Convergent Decomposition Solvers for Tree-reweighted Free Energies

Jeremy Jancsary
Austrian Research Institute
for Artificial Intelligence (OFAI)
jeremy.jancsary@ofai.at

Gerald Matz
Institute of Telecommunications
Vienna University of Technology
gerald.matz@tuwien.ac.at

Abstract

We investigate minimization of tree-reweighted free energies for the purpose of obtaining approximate marginal probabilities and upper bounds on the partition function of cyclic graphical models. The solvers we present for this problem work by directly tightening tree-reweighted upper bounds. As a result, they are particularly efficient for tree-reweighted energies arising from a small number of spanning trees. While this assumption may seem restrictive at first, we show how small sets of trees can be constructed in a principled manner. An appealing property of our algorithms, which results from the problem decomposition, is that they are embarrassingly parallel. In contrast to the original message passing algorithm introduced for this problem, we obtain global convergence guarantees.

1 INTRODUCTION

Exact computation of marginal probabilities and the partition function in general graphical models is an NP-hard problem that scales exponentially in the treewidth of the graph (Chandrasekaran et al., 2008). Much effort has been put into construction of approximate inference algorithms that remain tractable even for graphs of large treewidth, such as those involving many cycles. Good results were initially obtained using loopy belief propagation, which ignores the cycles and performs message updates as if the graph were a tree. Theoretical justification was later given to the method by showing that it can be understood to minimize the so-called Bethe free energy (Yedidia et al., 2003). However, the Bethe free energy is convex only for tree-structured graphs. Hence, loopy belief propagation cannot in general be expected to establish the global minimum, nor is it guaranteed to converge.

In the seminal work of Wainwright et al. (2005a), tree-reweighted (TRW) free energies were introduced to rectify this problem. These energies arise from a convex combination of log partition functions of spanning trees that forms a natural upper bound on the exact log partition function. The tree-reweighted free energy itself then only depends on edge occurrence probabilities resulting from the choice of trees. An adapted message passing algorithm was derived that is reminiscent of loopy belief propagation, but minimizes a tree-reweighted free energy instead of the Bethe free energy. However, despite convexity of its objective function, the algorithm is not guaranteed to converge. Indeed, we will demonstrate this failure of convergence.

Consequently, recent work has focused on establishing convergent variants of the original algorithm. Previous attempts have aimed at optimization of the tree-reweighted free energy itself, rather than direct minimization of the convex upper bound. In part, this is due to the original presentation by Wainwright et al. (2005a), who form the convex combination over *all* spanning trees of the cyclic graph. Naturally, direct minimization of this bound is infeasible. However, if the upper bound is restricted to a small number of spanning trees, this optimization problem has favorable properties. Moreover, approximate marginal probabilities result naturally as a byproduct.

In fact, for the related maximum-a-posteriori (MAP) problem, Komodakis et al. (2007) have shown that a similar convex upper bound, formed over a small number of trees, can be minimized efficiently using the projected subgradient algorithm. Optimization of the upper bound on the log partition function differs in two key ways. First, the problem is smooth, which suggests improved asymptotic properties. Second, the choice of spanning trees can have significant influence on the tightness of the optimum.

Appearing in Proceedings of the 14^{th} International Conference on Artificial Intelligence and Statistics (AISTATS) 2011, Fort Lauderdale, FL, USA. Volume 15 of JMLR: W&CP 15.

In this paper, we make the following contributions: (a) We investigate direct minimization of tree-reweighted upper bounds on the log partition function using the spectral projected gradient algorithm (Birgin et al., 2000) and the projected quasi-Newton algorithm (Schmidt et al., 2009). The core of the resulting algorithms is embarrassingly parallel and we demonstrate that it scales accordingly in the number of processors. (b) We present strategies for choosing small sets of spanning trees and study their effect on the error of marginal probabilities, tightness of the upper bound and computational cost. These results are of general interest as the choice of trees (or edge probabilities) is mandated by any tree-reweighted algorithm.

2 BACKGROUND

Next, we briefly review the most important concepts we will be concerned with.

2.1 UNDIRECTED GRAPHICAL MODELS

We consider undirected graphical models G with vertex set $\mathcal{V}$ and edge set $\mathcal{E}$ defined over n discrete random variables with pairwise interactions. The probability of a particular variable state $\boldsymbol{x} \in \mathcal{X}^n$ thus factors as

$$p(\boldsymbol{x};\boldsymbol{\theta}) = \exp\left(\sum_{s\in\mathcal{V}} \theta_s(x_s) + \sum_{(s,t)\in\mathcal{E}} \theta_{st}(\boldsymbol{x}_{st}) - \Phi(\boldsymbol{\theta})\right),$$

where the log partition function is defined as

$$\Phi(\boldsymbol{\theta}) = \log \sum_{\boldsymbol{x}\in\mathcal{X}^n} \exp\left(\sum_{s} \theta_s(x_s) + \sum_{(s,t)} \theta_{st}(\boldsymbol{x}_{st})\right).$$

We shall find it convenient to express the same factorization using a vector-valued indicator function $\boldsymbol{\phi}(\boldsymbol{x})$, which maps a variable state to binary indicators for the corresponding components of $\boldsymbol{\theta} \in \mathbb{R}^d$:

$$p(\boldsymbol{x};\boldsymbol{\theta}) = \exp\left(\boldsymbol{\phi}(\boldsymbol{x})\cdot\boldsymbol{\theta} - \Phi(\boldsymbol{\theta})\right), \tag{1}$$

and similarly,

$$\Phi(\boldsymbol{\theta}) = \log \sum_{\boldsymbol{x}\in\mathcal{X}^n} \exp\left(\boldsymbol{\phi}(\boldsymbol{x})\cdot\boldsymbol{\theta}\right). \tag{2}$$

Subsequently, we will be concerned with computation of approximations to $\Phi(\boldsymbol{\theta})$ and the marginal probabilities

$$\mathbb{E}\{\phi_\alpha(\boldsymbol{x})\} = \sum_{\boldsymbol{x}\in\mathcal{X}^n} p(\boldsymbol{x};\boldsymbol{\theta})\phi_\alpha(\boldsymbol{x}), \tag{3}$$

where we use α to refer to a single index corresponding to a particular state of a vertex s or an edge (s,t). Interestingly, the first and second derivatives of $\Phi(\boldsymbol{\theta})$ generate the cumulants

$$\frac{\partial\Phi(\boldsymbol{\theta})}{\partial\theta_\alpha} = \mathbb{E}\{\phi_\alpha(\boldsymbol{x})\} \text{ and } \frac{\partial^2\Phi(\boldsymbol{\theta})}{\partial\theta_\alpha\partial\theta_\beta} = \operatorname{cov}\{\phi_\alpha(\boldsymbol{x}),\phi_\beta(\boldsymbol{x})\}.$$

Hence, the marginal probabilities are given precisely by the gradient of the log partition function. Moreover, the covariance matrix, which is by definition positive semi-definite, forms the Hessian. Convexity of the log partition function follows from this property.

