Approximate MRF Inference Using Bounded Treewidth Subgraphs

Alexander Fix[1], Joyce Chen[1], Endre Boros[2], and Ramin Zabih[1]

[1]Cornell University, Computer Science Department, Ithaca, New York
{afix,yuhsin,rdz}@cs.cornell.edu
[2]Rutgers University, RUTCOR, New Brunswick, New Jersey
boros@rci.rutgers.edu

Abstract. Graph cut algorithms [9], commonly used in computer vision, solve a first-order MRF over binary variables. The state of the art for this NP-hard problem is QPBO [1, 2], which finds the values for a subset of the variables in the global minimum. While QPBO is very effective overall there are still many difficult problems where it can only label a small subset of the variables. We propose a new approach that, instead of optimizing the original graphical model, instead optimizes a *tractable sub-model*, defined as an energy function that uses a subset of the pairwise interactions of the original, but which for which exact inference can be done efficiently. Our Bounded Treewidth Subgraph (k-BTS) algorithm greedily computes a large weight treewidth-k subgraph of the signed graph, then solves the energy minimization problem for this subgraph by dynamic programming. The edges omitted by our greedy method provide a per-instance lower bound. We demonstrate promising experimental results for binary deconvolution, a challenging problem used to benchmark QPBO [2]: our algorithm performs an order of magnitude better than QPBO or its common variants [4], both in terms of energy and accuracy, and the visual quality of our output is strikingly better as well. We also obtain a significant improvement in energy and accuracy on a stereo benchmark with 2nd order priors [5], although the improvement in visual quality is more modest. Our method's running time is comparable to QPBO.

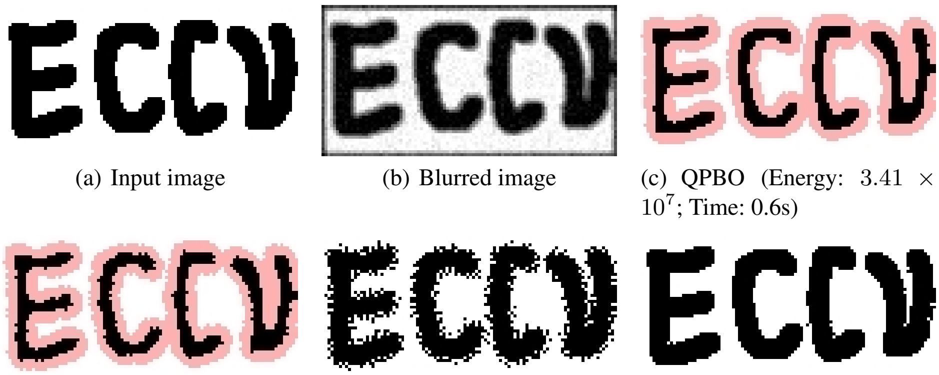
1 Introduction

Graph cuts, which are a popular method for solving first-order Markov Random Fields [9, 6], use max flow to solve an MRF over binary variables. Recall that first-order, binary MRFs are those which can be written as an energy function of the form:

$$f(x_1, \ldots, x_n) = \theta_{\text{const}} + \sum_i \theta_i(x_i) + \sum_{(i,j)\in E} \theta_{i,j}(x_i, x_j) \tag{1}$$

where the $x_i \in \{0, 1\}$. These MRFs are completely specified by their set of variables x_i, together with the unary costs θ_i and graph of pairwise interactions, $\theta_{i,j}$ for $(i, j) \in E$.

This is an NP-hard problem [7], but a technique commonly known as QPBO [1, 2] has proven to be very effective in practice. QPBO finds the correct values for a subset of the variables in the global minimum, which are called persistencies [1]. However,



(a) Input image

(b) Blurred image

(c) QPBO (Energy: 3.41×10^7; Time: 0.6s)

(d) QPBOP (Energy: 3.13×10^7; Time: 2.8s)

(e) QPBOI (Energy: 3.12×10^6; Time: 0.7s)

(f) 2-BTS (Energy: 3.82×10^5; Time 0.6s)

Fig. 1. Example image from deconvolution results (see section 4 for details). Pixels left unlabeled by QPBO (40.7% of the image) and QPBOP (38.3%) are shown in pink. 0.35% of pixels in the 2-BTS result and were wrong, compared to 8.54% for QPBOI.

QPBO is not guaranteed to return any persistencies at all, and the circumstances under which QPBO gives good results are currently not well understood.

Despite the fact that inference in MRFs is in general NP-hard, it is well known that there are families of graphs for which NP-hard graph problems become tractable [18]. We focus on low treewidth graphs, since effecient exact inference can be performed by dynamic programming provided the graph of pairwise interactions has low treewidth.

Unfortunately, for graphs that appear in computer vision, the treewidth is quite large (for grid graphs, it is $\Omega(\sqrt{n})$) [20]. In this case we cannot apply the exact-inference dynamic program, as the resulting algorithm would have exponential running time. Instead, we take the approach of representing the original graphical model with a simpler, tractable model, defined over a subgraph of the original. In particular, we are looking for a subset of edges, $E' \subseteq E$, for which we can exactly optimize the energy function

$$f'(x_1, \ldots, x_n) = \theta_{\text{const}} + \sum_i \theta_i(x_i) + \sum_{(i,j) \in E'} \theta_{i,j}(x_i, x_j). \tag{2}$$

Since f' uses a subset of the pairwise interactions of the original f, we will call it a sub-model of the original graphical model. If exact inference in f' can be performed efficiently (which for our purposes means its graph has low treewidth), we will refer to it as a *tractable sub-model*. The key idea of this paper is how best to find and use tractable sub-models to optimize first-order MRFs.

