Potts model, parametric maxflow and k-submodular functions

Igor Gridchyn
IST Austria
igor.gridchyn@ist.ac.at

Vladimir Kolmogorov
IST Austria
vnk@ist.ac.at

Abstract

The problem of minimizing the Potts energy function frequently occurs in computer vision applications. One way to tackle this NP-hard problem was proposed by Kovtun [20, 21]. It identifies a part of an optimal solution by running k maxflow computations, where k is the number of labels. The number of "labeled" pixels can be significant in some applications, e.g. 50-93% in our tests for stereo. We show how to reduce the runtime to $O(\log k)$ maxflow computations (or one parametric maxflow *computation). Furthermore, the output of our algorithm allows to speed-up the subsequent alpha expansion for the unlabeled part, or can be used as it is for time-critical applications.*

To derive our technique, we generalize the algorithm of Felzenszwalb et al. [7] for Tree Metrics. *We also show a connection to k-*submodular functions *from combinatorial optimization, and discuss k-*submodular relaxations *for general energy functions.*

1. Introduction

This paper addresses the problem of minimizing an energy function with *Potts* interaction terms. This energy has found a widespread usage in computer vision after the seminal work of Boykov et al. [4] who proposed an efficient approximation algorithm for this NP-hard problem called *alpha expansion*.

The algorithm of [4] is based on the maxflow algorithm, also known as *graph cuts*. Each iteration involves k maxflow computations, where k is the number of labels. Several techniques were proposed for improving the efficiency of these computations. The most relevant to us is the method of Kovtun [20, 21] which computes a part of an optimal solution via k maxflow computations. We can then fix "labeled" nodes and run the alpha expansion algorithm for the remaining nodes. Such scheme was a part of the "Reduce, Reuse, Recycle" approach of Alahari et al. [1].

Our main contribution is to improve the efficiency of Kovtun's method from k maxflow computations to $\lceil 1 + \log_2 k \rceil$ computations on graphs of equivalent sizes. In some applications the method labels a significant fraction of nodes [20, 21, 1], so our techique gives a substantial speed-up. We may get an improvement even when there are few labeled nodes: it is reported in [1] that using flow from Kovtun's computations always speeds up the alpha expansion algorithm for unlabeled nodes.

The idea of our approach is to cast the problem as another minimization problem with *Tree Metrics*, and then generalize the algorithm of Felzenszwalb et al. [7] for Tree Metrics by allowing more general unary terms. This generalization is our second contribution. Finally, we discuss some connections to *k-submodular functions*.

Other related work A theoretical analysis of Kovtun's approach was given by Shekhovtsov and Hlavac [25, 24]. It was shown that the method in [20, 21] does not improve on the alpha expansion in terms of the quality of the solution: if a node is labeled by Kovtun's approach then the alpha expansion would produce the same solution for this node upon convergence (assuming that all costs are unique; see [25] for a more general statement). Similarly, Kovtun's approach does not improve on the standard Schlesinger's LP relaxation of the energy [25].

We also mention the "FastPD" method of Komodakis et al. [18, 19]. The default version of FastPD for the Potts energy produces the same answer as the alpha expansion algorithm but faster, since it maintains not only primal variables (current solution) but also dual variables ("messages"). Intuitively, it allows to reuse flow between different maxflow computations. An alternative method for reusing flow was used by Alahari et al. [1], who reported similar speed-ups.

2. Preliminaries

The Potts energy for labeling $x \in \mathcal{L}^V$ is given by

$$f(x) = \sum_{i \in V} f_i(x_i) + \sum_{\{i,j\} \in E} \lambda_{ij} [x_i \neq x_j] \qquad (1)$$

Here V is the set of nodes, E is the set of edges, $\mathcal{L}$ is the set of labels, λ_{ij} are non-negative constants and $[\cdot]$ is the *Iverson bracket*. It is well-known that computing a minimizer of (1) is NP-hard when $|\mathcal{L}| \geq 3$ [4].

Let us review the method of Kovtun [20, 21] for obtaining a part of an optimal solution. (The method is applicable to general functions - see [20, 21, 25]; here we consider only the Potts energy, in which case the formulation simplifies considerably.) For a label $a \in \mathcal{L}$ denote $\bar{a} = \mathcal{L} - \{a\}$, and let $f_i(\bar{a}) = \min_{b \in \bar{a}} f_i(b)$. Define function $f^a : \{a, \bar{a}\}^V \to \mathbb{R}$ via

$$f^a(y) = \sum_{i \in V} f_i(y_i) + \sum_{\{i,j\} \in E} \lambda_{ij} [y_j \neq y_i] \qquad (2)$$

(A remark on notation: we typically use letter x for multi-valued labelings and y for binary labelings).

Theorem 1 ([20, 21])**.** *Let $y \in \{a, \bar{a}\}^V$ be a minimizer of f^a. For any $x \in \mathcal{L}^V$ there holds $f(x^y) \leq f(x)$ where labeling x^y is defined via* $(x^y)_i = \begin{cases} a & \text{if } y_i = a \\ x_i & \text{if } y_i = \bar{a} \end{cases}$ *for $i \in V$. Consequently, there exists minimizer $x^* \in \arg \min_{x \in \mathcal{L}^V} f(x)$ such that $x^*_i = a$ for all nodes $i \in V$ with $y_i = a$.*

Kovtun's approach requires minimizing function f^a for all $a \in \mathcal{L}$. A naive way to do this is to use k maxflow computations on a graph with $|V|$ nodes and $|E|$ edges, where $k = |\mathcal{L}|$. To reduce this to $O(\log k)$ maxflow computations, we will use the following strategy. First, we will define an auxiliary function $g : \mathcal{D}^V \to \mathbb{R}$ where $\mathcal{D} = \mathcal{L} \cup \{\mathbf{o}\}$, $\mathbf{o} \notin \mathcal{D}$. We will then present an efficient algorithm for minimizing g, and show that a minimizer $x \in \arg\min\{g(x) \mid x \in \mathcal{D}^V\}$ determines a minimizer $y \in \arg\min\{f^a(y) \mid y \in \{a, \bar{a}\}^V\}$ for each $a \in \mathcal{L}$ in the natural way, i.e. $y_i = a$ if $x_i = a$ and $y_i = \bar{a}$ otherwise. Function g will have the following form:

$$g(x) = \sum_{i \in V} g_i(x_i) + \sum_{\{i,j\} \in E} \lambda_{ij} d(x_i, x_j) \qquad (3)$$

where $d(\cdot, \cdot)$ is a *tree metric* with respect to a certain tree T:

Definition 2. *Let $T = (\mathcal{D}, \mathcal{E}, d)$ be a weighted undirected tree with positive weights $d(e)$, $e \in \mathcal{E}$. The* tree metric *on $\mathcal{D}$ is the function $d : \mathcal{D} \times \mathcal{D} \to \mathbb{R}$ defined as follows: $d(a, b)$ for $a, b \in \mathcal{D}$ is the length of the unique path from a to b in T, where $d(e)$ is treated as the length of edge $e \in \mathcal{E}$.*

We define T as the star graph rooted at $\mathbf{o}$, i.e. $\mathcal{E} = \{\{a, \mathbf{o}\} \mid a \in \mathcal{L}\}$. All edges are assigned length 1. The unary functions in (3) are set as follows: $g_i(\mathbf{o}) = 0$ and $g_i(a) = f_i(a) - f_i(\bar{a})$ for $a \in \mathcal{L}$. Function g in eq. (3) is now completely defined. It can be seen that minimizing f^a is equivalent to minimizing $g(y)$ over $y \in \{a, \mathbf{o}\}^V$.