2.2 TREE-REWEIGHTED BOUNDS

Consider now the set $\mathcal{T} = \{T\}$ of all spanning trees of a cyclic graph G. We use $\mathcal{I}(T)$ to denote the set of indices $\{\alpha\}$ corresponding to states x_s of vertices and $\boldsymbol{x}_{st}$ of edges that belong to a particular tree T. Each of the spanning trees is associated with a parameterization $\boldsymbol{\theta}(T)$ that is tractable by the structural assumption. Wainwright et al. (2005a) observe that a convex combination $\sum_T \rho(T)\Phi(\boldsymbol{\theta}(T))$ over trees yields an upper bound on $\Phi(\boldsymbol{\theta})$ if the tractable parameter vector $\vec{\boldsymbol{\theta}} = [\boldsymbol{\theta}(T_1),\ldots,\boldsymbol{\theta}(T_m)] \in \mathbb{R}^{md}$ lies in the convex set

$$\mathcal{C}(\boldsymbol{\theta}) = \left\{\vec{\boldsymbol{\theta}} \,\middle|\, \begin{array}{l}\theta_\alpha(T) = 0 \text{ for all } T, \alpha \notin \mathcal{I}(T)\\ \sum_T \rho(T)\boldsymbol{\theta}(T) = \boldsymbol{\theta}\end{array}\right\}, \tag{4}$$

and $\boldsymbol{\rho} = \{\rho(T)\}$ is constrained to belong to the simplex of distributions over $\mathcal{T}$,

$$\Delta = \left\{\boldsymbol{\rho} \mid \textstyle\sum_T \rho(T) = 1, \rho(T) \geq 0\right\}. \tag{5}$$

Observe that $\boldsymbol{\rho}$ must also be valid in the sense that each edge is covered with non-zero probability, otherwise $\mathcal{C}(\boldsymbol{\theta})$ is empty. The upper bound property now follows directly from Jensen's inequality:

$$\Phi(\boldsymbol{\theta}) = \Phi\big(\textstyle\sum_T \rho(T)\boldsymbol{\theta}(T)\big) \leq \sum_T \rho(T)\Phi(\boldsymbol{\theta}(T)).$$

The structural constraints $\theta_\alpha(T) = 0$ in $\mathcal{C}(\boldsymbol{\theta})$ are not required for the upper bound to hold, but we include them in our presentation to make explicit the fact that the parameterizations $\boldsymbol{\theta}(T)$ are tractable.

A natural question is then how to obtain the tightest upper bound possible within this framework. For a given distribution $\boldsymbol{\rho}$ over spanning trees, and target parameters $\boldsymbol{\theta}$, we can simply optimize over the set of tractable parameterizations $\vec{\boldsymbol{\theta}}$,

$$\min_{\vec{\boldsymbol{\theta}}\in\mathcal{C}(\boldsymbol{\theta})} \sum_T \rho(T)\Phi(\boldsymbol{\theta}(T)). \tag{6}$$

Since the upper bound is a convex combination of convex functions, and the constraint set is convex, this is a convex optimization problem.

2.3 TREE-REWEIGHTED ENERGIES

By forming the Lagrangian of (6) and exploiting the conjugate duality relation between the log partition function and the negative entropy of a distribution, one can obtain an equivalent dual problem:

$$\max_{\boldsymbol{\mu}\in\mathcal{L}(\mathcal{G})} \Big\{\boldsymbol{\mu}\cdot\boldsymbol{\theta} + \textstyle\sum_s H(\boldsymbol{\mu}_s) - \sum_{(s,t)} \nu_{st} I(\boldsymbol{\mu}_{st})\Big\}, \quad (7)$$

where $\boldsymbol{\mu}_s$ and $\boldsymbol{\mu}_{st}$ have interpretations as node and edge pseudomarginals, $H(\cdot)$ and $I(\cdot)$ denote the Shannon entropy and the mutual information, respectively, and the constraint set

$$\mathcal{L}(G) = \left\{ \boldsymbol{\mu} \geq \mathbf{o} \;\middle|\; \begin{array}{l} \sum_{x_s} \mu_s(x_s) = 1 \\ \sum_{\boldsymbol{x}_{st}\sim x_s} \mu_{st}(\boldsymbol{x}_{st}) = \mu_s(x_s) \end{array} \right\} \quad (8)$$

ensures proper local normalization and marginalization consistency[1]. The edge probabilities $\boldsymbol{\nu} = \{\nu_{st}\}$ are strictly positive and arise from the valid distribution $\boldsymbol{\rho} \in \Delta$ over spanning trees.

The objective function in (7) is the negative *tree-reweighted free energy*. As the dual of a convex function, it is concave in $\boldsymbol{\mu}$, and strong duality holds. The primary advantage of problem (7) over (6) is its reduced dimensionality: it is independent of the number of spanning trees involved. However, constraint set $\mathcal{L}(G)$ is considerably more complicated than $\mathcal{C}(\boldsymbol{\theta})$.

3 APPROACH

The original message passing algorithm by Wainwright et al. (2005a) can be understood to perform block coordinate updates in the Lagrangian of (7). However, without further precautions, the scheme is not guaranteed to converge. In practice, "damping" strategies are often applied to improve the convergence characteristics. In contrast, we investigate efficient methods for direct minimization of (6). The coupling constraints in $\mathcal{C}(\boldsymbol{\theta})$ are more easy to handle than $\mathcal{L}(G)$, and convergent minimization schemes thus arise naturally. We next discuss several key aspects of our approach.

3.1 OBTAINING MARGINALS

An approximation to the log partition function is naturally given by the optimum of problem (6). In contrast, it is not so obvious how to obtain approximate marginals from the solution. The key observation here arises en route of deriving (7) from (6): By forming the Lagrangian of (6), and taking derivatives with respect to θ_α, one obtains the stationary conditions

$$\mathbb{E}_{\boldsymbol{\theta}^\star(T)}\{\phi_\alpha(\boldsymbol{x})\} \stackrel{!}{=} \mu_\alpha \text{ for all } T, \alpha \in \mathcal{I}(T)\ .$$

[1] We use $\boldsymbol{x}_{st} \sim x_s$ to denote edge states $\boldsymbol{x}_{st}$ that are consistent with node state x_s.

Consequently, at the optimal solution $\vec{\boldsymbol{\theta}}^\star$, all trees share a single set of marginals. To construct a full set of pseudomarginals $\boldsymbol{\mu}$, for each index α, we can thus use the marginal probability of any tree T for which $\alpha \in \mathcal{I}(T)$ once (6) is solved to optimality. Notably, the marginals of any tree can be obtained efficiently.

3.2 COMPUTING THE GRADIENT

As we pointed out in section 2.1, the derivative of the log partition function $\Phi(\cdot)$ with respect to θ_α is given by the corresponding marginal probability, $\mathbb{E}\{\phi_\alpha(\boldsymbol{x})\}$. Given that (6) is a weighted sum of such partition functions, it is easy to see that the full gradient

$$\begin{aligned}\nabla_{\vec{\boldsymbol{\theta}}} &= \sum_T \big[0 \cdots \rho(T)\mathbb{E}_{\boldsymbol{\theta}(T)}\{\boldsymbol{\phi}(\boldsymbol{x})\} \cdots 0\big] \\ &= \big[\rho(T_1)\mathbb{E}_{\boldsymbol{\theta}(T_1)}\{\boldsymbol{\phi}(\boldsymbol{x})\}, \ldots, \rho(T_m)\mathbb{E}_{\boldsymbol{\theta}(T_m)}\{\boldsymbol{\phi}(\boldsymbol{x})\}\big]\end{aligned}$$

is thus given as a vector of weighted component gradients. In principle, this gradient can be computed very efficiently; the only concern is the number m of spanning trees involved. We discuss this issue in great detail in section 3.5.

3.3 HANDLING THE CONSTRAINTS

We now turn to discussion of the constraint set $\mathcal{C}(\boldsymbol{\theta})$, defined in (4). Both the coupling constraints $\sum_T \rho(T)\boldsymbol{\theta}(T) = \boldsymbol{\theta}$ and the structural constraints $\theta_\alpha(T) = 0$ are linear, so $\mathcal{C}(\boldsymbol{\theta})$ defines a convex polytope. As we shall point out, projection onto this set can be realized very efficiently. Formally, we search the solution to the following optimization problem:

$$\mathcal{P}_{\boldsymbol{\theta}}(\vec{\boldsymbol{\theta}}') = \operatorname*{argmin}_{\vec{\boldsymbol{\theta}}\in\mathcal{C}(\boldsymbol{\theta})} \big\|\vec{\boldsymbol{\theta}} - \vec{\boldsymbol{\theta}}'\big\|_2^2\ . \quad (9)$$

For all T, if $\alpha \notin \mathcal{I}(T)$, the structural constraints prescribe $\theta_\alpha(T) = 0$. These components are hence fully specified. Otherwise, the coupling constraints $\sum_T \rho(T)\theta_\alpha(T) = \theta_\alpha$ must be satisfied. Among the admissible $\{\theta_\alpha(T)\}$ for a given index α, whose weighted sum must be θ_α, the sum of squares is minimized if $(\theta_\alpha(T) - \theta'_\alpha(T))^2$ is equal for all trees T with $\alpha \in \mathcal{I}(T)$. Consider now the distance from the target parameter $\delta_\alpha = (\sum_T \rho(T)\theta'_\alpha(T) - \theta_\alpha)$ and the accumulated probability mass $\sigma_\alpha = \sum_{T:\alpha\in\mathcal{I}(T)} \rho(T)$. It can be verified that the projection given by

$$\mathcal{P}_{\boldsymbol{\theta}}(\vec{\boldsymbol{\theta}}') = \begin{cases} \theta_\alpha(T) = 0 & \text{if } \alpha \notin \mathcal{I}(T) \\ \theta_\alpha(T) = \theta'_\alpha(T) - \frac{\delta_\alpha}{\sigma_\alpha} & \text{otherwise} \end{cases} \quad (10)$$

ensures satisfaction of all constraints while adhering to the optimality criterion discussed above. Hence, it provides a solution to (9) which can be computed in $O(md)$, i.e. in time linear in the dimensionality of $\vec{\boldsymbol{\theta}}$.

