Inference in Higher Order MRF-MAP Problems with Small and Large Cliques

Ishant Shanu[1] Chetan Arora[1] S.N. Maheshwari[2]
[1]IIIT Delhi [2]IIT Delhi

Abstract

Higher Order MRF-MAP formulation has been a popular technique for solving many problems in computer vision. Inference in a general MRF-MAP problem is NP Hard, but can be performed in polynomial time for the special case when potential functions are submodular. Two popular combinatorial approaches for solving such formulations are flow based and polyhedral approaches. Flow based approaches work well with small cliques and in that mode can handle problems with millions of variables. Polyhedral approaches can handle large cliques but in small numbers. We show in this paper that the variables in these seemingly disparate techniques can be mapped to each other. This allows us to combine the two styles in a joint framework exploiting the strength of both of them. Using the proposed joint framework, we are able to perform tractable inference in MRF-MAP problems with millions of variables and a mix of small and large cliques, a formulation which can not be solved by either of the two styles individually. We show applicability of this hybrid framework on object segmentation problem as an example of a situation where quality of results is significantly better than systems which are based only on the use of small or large cliques.

1. Introduction

Many problems in computer vision can be posed as labeling problems [33, 16]. Modeling them as MRF and finding the labeling configuration with maximum a posteriori probability converts them to the following discrete optimization problem:

$$\mathbf{l}_{\mathcal{V}}{}^{*} = \arg\min_{\mathbf{l}_{\mathcal{V}}} \sum_{p\in\mathcal{V}} f_p(l_p) + \sum_{\mathbf{c}\in\mathcal{C}} f_{\mathbf{c}}(\mathbf{l}_{\mathbf{c}}). \qquad (1)$$

Here, a pixel is denoted by p and $\mathcal{V}$ is the set of all pixels. A clique is a set of pixels conditionally dependent upon each other and is denoted by $\mathbf{c}$, and $\mathbf{l}_{\mathbf{c}}$ is the set of labels assigned to the pixels in clique $\mathbf{c}$. $\mathcal{C}$ denotes the set of all cliques. $f_p(l_p)$, called unary energy, assigns the cost for labeling a particular pixel p as l_p, whereas $f_{\mathbf{c}}(\mathbf{l}_{\mathbf{c}})$, called clique potential, is the cost of assigning labeling configuration $\mathbf{l}_{\mathbf{c}}$ to clique $\mathbf{c}$. The inference problem, popularly known as MRF-MAP, is to find the labeling, $\mathbf{l}_{\mathcal{V}}{}^{*}$ which minimizes the right hand side of Eq. (1). Many computer vision problems like image restoration, segmentation of videos and images, super resolution, texture synthesis, stereo matching to object detection, can be modelled as MRF-MAP inference problems [33, 16].

MRF-MAP is computationally a hard problem even in the restricted setting of binary labels and pairwise cliques. However, its applicability to modeling computer vision problems is so well established that variety of approaches have been explored to solve the problem in the restricted setting mentioned above. These approaches have ranged from message passing [24], dual decomposition [26], semidefinite programming relaxation [34], and graph cut [1, 25].

Since higher order clique potentials are more expressive compared to pairwise and lead to improved visual quality in the inferred output, the above mentioned techniques have been generalized to handle such problems also. The inference problem, however, becomes significantly more complicated. Apart from generalization of message passing and dual decomposition [23, 31], reducing the higher order problem to an equivalent pseudo Boolean form and solving it approximately using the QPBO algorithm is the other notable approach to handle such generalizations [8, 11, 15].

It should be noted that all the methods mentioned above provide only approximate solutions. Therefore, developing efficient inference techniques for some special subclass of clique potentials of interest has become an area of active research [18, 19, 20, 29]. One such special class of clique potentials whose importance to solving MRF MAP inference problems in computer vision is increasingly getting recognized is based on the notion of submodularity (formally defined in the next section).

Algorithms that minimize submodular functions (SFM) in polynomial time have now been known for over two decades [9, 30]. These algorithms work directly with the submodular polyhedron (defined formally in the next section), but have been found to be unusable as their run time is bounded by very high degree polynomials. There have been attempts to adapt/improvise various SFM techniques

for MRF-MAP inference problem [14, 17, 32]. A significant breakthrough is Generic Cuts (GC) [2, 3], a flow based algorithm that solves the inference problem for sum of submodular potentials in low order polynomial time for fixed clique sizes. GC has been shown to run efficiently even when the number of cliques runs into hundreds of thousands but does not scale well on problems with cliques of size larger than 16.

The problem of minimizing sum of submodular functions is beginning to attract attention of researchers [22, 31]. Kolmogorov [22] has reported a polynomial time "submodular flow" [6] based algorithm using capacity scaling. No results exist, however, to assess it's viability on practical vision problems. Recently [31] has reported a submodular function minimization algorithm, SOSMNP (Sum of Submodular Min Norm Point), based on Minimum Norm Point (MNP) algorithm [9]. SOSMNP works on realistic vision problems (millions of variables) with few but very large clique sizes.

This paper arises out of a realization that a typical vision problem involving millions of variables may require a model in which there are a very large number of small cliques (of sizes in the range 2 to 16) and a relatively few but large cliques (sizes of the order of 1000). One typical example could be when smaller cliques enforce regularization while larger cliques penalize deviation from the region/patch level statistics (e.g. a prior on the expected object size). Such problems are not solvable practically either by flow based algorithms like [2] or submodular polyhedron based algorithm like [31]. The first cannot handle large cliques and the second is applicable only when the number of cliques is small. What is required is a framework which simultaneously runs GC, a flow based algorithm on the small cliques, and SOSMNP, a submodular polyhedron based min norm style of algorithm on the large cliques. We show that this is indeed possible. The specific contributions of this paper are as follows:

1. **Mapping:** We show that variables in the, seemingly disparate, flow based and submodular polyhedron based approaches can be mapped to one other.

2. **Fusion:** The mapping allows us to propose a hybrid framework where we can adopt different styles to solve different sub-problems within the overall inference problem.

