Optimizing Binary MRFs via Extended Roof Duality

Technical Report MSR-TR-2007-46

Carsten Rother[1], Vladimir Kolmogorov[2], Victor Lempitsky[3], Martin Szummer[1]

[1] Microsoft Research Cambridge {carrot, szummer}@microsoft.com
[2] University College London vnk@adastral.ucl.ac.uk
[3] Moscow State University victorlempitsky@gmail.com

http://research.microsoft.com/vision/cambridge/

Abstract

Many computer vision applications rely on the efficient optimization of challenging, so-called non-submodular, binary pairwise MRFs. A promising graph cut based approach for optimizing such MRFs known as "roof duality" was recently introduced into computer vision. We study two methods which extend this approach. First, we discuss an efficient implementation of the "probing" technique introduced recently by Boros et al. *[8]. It simplifies the MRF while preserving the global optimum. Our code is 400-700 faster on some graphs than the implementation of [8]. Second, we present a new technique which takes an arbitrary input labeling and tries to improve its energy. We give theoretical characterizations of local minima of this procedure.*

We applied both techniques to many applications, including image segmentation, new view synthesis, super-resolution, diagram recognition, parameter learning, texture restoration, and image deconvolution. For several applications we see that we are able to find the global minimum very efficiently, and considerably outperform the original roof duality approach. In comparison to existing techniques, such as graph cut, TRW, BP, ICM, and simulated annealing, we nearly always find a lower energy.

1. Introduction

Most early vision problems can be formulated in terms of Markov random fields (MRFs). Algorithms for MRF inference therefore are of fundamental importance for computer vision. The MAP-MRF approach (computing maximum a posteriori configurations in an MRF) has proven to be extremely successful for many vision applications such as stereo, image segmentation, image denoising, super-resolution, new view synthesis and others. We refer to [24] for an overview of MRF optimization techniques in vision.

Binary MRFs In this paper we focus on a special class of MRFs. Namely, we consider the problem of minimizing an energy function of the form

$$E(\mathbf{x}) = \theta_{\text{const}} + \sum_{p \in \mathcal{V}} \theta_p(x_p) + \sum_{(p,q) \in \mathcal{E}} \theta_{pq}(x_p, x_q) \; . \quad (1)$$

Here $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is an undirected graph. The set of nodes $\mathcal{V}$ usually corresponds to pixels, and $x_p \in \{0, 1\}$ denotes the label of node p. It is well-known that if the function E is submodular, i.e. every pairwise term θ_{pq} satisfies

$$\theta_{pq}(0,0) + \theta_{pq}(1,1) \leq \theta_{pq}(0,1) + \theta_{pq}(1,0), \quad (2)$$

then a global minimum of E can be computed in polynomial time as a minimum s-t cut in an appropriately constructed graph ("submodular graph cuts"). Submodular functions are very important, for example, for the image segmentation problem (see e.g. [9]). In many vision applications, however, condition (2) is not satisfied. We focus on the problem of minimizing non-submodular functions, which is a very challenging task (in general, NP-hard).

A promising approach for this problem called *roof duality* was proposed in [15] (see a review in [18]). It produces *part* of an optimal solution. Boros *et al.* [6] give an efficient algorithm for computing a roof dual. It can be viewed as a generalization of the standard graph cut algorithm used in vision: for submodular functions the two methods give the same answer and have exactly the same running time, except for a linear time overhead. We will refer to this method as the *QPBO* algorithm, which stands for *quadratic pseudo-boolean optimization* - this is what the minimization problem (1) is called in [15, 6]. Recently it was successfully applied to vision applications such as MR reconstruction [21] and texture restoration [18].

Our contributions In some cases the roof duality approach does not work very well, i.e. it leaves many nodes unlabeled. We investigate two extensions of the roof duality approach. The first one is the "probing" method introduced recently in [8]. This is an exact technique: it simplifies the

energy by contracting and fixing nodes while preserving the global optimum. We describe an efficient implementation and observe that our code is 400-700 times faster than the software of [8] on several 4-connected grid graphs (such grids are common in vision.) Our experiments show that this makes the algorithm practical for vision applications.

Second, we develop a new approximate technique: it takes an input solution and tries to improve its energy. The energy is guaranteed not to increase, and experimentally often decreases. Both techniques can be combined; in some cases such combination outperformed other methods that we tested (simulated annealing, ICM, max-product belief propagation, graph cut, and TRW).

Last but not least, we show the importance of the roof duality approach and its extensions for many vision applications, such as image segmentation, diagram recognition, new view synthesis, and image deconvolution. Note, experiments in [8] were outside computer vision.

Related work There is an extensive literature devoted to minimizing energy (1). Exact methods for this problem are usually branch-and-bound style methods, with different techniques for obtaining a lower bound. A large number of heuristic ideas have also been applied to this problem, e.g. tabu search, scatter search, simulated annealing, evolutionary algorithms. We refer to [3, 7] and references therein for an overview of different exact and approximate methods.

2. Optimizing Binary MRFs: Roof duality

In this section we give an overview of the roof duality approach for optimizing binary MRFs introduced in [15]. The idea is to solve a particular linear programming (LP) relaxation of the energy where integer constraints $x_p \in \{0,1\}$ are replaced with linear constraints $x_p \in [0,1]$. It can be shown that this LP has a half-integer optimal solution $\bar{\mathbf{x}}$, i.e. $\bar{x}_p \in \{0, 1, \frac{1}{2}\}$ for every node p. It is convenient to define the corresponding *partial* labeling $\mathbf{x}$ of the integer problem with $x_p \in \{0, 1, \varnothing\}$ where value $\varnothing$ means that the node is "unlabeled".

The LP relaxation above can be solved in several different ways. The algorithm in [6] is perhaps the most efficient. We review this method, which we call QPBO, in Section 2.1. As we mentioned, it produces a partial labeling $\mathbf{x}$. Properties of this labeling (in particular, *persistency*, or *partial optimality*) are discussed in Section 2.2.

2.1. The QPBO Algorithm

We describe the algorithm of [6] using the notion of *reparameterization*.

Reparameterization Let us introduce the following notation. The energy of eq. (1) is specified by the constant term θ_{const}, unary terms $\theta_p(i)$ and pairwise terms $\theta_{pq}(i,j)$ $(i,j \in \{0,1\})$. It will be convenient to denote the last two terms as $\theta_{p;i}$ and $\theta_{pq;ij}$, respectively. We can concatenate all these values into a single vector $\theta = \{\theta_\alpha \mid \alpha \in \mathcal{I}\}$ where the index set is

$$\mathcal{I} = \{\text{const}\} \cup \{(p;i)\} \cup \{(pq;ij)\} \ .$$

Note that $(pq;ij) \equiv (qp;ji)$, so $\theta_{pq;ij}$ and $\theta_{qp;ji}$ are the same element. We will use the notation θ_p to denote a vector of size 2 and θ_{pq} to denote a vector of size 4.

Vector θ' is called a *reparameterization* of vector θ if the energy functions E' and E that they define are the same, i.e. $E'(\mathbf{x}) = E(\mathbf{x})$ for all labelings $\mathbf{x}$. As a particular example, we can subtract some constant from vectors θ_p or θ_{pq} and add the same constant to θ_{const}. Another possible transformation involves edge $(p,q) \in \mathcal{E}$ and label $j \in \{0,1\}$: we can subtract a constant from components $\theta_{pq;ij}$ for all $i \in \{0,1\}$ and add the same constant to $\theta_{p;j}$.

Normal form We will say that the vector θ is in a *normal form* if it satisfies the following:

(a) $\min\{\theta_{p;0}, \theta_{p;1}\} = 0$ for all nodes p.
(b) $\min\{\theta_{pq;0j}, \theta_{pq;1j}\} = 0$ for all $(p,q) \in \mathcal{E}$ and $j \in \{0,1\}$.

Normal form implies the following: $\theta_{pq;00} = \theta_{pq;11} = 0$, $\theta_{pq;01}, \theta_{pq;10} \geq 0$ if edge (p,q) is submodular; and $\theta_{pq;01} = \theta_{pq;10} = 0$, $\theta_{pq;00}, \theta_{pq;11} \geq 0$ if (p,q) is supermodular (see fig. 1 in [18]).

Algorithm The first step of the QPBO algorithm is to reparameterize vector θ into a normal form. This can be done in linear time (see e.g. [18]). Then a directed weighted graph $G = (V, A)$ is constructed. For each node $p \in \mathcal{V}$, two nodes $p, \bar{p}$ are added to V. (They correspond to variable x_p and its negation $\bar{x}_p = 1 - x_p$, respectively). In addition, there are two special nodes - the source s and the sink t which correspond to labels 0 and 1. Thus, $V = \{p, \bar{p} \mid p \in \mathcal{V}\} \cup \{s, t\}$. For each non-zero element θ_α (except for θ_{const}) two directed arcs are added to the graph with weight θ_α; details can be found in [5, 18].

Finally, a minimum s-t cut (S, T) in G is computed by computing a maximum flow from s to t. This cut gives an optimal solution to the LP relaxation and corresponding partial labeling $\mathbf{x}$ as follows:

$$x_p = \begin{cases} 0 & \text{if } p \in S, \bar{p} \in T \\ 1 & \text{if } p \in T, \bar{p} \in S \\ \varnothing & \text{otherwise} \end{cases} . \qquad (3)$$

It is worth noting that the maximum flow in G defines a reparameterization of the energy. There are certain relations between this reparameterization and partial labeling $\mathbf{x}$ (*complementary slackness* conditions - see e.g. Appendix A).

Choosing a minimum cut One technical issue is that graph G may have several minimum cuts (S, T). They may correspond to different partial labelings $\mathbf{x}$ with different sets

of labeled nodes. In general, there exist "extreme" cuts $(S^{\min}, T^{\min})$ and $(S^{\max}, T^{\max})$ such that for any other minimum cut (S, T) there holds $\mathrm{dom}(\mathbf{x}^{\min}) \subseteq \mathrm{dom}(\mathbf{x}) \subseteq \mathrm{dom}(\mathbf{x}^{\max})$ where $\mathbf{x}^{\min}$, $\mathbf{x}^{\max}$, and $\mathbf{x}$ are the labelings defined by these cuts and $\mathrm{dom}(\mathbf{x})$ denotes the set of labeled nodes in $\mathbf{x}$ ("domain of $\mathbf{x}$").

Cut $(S^{\min}, T^{\min})$ can be set as follows: nodes reachable from s through non-saturated arcs are in $S^{\min}$, and all other nodes are in $T^{\min}$. (Alternatively, $T^{\min}$ can be set to be the set of nodes from which t can be reached through non-saturated arcs. It will yield the same labeling $\mathbf{x}^{\min}$). Computing cut $(S^{\max}, T^{\max})$ is a bit more complicated. It can be done, for example, by analyzing strongly connected components of the residual graph (details are reviewed in [18]).

Note that nodes in $\mathrm{dom}(\mathbf{x}^{\min})$ are labeled uniquely by any minimum cut (S, T). The labeling of nodes in $\mathrm{dom}(\mathbf{x}^{\max}) - \mathrm{dom}(\mathbf{x}^{\min})$, however, may depend on the cut.

