Minimizing non-submodular functions with graph cuts – a review

Vladimir Kolmogorov
University College London
vnk@adastral.ucl.ac.uk

Carsten Rother
Microsoft Research, Cambridge
carrot@microsoft.com

Abstract

Optimization techniques based on graph cuts have become a standard tool for many vision applications. These techniques allow to minimize efficiently certain energy functions corresponding to pairwise Markov Random Fields (MRFs). Currently, there is an accepted view within the computer vision community that graph cuts can only be used for optimizing a limited class of MRF energies (e.g. submodular functions).

In this survey we review some results that show that graph cuts can be applied to a much larger class of energy functions (in particular, non-submodular functions). While these results are well-known in the optimization community, to our knowledge they were not used in the context of computer vision and MRF optimization. We demonstrate the relevance of these results to vision on the problem of binary texture restoration.

Keywords: I.4.6.c Markov Random Fields, I.2.10.i Texture
Index terms: Energy minimization, Markov Random Fields, quadratic pseudo-boolean optimization, min cut/max flow, texture restoration

1 Introduction

Many early vision problems can be naturally formulated in terms of energy minimization where the energy function has the following form:

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

Here $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is an undirected graph. Set $\mathcal{V}$ usually corresponds to pixels; x_p denotes the label of pixel $p \in \mathcal{V}$ which must belong to a finite set of integers $\{0, 1, \ldots, K-1\}$. For motion or stereo, the labels are disparities, while for image restoration they represent intensities. The constant term of the energy is θ_{const}, the unary terms $\theta_p(\cdot)$ encode data penalty functions, and the pairwise terms $\theta_{pq}(\cdot,\cdot)$ are interaction potentials. This energy is often derived in the context of Markov Random Fields [6, 15]: a minimum of E corresponds to a *maximum a-posteriori* (MAP) labeling $\mathbf{x}$.

Minimizing energy (1) is a difficult problem (in general, it is NP-hard). Many approximate optimization methods have been developed, such as the augmenting DAG algorithm [25, 36], simulated annealing [21], iterated conditional modes [7], belief propagation [31], tree-reweighted message passing [35], or Swendsen-Wang Cuts [5]. We will focus on a particular branch of algorithms that are based on *graph cuts*, or the *s*-*t* min cut/max flow technique. They were introduced into computer vision in the late 80-s [17] and reintroduced in the 90-s [12, 20, 24]. Graph cuts proved to be very successful for many vision applications such as small baseline stereo, volumetric multi-view reconstruction, image segmentation, image synthesis and others (see, for example, [11] and references therein).

In this survey we review some graph cut-based algorithms for minimizing energy (1). We consider only the case when the variables are binary: $x_p \in \{0,1\}$. Note, however, that this case is highly relevant for vision problems involving non-binary variables. Indeed, one of the most successful MRF minimization algorithms, namely the expansion move method of Boykov *et al.* [12], reduces the problem with *multi-valued* variables to a sequence of minimization subproblems with *binary* variables[1]. There are also other ways to reduce the problem to a sequence of binary subproblems, e.g. swap move and jump move algorithms [12, 33].

1.1 Minimizing functions of binary variables via graph cuts

There is an accepted view within the computer vision community popularized by [24] that graph cuts can only be used for minimizing *submodular* energy functions, i.e. functions whose pairwise terms satisfy $\quad \theta_{pq}(0,0) + \theta_{pq}(1,1) \le \theta_{pq}(0,1) + \theta_{pq}(1,0)\ .$

For functions of multi-valued variables and the expansion move algorithm the corresponding condition is $\theta_{pq}(\beta,\gamma)+\theta_{pq}(\alpha,\alpha) \le \theta_{pq}(\beta,\alpha)+\theta_{pq}(\alpha,\gamma)$ which must hold for all labels $\alpha,\beta,\gamma \in \{0,\ldots,K-1\}$ [2]. While many important energy functions in vision (e.g. *Potts*) do satisfy these conditions, in some situations we get functions which are not submodular. For example, they may arise when parameters of the energy function are learned from training data.

Rother *et al.* [32] suggested to deal with non-submodular terms by "truncating" them, i.e.

[1]Recall that the expansion move algorithm [12] iteratively applies α-expansion operations for labels $\alpha \in \{0,\ldots,K-1\}$ in a certain order, starting with some initial configuration. If $\mathbf{x}$ is the current configuration, then during an α-expansion each pixel is allowed either to keep its old label x_p or to switch to the new label α. Since each pixel makes a binary decision, computing an optimal α-expansion move (i.e. a move with the smallest energy) is equivalent to minimizing a function of binary variables: 0 corresponds to keeping the old label, and 1 corresponds to the new label α. Upon convergence the algorithm produces a local minimum whose energy is within a known approximation factor from the global minimum (for metric interactions) [12].

