MRF Optimization via Dual Decomposition: Message-Passing Revisited[*]

Nikos Komodakis
MAS, Ecole Centrale Paris
komod@csd.uoc.gr

Nikos Paragios
MAS, Ecole Centrale Paris
nikos.paragios@ecp.fr

Georgios Tziritas
University of Crete
tziritas@csd.uoc.gr

Abstract

A new message-passing scheme for MRF optimization is proposed in this paper. This scheme inherits better theoretical properties than all other state-of-the-art message passing methods and in practice performs equally well/outperforms them. It is based on the very powerful technique of Dual Decomposition [1] and leads to an elegant and general framework for understanding/designing message-passing algorithms that can provide new insights into existing techniques. Promising experimental results and comparisons with the state of the art demonstrate the extreme theoretical and practical potentials of our approach.

1. Introduction

Discrete MRF optimization is of fundamental importance to computer vision. Given a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with nodes $\mathcal{V}$ and edges $\mathcal{E}$, the goal is to assign a label l_p (from a discrete label set $\mathcal{L}$) to each $p \in \mathcal{V}$, so that the MRF energy is minimized. This means solving the following problem:

$$\min \sum\nolimits_{p\in\mathcal{V}} \theta_p(l_p) + \sum\nolimits_{pq\in\mathcal{E}} \theta_{pq}(l_p, l_q). \tag{1}$$

Here, $\theta_p(\cdot)$, $\theta_{pq}(\cdot,\cdot)$ represent the unary and pairwise MRF potential functions respectively.

Currently, two classes of methods are the most prominent ones in MRF optimization: those based on graph-cuts [5, 2], and those based on message-passing. Regarding the latter class, a significant advance took place recently with the introduction of the so-called tree-reweighted message passing (TRW) algorithms [7, 3, 8]. Although they appear similar to the max-product Belief Propagation (BP) algorithm [6] on the surface, these methods are in fact quite different, as well as far more powerful. They rely on the following integer linear programming formulation of (1):

$$\begin{aligned} \min_{\mathbf{x}} \quad & E(\boldsymbol{\theta}, \mathbf{x}) = \boldsymbol{\theta}\cdot\mathbf{x} = \sum_{p\in\mathcal{V}} \boldsymbol{\theta_p}\cdot\mathbf{x}_p + \sum_{pq\in\mathcal{E}} \boldsymbol{\theta_{pq}}\cdot\mathbf{x}_{pq} \\ \text{s.t.} \quad & \mathbf{x} \in \mathcal{X}^{\mathcal{G}} \end{aligned} \tag{2}$$

Here, the vector $\boldsymbol{\theta}=\{\{\boldsymbol{\theta}_p\}, \{\boldsymbol{\theta}_{pq}\}\}$ of MRF-parameters consists of all unary $\boldsymbol{\theta}_p=\{\theta_p(\cdot)\}$ and pairwise $\boldsymbol{\theta}_{pq}=\{\theta_{pq}(\cdot,\cdot)\}$ vectorized potential functions, whereas $\mathbf{x} = \{\{\mathbf{x}_p\}, \{\mathbf{x}_{pq}\}\}$ is the vector of MRF-variables consisting of all unary subvectors $\mathbf{x}_p = \{x_p(\cdot)\}$ and pairwise subvectors $\mathbf{x}_{pq} = \{x_{pq}(\cdot,\cdot)\}$. The MRF-variables are $\{0,1\}$-variables that satisfy: $x_p(l) = 1 \Leftrightarrow$ label l is assigned to p, while $x_{pq}(l,l') = 1 \Leftrightarrow$ labels l, l' are assigned to p, q. To enforce these conditions, it suffices that vector $\mathbf{x}$ lies in the set $\mathcal{X}^{\mathcal{G}}$. For any graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, that set is defined as follows:

$$\mathcal{X}^{\mathcal{G}} = \left\{ \mathbf{x} \;\middle|\; \begin{array}{l} \sum_{l\in\mathcal{L}} x_p(l) = 1, \quad \forall\, p \in \mathcal{V} \\ \sum_{l'\in\mathcal{L}} x_{pq}(l,l') = x_p(l), \quad \forall\, (pq,l) \in \mathcal{E}\times\mathcal{L} \\ x_p(l) \in \{0,1\}, \quad x_{pq}(l,l') \in \{0,1\} \end{array} \right\}$$

The first constraints simply ensure that a unique label is assigned to each p, while the second constraints enforce consistency between $x_p(\cdot)$, $x_q(\cdot)$ and $x_{pq}(\cdot,\cdot)$, since they ensure that if $x_p(l) = x_q(l') = 1$, then $x_{pq}(l,l') = 1$ as well.

However, despite that TRW algorithms rely on formulation (2) in order to optimize an MRF, the key property that characterizes all these methods is that they do not actually attempt to minimize the energy of that MRF directly. Instead, their goal is to maximize a lower bound on this energy. To be more rigorous, instead of directly addressing the MRF problem, *i.e.* problem (2), these methods try to solve a dual problem. Specifically, the key idea behind them is to solve the dual to the LP relaxation of (2). Any feasible solution to this dual is a lower bound on the MRF energy, and so, by solving the dual, these methods aim to maximize this bound. Based on how good the resulting lower bound from the dual is, a solution to the primal, *i.e.* the MRF problem (2), is then extracted. To our surprise, however, we found out that, although the key to success of all TRW algorithms is solving that dual, none of them can actually guarantee that. In fact, as shown in [3], there are cases for which this is not true.

Motivated by this fact, we propose here a new message-passing MRF-optimization scheme, called DD-MRF (Dual Decomposition MRF). To the best of our knowledge, DD-MRF is the first such scheme that can also solve the above mentioned dual LP (*i.e.*, maximize the lower bound), which is the driving force behind all TRW algorithms. It enjoys better theoretical properties than TRW methods, thus providing new insights into these techniques, while it has given

[*]This work was partially supported from the French ANR-Blanc grant SURF (2005-2008) and Platon (2006-2007).

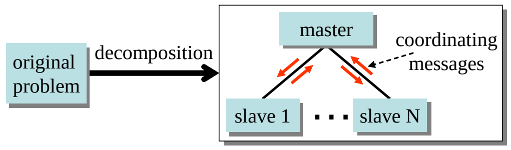


Fig. 1: The original (possibly difficult) optimization problem decomposes into easier subproblems (called the *slaves*) that are coordinated by a *master* problem via message exchanging.

very good experimental results on a variety of computer vision tasks. Moreover, it is derived based on very general principles, and thus leads to a simple, powerful and elegant framework for understanding/designing message-passing algorithms, that revisits some of the choices of previous methods, which we consider as another important contribution of this work. In particular, the theoretical setting of our method rests on the technique of *dual decomposition* [1]. This is an extremely powerful and general technique, well known to people in optimization. As a result of introducing this technique, we manage to reduce MRF optimization to a simple projected-subgradient method. This connection can prove to be of great benefit, since it could motivate new research and pave the way for better MRF optimization methods in the future.

