Beyond Pairwise Energies: Efficient Optimization for Higher-order MRFs

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

Nikos Paragios
Ecole Centrale de Paris/INRIA Saclay
nikos.paragios@ecp.fr

Abstract

In this paper, we introduce a higher-order MRF optimization framework. On the one hand, it is very general; we thus use it to derive a generic optimizer that can be applied to almost any higher-order MRF and that provably optimizes a dual relaxation related to the input MRF problem. On the other hand, it is also extremely flexible and thus can be easily adapted to yield far more powerful algorithms when dealing with subclasses of high-order MRFs. We thus introduce a new powerful class of high-order potentials, which are shown to offer enough expressive power and to be useful for many vision tasks. To address them, we derive, based on the same framework, a novel and extremely efficient message-passing algorithm, which goes beyond the aforementioned generic optimizer and is able to deliver almost optimal solutions of very high quality. Experimental results on vision problems demonstrate the extreme effectiveness of our approach. For instance, we show that in some cases we are even able to compute the global optimum for NP-hard higher-order MRFs in a very efficient manner.

1. Introduction

MRF inference is extremely popular in computer vision and related fields. However, with a few exceptions only, its use was mainly confined to the case of pairwise MRFs up to now. One reason is because optimization of higher order MRFs can be extremely difficult (*i.e*., algorithms that yield almost optimal solutions are hard to get in this case) and, furthermore, these algorithms often have a very high computational cost that is prohibitive in practice. Yet, many vision problems could greatly benefit from the use of higher order models as this would allow far more expressive priors to be encoded, and also multiple interactions to be captured. This would, in turn, lead to a far better and more accurate modelling, which is clearly needed by many vision tasks (*e.g*., notice that in many cases there is a large disagreement between the global optimum, that can often be computed for pairwise MRFs, and the ground truth).

Towards dealing with the above issues, we propose here a powerful framework for high-order MRF optimization (§ 2). It uses a *master-slave* based scheme, which relies on the core idea that even a hard high-order MRF problem (with, *e.g*., large cliques or complicated structure) can often be decomposed into high-order MRF subproblems that are very easy or even trivial to solve. This leads to a very general and flexible framework. Hence, on the one hand, we use it to derive a generic optimizer, which is applicable to almost any high-order MRF, and which provably computes the global optimum to a strong dual LP-relaxation of the MRF problem (§ 2.1). On the other hand, due to its flexibility, the proposed framework can also be easily adapted to lead to even more powerful algorithms when it comes to dealing with specific classes of high-order MRFs. To illustrate this, we also introduce here a new class of high-order potentials, called *pattern-based* potentials, which offer great expressive power and can be useful for many vision tasks (§ 3). By relying again on the same framework, a powerful and extremely efficient message-passing algorithm is proposed in this case. This algorithm goes beyond the aforementioned generic optimizer and is able to deliver solutions of very high quality (§ 3.1). As a result, for the first time, we show experimentally that in some cases we are even able to compute the global optimum for NP-hard high-order MRFs used in vision, and, furthermore, we can do that in a very efficient manner, *e.g*., at a fraction of the time that would be required by a generic message-passing scheme (§ 4). We thus hope that our framework will further promote the applicability of higher-order models to vision.

We should note that not many MRF algorithms for high-order vision problems have been proposed up to now. A notable exception is the recent work of Kohli *et al*. [3, 4], where an efficient inference technique was proposed for a specific class of higher-order MRFs. Lan *et al*. [8] presented an efficient but approximate version of BP, while Potetz [10] proposed a BP adaptation for a certain class of high-order graphical models. The n-ary max-sum diffusion method has been very recently proposed by Werner [12], while two other works [2, 11], that appeared concurrently with ours, address high-order MRF optimization by reducing it to a pairwise problem with binary or multi-label variables. This can lead to a very compact representation in some cases.

2. Dual Decomposition & high-order MRFs

The problem of MRF optimization is defined as follows: we are given a hypergraph $G = (\mathcal{V}, \mathcal{C})$, consisting of a set

978-1-4244-3991-1/09/$25.00 ©2009 IEEE

of nodes $\mathcal{V}$ and a set of hyperedges[1] $\mathcal{C}$. We also assume that each node $q \in \mathcal{V}$ can take a label x_q from a discrete set of labels $\mathcal{L}$. Our goal is then to find a labeling $\mathbf{x} = \{x_q\}$ that solves the following optimization task:

$$\mathrm{MRF}_G(\mathbf{U}, \mathbf{H}) \equiv \min_{\mathbf{x}} \sum_{q \in \mathcal{V}} U_q(x_q) + \sum_{c \in \mathcal{C}} H_c(\mathbf{x}_c) \qquad (1)$$

Here $\mathbf{U}_q = \{U_q(\cdot)\}$, $\mathbf{H}_c = \{H_c(\cdot)\}$ represent respectively the unary potentials (for node q) and the higher order potentials (for clique c), while notation $\mathbf{x}_c$ denotes the set $\{x_q | q \in c\}$. We will hereafter refer to the above problem as $\mathrm{MRF}_G(\mathbf{U}, \mathbf{H})$ to denote its dependence on the hypergraph G and on the potentials $\mathbf{U} = \{\mathbf{U}_q\}_{q \in \mathcal{V}}$, $\mathbf{H} = \{\mathbf{H}_c\}_{c \in \mathcal{C}}$.

