Submodularization for Binary Pairwise Energies

Lena Gorelick  Yuri Boykov  Olga Veksler  Ismail Ben Ayed  Andrew Delong
Computer Science Department, University of Western Ontario  GE Healthcare  Elect. & Comp. Engineering, University of Toronto

Abstract

Many computer vision problems require optimization of binary non-submodular energies. We propose a general optimization framework based on local submodular approximations (LSA). Unlike standard LP relaxation methods that linearize the whole energy globally, our approach iteratively approximates the energies locally. On the other hand, unlike standard local optimization methods (e.g., gradient descent or projection techniques) we use non-linear submodular approximations and optimize them without leaving the domain of integer solutions. We discuss two specific LSA algorithms based on trust region and auxiliary function principles, LSA-TR and LSA-AUX. These methods obtain state-of-the-art results on a wide range of applications outperforming many standard techniques such as LBP, QPBO, and TRWS. While our paper is focused on pairwise energies, our ideas extend to higher-order problems. The code is available online\(^1\).

1. Introduction

We address a general class of binary pairwise non-submodular energies, which are widely used in applications like segmentation, stereo, inpainting, deconvolution, and many others. Without loss of generality, the corresponding binary energies can be transformed into the form\(^2\)

\[
E(S) = S^T U + S^T M S, \quad S \in \{0, 1\}^{\Omega}
\]

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

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

\(^1\)http://vision.csd.uwo.ca/code/

\(^2\)Note that such transformations are up to a constant.

1.1. Standard linearization methods

Integer quadratic programming problems is a well-known challenging optimization problem with extensive literature in the combinatorial optimization community, e.g., see [15, 9, 3]. It often appears in computer vision where it can be addressed with many methods including spectral and semi-definite programming relaxations, e.g., see [19, 12].

Methods for solving (2) based on LP relaxations, e.g., QPBO [21] and TRWS [13], are considered among the most powerful in computer vision [11]. They approach integer quadratic problem (2) by global linearization of the objective function at a cost of introducing a large number of additional variables and linear constraints. These methods attempt to optimize the relaxed LP or its dual. However, the integer solution can differ from the relaxed solution circled in Fig. 1(a). This is a well-known integrality gap problem. Most heuristics for extracting an integer solution from the relaxed solution have no a priori quality guarantees.

Our work is more closely related to local linearization
techniques for approximating (2), e.g. parallel ICM, IPFP [16], and other similar methods [6]. Parallel ICM iteratively linearizes energy \( E(S) \) around current solution \( S_0 \) using Taylor expansion and makes a step by computing an integer minimizer \( S_{\text{int}} \) of the corresponding linear approximation, see Fig.1(b). However, similarly to Newton’s methods, this approach often gets stuck in bad local minima by making too large steps regardless of the quality of the approximation. IPFP attempts to escape such minima by reducing the step size. It explores the continuous line between integer minimizer \( S_{\text{int}} \) and current solution \( S_0 \) and finds optimal relaxed solution \( S_{\text{relx}} \) with respect to the original quadratic energy. Similarly to the global linearization methods, see Fig.1(a), such continuous solutions give no quality guarantees with respect to the original integer problem (2).

1.2. Overview of submodularization

Linearization has been a popular approximation approach to integer quadratic problem (1)-(2), but it often requires relaxation leading to the integrality gap problem. We propose a different approximation approach, which we refer to as submodularization. The main idea is to use submodular approximations of energy (1). We propose several approximation schemes that keep submodular terms in (1) and linearize non-submodular potentials in different ways leading to very different optimization algorithms. Standard truncation of non-submodular pairwise terms and some existing techniques for high-order energies [10, 17, 22, 2] can be seen as submodularization examples, as discussed later. Common properties of submodularization methods is that they compute globally optimal integer solutions of the approximation and do not need to leave the domain of discrete solutions avoiding integrality gaps. Symbodularization can be seen as a generalization of local linearization methods since it uses more accurate higher-order approximations.

One way to linearize non-submodular terms in (1) is to compute their Taylor expansion around current solution \( S_0 \). Taylor’s approach is similar to IPFP [16], but they linearize all terms including submodular ones. In contrast to IPFP, our overall approximation of \( E(S) \) at \( S_0 \) is not linear; it belongs to a more general class of submodular functions. Such non-linear approximations are more accurate while still permitting efficient optimization in the integer domain.