3. **Efficiency:** Using the proposed framework inference problems involving large number of variables with a mix of small and large cliques can be solved efficiently.

4. **Bridge:** The proposed framework is general and can act as a model for combining other algorithms chosen for applicability to other vision problems.

2. Background

Central to this paper is the notion of submodularity. A set function, $f : S \subseteq \mathcal{V} \rightarrow \mathbb{R}$ is called submodular if:

$$f(A) + f(B) \geq f(A \cap B) + f(A \cup B), \tag{2}$$

where $A, B \subseteq \mathcal{V}$. Convex polyhedron based SFM algorithms associate a base polyhedra, $B(f)$, with each submodular function f:

$$B(f) = \{x \mid x(S) \leq f(S), \forall S \subseteq \mathcal{V}; x(\mathcal{V}) = f(\mathcal{V})\},$$

where $x \in \mathbb{R}^{|\mathcal{V}|}$ and $x(U) = \sum_{u \in U} x(u)$. Any point $x \in B(f)$ is called a *base vector* or simply a *base*. For any base vector x, we denote $x^-(U) = \sum_{u \in U} \min\{0, x(u)\}$. It is easy to see that, $x^-(\mathcal{V}) \leq x(Y) \leq f(Y)$ holds for any $x \in B(f)$ and any subset $Y \subseteq \mathcal{V}$. An important result in SFM theory is the *Min Max Theorem* [12] which states that for a submodular function $f : S \subseteq \mathcal{V} \rightarrow \mathbb{R}$ with $f(\emptyset) = 0$:

$$\max\{x^-(\mathcal{V}) \mid x \in B(f)\} = \min\{f(S) \mid S \subseteq \mathcal{V}\}. \tag{3}$$

Strongly polynomial convex polyhedron based SFM algorithms actually solve the dual problem of maximizing $x^-(\mathcal{V})$ [13, 28, 30]. The Min Norm Point algorithm of Fuzishige et al. [9] on the other hand reduces the problem of finding a base vector, x, with maximum $x^-(\mathcal{V})$ to finding a base vector x^* with minimum norm: i.e., $x^* = \min_{x \in B(f)} \|x\|^2$. While no polynomial bound is known for the algorithm, experimental studies have shown that it runs much faster than the other available algorithms [14, 27, 31].

2.1. MRF-MAP and SFM Relationship

Eq. (1), with label set $\{a, b\}$ and the clique potential $f_{\mathbf{c}}$, can be looked upon as a set function over the set $\mathbf{c}$, such that $f_{\mathbf{c}}(\mathbf{l_c})$ defines the cost of a subset of pixels in $\mathbf{c}$ given a particular label (we use label 'a' as a convention here) in the labeling $\mathbf{l_c}$. Further, it is possible to consider $\sum_p f_p(l_p)$ in Eq. (1) as a 'modular' function[1] defined over a clique spanning all pixels of the image. With this, the MRF-MAP problem of Eq. (1) can be equivalently written as:

$$\mathbf{l_\mathcal{V}}^* = \arg\min_{\mathbf{l_\mathcal{V}}} \sum_{\mathbf{c} \in \mathcal{C}} f_{\mathbf{c}}(\mathbf{l_c}). \tag{4}$$

The above MRF-MAP formulation can be looked upon as minimizing a sum of submodular functions. Shanu et al. [31] have suggested a block coordinate descent framework to implement the Min Norm Point algorithm when cliques are large.

[1] Modular functions are special type of submodular function when Eq. (2) is satisfied with equality. Unlike submodular functions, modular functions can be minimized trivially by finding the minimum of each element independently.

2.2. Sum of Submodular Functions and Block Coordinate Descent

With each $f_{\mathbf{c}}$, as given in Eq. (4), one can associate a base polyhedron such that:

$$B(f_{\mathbf{c}}) := \Big\{ y_{\mathbf{c}} \in R^{|\mathbf{c}|} \mid y_{\mathbf{c}}(U) \leq f_{\mathbf{c}}(U),\ \forall U \subseteq \mathbf{c}; \quad (5)$$

$$y_{\mathbf{c}}(\mathbf{c}) = f_{\mathbf{c}}(\mathbf{c}) \Big\}. \quad (6)$$

Shanu et al. [31] have given the following results to relate a base vector, x, of the function f and a set of base vectors $y_{\mathbf{c}}$ of a $f_{\mathbf{c}}$:

Lemma 2.1. *Let $x(S) = \sum_{\mathbf{c}} y_{\mathbf{c}}(\mathbf{c} \cap S)$ where each $y_{\mathbf{c}}$ belongs to base polyhedra $B(f_{\mathbf{c}})$. Then the vector x belongs to base polyhedron $B(f)$.*

Lemma 2.2. *Let x be a vector belonging to the base polyhedron $B(f)$. Then, x can be expressed as the sum: $x(S) = \sum_{\mathbf{c}} y_{\mathbf{c}}(S \cap \mathbf{c})$, where each $y_{\mathbf{c}}$ belongs to the submodular polyhedron $B(f_{\mathbf{c}})$ i.e., $y_{\mathbf{c}} \in B(f_{\mathbf{c}})\ \forall\ \mathbf{c}$.*

The above results were used to minimize $\|x\|^2$ using a block coordinate descent approach, where each block represents a base vector $y_{\mathbf{c}}$ as defined above (c.f. [31]). Note that a base vector $y_{\mathbf{c}}$ is of dimension $|\mathbf{c}|$ (clique size), whereas a base x is of dimension $|\mathcal{V}|$ (number of pixels in an image). Since $|\mathbf{c}| \ll |\mathcal{V}|$, minimizing the norm of $y_{\mathbf{c}}$ over it's submodular polyhedron $B(f_{\mathbf{c}})$ is much more efficient than minimizing the norm of x by just applying the MNP algorithm. It should be noted that the above scheme is equally applicable with any other polyhedral based algorithm to minimize sum of submodular functions [9, 13, 28, 30].