[2]There is one exception to this rule: it is also possible to include hard constraints $H_{pq}(x_p,x_q)$ taking values in $\{0,+\infty\}$ as long as $H_{pq}(\alpha,\alpha)=0$ for all labels α [32].

replacing a function with a submodular approximation and minimizing the latter. For multi-valued variables and certain truncation schemes the energy is guaranteed not to increase during one α-expansion [32]. For the application of image stitching [1, 28, 32] this technique gives reasonable results, maybe because the number of non-submodular terms is very small (see details in section 3). For some other problems, however, the number of non-submodular terms is much higher, and truncation may not be appropriate. An example is binary texture restoration as done by Cremers and Grady [14]. They propose to ignore non-submodular terms during MRF learning in order to achieve efficient inference via graph cuts. We show in section 3 that ignoring such terms decreases the performance. It is therefore desirable to have a method which explicitly takes into account non-submodular terms instead of throwing them away.

In this paper we will review the algorithm in [10, 19] for minimizing functions with both submodular and non-submodular terms[3]. We refer to it as the QPBO method ("quadratic pseudo-boolean optimization")[4]. Its output is a *partial* labeling $\mathbf{x}$. In other words, $x_p \in \{0, 1, \varnothing\}$ for pixels $p \in \mathcal{V}$; the value $\varnothing$ is interpreted as "unknown". The algorithm has a number of properties reviewed below. (All of them were given in [19]).

Properties of the QPBO method [19] Perhaps the most important one is the following:

[$\mathcal{P}$1] *(Persistency) Let* $\mathbf{y}$ *be a complete labeling, and let* $\mathbf{z}$ *be the "fusion" of* $\mathbf{x}$ *and* $\mathbf{y}$*:* $z_p = x_p$ *if* $x_p \in \{0, 1\}$*, and* $z_p = y_p$ *otherwise. Then* $E(\mathbf{z}) \leq E(\mathbf{y})$*.*

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

[$\mathcal{P}$2] *(Partial optimality) There exists global minimum* $\mathbf{x}^*$ *of energy* (1) *such that* $x_p = x_p^*$ *for all labeled pixels* p *(i.e. pixels with* $x_p \in \{0, 1\}$*).*

Clearly, the usefulness of the algorithm depends on how many pixels are labeled. In general, we cannot expect that the method will label all nodes since minimizing energy (1) is an NP-hard problem. It is a question of experimentation of how the algorithm performs for a particular application, such as binary texture restoration as discussed in section 3.

In some special cases, however, the method is guaranteed to give a complete labeling:

[3] Algorithms in [19] and [10] compute the same answer, but the latter is more efficient.

[4] Energy (1) can be written as a quadratic polynomial: $E(\mathbf{x}) = const + \sum_p \alpha_p x_p + \sum_{(p,q)} \alpha_{p,q} x_p x_q$, hence the word *quadratic*. E is called *pseudo*-boolean since it maps boolean variables to $\mathbb{R}$ rather than to $\{0, 1\}$.

[$\mathcal{P}$3] *If all terms of the energy are submodular, then the algorithm will label all nodes.*

[$\mathcal{P}$4] *The algorithm is invariant with respect to "flipping" a subset of pixels $\mathcal{U} \subseteq \mathcal{V}$, i.e. swapping the meaning of 0 and 1 for pixels $p \in \mathcal{U}$. (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 [19]. This holds, in particular, for trees.

There remains a question of what to do with unlabeled pixels. This is a difficult question which probably does not have a single answer. In the context of the expansion move algorithm one possibility is to keep the old label for all such pixels. The persistency property then implies that the energy never goes up.

Energy minimization methods and partial optimality It is known [23] that the tree-reweighted message algorithm [35] also gives a part of an optimal solution when applied to functions of binary variables. (This is not surprising: it solves the same *linear programming relaxation* of the energy as the QPBO method). A different principle for obtaining partially optimal solutions for MAP-MRF problems is given by Kovtun [26].

2 QPBO algorithm for functions of binary variables

We will describe the algorithm using the notion of *reparameterization*. This concept is discussed in section 2.1. After that we will review the QPBO method for minimizing functions of eq. (1). For completeness, we first consider the simpler case when all terms of the energy are submodular (section 2.2). Then in section 2.3 we will review the QPBO method for arbitrary functions of the form (1). For both cases the algorithm consists of the following three steps: (i) construct the graph; (ii) compute the maximum flow and minimum cut; (iii) assign labels based on the minimum cut.

2.1 Reparameterization

The term *repameterization* was introduced in the machine learning community [13, 34] (an alternative term for this is *equivalent transformations* [25, 36]). It is a very useful tool for analyzing MRF inference algorithms such as belief propagation [34] and tree-reweighted message passing [35]. Reparameterization is also a convenient interpretation of maxflow-based algorithms (see sections 2.2 and 2.3).