We also propose a different mechanism for controlling the step size. Instead of exploring relaxed solutions on continuous interval \([S_0, S_{\text{int}}]\) in Fig.1(b), we compute integer intermediate solutions \( S \) by minimizing local submodular approximation over \([0, 1]^\Omega\) under additional distance constraints \( \|S - S_0\| < \delta \). Thus, our approach avoids integrality gap issues. For example, even linear approximation model in Fig.1b can produce solution \( S^* \) if Hamming distance constraint \( \|S - S_0\| \leq 1 \) is imposed. This local submodularization approach to (1)-(2) fits a general trust region framework [8, 25, 19, 10] and we refer to it as LSA-TR.

Our paper also proposes a different local submodularization approach to (1)-(2) based on the general auxiliary function framework [14, 17, 2]. Instead of Taylor expansion, non-submodular terms in \( E(S) \) are approximated by linear upper bounds specific to current solution \( S_0 \). Combining them with submodular terms in \( E(S) \) gives a submodular upper-bound approximation, a.k.a. an auxiliary function, for \( E(S) \) that can be globally minimized within integer solutions. This approach does not require to control the step sizes as the global minimizer of an auxiliary function is guaranteed to decrease the original energy \( E(S) \). Throughout the paper we refer to this type of local submodular approximation approach as LSA-AUX.

Some auxiliary functions were previously proposed in the context of high-order energies [17, 2]. For example, [17] divided the energy into submodular and supermodular parts and replaced the latter with a certain permutation-based linear upper-bound. The corresponding auxiliary function allows polynomial-time solvers. However, experiments in [22] (Sec. 3.2) demonstrated limited accuracy of the permutation-based bounds [17] on high-order segmentation problems. Recently, Jensen inequality was used in [2] to derive linear upper bounds for several important classes of high-order terms that gave practically useful approximation results. Our LSA-AUX method is first to apply auxiliary function approach to arbitrary (non-submodular) pairwise energies. We discuss all possible linear upper bounds for pairwise terms and study several specific cases. One of them corresponds to the permutation bounds [17] and is denoted by LSA-AUX-P.

Recently both trust region [8, 25, 19] and auxiliary function [14] frameworks proved to work well for optimization of energies with high-order regional terms [10, 2]. They derive specific linear [10] or upper bound [2] approximations for non-linear cardinality potentials, KL and other distances between segment and target appearance models. To the best of our knowledge, we are the first to develop trust region and auxiliary function methods for integer quadratic optimization problems (1)-(2).

Our contributions can be summarized as follows:

- A general submodularization framework for solving integer quadratic optimization problems (1)-(2) based on local submodular approximations (LSA). Unlike global linearization methods, LSA constructs an approximation model without additional variables. Unlike local linearization methods, LSA uses a more accurate approximation functional.
- In contrast to the majority of standard approximation

---

3 Truncation is known to give low quality results, e.g. Fig.4, Tab.1.

4 Auxiliary functions are also called surrogate functions or upper-bounds. The corresponding approximate optimization technique is also known as the majorize-minimize principle [14].
methods, LSA avoids integrality gap issue by working strictly within the domain of discrete solutions.

- State-of-the-art results on a wide range of applications. Our LSA algorithms outperformQPBO, LBP, IPFP, TRWS, its latest variant SRMP, and other standard techniques for (1)-(2).

2. Description of LSA Algorithms

In this section we discuss our framework in detail. Section 2.1 derives local submodular approximations and describes how to incorporate them in the trust region framework. Section 2.2 briefly reviews auxiliary function framework and shows how to derive local auxiliary bounds.

2.1. LSA-TR

Trust region methods are a class of iterative optimization algorithms. In each iteration, an approximate model of the optimization problem is constructed near the current solution \( S_0 \). The model is only accurate within a small region around the current solution called “trust region”. The approximate model is then globally optimized within the trust region to obtain a candidate solution. This step is called trust region sub-problem. The size of the trust region is adjusted in each iteration based on the quality of the current approximation. For a detailed review of trust region framework see [25].

Below we provide details for our trust region approach to the binary pairwise energy optimization (see pseudo-code in Algorithm 1). The goal is to minimize \( E(S) \) in (1). This energy can be decomposed into submodular and supermodular parts

\[
E(S) = E_{\text{sub}}(S) + E_{\text{sup}}(S)
\]

such that

\[
E_{\text{sub}}(S) = S^T U + S^T M^- S
\]

\[
E_{\text{sup}}(S) = S^T M^+ S
\]

where matrix \( M^- \) with negative elements \( m_{pq}^- \leq 0 \) represents the set of submodular pairwise potentials and matrix \( M^+ \) with positive elements \( m_{pq}^+ \geq 0 \) represents supermodular potentials. Given the current solution \( S_t \) energy \( E(S) \) can be approximated by submodular function

\[
E_t(S) = E_{\text{sub}}(S) + S^T U_t + \text{const}
\]

where \( U_t = 2 M^+ S_t \). The last two terms in (3) are the first-order Taylor expansion of supermodular part \( E_{\text{sup}}(S) \).