The following observation will be crucial.

Proposition 3. *For any $i \in V$ and $a, b \in \mathcal{L}$ with $a \neq b$ there holds $g_i(a) + g_i(b) \geq 0$.*

Proof. Let $a_1 \in \arg\min_{a \in \mathcal{L}} f_i(a)$ and $a_2 \in \arg\min_{a \in \bar{a}_1} f_i(a)$. We have $g_i(a_1) = f_i(a_1) - f_i(a_2) \leq 0$ and $g_i(a) = f_i(a) - f_i(a_1) \geq f_i(a_2) - f_i(a_1) \geq 0$ for any $a \in \bar{a}_1$. This implies the claim. □

More generally, we say that function $g_i : \mathcal{D} \to \mathbb{R}$ is T-*convex* if for any pair of edges $\{a, b\}, \{b, c\} \in \mathcal{E}$ with $a \neq c$ there holds

$$d(a, c) g_i(b) \leq d(b, c) g_i(a) + d(a, b) g_i(c) \qquad (4)$$

Clearly, terms g_i contructed above are T-convex. We will prove the following result for an arbitrary tree T and function g with T-convex unary terms g_i. (Part (a) will imply that Kovtun's approach indeed reduces to the minimization of the function g above; part (b) will motivate a divide-and-conquer algorithm for minimizing g.)

Theorem 4. *Let $\{a, b\}$ be an edge in $\mathcal{E}$. For labeling $x \in \mathcal{D}^V$ define binary labeling $x^{[ab]} \in \{a, b\}^V$ as follows: $x^{[ab]}_i$ is the label in $\{a, b\}$ closest to x_i in T.*
(a) If $x \in \mathcal{D}^V$ is a minimizer of g then $x^{[ab]} \in \arg\min\{g(y) \mid y \in \{a, b\}^V\}$.
(b) If $y \in \arg\min\{g(y) \mid y \in \{a, b\}^V\}$ then function g has a minimizer $x \in \mathcal{D}^V$ such that $x^{[ab]} = y$.

Note, part (a) and a repeated application of Theorem 1 give that any minimizer $x \in \mathcal{D}^V$ of g is a partially optimal labeling for f, i.e. f has a minimizer $x^* \in \mathcal{L}^V$ such that $x^*_i = x_i$ for all i with $x_i \neq \mathbf{o}$.

In the next section we consider the case of an arbitrary tree T, and present an efficient algorithm for minimizing function g with T-convex unary terms. In section 4 we discuss its specialization to the star graph T with unit edge length, and sketch some implementation details. Then in section 5 we describe a connection to *k-submodular functions*. Section 6 gives experimental results, and section 7 presents conclusions.

3. Minimization algorithm for general T

We build on the work of Kolen [15] and Felzenszwalb et al. [7]. They considered the case when unary functions $g_i(\cdot)$ are given by $g_i(x_i) = \lambda_i d(x_i, c_i)$ where $\lambda_i \geq 0$ and c_i is a constant node in $\mathcal{D}$. [15] showed that such function can be minimized via $|\mathcal{D}|$ maximum flow computations on graphs with $O(|V|)$ nodes and $O(|E|)$ edges. Using a divide-and-conquer approach, [7] improved this to $O(\log |\mathcal{D}|)$ maxflow computations (plus $O(|\mathcal{D}| \log |\mathcal{D}|)$ time for bookkeeping). Their algorithm can be viewed as a generalization of the algorithm in [12, 5, 6] for minimizing *Total Variation* functionals $g(x) = \sum_i g_i(x_i) + \sum_{\{i,j\}} \lambda_{ij} |x_j - x_i|$ with convex terms g_i over $x \in \{1, 2, \ldots, k\}^V$ (this corresponds to the case when T is a chain with unit lengths).

In this section we show that with an appropriate modification the algorithm of [7] can be applied to function (3)

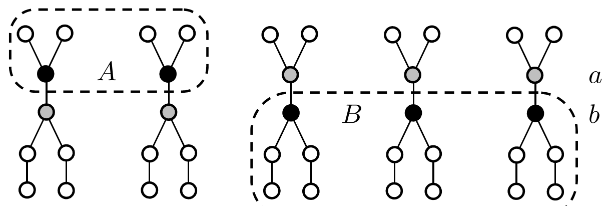

Figure 1. Algorithm's illustration. First, it computes $y \in \arg\min\{g(y) \mid y \in \{a,b\}^V\}$. Suppose that $y = (a,a,b,b,b)$. By Theorem 4(b), g has minimizer x that belongs to regions A and B. To find solution (x_1,x_2) for region A, the algorithm is called recursively while fixing variables x_3,x_4,x_5 to a (this is equivalent to fixing these variables to their optimal labels in B - the function changes only by a constant that does not depend on x_1,x_2). Solution (x_3,x_4,x_5) is computed similarly.

with T-convex unary terms, and present a self-contained proof of correctness.[1]

The main step of the algorithm is computing a minimizer $y \in \arg\min\{g(y) \mid y \in \{a,b\}^V\}$ for some edge $\{a,b\} \in \mathcal{E}$ (this can be done via a maxflow algorithm). By Theorem 4(b), y gives some information about a minimizer of g. This information allows to split the problem into two independent subproblems which can then be solved recursively. We arrive at the following algorithm.