To tackle problem $\mathrm{MRF}_G(\mathbf{U}, \mathbf{H})$ in its full generality, we will rely on the recently introduced MRF optimization framework of Dual Decomposition [7]. According to that framework, for solving a hard MRF problem (hereafter called the master), we first decompose it into a series of MRF subproblems that are easy to optimize (the so-called slave MRFs), and then we extract a solution to it by cleverly combining the solutions from the slaves. The latter can be done in a rigorous manner by using an iterative projected-subgradient scheme. This framework has been previously used for optimizing pairwise MRFs, but it turns out that it can be easily extended to the higher order case, *i.e.*, for solving problem $\mathrm{MRF}_G(\mathbf{U}, \mathbf{H})$ (see Fig. 2(a)). To this end, one simply needs to decompose the original hypergraph $G = (\mathcal{V}, \mathcal{C})$ into a set of sub-hypergraphs $\{G_i = (\mathcal{V}_i, \mathcal{C}_i)\}$ such that $\mathcal{V} = \cup \mathcal{V}_i$ and $\mathcal{C} = \cup \mathcal{C}_i$. An MRF with unary potentials $\mathbf{U}^{G_i}$ and higher order potentials $\mathbf{H}^{G_i}$ is then defined for each of these sub-hypergraphs. Furthermore, all these potentials are chosen so as to provide a decomposition of the original MRF potentials, *i.e.*, it holds $\mathbf{U} = \sum_i \mathbf{U}^{G_i}$, $\mathbf{H} = \sum_i \mathbf{H}^{G_i}$, which is easily ensured simply by setting $\mathbf{U}_q^{G_i} = \frac{\mathbf{U}_q^G}{|\{i | q \in \mathcal{V}_i\}|}$, $\mathbf{H}_c^{G_i} = \frac{\mathbf{H}_c^G}{|\{i | c \in \mathcal{C}_i\}|}$. By using these MRFs as slaves and applying the Dual Decomposition framework, we derive the algorithm appearing in Fig. 1.[2] As can be seen, it alternates between optimizing the slave MRFs (line 3) and updating their potentials $\mathbf{U}^{G_i}$ based on the extracted optimizers (line 5). The latter essentially relies on averaging these optimizers. Intuitively, the goal is thus to make the slave MRFs to reach a consensus and agree with each other on their labels for common nodes (in which case the labeling is a global optimum). Upon convergence a solution can be extracted by, *e.g.*, traversing the slave MRFs and copying their optimizers until all nodes of the master MRF get labeled. Metaphorically, slave optimizers $\mathbf{x}^{G_i}$ may also be viewed as amount of resources, and potentials $\mathbf{U}^{G_i}$ as corresponding market prices. Hence, master sets the prices and then lets the slaves choose how many resources to consume. Naturally, prices are adjusted so that the market finally clears: they thus increase/decrease for overutilized/underutilized resources in line 5. We note that the multipliers $\alpha_t \geq 0$ (one per iteration) used by the algorithm merely need to satisfy $\lim_{t \to \infty} \alpha_t = 0$, $\sum_t \alpha_t = \infty$.

[1]The term hyperedge or clique will refer to a subset of nodes from $\mathcal{V}$.

[2]Here we assumed that each clique $c \in \mathcal{C}$ belongs to only one $\mathcal{C}_i$ (*i.e.*, $\mathcal{C}_i \cap \mathcal{C}_j = \emptyset$, $\forall i \neq j$). This allows us not to have to update the high-order potentials $\mathbf{H}^{G_i}$ of the slave MRFs. We note, however, that the Dual Decomposition framework can be easily adapted to handle such cases.

1: **Input:** set of slaves $\{\mathrm{MRF}_{G_i}(\mathbf{U}^{G_i}, \mathbf{H}^{G_i})\}$
2: **repeat**
3: $\quad \mathbf{x}^{G_i} = \texttt{optimize}\big(\mathrm{MRF}_{G_i}(\mathbf{U}^{G_i}, \mathbf{H}^{G_i})\big)$, $\forall i$
4: $\quad \bar{\mathbf{x}}_q^{G_i} = \big\{\big[x_q^{G_i} = l\big]\big\}_{l \in \mathcal{L}}$, where $[\cdot]$ is Iverson bracket
5: $\quad \mathbf{U}_q^{G_i} \mathrel{+}= \alpha_t \cdot \left(\bar{\mathbf{x}}_q^{G_i} - \frac{\sum_{j \in \mathcal{G}(q)} \bar{\mathbf{x}}_q^{G_j}}{|\mathcal{G}(q)|}\right)$, with $\mathcal{G}(q) = \{j | q \in G^j\}$
6: **until** convergence

Fig. 1: Dual decomposition scheme for high-order MRFs.

Simply by choosing different decompositions $\{G_i\}$, different algorithms can be derived via the above scheme, all of which can be shown to provably optimize (possibly different) dual relaxations to the MRF problem. At each iteration the sum of the minimum energies of the slaves provides a lower bound to the minimum energy of the master MRF, and the maximum of these bounds coincides with the optimum of the underlying dual relaxation. As a result, the more difficult the slave MRFs (*e.g.*, the more complex the topology of sub-hypergraphs G_i), the better the lower bounds and thus the tighter the underlying dual relaxation (this, of course, means that it better approximates the master MRF).

2.1. A generic optimizer for higher order MRFs

The above framework provides great flexibility as to how the slave MRFs (or equivalently the sub-hypergraphs G_i) are chosen (the only practical requirement is that one must be able to compute the optimizers of the resulting slave MRFs). Based on this fact, we will derive in this section a generic algorithm for high-order MRF optimization by using the following choice of sub-hypergraphs: to each clique $c \in \mathcal{C}$ we will associate one sub-hypergraph $G_c = (\mathcal{V}_c, \mathcal{C}_c)$, whose only hyperedge will be c and its set of nodes will consist of all the nodes contained in that hyperedge, *i.e.*, $\mathcal{V}_c = \{q | q \in c\}$, $\mathcal{C}_c = \{c\}$. Note that the resulting slaves are much easier problems to solve, as, essentially, one only needs to be able to optimize MRFs with a single high-order clique (if this is still difficult to achieve, then there is probably not much hope for solving the original problem in the first place). Due to this fact, the resulting algorithm can be applied to almost any high-order MRF and thus can be considered as a generic optimizer. Not only that but, as the next theorem certifies, it actually computes the global optimum to a strong LP relaxation that often proves to be a very good approximation to problem $\mathrm{MRF}_G(\mathbf{U}, \mathbf{H})$.

Theorem 1 ([1]). *If the sub-hypergrpahs are chosen as described above (*i.e.*, one sub-hypergraph G_c per clique c), then the algorithm optimizes the LP relaxation of the follow-*

ing integer LP that is equivalent to problem $\mathrm{MRF}_G(\mathbf{U},\mathbf{H})$:

$$\min_{\mathbf{z}} \sum_q \sum_{x_q} U_q(x_q) z_q(x_q) + \sum_c \sum_{\mathbf{x}_c} H_c(\mathbf{x}_c) z_c(\mathbf{x}_c) \quad (2)$$

$$\text{s.t.} \sum\nolimits_{x_q} z_q(x_q) = 1\,, \quad \forall q \quad (3)$$

$$\sum\nolimits_{\mathbf{x}_c : x_q = l} z_c(\mathbf{x}_c) = z_q(l)\,, \quad \forall c \in \mathcal{C}, q \in c \quad (4)$$

$$z_q(\cdot), z_c(\cdot) \in \{0,1\}\ , \quad (5)$$

where a variable $z_q(x_q)$*,* $z_c(\mathbf{x}_c)$ *is associated respectively with each label* x_q *of node* q *and each label* $\mathbf{x}_c$ *of clique* c*.*

Note that in the case where all high-order potentials are pairwise, this relaxation reduces to the well known LP-relaxation upon which many state-of-the-art algorithms for pairwise MRFs currently rely [12].

