Towards More Efficient and Effective LP-Based Algorithms for MRF Optimization

Nikos Komodakis

University of Crete
Computer Science Department
komod@csd.uoc.gr

Abstract. This paper proposes a framework that provides significant speed-ups and also improves the effectiveness of general message passing algorithms based on dual LP relaxations. It is applicable to both pairwise and higher order MRFs, as well as to any type of dual relaxation. It relies on combining two ideas. The first one is inspired by algebraic multigrid approaches for linear systems, while the second one employs a novel decimation strategy that carefully fixes the labels for a growing subset of nodes during the course of a dual LP-based algorithm. Experimental results on a wide variety of vision problems demonstrate the great effectiveness of this framework.

1 Introduction

Message passing methods are extremely popular MRF optimization techniques in computer vision, with BP being the earliest method of this kind. Recently, many state of the art message-passing techniques have been proposed that rely on solving dual LP relaxations [1,2,3,4]. Compared to BP, they offer significant advantages such as better convergence properties, as well as the ability to provide suboptimality guarantees based on dual lower bounds. Moreover, they have been shown to significantly outperform BP and all other MAP estimation techniques [5]. On the other hand, one main drawback is that they often have a higher computational cost. As a result, given the large scale nature of the majority of vision problems, one of the key challenges in energy minimization is currently the acceleration of these methods. This is even more so considering the fact that computer vision researchers start gradually to resort to higher order MRF models, where such dual-based methods are expected to have much wider applicability due to their generality.

Motivated by the above observations, the goal of this work is to increase the overall efficiency of dual LP-based algorithms both for pairwise and higher order MRFs, while at the same time improving their effectiveness (*i.e.*, their accuracy). To this end, it proposes a framework that combines together two very general techniques in order to significantly speed up such algorithms. The first one is inspired by *algebraic multigrid* techniques for linear systems of equations, and uses a multiresolution hierarchy of dual relaxations for accelerating the convergence of dual-LP based methods. It relies on the premise that information

K. Daniilidis, P. Maragos, N. Paragios (Eds.): ECCV 2010, Part II, LNCS 6312, pp. 520–534, 2010.

is expected to propagate faster at lower resolutions. In the past, a geometric multigrid approach has been used for accelerating the BP algorithm for grid-structured graphs [6]. Here we extend and generalize such an approach to LP-based algorithms. Our algebraic multigrid framework can handle MRFs defined on any kind of graph, or having any kind of potentials. Moreover, it can be applied to higher order MRFs, as well as to LP relaxations that are tighter than the standard marginal polytope relaxation.

But to be able to achieve a significant speed up, besides accelerating the convergence, we also need to significantly reduce the time per iteration of a dual LP-based algorithm. To this end, we introduce a second technique, which consists of a decimation strategy that carefully fixes the labels for a growing subset of nodes during the course of the algorithm and thus one does not need to update their dual variables thereafter. It is based on the observation that, when using an algrebraic multigrid approach, a set of nodes typically exists that contribute a very small increase to the objective of the dual relaxation when their dual variables are updated. Similarly to the first technique, it is very general, and is applicable to both pairwise and higher order MRFs. Furthermore, it allows better primal solutions to be computed. Note that MRF decimation techniques have also been used in the past, and have been applied either to variants of BP [7,8] or to dual LP-based algorithms [9,10,11]. However, the latter techniques are not as widely applicable as our method.

After introducing in the next section the general setting used in the paper, we describe our framework in §3 - §7, while we discuss some extensions in §8. We present experimental results in §9 and finally conclude in §10.

2 Dual LP Relaxations for MRF Optimization

The problem of MAP estimation for discrete MRFs is typically formulated as follows. Given a graph $G = (\mathcal{V}, \mathcal{E})$ (where $\mathcal{V}$, $\mathcal{E}$ represent the nodes and edges of the graph) and a discrete set of labels $\mathcal{L}$, we want to assign a label x_p to each node p so that the total MRF energy (*i.e.*, the sum of all MRF potentials) is minimized, or

$$\mathrm{MRF}_G(\mathbf{U}, \mathbf{P}) := \min_{\mathbf{x}} \sum_{p \in \mathcal{V}} U_p(x_p) + \sum_{pq \in \mathcal{E}} P_{pq}(x_p, x_q) \ . \tag{1}$$

In the above, $\mathbf{U} = \{\mathbf{U}_p\}_{p \in \mathcal{V}}$ and $\mathbf{P} = \{\mathbf{P}_{pq}\}_{pq \in \mathcal{E}}$ denote respectively the set of all unary and pairwise potential functions.

As mentioned in the introduction, here we will concentrate on optimization methods that rely on dual LP relaxations. The most general setting for describing all these methods is based on the dual decomposition framework [3]. According to this framework, the original problem $\mathrm{MRF}_G(\mathbf{U}, \mathbf{P})$ (also called the master MRF) is decomposed into a set of simpler MRFs that are called the slaves and are denoted by $\mathrm{MRF}_{G_i}(\boldsymbol{\theta}^{G_i}, \mathbf{P})$. Here we assume that each slave MRF is defined on a subgraph $G_i = (\mathcal{V}_i, \mathcal{E}_i)$, has its own unary potentials (denoted by $\boldsymbol{\theta}^{G_i}$),

while it inherits[1] the pairwise potentials $\mathbf{P}$ of the master MRF. In this case, the dual variables are the unary potentials $\{\boldsymbol{\theta}^{G_i}\}$ of the slave MRFs, and the *key property* that these variables have to satisfy is $\sum_i \boldsymbol{\theta}^{G_i} = \mathbf{U}$, *i.e.*, the sum of the unary potentials of the slaves should give back the unary potentials of the master MRF.

