MRF Energy Minimization & Beyond via Dual Decomposition

Nikos Komodakis, Nikos Paragios and Georgios Tziritas

Abstract

This paper introduces a new rigorous theoretical framework to address discrete MRF-based optimization in computer vision. Such a framework exploits the powerful technique of Dual Decomposition. It is based on a projected subgradient scheme that attempts to solve an MRF optimization problem by first decomposing it into a set of appropriately chosen subproblems and then combining their solutions in a principled way. In order to determine the limits of this method, we analyze the conditions that these subproblems have to satisfy and we demonstrate the extreme generality and flexibility of such an approach. We thus show that, by appropriately choosing what subproblems to use, one can design novel and very powerful MRF optimization algorithms. For instance, in this manner we are able to derive algorithms that: 1) generalize and extend state-of-the-art message-passing methods, 2) optimize very tight LP-relaxations to MRF optimization, 3) and take full advantage of the special structure that may exist in particular MRFs, allowing the use of efficient inference techniques such as, *e.g*, graph-cut based methods. Theoretical analysis on the bounds related with the different algorithms derived from our framework and experimental results/comparisons using synthetic and real data for a variety of tasks in computer vision demonstrate the extreme potentials of our approach.

Index Terms

Discrete optimization, linear programming, Markov Random Fields, graphical models, message-passing, graph-cuts.

I. Introduction

Discrete Markov Random Fields (MRFs) are a popular class of undirected probabilistic graphical models that have been of fundamental importance to many computer vision problems. This is the reason why MRF optimization methods have been constantly attracting a significant amount of research for more than 20 years now. For instance, to mention a few of their

N. Komodakis (corresponding author) is with the Computer Science Department of the University of Crete, Greece. This work has been carried out during his affiliation with the Laboratoire de mathematiques appliquees (MAS) of Ecole Centrale de Paris, France (mailto: komod@csd.uoc.gr).

N. Paragios is with the Laboratoire de mathematiques appliquees (MAS) of Ecole Centrale de Paris and with the GALEN group of INRIA-Saclay Ile-de-France, France.

G. Tziritas is with the Computer Science Depaptment of the University of Crete, Greece.

applications in vision, discrete MRFs have been used extensively in stereo matching [47], [31], [19], image segmentation [27], optical flow estimation [37], image denoising [11], texture synthesis [29], image completion [25] and object recognition [10]. Moreover, their influence goes far beyond computer vision as they have been used with great success in many other areas as well, including computer graphics, medical image analysis, machine learning, digital communications, and statistical physics [12], [18].

MRF optimization is often posed as the task of finding the mode of a probability distribution (known as the MAP estimation problem). Here it will be directly described as an energy minimization task. In this context, we assume that a discrete set of labels $\mathcal{L}$, as well as a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, *i.e*, with nodes $\mathcal{V}$ and edges $\mathcal{E}$, are given as input to us. Furthermore, we assume that a so called unary potential function $\theta_p(\cdot) : \mathcal{L} \to \mathbb{R}$ and a pairwise potential function $\theta_{pq}(\cdot,\cdot) : \mathcal{L} \times \mathcal{L} \to \mathbb{R}$ are defined for each node p and edge pq of the input graph $\mathcal{G}$ respectively. The task of MRF optimization then amounts to assigning a label $l_p \in \mathcal{L}$ to each node $p \in \mathcal{V}$ such that the following objective function, known as the MRF energy, is minimized:

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

Currently, two classes of methods are the most prominent ones in MRF optimization [46], [47]: those based on graph-cuts, 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 [49], [21]. Although they appear similar to the max-product Belief Propagation (BP) algorithm [34] on the surface, these methods are in fact quite different, as well as far more powerful. They rely on the following linear integer program, which can be shown to be equivalent to the task of minimizing the MRF energy (1) [40], [7], [53]:

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

In the above formulation, $\boldsymbol{\theta} = \left\{ \{\boldsymbol{\theta}_p\}_{p \in \mathcal{V}}, \{\boldsymbol{\theta}_{pq}\}_{pq \in \mathcal{E}} \right\}$ represents a vector of MRF-parameters that consists of all unary $\boldsymbol{\theta}_p = \{\theta_p(\cdot)\}$ and pairwise $\boldsymbol{\theta}_{pq} = \{\theta_{pq}(\cdot,\cdot)\}$ vectorized potential functions. Similarly, $\mathbf{x} = \left\{ \{\mathbf{x}_p\}_{p \in \mathcal{V}}, \{\mathbf{x}_{pq}\}_{pq \in \mathcal{E}} \right\}$ denotes a vector of binary MRF-variables consisting of all unary subvectors $\mathbf{x}_p = \{x_p(\cdot)\}$ and pairwise subvectors $\mathbf{x}_{pq} = \{x_{pq}(\cdot,\cdot)\}$. These MRF-variables take $\{0, 1\}$ values and are essentially indicators of the labels that are assigned to each MRF node, as well as to each pair of nodes, *i.e*, they 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. Enforcing these conditions on the MRF-variables is easily seen as being equivalent to requiring that the vector $\mathbf{x}$ lies in the set $\mathcal{X}^{\mathcal{G}}$ defined below (which is exactly why that set is used as the feasible set of optimization problem (2)). For any graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, the set $\mathcal{X}^{\mathcal{G}}$ is defined as follows:

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

The first constraints simply express the fact that a unique label must be assigned to each node p, while the second constraints enforce consistency between the unary variables $x_p(\cdot)$, $x_q(\cdot)$ and the pairwise variables $x_{pq}(\cdot, \cdot)$, since they ensure that if $x_p(l) = x_q(l') = 1$, then $x_{pq}(l, l') = 1$ as well. The above set $\mathcal{X}^{\mathcal{G}}$ is also known as the *marginal polytope* [49].

As we already mentioned, TRW methods are tightly related to the above integer program (2). More precisely, they are connected with the linear programming (LP) relaxation of (2), which is formed by replacing the integer constraints $x_p(\cdot), x_{pq}(\cdot, \cdot) \in \{0, 1\}$ with the relaxed constraints $x_p(\cdot), x_{pq}(\cdot, \cdot) \geq 0$[1] (we will hereafter refer to this relaxation as the *standard LP relaxation* and we will denote it by LP$_{\text{MRF}}$). Based on the assumption that this relaxation provides a good approximation to the integer program, TRW methods hope to obtain an approximately optimal solution to the latter, *i.e*, to the MAP estimation task, via solving the former, *i.e*, the relaxation LP$_{\text{MRF}}$. However, TRW methods do not attempt to minimize LP$_{\text{MRF}}$ directly. Instead, they focus on solving the dual of that relaxation, which is exactly what motivates these methods. This motivation also lies behind some other MAP estimation algorithms such as the max-sum diffusion algorithm reviewed in [53] or the more recent algorithm of Globerson and Jaakkola that was proposed in [14]. These are methods that also operate on a dual of relaxation LP$_{\text{MRF}}$, and can essentially be understood as block coordinate ascent procedures applied to that dual. Obviously, the cost of any feasible solution to the dual of LP$_{\text{MRF}}$ yields a lower bound on the optimal MRF energy. Hence, solving the dual corresponds to a maximization of this lower bound, which is essentially the key idea behind all the above mentioned techniques. Based on the returned solution from the dual, a solution to the primal problem (2), *i.e* to the original MRF optimization task, can then be extracted based on a rounding procedure. Moreover, the quality of the resulting MRF solution depends critically on the quality of the estimated dual lower bound (*i.e*, how large that bound is). However, despite this fact, (*i.e*, despite that the key to the success of all TRW algorithms relies on deriving a dual solution that maximizes the LP lower bound to the MRF energy), none of them can actually provide such a guarantee. In fact, as shown in [21], there exist cases for which this is known not to be true.

Motivated by this observation, a novel MRF-optimization scheme is proposed in this paper. It is called DD-MRF (from Dual Decomposition MRF) and, unlike existing message-passing

[1]The resulting polytope is known as the *local marginal polytope* in the literature, and is denoted by LOCAL($\mathcal{G}$) hereafter.

techniques, it can provably solve the above mentioned dual LP (*i.e*, maximize the lower bound), which forms, as already explained, the driving force behind all TRW algorithms. Moreover, it is derived based on very general principles. In particular, the theoretical setting of our method rests on the technique of *dual decomposition* [2]. This is an extremely general technique, which is well known to people in optimization as it has been used with great success for solving many different kinds of problems up to now. When used in the particular case of the MRF problem, it leads to a simple, but very powerful, *projected-subgradient* scheme for MAP estimation. Here we analyze the conditions under which this MRF scheme is applicable and prove that it really enjoys great flexibility and generality. We thus show that it leads to a very elegant framework, which allows for designing powerful MAP estimation algorithms that are easily adaptable to the problem at hand. For instance, as an illustration, we use it to derive message-passing techniques that generalize and provide new insights into existing approaches (such as TRW methods), while they also enjoy better theoretical properties. Going a bit further, we also show how it can lead to optimization schemes that provably solve not just the standard LP relaxation associated to an MRF problem, but also much tighter relaxations, thus allowing for high-quality MRF solutions to be extracted even in more difficult cases. More generally, we demonstrate that one can use the proposed framework to easily design algorithms that are tailored to any particular class of MRFs. The derived methods can take full advantage of the structure in that class, and, *e.g*, allow the use of efficient inference techniques such as graph-cut based approaches.

The remainder of this paper focuses on analyzing and describing the proposed optimization framework. It is structured as follows: after reviewing related work in section II, we proceed by briefly describing the dual decomposition principle in section III. By using this principle, we are able to derive a projected subgradient scheme that computes a solution to a difficult or large optimization problem by first decomposing it into a set of easier subproblems and then combining the subproblems' solutions in a principled and optimal manner. We apply this idea to the case of MRF optimization in section IV, where we choose to use tree-structured MRFs as subproblems. This leads to a message passing algorithm that generalizes TRW methods and has stronger theoretical properties [24]. These properties are analyzed thoroughly in section V, where we also show that the proposed algorithm can provably optimize a standard LP relaxation to problem (2). To demonstrate the power and flexibility of our framework with regard to MRF optimization, we describe how it allows us to treat in a unified way much more general cases, *i.e*, when more general subproblems than tree-structured MRFs are being used. We thus analyze in section VI the very weak conditions that these subproblems have to satisfy and, as an illustration, we show that one can derive algorithms guaranteed to globally optimize even tighter LP relaxations to problem (2). A thorough theoretical analysis of the properties of these relaxations is provided in sections VI-A and VI-B. Furthermore, in sections VI-C and VI-D we illustrate how, by an appropriate choice of the subproblems, even non-message passing algorithms can be derived that are easily adaptable to take full advantage of the problem's structure. For instance, in section VI-C, we describe dual decompositions based on submodular problems, in which case we show that very efficient graph cut based techniques can be used for accelerating the subgradient algorithm. Experimental results on a variety of computer vision tasks as well as on synthetic problems are presented in section VII, verifying the state of the art performance of our approach. Finally, we conclude in section VIII. We note that the current article provides a significantly extended version of our earlier work [24].

II. Related work

MRF optimization has motivated a great deal of research over the last years and, as a result, many related algorithms have been proposed in this regard. Currently, two classes of methods seem to be dominating the literature on MRF optimization: graph-cut based techniques [4], [26], [38], [35], and algorithms that rely on message-passing. Our work is most closely connected with the latter class of methods. Message-passing algorithms have been shown to provide a powerful way for solving many problems related to probabilistic graphical models. Pearl's belief propagation has been perhaps one of the first message-passing algorithms for inference on Bayesian networks [34]. Although this algorithm is exact on tree-structured graphs (*i.e*, it provably computes the global optimum), no such guarantees can be provided when BP is applied to graphs with cycles (in fact, for loopy graphs, BP is not even guaranteed to converge). Nevertheless, the very simple idea of pretending that a loopy graph has no cycles and then repeatedly applying BP to it has often shown remarkably good experimental performance in practice [13] (for obvious reasons the resulting algorithm is called loopy BP). As a result, many generalizations of belief propagation have been proposed in the literature [57], [55], [45], [9], [5], [16]. However, despite the fact that there has been a lot of work on trying to provide a better theoretical analysis of BP-based algorithms [57], [51], [50], [15], an important drawback of these methods remains that their theoretical optimality and/or convergence properties are not yet well understood and, so, one does not really know when and why these methods fail.

An important advance took place recently with the so called tree-reweighted max-product (TRW-MP) algorithm introduced by Wainwright *et al* in their seminal work [49]. This message-passing algorithm is directly connected to the LP relaxation of the integer program (2) and is actually motivated by trying to optimize that relaxation. In fact, Wainwright *et al* proved that, under certain conditions, suitable fixed points of TRW-MP provide optimal solutions to this LP relaxation. In a subsequent work [21], Kolmogorov has proposed a sequential version of TRW-MP, called TRW-S, where messages are updated not in a parallel but in a sequential order. Unlike the original TRW-MP algorithm, which is not guaranteed to converge, TRW-S provides improved convergence properties. However, similarly to TRW-MP, the fixed points of TRW-S are not guaranteed to be LP-optimal solutions except for some very specific cases (*e.g*, they are

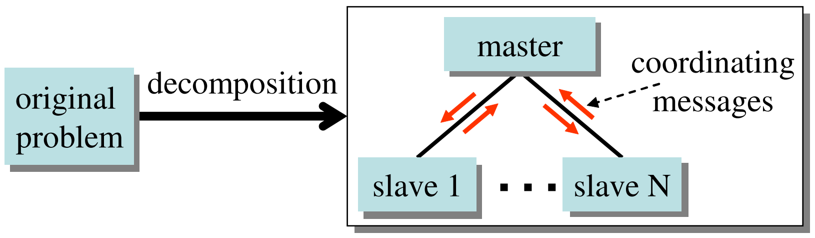


Fig. 1: The original (possibly difficult) optimization problem decomposes into easier subproblems (called the *slaves*) that are coordinated by a *master* problem via message exchanging.

known to be LP-optimal when the original MRF is binary and submodular [22]). This observation also applies to another convergent message-passing algorithm that has been recently proposed by Globerson and Jaakkola [14], as well as to the max-sum diffusion algorithm reviewed in [53], both of which are coordinate ascent procedures on some dual of relaxation $\mathrm{LP}_{\mathrm{MRF}}$. The importance of relaxation $\mathrm{LP}_{\mathrm{MRF}}$ to MRF optimization and/or its connection to message-passing methods has also been explored in some other interesting works (*e.g*, we point the interested reader to [52], [28], [43], [56] for more details). Recently, Ravikumar *et al* [36] proposed a method for solving relaxation $\mathrm{LP}_{\mathrm{MRF}}$ based on the theory of proximal minimization. Their method works directly in the primal domain as opposed to all other techniques mentioned above that operate on the dual variables. Also, Johnson *et al* [17] have very recently presented a dual algorithm for trying to optimize relaxation $\mathrm{LP}_{\mathrm{MRF}}$, where a smoothing-based procedure is used for modifying the objective function of $\mathrm{LP}_{\mathrm{MRF}}$ such that it becomes strictly convex.

Schlensinger and Giginyak [39] have also developed a dual method for the same purpose: based on the concept of equivalent transformations (also known as reparameterizations) of a max-sum problem, they use a subgradient algorithm for finding an equivalent transformation that optimizes relaxation $\mathrm{LP}_{\mathrm{MRF}}$. Furthermore, they apply this approach using a tree decomposition for the special-case where trees coincide with rows and columns of grid-structured MRFs.

