Thus, the augmenting DAG, max-sum diffusion and TRW algorithms have similar stopping criteria. In particular, neither technique is guaranteed to solve the LP relaxation exactly, as our counterexample shows. (Another counterexample due to M. I. Schlesinger is given in [34]).

Outline The paper is organized as follows. In section 2 we introduce our notation and review some results from [33], in particular the lower bound on the energy function via convex combination of trees and the duality result. In section 3 we present our new tree-reweighted algorithm (TRW-S), as well as its analysis. We recommend using a special case of the algorithm, namely TRW-S with *monotonic chains*. A reader who wants to implement this technique can skip sections 2 and 3 (except for the first three paragraphs of section 2) and go directly to section 4. This section gives a summary of the algorithm. Experimental results are described in section 5. Finally, we give conclusions in section 6.

2 Notation and background

In this paper we closely follow the notation used in [33]. 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 $\mathcal{X} = \mathcal{X}_1 \times \mathcal{X}_2 \times \ldots \times \mathcal{X}_n$. Unless noted otherwise, symbols s and t will denote nodes in $\mathcal{V}$, (s,t) is an edge in $\mathcal{E}$, and j and k are values in $\mathcal{X}_s$ and $\mathcal{X}_t$, respectively.

As stated in the previous section, our goal is minimizing the function $E(\mathbf{x} \mid \theta)$ (eqn. (1)) defined by parameter θ. This parameter is specified by constant term θ_{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 θ can be viewed as a vector $\theta = \{\theta_\alpha \mid \alpha \in \mathcal{I}\} \in \mathbb{R}^d$ 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 θ_s to denote a vector of size $|\mathcal{X}_s|$ and θ_{st} to denote a vector of size $|\mathcal{X}_s \times \mathcal{X}_t|$.

It is convenient to express $E(\mathbf{x} \mid \theta)$ as the Euclidean product of two vectors depending on θ and $\mathbf{x}$. Of course,

[1] Max-sum diffusion algorithm was developed indepedently by V. A. Kovalevsky and V. K. Koval in 1975 and by B. Flach in 1998. Neither of the works was published. We learned about this method from report [34].

we cannot write it as $\langle\theta, \mathbf{x}\rangle$ since θ and $\mathbf{x}$ belong to different spaces. However, we can introduce mapping $\boldsymbol{\phi} : \mathcal{X} \to \mathbb{R}^d$ so that

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

Mapping $\boldsymbol{\phi}$ called the *canonical overcomplete representation* [30, 32] consists of the following functions $\phi_\alpha : \mathcal{X} \to \mathbb{R}$:

$$\begin{array}{rcl} \phi_{const}(\mathbf{x}) & = & 1 \\ \phi_{s;j}(\mathbf{x}) & = & [x_s = j] \\ \phi_{st;jk}(\mathbf{x}) & = & [x_s = j, x_t = k] \end{array}$$

where $[\cdot]$ is one if its argument is true, and zero otherwise.

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

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

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 [22]. BP is an algorithm for approximate minimization of energy $E(\mathbf{x}|\bar{\theta})$ as in eqn. (1); it is exact if the graph is a tree[2]. Let us review how BP works. It maintains a *message* $M_{st} = \{M_{st;k} \mid k \in \mathcal{X}_t\}$ for each directed edge $(s \to t) \in \mathcal{E}$, 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 $(s \to t) \in \mathcal{E}$. It consists of updating vector M_{st} as follows:

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

where $const_t$ is a constant independent of k[3]. We will say that a message for directed edge $(s \to t)$ 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: inward (sending messages from leaves to a root) and outward (sending messages from the root to leaves). Note that any node can serve as a root.

[2]From now on, we will use notation $\bar{\theta}$ for the original energy function rather than θ. The reason for this will be clear in the next section.

[3]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.

2.2 Reparameterization

If two parameter vectors θ and $\bar{\theta}$ define the same energy function (i.e. $E(\mathbf{x} \mid \theta) = E(\mathbf{x} \mid \bar{\theta})$ for all $\mathbf{x} \in \mathcal{X}$) then θ is called a *reparameterization* of $\bar{\theta}$ [5, 25, 26, 30, 32] (in [25] this notion was called *equivalent transformations*). 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 ϕ_α. Indeed, any message vector $M = \{M_{st}\}$ defines reparameterization $\theta = \bar{\theta}[M]$ of the original parameter vector $\bar{\theta}$ as follows:

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

We prove in Appendix B that the opposite is also true: any reparameterization θ of vector $\bar{\theta}$ can be expressed via messages, up to a *constant parameter vector*[4].

Reparameterization provides an alternative way of implementing BP algorithm. As discussed in section 2.1, standard BP stores the original parameter vector $\bar{\theta}$ and messages M. Alternatively, we can store the reparameterization $\theta = \bar{\theta}[M]$, in which case the 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 θ_t and θ_{st} for node t and edge (s, t), respectively [32]. It can also be shown [32] that a message for directed edge $(s \to t)$ is valid iff

$$\min_{j\in\mathcal{X}_s} \{\theta_{s;j} + \theta_{st;jk}\} = const_{st} \qquad \forall\, k \in \mathcal{X}_t \tag{2}$$

If this condition holds, then sending a message from s to t does not change θ_{st} and θ_t (or change them by a constant independent of j or k). We say that θ 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 θ in a normal form have a particularly simple interpretation [32] - they correspond to min-marginals (up to a constant):

$$\Phi_{s;j}(\theta) = \theta_{s;j} + const_s \tag{3a}$$

$$\Phi_{st;jk}(\theta) = \{\theta_{s;j} + \theta_{st;jk} + \theta_{t;k}\} + const_{st} \tag{3b}$$

Canonical normal form We will say that vector θ is in a *canonical* normal form if in addition to eqn. (2) it satisfies the following conditions for all nodes and edges:

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

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

[4]We have recently learned that a proof of this fact was given by M. I. Schlesinger in his lectures (NTUU "KPI", Kiev). To our knowledge, this proof was not published.

2.3 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 [4, 17, 24, 33] uses a linear programming (LP) relaxation technique. Let us define the following constraint set:

$$LOCAL(\mathcal{G}) = \left\{ \tau \in \mathbb{R}^d_+ \;\middle|\; \begin{array}{l} \tau_{const} = 1 \\ \sum_{j \in \mathcal{X}_s} \tau_{s;j} = 1 \\ \sum_{j \in \mathcal{X}_s} \tau_{st;jk} = \tau_{t;k} \end{array} \right\}$$

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

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

As shown in [4], this relaxation has several interesting properties. Here we mention just one. Suppose that energy function $E(\mathbf{x} \mid \theta)$ has *Potts* interaction terms and parameter vector θ is non-negative. Then by solving this minimization problem and by using randomized technique proposed in [11] we obtain configuration $\mathbf{x}$ such that the expected value of $E(\mathbf{x} \mid \theta)$ is at most twice the optimal value of the energy.

Unfortunately, general linear programming algorithms such as interior point methods are rather slow, and currently solving problem (4) is not computationally feasible for large vision problems such as stereo. Specialized techniques exploiting the structure of the problem were developed in [18] and in [33]. Both approaches try to solve a *dual* to problem (4). First, a lower bound on the energy $E(\mathbf{x} \mid \theta)$ is formulated (i.e. a function of dual variables which is always smaller or equal than the minimum of E). The goal then becomes to maximize this bound. In this paper we will follow the approach in [33] in which the lower bound is formulated via *convex combination of trees*. This approach is described in the next section.