The remainder of the paper is organized as follows: we briefly review dual decomposition in §2. The DD-MRF algorithm is then presented in §3, while some of its theoretical properties are analyzed in §4. Experimental results are shown in §5, while we finally conclude in §6.

2. Dual decomposition

The main idea behind decomposition is surprisingly simple: first decompose your original complex problem into smaller solvable subproblems and then extract a solution by cleverly combining the solutions from these subproblems. Although simple as a concept, decomposition is extremely general and powerful, and has been used for solving many large or complex optimization problems. Typically, during decomposition one has to define 2 things: what the subproblems will be (also referred to as *slave* problems), as well as a so-called *master* problem that will act as a coordinator between the slave problems (see Fig. 1). In addition, one can either decompose the original problem (primal decomposition) or its Lagrangian dual (dual decomposition).

Here, we will only consider the latter type and give a simple example just to illustrate how it works. To this end, consider the following problem (where $\mathbf{x}$ denotes a vector):

$$\begin{array}{ll} \min_{\boldsymbol{x}} & \sum_i f^i(\boldsymbol{x}) \\ \text{s.t.} & \boldsymbol{x} \in \mathcal{C} \end{array}$$

We assume that separately minimizing each $f^i(\cdot)$ over vector $\boldsymbol{x}$ is easy, but minimizing $\sum_i f^i(\cdot)$ is hard. Using auxiliary variables $\{\mathbf{x}^i\}$, we thus transform our problem into:

$$\begin{array}{ll} \min_{\{\boldsymbol{x}^i\},\boldsymbol{x}} & \sum_i f^i(\boldsymbol{x}^i) \\ \text{s.t.} & \boldsymbol{x}^i \in \mathcal{C},\ \boldsymbol{x}^i = \boldsymbol{x} \end{array}$$

If the coupling constraints $\boldsymbol{x}^i = \boldsymbol{x}$ were absent, the problem would decouple. We therefore relax them (via multipliers $\{\boldsymbol{\lambda}^i\}$) and form the following Lagrangian dual function:

$$\begin{aligned} g(\{\boldsymbol{\lambda}^i\}) &= \min_{\{\boldsymbol{x}^i \in \mathcal{C}\},\boldsymbol{x}} \sum_i f^i(\boldsymbol{x}^i) + \sum_i \boldsymbol{\lambda}^i \cdot (\boldsymbol{x}^i - \boldsymbol{x}) \\ &= \min_{\{\boldsymbol{x}^i \in \mathcal{C}\},\boldsymbol{x}} \sum_i [f^i(\boldsymbol{x}^i) + \boldsymbol{\lambda}^i \cdot \boldsymbol{x}^i] - (\sum_i \boldsymbol{\lambda}^i)\boldsymbol{x} \end{aligned}$$

We next eliminate $\boldsymbol{x}$ from $g(\{\boldsymbol{\lambda}^i\})$ by minimizing over that variable. This just implies $\{\boldsymbol{\lambda}^i\} \in \Lambda = \big\{\{\boldsymbol{\lambda}^i\} | \sum_i \boldsymbol{\lambda}^i = 0\big\}$ (it is easy to check that if $\{\boldsymbol{\lambda}^i\} \notin \Lambda$ then $g(\{\boldsymbol{\lambda}^i\}) = -\infty$). Therefore, the resulting dual function becomes equal to:

$$g(\{\boldsymbol{\lambda}^i\}) = \min_{\{\boldsymbol{x}^i \in \mathcal{C}\}} \sum_i [f^i(\boldsymbol{x}^i) + \boldsymbol{\lambda}^i \cdot \boldsymbol{x}^i]$$

We can now setup a Lagrangian dual problem, *i.e.* maximize $g(\{\boldsymbol{\lambda}^i\})$ over the feasible set Λ, or

$$\max_{\{\boldsymbol{\lambda}^i\} \in \Lambda} \ g(\{\boldsymbol{\lambda}^i\}) = \sum_i g^i(\boldsymbol{\lambda}^i), \tag{3}$$

where this dual problem (also called the master) has now decoupled into the following slave problems (one per $g^i(\boldsymbol{\lambda}^i)$):

$$\begin{array}{ll} g^i(\boldsymbol{\lambda}^i) = & \min_{\boldsymbol{x}^i} \ f^i(\boldsymbol{x}^i) + \boldsymbol{\lambda}^i \cdot \boldsymbol{x}^i \\ & \text{s.t.} \quad \boldsymbol{x}^i \in \mathcal{C} \end{array} \tag{4}$$

Problem (3) is always convex and can be solved with the projected subgradient method (since $g(\cdot)$ is typically not differentiable). According to that method, at each iteration the dual variables $\{\boldsymbol{\lambda}^i\}$ are updated as $\boldsymbol{\lambda}^i \leftarrow \big[\boldsymbol{\lambda}^i + \alpha_t \nabla g^i(\boldsymbol{\lambda}^i)\big]_\Lambda$. Here, α_t denotes a positive multiplier (for the t-th iteration), $[\,\cdot\,]_\Lambda$ denotes projection onto the set Λ, while $\nabla g^i(\boldsymbol{\lambda}^i)$ denotes the subgradient of $g^i(\cdot)$. It thus remains to compute this subgradient, for which we can use the following well-known lemma:

Lemma. *Let* $q(\boldsymbol{\lambda}) = \min_{i \in \mathcal{I}} \{\mathbf{d}_i \cdot \boldsymbol{\lambda} + \mathbf{b}_i\}$. *Any* $\mathbf{d}_i$ *with* $i \in \mathcal{I}_{\boldsymbol{\lambda}} = \{i | \mathbf{d}_i \cdot \boldsymbol{\lambda} + \mathbf{b}_i = q(\boldsymbol{\lambda})\}$ *is a subgradient of* $q(\cdot)$.

Therefore, $\nabla g^i(\boldsymbol{\lambda}^i) = \bar{\boldsymbol{x}}^i$, where $\bar{\boldsymbol{x}}^i$ is any optimal solution to the i-th slave problem (4). To summarize, what happens in essence is that a solution to the dual is obtained by operating at two levels. At the higher level, the master problem (3) coordinates the slaves simply by updating $\{\boldsymbol{\lambda}^i\}$ based on the currently extracted optimal solutions $\{\bar{\boldsymbol{x}}^i\}$. And then, at the lower level, based on the updated $\{\boldsymbol{\lambda}^i\}$ each of the decoupled slave problems (4) is again solved independently to generate a new $\bar{\boldsymbol{x}}^i$ for the next iteration.

