On Approximate Non-submodular Minimization via Tree-Structured Supermodularity

Yoshinobu Kawahara
The Inst. of Sci. and Ind. Res. (ISIR),
Osaka University
ykawahara@sanken.osaka-u.ac.jp

Rishabh Iyer **Jeffery A. Bilmes**
Dept. of Electrical Engineering,
University of Washington,
{rkiyer,bilmes}@u.washington.edu

Abstract

We address the problem of minimizing non-submodular functions where the supermodularity is restricted to tree-structured pairwise terms. We are motivated by several real world applications, which require submodularity along with structured supermodularity, and this forms a rich class of expressive models, where the non-submodularity is restricted to a tree. While this problem is NP hard (as we show), we develop several practical algorithms to find approximate and near-optimal solutions for this problem, some of which provide lower and others of which provide upper bounds thereby allowing us to compute a tightness gap for any problem. We compare our algorithms on synthetic data, and also demonstrate the advantage of the formulation on the real world application of image segmentation, where we incorporate structured supermodularity into higher-order submodular energy minimization.

1 Introduction

Minimizing submodular functions, which appears in a variety of problems in machine learning and related fields, has been actively studied for several decades. This problem is polynomially-solvable and several efficient algorithms have been developed [15, 7]. While submodularity is natural in many applications, for others it is restricting from a modeling perspective, and thus much recent work has focused on non-submodular optimization [13, 5, 17, 9, 10]. Algorithms for this are either combinatorial (like greedy or local search) [13, 5] or rely on relaxations [22, 23]. This occurs very naturally, for example, in the context of inference in Markov random fields and image segmentation [22, 2, 10].

Unfortunately, the most general formulation of this problem is a difference of submodular functions, that is if $h(A) = f(A) + (-g(A)) = f(A) + \bar{g}(A)$ where g and f are submodular ($\bar{g} = -g$ is supermodular), then h can represent any discrete set function. Minimizing such functions is very hard and in fact inapproximable [9]. In some applications, however, we do not require this most general form where restricted supermodularity suffices. In this paper, we consider non-submodular minimization where supermodularity comes only from terms with specific structures, *i.e.*, tree-structured pairwise, and later generalize these to arbitrary pairwise terms. That is, $\bar{g} = \sum_{(i,j)\in\mathcal{E}} \phi_{ij}$ where $\mathcal{E}$ are the edges of a tree or a graph. This is an important special case of non-submodular minimization, where we could use its specific structure of the problem to obtain a practical algorithm or to incorporate some prior information into submodular minimization. Thus, we are additively combining the extremes of polytime solvable problems: on the one hand, we have f which is submodular, but regardless of the tree-width, we can minimize it in polynomial time; and on the other hand, we have $\bar{g}$ that, in the case of a tree, can be minimized exactly and efficiently using dynamic programming even when it uses non-submodular potentials.

We are motivated by several real world applications, where we want to model structured supermodularity along with submodularity. For example, in the context of image segmentation, submodularity represents the smoothness (attractive potentials) in an image while supermodularity represents the roughness (repulsive potentials). Thus, by incorporating supermodular terms on reliable edges obtained by some detector in addition to a submodular energy, we might expect to get better segmentation than when using the submodular energy alone. For example, a supermodular forest or tree could be used to add encouragement for certain pairs of pixels to be labeled unequally, and forest edges could be created perpendicularly across detected image edges obtained via a separate image edge detection algorithm. We show results for this application in Section 4.2.

We develop four distinct algorithms having different properties, alternating minimization, dual decomposition, continuous relaxation/rounding, and the submodular-supermodular procedure, each of which exploits the structure of the supermodular term and the submodularity of the submodular term. These algorithms together provide both upper and lower bounds to the problem, and thus are useful to obtain an approximate or ϵ-optimal solutions. We compare the performance of our algorithms on synthetic data thereby providing evidence for which should be used in applications, and also demonstrate the advantage of the formulation in image segmentation.

Notations and Preliminaries: We denote by $\hat{f}$ *the Lovász extension* of a set function f, *i.e.*, a continuous function $\hat{f} : \mathbb{R}^{\mathcal{V}} \rightarrow \mathbb{R}$ defined by $\hat{f}(\mathbf{x}) = \sum_{j=1}^{l-1}(\hat{x}_j - \hat{x}_{j+1})f(\mathcal{U}_j) + \hat{x}_l f(\mathcal{V})$, where $\mathcal{U}_j = \{i \in \mathcal{V} : x_i \geq \hat{x}_j\}$ and $\hat{x}_1 > \cdots > \hat{x}_l$ are the m distinct values in the elements of $\mathbf{x} \in \mathbb{R}^{\mathcal{V}}$.Also, a bi-set function $g : 2^{2\mathcal{V}} \rightarrow \mathbb{R}$ is called *simple bi-submodular* [21] if $g(\mathcal{S}, \mathcal{T}) + g(\mathcal{S}', \mathcal{T}') \geq g(\mathcal{S} \cup \mathcal{S}', \mathcal{T} \cup \mathcal{T}') + g(\mathcal{S} \cap \mathcal{S}', \mathcal{T} \cap \mathcal{T}')$ for all $(\mathcal{S}, \mathcal{T}), (\mathcal{S}', \mathcal{T}') \in 2^{2\mathcal{V}}$. This definition means that if we fix one of the coordinates of $g(\mathcal{S}, \mathcal{T})$, it is a submodular function in the other coordinate. We denote by $\boldsymbol{e}_{\mathcal{S}} \in \{0, 1\}^{\mathcal{V}}$, *the characteristic vector* of $\mathcal{S} \subseteq \mathcal{V}$, *i.e.*, $\boldsymbol{e}_{\mathcal{S}} = \sum_{i \in \mathcal{S}} \boldsymbol{e}_i$ ($\boldsymbol{e}_i$ is the i-th unit vector).