2.4 Convex combination of trees

First, we need to introduce some notation. Let $\mathcal{T}$ be a collection of trees in graph $\mathcal{G}$ and ρ^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 θ^T that must respect the structure of T. More precisely, the parameter θ^T must belong to the following linear constraint set:

$$\mathcal{A}^T = \{\theta^T \in \mathbb{R}^d \mid \theta^T_\alpha = 0 \;\; \forall\, \alpha \in \mathcal{I} \backslash \mathcal{I}^T\}$$

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 $\Phi_\rho : \mathcal{A} \to \mathbb{R}$ defined as follows:

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

[33] 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:

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

Φ_ρ is a concave function of $\boldsymbol{\theta}$; moreover, the constraints of problem (5) are linear in $\boldsymbol{\theta}$. The following theorem proved in [33] characterizes the dual to problem (5)[5].

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.

We start by modifying maximization problem (5). This problem inspired two algorithms in [33] - 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[6]. 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:

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

The following lemma justifies this formulation.

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

[5] [33] 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.

[6] Note that lemmas 5 and 11 in [33] 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.

Proof. See Appendix C. The proof involves showing that any reparameterization can be expressed via messages. □

As shown in [33], TRW-E and TRW-T algorithms maintain the constraint of problem (6). Unfortunately, they do not guarantee that the objective function Φ_ρ monotonically increases - in our experiments we have observed that sometimes it goes down. In fact, when the algorithms failed to converge the value of $\Phi_\rho(\boldsymbol{\theta})$ often had gone into a loop. Next we design a new algorithm with the property that Φ_ρ never decreases.

Our algorithm is shown in Fig. 1. It iterates between two steps: (a) reparameterizing vectors θ^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 ω, components of vectors θ^T corresponding to ω should give correct min-marginals for trees $T \in \mathcal{T}_\omega$ (eqn. (3)).*

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[7]. However, they update vectors θ^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 θ^T to normal forms by running the ordinary max-product BP. However, this would be very expensive if the 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 Φ_ρ. 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 Φ_ρ.

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

[7] 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 algorithms do not satisfy the min-marginal property.

0. Initialize $\boldsymbol{\theta}$ so that $\boldsymbol{\theta} \in \mathcal{A}$ and $\sum_T \rho^T \theta^T \equiv \bar{\theta}$.
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 $\mathcal{T}_\omega \subseteq \mathcal{T}$ containing ω. If there is more than one tree, then do the following:
   (a) For all trees $T \in \mathcal{T}_\omega$ reparameterize θ^T such that values $\theta^T_{s;j}$ (if $\omega = s$ is a node) or $\theta^T_{s;j} + \theta^T_{st;jk} + \theta^T_{t;k}$ (if $\omega = (s,t)$ is an edge) give correct min-marginals for tree T as in formulae (3).
   (b) "Averaging" operation:
   If $\omega = s$ is a node in $\mathcal{V}$ then
   - Compute $\widetilde{\theta}_s = \frac{1}{\rho_s} \sum_{T \in \mathcal{T}_s} \rho^T \theta^T_s$
   - Set $\theta^T_s := \widetilde{\theta}_s$ for trees $T \in \mathcal{T}_s$

   If $\omega = (s,t)$ is an edge in $\mathcal{E}$ then
   - Compute $\widetilde{\nu}_{st;jk} = \frac{1}{\rho_{st}} \sum_{T \in \mathcal{T}_{st}} \rho^T (\theta^T_{s;j} + \theta^T_{st;jk} + \theta^T_{t;k})$
   - Set θ^T_s, θ^T_{st}, θ^T_t for trees $T \in \mathcal{T}_{st}$ so that
   $$\theta^T_{s;j} + \theta^T_{st;jk} + \theta^T_{t;k} = \widetilde{\nu}_{st;jk}$$
2. Check whether a stopping criterion is satisfied; if yes, terminate, otherwise go to step 1.

Figure 1: **Sequential tree-reweighted algorithm (TRW-S).** *ρ_s is the* node appearance probability*, i.e. the probability that a tree chosen randomly under ρ contains node s. Similarly, ρ_{st} is the edge appearance probability.*

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 every tree T set $\mathbb{S}^T$ is non-empty.

(b) If node s is contained in trees T and T', then for every 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 every 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 $\boldsymbol{\theta} = \{\theta^T\} \in \mathcal{A}$ is said to satisfy the* weak tree agreement *condition if there exists collection $\mathbb{S} \subseteq OPT(\boldsymbol{\theta})$ which is consistent.*

Note that it can be viewed as a generalization of the *tree agreement* condition introduced in [33]: 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 θ^T are in a normal form and $\theta^T_\omega = \theta^{T'}_\omega$ 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 $\bar{\theta}$, i.e. it maintains the property $\sum_T \rho^T \theta^T \equiv \bar{\theta}$.*

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

Next we analyze the behaviour of objective function Φ_ρ during the algorithm. To be specific, we assume for the rest of this section that after reparameterization step 1(a) we have $\min_j \{\theta^T_{s;j}\} = 0$ (if $\omega = s$ is a node) or $\min_{j,k} \{\theta^T_{s;j} + \theta^T_{st;jk} + \theta^T_{t;k}\} = 0$ (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 Φ and Φ_ρ do not decrease.*

(b) If vector $\boldsymbol{\theta}$ does not satisfy WTA condition then after a finite number of iterations function Φ_ρ will increase.

(c) If vector $\boldsymbol{\theta}$ 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 Φ_ρ 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 $\boldsymbol{\theta}$ maximizes problem (6) then $\boldsymbol{\theta}$ 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 $\{\boldsymbol{\theta}^{(i)}\}_i$ be an infinite sequence of vectors obtained by applying step 1 of TRW-S algorithm. Let $\{\widetilde{\boldsymbol{\theta}}^{(i)}\}_i$ be the reparameterization of this sequence obtained by reparameterizing vectors $\boldsymbol{\theta}^{(i)}$ into the canonical normal form. Then there is a subsequence $\{\widetilde{\boldsymbol{\theta}}^{(i(m))}\}_m$ such that*

(a) It converges to some vector $\boldsymbol{\theta}^* \in \mathcal{A}$.

(b) Sequence $\{\Phi_\rho(\widetilde{\boldsymbol{\theta}}^{(i)})\}_i$ converges to $\Phi_\rho(\boldsymbol{\theta}^*)$.

(c) Vector $\boldsymbol{\theta}^*$ satisfies WTA condition.

Note that since the algorithm does not specify exactly how we reparameterize vectors $\boldsymbol{\theta}^{(i)}$, we cannot claim that sequence $\{\boldsymbol{\theta}^{(i)}\}_i$ always contains a converging subsequence. As a counterexample, we could choose $\{\boldsymbol{\theta}^{(i)}\}_i$ so that it is not bounded. Reparameterization $\{\widetilde{\boldsymbol{\theta}}^{(i)}\}_i$ 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. The most important property of the algorithm, namely theorem 3.4(a) (non-decreasing lower bound) is proven in the beginning of the next section. This proof is instrumental for understanding the structure of the algorithm. Other parts of the proof are more technical; the reader may want to skip them and go to section 3.4.

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 the averaging step 1(b). The following lemma shows the effect of the averaging operation on functions Φ.