Based on this property, it is easy to prove that the sum of the optimal energies of the slaves always provides a lower bound to the optimal energy of the master, and so the goal of the dual LP relaxation is exactly to adjust the dual variables so as to maximize this lower bound, or

$$\max_{\{\boldsymbol{\theta}^{G_i}\}} \sum_i \mathrm{MRF}_{G_i}(\boldsymbol{\theta}^{G_i}, \mathbf{P}) \tag{2}$$

$$\text{s.t. } \sum_i \boldsymbol{\theta}^{G_i} = \mathbf{U} \ . \tag{3}$$

Different dual-based optimization algorithms have been proposed in the literature, all of which try to solve the above dual relaxation, and the key property that has to be maintained (either implicitly or explicitly) is condition (3).

3 Accelerating Dual LP-Based Optimization Algorithms via an Algrebraic Multigrid Approach

Due to the decomposition of the master MRF into a set of smaller slave MRFs, the update of the dual variables is essentially done based only on local information. As a result, information travels slowly across the graph, and this has the undesirable effect of slowing down the convergence of dual LP-based algorithms, which thus require many iterations to converge to the correct solution. This issue is essentially very similar to the slow convergence problem faced by iterative algorithms for linear systems. Again, due to the local nature of the updates, such algorithms can recover very fast (*i.e.*, in few iterations) the high-frequency part of the solution, but they are very slow at recovering the lower frequencies. Multigrid is introduced to overcome this problem, where the basic idea is based on the trivial observation that low frequencies in the original grid reappear as high frequencies in a grid of lower resolution. A multigrid approach thus replaces the original linear system with a hierarchical multiresolution set of linear systems. The two key elements in a multigrid algorithm are the so called restriction and prolongation operators, that specify the transition between linear systems at adjacent levels in the hierarchy. These operators are combined to generate a so called V-cycle, which consists of a fine-to-coarse restriction phase followed by a coarse-to-fine prolongation phase.

Our aim here will be to apply a similar strategy to dual based MRF algorithms for quickly solving (2). This will be done by using a hierarchy of dual decompositions, defined on a sequence of graphs $G = G^{(0)}, G^{(1)}, \ldots, G^{(T)}$, where each

[1] In general, each slave can have its own pairwise potentials (just like the unary potentials) and does not need to inherit them from the master MRF. Here we assume they are inherited only to simplify the presentation and to reduce notational clutter, but everything described can be very easily extended to the more general case.

graph $G^{(t+1)}$ is assumed to be a "coarser" version of graph $G^{(t)}$ (we will explain what precisely we mean by "coarser" later). We will also define a restriction and prolongation operator, denoted hereafter by PROJ and LIFT respectively. The role of the restriction operator will be to take as input a master MRF and its dual decomposition at level t, and to project them onto level $t+1$, *i.e.*, create a corresponding master problem and a corresponding dual decomposition at level $t+1$

$$\begin{array}{c} \mathrm{MRF}_{G^{(t)}}(\mathbf{U}^{(t)}, \mathbf{P}^{(t)}) \\ \left\{\mathrm{MRF}_{G_i^{(t)}}(\boldsymbol{\theta}^{G_i^{(t)}}, \mathbf{P}^{(t)})\right\} \end{array} \xrightarrow{\text{PROJ}} \begin{array}{c} \mathrm{MRF}_{G^{(t+1)}}(\mathbf{U}^{(t+1)}, \mathbf{P}^{(t+1)}) \\ \left\{\mathrm{MRF}_{G_i^{(t+1)}}(\boldsymbol{\theta}^{G_i^{(t+1)}}, \mathbf{P}^{(t+1)})\right\} \end{array} \tag{4}$$

On the contrary, the role of the prolongation operator LIFT will be to take as input a feasible set of dual variables $\left\{\boldsymbol{\theta}^{G_i^{(t+1)}}\right\}$ for the decomposition defined at the "coarser" level $t+1$, and to lift them to a feasible set of dual variables $\left\{\boldsymbol{\theta}^{G_i^{(t)}}\right\}$ for the decomposition that has been previously defined at level t, *i.e.*,

$$\left\{\boldsymbol{\theta}^{G_i^{(t+1)}}\right\} \xrightarrow{\text{LIFT}} \left\{\boldsymbol{\theta}^{G_i^{(t)}}\right\} \ . \tag{5}$$

Just like in multigrid, a V-cycle in our case will consist of a restriction phase followed by a prolongation phase (see Fig. 1(a)). In the restriction phase we sequentially apply operator PROJ to all but the last level in the hierarchy, *i.e.*, we start from level $t = 0$ and go up to level $t = T - 1$. In this manner, a master MRF along with a dual decomposition is generated for each level. All of these decompositions are essentially projections of the original master problem and its dual decomposition. In the prolongation phase, we move in the opposite direction. This means that for each level t (where t now starts from $t = T$ and terminates at $t = 0$) we solve the dual relaxation corresponding to the decomposition at that level, and then we lift the resulting solution onto the next finer level (if one exists) via using the operator LIFT, thus initializing the dual variables for the decomposition at level $t - 1$. Due to the the information traveling much faster at the "coarser" levels of the hierarchy, the dual relaxations for these levels can be solved very fast, *i.e.*, in very few iterations. Furthermore, this quick spreading of the information that took place in the coarser levels is carried over to the finer levels, thanks to the initialization of the dual variables via the LIFT operator (assuming, of course, that this operator has been properly defined, which is crucial for the success of this scheme). This, in turn, results into accelerating the convergence of the dual relaxations at the finer levels as well.

Having explained the overall structure of our method, it still remains to describe how to generate the hierarchy of graphs, how the master problems and their dual decompositions are defined at each level, and, most importantly, how to efficiently compute the operators LIFT and PROJ, which is, of course, one of the key technical issues. Before doing so, we must note that we want our scheme to be applicable to any kind of graph G, and not only to grids, as well as to MRFs with any kind of potential functions. Drawing an analogy with multigrid methods, we want to derive an algebraic (and not a geometric) multigrid solver, as the former is much more widely applicable.

