INDRAPRASTHA INSTITUTE *of*
INFORMATION TECHNOLOGY **DELHI**

Optimal inference algorithms for higher order MRF-MAP problems

Submitted in partial fulfillment of the requirements for the degree of
Doctor of Philosophy

ISHANT

Ishants@iiitd.ac.in

Indraprastha Institute of Information Technology

Supervisor:
Prof. Chetan Arora

August 13, 2020

Certificate

This is to certify that the thesis titled **Optimal inference algorithms for higher order MRF-MAP problems** being submitted by **Ishant** to the Indraprastha Institute of Information Technology-Delhi, for the award of the degree of **Doctor of Philosophy**, is an original research work carried out by him under my supervision. In my opinion, the thesis has reached the standards fulfilling the requirements of the regulations relating to the degree.

The results contained in this thesis have not been submitted in part or full to any other university or institute for the award of any degree/diploma.

March, 2020

Prof. Chetan Arora

Adjunct Faculty,

Department of Computer Science,

Indraprastha Institute of Information Technology Delhi,

Okhla Industrial Estate, Phase III, New Delhi, Delhi 110020.

Acknowledgement

First of all, I would like to extend my heartiest gratitude towards my supervisor, Prof. Chetan Arora for his constant support throughout my PhD. He consistently gave me time for detailed technical discussions and for my paper reviews despite his busy schedule. He has given me insight into both theoretical as well as practical work. I really enjoyed all the long technical discussions with him. I would like to thank him for his continuous technical/non-technical help and his feedback and encouragement all the times during my PhD. I would like to thank him for guidance and motivation, which enabled me to complete this thesis. I would also like to thank Prof. Maheshwari for his precious time and constant support. He introduced me to the world of abstract thinking which enabled me to see through things. Being with him I learned whole new way of picking what is important. He not only helped me to learn more but also what are the limitations of thought and too much of the unprocessed information. My PhD would not have been possible without the emotional support of my mother. At last but not the least I would like to thank my lab-mates for their support.

Ishant

Publications

1. **Ishant Shanu**, Chetan Arora, and Parag Singla, Min Norm Point algorithm for higher order MRF-MAP inference, in IEEE Conference on Computer Vision and Pattern Recognition(CVPR), 2016.

2. **Ishant Shanu**, Chetan Arora, and S.N. Maheshwari, Inference in higher order MRF-MAP problems with small and large cliques, in IEEE Conference on Computer Vision and Pattern Recognition(CVPR), 2018.

3. **Ishant Shanu**, Siddhant Bharti, Chetan Arora, and S.N. Maheshwari, An Inference Algorithm for Multi-Label MRF-MAP Problems with Clique Size 100, In European Conference on Computer Vision(ECCV), 2020.

Abstract

Many computer vision problems can be formulated as finding the best labeling configuration. If labelings satisfy Markov property then finding best labeling configuration becomes MRF (Markov Random Field)- MAP (Maximum A Priori Posteriori) inference problem. Which is the minimization of cost of assigning labels to individual pixels and cost of assigning labelings to a collection of pixels (cliques). If we assume clique costs (or clique potentials) to be submodular then MRF-MAP inference becomes minimization of sum of submodular functions and can be done in polynomial time. Standard way to minimize a submodular function is by minimizing an equivalent dual objective defined on submodular polyhendron. As the first part of thesis in chapter 3, we develop an efficient inference algorithm for 2 label MRF-MAP problem named SoS-MNP. We show that the dual problem can be decomposed over cliques which enables the efficient optimization of dual in block co-ordinate descent (BCD) style. In our experiments we look at the image segmentation problem with clique size as large as 1000. We show that SoSMNP is very efficient and scalable to large cliques as compared to state of the art methods which scales only upto clique size of 16. In the second part of thesis in chapter 4, we develop the inference algorithm for 2-label MRF-MAP problem with a mix of small and large cliques. In such a configuration there are large number of small cliques which makes BCD style SoSMNP to be very slow. On the other hand there are other state of the art methods like Generic Cuts (GC) which run very fast for the problems with large number of small cliques but do not scale for the problems with large cliques. To overcome the limitations of both the algorithms we run GC for small cliques and SoS-MNP for large cliques in BCD style by proposing a mapping between the variables of SoS-MNP and GC. Even after this mapping the hybrid algorithm does not give optimal result because GC minimizes ℓ_1-norm and SOS-MNP minimizes ℓ_2-norm of the objective function. We propose a recursive method which calls GC multiple times to output ℓ_2-norm solution. In experiments we show that the quality of the pixelwise image segmentation results improve if we use both the small and large cliques as compared to if we use only large cliques. We also demonstrate the efficiency of the proposed hybrid method over SoSMNP on the configuration with small and large cliques. As the third and last part of thesis in chapter 5, we develop an inferencne algorithm for multi label MRF-MAP problems. The current state of the art methods only run for the configuration of cliques with size upto 4 and 4 labels only. Standard way to solve multi-label problem is by converting it into 2-label problem by some encoding, such encoding introduces many extra states. We show that there is enough structure

in the submodular polyhendron of the encoded clique potentials which can be exploited. We propose an efficient hybrid method (Hybrid-ML) which avoids computation over the extra states. In our experiments we show that using an MRF with clique size 800 can improve the results obtained by state of the art deep learning network Segnet on pixel-wise multi-object segmentation results. We also run Hybrid-ML on a stereo correspondence problem with clique size 100 and 16 labels. We also compare the running time of Hybrid-ML with SoSMNP and show a huge improvement in terms of efficiency.

Table of Contents

*

List of Tables

List of Figures

Chapter 1

Introduction

In our daily life, digital images are everywhere; they come from the cameras in mobile phones, CCTV cameras that surround us, Youtube videos, as well as photos uploaded on social networks like Facebook, Instagram etc. Computer scientists are contributing to how digital images can be analysed, processed and understood [39, 91]. Computer vision is now an established branch of Artificial Intelligence dedicated to "inferring information" from an input image or a video. At present some of the sub categories of computer vision that are being explored are object classification task which for a given image involves outputting the probability of each object class from a predefined set being present in the input image [63]. Object detection problem involves predicting the bounding box for each object's location along with the confidence of each class being present in the corresponding box [79]. Object segmentation task for an input image requires to output the boundary of objects present in the input image [10]. Image super-resolution task for at least one input low resolution images requires to output an image at better resolution [102]. Image synthesis task for an input image requires to output a new image based on a given description of the new image [46]. Stereo correspondence problem for a pair of images requires to output the relative depth of the objects from a predefined set [25].