III. Dual decomposition

In this section we briefly review the dual decomposition technique, upon which our framework relies. The core idea behind that technique is surprisingly simple: *given a difficult or large problem, we decompose it 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 many times in the operations research literature for solving optimization problems that are either large-scale or difficult. 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 so we will give a simple example just to illustrate how it works. To this end, consider the following problem (where x denotes a vector of variables and $\mathcal{C}$ is a closed[2] convex set):

$$\begin{aligned} \min_{\boldsymbol{x}} \quad & \textstyle\sum_i f^i(\boldsymbol{x}) \\ \text{s.t.} \quad & \boldsymbol{x} \in \mathcal{C} \end{aligned} \tag{3}$$

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

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

Obviously this is equivalent to our original problem (3). Furthermore, if the coupling constraints $\boldsymbol{x}^i = \boldsymbol{x}$ were absent, the problem would decouple. We therefore relax them via introducing multipliers $\{\boldsymbol{\lambda}^i\}$ and form the following Lagrangian dual function:

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

We next eliminate $\boldsymbol{x}$ from $g(\{\boldsymbol{\lambda}^i\})$ by minimizing over that variable. This simply results in having $\{\boldsymbol{\lambda}^i\} \in \Lambda = \{\{\boldsymbol{\lambda}^i\} | \sum_i \boldsymbol{\lambda}^i = 0\}$ (since 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:

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

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

$$\max_{\{\boldsymbol{\lambda}^i\} \in \Lambda} g(\{\boldsymbol{\lambda}^i\}) = \textstyle\sum_i g^i(\boldsymbol{\lambda}^i), \tag{4}$$

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

$$\begin{aligned} g^i(\boldsymbol{\lambda}^i) = \min_{\boldsymbol{x}^i} & f^i(\boldsymbol{x}^i) + \boldsymbol{\lambda}^i \cdot \boldsymbol{x}^i \\ \text{s.t.} \quad & \boldsymbol{x}^i \in \mathcal{C} \end{aligned} \tag{5}$$

Problem (4) is always convex[3] and so it can be solved using, *e.g*, a projected subgradient method (due to that $g(\cdot)$ is typically not differentiable). According to that method, at each iteration the dual variables $\{\boldsymbol{\lambda}^i\}$ are updated as $\boldsymbol{\lambda}^i \leftarrow [\boldsymbol{\lambda}^i + \alpha_t \nabla g^i(\boldsymbol{\lambda}^i)]_\Lambda$. Here, α_t denotes a positive multiplier (for the t-th iteration), $[\,\cdot\,]_\Lambda$ denotes projection onto the set Λ, while $\nabla g^i(\boldsymbol{\lambda}^i)$

[2]In case the set $\mathcal{C}$ is not closed, $\min$ has to be replaced with $\inf$ in the derivations that follow.

[3]Note that the convexity of problem (4) holds regardless of whether or not the objective function of problem (3) is convex.

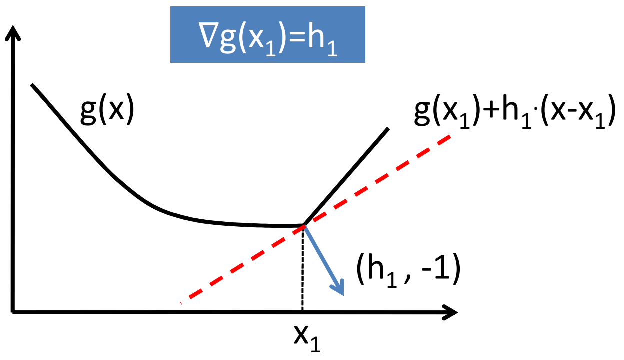


Fig. 2: The vector h_1 is a subgradient of function $g(\cdot)$ at x_1 if and only if $(h_1, -1)$ specifies a supporting hyperplane to the epigraph of $g(\cdot)$ at $(x_1, g(x_1))$.

denotes a subgradient[4] of $g^i(\cdot)$ at $\boldsymbol{\lambda}^i$. It thus remains to show how to compute such a subgradient $\nabla g^i(\boldsymbol{\lambda}^i)$. We recall that the subgradient of a convex function is a generalization of the notion of gradient for non-differentiable functions, and its estimation essentially corresponds to specifying a supporting hyperplane to a function's epigraph (see Fig. 2). For estimating a subgradient $\nabla g^i(\boldsymbol{\lambda}^i)$, we can rely on the following well-known lemma:

Lemma 1. *Let function $g(\boldsymbol{\lambda})$ be defined as $g(\boldsymbol{\lambda}) = \min_{x \in \mathcal{C}} \{a(\boldsymbol{x}) + \boldsymbol{\lambda} \cdot b(\boldsymbol{x})\}$, where $a(\cdot), b(\cdot)$ represent functions over a compact set $\mathcal{C}$. Let also vector $\bar{\boldsymbol{x}}$ be an optimal solution to problem* $\min_{x \in \mathcal{C}} \{a(\boldsymbol{x}) + \boldsymbol{\lambda} \cdot b(\boldsymbol{x})\}$, i.e, $g(\boldsymbol{\lambda}) = a(\bar{\boldsymbol{x}}) + \boldsymbol{\lambda} \cdot b(\bar{\boldsymbol{x}})$. *Then $b(\bar{\boldsymbol{x}})$ is a subgradient of $g(\cdot)$ at λ.*

Proof: The theorem follows from

$$g(\lambda') \leq a(\bar{\boldsymbol{x}}) + \boldsymbol{\lambda}' \cdot b(\bar{\boldsymbol{x}}) = (a(\bar{\boldsymbol{x}}) + \boldsymbol{\lambda} \cdot b(\bar{\boldsymbol{x}})) + (\boldsymbol{\lambda}' - \boldsymbol{\lambda}) \cdot b(\bar{\boldsymbol{x}}) = g(\boldsymbol{\lambda}) + (\boldsymbol{\lambda}' - \boldsymbol{\lambda}) \cdot b(\bar{\boldsymbol{x}})$$

■

From the above lemma it follows directly that it holds:

$$\nabla g^i(\boldsymbol{\lambda}^i) = \bar{\boldsymbol{x}}^i(\boldsymbol{\lambda}^i) \ ,$$

where $\bar{\boldsymbol{x}}^i(\boldsymbol{\lambda}^i)$ denotes any optimal solution to the i-th slave problem (5). By putting all of the above pieces together, the communication between the master and the slaves thus proceeds as follows:

1) The master sends the current $\{\boldsymbol{\lambda}^i\}$ to the slaves and asks them to optimize their problems.
2) The slaves respond to the master by solving their easy problems and sending back to him the resulting minimizers $\{\bar{\boldsymbol{x}}^i(\boldsymbol{\lambda}^i)\}$.
3) The master then collects the minimizers and updates each $\boldsymbol{\lambda}^i$ by setting $\boldsymbol{\lambda}^i \leftarrow [\boldsymbol{\lambda}^i + \alpha_t \bar{\boldsymbol{x}}^i(\boldsymbol{\lambda}^i)]_\Lambda$
4) The above three steps are repeated until convergence.

[4]Throughout the paper, by abuse of notation, we use $\nabla g(x)$ to denote a subgradient of function $g(\cdot)$ at point x, *i.e*, a vector that belongs in the subdifferential $\partial g(x)$. Note that this notation is non-conventional, since, in the literature $\nabla g(x)$ is used only to refer to the gradient of a differentiable function.

In essence, what happens is that a solution to the dual is obtained by operating at two levels. At the higher level, the master problem (4) coordinates the slaves simply by updating $\{\boldsymbol{\lambda}^i\}$ based on the currently extracted optimal solutions $\{\bar{\boldsymbol{x}}^i(\boldsymbol{\lambda}^i)\}$. And then, at the lower level, based on the updated $\{\boldsymbol{\lambda}^i\}$ each of the decoupled slave problems (5) is again solved independently to generate a new set of minimizers $\{\bar{\boldsymbol{x}}^i(\boldsymbol{\lambda}^i)\}$ for the next iteration.

IV. MRF optimization via Dual Decomposition

In this section, by following a reasoning similar to that in the previous example, we will describe how we can apply the dual decomposition method to MAP estimation for discrete MRFs. To prepare the reader for what is about to come next, our goal, at a high level, will be to decompose the original MRF optimization problem, which is NP-hard (since it is defined on a general graph $\mathcal{G}$), into a set of easier MRF subproblems, each one defined on a tree $\mathcal{T} \subset \mathcal{G}$. 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 $\mathcal{T}(\mathcal{G})$ be a set of subtrees of graph $\mathcal{G}$. The only requirement for $\mathcal{T}(\mathcal{G})$ is that its trees cover (at least once) every node and edge of graph $\mathcal{G}$. For each tree $\mathcal{T} \in \mathcal{T}(\mathcal{G})$ we will then imagine that there is a smaller MRF defined just on the nodes and edges of tree $\mathcal{T}$, and we will associate to it a vector of MRF-parameters $\boldsymbol{\theta}^{\mathcal{T}}$, as well as a vector of MRF-variables $\mathbf{x}^{\mathcal{T}}$ (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 $\mathbf{x}^{\mathcal{T}}$ 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 $\mathbf{x}^{\mathcal{T}} = \mathbf{x}_{|\mathcal{T}}$, where $\mathbf{x}_{|\mathcal{T}}$ represents the subvector of $\mathbf{x}$ containing MRF-variables only for nodes and edges of tree $\mathcal{T}$. In addition, all the vectors $\{\boldsymbol{\theta}^{\mathcal{T}}\}$ will be defined so that they satisfy the following conditions:

$$\sum_{\mathcal{T} \in \mathcal{T}(p)} \boldsymbol{\theta}_p^{\mathcal{T}} = \boldsymbol{\theta}_p, \qquad \sum_{\mathcal{T} \in \mathcal{T}(pq)} \boldsymbol{\theta}_{pq}^{\mathcal{T}} = \boldsymbol{\theta}_{pq}. \tag{6}$$

Here, $\mathcal{T}(p)$ and $\mathcal{T}(pq)$ denote the set of all trees of $\mathcal{T}(\mathcal{G})$ that contain node p and edge pq respectively. *E.g*, to ensure (6), one can simply set: $\boldsymbol{\theta}_p^{\mathcal{T}} = \frac{\boldsymbol{\theta}_p}{|\mathcal{T}(p)|}$ and $\boldsymbol{\theta}_{pq}^{\mathcal{T}} = \frac{\boldsymbol{\theta}_{pq}}{|\mathcal{T}(pq)|}$. Due to (6) and the fact that $\mathbf{x}^{\mathcal{T}} = \mathbf{x}_{|\mathcal{T}}$, energy $E(\boldsymbol{\theta}, \mathbf{x})$ thus decomposes into the energies $E(\boldsymbol{\theta}^{\mathcal{T}}, \mathbf{x}^{\mathcal{T}}) = \boldsymbol{\theta}^{\mathcal{T}} \cdot \mathbf{x}^{\mathcal{T}}$, or

$$E(\boldsymbol{\theta}, \mathbf{x}) = \sum_{\mathcal{T} \in \mathcal{T}(\mathcal{G})} E(\boldsymbol{\theta}^{\mathcal{T}}, \mathbf{x}^{\mathcal{T}}) \tag{7}$$

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

$$\mathbf{x}^{\mathcal{T}} \in \mathcal{X}^{\mathcal{T}}, \quad \mathbf{x}^{\mathcal{T}} = \mathbf{x}_{|\mathcal{T}}, \qquad \forall \mathcal{T} \in \mathcal{T}(\mathcal{G}) \tag{8}$$

Hence, our original MRF problem becomes equivalent to:

$$\begin{aligned} \min_{\{\mathbf{x}^{\mathcal{T}}\},\mathbf{x}} \quad & \textstyle\sum_{\mathcal{T}\in\mathcal{T}(\mathcal{G})} E(\boldsymbol{\theta}^{\mathcal{T}},\mathbf{x}^{\mathcal{T}}) \\ \text{s.t.} \quad & \mathbf{x}^{\mathcal{T}} \in \mathcal{X}^{\mathcal{T}} \;, \quad \forall \mathcal{T} \in \mathcal{T}(\mathcal{G}) \\ & \mathbf{x}^{\mathcal{T}} = \mathbf{x}_{|\mathcal{T}} \;, \quad \forall \mathcal{T} \in \mathcal{T}(\mathcal{G}) \end{aligned} \tag{9}$$

It is clear that without constraints $\mathbf{x}^{\mathcal{T}}\!=\!\boldsymbol{x}_{|\mathcal{T}}$, 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 $\boldsymbol{\lambda}^{\mathcal{T}} = \{\{\boldsymbol{\lambda}_p^{\mathcal{T}}\},\{\boldsymbol{\lambda}_{pq}^{\mathcal{T}}\}\}$) and form the Lagrangian dual function as:

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

Vector $\mathbf{x}$ can be eliminated from $g(\{\boldsymbol{\lambda}^{\mathcal{T}}\})$ by directly minimizing over it, which simply imposes the constraint $\{\boldsymbol{\lambda}^{\mathcal{T}}\}\!\in\!\Lambda$,[5] where the feasible set Λ is now defined as:

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

while the resulting Lagrangian dual function simplifies to:

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

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

$$\max_{\{\boldsymbol{\lambda}^{\mathcal{T}}\}\in\Lambda} g(\{\boldsymbol{\lambda}^{\mathcal{T}}\}) = \sum_{\mathcal{T}\in\mathcal{T}(\mathcal{G})} g^{\mathcal{T}}(\boldsymbol{\lambda}^{\mathcal{T}}), \tag{10}$$

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

$$\begin{aligned} g^{\mathcal{T}}(\boldsymbol{\lambda}^{\mathcal{T}}) = \min_{\mathbf{x}^{\mathcal{T}}} \quad & E(\boldsymbol{\theta}^{\mathcal{T}}\!+\!\boldsymbol{\lambda}^{\mathcal{T}},\mathbf{x}^{\mathcal{T}}) \\ \text{s.t.} \quad & \mathbf{x}^{\mathcal{T}} \in \mathcal{X}^{\mathcal{T}}. \end{aligned} \tag{11}$$

Problem (10) has thus become our master problem, and each slave problem (11) simply amounts to optimizing an MRF over a tree $\mathcal{T} \subset \mathcal{G}$, *i.e*, a much easier problem. For optimizing the master, we will use the projected subgradient method. As explained in §III, at each iteration of this method the dual variables $\boldsymbol{\lambda}^{\mathcal{T}}$ must first be updated as $\boldsymbol{\lambda}^{\mathcal{T}} \leftarrow \boldsymbol{\lambda}^{\mathcal{T}} + \alpha_t \nabla g^{\mathcal{T}}(\boldsymbol{\lambda}^{\mathcal{T}})$. Based on lemma 1, a subgradient of $g^{\mathcal{T}}(\cdot)$ equals $\nabla g^{\mathcal{T}}(\boldsymbol{\lambda}^{\mathcal{T}}) = \bar{\mathbf{x}}^{\mathcal{T}}$, where $\bar{\mathbf{x}}^{\mathcal{T}}$ represents any optimal solution to slave MRF (11), and so the above update amounts to setting $\boldsymbol{\lambda}^{\mathcal{T}} \leftarrow \boldsymbol{\lambda}^{\mathcal{T}} + \alpha_t \bar{\mathbf{x}}^{\mathcal{T}}$. It

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

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

Fig. 3: A basic update during the projected subgradient algorithm.

then only remains to project the resulting $\{\boldsymbol{\lambda}^{\mathcal{T}}\}$ onto the feasible set Λ. Due to the definition of Λ, this projection reduces to subtracting the average vector $\frac{\sum_{\mathcal{T}\in\mathcal{T}(p)} \boldsymbol{\lambda}_p^{\mathcal{T}}}{|\mathcal{T}(p)|}$ from each $\boldsymbol{\lambda}_p^{\mathcal{T}}$ (so that $\sum_{\mathcal{T}\in\mathcal{T}(p)} \boldsymbol{\lambda}_p^{\mathcal{T}} = 0$), as well as subtracting the average vector $\frac{\sum_{\mathcal{T}\in\mathcal{T}(pq)} \boldsymbol{\lambda}_{pq}^{\mathcal{T}}}{|\mathcal{T}(pq)|}$ from each $\boldsymbol{\lambda}_{pq}^{\mathcal{T}}$ (so that $\sum_{\mathcal{T}\in\mathcal{T}(pq)} \boldsymbol{\lambda}_{pq}^{\mathcal{T}} = 0$). By aggregating all of the above operations, a projected subgradient update is easily seen to reduce to $\boldsymbol{\lambda}_p^{\mathcal{T}} \mathrel{+}= \Delta\boldsymbol{\lambda}_p^{\mathcal{T}}$, $\boldsymbol{\lambda}_{pq}^{\mathcal{T}} \mathrel{+}= \Delta\boldsymbol{\lambda}_{pq}^{\mathcal{T}}$ where:

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

$$\Delta\boldsymbol{\lambda}_{pq}^{\mathcal{T}} = \alpha_t \cdot \left(\bar{\mathbf{x}}_{pq}^{\mathcal{T}} - \frac{\sum_{\mathcal{T}'\in\mathcal{T}(pq)} \bar{\mathbf{x}}_{pq}^{\mathcal{T}'}}{|\mathcal{T}(pq)|} \right) \tag{13}$$

Of course, each $\boldsymbol{\lambda}^{\mathcal{T}}$ is only used for defining the MRF-parameters $\boldsymbol{\theta}^{\mathcal{T}} + \boldsymbol{\lambda}^{\mathcal{T}}$ of the slave MRF in (11). Hence, instead of updating the Lagrange multipliers $\{\boldsymbol{\lambda}^{\mathcal{T}}\}$ at each iteration, one can choose to directly update the MRF-parameters $\{\boldsymbol{\theta}^{\mathcal{T}}\}$, *i.e*, set $\boldsymbol{\theta}_p^{\mathcal{T}} \mathrel{+}= \Delta\boldsymbol{\lambda}_p^{\mathcal{T}}$, $\boldsymbol{\theta}_{pq}^{\mathcal{T}} \mathrel{+}= \Delta\boldsymbol{\lambda}_{pq}^{\mathcal{T}}$. In this manner, the need for storing the dual variables $\{\boldsymbol{\lambda}^{\mathcal{T}}\}$ is avoided. This is actually how the pseudocode in Fig. 3 was formed, describing one basic update during the resulting subgradient algorithm.