Algorithm 1: TightenBound (SPG Variant)

input : set of trees $\mathcal{T}$ and valid distribution $\boldsymbol{\rho}$, target parameters $\boldsymbol{\theta}$, arbitrary initial $\vec{\boldsymbol{\theta}}^{(1)}$, step size interval $[\alpha_{\min}, \alpha_{\max}]$, history length h

output: pseudomarginals $\boldsymbol{\mu}$, upper bound $\tilde{\Phi} \geq \Phi(\boldsymbol{\theta})$

$\vec{\boldsymbol{\theta}}^{(1)} \leftarrow \mathcal{P}_{\boldsymbol{\theta}}(\vec{\boldsymbol{\theta}}^{(1)})$

$\tilde{\Phi}^{(1)} \leftarrow$ **parallelized** $\sum_T \rho(T)\Phi(\boldsymbol{\theta}^{(1)}(T))$

$\alpha^{(1)} \leftarrow 1/\|\mathcal{P}_{\boldsymbol{\theta}}(\vec{\boldsymbol{\theta}}^{(1)} - \nabla^{(1)}_{\vec{\boldsymbol{\theta}}}) - \vec{\boldsymbol{\theta}}^{(1)}\|$

$k \leftarrow 1$

while $\|\mathcal{P}_{\boldsymbol{\theta}}(\vec{\boldsymbol{\theta}}^{(k)} - \nabla^{(k)}_{\vec{\boldsymbol{\theta}}}) - \vec{\boldsymbol{\theta}}^{(k)}\| < \varepsilon$ **do**

  $d^{(k)} \leftarrow \mathcal{P}_{\boldsymbol{\theta}}(\vec{\boldsymbol{\theta}}^{(k)} - \alpha^{(k)}\nabla^{(k)}_{\vec{\boldsymbol{\theta}}}) - \vec{\boldsymbol{\theta}}^{(k)}$

  repeat

    choose $\lambda \in (0, 1)$ *; e.g. via interpolation*

    $\vec{\boldsymbol{\theta}}^{(k+1)} \leftarrow \vec{\boldsymbol{\theta}}^{(k)} + \lambda d^{(k)}$

    $\tilde{\Phi}^{(k+1)} \leftarrow$ **parallelized** $\sum_T \rho(T)\Phi(\boldsymbol{\theta}^{(k+1)}(T))$

  until $\tilde{\Phi}^{(k+1)} < \max\{\tilde{\Phi}^{(k)}, \ldots, \tilde{\Phi}^{(k-h)}\} + \epsilon\lambda\nabla^{(k)}_{\vec{\boldsymbol{\theta}}} \cdot d$

  $s^{(k)} \leftarrow \vec{\boldsymbol{\theta}}^{(k+1)} - \vec{\boldsymbol{\theta}}^{(k)}$

  $y^{(k)} \leftarrow \nabla^{(k+1)}_{\vec{\boldsymbol{\theta}}} - \nabla^{(k)}_{\vec{\boldsymbol{\theta}}}$

  $\alpha^{(k+1)} \leftarrow \min\{\alpha_{\max}, \max\{\alpha_{\min}, (s^{(k)} \cdot s^{(k)})/(s^{(k)} \cdot y^{(k)})\}\}$

  $k \leftarrow k + 1$

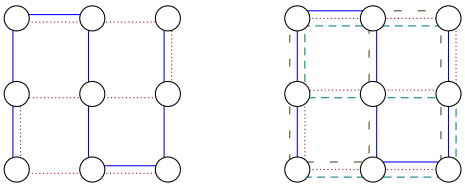
return $(\tilde{\Phi}^{(k)}, \text{marginals}\{\nabla^{(k)}_{\vec{\boldsymbol{\theta}}}\})$ *; see section 3.1*

3.4 TIGHTENING THE BOUND

For now, assume that $\rho(T) > 0$ for a small number of trees T only. The gradient of our objective in (6) can then be computed efficiently. Moreover, the constraint set $\mathcal{C}(\boldsymbol{\theta})$ is convex and can be projected onto at little cost. A principal method for optimization in such a setting is the projected gradient algorithm. However, this basic method can be improved on.

3.4.1 Spectral Projected Gradient Method

The main improvements of the spectral projected gradient (SPG) method (Birgin et al., 2000) over classic projected gradient descent are a particular choice of the step size (Barzilai and Borwein, 1988) and a non-monotone, yet convergent line search (Grippo et al., 1986). In the setting of unconstrained quadratics, the SPG algorithm has been observed to converge super-linearly towards the optimum. We outline its application to (6) in Algorithm 1. Besides the mandatory input $\mathcal{T}$, $\boldsymbol{\rho}$ and $\boldsymbol{\theta}$, the meta parameters $[\alpha_{\min}, \alpha_{\max}]$ specify the interval of admissible step sizes, and history length h specifies how many steps may be taken without sufficient decrease of the objective. If the number of steps is exceeded, backtracking is performed and the step size is decremented until sufficient decrease has been established. In our implementation, we chose $\alpha_{\min} = 10^{-10}$, $\alpha_{\max} = 10^{10}$ and $h = 10$. In the backtracking step, we simply multiply with a factor $\lambda = 0.3$. In practice, we found Algorithm 1 to be very robust with respect to the choice of meta parameters.



(a) (b)

Figure 1: (a) Two "snakes" cover any grid; (b) Two more mirrored replicas achieve symmetric edge probabilities.

Proposition 1. *For a given set of spanning trees $\mathcal{T}$, valid distribution over trees $\boldsymbol{\rho}$ and target parameters $\boldsymbol{\theta}$, Algorithm 1 converges to the global optimum of (6).*

Proof (Sketch). Convergence follows from the analysis of the SPG method by Wang et al. (2005). □

3.4.2 Projected Quasi-Newton Method

The projected quasi-Newton (PQN) method was recently introduced by Schmidt et al. (2009) and can be considered a generalization of L-BFGS (Nocedal, 1980) to constrained optimization. At each iteration, a feasible direction is found by minimizing a quadratic model subject to the original constraints:

$$\min_{\vec{\boldsymbol{\theta}} \in \mathcal{C}(\boldsymbol{\theta})} \tilde{\Phi}^{(k)} + (\vec{\boldsymbol{\theta}} - \vec{\boldsymbol{\theta}}^{(k)}) \cdot \nabla^{(k)}_{\vec{\boldsymbol{\theta}}} + \tfrac{1}{2}(\vec{\boldsymbol{\theta}} - \vec{\boldsymbol{\theta}}^{(k)})^T B^{(k)} (\vec{\boldsymbol{\theta}} - \vec{\boldsymbol{\theta}}^{(k)})$$

where $B^{(k)}$ is a positive-definitive approximation to the Hessian that is maintained in compact form in terms of the previous p iterates and gradients (Byrd et al., 1994). The SPG algorithm can be used to perform the above minimization effectively. We hypothesized that PQN might compensate for the larger per-iteration cost through improved asymptotic convergence and thus implemented a scheme similar to Algorithm 1. We do not give a complete specification here, as it only differs from Algorithm 1 in the choice of the direction and the use of a traditional line search.