In order for our sub-model to be useful, we would like it to represent the original energy as faithfully as possible, so that the optimum for the simpler energy function is still close to the optimum for the original, difficult to optimize function. In particular, we will show that by using the bi-form representation of [3] to assign weights to pairwise interaction, the more total weight the subgraph includes, the closer the optimum of the tractable sub-model will be to the true optimum of the original f.

Consequently, as the key intermediate step in our optimization algorithm, we would like to solve the following problem: given a weighted graph, find a subgraph which has treewidth at most k, but has as large a weight as possible over all subgraphs of treewidth k. This is known as the Maximum Bounded-Treewidth Subgraph problem. This is itself an NP-hard problem [22]; however, an approximation will suffice. We give a fast greedy algorithm for this problem, which we use to find tractable sub-models which represent the original energy functions well in practice.

The overall structure of our k-BTS algorithm (k-Bounded-Treewidth Subgraph) is:

- Begin with an energy function in the form of (1).
- Assign weights to the pairwise interactions using the bi-form representation.
- Apply our greedy algorithm to find a subgraph of treewidth k and of large weight.
- Exactly solve the inference problem in the subgraph using dynamic programming.
- Using a proof of lower-bound, argue that the solution for the subgraph is a good approximation to the optimum for the original energy.

The remainder of the paper is structured as follows: We begin with a review of existing graph cut algorithms for solving first-order MRFs, including a discussion of low-treewidth methods in general MRFs, and the bi-form representation and its associated signed graph. Our new algorithms are given in section 3. Experimental results and comparisons are given in section 4, with more data and examples in the supplementary material; particularly exciting results are obtained on difficult binary deconvolution problems, as illustrated in figure 1.

2 Related work

2.1 Graph cuts and QPBO

Graph cut methods solve energy minimization problems by constructing a graph and computing the minimum cut. The most popular graph cuts methods, such as the expansion move algorithm of [9], repeatedly solve an optimization problem over binary variables. Such problems have been extensively studied in the operations research community, where they are referred to as pseudo-Boolean optimization [10] (see [1] for a more detailed survey).

Minimizing an arbitrary energy function is NP-hard (see e.g. [7]). Certain binary optimization problems can be solved exactly by min cut, such as computing the optimal expansion move [9]. The most important class of optimization problems that can be solved exactly with graph cuts are quadratic pseudo-boolean functions (QPBF) in which every term is submodular [1, 11, 7]. Such a function can be written as a polynomial in the binary variables $(x_1, \ldots, x_n) \in \mathbb{B}^n$ in the form $\sum_i a_i x_i + \sum_{i,j} a_{i,j} x_i x_j$. The function is submodular exactly when $a_{i,j} < 0$ for every i, j.

The most widely used technique for minimizing binary energy functions with non-submodular terms relies on the roof duality technique of [12] and the associated graph construction [1]. This method, known as QPBO [2], uses min cut to compute the global minimum for any submodular energy function and for certain other functions as well (see [1, 2] for a discussion). Even when QPBO does not compute the global minimum, it provides a partial optimality guarantee called persistency [12]; QPBO computes a partial assignment which, when applied to an arbitrary complete assignment, will never cause the energy to increase. Such a partial assignment gives the variables it labels their values in the global minimum. Efficient techniques for computing persistencies, along with generalizations and implementations were proposed in [1, 13, 2]; most notable are the probing and improving methods of [4]. Probing extends the persistencies to cover more variables, while improving tries to reduce the energy of any input labeling. We will compare our methods with the work of [4] in the experimental results.

2.2 Relationship to junction trees

The idea of using dynamic programming on low treewidth graphs has appeared before in the machine learning literature. In the context of graphical models, if the entire graph has low treewidth, then the Junction Tree Algorithm (JTA) efficiently computes marginals and conditionals for the variables in the graphical model [19]. JTA is typically stated as a message-passing algorithm, but it is essentially equivalent to a dynamic program, similar to the one described in Lemma 3. However, in the case of vision problems, most inputs are either a grid or grid-like, and thus have treewidth at least $\Omega(\sqrt{n})$ [20], making JTA intractable due to the exponential dependence on treewidth.

Several papers have taken a similar approach to ours, by attempting to find a low treewidth subgraph of the entire model, and then performing exact inference on this approximation to the original problem. [21] considers the problem of finding low treewidth approximations to weighted *hypergraphs*, where each clique in the hypergraph has a weight representing the mutual information among a set of variables. They provide a constant fraction approximation algorithm; however, when applied to the special case where the input is just a weighted *graph* (and not a weighted hypergraph, i.e., cliques of size > 2 have weight zero) then their algorithm essentially computes a maximum spanning tree of the graph, which is subsumed by the $k = 1$ case of our algorithm.