A. *Analysis of the algorithm*

Let us now briefly summarize how the algorithm in Fig. 3 works. Like most other dual decomposition techniques, it operates on two levels (see Fig. 4). At the lower level, it has to solve each one of the decoupled slave problems (11). 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 $\mathcal{T} \in \mathcal{T}(\mathcal{G})$, and its MRF-parameters are specified by the vector $\boldsymbol{\theta}^{\mathcal{T}}$. 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 $\bar{\mathbf{x}}^{\mathcal{T}}$ for each $\mathcal{T} \in \mathcal{T}(\mathcal{G})$. 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 (10) is maximized. To this end, it thus has to update the MRF-parameters $\{\boldsymbol{\theta}^{\mathcal{T}}\}$ of all slave MRFs, based on the optimal solutions $\{\bar{\mathbf{x}}^{\mathcal{T}}\}$ 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 $\{\boldsymbol{\lambda}^{\mathcal{T}}\}$, but, as already explained, this is equivalent to updating the MRF-parameters $\{\boldsymbol{\theta}^{\mathcal{T}}\}$ instead).

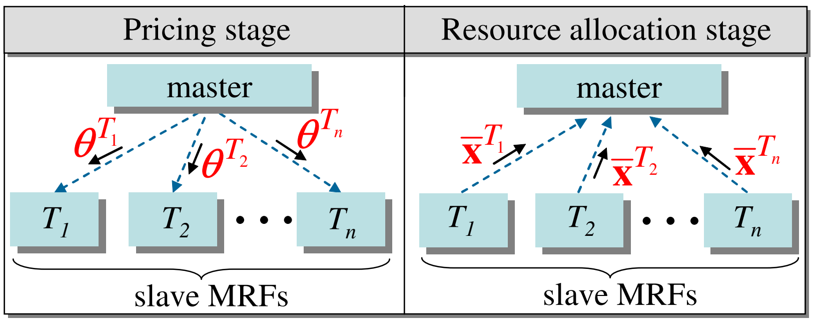


Fig. 4: Dual decomposition scheme for MRF optimization **Left:** Based on the current optimal solutions $\{\bar{\mathrm{x}}^{\mathcal{T}}\}$ (*i.e* the current resource allocation), the master assigns new MRF potentials $\{\boldsymbol{\theta}^{\mathcal{T}}\}$ (*i.e* new prices) to the slave MRFs. **Right:** Based on these new potentials, the slave MRFs immediately respond to the master by sending to him new optimal solutions $\{\bar{\mathrm{x}}^{\mathcal{T}}\}$ (*i.e* by readjusting their resource allocation).

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 $\mathcal{T} \in \mathcal{T}(p)$, the vector $\bar{\mathrm{x}}_p^{\mathcal{T}}$ will have the following form: $\bar{x}_p^{\mathcal{T}}(l) = 1$ if $l = l_p$, whereas $\bar{x}_p^{\mathcal{T}}(l) = 0$ if $l \neq l_p$. All these vectors will therefore coincide with each other and so $\boldsymbol{\Delta\lambda}_p^{\mathcal{T}} = \mathbf{0}$. Any vector $\boldsymbol{\theta}_p^{\mathcal{T}}$ 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 only to two trees, say $\mathcal{T}_1, \mathcal{T}_2$, 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 $\boldsymbol{\theta}_p^{\mathcal{T}_1}$, $\boldsymbol{\theta}_p^{\mathcal{T}_2}$ will take place:

$$\theta_p^{\mathcal{T}_1}(l) \mathrel{+}= \begin{cases} +\frac{\alpha_t}{2} & \text{if } l = l_1 \\ -\frac{\alpha_t}{2} & \text{if } l = l_2 \\ 0 & \text{otherwise} \end{cases}, \quad \theta_p^{\mathcal{T}_2}(l) \mathrel{+}= \begin{cases} -\frac{\alpha_t}{2} & \text{if } l = l_1 \\ +\frac{\alpha_t}{2} & \text{if } l = l_2 \\ 0 & \text{otherwise} \end{cases}$$

As can be seen, what happens is that the master tries to readjust the unary potentials for node p at $\mathcal{T}_1, \mathcal{T}_2$, 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 $\{\mathrm{x}^{\mathcal{T}}\}$ as amounts of resources consumed by the slave problems, with variables $\mathrm{x}^{\mathcal{T}}$ representing the amount of resources consumed by the MRF problem for tree $\mathcal{T}$. In dual decomposition, the master algorithm never sets these amounts explicitly. Instead, it just sets the prices for the resources, *i.e*, variables $\{\boldsymbol{\theta}^{\mathcal{T}}\}$ in our case. Then, based on these prices, each slave MRF is left free to independently decide how many resources it wants to 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 to know, 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 $\{\boldsymbol{\theta}^{\mathcal{T}}\}$. DD-MRF, however, relies on the optimal solutions $\bar{\mathbf{x}}^{\mathcal{T}}$ for that task. Furthermore, contrary to TRW algorithms, which modify all dual variables at each iteration, DD-MRF needs to modify a vector of dual variables $\boldsymbol{\theta}_p^{\mathcal{T}}$ at node p if the slave MRFs disagree about that node's label.

Before proceeding, we should also note that, since no Lagrange multipliers $\{\boldsymbol{\lambda}^{\mathcal{T}}\}$ need to be stored (as $\{\boldsymbol{\theta}^{\mathcal{T}}\}$ 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 [21].

B. 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 (9). To this end, we have opted to relax some of the complicating constraints in (9) 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 $\{\bar{\mathbf{x}}^{\mathcal{T}}\}$ that were generated for the slave problems. Recall that each $\bar{\mathbf{x}}^{\mathcal{T}}$ is a $\{0,1\}$ vector, which essentially specifies an optimal labeling for a slave MRF at tree $\mathcal{T}$. As explained in §IV-A, 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 $\mathcal{T} \in \mathcal{T}(\mathcal{G})$ is a spanning tree, then each $\bar{\mathbf{x}}^{\mathcal{T}}$ directly specifies a feasible solution to the MRF problem. In general, one will have to traverse the slave MRFs and copy the labels from the corresponding optimizers until all nodes of the master get labeled.

Of course, there are many other possible ways of getting good feasible primal solutions. For instance, for each node-label pair (p, l), one may count the number of times that variable $x_p(l)$ is active (*i.e*, label l is assigned to node p) in an optimal solution of a slave MRF, where all solutions up to the current iteration are taken into account. The label having the biggest count at a node can then be chosen as its new label. The astute reader will perhaps notice that this heuristic relates to the recovery of an optimal fractional solution to primal problem (9) via utilizing weighted averages of dual subgradients (see equations (15), (16) below). Another way to compute a feasible primal solution, that we found to work well in practice, is to use the messages exchanged during the max-product algorithm (for the slave MRFs), since these messages contain valuable information. *E.g*, a heuristic similar to the one proposed in [21] can be used for this purpose. In this case, we visit the MRF nodes in some predefined order, say, $p_1, p_2, \ldots, p_n$ and then assign to each node p_i the label $\hat{x}_i$ that minimizes the following expression:

$$\theta_{p_i}(\hat{x}_i) + \sum_{j<i} \theta_{p_j p_i}(\hat{x}_j, \hat{x}_i) + \sum_{j>i,\, \mathcal{T} \in \mathcal{T}(p_j p_i)} M^{\mathcal{T}}_{p_j p_i}(\hat{x}_i) \ , \tag{14}$$

where $M^{\mathcal{T}}_{pq}(\cdot)$ denotes the max-product messages computed for edge pq of tree $\mathcal{T}$. The justification for this heuristic comes from the observation made in [31] that if the set of nodes with multiple minima consists of disjoint chains then upon convergence the above procedure will yield a global optimum.

We should note at this point that the recovery of primal solutions based on dual subgradients has been a subject that attracted significant interest in the optimization literature for subgradient methods. An example of a method on this topic is the Volume algorithm proposed by Barahona and Anbil [1], which can essentially be viewed as fast approximation to the Dantzig-Wolfe decomposition method, and produces primal solutions by computing the volume below the faces that are active at an optimal dual solution. An even more popular way of generating primal solutions, that has been studied in a number of existing works, is also via utilizing ergodic sequences (*i.e* sequences of weighted averages) of dual subgradients (note that the use of an ergodic sequence forms a common technique for inducing convergence properties that an original sequence lacks). An early example of such an averaging scheme based on dual subgradient information is the method of Shor [42] for linear optimization problems. That work has been extended by Sherali and Choi [41] to allow for more general choices for the weights used during averaging. Furthermore, recently, Larsson *et al* [30] have significantly generalized these results to convex constrained optimization problems. The method proposed by Larsson *et al* utilizes ergodic sequences either of the form

$$x^k = \frac{\sum_{t=1}^k a_t s^t}{\sum_{t=1}^k a_t} \ , \quad k = 1, 2, \dots \tag{15}$$

or of the form

$$x^k = \frac{\sum_{t=1}^k s^t}{k} \ , \quad k = 1, 2, \dots \tag{16}$$

In the above formulas, s^t represents the dual subgradient at the t-th iteration, while a_t denotes the stepsize used at the t-th iteration. As shown in [30] the resulting sequence $\{x^k\}$ is guaranteed to converge to an optimal primal solution. In general, convergence happens only in the limit. However, more recent work [33] also provides convergence rates estimates, including per iteration estimates for the amount of feasibility violation, as well as upper and lower bounds for the primal objective function. We also want to note that in the case that the standard LP relaxation is not tight, the approximate primal solution given in Eqs. (15), (16) may be able to be used to (approximately) find violated constraints that, when included, would tighten the relaxation. For example, one could search for violated cycle inequalities [43].

V. Theoretical properties

As already explained, our method tries to solve problem (10), which is the Lagrangian relaxation of problem (9). 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* (10) *is equivalent to the LP relaxation of* (2), i.e *the LP relaxation of the original integer programming formulation for the MRF problem.*

Proof: To form the Lagrangian relaxation, we relaxed constraints $\mathbf{x}_p^{\mathcal{T}} = \mathbf{x}_p$ of (9), but we kept constraints $\mathbf{x}^{\mathcal{T}} \in \mathcal{X}^{\mathcal{T}}$. The Lagrangian dual is then known to be equivalent to the following relaxation of (9):

$$\min_{\{\mathbf{x}^{\mathcal{T}}\}, \mathbf{x}} \{E(\mathbf{x}, \boldsymbol{\theta}) \mid \mathbf{x}_p^{\mathcal{T}} = \mathbf{x}_p, \ \mathbf{x}_{pq}^{\mathcal{T}} = \mathbf{x}_{pq}, \ \mathbf{x}^{\mathcal{T}} \in \textsc{ConvexHull}(\mathcal{X}^{\mathcal{T}})\} \tag{17}$$

Indeed, it holds that:

$$\begin{aligned}
\textsc{Lagr. Dual} &= \max_{\{\boldsymbol{\lambda}^{\mathcal{T}}\}} \min_{\{\boldsymbol{x}^{\mathcal{T}} \in \mathcal{X}^{\mathcal{T}}\}, \boldsymbol{x}} \sum_{\mathcal{T} \in \mathcal{T}(\mathcal{G})} E(\boldsymbol{\theta}^{\mathcal{T}}, \boldsymbol{x}^{\mathcal{T}}) + \sum_{\mathcal{T} \in \mathcal{T}(\mathcal{G})} \boldsymbol{\lambda}^{\mathcal{T}} \cdot (\boldsymbol{x}^{\mathcal{T}} - \boldsymbol{x}_{|\mathcal{T}}) \\
&= \max_{\{\boldsymbol{\lambda}^{\mathcal{T}}\}} \min_{\{\boldsymbol{x}^{\mathcal{T}} \in \textsc{ConvexHull}(\mathcal{X}^{\mathcal{T}})\}, \boldsymbol{x}} \sum_{\mathcal{T} \in \mathcal{T}(\mathcal{G})} E(\boldsymbol{\theta}^{\mathcal{T}}, \boldsymbol{x}^{\mathcal{T}}) + \sum_{\mathcal{T} \in \mathcal{T}(\mathcal{G})} \boldsymbol{\lambda}^{\mathcal{T}} \cdot (\boldsymbol{x}^{\mathcal{T}} - \boldsymbol{x}_{|\mathcal{T}}) \\
&= \min_{\{\boldsymbol{x}^{\mathcal{T}} \in \textsc{ConvexHull}(\mathcal{X}^{\mathcal{T}})\}, \boldsymbol{x}} \max_{\{\boldsymbol{\lambda}^{\mathcal{T}}\}} \sum_{\mathcal{T} \in \mathcal{T}(\mathcal{G})} E(\boldsymbol{\theta}^{\mathcal{T}}, \boldsymbol{x}^{\mathcal{T}}) + \sum_{\mathcal{T} \in \mathcal{T}(\mathcal{G})} \boldsymbol{\lambda}^{\mathcal{T}} \cdot (\boldsymbol{x}^{\mathcal{T}} - \boldsymbol{x}_{|\mathcal{T}}) \\
&= \min_{\{\boldsymbol{x}^{\mathcal{T}} \in \textsc{ConvexHull}(\mathcal{X}^{\mathcal{T}})\}, \boldsymbol{x}^{\mathcal{T}} = \boldsymbol{x}_{|\mathcal{T}}} \sum_{\mathcal{T} \in \mathcal{T}(\mathcal{G})} E(\boldsymbol{\theta}^{\mathcal{T}}, \boldsymbol{x}^{\mathcal{T}}) \\
&= \min_{\{\mathbf{x}^{\mathcal{T}}\}, \mathbf{x}} \{ E(\mathbf{x}, \boldsymbol{\theta}) \mid \mathbf{x}_p^{\mathcal{T}} = \mathbf{x}_p, \ \mathbf{x}_{pq}^{\mathcal{T}} = \mathbf{x}_{pq}, \ \mathbf{x}^{\mathcal{T}} \in \textsc{ConvexHull}(\mathcal{X}^{\mathcal{T}}) \}.
\end{aligned}$$

For a tree $\mathcal{T}$, however, the set $\textsc{ConvexHull}(\mathcal{X}^{\mathcal{T}})$ coincides with the local marginal polytope that results from the set $\mathcal{X}^{\mathcal{T}}$ if we relax its $\{0, 1\}$ constraints to the constraints $\mathbf{x}^{\mathcal{T}} \geq 0$. Based on this fact, the theorem follows trivially from (17). ■

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 one of the simplest theorems related to that method (see proposition 2.2 in [32] for a proof of a generalized version of this theorem):

Theorem 2. *If the sequence of multipliers* $\{\alpha_t\}$ *satisfies*

$$\alpha_t \geq 0, \ \lim_{t \to \infty} \alpha_t = 0, \ \sum_{t=0}^{\infty} \alpha_t = \infty \ , \tag{18}$$

then the subgradient algorithm converges to the optimal solution of (10).

A sequence $\{a_t\}$ that satisfies condition (18) is also known as a *diminishing step size rule*. A typical such example is given by $a_t = \frac{a}{\sqrt{t}}$, where $a > 0$. Convergence to an optimal solution is also guaranteed in other cases as well, *e.g*, for a *nonsummable diminishing step length* sequence or a *square summable but not summable* sequence [32]. In the former case the multipliers $\{a_t\}$ are chosen as $a_t = \frac{b_t}{\|\nabla g_t\|_2}$, where ∇g_t denotes the subgradient computed at the t-th iteration and

$$b_t \geq 0, \ \lim_{t \to \infty} b_t = 0, \ \sum_{t=0}^{\infty} b_t = \infty \ , \tag{19}$$

whereas in the latter case the multipliers $\{a_t\}$ satisfy

$$\alpha_t \geq 0, \; \sum_{t=0}^{\infty} \alpha_t^2 < \infty, \; \sum_{t=0}^{\infty} \alpha_t = \infty \;\; , \tag{20}$$

with a typical example of a sequence satisfying (20) being $\alpha_t = \frac{a}{b+t}$, where $a > 0, b \geq 0$. A popular way for choosing multipliers $\{\alpha_t\}$ is also via using a so called *adaptive (or dynamic)* step size rule. Contrary to rules (18)-(20) that determine the multipliers $\{\alpha_t\}$ a priori, an adaptive step size rule dynamically adjusts the multipliers $\{\alpha_t\}$ during the execution of the subgradient method. To this end, it can even use information from previous iterations of the algorithm. Rules of this type are usually preferred in practice as they typically lead to faster convergence (see section VII for a concrete example of an adaptive step size rule).