1.1 Labeling Problems

The class of the computer vision problems described above as well as many more can be posed as labeling problems. In such problems the input is mapped onto a set of random variables, where each random variable may correspond to a pixel, a super pixel, a frame or some feature. The value (or label) that a random variable can take is from a predefined set of labels. The problem can be posed as finding the most likely labeling from a probability distribution defined over the set of random variables. The probability distribution depends upon the desired output for the problem and on the given input. If we assume that the set of random variables satisfy the markov property [22, 96] then the labeling problem becomes Markov Random Fields (MRF)- Maximum A Posteriori (MAP)

problem (formally defined in the next section). Now we will look at some of the labeling assignment problems where each random variable corresponds to a pixel in the input image. For example, image denoising requires a noisy input image transformed into clean image. Consider the image is gray scale then in the labeling problem, a label corresponds to one of the 256 intensity values. Assume that we can calculate the noise present in the input image then the cost of assigning a labeling can be set proportional to the noise in the image [11]. Object segmentation task for an image can be seen as requiring the assignment of labels from predefined set to pixels where labels represent the class of the objects. The most likely labeling ensures that each object in the image gets the corresponding unique label [49]. In the stereo correspondence problem, for a pair of input images, a label assigned to any pixel specifies the relative depth of corresponding surface voxel [92]. Images corresponding to a real scene follows that intensity values in a close region does not change rapidly along a line sweep for example, such regions are shown in Figure 1.1 as the superpixels. It is very highly likely that such region is within an object or background therefore the labels of the pixels in it must be uniform (or dependent). The size of the regions define the range of a local neighbourhood for which labels of the pixels in it depend on each other. In general, we can say that the label of a pixel is contextually dependent on the the label of pixels in immediate neighbourhood [66].

One of the ways to look at the labeling problems is using graphs where each pixel (random variable) can be seen as a node in the graph. Let there be an edge between two nodes if the labels of their corresponding pixel depend on each other. Since in a graph, a subset of nodes in which every pair of nodes are connected by an edge is called a clique. Therefore, a local neighbourhood in which the label of every pair of pixels is contextually dependent can be referred to as *clique* and the number of pixels in a clique as the *clique size* [49]. Consistency among labels associated with pixels in a clique can be enforced by assigning cost for each possible labeling associated with pixels in a clique [49]. Costs are so assigned that more likely labeling configurations have relatively lower cost as compared to the unlikely labeling configurations.

With costs so associated, the labeling problems can be formulated as an optimization problems. Let us consider a simple example of foreground-background segmentation. In this problem, the two labels 0 and 1 are used to represent if a pixel belongs to foreground or background. Since in the output image a set of neighbouring pixels should have similar labels, we can associate a consistency cost with clusters of pixels which can be looked upon as a measure of similarity among the labels in the cluster (formally called a clique). Cost associated with a clique is a function of the labels associated with the pixels in the clique and is called *clique potentials*. Clique size (number of pixels) used in modeling is essentially a measure of resolution in the model and is at least 2 (*first order* cliques) or larger (*higher order* cliques). With each pixel we can also associate a cost measuring the penalty (pixel cost) of assigning the label 0 or 1. Segmenting

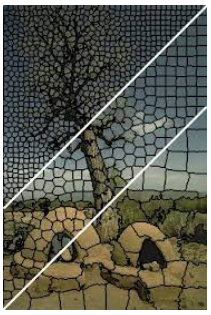

Figure 1.1: Figure shows 3 different sizes of regions from small to large differentiated by two lines as shown in [2]. In each region the intensity of pixels do not vary by a significant amount.

out the foreground object in the image can be modelled as determining a labeling of the pixels which minimizes some weighted combination of the total pixel and clique costs associated with the image. Formally, the set of pixels in the image is denoted by $\mathscr{P}$, the two labels that can be associated with any pixel by $\mathscr{L} = \{0, 1\}$. Each $p \in \mathscr{P}$ is associated with a label $l_p \in \mathscr{L}$. A clique is denoted as $\mathbf{c}$ which is a subset $\mathbf{c} \subseteq \mathscr{P}$, and $\mathscr{C}$ denotes the set of all the cliques in the image. The set of labels assigned to pixels in a clique $\mathbf{c}$ is denoted by $\mathbf{l_c}$ and is called the labeling configuration of the clique. For each pixel $p \in \mathscr{P}$ the cost of assigning the label l_p is denoted by $D_p(l_p)$ and is called the *unary potential*. The cost of assigning the labeling configuration $\mathbf{l_c}$ to a clique $\mathbf{c}$ is denoted by $W_{\mathbf{c}}(\mathbf{l_c})$ and is called the *clique potential*. The minimization problem defined earlier can formally be described as follows:

$$l^*_{\mathscr{P}} = \underset{l_{\mathscr{P}} \in \mathscr{L}^{|\mathscr{P}|}}{\arg\min} \sum_{p \in \mathscr{P}} D_p(l_p) + \sum_{\mathbf{c} \in \mathscr{C}} W_{\mathbf{c}}(\mathbf{l_c}), \tag{1.1}$$

where $l^*_{\mathscr{P}}$ denotes the labeling configuration over all the pixels which minimizes the r.h.s. of Eq. (1.1).

Let us consider multi object segmentation stated earlier which can be modelled as a label assignment problem. Let us assume there are m objects in the image. Label assignment, therefore, involves assigning each pixel $p \in \mathscr{P}$ a label from the set of $m+1$ labels $\mathscr{L} = \{0, 1, 2...m\}$ such that all pixels associated with an object get assigned the label corresponding to the object. note that the label 0 represents the background. Similar to image denoising example we can define the pixel cost of assigning a label to each pixel, and enforce label consistency within each clique by specifying the cost over each clique of possible labeling configurations. As before, optimization problem involves finding the labeling which minimizes some combination of the pixel costs and clique costs. Note that the only change is that the set $\mathscr{L}$ is now $\{0, 1, 2...m\}$. We now show that such an

optimization problem can be related to the well known Markov Random Fields (MRF)- Maximum A Posteriori (MAP) problem.