Algorithm 1 SPLIT(g)
Input: function $g : \mathcal{D}^V \to \mathbb{R}$ specified by graph (V,E), tree $T = (\mathcal{D},\mathcal{E},d)$, unary terms $g_i : \mathcal{D} \to \mathbb{R}$ and edge weights λ_{ij}
Output: labeling $x \in \arg\min\{g(x) \mid x \in \mathcal{D}^V\}$

```
1:  if $\mathcal{D} = \{a\}$ return $(a,\ldots,a)$
2:  pick edge $\{a,b\} \in \mathcal{E}$
3:  compute $y \in \arg\min\{g(y) \mid y \in \{a,b\}^V\}$
4:  let $T_a = (\mathcal{D}_a,\mathcal{E}_a,d_a)$, $T_b = (\mathcal{D}_b,\mathcal{E}_b,d_b)$ be the trees
    obtained from $T$ by removing edge $\{a,b\}$ (with $a \in \mathcal{D}_a$, $b \in \mathcal{D}_b$)
5:  for $c \in \{a,b\}$ do
6:      let $V_c = \{i \in V \mid y_i = c\}$
7:      let $g^c$ be the function $\mathcal{D}_c^{V_c} \to \mathbb{R}$ obtained from $g$ by
        fixing all nodes in $V - V_c$ to $c$, i.e. $g^c(x) = g(\bar{x})$
        where $\bar{x}_i = x_i$ for $i \in V_c$ and $\bar{x}_i = c$ for $i \in V - V_c$
8:      let $x^c :=$ SPLIT($g^c$)
9:  end for
10: merge labelings $x^a$, $x^b$ into labeling $x$, return $x$
```

Note that function g^c in line 7 is defined on the subgraph of (V,E) induced by V_c. Indeed, for each edge $\{i,j\} \in E$ with $i \in V_c$, $j \in V - V_c$ pairwise term $\lambda_{ij} d(x_i,x_j)$ is transformed to a unary term $\lambda_{ij} d(x_i,c)$. It can be checked that this unary term is T_c-convex.

[1]The proof in [7] relied on results in [15], and used a different argument. In our view, the new proof shows more clearly why the extension to T-convex unary terms is possible.

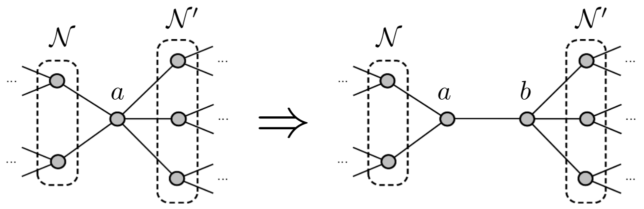

Figure 2. Inserting edge into T. Given node $a \in \mathcal{D}$ and the partition of its neighbors $\mathcal{N} \cup \mathcal{N}'$, tree T is modified as follows: (i) add new node $b \notin \mathcal{D}$; (ii) add new edge $\{a,b\}$; (iii) keep nodes $c \in \mathcal{N}$ as neighbors of a, but make nodes $c' \in \mathcal{N}'$ neighbors of b.

The following theorem implies that the algorithm is correct; its proof is given in section 3.1.

Theorem 5. *If x^c in line 9 is a minimizer of g^c over $\mathcal{D}_c^{V_c}$ for each $c \in \{a,b\}$ then labeling x in line 10 is a minimizer of g over $\mathcal{D}^V$.*

The algorithm leaves some freedom in line 2, namely the choice of edge $\{a,b\} \in \mathcal{E}$. Ideally, we would like to choose an edge that splits the tree into approximately equals parts ($|\mathcal{D}_a| \approx |\mathcal{D}_b|$). Unfortunately, this is not always possible; if, for example, T is a star graph then every split will be very unbalanced. To deal with this issue, [7] proposed to expand tree T (and modify the input function accordingly) so that the new tree T' admits a more balanced split. Details are given below.

Let a be a node in $\mathcal{D}$ with two or more neighbors. Let us split these neighbors into non-empty disjoint sets $\mathcal{N}, \mathcal{N}'$ and modify tree T as described in Fig. 2. (This step is inserted before line 2; the new edge $\{a,b\}$ becomes the output of line 2.) We denote $\mathcal{D}' = \mathcal{D} \cup \{b\}$; also, let $\mathcal{D}'_a$, $\mathcal{D}'_b$ be the connected components of T' after removing edge $\{a,b\}$ (with $a \in \mathcal{D}'_a$, $b \in \mathcal{D}'_b$).

The length of new edge $\{a,b\}$ is set to an infinitesimally small constant $\epsilon > 0$. The new unary function $g_i^\epsilon : \mathcal{D}' \to \mathbb{R}$ for node $i \in V$ is defined via

$$g_i^\epsilon(c) = g_i(c) + \epsilon \cdot u_i \cdot [c \in \mathcal{D}'_b] \qquad \forall c \in \mathcal{D}' \tag{5}$$

where we assume that $g_i(b) = g_i(a)$, and $u_i \in \mathbb{R}$ is chosen in such a way that function g_i^ϵ is T'-convex. (Such u_i always exists - see below). The new functional is thus $g^\epsilon(x) = \sum_{i\in V} g_i^\epsilon(x_i) + \sum_{\{i,j\}\in E} \lambda_{ij} d^\epsilon(x_i,x_j)$ for $x \in (\mathcal{D}')^V$, where d^ϵ is the new tree metric.

There holds $|g^\epsilon(x) - g(x^{b\mapsto a})| \le const \cdot \epsilon$ for any $x \in (\mathcal{D}')^V$, where $x^{b\mapsto a}$ is the labeling obtained from x by assigning label a to nodes with label b. Therefore, if ϵ is small enough then the following holds: if $x \in (\mathcal{D}')^V$ is an optimal solution of the modified problem then $x^{b\mapsto a}$ is an optimal solution of the original problem.

The cost function used in line 3 can be written as $g^{\epsilon}(y) = const + \epsilon \cdot g'(y)$ for all $y \in \{a,b\}^V$, where function $g' : \{a,b\}^V \to \mathbb{R}$ is defined via

$$g'(y) = \sum_{i \in V} u_i \cdot [y_i = b] + \sum_{\{i,j\} \in E} \lambda_{ij} \cdot [y_i \neq y_j] \qquad (6)$$

Therefore, minimizing g^{ϵ} over $\{a,b\}^V$ is equivalent to minimizing g' (and thus the minimizer does not depend on ϵ).

To summarize, we showed that the `SPLIT` algorithm remains correct if we replace line 2 with the tree modification step described above, and in line 3 compute $y \in \arg\min\{g'(y) \mid y \in \{a,b\}^V\}$. Also, in line 10 we need to convert labeling x^b to $(x^b)^{b \mapsto a}$ before merging with x^a.

Selecting u_i It remains to show that value u_i for node $i \in V$ can be set in such a way that function (5) is T'-convex.