3. Pattern-based high order potentials

To illustrate the extreme flexibility of our framework, in this section we will attempt to go beyond the above generic optimizer by concentrating on a broad subclass of MRFs that are based on a new set of high-order potentials, called *pattern-based* potentials. As we shall see, these potentials offer enough expressive power to be useful for modeling many vision tasks (*e.g.*, they include as special cases many high-order potentials frequently used in vision). In addition to that, however, they lend themselves to very powerful and efficient inference algorithms, as they allow much stronger decompositions to be used. A pattern-based potential $H_c(\cdot)$ of a clique c is defined as:

$$H_c(\mathbf{x}) = \begin{cases} \psi_c(\mathbf{x}) & \text{if } \mathbf{x} \in \mathcal{P} \\ \psi_c^{\max} & \text{otherwise}\ . \end{cases} \quad (6)$$

Here $\mathcal{P}$ is a set of vectors from $\mathcal{L}^{|c|}$ that correspond to labelings of clique c (these vectors will be called *patterns* hereafter), and the main assumption is that this set of patterns is sparse (*i.e.*, it contains much fewer elements than the set $\mathcal{L}^{|c|}$ of all possible labelings of c). The only other restriction that we impose is that the high-order function $\psi_c(\cdot)$ should satisfy $\psi_c(\mathbf{x}) \le \psi_c^{\max},\ \forall \mathbf{x} \in \mathcal{P}$ (other than that, $\psi_c(\cdot)$ can take arbitrary values).

To simplify the exposition and reduce notational clutter, for the remainder of this section we will assume w.l.o.g. that the hypergraph $G = \{\mathcal{V}, \mathcal{C}\}$ has a a grid-like structure (a typical case for many vision problems). In particular, we assume that its nodes $\mathcal{V}$ are arranged in a grid of size $N_y \times N_x$, while all its cliques $\mathcal{C}$ have the same size $K_y \cdot K_x$ and are in 1-1 correspondence with the set of all $K_y \times K_x$ subrectangles in the grid. For short, $G_{\{y:y',x:x'\}}$ will hereafter denote the sub-hypergraph induced by the nodes lying in the grid-rectangle $[y:y', x:x']$ (*e.g.*, under this notation $G = G_{\{1:N_y,1:N_x\}}$).

The generic algorithm of sec. 2.1 always uses a decomposition consisting of sub-hypergraphs of the form $G_{\{y+1:y+K_y,x+1:x+K_x\}}$. For pattern-based potentials, however, we will manage to use a stronger decomposition that consists of much larger sub-hypergraphs: namely of all horizontal sub-hypergraphs of the form $G_{\{y+1:y+K_y,1:N_x\}}$, as well as of all vertical sub-hypergraphs of the form $G_{\{1:N_y,x+1:x+K_x\}}$ (see Fig. 2(b)). As explained, this will lead us to a far more powerful algorithm that optimizes a tighter dual relaxation to the MRF problem. Of course, a critical question is how to compute the global optimum for the much more complicated high-order MRFs associated with the larger sub-hypergraphs. To this end, a very efficient message-passing algorithm is proposed in the next section.

3.1. Message-passing for higher-order slave MRFs

Since all sub-hypergraphs are isomorphic, it suffices to consider one of them, *e.g.*, $G_{\{1:K_y,1:N_x\}}$. We will first present our message-passing algorithm assuming that a clique's vertical dimension is $K_y = 1$, and we will explain afterwards how the case $K_y > 1$ reduces to this one. Under this assumption, our slave MRF consists of N nodes $q_1, q_2, \ldots, q_N$ (where $N = N_x$), as well as $N-K+1$ cliques each of size K (where $K = K_x$). The i-th clique (denoted c_i) contains nodes $q_i, \ldots, q_{i+K-1}$ and its associated potential $H_i(\cdot)$ will be of form (6), while the unary potential of node q_i will be denoted by $U_i(\cdot)$. Furthermore, in this case, each pattern $\mathbf{x} \in \mathcal{P}$ will be a vector of K components. Notation $\mathcal{P}_{[k:l]}$ will represent the set of all different *subpatterns* $\mathbf{x}_{[k:l]}$ that can be formed from patterns in $\mathcal{P}$, *i.e.*,

$$\mathcal{P}_{[k:l]} = \left\{ \mathbf{x}_{[k:l]} \,|\, \mathbf{x} \in \mathcal{P} \right\}\ ,$$

where, for any vector $\mathbf{x}$, we denote by $\mathbf{x}_{[k:l]}$ its subvector starting from the k-th and ending at the l-th component. Note that, provided $\mathcal{P}$ is sparse, each set $\mathcal{P}_{[k:l]}$ is sparse as well. Given the i-th MRF node (*i.e.*, node q_i), we define its partial energy, $E_i(\cdot)$, as the energy resulting from considering all potentials (excluding $U_i(\cdot)$) that involve only variables associated with nodes from the set $\{q_1, \ldots, q_i\}$, *i.e.*, $E_i(\cdot) = \sum_{j<i} U_j(\cdot) + \sum_{j \le i-K+1} H_j(\cdot)$. During the algorithm's execution, a set of partial energies, called *messages*, are maintained. These are denoted by $\mathcal{M}_i^k(\mathbf{x}_{[1:k]})$ and have the following interpretation: assuming that nodes $q_{i-k+1}, \ldots, q_i$ are labeled respectively with the components $x_1, \ldots, x_k$ of vector $\mathbf{x}_{[1:k]}$, then message $\mathcal{M}_i^k(\mathbf{x}_{[1:k]})$ represents the minimum partial energy up to node q_i attainable under these conditions.

The key observation to deriving an efficient algorithm is that, despite the existence of an exponentially large number of possible messages, only a very small subset of them needs to be maintained for computing the global optimum. In particular, we need to maintain the following messages:

$$\mathcal{M}_i^1(\mathbf{x})\,, \ \ \forall \mathbf{x} \in \mathcal{P}_{[1:1]} \cup \mathcal{P}_{[K:K]}\ , \quad (7)$$

$$\mathcal{M}_i^k(\mathbf{x})\,, \ \ \forall \mathbf{x} \in \mathcal{P}_{[1:k]}\ , \ \text{where } 1 < k < K\ . \quad (8)$$

Note that the above set of messages is small, because only $\mathbf{x}$ that are subpatterns have to be considered (recall that $\mathcal{P}_{[1:k]}$ and $\mathcal{P}_{[K:K]}$ are assumed to be sparse). To also take into account all other $\mathbf{x}$ (*i.e.*, $\mathbf{x}$ that are not subpatterns), it turns out we only need to consider just a few extra messages. These are denoted by $\widehat{\mathcal{M}}_i^1$ and have the following interpretation: assuming that node q_i is assigned a label not belonging to the set of subpatterns $\mathcal{P}_{[1:1]} \cup \mathcal{P}_{[K:K]}$, then message $\widehat{\mathcal{M}}_i^1$ represents the minimum partial energy up to node q_i attainable under these conditions, *i.e.*, it holds

$$\widehat{\mathcal{M}}_i^1 = \min_{\mathbf{x} \notin \mathcal{P}_{[1:1]} \cup \mathcal{P}_{[K:K]}} \mathcal{M}_i^1(\mathbf{x}) \tag{9}$$