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

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

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

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

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

$$\begin{aligned} \Phi_{s;j}(\widetilde{\theta}^T) &= E(\mathbf{x}^j \,|\, \widetilde{\theta}^T) = E(\mathbf{x}^j \,|\, \theta^T) - \theta^T_{s;j} + \widetilde{\theta}^T_{s;j} = \\ &= \Phi_{s;j}(\theta^T) - \theta^T_{s;j} + \widetilde{\theta}^T_{s;j} = \Phi(\theta^T) + \widetilde{\theta}^T_{s;j} \end{aligned}$$

□

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

$$\Phi(\widetilde{\theta}^T) = \min_{j \in \mathcal{X}_s} \Phi_{s;j}(\widetilde{\theta}^T) = \Phi(\theta^T) + \min_{j \in \mathcal{X}_s} \{\widetilde{\theta}^T_{s;j}\} \qquad (7)$$

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

Let us prove part (c). Suppose that $\boldsymbol{\theta}$ satisfies WTA with collection $\mathbb{S} = \{\mathbb{S}^T\}$. We need to prove that $\mathbb{S} \subseteq \mathrm{OPT}(\widetilde{\boldsymbol{\theta}})$. We can ignore trees T' that do not contain node s since vector $\theta^{T'}$ and set $\mathrm{OPT}(\theta^{T'})$ 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 every tree $T' \in \mathcal{T}_s$

there exists a configuration $\mathbf{x}'$ with $x'_s = j$ that is optimal for $\theta^{T'}$. This implies that $\theta^{T'}_{s;j} = 0$ for every $T' \in \mathcal{T}_s$. Therefore, we have $\widetilde{\theta}^T_{s;j} = 0$.

One consequence of this fact is that the minimum in eqn. (7) is zero, so $\Phi(\widetilde{\theta}^T) = \Phi(\theta^T)$. Also, it means that $E(\mathbf{x} \mid \widetilde{\theta}) = \Phi(\widetilde{\theta}^T)$, i.e. $\mathbf{x}$ is an optimal configuration for vector $\widetilde{\theta}^T$: $\mathbf{x} \in \mathrm{OPT}^T(\widetilde{\theta}^T)$. 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 $\boldsymbol{\theta}$ to vector $\widetilde{\boldsymbol{\theta}}$ and $\Phi_\rho(\boldsymbol{\theta}) = \Phi_\rho(\widetilde{\boldsymbol{\theta}})$. Then $OPT(\widetilde{\boldsymbol{\theta}})$ does not acquire new configurations, i.e. $OPT(\widetilde{\boldsymbol{\theta}}) \subseteq OPT(\boldsymbol{\theta})$. If in addition $OPT(\boldsymbol{\theta})$ is not consistent then it will shrink, i.e. the inclusion $OPT(\widetilde{\boldsymbol{\theta}}) \subset OPT(\boldsymbol{\theta})$ is strict.*

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

Now suppose that $OPT(\boldsymbol{\theta})$ is not consistent and a complete pass of step 1 does not change the value of function Φ_ρ. It means that in the beginning of step 1 the consistency condition for collection $\mathrm{OPT}(\boldsymbol{\theta})$ is violated for some element (either a node or an edge). Consider the moment when the algorithm reaches this element. If collection $\mathrm{OPT}(\boldsymbol{\theta})$ has already shrunk by that time, then we do not have to prove anything - the lemma holds. Let us assume that $\mathrm{OPT}(\boldsymbol{\theta})$ 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 θ^T (so $\theta^T_{s;j} = 0$) but there exists no configuration $\mathbf{x}'$ with $x'_s = j$ which is optimal for $\theta^{T'}$ (so $\theta^{T'}_{s;j} > 0$). This implies that $\widetilde{\theta}^T_s > 0$, therefore configuration $\mathbf{x}$ is no longer optimal for $\widetilde{\theta}^T$. Thus, collection $\mathrm{OPT}(\widetilde{\boldsymbol{\theta}})$ has shrunk. The lemma is proved. □

We now proceed with the proof of theorem 3.4(b). Let $\boldsymbol{\theta}^0, \boldsymbol{\theta}^1, \boldsymbol{\theta}^2, \ldots$ 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 Φ_ρ stays the same after any number of steps: $\Phi_\rho(\boldsymbol{\theta}^0) = \Phi_\rho(\boldsymbol{\theta}^1) = \Phi_\rho(\boldsymbol{\theta}^2) = \ldots$. Let us show that $\boldsymbol{\theta}$ satisfies WTA condition.

By lemma 3.8 we have $\mathrm{OPT}(\boldsymbol{\theta}) \supset \mathrm{OPT}(\boldsymbol{\theta}^1) \supset \mathrm{OPT}(\boldsymbol{\theta}^2) \supset \ldots$. Since $\mathrm{OPT}(\boldsymbol{\theta})$ is finite it cannot shrink indefinitely, therefore after a finite number of iterations (let us say, n) $\mathrm{OPT}(\boldsymbol{\theta}^n)$ will become consistent. We have $\mathrm{OPT}(\boldsymbol{\theta}^n) \subseteq \mathrm{OPT}(\boldsymbol{\theta})$ 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 $\widetilde{\boldsymbol{\theta}}^{(i)} = \boldsymbol{\theta}^{(i)}$. Appendix E proves that the 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 Φ_ρ is continuous. We now prove that for every 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 $\widetilde{\boldsymbol{\theta}}^*$ such that $\Phi_\rho(\widetilde{\boldsymbol{\theta}}^*) > \Phi_\rho(\boldsymbol{\theta}^*)$.

Let $\pi : \mathcal{A} \to \mathcal{A}$ 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 $\mathcal{A} \to \mathcal{A}$, therefore π is continuous as well. Thus,

$$\{\pi(\boldsymbol{\theta}^{(i(m))})\} \overset{m\to\infty}{\longrightarrow} \pi(\boldsymbol{\theta}^*) = \widetilde{\boldsymbol{\theta}}^*$$

Function Φ_ρ is also continuous, so

$$\{\Phi_\rho(\pi(\boldsymbol{\theta}^{(i(m))}))\} \overset{m\to\infty}{\longrightarrow} \Phi_\rho(\widetilde{\boldsymbol{\theta}}^*)$$

Thus, there exists index m such that $\Phi_\rho(\pi(\boldsymbol{\theta}^{(i(m))})) \geq \Phi_\rho(\boldsymbol{\theta}^*) + \epsilon$ where $\epsilon = \frac{1}{2}(\Phi_\rho(\widetilde{\boldsymbol{\theta}}^*) - \Phi_\rho(\boldsymbol{\theta}^*)) > 0$. 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 every 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 θ^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 inward pass of the ordinary max-product BP is needed for trees $T \in \mathcal{T}_s$ - sending messages from leaves to node s which we treat as a root[8]. However, this would make the algorithm very inefficient if the trees are large. Fortunately, a complete inward 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[9].

[8] Note that the outward pass (sending messages from the root to leaves) is not needed. It would convert vectors θ^T to normal forms but would not change vectors θ^T_s.

[9] Recall that our algorithm is message-free. As discussed in section 2.2, the phrase "message is valid for directed edge $(s \to t)$ in tree T" means that eqn. (2) holds (or that sending a message from node s to node t would not modify $\theta^T_{st;jk}$ and $\theta^T_{t;k}$ except for a constant independent of j or k).