3. MRF optimization via Dual Decomposition

In this section, we will describe how we can apply the dual decomposition method to the case of the MRF optimization problem. To prepare the reader, our goal will be to decompose our original MRF problem, which is NP-hard since it is defined on a general graph $\mathcal{G}$, into a set of easier MRF subproblems defined on trees $T \subset \mathcal{G}$. To this end, we will first need to transform our problem into a more appropriate form by introducing a set of auxiliary variables.

In particular, let $\mathcal{T}$ be a set of subtrees of graph $\mathcal{G}$. The only requirement for $\mathcal{T}$ is that its trees cover (at least once) every node and edge of graph $\mathcal{G}$. For each tree $T \in \mathcal{T}$ we will then imagine that there is a smaller MRF defined just on the nodes and edges of tree T, and we will associate to it a vector of MRF-parameters $\boldsymbol{\theta}^T$, as well as a vector of MRF-variables $\mathbf{x}^T$ (these have similar form to vectors $\boldsymbol{\theta}$ and $\mathbf{x}$ of the original MRF, except that they are smaller in size). MRF-variables contained in vector $\mathbf{x}^T$ will be redundant, since we will initially assume that they are all equal to the corresponding MRF-variables in vector $\mathbf{x}$, *i.e.* it will hold $\mathbf{x}^T = \mathbf{x}_{|T}$, where $\mathbf{x}_{|T}$ represents the subvector of $\mathbf{x}$ containing MRF-variables only for nodes and edges of tree T. In addition, all the vectors $\{\boldsymbol{\theta}^T\}$ will be defined so that they satisfy the following conditions:

$$\sum_{T \in \mathcal{T}(p)} \boldsymbol{\theta}^T_p = \boldsymbol{\theta}_p, \qquad \sum_{T \in \mathcal{T}(pq)} \boldsymbol{\theta}^T_{pq} = \boldsymbol{\theta}_{pq}. \tag{5}$$

Here, $\mathcal{T}(p)$ and $\mathcal{T}(pq)$ denote all trees of $\mathcal{T}$ that contain node p and edge pq respectively. *E.g.*, to ensure (5), one can simply set: $\boldsymbol{\theta}^T_p = \frac{\boldsymbol{\theta}_p}{|\mathcal{T}(p)|}$ and $\boldsymbol{\theta}^T_{pq} = \frac{\boldsymbol{\theta}_{pq}}{|\mathcal{T}(pq)|}$. Due to (5) and the fact that $\mathbf{x}^T = \mathbf{x}_{|T}$, energy $E(\boldsymbol{\theta}, \mathbf{x})$ thus decomposes into the energies $E(\boldsymbol{\theta}^T, \mathbf{x}^T) = \boldsymbol{\theta}^T \cdot \mathbf{x}^T$, or

$$E(\boldsymbol{\theta}, \mathbf{x}) = \sum_{T \in \mathcal{T}} E(\boldsymbol{\theta}^T, \mathbf{x}^T) \tag{6}$$

Also, by using the auxiliary variables $\mathbf{x}^T$, it is trivial to see that our original constraints $\mathbf{x} \in \mathcal{X}^{\mathcal{G}}$ reduce to:

$$\mathbf{x}^T \in \mathcal{X}^T, \quad \mathbf{x}^T = \mathbf{x}_{|T}, \qquad \forall T \in \mathcal{T} \tag{7}$$

Hence, our original MRF problem becomes equivalent to:

$$\begin{array}{ll} \min\limits_{\{\mathbf{x}^T\}, \mathbf{x}} & \sum_{T \in \mathcal{T}} E(\boldsymbol{\theta}^T, \mathbf{x}^T) \\ \text{s.t.} & \mathbf{x}^T \in \mathcal{X}^T, \quad \forall T \in \mathcal{T} \\ & \mathbf{x}^T = \mathbf{x}_{|T}, \quad \forall T \in \mathcal{T} \end{array} \tag{8}$$

It is clear that without constraints $\mathbf{x}^T = \boldsymbol{x}_{|T}$, this problem would decouple into a series of smaller MRF problems (one per tree $\mathcal{T}$). Therefore, it is natural to relax these coupling constraints (by introducing Lagrange multipliers $\boldsymbol{\lambda}^T = \{\{\boldsymbol{\lambda}^T_p\}, \{\boldsymbol{\lambda}^T_{pq}\}\}$) and form the Lagrangian dual function as:

$$\begin{aligned} g(\{\boldsymbol{\lambda}^T\}) &= \min_{\{\mathbf{x}^T \in \mathcal{X}^T\}, \mathbf{x}} \sum_{T \in \mathcal{T}} E(\boldsymbol{\theta}^T, \mathbf{x}^T) + \sum_{T \in \mathcal{T}} \boldsymbol{\lambda}^T \cdot (\mathbf{x}^T - \mathbf{x}_{|T}) \\ &= \min_{\{\mathbf{x}^T \in \mathcal{X}^T\}, \mathbf{x}} \sum_{T \in \mathcal{T}} E(\boldsymbol{\theta}^T + \boldsymbol{\lambda}^T, \mathbf{x}^T) - \sum_{T \in \mathcal{T}} \boldsymbol{\lambda}^T \cdot \mathbf{x}_{|T} \end{aligned}$$

Vector $\mathbf{x}$ can be eliminated from $g(\{\boldsymbol{\lambda}^T\})$ by directly minimizing over it, which simply imposes the constraint $\{\boldsymbol{\lambda}^T\} \in \Lambda$,[1] where the feasible set Λ is now defined as:

$$\Lambda = \left\{ \{\boldsymbol{\lambda}^T\} \;\middle|\; \sum_{T \in \mathcal{T}(p)} \boldsymbol{\lambda}^T_p = 0, \; \sum_{T \in \mathcal{T}(pq)} \boldsymbol{\lambda}^T_{pq} = 0 \right\},$$

while the resulting Lagrangian dual function simplifies to:

$$g(\{\boldsymbol{\lambda}^T\}) = \min_{\{\mathbf{x}^T \in \mathcal{X}^T\}} \sum_{T \in \mathcal{T}} E(\boldsymbol{\theta}^T + \boldsymbol{\lambda}^T, \mathbf{x}^T)$$

We can now setup a dual problem, *i.e.* maximize the above dual function $g(\{\boldsymbol{\lambda}^T\})$ over its feasible set Λ, or

$$\max_{\{\boldsymbol{\lambda}^T\} \in \Lambda} g(\{\boldsymbol{\lambda}^T\}) = \sum_{T \in \mathcal{T}} g^T(\boldsymbol{\lambda}^T), \tag{9}$$

where each function $g^T(\cdot)$ is defined as:

$$\begin{array}{lll} g^T(\boldsymbol{\lambda}^T) = & \min\limits_{\mathbf{x}^T} & E(\boldsymbol{\theta}^T + \boldsymbol{\lambda}^T, \mathbf{x}^T) \\ & \text{s.t.} & \mathbf{x}^T \in \mathcal{X}^T. \end{array} \tag{10}$$