2.2. Properties of QPBO

We now review properties of partial labeling $\mathbf{x}$ produced by the QPBO method ([15], see also [5]). Perhaps the most important one is the following:

[$\mathcal{P}1$] (Weak autarky) *Let $\mathbf{y}$ be an arbitrary complete labeling, and let $\mathbf{z}$=FUSE$(\mathbf{y}, \mathbf{x})$ be the "fusion" of $\mathbf{y}$ and $\mathbf{x}$: $z_p = x_p$ if $p \in \mathrm{dom}(\mathbf{x})$, and $z_p = y_p$ otherwise. Then $E(\mathbf{z}) \leq E(\mathbf{y})$.*

If we take $\mathbf{y}$ to be a global minimum, then we see that $\mathbf{x}$ is a part of some optimal solution:

[$\mathcal{P}2$] (Weak persistency, *or* partial optimality) *There exists a global minimum $\mathbf{x}^*$ of energy* (1) *such that $x^*_p = x_p$ for all labeled nodes $p \in \mathrm{dom}(\mathbf{x})$.*

Strong persistency Properties [$\mathcal{P}1$] and [$\mathcal{P}2$] are valid for any partial labeling $\mathbf{x}$ produced by QPBO. If we take $\mathbf{x} = \mathbf{x}^{\min}$, then these properties can be strengthened:

[$\mathcal{P}1'$] (Strong autarky) *Let $\mathbf{y}$ be a complete labeling, and let $\mathbf{z}$=FUSE$(\mathbf{y},\mathbf{x})$. If $\mathbf{z} \neq \mathbf{y}$ then $E(\mathbf{z}) < E(\mathbf{y})$.*

[$\mathcal{P}2'$] (Strong persistency) Any *global minimum $\mathbf{x}^*$ of energy* (1) *satisfies $x^*_p = x_p$ for all nodes $p \in \mathrm{dom}(\mathbf{x})$.*

Which nodes are labeled? Clearly, the usefulness of the algorithm depends on how many nodes are labeled. In general, we cannot expect that the method will label all nodes since minimizing energy (1) is an NP-hard problem. In some special cases, however, the method is guaranteed to label all nodes [15]:

[$\mathcal{P}3$] *If all terms of the energy are submodular then labelings $\mathbf{x}^{\max}$ produced by QPBO are complete, i.e. $\mathrm{dom}(\mathbf{x}^{\max}) = \mathcal{V}$.*

[$\mathcal{P}4$] *The algorithm is invariant with respect to "flipping" a subset of nodes $\mathcal{U} \subseteq \mathcal{V}$, i.e. swapping the meaning of 0 and 1 for nodes $p \in \mathcal{U}$. (This flipping transforms submodular terms between $\mathcal{U}$ and $\mathcal{V}\backslash\mathcal{U}$ into non-submodular, and vice versa).*

[$\mathcal{P}3$] and [$\mathcal{P}4$] imply that if there exists a flipping such that all terms become submodular then the QPBO method will label all nodes. By Harary's theorem, such a flipping exists if and only if there are no frustrated cycles in the graph. (A cycle is called *frustrated* if it contains an odd number of non-submodular terms). Thus, property [$\mathcal{P}3$] can be strengthened as follows [15]:

[$\mathcal{P}3'$] *If the energy does not have frustrated cycles then labelings $\mathbf{x}^{\max}$ produced by QPBO are complete.*

3. Extended Roof Duality

The roof duality works quite well in "simple" cases (e.g. when the number of non-submodular terms is small), but in more difficult cases it may leave many nodes unassigned (Sec. 4). In this paper we study two extensions of the roof duality approach and show that they outperform the basic algorithm for many vision applications. The first extension is the "probing" method introduced in [8]. We call it QPBOP where "P" stands for "probing". Its aim is to find the global optimum for nodes which QPBO failed to label. Sec. 3.1 reviews this method and also describes an efficient implementation. Then in Sec. 3.2 we propose a new algorithm which we call QPBOI, where "I" stands for "improve". Its aim is to efficiently improve a given reference solution. Unless noted otherwise, we will assume for simplicity that QPBO produces the (unique) strongly persistent solution $\mathbf{x} = \mathbf{x}^{\min}$, and write it as $\mathbf{x} = \mathrm{QPBO}(E)$. Note, in practice $\mathbf{x}^{\min}$ and $\mathbf{x}^{\max}$ are often the same[1].

We will use an operation called "fixing a node". Let $\mathbf{x} = \mathrm{QPBO}(E)$, and consider unlabeled node p with $x_p = \varnothing$ and given label $i \in \{0, 1\}$. Define energy $E' = E[p \leftarrow i]$ as follows: $E'(\mathbf{y}) = E(\mathbf{y}) + E_p(y_p)$ where E_p is a "hard constraint" term with $E_p(i) = 0, E_p(1-i) = C_p$ and C_p is a sufficiently large constant. If we run QPBO for energy E' then we obtain a new partial labeling $\mathbf{x}'$ in which more nodes may have become labeled. We refer to this behaviour as "spreading". It is easy to show the "monotonicity" property, i.e. that $x'_q = x_q$ for $q \in \mathrm{dom}(\mathbf{x})$ and $x'_p = i$ (see Appendix B).

Instead of adding term E_p, it is also possible to remove term θ_p from the energy along with all incident pairwise terms θ_{pq} while modifying unary terms θ_q. Note that adding

[1] This is not surprising. Indeed, QPBO may produce multiple partial labelings $\mathbf{x}$ only if the LP problem has multiple global minima; then there are several extreme points of a linear polytope which have the same cost. If, for example, costs θ_α are generated uniformly at random from some finite non-empty interval then the probability of this event is 0.

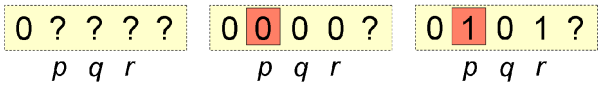

Figure 1. **Basic idea of QPBOP.** Left: QPBO labeling for the current energy, '?' means unlabeled. Middle & right: labelings after fixing node p (red). We can conclude that $x_q^* = 0$ and $x_r^* = x_p^*$ for any global minimum $\mathbf{x}^*$. Therefore, node q can be fixed to 0, and nodes p and r can be contracted.

the hard constraint term and removing node p are equivalent operations (see Appendix B).

3.1. QPBOP: Preserving Global Optimality

The basic idea of probing [8] is illustrated in Fig. 1. Let $\mathbf{x} = \text{QPBO}(E)$, and consider unlabeled node p. Let us fix p to 0 and to 1 and run QPBO in each case. We will obtain two partial labelings $\mathbf{x}^0 = \text{QPBO}(E[p \leftarrow 0])$ and $\mathbf{x}^1 = \text{QPBO}(E[p \leftarrow 1])$. Let $\mathcal{U}$ be the set

$$\mathcal{U} = \left[\text{dom}(\mathbf{x}^0) \cap \text{dom}(\mathbf{x}^1)\right] \; - \; \left[\text{dom}(\mathbf{x}) \cup \{p\}\right].$$

By the strong persistency property, we can draw the following information about global minima $\mathbf{x}^*$ of energy E:

$$x_p^* = i \quad \Rightarrow \quad x_q^* = x_q^i \qquad\qquad \forall\, i \in \{0,1\}, q \in \mathcal{U}.$$

Thus, nodes in $\mathcal{U}$ can be excluded from the energy without affecting the global minimum (or minima). Indeed, consider node $q \in \mathcal{U}$. Two cases are possible:

i) $x_q^0 = x_q^1 = j$. Then $x_q^* = j$ for all global minima $\mathbf{x}^*$, therefore x_q can be fixed to j.

ii) $x_q^0 \neq x_q^1$. This means that either (a) $x_q^0 = 0, x_q^1 = 1$, or (b) $x_q^0 = 1, x_q^1 = 0$. In case (a) we know that $x_p^* = x_q^*$ for all global minima $\mathbf{x}^*$, therefore we can "contract" p and q. In case (b) there holds $x_p^* = 1 - x_q^*$ for all global minima $\mathbf{x}$, therefore we can "flip" variable x_q (change the meaning of 0 and 1) and then contract p and q.

For details of the contract operation see Appendix B. In this operation edges (q, r) are replaced with edges (p, r), self-loops are deleted, and parallel edges are merged.

If set $\mathcal{U}$ is nonempty, then the operations above will modify the energy reducing the number of nodes; the new set of nodes is $\mathcal{V} - \mathcal{U}$. Then we run QPBO again for the new energy obtaining a new partial labeling $\mathbf{x}'$. A "monotonicity" property holds, i.e. $x'_q = x_q$ for $q \in \text{dom}(\mathbf{x})$ (see Appendix B). Thus, nodes in $\text{dom}(\mathbf{x}) - \mathcal{U}$ are labeled in $\mathbf{x}'$ (and in fact other nodes may become labeled as well).

We can repeat these operations for other nodes of the new energy. In the end we obtain a new energy E' defined on a graph $(\mathcal{V}', \mathcal{E}')$ and function $f : 2^{\mathcal{V}'} \to 2^{\mathcal{V}}$ which maps configurations $\mathbf{y}$ of energy E' to configurations $\mathbf{x}$ of the original energy[2].

[2]Function f can be described via two mappings $\alpha : \mathcal{V} \to \mathcal{V}' \cup \{0\}$, $\sigma : \mathcal{V} \to \{0,1\}$. For configuration $\mathbf{y} \in 2^{\mathcal{V}'}$ labeling $\mathbf{x} = f(\mathbf{y})$ is defined as follows: if $\alpha_p = 0$ for node $p \in \mathcal{V}$ then $x_p = \sigma_p$, otherwise $x_p = (y_q + \sigma_p) \bmod 2$ where $q = \alpha_p$. Note that mapping α defines a partitioning of the set $\mathcal{V}$.

Proposition 1. *(a) Function f gives a one-to-one mapping between the sets of optimal solutions of energies E' and E.*
(b) For any labeling $\mathbf{y} \in 2^{\mathcal{V}'}$ there holds $E(f(\mathbf{y})) = E'(\mathbf{y})$.

Adding directed constraints In practice it often happens that fixing node p to 0 and to 1 labels different sets of nodes: $\text{dom}(\mathbf{x}^0) \neq \text{dom}(\mathbf{x}^1)$. Consider node q which is labeled in $\mathbf{x}^i$ but not in $\mathbf{x}^{1-i}$. Running QPBO gave the information that has not been exploited yet; namely, if $x_p^* = i$ for a global minimum $\mathbf{x}^*$ then there must hold $x_q^* = x_q^i$.

To incorporate this information, one could add a pairwise term $E_{pq}(x_p, x_q)$ to the energy where $E_{pq}(i, 1 - x_q^i) = C$ and all other entries are zeros. (Here C is a sufficiently large constant; the exact value will not affect the correctness of the procedure, as long as C is non-negative.) QPBOP with directed constraints has the following properties which can be easily verified by induction:

Proposition 1′. *(a) (Same as in proposition 1).*
(b) For any labeling $\mathbf{y} \in 2^{\mathcal{V}'}$ with $E'(\mathbf{y}) < \min_{\mathbf{z}} E'(\mathbf{z}) + C$ there holds $E(f(\mathbf{y})) = E'(\mathbf{y})$.

One disadvantage of adding all possible constraints is that the graph might grow significantly. Unless noted otherwise, for experiments in this paper we used the following compromise: we add directed constraints only for already existing edges. Thus, no new edges are allocated. We compare this version with the full approach in Sec. 4.

In analogy with directed constraints, one could try to implement the operation of contracting nodes p and q by adding a pairwise term $E_{pq}(x_p, x_q)$ which would enforce the constraint $x_p = x_q$. It is worth mentioning, however, that this can much weaker than the contraction described above. For example, with "real" contraction parallel edges may become merged, which could make LP relaxation much tighter.

Algorithm's summary QPBOP algorithm first runs QPBO for energy E. Then it repeats the following steps until a certain stopping criterion:

- Pick unlabeled node p.
- Fix it to 0 and to 1, compute labelings $\mathbf{x}^0$, $\mathbf{x}^1$ and set $\mathcal{U}$.
- Fix or contract nodes in $\mathcal{U}$ thus removing them.
- For all edges $(p, q) \in \mathcal{E}$ with $q \in \text{dom}(\mathbf{x}^0) - \text{dom}(\mathbf{x}^1)$ or $q \in \text{dom}(\mathbf{x}^1) - \text{dom}(\mathbf{x}^0)$ add a directed constraint, unless such constraint has already been added.
- If the energy has changed, run QPBO again, update the set of unlabeled nodes.