While the use of Taylor expansion may seem strange in the context of functions of integer variables, Figure 2(a,b) illustrates its geometric motivation. Consider individual pairwise supermodular potentials \( f(x, y) \) in

\[
E_{\text{sup}}(S) = \sum_{pq} m_{pq}^+ \cdot s_p s_q = \sum_{pq} f_{pq}(s_p, s_q).
\]

Coincidentally, Taylor expansion of each relaxed supermodular potential \( f(x, y) = \alpha \cdot x y \) produces a linear approximation (planes in b) that agrees with \( f \) at three out of four possible discrete configurations (points A,B,C,D).

The standard trust region sub-problem is to minimize approximation \( E_t \) within the region defined by step size \( d_t \)

\[
S^* = \arg \min_{\|S - S_t\| < d_t} E_t(S)
\]

Hamming, \( L_2 \), and other useful metrics \( \|S - S_t\| \) can be represented by a sum of unary potentials [5].

Optimization problem (4) is NP-hard even for unary metrics\(^3\). One can solve Lagrangian dual of (4) by iterative sequence of graph cuts as in [23], but the corresponding duality gap may be large and the optimum for (4) is not guaranteed.

Instead of (4) we use a simpler formulation of the trust region subproblem proposed in [10]. It is based on unconstrained optimization of submodular Lagrangian

\[
L_t(S) = E_t(S) + \lambda_t \cdot \|S - S_t\|
\]

where parameter \( \lambda_t \) controls the trust region size indirectly. Each iteration of LSA-TR solves (5) for some fixed \( \lambda_t \) and adaptively changes \( \lambda_t \) for the next iteration (Alg. 1 line 10), as motivated by empirical inverse proportionality relation between \( \lambda_t \) and \( d_t \) discussed in [10].

Once a candidate solution \( S^* \) is obtained, the quality of the approximation is measured using the ratio between the actual and predicted reduction in energy. Based on this ratio, the solution is updated in line 8 and the step size (or \( \lambda \)) is adjusted in line 10. It is common to set the parameter \( \tau_1 \) in line 8 to zero, meaning that any candidate solution that decreases the actual energy gets accepted. The parameter \( \tau_2 \) in line 10 is usually set to 0.25 [25]. Reduction ratio above this value corresponds to good approximation model allowing increase in the trust region size.

Algorithm 1: General Trust Region Approach

1 Initialize \( t = 0, S_0, \lambda_0 \)
2 Repeat
3 //Solve Trust Region Sub-Problem
4 \( S^* \leftarrow \arg \min_{\Delta \in [0,1]} L_t(S) \) // as defined in (5)
5 \( P = E_t(S_t) - E_t(S^*) \) //predicted reduction in energy
6 \( R = E(S_t) - E(S^*) \) //actual reduction in energy
7 //Update current solution
8 \( S_{t+1} \leftarrow \begin{cases} S^* & \text{if } R/P > \tau_1 \\ S_t & \text{otherwise} \end{cases} \)
9 //Adjust the trust region
10 \( \lambda_{t+1} \leftarrow \begin{cases} \lambda_t / \alpha & \text{if } R/P > \tau_2 \\ \lambda_t \cdot \alpha & \text{otherwise} \end{cases} \)
11 Until Convergence

\(^3\)By a reduction to the balanced cut problem.
functions that agree with variables linear upper bounds at configurations (0, 0) and (1, 1) selects either orange or green plane in (b), or anything in-between, e.g. the purple plane passing through A and D in (c).

2.2. LSA-AUX

Bound optimization techniques are a class of iterative optimization algorithms constructing and optimizing upper bounds, a.k.a. auxiliary functions, for energy E. It is assumed that those bounds are easier to optimize than the original energy E. Given a current solution S_i, the function A_i(S) is an auxiliary function of E if it satisfies the following conditions:

\[ E(S) \leq A_i(S) \quad \text{for all } S \in \mathcal{S} \]

\[ E(S_i) = A_i(S_i) \quad \text{for } i = 0 \]

To approximate minimization of E, one can iteratively minimize a sequence of auxiliary functions:

\[ S_{t+1} = \arg \min_S A_i(S), \quad t = 1, 2, \ldots \quad (7) \]

Using (6a), (6b), and (7), it is straightforward to prove that the solutions in (7) correspond to a sequence of decreasing energy values E(S_t).

