Adaptive and Move Making Auxiliary Cuts for Binary Pairwise Energies

Lena Gorelick Yuri Boykov Olga Veksler
Computer Science Department
University of Western Ontario

Abstract

Many computer vision problems require optimization of binary non-submodular energies. In this context, local iterative submodularization techniques based on trust region (LSA-TR) and auxiliary functions (LSA-AUX) have been recently proposed [9]. They achieve state-of-the-art-results on a number of computer vision applications.

In this paper we extend the LSA-AUX framework in two directions. First, unlike LSA-AUX, which selects auxiliary functions based solely on the current solution, we propose to incorporate several additional criteria. This results in tighter bounds for configurations that are more likely or closer to the current solution. Second, we propose move-making extensions of LSA-AUX which achieve tighter bounds by restricting the search space.

Finally, we evaluate our methods on several applications. We show that for each application at least one of our extensions significantly outperforms the original LSA-AUX. Moreover, the best extension of LSA-AUX is comparable to or better than LSA-TR on four out of six applications.

1. Introduction

Minimization of binary pairwise non-submodular energies is a classical problem in combinatorial optimization, *e.g.* see [15, 8, 2]. In the last decade optimization methods based on LP relaxations, *e.g.* QPBO [21] and TRWS [13], became popular in vision and are thoroughly evaluated [12].

Another group of methods is based on local linearization, *e.g.* parallel ICM, IPFP [16], and similar methods [4]. These methods iteratively linearize the energy around the current solution and globally optimize the approximation within the integer domain. Such approach often gets stuck in poor local minimum by making large steps regardless of the quality of approximation. IPFP tries to control the step size by exploring relaxed solutions on a certain continuous line in the vicinity of the new minimum. Unfortunately, as with global LP relaxations, such relaxed solutions have no guarantees with respect to the original integer problem.

Recently, two local submodularization methods, LSA-TR and LSA-AUX have been proposed [9]. Instead of linear approximation, they use a more general class of submodular functions, which gives better approximation, yet allows efficient optimization.

LSA-TR is based on the trust region framework [23]. In each iteration, it approximates the original energy around current solution using a submodular function. The approximation is only trusted within a small region around current solution, called the trust region. The next candidate solution is obtained by globally optimizing the approximation within this trust region. The trust region size is then either increased or decreased based on the quality of the approximation, providing a better control of the step-size problem.

LSA-AUX is based on the auxiliary function framework [14, 18, 1], also known as bound optimization. In each iteration, it finds a submodular upper bound for the original energy around the current solution and minimizes it globally. There is no need to control the step size as the global minimizer is guaranteed to decrease the original energy.

Local submodularization methods were reported to obtain state-of-the-art results on several applications, involving quadratic [9, 11] and high-order [22] energies, with LSA-TR being the more accurate and LSA-AUX the faster.

In this paper we extend the LSA-AUX framework in two directions. First, we introduce a new way to select upper bounds for each supermodular potential. LSA-AUX uses a predetermined bound based solely on its configuration in the current solution. In contrast, we propose using an adaptive upper bound that is selected based on several additional criteria such as unary term preference, energetic preference, proximity to the current solution and combinations of the above. Intuitively, this gives tighter bounds for configurations that are more likely or closer to the current solution.

Second, inspired by move-making algorithms in the context of multilabel energies [3, 17] and binary energies [10], we propose two types of binary move-making extensions of LSA-AUX. They achieve more accurate bounds by restricting the search space. The first type is trivially based on α-expansion, where the current solution is either allowed to grow or to shrink. Such moves yield certain configurations invalid. Therefore, they can be ignored when selecting an upper bound, allowing tighter bounds for the remaining configurations. The second type of move-making is based on the geometry of the segment's boundary. Each move accurately models boundary change in a certain orientation and we iterate over the orientations.

Note that the proposed extensions could be used in conjunction with the recently published pseudo-bound optimization [22] that employs families of pseudo-bounds.

We evaluate the proposed adaptive and move-making auxiliary cuts on inpainting, segmentation with repulsion and binary deconvolution as in [9], squared curvature [19], compact shape [5] and multi-part object [6] priors. We show that for each application at least one of our extensions significantly outperforms the original LSA-AUX. Moreover, the best extension of LSA-AUX is comparable to or better than LSA-TR on four out of six applications.

This paper is structured as follows. Section 2 outlines the energy and reviews LSA-AUX. Sections 3 and 4 describe the proposed adaptive and move making auxiliary cuts respectively. Section 5 reports the experimental results.

2. Overview of LSA-AUX

We address a general class of binary pairwise non-submodular energies, which are popular in computer vision. Without loss of generality, such energy can be written[1] as

$$E(S) = S^T U + S^T M S, \qquad S \in \{0,1\}^\Omega \tag{1}$$

where $S = (s_p \,|\, p \in \Omega)$ is a vector of binary indicator variables defined on pixels $p \in \Omega$, vector $U = (u_p \in \mathcal{R} \,|\, p \in \Omega)$ represents unary potentials, and symmetric matrix $M = (m_{pq} \in \mathcal{R} \,|\, p, q \in \Omega)$ represents pairwise potentials. In many applications M is sparse with $m_{pq} = 0$ for all non-interacting pixel pairs. We seek solutions to the following integer quadratic optimization problem

$$\min_{S \in \{0,1\}^\Omega} E(S). \tag{2}$$

When $m_{pq} \le 0 \;\; \forall (p,q)$, the energy (1) is *submodular* and a global optimum for (2) can be found in polynomial time [2]. The general non-submodular case of (2) is NP hard.

LSA-AUX is based on the auxiliary functions framework. It is a class of iterative algorithms that in each iteration construct and optimize an upper bound of the original energy E around current solution. It is assumed that those bounds are easier to optimize than the original energy E.

[1]Note that the transformation is up to a constant.