2 Non-submodular Minimization with Tree-Structured Supermodularity

In this section, we first formulate non-submodular minimization with tree-structured supermodularity, and then further characterize this problem. Given a finite set $\mathcal{V} := \{1, \ldots, d\}$ and a tree $\mathcal{T} = (\mathcal{E}, \mathcal{V})$ whose vertices correspond to the elements in $\mathcal{V}$, we consider the following optimization problem:

$$\min_{\boldsymbol{x} \in \{0,1\}^{\mathcal{V}}} E(\boldsymbol{x}) = \min_{\boldsymbol{x} \in \{0,1\}^{\mathcal{V}}} f(\mathcal{S}(\boldsymbol{x})) + \sum_{(i,j) \in \mathcal{E}} \psi_{ij}(x_i, x_j), \tag{1}$$

where $f : 2^{\mathcal{V}} \rightarrow \mathbb{R}$ is a submodular function, $\mathcal{S} : \{0, 1\}^{\mathcal{V}} \rightarrow 2^{\mathcal{V}}$ is a mapping from a characteristic vector to the corresponding subset and $\psi_{ij} : \{0, 1\}^2 \rightarrow \mathbb{R}_+$ is strictly supermodular on a pair in $\mathcal{E}$. That is, ψ_{ij} satisfies the following inequality equation:

$$\psi_{ij}(0, 0) + \psi_{ij}(1, 1) > \psi_{ij}(1, 0) + \psi_{ij}(0, 1). \tag{2}$$

The objective in problem (1) is therefore not submodular. Also, there is no loss or gain in generality by requiring the pairwise functions ψ_{ij} to be strictly supermodular as any modularity (or pairwise submodularity) can be absorbed into f. Unfortunately, this problem is already NP hard.

Theorem 1. *Problem* (1)*, where $\mathcal{E}$ are the edges of a forest, is NP hard.*

Proof. The idea is to reduce this problem to the vertex cover problem. Given an instance of the vertex cover problem, *i.e.*, a graph $G = (V, E)$, define an auxiliary graph $\hat{G} = (\hat{V}, \hat{E})$ as follows. $\hat{V} = V \cup \bar{V}$, where $\bar{V}$ are a set of $|V|$ appended vertices $\bar{V} = \{\bar{i}, i \in V\}$. Furthermore, $\hat{E} = \{(i, j), (i, \bar{j}), (\bar{i}, j), (\bar{i}, \bar{j})\}, \forall (i, j) \in E$. Hence the auxiliary graph has $2|V|$ vertices and $4|E|$ edges. Now define the submodular function as $f(X) = \sum_{(i,j) \in E} C(1 - x_i)x_{\bar{j}} + \sum_{i \in V} x_i$. Note that f is a pairwise submodular function. Define the supermodular tree function as $T(X) = \sum_{i \in V} C.I(x_i = x_{\bar{i}})$. In both functions, ensure that the constant $C \geq n$. Then for any vertex cover, X in G, we have that $f(X) + T(X) = |X|$. Furthermore, if X is not a vertex cover, the term $\sum_{(i,j) \in E} C(1 - x_i)x_{\bar{j}} + \sum_{i \in V} C.I(x_i = x_{\bar{i}}) > 0$, and hence, $f(X) \geq C \geq n$. Correspondingly,the minimum solution to this problem, is the minimum vertex cover, which is NP hard to find. □

The minimization problem (1) is a special case of non-submodular minimization problems, where the non-submodularity comes only from the tree-structured term. The above theorem shows that even restricting the supermodularity to a tree does not help in terms of the hardness. We shall, however, provide several approximate and near optimal algorithms to this problem that are, fortunately, made possible by the fact that the supermodularity is restricted to a tree.

3 Optimization and Bounds Estimation

The four approaches to problem (1) are described in the following subsection: alternating minimization (AM) in subsection 3.1; dual decomposition (DD) in subsection 3.2, continuous relaxation (CR) in subsection 3.3; and submodular-supermodular procedure (SSP) in subsection 3.4. Our algorithms require submodular minimization as a subroutine, which can be performed efficiently in general and even more efficiently for sub-classes of submodular functions (e.g., generalized graph-cuts [11, 16]). Exact inference on trees is always efficient using dynamic programming.

Algorithm 1 Alternating minimization (AM).

Input: α and $\boldsymbol{y}_0$. **Output:** $\boldsymbol{x}_{t-1}$ and $\boldsymbol{y}_{t-1}$.