2.3. Generic Cuts (GC)

An important result from [2] states that an arbitrary submodular function, f, defined over a set $\mathcal{V}$, can always be factorized into a sum of modular function, f_m, and a submodular function, f', such that $f'(\emptyset) = f'(\mathcal{V}) = 0$ and $f'(S) \geq 0, \forall S \subset \mathcal{V}$. In GC a given function (Eq. (1)) is reparameterized to such a form, $f = f_m + f'_{\mathbf{c}}$. GC then formulates the reparameterized problem as an integer program, relaxes it to a linear program, and maximizes its Lagrangian dual given below:

$$\max \sum_{p \in \mathcal{V}} U_p, \quad \text{subject to}$$

$$U_p \leq f_m(l) + \sum_{\mathbf{c}: p \in \mathbf{c}} V_{\mathbf{c},p,l} \quad \forall p \in \mathcal{V}, \quad l \in \{a, b\}, \quad \text{and}$$

$$\sum_{p \in \mathbf{c}} V_{c,p,l^p_{\mathbf{c}}} \leq f'_{\mathbf{c}}(\mathbf{l}_{\mathbf{c}}), \quad \mathbf{c} \in \mathcal{C}, \quad \mathbf{l}_{\mathbf{c}} \in \{a, b\}^{|\mathbf{c}|}. \quad (7)$$

For all submodular functions, without loss of generality, all $V_{\mathbf{c},p,b}$ can be set to 0, thereby reducing Eq. (7) to:

$$\sum_{p: l^p_{\mathbf{c}} = a} V_{c,p,a} \leq f'_{\mathbf{c}}(\mathbf{l}_{\mathbf{c}}), \quad \mathbf{c} \in \mathcal{C}, \quad \mathbf{l}_{\mathbf{c}} \in \{a, b\}^{|\mathbf{c}|}. \quad (8)$$

The algorithm then creates a gadget based flow graph (a gadget for each clique) with a node corresponding to each pixel and two special nodes s and t for source and sink respectively. The capacities of the edges from s and t to pixels are set using modular function f_m, whereas $f'_{\mathbf{c}}(\mathbf{l}_{\mathbf{c}})$ are used to restrict sum of flows in the edges (measured by the value of variables $V_{c,p,a}$) of a gadget [2].

3. Proposed Framework

From hereon, we will use a slightly modified notation for flow variables in GC. We will denote the set of variables $V_{\mathbf{c},p,a}$ as a vector $v_{\mathbf{c}}$, with each variable $V_{\mathbf{c},p,a}$ denoted by $v_{\mathbf{c}}(p)$ (a being implicit in the notation, since all the variables corresponding to b always remain 0 in GC). Recall that a base vector is denoted by $y_{\mathbf{c}}$ and each element in the base vector is denoted as $y_{\mathbf{c}}(p)$.

Further, we will assume that each submodular function $f_{\mathbf{c}}$ is of the form such that $f_{\mathbf{c}}(\emptyset) = f_{\mathbf{c}}(\mathbf{c}) = 0$ and $f(S) \geq 0, \forall S \subset \mathbf{c}$. As shown in [2], the assumption is not restrictive as every submodular function can be reparameterized to such a form.

The following results establish the correspondence between $v_{\mathbf{c}}$ and $y_{\mathbf{c}}$.

Lemma 3.1. *Any vector $v_{\mathbf{c}}$ derived from a valid flow in the GC flow graph is a vector in the base polyhedron of $f_{\mathbf{c}}$.*

Proof. Equations (5) and (6) give the conditions for any vector to be in the base polyhedron. By virtue of Eq. (8), it is easy to see that the flow vector satisfies Eq. (5). Further, by the property of flow conservation in the GC flow graph $\sum_{p \in \mathbf{c}} v_{\mathbf{c}}(p) = 0 = f_{\mathbf{c}}(\mathbf{c})$ (c.f. [2]), thus satisfying Eq. (6) as well. □

Lemma 3.1 serves to indicate that equations (5) and (8) are essentially the same, and a flow vector also satisfies the conditions of a base vector. This allows us to directly map a flow vector $v_{\mathbf{c}}$ to $y_{\mathbf{c}}$. But, the converse is not true, since, a general base vector may not satisfy the flow conservation constraint at each vertex in the flow graph. In flow minus out flow, i.e. excess at the vertex, may be negative or positive. However, this is easy to fix as follows. If the excess is negative then an edge directed from the source s is to the vertex with negative excess is added with flow equal to the absolute value of the excess. If the excess is positive then an edge from the positive excess vertex to the sink t is added with flow equal to the excess. We have, in effect, shown the following:

Lemma 3.2. *For every base vector $y_{\mathbf{c}}$ there exists a GC flow graph with valid flow obtained through one to one mapping of $y_{\mathbf{c}}$ to the flow vector $v_{\mathbf{c}}$ in the gadget corresponding to clique $\mathbf{c}$ in the GC flow graph.*

4. Hybrid Algorithms

Lemmas 3.1 and 3.2 enable us to put in place a block coordinate descent strategy consistent with the framework of [31]. We make a coordinate block corresponding to sum of large clique potentials to be solved by a polyhedral method maximizing $x^-(\mathcal{V})$. Another coordinate block is made (corresponding to sum of smaller cliques) to be solved by GC by maximizing flow. At each iteration we choose a coordinate block and solve it's optimization problem using the chosen method. Consistent with the coordinate blocks, Eq. (4) can be rewritten as

$$\mathbf{l}_{\mathcal{V}}{}^* = \arg\min_{\mathbf{l}_{\mathcal{V}}} \sum_{\mathbf{c}\in\mathcal{L}} f_{\mathbf{c}}(\mathbf{l}_{\mathbf{c}}) + \sum_{\mathbf{c}\in\mathcal{S}} f_{\mathbf{c}}(\mathbf{l}_{\mathbf{c}}). \tag{9}$$