1.2 Markov Random Fields

Let us consider an input which we formally call an observation $o \in O$ from the set of all possible observations, denoted by O. We assume that o comprises of n features belonging to a set $\mathscr{P}$, and o_p denotes the value of feature $p \in \mathscr{P}$. We can associate a node (in the abuse of notation we denote it by p) with each feature and map o onto a graph. We assume each node can be assigned one label from the set of labels $\mathscr{L} = \{0, 1, ..., m\}$. We associate n discrete random variables with the set of nodes $l_p, \forall p \in \mathscr{P}$. Value that the random variable l_p can take can be interpreted as the label associated with node $p \in \mathscr{P}$. We denote the labeling configuration over all the nodes by $l = \{l_p | p \in \mathscr{P}\}$. Let $\mathscr{L}$ be the set of all possible labelings. Probabilistic inference assumes that there is some joint probability distribution $\mathbb{P}(\mathscr{L} \times O)$ over all possible observations and their associated labelings. We don't assume we have knowledge of this joint probability distribution but it does allow us to consider the conditional probability $\mathbb{P}(l|o)$ of an unknown labeling $l \in \mathscr{L}$ given that we observed $o \in O$. This conditional distribution is known as the posterior probability (since it is the conditional probability of the random vector l after having observed o). The Maximum A-Posteriori (MAP) problem is to find the state l^* with highest posterior probability

$$l^* = \arg\max_{l \in \mathscr{L}^n} \mathbb{P}_{l/o}(l/o).$$

Using the Bayes' rule we can rewrite the above expression as follows:

$$l^* = \arg\max_{l \in \mathscr{L}^n} \frac{\mathbb{P}(o/l)\mathbb{P}(l)}{\mathbb{P}(o)}. \tag{1.2}$$

Probability distribution $\mathbb{P}(o/l)$ is called the likelihood function. Many models assume that observation $o_p, \forall p \in \mathscr{P}$ depends only on l_p. Therefore probability distribution $\mathbb{P}(o/l)$ can be factorized as $\prod_{p \in \mathscr{P}} \mathbb{P}_p(o_p/l_p)$. The term $\mathbb{P}(l)$ is called prior probability distribution which defines the distribution of the joint labeling for all the nodes. The label of a node is contextually dependent upon the label of other nodes in the some neighbourhood/clique. Such cliques can been look upon as Markov Blankets (the Markov Blanket of a random variable is that subset of the random variables on which the random variable is dependent, i.e. conditioned upon nodes in the Markov Blanket it is independent of the random variables not in the Markov blanket) of the node under consideration. Therefore the joint probability $\mathbb{P}(l)$ can be factorized using Markovian property [65] as $\mathbb{P}(l) = \frac{1}{Z} \prod_{\mathbf{c} \in \mathscr{C}} \phi_{\mathbf{c}}(\mathbf{l_c})$ where $Z = \sum_{l \in L} \prod_{\mathbf{c} \in \mathscr{C}} \phi_{\mathbf{c}}(\mathbf{l_c})$. We can rewrite the Eq.

(1.2) as follows:

$$l^* = \arg\max_{l \in \mathscr{L}^n} \frac{\prod_{p \in \mathscr{P}} \mathbb{P}(o_p/l_p) \prod_{\mathbf{c} \in \mathscr{C}} \phi_{\mathbf{c}}(\mathbf{l_c})}{Z \times \mathbb{P}(o)}. \tag{1.3}$$

$\mathbb{P}(o)$ is constant with respect to l and therefore does not affect the arg max. The term Z is a constant with respect to l which can also be ignored. We can rewrite Eq. (1.3) as follows:

$$l^* = \arg\max_{l \in \mathscr{L}^n} \prod_{p \in \mathscr{P}} \mathbb{P}(o_p/l_p) \prod_{\mathbf{c} \in \mathscr{C}} \phi_{\mathbf{c}}(\mathbf{l_c}). \tag{1.4}$$

By taking the negative log likelihood of each term or putting $D_p(l_p) = -\log(\mathbb{P}(o_p/l_p))$ and $W_{\mathbf{c}}(\mathbf{l_c}) = -\log(\phi_{\mathbf{c}}(\mathbf{l_c}))$, we can rewrite equation (1.4) as:

$$l^* = \arg\min_{l \in \mathscr{L}^n} \sum_{p \in \mathscr{P}} D_p(l_p) + \sum_{\mathbf{c} \in \mathscr{C}} W_{\mathbf{c}}(\mathbf{l_c}). \tag{1.5}$$

Note that Eq. (1.5), an instance of traditional MRF-MAP problem, is identical to the optimization problem defined in Eq. (1.1). Labeling problems of interest to us are all instances of MRF-MAP inference. There is a vast literature on developing efficient algorithms for various sub categories of MRF-MAP optimization. We now survey the state of the art and define the agenda that we have focused on in this dissertation.

1.3 MRF-MAP Inference Methods

There are a total $|\mathscr{L}|^{|\mathscr{P}|}$ labeling configurations in general out of which finding the optimal configuration is NP hard even for the first order MRFs [18**?**]. Researchers have, therefore, focused on developing efficient approximate inference algorithms for the problem in general as well as with various restricted setting of MRF parameters. Restricting the clique size to first order and working with binary labels has been the most popular approach as it has been shown that in this situation when the potentials are submodular there exists an optimal inference algorithm that runs in low order polynomial time [17, 18, 38, 42, 83**?**]. In the following section, we discuss this celebrated algorithm known as Graphcut.

1.3.1 Graphcut

For submodular clique potentials, first order MRF-MAP energy function (given in Eq.(1.1)) is representable by flow graph [**?**]. Which has nodes corresponding to pixels (pixel nodes) and two more nodes source(S) and sink(T). In the graph, two pixel nodes are connected by edges if their respective pixels are in same clique, there are edges from S to each pixel node and from pixel nodes to T

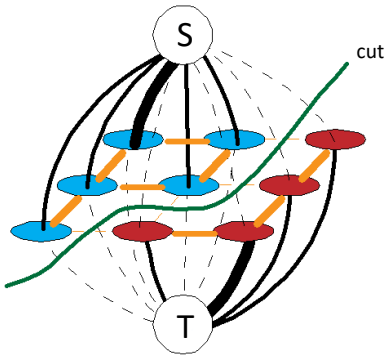

Figure 1.2: Graphcut calculates the max-flow on the graph shown above.

(shown in Figure 1.2). Capacity of the edges between two pixel nodes depend upon clique potentials and capacity of edges S to pixel nodes, pixel nodes to T depend upon unary potentials such that only one edge among the two has non-zero capacity. Graphcut optimizes the energy function by computing the max flow on this flow graph [**?**]. This technique has been applied for solving a variety of computer vision problems. Examples range from image regularisation and denoising [17, 18], image segmentation [16, 19, 82], estimation of optical flow [13, 18], to stereo vision [58, 101]. Graphcut techniques have also been extended to deal with non-submodular binary energy functions. Approximate inference for those non submodular clique potentials which can be represented as quadratic pseudo boolean functions is possible by a Graphcut like construction (QPBO) [15]. The approximation outputted by this algorithm satisfies weak persistence in that the algorithm assures that the pixels that get labeled would get the same label as in the optimal solution. The algorithm leaves a number of pixels unlabeled.