We decompose the energy E in (1) into submodular and supermodular parts $E(S) = E^{sub}(S) + E^{sup}(S)$ such that

$$E^{sub}(S) \;=\; S^T U + S^T M^- S \tag{3}$$
$$E^{sup}(S) \;=\; S^T M^+ S \tag{4}$$

where M^- with elements $m^-_{pq} \le 0$ and M^+ with $m^+_{pq} \ge 0$ hold submodular and supermodular potentials respectively.

Given current solution S_t, the function $A_t(S)$ is an auxiliary function of E if it satisfies the following conditions:

$$E(S) \le A_t(S) \tag{5a}$$
$$E(S_t) = A_t(S_t) \tag{5b}$$

We iteratively minimize a sequence of auxiliary functions:

$$S_{t+1} = \arg\min_S \; A_t(S)\,, \quad t = 1, 2, \ldots \tag{6}$$

Using (5a), (5b), and (6), it is easy to prove that the solutions in (6) yield a sequence of decreasing energy values.

$$E(S_{t+1}) \le A_t(S_{t+1}) \le A_t(S_t) = E(S_t).$$

The main challenge in bound optimization is designing an appropriate upper bound $A_t(.)$ satisfying (5a) and (5b). However, in case of integer quadratic optimization problem (2), this step is simple [9] and is summarized below.

Consider any supermodular potential[2] $\beta \cdot s_p s_q$ in (4). It can be bounded above by linear function $v_{pq}(s_p, s_q) := v_p \cdot s_p + v_q \cdot s_q \ge \beta \cdot s_p s_q$ for some positive scalars v_p and v_q.

Let (s^t_p, s^t_q) be the configuration of pixels p and q in current solution S_t. The table below specifies upper bounds for the four possible configurations of (s^t_p, s^t_q) as in [9]:

(s^t_p, s^t_q)	upper bound $v^t_{pq}(s_p, s_q)$
(0,0)	$\frac{\beta}{2} s_p + \frac{\beta}{2} s_q$
(0,1)	βs_p
(1,0)	βs_q
(1,1)	$\frac{\beta}{2} s_p + \frac{\beta}{2} s_q$

Table 1. Upper bounds used in [9].

Summing upper bounds for all supermodular potentials in (4) gives an overall linear upper bound

$$E^{sup}(S) \le S^T V_t \tag{7}$$

where vector $V_t = (v^t_p | p \in \Omega)$ consists of elements

$$v^t_p = \sum_q \frac{m^+_{pq}}{2}(1 + s^t_q - s^t_p)$$

and S_t is the current solution.

[2]Here we use β instead of m^+_{pq} for brevity.

Let $A_t(S)$ be an auxiliary function at S_t, defined by

$$A_t(S) := S^T V_t + E^{sub}(S). \qquad (8)$$

Using inequality (7) we satisfy condition (5a)

$$E(S) = E^{sup}(S) + E^{sub}(S) \leq A_t(S).$$

Since $S_t^T V_t = E^{sup}(S_t)$, our auxiliary function (8) also satisfies condition (5b). Function $A_t(S)$ is submodular. Thus, we can globally optimize it in each iteration guaranteeing a decrease in energy.

3. Adaptive Auxiliary functions

Bounds βs_p and βs_q provided in lines 2-3 of Tab. 1 for configurations $(s_p^t, s_q^t) = (0,1)$ and $(s_p^t, s_q^t) = (1,0)$ are the tightest possible, as they coincide with the original potential $f_{pq} = \beta s_p s_q$ on three out of four possible discrete configurations. This is not the case with the bound $\frac{\beta}{2} s_p + \frac{\beta}{2} s_q$ used for other two configurations $(s_p^t, s_q^t) = (0,0)$ and $(s_p^t, s_q^t) = (1,1)$. It coincides with the original potential f_{pq} only on two out of four discrete configurations.

One can find tighter bounds for configurations $(s_p^t, s_q^t) = (0,0)$ and $(s_p^t, s_q^t) = (1,1)$ in lines 1,4 of Tab. 1. For example, βs_p or βs_q can also be used in these cases. Note that bound βs_q coincides with potential f_{pq} on configurations $(0,0)$, $(1,1)$ and $(1,0)$ and bound βs_p coincides with f_{pq} on configurations $(0,0)$, $(1,1)$ and $(0,1)$. Selecting one of the bounds essentially chooses which of the configurations $(0,1)$ or $(1,0)$ is modeled exactly by the bound in addition to $(0,0)$ and $(1,1)$. Therefore throughout the paper we use statements "we select bound βs_p" and "we model configuration $(0,1)$ precisely" interchangeably. Similarly, the statements "we select bound βs_q" and "we model configuration $(1,0)$ precisely" are interchangeable.

Without additional information, it is not clear which of the two equally tight bounds βs_p and βs_q is better for configurations in lines 1,4 of Tab. 1. One way is to randomly select either βs_p or βs_q in those two cases. This is exactly the permutation based approach [18] proposed in the context of high-order supermodular terms when reduced to pairwise potentials. We call this approach LSA-AUX-P. However in [9], LSA-AUX-P was inferior or comparable to bounds in Tab. 1 in all but one application.

An alternative way is to decide whether configuration $(0,1)$ or $(1,0)$ is more "likely" in some sense. We choose bound βs_p if configuration $(0,1)$ is more likely, and βs_q otherwise. This way the chosen bound coincides with the original pairwise potential on the more likely configuration.

Below we provide several selection criteria that give rise to distinct auxiliary cuts algorithms. All these methods differ in the way they select bounds for current configurations $(s_p^t, s_q^t) = (0,0)$ and $(s_p^t, s_q^t) = (1,1)$, leaving the bounds for configurations $(s_p^t, s_q^t) = (0,1)$ and $(s_p^t, s_q^t) = (1,0)$ unchanged, as in Tab. 1.

3.1. Unary Preference

The simplest way to decide which of two configurations $(0,1)$ or $(1,0)$ is more likely is based on the unary terms. Let U be the vector of unary coefficients as in (1). If pixel p has a stronger preference for background compared to pixel q, that is $u_p > u_q$, we say that configuration $(s_p, s_q) = (0,1)$ is more likely than $(1,0)$ and select bound βs_p.