Here $\mathcal{L}$ and $\mathcal{S}$ are the set of large and small cliques respectively such that $\mathcal{C} = \mathcal{L} \cup \mathcal{S}$. If x is a base vector corresponding to the submodular function as given in Eq. (9), then using Lemma 2.2 it can also be written as:

$$x = \sum_{\mathbf{c}\in\mathcal{L}} y_{\mathbf{c}}(\mathbf{l}_{\mathbf{c}}) + \sum_{\mathbf{c}\in\mathcal{S}} y_{\mathbf{c}}(\mathbf{l}_{\mathbf{c}}), \text{ or} \tag{10}$$

$$x = x_l + x_s, \tag{11}$$

where $y_{\mathbf{c}}(\mathbf{l}_{\mathbf{c}}), \mathbf{c} \in \mathcal{L}$ and $y_{\mathbf{c}}(\mathbf{l}_{\mathbf{c}}), \mathbf{c} \in \mathcal{S}$ are the base vectors corresponding to large and small cliques respectively. Vectors, x_l and x_s denote the factors of x due to large and small cliques respectively.

A hybrid algorithm achieves the objective of maximizing $x^-(\mathcal{V})$ by alternately carrying out a block coordinate descent iterations on the blocks corresponding to $\mathcal{L}$ and $\mathcal{S}$. While maximizing $x_l^-(\mathcal{V})$ we keep x_s (and its associated y's) as constant. Any base polyhedron based SFM algorithm which works on the principle of maximizing $x^-(\mathcal{V})$ can be used for the maximizing step. Note that since there will be multiple number of large cliques it is quite likely that the sum of submodular function algorithm used may itself be based on block coordinate descent scheme.

We use GC to maximize $x_s^-(\mathcal{V})$ and when we do that, we keep x_l constant, in effect treating them as unary costs in GC flow graph. As explained in Section 2.3 the role of unary costs is in setting the terminal edge capacities in the GC graph. The flow graph is created as per the construction given in Section 3.

It may be noted that during multiple such block coordinate ascent iterations GC edge capacities remain constant and the minimum cut to be found changes due to changes in the terminal edge capacities. In principle, it is possible to exploit such structure by initializing the flow in an iteration from the flow in the last iteration, fixing the overflows and then send more flow if possible. This kind of structure have been exploited by Kohli and Torr [21] using graph cuts. Our current implementation, however, does not exploit this.

5. Hybrid Algorithm Based On MNP and GC

It must be pointed out that the above framework is not applicable as written if the base polyhedron based algorithm is the Min Norm Point Algorithm, the algorithm of choice for working with large cliques [27, 31]. We give below the reasoning in detail and the changes required to use MNP to handle large cliques.

5.1. Problems in Combining GC with MNP

Before we delve into the problem in combing MNP and GC, it is important to understand the differences between polyhedral methods which maximize $x^-(\mathcal{V})$ and MNP which minimizes $\|x\|^2$.

Polyhedral methods that maximize $x^-(\mathcal{V})$ can be considered to move from one base vector to another in which the value of $x^-(\mathcal{V})$ increases. The extent of increase is a function of the next base vector that the algorithm chooses. For example, in Schrijver's algorithm [30] this increase is effected by carrying out what is known as "block exchange". Given a base vector x and two elements say p and q in $\mathcal{V}$, in a block exchange, a δ is computed to generate a new base vector x', which differs from x only in that the value of $x'(p)$ is larger than $x(p)$ by δ and the value of $x'(q)$ is smaller than $x(q)$ by δ. In many polyhedral based methods the value of $x^-(\mathcal{V})$ is maximized through appropriately chosen block exchanges.

Let S be a set that minimizes the given submodular function. It can be shown that S can also be obtained by including all the elements with negative value in a base vector x that maximizes $x^-(\mathcal{V})$ along with some other elements of x, with zero value, chosen based upon some "reachability" criterion that is particular to specific SFM algorithms.

Note that if p and q are in S a block exchange may be possible which increases the value of $x(p)$ and decreases the value of $x(q)$ without making $x(p)$ positive. In this case neither the value of $x^-(\mathcal{V})$ changes nor the optimal set S changes, but new optimal base vector has been computed.

By picking two elements, which either both belong to S or it's complement, one can, therefore, generate infinitely many optimal base vectors that leave the optimal set S unchanged. It can be easily shown that when such block exchanges also leave the relative order of the values of elements unchanged the L_2 norm value of the resultant base vector reduces. The discussion not only indicates that polyhedral methods using block exchanges as the basic operation to maximize $x^-(\mathcal{V})$ may not minimize L_2 norm, but also motivates a strategy to do so using GC, as we show later in the next section.

Note that, apart from the edge emanating from the source and the edge incident at the sink, the edges in a flow augmentation path in GC consists of a sequence of pairs of edges each pair belonging to a gadget through which the path passes. During flow augmentation flow increases in one edge of such a pair and decreases by the same amount in the other resulting in change in the flow vetor value at just two vertices in the gadget involved. Lemma 3.2 implies that this results in change in the value of the corresponding base vector at just two vertices. In effect flow augmentation in GC, at the macro level, involves performing of a series of block exchange operations (corresponding to the gadgets through which the augmenting path passes), and results in maximized $x^-(\mathcal{V})$ at termination. The source of problem faced in combining GC and SOSMNP in a common block descent framework is, the additional requirement, that output of GC should also minimize the L_2 norm. When GC flow augmentation stops, the set of pixels gets partitioned into a (S,T) cut set: namely those which are reachable from the source s form the set S, and the remaining nodes form the set T. It can be easily shown that while the total flow in the edges across the cutset is the value of the maximum flow and also the value of the optimal solution to the sum of submodular function minimization problem, the values of the elements in S in the corresponding final base vector are all negative or zero and their sum is equal to the max flow in the GC flow graph.