The algorithm stops when there was a pass over all unlabeled nodes but no changes to the energy were made. At

this point no further contractions are possible, so the algorithm has converged. Clearly, the termination occurs after a polynomial number of passes over the nodes.

3.1.1 Efficient Implementation

We now describe an efficient implementation of QPBOP. As noted in [8], it is important to reuse previous calculations when the graph is updated (e.g. when the nodes are contracted). A correct implementation of the update operations, however, requires some care: the graph and flow constructed by the QPBO method have a certain property which must be preserved. Boros *et al.* [8] propose to maintain the *symmetry* condition which says that each arc holds the same flow as its "mate". This is achieved by modifying the maxflow algorithm[3]. We propose to maintain the "relaxed symmetry condition" instead (see Appendix A). One advantage of this is that the algorithm can work with integer capacities, whereas maintaining the symmetry condition requires floating point numbers (even if the original costs are integers).

We used the maxflow algorithm in [10], and reused flow and search trees as described in [16]. We modified the code so that it maintains a list of visited nodes; thus, set $\mathcal{U}$ can be traversed without going through the entire graph. Using the *relaxed* symmetry condition makes the update operation for nodes p, q quite fast: it involves only these nodes. In contrast, maintaining the symmetry condition using techniques in [10, 16] appears more difficult. After every maxflow computation we would need to go through all edges that have been accessed and restore the symmetry property. This could affect many edges, which would make reusing the search trees more complicated.

Our scheme also has a disadvantage: with the symmetry condition the lower bound on the function (represented by the amount of pushed flow) never decreases, which is not necessarily the case with the relaxed symmetry condition.

Order of processing nodes Experimentally, the order in which nodes are probed affects both the final result and the running time. The difference in final results appears insignificant in practice[4]. However, optimizing the order is quite important for reducing the number of probed nodes (and, thus, the running time). We found the following heuristic to work well. We split the execution into iterations; one iteration consists of processing nodes in a certain set. In a given iteration we record nodes which made progress, i.e. whose processing resulted in changes to the energy. We then dilate this set of nodes by a fixed amount (3 rounds), and in the next iteration test only nodes in the dilated set. When the set becomes empty, we process all nodes again. Note, in the beginning all nodes are in the set.

Comparison of implementations Results on 3 random 4-connected 100×100 grid graphs are shown below. (First column corresponds to the implementation of [8], second and third - to our implementation with standard and optimized ordering, respectively[5]). Our implementation is 400-700 times faster. It should be noted that there are many differences between the implementations: maxflow algorithm used, search tree recycling, symmetry vs. relaxed symmetry condition, integer vs. floating point capacities, data structures for storing the graph, and ordering of processing nodes. We believe that a major factor in the speed-up is the maxflow algorithm in [10] together with reusing search trees [16], whose use is simplified because of the relaxed symmetry condition. We make our code publicly available; we hope that this would have a significant practical impact in vision.

	unlabeled after QPBO (%)	unlabeled after QPBOP (%)			time (sec)		
1	92.1	5.8	5.8	5.8	718.9	2.00	1.72
2	97.9	42.2	42.2	42.6	5098.5	10.3	7.4
3	99.5	73.6	62.7	62.9	9589.6	37.3	21.4

3.2. QPBOI: Improving a Given Solution

So far we have discussed exact methods (QPBO and QPBOP) which give information about global minima of the energy. We now turn to the problem of obtaining a good approximate solution. Let us assume that we have an (complete) input labeling $\mathbf{x}$ obtained via some method (e.g. random or max-product BP). Our goal is to try to improve this labeling using QPBO.

Let us pick an arbitrary subset of nodes $\mathcal{S} \subset \mathcal{V}$ and fix nodes in $\mathcal{S}$ to labels given by $\mathbf{x}$. We denote the obtained energy as $E[\mathcal{S} \leftarrow \mathbf{x}]$. Now we can run QPBO for this energy obtaining partial labeling $\mathbf{y}$. Obviously, $y_p = x_p$ for nodes $p \in \mathcal{S}$. Property [$\mathcal{P}1'$] immediately implies the following

Proposition 2. *Let* $\mathbf{z} = \text{FUSE}(\mathbf{x}, \mathbf{y})$, *i.e.* $z_p = y_p$ *if* $p \in \text{dom}(\mathbf{y})$, *and* $z_p = x_p$ *otherwise. If* $\mathbf{z} \neq \mathbf{x}$ *then* $E(\mathbf{z}) < E(\mathbf{x})$.

Thus, we can set $\mathbf{x} := \mathbf{z}$ and repeat the procedure for a

[3] The implementation of [8] is based on Dinic algorithm, but every augmentation is performed on a pair of "mate" paths (personal communications with G. Tavares).

[4] We ran QPBOP on 200 random grid graphs of size 50×50 with two different random orders. After running QPBO 2418.4 pixels were unlabeled (on average), and after QPBOP the number of nodes in energy E' was 1219.0. Partitionings of the set $\mathcal{V}$ produced by the two runs were compared as follows. Let $\mathcal{U}$ to be the minimal set of pixels so that (i) each partition belongs either to $\mathcal{U}$ or to $\mathcal{V} - \mathcal{U}$, and (ii) partitions of $\mathcal{V} - \mathcal{U}$ in both runs are exactly the same. (Pixels fixed in one run but not in the other are included in $\mathcal{U}$; pixels fixed in both runs are excluded.) The average size of $\mathcal{U}$ was 22.2 pixels. In 37% of the cases results were identical.

[5] To conform to the implementation of [8], all methods add all possible directed constraints and use weakly persistent solution produced by QPBO in the main loop (but not in the probing operations). Weakly persistent solutions are not unique, which accounts for slightly different percentages of unlabeled nodes.

different subset $\mathcal{S}$. The construction guarantees that the energy of labeling $\mathbf{x}$ does not increase. We call this technique QPBOI, where "I" stands for improve.

To achieve efficiency, we propose to use a nested sequence of subsets $\mathcal{S}$ according to some ordering of nodes $\pi : \mathcal{V} \to \{1, \dots, |\mathcal{V}|\}$. Then flow and search trees can be reused as in [16]. One iteration of this procedure is given below.

- Select an ordering of nodes π.
- Initialization: Compute $\mathbf{y} = \text{QPBO}(E)$, set $\mathbf{x} := \text{FUSE}(\mathbf{x}, \mathbf{y})$, $\mathcal{S} := \text{dom}(\mathbf{y})$.
- For nodes $p \in \mathcal{V}$ do in the order π:
  - If $p \notin \mathcal{S}$ compute $\mathbf{y} = \text{QPBO}(E[\mathcal{S} \cup \{p\} \leftarrow \mathbf{x}])$, set $\mathbf{x} := \text{FUSE}(\mathbf{x}, \mathbf{y})$, $\mathcal{S} := \text{dom}(\mathbf{y})$.

It is worth mentioning that the QPBOI procedure can be generalized: rather than fixing nodes to values in $\mathbf{x}$, we can (i) enforce hard constraints satisfied by current labeling $\mathbf{x}$ via contracting nodes of the energy; (ii) Add non-negative numbers to elements θ_α such that the cost of $\mathbf{x}$ stays the same. It is not difficult to see that proposition 2 still holds. We informally tested a particular variant[6] but found that it performed worse than fixing nodes.

Local minima of QPBOI In order to understand the capabilities of QPBOI we now analyze local minima of this procedure.

Definition 3. *Labeling $\mathbf{x}$ is called* stable *(or* QPBO-stable*) if no QPBOI operation can change it, i.e. for any subset $\mathcal{S} \subset \mathcal{V}$ there holds $y_p = x_p$ for $p \in \text{dom}(\mathbf{y})$ where $\mathbf{y} = \text{QPBO}(E[\mathcal{S} \leftarrow \mathbf{x}])$.*

The theorem below exhibits a large class of functions for which stable labelings are essentially global minima.

Theorem 4. *Suppose energy E and labeling $\mathbf{x}$ satisfy at least one of the following conditions:*

(a) E does not have frustrated cycles.

(b) $\theta_{\text{const}} = 0$, $\theta_{p;i} \geq 0$ for all indexes $(p;i)$, $\theta_{pq;ij} \in \{0, C\}$ for all indexes $(pq;ij)$ where C is a positive constant, and $E(\mathbf{x}) < C$.

Then $\mathbf{x}$ is stable iff it is a global minimum of E.

[6] We tested contracting nodes p, q for existing edges (p, q). In the beginning we flip a subset of nodes so that the current labeling $\mathbf{x}$ becomes 0 for all nodes, and maintain this property afterwards. In other words, if after any of the following operations node p gets label 1 then we flip p.

Next, we choose a random order in which edges are processed. Then we iterate over all edges in the given order (possibly multiple times). Edges which are supermodular are contracted, and edges which are submodular are not contracted; instead, they are moved to the next iteration. The algorithm terminates when all edges of the energy become submodular. (Clearly, this happens after a polynomial number of steps.) Therefore, conditions of property [$\mathcal{P}3'$] are satisfied.

Note that since in case (b) there exists a solution whose cost is smaller than C, pairwise terms act as hard constraints; for any solution $\mathbf{y}$ with $E(\mathbf{y}) \leq E(\mathbf{x})$ there must hold $\theta_{pq}(y_p, y_q) = 0$ for all edges (p, q). A proof of the theorem is given in Appendix C; it relies on characterization [$\mathcal{P}3'$].

In general, however, there may be stable solutions which are not optimal. A simple example is the energy

$$E(x, y, z) = 3|x - y| + 3|y - z| + 2xy + (1 - x)(1 - y).$$

$(1, 1, 1)$ is a stable solution, and $(0, 0, 0)$ is the optimum. (Note that $E(1, 1, 1)$ is larger than $E(0, 0, 0)$ but smaller than the cost of all other labelings. Also, running QPBOI with the empty set $\mathcal{S}$ does not label any nodes.)

Using theorem 4 it is not difficult to show the following negative result (see Appendix D):

Theorem 5. *Testing whether a labeling is stable is a co-NP complete problem (under Turing reductions).*

Thus, obtaining good orderings in the QPBOI procedure is a difficult task. Nevertheless, experimental results in Section 4 show that in many cases random permutations do decrease the energy, at least during the first few iterations.

3.3. Summary of Algorithms

There are several options for using the techniques described in Sec. 3. An important question for QPBOI is how to initialize it. Furthermore, QPBOP and QPBOI can be combined. We settled on the following four methods:

QPBOP This technique is designed for obtaining partial optimal solutions. We demonstrate that in many cases it significantly outperforms QPBO, i.e. it produces fewer unlabeled nodes.

BP+I First we run QPBO, then max-product BP (only for unlabeled nodes) and finally improve the solution using QPBOI with random permutations of nodes.

We used a "sequential" schedule of BP as in [17]. Before starting BP, we reparameterized the energy so that $\theta_{pq;00} = \theta_{pq;11}$, and $\theta_{pq;01} = \theta_{pq;10}$ for each edge (p, q). (Note that it does not make sense to start with the reparameterization obtained after running QPBO, since such reparameterization is a fixed point of BP).

QPBOI is stopped when the energy has not improved for 5 iterations, and BP is run for a large number of iterations (here 1000) and the best result is taken.