α-expansion, the technique of choice for working with multi-label problems over first order potentials, is also Graphcut based [18]. α-expansion involves cycling through all the labels and making an expansion move with the current label α. The expansion move consists of minimizing the MRF potential with respect to the current label (in effect solving a two label problem). Inference using α-expansion is only approximate even for submodular clique potentials.

Ishikawa [40] has reported a polynomial time optimal inference algorithm for m-label MRF with submodular clique potentials by transforming the m-label problem to a flow problem on a graph. Interestingly Ishikawa's transformation in which the graph created is over first order potentials has turned out to be non generalisable in that nothing has been reported since publication of his algorithm in 2003. Successful handling of multi-label problems over submodular potentials been reported only when combinatorial flow based techniques were developed to

handle higher order potentials [7]. We will have more to say about this approach a little later in our discussion when we talk about MLGC.

It is well known that higher order clique potentials can capture the contextual statistics between large number of pixels and lead to improved visual quality in the inferred output [41, 52, 59, 80, 100]. Since algorithms which could handle higher order potentials were not reported for a long time, Graphcut based algorithms were experimented with for handling higher order MRFs also. One such approach is based on reducing the higher order cliques to pairwise ones and applying the QPBO algorithm on the resultant model [3, 14]. However, this leads to a large number of cliques with non-submodular potential in the resulting pairwise model. Practical usefulness of such reductions has been reported to be limited to third-order potentials only [14].

1.3.2 Belief Propagation

Since Graphcut operates over a very limited class of MRFs, approximate inference algorithms for general MRF has attracted attention of researchers [56, 62, 95, 103]. Belief propagation (BP) [26, 98] is one such class of algorithms. BP maps the primal problem (given in Eq. (1.1)) onto a graph which has nodes corresponding to pixels (pixel nodes) and cliques (clique nodes). If a pixel belongs to a clique then in the graph the corresponding pixel node is connected to the clique node by an edge (as shown in Figure 1.3). Corresponding to each pixel node there is a vector (whose size is the number of labels) which contains the effective costs of a pixel being assigned a particular label. This vector is called a belief vector. In BP algorithms the clique nodes collect the beliefs of its neighbouring pixel nodes through messages. The processed beliefs are then forwarded to the neighbouring pixel nodes. This local exchange of messages takes place at every clique node until the the beliefs stabilize at all pixel nodes [64]. Formally at iteration t, the message from a pixel node p to the clique node $\mathbf{c}$ for a label l_p is denoted as $m^t_{p\to\mathbf{c}}(l_p)$. A message from the clique node $\mathbf{c}$ to its neighbouring pixel node p for a label l_p at current iteration t is denoted as $m^t_{\mathbf{c}\to p}(l_p)$. The messages are updated as per the rules given below:

$$m^t_{p\to\mathbf{c}}(l_p) = \sum_{d\in\mathcal{N}(p)\setminus\mathbf{c}} m^{t-1}_{d\to p}(l_p),$$

$$m^t_{\mathbf{c}\to p}(l_p) = \min_{l_{\mathbf{c}\setminus p}} \left(W_c(l_c) + \sum_{q\in\mathbf{c}\setminus p} m^{t-1}_{q\to\mathbf{c}}(l_p)\right).$$

At any iteration the belief of a pixel node p is computed as the sum of the messages coming from its neighbouring clique nodes as given below:

$$b^t_p(l_p) = \sum_{\mathbf{c}\in\mathcal{N}(p)} m^t_{\mathbf{c}\to p}(l_p).$$

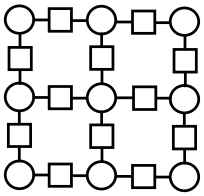

Figure 1.3: Factor graph corresponding to 3×3 image. Here we consider pairwise cliques where each pixel (represented as circle) is connected to their neighbours by a clique node (represented as square box).

This simplistic scheme given above to explain BP is not always guaranteed to converge. It has been shown to converge to optimal solution if the graph created is a tree [96]. There are sophisticated schemes reported in literature that guarantee convergence in general but they come at the cost of increased complexity and consequent inefficient running times [76].

1.3.3 Primal-Dual Framework

Another class of MRF inference algorithms are linear programming based primal–dual methods. The primal-dual frameworks have been at the heart of the successful MRF-MAP inference methods. Original MRF-MAP problem formulation 1.1 can be converted to an equivalent integer program (IP) as follows [61]. Any pixel $p \in \mathscr{P}$ can take a label from the set $\mathscr{L}$ of possible labels. The labeling configuration of clique $\mathbf{c}$ in which the label of pixel p is l in $\mathscr{L}$ is denoted by $l_{\mathbf{c},p,l}$. For each pixel there are $|\mathscr{L}|$ binary variables X_p^l. The value of X_p^l is 1 if pixel p is assigned label $l \in \mathscr{L}$ else its value is 0. Similarly for each possible labeling on a clique there is a binary variable $Y_c^{\mathbf{l_c}}$ which takes value 1 whenever clique $\mathbf{c}$ is assigned labeling configuration $\mathbf{l_c}$ and is 0 otherwise. The equivalent IP of the problem defined in Eq. (1.1) is as follows:

$$\min_{X_p^l, Y_c^{l_c}} \sum_{p \in \mathscr{P}} \sum_{l \in \mathscr{L}} D_p(l) X_p^l + \sum_{\mathbf{c} \in \mathscr{C}} \sum_{l_c \in \mathscr{L}^k} W_c(\mathbf{l_c}) Y_{\mathbf{c}}^{\mathbf{l_c}}, \tag{1.6}$$

subject to,

$$\sum_{l \in \mathscr{L}} X_p^l = 1, \qquad (p \in \mathscr{P})$$

$$\sum_{\forall l_{\mathbf{c},p,l}} Y_{\mathbf{c}}^{l_{\mathbf{c},p,l}} = X_p^l, \qquad (\mathbf{c} \in \mathscr{C}, p \in \mathbf{c}, l \in \mathscr{L})$$

$$X_p^l, Y_c^{l_c} \in \{0, 1\}. \tag{1.7}$$

Above IP can be converted to an LP by relaxing the constraints $X_p^a, Y_c^{l_c}$ as follows:

$$X_p^a, Y_c^{l_c} \geq 0. \tag{1.8}$$

Dual of the above LP formulation can shown to be the following [7]:

$$\max \sum_{p \in \mathscr{V}} U_p, \tag{1.9}$$

subject to

$$U_p = \min_{l \in \mathscr{L}} h_p(l) \tag{1.10}$$

$$h_p(l) = D_p(l) + \sum_{\mathbf{c}: p \in \mathbf{c}} V_{\mathbf{c},p,l}, \quad \forall l \in \mathscr{L}, \tag{1.11}$$

$$\sum_{p \in \mathbf{c}} V_{c,p,l_{\mathbf{c}}^p} \leq W_{\mathbf{c}}(\mathbf{l}_{\mathbf{c}}). \tag{1.12}$$