Problem (9) has thus become our master problem, and each slave problem (10) simply amounts to optimizing an MRF over a tree $T \subset \mathcal{G}$, *i.e.* a much easier problem. For optimizing the master, we will use the projected subgradient method. As explained in §2, at each iteration of this method the dual variables $\boldsymbol{\lambda}^T$ must first be updated as $\boldsymbol{\lambda}^T \leftarrow \boldsymbol{\lambda}^T + \alpha_t \nabla g^T(\boldsymbol{\lambda}^T)$. Based on lemma 2, the subgradient of $g^T(\cdot)$ equals $\nabla g^T(\boldsymbol{\lambda}^T) = \bar{\mathbf{x}}^T$, where $\bar{\mathbf{x}}^T$ represents any optimal solution to slave MRF (10), and so the above update amounts to setting $\boldsymbol{\lambda}^T \leftarrow \boldsymbol{\lambda}^T + \alpha_t \bar{\mathbf{x}}^T$. It then only remains to project the resulting $\{\boldsymbol{\lambda}^T\}$ onto the feasible set Λ. Due to the definition of Λ, this projection reduces to subtracting the average vector $\frac{\sum_{T \in \mathcal{T}(p)} \boldsymbol{\lambda}^T_p}{|\mathcal{T}(p)|}$ from each $\boldsymbol{\lambda}^T_p$ (so that $\sum_{T \in \mathcal{T}(p)} \boldsymbol{\lambda}^T_p = 0$), as well as subtracting the average vector $\frac{\sum_{T \in \mathcal{T}(pq)} \boldsymbol{\lambda}^T_{pq}}{|\mathcal{T}(pq)|}$ from each $\boldsymbol{\lambda}^T_{pq}$ (so that $\sum_{T \in \mathcal{T}(pq)} \boldsymbol{\lambda}^T_{pq} = 0$). By aggregating all of the above operations, a projected subgradient update is easily seen to reduce to $\boldsymbol{\lambda}^T_p \mathrel{+}= \boldsymbol{\Delta\lambda}^T_p$, $\boldsymbol{\lambda}^T_{pq} \mathrel{+}= \boldsymbol{\Delta\lambda}^T_{pq}$ where:

$$\boldsymbol{\Delta\lambda}^T_p = \alpha_t \cdot \left(\bar{\mathbf{x}}^T_p - \frac{\sum_{T' \in \mathcal{T}(p)} \bar{\mathbf{x}}^{T'}_p}{|\mathcal{T}(p)|} \right) \tag{11}$$

$$\boldsymbol{\Delta\lambda}^T_{pq} = \alpha_t \cdot \left(\bar{\mathbf{x}}^T_{pq} - \frac{\sum_{T' \in \mathcal{T}(pq)} \bar{\mathbf{x}}^{T'}_{pq}}{|\mathcal{T}(pq)|} \right) \tag{12}$$

Of course, each $\boldsymbol{\lambda}^T$ is only used for defining the MRF-parameters $\boldsymbol{\theta}^T + \boldsymbol{\lambda}^T$ of the slave MRF in (10). Hence, instead of updating the Lagrange multipliers $\{\boldsymbol{\lambda}^T\}$ at each

[1] It is easy to see that if $\{\boldsymbol{\lambda}^T\} \notin \Lambda$, then $g(\{\boldsymbol{\lambda}^T\}) = -\infty$.

```
– Solve slave MRFs using max-product BP, i.e.:
    $\forall T \in \mathcal{T}$, compute $\bar{\mathbf{x}}^T = \operatorname{argmin}_{\mathbf{x}^T \in \mathcal{X}^T} E(\boldsymbol{\theta}^T, \mathbf{x}^T)$
– Update parameters for slave MRFs using $\{\bar{\mathbf{x}}^T\}$, i.e.:
    $\forall T \in \mathcal{T}$, $\boldsymbol{\theta}_p^T \mathrel{+}= \boldsymbol{\Delta\lambda}_p^T$, $\boldsymbol{\theta}_{pq}^T \mathrel{+}= \boldsymbol{\Delta\lambda}_{pq}^T$
```

Fig. 2: A basic update during the projected subgradient algorithm.

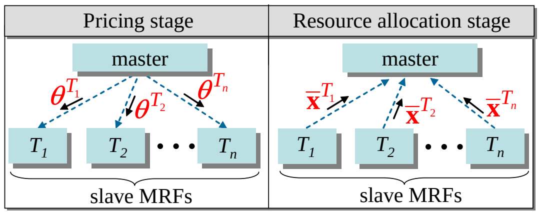


Fig. 3: Dual decomposition scheme for MRF optimization **Left:** Based on the current optimal solutions $\{\bar{\mathbf{x}}^T\}$ (*i.e.* the current resource allocation), the master assigns new MRF potentials $\{\boldsymbol{\theta}^T\}$ (*i.e.* new prices) to the slave MRFs. **Right:** Based on these new potentials, the slave MRFs immediately respond to the master by sending to him new optimal solutions $\{\bar{\mathbf{x}}^T\}$ (*i.e.* by readjusting their resource allocation).

iteration, one can choose to directly update the MRF-parameters $\{\boldsymbol{\theta}^T\}$, *i.e.*, set $\boldsymbol{\theta}_p^T \mathrel{+}= \boldsymbol{\Delta\lambda}_p^T$, $\boldsymbol{\theta}_{pq}^T \mathrel{+}= \boldsymbol{\Delta\lambda}_{pq}^T$. In this manner, the need for storing the dual variables $\{\boldsymbol{\lambda}^T\}$ is avoided. This is actually how the pseudocode in Fig. 2 was formed, describing one basic update during the resulting subgradient algorithm.

3.1. Analysis of the algorithm

Let us now briefly summarize how the algorithm in Fig. 2 works. Like most other dual decomposition techniques, it operates on two levels (see Fig. 3). At the lower level, it has to solve each one of the decoupled slave problems (10). In this case, the slave problems turn out to be MRF optimization problems for tree-structured graphs. There exists one such MRF for each tree $T \in \mathcal{T}$, and its MRF-parameters are specified by the vector $\boldsymbol{\theta}^T$. Since the underlying graphs for all slave MRFs are tree-structured, these are easy problems to solve. *E.g.*, one can use the max-product algorithm to estimate an exact optimal solution $\bar{\mathbf{x}}^T$ for each $T \in \mathcal{T}$. At the higher level, on the other hand, there exists the master problem, whose sole mission is to coordinate the slave problems so that the dual function in (9) is maximized. To this end, it thus has to update the MRF-parameters $\{\boldsymbol{\theta}^T\}$ of all slave MRFs, based on the optimal solutions $\{\bar{\mathbf{x}}^T\}$ that have been estimated previously at the current iteration (strictly speaking, the master is responsible for updating the dual variables, *i.e.* the Lagrange multipliers $\{\boldsymbol{\lambda}^T\}$, but, as already explained, this is equivalent to updating the MRF-parameters $\{\boldsymbol{\theta}^T\}$ instead).