Interestingly, the key quantity to proving convergence of the subgradient method in the case that any of the above mentioned step size rules is used (but also in any other case) is not the objective function value (which may temporarily fluctuate), but the Euclidean distance to an optimal solution. This is made more precise in the following statement, whose proof follows from the fact that the angle between the current subgradient and the vector formed by the difference of the current iterate with an optimal solution is less than 90 degrees:

Theorem 3. *The Euclidean distance of the current solution $\{\boldsymbol{\theta}^{\mathcal{T}}\}$ to the set of optimal solutions decreases per iteration (for a proof see proposition 6.3.1 in [2]).*

The above theorem is to be distinguished from the type of guarantees given by the TRW-S method: in TRW-S, the dual objective improves monotonically, but the algorithm can get stuck; with subgradient methods, the dual objective does not improve monotonically, but the distance to the optimal solution does. Note that one can construct a monotonically improving variant of the subgradient method by working with the so called ϵ-subdifferential of the objective function (see also section VII). Intuitively, such a set includes not only the subgradients at the current iterate, but also the subgradients for points that lie nearby (called ϵ-subgradients). The resulting algorithm chooses an ϵ-subgradient of minimum norm as the next ascent direction, and is known as the ϵ-ascent method [3]. This method can be implemented by either doing an explicit computation of the ϵ-subdifferential (whenever the problem structure allows for that possibility) or by approximating that set with a finite number of ϵ-subgradients.

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 [49]. If a TRW fixed point, say $\{\bar{\boldsymbol{\theta}}^{\mathcal{T}}\}$, satisfies TA, an optimal solution to the original MRF problem can then be extracted. A much more general condition was later introduced in [21], called the *weak tree agreement* (WTA). This condition has also been used to provide further optimality results for TRW algorithms [22]. 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 §VII, the converse is not true, *i.e*, a solution $\{\bar{\boldsymbol{\theta}}^{\mathcal{T}}\}$ satisfying WTA is not necessarily optimal with respect to the Lagrangian dual problem (10)).

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

Proof: Let $\{\bar{\boldsymbol{\theta}}^{\mathcal{T}}\}$ be a solution generated by our algorithm (*i.e*, $\bar{\boldsymbol{\theta}}^{\mathcal{T}}$ are the current potentials for the slave MRF corresponding to tree T). Let us suppose it does not satisfy WTA. We will then show that $\{\bar{\boldsymbol{\theta}}^{\mathcal{T}}\}$ can be perturbed to give a solution that achieves a higher dual objective. Indeed, in this case there will be, *e.g*, a node p for which none of its labels will be optimal for all the tree-structured slave MRFs containing that node. If we assume w.l.o.g. that p is contained only in trees $\mathcal{T}_1$ and $\mathcal{T}_2$, it will then hold:

$$\textsc{Opt}_{\mathcal{T}_1} \cap \textsc{Opt}_{\mathcal{T}_2} = \varnothing \ ,$$

where $\textsc{Opt}_{\mathcal{T}_1}$, $\textsc{Opt}_{\mathcal{T}_2}$ denote the sets of optimal labels for node p in the slave MRFs corresponding to $\mathcal{T}_1$ and $\mathcal{T}_2$ respectively. By applying the following transformation to the unary potentials at p:

$$\bar{\theta}_p^{\mathcal{T}_1}(l) \mathrel{+}= \epsilon, \quad \bar{\theta}_p^{\mathcal{T}_2}(l) \mathrel{-}= \epsilon \ , \quad \forall l \in \textsc{Opt}_{\mathcal{T}_1}$$

it is obvious that the objective value of the Lagrangian dual (10) will increase by ϵ (for an ϵ that is small enough). This is impossible, however, since, by theorem 2 above, $\{\bar{\boldsymbol{\theta}}^{\mathcal{T}}\}$ is already an optimal solution to (10). ■

The above theorem implies that all optimality results related to WTA carry over to our algorithm. Here we simply state just one of them [22]:

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

VI. Beyond tree-structured subproblems

Based on what was discussed up to now, one may get the false impression that our framework applies only to the case where the subproblems (used in the decomposition of the MRF) are tree structured. However, the proposed framework is much more general than that and allows decomposing the MRF optimization task into subproblems in many other ways (as we shall see, this can even lead to algorithms that do not rely entirely on message-passing). Roughly speaking, the only requirement needed is that a global optimizer can be computed for each of the chosen subproblems (something which is obviously true for any tree-structured MRF). If this condition is satisfied, our framework can then be applied, thus allowing a *unified* treatment of all such cases.

More generally, let us consider N subproblems denoted hereafter by SP^i, where $i \in \{1, 2, \dots, N\}$. These subproblems need not be MRF optimization problems, but we simply assume that they are defined as follows:

$$\begin{aligned} \mathrm{SP}^i(\boldsymbol{\theta}^{\mathcal{G}_i}) = \min_{\boldsymbol{x}^{\mathcal{G}_i}} \quad & \boldsymbol{\theta}^{\mathcal{G}_i} \cdot \boldsymbol{x}^{\mathcal{G}_i} \\ \text{s.t.} \quad & \boldsymbol{x}^{\mathcal{G}_i} \in \textsc{feasible}^i . \end{aligned} \tag{21}$$

In the above definition, $\mathcal{G}_i = (\mathcal{V}_i, \mathcal{E}_i)$ denotes a subgraph of $\mathcal{G}$ (*i.e*, $\mathcal{G}_i \subseteq \mathcal{G}$) associated with the i-th subproblem. Also, the vector $\boldsymbol{x}^{\mathcal{G}_i}$ denotes a (different) set of variables associated with each subproblem SP^i, while the objective function of SP^i is a linear function with parameters specified by the vector $\boldsymbol{\theta}^{\mathcal{G}_i}$. Note that there is one to one correspondence between the elements of vectors $\boldsymbol{\theta}^{\mathcal{G}_i}$, $\boldsymbol{x}^{\mathcal{G}_i}$ and the elements (*i.e*, the vertices and edges) of the graph $\mathcal{G}_i$ (*e.g*, it holds $\boldsymbol{x}^{\mathcal{G}_i} = \{x_p^{\mathcal{G}_i}\}_{p \in \mathcal{V}_i \cup \mathcal{E}_i}$ and similarly for $\boldsymbol{\theta}^{\mathcal{G}_i}$). Vector $\boldsymbol{x}^{\mathcal{G}_i}$ must also belong to the set $\textsc{feasible}^i$ associated with the i-th subproblem, which encodes the constraints that the solutions of SP^i must satisfy.

Let us also denote by $\boldsymbol{\theta}^{\mathcal{G}}$ and $\boldsymbol{x}^{\mathcal{G}}$ the parameters and variables of the MRF whose energy $\boldsymbol{\theta}^{\mathcal{G}} \cdot \boldsymbol{x}^{\mathcal{G}}$ we want to minimize. Note that, in general, it will hold $\mathcal{G}_i \subset \mathcal{G}$, as well as $\mathcal{G}_i \neq \mathcal{G}_j$. As a result, the elements of vectors $\boldsymbol{\theta}^{\mathcal{G}_i}$, $\boldsymbol{\theta}^{\mathcal{G}_j}$, $\boldsymbol{\theta}^{\mathcal{G}}$ may not be in one to one correspondence. However, to reduce notational clutter when manipulating (*e.g*, comparing or adding) such vectors, we will hereafter assume that all of them are defined on the vertices and edges of graph $\mathcal{G}$ by appropriately padding them with zero elements (the same, of course, applies to vectors $\boldsymbol{x}^{\mathcal{G}_i}$, $\boldsymbol{x}^{\mathcal{G}_j}$, $\boldsymbol{x}^{\mathcal{G}}$).

Based on the above definitions, we will hereafter say that subproblems $\{\mathrm{SP}^i\}$ provide a *decomposition* of an MRF whose parameters are $\boldsymbol{\theta}^{\mathcal{G}}$ if the following 2 conditions hold true:

- The first condition requires that the MRF objective function (*i.e*, the MRF energy) should be decomposed as the sum of the objective functions of the subproblems. To this end, we require:

  $$\boldsymbol{\theta}^{\mathcal{G}} = \sum_i \boldsymbol{\theta}^{\mathcal{G}_i} \quad . \tag{22}$$

  Obviously, the above condition implies that it should also hold:

  $$\mathcal{G} = \cup \mathcal{G}_i \quad . \tag{23}$$

  Note that it is not required to hold $\mathcal{G}_i \cap \mathcal{G}_j = \varnothing$ for $i \neq j$.
- The second condition is related to the sets $\textsc{feasible}^i$ encoding the constraints that the solutions of the subproblems have to satisfy. More specifically, it should hold:

  $$\left\{ \{\boldsymbol{x}^{\mathcal{G}_i}\}, \boldsymbol{x}^{\mathcal{G}} \mid \boldsymbol{x}^{\mathcal{G}_i} \in \textsc{feasible}^i, \boldsymbol{x}^{\mathcal{G}_i}_{|\mathcal{G}_i \cap \mathcal{G}} = \boldsymbol{x}^{\mathcal{G}}_{|\mathcal{G}_i \cap \mathcal{G}} \right\} \supseteq \left\{ \{\boldsymbol{x}^{\mathcal{G}_i}\}, \boldsymbol{x}^{\mathcal{G}} \mid \boldsymbol{x}^{\mathcal{G}} \in \mathcal{X}^{\mathcal{G}}, \boldsymbol{x}^{\mathcal{G}_i}_{|\mathcal{G}_i \cap \mathcal{G}} = \boldsymbol{x}^{\mathcal{G}}_{|\mathcal{G}_i \cap \mathcal{G}} \right\} \quad . \tag{24}$$

  For example, due to (23), one way to ensure the above condition is by simply requiring:

  $$\textsc{feasible}^i \supseteq \mathcal{X}^{\mathcal{G}_i}, \quad \forall i \in \{1, 2, \ldots, N\} \quad . \tag{25}$$

Algorithm 1: DD-MRF

```
**INPUT:**
  A decomposition $\{\mathrm{SP}^i(\cdot)\}_{i=1}^N$ of the form (21).

**PROCEDURE:**
  **repeat** until convergence
    – $\mathrm{SLAVE}^i(\boldsymbol{\lambda}^{\mathcal{G}_i}) := \mathrm{SP}^i(\boldsymbol{\theta}^{\mathcal{G}_i} + \boldsymbol{\lambda}^{\mathcal{G}_i})$
    – Solve slave problems by computing $\bar{\mathbf{x}}^{\mathcal{G}_i} = \mathtt{minimizer}(\mathrm{SLAVE}^i(\boldsymbol{\lambda}^{\mathcal{G}_i}))$
    – For each $i$, update parameters $\boldsymbol{\lambda}^{\mathcal{G}_i}$ of the $i$-th slave problem:
      $\forall$ node $p$ or edge $pq \in \mathcal{G}_i$ apply formulas (27), (28)
  **end repeat**
```

Note that in constraint (25) the set FEASIBLE^i may well be strictly larger than $\mathcal{X}^{\mathcal{G}_i}$, which corresponds to a relaxation of the marginal polytope.

If conditions (22), (24) hold true, then by using a similar reasoning as in section IV it is easy to prove that the DD-MRF algorithm reduces to the algorithm 1 shown on the next page. In this case, the i-th slave problem (denoted by SLAVE^i) turns out to have the same form as an i-th subproblem SP^i. More specifically, it holds:

$$\mathrm{SLAVE}^i(\boldsymbol{\lambda}^{\mathcal{G}_i}) \equiv \mathrm{SP}^i(\boldsymbol{\theta}^{\mathcal{G}_i} + \boldsymbol{\lambda}^{\mathcal{G}_i}) \ . \tag{26}$$

Furthermore, the Lagrangian multipliers are updated according to the following formulas:

$$\boldsymbol{\lambda}_p^{\mathcal{G}_i} {+}{=}\ \alpha_t \cdot \left(\bar{\mathbf{x}}_p^{\mathcal{G}_i} - \frac{\sum_{j \in \mathcal{J}(p)} \bar{\mathbf{x}}_p^{\mathcal{G}_j}}{|\mathcal{J}(p)|} \right) \ , \quad \text{where } \mathcal{J}(p) = \{ j \,|\, p \in G_j \} \tag{27}$$

$$\boldsymbol{\lambda}_{pq}^{\mathcal{G}_i} {+}{=}\ \alpha_t \cdot \left(\bar{\mathbf{x}}_{pq}^{\mathcal{G}_i} - \frac{\sum_{j \in \mathcal{J}(pq)} \bar{\mathbf{x}}_{pq}^{\mathcal{G}_j}}{|\mathcal{J}(pq)|} \right) \ , \quad \text{where } \mathcal{J}(pq) = \{ j \,|\, pq \in G_j \} \ . \tag{28}$$

The following theorem is then derived:

Theorem 6. *The optimization problem corresponding to any decomposition $\{\mathrm{SP}^i\}$ is a relaxation to the original MRF optimization problem (and the optimum of that relaxation is computed by the DD-MRF algorithm).*

Before proceeding, let us mention that in the particular case where FEASIBLE^i $= \mathcal{X}^{\mathcal{G}_i}$, then the i-th subproblem as well as the i-th slave problem reduce to optimizing the energy of an MRF defined on graph $\mathcal{G}_i$. Note, however, that this may not be necessarily the case and even more general subproblems can be used as well. For instance, matchings is an explicit example of another type of non tree-structured subproblem that can be used with our framework [8]. We refer the interested reader to [48] for a nice application of the dual decomposition framework in this regard (capable of obtaining globally optimal solutions to the graph matching problem)..

A. *Deciding what subproblems to use*

As explained above, there is a great flexibility with regard to the choice of the subproblems that can be used for decomposing an MRF optimization problem. Hence, a question that naturally arises is the following one: how does this choice influence the effectiveness of MRF optimization? Put otherwise, given two different decompositions (that both satisfy the above mentioned conditions (22), (24)), how can we decide which one is more powerful? The answer to this question is provided in a rigorous manner in the following theorem:

Theorem 7. *Given any decomposition* $\{\mathrm{SP}^i\}$, *let us define the following set:*[6]

$$\textsc{Domain}(\{\mathrm{SP}^i\}) = \left\{ \{\boldsymbol{x}^{\mathcal{G}_i}\}, \boldsymbol{x}^{\mathcal{G}} \;\middle|\; \boldsymbol{x}^{\mathcal{G}_i} \in \textsc{ConvexHull}(\textsc{feasible}^i),\ \boldsymbol{x}^{\mathcal{G}_i}_{|\mathcal{G}_i \cap \mathcal{G}} = \boldsymbol{x}^{\mathcal{G}}_{|\mathcal{G}_i \cap \mathcal{G}} \right\} \ . \tag{29}$$

Let $\{\mathrm{SP}^i_0\}$, $\{\mathrm{SP}^j_1\}$ *be 2 different decompositions of a given MRF optimization problem. It then holds that decomposition* $\{\mathrm{SP}^i_0\}$ *is stronger*[7] *than decomposition* $\{\mathrm{SP}^j_1\}$ *if*

$$\textsc{Domain}(\{\mathrm{SP}^i_0\}) \subseteq \textsc{Domain}(\{\mathrm{SP}^j_1\}) \ . \tag{30}$$