Messages (7), (8) and (9) can be computed very efficiently in a recursive manner by traversing the MRF nodes in the order $q_1, q_2, \ldots, q_N$. When the current node is q_i, we compute all messages related to that node, *i.e.*, messages $\{\mathcal{M}_i^k(\cdot), \widehat{\mathcal{M}}_i^1\}$, and this is done based on the messages related to the previous node q_{i-1}, *i.e.*, messages $\{\mathcal{M}_{i-1}^k(\cdot), \widehat{\mathcal{M}}_{i-1}^1\}$ (which have been computed at the previous step). The pseudocode for these recursive message updates is shown in Fig. 2. The updating of message $\widehat{\mathcal{M}}_i^1$ is done at lines 4-5. The role of the loop at lines 7-9 is to update the rest of the messages $\mathcal{M}_i^k(\cdot)$ by first considering the case where a pattern $\mathbf{x}_{[1:K]} \in \mathcal{P}$ is assigned to the nodes of the clique ending at the current node q_i (*i.e.*, the $(i-K+1)$-th clique c_{i-K+1}). Subject to this condition we compute the resulting minimum partial energy up to q_i, denoted as E_{pattern} at line 8, which suffices for updating $\mathcal{M}_i^k(\cdot)$ at line 9. E_{pattern} obviously equals the minimum partial energy up to the previous node q_{i-1} (given by message $\mathcal{M}_{i-1}^{K-1}(\mathbf{x}_{[1:K-1]})$) plus the two potentials not taken into account by that message[3], *i.e.*, the unary potential $U_{i-1}(\mathbf{x}_{[K-1:K-1]})$ at q_{i-1} and the high order potential $\psi_{i-K+1}(\mathbf{x}_{[1:K]})$ for the clique c_{i-K+1} terminating at q_i. Proceeding in a similar manner, we finish the update of messages $\mathcal{M}_i^k(\cdot)$ by considering at lines 11-13 the remaining case where a non-pattern is assigned to the nodes of the clique c_{i-K+1} ending at q_i. The potential of c_{i-K+1} will then equal $\psi_{i-K+1}^{\max}$, which leads to the update at line 13. The important thing to notice here is that, as can be seen from the pseudocode in Fig. 2, the time needed for updating the messages is linear with respect to the size of the set of patterns $\mathcal{P}$. This, in turn, means that the messages can be computed very fast, which makes the algorithm very efficient and thus practical even for large scale problems with large clique sizes.

Upon reaching last node q_N, each sum $\mathcal{M}_N^1(\cdot) + U_N(\cdot)$ (or $\widehat{\mathcal{M}}_N^1 + \min_l U_N(l)$) of a message and a unary term corresponds, by definition, to a min-marginal energy, and so q_N gets the label that attains the minimum of all these sums, *i.e.*, either label $\arg\min_l(\mathcal{M}_N^1(l) + U_N(l))$ or label $\arg\min_l U_N(l)$ (based on whether it holds $\min_l(\mathcal{M}_N^1(l) + U_N(l)) < \widehat{\mathcal{M}}_N^1 + \min_l U_N(l)$). In the latter case the active message for node q_N is $\widehat{\mathcal{M}}_N^1$, whereas in the former case it is $\mathcal{M}_N^1(l_N)$, where l_N is the label assigned to q_N (the active message of a node represents its partial energy at the optimal labeling). To recover the optimal labels for the rest of the nodes, we simply need to do a backtracking, *i.e.*, we traverse nodes in reverse order and keep track of active messages (this turns out to reduce to a series of lookup operations due to that all necessary information can already be computed for free during the forward pass).

To apply the above algorithm for $K_y > 1$, one simply uses columns of nodes (each of length K_y) instead of single nodes. In such a case, a series of N columns $q_1, \ldots, q_N$ results, while $\mathbf{x}_{[k:l]}$ represents the submatrix (of a matrix $\mathbf{x}$) starting at the k-th and ending at the l-th column of $\mathbf{x}$. Furthermore, for any column vector $\mathbf{x} = \{x_j\}$, terms $U_i(\mathbf{x})$ and $\min_{\mathbf{x}}(U_i(\mathbf{x}))$ in Fig. 2 must be replaced with $\sum_j U_{ij}(x_j)$ and $\sum_j \min_{x_j}(U_{ij}(x_j))$, where j indexes the nodes in column q_i. The same algorithm is then applicable.

3.2. Min-marginals for slave MRFs

Given a decomposition into sub-hypergraphs $\{G_i\}$, the pseudocode in Fig. 1 uses the optimizers $\mathbf{x}^{G_i}$ of the corresponding slave MRFs to update the unary potentials $\mathbf{U}^{G_i}$ Alternatively, the node min-marginals can be used for that purpose. In this case, assuming that $\boldsymbol{\Delta}_q^{G_i}$ denotes the vector of min-marginals of node q (with respect to the slave MRF on G_i), the potentials $\mathbf{U}^{G_i}$ are updated as follows:

$$\mathbf{U}_q^{G_i} \mathrel{+}= \frac{\sum_{j \in \mathcal{G}(q)} \boldsymbol{\Delta}_q^{G_j}}{|\mathcal{G}(q)|} - \boldsymbol{\Delta}_q^{G_i}, \text{where } \mathcal{G}(q) = \{i | q \in G_i\}.$$

The above update can also be viewed as generalizing TRW methods [5, 6] to the case of higher-order MRFs.

In the case of pattern-based potentials, to compute the min-marginals $\Delta_i(\cdot)$ of node q_i, one needs to compute an additional set of messages, called *reverse messages*:

$$\widehat{\mathfrak{M}}_i^1, \mathfrak{M}_i^1(\mathbf{x}), \quad \forall \mathbf{x} \in \mathcal{P}_{[1:1]} \cup \mathcal{P}_{[K:K]} \ , \tag{10}$$

$$\mathfrak{M}_i^k(\mathbf{x}), \quad \forall \mathbf{x} \in \mathcal{P}_{[K-k+1:K]}, \ 1 < k < K \ . \tag{11}$$

These play the same role with normal messages if the order of nodes is reversed, *e.g.*, $\mathfrak{M}_i^k(\cdot)$ represent minimum partial energies that take into account all potentials involving nodes from $\{q_i, \ldots, q_N\}$ (we again exclude potential $U_i(\cdot)$ as in normal messages). Reverse messages can thus be computed during a backward pass by using the same algorithm as in Fig. 2. Given both reverse and normal messages, the pseudocode in Fig. 3 is used for the min-marginals estimation. In this code, min-marginals $\Delta_i(\cdot)$ of q_i are updated using sums of the form $\mathcal{M}_i(\cdot) + \mathfrak{M}_i(\cdot) + U_i(\cdot) + \overline{\Delta}$,