We refer to such version of LSA-AUX as AUX-U. The table below specifies all cases of the selection criterion and the resulting bounds for configurations $(0,0)$ and $(1,1)$:

unary preference	upper bound $v_{pq}^t(s_p, s_q)$
$(u_p < u_q)$	βs_q
$(u_p > u_q)$	βs_p
$(u_p = u_q)$	$\frac{\beta}{2} s_p + \frac{\beta}{2} s_q$

3.2. ICM preference

A more involved way to choose between the bounds βs_p βs_q is inspired by the principle of the parallel ICM [16]. In ICM, given current solution S_t, we fix all labels except one and optimize over the remaining variables. In parallel ICM, this is done simultaneously for all variables yielding a linear approximation function, which can be efficiently optimized. Such an approximation function is used in LSA-TR approach in [9]. For selecting adaptive upper bounds we use pairs of variables instead.

Given supermodular potential $\beta \cdot s_p s_q$, we fix all variables except s_p and s_q to current solution S_t. We then evaluate energy E of solution S_t, first substituting $(s_p, s_q) = (0,1)$ and then substituting $(s_p, s_q) = (1,0)$, denoted by S_t^{01} and S_t^{10} respectively. The configuration with the lower energy is selected to be modeled precisely. This is done simultaneously for all supermodular potentials. We refer to this version of LSA-AUX as AUX-ICM and specify the selected bounds for each case in the table below.

ICM preference	upper bound $v_{pq}^t(s_p, s_q)$
$E(S_t^{01}) < E(S_t^{10})$	βs_p
$E(S_t^{01}) > E(S_t^{10})$	βs_q
$E(S_t^{01}) = E(S_t^{10})$	$\frac{\beta}{2} s_p + \frac{\beta}{2} s_q$

3.3. Distance from current solution S_t

Another approach to decide which of the configurations $(0,1)$ or $(1,0)$ should be modeled precisely is based on the distance from the current solution. Consider supermodular potential $\beta \cdot s_p s_q$ and let $D_t = (d_p^t | p \in \Omega)$ be the vector of shortest distances from each pixel to the boundary of current solution S_t. We say that changes closer to the boundary are more desirable than changes farther from it. Assume current configuration $(s_p^t, s_q^t) = (0,0)$. If pixel p is closer to the boundary than pixel q, that is $d_p^t < d_q^t$, we prefer to model precisely the configuration in which p changes its label rather than q. That is, we select configuration $(1,0)$.

Similarly, if current configuration is $(s_p^t, s_q^t) = (1, 1)$ and pixel p is closer to the boundary than pixel q, we select to model precisely configuration $(0, 1)$. The table below specifies all cases and the resulting bounds for configurations $(s_p^t, s_q^t) = (0, 0)$ and $(s_p^t, s_q^t) = (1, 1)$:

$(s_p^t, s_q^t) \wedge$ distance preference	upper bound $v_{pq}^t(s_p, s_q)$
$(0, 0) \wedge (d_p^t < d_q^t)$	βs_q
$(0, 0) \wedge (d_p^t > d_q^t)$	βs_p
$(d_p^t = d_q^t)$	$\frac{\beta}{2} s_p + \frac{\beta}{2} s_q$
$(1, 1) \wedge (d_p^t > d_q^t)$	βs_q
$(1, 1) \wedge (d_p^t < d_q^t)$	βs_p

Distance based approach for selecting bounds has an interesting property. Implicitly, it defines a large family of segments $C_t = \{S \subset \Omega | S^T V_t = E^{sup}(S)\}$ for which the upper bound $S^T V_t$ coincides with the original supermodular energy term $E^{sup}(S)$. This means that we optimize the original energy over segments in C_t exactly.

The family C_t is quite large. For example, it contains all segments corresponding to the level sets of the distance transform D_t computed from current solution S_t, see (a-left) of Fig. 1. But this is just a small subset of the segments in C_t. In fact, any segment, in which the transition from label 1 to 0 occurs along the gradient orientation of the distance transform, is also in C_t, see (a-right) of Fig. 1.

It is worth noting that many segments that are not in C_t may still have large sections of their contour modeled precisely by the upper bound $S^T V_t$. Figure 1, (b) shows examples of segments with precisely modeled sections shown in green and imprecise sections shown in red. The transition from 1 to 0 across the red sections of the contour is either in opposite orientation w.r.t. the gradient (b-left) or is parallel to the gradient (b-right). The bound for such transitions is not tight.

Due to regularizing properties of the distance transform, family C_t consists of coherent segments with smooth boundaries. Such segments are more likely candidate solutions for vision applications. In contrast, choosing bounds at random as in [18] yields a family C_t that is not likely to contain (*i.e.* model precisely) segments with smooth boundaries, since a set of random configurations is not likely to coincide with a smooth contour. We refer to distance based version of LSA-AUX as AUX-DIST.

While distance based solutions are very good candidates, combining distance criterion with the unary preference allows an additional subset of candidates. In this subset, candidate solutions deviate from the level sets of the distance transform based on the unary preference. We refer to such version of LSA-AUX as AUX-DIST-U. The table below specifies the selection criterion and the resulting bounds for configurations $(0, 0)$ and $(1, 1)$:

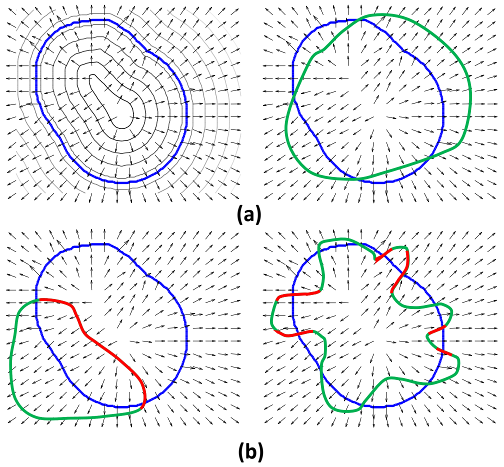


Figure 1. Distance Preference. a) Left: Current solution S_t (blue) and the level sets of its distance transform (gray) belong to C_t. Right: a segment (green) in C_t where transition from 1 to 0 aligns with the gradient orientation (black arrows). b) Segments that are not in C_t, yet have large boundary sections that are modeled precisely (green). Sections with inaccurate bounds are shown in red.