4 Defining the Hierarchy of Graphs

Given a graph $G = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \{p_1, p_2, \ldots, p_n\}$, we want to define a "coarser" graph $\bar{G} = (\bar{\mathcal{V}}, \bar{\mathcal{E}})$. All that is needed as input for this purpose, is a partition $\{\bar{p}_1, \bar{p}_2, \ldots, \bar{p}_{\bar{n}}\}$ of $\mathcal{V}$, *i.e.*, $\cup \bar{p}_i = \mathcal{V}$ and $\bar{p}_i \cap \bar{p}_j = \emptyset$. Each node of the "coarser" graph $\bar{G}$ will then correspond to a subset of this partition, *i.e.*, it will hold $\bar{\mathcal{V}} = \{\bar{p}_1, \bar{p}_2, \ldots, \bar{p}_{\bar{n}}\}$, where we hereafter use $\bar{p}_i$ to denote both a node of $\bar{\mathcal{V}}$ as well as a subset of nodes of $\mathcal{V}$. Under this convention, the projection (denoted by $\text{proj}(p)$) of a node $p \in \mathcal{V}$ is defined as the unique node $\bar{p} \in \bar{\mathcal{V}}$ that satisfies the condition $p \in \bar{p}$, while the projection of a subset of nodes $\{p_k\} \subseteq \mathcal{V}$ is naturally equal to the union of the individual projections, *i.e.*, $\text{proj}(\{p_k\}) = \cup \text{proj}(p_k)$. Based on this notation, the set of edges $\bar{\mathcal{E}}$ of $\bar{G}$ is then defined as $\bar{\mathcal{E}} = \{\text{proj}(p_i p_j) | p_i p_j \in \mathcal{E}, \text{proj}(p_i) \neq \text{proj}(p_j)\}$. The resulting "coarser" graph $\bar{G} = (\bar{\mathcal{V}}, \bar{\mathcal{E}})$ is called the projection of graph G, and is denoted by $\text{proj}(G)$ (*e.g.*, see Fig. 1(b)). Therefore, to define a hierarchy of graphs, it suffices to set $G^{(t+1)} = \text{proj}(G^{(t)})$, where we assume that a partition has been specified by the user for each of the projections and $G^{(0)} = G$.

Assigning a label to a node $\bar{p} \in \bar{\mathcal{V}}$ of the "coarser" graph $\bar{G} = \text{proj}(G)$ will mean that this label is assigned to all nodes of G in the set $\{p \in \mathcal{V} \,|\, \text{proj}(p) = \bar{p}\}$. Based on this convention, if $\text{MRF}_G(\mathbf{U}, \mathbf{P})$ is an MRF[2] on the graph G, its projection on $\bar{G}$ will be an MRF, denoted by $\text{proj}(\text{MRF}_G(\mathbf{U}, \mathbf{P})) := \text{MRF}_{\bar{G}}(\bar{\mathbf{U}}, \bar{\mathbf{P}})$, whose potentials $\bar{\mathbf{U}}$, $\bar{\mathbf{P}}$ are defined as follows[3]:

$$\bar{U}_{\bar{p}}(l) = \sum\nolimits_{p:\text{proj}(p)=\bar{p}} U_p(l), \quad \bar{P}_{\bar{p}\bar{q}}(l, l') = \sum\nolimits_{pq:\text{proj}(pq)=\bar{p}\bar{q}} P_{pq}(l, l') \ . \tag{6}$$

Naturally, we want the master MRF at each level of our hierarchy to be a projection of the original MRF.

5 Defining the Restriction Operator PROJ

It suffices to show how to define this operator for one level of the hierarchy, *i.e.*, during a transition from a graph G to a coarser graph $\bar{G} = \text{proj}(G)$. Let $\text{MRF}_G(\mathbf{U}, \mathbf{P})$ be the master MRF on G, and let $\left\{\text{MRF}_{G_i}(\boldsymbol{\theta}^{G_i}, \mathbf{P})\right\}$ be its dual decomposition (*i.e.*, the set of slaves defined on subgraphs $\{G_i\}$). The main role of operator PROJ will be to define the corresponding dual decomposition for the graph $\bar{G}$, denoted by $\left\{\text{MRF}_{\bar{G}_j}(\bar{\boldsymbol{\theta}}^{\bar{G}_j}, \bar{\mathbf{P}})\right\}$. To this end, it first needs to determine the set of subgraphs $\{\bar{G}_j\}$ on which the new slaves will be defined. This set will consist of all subgraphs of the form $\text{proj}(G_i)$, *i.e.*,

$$\{\bar{G}_1, \bar{G}_2, \ldots, \bar{G}_{\mathcal{J}}\} = \{\text{proj}(G_1), \text{proj}(G_2), \ldots, \text{proj}(G_{\mathcal{I}})\} \ . \tag{7}$$

[2] Depending on the context, $\text{MRF}_G(\mathbf{U}, \mathbf{P})$ denotes either a MRF (on a graph G) with unary and pairwise potentials $\mathbf{U}$, $\mathbf{P}$ or a minimum MRF energy (as in (1)).

[3] To reduce notational clutter, we assume it holds $P(l, l) = 0$ when defining the potentials $\bar{\mathbf{U}}$, otherwise we must set $\bar{U}_{\bar{p}}(l) = \sum_{p:\text{proj}(p)=\bar{p}} U_p(l) + \sum_{pq:\text{proj}(pq)=\bar{p}} P_{pq}(l, l)$.