Proposition 6. *Define*

$$u_i^{\min} = -\min_{c \in \mathcal{N}} \frac{g_i(c) - g_i(a)}{d(a,c)} \qquad u_i^{\max} = \min_{c' \in \mathcal{N}'} \frac{g_i(c') - g_i(a)}{d(a,c')}$$

There holds $u_i^{\min} \leq u_i^{\max}$, and for any $u_i \in [u_i^{\min}, u_i^{\max}]$ and $\epsilon > 0$ function g_i^{ϵ} in (5) is T'-convex.

A proof is given in [9].

3.1. Proof of theorems 4 and 5

The proof is based on the theorem below. Versions of this theorem in the case when T is a chain with unit weights appeared in [12, 28, 5, 6]; eq. (7) was then called the *coarea formula* [5, 6].

In part (b) we exploit the fact that unary functions g_i are T-convex, and make use of a well-known result about the *parametric maxflow* problem [8].

Theorem 7. **(a)** [Coarea formula] *There holds*

$$g(x) = const + \sum_{\{a,b\} \in \mathcal{E}} g(x^{[ab]}) \qquad \forall x \in \mathcal{D}^V \qquad (7)$$

where $x^{[ab]} \in \{a,b\}^V$ is defined as in Theorem 4.
(b) *Consider edges $\{a,b\}, \{b,c\} \in \mathcal{E}$ with $a \neq c$. Let y^{bc} be a minimizer of $\{g(y) \mid y \in \{b,c\}^V\}$. If y^{ab} is a minimizer of $\{g(y) \mid y \in \{a,b\}^V\}$ then so is labeling $y^{ab} \downarrow y^{bc} \in \{a,b\}^V$ where binary operation $\downarrow$ is defined component-wise via*

$$\ell \downarrow \ell' = \begin{cases} \ell & \text{if } \ell' = b \\ b & \text{if } \ell' = c \end{cases} \qquad \forall \ell \in \{a,b\}, \ell' \in \{b,c\}$$

Proof. **Part (a)** It is straightforward to check that the following holds for nodes $i \in V$ and edges $\{i,j\} \in E$ respectively:

$$\begin{aligned} g_i(x_i) &= \left[\sum_{a \in \mathcal{D}} (1 - deg(a)) g_i(a)\right] + \sum_{\{a,b\} \in \mathcal{E}} g(x_i^{[ab]}) \\ \lambda_{ij} d(x_i, x_j) &= \lambda_{ij} \sum_{\{a,b\} \in \mathcal{E}} d(x_i^{[ab]}, x_j^{[ab]}) \end{aligned}$$

where $deg(a)$ is the number of neighbors of a in T. Summing these equations gives (7).

Part (b) Let $g' : \{0,1\}^V \to \mathbb{R}$ be the function obtained from g by associating $0 \mapsto a$, $1 \mapsto b$. Similarly, let $g'' : \{0,1\}^V \to \mathbb{R}$ be the function obtained from g by associating $0 \mapsto b$, $1 \mapsto c$. We can write

$$\begin{aligned} h'(y) &\triangleq \frac{g'(y)}{d(a,b)} = const + \sum_{i \in V} u'_i y_i + \sum_{\{i,j\} \in E} \lambda_{ij} |y_j - y_i| \\ h''(y) &\triangleq \frac{g''(y)}{d(b,c)} = const + \sum_{i \in V} u''_i y_i + \sum_{\{i,j\} \in E} \lambda_{ij} |y_j - y_i| \end{aligned}$$

where

$$u'_i = \frac{g_i(b) - g_i(a)}{d(a,b)} \qquad u''_i = \frac{g_i(c) - g_i(b)}{d(b,c)}$$

for $i \in V$. The T-convexity of g_i implies that $u'_i \leq u''_i$. We need to show the following: if $y', y'' \in \{0,1\}^V$ are minimizers of h' and h'' respectively then labeling $y' \vee y''$ is a minimizer of h'. This is a well-known fact about the parametric maxflow problem ([8], Lemma 2.8). Indeed,

$$\begin{aligned} & h'(y' \vee y'') - h'(y') \leq h''(y'') - h'(y' \wedge y'') \\ & = h''(y'') - h''(y' \wedge y'') + \sum_{i : (y'_i, y''_i) = (0,1)} [u'_i - u''_i] \leq 0 \end{aligned}$$

$\square$

We say that a family of binary labelings $\boldsymbol{y} = (y^{ab} \in \{a,b\}^V \mid \{a,b\} \in \mathcal{E})$ is *consistent* if there exists labeling $x \in \mathcal{D}^V$ such that $x^{[ab]} = y^{ab}$ for all $\{a,b\} \in \mathcal{E}$. Theorem 7(a) implies that the minimization of $g(x)$ over $x \in \mathcal{D}^V$ is equivalent to the minimization of

$$G(\boldsymbol{y}) = \sum_{\{a,b\} \in \mathcal{E}} g(y^{ab}) \qquad (8)$$

over consistent labelings $\boldsymbol{y} = (y^{ab} \in \{a,b\}^V \mid \{a,b\} \in \mathcal{E})$. Next, we analyze the consistency constraint.

Proposition 8. *Family $\boldsymbol{y}$ is consistent iff for every for any pair of edges $\{a,b\}, \{b,c\} \in \mathcal{E}$ with $a \neq c$ and any node $i \in V$ there holds $(y_i^{ab}, y_i^{bc}) \neq (a,c)$.*

Proof. Let us fix a node $i \in V$, and denote $\boldsymbol{y}_i = (y_i^{ab} \mid \{a,b\} \in \mathcal{E})$. Clearly, there is one-to-one correspondence between possible labelings $\boldsymbol{y}_i$ and orientations of tree T. Namely, to each $\boldsymbol{y}_i$ we associate a directed graph $\mathcal{G}[\boldsymbol{y}_i] = (\mathcal{D}, \vec{\mathcal{E}}[\boldsymbol{y}_i])$ with $\vec{\mathcal{E}}[\boldsymbol{y}_i] = \{(a,b) \mid \{a,b\} \in \mathcal{E}, y_i^{ab} = b\}$.

It can be seen that $\boldsymbol{y}_i$ is consistent (i.e. there exists $x_i \in \mathcal{D}$ with $y_i^{ab} = x_i^{[ab]}$ for all $\{a,b\} \in \mathcal{E}$) iff graph $G[\boldsymbol{y}_i]$ has exactly one sink, i.e. a node without outgoing edges. This is equivalent to the condition that each node $a \in \mathcal{D}$ has at most one outgoing edge in $G[\boldsymbol{y}_i]$. This is exactly what the condition in the proposition encodes. □

We can now prove Theorems 4 and 5. Here we prove only Theorem 4(b); for other parts were refer to [9].

