A Primal-Dual Algorithm for Higher-Order Multilabel Markov Random Fields

Alex Fix Chen Wang Ramin Zabih
Cornell University
{afix,chenwang,rdz}@cs.cornell.edu

Abstract

Graph cuts method such as α-expansion [4] and fusion moves [22] have been successful at solving many optimization problems in computer vision. Higher-order Markov Random Fields (MRF's), which are important for numerous applications, have proven to be very difficult, especially for multilabel MRF's (i.e. more than 2 labels). In this paper we propose a new primal-dual energy minimization method for arbitrary higher-order multilabel MRF's. Primal-dual methods provide guaranteed approximation bounds, and can exploit information in the dual variables to improve their efficiency. Our algorithm generalizes the PD3 [19] technique for first-order MRFs, and relies on a variant of max-flow that can exactly optimize certain higher-order binary MRF's [14]. We provide approximation bounds similar to PD3 [19], and the method is fast in practice. It can optimize non-submodular MRF's, and additionally can incorporate problem-specific knowledge in the form of fusion proposals. We compare experimentally against the existing approaches that can efficiently handle these difficult energy functions [6, 10, 11]. For higher-order denoising and stereo MRF's, we produce lower energy while running significantly faster.

1. Higher-order MRFs

There is widespread interest in higher-order MRF's for problems like denoising [23]and stereo [30], yet the resulting energy functions have proven to be very difficult to minimize. The optimization problem for a higher-order MRF is defined over a hypergraph with vertices $\mathcal{V}$ and cliques $\mathcal{C}$ plus a label set $\mathcal{L}$. We minimize the cost of the labeling $f : \mathcal{L}^{|\mathcal{V}|} \to \Re$ defined by

$$f(x) = \sum_i f_i(x_i) + \sum_{C \in \mathcal{C}} f_C(x_C). \qquad (1)$$

f_C is a function of just the variables x_i with $i \in C$ (a subvector of x which we denote x_C). We assume, without loss of generality, that $f_i, f_C \geq 0$ (we can just add a constant to make this hold). Special cases include first-order MRFs where $|C| = 2$, and binary MRFs where $|\mathcal{L}| = 2$; we are interested in the general case of higher-order multilabel MRF's, where we restrict neither $|C|$ nor $|\mathcal{L}|$.

Graph cut methods are a popular approach to first-order MRF's, and produce strong performance on standard benchmarks such as [26]. The most widely used graph cut techniques, including α-expansion [4] and its generalization to fusion moves [22], repeatedly solve a first-order binary MRF in order to minimize the original multilabel energy function. If the first-order binary MRF satisfies a condition known as submodularity (sometimes called regularity [16]), then expansion move produces a solution that is guaranteed to be within a known factor of the global minimum [4].

The connection between graph cuts and primal-dual techniques was established by [19] who showed that α-expansion could be interpreted as simultaneously optimizing primal and dual solutions. [19] proposed several primal-dual algorithms that generalized α-expansion and provided both theoretical and practical advantages. These methods apply to much more general energy functions and extend the approximation bounds of [12]. Empirically, keeping track of the dual variables allows a number of implementation speedups compared to α-expansion, resulting in the very efficient algorithm FastPD [21].

1.1. Summary of our method

In this paper, we provide an generalization of the primal-dual algorithm PD3 of [19] that can efficiently minimize an arbitrary higher-order multilabel energy function. Briefly: PD3 relies on max-flow; the flow values update the dual variables and the min-cut updates the primal variables. Our method instead uses a variant of max-flow proposed by [14], who demonstrates that it can exactly minimize an interesting class of higher-order binary MRF's.

Primal-dual methods rely on a condition called complementary slackness. Our algorithm begins with a common linear programming relaxation to (1). This linear program has two kinds of constraints, corresponding to the unary terms and clique-based terms in equation (1). We refer to the respective complementary slackness conditions as unary and clique slackness conditions. We will keep track of a primal solution x and (not necessarily feasible) dual solution λ. We will ensure that x, λ always satisfy the clique slackness conditions, and at each step of the algorithm, we will try to move x to be closer to satisfying unary slackness. The algorithm converges to a solution where both slackness conditions hold, but we generally lose feasibility. However, there exists some ρ such that λ/ρ is dual-feasible. This gives us a ρ-approximation algorithm for a class of functions we call weakly associative.

We review the related work in Section 2. Our algorithm is presented in Section 3, and we conclude with an experimental evaluation in Section 4.

2. Related Work

While our focus is on graph cut methods, which have been very successful for first-order MRF's [26], there are also several interesting algorithms for higher-order MRF's that do not use graph cuts. [29] took the LP relaxation of the energy function and generalized the max-sum diffusion algorithm for high-order MRF's. [17] extended the dual decomposition scheme, which is reviewed in [18]. [27] proposed a message passing algorithm for two important classes of higher order energy functions. Other message-passing schemes include the Generalized TRW-S algorithm of [15], which also optimizes a dual linear program, using max-product message passing on chains of cliques.

2.1. Graph Cut Methods and Higher-Order MRF's

The most popular graph cut methods for multilabel first-order MRF's rely on move-making techniques. Those methods, which notably include α-expansion [4] and fusion moves [22], reduce the multilabel problem to a series of binary subproblems which are then solved by max-flow [3, 16]. In α-expansion [4], the binary problem involves each pixel deciding whether to keep its current label or adopt a particular new label α. The expansion move algorithm also provides a guaranteed approximation bound.

[19, 20] proposed a primal-dual framework that generalizes α-expansion. They interpreted this algorithm as optimizing the primal and dual problem of the LP-relaxation of the MRF energy function simultaneously. In addition, the general primal-dual algorithm overcomes the most important limitation of the α-expansion algorithm, which is the requirement that the pairwise energy must be a metric [4]. The same approximation ratio still holds for a much broader class of energy functions. Furthermore, by tracking the dual variables to speedup the optimization, it can be 3-9 times faster in practice [21].