[3] Recall that, by definition, a message $\mathcal{M}_{i-1}^k(\cdot)$ takes into account potentials only up to node q_{i-1} excluding unary potential $U_{i-1}(\cdot)$

INPUT: messages $\widehat{\mathcal{M}}^1_{i-1}, \mathcal{M}^k_{i-1}(\cdot)$, **OUTPUT:** messages $\widehat{\mathcal{M}}^1_i, \mathcal{M}^k_i(\cdot)$

$\widehat{\mathcal{M}}^1_i \leftarrow +\infty,\ \mathcal{M}^k_i(\cdot) \leftarrow +\infty$

$\widehat{\mathcal{M}}^1_i \leftarrow \min_{\mathbf{x}\in\mathcal{P}_{[K:K]}}(\mathcal{M}^1_{i-1}(\mathbf{x}) + U_{i-1}(\mathbf{x}) + \psi^{\max}_{i-K+1})$

$\widehat{\mathcal{M}}^1_i \overset{\text{MIN}}{\leftarrow} \widehat{\mathcal{M}}^1_{i-1} + \min_{\mathbf{x}}(U_{i-1}(\mathbf{x})) + \psi^{\max}_{i-K+1}$

For each pattern $\mathbf{x} \in \mathcal{P}$ **do**

  $E_{\text{pattern}} \leftarrow \mathcal{M}^{K-1}_{i-1}(\mathbf{x}_{[1:K-1]}) + U_{i-1}(\mathbf{x}_{[K-1:K-1]}) + \psi_{i-K+1}(\mathbf{x})$

  if $\mathbf{x}_{[K-k+1:K]} \in \mathcal{P}_{[1:k]}$ **then** $\mathcal{M}^k_i(\mathbf{x}_{[K-k+1:K]}) \overset{\text{MIN}}{\leftarrow} E_{\text{pattern}},\ 1 \le k < K$

$\mathcal{M}^1_i(\mathbf{x}) \overset{\text{MIN}}{\leftarrow} \widehat{\mathcal{M}}^1_i,\ \forall \mathbf{x} \in \mathcal{P}_{[1:1]} \cup \mathcal{P}_{[K:K]}$

For each pattern $\mathbf{x} \in \mathcal{P},\ 1 < k < K$ **do**

  $\mathcal{M}^k_i(\mathbf{x}_{[1:k]}) \overset{\text{MIN}}{\leftarrow} \mathcal{M}^{k-1}_{i-1}(\mathbf{x}_{[1:k-1]}) + U_{i-1}(\mathbf{x}_{[k-1:k-1]}) + \psi^{\max}_{i-K+1}$

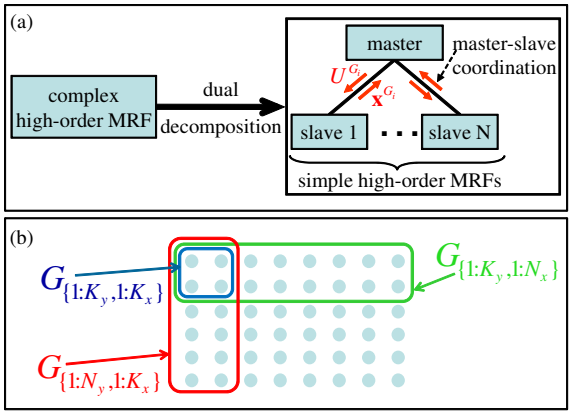


Fig. 2: (Left) Pseudocode of recursive message updates (see text for explanation). $\mathbf{x} \overset{\text{MIN}}{\leftarrow} \mathbf{y}$ represents $\mathbf{x} \leftarrow \min(\mathbf{x}, \mathbf{y})$. **(a)** Dual decomposition for high-order MRFs. **(b)** Grid of size $N_x = 8$, $N_y = 6$, with cliques of size $K_x = 2$, $K_y = 2$. A single-clique slave (blue) used by the generic optimizer, as well as an horizontal (green) and a vertical (red) slave used in the decomposition for pattern-based potentials.

$\overline{\mathcal{C}} = \{j \mid \text{clique } c_j \not\subseteq \{q_1, \dots, q_i\}, c_j \not\subseteq \{q_i, \dots, q_N\}\}$

$\overline{\Delta} = \sum_{j\in\overline{\mathcal{C}}} \psi^{\max}_j$

$\Delta_i(\cdot) \leftarrow \widehat{\mathcal{M}}_i + \widehat{\mathfrak{M}}_i + U_i(\cdot) + \overline{\Delta}$

$\Delta_i(\mathbf{x}) \overset{\text{MIN}}{\leftarrow} \mathcal{M}_i(\mathbf{x}) + \mathfrak{M}_i(\mathbf{x}) + U_i(\mathbf{x}) + \overline{\Delta},\ \forall \mathbf{x} \in \mathcal{P}_{[1:1]} \cup \mathcal{P}_{[K:K]}$

$\forall \mathbf{x}_{[1:K+\delta]}$ with $\mathbf{x}_{[1:K]} \in \mathcal{P},\ \mathbf{x}_{[\delta+1:K]} \in \mathcal{P},\ 0 \le \delta < K-2$

  For each $k \in \{\delta+2, \dots, K-1\}$ **do**

    $\overline{\mathcal{C}}_0 = \{j \in \overline{\mathcal{C}} \mid \text{clique } c_j \subseteq \{q_{i-k+1}, \dots, q_{i-k+K+\delta}\}\}$

    Assume $\{q_{i-k+1}, \dots, q_{i-k+K+\delta}\}$ take the labels $\mathbf{x}_{[1:K+\delta]}$

    and let ψ'_j be the evaluated potential for clique c_j with $j \in \overline{\mathcal{C}}_0$

    $\overline{\Delta} = \sum_{j\in\overline{\mathcal{C}}\setminus\overline{\mathcal{C}}_0} \psi^{\max}_j + \sum_{j\in\overline{\mathcal{C}}_0} \psi'_j$

    $\Delta_i(x_k) \overset{\text{MIN}}{\leftarrow} \mathcal{M}_i(\mathbf{x}_{[1:k]}) + \mathfrak{M}_i(\mathbf{x}_{[k:K+\delta]}) + U_i(\mathbf{x}_k) + \overline{\Delta}$

  end for

Fig. 3: Pseudocode for computing the min-marginals $\Delta_i(\cdot)$ of node q_i (see text for explanation). Superscripts for messages $\mathcal{M}_i(\mathbf{x})$, $\mathfrak{M}_i(\mathbf{x})$ can be deduced from the size of $\mathbf{x}$, and, so we drop them to reduce notational clutter.

where we drop superscripts to reduce notational clutter. The terms $\mathcal{M}_i(\cdot)$, $\mathfrak{M}_i(\cdot)$ account for all potentials involving respectively nodes $\{q_1, \dots, q_i\}$ and $\{q_i, \dots, q_N\}$ (excluding potential $U_i(\cdot)$). Hence the first 3 terms account for all potentials except for those corresponding to cliques in the set $\overline{\mathcal{C}} = \{\text{cliques } c \mid c \not\subseteq \{q_1, \dots, q_i\}, c \not\subseteq \{q_i, \dots, q_N\}\}$, which are taken into account by the last term $\overline{\Delta}$. Based also on the fact that $\psi_i(\mathbf{x}) \le \psi^{\max}_i$, it is easy to verify that the above code correctly estimates the min-marginals of q_i.