0. Initialize $\boldsymbol{\theta}$ so that $\boldsymbol{\theta} \in \mathcal{A}$ 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 θ_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.**

Therefore, we can "reuse" some of the messages passed in previous steps, i.e. not pass them again[10].

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 *ω.*

Proof. Consider edge $(s \to t)$ oriented towards ω in tree T. The averaging operation can affect only the endpoint vector θ_t^T. However, condition (2) of a valid message involves vectors θ_s^T and θ_{st}^T but not θ_t^T. □

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 $(u \to v)$ in trees $T \in \mathcal{T}_{uv}$ are valid. This property also holds for node $u = s$ in the end of step 1(b).

(b) For each edge $(u,v) \in \mathcal{E}$ with $i(s) < i(u) < i(v)$ messages $(v \to u)$ 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 $(s \to t)$ in this tree becomes valid. □

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 ω. Therefore, passing messages from leaves to ω would not change parameters θ^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[11]. First, it can be seen that the algorithm maintains 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 θ^T can be stored using messages $M_{st} = \{M_{st;k} \mid k \in \mathcal{X}_t\}$ for directed edges $(s \to t) \in \mathcal{E}$

[10] The idea of reusing messages in junction trees was used in the context of iterative proportional fitting [29] and Rao-Blackwellized sampling [21].

[11] 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.

0. Set all messages to zero.

1. Set $E_{bound} = \bar{\theta}_{const}$.
   For nodes $s \in \mathcal{V}$ do the following operations in the order of increasing $i(s)$:

   - Compute $\widehat{\theta}_s = \bar{\theta}_s + \sum_{(u,s)\in\mathcal{E}} M_{us}$. Normalize vector $\widehat{\theta}_s$ as follows:

     $$\delta := \min_j \widehat{\theta}_{s;j} \qquad \widehat{\theta}_{s;j} := \widehat{\theta}_{s;j} - \delta \qquad E_{bound} := E_{bound} + \delta$$

   - For every edge $(s,t) \in \mathcal{E}$ with $i(s) < i(t)$ update and normalize message M_{st} as follows:

     $$M_{st;k} := \min_j \{(\gamma_{st}\widehat{\theta}_{s;j} - M_{ts;j}) + \bar{\theta}_{st;jk}\}$$

     $$\delta := \min_k M_{st;k} \quad M_{st;k} := M_{st;k} - \delta \quad E_{bound} := E_{bound} + \delta$$

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.** *For a description of ordering $i(\cdot)$ and coefficients γ_{st}, see section 4. In the end of step 1 value E_{bound} gives a lower bound on the energy $E(\mathbf{x} \mid \bar{\theta})$. This value cannot decrease with time.*

according to the following formulae:[12]

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

The resulting algorithm is shown in Fig. 3. (We introduced notation $\gamma_{st} = \rho_{st}/\rho_s$ for directed edge $(s \to t) \in \mathcal{E}$.)

4 Summary of TRW-S algorithm

In the previous section we described several versions of TRW-S algorithm. For a practical implementation we recommend using the technique in Fig. 3. We now summarize various algorithmic details.

The input to the algorithm is an energy function specified by parameter vector $\bar{\theta}$. The method works by passing messages; for each directed edge $(s \to t) \in \mathcal{E}$ there is message M_{st} which is a vector with $|\mathcal{X}_t|$ components. Before running the algorithm we need to make the following choices:

- Select ordering of nodes $i(\cdot)$ (i.e. mapping of nodes in $\mathcal{V}$ onto the set $\{1, 2, \ldots, |\mathcal{V}|\}$).

- Select chains $T \in \mathcal{T}$ which are monotonic with respect to $i(\cdot)$ (see definition 3.10). Each edge must be covered by at least one chain.

- Choose probability distribution ρ over chains $T \in \mathcal{T}$ such that $\rho^T > 0$, $\sum_T \rho^T = 1$.

These choices define coefficients γ_{st} in Fig. 3 in the following way: $\gamma_{st} = \rho_{st}/\rho_s$ where ρ_{st} and ρ_s are edge and node appearance probabilities, respectively. In other words, γ_{st} is the probability that a tree chosen randomly under ρ contains edge (s,t) *given* that it contains s.

An important property of our algorithm is that it requires half as much memory compared to traditional BP. Indeed, the latter needs to store messages in both directions 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 $\gamma_{st} = 1$. The fact that memory requirements of BP can be reduced by half was first observed in [6]. 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 [6]. Indeed, the schedule used in [6] 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.

Note that for many important choices of terms $\bar{\theta}_{st}$ message update in step 1 can be done very efficiently in time $|\mathcal{X}_t|$ using distance transforms [6]. Then the complexity of one pass of our algorithm is $O(|\mathcal{E}| \cdot K)$ where $K = \max_s |\mathcal{X}_s|$.

We conclude this section with the discussion of various implementational details.

Choice of node ordering and monotonic chains Automatic selection of node ordering for an arbitrary graph is an interesting open question, which is not addressed in this paper. Intuitively, good ordering should allow long monotonic chains. We tested two types of graphs: 2D grids with 4 or 8 neighborhood system and complete graphs. We used a natural row-major order for the former. Note that for complete graphs all orderings are equivalent.

Given an ordering, we constructed monotonic chains in a greedy manner as follows. 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.

Although this method can produce different sets of trees depending on what chains we choose, the behaviour of TRW-S algorithm is specified uniquely (assuming that the order of nodes is fixed). Indeed, the algorithm depends only on coefficients $\gamma_{st} = \rho_{st}/\rho_s$ for edges $(s \to t)$. It can

[12] Note that messages M_{st} that we use here are slightly different from messages $M_{st}^{[13]}$ used in [13, 33]. The relationship between the two is as follows: $M_{st} = \rho_{st} M_{st}^{[13]}$. We decided to scale the messages because it simplified some of the equations.

be seen that the number of trees containing node s is

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

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

Stopping criterion A conservative way for checking whether the WTA condition has been achieved follows from the definition in section 3.1: keep adding minimum configurations to the set $\mathbb{S}$ until either $\mathbb{S}$ becomes consistent or no more configurations can be added. This, however, may be expensive. As a practical alternative, we suggest the following heuristic criterion inspired by theorem 3.4(b): we stop if the value of the lower bound E_{bound} has not increased (within some precision) during, say, the last 10 iterations. It is worth noting that even if WTA has been achieved and the lower bound does not change anymore, the messages may still keep changing as well as configuration $\mathbf{x}$ computed from the messages.

One could also imagine other stopping criteria, e.g. fixing the number of iterations. Different criteria will lead to different tradeoffs between speed and accuracy.

Choosing solution An important question is how to construct solution $\mathbf{x}$ given reparameterization $\widehat{\theta} = \sum_T \rho^T \theta^T$. A possible approach is to choose label x_s for node s that minimizes $\widehat{\theta}_s(x_s)$. However, it is often the case that the minimum is not unique within the precision of floating point numbers. This is not suprising. Indeed, if all nodes have unique minimum then it means that we found the optimal solution, as shown in [33]. 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 $\widehat{\theta}_s(x_s) + \sum_{i(u)<i(s)} \widehat{\theta}_{us}(x_u, x_s)$ 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 may still be assigned essentially at random (more precisely, the solution is determined by numerical errors)[13].