To gain a better understanding of how the master problem tries to coordinate the slave MRFs, let us now consider a node p in our original graph $\mathcal{G}$ and let us also assume that, during the current iteration, node p is assigned the same label, say l_p, by all slave MRFs. This means that, for each $T \in \mathcal{T}(p)$, the vector $\bar{\mathbf{x}}_p^T$ will have the following form: $\bar{x}_p^T(l) = 1$ if $l = l_p$, whereas $\bar{x}_p^T(l) = 0$ if $l \neq l_p$. All these vectors will therefore coincide with each other and so $\boldsymbol{\Delta\lambda}_p^T = \mathbf{0}$. Any vector $\boldsymbol{\theta}_p^T$ will thus remain untouched during the current iteration, which, in other words, means that if all slave MRFs agree on a node p, then the master problem does not modify the unary potentials associated to that node.

On the other hand, let us assume that not all slave MRFs assign the same label to p. For simplicity, let us assume that p belongs just to two trees, say T_1, T_2, and let the corresponding slave MRFs assign labels l_1, l_2 to that node (with $l_1 \neq l_2$). It is then easy to check that the following update of the vectors $\boldsymbol{\theta}_p^{T_1}$, $\boldsymbol{\theta}_p^{T_2}$ will take place:

$$\theta_p^{T_1}(l) \mathrel{+}= \begin{cases} +\frac{\alpha_t}{2} & \text{if } l = l_1 \\ -\frac{\alpha_t}{2} & \text{if } l = l_2 \\ 0 & \text{otherwise} \end{cases}, \quad \theta_p^{T_2}(l) \mathrel{+}= \begin{cases} -\frac{\alpha_t}{2} & \text{if } l = l_1 \\ +\frac{\alpha_t}{2} & \text{if } l = l_2 \\ 0 & \text{otherwise} \end{cases}$$

As can be seen, what happens is that the master tries to readjust the unary potentials for node p at T_1, T_2, so that a common label assignment to that node (by both slave MRFs) has higher chances during the next iteration, *i.e.* the master encourages slave MRFs to agree on a common label for p. As a result, all slave MRFs will agree on more and more nodes, as the algorithm progresses. Note, however, that this agreement is not enforced explicitly by the algorithm.

The above behavior is typical in dual decomposition schemes. In fact, due to an economic interpretation, dual decomposition corresponds to what is also known as resource allocation via pricing. According to this interpretation, we can think of the primal variables $\{\mathbf{x}^T\}$ as amounts of resources consumed by the slave problems, with variables $\mathbf{x}^T$ representing the amount of resources consumed by the MRF problem for tree T. In dual decomposition, the master algorithm never sets these amounts explicitly. Instead, it just sets the prices, *i.e.* the variables $\{\boldsymbol{\theta}^T\}$ in our case, for the resources. Then, based on these prices, each slave MRF has to independently decide how many resources it will use. Of course, the prices do not remain static, but are adjusted at every iteration by the master algorithm. This adjustment is naturally done as follows: prices for overutilized resources are increased, whereas prices for underutilized resources are decreased.

At this point, it is also worth noting some of the resulting differences between DD-MRF and existing TRW algorithms. These differences are useful, since they reveal some of the algorithmic choices of TRW algorithms that are revisited by DD-MRF. *E.g.*, all TRW algorithms use the tree min-marginals in order to update the dual variables

$\{\boldsymbol{\theta}^T\}$. DD-MRF, however, relies solely on the optimal solutions $\bar{\mathbf{x}}^T$ for that task. This also implies that no tree min-marginals have to be explicitly computed by DD-MRF. Furthermore, contrary to TRW algorithms, which modify all dual variables (either sequentially or in parallel) at each iteration, DD-MRF modifies a vector, *e.g.*, $\boldsymbol{\theta}_p^T$ of dual variables at a node p only if the slave MRFs disagree about that node's label, which is another important difference.

Before proceeding, we should also note that, since no Lagrange multipliers $\{\boldsymbol{\lambda}^T\}$ need to be stored (as $\{\boldsymbol{\theta}^T\}$ can be updated directly), DD-MRF has similar memory requirements to the belief propagation algorithm. In fact, any of the recently proposed techniques for improving the memory usage of BP, apply here as well [3].

3.2. Obtaining primal solutions

Let us now briefly recapitulate what we have accomplished so far. We wanted to find a solution to our original MRF problem (2), or equivalently to the primal problem (8). To this end, we have opted to relax some of the complicating constraints in (8) and solve the resulting Lagrangian dual, by decomposing it into easier subproblems (in fact, as we shall prove in the next section, the resulting Lagrangian dual is equivalent to the linear programming relaxation of the original MRF problem, *i.e.* it is the same problem that all TRW algorithms are attempting to solve). What still remains to be done is to obtain a feasible primal solution to our initial problem, *i.e.* to the MRF problem, based on the estimated solution from the Lagrangian dual.

The above situation is typical for schemes with Lagrangian relaxation. The Lagrangian solutions will in general be infeasible with respect to the original primal, *i.e.* the one without relaxed constraints. Yet, they will usually be nearly feasible, since large constraints violations got penalized. Hence, one may construct feasible solutions by, *e.g.*, correcting the minor infeasibilities of the Lagrangian solutions, which implies that the cost of the resulting solutions will not be far from the optimum. In fact, one usually constructs many feasible solutions in this manner (the more the better) and chooses the best one at the end.

In our case, for instance, we can take advantage of the optimal solutions $\{\bar{\mathbf{x}}^T\}$ that were generated for the slave problems. Recall that each $\bar{\mathbf{x}}^T$ is a $\{0,1\}$ vector, which essentially specifies an optimal labeling for a slave MRF at tree T. As explained in §3.1, these labelings will typically agree on all but a few of the MRF nodes (if they agree everywhere, they are equal to the MRF optimal solution). Due to this fact, many good primal solutions are expected to be constructed by using these labelings. Moreover, this can be done very easily. *E.g.*, if every $T \in \mathcal{T}$ is a spanning tree, then each $\bar{\mathbf{x}}^T$ directly specifies a feasible solution to the MRF problem.

Of course, there are many other possible ways of getting good feasible primal solutions. One such way, that we found to work well in practice, was to use the messages exchanged during the max-product algorithms (for the slave MRFs), since these messages contain valuable information. *E.g.*, a heuristic similar to the one proposed in [3] can be used for this purpose.

4. Theoretical properties

As already explained, our method tries to solve problem (9), which is the Lagrangian relaxation of problem (8). The subject of the next theorem is to show that this is equivalent to trying to solve the Linear Programming (LP) relaxation of problem (2).