The most relevant comparison to our work comes from [22], which considers the same formulation as our problem: given a weighted graph, find the maximum weight subgraph of treewidth k. They construct a tree decomposition by finding a good separator, and recursively finding tree decompositions of each half of the partition. However, to do so, they solve up to k large-sized linear programs (with variables for each vertex and edge). Consequently, it is unlikely that their algorithm scales as well as the simple greedy approach taken by this paper. While their paper reports running times of a few minutes, their examples are vastly smaller than those that arise in computer vision.

2.3 The bi-form representation and its signed graph

In order to assign weights to the pairwise terms, it will be convenient to use the bi-form representation of [3]. Bi-forms have several connections to other combinatorial and computer vision problems, which we will describe below. A bi-form is written in terms of the *relational variables* $y_{i,j}$ and $\overline{y}_{i,j}$. These are functions of the original variables x_i

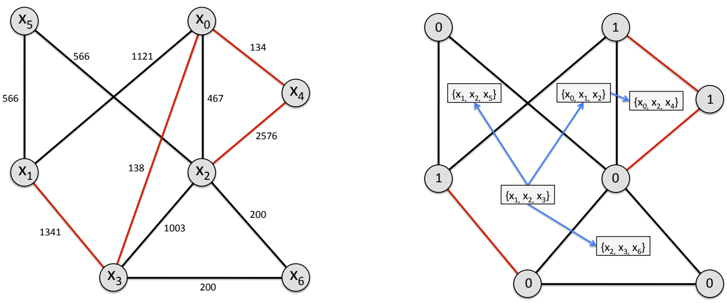


Fig. 2. Left: The bi-form graph from the posiform in equation (7). Black edges are positive (attractive), red edges are negative (repulsive). **Right:** The maximum weight treewidth 2 subgraph, with tree decomposition in rectangular nodes. Labels on the vertices are the optimal labeling for this tree decomposition.

and x_j, and indicate whether x_i and x_j are equal or not. That is, $y_{i,j} = 1$ iff $x_i \neq x_j$ while $\overline{y}_{i,j} = 1$ iff $x_i = x_j$. These functions can be written in terms of the x_i and x_j as:

$$y_{i,j} \equiv x_i x_j + \bar{x}_i \bar{x}_j \qquad \overline{y}_{i,j} \equiv x_i x_j + \bar{x}_i \bar{x}_j \tag{3}$$

Definition 1 (Bi-form). *A bi-form is a sum of relational variables with positive coefficients*

$$\phi = \sum_{\{i,j\} \in E^+} c_{i,j} y_{i,j} + \sum_{\{i,j\} \in E^-} c_{i,j} \overline{y}_{i,j} \tag{4}$$

where E^+, E^- are sets of pairs $\{i, j\}$, and $c_{i,j} > 0$. Since $y_{i,j} + \overline{y}_{i,j} = 1$, we will make the simplifing assumption that no pair $\{i, j\}$ is in both E^+ and E^-.

[3] shows that for any QPBF $f(x_1, \dots, x_n)$, we can write it as a biform by adding a single additional variable x_0 (representing the constant value $x_0 = 1$), by first applying the transformations

$$x_i x_j \Longrightarrow \frac{1}{2}[y_{i,j} + (x_i + x_j) - 1], \quad -x_i x_j \Longrightarrow \frac{1}{2}[\overline{y}_{i,j} - (x_i + x_j)] \tag{5}$$

to the quadratic terms, and then applying the transformations

$$x_i \Longrightarrow y_{0,i}, \quad -x_i \Longrightarrow \overline{y}_{0,i} - 1 \tag{6}$$

to the linear terms of the multilinear representation of f.

Note that, by substituting in the identities of (3) into the definition of a bi-form (4), we get back a QPBF. Since the transformations are algebraic equalities, the biform is just a reparameterization of the original energy. However, in this form, every $\theta_{i,j}$ can be specified by just two values, a non-negative weight $c_{i,j}$ (giving the coefficients in (4)) and a sign ± 1, according to whether (i, j) is in E^+ or E^-. We call such a graph, where the edges come in two kinds, a *signed graph*. [8] noted that quadratic binary functions have a unique signed graph representation arising from the bi-form.

For concreteness, we can illustrate this with an example QPBF, taken from the residual network after running QPBO on one of our test cases:

$$\begin{aligned} &549x_1x_3 \ +2133\overline{x_1}\overline{x_3} \ +237x_1\overline{x_5} \ +895\overline{x_1}x_5 \ +273x_2\overline{x_3} \ +1733\overline{x_2}x_3 \\ +\ &2710x_2x_4 \ +2442\overline{x_2}\overline{x_4} \ +895x_2\overline{x_5} \ +237\overline{x_2}x_5 \ +400\overline{x_2}x_6 \ +400x_3\overline{x_6} \end{aligned} \tag{7}$$

We can apply the transformations given above to get the following equivalent bi-form for the same energy function (with the associated signed graph in Figure 2):

$$\begin{aligned} -930 \ &+1121y_{0,1} \ +467y_{0,2} \ +138\overline{y}_{0,3} \ +134\overline{y}_{0,4} \ +1341\overline{y}_{1,3} \ +566y_{1,5} \\ &+1003y_{2,3} \ +2576\overline{y}_{2,4} \ +566y_{2,5} \ +200y_{2,6} \ +200y_{3,6}. \end{aligned} \tag{8}$$

Signed graphs, introduced in [15], have numerous applications and many variations (see e.g., [14]). It was noted early that the problem of *balancing* signed graphs is a equivalent with finding the maximum cut in simple graphs, which in its turn is equivalent with quadratic binary minimization (see e.g., [16]).