Graph cut methods for higher order priors [6, 8, 10, 11, 24] take a reduction-based approach. When a move-making algorithm, such as α-expansion or fusion moves, is used to solve a higher-order MRF, the optimal move is computed by solving a higher-order binary MRF. [10] and [6] propose methods for turning an arbitrary binary higher-order MRF into a binary first-order MRF.

Recall that a function f defined on subsets of a base set is submodular if for all subsets S, T we have $f(S \cup T) + f(S \cap T) \leq f(S) + f(T)$. If the binary first-order MRF is submodular it can be solved exactly via graph cuts [3, 16]; otherwise the solution can be approximated [3]. Certain higher-order binary MRF's have efficient reductions [13, 24]. There are also methods that find the best reduction within a particular class of reductions [8, 11].

2.2. Linear programming and duality for MRFs

Much of the theory of MRF optimization algorithms revolves around a specific linear programming relaxation of (1) known as the Local Marginal Polytope formulation [25], which was extended to higher-order MRFs in [28]. Every linear program (LP) has a corresponding dual, and the dual program has resulted in efficient algorithms such as [18, 19, 21]. We will omit the primal program, and proceed straight to the dual program, which is a higher-order version of the dual program from [19].

The dual program has variables for each clique C, $i \in C$ and label x_i, denoted $\lambda_{C,i}(x_i)$; and is given by

$$\max_{\lambda} \quad \sum_i \min_{x_i} h_i(x_i) \tag{2a}$$

$$h_i(x_i) = f_i(x_i) + \sum_C \lambda_{C,i}(x_i) \quad \forall i \tag{2b}$$

$$\sum_{i \in C} \lambda_{C,i}(x_i) \leq f_C(x_C) \quad \forall C, x_C \tag{2c}$$

We can informally think of the dual variable $\lambda_{C,i}(x_i)$ as taking part of the cost $f_C(x_C)$, and redistributing it to the unary terms. Following [19], the functions $h_i(x_i)$ will be called the "height" of label x_i at variable i, and semantically can be thought of as the original cost $f_i(x_i)$, plus any redistribution $\lambda_{C,i}$ from the cliques to the unary terms at i. The dual is always a lower bound on the value $f(\overline{x})$ of any labeling. Throughout, when there is a distinguished labeling (such as the current labeling in α-expansion) we'll denote it $\overline{x}$, while generic labels will simply be x.

2.3. Sum-of-submodular (SoS) flow

[14] introduced SoS minimization via SoS flow, and described an SoS flow algorithm. Alternative SoS flow algorithm were described by [1, 7], with the method of [7] specifically designed for typical computer vision problems.

The sum-of-submodular flow problem of [14] is similar to max flow. We have a set of vertices $\mathcal{V}$ plus the source s and sink t, and arcs (s, i) and (i, t) for each $i \in \mathcal{V}$. We are also given a **sum-of-submodular function**:

$$g(S) = \sum_{C \in \mathcal{C}} g_C(S \cap C) + \sum_{i \in S} c_{i,t} + \sum_{i \notin S} c_{s,i} \tag{3}$$

Algorithm 1 Our SoSPD algorithm.

```
Initialize $\overline{x}$ arbitrarily.
Initialize $\lambda_{C,i}(\overline{x}_i) = \frac{1}{|C|} f_C(\overline{x}_C)$, and $\lambda_{C,i}(x_i) = 0$ for $x_i \neq \overline{x}_i$.
while unary slackness condititions are not satisfied do
    $\overline{y} \leftarrow$ result of proposal generator
    PRE-EDIT-DUALS($\overline{x}, \overline{y}, \lambda$)
    $\overline{x}', \lambda' \leftarrow$ UPDATE-DUALS-PRIMALS($\overline{x}, \overline{y}, \lambda$)
    POST-EDIT-DUALS($\overline{x}', \lambda'$)
end while
return $\overline{x}$
```

where $C \in \mathcal{C}$ are called cliques in $\mathcal{V}$, and each g_C is a *submodular function*, called a clique function, with

$$g_C(\emptyset) = g_C(C) = \min_S g_C(S \cap C) = 0. \tag{4}$$

Intuitively, the difference between max flow and sum-of-submodular flow is that in addition to capacity and conservation constraints, we will also require that the flow out of any set S is at most $g_C(S \cap C)$. To be precise, a sum-of-submodular flow has flow values $\phi_{s,i}$ and $\phi_{i,t}$ on the source and sink edges, as well as flow values $\phi_{C,i}$ for each clique C and $i \in C$. Then, a maximum sum-of-submodular flow is a solution to the following LP:

$$\max_\phi \quad \sum_i \phi_{s,i} \tag{5a}$$
$$\text{s.t.} \quad \phi_{s,i} \leq c_{s,i}, \quad \phi_{i,t} \leq c_{i,t} \qquad \forall i \tag{5b}$$
$$\phi_{s,i} - \phi_{i,t} - \sum_{C \ni i} \phi_{C,i} = 0 \quad \forall i \tag{5c}$$
$$\sum_{i \in S} \phi_{C,i} \leq g_C(S) \qquad \forall C, S \subseteq C \tag{5d}$$

Here, (5b) are the capacity constraints for source and sink edges, with capacities given by the unary terms $c_{s,i}, c_{i,t}$, (5c) are the flow-conservation constraints at i and (5d) are the additional constraints that the ϕ_C in a set S are at most $g_C(S)$. [14] shows that this LP can be solved by a generalized flow algorithm.

Finally, [14] provides a sum-of-submodular version of the min-cut max-flow theorem. If ϕ maximizes (5), and S minimizes (3), then the objective value (5a) of ϕ is equal to $g(S)$. Furthermore, the notion of saturated edges extends to the clique function: (1) if $i \in S$ then $\phi_{i,t} = c_{i,t}$ (2) if $i \notin S$ then $\phi_{s,i} = c_{s,i}$, and most importantly (3) for every clique C, $g_C(S \cap C) = \sum_{i \in S} \phi_{C,i}$.