Theorem 1. *Lagrangian relaxation* (9) *is equivalent to the LP relaxation of* (2), i.e. *the LP relaxation of the original integer programming formulation for the MRF problem.*

Sketch of proof. To form the Lagrangian relaxation, we relaxed constraints $\mathbf{x}_p^T = \mathbf{x}_p$ of (8), but we kept constraints $\mathbf{x}^T \in \mathcal{X}^T$. The Lagrangian dual is then known to be equivalent to the following relaxation of (8):

$$\min_{\{\mathbf{x}^T\},\mathbf{x}} \{E(\mathbf{x},\boldsymbol{\theta}) \mid \mathbf{x}_p^T = \mathbf{x}_p,\ \mathbf{x}^T \in \textsc{ConvexHull}(\mathcal{X}^T)\}$$

For a tree T, however, the set $\textsc{ConvexHull}(\mathcal{X}^T)$ will not change if we modify $\mathcal{X}^T$ by replacing the $\{0,1\}$ constraints with $\mathbf{x}^T \geq 0$. Based on this fact, the theorem follows trivially. □

The above theorem certifies that our method tries to solve exactly the same problem as all state-of-the-art tree-reweighted message-passing algorithms, such as TRW-T, TRW-E or TRW-S. However, unlike those algorithms, which can only guarantee a local optimum in general, an important advantage of our method is that it can provably compute the global optimum of that problem. This is an immediate result of the fact that we are using the subgradient algorithm, which is a very well studied technique in optimization, with a vast literature devoted to it. Here, we simply state two of the simplest theorems related to it [1].

Theorem 2. *If the sequence of multiplies* $\{\alpha_t\}$ *satisfies* $\alpha_t \geq 0$, $\lim_{t\to\infty} \alpha_t = 0$, $\sum_{t=0}^{\infty} \alpha_t = \infty$, *then the subgradient algorithm converges to the optimal solution of* (9).

In fact, one can even make the following statement:

Theorem 3. *The distance of the current solution* $\{\boldsymbol{\theta}^T\}$ *to the optimal solution, say,* $\{\bar{\boldsymbol{\theta}}^T\}$ *decreases at every iteration.*

State-of-the-art tree-reweighted (TRW) max-product algorithms can also provide certain correctness guarantees regarding their fixed points. One such example is the strong *tree agreement* (TA) condition that was first introduced in

[7]. If a TRW fixed point, say $\{\bar{\boldsymbol{\theta}}^T\}$, satisfies TA, an optimal solution to the original MRF problem can then be extracted. A much more general condition was later introduced in [3], called the *weak tree agreement* (WTA). This condition has also been used to provide further optimality results for TRW algorithms [4]. We next show that our method provides a generalization of the WTA condition (and hence of TA as well), in the sense that any solution of our algorithm satisfies the WTA condition (but, as we shall see in §5, the converse is not true, *i.e.*, a solution $\{\bar{\boldsymbol{\theta}}^T\}$ satisfying WTA is not necessarily optimal with respect to the Lagrangian dual problem (9)).

Theorem 4. *Any solution obtained by our method satisfies the WTA condition.*

Sketch of proof. Let $\{\bar{\boldsymbol{\theta}}^T\}$ be a solution generated by our algorithm. Let us suppose it does not satisfy WTA. One can then show that $\{\bar{\boldsymbol{\theta}}^T\}$ can be perturbed to give a solution that achieves a higher objective value for the Lagrangian dual (9). This is impossible, however, since, by theorem 2 above, $\{\bar{\boldsymbol{\theta}}^T\}$ is already an optimal solution to (9) □

The above theorem implies that all optimality results related to WTA carry over to our algorithm. Here we simply state just one of them [4]:

Theorem 5. *For binary MRFs with submodular energies, our method computes a globally optimal solution.*

5. Experimental results

Here we will present some experimental results produced by our method. We will also compare DD-MRF to existing TRW algorithms. These are the TRW-T and TRW-E algorithms presented in [7], as well the TRW-S algorithm presented in [3]. The only difference between TRW-E and TRW-S is that the former algorithm updates its messages in parallel, whereas TRW-S updates messages in a sequential order. Furthermore, since TRW-E did worse than the other TRW algorithms in our experiments, no results for TRW-E will be shown, so as to also keep the plots cleaner.

We have first tested our method on the task of interactive binary image segmentation. In this case, the unary MRF potentials were set according to the log-likelihood of a pixel belonging either to foreground or background (these likelihoods were learned based on user specified masks), whereas the pairwise potentials were set using a standard Potts model. According to theorem 5, DD-MRF should be able to find the global optimum in this case and so the main goal of this experiment was to confirm this fact. 10 natural images were thus segmented and Fig. 4 shows a typical plot of how the MRF energy (*i.e.* the cost of the primal problem) varies during a segmentation test. We have also plotted the cost of the dual problem (9), since this cost forms a lower bound on the minimum MRF energy. As can be seen, DD-MRF manages to extract the global optimum, since the primal-dual gap (*i.e.* the difference between the primal cost and the dual cost) reaches 0 at the end. Another way to verify this, is by using the max-flow algorithm to compute the optimal solution.

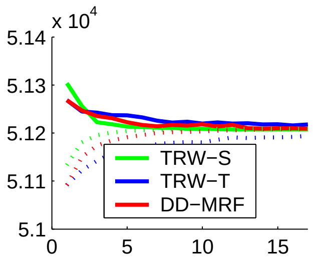
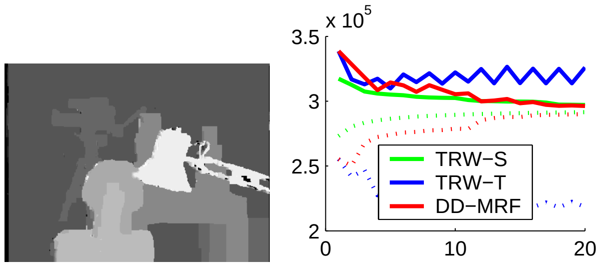


Fig. 4: Plots for the binary segmentation problem. Solid curves represent the MRF energy per iteration (these curves thus form an upper bound on the minimum MRF energy), whereas dashed curves represent the cost of the Lagrangian dual (9) (*i.e.* form lower bounds on that energy).

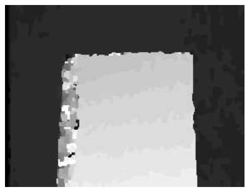


(a) Estimated disparity (b) Energy and lower bound plots

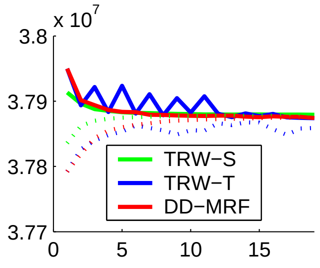
Fig. 5: *Tsukuba* results.

