On Approximate Non-submodular Minimization via Tree-Structured Supermodularity

Yoshinobu Kawahara
Osaka University
ykawahara@sanken.osaka-u.ac.jp

Rishabh Iyer
University of Washington, Seattle
rkiyer@u.washington.edu

Jeffery A. Bilmes
University of Washington, Seattle
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. We also show that some of our algorithms can be extended to handle more general forms of supermodularity restricted to arbitrary pairwise terms. 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 [9, 22]. 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 [7, 13, 16, 19, 24]. Algorithms for this are either combinatorial (like greedy or local search) [7, 19] or rely on relaxations [29, 31]. This occurs very naturally, for example, in the context of inference in Markov random fields and image segmentation [4, 16, 30].

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 [24]. Minimizing such functions is very hard and in fact inapproximable [13]. In some applications, however, we do not require this most general form, and a restricted form of 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 might use the specific structure of the problem to obtain a practical algorithm or to incorporate 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, and regardless of the treewidth, 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 and non-supermodular 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

Appearing in Proceedings of the 18^{th} International Conference on Artificial Intelligence and Statistics (AISTATS) 2015, San Diego, CA, USA. JMLR: W&CP volume 38.

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 proof-of-concept results for this application in Section 5.2.

Our primary contributions are the development of 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.

The remainder of this paper is organized as follows. We first define notation and preliminaries. Then, in Section 2, we first formulate non-submodular minimization with a tree-structured supermodular term, and characterize this problem. Then in Section 3, we develop the four aforementioned algorithms. In Section 4, we consider the case where the supermodular term is more general, *i.e.*, consists of arbitrary pairwise supermodular functions. Finally, we investigate the empirical performance of our proposed algorithms in Section 5, and conclude in Section 6. We also describe the details for the derivation of the continuous relaxation algorithm in Appendix A. Most proofs are described in the supplementary material.

Notations and Preliminaries: A set function $f : 2^{\mathcal{V}} \to \mathbb{R}$ is called *submodular* if $f(\mathcal{S}) + f(\mathcal{T}) \geq f(\mathcal{S} \cup \mathcal{T}) + f(\mathcal{S} \cap \mathcal{T})$ for all $\mathcal{S}, \mathcal{T} \subseteq \mathcal{V}$, where $\mathcal{V} = \{1, \ldots, d\}$ is a finite set [9, 22]. If $(-f)$ is submodular, then f is called *supermodular*. We denote by $\hat{f}$ *the Lovász extension* of a set function f, *i.e.*, a continuous function $\hat{f} : \mathbb{R}^{\mathcal{V}} \to \mathbb{R}$ defined by

$$\hat{f}(\mathbf{x}) = \textstyle\sum_{j=1}^{l-1} (\hat{x}_j - \hat{x}_{j+1}) f(\mathcal{U}_j) + \hat{x}_l f(\mathcal{V}),$$

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 $\mathbf{x} \in \mathbb{R}^{\mathcal{V}}$. It is known that f is submodular if and only if $\hat{f}$ is convex [22]. Also, a bi-set function $g : 2^{2\mathcal{V}} \to \mathbb{R}$ is called *simple bi-submodular* [27] 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 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.*, $\boldsymbol{e}_{\mathcal{S}} = \sum_{i \in \mathcal{S}} \boldsymbol{e}_i$, where $\boldsymbol{e}_i$ 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 (the non-tree case is considered in Section 4). Given a finite set $\mathcal{V} := \{1, \ldots, d\}$ and a tree $\mathcal{T} = (\mathcal{V}, \mathcal{E})$ 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 $f : 2^{\mathcal{V}} \to \mathbb{R}$ is a submodular function, $\mathcal{S} : \{0,1\}^{\mathcal{V}} \to 2^{\mathcal{V}}$ is a mapping from a characteristic vector to the corresponding subset and $\psi_{ij} : \{0,1\}^2 \to \mathbb{R}_+$ 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.

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: $\quad \boldsymbol{x}_t \leftarrow \operatorname{argmin}_{\boldsymbol{x} \in \{0,1\}^{\mathcal{V}}} \mathcal{E}_\alpha(\boldsymbol{x}, \boldsymbol{y}_{t-1})$.
4: $\quad \boldsymbol{y}_t \leftarrow \operatorname{argmin}_{\boldsymbol{y} \in \{0,1\}^{\mathcal{V}}} \mathcal{E}_\alpha(\boldsymbol{x}_t, \boldsymbol{y})$ and then $t \leftarrow t + 1$.
5: **end while**

Algorithm 2 AM with simple scheduling (AM-Simple).