P+BP+I First we run QPBOP obtaining new energy E' and mapping $f : 2^{\mathcal{V}'} \to 2^{\mathcal{V}}$. Then we apply BP+I for energy E'; this gives solution $\mathbf{y} \in 2^{\mathcal{V}'}$. The output of P+BP+I is the labeling $\mathbf{x} = f(\mathbf{y})$ [7]

[7] Note that QPBOI applied to energy E' never increases the cost $E'(\mathbf{y})$. By property 1′(b) cost $E(\mathbf{x})$ where $\mathbf{x} = f(\mathbf{y})$ also does not increase,

P+I In some scenarios an input labeling $\mathbf{x}$ for energy E is available, and it is desirable that the method does not increase it. (An example is the expansion move algorithm [11]; the input labeling $(0, \dots, 0)$ corresponds to the current configuration.) We now show how to combine QPBOP and QPBOI to ensure this property. The basic step of the QPBOP is to fix node p to 0 and to 1 and compute corresponding partial labelings $\mathbf{x}^0$ and $\mathbf{x}^1$. We propose to update labeling $\mathbf{x}$ as follows: $\mathbf{x} := \text{FUSE}(\mathbf{x}, \mathbf{x}^i)$ where $i = x_p$. It can be seen that fix and contract operations preserve the "structure" of $\mathbf{x}$, e.g. if node q is fixed to label j during QPBOP then there must hold $x_q = j$. Let $\mathbf{y}$ be the transformed labeling for energy E'. It is easy to verify by induction that if constant C in QPBOP is sufficiently large (namely, $E(\mathbf{x}) < \min_{\mathbf{z}} E(\mathbf{z}) + C$) then there holds $E'(\mathbf{y}) = E(f(\mathbf{y})) \leq E(\mathbf{x})$. After QPBOP we run QPBOI for energy E' starting with labeling $\mathbf{y}$.

4. Experiments

In this section we will first investigate the performance of the methods described above with respect to various MRF settings. Then we consider six different applications with non-submodular MRFs and compare them to a standard set of MRF optimization methods. Finally we show the usefulness of our new P+I method within the standard alpha-expansion procedure [11] to optimize a multi-labeled MRF.

4.1. Performance of QPBOP and QPBOI

In the following we measure the improvement of QPBOP over QPBO in terms of additionally labeled pixels. For QPBOI the improvement, with respect to a given reference solution, is measured in terms of lower energy. We are also interested in the runtime overhead for both methods with respect to QPBO. In general, the performance of QPBO (and extended versions) strongly depends on three factors: number of non-submodular terms (ideally, the number of frustrated cycles), connectivity (i.e. average degree of a node), and strength of unary versus pairwise terms. This strength is computed as: $\text{mean}_{p,i}\theta_{p;i}/\text{mean}_{p,q,i,j}\theta_{pq;ij}$, after conversion into normal form (similar to [7]). Fig. 2 shows several variations of these factors.

QPBOP significantly extends QPBO in a way that is not achievable by common exact methods. To illustrate this, we took a random 4-connected graph of size 100×100 for which QPBO failed to label 73% of nodes (runtime was 0.08 sec). We found that the unlabeled nodes induce a clique of size 45 (treewidth 44), based on the min-fill method [4], with other triangulations being even larger. Thus the exact junction tree algorithm is completely infeasible here, requiring Terabytes of RAM just to store the

assuming that the output of BP $\mathbf{y}$ satisfies $E'(\mathbf{y}) < \min_{\mathbf{z}} E'(\mathbf{z}) + C$. Note that if constant C is sufficiently large, then it is easy to find labeling $\mathbf{y}$ that satisfies this - see P+I.

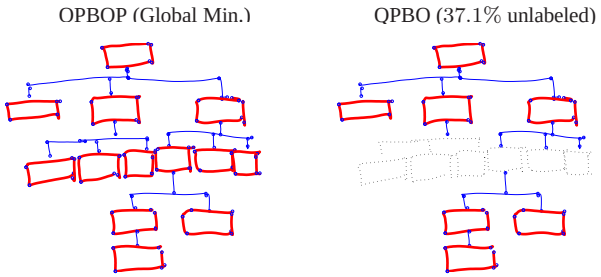

Figure 4. **Diagram recognition.** Given a raw unlabeled hand-drawing, the task is to classify whether each pen stroke is part of a container (red and bold) or a connector (blue). QPBOP finds the global minimum and labels all strokes correctly (left), whereas standard QPBO finds only part of the global solution and leaves 37.1% of the pen strokes unlabeled (dashed in the right diagram).

clique. In contrast, QPBOP found the global minimum of this problem in 0.4 sec.

4.2. Applications

In the following we will compare a standard set of optimization techniques [24] (ICM, BP, TRW-S, Graph Cut, QPBO, and Simulated Annealing) with the new methods on real world applications where binary non-submodular MRFs occur. We are interested to see which method achieves the best performance in a "reasonable" time, i.e. up to several seconds depending on the application. Therefore, we do not plot runtime versus energy but simply report the energy and runtime of the best result for each method. Some notes on the competitive methods: Iterative Conditional Modes (ICM) is run with a random traversal order until convergence. Details of Belief Propagation (BP) are described in Sec. 3.3. Since graph cut (GC) cannot handle non-submodular terms we truncated them as in [23]. Simulated annealing (SA) is capable of producing high quality results with potentially long runtimes. We tweaked the parameters of SA for each *individual* problem, to achieve best results. Finally, TRW-S is guaranteed to give the same answer as QPBO [19], therefore we omit it. (We verified experimentally that the result for labeled nodes is identical. Furthermore, running TRW-S until convergence of the lower bound is much slower than QPBO in practice.)

Table 1 lists the comparison of all methods for one or two examples of each of the six applications.

Diagram Recognition Shape recognition in hand-drawn diagrams is an application where the QPBOP method considerably outperforms standard QPBO. We tested 2700 diagram problems, with an average of 64 nodes and connectivity of 4.1. The MRF model is described in [25]. QPBOP returned the global minimum for **all** problems, whereas QPBO failed to label all nodes in 97 cases, with between 5% and 56% of nodes unlabeled. One of those challenging examples is shown in fig. 4. Another difficult example is listed in table 1 where the new methods (P+BP+I and BP+I) attain the lowest energy, and QPBOP confirms that

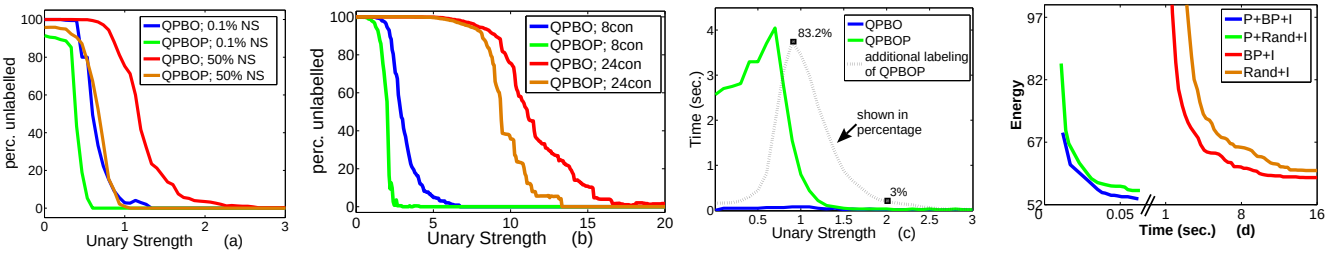

Figure 2. **Performance of QPBOP and QPBOI.** As default we use a 4-connected random graph in normal form of size 100×100 pixels with 50% non-submodular terms and unary strength 0.8. (a,b) compares QPBO and QPBOP with respect to varying non-submodularity (NS) and connectivity (con). The percentage of labeled nodes decreases for: (i) a large number of non-submodular terms, (ii) high connectivity, or (iii) a small unary strength. Note that the improvement of QPBOP over QPBO differs between 0% and 90% of additionally labeled nodes depending on the MRF settings. (c) compares the runtime of QPBO and QPBOP. As to be expected, the runtime increases significantly when QPBOP labels considerably more nodes than QPBO. For practical use an important range is the unary strength between 1.5 and 3, where QPBOP is able to compute (most of the time) the global minimum with only a small runtime overhead. (d) illustrates energy versus runtime for P+BP+I, BP+I, and both methods with a random starting point (P+Rand+I, Rand+I), i.e. a reference solution with random labeling. Note, QPBOI was stopped when the energy did not increase for 5 iterations. We see that using BP as the starting point consistently gives a better result. For this particular problem running QPBOP first gives a large improvement in runtime, however, for certain applications the runtime overhead of QPBOP can be considerable and therefore BP+I may sometimes be preferred in practice.

Applications	Sim. An.	ICM	GC	BP	BP+I	P+BP+I	QPBO	QPBOP
Diagram recognition (4.8con)	0 (0.28s)	999 (0s)	119 (0s)	25 (0s)	0 (0s)	0 (0s)	56.3% (0s)	0% (0s) **GM**
New View Synthesis (8con)	- (-s)	999 (0.2s)	2 (0.3s)	18 (0.6s)	0 (2.3s)	0 (1.4s)	3.9%(0.7s)	0% (1.4s) **GM**
Super-resolution (8con)	7 (52s)	68 (0.02s)	999 (0s)	0.03 (0.01s)	0.001 (0.06s)	0 (0.047s)	0.5% (0.016s)	0% (0.047s) **GM**
Image Segm. 9BC + 1 Fgd Pixel (4con)	983 (50s)	999 (0.07s)	0 (28s)	28 (0.2s)	0 (31s)	0 (10.5s)	99.9% (0.08s)	0% (10.5s) **GM**
Image Segm. 9BC; 4RC (4con)	900 (50s)	999 (0.04s)	0 (14s)	24 (0.2s)	0 (3s)	0 (1.48s)	1% (1.46s)	0% (1.48s) **GM**
Texture restoration (15con)	15 (165s)	636 (0.26)	999 (0.05s)	19 (0.18s)	0.01 (2.4s)	0 (14s)	16.5% (1.4s)	0% (14s) **GM**
Deconvolution 3×3 kernel (24con)	0 (0.4s)	14 (0s)	999 (0s)	5 (0.5s)	3.6 (1s)	0 (0.4s)	45% (0.01s)	43% (0.4s)
Deconvolution 5×5 kernel (80con)	0 (1.3s)	6 (0.03s)	999 (0s)	71 (0.9s)	8.1 (31s)	8.1 (31s)	80% (0.1s)	80% (9s)

Table 1. **Comparison table for different applications.** Results are given as: Energy (runtime in seconds). For each problem the energies are scaled to the range of 0 to 999. Note that an energy of 0 in the last two rows does not mean that this is the global optimal solution. The last two columns show the percentage of unlabeled nodes for QPBO and QPBOP, where GM means global minimum. For segmentation BC means boundary constraint and RC region constraint. Also, for segmentation, graph cut was run 2^n (n = number BC) times with flow and search tree recycling to obtain the global minimum. Note, ICM and simulated annealing do not perform well for applications with hard pairwise constraints (infinite links), such as segmentation and new view synthesis.

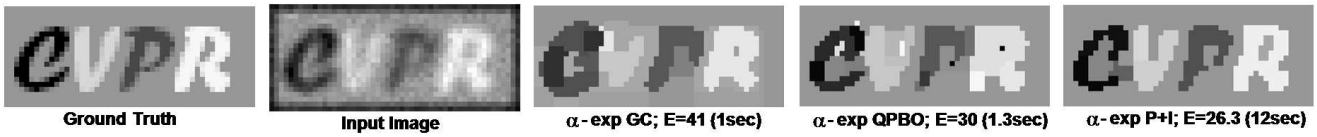

Figure 3. **Image deconvolution.** Given a blurry and noisy input image with 32 gray-levels, the task is to reconstruct the ground truth. The new P+I method within alpha-expansion improves results compared to graph cut and QPBO-based alpha expansion both in terms of energy and visually. Note, for all alpha expansion based methods the order is crucial. Energies of P+I differed between 26.3 and 27.9 and runtime between 12 and 31 sec, best result shown. Also, to improve runtime of P+I we initialized it with standard graph cut based alpha expansion. BP (E=103), TRW (E=112) and ICM (E=54) perform poorly. Simulated annealing achieved a similar result to P+I in 60 sec.