1: Set $t \leftarrow 1$.
2: **while** not converged **do**
3: $\quad \boldsymbol{x}_t \leftarrow \mathrm{argmin}_{\boldsymbol{x}\in\{0,1\}^{\mathcal{V}}} \mathcal{E}_\alpha(\boldsymbol{x}, \boldsymbol{y}_{t-1})$.
4: $\quad \boldsymbol{y}_t \leftarrow \mathrm{argmin}_{\boldsymbol{y}\in\{0,1\}^{\mathcal{V}}} \mathcal{E}_\alpha(\boldsymbol{x}_t, \boldsymbol{y})$.
5: $\quad t \leftarrow t+1$.
6: **end while**

Algorithm 2 AM with greedy scheduling (AM-Greedy).

Input: $\boldsymbol{y}_0$. **Output:** $\boldsymbol{x}_{t-1}$.

1: Set $\alpha_0 \leftarrow 0$ and $t \leftarrow 1$.
2: **repeat**
3: $\quad \alpha_t \leftarrow \mathrm{argmax}_\alpha [\mathcal{E}_\alpha(\boldsymbol{x}_{t-1}, \boldsymbol{y}_{t-1}) - \mathcal{E}_\alpha(\boldsymbol{x}^*_{\alpha,\boldsymbol{y}_{t-1}}, \boldsymbol{y}^*_{\alpha,\boldsymbol{y}_{t-1}})]$
$\quad$ $((\boldsymbol{x}^*_{\alpha,\boldsymbol{y}_{t-1}}, \boldsymbol{y}^*_{\alpha,\boldsymbol{y}_{t-1}})$ is the output of $AM(\alpha, \boldsymbol{y}_{t-1}))$.
4: $\quad (\boldsymbol{x}_t, \boldsymbol{y}_t) \leftarrow AM(\alpha_t, \boldsymbol{y}_{t-1})$.
5: $\quad t \leftarrow t+1$.
6: **until** $\boldsymbol{x}_{t-1} = \boldsymbol{y}_{t-1}$ holds

3.1 Alternating Minimization with Scheduling (AM)

Since exact minimization of each term individually in Eq. (1) can be efficiently solved, one can apply a simple alternating minimization (AM) procedure (Algorithm 1). Define the following:

$$\mathcal{E}_\alpha(\boldsymbol{x}, \boldsymbol{y}) = f(\mathcal{S}(\boldsymbol{x})) + \textstyle\sum_{(i,j)\in\mathcal{E}} \psi_{ij}(y_i, y_j) + \alpha \cdot d_h(\boldsymbol{x}, \boldsymbol{y}), \tag{3}$$

where $\alpha \geq 0$ and $d_h : \{0,1\}^{\mathcal{V}\times\mathcal{V}} \rightarrow \mathbb{R}$ is the Hamming distance. Starting with an arbitrary initial $\boldsymbol{y}_0 \in \{0,1\}^{\mathcal{V}}$, AM alternately minimizes $\mathcal{E}_\alpha(\boldsymbol{x}, \boldsymbol{y})$ with respect to $\boldsymbol{x}$ while $\boldsymbol{y}$ is held fixed and then vice-verse. That is, we iterate

$$\boldsymbol{x}_t = \mathrm{argmin}_{\boldsymbol{x}\in\{0,1\}^{\mathcal{V}}} \mathcal{E}_\alpha(\boldsymbol{x}, \boldsymbol{y}_{t-1}) \quad \text{and} \quad \boldsymbol{y}_t = \mathrm{argmin}_{\boldsymbol{y}\in\{0,1\}^{\mathcal{V}}} \mathcal{E}_\alpha(\boldsymbol{x}_t, \boldsymbol{y}).$$

This procedure always decreases the value of $\mathcal{E}_\alpha$ until convergence. Although, at convergence, $\boldsymbol{x}$ and $\boldsymbol{y}$ are not necessarily equal, the value of $\mathcal{E}_\alpha$ with these solutions offers an upper bound of the original problem (1) due to the following proposition:

Proposition 2. *Given any $\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z} \in \{0,1\}^{\mathcal{V}}$ such that $\boldsymbol{x} \neq \boldsymbol{y}$, then there exists a finite $\bar{\alpha}$ such that*

$$\mathcal{E}_{\bar{\alpha}}(\boldsymbol{x}, \boldsymbol{y}) \geq \mathcal{E}_{\bar{\alpha}}(\boldsymbol{z}, \boldsymbol{z}) = E(\boldsymbol{z}).$$

Since $d_h(\boldsymbol{x}, \boldsymbol{y}) > 0$ (due to $\boldsymbol{x} \neq \boldsymbol{y}$), the proof of this proposition is obvious from Eq. (3). The direct application of AM to Problem (3), however, does not necessarily produce a useful solution to the original problem (1). This is because, if we set a relatively large α to make $\boldsymbol{x} = \boldsymbol{y}$ at termination, the distance term in Eq. (3) is dominant and the procedure quickly gets stuck near $\boldsymbol{y}_0$. A solution is to utilize appropriate scheduling of α to ensure a sequence of solutions gradually move towards each other. How to do this optimally is an interesting issue, but in this work we introduce a simple (but reasonable) strategy to increase the value of α gradually until both solutions are equivalent. Denote a multiplier by $C > 1$, and then update α as $\alpha \leftarrow C \times \alpha$ (AM-Simple). This scheduling never makes the objective worse due to the following:

Proposition 3. *Let $\alpha_1 < \alpha_2$ and $\boldsymbol{y}_0 \in \{0,1\}^{\mathcal{V}}$. Then, if $(\boldsymbol{x}^*_1, \boldsymbol{y}^*_1) \leftarrow AM(\alpha_1, \boldsymbol{y}_0)$ and $(\boldsymbol{x}^*_2, \boldsymbol{y}^*_2) \leftarrow AM(\alpha_2, \boldsymbol{y}^*_1)$ and $(\bar{\boldsymbol{x}}^*, \bar{\boldsymbol{y}}^*) \leftarrow AM(\alpha_2, \boldsymbol{y}_0)$, then there exists some minimal value, say $\tilde{\alpha}$, for α_1 such that for all $\alpha_1 \geq \tilde{\alpha}$, we have that the objective at α_2 satisfies $\mathcal{E}_{\alpha_2}(\boldsymbol{x}^*_2, \boldsymbol{y}^*_2) \leq \mathcal{E}_{\alpha_2}(\bar{\boldsymbol{x}}^*, \bar{\boldsymbol{y}}^*)$.*

We further enhance the scheduling of α using a greedy-like procedure, as described in Algorithm 2 (AM-Greedy). Here, at each outer iteration, we choose an α (Line 3 of Algorithm 2), chosen via binary search, such that $\mathcal{E}_\alpha$ decreases the most from the previous candidate solution. This scheduling, in fact, works very well in practice as will be seen below in the experimental section.

3.2 Optimization via Dual Decomposition (DD)

In the AM approach, we ensure feasible solutions by adjusting α in a manner that works well in practice, but it is indeed heuristic. A more systematic approach (and often applied to ML problems) is dual decomposition [18], or Lagrangian relaxation, which is quite naturally applied to problem (1). We reformulate the original problem (1) in the following (equivalent) form:

$$\min_{\boldsymbol{x},\boldsymbol{y}\in\{0,1\}^{\mathcal{V}}} E(\boldsymbol{x}, \boldsymbol{y}) = \min_{\boldsymbol{x},\boldsymbol{y}\in\{0,1\}^{\mathcal{V}}} f(\mathcal{S}(\boldsymbol{x})) + \textstyle\sum_{(i,j)\in\mathcal{E}} \psi_{ij}(y_i, y_j) \ \text{ s.t. } x_i = y_i \ (i \in \mathcal{V}). \tag{4}$$

It is obvious that the solutions of the problems (1) and (4) are equivalent. Note that the minimization with respect to each subproblem in Eq. (4) is solvable efficiently via submodular minimization (the first term) or dynamic programming (the second), respectively. This motivates us to solve the dual

$$\max_{\boldsymbol{\delta}} L(\boldsymbol{\delta}) = \max_{\boldsymbol{\delta}} \min_{\boldsymbol{x},\boldsymbol{y}\in\{0,1\}^{\mathcal{V}}} E(\boldsymbol{x}, \boldsymbol{y}) + \boldsymbol{\delta}^\top (\boldsymbol{x} - \boldsymbol{y}), \tag{5}$$

where $\boldsymbol{\delta} \in \mathbb{R}^{\mathcal{V}}$ is the Lagrangian coefficients, in place of the primal one (4). This always provides a lower bound of the primal although we do not necessarily have strong duality. However, for some function $E(\boldsymbol{x})$, strong duality might hold, as stated in the following:

Proposition 4. *If there exist $\boldsymbol{\delta}^*$ and $\boldsymbol{x}^*$ such that*

$$\boldsymbol{x}^* \in \mathrm{argmin}_{\boldsymbol{x}} f(\mathcal{S}(\boldsymbol{x})) + (\boldsymbol{\delta}^*)^\top \boldsymbol{x} \ \text{ and } \ \boldsymbol{x}^* \in \mathrm{argmin}_{\boldsymbol{x}} \textstyle\sum_{(i,j)\in\mathcal{E}} \psi_{ij}(x_i, x_j) - (\boldsymbol{\delta}^*)^\top \boldsymbol{x},$$

then $\boldsymbol{x}^*$ is an optimal solution to problem (1) and hence $L(\boldsymbol{\delta}^*) = E(\boldsymbol{x}^*)$.

Similar statements appear in multiple papers on the MAP inference on MRFs [8, 18]. The conditions of the proposition correspond to the subproblems agreeing on a minimizing $\boldsymbol{x}$ of $E(\boldsymbol{x})$. Since agreement implies optimality of the dual, it can only occur after the algorithm finds the tightest lower bound. Although agreement is not guaranteed, if we do reach such a state, then Proposition 4 ensures an exact solution to problem (1). Each subproblem is still solvable efficiently. Since $L(\boldsymbol{\delta})$ is concave on $\boldsymbol{\delta}$ (and non-differentiable), the outer optimization (with respect to $\boldsymbol{\delta}$) can be performed by a subgradient methods and so on.