5.2. Outputting a Min L_2 Norm Solution in GC

As shown in the previous section, GC can be interpreted as a block exchange algorithm. Therefore, the reachability property from nodes s and t to verticices in sets S and T in the max flow (S,T) cutset in the GC flowgraph are the same reachability properties as those by the attributes in the sets S and T outputted by any other block exchange based algorithm, say Schrijver's. Therefore, the counterpart of block exchange among nodes in set S in Schrijver's algorithm that results in lowering of the L_2 norm in GC is nothing but the result of sending flow from s to some node p in S. The following lemma can be shown to hold.

Lemma 5.1. *The base vector x corresponding to max flow flow in GC with (S,T) cut set does not satisfy the L_2 norm as long as there exist augmenting paths*

1. *from s to any vertex p in S such that the value of $x(q)$, q being the first vertex on that path after s, is less than $x(p)$, and*
2. *to t from any vertex p in T such that the value of $x(q)$, q being the first vertex on that path after t, is larger than $x(p)$.*

We propose the following scheme for moving towards minimum L_2 norm solution in GC. Let (S,T) be the min cut in the flow graph outputted by GC. We will explain below the process for the set S. The process for set T will be identical and consistent with condition 2 of Lemma 5.1 and effectively be just the mirror image of process explained below.

Let x denote the base vector corresponding to a GC block coordinate descent iteration. Let $x^-(S)$ denote the sum of the negative elements (corresponding to pixels) in set S. It is easy to see that the ideal redistribution of values in vector x such that the $x^-(S)$ remains same, but the norm $\|x(S)\|^2$ achieves minimum is when all non zero elements in set S have the value $x^-(S)/|S|$.

To move towards the desired objective, we create a new flow graph containing all the nodes in S and two auxiliary nodes s and t. We create an edge from source s to a vertex, p, whenever $x(p) < x^-(S)/|S|$. The capacity of the edge is set to $\frac{x^-(S)}{|S|} - x(p)$. From the rest of the negative valued vertices in S we direct edges to sink t. Capacity of an edge from such a node q to t is set equal to $x(q) - \frac{x^+(S)}{|S|}$. All the other edges between two pixel nodes of the set S are the edges of the original GC graph with their capacities set equal to the residual capacities they had when GC's previous iteration terminated.

We now solve the max flow problem in the flow subgraph so created. If all the edges out of s are saturated then we have effectively shown that there exists an optimal solution to the original problem in which all the elements on the S side have the same value, which the optimal min norm solution should have. Otherwise, we have discovered a new (S,T) cut in the subproblem the base vector corresponding to which can be moved towards the min L_2 norm recursively by solving two max flow problems (for the new S and T sets) as explained above on increasingly smaller flow graphs. Note that the this method of moving towards the min L_2 norm solution, consistent with Lemma 5.1, has the merit that the number of additional max flow problems that need to be solved are not only finite but are bounded by the number of nodes in the original GC graph.

5.3. Proposed Algorithm

We formalize the proposed hybrid strategy for minimizing the sum of submodular functions. As it is common in SFM literature, we assume that $f_{\mathbf{c}}(\emptyset) = 0$ for large clique potentials. For small clique potentials, as is done in GC, we assume that the function has been appropriately reparameterized such that $f_{\mathbf{c}}(\emptyset) = f_{\mathbf{c}}(\mathbf{c}) = 0$ and the value of clique potential for all other configurations is non-negative. No such reparameterization is done for large clique potential. Values of modular function, f_m, can be arbitrary (positive/negative/zero), but we reparameterize it in our algorithm such that $f_m(l_p = a) = f_m(l_p = a) - f_m(l_p = b)$ and $f_m(l_p = b) = 0$. This causes a decrease of constant, $\sum_p f_m(l_p = b)$, in all the function values, which we add

Procedure 1 HybridMNGC: Algorithm for minimizing a sum of submodular function with mixed sized cliques

Input: $f = \sum f_m + f_s + f_l$.

Output: $x^* = y_m + x_s + x_l = \arg\min \|x\|^2$ subject to $x \in B(f)$.

Set $y_m = f_m$ and $y_{\mathbf{c}} = 0, \mathbf{c} \in \mathcal{S}$.
$\forall \mathbf{c} \in \mathcal{L}$ initialize $y_{\mathbf{c}}$ from a random base vector.
while Norm of x decreases **do**
    Call SOSMNP with $y_m + x_s$ as unary cost and f_l as the clique potential;
    MinNormGC($\mathcal{V}, f_s, y_m + x_l, 0$);
end while

Procedure 2 MinNormGC($\mathcal{V}, f_s, U, b$)

Input: $\mathcal{V}$: Set of Nodes/Elements for which norm is to computed.

Input: U: Function specifying unary/modular cost for each pixel in $\mathcal{V}$.

Input: f_s: Clique potential for the small cliques.

Input: b: Base value above which we add edge from s.

We create the GC gadget graph using f_s and assume that the flow graph once created is available globally during the recursive calls. Only the edges from s and to t change.

for $\forall p \in \mathcal{V}$ **do**
    if $U(p) > b$ **then**
        Add s to p edge with capacity $U(p) - b$;
    else
        Add p to t edge with capacity $b - U(p)$;
    end if
end for
Run GC; Find minimum cut and corresponding S and T sets;
if $|S| > 1$ **then**
    Denote by U_S, the residual edge capacity between source s and pixels $p \in S$;
    Denote average of U_S by b_S;
    MinNormGC(S, f_s, U_S, b_S);
end if
if $|T| > 1$ **then**
    Denote by U_T, the residual edge capacity between pixels $p \in T$ and sink node t;
    Denote average of U_T by b_T;
    MinNormGC(T, f_s, U_T, b_T);
end if

back in the returned value of the minimum.

We solve the minimization problem in the dual domain by finding a vector with minimum norm in the base polyhedron of sum of $f_m + f_l + f_s$. We make use of the block coordinate descent framework of [31] and maintain a base vector, x, of the overall function, as the sum of base vectors y_m, x_s and x_l, corresponding to modular, small and large clique potential functions. We initialize the vector y_m as the value of function f_m and keep it constant throughout the algorithm. We initialize the $y_{\mathbf{c}}$ corresponding to small cliques to be zero, which is equivalent to initializing the GC flow graph with flow of zero. For large cliques, we start with an arbitrary base as the starting $y_{\mathbf{c}}$.