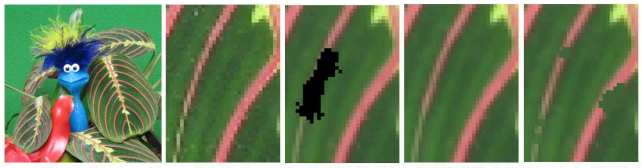

Figure 5. **New View Synthesis** where QPBO leaves 5731 pixels (3.9%) unlabeled (black), QPBOP finds the global minimum, and graph cut (best of all competitors) has visually noticeable artifacts.

this is indeed the global minimum.

Super-resolution and new view synthesis For super-resolution we used the approach of [14] where a node label corresponds to a patch from a reference patch dictionary. The MRF pairwise terms encode the compatibility of overlapping patches of neighboring nodes. The amount of non-submodularity can be high, e.g. 45%. The unary terms encode color consistency with the low-resolution image. To make it suitable for our purpose we use two labels and a 5×5 patch size which correspond to an 8-connected MRF (no overlap in the 3×3 center as in [14]). For New

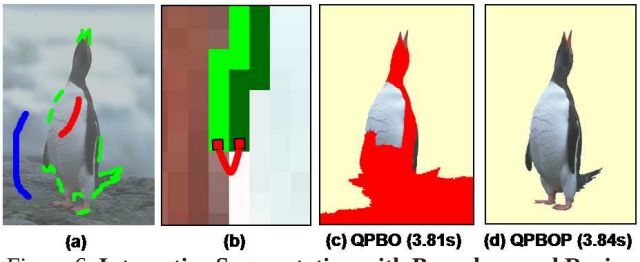

Figure 6. **Interactive Segmentation with Boundary and Region Constraints.** (a) Input image with superimposed user inputs: one inside brush (red), one outside brush (blue) and 9 unconnected boundary constraints (green), bold for better visibility. (b) Zoom into a boundary constraint: Pixels on each line (light and dark green) are constrained to have the same labeling and one extra non-submodular link (red) constrains both lines to have opposite labels. For optimal speed QPBOP first probes pixels at the red links. Note that alternative formulations, e.g. a "fat" intelligent scissors brush, with no specific start and endpoints, are possible and give similar results. The segmentation result of the penguin using QPBO (c) has 26.7% of unlabeled pixels (red), where QPBOP (d) finds the global minimum in about the same time as QPBO.

View Synthesis as introduced in [13] we may use the same MRF structure, where labels are now color modes derived from depth images (details omitted). We have tested several examples and parameter settings for both applications and may conclude that QPBO typically has a smaller number of unlabeled nodes, e.g. up to 3.9% (in total 5731 pixels) for example in fig. 5. QPBOP is able to find the global minimum most of the time with very little extra runtime (see examples in table 1).

Image segmentation An important issue for interactive image segmentation is the combination of boundary constraints, as in intelligent scissors [20], and region constraints, as in [9]. Here we show that this is possible by including a few non-submodular terms, see fig. 6. We have tested our system for many images, where two examples are listed in table 1. The conclusion is that QPBOP is able to give the global minimum for all examples we have tested, and outperforms QPBO considerably. The speed of QPBOP is affected by the number of brush strokes, the more the faster. All other methods perform very poorly for this application. Note, an alternative approach to compute the global minimum is to run standard graph cut 2^n times where n is the number of unconnected boundary constraints. For the example in fig. 6 where $n = 9$, running 512 graph cuts with flow and tree recycling [16] and an optimized order took in the best case 16 sec (84 sec without recycling) which is considerably more than the 3.8 sec of QPBOP.

Parameter learning for binary texture restoration In this application we restore a noisy test image of a texture, based on an MRF model learned from a training image of the same texture type. We used the same learning procedure as described in [18] based on [12] with the only difference that QPBO is replaced by P+BP+I. We have done this for one Brodatz texture D103 (see [18]) where the test error (averaged over 20 examples) reduces from 25.4 to 25.1 when using P+BP+I instead of QPBO. One example is listed in table 1 where P+BP+I achieved the global minimum. For this application BP+I achieved nearly the same result with a speed-up factor of 6.

Image deconvolution In [22] image deconvolution was formulated as a labeling problem with a pairwise MRF and solved using graph cut based alpha-expansion. Given an $n \times n$ convolution kernel the MRF connectivity is $(2n-1) \times (2n-1) - 1$. Fig. 3 shows an example of reconstructing an input image with 32 different gray-scales and convolution with a 3×3 kernel. To solve this 32 label problem we use alpha-expansion where the P+I method (Sec. 3.3) is used as the binary optimizer. Raj *et al.* [21] also use QPBO-based alpha expansion to reconstruct MR images, although with a sparsely connected MRF.

We have also used the deconvolution MRF with only two labels to reconstruct binary images. Table 1 gives two results with different convolution kernels. The main conclusion is that for highly connected MRFs, e.g. connectivity 80, simulated annealing outperforms all other methods including P+BP+I, and QPBOP performs similarly to QPBO.

5. Conclusions and Future Work

We presented an efficient implementation of the QPBOP method in [8] which is 2-3 orders of magnitude faster than the implementation of [8] on some vision related graphs. We introduced a new technique called QPBOI for optimizing binary non-submodular MRFs, and proved theoretical properties of this method. We have verified experimentally that QPBOP finds the global minimum for many vision applications and that QPBOI nearly always achieves a lower energy with respect to any given reference solution that we have tested. Both techniques are efficient due to graph cut with flow and search tree recycling.

We believe that the main impact of our work lies on the application side, where we plan to further investigate MRFs with high order cliques and multiple labels. Also, most handcrafted MRFs in computer vision are submodular, which is not necessarily true for learned MRFs. Consequently we believe that the demand for efficient, but general, optimizers both during MRF learning and inference, will increase considerably in the future. Finally, we will make the code and energies publicly available, as a step towards a benchmarking system for optimizing challenging, non-submodular MRFs in computer vision, similar to [24].

Acknowledgements We thank the anonymous reviewers for pointing out the recent work [8], Gabriel Tavares for running some tests on our datasets and sharing the implementation of "probing" used in [8], and Oliver Woodford for providing us with energies for the new view synthesis problem.

Appendix A: Relaxed symmetry condition

Recall that the QPBO method constructs a directed weighed graph $G = (V, A, c)$. The vertex set is $V = \{p, \bar{p} \mid p \in \mathcal{V}\} \cup \{s, t\}$ and arcs in A correspond to non-zero components of vector θ (see e.g. [18]). Each node $u \in V$ has a "mate" $\bar{u} \in V$ such that $\bar{\bar{u}} = u$; similarly, each arc $a = (u \to v) \in A$ has a "mate" arc $\bar{a} = (\bar{v} \to \bar{u}) \in A$. Graph G with residual capacities $c \geq 0$ defines an energy function $E(\cdot \mid \theta)$ with the following parameter vector θ:

$$\begin{array}{ll} \theta_{p;0} = c_{pt} + c_{s\bar{p}} & \theta_{p;1} = c_{sp} + c_{\bar{p}t} \\ \theta_{pq;00} = c_{p\bar{q}} + c_{q\bar{p}} & \theta_{pq;01} = c_{pq} + c_{\bar{q}\bar{p}} \\ \theta_{pq;10} = c_{\bar{p}\bar{q}} + c_{qp} & \theta_{pq;11} = c_{\bar{p}q} + c_{\bar{q}p} \end{array} \quad (4)$$

(We assume that $c_{uv} = 0$ if there is no arc $(u \to v)$ in A.)

For convenience, we will assume that A contains arcs $(s \to u)$, $(u \to t)$, $(u \to s)$, $(t \to u)$ for every node $v \in V - \{s, t\}$. Let us also define the following sets:

$$\begin{array}{ll} V^{\text{int}} = V - \{s, t\} & \text{(set of "interior" nodes)} \\ A^{\text{int}} = \{(u \to v) \in A \mid u, v \in V^{\text{int}}\} & \text{(set of "interior" arcs)} \\ A^{st} = \{(u \to v) \in A \mid u \neq t, v \neq s\} & \end{array}$$

Boros et al. [8] maintain the *symmetry condition* which says that each arc $a \in A$ has the same flow as its mate, so $c_a = c_{\bar{a}}$. This can be achieved by modifying the maxflow algorithm, as in [8]. Alternatively, after every maxflow computation one could go through arcs and set $c'_a := \frac{1}{2}(c_a + c_{\bar{a}})$, $c'_{\bar{a}} := \frac{1}{2}(c_a + c_{\bar{a}})$. Both schemes require floating point numbers. The latter scheme also has the following disadvantage: because of rounding errors, after restoring the symmetry condition the flow is not necessarily optimal, i.e. there may be augmenting paths from s to t of very small capacity. (We observed this in practice). There are several ways to handle this complication, e.g. iterate the maxflow algorithm until a symmetric optimal flow is found.

We maintain the following *relaxed symmetry condition* instead:

$$-c_u + \sum_{(u \to v) \in A^{\text{int}}} c_{uv} = c_{\bar{u}} + \sum_{(\bar{v} \to \bar{u}) \in A^{\text{int}}} c_{\bar{v}\bar{u}} \qquad \forall\, u \in V^{\text{int}} \quad (5a)$$

$$c_{uv} + c_{vu} = c_{\bar{v}\bar{u}} + c_{\bar{u}\bar{v}} \qquad \forall\, (u \to v) \in A^{\text{int}} \quad (5b)$$

where $c_u = c_{su} - c_{ut}$ for node $u \in V^{\text{int}}$. These conditions essentially say that (V, A, c) can be obtained by pushing flow in a graph satisfying the symmetry condition:

Proposition 6. *For arc $a = (u \to v) \in A^{st}$ let $c'_a = \frac{1}{2}(c_a + c_{\bar{a}})$ and $f_a = c'_a - c_a = \frac{1}{2}(c_{\bar{a}} - c_a)$. Also, let $f_a = -f_a$ for $a \in A - A^{st}$. Then f is a circulation in G, i.e. it satisfies*

$$\begin{array}{lll} f_{uv} = -f_{vu} & \forall\, (u \to v) \in A & \textit{(antisymmetry)} \\ \sum\limits_{(u \to v) \in A} f_{uv} = 0 & \forall\, u \in V & \textit{(flow conservation)} \end{array}$$

Proof. The antisymmetry property for arc $(u \to v) \in A^{\text{int}}$ follows from (5b):

$$\begin{aligned} 2(f_{uv} + f_{vu}) &= (c_{\bar{v}\bar{u}} - c_{uv}) + (c_{\bar{u}\bar{v}} - c_{vu}) \\ &= (c_{\bar{v}\bar{u}} + c_{\bar{u}\bar{v}}) - (c_{uv} + c_{vu}) \;=\; 0 \end{aligned}$$

Let us verify the flow conservation property. For node $u \in V^{\text{int}}$, we have

$$\begin{aligned} 2 \sum_{(u \to v) \in A} f_{uv} &= 2 \left[-f_{su} + f_{ut} + \sum_{(u \to v) \in A^{\text{int}}} f_{uv} \right] \\ &= -(c_{\bar{u}t} - c_{su}) + (c_{s\bar{u}} - c_{ut}) + \sum_{(u \to v) \in A^{\text{int}}} (c_{\bar{v}\bar{u}} - c_{uv}) \\ &= \left[c_{\bar{u}} + \sum_{(\bar{v} \to \bar{u}) \in A^{\text{int}}} c_{\bar{v}\bar{u}} \right] - \left[-c_u + \sum_{(u \to v) \in A^{\text{int}}} c_{uv} \right] = 0 \end{aligned}$$

To show flow conservation at the source s, let us sum expression (5a) for all nodes $u \in V^{\text{int}}$ and subtract half of the sum of (5b) for all arcs $(u \to v) \in A^{\text{int}}$. Then we obtain

$$\sum_{u \in V^{\text{int}}} -c_u = \sum_{u \in V^{\text{int}}} c_{\bar{u}}$$

The expression on the LHS is the negation of the expression on the RHS, so it must equal zero. Thus,

$$\begin{aligned} 2 \sum_{(s \to u) \in A} f_{su} &= \sum_{u \in V^{\text{int}}} (c_{\bar{u}t} - c_{su}) \\ &= \sum_{u \in V^{\text{int}}} (c_{ut} - c_{su}) = \sum_{u \in V^{\text{int}}} -c_u = 0 \end{aligned}$$

Conservation at the sink t can be shown in a similar way. (In fact, it follows from the conservation at other nodes.) □

This proposition together with results in [5] imply the correctness of the relaxed symmetry condition. For completeness, let us state and prove main properties.