Input: $\alpha_0 \approx 0 (> 0)$, $\boldsymbol{y}_0$ and C. **Output:** $\boldsymbol{x}_{t-1}$.
1: Set $t \leftarrow 1$.
2: **repeat**
3: $\quad (\boldsymbol{x}_t, \boldsymbol{y}_t) \leftarrow AM(\alpha_{t-1}, \boldsymbol{y}_{t-1})$.
4: $\quad \alpha_t \leftarrow C \times \alpha_{t-1}$ and then $t \leftarrow t + 1$.
5: **until** $\boldsymbol{x}_{t-1} = \boldsymbol{y}_{t-1}$ holds

Algorithm 3 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: $\quad \alpha_t \leftarrow \operatorname{argmax}_\alpha [\mathcal{E}_\alpha(\boldsymbol{x}_{t-1}, \boldsymbol{y}_{t-1}) - \mathcal{E}_\alpha(\boldsymbol{x}^*_{\alpha, \boldsymbol{y}_{t-1}}, \boldsymbol{y}^*_{\alpha, \boldsymbol{y}_{t-1}})]$, where $(\boldsymbol{x}^*_{\alpha, \boldsymbol{y}_{t-1}}, \boldsymbol{y}^*_{\alpha, \boldsymbol{y}_{t-1}})$ is the output of $AM(\alpha, \boldsymbol{y}_{t-1})$.
4: $\quad (\boldsymbol{x}_t, \boldsymbol{y}_t) \leftarrow AM(\alpha_t, \boldsymbol{y}_{t-1})$ and then $t \leftarrow t + 1$.
5: **until** $\boldsymbol{x}_{t-1} = \boldsymbol{y}_{t-1}$ holds

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 problem (1) is a special case of non-submodular minimization problems, but here 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.

Approaches to Optimization We consider four approaches to Problem (1): Alternating Minimization (AM), Dual Decomposition (DD), Continuous Relaxation with rounding (CR), and the Submodular-Supermodular Procedure (SSP). Basically, DD and CR optimize lower bounds of the original problem while AM and SSP provide upper bounds. In each case, we theoretically analyze these algorithms, and finally empirically compare them in Section 5.

3 Optimization

The four approaches to problem (1) are described in the following subsections: AM in subsection 3.1; DD in subsection 3.2, CR in subsection 3.3; and SSP in subsection 3.4. Our algorithms require submodular minimization as a subroutine, which can be performed efficiently in general [2] and even more efficiently for sub-classes of submodular functions (e.g., generalized graph-cuts [17, 23]). Exact inference on trees is always efficient and easy using dynamic programming.

3.1 Alternating Minimization (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:

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

where $\alpha \geq 0$ and $d_h : \{0,1\}^{\mathcal{V}\times\mathcal{V}} \to \mathbb{R}$ 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.[1] That is, we iterate

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

This procedure always decreases the value of $\mathcal{E}_\alpha$ until convergence.[2] Although, at convergence, $\boldsymbol{x}$ and $\boldsymbol{y}$ are not necessarily equal, the final value of $\mathcal{E}_\alpha$ 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, Algorithm 2). This scheduling is not worse at each step than just Algorithm 1 in the sense of the following proposition:

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 $(\boldsymbol{x}_2^*,\boldsymbol{y}_2^*) \leftarrow AM(\alpha_2,\boldsymbol{y}_1^*)$*

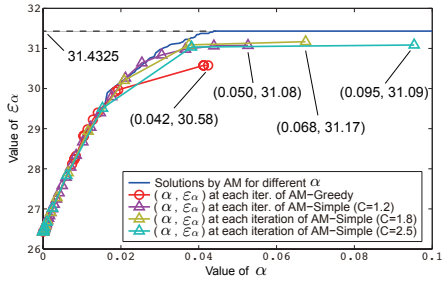

Figure 1: Sols. by AM for various α, and pairs $(\mathcal{E}_\alpha, \alpha)$ at each iter. of AM-Simple (with various Cs) and AM-Greedy.

[1] We do not necessarily need to start from $\boldsymbol{y}$, we could start from $\boldsymbol{x}$ just as well.