3.3 Continuous Relaxation and Rounding (CR)

For MAP inference on MRFs, it is well known that dual decomposition is equivalent to solving the dual of a linear programming (LP) relaxation of the original problem [22]. Therefore, many algorithms based on the LP relaxation have been actively discussed in this context. In this subsection, we consider a convex relaxation approach to problem (1). Consider the following relaxation of problem (1) to the domain $[0,1]^{\mathcal{V}}$:

$$\begin{aligned} \min_{\mathbf{x}\in[0,1]^{\mathcal{V}},\boldsymbol{\mu}\in[0,1]^{4|\mathcal{E}|}} & \hat{f}(\mathbf{x}) + \sum_{(i,j)\in\mathcal{E}} \sum_{\bar{\boldsymbol{x}}_{ij}\in\{0,1\}^2} \psi_{ij}(\bar{x}_i,\bar{x}_j)\mu_{ij,\bar{\boldsymbol{x}}_{ij}}, \\ \text{s.t.} & \sum_{\bar{\boldsymbol{x}}_{ij}\in\{0,1\}^2} \mu_{ij,\bar{\boldsymbol{x}}_{ij}} = 1,\ \sum_{i\in\mathcal{V}} \mu_{ij,\bar{\boldsymbol{x}}_{ij}} = x_j,\ \sum_{j\in\mathcal{V}} \mu_{ij,\bar{\boldsymbol{x}}_{ij}} = x_i, \end{aligned} \tag{6}$$

where $\boldsymbol{\mu} = \{\mu_{ij,\bar{\boldsymbol{x}}_{ij}}\}$ are the alternative variables for this representation, and $\hat{f}$ is the Lovász extension of f and hence is convex. Therefore, this is basically an extended formulation of LP relaxation methods for MAP inference on MRFs, but using the Lovász extension. Although several optimization methods, including dual decomposition, can be applied to this problem, we develop an algorithm based on Alternating Direction Method of Multipliers (ADMM) [6, 3] in the current subsection. This is because ADMM possesses superior convergence properties. Note that it can be intractable to solve Eq. (6) directly by a constrained convex solver due to the representation of the Lovász extension. Also note that the application of ADMM to a discrete problem is not straightforward and thus it is beyond the scope of this paper to apply it directly to problem (4).

For simplicity, we denote by $\langle\boldsymbol{\psi},\boldsymbol{\mu}\rangle$ the second term in the objective and by $A\boldsymbol{\mu} = \mathbf{1}_{|\mathcal{E}|}$ and $B\boldsymbol{\mu} = C\boldsymbol{x}$ the sets of the equality constraints. The augmented Lagrangian for ADMM is then given by

$$L_\rho(\mathbf{x},\boldsymbol{\mu},\boldsymbol{\delta}) = \hat{f}(\mathbf{x}) + \langle\boldsymbol{\psi},\boldsymbol{\mu}\rangle + \begin{bmatrix} \boldsymbol{\delta}_1 \\ \boldsymbol{\delta}_2 \end{bmatrix}^\top \begin{bmatrix} A\boldsymbol{\mu} - \mathbf{1}_{|\mathcal{E}|} \\ B\boldsymbol{\mu} - C\boldsymbol{x} \end{bmatrix} + (\rho/2)(\|A\boldsymbol{\mu} - \mathbf{1}_{|\mathcal{E}|}\|_2^2 + \|B\boldsymbol{\mu} - C\boldsymbol{x}\|_2^2),$$

where $\boldsymbol{\delta}_1 \in \mathbb{R}^{|\mathcal{E}|}$ and $\boldsymbol{\delta}_2 \in \mathbb{R}^{2|\mathcal{E}|}$ are the Lagrangian coefficient vectors and $\rho > 0$ is the penalty parameter. Then, ADMM consists of the iterates:

$$\boxed{\mathbf{x}^{k+1} \Leftarrow \operatorname{argmin}_{\mathbf{x}\in[0,1]^{\mathcal{V}}} L_\rho(\mathbf{x},\boldsymbol{\mu}^k,\boldsymbol{\delta}^k)} \quad \text{(7a)} \qquad \boxed{\boldsymbol{\mu}^{k+1} \Leftarrow \operatorname{argmin}_{\boldsymbol{\mu}\in[0,1]^{2\mathcal{V}}} L_\rho(\mathbf{x}^{k+1},\boldsymbol{\mu},\boldsymbol{\delta}^k)} \quad \text{(7b)}$$

$$\boxed{\boldsymbol{\delta}_1^{k+1} \Leftarrow \boldsymbol{\delta}_1^k + \rho(A\boldsymbol{\mu}^{k+1} - \mathbf{1}_{|\mathcal{E}|}),\ \boldsymbol{\delta}_2^{k+1} \Leftarrow \boldsymbol{\delta}_2^{k+1} + \rho(B\boldsymbol{\mu}^{k+1} - C\boldsymbol{x}^{k+1})} \quad \text{(7c)}$$

While the minimization for $\boldsymbol{\mu}$ in step (7b) is a quadratic problem, the one for $\mathbf{x}$ in step (7a) is a non-smooth convex problem. Thus, for example, we can apply proximal gradient methods to solve this step. Since the objective consists of the Lovász extension of a submodular function and a least-squares term, we can calculate the proximal operator as a minimum-norm-point (MNP) problem, as in a (not entirely straightforward) fashion similar to [1, 16]. A solution for problem (6) is not necessarily integral, and hence we must apply a rounding algorithm. Although there are several possible deterministic and randomized rounding algorithms that could be applied directly to our case (from [19]), we use the node-based rounding in the experiments below.