The various iterations of the algorithm follow the scheme as detailed out in Section 4. The above detailing of the initialization process takes into account modular function costs which were subsumed earlier. Algorithms 1 and 2 give the details of proposed algorithm in pseudocode format.

5.4. Convergence

Overall algorithm implements block coordinate descent over two blocks. One consists of all the large cliques, and the other contains all the small cliques. Each iteration minimizes the norm of the solution vector x corresponding to the block chosen. Convergence of the over all algorithm follows from the in principle convergence of Min Norm algorithm [10], polynomial time convergence of GC [2], and convergence of the block coordinate descent based min norm algorithm [14, 31]. The additional work that is happening is transformation of GC output in every run of GC block to satisfy L_2 norm. That can take as little as $O(\log n)$ GC iterations in the best case to $O(n)$ iterations in the worst case, where n is the number of pixels. Experiments reported in the next section indicate that HybridMNGC is significantly faster than SOSMNP which, experiments reported earlier [31] indicate, is the algorithm of choice for polyhedral based algorithms working with large cliques.

6. Experiments

We have conducted all the experiments on a regular workstation with Intel Core i7 CPU, 8 GB of RAM and running Windows 10 OS. We have implemented the proposed algorithm in C++. We have taken implementations of SOSMNP and GC from the authors' website and modified them for our purpose. All timings reported are in seconds.

The focus of the experiments is to establish the efficacy of the hybrid scheme both in terms of the efficiency and quality of the results outputted. The experiments focussing on efficiency have been on synthetic problems of small sizes. The results and discussion are in Section 6.1. Quality comparison has been done on the pixelwise object segmentation problem where the images for the demonstration are taken from Pascal VOC dataset [7]. Outputs of HybridMNGC are compared with that of SegNet [4] (chosen to represent state of the art methods based on deep neural networks). GC runs with cliques of size two, and SOSMNPs run with cliques of size approximately 1000.

In experiments which focus on efficiency, as in [31], we consider edge based clique potentials based on counting the

	Big Clique Size =100, No. of cliques =25				
Small Clique Size	2×1	2×2	3×2	3×3	4×3
HybridMNGC	11.54	9.42	9.68	14.35	58.48
SOSMNP	274.27	538.62	184.53	366.77	231.78

Table 1. Clique potentials are Count Based. Small cliques span the image in sliding window style. All running time are in seconds.

	Big Clique Size =100, No. of cliques =25				
Small Clique Size	2×1	2×2	3×2	3×3	4×3
HybridMNGC	17.41	21.27	21.51	28.56	93.96
SOSMNP	250.67	350.17	375.78	268.42	228.54

Table 2. Similar comparison as in Table 1 with the proviso that clique potentials are edge based. All running time are in seconds.

	No. of Pixels 2500				
#No. of Big Cliques	40	60	80	100	120
HybridMNGC	31.28	34.98	62.75	66.11	94.66
SOSMNP	295.73	367.97	396.08	403.61	428.46

Table 3. Count based big cliques with size=100, pairwise small cliques span whole image. All running time are in seconds.

	No. of Big Cliques = 50			
#No. of Pixels	2500	3600	4900	6400
HybridMNGC	34.69	30.02	31.95	32.50
SOSMNP	234.17	271.79	301.67	349.71

Table 4. Time as a function of Image size. Count based big cliques with size=100, pairwise small cliques span whole image. All running time are in seconds.

square root of the number of edges, and count based potential which involve counting the product of number of ones and zeros in the clique. The unary potentials are assigned randomly. In experiments on real images from Pascal VOC dataset we have used count based potentials and unary costs are based on the score received from SegNet for the image under consideration.

6.1. Performance Comparison

Tables 1 and 2 compare the performance of the two algorithms on images of size (50×50) as small clique size is changed. The unary potentials, seeding of the large cliques, as well as the number of small cliques is the same. Note that HybridMNGC takes significantly less time than SOSMNP at all points of comparison. Observe that run time for SOSMNP does not show any monotonic pattern. Since at each iteration both algorithms work with optimal

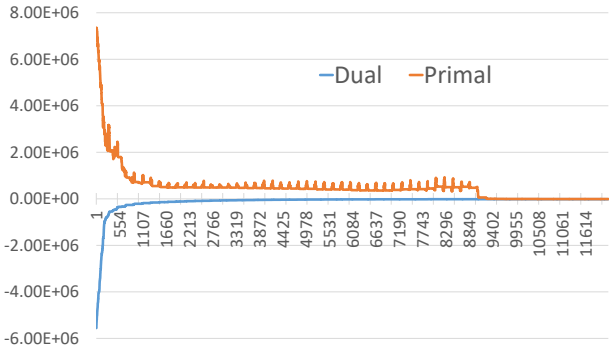

Figure 1. Energy as a function of the number of iterations of the HybridMNGC algorithm. Primal is the function value for the set defined by the negative elements of the current base vector.

subsolutions this fluctuation is possibly due to different ways in which values get transmitted among large number of small cliques under SOSMNP. In comparison HybridMNGC, which is an order of magnitude faster than SOSMNP, exhibits typical GC trends where performance decreases with increase in clique size. Similar trend is observed with both count as well as edge based potentials.

Tables 3 and 4 give the results of changes in time taken when both the number of big cliques and the image size is increased. The time for both the algorithms increase, but HybridMNGC continues to outperform the baseline SOSMNP.