3.5 CHOOSING THE SET OF TREES

It is clear that Algorithm 1 is only efficient for a reasonably small number of selected trees with $\rho(T) > 0$. We refer to this set as $\mathcal{S}$ and denote the corresponding vector of non-zero coefficients by $\boldsymbol{\rho}_s$. Subsequently, we discuss how to obtain $\mathcal{S}$ and $\boldsymbol{\rho}_s$ in a principled manner.

3.5.1 Uniform Edge Probabilities

According to the Laplacian principle of insufficient reasoning, one might choose uniform edge occurrence probabilities given by $\nu_{st} = (|\mathcal{V}| - 1)/|\mathcal{E}|$. However, in our formulation, we need to find a pair $(\mathcal{S}, \boldsymbol{\rho}_s)$ that *results* in these probabilities. The dual coupling between $(\mathcal{S}, \boldsymbol{\rho}_s)$ and $\boldsymbol{\nu}$ is defined in terms of the map-

Algorithm 2: CoveringTrees

input : graph G, stopping criterion
output: selected trees $\mathcal{S}$, valid $\boldsymbol{\rho}_s$
$\mathcal{S}^{(1)} \leftarrow \{\text{random spanning tree}\}$, $\boldsymbol{\rho}_s^{(1)} \leftarrow [1]$, $k \leftarrow 1$
while not criterion **do**
  $\boldsymbol{\nu}^{(k)} \leftarrow \boldsymbol{\nu}(\mathcal{S}^{(k)}, \boldsymbol{\rho}_s^{(k)})$ *; compute edge probabilities*
  $\mathcal{S}^{(k+1)} \leftarrow \mathcal{S}^{(k)} \cup \mathrm{MST}(G, \boldsymbol{\nu}^{(k)})$ *; MST f. edge cost* $\boldsymbol{\nu}^{(k)}$
  $\boldsymbol{\rho}_s^{(k+1)} \leftarrow \mathbf{1}/(k+1)$ *; for* $\mathbf{1} \in \mathbb{R}^{k+1}$
  $k \leftarrow k+1$
return $(\mathcal{S}^{(k)}, \boldsymbol{\rho}_s^{(k)})$

ping $\boldsymbol{\nu}(\mathcal{S}, \boldsymbol{\rho}_s) = \sum_{T \in \mathcal{S}} \rho_s(T) \boldsymbol{\nu}(T)$, where $\boldsymbol{\nu}(T) \in \mathbb{R}^{|\mathcal{E}|}$ indicates the edges contained in T, such that $\nu_{st}(T) = [(s,t) \in \mathcal{E}_T]$. Algorithm 2 establishes a suitable pair $(\mathcal{S}, \boldsymbol{\rho}_s)$ in a greedy manner. At each step, we add a minimum spanning tree (MST) for weights given by the current edge probabilities. We stop when $\boldsymbol{\nu}(\mathcal{S}, \boldsymbol{\rho}_s)$ is sufficiently uniform, which allows to trade off the number of resulting trees against uniformity.

Proposition 2. *Algorithm 2 determines a sequence* $\{\boldsymbol{\nu}(\mathcal{S}^{(k)}, \boldsymbol{\rho}_s^{(k)})\}$ *that converges to a vector* $\boldsymbol{u}$ *with components given by* $u_{st} = (|\mathcal{V}|-1)/|\mathcal{E}|$ *as* $k \to \infty$.

We outline a proof in the supplementary material, appendix A.1.1; Algorithm 2 takes conditional gradient steps that seek to minimize $\|\boldsymbol{\nu}(\mathcal{S}, \boldsymbol{\rho}_s) - \boldsymbol{u}\|_2^2$.

3.5.2 Snake-Based Strategy

For grid-structured graphs, we also found that fairly uniform edge occurrence probabilities could be obtained using four "snake"-shaped trees that in sum cover all edges. This is best seen in terms of an illustration, which we provide in Figure 1. If we choose $\boldsymbol{\rho}_s = \mathbf{1}/|\mathcal{S}|$, the edges in the interior assume $\nu_{st} = 1/2$, whereas those on the boundary are given by $\nu_{st} = 3/4$.

3.5.3 Constructing an Almost Minimal Set

If we choose a different stopping criterion, namely $\nu_{st} > 0 \; \forall (s,t)$, Algorithm 2 can also be used to greedily establish a set of trees that is almost minimal in the sense that its cardinality is close to the minimum number of spanning trees required to cover all edges of G. Note that there is no guarantee of optimality in this respect. However, in practice, we found that Algorithm 2 was very effective at establishing such sets.

3.5.4 Obtaining an Optimal Set

Wainwright et al. (2005a) show that one can obtain even tighter upper bounds by optimizing (7) over the edge occurrence probabilities $\boldsymbol{\nu}$. This is achieved using conditional gradient steps, where each such outer iteration involves solution of (7) for the current iterate $\boldsymbol{\nu}^{(k)}$ and a subsequent minimum spanning tree (MST) search with edge weights given by the negative mutual information (MI) of the current edge pseudo-marginals, denoted by $I(\boldsymbol{\mu}_{st})$. The resulting bound is jointly optimal over $\boldsymbol{\nu}$ and $\boldsymbol{\mu}$. Algorithm 3 defines a similar procedure for the primal space we are operating in. It successively establishes pairs $(\mathcal{S}, \boldsymbol{\rho}_s)$ resulting in increasingly tighter upper bounds $\tilde{\Phi}$. The invocation of CoveringTrees$(\cdot)$ in the initialization phase ensures that we start from a valid distribution $\boldsymbol{\rho}_s$ and a small set $\mathcal{S}$ such that each edge is covered with non-zero probability and Algorithm 1 can be applied. In practice, the biggest gains are achieved in the first few iterations. Hence, although it is expensive to find a suitable tree at each iteration, the number of trees stays relatively small, and we approach the joint optimum in the process.

Algorithm 3: OptimalTrees

input : graph G, target parameters $\boldsymbol{\theta}$
output: selected trees $\mathcal{S}$, valid $\boldsymbol{\rho}_s$, $\boldsymbol{\mu}$, $\tilde{\Phi} \geq \Phi(\boldsymbol{\theta})$
$(\mathcal{S}^{(l)}, \boldsymbol{\rho}_s^{(l)}) \leftarrow$ CoveringTrees$(G, \nu_{st} > 0 \; \forall (s,t))$
$k \leftarrow l$
while not converged **do**
  $(\tilde{\Phi}^{(k)}, \boldsymbol{\mu}^{(k)}) \leftarrow$ TightenBound$(\mathcal{S}^{(k)}, \boldsymbol{\rho}_s^{(k)}, \boldsymbol{\theta})$
  $\boldsymbol{i}^{(k)} \leftarrow [-I(\boldsymbol{\mu}_{st}^{(k)}), -I(\boldsymbol{\mu}_{uv}^{(k)}), \ldots]$ *; negative MI per edge*
  $\mathcal{S}^{(k+1)} \leftarrow \mathcal{S}^{(k)} \cup \mathrm{MST}(G, \boldsymbol{i}^{(k)})$ *; MST f. edge cost* $\boldsymbol{i}^{(k)}$
  $\boldsymbol{\rho}_s^{(k+1)} \leftarrow \mathbf{1}/(k+1)$ *; for* $\mathbf{1} \in \mathbb{R}^{k+1}$
  $k \leftarrow k+1$
return $(\mathcal{S}^{(k)}, \boldsymbol{\rho}^{(k)}, \text{TightenBound}(\mathcal{S}^{(k)}, \boldsymbol{\rho}_s^{(k)}, \boldsymbol{\theta}))$

Proposition 3. *Algorithm 3 determines a sequence* $\{\tilde{\Phi}^{(k)}\}$ *converging to an upper bound* $\tilde{\Phi}^\star \geq \Phi(\boldsymbol{\theta})$ *that is jointly optimal over the choice of trees* $\mathcal{S}$*, the distribution over trees* $\boldsymbol{\rho}_s$*, and the tractable parameterization* $\vec{\boldsymbol{\theta}}$*, as* $k \to \infty$.