We have also tested our method on stereo matching. In Fig. 5(a), we show the disparity produced by DD-MRF for the case of the well-known *Tsukuba* stereo pair. In this example, the truncated linear distance $\theta_{pq}(x_p, x_q) = w_{pq} \cdot \min(|x_p - x_q|, \theta_{\max})$ (with $w_{pq} = 20$, $\theta_{\max} = 2$) has been used as the MRF pairwise potential function. Fig. 5(b) contains the corresponding plot that shows how the costs of the primal and the dual problem (*i.e.* the MRF energy and the lower bound) vary during the execution of the algorithm. As in all other examples, here as well we have included the corresponding plots for the TRW-T and TRW-S algorithms. It is worth noting that, in this example, TRW-T did not manage to reduce the MRF energy (or increase the lower bound) as effectively as the DD-MRF algorithm. This is despite the fact that, as in all of this paper's experiments, exactly the same set of spanning trees has been used by both algorithms (we recall here that TRW-T uses a set of spanning trees for doing its message-passing).

Another issue that we investigated was how to set the

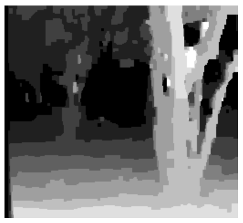


(a) Estimated disparity

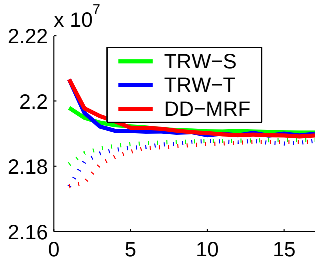


(b) Energy and lower bound plots

Fig. 6: Results for the *Map* stereo pair.

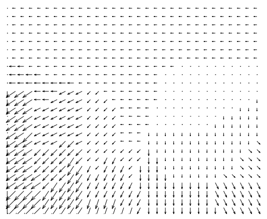


(a) Estimated disparity

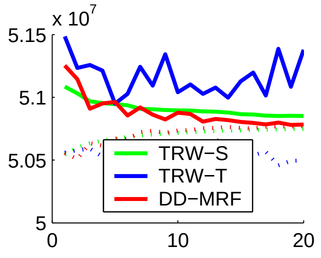


(b) Energy and lower bound plots

Fig. 7: Results for the *SRI tree* stereo pair.



(a) Estimated optical flow



(b) Energy and lower bound plots

Fig. 8: Optical flow for the *Yosemite* image sequence.

positive multipliers $\{\alpha_t\}$. These multipliers are used for updating the dual variables during the subgradient method. Theorem 2 describes just one of the simplest methods that can be used for this task. We have also experimented with a few other schemes as well, but we still intend to experiment with many more in the future, since there is a large literature on this subject [1]. *E.g.*, one of the schemes that we found to work well in practice was to update the multipliers $\{\alpha_t\}$ using the following formula:

$$\alpha_t = \gamma \frac{\textsc{BestPrimal}_t - \textsc{Dual}_t}{\|\nabla g_t\|^2}. \tag{13}$$

Here, $\textsc{BestPrimal}_t$ denotes the MRF energy of the best primal solution up to iteration t, $\textsc{Dual}_t$ denotes the current value of the dual function at the t-th iteration, while ∇g_t denotes the subgradient of the dual function at time t. Also, γ denotes a constant taking values in $(0, 2]$. The intuition behind this formula is that, initially, when the primal-dual gap (and hence the quantity $\textsc{BestPrimal}_t - \textsc{Dual}_t$) is large, $\{\alpha_t\}$ will take large values. This means that large changes will be initially applied to the dual variables (and hence to the primal variables as well), which makes sense since we are still far from the optimum. During the last iterations, however, as the primal-dual gap will be smaller, $\{\alpha_t\}$ will be assigned smaller values and hence the dual variables will be modified using finer updates. Another thing we have experimented with was using an incremental subgradient method [1] (instead of a standard subgradient algorithm). By doing so, we found that this method can give improved results in some cases.

Figures 6, 7 contain further results on stereo matching. Specifically, Fig. 6(a) displays the produced disparity for the *Map* stereo pair, while Fig. 6(b) contains the corresponding energy plots generated during the algorithm's execution. Similarly, the corresponding results for the *SRI-tree* stereo pair are displayed in Figures 7(a) and 7(b). For the case of the *Map* stereo pair, the MRF pairwise potentials were set equal to $\theta_{pq}(x_p, x_q) = 4 \cdot \min(|x_p - x_q|, 3)$, whereas for the case of the *SRI-tree* example the pairwise potentials were defined using the following truncated linear distance $\theta_{pq}(x_p, x_q) = 6 \cdot \min(|x_p - x_q|, 5)$.

As a further test, we have also applied our method to the optical flow estimation problem. In this case, labels correspond to 2D displacement vectors, while the unary potential, for assigning vector $x_p = (u_x, u_y)$ to pixel $p = (p_x, p_y)$, equals $\theta_p(x_p) = |\mathcal{I}_{\text{next}}(p_x + u_x, p_y + u_y) - \mathcal{I}_{\text{cur}}(p_x, p_y)|$, where $\mathcal{I}_{\text{cur}}, \mathcal{I}_{\text{next}}$ denote the current and next image frame. Also, the pairwise potential between labels $x_p = (u_x, u_y)$, $x_q = (v_x, v_y)$ equals the following truncated squared Euclidean distance $\theta_{pq}(x_p, x_q) = w_{pq} \min(\|(u_x - v_x, u_y - v_y)\|^2, \theta_{\max})$. An optical flow result, generated by applying DD-MRF to the well-known Yosemite sequence (with $w_{pq} = 10, \theta_{\max} = 20$), is shown in Fig. 8, along with plots for the corresponding upper and lower bounds. Note again that, contrary to our method, TRW-T has not managed to effectively reduce the MRF energy in this case.