3.4 Submodular-Supermodular Procedure Using Pairwise Structures (SSP)

The Submodular-Supermodular procedure [17, 9] is a set of heuristics for minimizing a general set functions. [9] propose a number of different variants of the submodular-supermodular procedures that have worked well for a variety of problems. The most general algorithms do not specifically exploit the structure of the problem, and work for arbitrary sums of submodular and supermodular terms.

In SSP, we iteratively minimize the sum of a submodular function and a supermodular function by replacing the supermodular part by its (typically, modular) upper bound at every iteration. Here, we describe an efficient special form of the submodular-supermodular procedure for minimizing $E(\boldsymbol{x})$ in Eq. (1) with a modular upper bound that can be calculated easily based on the structure of a pairwise supermodular function. For each pair $(i,j) \in \mathcal{E}(\boldsymbol{x})$, let us define $\mathbf{b}_{ij}^1, \mathbf{b}_{ij}^2 \in \mathbb{R}^2$, respectively, as

$$\mathbf{b}_{ij}^1 = \begin{bmatrix} \psi_{ij}(1,0) \\ \psi_{ij}(1,1) + \psi_{ij}(0,0) - \psi_{ij}(1,0) \end{bmatrix} \text{ and } \mathbf{b}_{ij}^2 = \begin{bmatrix} \psi_{ij}(1,1) + \psi_{ij}(0,0) - \psi_{ij}(0,1) \\ \psi_{ij}(0,1) \end{bmatrix}$$

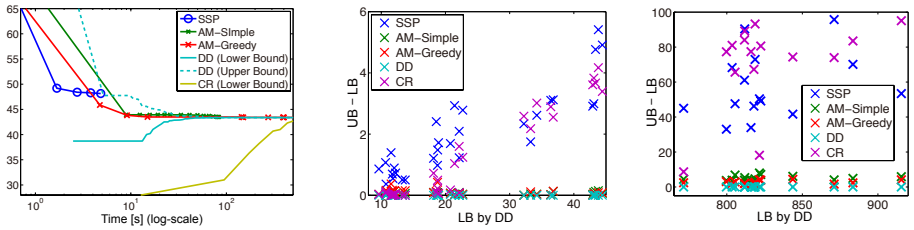

Figure 1: Typical examples (for a concave-of-modular function of $|\mathcal{V}| = 324$) of solution sequences by the algorithms for the cases with a tree-structured supermodular term, and lower bounds (by DD) vs. ϵ-optimality, *i.e.* (upper bounds by the algorithms)− (LB (by DD)), for several instances (left: concave-of-modular funct., right: objective in [14] with tree-structured term).

Note that all elements in $\mathbf{b}^1_{ij}$ and $\mathbf{b}^2_{ij}$ are always positive from the supermodularity and the positivity of ψ_{ij}. Then, $\psi_{ij}(x_i, x_j)$ can be represented as

$$\psi_{ij}(x_i, x_j) = \psi_{ij}(0,0) \cdot (1 - \boldsymbol{x}_{ij}^\top \mathbf{1}_2) + \min\{\boldsymbol{x}_{ij}^\top \mathbf{b}^1_{ij}, \boldsymbol{x}_{ij}^\top \mathbf{b}^2_{ij}\}, \tag{8}$$

where $\boldsymbol{x}_{ij} = [x_i, x_j]^\top$. This function is bi-submodular. Then, for a given $\bar{\boldsymbol{x}}_{ij}$, define a vector $\tilde{\mathbf{b}}_{ij}$ as

$$\tilde{\mathbf{b}}_{ij} = \mathbf{b}^1_{ij} \text{ (if } \bar{\boldsymbol{x}}_{ij} = (1,0)^\top), \ \mathbf{b}^2_{ij} \text{ (if } \bar{\boldsymbol{x}}_{ij} = (0,1)^\top) \text{ and } (\mathbf{b}^1_{ij} + \mathbf{b}^2_{ij})/2 \text{ (otherwise)}. \tag{9}$$

Then, $h_{\bar{\boldsymbol{x}}_{ij}}(\boldsymbol{x}_{ij}) := \psi_{ij}(0,0)(1 - \boldsymbol{x}_{ij}^\top \mathbf{1}_2) + \boldsymbol{x}_{ij}^\top \tilde{\mathbf{b}}_{ij}$is a modular upper bound that is tight with respect to a given point $\bar{\boldsymbol{x}}_{ij}$, which can be calculated easily. By summing up this for all pairs in $\mathcal{E}$, we have a modular upper bound of the original supermodular term. Thus, we can apply SSP with $h_{\bar{\boldsymbol{x}}_{ij}}$, and iteratively solve $\boldsymbol{x}_{t+1} \leftarrow \operatorname{argmin}_{\boldsymbol{x} \in \{0,1\}^\nu} f(\mathcal{S}(\boldsymbol{x})) + \sum_{(i,j)\in\mathcal{E}} h_{\bar{\boldsymbol{x}}_{ij}}(\boldsymbol{x}_{ij})$, until convergence.