$(s_p^t, s_q^t) \wedge$ combined preference	upper bound $v_{pq}^t(s_p, s_q)$
$(0, 0) \wedge (d_p + u_p < d_q + u_q)$	βs_q
$(0, 0) \wedge (d_p + u_p > d_q + u_q)$	βs_p
$(0, 0) \wedge (d_p + u_p = d_q + u_q)$	$\frac{\beta}{2} s_p + \frac{\beta}{2} s_q$
$(1, 1) \wedge (d_p - u_p < d_q - u_q)$	βs_q
$(1, 1) \wedge (d_p - u_p > d_q - u_q)$	βs_p
$(1, 1) \wedge (d_p - u_p = d_q - u_q)$	$\frac{\beta}{2} s_p + \frac{\beta}{2} s_q$

3.4. Supermodular Neighborhood Size

Another way to choose between the bounds is based on the size of *supermodular neighborhood*. Given current solution S_t we can measure for each pixel p the number n_p^t of supermodular potentials (p, r) with r in the foreground:

$$n_p^t = \sum_{(r \in \mathcal{N}_p) \wedge (m_{pr} > 0)} s_r^t. \quad (9)$$

Consider supermodular potential (p, q). If both pixels are in the foreground and $(n_p^t > n_q^t)$, this means in most cases that q is closer to the boundary. Therefore we model precisely configuration $(1, 0)$ where p stays unchanged. If both pixels are in the background and $(n_p^t > n_q^t)$, this means that p is closer to the boundary. Therefore we model precisely configuration $(1, 0)$, where q stays unchanged. Pixels that are far from the boundary have a flat value n_p^t that is either zero (in the background) or the size of the supermodular neighborhood (inside the object). The table below specifies the selection criterion and the resulting bounds for configurations $(0, 0)$ and $(1, 1)$:

sup. neighborhood size	upper bound $v^t_{pq}(s_p, s_q)$
$(n^t_p < n^t_q)$	βs_q
$(n^t_p > n^t_q)$	βs_p
$(n^t_p = n^t_q)$	$\frac{\beta}{2} s_p + \frac{\beta}{2} s_q$

In the experiments we use a weighted version of (9), namely $n^t_p = \sum_{(r \in \mathcal{N}_p) \wedge (m_{pr} > 0)} m_{pr} \cdot s^t_r$, since it works slightly but consistently better in practice. We refer to this version of LSA-AUX as AUX-SNS.

4. Move-Making Auxiliary Cuts

Inspired by move-making algorithms in the context of multilabel [3, 17] and binary [10] energies, below we propose two types of binary move-making extensions. In contrast to the adaptive bounds approach, where one seeks upper bounds that are precise for certain configurations, move-making extensions achieve better performance through restricting the search space either explicitly or implicitly.

4.1. α-expansion Auxiliary Cuts

The first type is trivially based on α-expansion, where the current solution is either allowed to grow or to shrink, as in [10]. Such moves yield certain configurations invalid, allowing the remaining, valid configurations to be modeled precisely by an approximation. In each α-expansion move, we hard constraint all variables that currently have label α and run the auxiliary cuts method till convergence. We repeat the process of alternating α till convergence. Note, that α-expansion can be used in conjunction with LSA-AUX [9] and any of the adaptive auxiliary cuts in Sec. 3.

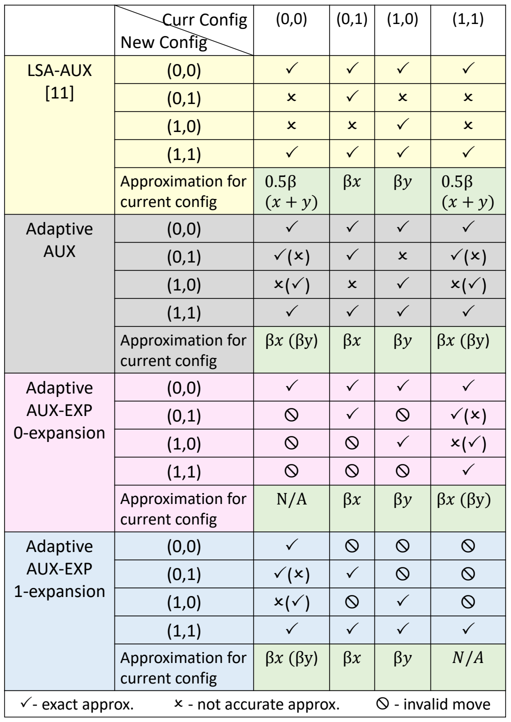
Figure 2 demonstrates the advantage of restricting the search space for auxiliary cuts. The columns correspond to four possible configurations of pairwise potential (s^t_p, s^t_q) in the current solution. The rows correspond to four configurations of potentials (s_p, s_q) in a candidate solution. Based on the current configuration an appropriate bound is selected in each method (green line in each section). If there are two choices, an alternative is specified in the brackets. We then specify whether it is precise approximation or not for each possible configuration of (s_p, s_q) in a candidate solution.

It can be seen that LSA-AUX [9] (yellow) is precise in ten out of 16 cases. Adaptive auxiliary cuts (gray) is precise in twelve out of 16 cases. Finally, $\alpha-$expansion of the adaptive approach (blue and pink) is precise in eight out of nine cases, since some of the configurations are invalid.

Whenever we use expansion moves with any of the methods proposed in Sec. 3, we add the -EXP suffix to their name. For example, when AUX-U is used with expansion moves, we call it AUX-U-EXP.