Consider the following algorithm for constructing a family of binary labelings $\boldsymbol{y}$. Initially, we set $\boldsymbol{y} = (y^{ab})$ where $y^{ab} = y$ is the labeling chosen in Theorem 4(b). We also initialize subtree $T' = (\mathcal{D}', \mathcal{E}')$ of T via $\mathcal{D}' = \{a,b\}$, $\mathcal{E}' = \{\{a,b\}\}$, and then repeat the following while $T' \neq T$: (i) pick edge $\{a',b'\} \in \mathcal{E} - \mathcal{E}'$ with $a' \in \mathcal{D} - \mathcal{D}'$, $b' \in \mathcal{D}'$, add a' to $\mathcal{D}'$ and $\{a',b'\}$ to $\mathcal{E}'$; (ii) pick $y^{a'b'} \in \arg\min\{g(y) \mid y \in \{a',b'\}^V\}$; (iii) go through edges $\{b',c'\} \in \mathcal{E}'$ with $c' \neq a'$ (in some order) and replace $y^{a'b'}$ with $y^{a'b'} \downarrow y^{b'c'}$.

By Theorem 7(b), the constructed family of binary labelings $\boldsymbol{y}$ satisfies the following: $y^{a'b'} \in \arg\min\{g(y) \mid y \in \{a',b'\}^V\}$ for all $\{a',b'\} \in \mathcal{E}$. Using Proposition 8, it is also easy to check that family $\boldsymbol{y}$ is consistent; let $x \in \mathcal{D}^V$ be the corresponding labeling. Theorem 7(a) implies that x is a minimizer of g.

4. Implementation details

In this section we sketch implementation details of Algorithm 1 applied to the function constructed in section 2 (so T is a star graph with nodes $\mathcal{D} = \mathcal{L} \cup \{\mathbf{o}\}$). We will discuss, in particular, how to extract optimal flows.

We use the edge insertion operation at each call of `SPLIT` except when $\mathcal{D} = \{a, \mathbf{o}\}$ for some $a \in \mathcal{L}$. Thus, computations can be described in terms of a binary tree whose nodes correspond to subsets of labels $\mathcal{A} \subseteq \mathcal{L}$ (Fig. 3). Let Ω be the set of nodes of this tree. For each $\mathcal{A} \in \Omega$ we run a maxflow algorithm; let $V_{\mathcal{A}} \subseteq V$ be the set of nodes involved in this computation. Note that sets $V_{\mathcal{A}}$ for nodes $\mathcal{A}$ at a fixed depth form a disjoint union of V (except possibly the last level). Therefore, these maxflow computations can be treated as a single maxflow on the graph of the original size. The total number of such computations is $\lceil 1 + \log_2 k \rceil$ (the number of levels of the tree).

For each $\mathcal{A} \in \Omega$ we set up a graph with the set of nodes $V_{\mathcal{A}} \cup \{s,t\}$ and the cut function

$$f_{\mathcal{A}}(S \cup \{s\}, T \cup \{t\}) = \sum_{i \in \mathcal{V}_{\mathcal{A}}} u_i^{\mathcal{A}}[i \in T] + \sum_{\{i,j\}} \lambda_{ij}[i \in S, j \in T]$$

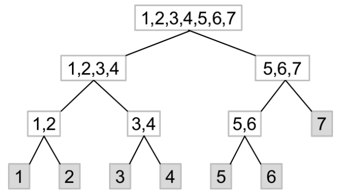

Figure 3. Binary tree for the set $\mathcal{L} = \{1, \ldots, 7\}$. Each node is a subset $\mathcal{A} \subseteq \mathcal{L}$; $\mathcal{L}$ is the root and singleton subsets are the leaves.

$(S \dot\cup T = V_{\mathcal{A}})$. To define $u_i^{\mathcal{A}}$, we need to specify the meaning of the source s and the sink t. For non-leaf nodes $\mathcal{A} \in \Omega$ the source corresponds to the left child $\mathcal{A}_\ell$ and the sink corresponds to the right child $\mathcal{A}_r$; we then have

$$u_i^{\mathcal{A}} \in [g_i(\mathbf{o}) - \min_{a \in \mathcal{A}_\ell} g_i(a), -g_i(\mathbf{o}) + \min_{a \in \mathcal{A}_r} g_i(a)] \qquad (9)$$

where we use the current value of $g_i(\mathbf{o})$ (it is zero initially and then gets decreased). For a leaf $\mathcal{A} = \{a\}$ we use a different intepretation: s corresponds to label a and t corresponds to label $\mathbf{o}$, therefore $u_i^{\mathcal{A}} = g_i(\mathbf{o}) - g_i(a)$.

We perform all maxflow computations on a single graph. We use the Boykov-Kolmogorov algorithm [3] with flow and search trees recycling [14]. We maintain values u_i for nodes $i \in V$ that give the current cut functions encoded by the residual graph. After computing maxflow at a non-leaf node $\mathcal{A}$ the residual graph is modified as follows. First, for each arc $(i \to j)$ from the source to the sink component we do the following:

1. Set $u_i := u_i - \lambda_{ij}$ and $u_j := u_j + \lambda_{ij}$; this simulates pushing flow λ_{ij} along the path $t \to j \to i \to s$.
2. Remove arcs $(i \to j), (j \to i)$ from the graph.
3. Update $g_i(\mathbf{o}) := g_i(\mathbf{o}) - \lambda_{ij}$.

Now we need to set unary costs for maxflow computations at the children $\mathcal{A}_\ell$, $\mathcal{A}_r$ of $\mathcal{A}$. Consider node $i \in V_{\mathcal{A}_c}$, $c \in \{\ell, r\}$. First, we compute the appropriate value $u_i^{\mathcal{A}_c}$; if $\mathcal{A}_c$ is not a leaf then we compute interval (9) for $\mathcal{A}_c$ and choose the value $u_i^{\mathcal{A}_c}$ from the interval closest to u_i.[2] Then we change the graph by adding $\delta_i^{\mathcal{A}_c} = u_i^{\mathcal{A}_c} - u_i$ to the capacity of $(s \to i)$ (or subtracting from the capacity of $(i \to t)$), and update $u_i := u_i^{\mathcal{A}_c}$.

Remark 1 The following property can be shown. Suppose that node $i \in V$ ended up at a leaf $\{a\} \in \Omega$. Let $\mathcal{P}$ be the path from $\mathcal{L}$ to $\{a\}$, and define values $c_i^{\mathcal{A}}$ for $\mathcal{A} \in \mathcal{P}$ so that $c_i^{\mathcal{L}} = u_i^{\mathcal{L}}$ and $c_i^{\mathcal{B}} = c_i^{\mathcal{A}} + \delta_i^{\mathcal{B}}$ for edges $(\mathcal{A}, \mathcal{B}) \in \mathcal{P}$. Then values $c_i^{\mathcal{A}}$ for nodes $\mathcal{A} \in \mathcal{P} - \{a\}$ are non-decreasing w.r.t. the **inorder** of the binary tree.[3]

[2]It can be shown that we only need to know $a^* \in \arg\min_{a \in \mathcal{L}} g_i(a)$, $g_i(a^*)$ and $g_i(\mathbf{o})$ for that.