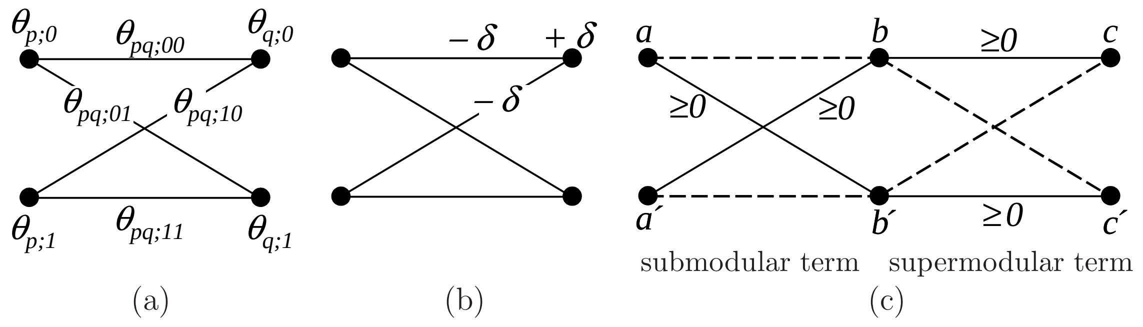


Figure 1: **(a) Convention for displaying parameters θ_p, θ_{pq}, θ_q. (b) Example of a reparameterization operation. (c) Normal form. Dotted lines denote links with zero cost. The first term is submodular, the second is supermodular. Unary parameters must satisfy** $\min\{a, a'\} = \min\{b, b'\} = \min\{c, c'\} = 0$**.**

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 $\mathcal{I}$ 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.

The energy in eq. (1) is therefore completely specified by parameter vector θ. In cases when the parameter vector of an energy is not clear from the context, we will write it explicitly as $E(\mathbf{x} \mid \theta)$. We will display parameters of the energy as shown in Fig. 1(a).

Definition 2.1. *If two parameter vectors θ and θ' define the same energy function (i.e. $E(\mathbf{x} \mid \theta) = E(\mathbf{x} \mid \theta')$ for all configurations $\mathbf{x}$), then θ is called a* reparameterization *of θ', and the relation is denoted by $\theta \equiv \theta'$.*

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 directed edge $(p \to q) \in \mathcal{E}$ 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 $\theta_{p;j}$ (Fig. 1(b)). It is easy to see that the cost of any configuration $\mathbf{x}$ stays the same.

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 pixels p.

(b) $\min\{\theta_{pq;0j}, \theta_{pq;1j}\} = 0$ for all directed edges $(p \to q)$ and labels $j \in \{0, 1\}$.

Note that these conditions imply that all components of vector θ are non-negative (except maybe for the constant term θ_{const}). The normal form is not unique, i.e. many reparameterizations of the same energy function may satisfy (a)-(b).

We will need this definition when we describe the algorithms in sections 2.2 and 2.3. The first step for both algorithms is to convert vector θ into a normal form. This can be done in two phases as follows.

1. While there is edge $(p \to q)$ and label j violating condition (b), do the following: Compute $\delta = \min\{\theta_{pq;0j}, \theta_{pq;1j}\}$; set $\theta_{pq;0j} := \theta_{pq;0j} - \delta$, $\theta_{pq;1j} := \theta_{pq;1j} - \delta$, $\theta_{q;j} := \theta_{q;j} + \delta$.

2. For every pixel p compute $\delta = \min\{\theta_{p;0}, \theta_{p;1}\}$ and set $\theta_{p;0} := \theta_{p;0} - \delta$, $\theta_{p;1} := \theta_{p;1} - \delta$, $\theta_{\text{const}} := \theta_{\text{const}} + \delta$.

Note that every operation above is a reparameterization. Furthermore, each operation in the first phase decreases the number of elements violating condition (b), therefore we are guaranteed to terminate in a finite number of steps (which is in fact linear in $|\mathcal{V}| + |\mathcal{E}|$).

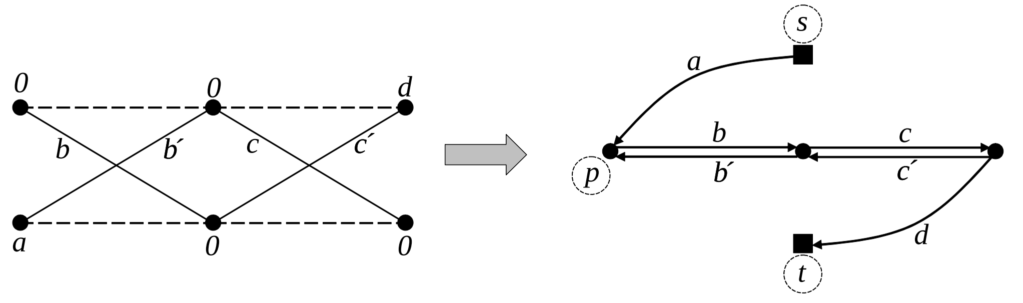
Upon termination, for each edge (p, q) we will have either $\theta_{pq;00} = \theta_{pq;11} = 0$ or $\theta_{pq;01} = \theta_{pq;10} = 0$ (Fig. 1(c)). In the former case the corresponding term is submodular, and in the latter it is supermodular.