[2] Convergence here means $\mathcal{E}_\alpha(x_{t+1},y_{t+1}) \geq \mathcal{E}_\alpha(x_t,y_t) - \text{TOL}$ for a given α and some tolerance $\text{TOL} \geq 0$.

Algorithm 4 Dual decomposition (DD) for problem (1).

Input: $\boldsymbol{\delta}_0 \approx 0$. **Output:** $\boldsymbol{x}_t$ (and $\boldsymbol{y}_t$).
1: Set $t \leftarrow 1$.
2: **while** $\boldsymbol{\delta}_t$ does not converge **do**
3: Set $c_t \leftarrow 1/t$.
4: $\boldsymbol{x}_t \leftarrow \operatorname{argmin}_{\boldsymbol{x}\in\{0,1\}^{\mathcal{V}}} f(\mathcal{S}(\boldsymbol{x})) + \boldsymbol{\delta}_{t-1}^{\top}\boldsymbol{x}$.
5: $\boldsymbol{y}_t \leftarrow \operatorname{argmin}_{\boldsymbol{y}\in\{0,1\}^{\mathcal{V}}} \sum_{(i,j)\in\mathcal{E}} \psi_{ij}(y_i, y_j) - \boldsymbol{\delta}_{t-1}^{\top}\boldsymbol{y}$.
6: $\boldsymbol{\delta}_t \leftarrow \boldsymbol{\delta}_{t-1} + c_t \cdot \boldsymbol{g}(\boldsymbol{x}_t, \boldsymbol{y}_t)$ ($\boldsymbol{g}$ is a subgradient of L with respect to $\boldsymbol{\delta}$)
7: **end while**

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

Figure 1 shows a typical example of solutions by AM with different α and sequences of solutions (until convergence) by AM-Simple and AM-Greedy, where we note that $\boldsymbol{x} = \boldsymbol{y}$ holds only for solutions by AM with $\alpha > 0.044$ and also for solutions at convergence by AM-Simple and AM-Greedy. For AM, the α values are chosen by hand, and for the other procedures, the sequence α values are determined automatically (and increase at each iteration).[3] We note that for small α, the final $\mathcal{E}_\alpha$ can get smaller since the two sides are less constrained to be equal, and the submodular and supermodular terms can be relatively independently optimized. The figure shows that the objective value for AM with $\boldsymbol{x} = \boldsymbol{y}$ (which is 31.4325) is worse than the ones for the other variants. In particular, AM-Greedy achieves an objective value, with $\boldsymbol{x} = \boldsymbol{y}$, of 30.58.

3.2 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 [20], or Lagrangian relaxation, which is quite naturally applied to problem (1). We note that [20] considers this approach for Markov random field submodular sub-problems with pairwise potentials (where graph cuts can provide a solution), but the approach is just as applicable for an arbitrary submodular function f. We reformulate the original problem (1) in the following (equivalent) form:

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

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

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

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*

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

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 [10, 20, 28]. 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 or block coordinate descent. In Algorithm 4, we show pseudo-code of DD with a subgradient method.

Dual decomposition is also useful since it provides both a lower bound $L(\boldsymbol{\delta})$ to the optimal objective value, and also an upper bound (we use $\min[E(\boldsymbol{x}, \boldsymbol{x}), E(\boldsymbol{y}, \boldsymbol{y})]$). In both cases, for our results below, the bounds are computed right after line 5 of Algorithm 4

3.3 Continuous Relaxation (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 [30]. Therefore, many algorithms based on the LP relaxation have been actively discussed in

[3] This is an example with data generated as in the manner described in Section 5.1.

this context. In this subsection, we consider a convex relaxation approach to problem (1). Consider the following relaxation of problem (1) to the domain $[0,1]^{\mathcal{V}}$:

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

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) [8, 5] 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:

$$\mathbf{x}^{k+1} \Leftarrow \operatorname*{argmin}_{\mathbf{x}\in[0,1]^{\mathcal{V}}} L_\rho(\mathbf{x},\boldsymbol{\mu}^k,\boldsymbol{\delta}^k) \tag{7a}$$
$$\boldsymbol{\mu}^{k+1} \Leftarrow \operatorname*{argmin}_{\boldsymbol{\mu}\in[0,1]^{2\mathcal{V}}} L_\rho(\mathbf{x}^{k+1},\boldsymbol{\mu},\boldsymbol{\delta}^k) \tag{7b}$$
$$\begin{aligned}\boldsymbol{\delta}_1^{k+1} &\Leftarrow \boldsymbol{\delta}_1^k + \rho(A\boldsymbol{\mu}^{k+1} - \mathbf{1}_{|\mathcal{E}|}),\\ \boldsymbol{\delta}_2^{k+1} &\Leftarrow \boldsymbol{\delta}_2^{k+1} + \rho(B\boldsymbol{\mu}^{k+1} - C\boldsymbol{x}^{k+1})\end{aligned} \tag{7c}$$