Recall that a bi-form is just a reparameterization of a QPBF, where each pairwise term $\theta_{i,j}$ is either 0 or $c_{i,j}$ according to whether the variables x_i and x_j are equal or unequal (depending on the sign of the edge (i, j)). Given a binary labeling of the vertices of a signed graph, we will say that an edge e is *happy* if it agrees with the labels of its two endpoints. More precisely: a positive edge e (resp. negative edge), is happy if and only if its endpoints have the same label (resp. different labels). Otherwise, we say that e is unhappy.

Definition 2. *The* Max Balanced Subgraph *problem is to find a binary labeling of the vertices of a signed graph that minimizes the total weight of unhappy edges.*

It is clear that the Max Balanced Subgraph is equivalent to the optimization of the original QPBF, since according to the reparameterization given by the bi-form, the cost $\theta_{i,j}$ is 0 whenever an edge is happy, and is $c_{i,j}$ whenever it is unhappy. Therefore, the total cost of unhappy edges for any labeling x is precisely the value of the original energy function $f(x)$. Because these problems are equivalent, we will use the language of signed graph balancing wherever it leads to simpler exposition.

Prior work using bi-forms includes [4] who proposed new methods based on QPBO and applied these methods to vision problems. Their paper was the first computer vision publication to use this representation, which allowed them to efficiently implement several post-processing steps that run on the output of QPBO. One such technique is QPBOP, a fast implementation of the probing technique of [3]. They also proposed a novel improve heuristic QPBOI based on the idea of using QPBO iteratively for randomly selected small subsets of the variables.

Our work is directly comparable to [4], and we experimentally compare against their methods in section 4. The most important difference is that while they use bi-forms to make their algorithms more efficient, we use bi-forms to derive new algorithms. As with [4], our techniques are naturally applied as a post-processing step after QPBO.

3 The k-Bounded Treewidth Subgraph algorithm

It is well known that many NP-hard graph problems can be solved in polynomial time by dynamic programming if the graph is "long and thin", under several different technical definitions of this notion such as graph bandwidth, treewidth, etc. [18]. In particular, there is an efficient dynamic program for Maximum Balanced Subgraph on graphs of constant treewidth.

Lemma 1. *Given a tree-decomposition of a graph with treewidth k, the optimal labeling for the Maximum Balanced Subgraph problem can be found by dynamic programming in $O(2^k m)$ time.*

Proof. Recall that a tree-decomposition is a tree whose nodes are cliques in the original graph, and which satisfies the running-intersection property (the set of cliques that contain a vertex v of the original graph form a connected region of the tree). Without loss of generality, we can always add extra cliques to the tree such that each clique C differs from its parent C' by removing one node, and adding another (i.e., $|C| = |C'| = k$ and $|C \cap C'| = k - 1$).

Pick an arbitrary node to be the root, r. We want each edge to be assigned to a unique clique, so we say that e belongs to the clique C that fully contains it and which is closest to the root. This C is unique by the running-intersection property.

For each clique C in the tree decomposition, maintain a table $T(C, x)$ of size $2^{|C|}$, indexed by all binary labelings x of the vertices in C. Initialize each entry to be the cost for the corresponding labeling of the clique, just for those edges that "belong" to C.

Now, working from the leaves up to the root, do the following: At a clique C, for every child C' in the tree, update $T(C, x) \leftarrow \min_{x'} T(C, x) + T(C', x')$. The minimum is computed over all labelings x' of C' that agree with x on $C \cap C'$. Since C and C' differ by adding a node and removing another, this minimum is only over two choices. Additionally, if C has multiple children, their intersection is contained in C, so independent choices for the minimum in each merge will still give a consistent labeling for the whole graph.

The initialization takes time at most $O(2^k m)$, since for each edge, we possibly add its weight to each entry in the table of the clique it belongs to. Each merge step takes time proportional to $O(2^k)$, and we must do one merge step for each edge in the tree decomposition, of which there are at most n. So, overall the entire algorithm is $O(2^k m)$. □

We can thus solve the problem exactly for low-treewidth graphs, but the running time quickly grows infeasible for large k. To get around this limitation, our proposed algorithm, k-BTS computes a k Bounded-Treewidth Subgraph of the original graph, and then solves this exactly by dynamic programming.

An example is given in Figure 2 where we compute the the largest weight treewidth 2 subgraph. Note that the nodes x_0, x_1, x_2, x_3, x_5 have K_4 as a minor, and thus can't be contained in any treewidth 2 subgraph. The tree decomposition found by the greedy algorithm of Section 3.1 is shown in the figure. Note that the only edge it is missing happens to be the the lightest edge in the forbidden subgraph, so in this case it is the maximum over all subgraphs of treewidth 2. This need not be the case in general. Furthermore, since the optimal labeling for the subgraph agrees with the only edge left out (since (x_0, x_3) wants its endpoints to take different labels) this is actually an optimal labeling of the entire graph.