2.2 Algorithm for submodular functions

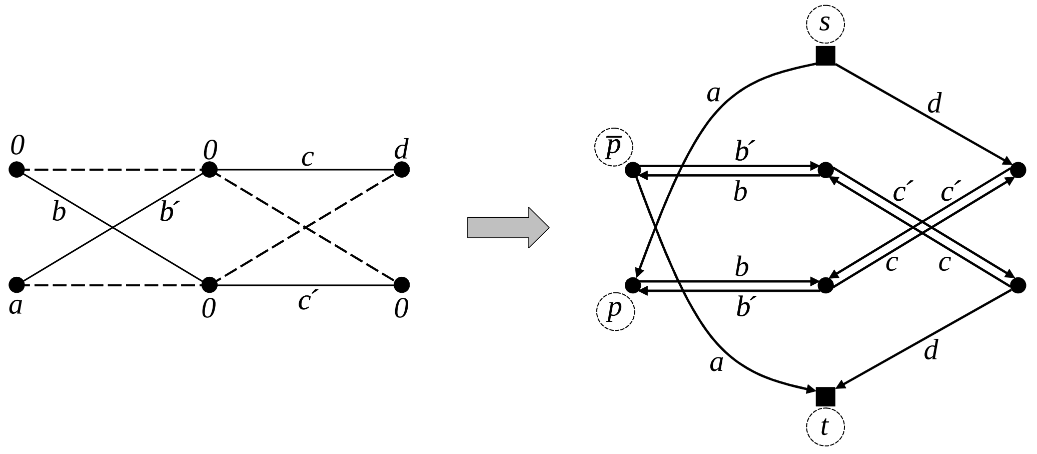
First we will review an algorithm for the case when all terms of energy (1) are submodular. The method (reduction to min cut/max flow) has been known for at least 40 years [18].

The first step is to convert the energy to a normal form (sec. 2.1). Then a directed weighted graph $G = (V, A, c)$ is created whose nodes correspond to pixels in $\mathcal{V}$. In addition, there are two special nodes - source s and sink t; they are called the *terminals*. Thus, $V = \mathcal{V} \cup \{s, t\}$. For every non-zero component of θ an edge is added to A according to the following rules:

component of θ	corresponding edge $a \in A$	capacity c_a
$\theta_{p;0}$	$(p \to t)$	$\theta_{p;0}$
$\theta_{p;1}$	$(s \to p)$	$\theta_{p;1}$
$\theta_{pq;01}$	$(p \to q)$	$\theta_{pq;01}$
$\theta_{pq;10}$	$(q \to p)$	$\theta_{pq;10}$



(a) Construction for submodular functions



(b) Construction for arbitrary functions (QPBO method)

Figure 2: **Graph construction. Left: input energy function (in a normal form) with three nodes and two edges. Right: corresponding graph G. Note that in the example in (b) graph G consists of two disjoint subgraphs. This is because we can "flip" a subset of pixels so that the energy becomes submodular (properties $[\mathcal{P}3, \mathcal{P}4]$). Such flipping can always be done if graph $\mathcal{G}$ is a tree.**

(For example, if $\theta_{p;0} > 0$ then we add edge $(p \to t)$ with weight $\theta_{p;0}$). Note that the constant term θ_{const} is ignored in this construction. These rules are illustrated in Fig. 2(a). After constructing the graph we compute a minimum s-t cut (S, T) by computing maximum flow from the source to the sink [2]. This cut defines configuration $\mathbf{x}$ as follows:

$$x_p = \begin{cases} 0 & \text{if } p \in S \\ 1 & \text{if } p \in T \end{cases} .$$

It can be seen that the cost of any cut is equal to the energy of the corresponding configuration plus the constant term θ_{const}. Therefore, a minimum s-t cut in G yields a global minimum of the energy E.

It is worth noting that the maxflow algorithm can be regarded as performing a reparameterization of the energy function. Indeed, pushing flow updates residual capacities c_{uv} for edges $(u \to v) \in A$ (last column of the table above). This in turn corresponds to modifying components of vector θ. The constant term of the energy changes as well: if we push flow of value C from the source to the sink, then we need to set $\theta_{\text{const}} := \theta_{\text{const}} + C$.

2.3 Algorithm for arbitrary functions: QPBO method

We now review the network model of [10] for solving the problem formulated in [19]. It computes a part of an optimal solution for an arbitrary function of eq. (1). Similar to the previous case, the problem is reduced to the computation of a minimum s-t cut in a certain graph $G = (V, A, c)$. However, the size of the graph is now doubled. For each pixel $p \in \mathcal{V}$ there will be *two* nodes p and $\bar{p}$; therefore, $V = \{p, \bar{p} \,|\, p \in \mathcal{V}\} \cup \{s, t\}$. For every non-zero element of θ two edges are added to A according to the following rules (Fig. 2(b)):

component of θ	corresponding edges $a, \bar{a} \in A$	capacities $c_a = c_{\bar{a}}$
$\theta_{p;0}$	$(p \to t)$, $(s \to \bar{p})$	$\frac{1}{2}\theta_{p;0}$
$\theta_{p;1}$	$(s \to p)$, $(\bar{p} \to t)$	$\frac{1}{2}\theta_{p;1}$
$\theta_{pq;01}$	$(p \to q)$, $(\bar{q} \to \bar{p})$	$\frac{1}{2}\theta_{pq;01}$
$\theta_{pq;10}$	$(q \to p)$, $(\bar{p} \to \bar{q})$	$\frac{1}{2}\theta_{pq;10}$
$\theta_{pq;00}$	$(p \to \bar{q})$, $(q \to \bar{p})$	$\frac{1}{2}\theta_{pq;00}$
$\theta_{pq;11}$	$(\bar{q} \to p)$, $(\bar{p} \to q)$	$\frac{1}{2}\theta_{pq;11}$

After computing a minimum s-t cut (S, T) the partial labeling $\mathbf{x}$ is determined as follows:

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

This labeling $\mathbf{x}$ has the persistency property described in sec. 1 (see [9] or Appendix in [22]).