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, 2, 23] (for details, see Appendix A below). In general, this problem might not be tractable in practice for large problems. However, if the submodular function has a specific structure and can be solved efficiently, such as a generalized graph-cut function, often found in practice, this could lead to a scalable algorithm for solving the problem [17, 23]. Also we note that since it is known that sparsity or even tree-structures can be used to accelerate the calculation of the quadratic problem, the structure of our problem (1) can accelerate the computation of Eq. (7b).

Algorithm 5 Continuous relaxation with rounding (CR) for problem (1).

Input: $\boldsymbol{\mu}_0$, $\boldsymbol{\delta}_{1,0}$, $\boldsymbol{\delta}_{2,0}$ and $\rho > 0$. **Output:** $\boldsymbol{x}^*$.
1: Set $t \leftarrow 1$.
2: **while** not converged **do**
3: $\mathbf{x}_{t+1} \leftarrow \operatorname{argmin}_{\mathbf{x}\in[0,1]^{\mathcal{V}}} L_\rho(\mathbf{x},\boldsymbol{\mu}_t,\boldsymbol{\delta}_t)$ (by solving MNP, see Appendix A).
4: $\boldsymbol{\mu}_{t+1} \leftarrow \operatorname{argmin}_{\boldsymbol{\mu}\in[0,1]^{4|\mathcal{E}|}} L_\rho(\mathbf{x}_{t+1},\boldsymbol{\mu},\boldsymbol{\delta}_t)$ (by solving the convex minimization).
5: $\boldsymbol{\delta}_{1,t+1} \leftarrow \boldsymbol{\delta}_{1,t} + \rho(A\boldsymbol{\mu}_{t+1} - \mathbf{1}_{|\mathcal{E}|})$, $\boldsymbol{\delta}_{2,t+1} \Leftarrow \boldsymbol{\delta}_{2,t} + \rho(B\boldsymbol{\mu}_{t+1} - C\boldsymbol{x}_{t+1})$.
6: **end while**
7: Round $\mathbf{x}_{t+1}$ to a integral solution $\boldsymbol{x}^*$.

A solution for problem (6) is not necessarily integral, and hence we must apply rounding. Although there are several possible deterministic and randomized rounding algorithms [25] that could be applied directly to our case, we use the node-based rounding of [25] in the experiments below. Algorithm 5 shows the pseudo-code of our continuous relaxation algorithm.

3.4 Submodular-Supermodular Procedure Using Pairwise Structures (SSP)

The Submodular-Supermodular procedure (SSP) [24, 13] is a set of heuristics for minimizing a general set functions. [13] 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 particular, they use the modular upper and lower bounds of a submodular function, in a Majorization-Minimization framework [15, 14, 13, 12]. We consider two variants here, which exploit the specific structure.

In one variant of 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 [11] 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 $\mathbf{b}^1_{ij}, \mathbf{b}^2_{ij} \in \mathbb{R}^2$, respectively, as

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

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, $h_{\bar{\boldsymbol{x}}_{ij}}(\boldsymbol{x}_{ij}) := \psi_{ij}(0,0)(1 - \boldsymbol{x}_{ij}^\top \mathbf{1}_2) + \boldsymbol{x}_{ij}^\top \tilde{\mathbf{b}}_{ij} = \psi_{ij}(0,0) + \boldsymbol{x}_{ij}^\top (\tilde{\mathbf{b}}_{ij} - \psi_{ij}(0,0)\mathbf{1}_2)$ 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.

Another variant of SSP is the Supermodular-Submodular procedure, where one replaces the submodular function f by its modular upper bound [11], and then minimizes the supermodular function [13]. Here, we replace f by its upper bound m^f, and optimize $m^f(\mathcal{S}(\boldsymbol{x})) + \sum_{(i,j)\in\mathcal{E}} \psi_{ij}(\boldsymbol{x}_{ij})$. In this case, the resulting term is expressed as a tree, which can be minimized exactly via dynamic programming. Moreover, under certain assumptions on f and ψ, these admit approximation guarantees.