3. The SoS Primal Dual Algorithm

Our algorithm, which we will call SoSPD, is designed around ensuring that two main conditions are satisfied regarding the primal and dual solutions. These conditions give us our approximation bound, as well as help design the rest of the algorithm. The conditions are complementary slackness conditions, in which the inequalities in the dual that correspond to a particular primal solution are actually satisfied with equality.

Definition 1. *Given a labeling $\overline{x}$ and dual solution λ, we say that $\overline{x}, \lambda$ satisfy the* **clique slackness** *conditions if the constraints in (2c) corresponding to $\overline{x}_C$ are satisfied with equality. That is, we have*

$$\sum_{i \in C} \lambda_{C,i}(\overline{x}_i) = f_C(\overline{x}_C) \qquad \forall C \tag{6}$$

Proposition 2. *If $\overline{x}, \lambda$ satisfy the clique slackness conditions, then $f(\overline{x}) = \sum_i h_i(\overline{x}_i)$.*

Proof. Remembering our redistribution argument, this means we have exactly partitioned $f_C(\overline{x}_C)$ among the λ, so the sum of the heights is the original cost $f(\overline{x})$. I.e.,

$$\sum_i h_i(\overline{x}_i) = \Big(\sum_i f_i(\overline{x}_i)\Big) + \Big(\sum_C \sum_i \lambda_{C,i}(\overline{x}_i)\Big)$$
$$= \sum_i f_i(\overline{x}_i) + \sum_C f_C(\overline{x}_C) = f(\overline{x}) \qquad \square$$

Definition 3. *$\overline{x}, \lambda$ satisfy the* **unary slackness** *conditions if for each i we have $h_i(\overline{x}_i) = \min_{x_i} h_i(x_i)$.*

Corollary 4. *If $\overline{x}, \lambda$ satisfy both the clique and unary slackness conditions, and λ is feasible, then $\overline{x}$ minimizes f.*

Proof. From Proposition 2, the sum of heights $\sum_i h_i(\overline{x}_i)$ is equal to $f(\overline{x})$, and by the definition of unary slackness, the sum of heights is also equal to the dual objective, the lower-bound on all possible values $f(x)$. $\square$

Since our original problem is NP-hard we can't expect both slackness conditions to hold for a feasible dual λ. We instead apply dual scaling [19], and allow our duals to become slightly infeasible. More specifically, the structure of (2) always allows us to scale down λ by $\frac{1}{\rho}$ for some $\rho \geq 1$ to get a feasible solution. This gives us approximate optimality.

Lemma 5. *If $\overline{x}$ and λ satisfy the unary and clique slackness conditions, and λ/ρ is dual feasible, then $f(\overline{x}) \leq \rho f(x^*)$, where x^* is the true optimum.*

Proof. Since $\overline{x}, \lambda$ satisfy both slackness conditions, we know that $f(\overline{x}) = \sum_i \min_{x_i} h_i(x_i)$, hence

$$f(\overline{x}) = \rho \sum_i \min_{x_i} \frac{1}{\rho}\Big[f_i(x_i) + \sum_C \lambda_{C,i}(x_i)\Big]$$
$$\leq \rho \sum_i \min_{x_i} \Big[f_i(x_i) + \sum_C \frac{1}{\rho}\lambda_{C,i}(x_i)\Big] \leq \rho f(x^*)$$

where the first inequality is because $f_i \geq 0$, and the second from λ/ρ being dual-feasible. $\square$

Lemma 5 gives the basic motivation behind our algorithm. Between iterations, $\overline{x}, \lambda$ will always satisfy the clique slackness conditions, and the goal of each iteration is to change $\overline{x}$ to move to lower height labels. At the end of the algorithm, all the $\overline{x}_i$ will be the lowest height labels for each i, and the unary slackness conditions are satisfied. Then, we'll prove that there exists some ρ such that λ/ρ is dual-feasible, and hence we have a ρ-approximation algorithm.

The difficult step in this algorithm is that when we change the labeling $\overline{x}$ to decrease the height, we must still maintain the clique slackness conditions. We cannot simply set each $\overline{x}_i$ to the lowest height label, lest the clique slackness conditions cease to hold. Instead we simultaneously pick a set of labels to change, and adjust the dual variables such that the new clique slackness conditions are tight. For the higher order case, we can show that sum-of-submodular flow is exactly the tool we need to ensure the clique slackness conditions still hold when changing labels.

At a high-level, the algorithm works as follows. At each iteration, much like the α-expansion or fusion move algorithms, we have a current labeling $\overline{x}$ and a proposed labeling $\overline{y}$. We use sum-of-submodular flow to pick a set S of variables that switch labels, and the max-flow min-cut theorem for sum-of-submodular flow will ensure that the new variables $\overline{x}', \lambda'$ also satisfy the clique slackness conditions.

Our SoSPD technique is summarized in Algorithm 1, and each iteration has 3 subroutines. The main work of the algorithm occurs in UPDATE-DUALS-PRIMALS, which sets up the sum-of-submodular flow problem, and picks a set of variables to swap. We will describe this subroutine first, in Section 3.1, making some assumptions about $\overline{x}, \lambda$ which may not hold in general. Then, it is the job of the other two subroutines, PRE-EDIT-DUALS and POST-EDIT-DUALS (Sections 3.2 and 3.3) to make sure these assumptions do hold, and that therefore the algorithm functions correctly.

3.1. Update-Duals-Primals

To begin with, we need notation for fusion moves [22]. If we have current and proposed labelings $\overline{x}$ and $\overline{y}$, and S is the set of variables that change label, we'll denote the fused labeling by $\overline{x}' = \overline{x}[S \leftarrow \overline{y}]$, which has $\overline{x}'_i = \overline{y}_i$ if $i \in S$, and $\overline{x}'_i = \overline{x}_i$ if $i \notin S$.