The sketch of a proof is given in appendix A.1.2.

3.6 PARALLELIZING COMPUTATION

The computational cost of Algorithm 1 is dominated by computation of $\sum_T \rho(T) \Phi(\boldsymbol{\theta}^{(k+1)}(T))$, which requires sum-product belief propagation on each tree $T \in \mathcal{S}$. One might then assume that compared to traditional message passing algorithms, Algorithm 1 incurs an overhead that is asymptotically linear in the number of selected trees. However, observe that the terms $\{\Phi(\boldsymbol{\theta}^{(k+1)}(T))\}$ are completely independent of each other. Hence, as long as the number of CPU cores is greater than or equal to the number of trees, we can avoid the additional cost by scheduling each run of belief propagation on a different core. As we shall see in section 4.2.3, this works very well in practice.

Table 1: Impact of the set of spanning trees on the approximation error

	Grid IsingGauss		Grid IsingUniform		Regular IsingGauss		Complete ExpGauss	
	$e(\tilde{\Phi})$	$e(\boldsymbol{\mu})$	$e(\tilde{\Phi})$	$e(\boldsymbol{\mu})$	$e(\tilde{\Phi})$	$e(\boldsymbol{\mu})$	$e(\tilde{\Phi})$	$e(\boldsymbol{\mu})$
4Snakes	0.085 ± 0.01	0.112 ± 0.01	0.104 ± 0.01	0.087 ± 0.00	$\sim$		$\sim$	
Minimal	0.088 ± 0.01	0.113 ± 0.01	0.109 ± 0.01	0.090 ± 0.00	0.833 ± 0.10	0.308 ± 0.05	0.397 ± 0.07	0.074 ± 0.01
Uniform	0.084 ± 0.01	0.110 ± 0.01	0.102 ± 0.01	0.085 ± 0.00	0.833 ± 0.10	0.308 ± 0.05	0.394 ± 0.07	0.074 ± 0.01
*Uniform	0.083 ± 0.01	0.110 ± 0.01	0.101 ± 0.01	0.085 ± 0.00	0.833 ± 0.10	0.308 ± 0.05	0.394 ± 0.07	0.074 ± 0.01
Optimal	0.031 ± 0.01	0.091 ± 0.02	0.053 ± 0.01	0.079 ± 0.01	0.832 ± 0.10	0.308 ± 0.05	0.377 ± 0.07	0.075 ± 0.01

4 EXPERIMENTS

We wanted to assess several aspects of our algorithms empirically. Towards this end, we considered four types of random graphs that varied with respect to their structure and the exponential parameters $\boldsymbol{\theta}$.[2]

Grid IsingGauss: an $n_g \times n_g$ grid of binary variables ($\mathcal{X} = \{-1, +1\}$), with potentials chosen as $\theta_s(x_s) = \theta x_s$ and $\theta_{st}(\boldsymbol{x}_{st}) = \theta x_s x_t$, where $\theta \sim \mathcal{N}(0, 1)$ was drawn independently for each node and edge.

Grid IsingUniform: Equal to the above, except that θ was drawn from $\mathcal{U}(-1, +1)$.

Regular IsingGauss: A random regular graph with n_r binary variables, each of which was connected to n_d others, and potentials akin to Grid IsingGauss.

Complete ExpGauss: A complete graph with n_c variables ($\mathcal{X} = \{0, 1, 2, 3\}$) and potentials independently drawn as $\theta_s(x_s) = 0$ and $\theta_{st}(\boldsymbol{x}_{st}) \sim \mathcal{N}(0, 1)$.

4.1 IMPACT OF TREE SELECTION

We considered four different ways of decomposing the cyclic graphs into spanning trees: 4Snakes, described in section 3.5.2; Minimal, described in section 3.5.3; Uniform described in section 3.5.1; and finally Optimal (section 3.5.4). For the Uniform decomposition, we stopped once $\min_{(s,t)} \nu_{st} \geq 0.9 \max_{(s,t)} \nu_{st}$. The 4Snakes decomposition was only applicable to grids.

First, we wanted to assess the impact of the decomposition scheme on the approximation errors $e(\tilde{\Phi}) = |\tilde{\Phi} - \Phi(\boldsymbol{\theta})|/\Phi(\boldsymbol{\theta})$ and $e(\boldsymbol{\mu}) = \|\boldsymbol{\mu} - \mathbb{E}\{\phi(\boldsymbol{x})\}\|_1/d$. We generated 30 instances of each type of graph considered (with $n_g = 15, n_r = 30, n_d = 10$ and $n_c = 10$) and solved the corresponding instance of (6) to a tolerance of $\varepsilon = 10^{-5}$ using Algorithm 1. For the Optimal scheme, we used 50 outer iterations. The gains were minuscule beyond this point. The reference values $\Phi(\boldsymbol{\theta})$ and $\mathbb{E}\{\phi(\boldsymbol{x})\}$ were computed using join trees or brute force, depending on the type of graph.

Table 1 shows the average and the standard deviation (indicated using $\pm$) of the error over the 30 instances of each type of graph. The tree decomposition was computed anew for each instance. Unsurprisingly, the Optimal scheme performed best almost universally, with large gains in some instances. More interestingly, the other three schemes were rather closely tied, with only a slight edge for the Uniform decomposition. For comparison, we also computed the approximation errors resulting from analytically determined uniform edge probabilities (*Uniform), which corresponds to an infinite number of iterations of Algorithm 2; the gains over the Uniform scheme are negligible. Finally, we checked how deterministically the Minimal scheme behaved on a single given graph (considering its random nature). Table 2 shows that the deviation over 30 independent decompositions was very low.

Table 2: Standard deviation of the approximation error for 30 runs over the *same* graphs and potentials, using different Minimal sets of trees at each run.

	$e(\tilde{\Phi})$	$e(\boldsymbol{\mu})$
Grid IsingGauss	0.096 ± 0.0018	0.113 ± 0.0012
Grid IsingUniform	0.112 ± 0.0019	0.090 ± 0.0010
Regular IsingGauss	0.866 ± 0.0003	0.351 ± 0.0002
Complete ExpGauss	0.355 ± 0.0023	0.076 ± 0.0004

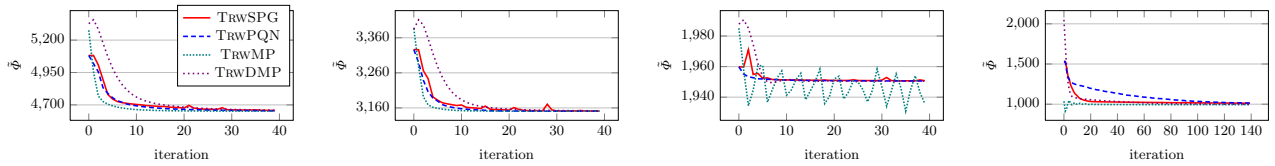
4.2 EFFECTIVENESS OF SOLVERS

In a second series of experiments, we compared our own solvers (TrwSPG, outlined by Algorithm 1, and TrwPQN with $p = 4$) to the message passing algorithm (TrwMP) of Wainwright et al. (2005a) and a variant thereof (TrwDMP) that employs "damping" ($\alpha = 0.5$). In our implementation of the latter, we updated the messages by iterating over the edges uniformly at random. For comparison, we used the same types of graphs as in section 4.1, with $n_g = 50$, $n_r = 100$, $n_d = 10$ and $n_c = 50$.