Lemma 5. *Assuming that f is monotone submodular, and the tree function ψ is non-negative, the supermodular-submodular procedure achieves an approximation factor of $|\mathcal{S}(\mathbf{x}^*)| \leq n$.*

We can improve this factor, based on the curvature of f [14, 12]. The proof is given in Appendix A. The submodular-supermodular procedure also attains an approximation factor if we slightly change the formulation. Assume we are given a submodular tree function ψ, and a supermodular function f. Instead of the minimization problem, consider the problem of maximizing $f(X) + \psi(X)$. Notice that both problems are equivalent from an optimization perspective. The submodular-supermodular procedure then achieves a $1/2$ approximation under certain assumptions.

Lemma 6. *The submodular-supermodular procedure achieves a $1/2$ approximation, as long as the submodular function ψ is monotone submodular, and the function f is non-negative.*

4 Arbitrary Pairwise Supermodularity

The optimization methods described above can be applied to the case of arbitrary pairwise supermodular terms, directly or with minor changes, except for AM. That is, the applicability of CR and SSP do not change for a general graph $\mathcal{G} = (\mathcal{E}, \mathcal{V})$, in place of a tree $\mathcal{T}$, for defining the pairwise term in Eq. (1). And, DD can be applied similarity simply by solving three inner subproblems.

Here, we describe a particular instance of DD applicable to arbitrary pairwise terms. Let $\mathcal{G}_i = (\mathcal{V}_i, \mathcal{E}_i)$ $(i \in \{1,2\})$ be two acyclic subgraphs of the master graph $\mathcal{G}$, with $\mathcal{V}_1 = \mathcal{V}_2 = \mathcal{V}$, $\mathcal{E}_1 \cup \mathcal{E}_2 = \mathcal{E}$ and $\mathcal{E}_1 \cap \mathcal{E}_2 = \emptyset$ (for example, if $\mathcal{G}$ is a grid graph, then $\mathcal{E}_1$ may contain all horizontal edges of $\mathcal{G}$ and $\mathcal{E}_2$ all vertical ones). Then, the objective $E(\boldsymbol{x})$ becomes the sum of the submodular part and the pairwise terms corresponding to these subgraphs $E(\boldsymbol{x}) = f(\mathcal{S}(\boldsymbol{x})) + \sum_{(i,j)\in\mathcal{E}_1} \psi_{ij}(x_i, x_j) + \sum_{(i,j)\in\mathcal{E}_2} \psi_{ij}(x_i, x_j)$. Then, by preparing variables for each term as well as Eq. (4), we have again an equivalent problem for the minimization:

$$\min_{\boldsymbol{x},\boldsymbol{y},\boldsymbol{z}\in\{0,1\}^\nu} f(\mathcal{S}(\boldsymbol{x})) + \sum_{(i,j)\in\mathcal{E}_1} \psi_{ij}(y_i, y_j) + \sum_{(i,j)\in\mathcal{E}_2} \psi_{ij}(z_i, z_j)$$
$$\text{s.t. } x_i = y_i, \ x_i = z_i.$$

The minimization can be solved efficiently for each subproblem. Hence, we solve the dual given by

$$\max_{\boldsymbol{\delta}_1, \boldsymbol{\delta}_2} L(\boldsymbol{\delta}_1, \boldsymbol{\delta}_2) = \max_{\boldsymbol{\delta}_1, \boldsymbol{\delta}_2} E(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}) + \boldsymbol{\delta}_1^\top (\boldsymbol{x} - \boldsymbol{y}) + \boldsymbol{\delta}_2^\top (\boldsymbol{x} - \boldsymbol{z})$$

where $E(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})$ is the objective of the primal problem and $\boldsymbol{\delta}_1, \boldsymbol{\delta}_2$ are the Lagrangian coefficients. Similar to the tree case, $L(\boldsymbol{\delta}_1, \boldsymbol{\delta}_2)$ is concave on $(\boldsymbol{\delta}_1, \boldsymbol{\delta}_2)$, we can apply the subgradient method or the coordinate descent method to solve the dual.

5 Experimental Evaluation

In Subsection 5.1, we investigate and compare the performance of the four proposed approaches on synthetic data. Then, we apply them to image segmentation in Subsection 5.2. The experiments below were run on a 2.6 GHz 64-bit computer 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. [18].