Given our current state $\overline{x}, \lambda$, we're going to construct a sum-of-submodular flow network. The values $\phi_{C,i}$ will be the amount we add or subtract from $\lambda_{C,i}(\overline{y}_i)$, and the source-sink flow $\phi_{s,i}, \phi_{i,t}$ will give the change in height of $h_i(\overline{y}_i)$. We will only ever adjust the dual variables $\lambda_{C,i}(\overline{y}_i)$ corresponding to the proposed labeling $\overline{y}$.[1]

[1]Note that if $\overline{x}_i = \overline{y}_i$, we do not change $\lambda_{C,i}(\overline{x}_i)$. We could accomplish this by simply removing such i from the flow network. However such vertices i will have, by construction, no outgoing capacity in the network, so $\phi_{C,i}$ must always be 0.

The easy part is defining the source-sink capacities. If $h_i(\overline{y}_i) < h_i(\overline{x}_i)$ then we can raise the height of label $\overline{y}_i$ by the difference, and still prefer to switch labels. Similarly, if $h_i(\overline{y}_i) > h_i(\overline{x}_i)$, we can lower the height of $\overline{y}_i$ by the difference without creating a new label we'd prefer to swap to. We define source-sink capacities by $c_{s,i} = h_i(\overline{x}_i) - h_i(\overline{y}_i)$, $c_{i,t} = 0$ in the first case, and $c_{s,i} = 0$, $c_{i,t} = h_i(\overline{y}_i) - h_i(\overline{x}_i)$ in the second case.

In addition to decreasing the heights of the variables, our other main concern is making sure that the clique slackness conditions continue to hold. Consider an individual clique C for now, and let us examine what our labeling $\overline{x}'_C$ could look like after a fusion step. The possibile labelings are $\overline{x}_C[S \leftarrow \overline{y}_C]$ for each subset S of C. We want to make sure that after the swap, $\sum_i \lambda_{C,i}(\overline{x}'_i) = f_C(\overline{x}'_C)$, so define a function g_C equal to the difference:

$$g_C(S) := f_C(\overline{x}_C[S \leftarrow \overline{y}_C]) - \sum_{i \in S} \lambda_{C,i}(\overline{y}_i) - \sum_{i \notin S} \lambda_{C,i}(\overline{x}_i) \tag{7}$$

For now, we'll assume that (1) g_C is a submodular function and (2) $g_C(\emptyset) = g_C(C) = 0$, $g_C(S) \geq 0$. These assumptions will end up being enforced by PRE-EDIT-DUALS, which we describe below.

Under these assumptions the capacities c and functions g_C define a sum-of-submodular flow network, so we can find a flow ϕ and cut S such that $g_C(S \cap C) = \sum_{i \in S} \phi_{C,i}$ (by the sum-of-submodular version of the max-flow min-cut theorem [14], paraphrased at the end of Section 2.3). Then, we set $\overline{x}' = \overline{x}[S \leftarrow \overline{y}]$, and $\lambda'_{C,i}(\overline{y}_i) = \lambda_{C,i}(\overline{y}_i) + \phi_{C,i}$. By definition of g_C, we have

$$\begin{aligned} f_C(\overline{x}'_C) &= g_C(S \cap C) + \sum_{i \in S} \lambda_{C,i}(\overline{y}_i) + \sum_{i \notin S} \lambda_{C,i}(\overline{x}_i) \\ &= \sum_{i \in S} [\lambda_{C,i}(\overline{y}_i) + \phi_{C,i}] + \sum_{i \notin S} \lambda_{C,i}(\overline{x}_i) = \sum_i \lambda'_{C,i}(\overline{x}'_i). \end{aligned}$$

Therefore, the primal and dual solutions satisfy the clique slackness conditions, and our source-sink capacities were chosen so that $h'_i(\overline{x}'_i) \leq h_i(\overline{x}_i)$. Finally, unless every edge out of s gets saturated (and hence $S = \emptyset$) then at least one height has strictly decreased.

3.2. Pre-Edit-Duals

The job of PRE-EDIT-DUALS is to ensure that the assumptions we made in UPDATE-DUALS-PRIMALS are actually true. Namely, we need (1) the function g_C must be submodular and (2) $g_C(\emptyset) = g_C(C) = 0$ and $g_C(S) \geq 0$.

For (1), first note that if $f_C(\overline{x}_C[S \leftarrow \overline{y}_C])$ is submodular, then so is g_C, since a submodular function plus a linear function is still submodular. Such functions were called *expansion-submodular* in [7]. To handle general energy functions, we need an approach for the case where the fusion move is not submodular.

We take a similar approach to the PD3 variant PD3$_a$ [19], which finds an overestimate of the original energy function. For pairwise energies finding a submodular overestimate simply consists of truncating negative capacities to 0. In our case, we must find a submodular upper bound, $\widetilde{f}_C(S)$, such that $\widetilde{f}_C(S) \geq f_C(\overline{x}_C[S \leftarrow \overline{y}_C])$. Our only other requirements are that $\widetilde{f}_C(\emptyset) = f_C(\overline{x}_C)$, $\widetilde{f}_C(C) = f_C(\overline{y}_C)$, and that $\widetilde{f}(\{i\}) \leq \max_{x_C} f_C(x_C)$ for $i \in C$.

The problem of finding a submodular upper bound of a function is underspecified. In our experiments we use a simple heuristic, which we describe in the supplementary material. We leave as an open-question finding the "best" submodular upper-bound (as well as what the metric for "what is best" should even be).

Having computed $\widetilde{f}_C$, we then substitute it for f_C, just for this iteration. To simplify the notation we will write $\widetilde{f}_C(\overline{x}'_C)$ to mean $\widetilde{f}_C(S)$ wherever $\overline{x}'_C = \overline{x}_C[S \leftarrow \overline{y}_C]$.

To establish assumption (2), we make use of the following fact (which follows from the greedy algorithm of Edmonds [5], and is used for a similar purpose in [14]). For any submodular function g with $g(\emptyset) = 0$, there is a vector ψ such that $g(S) + \psi(S) \geq 0$ and $g(C) = \psi(C)$ (where we are using the standard notation $\psi(S) := \sum_{i \in S} \psi_i$). In fact, the vector defined by $\psi_i = g(\{1, \ldots, i-1\}) - g(\{1, \ldots, i\})$ will suffice.