6.2. Comparing Object Segmentation Quality

We show some indicative results in Figures 2 and 3 for pixel wise object segmentation. Each image has a single object consistent with our binary labeling formulation. In the images in Figure 3 Gaussian noise with zero mean and intensity dependent variance has been added. We use Segnet [4], output in two ways. First as the basis for comparison of deep network based object segmentation, and second as a source for generating singleton confidence scores for independent pixel labels. The other points of comparison of the output of HybridMNGC are GC as an example of an algorithm for MRF MAP optimization using small clique models, and SOSMNP for an algorithm which can handle large cliques. The pairwise cliques span the image in 4-connected neighborhood style. Big cliques are region grown as in [32]. The number of big cliques and their size varies from image to image. Suffice to say that the number of big cliques range from 300 to 500. The average big clique size is 1000 with the max being 1500. The images themselves are 250×250 pixels in size.

Since both the images and the big clique sizes are too large for HybridMNGC and SOSMNP to terminate with optimal solutions in acceptable times, these algorithms are run in ϵ-approximate mode where ϵ is defined as in [5]. We run

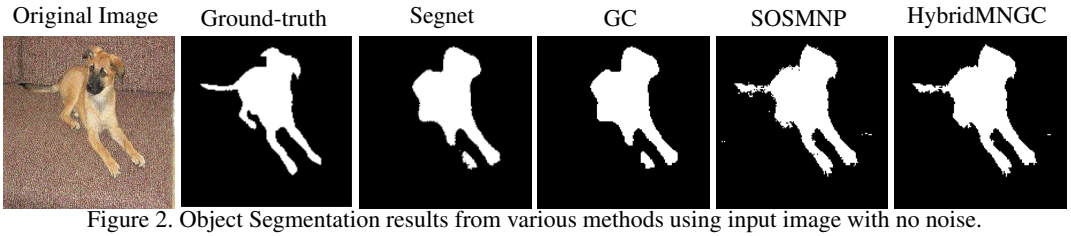

Figure 2. Object Segmentation results from various methods using input image with no noise.

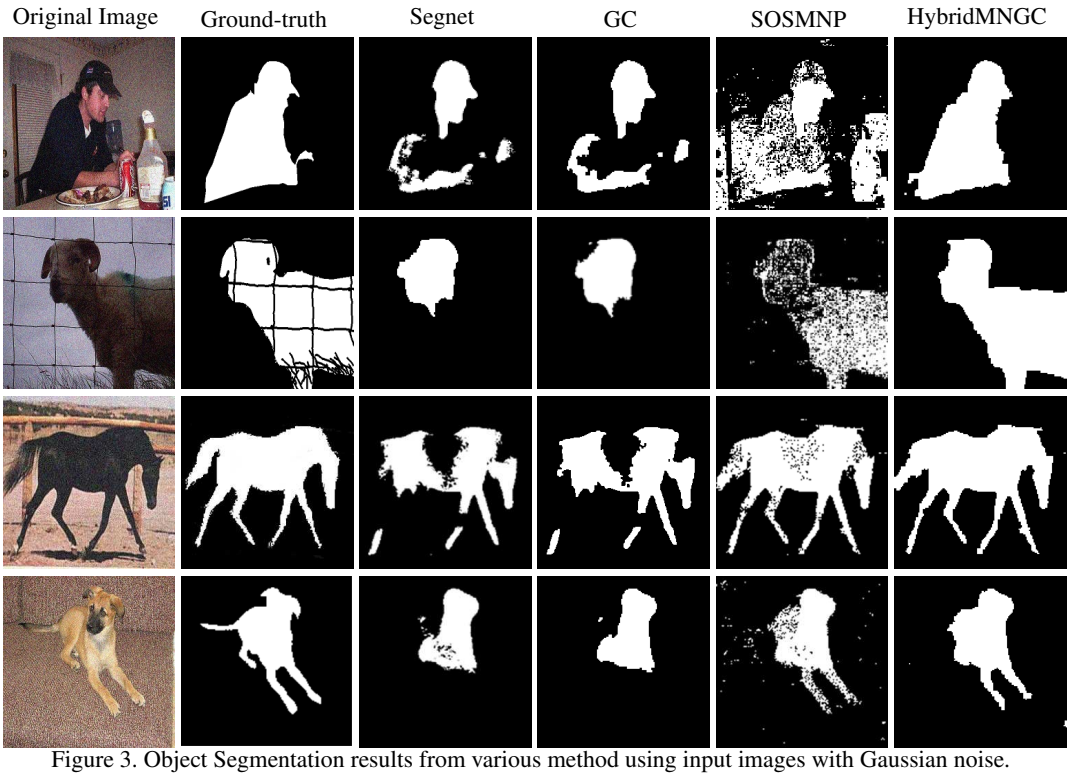

Figure 3. Object Segmentation results from various method using input images with Gaussian noise.

these algorithms with ϵ as 100. This condition seems to be arrived, on the average, after about 12000 iterations. Experimental evidence indicates Figure 1 that the primal labeling stabilizes much before the termination condition is reached. Figure 3 has the comparative output images. Note the significant improvement in HybridMNGC output in comparison to that of GC and SOSMNP.

Figure 2 contains an example of object segmentation with no added noise. Note that Segnet output deteriorates greatly with added noise. Output of HybridMNGC seems to be much more resilient to noise.

The object segmentation experiments have been done with the objective of establishing the usefulness of working simultaneously with both higher order and small cliques. What should their sizes be, how should they be seeded in the image, how should the potentials be chosen for an object segmentation system based on the ideas outlined here is a subject for future research.

7. Conclusions

Our objective in this paper has been to establish the efficacy of combining algorithms like Generic Cuts [2] and SOSMNP [31] which have complimentary strengths and weaknesses. The resultant algorithm, HybridMNGC, not only is shown to be more efficient, but has the potential of providing very high quality solutions to computer vision problems. Not only does it open up possibilities of developing new algorithms around HybridMNGC, it shows the way in which seemingly different techniques can be combined under the block coordinate descent framework. Solving problems in parallel by suitably defining blocks is an obvious possibility and future research direction for us.

Acknowledgement: Chetan Arora has been supported by Infosys Center for AI, Visvesaraya Young Faculty Research Fellowship, and EMR Funding by DST-SERB, Government of India. Ishant Shanu has been supported by Visvesaraya PhD Fellowship and a travel grant by Google.