4.2. Compass Moves

The second type of move-making is based on geometry of the segment's boundary. It is inspired by the order-



	Curr Config / New Config	(0,0)	(0,1)	(1,0)	(1,1)
LSA-AUX [11]	(0,0)	✓	✓	✓	✓
	(0,1)	×	✓	×	×
	(1,0)	×	×	✓	×
	(1,1)	✓	✓	✓	✓
	Approximation for current config	$0.5\beta(x+y)$	βx	βy	$0.5\beta(x+y)$
Adaptive AUX	(0,0)	✓	✓	✓	✓
	(0,1)	✓(×)	✓	×	✓(×)
	(1,0)	×(✓)	×	✓	×(✓)
	(1,1)	✓	✓	✓	✓
	Approximation for current config	βx (βy)	βx	βy	βx (βy)
Adaptive AUX-EXP 0-expansion	(0,0)	✓	✓	✓	✓
	(0,1)	⊘	✓	⊘	✓(×)
	(1,0)	⊘	⊘	✓	×(✓)
	(1,1)	⊘	⊘	⊘	✓
	Approximation for current config	N/A	βx	βy	βx (βy)
Adaptive AUX-EXP 1-expansion	(0,0)	✓	⊘	⊘	⊘
	(0,1)	✓(×)	✓	⊘	⊘
	(1,0)	×(✓)	⊘	✓	⊘
	(1,1)	✓	✓	✓	✓
	Approximation for current config	βx (βy)	βx	βy	N/A

✓ - exact approx. × - not accurate approx. ⊘ - invalid move

Figure 2. Adaptive auxiliary cuts with expansion moves have higher proportion of precise bounds due to restricted search space.

preserving moves proposed in the context of multilabel energies [17]. In order preserving moves, each pixel expands on a chosen set of labels based on orientation of the move. In contrast, in each compass move, an orientation (up, down, left, right) is chosen and, for all supermodular pairwise potentials in that orientation, we select configuration $(1, 0)$ to be modeled precisely by the upper bound. This allows precise upper bounds for changes along either top-, bottom-, left- or right-facing boundaries. We keep alternating the moves and for each move, we run auxiliary cuts till convergence. We call such approach compass moves.

Figure 3, (top) illustrates compass moves on a small example. Consider orientation "right". For any supermodular potential (p, q) where q is to the right of p, we model configuration $(s_p, s_q) = (1, 0)$ precisely. The striped region specifies possible location of pixel q w.r.t. pixel p. Note that we arbitrarily decided that pixels above p are considered to be to its right. Other orientations are handled similarly.

As with the distance based preference, given current solution S_t, compass moves also define a coherent family C_t of segments for which the upper bound is precise. Fig. 3, (bottom) shows examples of segments in family C_t for a specific solution S_t and orientation "right". The green sections of the boundary are modeled precisely because they align with the selected orientation. The blue sections are modeled precisely due to requirement 5b.

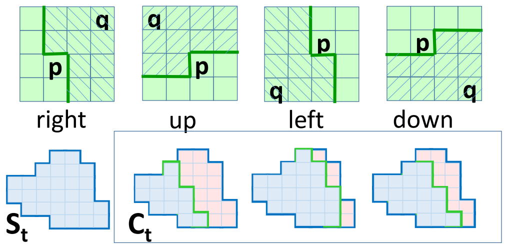


Figure 3. Compass Moves: (top) - Given pixel p and any pixel q in the striped region, configuration $(1, 0)$ is modeled precisely for orientations "right", "up", "left", "down". (bottom) - examples of segments in family C_t for given S_t and orientation "right".

Note, that unlike expansion moves that explicitly restrict the search space, in compass moves the restriction is implicit. Changes in the wrong orientation are possible but less likely since they have a looser upper bound. This version of auxiliary cuts is denoted by AUX-COMPASS.

5. Applications

Below we apply our methods to squared curvature regularization [19], several benchmark energies from [9] such as inpainting, segmentation with repulsion and binary deconvolution, as well as segmentation with compact [5] and multi-region [6] shape priors. For each application we compare the proposed adaptive and move-making extensions with the original LSA-AUX (AUX here), its random permutation version LSA-AUXP (AUXP) and LSA-TR. For completness, we also compare to other standards optimization methods, such as QPBO [21], LBP [20], IPFP [16], TRWS and SRMP [13]. Unless otherwise stated, all local approximation methods are initialized with the entire domain assigned to the foreground. All global optimization methods are run for up to 5000 iterations.

5.1. Squared Curvature

Below we focus on the squared curvature model in [19]. In combination with appearance terms, the model yields discrete binary energy with submodular and non-submodular pairwise terms. The weight of curvature term relative to appearance term is controlled by parameter λ_{curv}.

For this application we use Picasso's Don Quixote drawing. We vary the weight λ_{curv} and evaluate the performance of the proposed extensions. The best three are compared to LSA-TR, AUX and AUXP, see Fig. 4. All methods start with the maximum likelihood solution based on the appearance terms. When the weight of supermodular curvature terms increases, the proposed methods consistently outperform LSA-TR (blue line), AUX (red) and AUXP (green). All other standard optimization methods such as QPBO, LBP, TRWS, SRMP, and IPFP were significantly inferior even to the worst of the proposed extensions, AUX-DIST-U-EXP, see supplementary material for details.

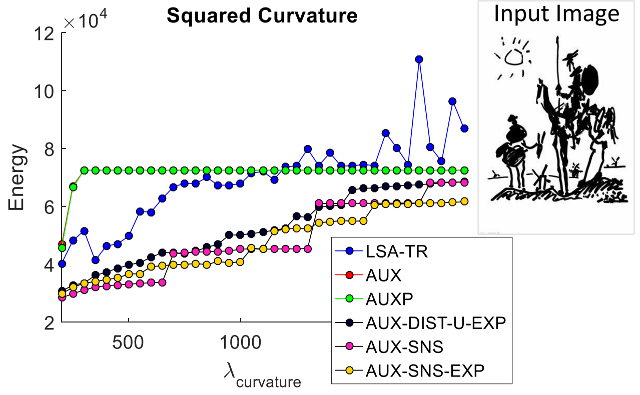


Figure 4. Squared curvature model. We used Gaussian with ($\mu = 0, \sigma = 0.2$) and ($\mu = 1, \sigma = 0.2$) for the foreground and background appearance and 7×7 stencils for angular resolution.