As clear from Fig.2(b,c), there are many other possible linear upper bounds for pairwise terms f. Interestingly, the "permutation" approach to high-order supermodular terms in [17] reduces to linear upper bounds for f(x, y) where each configuration (0, 0) or (1, 1) selects either orange or green plane randomly (depending on a permutation). Our tests showed inferior performance of such bounds for pairwise energies. The upper bounds using purple planes for (0, 0) and (1, 1), as in the table, work better in practice.
Summing upper bounds for all pairwise potentials $f_{pq}$ in (8) using linear terms in this table gives an overall linear upper bound for supermodular part of energy (1): 
\[
E^{\text{sup}}(S) \leq S^T U_t \tag{9}
\]
where vector $U_t = \{u_p^t | p \in \Omega\}$ consists of elements 
\[
u^t_p = \sum_q \frac{m^t_{pq}}{2} \left(1 + s^t_p - 2s^t_p s^t_q\right)
\]
and $S_t = \{s^t_p | p \in \Omega\}$ is the current solution configuration for all pixels. Defining our auxiliary function as 
\[A_t(S) := S^T U_t + E^{\text{sub}}(S) \tag{10}\]
and using inequality (9) we satisfy condition (6a): 
\[E(S) = E^{\text{sup}}(S) + E^{\text{sub}}(S) \leq A_t(S)\]
Since $S^T U_t = E^{\text{sup}}(S_t)$ then our auxiliary function (10) also satisfies condition (6b): 
\[E(S_t) = E^{\text{sup}}(S_t) + E^{\text{sub}}(S_t) = A_t(S_t)\]
Function $A_t(S)$ is submodular. Thus, we can globally optimize it in each iterations guaranteeing energy decrease.

3. Applications

Below we apply our method in several applications such as binary deconvolution, segmentation with repulsion, curvature regularization and inpainting. We report results for both LSA-TR and LSA-AUX frameworks and compare to existing state of the art methods such as QPBO [21], LBP [20], IPFP [16], TRWS and SRMP [13] in terms of energy and running time. For the sake of completeness, and to demonstrate the advantage of non-linear submodular approximations over linear approximations, we also compare to a version of LSA-TR where both submodular and supermodular terms are linearized, denoted by LSA-TR-L. In the following experiments, all local approximation methods, e.g. IPFP, LSA-AUX, LSA-AUX-P, LSA-TR, LSA-TR-L are initialized with the entire domain assigned to the foreground. All global linearization methods, e.g. TRWS, SRMP and LBP, are run for 50, 100, 1000 and 5000 iterations. For QPBO results, unlabeled pixels are shown in gray color. Running time is shown in log-scale for clarity.

3.1. Binary Deconvolution

Figures 3 (top-left) shows a binary image after convolution with a uniform $3 \times 3$ and adding Gaussian noise ($\sigma = 0.05$). The goal of binary deconvolution is to recover

---

*We used [code](http://pub.ist.ac.at/~vnk/software.html) for SRMP and [code](http://www.robots.ox.ac.uk/~ojw) for QPBO, TRWS, and LBP. The corresponding version of LPB is sequential without damping.*
the original binary image and the energy is defined as

\[ E(S) = \sum_{p \in \Omega} \left( I_p - \frac{1}{9} \sum_{q \in N_p} s_q \right)^2 \]  

(11)

Here \( N_p \) denotes the 3x3 neighborhood window around pixel \( p \) and all pairwise interactions are supermodular. We did not use length regularization, since it would make the energy easier to optimize. Fig. 3 demonstrates the performance of our approach (LSA-TR/LSA-AUX) and compares to standard optimization methods such as QPBO, LBP, IPFP, TRWS and SRMP. In this case LSA-TR-L and LSA-TR are identical since energy (11) has no submodular pairwise terms. The bottom of Fig. 3 shows the mean energy as a function of noise level \( \sigma \). For each experiment the results are averaged over ten instances of random noise. The mean time is reported for the experiments with \( \sigma = 0.05 \).