References

[1] C. Arora, S. Banerjee, P. Kalra, and S. Maheshwari. An efficient graph cut algorithm for computer vision problems. In *ECCV*, pages 552–565. Springer, 2010. 1

[2] C. Arora, S. Banerjee, P. Kalra, and S. Maheshwari. Generalized flows for optimal inference in higher order MRF-MAP. *TPAMI*, 37(7):1323–1335, 2015. 2, 3, 6, 8

[3] C. Arora and S. Maheshwari. Multi label generic cuts: Optimal inference in multi label multi clique MRF-MAP problems. In *CVPR*, pages 1346–1353, 2014. 2

[4] V. Badrinarayanan, A. Kendall, and R. Cipolla. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *arXiv preprint arXiv:1511.00561*, 2015. 6, 7

[5] D. Chakrabarty, P. Jain, and P. Kothari. Provable submodular minimization using wolfe's algorithm. In *NIPS*, pages 802–809, 2014. 7

[6] J. Edmonds and R. Giles. A min-max relation for submodular functions on graphs. *Annals of Discrete Mathematics*, 1:185–204, 1977. 2

[7] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results. http://www.pascal-network.org/challenges/VOC/voc2012/workshop/index.html. 6

[8] A. Fix, A. Gruber, E. Boros, and R. Zabih. A graph cut algorithm for higher-order markov random fields. In *ICCV*, pages 1020–1027, 2011. 1

[9] S. Fujishige, T. Hayashi, and S. Isotani. *The minimum-norm-point algorithm applied to submodular function minimization and linear programming*. 2006. 1, 2, 3

[10] S. Fujishige, T. Hayashi, and S. Isotani. *The minimum-norm-point algorithm applied to submodular function minimization and linear programming*. 2006. 6

[11] H. Ishikawa. Higher-order clique reduction without auxiliary variables. In *CVPR*, pages 1362–1369, 2014. 1

[12] S. Iwata. Submodular function minimization. *Mathematical Programming*, 112(1):45, 2008. 2

[13] S. Iwata, L. Fleischer, and S. Fujishige. A combinatorial strongly polynomial algorithm for minimizing submodular functions. *JACM*, 48(4):761–777, 2001. 2, 3

[14] S. Jegelka, F. Bach, and S. Sra. Reflection methods for user-friendly submodular optimization. In *NIPS*, pages 1313–1321, 2013. 2, 6

[15] F. Kahl and P. Strandmark. Generalized roof duality for pseudo-boolean optimization. In *ICCV*, pages 255–262, 2011. 1

[16] J. H. Kappes, B. Andres, F. A. Hamprecht, C. Schnörr, S. Nowozin, D. Batra, S. Kim, B. X. Kausler, T. Kröger, J. Lellmann, N. Komodakis, B. Savchynskyy, and C. Rother. A comparative study of modern inference techniques for structured discrete energy minimization problems. *IJCV*, 115(2):155–184, 2015. 1

[17] D. Khandelwal, K. Bhatia, C. Arora, and P. Singla. Lazy generic cuts. *CVIU*, 143:80–91, 2016. 2

[18] P. Kohli. Graph cuts for minimizing robust higher order potentials. 1

[19] P. Kohli, M. P. Kumar, and P. H. Torr. P3 & beyond: Solving energies with higher order cliques. In *CVPR*, pages 1–8. IEEE, 2007. 1

[20] P. Kohli, P. H. Torr, et al. Robust higher order potentials for enforcing label consistency. *IJCV*, 82(3):302–324, 2009. 1

[21] P. Kohli and P. H. S. Torr. Dynamic graph cuts for efficient inference in markov random fields. *TPAMI*, 29(12):2079–2088, Dec. 2007. 4

[22] V. Kolmogorov. Minimizing a sum of submodular functions. *Discrete Applied Mathematics*, 160(15):2246–2258, 2012. 2

[23] V. Kolmogorov. A new look at reweighted message passing. *TPAMI*, 37(5):919–930, 2015. 1

[24] V. Kolmogorov and M. Wainwright. On the optimality of tree-reweighted max-product message-passing. *arXiv preprint arXiv:1207.1395*, 2012. 1

[25] V. Kolmogorov and R. Zabin. What energy functions can be minimized via graph cuts? *TPAMI*, 26(2):147–159, 2004. 1

[26] N. Komodakis, N. Paragios, and G. Tziritas. Mrf energy minimization and beyond via dual decomposition. *TPAMI*, 33(3):531–552, 2011. 1

[27] S. McCormick. Submodular function minimization. 2013. 2, 4

[28] J. B. Orlin. A faster strongly polynomial time algorithm for submodular function minimization. *Mathematical Programming*, 118(2):237–251, 2009. 2, 3

[29] C. Rother, P. Kohli, W. Feng, and J. Jia. Minimizing sparse higher order energy functions of discrete variables. In *CVPR*, pages 1382–1389, 2009. 1

[30] A. Schrijver. A combinatorial algorithm minimizing submodular functions in strongly polynomial time. *Journal of Combinatorial Theory, Series B*, 80(2):346–355, 2000. 1, 2, 3, 4

[31] I. Shanu, C. Arora, and P. Singla. Min norm point algorithm for higher order mrf-map inference. In *CVPR*, pages 5365–5374, 2016. 1, 2, 3, 4, 6, 8

[32] P. Stobbe and A. Krause. Efficient minimization of decomposable submodular functions. In *NIPS*, pages 2208–2216, 2010. 2, 7

[33] R. Szeliski, R. Zabih, D. Scharstein, O. Veksler, V. Kolmogorov, A. Agarwala, M. Tappen, and C. Rother. A comparative study of energy minimization methods for Markov random fields with smoothness-based priors. *TPAMI*, 30(6):1068–1080, June 2008. 1

[34] P. Wang, C. Shen, A. van den Hengel, and P. H. Torr. Efficient semidefinite branch-and-cut for MAP-MRF inference. *IJCV*, 117(3):269–289, 2016. 1