Alg. Name	Mean Runtime	Mean Energy	#best	Rank
MCBC	2053.89 sec	-49550.1	59	1
BPS (LBP)*	72.85 sec	-49537.08	6	6
ILP	3580.93 sec	-49536.59	5	7
SA	NaN sec	-49533.08	11	4
TRWS-LF	2106.94 sec	-49519.44	4	8
LSA	0.08 sec	-49547.61	16	3
AUX-DIST	0.07 sec	-49538.72	7	5
AUX-DIST-U	0.07 sec	-49548.78	33	2
AUX-ICM	0.07 sec	-49533.09	4	8

Table 2. Chinese characters in-painting database [12].

5.2. Chinese Characters Inpainting

Below we consider the task of in-painting of Chinese characters, *dtf-chinesechar* [12]. We use a set of pre-trained unary and pairwise potentials provided by the authors with the dataset. Table 2 reports the performance of the best three of the proposed extensions and previously published results in [12, 9]. We omit rows with methods ranked below eight. AUX-DIST-U is ranked second, but runs five orders of magnitude faster than MCBC which is ranked first.

5.3. Segmentation with Repulsion

Below we consider segmentation with attraction and repulsion pairwise potentials [9] based on pixels' appearance and relative position. We use 16-neighborhood system and define the potentials as follows: $\omega(p, q) = \frac{-\Delta(p,q)+c}{\text{dist(p,q)}}$. Here dist(p,q) denotes the distance between image pixels p and q and $\Delta(p, q)$ is the difference in their respective intensities. The constant c makes similar neighbors attract, resulting in submodular potentials, and repulse otherwise, giving supermodular potentials. λ_{reg} is the weight of the pairwise term.

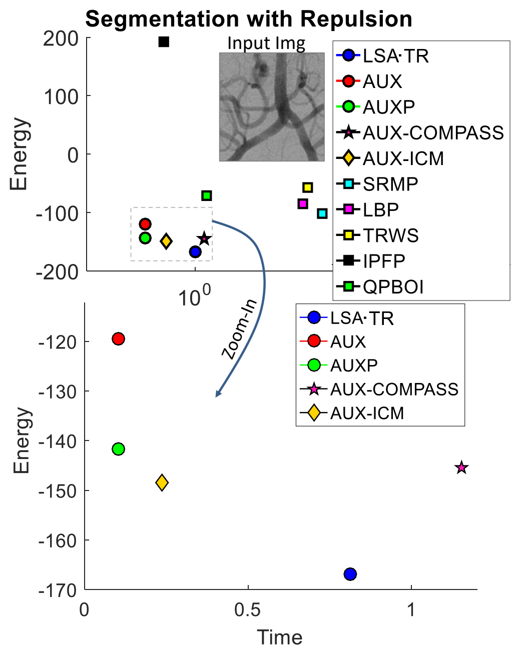


Figure 5. Segmentation with repulsion and attraction. We used μ_{fg}=0.4, μ_{bg}=0.6, σ=0.4 for appearance, λ_{reg}=100 and c=0.06.

Figure 5 reports the results. The best three of the proposed extensions outperform both AUX, AUXP and all standard optimization methods except LSA-TR. This is the only application in which many of the proposed extensions were inferior to the original AUX, suggesting that each application needs specifically designed adaptive bounds.

5.4. Binary Deconvolution

Consider a binary image convolved with a uniform 3×3 filter and combined with a Gaussian noise $\sim N(0, \sigma_{\text{noise}})$. The goal of binary deconvolution is to recover the original binary image. The energy is defined as

$$E(S) = \sum_{p \in \Omega} (I_p - \frac{1}{9} \sum_{q \in \mathcal{N}_p} s_q)^2, \tag{10}$$

where $\mathcal{N}_p$ denotes the 3×3 neighborhood window around pixel p and all pairwise interactions are supermodular.

Figure 6 reports the results. The best among the proposed extensions, namely AUX-ICM and AUX-U, significantly outperform the original AUX and AUXP, but not LSA-TR. For this application, LSA-TR shares the first rank with IPFP. This is consistent with the results in [9].

5.5. Segmentation of Multi-Region Objects

In this section we focus on MRI liver segmentation. The input image contains a liver with four tumors. The problem is formulated as minimization of a non-submodular energy [10]. It employs a multi-region model [6] and is able to encode geometric interactions such as inclusion and exclusion between the regions.

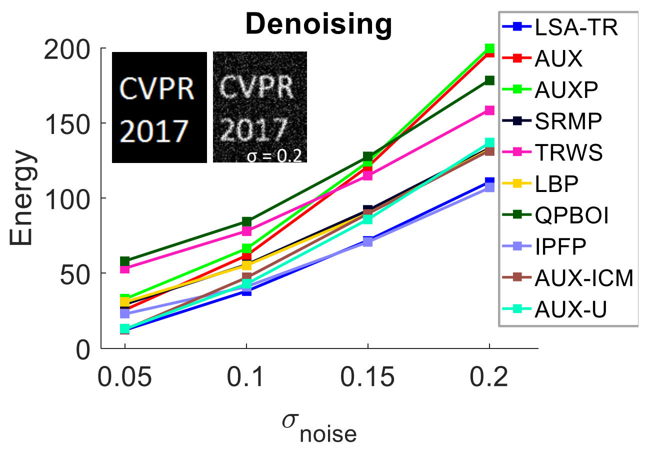


Figure 6. Binary deconvolution. We report the energy as a function of noise level σ, averaged over ten random instances.

The liver is modeled by a graph with five layers of binary variables, corresponding to liver (Fg), and four tumors (A, B, C, D). See Fig. 7, (a-b) for a schematic illustration. Inclusion of tumors within liver and exclusion constraints between tumors are implemented using submodular and supermodular inter-layer pairwise potentials respectively. In addition, we use Potts regularization on each layer. Please see supplementary material for the technical details.