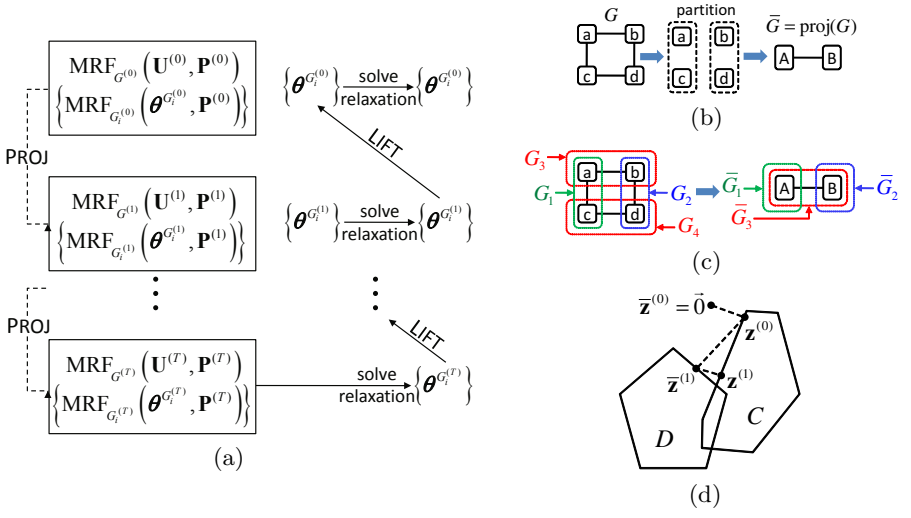


Fig. 1. (a) V-cycle of the 'algrebraic multigrid' approach for dual LP-based algorithms **(b)** $\bar{G}$ is the projection of graph G based on the partition $\{a, c\}, \{b, d\}$. **(c)** If G_1, G_2, G_3, G_4 are the subgraphs of the slaves in G, then $\bar{G}_1$, $\bar{G}_2$, $\bar{G}_3$ will be the subgraphs of the 3 slaves in $\bar{G}$. Note that $\bar{G}$ has fewer slaves since both G_3, G_4 project onto $\bar{G}_3$. Also note that the slaves for $\bar{G}_1$, $\bar{G}_2$ have no pairwise potentials. **(d)** The projection of $\mathbf{0}$ onto $C \cap D$ is computed via alternating projections on C and D (note that although C and D are drawn here as polytopes, they are actually affine subspaces in our case).

Since it can hold $\text{proj}(G_i) = \text{proj}(G_{i'})$ for $i \neq i'$, it is important to emphasize that the number of subgraphs $\bar{G}_j$ may be *strictly less* than the number of subgraphs G_i (see Fig. 1(c)). The operator PROJ then associates to each different subgraph $\bar{G}_j$ a slave $\text{MRF}_{\bar{G}_j}(\bar{\boldsymbol{\theta}}^{\bar{G}_j}, \bar{\mathbf{P}})$ whose potential functions are defined as follows:

$$\bar{\boldsymbol{\theta}}_{\bar{p}}^{\bar{G}_j}(l) = \sum\nolimits_{i:\text{proj}(G_i)=\bar{G}_j} \sum\nolimits_{p:\text{proj}(p)=\bar{p}} \boldsymbol{\theta}_p^{G_i}(l) \ , \tag{8}$$

$$\bar{P}_{\bar{p}\bar{q}}(l, l') = \sum\nolimits_{pq:\text{proj}(pq)=\bar{p}\bar{q}} P_{pq}(l, l') \ , \tag{9}$$

i.e., essentially it holds $\text{MRF}_{\bar{G}_j}(\cdot, \cdot) = \sum_{i:\text{proj}(G_i)=\bar{G}_j} \text{MRF}_{G_i}(\cdot, \cdot)$. Eqs. (7)-(9) completely specify the dual decomposition for graph $\bar{G}$. Furthermore, this decomposition, in turn, completely specifies the potentials of the master MRF for $\bar{G}$, denoted by $\text{MRF}_{\bar{G}}(\bar{\mathbf{U}}, \bar{\mathbf{P}})$, since it must hold $\bar{\mathbf{U}} = \sum_j \bar{\boldsymbol{\theta}}^{\bar{G}_j}$ due to (3). However, there still remains one critical question that must be answered: is the resulting master MRF a projection onto $\bar{G}$ of the master MRF for G, as we want? It turns out that this is indeed the case, as the following theorem certifies:

Theorem 1 ([12]). *If* $\text{MRF}_{\bar{G}}(\bar{\mathbf{U}}, \bar{\mathbf{P}})$ *is the master MRF resulting from the dual decomposition defined by eqs.* (7)-(9)*, it then holds* $\text{MRF}_{\bar{G}}(\bar{\mathbf{U}}, \bar{\mathbf{P}}) = \text{proj}(\text{MRF}_G(\mathbf{U}, \mathbf{P}))$.

6 Defining the Prolongation Operator LIFT

Let $\text{MRF}_G(\mathbf{U}, \mathbf{P})$, $\{\text{MRF}_{G_i}(\boldsymbol{\theta}^{G_i}, \mathbf{P})\}$ and $\text{MRF}_{\bar{G}}(\bar{\mathbf{U}}, \bar{\mathbf{P}})$, $\{\text{MRF}_{\bar{G}_j}(\bar{\boldsymbol{\theta}}^{\bar{G}_j}, \bar{\mathbf{P}})\}$ be the master MRFs along with their set of slaves for two graphs G, $\bar{G} = \text{proj}(G)$ that are adjacent in the hierarchy. We assume that all these MRFs have been constructed during the restriction phase. We are now at the prolongation phase, where we assume that the dual relaxation for $\bar{G}$ has already been solved (*i.e.*, the dual variables $\{\bar{\boldsymbol{\theta}}^{\bar{G}_j}\}$ are set to their optimal values), and we now want to compute the LIFT operator whose role is to initialize the dual variables $\{\boldsymbol{\theta}^{G_i}\}$ for graph G. Note that, since $\{\bar{\boldsymbol{\theta}}^{\bar{G}_j}\}$ are already set to their optimal values, this implies that an important amount of information has already been spread across the whole graph $\bar{G}$ (and hence across G as well, since $\bar{G} = \text{proj}(G)$). Therefore, if we manage to properly take into account this information when initializing $\{\boldsymbol{\theta}^{G_i}\}$, we will succeed in accelerating the convergence of the dual relaxation for graph G as well.