To ensure (2) holds, we start with $g_C(S)$ defined as in (7). Note that we have $g_C(\emptyset) = f_C(\overline{x}_C) - \sum_{i \in C} \lambda_{C,i}(\overline{x}_i)$, which by the clique slackness condition, we know is 0. We can therefore compute a ψ as just described, and update $\lambda_{C,i}(\overline{y}_i) \leftarrow \lambda_{C,i}(\overline{y}_i) - \psi_i$. Since $g_C(S) + \psi(S) \geq 0$ and $g_C(C) + \psi(C) = 0$, when we update $g_C \leftarrow g_C + \psi$ with the new values of λ, we satisfy $g_C(S) \geq 0$ and $g_C(C) = 0$.

3.3. Post-Edit-Duals

Having run UPDATE-DUALS-PRIMALS, we know that $\widetilde{f}_C(\overline{x}'_C) = \sum_i \lambda'_{C,i}(\overline{x}'_i)$. However, from PRE-EDIT-DUALS, $\widetilde{f}$ might be an overestimate of f.

The subroutine POST-EDIT-DUALS enforces the clique slackness conditions, by setting $\lambda'_{C,i}(\overline{x}'_i) = \frac{1}{|C|} f_C(\overline{x}'_C)$ for each clique C. Note that if $\widetilde{f}$ is an overestimate, this can only ever decrease the sum of heights $h_i(\overline{x}'_i)$ (since we first average, and then subtract the overestimate from λ).

One final property of POST-EDIT-DUALS: since $f_C(\overline{x}'_C) \geq 0$, we always know that $\lambda_{C,i}(\overline{x}'_i) \geq 0$. We will use this in the proof of approximation ratio, momentarily.

3.4. Convergence and approximation bound

Much like the pairwise algorithms α-expansion and fusion move, we have monotonically decreasing energy.

Lemma 6. *The objective value $f(\overline{x})$ is non-increasing.*

Proof. First, recall that $\overline{x}, \lambda$ satisfy the clique slackness conditions, so $f(\overline{x}) = \sum_i h_i(\overline{x}_i)$. We also know that PRE-EDIT-DUALS doesn't change any of the heights $h_i(\overline{x}_i)$, UPDATE-PRIMALS-DUALS can only decrease $h_i(\overline{x}_i)$ (by definition of the source-sink capacities) and POST-EDIT-DUALS also doesn't increase the sum of heights. □

The convergence of our method is not guaranteed for arbitrarily bad fusion moves (for instance, we could have a bad proposal generator which always suggests labels which have greater height than $\overline{x}_i$). For α-expansion proposals, however, convergence is guaranteed.

Proposition 7. *With the proposal $\overline{y}_i = \alpha$ for each i, at the end of the iteration either $f(\overline{x}') < f(\overline{x})$ or $h_i(\overline{x}_i) \leq h_i(\alpha)$ for all i.*

Proof. From the discussion of UPDATE-DUALS-PRIMALS, one of two things happens: (1) the height of at least one variable is strictly decreased, or (2) the minimum cut is $S = \emptyset$. If (1), then neither of the other subroutines increases the sum of heights, so by Proposition 2 we have $f(\overline{x}') < f(\overline{x})$. If (2) then all edges out of s are saturated, so UPDATE-DUALS-PRIMALS increased $h_i(\alpha)$ to be at least $h_i(\overline{x}_i)$. Furthermore, $\overline{x}' = \overline{x}$ and so neither PRE-EDIT-DUALS nor POST-EDIT-DUALS changes any of the $\lambda_{C,i}(\overline{x}_i)$, and therefore $h_i(\overline{x}_i) \leq h_i(\alpha)$ holds at the end of the iteration. □

Lemma 8. *If after running through iterations of α-expansion for every label α, $f(\overline{x})$ does not strictly decrease, then the unary slackness conditions must hold, and the algorithm terminates.*

Proof. Since every α-expansion iteration didn't change the objective $f(\overline{x})$, by Proposition 7 each such iteration ensures that $h_i(\alpha) \geq h_i(\overline{x}_i)$. Also note that a β-expansion for $\beta \neq \alpha$ doesn't change any of the $h_i(\alpha)$. Therefore, the $\overline{x}_i$ are all minimum height labels, and the unary slackness conditions are satisfied. □

Overall, with integer costs, the objective decreases by at least 1 each outer-iteration and therefore eventually halts. The running time of each iteration is dominated by the SoS flow computation — we use SoS-IBFS [7] which has runtime $O(|\mathcal{V}|^2 |\mathcal{C}|\, 2^k)$, where $k = \max |C|$. It is difficult to provide a non-trivial bound on the number of α-expansion iterations, but in practice we always observe convergence after 4 passes through the label set. Note that this is exponential in the clique size, since we represent submodular functions as tables of 2^k values. However, this is also true of other state of the art methods for higher-order MRF's such as [6, 10, 17].

Let $f^{\max} = \max f_C(x_C)$, $f^{\min} = \min f_C(x_C)$, where the max and min are over all cliques C and all non-constant labelings x_C. There is a natural class of MRF's where $f^{\min} > 0$ (i.e. all non-constant labelings have positive costs), and where constant labelings have zero cost. We call such MRF's **weakly associative**; they encourage all variables in a clique to have the same label, but are otherwise unrestricted on non-constant labelings. This generalizes what [19] calls non-metric energies.

Our approximation ratio will be $\rho = k\frac{f^{\max}}{f^{\min}}$. Note that ρ is finite only for a weakly associative MRF. This generalizes the approximation ratio for PD3, which is $2\frac{f^{\max}}{f^{\min}}$.