Proposition 7. *Suppose that residual capacities c in graph G satisfy equations* (5)*. Let $E(\cdot \mid \theta)$ be the energy defined by* (4)*. Let (S, T) be a minimum cut in G and $\mathbf{x}$ be the corresponding partial labeling defined by* (3)*.*

(a) Pushing s-t flow of value C through graph G maintains the relaxed symmetry condition (5)*. Furthermore, it corresponds to a reparameterization of energy $E(\cdot \mid \theta)$ if component θ_{const} is increased by C.*

(b) Property [$\mathcal{P}1$] ("weak autarky") holds for $\mathbf{x}$ and E.

(c) If there are no augmenting paths from s to t, then the following complimentary slackness conditions hold:

$$\begin{array}{ll} \theta_\alpha \geq 0 & \forall\, \alpha \in \mathcal{I} - \{\text{const}\} \\ \theta_p(x_p) = 0 & \forall\, p \in \text{dom}(\mathbf{x}) \\ \theta_p(i) = 0 & \forall\, p \notin \text{dom}(\mathbf{x}), i \in \{0, 1\} \\ \theta_{pq}(x_p, x_q) = 0 & \forall\, p, q \in \text{dom}(\mathbf{x}) \\ \theta_{pq}(x_p, j) = 0 & \forall\, p \in \text{dom}(\mathbf{x}), q \notin \text{dom}(\mathbf{x}), j \in \{0, 1\} \end{array} \quad (6)$$

(Note, by a "cut" we always mean an s-t cut).

Proof. Checking (a) is a straightforward calculation [5]. Part (b) follows easily from (a) and (c). Indeed, by (a) we can assume without loss of generality that we have a maximum flow in G, and thus (c) can be applied. If $\mathbf{y}$ is a complete labeling and $\mathbf{z} = \text{FUSE}(\mathbf{y}, \mathbf{x})$ then $E(\mathbf{z} \mid \theta) \leq E(\mathbf{y} \mid \theta)$ since by (6) each term in the sum $E(\mathbf{z}) = \theta_{\text{const}} + \sum_p \theta_p(z_p) + \sum_{(p,q)} \theta_{pq}(z_p, z_q)$ is the same or smaller than the corresponding term in the sum $E(\mathbf{x}) = \theta_{\text{const}} + \sum_p \theta_p(x_p) + \sum_{(p,q)} \theta_{pq}(x_p, x_q)$.

Let us prove part (c). For subset $U \subseteq V$, define $U^\star = V - \{u \mid \bar{u} \in U\}$. The relaxed symmetry condition implies the following key property:

$$cost(S,T) = cost(S^\star, T^\star) \qquad \text{for any cut } (S,T) \qquad (7)$$

In particular, if (S,T) is a minimum cut then so is $(S^\star, T^\star)$. Indeed, (7) clearly holds if capacities c in graph G are symmetric. Pushing flow in G preserves the cost of a cut, so by proposition 6 property (7) holds if capacities c satisfy the relaxed symmetry condition.

Consider arc $a = (u \to v) \in A$, $u \in S$, $v \in T$. We have $\bar{v} \in S^\star$, $\bar{u} \in T^\star$, $\bar{a} = (\bar{v} \to \bar{u}) \in A$. By the Ford-Fulkerson theorem, arcs a and $\bar{a}$ are saturated: $c_a = c_{\bar{a}} = 0$. Using this property, checking part (c) amounts to considering all possible cases. Consider, for example, node $p \in \mathcal{V}$. Three cases are possible:

- $p \in S, \bar{p} \in T, x_p = 0$. Then $\theta_p(x_p) = c_{pt} + c_{s\bar{p}} = 0$.
- $p \in T, \bar{p} \in S, x_p = 1$. Then $\theta_p(x_p) = c_{sp} + c_{\bar{p}t} = 0$.
- $p, \bar{p} \in S$ or $p, \bar{p} \in S$, so $x_p = \varnothing$. In each case $\theta_p(0) = c_{pt} + c_{s\bar{p}} = 0$, $\theta_p(1) = c_{sp} + c_{\bar{p}t} = 0$.

The last two equations in (6) for edge $(p,q) \in \mathcal{E}$ can be verified in a similar way. □

Computing solutions $\mathbf{x}^{\min}$ and $\mathbf{x}^{\max}$ In general, graph G may have several minimum cuts (S,T) yielding solutions $\mathbf{x}$ with different number of labeled nodes. As discussed in section 2.1, there exist "extreme" cuts $(S^{\min}, T^{\min})$ and $(S^{\max}, T^{\max})$ such that for any other minimum cut (S,T) there holds $\text{dom}(\mathbf{x}^{\min}) \subseteq \text{dom}(\mathbf{x}) \subseteq \text{dom}(\mathbf{x}^{\max})$ where $\mathbf{x}^{\min}, \mathbf{x}^{\max}, \mathbf{x}$ are the labelings defined by these cuts. Below we review how these cuts can be computed. We assume that $G = (V, A, c)$ is a residual graph satisfying the relaxed symmetry condition (5) in which there are no augmenting paths from s to t, i.e. a maximum flow has been found.

Computing $\mathbf{x}^{\min}$ Let $S^\circ \subset V$ be the set nodes reachable from s via non-saturated arcs, and $T^\circ \subset V$ be the set of nodes from which t can be reached via non-saturated arcs. It is well-known that S°, T° are disjoint, and $(S^\circ, V - S^\circ)$, $(V - T^\circ, T^\circ)$ are the *minimal* and the *maximal* minimum cuts, i.e. minimum cuts with the smallest and the largest source component, respectively. Both of these cuts define the labeling $\mathbf{x}^{\min}$ [5]. Indeed, property (7) and the fact $|S^\star| = |V| - |S|$ imply that $((S^\circ)^\star, (V - S^\circ)^\star)$ is the *maximal* minimum cut, so $V - T^\circ = (S^\circ)^\star$. Thus, for node $u \in V$ we have $(u \in S^\circ) \Leftrightarrow (\bar{u} \in T^\circ)$. This implies that $(S^\circ, V - S^\circ)$, $(V - T^\circ, T^\circ)$ define the same labeling $\mathbf{x}^{\min}$.

The "extremality" of $\mathbf{x}^{\min}$ can be easily verified. Suppose node p has label $x_p^{\min} = 0$, i.e. $p \in S^\circ, \bar{p} \in T^\circ$. Then $p \in S^\circ \subseteq S, \bar{p} \in T^\circ \subseteq T$ for any other minimum cut (S,T), so labeling $\mathbf{x}$ defined by (S,T) satisfies $x_p = 0$. The case $x_p^{\min} = 1$ is analogous.

For a proof of the strong autarky and persistency properties of $\mathbf{x}^{\min}$ we refer to [5].

Computing $\mathbf{x}^{\max}$ Let us show that partial labeling $\mathbf{x}^{\max}$ with the maximum number of labeled nodes can be computed from G *without restoring the symmetry condition*. Consider the following algorithm:

- Compute set $\hat{V} = V^{\text{int}} - S^\circ - T^\circ$. Take $\hat{A}$ to be the set of non-saturated arcs $(u \to v) \in A$, $u, v \in \hat{V}$.
- Compute strongly connected components in $(\hat{V}, \hat{A})$, contract them to single nodes.
- Run a topological sort algorithm on the obtained directed acyclic graph. The result is an ordering of nodes $\pi : \hat{V} \to \mathbb{Z}$ such that for all arcs $(u \to v) \in \hat{A}$ there holds $\pi(u) < \pi(v)$ (unless u and v belong to the same strongly connected component, in which case $\pi(u) = \pi(v)$).
- Extend π to all nodes in V: set $\pi(u) = +\infty$ if $u \in S^\circ$, and $\pi(u) = -\infty$ if $u \in T^\circ$.
- Set cut $(S^{\max}, T^{\max})$ as follows: if $\pi(u) \geq \pi(\bar{u})$ then $u \in S^{\max}$, otherwise $u \in T^{\max}$. The corresponding partial labeling $\mathbf{x}^{\max}$ is determined as follows: (i) If $\pi(p) > \pi(\bar{p})$ then $x_p = 0$. (ii) If $\pi(p) < \pi(\bar{p})$ then $x_p = 1$. (iii) If $\pi(p) = \pi(\bar{p})$ then $x_p = \varnothing$.

If the symmetry property is satisfied, then this procedure is equivalent to the method in [2]. Let us show that the algorithm works correctly if the relaxed symmetry condition holds.

Lemma 8. *Define c' and f as in proposition 6.*

(a) If $c_a > 0$ for arc $a = (u \to v) \in A^{st}$ then $\pi(u) \leq \pi(v)$.
(b) If $f_a \neq 0$ for arc $a = (u \to v) \in A$ then $\pi(u) = \pi(v)$.
(c) If $c'_a > 0$ for arc $a = (u \to v) \in A^{st}$ then $\pi(u) \leq \pi(v)$.

Proof. Part (a) follows from the construction of mapping π. Together with part (b) it implies part (c) since for arc $a \in A^{st}$ either $c_a = c'_a$ or $f_a \neq 0$. Let us prove part (b).

By the flow decomposition theorem [1], circulation f can be written as $f = f^1 + \ldots, f^r$ where each f^j is a flow

in a simple cycle and for all arcs $a \in A$ elements $f_a^1, \ldots, f_a^r$ are either all non-positive or all non-negative. Consider arc $a \in A$ with $f_a < 0$. Arc a must belong to a cycle defined by one of the flows f^j. Suppose that the cycle contains arcs $a_1, a_2, \ldots, a_k$ with $f_{a_i} < 0$, $a_i = (u_i \to u_{i+1})$, $u_{k+1} = u_1$. We claim that $\pi(u_1) = \ldots = \pi(u_k)$. Indeed, for arc $a_i \in A^{st}$ we have $\frac{1}{2}(c_{\bar{a}_i} - c_{a_i}) = f_{a_i} < 0$, $c_{\bar{a}_i} \geq 0$, therefore $c_{a_i} > 0$. Now consider four possible cases:

- The cycle does not contain s, t. Then all arcs a_i in the cycle belong to A^{st}, so $c_{a_i} > 0$. Using part (a), we obtain $\pi(u_1) \leq \ldots \leq \pi(u_k) \leq \pi(u_1)$, which implies our claim.
- The cycle contains s but not t. By renaming indexes we can ensure that $u_1 = s$. All arcs a_i except a_k belong to A^{st}, so all nodes u_i in the cycle can be reached from s via non-saturated arcs. Thus, $\pi(u_i) = +\infty$.
- The cycle contains t but not s. By renaming indexes we can ensure that $u_k = t$. All arcs a_i except a_k belong to A^{st}, so t can be reached from all nodes u_i in the cycle via non-saturated arcs. Thus, $\pi(u_i) = -\infty$.
- The cycle contains s, t. Then there is a path from s to t containing arcs $a_i \in A^{st}$. Arcs in this path are non-saturated, which contradicts to the fact that we have a maximum flow in G.