Figure 7, (c-d) shows the results. We use scribbles for appearance and as hard constraints. The top plot compares the methods in terms of energy and running time. The bottom plot zooms in on the most interesting part. Most methods arrived at poor solutions that have violations of inclusion and exclusion constraints. LSA-TR, AUX-DIST, AUX-DIST-U achieve the same lowest energy, with AUX-DIST being an order of magnitude faster.

5.6. Generalized Compact Shape Prior

Below we apply our methods to segmentation with generalized compact shape prior proposed in [10]. The original compact shape prior in [5] partitions an object into four quadrants around a given center, provided by the user. Within each quadrant, the contour must follow the allowed orientation for that quadrant, see Fig. 8, (a).

Instead of dividing the object into four quadrants, the generalized prior [10] divides the background into four regions as in Fig. 8, (b), corresponding to labels: top-left (TL), top-right (TR), bottom-left (BL), bottom-right (BR). There is also a label for the foreground object (fg). Similarly to the multi-region model, this model uses a graph with four binary layers: TL, TR, BL, BR. Each layer allows discontinuities only in certain orientation using Potts. There are also pairwise supermodular exclusion constraints between the corresponding nodes of the binary layers to enforce a coherent foreground object. See Fig. 8, (c-d) for the schematic representation of the generalized model, and

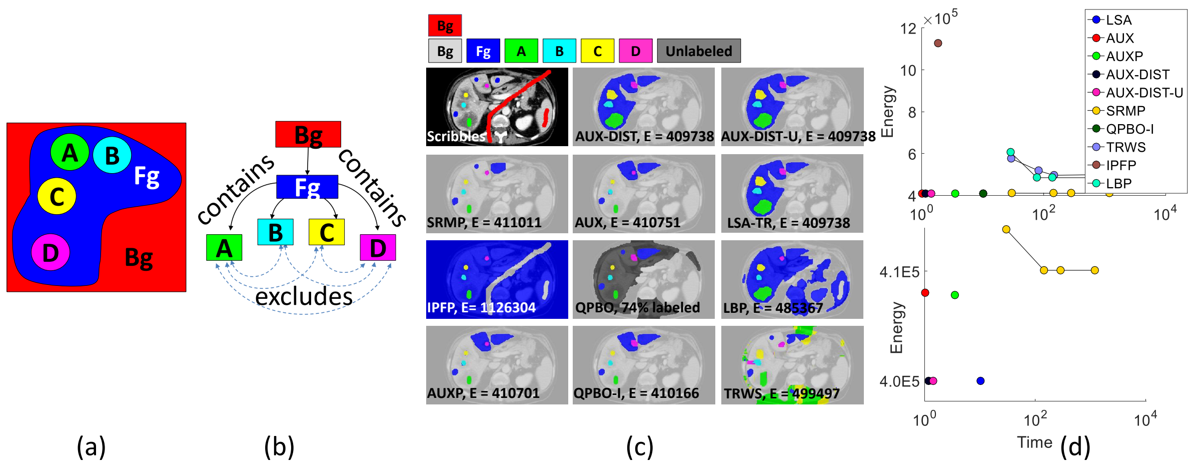


Figure 7. Multi-region liver segmentation. (a) schematic representation of liver and tumors, (b) inclusion and exclusion interactions between corresponding nodes of different graph layers, (c) input scribbles - liver (blue) and tumors (green, yellow, cyan, magenta) along with segmentation results, (d) energy vs. time plot (top) with zoom in (bottom). We set weights $\lambda_{sub} = \lambda_{sup} = 100$ and $\lambda_{\text{Potts}} = 25$.

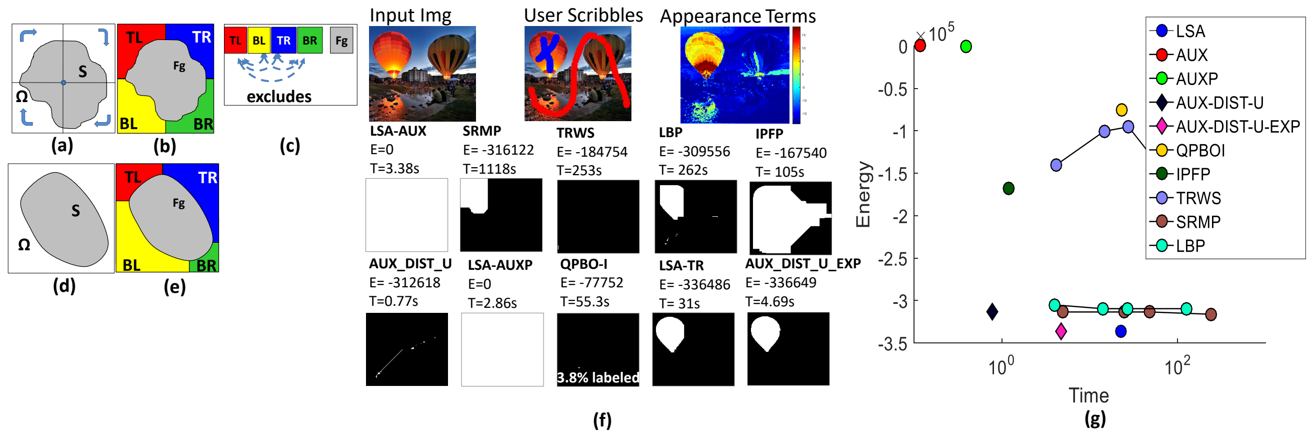


Figure 8. Generalized Compact Shape Prior: (a) the model in [5], (b) the generalized model in [10], (c) schematic representation of the generalized model in [10], (d) a segment that cannot be modeled using [5], (e) can be segmented with [10], (f-e) results and comparison.

supplementary material for the technical details.