4 Experimental Evaluation

In Subsection 4.1, we investigate and compare the performance of the four proposed approaches on synthetic data. Then, we apply them to image segmentation in Subsection 4.2. The experiments below were run on a 2.6 GHz 64-bit WS using Matlab. For the calculation of a maximum flow problem (for the second), we used a C++ implementation modified from the shared code by Kohli et al. [12].[1]

4.1 Evaluation on Synthetic Data

Our first experiment was performed with synthetic data generated as follows. First for a submodular function, we used a concave-over-modular function $\sqrt{\mathbf{w}_1(\mathcal{S}) + \alpha \mathbf{w}_2(\mathcal{V} \setminus \mathcal{S})}$ with randomly chosen vectors $\mathbf{w}_1$, $\mathbf{w}_2$ in $[0,1]^n$, and the objective in [14], $f(\mathcal{S}) = \sum_{i\in\mathcal{V}} \sum_{j\in\mathcal{S}} s_{ij} - \lambda \sum_{i,j\in\mathcal{S}} s_{ij}$, where λ is a redundancy parameter and $\{s_{ij}\}$ is a random similarity matrix. And for the supermodular part, we randomly generated a tree or an undirected graph over nodes $\mathcal{V}$, where we used GENRMF available from DIMACS Challenge[2] to generate a graph and also find a minimum spanning tree on the graph for the tree case. We created a supermodular potential by first randomly assigning values $\psi_{ij}(0,0)$ and $\psi_{ij}(1,1)$ in $[0,1]$, and then randomly giving values on $\psi_{ij}(0,1)$ and $\psi_{ij}(1,0)$ such that these satisfy the inequality in Eq. (2). Figure 1 shows typical examples of algorithmic convergence for a concave-over-modular function with a supermodular tree-structured, where $|\mathcal{V}| = 324$. Also, Figure 1 shows the plots of lower bounds (by CR) vs. upper bounds by the algorithms for different sizes of instances for the cases with two submodular functions and a tree-structured supermodular term. In general, CR ran significantly slower and didn't complete in some cases. SSP obtained a reasonable solution very quickly, but then got stuck at local optima — a reasonable hybrid approach would be to warm-start DD with SSP (not shown).

4.2 Application to Image Segmentation

We formulated segmentation in an image as problem (1), where we used the tree-structured term to incorporate information about edges into submodular energy minimization. As a submodular potential, we used the robust P^n Potts model [12] (with binary labels), where the unary, pairwise and higher-order terms are respectively given by

$$\phi_i(x_i) = \theta_T \phi_T(x_i) + \theta_{col}\phi_{col}(x_i) + \theta_l \phi_l(x_i) \ (i \in \mathcal{V}),$$
$$\phi_{ij}(x_i, x_j) = 0 \text{ (if } x_i = x_j), \ \theta_p + \theta_v \exp(\theta_\beta \|I_i - I_j\|^2) \text{ (otherwise)},$$
$$\phi_c(\boldsymbol{x}_c) = 0 \text{ (if } x_i = l_c, \ \forall i \in c), |c|^{\theta_\alpha}(\theta^h_p + \theta^h_v \exp(-(\theta^h_\beta/|c|)\|\textstyle\sum_{i\in c}(f(i) - \mu)^2\|) \text{ (otherwise)},$$

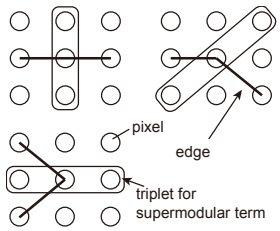

Figure 2: Construction of triplets for the supermodular term.

[1] http://research.microsoft.com/en-us/um/people/pkohli/code.html

[2] The 1st DIMACS Int'l Algo. Implmentation Challenge, 1990. See http://dimacs.rutgers.edu/Challenges/.

Figure 3: Original image (left), supermodular forest (second column), segmentation result with the robust P^n model (third column) and segmentation result with our formulation (forth column).

where ϕ_T, ϕ_{col} and ϕ_l are potentials from TextonBoost [20],[3] color and location, I_i and I_j are the color vectors of pixel i and j, $f(\cdot)$ is a function evaluated on all constituent pixels of the superpixel c and $\mu = \sum_{i \in c} f(i)/|c|$, respectively. Here, we used binary-segmented images in MSRC data [20] ($1/2$ images were used for training) and applied similar parameters to the ones used in [12]:

$$\theta_T = 0.7, \theta_{col} = 0.2, \theta_l = 0.27, \theta_p = 1.0, \theta_v = 1.5, \theta_\beta = 8.0, \theta_\alpha = 0.8, \theta_p^h = 0.2, \theta_v^h = 0.5, \theta_\beta^h = 12.0.$$