But how can we go about doing that? A first idea that comes in mind is the following one: Let $\text{Opt}_{\bar{G}}$ be the already computed optimal value of the dual relaxation for $\bar{G}$. Recall that our goal is to maximize the dual objective function for graph G as well. Therefore, perhaps we should aim at initializing the dual variables $\{\boldsymbol{\theta}^{G_i}\}$ such that the resulting dual objective is at least as large as $\text{Opt}_{\bar{G}}$. Unfortunately, this is not, in general, possible, as the following theorem shows:

Theorem 2 ([12]). *Let* $\text{Opt}_{\bar{G}}$, Opt_G *denote the optimal values of the dual relaxations for graphs* $\bar{G}$ *and* G *respectively. Then, in general, it holds* $\text{Opt}_{\bar{G}} > \text{Opt}_G$.

However, dual variables $\{\bar{\boldsymbol{\theta}}^{\bar{G}_j}\}$ still provide very important information about dual variables $\{\boldsymbol{\theta}^{G_i}\}$ that we can take advantage of. In particular, they provide the linear constraints (8), where values $\bar{\boldsymbol{\theta}}_{\bar{p}}^{\bar{G}_j}(\cdot)$ are now assumed to be known. By imposing these constraints when initializing variables $\{\boldsymbol{\theta}^{G_i}\}$, we implicitly take into account all information that is encoded in $\{\bar{\boldsymbol{\theta}}^{\bar{G}_j}\}$ and has propagated across graph $\bar{G}$. Of course, besides eqs. (8), $\{\boldsymbol{\theta}^{G_i}\}$ must also satisfy the dual feasibility constraints (3). Therefore, in total, variables $\{\boldsymbol{\theta}^{G_i}\}$ should be initialized so as to satisfy the linear system composed of Eqs. (3) and (8). Among the many solutions of this underdetermined linear system, we must compute the one that has minimum Euclidean norm. Intuitively, this regularization of the solution is important because otherwise the resulting initial dual variables $\{\boldsymbol{\theta}^{G_i}\}$ for the finer graph may exhibit large variations in magnitude, which can have as a result that too much energy/information is concentrated on local parts of the fine graph. This can destroy the propagation of "information" that took place at the coarser level and can thus hinder convergence. We next show how to efficiently perform this minimum norm computation.

6.1 Solving for $\{\boldsymbol{\theta}^{G_i}\}$

During this section, in order to make the exposition more clear, we will use $\mathbf{z} = \{z_k\}_{k=1}^K$ to denote the vector from concatenating all $\{\boldsymbol{\theta}^{G_i}\}$. Our goal is to find the least norm solution of an underdetermined linear system, *i.e.*,

$$\min_{\mathbf{z}} \|\mathbf{z}\|^2 \tag{10}$$

$$\text{s.t. } \mathbf{A}\mathbf{z} = \mathbf{b} \ , \tag{11}$$

where $\mathbf{A}\mathbf{z} = \mathbf{b}$ encodes the linear constraints (3) and (8). Theoretically, such a $\mathbf{z}$ can be computed as $\mathbf{z} = \mathbf{A}^T(\mathbf{A}\mathbf{A}^T)^{-1}\mathbf{b}$, but this may be too slow for our purposes. Fortunately, a solution to (10) can be computed extremely fast by exploiting the special structure existing in the constraints (3), (8). To this end, we first rewrite the above optimization problem as follows:

$$\min_{\mathbf{z}} \|\mathbf{z} - \mathbf{0}\|^2 \tag{12}$$

$$\text{s.t. } \mathbf{z} \in C \cap D \ , \tag{13}$$

where C, D denote the linear subspaces of $\mathbb{R}^K$ corresponding to the linear equations (3) and (8) respectively. Therefore, the optimal $\mathbf{z}$ coincides with the orthogonal projection of the zero vector onto the intersection of the two linear subspaces C and D. To compute this projection, we apply the well known Dykstra algorithm [13], which is an alternating projection method, *i.e.*, it starts from the zero vector $\bar{\mathbf{z}}^{(0)} = \mathbf{0}$, and then alternately projects onto C and D:

$$\mathbf{z}^{(n)} = \mathcal{P}_C(\bar{\mathbf{z}}^{(n)}), \quad \bar{\mathbf{z}}^{(n+1)} = \mathcal{P}_D(\mathbf{z}^{(n)}), \quad n = 0, 1, 2, \ldots \tag{14}$$

where $\mathcal{P}_C(\cdot)$ and $\mathcal{P}_D(\cdot)$ denote projection onto C and D, respectively (see Fig 1(d)). This generates a sequence $\mathbf{z}^{(n)} \in C$ which provably converges to the optimal solution. The advantage in doing so is that the projections $\mathcal{P}_C(\cdot), \mathcal{P}_D(\cdot)$ can be computed extremely fast in our case due to the special structure of the linear subspaces C and D. Namely, it is easy to verify that both subspaces are specified by a set of equations of the following form:

$$\sum_{k \in I_j} z_k = b_j, \quad j = 1, 2, \ldots, J \ , \tag{15}$$

where the sets $\{I_j\}_{j=1}^J$ form a partition of the set of indices $I = \{1, 2, \ldots, K\}$, *i.e.*, $\cup_j I_j = I$ and $I_j \cap I_{j'} = \emptyset$ for $j \neq j'$. The projection of a point $\mathbf{z}'$ onto such a linear subspace is easily seen to be given by the following vector $\mathbf{z}$:

$$\forall k \in I_j, \quad z_k = z'_k + (b_j - \sum_{i \in I_j} z'_i)/|I_j|, \quad j = 1, 2, \ldots, J \ . \tag{16}$$

Furthermore, the Dykstra algorithm converges very fast in our case (*i.e.*, extremely few alternating projections are required). Theoretically this can be attributed to the fact that the rate of convergence of this algorithm increases with the angle $\theta \in [0, \frac{\pi}{2}]$ between the two subspaces, *i.e.*, the more orthogonal the subspaces are, the faster the convergence. Hence, overall, this algorithm leads to a very fast method for minimizing (10), *i.e.*, for initializing $\{\boldsymbol{\theta}^{G_i}\}$.