3.3. Instances of pattern-based potentials

Just as an illustration, we describe here a few instances of pattern-based potentials that can be useful in vision tasks.

Arbitrary potentials for small $|\mathcal{L}|$ and not very large cliques: The number of all possible labeling for a clique of size K equals $|\mathcal{L}|^K$. Hence, if both K and the cardinality of the label set $|\mathcal{L}|$ are relatively small, one can very efficiently model an arbitrary potential function as a pattern-based potential simply by using all possible clique labelings as patterns. This applies especially to the case of binary optimization problems (*i.e.*, $|\mathcal{L}| = 2$), which are so frequently used in computer vision. For instance, a very useful example in this class, that could be applied to a wide variety of vision tasks, would be the case of binary fusion moves [13] with low-size cliques.

Higher-order truncated potentials: One common way to regularize many low level vision problems is by penalizing (in a robust way) solutions that fail to satisfy certain high order smoothness assumptions. *E.g.*, the following robust 3rd order potential serves such a role:

$$H(\mathbf{x}) = \begin{cases} 0 & \text{if } |\mathcal{F}\mathbf{x}| = 0,\ |\nabla\mathbf{x}|_\infty \le \alpha_{\max} \\ \kappa & \text{otherwise} \end{cases} \tag{12}$$

It uses the 2nd order derivative $\mathcal{F}\mathbf{x} = x_1 - 2x_2 + x_3$ (computed via filter $\mathcal{F} = [1\ -2\ \ 1]$) to robustly favor piecewise planar solutions. Furthermore, $\alpha_{\max}$ denotes the maximum gradient magnitude (in $|\cdot|_\infty$ norm) that can be attained inside the smooth part of a recovered solution. This is an instance of a pattern-based potential with patterns of the form $(x, x+\alpha, x+2\alpha)$ or $(x+2\alpha, x+\alpha, x)$, whose total number equals $2(\alpha_{\max}+1)(|\mathcal{L}| - \alpha_{\max}) - |\mathcal{L}|$. Given that $\alpha_{\max}$ can be safely assumed to be small in practice, the number of patterns is linear with respect to $|\mathcal{L}|$ and thus the message-passing algorithm of sec. 3 becomes very efficient in this case. Note, of course, that this can be true if other filters, possibly of even higher order, are used as well.

A similar reasoning also applies to the following more general class of robust potentias, which serve a similar role:

$$H(\mathbf{x}) = \begin{cases} |\mathcal{F}\mathbf{x}| & \text{if } |\mathcal{F}\mathbf{x}| \le \kappa,\ |\nabla\mathbf{x}|_\infty \le \alpha_{\max} \\ \kappa & \text{otherwise} \end{cases} \tag{13}$$

Functions of this form are again very sparse, since they assign a constant value to the great majority of input vectors $\mathbf{x}$, and thus they can be expressed as pattern based potentials too (*e.g.*, if $\mathcal{F} = [1\ -2\ \ 1]$ then a very coarse upper bound on

the resulting number of patterns is $4\alpha_{\max}^2|\mathcal{L}|$). As a result, these potentials can be easily handled in a unified manner by our framework. Note that potentials (13) generalize the so called robust truncated potentials such as $\min(|\mathcal{F}\mathbf{x}|, \kappa)$, which are known to preserve discontinuities very well and, due to this, are heavily used in vision (*e.g.*, in stereo matching or optical flow estimation).

$\mathcal{P}^n$ **Potts potential:** This potential was recently introduced into computer vision by [3] and has been successfully used for segmentation purposes. Given a label set $\mathcal{L} = \{l_j\}$, that potential is defined as follows (for a K-size clique):

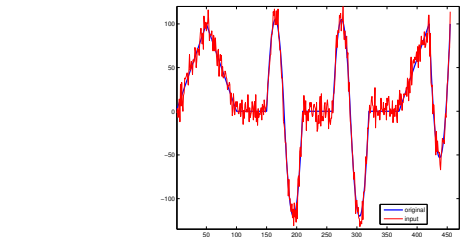
$$H(\mathbf{x}) = \begin{cases} \gamma_j & \text{if } x_1 = x_2 = \ldots = x_K = l_j \\ \gamma_{\max} & \text{otherwise} \end{cases}, \quad (14)$$

where $\gamma_{\max} \geq \gamma_j, \forall j$. Obviously, this is an instance of a pattern-based potential with $|\mathcal{L}|$ patterns of the form $(l, l, \ldots, l), \forall l \in \mathcal{L}$. Note that, in this case, the number of patterns is independent of the clique size. As a result, the message-passing algorithm of sec. 3 remains efficient even when applied to cliques of very large sizes.

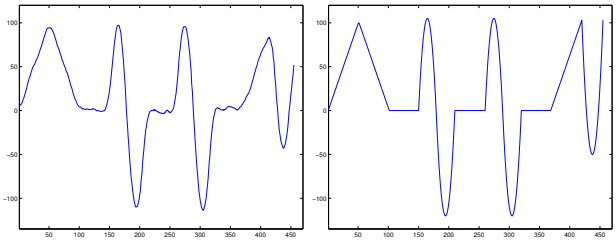
Exemplar-based priors/higher-order shape priors: *Exemplar-based priors* are known to be extremely useful for many vision tasks and can be encoded very easily by using pattern-based potentials. *E.g.*, in the context of class-specific binary segmentation, each pattern may correspond to a binary fragment (object-background mask) related to a frequently occuring object part. Hence, in this case, patterns essentially act as exemplars for frequently occuring segments at the clique level, and they are thus used for implicitly encoding higher-order shape information. Note that, as demonstrated by Levin and Weiss [9], a small set of such fragments is typically required, and they can be efficiently learnt via a training procedure. Instead of deriving a class-specific shape prior as above, one may also use patterns to derive a generic *higher-order shape prior*. For instance, one can associate patterns with a sparse set of generic binary fragments with smooth boundaries (*e.g.*, with small curvature). By favoring these patterns, one essentially encodes higher-order boundary terms such as curvature in the potential functions. This type of higher-order shape information can be very useful for improving the segmentation process.