The graph construction can be motivated as follows [9]. Node $\bar{p}$ can be associated with variable $x_{\bar{p}}$ which ideally should be the negation of x_p: $x_{\bar{p}} = 1 - x_p$. Then the graph represents energy $E(\mathbf{x})$ expressed as a function of old variables $\{x_p\}$ and new variables $\{x_{\bar{p}} = 1 - x_p\}$. Indeed, consider, for example, component $\theta_{pq;11}$ which contributes term $\theta_{pq;11} x_p x_q$ to the energy. The term can be rewritten as $\frac{1}{2}\theta_{pq;11}(x_p \bar{x}_{\bar{q}} + \bar{x}_{\bar{p}} x_q)$, which corresponds to edges $(\bar{q} \to p)$, $(\bar{p} \to q)$ added to the graph. An important observation is that the new energy of variables $\{x_p, x_{\bar{p}}\}$ is submodular, and thus can be minimized in polynomial time. If in addition we could enforce constraints $x_{\bar{p}} = 1 - x_p$ during minimization (i.e. enforcing that nodes p and $\bar{p}$ belong to different sets of the cut) then we would obtain a global minimum of the energy. Without these constraints only a part of an optimal solution is found.

As in the previous case, pushing flow through graph G can be regarded as a reparameterization of energy $E(\mathbf{x} \,|\, \theta)$. More precisely, the residual capacities c in G define the parameter vector θ as follows: if component θ_α corresponds to edges $a, \bar{a} \in A$ then $\theta_\alpha = c_a + c_{\bar{a}}$.

Efficient implementation Computation of maximum flow in graph G can be speeded up using the following heuristics. First consider only edges in G corresponding to submodular terms. These edges form two independent networks such that one is the "reversed" copy of the other, i.e. the source and the sink are swapped and direction of all the edges is reversed. Instead of solving the same problem twice, we can compute a maximum flow in one network and then copy the result to the other network. After that we can add the remaining edges corresponding to non-submodular terms and compute the maximum flow in G, starting from the flow obtained in the first step. An advantage of this heuristics is that if all edges are submodular, then the algorithm has exactly the same running time as the method described in section 2.2, except for a linear time overhead.

Involution property It will be convenient to introduce the following notation. Each node u in graph G has a corresponding node denoted as $\bar{u}$. (In particular, source and sink correspond to each other, so $\bar{s} = t$ and $\bar{t} = s$). It follows from the definition that mapping $^{-} : V \to V$ is an involution, i.e. $\bar{\bar{u}} = u$ for all nodes u. An important property of this involution is that for any edge $a = (u \to v) \in A$ there is a corresponding edge $\bar{a} = (\bar{v} \to \bar{u}) \in A$ with the same initial capacity. We will assume without loss of generality that the *residual* capacities of a and $\bar{a}$ are also the same[5].

Choosing a minimum cut We stated in the beginning that if all terms are submodular then the algorithm will label all nodes (property [$\mathcal{P}3$]). This holds, however, only if we pick a particular minimum cut (S, T). (Recall that a graph may have many minimum cuts of the same cost). We now discuss how to choose a minimum cut that labels as many pixels as possible. This is done by analyzing the residual graph $G = (V, A, c)$ obtained after pushing the maximum flow (and restoring the involution property). We assume that A contains only edges with positive residual capacities, and there are no edges to the source or from the sink.

Consider pixel $p \in \mathcal{V}$, and suppose that there exist paths in graph G from p to $\bar{p}$ and from $\bar{p}$ to p (i.e. p to $\bar{p}$ belong to the same strongly connected component). Then by the Ford-Fulkerson theorem there is no minimum cut (S, T) that separates these nodes. Therefore, node p cannot be labeled. Let $\mathcal{U}^0 \subseteq \mathcal{V}$ be the set of all other pixels (i.e. pixels p such that

[5]The involution property can be restored as follows: given the residual graph with capacities c we compute the corresponding reparameterization θ and then construct a new graph $G = (V, A, \tilde{c})$ for this reparameterization. This is equivalent to setting $\tilde{c}_a = \tilde{c}_{\bar{a}} := \frac{1}{2}(c_a + c_{\bar{a}})$ for all pairs of corresponding edges $a, \bar{a}$.

p and $\bar{p}$ belong to different strongly connected components). It turns out that there exists a minimum cut that labels all pixels in $\mathcal{U}^0$ [4]. This cut and corresponding labeling can be found, for example, by the following algorithm:

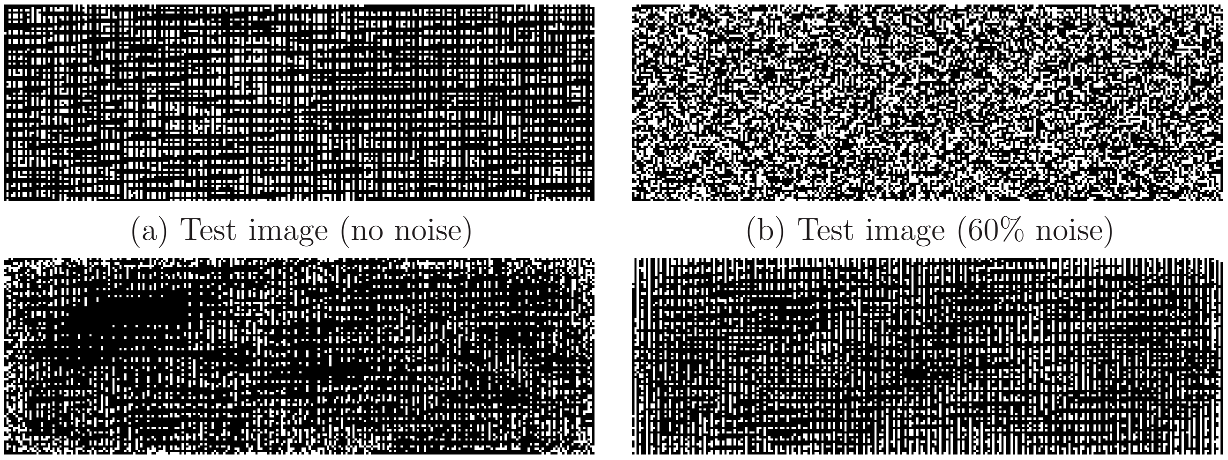
- Add edge $(t \to s)$ to A.
- Compute strongly connected components in G, contract them to single nodes.
- Run the topological sort algorithm on the obtained directed acyclic graph. The result is an ordering of nodes $\pi : V \to \mathbb{Z}$ such that for all edges $(u \to v) \in A$ there holds $\pi(u) < \pi(v)$ (unless u and v belong to the same strongly connected component, in which case $\pi(u) = \pi(v)$).
- Set cut (S, T) as follows: if $\pi(u) \geq \pi(\bar{u})$ then $u \in S$, otherwise $u \in T$. The corresponding partial labeling $\mathbf{x}$ 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 $x_p = \varnothing$.

Note that adding edge $(t \to s)$ does not affect strongly connected components of G since there is no path from s to t. The presence of this edge ensures that $s \in S$ and $t \in T$.

Decomposition into strongly connected components Let us consider a strongly connected component $U \subset V$ such that $u, \bar{u} \in U$ for some node u. The involution property then implies that for any node $v \in U$ there holds $\bar{v} \in U$. Therefore, strongly connected components of G partition the set of pixels $\mathcal{V} \backslash \mathcal{U}^0$ into disjoint regions $\mathcal{U}^1, \ldots, \mathcal{U}^k$.

Recall that the QPBO algorithm does not label pixels in $\mathcal{U}^1 \cup \ldots \cup \mathcal{U}^k$. If the number of unlabeled pixels is small, then we could try to use some other algorithm for the remaining pixels, e.g. exhaustive search. This can be speeded up by the method in [8] reviewed below.

For each region $\mathcal{U}^r$ consider the part of the energy $E(\mathbf{x} \,|\, \theta)$ that involves only pixels and edges inside $\mathcal{U}^r$. (We assume that θ is the reparameterization computed from the residual graph, not the original parameter vector). Suppose that somehow we computed a global minimum $\mathbf{x}^r$ of this part of the energy (note that computing such a minimum is an NP-hard problem). [8] showed that solutions $\mathbf{x}^1, \ldots, \mathbf{x}^k$ can be combined in linear time to obtain a complete optimal solution $\mathbf{x}^*$ for the full energy E. The first step is to modify the vector θ as follows: for each edge (p, q) inside region $\mathcal{U}^r$ set $\theta_{pq}(x_p^r, x_q^r) := 0$. After that we can



(a) Test image (no noise) (b) Test image (60% noise)

(c) Submodular graph cuts (error 18.1) (d) QPBO (error 16.0)

Figure 3: **The task is to restore a noisy test image (b) of a binarized Brodatz texture (D21) (a). The result of submodular graph cuts (c) is worse than QPBO (d), where learning gave** 6 **submodular terms for SGC and** 3 **submodular and** 9 **non-submodular for QPBO.**

run the algorithm described earlier ("Choosing a minimum cut"). It will label all nodes and produce a global minimum of $E(\mathbf{x})$[6].

3 Experimental results

So far, vision researchers have used only the standard graph cuts method for minimizing submodular functions described in section 2.2. (We will refer to it as the SGC method - "submodular graph cuts"). To ensure that the function is submodular, two techniques were employed: (i) enforce submodularity constraint while *learning* parameters of the energy [3,14, 27]; (ii) "Truncate" non-submodular terms during *inference* [32]. A natural idea is to use the QPBO method instead, which means in the first case that we allow arbitrary terms during learning. Here we consider two application areas with non-submodular energies: texture restoration and image stitching. The main conclusion we can draw is that if a substantial amount of pairwise terms are non-submodular the QPBO method outperforms SGC, as for texture restoration. If only a small fraction of the terms are non-submodular both methods produce close to identical results, as for the application of image stitching.