[3]The monotonicity would also hold for the leaf $\{a\}$ if we changed the meaning of the source and the sink for computations at the leaves $\{a\} \in \Omega$ that are right children. However, we found it more convenient to use our interpretation.

Such monotonicity implies that computations at non-leaf nodes fall into the framework of *parametric maxflow* of Gallo et al. [8]. As shown in [8], all computations can be done with the same worst-case complexity as a single maxflow computation. However, this requires a more complex implementation, namely running in parallel two push-relabel algorithms. Experiments in [2] suggest that this is less efficient than a naive scheme.

Extracting flows Let us fix label $a \in \mathcal{L}$. Recall that Algorithm 1 yields the minimum of function f^a given by (2). An important question is how to obtain an optimal flow correspoding to this computation; as reported in [1], using this flow speeds up the alpha expansion algorithm.

It suffices to specify the flow ξ_{ij} for each arc $(i \to j)$ with $\{i,j\} \in E$ (the flow from the source and to the sink can then be easily computed). We used the following rule. For each edge we store flow ξ''_{ij} after the final maxflow and flow ξ'_{ij} immediately before maxflows at the leaves. For each node $i \in V$ we also store leaf $\mathcal{A}_i \in \Omega$ at which node i ended up. We now set flow ξ_{ij} as follows:

- if $\mathcal{A}_i = \mathcal{A}_j = \{a\}$ set $\xi_{ij} := \xi''_{ij}$
- otherwise let $\mathcal{A} \in \Omega$ be the least common ancestor of $\mathcal{A}_i, \mathcal{A}_j, \{a\}$ in the binary tree. If $\{a\}$ belongs to the left subtree of $\mathcal{A}$ then set $\xi_{ij} := \xi'_{ij}$, otherwise set $\xi_{ij} := \xi'_{ji}$.

We hope to prove to correctness of this procedure in a future publication; at the moment we state it as a conjecture that was verified experimentally. We mention that we were not able to find a scheme that would store only one flow per arc.

5. Relation to k-submodular functions

In this section we discuss some connections between techniques described earlier and *k-submodular functions* introduced in [16, 13]. We also define *k-submodular relaxations* of discrete functions $f : \mathcal{L}^V \to \mathbb{R}$ which generalize *bisubmodular relaxations* [17] of pseudo-Boolean functions in a natural way.

Definition 9 (k-submodularity)**.** *Let $\preceq$ be the partial order on $\mathcal{D} = \mathcal{L} \cup \{\mathbf{o}\}$ such that $a \prec b$ iff $a = \mathbf{o}$ and $b \in \mathcal{L}$. Define binary operations $\sqcap, \sqcup : \mathcal{D} \times \mathcal{D} \to \mathcal{D}$ via*

$$(a \sqcap b, a \sqcup b) = \begin{cases} (\mathbf{o}, \mathbf{o}) & \text{if } a, b \in \mathcal{L}, a \neq b \\ (\min\{a,b\}, \max\{a,b\}) & \text{otherwise} \end{cases}$$

where min *and* max *are taken w.r.t. partial order $\preceq$. Function $g : \mathcal{D}^V \to \mathbb{R}$ is called k-*submodular *(with $k = |\mathcal{L}|$) if*

$$g(x \sqcap y) + g(x \sqcup y) \leq g(x) + g(y) \qquad \forall x, y \in \mathcal{D}^V \quad (10)$$

where operations $\sqcap, \sqcup$ are applied component-wise.

It is easy to check that function g constructed in section 2 is k-submodular. Another way to obtain a k-submodular function is as follows. Consider some function $f : \mathcal{L}^V \to \mathbb{R}$. We say that function $g : \mathcal{D}^V \to \mathbb{R}$ is a *k-submodular relaxation* of f if $g(x) = f(x)$ for all $x \in \mathcal{L}^V$, and function g is k-submodular. It can be seen that any function $f : \mathcal{L}^V \to \mathbb{R}$ admits a k-submodular relaxation; we can set, for example, $g(x) = f(x)$ for $x \in \mathcal{L}^V$ and $g(x) = C$ for $x \in \mathcal{D}^V - \mathcal{L}^V$, where $C \leq \min_{x \in \mathcal{L}^V} f(x)$.

k-submodular relaxations for $k = 2$ have been studied in [17] under the name *bisubmodular relaxations*. It was shown that if f is a *quadratic* pseudo-Boolean function then the tightest bisubmodular relaxation is equivalent to the roof duality relaxation [10]. It was also proved that bisubmodular relaxations possess the *persistency*, or *partial optimality* property. The argument of [17] extends trivially to k-submodular relaxations, as the following proposition shows.

Proposition 10. *Let g be a k-submodular relaxation of f and $y^* \in \mathcal{D}^V$ be a minimizer of g. Function f has a minimizer $x^* \in \mathcal{L}^V$ such that $x^*_i = y^*_i$ for all $i \in V$ with $y^*_i \in \mathcal{L}$.*

Proof. First, observe that for any $z \in \mathcal{D}^V$ there holds $g(z \sqcup y^*) \leq g(z)$ since $g(z \sqcup y^*) - g(z) \leq g(y^*) - g(z \sqcap y^*) \leq 0$.

Let $x \in \mathcal{L}^V$ be a minimizer of f, and define $x^* = (x \sqcup y^*) \sqcup y^*$. It can be checked that $x^*_i = y^*_i$ if $y^*_i \in \mathcal{L}$, and $x^*_i = x_i$ if $y^*_i = \mathbf{o}$. Thus, $x^* \in \mathcal{L}^V$. Labeling x^* is a minimizer of f since

$$f(x^*) = g((x \sqcup y^*) \sqcup y^*) \leq g(x \sqcup y^*) \leq g(x) = f(x)$$

□

Thus, k-submodular relaxations can be viewed as a generalization of the roof duality relaxation to the case of multiple labels. Recently, Thapper and Živný showed [26] that a k-submodular function g can be minimized in polynomial time if g is represented as a sum of low-order k-submodular terms. (This was proved by showing the tightness of the *Basic LP relaxation* (BLP); when g is a sum of unary and pairwise terms, BLP is equivalent to the standard Schlesinger's LP [27].) This suggests a new possibility for obtaining partial optimality for discrete functions f.