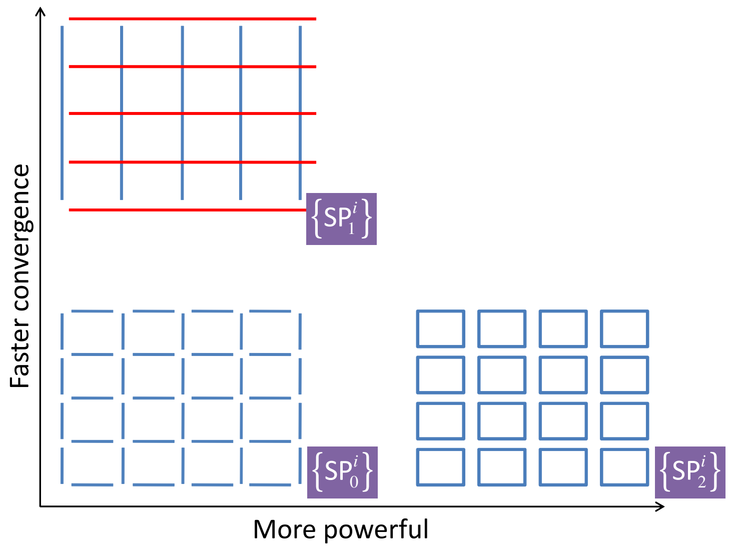
Proof: Let LR_0, LR_1 be the relaxations corresponding to $\{\mathrm{SP}^i_0\}$, $\{\mathrm{SP}^j_1\}$. Since both $\{\mathrm{SP}^i_0\}$, $\{\mathrm{SP}^j_1\}$ are decompositions of the same MRF, the objective functions of relaxations LR_0, LR_1 will be equal to each other (due to (22)). To prove the theorem, it thus suffices to show that $\textsc{Domain}(\{\mathrm{SP}^i_0\})$ (respectively $\textsc{Domain}(\{\mathrm{SP}^j_1\})$) is the feasible set of relaxation LR_0 (respectively LR_1). Indeed, it holds that:

$$\begin{aligned}
\mathrm{LR}_0 &= \max_{\boldsymbol{\lambda}^{\mathcal{G}_i}} \min_{\boldsymbol{x}^{\mathcal{G}_i} \in \textsc{feasible}^i} \sum_i \boldsymbol{\theta}^{\mathcal{G}_i} \cdot \boldsymbol{x}^{\mathcal{G}_i} + \sum_i \boldsymbol{\lambda}^{\mathcal{G}_i} \cdot (\boldsymbol{x}^{\mathcal{G}_i}_{|\mathcal{G}_i \cap \mathcal{G}} - \boldsymbol{x}^{\mathcal{G}}_{|\mathcal{G}_i \cap \mathcal{G}}) &(31)\\
&= \max_{\boldsymbol{\lambda}^{\mathcal{G}_i}} \min_{\boldsymbol{x}^{\mathcal{G}_i} \in \textsc{ConvexHull}(\textsc{feasible}^i)} \sum_i \boldsymbol{\theta}^{\mathcal{G}_i} \cdot \boldsymbol{x}^{\mathcal{G}_i} + \sum_i \boldsymbol{\lambda}^{\mathcal{G}_i} \cdot (\boldsymbol{x}^{\mathcal{G}_i}_{|\mathcal{G}_i \cap \mathcal{G}} - \boldsymbol{x}^{\mathcal{G}}_{|\mathcal{G}_i \cap \mathcal{G}}) &(32)\\
&= \min_{\boldsymbol{x}^{\mathcal{G}_i} \in \textsc{ConvexHull}(\textsc{feasible}^i)} \max_{\boldsymbol{\lambda}^{\mathcal{G}_i}} \sum_i \boldsymbol{\theta}^{\mathcal{G}_i} \cdot \boldsymbol{x}^{\mathcal{G}_i} + \sum_i \boldsymbol{\lambda}^{\mathcal{G}_i} \cdot (\boldsymbol{x}^{\mathcal{G}_i}_{|\mathcal{G}_i \cap \mathcal{G}} - \boldsymbol{x}^{\mathcal{G}}_{|\mathcal{G}_i \cap \mathcal{G}}) &(33)\\
&= \min_{\left\{\{\boldsymbol{x}^{\mathcal{G}_i}\}, \boldsymbol{x}^{\mathcal{G}}\right\} \in \textsc{Domain}(\{\mathrm{SP}^i_0\})} \sum_i \boldsymbol{\theta}^{\mathcal{G}_i} \cdot \boldsymbol{x}^{\mathcal{G}_i} \ . &(34)
\end{aligned}$$

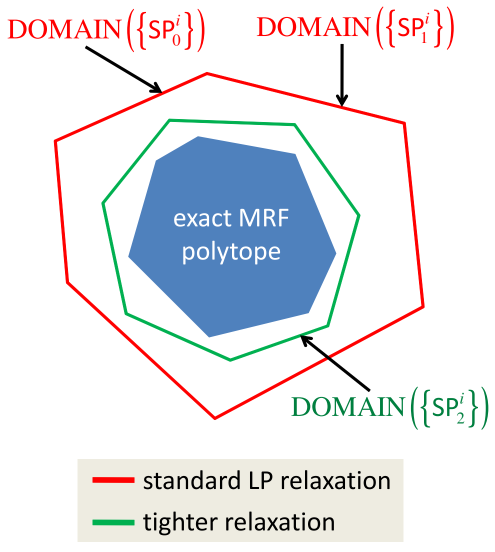
And similarly for LR_1. Note that (32) results from the fact that a linear function always attains its optimum at an extreme point of its domain, while the exchange of $\min$ and $\max$ in (33) is a result of strong duality. ■

[6]Note that $\textsc{feasible}^i$ refers merely to a superset of $\mathcal{G}_i$ (see Eq. (25)). Since the latter set is discrete (and thus non-convex), $\textsc{feasible}^i$ may be non-convex as well. Therefore, $\textsc{ConvexHull}(\textsc{feasible}^i)$ does not necessarily coincide with $\textsc{feasible}^i$ in Theorem 7 (and elsewhere).

[7]By a stronger decomposition, we mean one that leads to a tighter (*i.e*, higher) dual lower bound, thus yielding a tighter dual relaxation.



(a)



(b)

Fig. 5: **(a)** Three different decompositions of an MRF (defined on a 4×4 grid $\mathcal{G}$) into smaller subproblems (which, in this case, are also MRFs defined on subgraphs of $\mathcal{G}$): Decomposition $\{\mathrm{SP}_0^i\}$ consists of one subproblem per edge. Decomposition $\{\mathrm{SP}_1^i\}$ consists of one subproblem per row and column. Decomposition $\{\mathrm{SP}_2^i\}$ consists of one subproblem per 1×1 cell. Due to the fact that decomposition $\{\mathrm{SP}_1^i\}$ consists of larger subproblems than decomposition $\{\mathrm{SP}_0^i\}$, it is expected to converge faster. However, with regard to tightness, these two decompositions are completely equivalent to each other. On the other hand, decomposition $\{\mathrm{SP}_2^i\}$ is stronger than both $\{\mathrm{SP}_0^i\}$ and $\{\mathrm{SP}_1^i\}$ as its slave MRFs are not expected to have the integrality property since they are defined on loopy subgraphs. As a result, when moving upwards along the vertical axis we gain in speed, whereas when moving right along the horizontal axis we gain in power. **(b)** The relaxations corresponding to decompositions $\{\mathrm{SP}_0^i\}$ and $\{\mathrm{SP}_1^i\}$ coincide with the standard LP relaxation of MRF problem (2) and thus all have the same feasible set. On the other hand, the relaxation corresponding to decomposition $\{\mathrm{SP}_2^i\}$ is tighter and thus has a smaller feasible set.

As an illustration, we next provide some examples of possible subproblems that can be used in our decomposition framework.

B. Tightening the LP relaxation

One choice for the subproblems is to optimize discrete MRFs restricted on subgraphs $\mathcal{G}_i$ of $\mathcal{G}$ (this will be the case when $\textsc{feasible}^i = \mathcal{X}^{\mathcal{G}_i}$). Let us hereafter denote the resulting MRFs by $\mathrm{MRF}(G_i)$ and the original MRF by $\mathrm{MRF}(G)$. Note that the subgraphs $\mathcal{G}_i$ do not necessarily have to be trees and so this is a more general decomposition than the tree-structured decomposition used in section IV. On the other hand, since $\mathcal{G}_i$ may not be trees, the resulting subproblems will be more difficult to solve. Hence, a natural question that arises is if we actually gain anything by putting an extra effort in solving these more difficult subproblems. The answer is provided by theorem 9 below (whose proof is based on theorem 7 above). Before stating that theorem, let us first make the following definition:

Definition 8. *We say that a discrete Markov Random Field (defined on a graph G) has the* *integrality property if the standard LP relaxation associated to that MRF is tight, ie, the feasible set of that relaxation coincides with the convex hull of* $\mathcal{X}^G$.

Theorem 9. *The DD-MRF algorithm solves a relaxation which is strictly tighter than the standard LP relaxation associated with* $\text{MRF}(G)$ *only if at least one of the* $\text{MRF}(\mathcal{G}_i)$ *does not have the integrality property.*

Proof: If all slaves $\text{MRF}(\mathcal{G}_i)$ have the integrality property then none of the convex sets $\textsc{ConvexHull}(\textsc{feasible}^i) = \textsc{ConvexHull}(\mathcal{X}^{\mathcal{G}_i})$ will change if we replace the $\{0,1\}$ constrains in $\mathcal{X}^{\mathcal{G}_i}$ by the constraints $\boldsymbol{x}^{\mathcal{G}_i} \geq 0$. This implies that $\textsc{Domain}(\{\text{MRF}(\mathcal{G}_i)\})$ will coincide with the feasible set of the standard LP-relaxation, *i.e*, the local marginal polytope $\textsc{Local}(\mathcal{G}_i)$, and so the theorem follows directly from theorem 7. ■

The above theorem essentially generalizes theorem 1. On the one hand, it thus confirms the known fact that all tree structured decompositions are equally powerful, *i.e*, the choice of trees does not matter with regard to the tightness of the relaxation optimized by DD-MRF as long as these trees cover the graph $\mathcal{G}$ (this, of course, results directly from theorem 9 and the fact that any tree structured MRF is known to have the integrality property). On the other hand, it tells us that if we want to better approximate our original NP-hard MRF optimization problem (and thus obtain better solutions), we can, *e.g*, use loopy MRFs as subproblems (this is due to the fact that, in general, loopy MRFs are known not to have the integrality property). Of course, DD-MRF requires that a global minimizer can be computed for the subproblems. Hence, for having a practical algorithm, these global minimizers need to be computed efficiently. One possible choice is thus to use loopy MRFs with, *e.g*, small tree-width, as these MRFs can be efficiently optimized via the Junction Tree algorithm. Before proceeding, we should also note that, although all tree structured decompositions are equivalent to each other, the choice of trees does affect the speed of convergence of the DD-MRF algorithm, *i.e*, larger trees (and more generally larger subgraphs) do help DD-MRF to converge faster. The above conclusions are summarized and illustrated in Fig. 5.

C. Submodular subproblems

Of course, if we decide to use MRFs as subproblems, the topology of graphs $\mathcal{G}_i$ may not be the only criterion for deciding what MRFs to select. Instead, one may choose MRFs based on the type of their potentials, something which may lead to algorithms that do not rely entirely on message-passing. For instance, one may choose MRFs that are submodular[8]. One reason is that submodular MRFs can be very efficiently optimized using graph-cut based algorithms, regardless of whether

[8]A function f on a lattice $(X, \leq)$ is called submodular if the following relation holds for all $x, y \in X$: $f(x \wedge y) + f(x \vee y) \leq f(x) + f(y)$, where $\wedge$, $\vee$ represent respectively the meet and join operation.

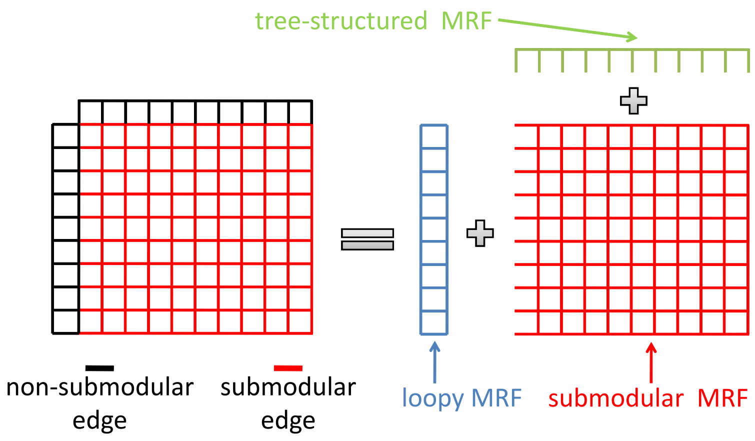


Fig. 6: A decomposition of an MRF into 3 subproblems for which their global minimum can be computed efficiently: a loopy MRF with small tree width, a submodular MRF and a tree-structured MRF. Our framework can be applied to this case as well and provides a principled way of combining the subproblems' solutions for optimizing the MRF on the full grid.

their associated graphs contain cycles or not. As a result, if, for example, a large part of an existing MRF is submodular, the MRF corresponding to that part may be used as a subproblem, something which can significantly accelerate the convergence of the DD-MRF algorithm. Furthermore, an additional advantage from using submodular MRFs as subproblems comes from the fact that there have also been developed extremely efficient graph-cut based techniques for optimizing submodular MRFs that are dynamic (*i.e*, that have time varying parameters) [20], [26]. This becomes very relevant in the case of the DD-MRF algorithm, because (if $\text{FEASIBLE}^i = \mathcal{X}^{\mathcal{G}_i}$) that algorithm essentially proceeds by solving N dynamic MRF problems. To better see that, it suffices to notice that, in this case, the i-th slave problem corresponds to an MRF whose parameters (*i.e*, potentials) vary according to $\boldsymbol{\theta}^{\mathcal{G}_i} + \boldsymbol{\lambda}^{\mathcal{G}_i}$ per iteration, *i.e*, it is dynamic. In fact, as the DD-MRF algorithm converges, only a small number of elements of vector $\boldsymbol{\lambda}^{\mathcal{G}_i}$ will typically change per iteration (to see this, recall that if all subproblems agree on their label for a node p, then, the element $\lambda_p^{\mathcal{G}_i}$ of $\boldsymbol{\lambda}^{\mathcal{G}_i}$ corresponding to that node does not change). This implies that as the DD-MRF algorithm converges, smaller and smaller changes will be applied to the parameters of each dynamic MRF. As a result, its optimization will become even faster. Although using submodular MRFs can potentially be of great benefit with regard to speed, we note, however, that submodular MRFs have the integrality property and, so, (by theorem 9) their use will not tighten the solved relaxation. For a practical implementation of submodular decomposition, one also needs to be able to detect what submodular subproblems to use. This can, for instance, be done on a per application basis, by relying on the a priori knowledge that one has about the structure of a particular MRF or a certain class of MRFs (in this case the usage of the method will be problem dependent). However, detection may also proceed in an automatic manner, given that checking the submodularity of a pairwise MRF subproblem is equivalent to checking the submodularity of all its pairwise potentials, an operation that can be performed very efficiently in many cases. Of course, for being to apply a submodular decomposition to a MAP estimation task, we must ensure that the submodularity of each slave MRF(G_i) is maintained as $\boldsymbol{\lambda}^{\mathcal{G}_i}$ changes. This is very easily achieved in our framework: for instance, it can be done simply by choosing to update only the unary potentials of the corresponding slave MRFs, but not their pairwise potentials. The latter will thus remain submodular, and so the same thing will hold for the energies of the corresponding subproblems.

An additional comment is in order here: for submodular MRFs, even if we choose to do a partial update by modifying only the unary potentials of the slaves, we can still guarantee that we are solving the same dual relaxation as before. This is more generally true when all slave MRFs satisfy the integrality property (see Definition 8). It is very easy to see this by moving to the primal domain, and examining the equivalent primal relaxations R_{partial}, R_{full} for the partial and full updates respectively. As explained earlier, these relaxations take the following form if the slave MRFs have the integrality property:

$$R_{\text{partial}} = \min_{\{\mathbf{x}^{\mathcal{G}_i}\},\mathbf{x}} \{\sum_i E(\boldsymbol{\theta}^{\mathcal{G}_i}, \mathbf{x}^{\mathcal{G}_i}) \mid \mathbf{x}_p^{\mathcal{G}_i} = \mathbf{x}_p,\ \mathbf{x}^{\mathcal{G}_i} \in \text{LOCAL}(\mathcal{G}_i)\} \tag{35}$$

$$R_{\text{full}} = \min_{\{\mathbf{x}^{\mathcal{G}_i}\},\mathbf{x}} \{\sum_i E(\boldsymbol{\theta}^{\mathcal{G}_i}, \mathbf{x}^{\mathcal{G}_i}) \mid \mathbf{x}_p^{\mathcal{G}_i} = \mathbf{x}_p,\ \mathbf{x}_{pq}^{\mathcal{G}_i} = \mathbf{x}_{pq},\ \mathbf{x}^{\mathcal{G}_i} \in \text{LOCAL}(\mathcal{G}_i)\}\ , \tag{36}$$

where $\text{LOCAL}(\mathcal{G}_i)$ refers to the local marginal polytope for subgraph $\mathcal{G}_i$ (recall that this polytope results from the marginal polytope $\mathcal{X}^{\mathcal{G}_i}$ after we relax its integrality constraints to $\mathbf{x}_p^{\mathcal{G}_i}, \mathbf{x}_{pq}^{\mathcal{G}_i} \geq 0$). Therefore, to prove that R_{partial} and R_{full} are equivalent relaxations, it suffices to show that if $\{\bar{\mathbf{x}}^{\mathcal{G}_i}\}, \bar{\mathbf{x}}$ is an optimal solution to R_{partial} then there exists a feasible solution to R_{full} of equal cost. Indeed, let us assume without loss of generality that $\boldsymbol{\theta}_p^{\mathcal{G}_i} = \boldsymbol{\theta}_p^{\mathcal{G}_j}$, $\boldsymbol{\theta}_{pq}^{\mathcal{G}_i} = \boldsymbol{\theta}_{pq}^{\mathcal{G}_j}$, $\forall i, j$. Then if we define $\hat{\mathbf{x}}_p^{G_i} = \big(\sum_{j\in\mathcal{J}(p)} \bar{\mathbf{x}}_p^{G_j}\big)/|\mathcal{J}(p)|$, $\hat{\mathbf{x}}_{pq}^{G_i} = \big(\sum_{j\in\mathcal{J}(pq)} \bar{\mathbf{x}}_{pq}^{G_j}\big)/|\mathcal{J}(pq)|$, it is trivial to check that the resulting solution belongs to the feasible set of R_{full} and has the same cost with solution $\{\bar{\mathbf{x}}^{\mathcal{G}_i}\}, \bar{\mathbf{x}}$.