3.1 The greedy subgraph algorithm

The main issue is how to compute the low treewidth approximation. We have found that a simple greedy approach works quite well for vision applications. Our greedy technique for finding a large weight subgraph of treewidth k proceeds as follows.

First, we want to find a large weight clique C of size $k+1$ to be our root of the tree-decomposition:

– Pick an arbitrary node v, and initialize $C \leftarrow \{v\}$.
– For any subset $S \subset V$, define *BestExtension*(S) to be the vertex v that is not currently part of the subgraph, and which has the highest combined weight of edges from v to vertices in S.
– While $|C| < k+1$, add the best extension, $C \leftarrow$ *BestExtension*(C).

Make C the root node of the tree decomposition. Now, we will repeatedly add child cliques to cliques already in the tree decomposition, until our subgraph spans the entire graph.

As noted in the description of the dynamic program above, we can assume that if C is a child of C' in the tree decomposition, then we can get C by removing a vertex of C' and adding a new one.

So, while our tree decomposition doesn't yet cover all nodes: find the best over all cliques C in the tree decomposition, and over all v in C of *BestExtension*$(C-v)$, call it v'. We then add $C - v + v'$ as a child of C in the tree decomposition.

Lemma 2. *The running time of the greedy algorithm is* $O(km\log(m))$

Proof. We defer the full proof to the supplementary material. The basic idea is that we maintain a priority queue of all possible extensions, and at each step pull off the one with largest weight. By eliminating strictly dominated choices (i.e. extensions of less weight than another extension covering the same new nodes), there are only m possible extensions we have to put in this queue, each of which takes time $O(k\log(m))$ to find, for a total time of $O(km\log(m))$. □

As discussed in the experimental section, we get strong results using small choices of k. Two special cases are worth discussing, namely $k=1$, which involves a maximum spanning tree, and $k=2$ which uses triangulation.

3.2 Maximum spanning tree ($k = 1$)

Recall that treewidth 1 graphs are the same as trees. In the case of $k=1$, the above greedy algorithm reduces to Prim's algorithm for computing the maximum spanning tree.

To give some intuition on why choosing a spanning tree is a reasonable idea, note that the dynamic program for finding the optimal labeling on a *tree* is particularly simple. Simply label the root to be 1 or 0 arbitrarily, and then do depth first search in the tree. For every vertex v with parent p, if the edge (p,v) is a positive edge, we give v the same label as its parent. If it's a negative edge, we give it the opposite label. Since this is the unique optimal labeling for the tree (up to flipping all the labels simultaneously), we'll say that the spanning tree induces this labeling on the graph.

Lemma 3. *For any signed graph, there exists a spanning tree that induces an optimal labeling for the maximum balanced subgraph problem.*

Proof. Consider the optimal labeling x, and let E' be the set of happy edges in this labeling. We assume that the original graph (V, E) is connected, otherwise apply this argument to each connected component separately. By way of contradiction, assume the subgraph (V, E') is disconnected, and take any connected component A. Note that according to the current labeling, $\delta(A)$, the set of edges with one endpoint in A, only contains unhappy edges. Flip all the labels in A. This doesn't change the happiness of any edge entirely in A or entirely in $V - A$, but all the edges in $\delta(A)$ have had exactly one endpoint flipped, so they all become happy.

Since the original graph was connected, $\delta(A)$ is nonempty, so we have strictly reduced the total cost of unhappy edges, contradicting the assumption that x was optimal. Therefore, (V, E') is connected, so take any spanning tree of (V, E') and it will induce the optimal labeling. □

The maximum spanning tree encourages heavy edges to be selected, while selectively ignoring lighter edges to find a low-energy labeling of variables and relations. By choosing the maximum spanning tree, we essentially select the highest weight set of edges that care the most about their labelings.

3.3 Graph triangulation ($k = 2$)

The greedy algorithm for $k = 2$ is similar to Prim's, but instead of adding the maximum weight *edge* out of the current subgraph, we add the maximum weight *triangle*. Overall, the algorithm is as follows:

- Find the maximum weight edge out of the starting vertex v. Add it to the subgraph.
- While the subgraph doesn't span the entire graph, add the biggest triangle you can find, such that the base of the new triangle was already an edge in the graph. For the purpose of simplicity of description, assume we have a complete graph where all non-existent edges have weight 0.

The algorithms for higher k are similar, but instead of adding triangles to the graph, we add the biggest "cone", defined as a vertex v which is connected to k other vertices, such that these k vertices are all contained together in some clique of the tree decomposition we've constructed so far.

3.4 The k-BTS lower bound

An important measure of how well this algorithm is doing is how much total weight ends up in the subgraph. Intuitively, if we capture more weight in the subgraph, then the optimal solution we find on this subgraph will better approximate the costs in the original graph. Note that this argues for larger values of k, which of course trade off against the higher computational costs.

We can formalize this intuition by providing a lower bound on the optimal solution. For a labeling $x : V \to \{0, 1\}$, let $G(x)$ be the cost of this labeling in the original graph, and let $G'(x)$ be the cost just according to the edges in the low-treewidth subgraph. Let W be the total weight of edges not included in the subgraph.