Texture restoration The field of texture restoration and texture modeling has received considerable attention in the past. We refer to [29] for a survey of several MRF models of

[6]Note that this "stitching" of solutions can be applied even if $\mathbf{x}^r$ are approximate rather than global minima for regions $\mathcal{U}^r$.

texture such as auto-binomial, auto-normal, multi-level logistics. Gimel'farb [16] and Zhu *et al.* [37, 38] proposed non-parametric MRF models and methods for their learning. Cremers and Grady [14] focussed on efficient learning and inference via graph cuts for *binary* texture restoration. In this paper we closely follow [14]. We tested MRFs with *pairwise* terms only. This rather simple texture model is probably not the best, but it serves as a good testbed for QPBO (see [14, 37, 38] for texture models with higher-order cliques). Note that QPBO can potentially be extended to more sophisticated models: gray-scale textures can potentially be handled via α-expansion [12], and any higher-order clique can be reduced to pairwise terms [9], although the complexity grows exponentially with the clique size.

We binarized 10 Brodatz textures[7] where fig. 3 shows an example. Each texture image was split into three equally-sized subimages which were used later for training, validation and testing. The same amount of noise was added to all three (60% salt-and pepper noise, i.e. 60% of the pixels are replaced by a random value). We utilized the following MRF from [14]. The unary potentials are defined as $\sum_{p \in \mathcal{V}} \frac{-\lambda}{1+|I_p - x_p|}$ where I_p is the color of pixel p and x_p the binary output label of pixel p. The value λ trades off the importance of the unary versus pairwise potentials and has to be learned for each texture individually as described later. The pairwise potentials for an edge with shift (s_x, s_y) were learned by computing the joint histograms of all pixel pairs with the same shift from the training data: $\theta_{pq}(x_p, x_q) = -\log Pr(x_p, x_q)$. We considered only edges with $\max\{|s_x|, |s_y|\} \leq w$. The neighborhood size w has to be large enough to capture the repetitive structure of the pattern and was set by hand for each texture. In order to avoid over-fitting we learned a subset of pairwise potentials which gave the lowest error rate on a validation image, which has the same noise statistic as a potential test image. As error rate we used the number of misclassified pixels, where unlabeled pixels of the QPBO method are counted as misclassified. The learning is a greedy search: the optimal subset is build by picking sequentially the best pairwise terms, among all terms, where for each potential term a full search over $\lambda \in [0, 100]$ with step-size 0.5 was conducted. Since SGC can only deal with submodular terms we ran the learning procedure twice: (i) over all pairwise terms and (ii) all submodular terms only. This brute-force learning approach took 12 hours per texture on a 3GHerz machine

[7]http://research.microsoft.com/vision/cambridge/i3l/segmentation/DATA/TextureRestoration.zip

and alternative MRF learning approaches could be considered, which is however beyond the scope of this paper. The learning yield on average 6.8 pairwise potentials when restricted to submodular terms only, and 10.1 pairwise potentials without restriction where 34.7% of the terms are non-submodular.

The error rate on the validation set over all textures was 18.28% for SGC and 17.24% for QPBO, and on the test set - 19.96% for SGC and 19.18% for QPBO. (For each texture in the test set we generated 20 instances of random noise). Fig. 3(c,d) shows one example (all results available online[7]). It is worth mentioning that the optimal setting for QPBO was often close to a "critical" settings, i.e. reducing λ by a small amount results in many unlabeled pixels on the validation image. It happened, for 3 textures, that QPBO reconstructed test images with many unlabeled pixels. We used an automated procedure which increased λ by a small amount until most pixels of the test images are fully labeled.

To summarize, for texture restoration it is important to model non-submodular terms and QPBO can handle these terms to a certain extent. A further observation is that an MRF with only submodular terms has a bias towards a solution with uniform labels, i.e. a black or white image. In the absence of unary terms the optimal solution is a uniform labeling. In terms of runtime, the QPBO method was on average about 8 times slower (e.g. 1.8 sec. SGC versus 14.3 sec. QPBO on a 3GHerz machine). Also, as expected the runtime of QPBO is close to identical to SGC if only submodular terms are provided.

Image stitching The problem of image stitching is to merge a set of input images into one output image, e.g. a panoramic view. Several methods [1,28,32] have approached this task as a labeling problem, where each pixel in the output image is assigned a label which corresponds to an input image. Several of the proposed energies (depending on the application) contain non-submodular terms. Since these methods utilize SGC we pose the question if QPBO will improve the results. We have tested the energy defined in [28] (eq. 5) for 5 different scenarios of merging 20 images into one output image (similar to fig. 6 [32]). When running SGC the energy is made submodular by truncating the terms $\theta_{pq}(0, 0)$ (see details in [32]). The percentage of non-submodular terms was on average 0.004% for one alpha-expansion move. The number of differently labeled pixels was between 0% (4 examples) and 6.5% (1 example) and the minimum energy was also close to identical. We may conclude that SGC and QPBO perform very similar when the percentage of non-submodular terms is small.

4 Conclusions and future work

In this survey we reviewed the method in [10,19] for minimizing functions of binary variables with unary and pairwise terms. We believe that this algorithm can make a significant impact for certain vision applications. We demonstrated that the QPBO method improves the results for the binary texture restoration problem. We hope that this positive result will encourage research in other areas such as recognition, e.g. [27], or super resolution, e.g. [30], to exploit energies with non-submodular terms and utilize the QPBO method for inference. (In the past such terms were either disallowed during learning or "truncated" during the inference via graph cuts). It is also interesting to incorporate higher-order cliques of binary variables, as in [14]: it is known that any such clique can be reduced to pairwise terms [9], however, the number of terms grows exponentially with the clique size.