5.1 Evaluation on Synthetic Data

Our first experiment was performed with synthetic data generated as follows. First for a submodular function, we used a modular plus 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 [21],

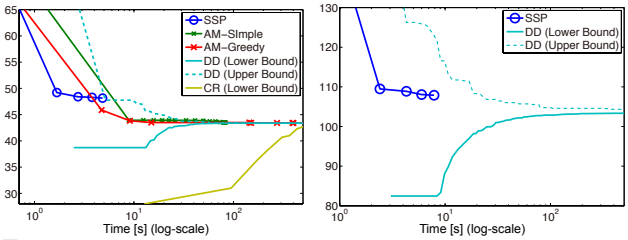

Figure 2: Typical examples (for a concave-of-modular function of $|\mathcal{V}| = 324$) of solution sequences by the algorithms for the cases with a tree-structured supermodular term (left) and arbitrary pairwise ones (right).

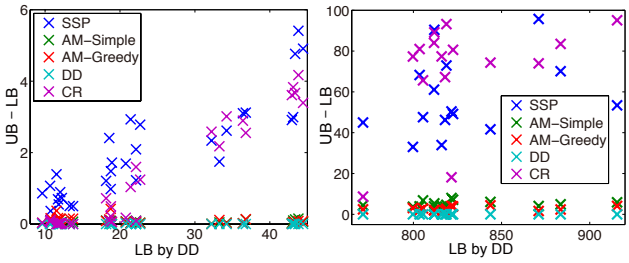

Figure 3: Lower bounds (by DD) vs. ϵ-optimality, *i.e.* (upper bounds by the algorithms)− (LB (by DD)), for several instances (left: concave-of-modular funct., right: objective in [21] with the tree-structured term).

$f(\mathcal{S}) = \sum_{i\in\mathcal{V}} \sum_{j\in\mathcal{S}} s_{ij} - \lambda \sum_{i,j\in\mathcal{S}} s_{ij}$, 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[4] 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 2 shows typical examples of algorithmic convergence for a concave-over-modular function with a supermodular tree-structured or arbitrary pairwise term, where $|\mathcal{V}| = 324$. Figure 3 shows the plots of lower bounds (by DD) 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).

5.2 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 [18] (with binary labels), where the unary, pairwise and higher-order

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

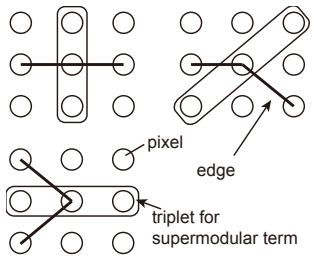

Figure 4: Construction of triplets for the supermodular term.

terms are respectively given by

$$\phi_i(x_i) = \theta_T \phi_T(x_i) + \theta_{col}\phi_{col}(x_i) + \theta_l \phi_l(x_i) \ (i \in \mathcal{V}),$$

$$\phi_{ij}(x_i, x_j) = \begin{cases} 0 & \text{if } \ x_i = x_j, \\ \theta_p + \theta_v \exp(\theta_\beta \|I_i - I_j\|^2) & \text{otherwise} \end{cases},$$

$$\phi_c(\boldsymbol{x}_c) = \begin{cases} 0 & \text{if } \ x_i = l_c, \ \forall i \in c, \\ |c|^{\theta_\alpha}(\theta_p^h + \theta_v^h \exp(-(\theta_\beta^h/|c|)\|\sum_{i\in c}(f(i)-\mu)^2\|) & \text{otherwise} \end{cases},$$

where ϕ_T, ϕ_{col} and ϕ_l are potentials from TextonBoost [26],[5] 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 [26] (1/2 images were used for training) and applied similar parameters to the ones used in [18]:

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

Segments $\mathcal{C}$ for the higher-order potential were generated from mean-shift [6]. 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 Figures 4 and 5 (upper right). 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 5 (bottom left) shows a typical example 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 (bottom right). For the optimization in our formulation, we used AM-Greedy (Algo. 3) 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.

[5]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.

Figure 5: Original image (upper left), supermodular forest (upper right), segmentation result with the robust P^n model (bottom left) and segmentation result with our formulation (bottom right).