The model in [5] is a special case of the generalized model [10], when the transitions between background labels are horizontally and vertically aligned as in (b). The generalized model does not require such alignment. For example, it can model the object in (d) using four regions shown in (e). This object cannot be segmented using [5] due to restrictions on the orientation of the contours.

Below we report the results. Figure 8, (f) shows an input image, user scribbles and the resulting appearance terms, followed by final segmentations for each method. Figure 8, (g) compares the methods in terms of energy and time. Most methods arrived at poor solutions with violations of the orientation and coherence constraints. LSA-TR and AUX-DIST-U-EXP are the only methods that could optimize such energy, with AUX-DIST-U-EXP obtaining the lowest energy in shorter time.

6. Conclusions and Future Work

We proposed two extensions to the LSA-AUX framework. First, we incorporate energetic and distance based criteria for selection of upper bounds. This results in tighter bounds for configurations that are more likely or closer to the current solution. Second, we proposed two move-making extensions of LSA-AUX which achieve more accurate upper bounds by means of restricted search space.

We conclude that different extensions work better for different applications, leaving room for designing application specific adaptive bounds and moves. We also plan to extend our approach to higher-order and multilabel energies. In addition, it would be interesting to design upper bounds based on the theory of submodular functions [7].

References

[1] I. Ben Ayed, L. Gorelick, and Y. Boykov. Auxiliary cuts for general classes of higher order functionals. In *IEEE conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1304–1311, Portland, Oregon, June 2013. 1

[2] E. Boros and P. L. Hammer. Pseudo-boolean optimization. *Discrete Applied Mathematics*, 123:2002, 2001. 1, 2

[3] Y. Boykov, O. Veksler, and R. Zabih. Fast approximate energy minimization via graph cuts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23:1222–1239, November 2001. 1, 5

[4] W. Brendel and S. Todorovic. Segmentation as maximum-weight independent set. In *Neural Information Processing Systems (NIPS)*, pages 307–315, 2010. 1

[5] P. Das, O. Veksler, V. Zavadsky, and Y. Boykov. Semiautomatic segmentation with compact shape prior. *Image Vision Computing*, 27(1-2):206–219, 2009. 2, 6, 7, 8

[6] A. Delong and Y. Boykov. Globally optimal segmentation of multi-region objects. In *IEEE 12th International Conference on Computer Vision, ICCV 2009, Kyoto, Japan, September 27 - October 4, 2009*, pages 285–292, 2009. 2, 6, 7

[7] S. Fujishige. *Submodular Functions and Optimization*. Elsevier, 2005. 8

[8] M. Goemans and D. Wiliamson. Improved approximation algorithms for maximum cut and satisfiability problem using semidefinite problem. *ACM*, 42(6):1115–1145, 1995. 1

[9] L. Gorelick, Y. Boykov, O. Veksler, I. B. Ayed, and A. Delong. Submodularization for binary pairwise energies. In *Computer Vision and Pattern Recognition (CVPR)*, 2014. 1, 2, 3, 5, 6, 7

[10] L. Gorelick, Y. Boykov, O. Veksler, I. B. Ayed, and A. Delong. Local submodularization for binary pairwise energies. In *PAMI*, page accepted, 2017. 1, 5, 7, 8

[11] J. H. Kappes, B. Andres, F. A. Hamprecht, C. Schnörr, S. Nowozin, D. Batra, S. Kim, B. X. Kausler, T. Kröger, J. Lellmann, N. Komodakis, B. Savchynskyy, and C. Rother. A comparative study of modern inference techniques for structured discrete energy minimization problems. *International Journal of Computer Vision*, pages 1–30, 2015. 1. 1

[12] J. H. Kappes, B. Andres, F. A. Hamprecht, C. Schnorr, S. Nowozin, D. Batra, S. Kim, B. X. Kausler, J. Lellmann, N. Komodakis, and C. Rother. A comparative study of modern inference techniques for discrete energy minimization problem. In *IEEE conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1328–1335, 2013. 1, 6

[13] V. Kolmogorov and T. Schoenemann. Generalized seq. tree-reweighted message passing. *arXiv:1205.6352*, 2012. 1, 6

[14] K. Lange, D. R. Hunter, and I. Yang. Optimization transfer using surrogate objective functions. *Journal of Computational and Graphical Statistics*, 9(1):1–20, 2000. 1

[15] R. Lazimy. Mixed integer quadratic programming. *Mathematical Programming*, 22:332–349, 1982. 1

[16] M. Leordeanu, M. Hebert, and R. Sukthankar. An integer projected fixed point method for graph matching and map inference. In *Neural Information Processing Systems (NIPS)*, pages 1114–1122, 2009. 1, 3, 6

[17] X. Liu, O. Veksler, and J. Samarabandu. Order-preserving moves for graph-cut-based optimization. *IEEE Trans. Pattern Anal. Mach. Intell.*, 32(7):1182–1196, July 2010. 1, 5

[18] M. Narasimhan and J. A. Bilmes. A submodular-supermodular procedure with applications to discriminative structure learning. In *UAI*, pages 404–412, 2005. 1, 3, 4

[19] C. Nieuwenhuis, E. Toeppe, L. Gorelick, O. Veksler, and Y. Boykov. Efficient Squared Curvature. In *IEEE conf. on Comp. Vision and Pattern Recognition (CVPR)*, 2014. 2, 6

[20] J. Pearl. Reverend bayes on inference engines: A distributed hierarchical approach. In *National Conference on Artificial Intelligence*, pages 133–136, 1982. 6

[21] C. Rother, V. Kolmogorov, V. Lempitsky, and M. Szummer. Optimizing binary MRFs via extended roof duality. In *IEEE conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–8, 2007. 1, 6

[22] M. Tang, I. B. Ayed, and Y. Boykov. Pseudo-bound optimization for binary energies. In *European Conference on Computer Vision (ECCV)*, Zurich, Switzerland, September 2014. 1, 2

[23] Y. Yuan. A review of trust region algorithms for optimization. In *Proceedings of the Fourth International Congress on Industrial & Applied Mathematics (ICIAM)*, 1999. 1