We should mention that in the more general case, where the integrality property is not satisfied by the slave MRFs, the relaxation R_{full} may be tighter than R_{partial}. We should note, however, that even in cases where these two relaxations are equivalent, there may still be a difference in the speed of convergence when using partial updates.

D. Combining subproblems of different types

When applying the DD-MRF algorithm, nothing prevents us from combining different types of subproblems. This is illustrated in the example of Fig. 6, where a large part of the MRF shown in that figure is submodular. Hence, that MRF has been decomposed into a large submodular subproblem, a loopy MRF with small tree width, and a tree-structured MRF. Note that a global minimizer can be computed efficiently for all 3 types of subproblems using respectively graph-cut based techniques, the junction tree algorithm and belief propagation. Of course, as explained earlier, many other type of subproblems are allowed to be used, and the same thing holds for the corresponding inference algorithms, which may chosen to be even more advanced and/or efficient. In this manner, one can easily adapt DD-MRF to fully exploit the structure of the input problem, which is a very important advantage of the proposed framework.

VII. Experimental results

In this section we discuss a few other useful practical aspects of our approach, we describe some further extensions, and we also present experimental results on a wide variety of problems.

Choice of subgradient. As explained in Section IV, a possible subgradient of the function $g^{\mathcal{T}}(\boldsymbol{\lambda}^{\mathcal{T}})$ in Eq. (11) associated with the subproblem for tree $\mathcal{T}$ is given by vector $\bar{\mathbf{x}}^{\mathcal{T}}$, where $\bar{\mathbf{x}}^{\mathcal{T}}$ is set equal to any binary vector that minimizes the slave MRF for $\mathcal{T}$. Similarly, in the more general scenario of Section VI where subproblems correspond to subgraphs $\mathcal{G}_i$, a subgradient of function $g^i(\boldsymbol{\lambda}^{\mathcal{G}_i}) \equiv \textsc{Slave}^i(\boldsymbol{\lambda}^{\mathcal{G}_i})$ in Eq. (26) is given by vector $\bar{\mathbf{x}}^{\mathcal{G}_i}$, which is set equal to any minimizer of the i-th slave subproblem. The choice of these subgradient vectors $\bar{\mathbf{x}}^{\mathcal{G}_i}$ is important since they are used in Eqs. (27), (28) for updating the Lagrangian multipliers $\boldsymbol{\lambda}^{\mathcal{G}_i}$. However, often there exist many possible choices for these subgradients. In fact, the subdifferential $\partial g^i(\boldsymbol{\lambda}^{\mathcal{G}_i})$, *i.e*, the set of all possible subgradients of $g^i(\cdot)$ at $\boldsymbol{\lambda}^{\mathcal{G}_i}$, is equal to the following convex set

$$\partial g^i(\boldsymbol{\lambda}^{\mathcal{G}_i}) = \textsc{ConvexHull}(\bar{\mathbf{x}}^{\mathcal{G}_i} \,|\, \bar{\mathbf{x}}^{\mathcal{G}_i} \text{ minimizer of } i\text{-th subproblem } \textsc{Slave}^i(\boldsymbol{\lambda}^{\mathcal{G}_i}))\ . \tag{37}$$

Therefore one can use as subgradients $\bar{\mathbf{x}}^{\mathcal{G}_i}$ in Eqs. (27), (28) any vector (not necessarily integral)[9] that belongs to the above set $\partial g^i(\boldsymbol{\lambda}^{\mathcal{G}_i})$, *i.e*, is equal to a convex combination of slave minimizers. We can make use of this fact to accelerate the convergence of the algorithm, since some subgradients may be better than others in this regard. For example, in the case of a tree-structured decomposition, one tree may have multiple MAP assignments. All or a subset of these assignments can be computed based on the min-marginals provided by the max-product algorithm. By then choosing the one that agrees the most with the other trees' assignments, one may be able to obtain faster convergence. More generally, due to (37), one can achieve the same goal by trying to choose the best convex combination of these assignments.

Incremental/stochastic subgradient updates. Another very useful idea is that of incremental subgradient updates, where at each iteration the Lagrangian multipliers are changed gradually through a sequence of steps. At each step only a small number of slave MRFs are chosen and only their Lagrangian multipliers are updated. In general, at least two slave MRFs have to be chosen per step, where the second slave is needed so as to balance the first one out. Selecting the slaves can be done at random, in which case a *stochastic subgradient* method results. The slave selection can also be done deterministically, leading to a so called *incremental subgradient* method. One

[9]Note that even in the case where the slaves are chosen to be discrete MRFs (which means that their minimizers $\bar{\mathbf{x}}^{\mathcal{G}_i}$ have to be binary vectors), the subgradient vectors $\bar{\mathbf{x}}^{\mathcal{G}_i}$ used in Eqs. (27), (28) may still be chosen to be non-integral due to (37).

interesting strategy for deterministically choosing the subproblems would be to select a slave whose MAP assignment is very different from the other slaves' MAP assignments (of course, we still need to select another slave to balance it out as mentioned earlier). A simpler strategy is to sequentially visit the nodes and edges of the graph $\mathcal{G}$ in a predefined or random order and at each step to update the Lagrangian multipliers only for those slaves that contain the currently visited element. This essentially amounts to updating at each step the Lagrangian multipliers associated with the current node p (or edge pq) through Eq. (27) (or Eq. (28) respectively). We show for this case a very useful way of setting the subgradient $\bar{\mathbf{x}}_p^{\mathcal{G}_i}$ in Eq. (27) when visiting a node p (the case of visiting an edge pq is analogous, and is thus omitted). To this end, we make use of the following notation: $\mathbf{M}_p^i$ is the vector of min-marginals at node p for the i-th subproblem, while $\bar{\mathbf{M}}_p^i = \mathbf{M}_p^i - \min_{l \in \mathcal{L}} M_p^i(l)$ represents its non-negative normalization (resulting after we subtract the minimum energy $\min_{l \in \mathcal{L}} M_p^i(l)$ of the i-th slave). Also, OPT_p^i denotes the set of optimal labels at p for the i-th subproblem, *i.e*, $\text{OPT}_p^i = \{l \in \mathcal{L} \,|\, \bar{M}_p^i(l) = 0\}$, while $\mathcal{J}(p) = \{j \,|\, p \in \mathcal{G}_j\}$ represents the set of slaves containing p. The subgradient vector $\bar{\mathbf{x}}_p^{\mathcal{G}_i}$ in Eq. (27) can then be set as follows

$$\bar{x}_p^{\mathcal{G}_i}(l) = \begin{cases} \dfrac{\sum_{j \in \mathcal{J}(p), j \neq i} (\bar{M}_p^j(l) + \varepsilon)}{Z_p^i} & , \text{ if } l \in \text{OPT}_p^i \\ 0 & , \text{ if } l \notin \text{OPT}_p^i \end{cases} , \tag{38}$$

where $Z_p^i = \sum_{l \in \text{OPT}_p^i} \left(\sum_{j \in \mathcal{J}(p), j \neq i} (\bar{M}_p^j(l) + \varepsilon) \right)$ is a normalization factor that simply enforces the elements of $\bar{\mathbf{x}}_p^{\mathcal{G}_i}$ to sum to 1 (as they should), while ε is a small positive constant used merely to ensure that at least one component of $\bar{\mathbf{x}}_p^{\mathcal{G}_i}$ is positive and hence $Z_p^i \neq 0$. For example, the use of an $\varepsilon > 0$ is necessary if $\text{OPT}_p^i \subseteq \cap_{j \neq i, j \in \mathcal{J}(p)} \text{OPT}_p^j$, in which case it holds $Z_p^i = 0$ for $\varepsilon = 0$, whereas for $\varepsilon > 0$ Eq. (38) gives

$$\bar{x}_p^{\mathcal{G}_i}(l) = \frac{1}{|\text{OPT}_p^i|} \quad , \quad \forall l \in \text{OPT}_p^i \ .$$

Given that for each $l \in \text{OPT}_p^i$ there exists, by definition, a minimizer of the i-th slave, say $\mathbf{x}^{[l]}$, that satisfies $x_p^{[l]}(l) = 1$, it is then trivial to see that the vector $\bar{\mathbf{x}}_p^{\mathcal{G}_i}$ in (38) is a convex combination of the vectors $\{\mathbf{x}_p^{[l]}\}_{l \in \text{OPT}_p^i}$ and thus provably belongs to the current subdifferential. The intuition behind Eq. (38) is the following one: if $l \in \text{OPT}_p^i$ and the sum $\sum_{j \in \mathcal{J}(p), j \neq i} \bar{M}_p^j(l)$ is large then this means that the slaves have a large disagreement about the assignment of label l to node p (*e.g*, the i-th slave considers l to be optimal, whereas some of the other slaves assign a large min-marginal to it). To reach a consensus about label l, the slaves therefore need to make a large correction to their unary potentials (at p) for label l via applying Eq. (27), which is exactly why element $\bar{x}_p^{\mathcal{G}_i}(l)$ is good to be set proportional to the above sum.

In the above case, besides the standard rules for setting the multipliers a_t in Eqs. (27) and (28), the following rule can be used as well, which also guarantees that the dual objective function increases per iteration:

$$a_t = \min_{j \in \mathcal{J}(p)} Z_p^j \ , \tag{39}$$

where p denotes the node visited at the current step. We often found that the above rule can accelerate the convergence of the incremental subgradient method. Also, it is useful to have an error tolerance ϵ when determining the set of optimal labels OPT_p^i in Eq. (38), *i.e*, we can set $\text{OPT}_p^i = \{l \in \mathcal{L} \,|\, \bar{M}_p^i(l) \leq \epsilon\}$. This also relates to the use of ϵ-subgradients during the algorithm.

Dynamic set of subproblems. Another possibility is that of having the set of subproblems to be updated dynamically. This means that slaves can be added or removed from this set at run time, and this can be decided based on the current information maintained by the algorithm.

Choice of decomposition & decoding method. The decomposition that is chosen to be used by the algorithm plays a very important role. As thoroughly explained in Section VI, it determines how tight the underlying relaxation is, which is closely related to the quality of the estimated solutions, while it also affects the speed of convergence. Unless noted otherwise, we use a tree-structured decomposition in the following examples. Regarding the choice of trees, we prefer large trees (*e.g*, spanning trees) since the larger the trees are the faster the convergence of the algorithm. The choice of decomposition also influences a lot the actual effectiveness of the decoding method used for obtaining the primal solutions. This happens because if the relaxation resulting from the decomposition comprises a bad approximation to the original MRF optimization problem, then we observed that most decoding methods were not very effective in practice. For computer vision problems, we found that overall the decoding method (14) based on BP messages performs better than the Larsson *et al* decoding method in the sense that it often yields good solutions earlier. However, both decoding methods have a minimal computational cost. Therefore, as already mentioned in Section IV-B, the best strategy in a Lagrangian based approach is to generate many feasible primal solutions via the use of non-expensive heuristic procedures and always pick the best one.

Adaptive multiplier update rules. Another issue that we investigated was how to set the positive multipliers $\{\alpha_t\}$, which are used for the update of the dual variables during the subgradient method. In section V we described a few of the simplest methods that can be used for this task. We have also experimented with other schemes as well, including adaptive step size rules which dynamically adjust the multipliers during the execution of the subgradient method and typically lead to faster convergence in practice. Before proceeding, though, we want to note that there is not a single choice of adaptive multiplier updates that yields the optimal performance in each and every case (for example, we often found that heuristic step size rules with no theoretical guarantees could achieve superlinear convergence). However, the adaptive step size rule (40) shown below is theoretically justified and has performed well on average, hence it has been used in the experiments. That rule has the following form:

$$\alpha_t = \gamma_t \frac{\textsc{Approx}_t - \textsc{Dual}_t}{\|\nabla g_t\|^2} . \tag{40}$$

Here, $\textsc{Approx}_t$ is supposed to be an approximation to the unknown optimal dual value, $\textsc{Dual}_t$ denotes the current value of the dual function at the t-th iteration, while ∇g_t denotes the subgradient of the dual function computed at time t. Also, γ_t denotes a positive scalar that is allowed to vary per iteration and typically takes values in the interval $(0, 2)$. The value of $\textsc{Approx}_t$ that approximates the optimal dual value can be chosen in many ways. For instance, one can use the cost of the best primal solution obtained so far, say, $\textsc{BestPrimal}_t$ as an upper bound approximation to the dual optimum, in which case we set $\textsc{Approx}_t = \textsc{BestPrimal}_t$. The intuition behind the resulting multiplier update rule is that, initially, when the primal-dual gap (and hence the numerator $\textsc{BestPrimal}_t - \textsc{Dual}_t$ in (40)) 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 become smaller, $\{\alpha_t\}$ will be assigned smaller values and hence the dual variables will be modified using finer updates. Another option for determining $\textsc{Approx}_t$ is to use a lower bound approximation of the dual optimum based on the best dual value obtained so far, say, $\textsc{BestDual}_t$. In this case we set:

$$\textsc{Approx}_t = \textsc{BestDual}_t + \delta_t \ , \tag{41}$$

where δ_t is updated as follows:

$$\delta_{t+1} = \begin{cases} \rho_0 \delta_t \ , & \text{if dual function improved by } \delta_t \\ \max(\rho_1 \delta_t, \delta) \ , & \text{otherwise} \ . \end{cases}$$

The rationale behind formula (41) is that we essentially aspire to increase the dual objective by δ_t. If we succeed in doing that at the current iteration, then we increase δ_t by $\rho_0 > 1$, otherwise we reduce it by $0 < \rho_1 < 1$ (but only up to a minimum threshold δ).

For comparing how adaptive and non-adaptive multiplier update rules perform, we have also applied our projected subgradient scheme to random synthetic MRF optimization problems. The plot in Fig. 7 shows a typical result of how the dual objective value varies in this case where an adaptive step size rule of the form (40), as well as a typical diminishing step size rule of the form $\{a_t = \frac{a}{\sqrt{t}}\}$ have been used. As expected, the non-adaptive step size rule converges more slowly, since it determines $\{a_t\}$ a priori, *i.e*, before the algorithm is run.