6 Conclusions

We have developed algorithms for minimizing the sum of a submodular function and a tree-structured supermodular term based on four different approaches. We further describe extensions of the framework to the case with arbitrary pairwise terms. We investigated the performances of the proposed algorithms with synthetic data and applied the formulation to image segmentation, where information on image edges detected beforehand is incorporated through the supermodular term into the energy, showing improved results.

A Derivation of the MNP Problem for Eq. (7a)

We first offer some preliminaries necessary for this section. For a submodular function f with $f(\emptyset) = 0$, the *submodular polyhedron* $P(f) \subseteq \mathbb{R}^{\mathcal{V}}$ and the *base polyhedron* $B(f) \subseteq \mathbb{R}^{\mathcal{V}}$ are respectively defined as

$$P(f) = \{\mathbf{z} \in \mathbb{R}^{\mathcal{V}} : \mathbf{z}(\mathcal{S}) \leq f(\mathcal{S})(\forall \mathcal{S} \subseteq \mathcal{V})\} \text{ and}$$
$$B(f) = \{\mathbf{z} \in P(f) : \mathbf{z}(\mathcal{V}) = f(\mathcal{V})\},$$

where $\mathbf{z}(\mathcal{S}) = \sum_{v \in \mathcal{S}} z_v$ (for $\mathcal{S} \subseteq \mathcal{V}$). For any vector $\mathbf{c} \in \mathbb{R}^{\mathcal{V}}$, we denote by $f^c(\mathcal{S})$ the *reduction* of f by $\mathbf{c}$, *i.e.*, $f^{\mathbf{c}}$ is defined as $f^{\mathbf{c}}(\mathcal{S}) = \min_{\mathcal{T} \subseteq \mathcal{S}} f(\mathcal{T}) + \mathbf{c}(\mathcal{S} \setminus \mathcal{T})$ $(\mathcal{S} \subseteq \mathcal{V})$. Note that $P(f^{\mathbf{c}}) = \{\mathbf{z} \in P(f) : \mathbf{z} \leq \mathbf{c}\}$ (cf. Figure 6 (left)). Also for any vector $\mathbf{c} \in \mathbb{R}^{\mathcal{V}}$, the *translation* of a submodular function f by $\mathbf{c}$, *i.e.*, $(f + \mathbf{c})(\mathcal{S}) = f(\mathcal{S}) + \mathbf{c}(\mathcal{S})$, is submodular (cf. Figure 6 (right)).

For a vector $\mathbf{v} \in \mathbb{R}^{\mathcal{V}}$, $\mathbf{v}_{[a,b]}$ means a vector whose i-th element is defined as

$$v_{[a,b],i} = \begin{cases} a & \text{for } i \text{ s.t. } v_i \leq a, \\ v_i & \text{for } i \text{ s.t. } a \leq v_i \leq a, \\ b & \text{for } i \text{ s.t. } v_i \geq b. \end{cases}$$

In this appendix, we describe the detail of the derivation of the minimum-norm-point (MNP) problem to calculate the proximal operator for solving Eq. (7a). As described in Section 3.3, Eq. (7a) involves the minimization of the sum of the Lovász extension of a submodular function and a quadratic term. Since this is a non-smooth convex problem, it can be solved with the proximal gradient method, such as FISTA (the fast iterative shrinkage-thresholding algorithm) [3]. Here, the calculation of the proximal operator in our case is reduced to the following problem:

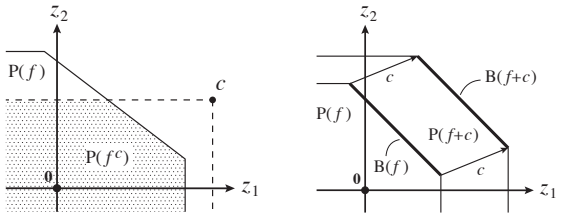

Figure 6: Illustrations of the submodular/base polyhedra of the reduction of submodular function f by a vector $\mathbf{c}$ (left) and the translation of f by a vector $\mathbf{c}$ (right).