Lemma 4. *If x^* is any optimal labeling of the original graph G, and x' is any optimal solution to the subgraph G', then $G(x') - W \leq G(x^*) \leq G(x')$*

Proof. For any labeling x we have $G'(x) \leq G(x)$, since there are fewer edges in the subgraph to be unhappy with x. We also have $G(x) - W \leq G'(x)$, since in the worst case, we have to pay for every single edge we've left out to get the extra cost of $G(x)$ over $G'(x)$.

Therefore, since x^* and x' are optimal for G and G' respectively, $G(x^*) \geq G'(x^*) \geq G'(x') \geq G(x') - W$ (the middle inequality holds because x' is the optimal for G').

Therefore, we've bounded the true optimal cost by $G(x') - W \leq G(x^*) \leq G(x')$. Overall, the less weight W we have to leave out of the subgraph, the tighter this bound and the closer our solution x' must be to the optimal solution. □

4 Experimental results

We primarily compare these methods against the common extensions of QPBO, namely QPBOP and QPBOI from [4]. QPBOP is the most computationally intensive variant and labels the most variables. QPBOI takes in an arbitrary labeling of the variables, and using the persistencies found from repeated flow calculations, improves this solution to one with at least as good energy. In the following experiments, QPBOI refers to running Improve on the all-zeros solution, which is the default behavior in the QPBO code.

We also compare against two other non-graph-cuts methods, ICM and loopy BP. These methods were also compared against in [4], and were generally inferior to QPBOP and QPBOI; however, ICM had reasonable performance on their binary deconvolution dataset, which we see repeated here. Belief Propagation was done with sequential updates, ordered with maximum marginals. As an implementation we used libDAI [23].

	Binary deconvolution		Stereo reconstruction	
Method	Energy	Time	Energy	Time
QPBO	999	21.5	218	6.8
QPBOP	991	128.4	–	–
QPBOI	94	92.3	179	17.6
ICM	5.8	1.20	999	4.9
BP	453	397.47	–	–
1-BTS	0.12	21.5	91.1	10.0
2-BTS	0.43	26.6	29.5	17.9
6-BTS	2.4	27.0	19.6	32.2
2-BTS+I	0	108.4(16.1)	0	27.0(9.4)

Fig. 3. Following [4] energy values are scaled so that highest energy seen by any method is 999, and the lowest is 0. The entries omitted for QPBOP and BP took too long to complete (see Section 4.2). For 2-BTS+I we report the running time due purely to 2-BTS in parentheses. All timings include the initial running of QPBO, and are measured in seconds on a 2-year old desktop PC.

We have focused on difficult optimization problems where QPBO by itself fails to label a significant number of pixels. We compare QPBOI and QPBOP with our k-BTS method. Like QPBOI and QPBOP, k-BTS is most naturally run after QPBO, which is usually quite fast, and has a possibility of splitting the problem into multiple smaller subproblems.

Finally, since QPBOI can take an *arbitrary* labeling of the variables, and attempt to improve it, we can also take the result of running BTS first and then applying QPBOI to this solution. We'll denote this approach by k-BTS+I. A summary of the numerical data for all experiments is in Figure 3. Additional examples and more details are provided in the supplementary material. All code and data for these experiments will be available at `rgb.cs.cornell.edu/projects`.

4.1 Binary Deconvolution

Our first dataset for comparison uses a deconvolution algorithm based on [24]. The image data comprised 10 images in the Brodatz texture data, thresholded at varying intensities to get black and white images. Each was then convolved with a 3×3 kernel and independent noise applied at each pixel to obtain blurred images, as in Figure 4.

Reconstructing the original non-blurry image requires optimizing a quadratic binary function with high connectivity: each pixel is involved in pairwise terms with the 24 other variables in a 5×5 box. Additionally, all the pairwise terms between pixels are nonsubmodular, with no submodular terms at all. Previously, it was observed in [6] that energy functions with a high degree of nonsubmodularity are very difficult to optimize.

We observed that QPBO and QPBOP managed to label very few of the variables in these energy functions (13.3% and 13.6% respectively). Overall, the BTS-based algorithms performed very similiarly in terms of final energy, with results significantly lower than existing methods, as seen in the table above. For the example image in Figure 4, the final energy for the BTS images were more than an order of magnitude lower than the QPBOI result, and over 2 orders of magnitude lower than QPBO and QPBOP.

Note that while the true global optima for these energy functions are unknown, by visual inspection of the resulting images are very close to the original, un-blurred images. In fact, the results for 2-BTS differed from the original in 5.1% of pixels, versus 25.6% for QPBOI, while taking only 28% as long to compute. Other BTS-based algorithms yielded similar accuracy.

For additional confirmation, we also used a small set of 4 hand-drawn text images, similar to Figure 1. The average energy for the 2-BTS was 6 times better than the QPBOI result, and 58 times better than the result for QPBOP (average energy values of 7.5×10^5, 4.4×10^6 and 4.4×10^7 respectively). Furthermore, the results for 2-BTS differed from the original image in 0.15% of pixels, compared to 4.5% for QPBOI.