5 Experimental results

We have compared four algorithms: ordinary max-product BP and three tree-reweighted algorithms (TRW-E, TRW-T, TRW-S). Trees and their probabilities have been chosen as described in the previous section.

For TRW-E and TRW-T algorithms we also used damping parameter $\gamma \in (0,1]$; as reported in [33], the algorithms converge if sufficiently damped. For each problem described below we chose γ 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.

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.

5.1 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 λ_{st} was generated as $|\mathcal{N}(0,\sigma^2)|$ for attractive potentials and as $\mathcal{N}(0,\sigma^2)$ for mixed potentials. Parameter σ 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.

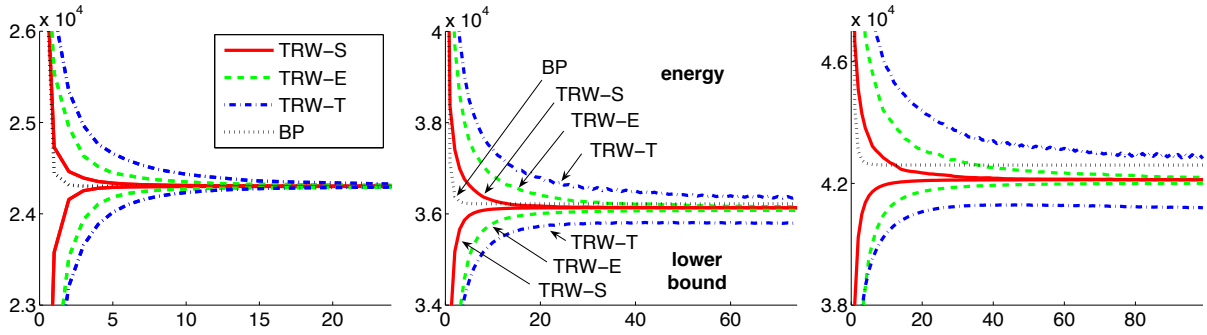
As a measure of performance we used two quantities: the value of the lower bound and the energy of current solution (except for the BP algorithm where we can determine only one of these quantities). Note that the former is a lower bound on the optimal value of the energy, and the latter is an upper bound. 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.

Note that for functions of binary variables TRW computes that same solution as maxflow algorithm [12]. In particular, for attractive potentials TRW is guaranteed to find a global minimum, while for mixed potentials it finds *part* of an optimal solution. It should also be noted that in both cases maxflow would be significantly faster. The goal of this section is to compare different message passing techniques.

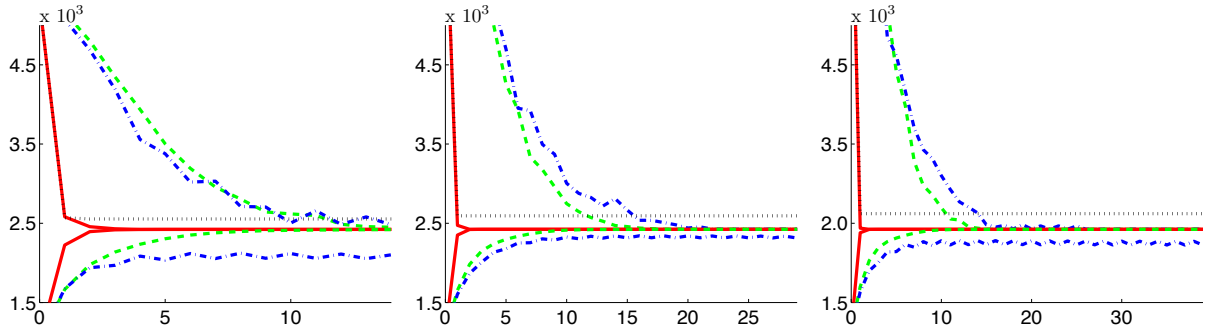
Attractive potentials The results for attractive potentials are shown in Fig. 4(a,b). Note that in this case the global minimum can be found in polynomial time using the maxflow algorithm. In all cases TRW-S outperforms TRW-E and TRW-T. It also outperforms BP since it converges to the global minimum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

Mixed potentials The results for this case are shown in Fig. 4(c,d). The situation is very different from the case of attractive potentials. In all cases BP outperforms TRW algorithms. We believe that this indicates that LP relaxation (4) is not tight in this case.

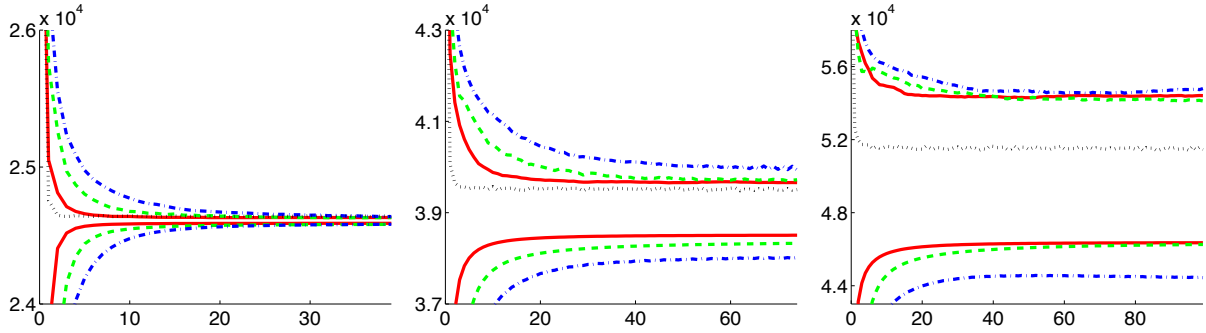
[13] This technique can be motivated by the following observation: if a reparameterization satisfies WTA condition and the set of nodes with multiple minima consists of disjoint chains which are monotonic with respect to the ordering used, then the procedure will find a global minimum (see [20]).



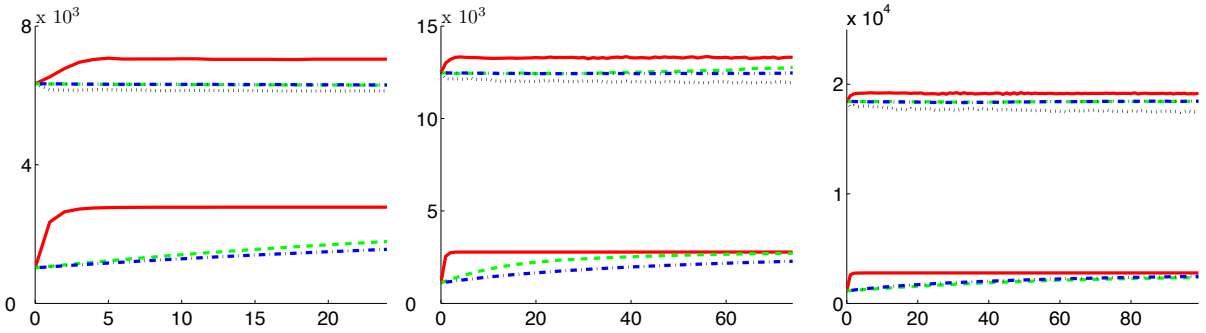
(a) Grid graph 30×30, attractive potentials