Segments $\mathcal{C}$ for the higher-order potential were generated from mean-shift [4]. Moreover, with edges obtained by a popular edge detector (the Prewitt method), the tree-structured supermodular term was constructed as follows: first, we generated a set of triplets of pixels that stride across detected image edge boundaries, as shown in Figure 2 and Figure 3 (2nd col.). Then, we found a minimum-spanning tree (forest) on pixels included in all triplets and set a supermodular potential on each pair in the tree. Figure 3 (3rd col.) shows an example of a typical result by the robust P^n Potts model (which either entirely misses the background between the bench slats (shown), or removes significant portions of the bench from the foreground (not shown)), and our formulation which reduces this problem (4th col.). For the optimization in our formulation, we used AM-Greedy (Algorithm 2) with the maximum flow algorithm. As can be seen, the addition of the supermodular forest significantly improves the quality of the segmentation in background regions where the submodular-only potential fails.

References

[1] F. Bach. Structured sparsity-inducing norms through submodular functions. In *Adv. in NIPS*, volume 23, pages 118–126. 2010.

[2] L. Bordeaux, Y. Hamadi, and P. Kohli. *Tractability: Practical Approaches to Hard Problems*. Cambridge University Press, 2014.

[3] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Found. & Trends in Mach. Learn.*, 3(1):1–122, 2011.

[4] D. Comaniciu and P. Meer. Mean shift: A robust approach toward feature space analysis. *IEEE PAMI*, 24(5):603–619, 2002.

[5] D.-Z. Du, R.L. Graham, P.M. Pardalos, P.-J. Wan, W. Wu, and W. Zhal. Analysis of greedy approximations with nonsubmodular potential functions. In *Proc. of the 19th Ann. ACM-SIAM Symp. on Discrete Algorithm (SODA'08)*, pages 167–175, 2008.

[6] J. Eckstein and D.P. Bertsekas. On the Douglas-Rachford splitting method and the proximal point algorithm for maximal monotone operators. *Mathematical Programming*, 55(1-3):293–318, 1992.

[7] S. Fujishige. *Submodular Functions and Optimization*. Elsevier, 2nd edition, 2005.

[8] A.M. Geoffrion. Lagrangian relaxation for integer programming. *Math. Prog. Study*, 2:82–114, 1974.

[9] R. Iyer and J.A. Bilmes. Algorithms for approximate minimization of the difference between submodular functions, with applications. In *UAI*, 2012.

[10] S. Jegelka and J. Bilmes. Submodularity beyond submodular energies: Coupling edges in graph cuts. In *CVPR*, pages 1897–1904, 2011.

[11] S. Jegelka, H. Liu, and J.A. Bilmes. On fast approximate submodular minimization. In *Adv. in NIPS*, volume 24, pages 460–468. 2011.

[12] P. Kohli, L. Ladický, and P.H.S. Torr. Robust higher order potentials for enforcing label consistency. *International Journal of Computer Vision*, 82:302–324, 2009.

[13] V. Kolmogorov and C. Rother. Minimizing nonsubmodular functions with graph cuts – A review. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 29(7):1274–1279, 2007.

[14] H. Lin and J. Bilmes. Multi-document summarization via budgeted maximization of submodular functions. In *NAACL-HLT*, pages 912–920, 2010.

[15] L. Lovász. Submodular functions and convexity. In *Mathematical Programming – The State of the Art*, pages 235–257. 1983.

[16] K. Nagano and Y. Kawahara. Structured convex optimization under submodular constraints. In *UAI'13*, pages 459–468, 2013.

[17] M. Narasimhan and J.A. Bilmes. A submodular-supermodular procedure with applications to discriminative structure learning. In *UAI'05*, pages 404–412, 2005.

[18] N. N.Komodakis, N. Paragios, and G. Tziritas. Mrf energy minimization and beyond via dual decomposition. *IEEE PAMI*, 33(3):531–552, 2011.

[19] P. Ravikumar, A. Agarwal, and M.J. Wainwright. Message-passing for graph-structured linear programs: Proximal methods and rounding schemes. *Journal of Machine Learning Research*, 11:1043–1080, 2010.

[20] J. Shotton, J. Winn, C. Rother, and A. Criminisi. Textonboost: Joint appearance, shape and context modeling for multi-class object recognition and segmentation. In *ECCV*, pages 1–15, 2006.

[21] A.P. Singh, A. Guillory, and J.A. Bilmes. On bisubmodular maximization. In *Proc. of the 15th Int'l Conf. on Artificial Intelligence and Statistics (AISTATS'12)*, pages 1055–1063, 2012.

[22] M.J. Wainwright and M.I. Jordan. Graphical models, exponential families, and variational inference. *Foundations and Trends in Machine Learning*, 1(1–2):1–305, 2008.

[23] T. Werner. Revisiting the linear programming relaxation approach to Gibbs energy minimization and weighted constraint satisfaction. *IEEE PAMI*, 32(8):1474–1488, 2010.

[3] TextonBoost is originally a method for multiple classes segmentation. We used its outputs as the labeling on a main object and others. For the training, the half of all images in the whole MSRC data was used.