4.2 Stereo reconstruction

The stereo reconstruction method of [5] uses fusion moves to solve a second-order curvature prior. We experimented with their "SegPln" proposal method, which generates QBPFs which are difficult for QPBO to optimize, with an average of 66% of variables

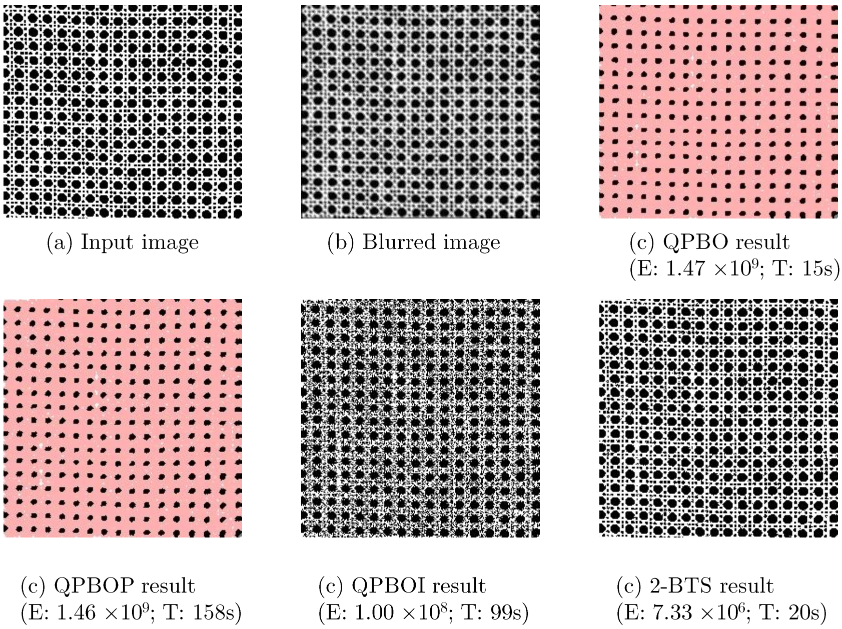

(a) Input image
(b) Blurred image
(c) QPBO result (E: 1.47 $\times 10^9$; T: 15s)
(c) QPBOP result (E: 1.46 $\times 10^9$; T: 158s)
(c) QPBOI result (E: 1.00 $\times 10^8$; T: 99s)
(c) 2-BTS result (E: 7.33 $\times 10^6$; T: 20s)

Fig. 4. Deconvolution results based on [2] benchmark. Pixels left unlabeled by QPBO (85.9% of the image) and QPBOP (84.9%) are shown in pink. 1.1% of pixels in the 2-BTS result and were wrong, compared to 21.5% for QPBOI.

labeled per problem. QPBOP performed so poorly on this dataset as to be impractical. Most QPBOP executions lasting longer than 1200 seconds, at which point the experiment was terminated. In one instance, running QPBOP to completion was observed to take 14,000 seconds. Note that this is just to compute a single fusion move. Similarly, all but one execution of BP failed to converge in 1200 seconds.

Overall, the BTS algorithms all acheived significantly better results than QPBO or QPBOI. To see the additional improvement of these methods, we compare to the improvement of QPBOI over QPBO. Relative to this difference, the algorithms BTS-1, BTS-2 and BTS-6 reduced the energy on average by an additional 11%, 66% and 85% respectively. The best result, by the hybrid algorithm 2-BTS+I reduced the energy 113% over the baseline, in less than double the time ($\times 1.82$).

To compare the effects of running a single optimization algorithm through an entire run of fusion move, we ran the algorithm to convergence with each method. The best BTS algorithm, 2-BTS+I, produced the image in Figure 5.

5 Conclusions and Extensions

It is interesting to note the relationship between increasing k and the overall performance of the k-BTS algorithms in the summary table at the beginning of this section. For the stereo reconstruction experiments, increasing the width of the subgraph found

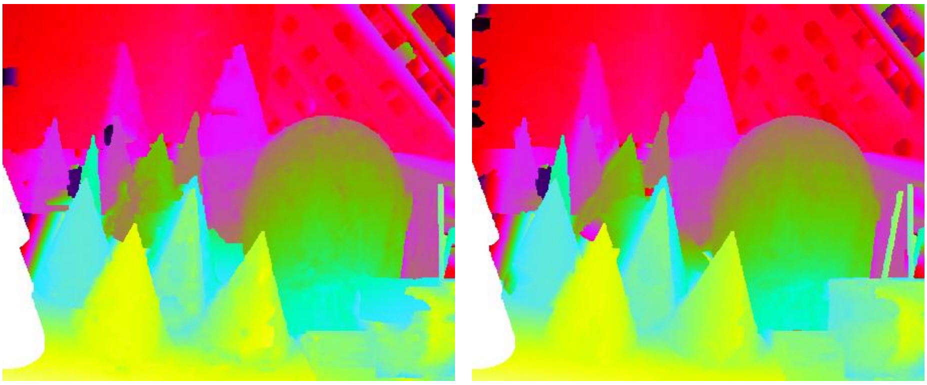

(a) QPBOI. E: 1.172×10^8; Correct pixels: 1,416 (4,586 within ± 1 disparity)
(b) 2-BTS+I. E: 1.167×10^8; Correct pixels: 2,525 (7,804 within ± 1 disparity)

Fig. 5. Stereo reconstruction results running fusion moves to convergence. 2-BTS+I took fewer iterations (8 vs. 20 for QPBOI) and slightly less time time (420 seconds vs. 450 seconds). The visual results are similiar, but note that the BTS result is consistently smoother than the QPBOI result, especially in the reconstruction of the mask, and the yellow cone in the foreground. 2-BTS-I had substantially better accuracy with respect to the ground truth.

reduced the final energy signficantly, between 1-BTS and 6-BTS. This aligns with our expectations that higher treewidth subgraphs can cover more of the original energy, and thereby get better results.