(b) Complete graph with 50 nodes, attractive potentials



(c) Grid graph 30×30, mixed potentials



(d) Complete graph with 50 nodes, mixed potentials

Figure 4: **Synthetic problems. Horizontal axis: number of iterations. Vertical axis, upper curves: average value of the energy $E(\mathbf{x} \mid \bar{\theta})$. Vertical axis, lower curves for TRW algorithms: average value of $\Phi_\rho(\boldsymbol{\theta})$. Columns 1-3: different strengths of the interaction potential ($\sigma = \frac{1}{\sqrt{d}}, \frac{2}{\sqrt{d}}, \frac{3}{\sqrt{d}}$).**

5.2 Real problems

Binary segmentation First we tested the algorithms on the energy function arising in the method of Boykov *et al.* [2]. 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 (back-

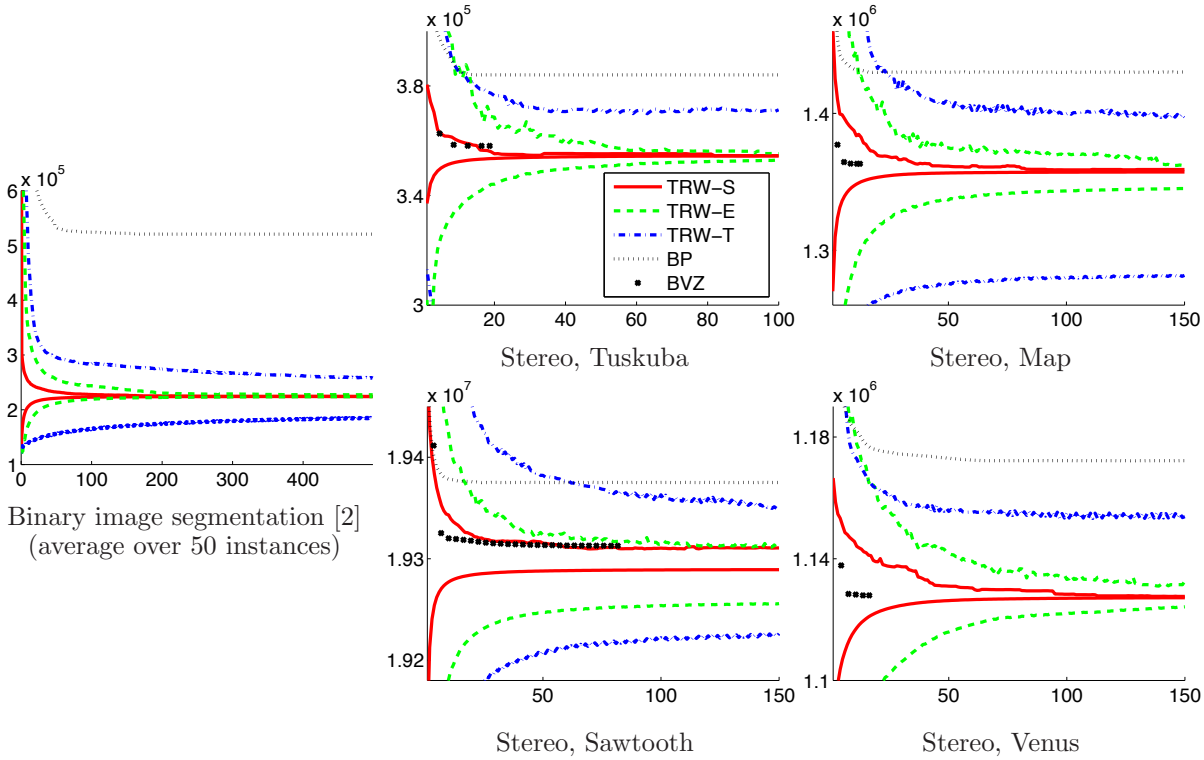


Figure 5: **Results on real problems. For description of BVZ, see text.**

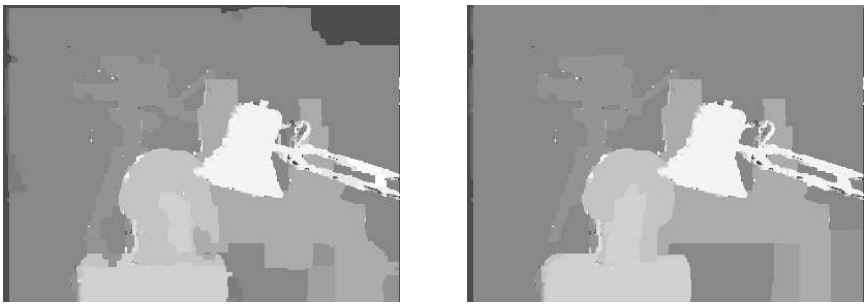


Figure 6: **Left: result of BP after convergence. Right: result of TRW-S after 100 iterations. The result of BVZ is visually indistinguishable from the latter.**

ground) constraints may exist only at the grid boundary; they correspond to a rectangle dragged around the object.

We used 49 different instances with the average size 1.07×10^5 nodes. Energy plots are shown in Fig. 5. The average running time for TRW-S algorithm was 0.067 secs per iteration on Pentium IV 2.8 GHz processor. For comparison, maxflow algorithm in [1] took 0.114 secs (on average). This shows that for this problem maxflow significantly outperforms message passing algorithms.

Stereo matching We have tested the algorithms on the energy function arising in the stereo matching problem with Potts interactions [3]. 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 [23] - Tsukuba, Map, Sawtooth and Venus. Fig. 5 shows the energy plots, and Fig. 6 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 [3], which is referred to as BVZ. (Recall that it is based on the maxflow algorithm). 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[14]. Each point in Fig. 5 corresponds to one iteration of BVZ algorithm, and the last point indicates that BVZ has converged.

[14] 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, in our implementation one iteration of BP is slightly faster and needs less memory than TRW-S since we use integers for BP and double precision numbers for TRW-S. We also tested informally single precision numbers for TRW-S; then the value of the lower bound becomes very inaccurate, but the value of energy gets worse only slightly.

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 (W x H x D)	TRW-S time	TRW-S accuracy	BVZ 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.

6 Discussion and conclusions

We have developed a new algorithm which can be viewed as a method for direct maximization of objective function Φ_ρ 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 [3] which is considered to be the most accurate energy minimization technique for such problems. On real vision problems that we have tested TRW-S outperforms both TRW algorithms in [33] and ordinary max-product BP.

It should be noted that TRW-S (and TRW algorithms in general) have some limitations. First, they do not work well when LP relaxation (4) is not tight. Second, TRW algorithms are slower than maxflow-based techniques (in cases when such techniques *can* be applied). However, one advantage of message-passing techniques over maxflow-based techniques is that the former are easily parallelizable; one could imagine GPU or hardware implementations. We also believe that TRW-S could be a winner for problems which, on one hand, are sufficiently "easy" in the sense that LP relaxation (4) is tight, but on the other, do not have a "structured" set of labels so that maxflow-based techniques cannot be applied. It is interesting, for example, to test the problem of super resolution [7].

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, achieving convergence may not be necessary since running the algorithm after obtaining WTA will not improve the lower bound on the energy.

Acknowledgments I would like to thank Thomas Minka for helpful discussions, anonymous reviewers for suggestions that helped to improve the presentation of the paper, and Philip Torr for careful proofreading of the manuscript. This work was mainly done while the author was with Microsoft Research, Cambridge.