4. Experimental results

For the remainder of this paper we will refer to the algorithm of sec. 3 as *PatB* (from Pattern-Based). We tested that algorithm, as well as the generic optimizer of sec. 2.1 (referred to as simply generic optimizer hereafter), on various problems. The first one was the problem of signal reconstruction, where we are given a corrupted version of a one-dimensional piecewise smooth signal $\mathbf{y}$ and we want to recover the original signal $\mathbf{x}$. In this case, the data term was set equal to the unary potential $U(x_p) = |y_p - x_p|$. To model piecewise smoothness, we chose to take into account the 3^{nd} order derivative $\mathcal{F}\mathbf{x}$ of signal $\mathbf{x}$, where $\mathcal{F} = [1\ -3\ \ 3\ \ -1]$. In order to do that in a robust way, we used a 4^{th}-order potential of the form (12), and, so, the resulting MRF had cliques of size 4. Fig. 4 shows an example of using this technique for reconstructing a piecewise quadratic signal. Notice how accurately the signal's structure is recovered by our algorithm. For comparison, we also show the result from the bilateral filter, which is a state-of-the art filtering technique. Note that, in this case, the PatB algorithm is guaranteed to compute the global optimum of the resulting high-order MRF.

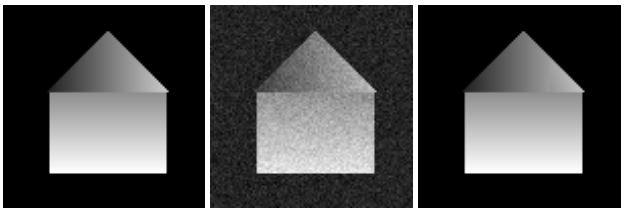


(a) corrupted signal (red) and original signal (blue)



(b) bilateral filter result (c) our result

Fig. 4: 1D signal reconstruction using a 4^{th} order MRF. The PatB algorithm computes the global optimum in this case.



(a) original (b) noisy input (c) our result

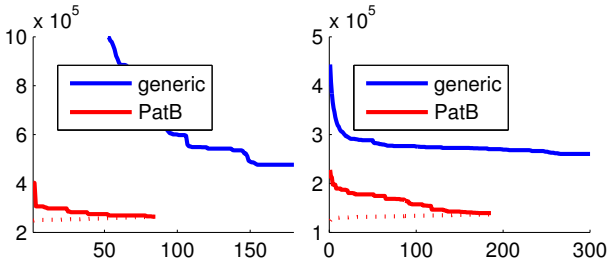
Fig. 5: Image denoising with a 3^{rd}-order MRF.

We next show results from applying our algorithm to image denoising. Here we are using a 3^{rd}-order potential of the form (12), while we use absolute differences for the unary potentials. Fig. 5 shows an image restoration result for a synthetic example of size 100×100. An almost perfect restoration is achieved in this case as the average error in intensity is only 0.3 (note that pairwise MRFs do very badly if applied to this example). Fig. 6 shows a corresponding result for a real image of size 128×96 that has been altered by Gaussian noise. For comparison we also show the result from using a pairwise MRF with a truncated linear function as potential (notice the characteristic piecewise constant

(a) original (b) noisy input

(c) pairwise MRF (d) 3^{rd}-order MRF

Fig. 6: Another image denoising example.

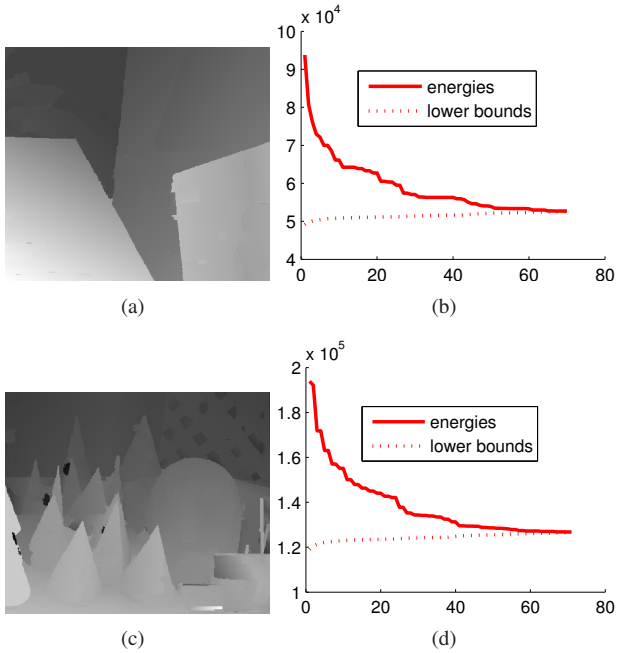


(a) primal-dual costs for the denoising result in Fig. 5 (b) primal-dual costs for the denoising result in Fig. 6

Fig. 7: Primal costs (solid lines) represent MRF energies, while dual costs (dashed lines) represent lower bounds on the optimum energy. Unlike the generic optimizer of sec. 2.1, PatB computes an almost optimal solution for the denoising examples since the sequences of energies and lower bounds converge.

patches that appear in the denoised image in this case). The plots in Fig. 7 display the sequences of primal and dual costs generated by the PatB algorithm when applied to the examples of Figs. 5, 6 (recall that each dual cost equals to the sum of minimum energies of the slave MRFs and always provides a lower bound to the optimum energy). Due to that the two sequences converge in both examples, this implies that the solutions computed by PatB are almost optimal. We also plotted the energies produced by the generic optimizer of sec. 2.1. As can be seen, it fails to perform as well as PatB (*i.e.*, produces solutions with higher energies), while it also converges much slower. Despite using heavily unoptimized code, the average running time per iteration of PatB was only 7 secs for the denoising examples (256 labels).