Acknowledgements We thank D. Cremers and L. Grady for communicating the recent paper [14] and providing images, E. Boros for pointing out reference [8], Y. Boykov, V. Lempitsky and anonymous reviewers for suggestions that helped to improve the presentation.

References

[1] A. Agarwala, M. Dontcheva, M. Agrawala, S. Drucker, A. Colburn, B. Curless, D. Salesin, and M. Cohen. Interactive digital photomontage. *SIGGRAPH*, 2004.

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

[3] D. Anguelov, B. Taskar, V. Chatalbashev, D. Koller, D. Gupta, G. Heitz, and A. Ng. Discriminative learning of Markov Random Fields for segmentation of 3D scan data. In *CVPR*, 2005.

[4] 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).

[5] A. Barbu and S.-C. Zhu. Generalizing Swendsen-Wang to sampling arbitrary posterior probabilities. *PAMI*, 27(8):1239–1253, August 2005.

[6] J. Besag. Spatial interaction and the statistical analysis of lattice systems. *J. of the Royal Statistical Society, Series B*, 36:192–236, 1974.

[7] J. Besag. On the statistical analysis of dirty pictures. *J. of the Royal Statistical Society, Series B*, 48(3):259–302, 1986.

[8] A. Billionnet and B. Jaumard. A decomposition method for minimizing quadratic pseudo-boolean functions. *Operation Research Letters*, 8:161–163, 1989.

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

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

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

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

[13] R. G. Cowell, A. P. Dawid, S. L. Lauritzen, and D. J. Spiegelhalter. *Probabilistic Networks and Expert Systems*. Springer-Verlag, 1999.

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

[15] S. Geman and D. Geman. Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images. *PAMI*, 6:721–741, 1984.

[16] G. L. Gimel'farb. Texture modeling by multiple pairwise pixel interactions. *PAMI*, 18(11), 1996.

[17] D. Greig, B. Porteous, and A. Seheult. Exact maximum a posteriori estimation for binary images. *J. of the Royal Statistical Society, Series B*, 51(2):271–279, 1989.

[18] P. L. Hammer. Some network flow problems solved with pseudo-boolean programming. *Operations Research*, 13:388–399, 1965.

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

[20] H. Ishikawa. Exact optimization for Markov Random Fields with convex priors. *PAMI*, 25(10):1333–1336, October 2003.

[21] S. Kirkpatrick, C. D. Gelatt Jr., and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220, 4598:671–680, 1983.

[22] V. Kolmogorov and C. Rother. Minimizing non-submodular functions with graph cuts - a review. Technical Report MSR-TR-2006-100, Microsoft Research, July 2006.

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

[24] V. Kolmogorov and R. Zabih. What energy functions can be minimized via graph cuts? *PAMI*, 26(2):147–159, February 2004.

[25] V. K. Koval and M. I. Schlesinger. Two-dimensional programming in image analysis problems. *USSR Academy of Science, Automatics and Telemechanics*, 8:149–168, 1976.

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

[27] S. Kumar and M. Herbert. Discriminative fields for modeling spatial dependencies in natural images. In *NIPS*, 2004.

[28] V. Kwatra, A. Schödl, I. Essa, G. Turk, and A. Bobick. Graphcut textures: Image and video synthesis using graph cuts. *SIGGRAPH*, July 2003.

[29] S. Z. Li. *Markov Random Field Modeling in Computer Vision*. Springer-Verlag, 1995.

[30] U. Mudenagudi, R. Singla, P. Kalra, and S. Banerjee. Super resolution using graphcut. In *ACCV*, 2006.

[31] J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann Publishers, 1988.

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

[33] O. Veksler. *Efficient graph-based energy minimization methods in computer vision.* PhD thesis, Cornell University, Dept. of Computer Science, Ithaca, NY, 1999.

[34] M. J. Wainwright, T. S. Jaakkola, and A. S. Willsky. Tree consistency and bounds on the performance of the max-product algorithm and its generalizations. *Statistics and Computing*, 14(2):143–166, April 2004.

[35] M.J. Wainwright, T.S. Jaakkola, and A.S. Willsky. MAP estimation via agreement on (hyper)trees: Message-passing and linear-programming approaches. *IEEE Trans. Information Theory*, 51(11):3697–3717, 2005.

[36] Tomáš Werner. A linear programming approach to max-sum problem: A review. Technical Report CTU–CMP–2005–25, Center for Machine Perception, K13133 FEE Czech Technical University, Prague, Czech Republic, December 2005.

[37] S.-C. Zhu and D. Mumford. Prior learning and Gibbs reaction-diffusion. *PAMI*, 19(11):1236–1250, 1997.

[38] S.-C. Zhu, Y. Wu, and D. Mumford. FRAME: Filters, Random fields And Maximum Entropy— towards a unified theory for texture modeling. *IJCV*, 27:1–20, 1998.