7 Accelerating Dual LP-Based Methods via Fixing Variables

The multigrid approach described above allows information to propagate faster across the MRF graph, and this helps us to reduce the number of iterations to convergence at the finest level. But to be able to take full advantage of this fact and achieve a significant speed up, we also need to reduce the time spent per iteration at that level. To this end, we now describe a technique that is applied only at the finest level of the hierarchy during the multigrid approach. As mentioned above, its main role is to bring a significant reduction in the time per iteration at that level (but, in addition to that, it also helps us to speed up the convergence of the algorithm). This reduction is achieved via a decimation strategy, where we carefully fix the labels for a dynamically growing subset of nodes during the algorithm, and do not update their dual variables thereafter. Recall that the cost of an iteration essentially comes from locally updating the dual variables $\{\boldsymbol{\theta}_p^{G_i}(\cdot)\}$ for each node p in the graph. These updates aim to improve the dual objective. However, it is often the case that the rate of improvement per iteration is very small despite the great computational effort, *i.e.*, the dual function increases only slightly per iteration, and this in turn leads to a slow progress towards a good primal solution. The reason for this behaviour comes from the fact that many nodes cannot contribute a positive increase when their local dual variables are updated during an iteration. The following definition is important in this regard: we say that a node p is *stabilized* at the t-th iteration if, exactly before the update of the local variables $\{\boldsymbol{\theta}_p^{G_i}(\cdot)\}$ at that iteration, there exists a label that optimizes all the current instances of slaves containing p (any such label will be called *stable* w.r.t. p). It is easy to verify the following proposition:

Proposition 1 ([12]). *If a node p is stabilized then no update of its local dual variables $\{\boldsymbol{\theta}_p^{G_i}(\cdot)\}$ can increase the dual objective. Conversely, if p is non-stabilized, then there always exists an update of variables $\{\boldsymbol{\theta}_p^{G_i}(\cdot)\}$ that improves the dual.*

According to this proposition, for example, stabilized nodes leave the dual function unmodified in sequential algorithms such as TRW-S or max-diffusion. But stabilized nodes also lead us to the central concept in our decimation method, that of an *R-nested* node: we say that node p is *R-nested* for the t-th iteration if both p and all other nodes of graph G within distance[4] R from p were found to be stabilized at that iteration (see Fig. 2(a)). Motivated by proposition 1, we have empirically verified the following two important observations: in practice, many nodes quickly become stabilized during a dual-based algorithm when a multigrid scheme is used, and, furthermore, stabilized nodes that consistently remain R-nested for a number of iterations (with R large enough) turn out to contribute a very small (even zero) total change to the dual objective thereafter. This leads to the following decimation strategy (that depends on two positive

[4] The distance of two nodes is the number of edges of their minimum connecting path.

integer parameters R, D): at each iteration, we fix all nodes that are stabilized at the current iteration and that were R-nested for the past D iterations (each such node is simply assigned one of its current stable labels). This strategy is applied after a few initial iterations have passed, while parameters R and D determine how fast nodes can become fixed, and must be set to some reasonably large values.

To intuitively understand the necessity for the conditions of the above decimation strategy notice that an R-nested node is essentially surrounded by a 'layer' (of width R) of stabilized nodes. Note also that if a node, say q, becomes non-stabilized at the current iteration, this means that q is able to contribute to the dual objective. This in turn implies that extra dual information (in the form of messages) can originate from q and propagate to nearby nodes, thus possibly affecting the labels of any node p within a certain distance, say R, from q. This explains why p must be R-nested. On the other hand, if a certain number of iterations, say D, have passed since the start of this propagation and p has still remained stabilized during all that time, it is highly likely that the new messages did not actually affect that node.

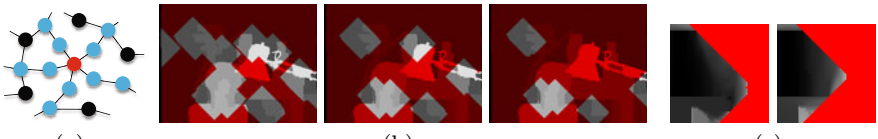


Fig. 2. (a) The red node is 2-nested, if itself and all blue nodes are stabilized. **(b)** Distribution of fixed nodes (red pixels) at 3 different iterations. **(c)** The same part of the 'confidence' map at 2 iterations of Tsukuba. More fixed nodes exist in the right map, which results into some non-fixed nodes becoming more 'confident' (*i.e.*, brighter).

As the dual-based algorithm progresses towards convergence, more and more nodes become fixed. This results into significant computational savings per iteration as only a very small number of dual variables have to be updated, which in turn results into a larger rate of improvement of the dual objective per iteration and thus in faster convergence. Fig. 2(b) shows examples from the distribution of the fixed nodes at different iterations of the multigrid algorithm for Tsukuba. Notice the order by which nodes become fixed: 'easier' nodes fix their labels earlier, while 'uncertain' nodes are fixed towards the end.

Another very important advantage of the decimation strategy is that, by fixing some of the labels, it manages to propagate additional information into the graph, which further increases the rate of improvement of the dual. This was found to considerably speed up convergence in our experiments. This propagation is illustrated by the 'confidence' maps for the 'tsukuba' example in Fig. 2(c), which show that, as a result of the decimation process, the 'confidence' of non-fixed nodes increases as well. Note that the confidence of a node p is calculated

by computing for each label the sum of its min-marginals for all the slave MRFs containing p and taking the difference between the two lowest sums.