Potts model Let us compare the approach above with the Kovtun's approach in the case of the Potts energy function f from eq. (1). A natural k-submodular relaxation of f is the function

$$\tilde{g}(x) = \sum_{i \in V} \tilde{g}_i(x_i) + \frac{1}{2} \sum_{\{i,j\} \in E} \lambda_{ij} d(x_i, x_j) \qquad (11)$$

where $\tilde{g}_i$ is a k-submodular relaxation of f_i and d is the tree metric used in section 2. It is natural to set $\tilde{g}_i(\mathbf{o})$ to the maximum possible value such that $\tilde{g}_i$ is k-submodular; this is achieved by $\tilde{g}_i(\mathbf{o}) = \frac{1}{2}[f_i(a_1) + f_i(a_2)]$ where $f_i(a_1)$ is the smallest value of f_i and $f(a_2)$ is the second smallest.

The proposition below shows that minimizing $\tilde{g}$ yields the same or fewer number of labeled nodes compared to the Kovtun's approach. Its proof is given in [9].

Proposition 11. *Let g be the function (3) corresponding to the Kovtun's approach, and $\tilde{g}$ be the k-submodular relaxation of f given by (11). Assume for simplicity that g and $\tilde{g}$ have unique minimizers x and $\tilde{x}$ respectively. If $\tilde{x}_i = a \neq \mathbf{o}$ for node $i \in V$ then $x_i = a$.*

Although a k-submodular relaxation of the Potts energy turns out to be worse than Kovtun's approach, there are clear similarities between the two (e.g. they can be solved by the same technique). We believe that exploring both approaches (or their combination) can be a fruitful direction for obtaining partial optimality for more general functions.

6. Experimental results

We applied our technique to the stereo segmentation problem on the Middlebury data [22, 23, 11]. Computations consist of two phases: (1) solve the Kovtun's approach, and (2) run the alpha-expansion algorithm for the unlabeled (or "*non-persistent*") part until convergence. For phase 1 we compared the speed of our algorithm (which we call "*k-sub Kovtun*") with the 'Reduce' method of Alahari et al. [1]. For phase 2 we used the FastPD method of Komodakis et al. [18, 19]. We used original implementations from [1] and [18, 19] and a Core i7 machine with 2.3GHz.

As a by-product, k-sub Kovtun produces a labeling which we call a *Kovtun labeling*: pixel i is assigned the label a where it ended up, as described in Sec. 4. Empirically, this labeling has a good quality - see below.

Matching costs The number of labeled pixels strongly depends on the method for computing matching costs $f_i(\cdot)$ and on the regularization parameter λ (which is the same for all edges). We tested the SSD matching costs and SSD cost averaged over the 9×9 window centered at pixel i. The latter method gave a lower error[4] in 6 out of 8 images (see Fig. 4(c)) and labeled significantly more pixels. We thus used aggregated SSD costs for all experiments.

Regularization parameter The effect of λ is shown in Fig. 4(a). Larger values of λ typically give fewer labeled pixels. For subsequent experiments we fixed $\lambda = 20$ (which is also the default value in the stereo package that comes with [22]); this value appears to work well for most of the images.

Speed comparisons The speed of different algorithms is given in Table 1. k-sub Kovtun is approximately 10 times faster than the 'Reduce' method [1] (except for Venus and Tsukuba, which have fewer labels). The fraction of non-persistent pixels ranged from 7% to 50%, which made the second phase significantly faster.

[4]As in [22], we define the error rate as the percentage of pixels whose predicted label differs from the ground truth label by more than 1.

Image(# labels)	Alahari et al.	k-sub	k-sub +FastPD	% non-persistent
Teddy(60)	3423	320	1016	18.1
Cones(60)	3858	243	466	6.8
Tsukuba(16)	519	254	469	19.3
Venus(20)	903	266	570	16.0
Lampshade1(60)	5006	523	3850	48.8
Aloe(60)	2786	236	819	10.5
Flowerpots(60)	5492	568	3489	50.5
Baby1(45)	2766	285	1095	22.0

Table 1. Runtimes (in milliseconds) and % of unlabeled pixels.

We also tested how the running time of the first phase depends on the number of labels. For this experiment we subsampled the set of allowed labels; the unary cost was set as the minimum over the interval that was merged to a given label. Results are shown in Fig. 4(b). As expected, we get a larger speed-up with more labels.

It is reported in [1] that the flow from Kovtun's computations speeds the alpha-expansion algorithm. We were unable to replicate this in our implementation. However, we observed that initializing FastPD with the Kovtun's labeling speeds it up compared to the "$\ell_{\min}$-initialization" [1].[5] The average speed-up was 14.2% (details are given in [9]).

Quality of the Kovtun's labeling We found that in the majority of cases Kovtun's labeling actually has a lower error rate compared to the alpha-expansion solution (even though the energy of the latter is better) - see Fig. 4(a). Disparity maps are shown in [9]. Since computing Kovtun's labeling requires much less computation time, we argue that it could be used in time-critical applications.

Not surprisingly, Kovtun's labeling is more reliable in the labeled part, i.e. the error rate over persistent pixels is lower compared to the rate over the entire image (Fig. 4(a)). Thus, for applications that require higher accuracy one might use an alternative technique for the unlabeled part.

7. Conclusions

We see the contributions of this work as two-fold. On the practical side, we showed how to improve the running time for the frequently used Potts model. We tested it on the stereo problem (partly because there is an established dataset for that), but we expect similar speed-ups for segmentation problems where labels correspond to different semantic classes. If the number of persistent pixels is low for a given application then one could use the cost aggregation trick to get more discriminative unary functions; as we saw for stereo, this only improves the accuracy. For time-critical applications one could potentially skip the second phase and use the Kovtun's labeling as the final output.

On the theoretical side, we introduced several concepts (such as k-submodular relaxations) that may turn out to be

[5]In this method we set $x_i \in \arg\min_a f_i(a)$. This initialization was shown in [1] to outperform the uniform initialization $x_i = 0$.