Besides using an adaptive update of the multipliers, another way to speed up subgradient methods is to add memory to the updates of the dual variables. One such standard method is to use an update direction s_t that is equal to a convex combination of the current subgradient ∇g_t and the last search direction s_{t-1}. In this case, the dual variables at the t-the iteration are updated

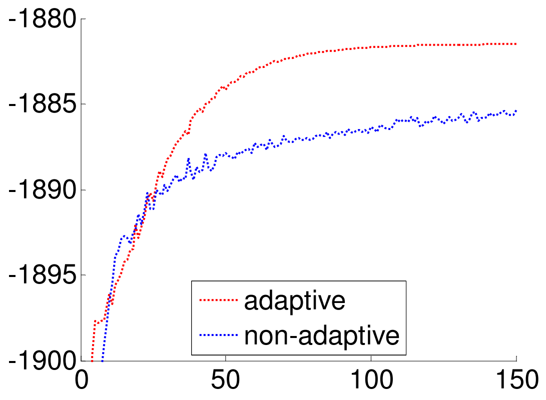


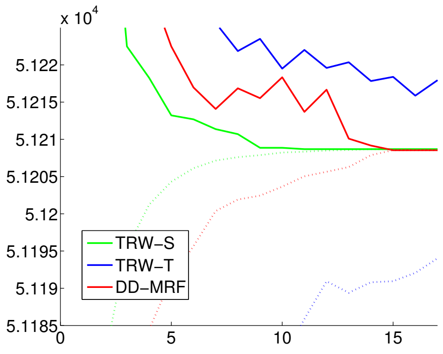
Fig. 7: We have applied DD-MRF to random synthetic multi-label MRFs defined on a 4-connected 50×50 grid using both an adaptive rule of the form (40) and a diminishing step size rule of the form $\{a_t = \frac{a}{\sqrt{t}}\}$. We show an example of how the dual objective varies in the two cases. The unary potentials of the synthetic MRFs were drawn from $\mathcal{N}(0,1)$ and the pairwise potentials from $\sigma \cdot |\mathcal{N}(0,1)|$ ($\sigma = 0.2$ for the example shown).

by adding the vector $a_t \cdot s_t$ (instead of $a_t \cdot \nabla g_t$), where $s_t = (1-w) \cdot \nabla g_t + w \cdot s_{t-1}$ and $0 \leq w < 1$ is a parameter that controls the desired amount of memory for the subgradient method (w can be also dynamically adapted as in [6]). Algorithms of this form are sometimes called the heavy ball method and appear in many variants. The rationale behind them is to prevent the appearance of a zig-zagging behaviour during the subgradient algorithm.

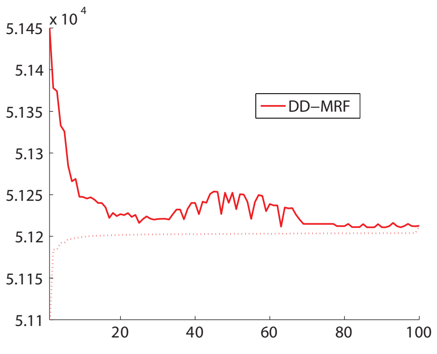
A related method for speeding up the subgradient algorithm that we often found to work well in practice was to modify the update direction by applying a coordinate ascent step on all the dual variables after each subgradient update. In this case, a subgradient vector such as (38) in conjunction with rule (39) can be used for the coordinate ascent steps. Before proceeding, we should note at this point that there is still a very large literature on how to dynamically update the multipliers or the search directions used by the subgradient method (*e.g*, see [2], [6] for more examples).

We have applied DD-MRF on various computer vision problems such as segmentation, stereo matching and optical flow estimation, as well as on synthetic problems, and we next present related experimental results. We also compare DD-MRF to existing TRW algorithms. These are TRW-T and TRW-E from [49], as well TRW-S from [21]. 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 less cluttered.

We have first tested our method on the task of interactive binary image segmentation [46]. 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

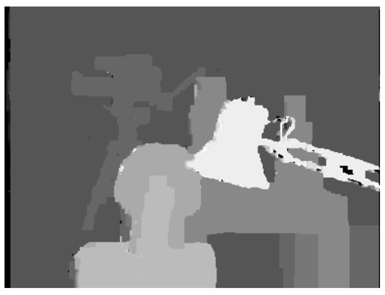


(a)

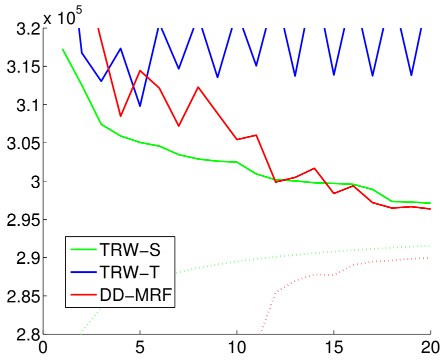


(b)

Fig. 8: (a) Plots for the binary segmentation problem (grid size = 256×170, number of labels = 2). 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 (10), *i.e*, lower bounds on that energy. **(b)** DD-MRF applied to the same problem as (a) using the Larsson *et al* decoding method. In this case, more iterations were required for reaching the global optimum.



(a) Estimated disparity

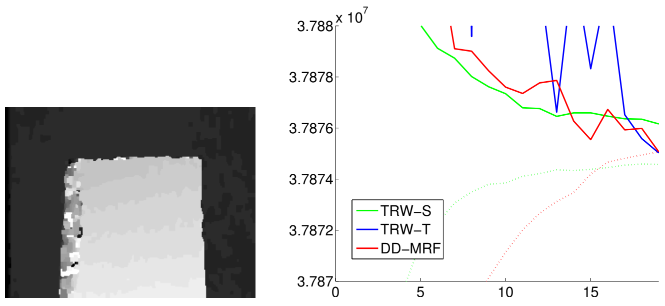


(b) Energy and lower bound plots

Fig. 9: *Tsukuba* results (grid size = 384×288, number of labels = 16).

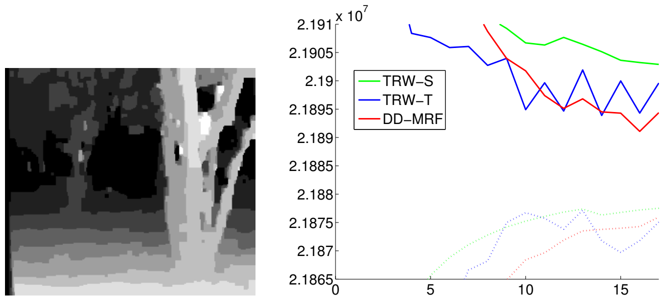
theorem 5, DD-MRF should be able to find the global optimum in this case and so the main goal of this experiment was to confirm this fact. 10 natural images were thus segmented and Fig. 8(a) shows a typical plot of how the MRF energy (*i.e* the cost of the primal problem) varies during a segmentation test. We also included in this plot the cost of the dual problem (10), 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 zero at the end (another way to verify this, is by using the max-flow algorithm to compute the optimal solution). In Fig. 8(b) we apply DD-MRF to the same problem using the Larsson *et al* decoding method. As can be seen, more iterations were required to compute the optimal solution in this case.

We have also tested our method on stereo matching [46]. In Fig. 9(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. 9(b) contains the corresponding plot that



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

Fig. 10: Results for the *Map* stereo pair (grid size = 284×216, number of labels = 30).



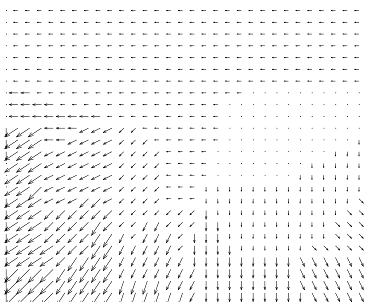
(a) Estimated disparity (b) Energy and lower bound plots

Fig. 11: Results for the *SRI tree* stereo pair (grid size = 256×233, number of labels = 10).

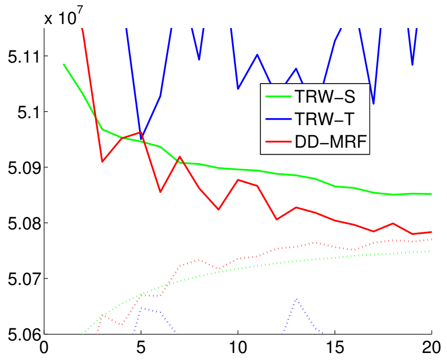
shows how the costs of the primal and the dual problem (*i.e* the MRF energy and the lower bound) vary during the execution of the algorithm. As in all other examples, here as well we have included the corresponding plots for the TRW-T and TRW-S algorithms. It is worth noting that, in this example, TRW-T did not manage to reduce the MRF energy (or increase the lower bound) as effectively as the DD-MRF algorithm.

Figures 10, 11 contain further results on stereo matching. Specifically, Fig. 10(a) displays the produced disparity for the *Map* stereo pair, while Fig. 10(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 11(a) and 11(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 [4]. 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.



(a) Estimated optical flow



(b) Energy and lower bound plots

Fig. 12: Optical flow for the *Yosemite* image sequence (grid size = 316×252, number of labels = 35).

12, 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 9-12, 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 has relative distance less than 0.0094, 0.0081, 0.00042, 0.00012 from the minimum energy for *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). Regarding the termination criterion that has been used, the algorithm stops when either the primal-dual gap has not improved significantly for a certain number of iterations (where by primal-dual gap we mean the difference between the best obtained upper bound and the best obtained lower bound), or a maximum number of iterations has been exceeded. We should note that theoretical stopping criteria do exist for the subgradient method, however they usually do not prove to be very useful in practice.

Also, to more thoroughly examine the differences in performance between our scheme and a method such as TRW-S, we conducted experiments in which both of these methods were applied to synthetic MRF problems. Figure 13 shows how the convergence of the dual objective proceeds in this case for an MRF with 4 labels (for this example, coordinate ascent steps were applied during the subgradient method as explained earlier). The multi-label MRFs were defined on a 50×50 grid, and the unary potentials for each node were drawn independently from $\mathcal{N}(0,1)$, while the matrix of pairwise potentials for each MRF edge was symmetric with zero diagonal elements and non-diagonal elements drawn from $d_{\max} \cdot |\mathcal{N}(0,1)|$ (*e.g*, $d_{\max}=3$ for the example shown in Fig. 13). As can be seen, TRW-S manages to increase the dual objective faster in the beginning. This should be expected given that TRW-S is a greedy block-coordinate ascent technique. On the other hand, compared to a subgradient method, it may eventually get stuck

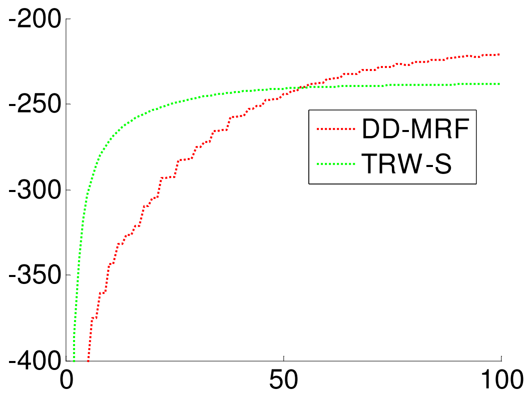
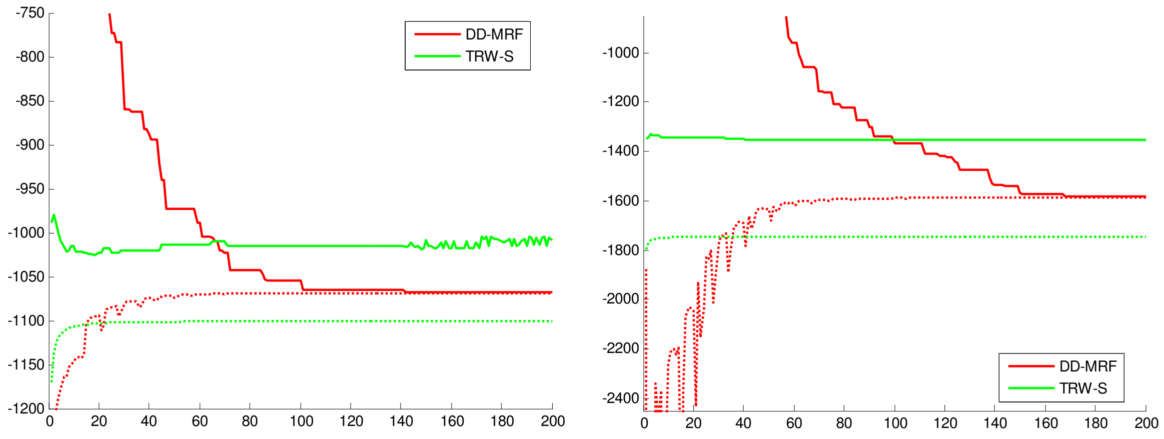


Fig. 13: An example of convergence of the dual objective when TRW-S and DD-MRF are applied to a random synthetic multi-label MRF with 4 labels (see text for more details).



(a) (b)

Fig. 14: This example illustrates, on the one hand, that DD-MRF is indeed able to optimize tighter LP-relaxations and, on the other hand, that the quality of the used relaxations can have a significant effect on the quality of the obtained solutions. We show sequences of primal costs (solid lines) and dual costs (dashed lines) generated by DD-MRF and TRW-S when applied to Ising models with mixed potentials defined on a 50$\times$50 grid. In this case, the success of DD-MRF is due to the fact that it uses loopy MRFs as slaves, which lead to a relaxation that turns out to be tight for this example. On the other hand, TRW-S relies on the use of tree-structured MRFs and thus optimizes a much less tight relaxation (*i.e*, one that does not approximate the MAP estimation problem very well), which explains why no good solutions (*i.e*, with low energy) are computed. The potentials were generated using **(a)** $\sigma = 1.5$ and **(b)** $\sigma = 3.5$ in equation (42).

to a lower dual objective. This is due to that the dual function to be maximized is non-smooth, and in this case it is well known that coordinate-ascent methods may get stuck to suboptimal solutions.

To test the ability of DD-MRF to optimize tighter relaxations, we next applied our algorithm to a set of hard binary optimization problems. The MRFs for these problems were Ising models with mixed pairwise potentials defined on a $N \times N$ grid. More specifically, the unary potential potentials $\boldsymbol{\theta}_p$ of these MRFs were generated by drawing independent samples from a zero-mean unit-variance Gaussian distribution, while their pairwise potentials $\boldsymbol{\theta}_{pq}$ were set as follows:

$$\theta_{pq}(0,0) = \theta_{pq}(1,1) = 0, \quad \theta_{pq}(0,1) = \theta_{pq}(1,0) = \lambda_{pq} \ ,$$

where each λ_{pq} was drawn independently from the following Gaussian distribution

$$\lambda_{pq} \sim \mathcal{N}(0, \sigma^2) \ . \tag{42}$$

Note that, due to the use of mixed potentials, the above binary problems are non-submodular and are therefore very difficult to optimize. Essentially, this difficulty stems from the fact that the standard LP relaxation associated with these MRFs is non-tight and thus does not approximate well the MRF optimization task (which is completely the opposite of what happens when dealing with submodular MRFs where the standard LP-relaxation is known to be exact). Obviously, one way to counter that problem is by attempting to strengthen the underlying relaxation. As explained in section VI, this can be achieved by using loopy MRFs as subproblems when applying the DD-MRF algorithm. Since the graph of the binary problems is a $N \times N$ grid, it suffices that a slave MRF is defined for each cell of that grid (*i.e*, there exists one slave loopy-MRF per quadruple of nodes (i, j), $(i, j+1)$, $(i+1, j+1)$, $(i+1, j)$). Fig. 14 shows two typical plots of sequences of primal and dual costs that are generated by the DD-MRF algorithm when applied to such a case for grids of size 50×50 (the two different plots correspond to binary MRFs generated with two different values of σ). As it is immediately seen, the sequences of primal and dual costs converge to each other, hence meaning that DD-MRF is able to compute a globally optimal solution even when dealing with non-submodular MRFs, and this is due to the fact that it relies on a tighter relaxation (see also [44],[54],[23] for some other dual coordinate-ascent methods that optimize tighter relaxations of the same type). On the other hand, if a standard LP-relaxation is used in this case, the resulting solutions are of much worse energy as shown in Fig. 14.

As a proof of concept, we also experimented with the idea of utilizing non tree-structured MRFs (like, *e.g*, loopy submodular MRFs) as subproblems during dual decomposition. To this end, given as input binary submodular MRFs[10] defined on a 50×50 grid, we applied the DD-MRF algorithm using two different dual decompositions: a typical tree-structured decomposition consisting of a pair of spanning trees that covered the input MRF graph, as well as a decomposition consisting of two submodular MRFs resulting from a vertical splitting of the input 50×50 grid into two grids of size 50×26 that overlap each other at the middle column of the original grid. Note that, as explained in section VI-C, both decompositions are going to lead to an equally strong relaxation (in fact, due to the submodularity of the input MRF, the resulting relaxation will be tight, and so both methods will yield a global optimum in this case). However, the rate

[10]To ensure submodularity, the pairwise potentials of the MRFs were chosen to be of the form $w_{pq}(1-I_2)$, where I_2 represents the 2×2 identity matrix and w_{pq} are weights drawn from $|\mathcal{N}(0, \sigma^2)|$ (unary potentials were sampled from $\mathcal{N}(0, 1)$).