Appendix A: Properties of a canonical normal form

Lemma 6.1. *Suppose that vector θ is in a canonical normal form. Then*

- *Constant $const_{st}$ in eqn. (2) is zero:*

$$\min_{j \in \mathcal{X}_s} \{\theta_{s;j} + \theta_{st;jk}\} = 0 \qquad \forall\, k \in \mathcal{X}_t \tag{8}$$

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

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

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

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

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

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

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 θ (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 eqn. (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 eqn. (3b). The second property is proved. □

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 6.2. *Vector $\widetilde{\theta} \in \mathbb{R}^d$ 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 $\widetilde{\theta}_{s;j}$ and $\widetilde{\theta}_{st;jk}$ are constants independent of j,k.*

Lemma 6.3. *Suppose that $\theta^2 \equiv \theta^1$ (i.e. θ^2 is a reparameterization of θ^1). Then there exists message vector M and constant parameter vector $\widetilde{\theta}$ such that $\theta^2 = \theta^1[M] + \widetilde{\theta}$ and $\widetilde{\theta} \equiv \mathbf{0}$.*

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

($\mathcal{P}1$) For any vectors θ, θ' and M there holds $(\theta+\theta')[M] = \theta[M] + \theta' = \theta + \theta'[M]$.

($\mathcal{P}2$) For any vectors θ, M and M' there holds $\theta[M+M'] = \theta[M][M']$.

In addition, we will need the following proposition.

Proposition 6.4. *For any edge $(s,t) \in \mathcal{E}$ and variables $j,j' \in \mathcal{X}_s$, $k,k' \in \mathcal{X}_t$*

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

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

$$x_s^{jk} = j\ ,\ x_t^{jk} = k \qquad x_s^{jk'} = j\ ,\ x_t^{jk'} = k'$$

$$x_s^{j'k} = j'\ ,\ x_t^{j'k} = k \qquad x_s^{j'k'} = j'\ ,\ x_t^{j'k'} = k'$$

We can write

$$E(\mathbf{x}^{jk} \mid \theta^1) + E(\mathbf{x}^{j'k'} \mid \theta^1) - E(\mathbf{x}^{jk'} \mid \theta^1) - E(\mathbf{x}^{j'k} \mid \theta^1) =$$

$$= \theta^1_{st;jk} + \theta^1_{st;j'k'} - \theta^1_{st;jk'} - \theta^1_{st;j'k}$$

- it is easy to see that all other terms cancel each other. We can write similar expression for vector θ^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. □

We now proceed with the proof of the lemma. Let $\theta = \theta^1 - \theta^2$. As showed in [32], there exists message vector M such that $\widetilde{\theta} = \theta[M]$ is a fixed point of max-product BP. We have

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

Also, $\theta^1 \equiv \theta^2$ implies that $\widetilde{\theta} \equiv \theta \equiv \mathbf{0}$. Thus, we can prove lemma 6.3 by proving that $\widetilde{\theta}$ is a constant parameter vector.

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

Consider an edge $(s,t) \in \mathcal{E}$. Since $\widehat{\theta}$ is a reparameterization of a constant function, by proposition 6.4 we have

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

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

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

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

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

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

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

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

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

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

We have showed that $\widehat{\theta}_s = \mathbf{0}$ for all nodes s and that $\widehat{\theta}_{st} = \mathbf{0}$ for all edges (s,t), i.e. that $\widehat{\theta}_s$ is a constant parameter vector. Lemma 6.3 is proved. □

Appendix C: Proof of lemma 3.1

Let us prove the following lemma first.

Lemma 6.5. *If $\theta^1 \equiv \theta^2$ then for any vector $\tau \in LOCAL(\mathcal{G})$ there holds $\langle \theta^1, \tau \rangle = \langle \theta^2, \tau \rangle$.*

Proof. By lemma 6.3 in Appendix B there exists message vector M and constant parameter vector $\widetilde{\theta}$ such that $\theta^2 = \theta^1[M] + \widetilde{\theta}$. We will show that $\langle \theta^1[M], \tau \rangle = \langle \theta^1, \tau \rangle$ and $\left\langle \widetilde{\theta}, \tau \right\rangle = 0$ 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 $\mathcal{P}2$ 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

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

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

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

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

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

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

$$\begin{aligned}\Phi_\rho(\boldsymbol{\theta}) &\le \max_{\boldsymbol{\theta}\in\mathcal{A}, \sum_T \rho^T \theta^T = \widehat{\theta}} \Phi_\rho(\boldsymbol{\theta}) = \\ &= \min_{\tau \in LOCAL(\mathcal{G})} \Big\langle \widehat{\theta}, \tau \Big\rangle = \min_{\tau \in LOCAL(\mathcal{G})} \big\langle \bar{\theta}, \tau \big\rangle = \Phi_\rho(\boldsymbol{\theta}^*)\end{aligned}$$

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

Appendix D: Local maxima of TRW-S algorithm

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

The graph with two trees is shown in Figure 7. Vectors $\boldsymbol{\theta} = \{\theta^1, \theta^2\}$ and $\widetilde{\boldsymbol{\theta}} = \{\widetilde{\theta}^1, \widetilde{\theta}^2\}$ are given as follows. All single-node potentials are zero: $\theta^T_{s;j} = \widetilde{\theta}^T_{s;j} = 0$. The constant terms are $\theta^T_{const} = 0$, $\widetilde{\theta}^T_{const} = 0.5$. Other elements of vectors θ^1 and θ^2 are given by

$$\theta^1_{ab} = \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix} \quad \theta^1_{bd} = \begin{bmatrix} 0 & 2 & 2 \\ 2 & 0 & 0 \end{bmatrix} \quad \theta^1_{de} = \begin{bmatrix} 2 & 0 \\ 0 & 2 \\ 2 & 0 \end{bmatrix} \quad \theta^1_{eg} = \begin{bmatrix} 0 & 2 \\ 2 & 0 \end{bmatrix}$$

$$\theta^2_{ac} = \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix} \quad \theta^2_{cd} = \begin{bmatrix} 2 & 0 & 0 \\ 0 & 2 & 2 \end{bmatrix} \quad \theta^2_{df} = \begin{bmatrix} 2 & 0 \\ 2 & 0 \\ 0 & 2 \end{bmatrix} \quad \theta^2_{fg} = \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix}$$

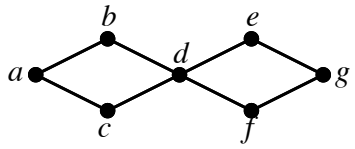
and elements of vectors $\widetilde{\theta}^1$ and $\widetilde{\theta}^2$ - by

$$\widetilde{\theta}^1_{ab} = \begin{bmatrix} 3 & 0 \\ 0 & 1 \end{bmatrix} \quad \widetilde{\theta}^1_{bd} = \begin{bmatrix} 0 & 1 & 1 \\ 3 & 0 & 0 \end{bmatrix} \quad \widetilde{\theta}^1_{de} = \begin{bmatrix} 0 & 0 \\ 0 & 4 \\ 0 & 0 \end{bmatrix} \quad \widetilde{\theta}^1_{eg} = \begin{bmatrix} 0 & 4 \\ 0 & 0 \end{bmatrix}$$

$$\widetilde{\theta}^2_{ac} = \begin{bmatrix} 1 & 0 \\ 0 & 3 \end{bmatrix} \quad \widetilde{\theta}^2_{cd} = \begin{bmatrix} 3 & 0 & 0 \\ 0 & 1 & 1 \end{bmatrix} \quad \widetilde{\theta}^2_{df} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 4 \end{bmatrix} \quad \widetilde{\theta}^2_{fg} = \begin{bmatrix} 4 & 0 \\ 0 & 0 \end{bmatrix}$$

Condition $\sum_T \rho^T \widetilde{\theta}^T \equiv \sum_T \rho^T \theta^T$ 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 $\boldsymbol{\theta}$ and $\widetilde{\boldsymbol{\theta}}$. However, we have $\Phi_\rho(\boldsymbol{\theta}) = 0$ and $\Phi_\rho(\widetilde{\boldsymbol{\theta}}) = 0.5$.