### 3.2. Segmentation with Repulsion

In this section we consider segmentation with attraction and repulsion pairwise potentials. Adding repulsion is similar to correlation clustering \([1]\), where data points either attract or repulse each other. Using negative repulsion in segmentation can avoid the bias of submodular length regularizer to short-cutting, whereby elongated structures are shortened to avoid high length penalty. Figure 4 (top-left) shows an example of an angiogram image with elongated structures. We use 16-neighborhood system and the pairwise potentials are defined as follows:

\[ \omega(p, q) = -\Delta(p, q) + c \]

Here \( \Delta(p, q) \) denotes the distance between image pixels \( p \) and \( q \) and \( \Delta(p, q) \) is the difference in their respective intensities (see pairwise potentials in Fig. 4, bottom-left). The constant \( c \) is used to make neighboring pixels with similar intensities attract and repulse otherwise. Being supermodular, repulsions potentials make the segmentation energy more difficult to optimize, but are capable to extract thin elongated structures. To demonstrate the usefulness of “repulsion” potentials we also show segmentation results with graph-cut a la Boykov-Jolly \([4]\) where negative pairwise potentials were removed/truncated (top-right).

### 3.3. Curvature

Below we apply our optimization method to curvature regularization. We focus on the curvature model proposed in \([7]\). The model is defined in terms of 4-neighborhood system and accounts for 90 degrees angles. In combination with appearance terms, the model yields discrete bi-
nary energy that has both submodular and non-submodular pairwise potentials. Originally, the authors of [7] proposed using QPBO for optimization of the curvature regularizer. We show that our method significantly outperforms QPBO and other state-of-the-art optimization techniques, especially with large regularizer weights.

First we deliberately choose a toy example (white circle on a black background, see Fig. 5) where we know what an optimal solution should look like. When using 4-neighborhood system, as the weight of the curvature regularizer increases, the solution should minimize the number of 90 degrees angles (corners) while maximizing the appearance terms. Therefore, when the weight of curvature regularizer is high, the solution should look more like a square than a circle. Consider the segmentation results in Fig. 5. With low curvature weight, i.e., $\lambda_{\text{curv}} = 0.1$, all compared methods perform equally well (see Fig. 5 top row). In this case appearance data terms are strong compared to the non-submodular pairwise terms. However, when we increase the curvature weight and set $\lambda_{\text{curv}} = 0.5$ or 2 there is a significant difference between the optimization methods both in terms of the energy and the resulting solutions (see Fig. 5 middle and bottom).

Next, we selected an angiogram image example from [7] and evaluate the performance of the optimization methods with two values of regularizer weight $\lambda_{\text{curv}} = 19$ and $\lambda_{\text{curv}} = 21$ (see Fig. 7). Although the weight $\lambda$ did not change significantly, the quality of the segmentation deteriorated for all global linearization methods, namely QPBO, TRWS, LBP. The proposed methods LSA-TR and LSA-AUX seem to be robust with respect to the weight of the supermodular part of the energy.

3.4. Chinese Characters Inpainting

Below we consider the task of in-painting in binary images of Chinese characters, df-chinesechar [11]. We used a set of pre-trained unary and pairwise potentials provided by the authors with the dataset. While each pixel variable has only two possible labels, the topology of the resulting graph and the non-submodularity of its pairwise potentials makes this problem challenging. Figure 6 shows two examples of inpainting. Table 1 reports the performance of our LSA-TR and LSA-AUX methods on this problem and compares to other standard optimization methods reported in [11], as well as, to truncation of non-submodular terms. LSA-TR is ranked second, but runs three orders of magnitudes faster.

4. Conclusions and Future Work

There are additional applications (beyond the scope of this paper) that can benefit from efficient optimization of binary non-submodular pairwise energies. For instance, our experiments show that our approach can improve non-submodular $\alpha$-expansion and fusion moves for multilabel energies. Moreover, while our paper focuses on pairwise interactions, our approach naturally extends to high-order potentials that appear in computer vision problems such as curvature regularization, convexity shape prior, visibility and silhouette consistency in multi-view reconstruction.

In the companion paper [18] we apply our method for optimization of a new highly accurate curvature regularization model. The model yields energy with triple clique interactions and our method achieves state-of-the-art performance.

Acknowledgments

We greatly thank V. Kolmogorov and our anonymous reviewers for their thorough feedback. We also thank NSERC and NIH for their grants supporting this project.

References

Figure 7. Curvature regularizer [7]: we show segmentation results and energy plots for $\lambda_{\text{curv}}=19$ (left), $\lambda_{\text{curv}}=21$ (right).