Theorem 9. *SoSPD with α-expansion for a weakly associative MRF f is a ρ-approximation algorithm, i.e., the primal solution $\overline{x}$ at the end will have $f(\overline{x}) \le \rho f(x^*)$.*

Proof. The first task is to show that λ doesn't get too big. In particular, after any iteration, $\lambda_{C,i}(x_i) \le f^{\max}$ for all x_i. Note that after UPDATE-DUALS-PRIMALS, we have

$$\phi_{C,i} \le g_C(\{i\}) := \widetilde{f}_C(\{i\}) - \sum_{j\neq i} \lambda_{C,i}(\overline{x}_i) - \lambda_{C,i}(\overline{y}_i)$$

Since we constructed $\widetilde{f}$ to have $\widetilde{f}(\{i\}) \le f^{\max}$ and POST-EDIT-DUALS from the previous iteration makes sure $\lambda_{C,i}(\overline{x}_i) \ge 0$, we get $\lambda'_{C,i}(\overline{y}_i) = \lambda_{C,i}(\overline{y}_i) + \phi_{C,i} \le f_C^{\max}$. If POST-EDIT-DUALS in the present iteration changes $\lambda_{C,i}(\overline{y}_i)$, it sets it to $\frac{1}{|C|} f_C(\overline{x}') \le f_C^{\max}$. Therefore, $\lambda'_{C,i}(\overline{y}_i) \le f_C^{\max}$, and we don't change $\lambda_{C,i}(x_i)$ for any $x_i \neq \overline{y}_i$ in this iteration, so inductively, at the end of the algorithm $\lambda_{C,i}(x_i) \le f_C^{\max}$ for all labels x_i.

For feasibility, we need to show that (2c) holds for each clique C and labeling x_C. For non-constant x_C we have

$$\sum_i \frac{1}{\rho}\lambda_{C,i}(x_i) \le \frac{|C| f_C^{\max}}{\rho} \le f_C^{\min} \le f_C(x_C)$$

For constant labeling $x_C = \alpha$, note that in the last α-expansion, PRE-EDIT-DUALS enforces that $\widetilde{f}_C(C) - \sum_i \lambda_{C,i}(\alpha) = g_C(C) = 0$, and neither of the other subroutines violate this. Therefore $\sum_i \frac{1}{\rho}\lambda_{C,i}(\alpha) = \frac{1}{\rho}\widetilde{f}_C(C) = \frac{1}{\rho} f_C(\alpha) = 0$, where the second equality is because we constructed $\widetilde{f}_C$ with $\widetilde{f}_C(C) = \widetilde{f}(\overline{y}_C)$, and the last is since f is weakly associative.

Finally, Lemma 5 says that at convergence, $f(\overline{x})$ is no more than $\rho f(x^*)$. $\square$

4. Experimental Evaluation

To evaluate the experimental performance of our algorithm, we considered two different higher-order multilabel problems: stereo reconstruction, using the curvature-regularizing prior of [30], and Field of Experts denoising [23]. All data and experiments can be found at the author's website, www.cs.cornell.edu/~afix. The code is in C++, and is available under an open source license.

"Teddy"	Pixels within ± 1	Final energy	Time
FGBZ-Fusion	83.3%	9.320×10^9	468s
HOCR-Fusion	83.8%	9.298×10^9	210s
GRD-Fusion	84.9%	9.256×10^9	1116s
SoSPD-Fusion	84.8%	9.172×10^9	129s
"Cones"	Pixels within ± 1	Final energy	Time
FGBZ-Fusion	74.9%	1.1765×10^{10}	340s
HOCR-Fusion	74.2%	1.1789×10^{10}	172s
GRD-Fusion	75.2%	1.1690×10^{10}	1138s
SoSPD-Fusion	75.2%	1.1664×10^{10}	133s

Table 1. Numerical results for stereo reconstruction, for the two images in Figure 1.

	Energy @ 10s	Final energy	Time
FGBZ-Gradient	4.17×10^8	2.353×10^8	86s
HOCR-Gradient	4.35×10^8	2.368×10^8	78s
GRD-Gradient	6.72×10^8	2.348×10^8	776s
SoSPD-Gradient	2.87×10^8	2.347×10^8	42s

Table 2. Numerical results for denoising, averaged over the 100 images in the test set. For the second column, we stop both methods after 10 seconds, and compare energy values.

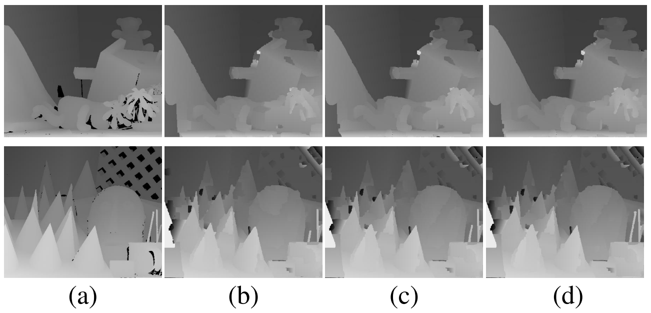

Figure 1. (a) Ground truth disparities, with results from (b) FGBZ-Fusion (c) SoSPD-Fusion and (d) SoSPD-Best-Fusion. Top row is the "teddy" image, bottom row is "cones". Results for SoSPD have slightly more correct pixels, and converged much faster — see Table 1 for details.

For experimental comparisons, the method of [17] does not currently have publicly available code, so we are left with the class of fusion-reduction methods [6, 10, 11]. While the Generalized Roof Duality method of [11] can produce good solutions, it is typically much slower than [6, 10], and is restricted to cliques of size at most 4. We observed that it obtains similar or slightly-worse energy values to SoSPD, while taking at least 10x more time, even for the heuristic version of GRD. We therefore focus on FGBZ [6] and HOCR [10] due to their speed and generality. Additional experimental data is provided in the supplementary material.