To further test the effectiveness of our framework, we also applied it to the problem of stereo matching. Given recent work which shows that a second order smoothness prior can provide a very good model for this problem,

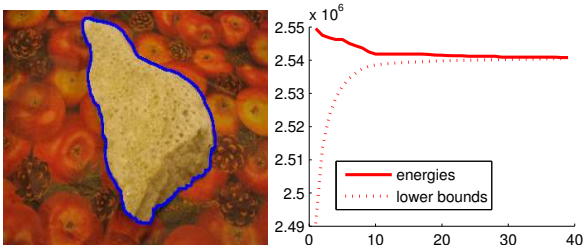


(a) (b)

(c) (d)

Fig. 8: Stereo matching results for 'venus' and 'teddy' using 15 proposal depth maps. These are concurrently fused by PatB at a single step, and this is done in an almost optimal manner since the corresponding energies and lower bounds converge to each other.

we used a higher-order potential of the form (13), which is essentially a generalization of the robust potential $H(x_1, x_2, x_3) = \min(|x_1 - 2x_2 + x_3|, \kappa)$ that approximates a 2^{nd} order derivative. The resulting MRF will thus have as higher-order cliques all horizontal 1×3 and vertical 3×1 patches in the image grid. Since we wanted to obtain disparity with subpixel accuracy, a set of proposal depth maps, generated similarly to [13], were fused for extracting pixel disparities. Unlike in [13], however, where many separate binary-fusion moves are applied for that purpose, here we can naturally handle the fusion in a much more integrated manner by considering all the depth maps *concurrently* in a single multilabel high-order MRF. For each node of this MRF we simply must choose a label from the corresponding pixels of the proposal depth maps instead from a standard label set. Put otherwise a *multi-fusion* move is performed at a single step (an operation that can, of course, be extremely useful for many other problems in vision as well). Fig. 8 displays some of the resulting disparity maps obtained with this method, where 15 proposal depth maps have been fused. Due to the use of higher order potentials, it is natural to expect that the resulting MRFs will be much harder to optimize. The plots in Fig. 8 show the corresponding sequences of energies and lower bounds generated by the PatB algorithm in this case. These plots confirm that, despite the high degree of difficulty of this problem, PatB has been able to fuse the depth maps almost optimally as the resulting solutions had



(a) (b)

Fig. 9: (a) A binary image segmentation result using a $\mathcal{P}^{3\times3}$ Potts model. **(b)** In this case PatB computes the global optimum of the corresponding 9^{th}-order MRF since the energies and the lower bounds finally become equal to each other.

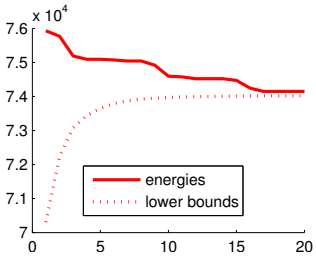


Fig. 10: A plot from applying PatB to a random high-order MRF based on a $\mathcal{P}^{3\times3}$ Potts model with 10 labels on a 50×50 grid. PatB computes the global optimum in this case as well.

energies extremely close to the optimum ones. The average running time per iteration was only 1.6 secs. Obviously the use of more depth map proposals can further improve the results, but our main goal is to demonstrate that PatB can perform the fusion in a provably almost optimal manner.

As a last application, we tested how well PatB can handle the $\mathcal{P}^n$ Potts potentials. We thus applied it to the task of interactive image segmentation, and used MRFs that had as cliques all 3×3 patches in the image grid (*i.e.*, the clique size was 9). The clique potential H_c for a patch c was defined such that it forms a $\mathcal{P}^{3\times3}$ model as follows:

$$H_c(\mathbf{x}) = \begin{cases} \text{SSD}_c(l) & \text{if } x_k = l \ \ \forall k \\ \lambda_0 \max_l \text{SSD}_c(l) & \text{otherwise} \ . \end{cases}$$

Here $\text{SSD}_c(l)$ represents the minimum sum of squared differences between the RGB values of patch c and all patches belonging to the l-th user-specified mask, while we set $\lambda_0 = 1.2$. Furthermore, all unary potentials were set equal to zero. Fig. 9(a) shows a binary segementation result for an image from the middlebury data set. The corresponding primal and dual costs are shown in Fig. 9(b), and prove that PatB manages to compute the exact global optimum in this case (this behavior was typical for other segmentation examples due to that the corresponding LP-relaxation was tight). To further examine how our algorithm handles the $\mathcal{P}^n$ Potts model, we also applied PatB to a series of NP-hard instances based on synthetic multilabel MRFs defined on a 50×50 grid. The values of their unary potentials were sampled independently from $100\mathcal{N}(0,1)$, while the parameters γ_j (in (14)) were sampled independently from $|100\mathcal{N}(0,1)|$ for each 3×3 clique potential (furthermore we set $\gamma_{\max} = \max_j \gamma_j$). Interestingly enough PatB has been able to compute the global optimum in all cases, and Fig. 10 shows a typical run. Despite this fact, the running time per iteration was very low. *E.g.*, for a 100×100 MRF with 3×3 cliques and 10 labels, the average running time per iteration was only 0.18 secs. Contrast this with the exponential time that would be required if we were to apply a generic message passing algorithm such as BP in this case.

5. Conclusions

We proposed a powerful master-slave based framework for high-order MRF optimization. It allows decomposing (in a principled manner) a difficult high-order MRF problem into a set of easy-to-handle optimization subtasks for which efficient customized inference techniques can be easily exploited. As a result, besides being extremely general, it is also very flexible since it is easily adaptable to take advantage of the special structure that may exist in high-order MRFs encountered in practice. We showed an example of this by introducing a very useful class of high-order potentials, called pattern-based potentials, along with novel and efficient message-passing algorithms. Experimental results on a variety of problems verified our framework's effectiveness and its ability to deliver high-quality solutions. Given that high-order models are of broad interest, we hope that our framework will further promote their applicability to vision tasks and will be of value to a large class of problems.

References

[1] supplemental material.
[2] H. Ishikawa. Higher-order clique reduction in binary graph cut. In *CVPR*, 2009.
[3] P. Kohli, P. Kumar, and P. Torr. P3 and beyond: Solving energies with higher order cliques. In *CVPR*, 2007.
[4] P. Kohli, L. Ladicky, and P. Torr. Robust higher order potentials for enforcing label consistency. In *CVPR*, 2008.
[5] M. Wainwright, T. Jaakkola, and A. Willsky. Map estimation via agreement on trees: message-passing and linear programming. *IEEE Trans. on Information Theory*, 2005.
[6] V. Kolmogorov. Convergent tree-reweighted message passing for energy minimization. *PAMI*, 2006.
[7] N. Komodakis, N. Paragios, and G. Tziritas. MRF optimization via dual decomposition: Message-passing revisited. In *ICCV*, 2007.
[8] X. Lan, S. Roth, D. P. Huttenlocher, and M. J. Black. Efficient belief propagation with learned higher-order markov random fields. In *ECCV*, 2006.
[9] A. Levin and Y. Weiss. Learning to combine bottom-up and top-down segmentation. In *ECCV*, 2006.
[10] B. Potetz. Efficient belief propagation for vision using linear constraint nodes. In *CVPR*, 2007.
[11] C. Rother, P. Kohli, W. Feng, and J. Jia. Minimizing sparse higher order energy functions of discrete variables. In *CVPR*, 2009.
[12] T. Werner. High-arity interactions, polyhedral relaxations, and cutting plane algorithm for soft constraint optimization (MAP-MRF). In *CVPR*, 2008.
[13] O. J. Woodford, P. H. S. Torr, I. D. Reid, and A. W. Fitzgibbon. Global stereo reconstruction under second order smoothness priors. In *CVPR*, 2008.