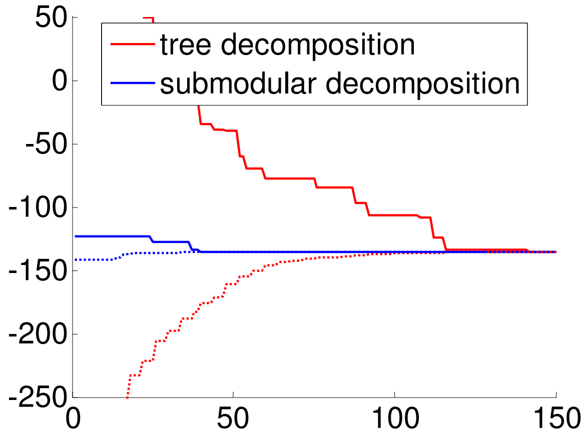
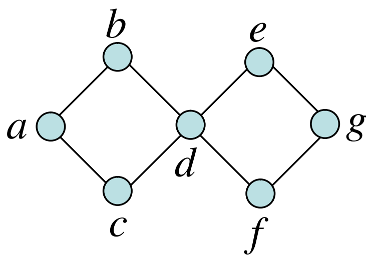
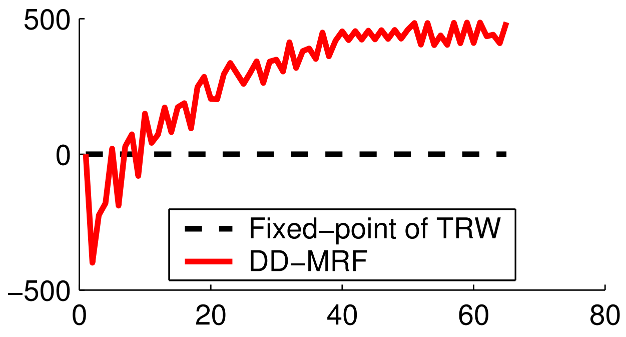


Fig. 15: Applying DD-MRF to a submodular binary MRF using a tree-structured decomposition and a submodular decomposition (see text for details).



(a)



(b)

Fig. 16: (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 $\mathcal{T}_1 = (a, b, d, e, g)$, $\mathcal{T}_2 = (a, c, d, f, g)$. **(b)** The solid and dashed lines show respectively the lower bounds produced by DD-MRF and the TRW algorithms (including TRW-S) for the graph in Fig. 16(a), when $\kappa = 1000$ (see text). Notice the large gap between these two bounds. In fact, the value of this gap can be made arbitrarily large by, *e.g*, increasing κ.

of convergence is expected to differ between the two cases. Indeed, as shown in the example of Fig. 15, the submodular decomposition provides a significant speed-up, as it requires much fewer iterations to converge. The actual speed-up is, in fact, even greater considering also that the optimization of the submodular MRF subproblems can be implemented extremely fast via the use of dynamic graph-cut based techniques such as [26], as explained in section VI-C.

We finally borrow an example from [21] to illustrate that DD-MRF can maximize the dual problem (10) (*i.e* the lower bound on the MRF energy), even in cases where the TRW algorithms may get stuck to a lower bound that can be arbitrarily far from the true maximum lower bound. The graph for this example is shown in Fig. 16(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 $\mathcal{T}_1 = (a, b, d, e, g)$, $\mathcal{T}_2 = (a, c, d, f, g)$ are used in this case, both of which are supposed to have zero unary potentials, *i.e* $\boldsymbol{\theta}_p^{\mathcal{T}_1} = \mathbf{0}\ \forall p \in \mathcal{T}_1, \boldsymbol{\theta}_p^{\mathcal{T}_2} = \mathbf{0}\ \forall p \in \mathcal{T}_2$. Also, the pairwise potentials for these trees are set as follows:

$$\boldsymbol{\theta}_{ab}^{\mathcal{T}_1} = \begin{bmatrix} \kappa & 0 \\ 0 & \kappa \end{bmatrix}, \boldsymbol{\theta}_{bd}^{\mathcal{T}_1} = \begin{bmatrix} 0 & \kappa & \kappa \\ \kappa & 0 & 0 \end{bmatrix}, \boldsymbol{\theta}_{de}^{\mathcal{T}_1} = \begin{bmatrix} \kappa & 0 \\ 0 & \kappa \\ \kappa & 0 \end{bmatrix}, \boldsymbol{\theta}_{eg}^{\mathcal{T}_1} = \begin{bmatrix} 0 & \kappa \\ \kappa & 0 \end{bmatrix},$$

$$\boldsymbol{\theta}_{ac}^{\mathcal{T}_2} = \begin{bmatrix} \kappa & 0 \\ 0 & \kappa \end{bmatrix}, \boldsymbol{\theta}_{cd}^{\mathcal{T}_2} = \begin{bmatrix} \kappa & 0 & 0 \\ 0 & \kappa & \kappa \end{bmatrix}, \boldsymbol{\theta}_{df}^{\mathcal{T}_2} = \begin{bmatrix} \kappa & 0 \\ \kappa & 0 \\ 0 & \kappa \end{bmatrix}, \boldsymbol{\theta}_{fg}^{\mathcal{T}_2} = \begin{bmatrix} \kappa & 0 \\ 0 & \kappa \end{bmatrix},$$

where κ denotes a positive constant. As it was shown in [21], the above dual variables $\boldsymbol{\theta}^{\mathcal{T}_1}$, $\boldsymbol{\theta}^{\mathcal{T}_2}$ form a fixed point for all TRW algorithms (as $\boldsymbol{\theta}^{\mathcal{T}_1}, \boldsymbol{\theta}^{\mathcal{T}_2}$ 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. 16(b), DD-MRF does not get stuck to such a bad lower bound when starting from $\boldsymbol{\theta}^{\mathcal{T}_1}$, $\boldsymbol{\theta}^{\mathcal{T}_2}$.

VIII. 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. At its core lies a simple, but powerful, subgradient-based dual decomposition scheme. Furthermore, one of its advantages is that it leads to MAP estimation algorithms that are easily adaptable, while they can also make full use of the structure that may exist in particular classes of MRFs. Due to its flexibility, we thus expect that our framework will find good use in a wide variety of applications in the future. Here we demonstrated this with 3 examples, *i.e*, by deriving algorithms that rely on 3 different kind of decompositions: tree-structured decompositions, decompositions with loopy graphs, and submodular decompositions. The corresponding algorithms respectively generalize prior-art message-passing schemes, optimize tighter relaxations, and allow for using fast inference techniques such as graph-cut based methods. We also provided a thorough theoretical analysis both for the above 3 cases and also for more general cases. On another note, an additional advantage of our framework is that it 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 more advanced techniques for non-smooth optimization would make a very interesting avenue of future research, and could potentially lead to even more powerful MRF optimization techniques. For instance, bundle methods or stabilized cutting plane methods would be particularly interesting candidates in this regard. This type of techniques essentially rely on approximating a non-smooth objective by a piecewise linear model. This approximate model is constructed based on a set (a "bundle") of subgradients of the objective function, which is continuously refined throughout the algorithm. These techniques can thus be considered as a natural extension to subgradient methods since they do not merely rely on the subgradient computed at the current iterate, but they also take into account past information (*i.e*, previous subgradients) for updating their model. To conclude, a novel and very general energy minimization framework has been presented, which we hope to further promote the use of MRFs in the future.

Acknowledgment. We would like to thank the anonymous reviewers for their insightful and constructive comments that helped us to improve the clarity and presentation of the paper.

References

[1] F. Barahona and R. Anbil. The volume algorithm: producing primal solutions with a subgradient method. *Mathematical Programming*, 87:385–399, 2000. 14

[2] D. Bertsekas. *Nonlinear Programming*. Athena Scientific, 1999. 4, 17, 30

[3] D. P. Bertsekas and S. K. Mitter. A descent numerical method for optimization problems with nondifferentiable cost functionals. *SIAM Journal on Control*, 11:637–652, 1973. 17

[4] Y. Boykov, O. Veksler, and R. Zabih. Fast approximate energy minimization via graph cuts. *PAMI*, Nov. 2001. 5, 32

[5] A. Braunstein, M. Mezard, and R. Zecchina. Survey propagation: an algorithm for satisfiability. *Random Structures and Algorithms*, 27:201, 2005. 5

[6] P. Camerini, L. Fratta, and F. Maffioli. On improving relaxation methods by modifying gradient techniques. *Math. Programming Study*, 3:26–34, 1975. 30

[7] C. Chekuri, S. Khanna, J. Naor, and L. Zosin. Approximation algorithms for the metric labeling problem via a new linear programming formulation. In *12^{th} Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 109–118, 2001. 2

[8] J. Duchi, D. Tarlow, G. Elidan, and D. Koller. Using combinatorial optimization within max-product belief. In *NIPS*, 2006. 20

[9] G. Elidan, I. McGraw, and D. Koller. Residual belief propagation: Informed scheduling for asynchronous message passing. In *Proceedings of the Twenty-second Conference on Uncertainty in AI (UAI)*, 2006. 5

[10] P. Felzenszwalb and D. Huttenlocher. Pictorial structures for object recognition. *IJCV*, 61, 2005. 2

[11] W. T. Freeman, E. C. Pasztor, and O. T. Carmichael. Learning low-level vision. *International Journal of Computer Vision*, 40(1):25–47, 2000. 2

[12] B. Frey. *Graphical models for machine learning and digital communication*. MIT Press, 1998. 2

[13] B. Frey and D. MacKay. A revolution: Belief propagation in graphs with cycles. In *NIPS*, 1998. 5

[14] A. Globerson and T. Jaakkola. Fixing max-product: Convergent message passing algorithms for map lp-relaxations. In *NIPS*, pages 553–560, 2008. 3, 6

[15] T. Heskes. Stable fixed points of loopy belief propagation are minima of the Bethe free energy. In *NIPS*, pages 359–366, 2003. 5

[16] T. Heskes. Convexity arguments for efficient minimization of the Bethe and Kikuchi free energies. *Journal of Artificial Intelligence Research*, 26:153–190, 2006. 5

[17] J. Johnson, D. Malioutov, and A. Willsky. Lagrangian relaxation for MAP estimation in graphical models. In *Allerton Conf. Communication, Control and Computing*, 2007. 6

[18] M. Jordan. *Learning in graphical models*. MIT Press, 1999. 2

[19] J. Kim, V. Kolmogorov, and R. Zabih. Visual correspondence using energy minimization and mutual information. In *ICCV*, 2003. 2

[20] P. Kohli and P. Torr. Dynamic graph cuts for efficient inference in markov random fields. *PAMI*, 2007. 24

[21] V. Kolmogorov. Convergent tree-reweighted message passing for energy minimization. *PAMI*, 2006. 2, 3, 5, 13, 14, 17, 30, 36, 37

[22] V. Kolmogorov and M. Wainwright. On the optimality of tree-reweighted max-product message passing. In *UAI*, 2005. 6, 17, 18

[23] N. Komodakis and N. Paragios. Beyond loose lp-relaxations: Optimizing mrfs by repairing cycles. In *ECCV*, 2008. 35

[24] N. Komodakis, N. Paragios, and G. Tziritas. MRF Optimization via Dual Decomposition: Message-passing revisited. In *ICCV*, 2007. 4, 5

[25] N. Komodakis and G. Tziritas. Image completion using global optimization. In *CVPR*, 2006. 2

[26] N. Komodakis, G. Tziritas, and N. Paragios. Fast, approximately optimal solutions for single and dynamic MRFs. In *CVPR*, 2007. 5, 24, 36

[27] M. Kumar, P. Torr, and A. Zisserman. Obj cut. In *CVPR*, 2005. 2

[28] M. P. Kumar, V. Kolmogorov, and P. H. S. Torr. An analysis of convex relaxations for MAP estimation. In *NIPS*, 2007. 6

[29] V. Kwatra, A. Schodl, I. Essa, G. Turk, and A. Bobick. Graphcut textures: Image and video synthesis using graph cuts. In *SIGGRAPH*, 2003. 2

[30] T. Larsson, M. Patriksson, and A. Stromberg. Ergodic primal convergence in dual subgradient schemes for convex programming. *Mathematical Programming*, 86:283–312, 1999. 15

[31] T. Meltzer, C. Yanover, and Y. Weiss. Globally optimal solutions for energy minimization in stereo vision using reweighted belief propagation. In *ICCV*, 2005. 2, 14

[32] A. Nedic and D. P. Bertsekas. Incremental subgradient methods for nondifferentiable optimization. *SIAM Journal on Optimization*, 12:109–138, 2001. 16

[33] A. Nedic and A. Ozdaglar. Approximate primal solutions and rate analysis for dual subgradient methods. *SIAM Journal on Optimization*, 19:1757–1780, 2009. 15

[34] J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, 1988. 2, 5

[35] A. Raj, G. Singh, and R. Zabih. MRF's for MRI's: Bayesian Reconstruction of MR Images via Graph Cuts. In *CVPR*, 2006. 5

[36] P. Ravikumar, A. Agarwal, and M. Wainwright. Message-passing for graph-structured linear programs: proximal projections, convergence and rounding schemes. In *ICML*, pages 800–807, 2008. 6

[37] S. Roth and M. Black. Fields of experts: A framework for learning image priors. In *CVPR*, 2005. 2

[38] C. Rother, V. Kolmogorov, V. S. Lempitsky, and M. Szummer. Optimizing binary mrfs via extended roof duality. In *CVPR*, 2007. 5

[39] M. Schlesinger and V. Giginyak. Solution to structural recognition (MAX,+)-problems by their equivalent transformations. *Control Systems and Computers*, 2007. 6

[40] M. I. Schlesinger. Syntactic analysis of two-dimensional visual signals in noisy conditions. *Kybernetica*, 1976. 2

[41] H. Sherali and G. Choi. Recovery of primal solutions when using subgradient optimization methods to solve lagrangian duals of linear programs. *Operations Research Letters*, 19:105–113, 1996. 14

[42] N. Shor. *Minimization methods for nondifferentiable functions*. Springer, Berlin, 1985. 14

[43] D. Sontag and T. Jaakkola. New outer bounds on the marginal polytope. In *NIPS*, 2008. 6, 15

[44] D. Sontag, T. Meltzer, A. Globerson, Y. Weiss, and T. Jaakkola. Tightening lp relaxations for map using message passing. In *UAI*, 2008. 35

[45] E. B. Sudderth, A. T. Ihler, W. T. Freeman, and A. S. Willsky. Nonparametric belief propagation. In *CVPR*, 2003. 5

[46] R. Szeliski, R. Zabih, D. Scharstein, O. Veksler, V. Kolmogorov, A. Agarwala, M. Tappen, and C. Rother. A comparative study of energy minimization methods for markov random fields with smoothness-based priors. *PAMI*, 2008. 2, 30, 31

[47] M. F. Tappen and W. T. Freeman. Comparison of graph cuts with belief propagation for stereo, using identical mrf parameters. In *ICCV*, pages 900–907, 2003. 2

[48] L. Torresani, V. Kolmogorov, and C. Rother. Feature correspondence via graph matching: Models and global optimization. In *ECCV*, 2008. 20

[49] M. Wainwright, T. Jaakkola, and A. Willsky. Map estimation via agreement on trees: message-passing and linear programming. *IEEE Trans. on Information Theory*, 2005. 2, 3, 5, 17, 30

[50] Y. Weiss and W. T. Freeman. Correctness of belief propagation in gaussian graphical models of arbitrary topology. *Neural Computation*, 13(10):2173–2200, 2001. 5

[51] Y. Weiss and W. T. Freeman. On the optimality of solutions of the max-product belief-propagation algorithm in arbitrary graphs. *IEEE Transactions on Information Theory*, 47(2):736–744, 2001. 5

[52] Y. Weiss, C. Yanover, and T. Meltzer. MAP estimation, linear programming and belief propagation with convex free

energies. In *Proceedings of the Twenty-second Conference on Uncertainty in AI (UAI)*, 2007. 6

[53] T. Werner. A linear programming approach to max-sum problem: A review. *PAMI*, 2007. 2, 3, 6

[54] T. Werner. High-arity interactions, polyhedral relaxations, and cutting plane algorithm for soft constraint optimisation (map-mrf). In *CVPR*, 2008. 35

[55] W. Wiegerinck and T. Heskes. Fractional belief propagation. In *NIPS*, pages 438–445, 2003. 5

[56] C. Yanover, T. Meltzer, and Y. Weiss. Linear Programming Relaxations and Belief Propagation – An Empirical Study. *JMLR*, 7, 2006. 6

[57] J. Yedidia, W. Freeman, and Y. Weiss. Constructing free energy approximations and generalized belief propagation algorithms. *IEEE Transactions on Information Theory*, 2005. 5