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 $\boldsymbol{\theta}$ is a fixed point of TRW algorithms.



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

Figure 7: **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 $T_2 = (a, c, d, f, g)$.**

Note that in the example above one of the nodes has three possible states. This is not accidental: if all nodes have binary states, then WTA condition always yields a global maximum of the lower bound, as our paper (with M. Wainwright) shows [12].

Appendix E: Proof of theorem 3.6

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

(1) According to theorem 3.4(a) terms $\Phi(\theta^T)$ are non-decreasing. By lemma 6.1 we have $\theta^T_{const} = \Phi(\theta^T)$. This shows that θ^T_{const} are bounded from below. The following inequality then implies that they are also bounded from above:

$$\textstyle\sum_T \rho^T \theta^T_{const} = \sum_T \rho^T \Phi(\theta^T) \le \Phi(\bar{\theta})$$

(2) Consider node $s \in \mathcal{V}$ and variable $j \in \mathcal{X}_s$. By definition of the canonical normal form, elements $\theta^T_{s;j}$ 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

$$\begin{aligned}\textstyle\sum_T \rho^T (\theta^T_{const} + \theta^T_{s;j}) &= \textstyle\sum_T \rho^T \Phi_{s;j}(\theta^T) \\ &\le \Phi_{s;j}(\textstyle\sum_T \rho^T \theta^T) = \Phi_{s;j}(\bar{\theta})\end{aligned}$$

We already showed that θ^T_{const} are bounded, so this implies that $\theta^T_{s;j}$ 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, $\theta^T_{s;j} +$

$\theta^T_{st;jk} + \theta^T_{t;k} \geq 0$. It implies that $\theta^T_{st;jk}$ is bounded from below since $\theta^T_{s;j}$ and $\theta^T_{t;k}$ 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] Y. Boykov and V. Kolmogorov. An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. *PAMI*, 26(9), September 2004.

[2] Y. Boykov and M.-P. Jolly. Interactive graph cuts for optimal boundary and region segmentation of objects in N-D images. In *ICCV*, 2001.

[3] Y. Boykov, O. Veksler, and R. Zabih. Fast approximate energy minimization via graph cuts. *PAMI*, 23(11), November 2001.

[4] C. Chekuri, S. Khanna, J. Naor, and L. Zosin. Approximation algorithms for the metric labeling problem via a new linear programming formulation. In *SODA*, 2000.

[5] R. G. Cowell, A. P. Dawid, S. L. Lauritzen, and D. J. Spiegelhalter. *Probabilistic Networks and Expert Systems*. Springer-Verlag, 1999.

[6] Pedro F. Felzenszwalb and Daniel P. Huttenlocher. Efficient belief propagation for early vision. In *CVPR*, 2004.

[7] W. Freeman, E. Pasztor, and O. Carmichael. Learning low-level vision. *IJCV*, 40(1):25–47, 2000.

[8] S. Geman and D. Geman. Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images. *PAMI*, 6:721–741, 1984.

[9] 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.

[10] Hiroshi Ishikawa. Exact optimization for Markov Random Fields with convex priors. *PAMI*, 25(10):1333–1336, October 2003.

[11] J. Kleinberg and E. Tardos. Approximation algorithms for classification problems with pairwise relationships: Metric labeling and Markov Random Fields. In *FOCS*, 1999.

[12] V. Kolmogorov and M. Wainwright. On the optimality of tree-reweighted max-product message passing. In *UAI*, July 2005.

[13] V. Kolmogorov. Convergent tree-reweighted message passing for energy minimization. In *AISTATS*, January 2005.

[14] V. Kolmogorov and R. Zabih. Multi-camera scene reconstruction via graph cuts. In *ECCV*, volume 3, pages 82–96, 2002.

[15] V. Kolmogorov and R. Zabih. What energy functions can be minimized via graph cuts? *PAMI*, 26(2):147–159, February 2004.

[16] N. Komodakis and G. Tziritas. A New Framework for Approximate Labeling via Graph Cuts. In *ICCV*, 2005.

[17] A. Koster, C. P. M. van Hoesel, and A. W. J. Kolen. The partial constraint satisfaction problem: Facets and lifting theorems. *Operation Research Letters*, 23(3-5):89–97, 1998.

[18] V. K. Koval and M. I. Schlesinger. Two-dimensional programming in image analysis problems. *Automatics and Telemechanics*, 2:149–168, 1976. In Russian.

[19] Vivek Kwatra, Arno Schödl, Irfan Essa, Greg Turk, and Aaron Bobick. Graphcut textures: Image and video synthesis using graph cuts. *SIGGRAPH*, July 2003.

[20] T. Meltzer, C. Yanover, and Y. Weiss. Globally optimal solutions for energy minimization in stereo vision using reweighted belief propagation. In *ICCV*, 2005.

[21] Mark A. Paskin. Sample propagation. In *NIPS*, 2003.

[22] J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann Publishers, 1988.

[23] D. Scharstein and R. Szeliski. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *IJCV*, 47:7–42, April 2002.

[24] M. I. Schlesinger. Syntactic analysis of two-dimensional visual signals in noisy conditions. *Kibernetika*, 4:113–130, 1976. In Russian.

[25] M. I. Schlesinger. *Mathematical tools for image processing*. Kiev: Nauk. dumka, 1989. In Russian.

[26] M. I. Schlesinger and B. Flach. Some solvable subclass of structural recognition problems. In *Czech Pattern Recognition Workshop*, 2000.

[27] J. Sun, N. Zheng, and H. Shum. Stereo matching using belief propagation. *PAMI*, 25(7):787–800, 2003.

[28] M. F. Tappen and W. T. Freeman. Comparison of graph cuts with belief propagation for stereo, using identical MRF parameters. In *ICCV*, October 2003.

[29] Y. W. Teh and M. Welling. On improving the efficiency of the iterative proportional fitting procedure. In *AISTATS*, 2003.

[30] 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.

[31] M.J. Wainwright, T.S. Jaakkola, and A.S. Willsky. Tree-reweighted belief propagation and approximate ML estimation by pseudo-moment matching. In *AISTATS*, January 2003.

[32] 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.

[33] M.J. Wainwright, T.S. Jaakkola, and A.S. Willsky. MAP estimation via agreement on (hyper)trees: Message-passing and linear-programming approaches. *IEEE Transactions on Information Theory*, 51(11):3697–3717, November 2005.

[34] Tomáš Werner. A linear programming approach to max-sum problem: A review. Technical Report CTU–CMP–2005–25, Center for Machine Perception, Czech Technical University, December 2005.

[35] Jonathan S. Yedidia, William T. Freeman, and Yair Weiss. Generalized belief propagation. In *NIPS*, pages 689–695, 2000.