$$\min_{\mathbf{x} \in [0,1]^{\mathcal{V}}} \hat{f}(\mathbf{x}) + \lambda \|\mathbf{x} - \mathbf{a}\|_2^2. \tag{10}$$

where $\mathbf{a} \in \mathbb{R}^{\mathcal{V}}$. Note that we have the constraint on the range of $\mathbf{x}$. Although we might be able to solve problem (10) by a general non-smooth convex solver, it can take much time due to the structure of the Lovász extension. However, we can find a solution to Eq. (10) by solving the problem without the constraint $[0,1]^{\mathcal{V}}$:

$$\min_{\tilde{\mathbf{x}} \in \mathbb{R}^{\mathcal{V}}} \hat{f}(\tilde{\mathbf{x}}) + \lambda \|\tilde{\mathbf{x}} - \mathbf{a}\|_2^2 = \min_{\tilde{\mathbf{x}} \in \mathbb{R}^{\mathcal{V}}} \max_{\mathbf{t} \in B(f)} \tilde{\mathbf{x}}^\top \mathbf{t} + \lambda \|\tilde{\mathbf{x}} - \mathbf{a}\|_2^2$$
$$= \max_{\mathbf{t} \in B(f)} -\sum_{i \in \mathcal{V}} t_i^2/(4\lambda) + t_i a_i, \tag{11}$$

where note that the last problem is equivalent to the so-called minimum-norm-point (MNP) problem (see, for the details of the derivation, [2]).

Theorem 7. *Let* $(\tilde{\mathbf{x}}^*, \mathbf{t}^*)$ *be an optimal pair to problem* (11)*. Then,* $\mathbf{x}^* := \tilde{\mathbf{x}}^*_{[0,1]}$ *is an optimal solution to problem* (10)*.*

Thus, we can solve Eq. (7a) by iteratively (in the proximal gradient method) solving the MNP problem (w/o the constraint) (11) and thresholding the solution.

Acknowledgments: We thank the UW-MELODI for discussions. This material is based upon work supported by the National Science Foundation under Grant No. IIS-1162606, and by a Google, a Microsoft, and an Intel research award. Rishabh Iyer also acknowledges support from the Microsoft Research Ph.D Fellowship Award.

References

[1] F. Bach. Structured sparsity-inducing norms through submodular functions. In *Advances in Neural Infor-*

mation Processing Systems, volume 23, pages 118–126. 2010.

[2] F. Bach. Learning with submodular functions: A convex optimization perspective. *Foundations and Trends in Machine Learning*, 6(2–3):145–373, 2013.

[3] A. Beck and M. Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM Journal on Imaging Sciences*, 2(1):183–202, 2009.

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

[5] 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.

[6] D. Comaniciu and P. Meer. Mean shift: A robust approach toward feature space analysis. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 24(5):603–619, 2002.

[7] 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.

[8] 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.

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

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

[11] R. Iyer and J. Bilmes. The submodular Bregman and Lovász-Bregman divergences with applications. In *NIPS*, 2012.

[12] R. Iyer and J. Bilmes. Submodular Optimization with Submodular Cover and Submodular Knapsack Constraints. In *NIPS*, 2013.

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

[14] R. Iyer, S. Jegelka, and J. Bilmes. Curvature and optimal algorithms for learning and minimizing submodular functions. In *NIPS*, 2013.

[15] R. Iyer, S. Jegelka, and J. Bilmes. Fast semidifferential-based submodular function optimization. In *ICML*, pages 855–863, 2013.

[16] S. Jegelka and J.A. Bilmes. Submodularity beyond submodular energies: coupling edges in graph cuts. In *Computer Vision and Pattern Recognition (CVPR)*, Colorado Springs, CO, June 2011.

[17] S. Jegelka, H. Liu, and J.A. Bilmes. On fast approximate submodular minimization. In *Advances in Neural Information Processing Systems*, volume 24, pages 460–468. 2011.

[18] 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.

[19] 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.

[20] N. Komodakis, N. Paragios, and G. Tziritas. MRF energy minimization and beyond via dual decomposition. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 33(3):531–552, 2011.

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

[22] L. Lovász. Submodular functions and convexity. In A. Bachem, M. Grötschel, and B. Korte, editors, *Mathematical Programming – The State of the Art*, pages 235–257. 1983.

[23] K. Nagano and Y. Kawahara. Structured convex optimization under submodular constraints. In *Proc. of the 29th Ann. Conf. on Uncertainty in Artificial Intelligence (UAI'13)*, pages 459–468, 2013.

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

[25] 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.

[26] 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.

[27] 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.

[28] D. Sontag, A. Globerson, and T. Jaakkola. Introduction to dual decomposition for inference. In S. Sra, S. Nowozin, and S. J. Wright, editors, *Optimization for Machine Learning*. MIT Press, 2011.

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

[30] 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.

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