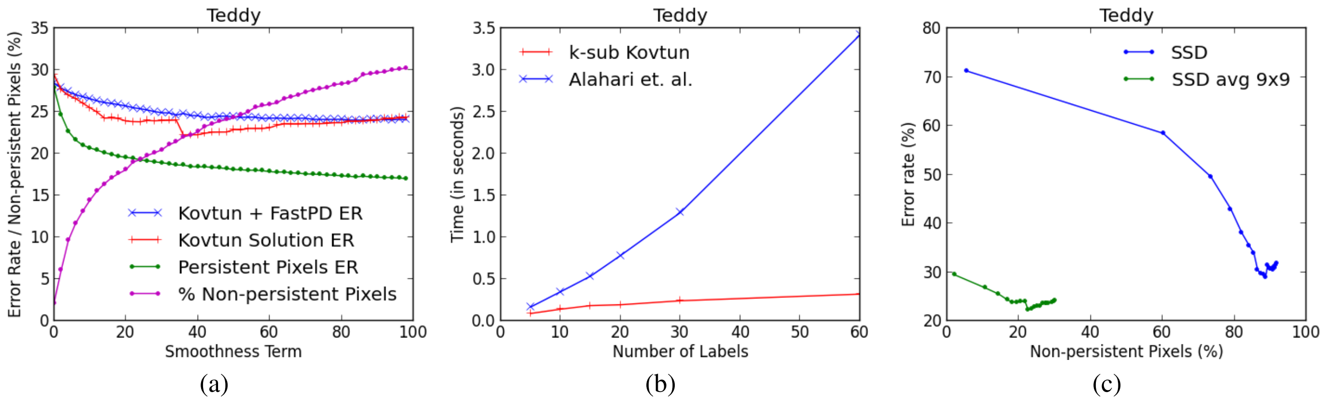

Figure 4. Results for images "Teddy": (a) dependency of the error rate on the smoothness term λ, (b) comparison of run-times of the 'Reduce' approach of [1] and k-sub Kovtun, (c) effect of the average costs aggregation - data points correspond to different values of the smoothness term from 0 to 100. **Results for other stereo pairs are given in [9].** The plots exhibit a similar behaviour, except that occasionally the ER of the Kovtun's labeling becomes worse than the ER of alpha-expansion for a sufficiently large λ (plots (a)), and for Lampshade1 and Aloe the best ER of SSD was lower than the best ER of aggregated SSD (plots (c)).

useful for other energy functions. We hope that these concepts could lead to new directions for obtaining partially optimal solutions for MAP-MRF inference.

Acknowledgements We thank the authors of [1] for answering questions about their implementation.

References

[1] K. Alahari, P. Kohli, and P. H. S. Torr. Dynamic hybrid algorithms for MAP inference in discrete MRFs. *PAMI*, 32(10):1846–1857, 2010. 1, 6, 7, 8

[2] M. Babenko, J. Derryberry, A. Goldberg, R. Tarjan, and Y. Zhou. Experimental evaluation of parametric max-flow algorithms. In *6th Int'l conference on Experimental Algorithms (WEA)*, pages 256–269, 2007. 6

[3] Y. Boykov and V. Kolmogorov. An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. *PAMI*, 26(9), September 2004. 5

[4] Y. Boykov, O. Veksler, and R. Zabih. Fast approximate energy minimization via graph cuts. *PAMI*, 23(11), 2001. 1

[5] A. Chambolle. Total variation minimization and a class of binary MRF models. In *EMMCVPR*, pages 136–152, November 2005. 2, 4

[6] J. Darbon and M. Sigelle. Image restoration with discrete constrained total variation part I: Fast and exact optimization. *J. of Math. Imaging and Vision*, 26(3):261–276, 2006. 2, 4

[7] P. Felzenszwalb, G. Pap, E. Tardos, and R. Zabih. Globally optimal pixel labeling algorithms for tree metrics. In *CVPR*, 2010. 1, 2, 3

[8] G. Gallo, M. D. Grigoriadis, and R. E. Tarjan. A fast parametric maximum flow algorithm and applications. *SIAM J. Computing*, 18:30–55, 1989. 4, 6

[9] I. Gridchyn and V. Kolmogorov. Potts model, parametric maxflow and k-submodular functions. *CoRR*, abs/1310.1771, 2013. 4, 5, 7, 8

[10] P. L. Hammer, P. Hansen, and B. Simeone. Roof duality, complementation and persistency in quadratic 0-1 optimization. *Math. Programming*, 28:121–155, 1984. 6

[11] H. Hirschmüller. Evaluation of cost functions for stereo matching. In *CVPR*, 2007. 7

[12] D. S. Hochbaum. An efficient algorithm for image segmentation, Markov Random Fields and related problems. *J. ACM*, 48:2:686–701, July 2001. 2, 4

[13] A. Huber and V. Kolmogorov. Towards minimizing k-submodular functions. In *International Symposium on Combinatorial Optimization (ISCO)*, Apr. 2012. 6

[14] P. Kohli and P. H. S. Torr. Efficiently solving dynamic Markov random fields using graph cuts. In *ICCV*, 2005. 5

[15] A. J. W. Kolen. *Tree Network and Planar Rectilinear Location Theory*. Vol. 25 of CWI Tracts. CWI, 1986. 2, 3

[16] V. Kolmogorov. Submodularity on a tree: Unifying $L^\natural$-convex and bisubmodular functions. In *36th Int'l Symposium on Math. Foundations of Comp. Science*, Aug. 2011. 6

[17] V. Kolmogorov. Generalized roof duality and bisubmodular functions. *Discrete Applied Mathematics*, 160(4-5):416–426, March 2012. 6

[18] N. Komodakis and G. Tziritas. Approximate labeling via graph cuts based on linear programming. *PAMI*, 29(8):1436–1453, 2007. 1, 7

[19] N. Komodakis, G. Tziritas, and N. Paragios. Performance vs computational efficiency for optimizing single and dynamic MRFs: Setting the state of the art with primal-dual strategies. *CVIU*, 112(1):14 – 29, 2008. 1, 7

[20] I. Kovtun. Partial optimal labeling search for a NP-hard subclass of (max,+) problems. In *DAGM*, pages 402–409, 2003. 1, 2

[21] I. V. Kovtun. *Image segmentation based on sufficient conditions of optimality in NP-complete classes of structural labelling problems*. PhD thesis, IRTC ITS National Academy of Sciences, Ukraine, 2004. (In Ukranian). 1, 2

[22] D. Scharstein and R. Szeliski. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *IJCV*, 47(1-3):7–42, Apr. 2002. 7

[23] D. Scharstein and R. Szeliski. High-accuracy stereo depth maps using structured light. In *CVPR*, pages 195–202, 2003. 7

[24] A. Shekhovtsov. *Efficient graph-based energy minimization methods in computer vision*. PhD thesis, Czech Technical University, CMP, Prague, 2013. 1

[25] A. Shekhovtsov and V. Hlavac. On partial opimality by auxiliary submodular problems. *Control Systems and Computers*, 2:71–78, 2012. 1, 2

[26] J. Thapper and S. Živný. The power of linear programming for valued CSPs. In *FOCS*, 2012. 6

[27] T. Werner. A linear programming approach to max-sum problem: A review. *PAMI*, 29(7):1165–1179, 2007. 6

[28] B. A. Zalesky. Network flow optimization for restoration of images. *J. Appl. Math.*, 2(4):199–218, 2002. 4