Komodakis and Tziritas [61, 62, 72] have used this framework to develop inference algorithms for multi-label first order MRFs. In their wells and balls model each pixel p has a well associated with it in which there are $|\mathscr{L}|$ balls (one for each label l) "floating". In a well corresponding to pixel p, the ball corresponding to label l floats at height $h_p(l)$ as shown in Figure 1.4. The free variables $V_{c,p,l}$ control the heights at which the balls float. The maximization of the value of objective function given in Eq. (1.9) can be looked upon as raising the lowest ball in all the wells subject to constraints given in Eq. (1.11) and (1.12) being satisfied.

Since this optimization problem is NP-Hard in general, their approximation scheme involves handling the objective function defined by Eq. (1.9) by reducing each iteration to a two ball problem, viz. the current active ball (for which $X_p^l > 0$) and the ball corresponding to the label chosen for the iteration. The iteration involves increasing the heights of the lowest ball in all the wells subject to feasibility of the dual constraints. Each iteration is modeled as a max flow problem very much in the spirit of Graphcut.

In the graph that is created for a flow iteration there is a node for each pixel and two distinguished nodes s and t the source and sink respectively. A flow pushing iteration involves choosing a well (say p) in which the ball corresponding to the chosen label is above the active ball (flow pushing starts from this well) and another (say q) in which the ball corresponding to the chosen label is below the active ball (flow pushing ends at this well). Pushing flow δ from p to q is supposed to decrease the height of chosen ball in well p by δ and increase the height of the chosen ball by δ in well q as shown in 1.5.

In line with this objective the edge from s is directed to node p with capacity equal to the difference in heights of the two balls in well p initially (active ball

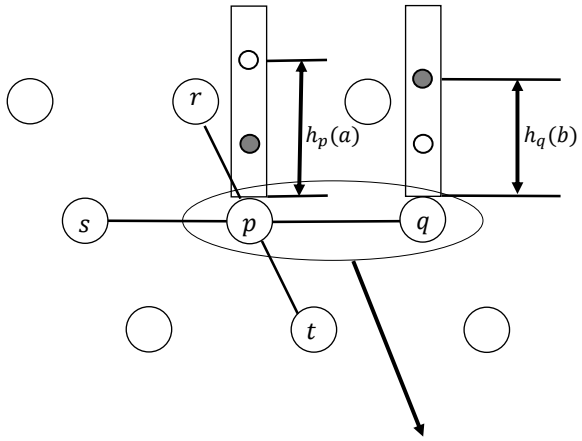

Figure 1.4: Shows two wells associated with two pixels in a clique. Here we assume only two labels a and b as shown in [4].

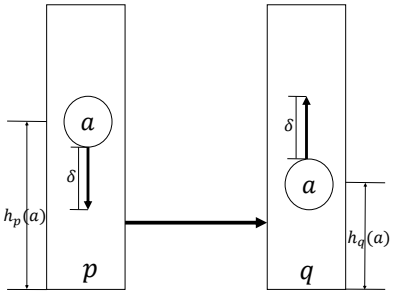

Figure 1.5: Shows a δ flow between two wells associated with the pixels p and q in a clique as shown in [4].

is floating below the chosen ball). The edge to t is directed from node q and its capacity is equal to the difference in heights in well q (active ball is floating above the chosen ball). The capacity of edges between the pixel nodes is controlled by the conditions given in Eq. (1.12). It is not too difficult to see that when flow is maximised in such a network the chosen balls get raised to the maximum possible height subject to the capacity constraints (ensuring feasibility) and effectively optimising the objective function, given in Eq. (1.9), for the chosen label.

Komodakis and Tziritas [60] use the above two label schema to approximate the multi label problem by iterating over the chosen label. Note that the dual variables define capacity constraints and the active label corresponds to the current feasible primal. The over all scheme is primal-dual in this sense.

1.3.4 Dual Decomposition

Unlike primal-dual based methods, dual only methods [56, 62, 95, 99] do not update the primal variables at each iteration and directly work on the dual. Dual of the problem can be decomposed as the sum of many sub problems of smaller size each of which can be solved individually. Solving smaller size subproblems increases the tractability of the problem that can be handled. Such methods are called Dual Decomposition (DD) methods. we give below a formal justification for the DD strategy. For simplicity we absorb the unary terms in the clique potentials [7]. The primal MAP problem takes the following form:

$$\min_{l \in \mathscr{L}^n} \sum_{\mathbf{c} \in \mathscr{C}} W_{\mathbf{c}}(l_{\mathbf{c}}).$$

Here we assume that labelings l_c's are in $\mathscr{L}^n$ such that the value of potential W_c is computed over the labeling corresponding to elements in c (ignoring rest of the elements). The above problem can be rewritten as follows:

$$\begin{aligned} &\min_{l \in \mathscr{L}^n} \sum_{\mathbf{c} \in \mathscr{C}} W_{\mathbf{c}}(l_{\mathbf{c}}), \\ &\text{s.t } l_{\mathbf{c}} = l, \quad \forall \mathbf{c} \in \mathscr{C}. \end{aligned}$$

Lagrange dual of the above problem can be written as:

$$\begin{aligned} g(\{\lambda_{\mathbf{c}}\}) &= \min_{l \in \mathscr{L}^n} \sum_{\mathbf{c} \in \mathscr{C}} W_{\mathbf{c}}(l_{\mathbf{c}}) + \sum_{\mathbf{c} \in \mathscr{C}} \lambda_{\mathbf{c}}(l_{\mathbf{c}} - l), \\ g(\{\lambda_{\mathbf{c}}\}) &= \min_{l \in \mathscr{L}^n} \sum_{\mathbf{c} \in \mathscr{C}} W_{\mathbf{c}}(l_{\mathbf{c}}) + \lambda_{\mathbf{c}}.l_{\mathbf{c}} - \sum_{\mathbf{c} \in \mathscr{C}} \lambda_{\mathbf{c}}.l. \end{aligned}$$