□

Now let us prove that $(S^{\max}, T^{\max})$ is a minimum cut in (V, A, c') (and, thus, in (V, A, c)). Suppose that $u \in S^{\max}, v \in T^{\max}$, $(u \to v) \in A$. We need to show that $c'_{uv} = 0$. Suppose this is not the case: $c'_{uv} = c'_{\bar{v}\bar{u}} > 0$. Then we arrive at a contradiction:

$$\pi(u) \geq \pi(\bar{u}) \geq \pi(\bar{v}) > \pi(v) \geq \pi(u)$$

(This first inequality holds since $u \in S^{\max}$, the third one holds since $v \in T^{\max}$, the second and fourth follow from part (c) of lemma 8.)

It remains to show the "extremality" of the corresponding solution $\mathbf{x}^{\max}$. Suppose that node p is not labeled in $\mathbf{x}^{\max}$, i.e. $\pi(p) = \pi(\bar{p})$. This means that $p, \bar{p} \in \hat{V}$ and $p, \bar{p}$ belong to the same strongly connected component in $(\hat{V}, \hat{A})$, i.e. there exists non-saturated paths from p to $\bar{p}$ and from $\bar{p}$ to p. By the Ford-Fulkerson theorem a minimum cut (S, T) cannot separate p and $\bar{p}$, so p is unlabeled by any minimum cut.

Appendix B: Implementational details

In this section we describe details of our implementation. We use an adjacency list representation for storing the graph. Nodes are stored in a single array so that there is a constant memory shift between a node and its mate. A similar scheme is used for arcs. This allows to obtain efficiently the mate of a given node or arc without using additional pointers. The source and sink are not stored explicitly; instead, one number per node c_u is stored. With our data structure, removing arc $(u \to v)$ takes $O(\deg(u))$ time where $\deg(u)$ is the degree of node u. Removing node v takes $O(\sum_{(u \to v) \in A^{\text{int}}} \deg(u))$ time. In vision applications nodes typically have a small degree, so such operations are quite efficient.

Let us describe details of "fix" and "contract" operations in QPBOP. As discussed in Appendix A, we need to update the graph and residual capacities so that the relaxed symmetry condition (5) is preserved and the reparameterization given by (4) gives the desired energy. We will use the following operation for arc $(u \to v) \in A^{\text{int}}$:

$$\begin{array}{ll} c_u := c_u - \delta, & c_{uv} := c_{uv} - \delta, \\ c_{vu} := c_{vu} + \delta, & c_v := c_v + \delta \end{array} \tag{8}$$

This operation can be viewed as sending a flow in G between the terminals, possibly from t to s. Therefore, it preserves invariant (5) and corresponds to a reparameterization of the energy. When pushing such a flow, we will always ensure that capacities of all arcs in A^{int} stay non-negative.

We consider fixing node p to only one of the labels, say 0. (Fixing p to label 1 can be reduced to this operation by "flipping" p, i.e. swapping the meaning of 0 and 1. Flipping simply means renaming nodes $p \leftrightarrow \bar{p}$.)

Fixing node $p \in \mathcal{V}$ to label 0 This is done by adding arcs $(s \to p)$ and $(\bar{p} \to t)$ with a large weight C, or setting

$$c_p := c_p + C, \quad c_{\bar{p}} := c_{\bar{p}} - C$$

After that flow is pushed in G using operations (8) so that all arcs $(p \to v), (\bar{v} \to \bar{p}) \in A^{\text{int}}$ become saturated (i.e. their residual capacities become zeros), c_p becomes positive, and $c_{\bar{p}}$ becomes negative. Clearly, these conditions can always be ensured if C is sufficiently large.

It is easy to verify by induction that from now no augmenting path from s to t will go through nodes $p, \bar{p}$. (Indeed, there are no non-saturated outgoing arcs from p other than to s, and there no non-saturated incoming arcs to $\bar{p}$ other than from t). Therefore, removing nodes $p, \bar{p}$ together with incident edges will not affect the output of QPBO. This implies our claim that adding a hard constraint term $E_p(x_p)$ and removing term θ_p along with incident pairwise terms θ_{pq} are equivalent operations.

Contracting nodes $p, q \in \mathcal{V}$ First, all arcs involving node q are reassigned to p (e.g. arc $(q \to u)$ is replaced with the arc $(p \to u)$ with the same weight), and arcs involving node $\bar{q}$ are reassigned to node $\bar{p}$. This includes, in particular, arcs from s and to t; in other words, we set $c_p := c_p + c_q$, $c_{\bar{p}} := c_{\bar{p}} + c_{\bar{q}}$. After that we scan arcs $(p \to u) \in A$ and do the following:

- Suppose there are two arcs from p to u where $u = r$ or $u = \bar{r}$ for some node $r \in \mathcal{V} - \{p\}$. Then there must be

three other pairs of parallel arcs (from u to p, from $\bar{u}$ to $\bar{p}$, and from $\bar{p}$ to $\bar{u}$). For each such pair one of the arcs is removed and its capacity is added to the other arc.

These operations are equivalent to adding two pairwise terms θ_{pr} and θ'_{pr} which are either both submodular or both supermodular.

- Suppose there are arcs $(p \to r)$ and $(p \to \bar{r})$ where $r \in \mathcal{V} - \{p\}$. Then we need to merge a submodular and a supermodular term. First, we push flow in G using operations (8) to ensure that

$$c_{pr} = c_{\bar{r}\bar{p}}, c_{p\bar{r}} = c_{r\bar{p}} \tag{9}$$

(this will imply that $c_{rp} = c_{\bar{p}\bar{r}}, c_{\bar{r}p} = c_{\bar{p}r}$ because of (5b)). These arcs now contribute to the energy one submodular term $\theta^1_{pr} = [0, 2c_{pr}, 2c_{qr}, 0]$ and one supermodular term $\theta^2_{pr} = [2c_{p\bar{r}}, 0, 0, 2c_{\bar{r}p}]$. (We use the following convention for pairwise terms: $\theta_{pr} = [\theta_{pr;00}, \theta_{pr;01}, \theta_{pr;10}, \theta_{pr;11}]$.) We compute the sum $\theta_{pr} = \theta^1_{pr} + \theta^2_{pr}$ and remove arcs between $\{p, \bar{p}\}$ and $\{r, \bar{r}\}$. Invariant (5a) will still hold because of condition (9).

The final step is to convert term θ_{pr} into a normal form (thus, unary terms θ_p and θ_r may appear) and add two arcs for each non-zero component of $\theta_p, \theta_{pr}, \theta_r$, as described e.g. in [18].

- Suppose there is an arc $(p \to u)$ where $u \in \{p, \bar{p}\}$; then there is a corresponding arc $(\bar{u} \to \bar{p})$. These arcs appeared because there was an edge $(p, q) \in \mathcal{E}$ before p and q were contracted. First, we send flow in G using operations (8) to ensure that $c_{pu} = c_{\bar{u}\bar{p}}$. If $u = \bar{p}$ then we set $c_p := c_p + (c_{p\bar{p}} - c_{\bar{p}p})$, $c_{\bar{p}} := c_{\bar{p}} + (c_{p\bar{p}} - c_{\bar{p}p})$. Finally, we remove arcs $(p \to u), (\bar{u} \to \bar{p})$.

This can be justified as follows. Suppose that before contraction the arcs contributed term θ_{pq}. (We assume that the symmetry condition holds for arcs corresponding to (p, q)). Label $x_p = i$ in the new energy corresponds to the labels $x_p = x_q = i$ in the old energy ($i \in \{0, 1\}$). Therefore we need to add unary term θ_p where $\theta_{p;0} = \theta_{pq;00}$, $\theta_{p;1} = \theta_{pq;11}$. This is done by setting $c_p := c_p + \frac{1}{2}(\theta_{p;0} - \theta_{p;1})$, $c_{\bar{p}} := c_{\bar{p}} - \frac{1}{2}(\theta_{p;0} - \theta_{p;1})$. It remains to check that $\theta_{p;0} = \theta_{p;1} = 0$ if $u = p$, and $\theta_{p;0} = 2c_{p\bar{p}}, \theta_{p;1} = 2c_{\bar{p}p}$ if $u = \bar{p}$.

Adding directed constraints To add constraint $(x_p = 0) \Rightarrow (x_q = 0)$ for an existing edge $(p, q) \in \mathcal{E}$, we add arcs $(p \to q)$, $(\bar{q} \to \bar{p})$ with a large capacity C representing the term $[0, 2C, 0, 0]$, and then merge it with the existing term as described above. An important practical question is how to recognize whether a constraint has already been added, and how to choose a capacity which would not cause an overflow but would be sufficiently large to enforce a "hard" constraint. We use the following approach. First, we select a constant C_0 which is larger than the maximum capacity in the graph. After probing node p, we check whether the current term θ_{pq} satisfies

$$\theta_{pq;01} + \theta_{pq;10} - \theta_{pq;00} - \theta_{pq;11} \geq C_0 \tag{10}$$

If not, the constraint has not been added yet; then we add a directed constraint with the value of C so that an equality would hold in (10). (This could happen at most once for this particular constraint, since the expression in (10) is invariant to reparameterization).

These operations, however, do not guarantee that we really enforce the "hard" constraint. To get such a guarantee, after updating the graph we compute a maximum flow in G and check arcs $(p \to q)$, $(\bar{q} \to \bar{p})$. If at least one of them is saturated then we add C_0 to their residual capacities, and continue with the probing algorithm.

"Monotonicity" property Let us show the "monotonicity" property mentioned in Section 3. Let $\mathbf{x} = \text{QPBO}(E)$ where E is the original energy. Assume for simplicity of notation that $x_p = 0$ for all nodes $p \in \text{dom}(\mathbf{x})$ (this condition can be ensured by flipping variables). Since QPBO is invariant to reparameterization, we also assume without loss of generality that before fix/contract operations we had a maximum flow in graph G, i.e. there was no path from s to t via non-saturated arcs $a \in A$ with $c_a > 0$.

Consider cut $(S^\circ, V - S^\circ)$ where $S^\circ = \{p \in \text{dom}(\mathbf{x})\} \cup \{s\}$. According to the discussion in Appendix A, the following properties hold:

(a) All nodes in S° can be reached from the source via non-saturated arcs.

(b) All arcs from $S^\circ - \{s\}$ to $V - S^\circ$ are saturated.

(By the Ford-Fulkerson theorem, (b) follows from the fact that $(S^\circ, V - S^\circ)$ is a minimum cut). "Fix" and "contract" operations are applied only to currently unlabeled nodes; therefore, properties (a,b) still hold after the graph is modified. Note that the set of nodes in the new graph may change; however, nodes in S° are preserved.

It follows from (a,b) that an augmenting path from s to t cannot contain nodes in $S^\circ - \{s\}$. Therefore, pushing flow from s to t will not violate (a,b). Using a straightforward induction, we obtain that after a maximum flow is computed properties (a,b) still hold.

Let $(S^{\circ\prime}, V - S^{\circ\prime})$ be the minimal minimum cut in the new graph which defines the strongly persistent solution $\mathbf{x}'$. It follows from (a) that $S^\circ \subseteq S^{\circ\prime}$, which implies the monotonicity property: $x'_p = x_p$ for all nodes $p \in \text{dom}(\mathbf{x})$.