But how can we empirically test the soundness of the above decimation process? A very strong empirical indication comes from the following fact: let us assume that the original dual LP relaxation is tight (or almost tight), *i.e.*, the resulting labels are (almost) optimal, which is the main case of interest. Note that each time we fix the label of a node, we are essentially modifying that relaxation. Moreover, the optimum of the modified dual relaxation increases only whenever a newly fixed node is assigned a suboptimal label. Therefore, in this case we can check how well the decimation process performed by simply comparing the original dual optimum with the dual optimum of the modified relaxation that results from fixing all the nodes. In all the real examples that we have tried, the two dual optima were either exactly the same (when the original relaxation was exact) or differed by a very small amount (when the original relaxation was almost tight). We have also verified this property with experiments on synthetic problems. Moreover, the obtained MRF energies were always better than the ones of the full algorithm (we found no case where this was not true).

Intuitively, the reason that we are able to obtain better primal solutions is because, by fixing some of the labels, we implicitly manage to gradually tighten the relaxation. Typically, LP-based solvers for MAP estimation function by solving the LP and then rounding each node to generate an integer solution. Instead, a better approach would be that, after rounding each node, we add its fixed state as an additional constraint to the LP and then solve this new LP before rounding the next node. This second approach, however, is very expensive but gives better solutions as the LP guiding the rounding scheme gets progressively tighter. The proposed decimation strategy can be thought of as an efficient way to approximately perform such an expensive series of computations. Stable nodes will have the same reparameterization in the final stage of the LP as they do now. Therefore, they can be immediately rounded, and their new solution propagated as a constraint. Note that the benefit of a decimation process to solving difficult problems has also been observed in other cases as well, *e.g.*, for solving SAT instances using survey propagation [8].

8 Extensions

Higher order MRFs: Due to the generality of the proposed formulation, the "algebraic multigrid" approach can also be extended to higher-order MRF optimization problems. These problems have the following form:

$$\mathrm{MRF}_G(\mathbf{U}, \mathbf{H}) := \min_{\mathbf{x}} \sum_{p \in \mathcal{V}} U_p(x_p) + \sum_{c \in \mathcal{C}} H_c(\mathbf{x}_c) \ , \tag{17}$$

where $\mathbf{H} = \{H_c\}$ are the higher order potential functions, which are now defined on cliques $c \in \mathcal{C}$ and replace the pairwise potentials $\mathbf{P}$.

Therefore, the dual objective function (2) now involves higher order potentials $\mathbf{H}$ (instead of $\mathbf{P}$), while the slave MRFs are defined on sub-hypergraphs G_i of

a hypergraph G [14]. The projection $\text{proj}(G)$ of any hypergraph G is defined analogously to the projection of a graph, *i.e.*, as the projection of its cliques. Similarly, the projection of an MRF with higher potentials $\mathbf{H}$ gives rise to an MRF with higher potentials $\bar{\mathbf{H}}$, which are again defined analogously to (6), *i.e.*,

$$\bar{H}_{\bar{c}}(\cdot) = \sum\nolimits_{c:\text{proj}(c)=\bar{c}} H_c(\cdot) \ . \tag{18}$$

Hence, by replacing $\mathbf{P}$ and $\bar{\mathbf{P}}$ with $\mathbf{H}$ and $\bar{\mathbf{H}}$ respectively, the restriction and prolongation operators PROJ and LIFT can then be computed using exactly the same algorithms as described in sections 5 and 6.

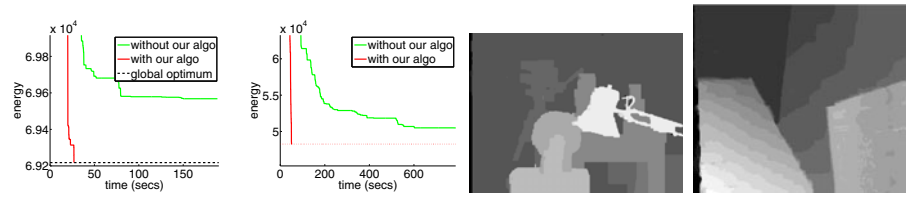
Tighter LP relaxations: In the dual decomposition framework, a tighter dual relaxation can result simply by choosing a set of non tree-structured slave MRFs. For instance, one can use loopy subgraphs of small tree-width for this purpose (intuitively, such a relaxation is tighter because the slaves now have higher optimal energies, and thus lead to better lower bounds). As a result, exactly the same algrebraic multigrid framework can be applied, thus leading to a multiresolution set of tighter relaxations in this case.

Data-driven projections: Typically the partitions that determine each projection in the hierarchy are chosen a priori (*e.g.*, for grids, a node at one level can project to a block of nodes at a coarser level). However, due to the generality of the proposed formulation, this could very well not be the case. Instead, one can use data driven partitions for defining these projections. In vision problems, for instance, it would be very useful to define these partitions so as to align with some of the edges in the image. If chosen properly, such data driven projections can lead to even greater computational savings.

9 Experimental Results

We have applied our method to a wide variety of vision problems. We first report results on pairwise MRFs. To this end, we tested our algorithm on the Middlebury dataset [5], which contains a variety of MRF problems on stereo matching, image segmentation and image denoising (all MRF potentials were set exactly the same as in that dataset). To demonstrate our framework for pairwise MRFs, we have used it to improve the TRW-S algorithm [2], which is a popular dual LP-based method for pairwise energies. We thus report results when we apply that algorithm with and without our framework. In both cases we use the same implementation of TRW-S as well as the same set of settings.[5] Slaves were chosen to be trees, with one tree per horizontal and vertical line of the input grid structured graph. We show typical plots of how the energy varies in Figs. 3(a),3(b) and the corresponding solutions in Figs. 3(c),3(d). Notice the much faster convergence when our framework is used. Further running times and energies for problems from the middlebury dataset are reported in Fig. 4. As

[5] For completeness we also compared our method with the original implementation of TRW-S by V. Kolmogorov (see the supplemental material [12] for these results).