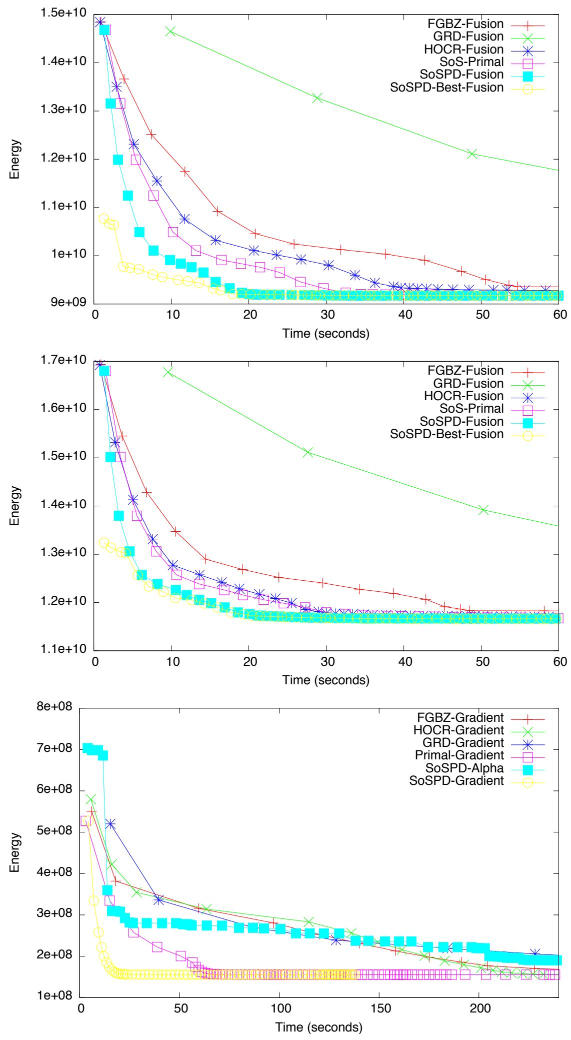

Figure 2. Energy reduction over time for the stereo images (top) "teddy" (center) "cones". (bottom) Energy reduction over time for the denoising image "penguin". Note that, in addition to converging faster, for a fixed time budget we achieve much better energy than the baseline.

4.1. Stereo reconstruction

The stereo reconstruction algorithm of [30] encourages the disparity map to be piecewise smooth using a 2nd order prior, composed of all 1×3 and 3×1 patches in the image, which each penalize a robust function of the curvature.

A number of optimization methods are proposed in [30], which are composed to get the final result. The most important step consists of pre-generating a set of 14 piecewise-planar proposed disparity maps, and then using these as proposals to the fusion move algorithm to improve the current disparity until convergence. This is called SEGPLN in [30].

We solve this using SoSPD by setting up a multilabel energy, where we have 14 labels, one for each pre-generated

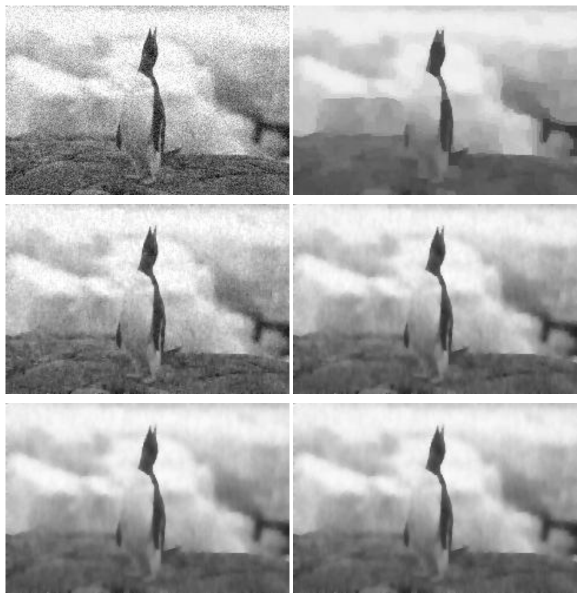

Figure 3. Denoising results. (top left) noisy image (top right) SoSPD-α (center left) FGBZ-Gradient, 10 sec (center right) SoSPD-Gradient, 10 sec (bottom left) FGBZ-Gradient at convergence (bottom right) SoSPD-Gradient at convergence.

proposal. Note that this allows estimating sub-pixel disparities (since the proposals can take any floating point value) using only 14 labels. We omit the binary visibility variables and edges from the original paper [30] to get a simpler problem, mainly so that all variables have the same number of labels (though in principle this could be handled by SoSPD). Note that this is the same experimental setup as [17]. Data was obtained by running the code[2] for [30] and recording the proposed fusion moves and corresponding unary terms.

We have two variants of SoSPD for this experiment, which only differ in the choice of proposed moves. The first, SoSPD-Fusion, rotates through the 14 labels, and successively chooses each to be an α-expansion proposal for that iteration. The second, SoSPD-Best-Fusion, uses an idea from [2] to pick the best α for each iteration. More specifically, we choose the α which will have the greatest total capacity leaving the source, in order to encourage as many nodes to switch to lower height labels as possible. We compared with the baselines, FGBZ-Fusion using the reduction [6], and HOCR-Fusion using the reduction [10]. Both methods cycle through the pre-generated proposals and perform fusion move.

Numerical results are in Table 1 and images in Figure 1. Overall, the SoSPD variants and reduction methods reach similar energy and visual results; however, SoSPD is fastest overall (2.5x-3.5x vs FGBZ, 1.3x-1.5x vs HOCR).

[2]http://www.robots.ox.ac.uk/~ojw/software.htm.

4.2. Field of Experts denoising

Field of Experts (FoE) for image denoising has been used as a benchmark in several higher-order optimization papers [10, 6, 11]. FoE is a patch-based natural image prior, with cliques for each 2×2 patch in the image.

SoSPD with α-expansion decreases the energy quickly initially, but gets stuck in poor local optima, with flat images as seen in Figure 3. Fortunately, gradient descent proposals [9] have been shown to be very effective at optimizing FoE priors. We call the combination of SoSPD with these fusion proposals SoSPD-Gradient.