Appendix C: proof of theorem 4

One direction of the theorem follows immediately from proposition 2: if $\mathbf{x}$ is optimal then it must be stable. Assume

now that $\mathbf{x}$ is stable. In this section $\bar{\mathcal{S}} = \mathcal{V} - \mathcal{S}$ denotes the complement of subset $\mathcal{S} \subseteq \mathcal{V}$, and $\mathbf{x}[\bar{\mathcal{S}}] : \bar{\mathcal{S}} \rightarrow \{0,1\}$ is the restriction of $\mathbf{x}$ to $\bar{\mathcal{S}}$. The definition of stability implies the following useful fact:

Proposition 9. *Let $E' = E[\mathcal{S} \leftarrow \mathbf{x}]$ be the energy obtained by fixing nodes in $\mathcal{S} \subseteq \mathcal{V}$ to values in $\mathbf{x}$, and let $\mathbf{x}' = \mathbf{x}[\bar{\mathcal{S}}]$. If $\mathbf{x}$ is stable for E, then $\mathbf{x}'$ is stable for E'.*

Case (a) First we consider the case when energy E has a unique global minimum. Let us run QPBOI with the empty set $\mathcal{S}$, i.e. compute the strongly persistent solution $\mathbf{y} = \text{QPBO}(E)$. We claim that the graph constructed by QPBO has a unique minimum cut (S,T). Indeed, let us flip a subset of nodes of function E so that it becomes submodular. Analyzing the graph constructed in the QPBO method one can see that it consists of two independent networks. The first one contains nodes $\{p \mid p \in \mathcal{V}\} \cup \{s,t\}$ and represents energy E, while the second network contains nodes $\{\bar{p} \mid p \in \mathcal{V}\} \cup \{s,t\}$ and represents energy E where all nodes are flipped. Therefore, each network (and thus the whole graph) has the unique minimum cut.

Thus, partial labelings $\mathbf{y}$ and $\mathbf{y}^{\max}$ defined by cuts $(S^{\min}, T^{\min}) = (S^{\max}, T^{\max})$ are the same, so by property [$\mathcal{P}3'$] $\mathbf{y}$ is a complete labeling. Since $\mathbf{x}$ is stable, there holds $\mathbf{x} = \text{FUSE}(\mathbf{x}, \mathbf{y}) = \mathbf{y}$ so $\mathbf{x}$ a global minimum of E.

Now consider a more general case when E may have multiple global minima. Let us choose an optimal solution $\mathbf{x}^*$ which maximizes $|\{p \in \mathcal{V} \mid x^*_p = x_p\}|$. Let $\mathcal{S} = \{p \in \mathcal{V} \mid x^*_p = x_p\}$, and consider energy $E' = E[\mathcal{S} \leftarrow \mathbf{x}]$. It is easy to see that E' has a unique global minimum $\mathbf{x}^*[\bar{\mathcal{S}}]$. (Indeed, if $\mathbf{x}^{**}[\bar{\mathcal{S}}]$ is another global minimum of E' with $x^{**}_p = x^*_p$ for $p \in \mathcal{S}$ then $\mathbf{x}^{**}$ is a global minimum of E and produces a larger set $\{p \in \mathcal{V} \mid x^{**}_p = x_p\}$.) Furthermore, $\mathbf{x}[\bar{\mathcal{S}}]$ is stable for E'. As we have shown above, this implies that $\mathbf{x}[\bar{\mathcal{S}}]$ is optimal for E', and thus $\mathbf{x}$ is optimal for E.

Case (b) We can assume without loss of generality that there are no edges (p,q) with $\theta_{pq} = \mathbf{0}$ (they can be removed without affecting the LP relaxation and, thus, the definition of stability). Let $\mathbf{x}^*$ be a global minimum of E, and let $\mathcal{S} = \{p \in \mathcal{V} \mid x^*_p = x_p\}$. We will prove below that energy $E' = E[\mathcal{S} \leftarrow \mathbf{x}]$ does not have frustrated cycles. Theorem 4(a) will then imply that $\mathbf{x}[\bar{\mathcal{S}}]$ is optimal for E' and, thus, $\mathbf{x}$ is optimal for E.

Lemma 10. *Consider path P in graph $\mathcal{G}$ from node p to node q whose nodes lie in $\bar{\mathcal{S}}$. Let $N(P)$ be the number of non-submodular edges in P. Then $N(P) = x_q - x_p \pmod 2$.*

Proof. We will use induction on the length of the path $|P|$. The base case ($|P| = 0$, $p = q$) is trivial. Suppose that the lemma holds for all paths of length $k \geq 0$, and consider path $P' = (p, \ldots, q, r)$ of length $k+1$ obtained by appending edge (q,r) to path P. Since $q, r \in \bar{\mathcal{S}}$ we have $x^*_q = 1 - x_q$, $x^*_r = 1 - x_r$. Costs of labelings $\mathbf{x}, \mathbf{x}^*$ is smaller than C, therefore

$$\theta_{qr}(x_q, x_r) = \theta_{qr}(x^*_q, x^*_r) = \theta_{qr}(1-x_q, 1-x_r) = 0. \quad (11)$$

Two cases are possible:

- $x_q = x_r$. From (11) we get $\theta_{qr}(0,0) = \theta_{qr}(1,1) = 0$. Thus, edge (q,r) is submodular, so
$$N(P') = N(P) = x_q - x_p = x_r - x_p \pmod 2.$$
- $x_q \neq x_r$. From (11) we get $\theta_{qr}(0,1) = \theta_{qr}(1,0) = 0$. Thus, edge (q,r) is non-submodular, so
$$N(P') = N(P) + 1 = x_q - x_p + 1 = x_r - x_p \pmod 2.$$

□

It follows from the lemma that every cycle whose nodes lie in $\bar{\mathcal{S}}$ contains an even number of non-submodular terms, i.e. energy E' does not have frustrated cycles, as claimed.

Appendix D: proof of theorem 5

Let $(\mathcal{P})$ be a procedure which computes whether a given labeling $\mathbf{x}$ for energy E is stable or not. Using a polynomial number of calls to $(\mathcal{P})$ we can determine a set $\mathcal{S} \subset \mathcal{V}$ which decreases the energy, assuming that labeling $\mathbf{x}$ is not stable. Indeed, we can use the following greedy method. Start with the empty set $\mathcal{S}$. Pick a node $p \in \mathcal{V}$, fix it to label x_p and run $(\mathcal{P})$ for the new energy. If labeling $\mathbf{x}$ is still not stable then p is added to set $\mathcal{S}$. Otherwise other nodes are tested. Clearly, the method terminates in a polynomial number of steps and produces the desired set $\mathcal{S}$.

It is clear that the decision problem in theorem 5 belongs to the class co-NP: if the answer is negative (i.e. $\mathbf{x}$ is not stable) then there exists a certificate (subset $\mathcal{S}$) which can be used for verifying in polynomial time that $\mathbf{x}$ is indeed not stable. Thus, in order to prove the theorem we need to show that a certain NP-hard problem can be solved via a polynomial number of calls to $(\mathcal{P})$.

We consider the minimum vertex cover (VC) problem which is NP-hard. The instance of VC is given by an undirected graph $(\mathcal{V}, \mathcal{E})$. Set $\mathcal{S} \subseteq \mathcal{V}$ is called a *vertex cover* if for each edge $(p,q) \in \mathcal{E}$ at least one of the nodes p, q is in $\mathcal{S}$. The goal is to compute a vertex cover of minimum cardinality. Set $\mathcal{S}$ can be described by labeling $\mathbf{x} : \mathcal{V} \rightarrow \{0,1\}$ such that $x_p = 1$ if $p \in \mathcal{S}$. The following energy function E represents the objective function of VC: $\theta_{p;1} = 1$, $\theta_{pq;00} = |\mathcal{V}| + 1$, all other entries are 0.

Let us use the following method for minimizing E. We start with solution $\mathbf{x} = (1, \ldots, 1)$ whose cost is $|\mathcal{V}|$. Using procedure $(\mathcal{P})$ we test whether $\mathbf{x}$ is stable or not. If not, we compute a corresponding subset $\mathcal{S}$ and apply QPBOI

method with subset $\mathcal{S}$. It gives labeling $\mathbf{x}$ with smaller energy. We repeat this procedure until convergence. Clearly, there are at most $|\mathcal{V}|$ steps.

Upon convergence we obtain a stable labeling $\mathbf{x}$. By theorem 4 such a labeling is a global minimum of E. Thus, we have solved the VC problem.

References

[1] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin. *Network Flows: Theory, Algorithms, and Applications*. Prentice Hall, 1993.

[2] B. Aspvall, M. F. Plass, and R. E. Tarjan. A linear time algorithm for testing the truth of certain quantified boolean formulas. *Inform. Process. Lett*, 8:121–123, 1979. Erratum, Inform. Process. Lett. 14(4), 195 (1982).

[3] J. E. Beasley. Heuristic algorithms for the unconstrained binary quadratic programming problem. Technical report, Management School, Imperial College, London, UK, 1998.

[4] H. Bodlaender. Discovering treewidth. In *Conf. Current Trends in Theory and Practice of Informatics (SOFSEM)*, pages 1–16, 2005.

[5] E. Boros and P. L. Hammer. Pseudo-boolean optimization. *Discrete Applied Mathematics*, 123(1-3):155 – 225, 2002.

[6] E. Boros, P. L. Hammer, and X. Sun. Network flows and minimization of quadratic pseudo-Boolean functions. Technical Report RRR 17-1991, RUTCOR, May 1991.

[7] E. Boros, P. L. Hammer, and G. Tavares. Local search heuristics for unconstrained quadratic binary optimization. Technical Report RRR 9-2005, RUTCOR, Feb. 2005.

[8] E. Boros, P. L. Hammer, and G. Tavares. Preprocessing of unconstrained quadratic binary optimization. Technical Report RRR 10-2006, RUTCOR, Apr. 2006.

[9] Y. Boykov and M.-P. Jolly. Interactive graph cuts for optimal boundary and region segmentation of objects in N-D images. In *ICCV*, 2001.

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

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

[12] D. Cremers and L. Grady. Learning statistical priors for efficient combinatorial optimization via graph cuts. In *ECCV*, 2006.

[13] A. W. Fitzgibbon, Y. Wexler, and A. Zisserman. Image-based rendering using image-based priors. In *ICCV*, 2003.

[14] W. Freeman, E. Pasztor, and O. Carmichael. Learning low-level vision. *IJCV*, 40(1):24–57, 2000.

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

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

[17] V. Kolmogorov. Convergent tree-reweighted message passing for energy minimization. *PAMI*, 28(10):1568–1583, Oct. 2006.

[18] V. Kolmogorov and C. Rother. Minimizing non-submodular functions with graph cuts - a review. *PAMI*, 2007. To appear. Online version at http://research.microsoft.com/ ˜carrot.

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

[20] E. Mortensen and W. Barrett. Intelligent scissors for image composition. *SIGGRAPH*, 1995.

[21] A. Raj, G. Singh, and R. Zabih. MRF's for MRI's: Bayesian reconstruction of MR images via graph cuts. In *CVPR*, 2006.

[22] A. Raj and R. Zabih. A graph cut algorithm for generalized image deconvolution. In *ICCV*, 2005.

[23] C. Rother, S. Kumar, V. Kolmogorov, and A. Blake. Digital tapestry. In *CVPR*, 2005.

[24] 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. In *ECCV*, May 2006.

[25] M. Szummer and Y. Qi. Contextual recognition of hand-drawn diagrams with conditional random fields. In *9th Intl. Wkshp. Frontiers in Handwriting Recognition*, 2004.