We can eliminate l by minimizing over it. Which implies that the sum of dual variables should be zero, i.e. $\sum_{\mathbf{c} \in \mathscr{C}} \lambda_{\mathbf{c}} = 0$ otherwise $g(\{\lambda_{\mathbf{c}}\})$ would not be feasible ($g(\{\lambda_{\mathbf{c}}\}) = -\infty$). Let set $\Lambda = \{\lambda_{\mathbf{c}}\}$ be the feasible set of dual variables. The dual maximization problem takes the form

$$\begin{aligned} &\max_{\{\lambda_{\mathbf{c}}\} \in \Lambda} g(\{\lambda_{\mathbf{c}}\}) = \sum_{\mathbf{c} \in \mathscr{C}} g_c(\lambda_c), \\ &\text{where, } g_c(\lambda_c) = \min_{l_{\mathbf{c}}} W_{\mathbf{c}}(l_{\mathbf{c}}) + \lambda_{\mathbf{c}}.l_{\mathbf{c}}. \end{aligned}$$

Each subproblem $g_c(\lambda_c)$ is minimized by projected subgradient method [60]. However minimization of the decomposed subproblems may not guarantee the minimization of the overall problem. The sum of the minima of the subproblems provides only a lower bound on the energy of the original MRF-MAP problem [59]. DD based methods have been adapted to do inference in higher order MRFs in [59, 97, 103].

1.3.5 Submodularity and its Implications for MRF-MAP

It should be noted that the inference methods mentioned above provide only approximate solutions for general MRF-MAP inference problem. Therefore, developing efficient and optimal inference techniques for some special subclass of clique potentials of interest has become an area of active research [50, 51, 54, 81]. One such special class of clique potentials whose importance in solving the MRF-MAP inference problems in computer vision is increasingly getting recognized is based on the notion of submodularity. For a 2 label problem, label 0 and 1 on a pixel can be seen as inclusion and exclusion of the pixel in a set. Therefore, clique potential can be seen as a set function. A set function f is said to be submodular [68] if it satisfies the following condition:

$$f(A) + f(B) \geq f(A \cup B) + f(A \cap B), \quad \forall A, B \subseteq V.$$

We will see the notion of submodularity for multi-label clique potentials in Chapter 5. Submodularity in clique potentials is important because strongly polynomial algorithms that output optimal results for minimizing such potentials are known to exist [44, 74, 86]. These algorithms cannot be used for solving practical MRF-MAP problems because the time complexity of these algorithms is bounded by very high order polynomials in the size of the problem. For any reasonable size problem the time taken is simply too large. The fact that strongly polynomial algorithms do exist when potentials are submodular opens up a new line of research which focuses on developing practically efficient algorithms for vision size problems. Hence, for the last two decades much research has been reported on such algorithms [1, 7, 8, 40**?**]. We will now summarize the progress in this direction.

The early inference methods for large scale MRF-MAP problems are based on variants of Graphcut [**?**]. It is well known that Graphcut gives optimal results for 2 label pairwise MRFs with the submodular potentials. We have already described the applications which have been modeled using Graphcut in Section 1.3. Researchers have also worked on improving the time complexity of the Graphcut. There have been series of works along this line. For example, Active cuts [47], capacity scaling algorithm [48], push-relabel based method [35] and Voronoi based Preflow Push (VPP) [5].

For multi-label MRFs the first polynomial time optimal algorithm was given by Ishikawa [40]. This algorithm is applicable to only first order submodular potentials. The algorithm is based on an encoding which converts the multi-label problem to an equivalent 2-label problem on an specifically created graph. As explained in [8] the graph has a path with number of nodes equal to the number of labels for each pixel. A node and the edge emanating from it on such a path corresponds to a label associated with the pixel. One end of all the paths are connected to a distinguished node s called the source and the other to t the sink. Capacities of the edges on such a path and the other edges introduced by

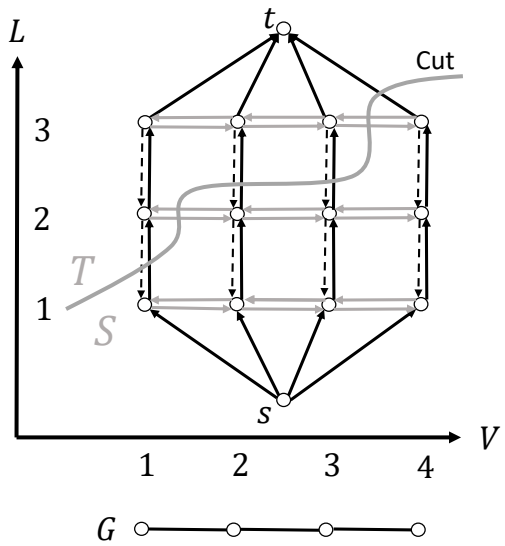

Figure 1.6: Graph corresponding to the problem with 4 pixels with 3 labels as drawn in [40].

the transformation are such that an (s,t) cut in the flow graph includes only one edge from such paths. The label of the pixel corresponds to that edge in the cut. The formal encoding which is implied in the flow graph construction corresponds can be recovered by labeling all the nodes on s side of the cut as 0 and the rest as 1. The sequence of labels associated with the vertices of a path are the same as the encoding formally defined in [8]. Figure (1.6) is an example of the flow graph created for a problem with 4 pixel and 3 labels. For each pixel, shown along horizontal axis in Figure (1.6), the path over the 3 labels is shown along the vertical axis. Note that it is assumed that the 4 pixels are interconnected as in the pixel graph shown in Figure 1.6. Note that all the edges between the nodes corresponding to pixels in the same clique are not shown in the figure. Interested readers are referred to [40] for the missing but crucial details. The construction is very rigid and works only for first order potentials.