For deconvolution, the results are less clear (though it's worth noting that the differences between varying k are much less than the difference with QPBOI). At the very least, these results suggest that there is room for improvement in choosing a smarter algorithm for finding bounded-treewidth subgraphs.

For future work, we are currently investigating more complicated for pricing cliques than the greedy algorithm of section 3.1. In particular, we are examining whether a primal-dual approach to greedily choosing cliques can give subgraphs with larger weight, especially for increasing k.

Additionally, the idea of using dynamic programming to optimize a low-treewidth subgraph can be applied to give a local search algorithm with a very large neighborhood size. If the entire graphical model is covered by a collection of low treewidth subgraphs, each can be independently optimized (holding the rest of the graph fixed) until no improvement can be made. For grid-graphs, these subgraphs can be chosen as bands, or small patches of the image. However, for general graphs, it remains a topic of open research how best to pick what low-treewidth subgraphs to use.

Acknowledgements: This research has been supported by National Science Foundation grants IIS-0803705/0803444 and IIS-1161860/1161476.

References

1. Boros, E., Hammer, P.L.: Pseudo-boolean optimization. Discrete Applied Mathematics **123** (2002)

2. Kolmogorov, V., Rother, C.: Minimizing nonsubmodular functions with graph cuts-a review. TPAMI **29** (2007) 1274–1279 Earlier version appears as technical report MSR-TR-2006-100.
3. Boros, E., Hammer, P., Sun, R., Tavares, G.: A max-flow approach to improved lower bounds for quadratic $0-1$ minimization. Discrete Optimization **5** (2008) 501–529 Also appeared as 2006 RUTCOR technical report.
4. Rother, C., Kolmogorov, V., Lempitsky, V., Szummer, M.: Optimizing binary MRFs via extended roof duality. In: CVPR. (2007)
5. Woodford, O., Torr, P., Reid, I., Fitzgibbon, A.: Global stereo reconstruction under second-order smoothness priors. TPAMI **31** (2009) 2115–2128
6. Szeliski, R., Zabih, R., Scharstein, D., Veksler, O., Kolmogorov, V., Agarwala, A., Tappen, M., Rother, C.: A comparative study of energy minimization methods for Markov Random Fields. TPAMI **30** (2008) 1068–1080
7. Kolmogorov, V., Zabih, R.: What energy functions can be minimized via graph cuts? TPAMI **26** (2004) 147–59
8. Boros, E., Hammer, P.: A max-flow approach to improved roof-duality in quadratic $0-1$ minimization. Technical report, RUTCOR (1989)
9. Boykov, Y., Veksler, O., Zabih, R.: Fast approximate energy minimization via graph cuts. TPAMI **23** (2001) 1222–1239
10. Hammer, P., Rudeanu, S.: Boolean Methods in Operations Research and Related Areas. Springer (1968)
11. Hammer, P.: Some network flow problems solved with pseudo-boolean programming. Operations Research **13** (1965) 388–399
12. Hammer, P.L., Hansen, P., Simeone, B.: Roof duality, complementation and persistency in quadratic 0-1 optimization. Mathematical Programming **28** (1984) 121–155
13. Feige, U., Goemans, M.: Approximating the value of two power proof systems, with applications to max 2sat and max dicut. 3rd Israel Symposium on the Theory of Computing Systems **00** (1995) 182
14. Zaslavsky, T.: Signed graphs. Discrete Applied Mathematics **4** (1982) 47–74
15. Harary, F.: On the notion of balance of a signed graph. Michigan Mathematical Journal **2** (1953) 143–146
16. Hammer, P.L.: Pseudo-boolean remarks on balanced graphs. International Series of Numerical Mathematics **36** (1977) 69–78
17. Goemans, M.X., Williamson, D.P.: Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. Journal of the ACM **42** (1995) 1115–1145
18. Garey, M., Johnson, D.: Computers and Intractability. W. H. Freeman and Company (1979)
19. Wainwright, M.J., Jordan, M.I.: Graphical Models, Exponential Families, and Variational Inference. Now Publishers Inc., Hanover, MA, USA (2008)
20. Lipton, R., Tarjan, R.: Applications of a planar separator theorem. SIAM Journal on Computing (1980) 615–627
21. Karger, D.R., Srebro, N.: Learning markov networks: maximum bounded tree-width graphs. In: SODA. (2001) 392–401
22. Shahaf, D., Chechetka, A., Guestrin, C.: Learning thin junction trees via graph cuts. In: Artificial Intelligence and Statistics (AISTATS). (2009)
23. Joris M. Mooij: libDAI: A free & open source C++ library for Discrete Approximate Inference in graphical models. In: Journal of Machine Learning Research, 11(Aug), 2010.
24. Raj, A., Zabih, R.: A graph cut algorithm for generalized image deconvolution. In: International Conference on Computer Vision (ICCV). (2005)