4.2.1 Asymptotic Efficiency

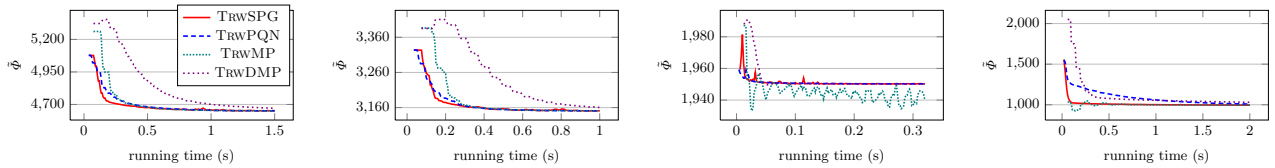
First, we compared the asymptotic behavior of the competing solvers. To this end, we ran them on the same randomly generated instances of each type of graph. Figure 2 shows the progress of the objective as a function of iterations of the respective algorithm. The plot displays only a single run of each solver

[2] We used libDAI (Mooij, 2010) to generate instances.

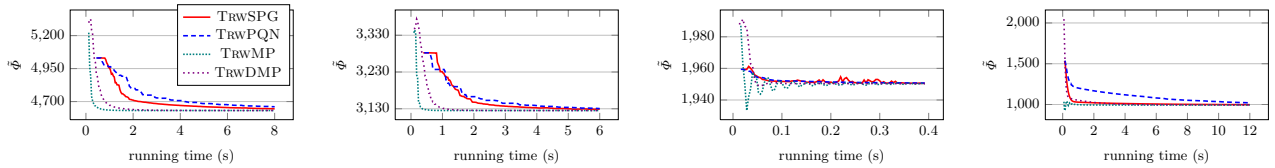


(a) Grid IsingGauss, Minimal (b) Grid IsingUniform, Minimal (c) Reg. IsingGauss, Minimal (d) Comp. ExpGauss, Minimal

Figure 2: Asymptotic efficiency—only a single run is depicted to highlight convergence characteristics.



(a) Grid IsingGauss, Minimal (b) Grid IsingUniform, Minimal (c) Reg. IsingGauss, Minimal (d) Comp. ExpGauss, Minimal



(e) Grid IsingGauss, Uniform (f) Grid IsingUniform, Uniform (g) Reg. IsingGauss, Uniform (h) Comp. ExpGauss, Uniform

Figure 3: Computational efficiency of competing solvers—average over 10 runs is shown here.

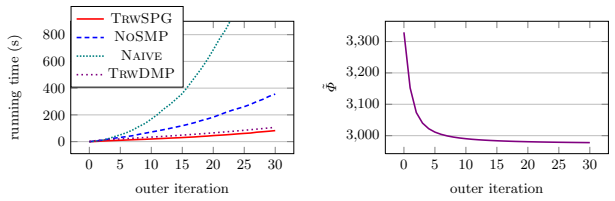
(rather than an average over multiple runs), so as not to "average out" the convergence characteristics. Due to a lack of space, we only show the curves for a particular set of trees obtained using the Minimal scheme; the others triggered similar asymptotic behavior.

As one can see from Figure 2, the message updates performed by TrwMP decrease the objective very rapidly. However, this comes at a price. In some cases, e.g. panel (c), the process diverges. We also note that the iterates produced by TrwMP need not lie within $\mathcal{L}(G)$. Feasibility is only guaranteed for the optimal solution; hence, the curve can fluctuate about the optimum, see panel (d). This can even happen for TrwDMP, which generally improves smoothness of convergence considerably, but decreases the objective more slowly. In contrast, the iterates of TrwSPG and TrwPQN are always guaranteed to yield an upper bound. In terms of smoothness of convergence, TrwPQN exposes the most desirable behavior. On the other hand, TrwSPG implements a compromise between smoothness and rapid decrease; while its non-monotone line search can yield sporadic "bumps", it ultimately converges to the global optimum.

4.2.2 Computational Efficiency

Next, we assessed the solvers in terms of their computational efficiency. For this purpose, we measured the progress of the objective as a function of running time, rather than iterations. We averaged the curve of each solver over 10 runs in order to smoothen any effects caused by the random nature of the updates performed by TrwMP, or scheduling of the multiple CPU threads used by TrwSPG and TrwPQN. All results were obtained on a machine with eight Intel Xeon CPU cores running at 2.4 GHz.

Figure 3 shows the resulting plots. We employed two decomposition schemes at opposing ends of the spectrum, Minimal (top row), and Uniform (bottom row). As expected, the results varied significantly with the number of trees in use. For Minimal sets, we found that TrwSPG approached the optimum even more quickly than TrwMP, and much more so than TrwDMP. TrwPQN was also competitive in some cases, but was generally dominated by TrwSPG due to the lower per-iteration cost. On the other hand, TrwMP and its damped variant were more efficient for the larger Uniform sets, since they only depend on the edge occurrence probabilities. This is particularly apparent in panels (e) and (f); over 50 spanning trees were required to achieve uniform edge probabilities, which significantly outnumbered the available CPU cores. However, in section 4.1, we found that there is only limited gain in establishing Uniform sets. Hence, one should definitely opt for a Minimal or 4Snakes strategy with TrwSPG and TrwPQN. In this regime, TrwSPG outperformed both TrwMP and TrwDMP while guaranteeing convergence.



(a) accumulated running time (b) progress of objective

Figure 4: Construction of Optimal sets of spanning trees for Grid IsingUniform—TrwSPG scales linearly until after the number of trees exceeds the number of CPU cores.

4.2.3 Scalability of Optimal Tree Selection

We next considered Optimal tree selection. Here, at each iteration, the set of trees grows. One might then expect the running time of Algorithm 1 to increase at each iteration, such that the accumulated running time grows superlinearly. We drew on two strategies in order to suppress this effect. First, by parallelizing computation, the cost of each iteration could be kept constant until the number of trees exceeds the number of cores. Second, by warm-starting Algorithm 1, almost-constant cost could be maintained up to a relevant number of iterations: At each outer iteration, we started from the previous solution $\vec{\boldsymbol{\theta}}^\star$; the additional parameters $\boldsymbol{\theta}(T^\star)$ of the newly added MST were obtained from the weighted average over the other trees, $\boldsymbol{\theta}(T^\star) = \sum_{T' \in \mathcal{S} \setminus T^\star} \boldsymbol{\rho}_s(T') \boldsymbol{\theta}^\star(T')$. All parameters were then projected to obtain an initial feasible point.

Figure 4 shows a run of the OptimalTrees algorithm. We compared our actual implementation (TrwSPG) to an implementation that does not use multi-processing (NoSMP) and a naive implementation that uses neither warm-starting nor multi-processing (Naive). As one can see, the differences are dramatic. Finally, we assessed an implementation (TrwDMP) that uses damped ($\alpha = 0.5$) message passing to solve the inner problem, as in Wainwright et al. (2005a). Figure 4 shows that up to a relevant number of iterations, this is less efficient than the TrwSPG-based scheme. Moreover, one does not know in advance which damping factor to choose.

5 RELATED WORK

Our formulation is most closely related to the dual decomposition scheme of Komodakis et al. (2007), who optimize an upper bound on the MAP score. As opposed to our setting, there is no strong duality between the (discrete) primal MAP problem and minimization of the convex upper bound, hence primal solutions must be generated heuristically. Moreover, the upper bound on the MAP score is non-differentiable, which has recently been dealt with using proximal regularization (Jojic et al., 2010). On the other hand, the upper bound on the log partition function depends on the choice of trees, a different source of complication.

Several independent lines of work have focused on convergent algorithms for convex free energies. Heskes (2006) derives convergent double-loop algorithms. He also argues that given sufficient damping, the original algorithm of Wainwright et al. (2005b) should converge. Globerson and Jaakkola (2007) provide a convergent algorithm for tree-reweighted free energies that solves an unconstrained geometric program. However, the authors note their work is mostly of theoretical interest, since "damped" message passing converges more rapidly. Hazan and Shashua (2008) devise a convergent algorithm for general convex energies by imposing strict non-negativity constraints on certain coefficients of the entropy terms. Meltzer et al. (2009) provide a unifying view that relates convergence to the order in which message updates are performed.