1.3.6 Generic Cuts(GC)

The next big advance in the area is GC, the work of Arora et al. [7] where a flow based algorithm is given for 2-label high order submodular clique potentials (order greater than 2). This method is based on the primal-dual framework explained in Section 1.3.3. The algorithm optimizes the dual objective function given in Eq. (1.9) by transforming the dual into a max flow problem. The flow

problem interpretation is very similar to the wells and balls model [60]. Crucial to the flow graph construction is the combinatorial gadget corresponding to each clique shown in Figure 1.7. The gadget contains a node for each pixel in a clique and has two more auxiliary nodes n and m. Auxiliary node n has outgoing edges to the pixel nodes and node m has edges incident to it from the pixel nodes. The edge from m to n has infinite capacity. For submodular clique potentials, without loss of generality, all $V_{\mathbf{c},p,b}$ can be set to 0, thereby reducing expression (1.12) to:

$$\sum_{p:l^p_{\mathbf{c}}=a} V_{c,p,a} \leq W_{\mathbf{c}}(\mathbf{l_c}). \quad \mathbf{c} \in \mathscr{C}, \quad \mathbf{l_c} \in |\mathscr{L}|^{|\mathbf{c}|}. \tag{1.13}$$

Capacity of the edge from a auxiliary node n to the pixel nodes in a gadget is determined by the maximum amount of increase possible in $V_{c,p,a}$ such that the constraints given in Eq. (1.13) are satisfied. Consider the situation when pixel p sends δ flow to pixel q in the clique. This flow gets modeled by sending δ flow along the path $p \rightarrow m \rightarrow n \rightarrow q$ in the gadget as shown in Figure 1.7. Let the flow in edge $p \rightarrow m$ in the gadget be denoted by f^c_{pm} (c is the clique corresponding to the gadget). The affect of this flow is to bring down ball a in well p by amount f^c_{pm} and push up ball a in well q by the same amount. Therefore the relationship between the dual variables and flow is captured by $V_{c,p,a} = f^c_{np} - f^c_{pm}$. Figure 1.8 shows the flow path in a more general setting of flow pushing from pixel p to pixel r through pixel q of three different 4 cliques where the cliques have some pixels in common. The capacities of edges from node n to pixel nodes are recalculated w.r.t new values of $V_{c,p,a}$ variables after each flow augmentation. In the flow graph that gets created this way if flow is being modeled for the movement of ball a then there is a directed edge from s to each pixel node p. Similarly there is a directed edge from each pixel node p in whose well ball a is active to node t. The overall objective given in Eq.(1.9) can be maximized computing the maximum flow in the flow graph.

Experiments reported in [7] indicate that the algorithm optimises the MRF-MAP energy function over real life images involving millions of pixels very efficiently. However, note that the residual capacity calculation problem itself is a form of submodular function minimisation problem. Experiments in [7] have suggested that for cliques of size 16 or smaller it is more efficient to simply calculate residual capacity by enumeration. Putting this another way usefulness of GC in real images is limited to cliques of size 16 or smaller.

Arora and Maheshwari [6] have shown that this gadget based approach to MRF-MAP inference problem can be used to find approximate solutions for non-submodular potentials also. The importance of the method in [6] is that its output satisfies the property of weak persistence [15]. The method presented in [7] has also been extended to work for multi-label higher order MRF-MAP problem known as MLGC [8]. As discussed in the following section.

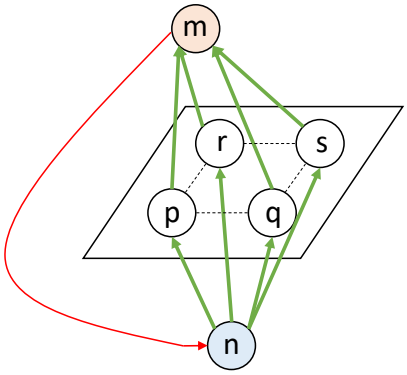

Figure 1.7: A gadget corresponding to a clique having 4 pixels as drawn in [7].

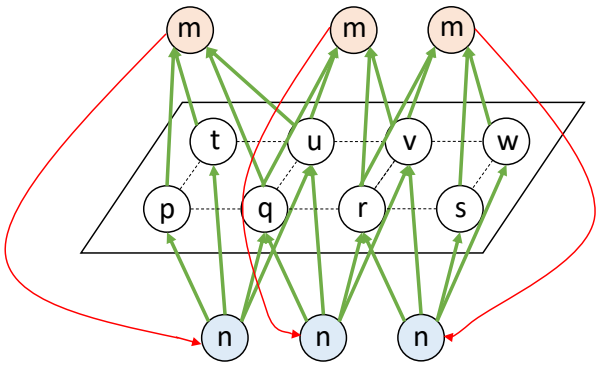

Figure 1.8: Flow graph corresponding to 3 cliques of size 4 as drawn in [7].

1.3.7 MLGC

The algorithm uses an encoding to convert the multi-label problem into 2 label problem. The encoding used in [8] is effectively the same as the labeling interpreted over the nodes along vertical axis (in Figure 1.6) in the flow graph construction of [40]. Encoding maps a label i to an m dimensional binary vector such that its first $m - i$ elements are 0 and the remaining elements are 1. The transformation of an m-label k clique problem using the encoding given in [8] (this encoding will be explained in detail in Chapter 5) results in a 2-label km clique problem. The size of the state space (subsets of the set of pixels in the clique) increases from m^k to 2^{mk} of which $2^{mk} - m^k$ states do not correspond to the states in original problem. That these extra states do not contribute to the minima is assured by assigning a very large value to the clique potential

for subsets corresponding to the extra states. There is a one to one mapping between the states of original (m-label k clique) problem and the remaining states (which are not extra) of the transformed 2-label mk clique. This 2-label mk clique problem can now be solved using GC [7]. Using this algorithm, the largest configuration that has been efficiently solved is 4-label 4 cliques [8]. This is because the original limitations of GC are carried over.

Finally we would like to point out that the submodular flows of the type captured by the gadget of GC are also the basis of another flow based method for minimizing submodular functions [1]. The crucial difference of this approach in comparison to GC is that it attempts to model flow between any two nodes of clique independently. By doing so the conditions when no further flow can be sent to a node from anywhere do not have the clear combinatorial structure that the gadget of GC enables. By using capacity scaling it has been possible to prove both termination as well pseudo polynomial time bounds similar to those of other submodular polyhdron based algorithms like those in [44]. The algorithm, however, has not generated any interest in researchers to test out its practical applicability so far. In the following section we look at the contribution of this thesis.