(a) Tsukuba energies (b) Venus energies (c) (d)

Fig. 3. Convergence plots and results for Tsukuba and Venus

can be seen, our method provides a very significant speedup in all cases, while at the same time it increases the effectiveness of the optimization. In fact, it always computed solutions whose energy was lower than the best energy reported in the Middlebury dataset. This behaviour was consistent throughout all our experiments. For instance, for the 'tsukuba' example, our method computed the global optimum in a time that was at least an order of magnitude faster than the method in [15] (global optimality can be verified based on the dual lower bounds). For obtaining these results, we used an MRF hierarchy consisting of 3-5 levels, where the partition at each level was consisting of sets of 2×2 pixels. Also, parameters R and D (used in the decimation strategy) were set to some reasonably large values (*e.g.*, $R \geq 30$ and $D \geq 10$ on average).

We also tested our method on problems with higher order MRFs. To this end, we applied it to image segmentation and stereo matching problems, where we used a $\mathcal{P}^n$ Potts model [16] and a truncated second order derivative as higher order potentials, respectively. Both of them were solved using the framework of pattern-based potentials from [14]. We report indicative energies and running times for two such cases in Fig. 5(a), while Fig. 5(b) shows the corresponding result for stereo matching. As can be observed, even for high order MRF problems, our framework enables us to obtain high quality solutions much faster. It also increases the effectiveness of the optimization, as it still consistently leads to solutions of lower energy even in this case.

Finally, for completeness, we also compare our algorithm to the algorithm from [6] that uses BP in conjunction with a geometric multigrid method. Fig. 5(c)

	Without our algo		With our algo		
problem	energy	time(secs)	energy	time(secs)	speedup
Tsukuba	69568	188.53	69218*	26.93	7.00x
Venus	50392	795.11	48255	51.87	15.58x
Teddy	44505	1052.10	38299	131.25	8.00x
Flower	11274	2.74	11274*	0.46	5.95x
Sponge	27165	3.33	27165*	0.49	6.79x
Person	04852	5.94	04852*	0.65	9.13x
Penguin	54664	1016.30	48787	133.08	7.63x

* = global optimum

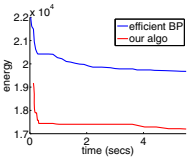
Fig. 4. Energies and running times for MRFs from the Middlebury dataset with and without our framework (energies have been normalized by subtracting a constant)

problem	Without our algo		With our algo		
	energy	time(secs)	energy	time(secs)	speedup
Segment (P^n potts)	325603	3.49	325603*	0.79	4.40x
Stereo (cones)	120840	181.27	118393	61.87	2.93x

* = global optimum

(a)

(b)



(c)

Fig. 5. (a) Energies and running times for high order MRFs. **(b)** Disparity for 'cones'. **(c)** Comparison between our method and the method in [6].

shows the convergence of the energy when these two algorithms are run on the stereo example from [6]. As can be seen, although the BP algorithm is very fast, our method computes a solution of lower energy even faster.

10 Conclusions

A framework for significantly improving the overall efficiency and effectiveness of dual-LP based methods was proposed in this paper, which is currently one of the main challenges encountered in energy minimization problems for vision. It relies on an algebraic multigrid approach and an efficient decimation strategy. It is also extremely general, and can be applied to both pairwise and higher order MRF problems. Due to this fact, and the very wide applicability of dual-LP based methods, we hope that our framework will help in making such methods much more practical for a wider class of vision problems in the future.

References

1. Wainwright, M., Jaakkola, T., Willsky, A.: Map estimation via agreement on trees: message-passing and linear programming. IEEE Trans. on Info. Theory (2005)
2. Kolmogorov, V.: Convergent tree-reweighted message passing for energy minimization. PAMI (2006)
3. Komodakis, N., Paragios, N., Tziritas, G.: MRF optimization via dual decomposition: Message-passing revisited. In: ICCV (2007)
4. Werner, T.: A linear programming approach to max-sum problem: A review. PAMI (2007)
5. Szeliski, R., et al.: A comparative study of energy minimization methods for markov random fields with smoothness-based priors. PAMI (2008)
6. Felzenszwalb, P.F., Huttenlocher, D.P.: Efficient belief propagation for early vision. IJCV 70, 41–54 (2006)
7. Komodakis, N., Tziritas, G.: Image completion using efficient belief propagation via priority scheduling and dynamic pruning. In: IEEE TIP (2007)
8. Braunstein, A., Mézard, M., Zecchina, R.: Survey propagation: An algorithm for satisfiability. Random Struct. Algorithms 27, 201–226 (2005)
9. Kovtun, I.: Partial optimal labeling search for a np-hard subclass of (max,+) problems. In: Michaelis, B., Krell, G. (eds.) DAGM 2003. LNCS, vol. 2781, pp. 402–409. Springer, Heidelberg (2003)

10. Alahari, K., Kohli, P., Torr, P.: Reduce, reuse and recycle: Efficiently solving multi-label mrfs. In: CVPR (2008)
11. Shekhovtsov, A., Kovtun, I., Hlaváč, V.: Efficient mrf deformation model for non-rigid image matching. In: CVIU (2008)
12. http://www.csd.uoc.gr/~komod/publications/docs/eccv10_supp.pdf
13. Dykstra, R.L.: An iterative procedure for obtaining i-projections onto the intersection of convex sets. Annals of Probability (1985)
14. Komodakis, N., Paragios, N.: Beyond pairwise energies: Efficient optimization for higher-order MRFs. In: CVPR (2009)
15. Meltzer, T., Yanover, C., Weiss, Y.: Globally optimal solutions for energy minimization in stereo vision using reweighted belief propagation. In: ICCV (2005)
16. Kohli, P., Kumar, P., Torr, P.: P3 and beyond: Solving energies with higher order cliques. In: CVPR (2007)