Concerning parallelization, Gonzalez et al. (2009) devise an efficient concurrent implementation of belief propagation. They show that synchronous schedules, which are naturally parallel, converge less rapidly—both empirically and theoretically. Hence, the authors parallelize a residual-based asynchronous schedule, which requires locking and considerable engineering effort. Moreover, their algorithm is not guaranteed to converge. On the other hand, some schemes that *do* guarantee convergence—such as that of Meltzer et al. (2009)—rely on the *order* of updates, which makes it inherently hard to gainfully employ parallelization.

6 CONCLUSION

We presented convergent optimization schemes for computation of approximate marginal probabilities in cyclic graphical models. For tree-reweighted energies arising from a small number of spanning trees, our SPG-based solver was shown to be more efficient than the original message passing algorithm, while guaranteeing convergence. Moreover, we found empirically that such energies provide approximations of reasonable quality. If more accurate approximations are desired, one can additionally optimize over the choice of trees. Towards this end, we outlined an efficient algorithm that draws on our convergent solvers at each iteration to establish the joint global optimum.

Acknowledgements

We thank our reviewers. OFAI is supported by the Austrian Federal Ministry for Transport, Innovation, and Technology. JJ and GM acknowledge funding by WWTF Grant ICT10-049 and FWF Grant S10606.

References

J. Barzilai and J. M. Borwein. Two-point step size gradient methods. *IMA Journal of Numerical Analysis*, 8:141–148, 1988.

D. P. Bertsekas. *Nonlinear Programming*. Athena Scientific, 2nd edition, 1999.

E. G. Birgin, J. M. Martinez, and M. Raydan. Nonmonotone spectral projected gradient methods on convex sets. *SIAM Journal of Optimization*, 10(4): 1196–1211, 2000.

R. Byrd, J. Nocedal, and R. Schnabel. Representations of quasi-Newton matrices and their use in limited memory methods. *Mathematical Programming*, 63 (1):129–156, 1994.

V. Chandrasekaran, N. Srebro, and P. Harsha. Complexity of inference in graphical models. In *24th Conference on Uncertainty in Artificial Intelligence (UAI)*, 2008.

J. Edmonds. Matroids and the greedy algorithm. *Mathematical Programming*, 1(1):127–136, 1971.

A. Globerson and T. S. Jaakkola. Convergent propagation algorithms via oriented trees. In *23rd Conference on Uncertainty in Artificial Intelligence (UAI)*, 2007.

J. E. Gonzalez, Y. Low, and C. Guestrin. Residual splash for optimally parallelizing belief propagation. In *12th International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2009.

L. Grippo, F. Lampariello, and S. Lucidi. A nonmonotone line search technique for Newton's method. *SIAM Journal on Numerical Analysis*, 23:707–716, 1986.

T. Hazan and A. Shashua. Convergent message-passing algorithms for inference over general graphs with convex free energies. In *24th Conference on Uncertainty in Artificial Intelligence (UAI)*, 2008.

T. Heskes. Convexity arguments for efficient minimization of the Bethe and Kikuchi free energies. *Journal of Artificial Intelligence Research*, 26:153–190, 2006.

V. Jojic, S. Gould, and D. Koller. Accelerated dual decomposition for MAP inference. In *27th International Conference on Machine Learning (ICML)*, 2010.

N. Komodakis, N. Paragios, and G. Tziritas. MRF optimization via dual decomposition: Message-passing revisited. In *IEEE 11th International Conference on Computer Vision (ICCV)*, pages 1–8, 2007.

T. Meltzer, A. Globerson, and Y. Weiss. Convergent message passing algorithms - a unifying view. In *Proceedings of the 25th Conference on Uncertainty in Artificial Intelligence (UAI)*, 2009.

J. M. Mooij. libDAI: A free and open source C++ library for discrete approximate inference in graphical models. *Journal of Machine Learning Research*, 11:2169–2173, 2010.

A. Nedić and V. G. Subramanian. Approximately optimal utility maximization. In *IEEE Information Workshop on Networking and Information Theory (ITW)*, pages 206–210, 2009.

J. Nocedal. Updating quasi-Newton matrices with limited storage. *Mathematics of Computation*, 35:773–782, 1980.

M. Schmidt, E. van den Berg, M. Friedlander, and K. Murphy. Optimizing costly functions with simple constraints: A limited-memory projected quasi-Newton algorithm. In *12th International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2009.

M. J. Wainwright, T. S. Jaakkola, and A. S. Willsky. A new class of upper bounds on the log partition function. *IEEE Transactions on Information Theory*, 51:2313–2335, July 2005a.

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, 2005b.

C. Wang, Q. Liu, and X. Yang. Convergence properties of nonmonotone spectral projected gradient methods. *Journal of Computational and Applied Mathematics*, 181(1):51–66, 2005.

J. S. Yedidia, W. T. Freeman, and Y. Weiss. Understanding belief propagation and its generalizations. In *Exploring artificial intelligence in the new millennium*, pages 239–269. Morgan Kaufmann Publishers Inc., 2003.

A SUPPLEMENTARY MATERIAL

We provide here additional material that is not officially part of the paper.

A.1 PROOFS W.R.T. TREE SELECTION

We start with a general discussion, as Proposition 2 and Proposition 3 are both based on the same framework. In particular, both algorithms seek the solution to a convex optimization problem

$$\min_{\boldsymbol{\nu}\in\mathbb{T}(G)} f(\boldsymbol{\nu}),$$

where $f(\cdot)$ is a convex function of the edge occurrence probabilities $\boldsymbol{\nu}$ and $\mathbb{T}(G)$ is the so-called *spanning tree*

polytope of a graph G (e.g. Edmonds, 1971). The latter is described by a number of inequalities that is exponential in the size of G. Nonetheless, one can optimize efficiently over $\boldsymbol{\nu}$ using the *conditional gradient* or Frank-Wolfe method (e.g. Bertsekas, 1999). Here, at each iteration k, we determine a feasible descent direction $\boldsymbol{p}^{(k)}$ through the solution of the first-order Taylor expansion of $f(\cdot)$ around $\boldsymbol{\nu}^{(k)}$,

$$\min_{\boldsymbol{\nu}\in\mathbb{T}(G)} \left\{ f(\boldsymbol{\nu}^{(k)}) + \nabla f(\boldsymbol{\nu}^{(k)}) \cdot (\boldsymbol{\nu} - \boldsymbol{\nu}^{(k)}) \right\}.$$

We use $\boldsymbol{\nu}^{\star(k)}$ to denote the minimizer of the above. The feasible descent direction is then given by $\boldsymbol{p}^{(k)} = \boldsymbol{\nu}^{\star(k)} - \boldsymbol{\nu}^{(k)}$, and the next iterate is obtained as

$$\boldsymbol{\nu}^{(k+1)} = \boldsymbol{\nu}^{(k)} + \alpha^{(k)}\boldsymbol{p}^{(k)}, \quad \alpha^{(k)} \in [0,1].$$

Observe that this is equivalent to

$$\boldsymbol{\nu}^{(k+1)} = \alpha^{(k)}\boldsymbol{\nu}^{\star(k)} + (1-\alpha^{(k)})\boldsymbol{\nu}^{(k)}, \quad \alpha^{(k)} \in [0,1],$$

i.e., the new iterate is obtained as a convex combination of the previous iterate and the extreme point $\boldsymbol{\nu}^{\star(k)}$, which can be found efficiently using the minimum spanning tree (MST) algorithm with edge weights given by $\nabla f(\boldsymbol{\nu}^{(k)})$. Hence, the MST algorithm solves the linear program $\min_{\boldsymbol{\nu}\in\mathbb{T}(G)} \nabla f(\boldsymbol{\nu}^{(k)}) \cdot \boldsymbol{\nu}$ over the spanning tree polytope.