We compare against fusion move with the same proposals, and the reductions of [6] and [10]. Overall, when comparing SoSPD vs. [6] for the same proposal method, SoSPD is significantly faster, and achieves slightly lower energy at convergence. Additionally, given a fixed time budget of 10 seconds, both the energy and visual results of SoSPD are significantly better, as seen in Figure 3 and Table 2.

Acknowledgments: This research has been funded by NSF grant IIS-1161282.

References

[1] C. Arora, S. Banerjee, P. Kalra, and S. N. Maheshwari. Generic cuts: an efficient algorithm for optimal inference in higher order MRF-MAP. In *ECCV*, 2012. 2

[2] D. Batra and P. Kohli. Making the right moves: Guiding alpha-expansion using local primal-dual gaps. In *CVPR*, pages 1865–1872, 2011. 7

[3] E. Boros and P. L. Hammer. Pseudo-boolean optimization. *Discrete Applied Mathematics*, 123(1-3), 2002. 2

[4] Y. Boykov, O. Veksler, and R. Zabih. Fast approximate energy minimization via graph cuts. *TPAMI*, 23(11):1222–1239, 2001. 1, 2

[5] J. Edmonds. Submodular functions, matroids, and certain polyhedra. In M. Jnger, G. Reinelt, and G. Rinaldi, editors, *Combinatorial Optimization Eureka, You Shrink!*, volume 2570 of *Lecture Notes in Computer Science*, pages 11–26. Springer Berlin Heidelberg, 2003. 5

[6] A. Fix, A. Gruber, E. Boros, and R. Zabih. A graph cut algorithm for higher-order Markov Random Fields. In *ICCV*, 2011. 1, 2, 5, 6, 7, 8

[7] A. Fix, T. Joachims, S. Park, and R. Zabih. Structured learning of sum-of-submodular higher order energy functions. *arXiv*, 1309.7512, Sept. 2013. 2, 4, 5

[8] A. C. Gallagher, D. Batra, and D. Parikh. Inference for order reduction in Markov Random Fields. In *CVPR*, pages 1857–1864, 2011. 2

[9] H. Ishikawa. Higher-order gradient descent by fusion move graph cut. In *ICCV*, pages 568–574, 2009. 8

[10] H. Ishikawa. Transformation of general binary MRF minimization to the first order case. *TPAMI*, 33(6), 2010. 1, 2, 5, 6, 7, 8

[11] F. Kahl and P. Strandmark. Generalized roof duality for pseudo-boolean optimization. In *ICCV*, pages 255–262, 2011. 1, 2, 6, 8

[12] J. Kleinberg and E. Tardos. Approximation algorithms for classification problems with pairwise relationships: metric labeling and Markov Random Fields. *J. ACM*, 49(5):616–639, 2002. 1

[13] P. Kohli, M. P. Kumar, and P. H. Torr. P3 and beyond: Move making algorithms for solving higher order functions. *TPAMI*, 31(9):1645–1656, 2008. 2

[14] V. Kolmogorov. Minimizing a sum of submodular functions. *Discrete Appl. Math.*, 160(15):2246–2258, Oct. 2012. 1, 2, 3, 4, 5

[15] V. Kolmogorov and T. Schoenemann. Generalized sequential tree-reweighted message passing. *arXiv*, 1205.6352, Dec. 2012. 2

[16] V. Kolmogorov and R. Zabih. What energy functions can be minimized via graph cuts? *TPAMI*, 26(2):147–59, 2004. 1, 2

[17] N. Komodakis and N. Paragios. Beyond pairwise energies: Efficient optimization for higher-order MRFs. In *CVPR*, pages 2985–2992, 2009. 2, 5, 6, 7

[18] N. Komodakis, N. Paragios, and G. Tziritas. MRF energy minimization and beyond via dual decomposition. *TPAMI*, 33(3):531–552, 2011. 2

[19] N. Komodakis and G. Tziritas. A new framework for approximate labeling via graph cuts. In *ICCV*, 2005. 1, 2, 3, 5, 6

[20] N. Komodakis and G. Tziritas. Approximate labeling via graph cuts based on linear programming. *TPAMI*, 29(8):1436–1453, 2007. 2

[21] N. Komodakis, G. Tziritas, and N. Paragios. Fast primal-dual strategies for MRF optimization. Technical Report 0605, Ecole Centrale de Paris, 2006. 1, 2

[22] V. Lempitsky, C. Rother, S. Roth, and A. Blake. Fusion moves for Markov Random Field optimization. *TPAMI*, 32(8):1392–1405, Aug 2010. 1, 2, 4

[23] S. Roth and M. Black. Fields of experts. *IJCV*, 82:205–229, 2009. 1, 6

[24] C. Rother, P. Kohli, W. Feng, and J. Jia. Minimizing sparse higher order energy functions of discrete variables. In *CVPR*, pages 1382–1389, 2009. 2

[25] M. Shlezinger. Syntactic analysis of two-dimensional visual signals in the presence of noise. *Cybernetics*, 12(4):612–628, 1976. 2

[26] R. Szeliski, R. Zabih, D. Scharstein, O. Veksler, V. Kolmogorov, A. Agarwala, M. Tappen, and C. Rother. A comparative study of energy minimization methods for Markov Random Fields. *TPAMI*, 30(6):1068–1080, 2008. 1, 2

[27] D. Tarlow, I. E. Givoni, and R. S. Zemel. HOP-MAP: Efficient message passing with high order potentials. In *AISTATS*, pages 812–819, 2010. 2

[28] Y. Weiss, C. Yanover, and T. Meltzer. Map estimation, linear programming and belief propagation with convex free energies. In *UAI*, 2007. 2

[29] T. Werner. High-arity interactions, polyhedral relaxations, and cutting plane algorithm for soft constraint optimisation (map-mrf). In *CVPR*, 2008. 2

[30] O. Woodford, P. Torr, I. Reid, and A. Fitzgibbon. Global stereo reconstruction under second-order smoothness priors. *TPAMI*, 31:2115–2128, 2009. 1, 6, 7