Also, note that DD-MRF has been able to find very low MRF energy in all of the examples. In fact, based on the lower bounds estimated from the plots in Figures 5-8, one can actually show that the generated energy is extremely close to the minimum MRF energy. *E.g.*, based on these bounds, the energy found by DD-MRF is within relative distance 0.0094, 0.0081, 0.00042, 0.00012 from the minimum energy corresponding to *Tsukuba*, *map*, *SRI-tree* and *Yosemite* respectively (relative distance is measured as $\frac{\text{ENERGY}-\text{LOWER\_BOUND}}{\text{LOWER\_BOUND}}$). Also, the corresponding running times (per iteration) of the algorithm were 0.32, 0.34, 0.17, 0.41 secs respectively (measured on a 2GHz CPU). Regard-

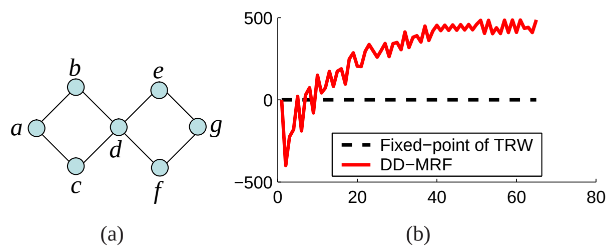


Fig. 9: **(a)** A simple graph that can be used for showing that TRW algorithms cannot maximize the lower bound on the MRF energy. The graph shown here is decomposed into 2 trees $T_1 = (a, b, d, e, g)$, $T_2 = (a, c, d, f, g)$. **(b)** A plot of the lower bounds produced by Dual-DT and TRW algorithms for the graph in Fig. 9(a), when $\kappa = 1000$ (see text). Notice the large gap between these 2 bounds. In fact, the value of this gap can be made arbitrarily large by, *e.g.*, increasing κ.

ing the choice of the trees that are associated with the slave MRFs, we found that, in practice, the smaller these trees are, the slower the convergence of the algorithm was. For this reason, each slave MRF was usually associated with a separate spanning tree of the original graph. Furthermore, the following termination criterion has been used: the algorithm stops when either the primal-dual gap has not decreased significantly for a certain number of iterations, or a maximum number of iterations has been exceeded.

We finally borrow an example from [3] to illustrate that DD-MRF can maximize the dual problem (9) (*i.e.* the lower bound on the MRF energy), even in cases where the TRW algorithms fail to do so. In fact, as this example shows, TRW algorithms may get stuck to a lower bound, which can be arbitrarily far from the maximum lower bound. The graph for this example is shown in Fig. 9(a), where we assume that nodes a, b, c, e, f, g, have two possible labels, while node d has three possible labels. The following two trees $T_1 = (a, b, d, e, g)$, $T_2 = (a, c, d, f, g)$ are used in this case, both of which are supposed to have zero unary potentials, *i.e.* $\boldsymbol{\theta}_p^{T_1} = \mathbf{0}\ \forall p \in T_1, \boldsymbol{\theta}_p^{T_2} = \mathbf{0}\ \forall p \in T_2$. Also, the pairwise potentials for these trees are set as follows:

$$\boldsymbol{\theta}_{ab}^{T_1}=\begin{bmatrix}\kappa & 0\\ 0 & \kappa\end{bmatrix}, \boldsymbol{\theta}_{bd}^{T_1}=\begin{bmatrix}0 & \kappa & \kappa\\ \kappa & 0 & 0\end{bmatrix}, \boldsymbol{\theta}_{de}^{T_1}=\begin{bmatrix}\kappa & 0\\ 0 & \kappa\\ \kappa & 0\end{bmatrix}, \boldsymbol{\theta}_{eg}^{T_1}=\begin{bmatrix}0 & \kappa\\ \kappa & 0\end{bmatrix},$$

$$\boldsymbol{\theta}_{ac}^{T_2}=\begin{bmatrix}\kappa & 0\\ 0 & \kappa\end{bmatrix}, \boldsymbol{\theta}_{cd}^{T_2}=\begin{bmatrix}\kappa & 0 & 0\\ 0 & \kappa & \kappa\end{bmatrix}, \boldsymbol{\theta}_{df}^{T_2}=\begin{bmatrix}\kappa & 0\\ \kappa & 0\\ 0 & \kappa\end{bmatrix}, \boldsymbol{\theta}_{fg}^{T_2}=\begin{bmatrix}\kappa & 0\\ 0 & \kappa\end{bmatrix},$$

where κ denotes a positive constant. As it was shown in [3], the above dual variables $\boldsymbol{\theta}^{T_1}$, $\boldsymbol{\theta}^{T_2}$ form a fixed point for all TRW algorithms (as $\boldsymbol{\theta}^{T_1}, \boldsymbol{\theta}^{T_2}$ satisfy the WTA condition). Hence, in this case, these algorithms will get stuck to a lower bound of value zero, *i.e.* arbitrarily far from the true maximum lower bound that can grow indefinitely by increasing parameter κ. On the contrary, as shown in Fig. 9(b), DD-MRF does not get stuck to such a bad lower bound when starting from $\boldsymbol{\theta}^{T_1}$, $\boldsymbol{\theta}^{T_2}$.

6. Extensions and conclusions

By being based on the technique of dual decomposition, *i.e.* one of the most powerful and widely used techniques in optimization, the proposed framework gains extreme generality and flexibility. For instance, one of the extensions we plan to explore in the future is to use exactly the same framework, but for optimizing MRFs with higher order cliques. A similar subgradient algorithm will result in this case, which can again provably maximize a lower bound on the energy. The only difference will be that, instead of the standard max-product, a factor graph max-product algorithm will have to be used for computing the subgradients. Another closely related idea, that we also wish to explore, is to use the proposed framework for constructing tighter LP relaxations to the MRF optimization problem (1) by considering groups of variables (again, minor modifications would need to be applied to our framework for this). The resulting algorithm would again be able to maximize an even stronger lower bound on the MRF energy, thus leading to better primal solutions in case of difficult MRF problems. On another note, an additional advantage is that our framework reduces MRF optimization to a projected subgradient algorithm. This connection can motivate new research, while it can also prove to be of great benefit, since subgradient methods form a very well studied topic in optimization, with a vast literature devoted to it. In fact, exploring some of the existing, but more advanced subgradient optimization techniques is one very interesting avenue of future research, that could potentially lead to even more powerful MRF optimization techniques in the future. To conclude, a novel and very general message-passing framework for MRF optimization has been presented, which possesses stronger theoretical properties (compared to existing message-passing methods), while also giving very good results in practice.

References

[1] D. Bertsekas. *Nonlinear Programming*. 1999. 1, 2, 5, 7

[2] Y. Boykov, O. Veksler, and R. Zabih. Fast approximate energy minimization via graph cuts. *PAMI*, Nov. 2001. 1

[3] V. Kolmogorov. Convergent tree-reweighted message passing for energy minimization. *PAMI*, 2006. 1, 5, 6, 8

[4] V. Kolmogorov and M. Wainwright. On the optimality of tree-reweighted max-product message passing. In *UAI*, 2005. 6

[5] N. Komodakis, G. Tziritas, and N. Paragios. Fast, approximately optimal solutions for single and dynamic MRFs. In *CVPR*, 2007. 1

[6] J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. 1988. 1

[7] M. Wainwright, T. Jaakkola, and A. Willsky. Map estimation via agreement on trees: message-passing and linear programming. *IEEE Trans. on Information Theory*, 2005. 1, 6

[8] J. Yedidia, W. Freeman, and Y. Weiss. Constructing free energy approximations and generalized belief propagation algorithms. *IEEE Transactions on Information Theory*, 2005. 1