Lemma 1. *The steps taken by Algorithm 2 and Algorithm 3 are exactly of the form described above.*

Proof. To see this, observe that at each step, the current edge occurrence probabilities $\boldsymbol{\nu}^{(k)}$ are maintained through the mapping

$$\boldsymbol{\nu}^{(k)} = \boldsymbol{\nu}(\mathcal{S}^{(k)}, \boldsymbol{\rho}_s^{(k)}) = \sum_{T\in\mathcal{S}^{(k)}} \rho_s^{(k)}(T)\boldsymbol{\nu}(T),$$

where $\boldsymbol{\nu}(T) \in \mathbb{R}^{|\mathcal{E}|}$ indicates which edges are contained in T, that is, $\nu_{st}(T) = [(s,t) \in \mathcal{E}_T]$. Each step chooses $\boldsymbol{\rho}_s^{(k+1)}$ as $\mathbf{1}/(k+1)$. Equivalently, we can develop the iterate as $\boldsymbol{\rho}_s^{(k+1)} = [\alpha^{(k)}, (1-\alpha^{(k)})\boldsymbol{\rho}_s^{(k)}]$ with $\alpha^{(k)} = 1/(k+1)$. Moreover, we note that the extreme point $\boldsymbol{\nu}^{\star(k)}$ corresponds to a particular tree $T^{\star(k)}$ via the relation $\boldsymbol{\nu}^{\star(k)} = \boldsymbol{\nu}(T^{\star(k)})$. It is precisely this tree $T^{\star(k)}$ that Algorithm 2 and Algorithm 3 add to $\mathcal{S}^{(k)}$ with associated probability $\alpha^{(k)}$ at each step. But then, through the mapping between $(\mathcal{S}, \boldsymbol{\rho}_s)$ and $\boldsymbol{\nu}$, we obtain

$$\begin{aligned}\boldsymbol{\nu}^{(k+1)} &= \boldsymbol{\nu}(\mathcal{S}^{(k+1)}, \boldsymbol{\rho}_s^{(k+1)}) \\ &= \alpha^{(k)}\boldsymbol{\nu}(T^{\star(k)}) + \boldsymbol{\nu}(\mathcal{S}^{(k)}, (1-\alpha^{(k)})\boldsymbol{\rho}_s^{(k)}) \\ &= \alpha^{(k)}\boldsymbol{\nu}^{\star(k)} + (1-\alpha^{(k)})\boldsymbol{\nu}^{(k)},\end{aligned}$$

which is what we wanted to show. □

To guarantee convergence of the framework, we also need the following lemma.

Lemma 2. *For the sequence of step sizes $\{\alpha^{(k)}\}$ chosen as $\alpha^{(k)} = 1/(k+1)$, the conditional gradient algorithm converges to the global minimum of $f(\cdot)$.*

Proof (Sketch). We do not give an explicit proof here. Global convergence of a conditional gradient algorithm with $\{\alpha^{(k)}\}$ chosen as a $1/(k+1)$ is shown by Nedić and Subramanian (2009), among others. □

Note that it is also possible to choose $\alpha^{(k)}$ such that sufficient decrease is obtained at each step by imposing the Armijo condition (Bertsekas, 1999). It remains to discuss the objective functions $f(\cdot)$ optimized by Algorithm 2 and Algorithm 3.

A.1.1 Proof of Proposition 2

We wish to show that Algorithm 2 determines a sequence $\{\boldsymbol{\nu}(\mathcal{S}^{(k)}, \boldsymbol{\rho}_s^{(k)})\}$ that converges to a vector $\boldsymbol{u}$ with components given by $u_{st} = (|\mathcal{V}|-1)/|\mathcal{E}|$. To see this, consider the optimization problem

$$\min_{\boldsymbol{\nu}\in\mathbb{T}(G)} f_{\text{uni}}(\boldsymbol{\nu}), \quad f_{\text{uni}}(\boldsymbol{\nu}) \stackrel{\text{def}}{=} \|\boldsymbol{\nu} - \boldsymbol{u}\|_2^2.$$

To apply the conditional gradient algorithm, we require the gradient of the objective, which we develop as $\nabla f_{\text{uni}}(\boldsymbol{\nu}) = 2(\boldsymbol{\nu} - \boldsymbol{u})$. At each iteration k, to determine the extreme point $\boldsymbol{\nu}^{\star(k)}$, we thus solve a minimum spanning tree problem with edge weights given by $2(\boldsymbol{\nu}^{(k)} - \boldsymbol{u})$. The constant factor 2 does not affect the solution, nor does the constant vector $\boldsymbol{u}$, the components of which are all equal. Consequently, we can solve the MST problem with edge weights given by $\boldsymbol{\nu}^{(k)}$. This is exactly what Algorithm 2 does. Finally, we note that $\boldsymbol{u} \in \mathbb{T}(G)$ such that $\boldsymbol{\nu} = \boldsymbol{u}$ can be achieved. Proposition 2 then follows from Lemma 1 and Lemma 2.

A.1.2 Proof of Proposition 3

We wish to show that Algorithm 3 determines a sequence $\{\tilde{\Phi}^{(k)}\}$ converging to an upper bound $\tilde{\Phi}^\star \geq \Phi(\boldsymbol{\theta})$ that is jointly optimal over the choice of trees $\mathcal{S}$, the distribution over trees $\boldsymbol{\rho}_s$, and the tractable parameterization $\vec{\boldsymbol{\theta}}$. To see this, consider the problem

$$\min_{\boldsymbol{\nu}\in\mathbb{T}(G)} f_{\text{opt}}(\boldsymbol{\nu}), \quad f_{\text{opt}} \text{ given by (7)}.$$

It can be shown (Wainwright et al., 2005a) that $f_{\text{opt}}(\cdot)$ is convex and differentiable in $\boldsymbol{\nu}$, and that its partial derivatives are given by $\frac{\partial f_{\text{opt}}}{\partial \nu_{st}} = -I(\boldsymbol{\mu}_{st}^\star)$, where $I(\boldsymbol{\mu}_{st}^\star)$ denotes the mutual information of (s,t) given the pseudomarginals $\boldsymbol{\mu}^\star$ that maximize the objective in (7).

Note that at each iteration k, the term $\boldsymbol{\mu}^{\star(k)}$ depends on the solution of (7) given the current iterate $\boldsymbol{\nu}^{(k)}$. Consequently, at each step, the conditional gradient algorithm first determines $\boldsymbol{\mu}^{\star(k)}$, and then finds the minimum spanning tree with the weight of edge (s,t) given by $-I(\boldsymbol{\mu}_{st}^{\star(k)})$. Wainwright et al. (2005a) show that by minimizing $f_{\text{opt}}(\cdot)$ over $\boldsymbol{\nu}$, one obtains an upper bound $\tilde{\Phi}^{\star} \geq \Phi(\boldsymbol{\theta})$ that is jointly optimal over $\boldsymbol{\nu}$ and $\boldsymbol{\mu}$. Now, from Lemma 1, we conclude that Algorithm 3 takes the same steps as the conditional gradient algorithm of Wainwright et al. (2005a). The only difference lies in the fact that the edge occurrence probabilities $\boldsymbol{\nu}^{(k)}$ are implicitly maintained in terms of the mapping $\boldsymbol{\nu}(\mathcal{S}^{(k)}, \boldsymbol{\rho}_s^{(k)})$, and that the pseudomarginals $\boldsymbol{\mu}^{\star(k)}$ are (equivalently) computed using Algorithm 1, which at each step determines the parameterization $\vec{\boldsymbol{\theta}}^{\star(k)}$ that minimizes the upper bound for the current iterate $(\mathcal{S}^{(k)}, \boldsymbol{\rho}_s^{(k)})$. Furthermore, Lemma 2 guarantees convergence for our choice of step sizes. It then follows that the sequence of upper bounds $\{\tilde{\Phi}^{(k)}\}$ converges to a jointly optimal upper bound $\tilde{\Phi}^{\star}$.

A.2 AVAILABILITY OF SOFTWARE

Our SPG- and PQN-based solvers are made available as opensource software in the PhiWeave machine learning library for approximate discriminative training of graphical models at `http://phiweave.sf.net`. The library is still in its infancy, but is approaching maturity at rapid speed.

Specifically, the implementation of our solvers can be viewed online at `http://phiweave.svn.sf.net/viewvc/phiweave/trunk/src/main/scala/net/sf/phiweave/inference/SumDualDecomposition.scala?view=markup`. PhiWeave is currently lacking documentation, but the situation will improve substantially once the set of supported features has stabilized.