1.4 Thesis Contribution

The state of the art optimal inference algorithm which assume submodular potentials can handle practically cliques of size up to 16 only. At the same time there is consensus among researchers on the belief that quality of results will improve if larger cliques are used in MRF-MAP modelling of vision problems [41, 52, 59, 81]. There have been attempts to solve the problems like object segmentation using cliques of size 100 [45, 90]. But the clique potentials used are very restricted, and the energy minimisation does not always output optimal results. Nevertheless the results were interesting as the quality of segmentation improved. This opens up a broad spectrum of futuristic research agenda ranging from developing practical algorithms that can operate on cliques of size 1000 or larger, enlarging the class of potentials that can be handled by such algorithms, to study the actual usefulness of large cliques in improving the output quality of vision problems.

We have started with the premise that in this initiative, algorithmic ideas that would need to be explored would be different from the frameworks that define the current state of the art. Submodular polyhedron based strongly polynomial time algorithms [74] have a time complexity $O(n^6)$ or larger. Unless there is a significant breakthrough, these algorithms cannot be the basis for handling cliques of size 1000. Combinatorial GC like algorithms are ruled out as long as residual capacity calculation is done by enumeration.

In this thesis we have focused on investigating the usefulness of the Min Norm Point (MNP) algorithm [31] for resolving problems posed earlier. MNP algorithm is attractive because it is empirically the fastest submodular polyhedron based algorithm for minimizing a submodular function. The algorithm has so far not generated interest among researchers as while it has been shown to converge in principle, no good formal time complexity bound has been reported. The only complexity analysis [20] is similar to high order pseudo polynomial bound for algorithms in [44, 86]. Also, there is no obvious method to use it on practical problems which typically have millions of pixels with a large number of cliques embedded.

In this thesis the first problem we have worked on is to investigate the existence of a practically efficient Min Norm Point based algorithm which outputs optimal solutions to the MRF-MAP problem defined over images with around a million pixels and large cliques of size upto 1000. We show that block coordinate descent based framework can be used to handle such MRF-MAP structures. The results of this investigation are reported in Chapter 3.

Higher order cliques are good at capturing large scale spatial statistics in an image. But using small cliques along with large cliques can be used to get better and smoother results [104]. Such configurations have a large number of small cliques and smaller number of very large cliques. While in principle our block coordinate descent framework would be applicable, in practice the algorithm suffers from slow convergence. This slow convergence is mostly due to the very large number of small cliques. We have, therefore, asked the obvious question as to why an algorithm cannot be developed which handles small cliques by a combinatorial algorithm like GC and large cliques by MNP. The challenge, it turns out, is in the two different norms satisfied by GC and MNP outputs. Note that GC outputs solutions in ℓ_1 norm and MNP in ℓ_2 norm. We show that ℓ_1 norm solutions of GC can be transformed to satisfy ℓ_2 norm. Chapter 4 summarises our investigations in the development of such a hybrid algorithm.

The previous two problems we worked on assumed that MRF-MAP modeling required only 2 labels. As pointed out earlier many problems in computer vision (such as stereo correspondence problem and multi object segmentation) require each pixel to be labelled from a set that has more than 2 labels. We know that a multi-label problem can be transformed into an equivalent 2 label problem [8, 40]. Can a multi-label problem defined over large cliques be solved after it has been transformed to a 2 label problem using the algorithms developed so far? The straight forward answer is no. As pointed out earlier while explaining [8], such an encoding expands the clique size by a factor (number of labels) and introduces exponential number of elements in the state space which do not correspond to the states in the multi-label problem. In the combinatorial flow based method [8] the extra states get ignored as by assigning a very large value to them it is ensured that the optimal solution remains within the original state

space. As pointed out earlier, this enables the residual capacity calculation to be carried out in the GC style by enumeration. This strategy does not carry over in the large clique scenario as MNP algorithm works on the whole transformed state space which has an additional high degree exponential number of elements. In addition since the extra states introduced have very large values, the matrix inversion operation crucial in computing affine minimizer in the MNP algorithm fails due to floating point errors. As to how this challenge can be overcome is the subject of discussion in Chapter 5. We propose a strategy which while working on the transformed state space limits the use of the MNP algorithm only to those situations where the intermediate solutions lie in the original state space. When the intermediate solution moves out of the original state space we show that it is possible to move the intermediate solution back to the original state space by solving a series of max flow problems. What is interesting is that the flow graph over which this stage of optimization is carried out is a result of a very careful analysis of the structure of the expanded state space. The reader is encouraged to carefully go through this discussion as we feel that the ideas explored in this process may have wider applicability.

Chapter 2 summarises the formal notation and the basic theory of submodular polyhedron required to follow the discussion in Chapters 3, 4, and 5. This has been deliberately kept terse as the material is now very standard and available in many well written monographs and survey papers [30, 68, 93].

Chapter 2

Background

In this chapter we discuss the standard results used in submodular function minimization (SFM). We also briefly describe how MRF-MAP problem is linked with SFM. Let $f : 2^{\mathscr{V}} \rightarrow \mathbb{R}$ be a submodular function defined over a set $\mathscr{V}$. The objective for SFM is to find a minimizer set, $S^* = \arg\min_{S \subseteq \mathscr{V}} f(S)$. Without loss of generality, we will assume that $f(\{\phi\}) = 0$. The *submodular polyhedron* $P(f)$ defined as follows:

$$P(f) = \{x \mid x \in \mathbb{R}^{\mathscr{V}},\ \forall\ U \subseteq \mathscr{V} : x(U) \leq f(U)\}, \tag{2.1}$$

where $x(U) = \sum_{v \in U} x(v)$. The term $x(v)$ denotes the element at index v in the vector x. It is useful to consider the face of $P(f)$ for which $x(\mathscr{V}) = f(\mathscr{V})$. The *base polyhedron* $B(f)$ is defined as:

$$B(f) = \{x \mid x \in P(f), x(\mathscr{V}) = f(\mathscr{V})\}. \tag{2.2}$$

Figure 2.1 shows $P(f)$ and $B(f)$ of a submodular function defined over set with two elements. A vector in the base polyhedron $B(f)$ is called a *base*. A base is called an *extreme base* if it is a corner point of the base polyhedron. An extreme base is essentially the solution to the following Linear Program LP

$$b = \arg \max_{x \in B(f)} x^T w. \tag{2.3}$$

Here w is a line orientation with norm $\|w\| = 1$ [93]. From now on the extreme bases will be represented by a vector b. We now discuss the problem of determining an extreme base of $B(f)$. Suppose that elements of w are ordered such that

$$w_1 \geq w_2 \geq w_n \geq 0. \tag{2.4}$$

Note that all elements of w have to be positive because otherwise solution of LP defined in Eq.(2.3) becomes unbounded [93]. Let $v_1, v_2, ..., v_n$ be the order of elements in $\mathscr{V}$ corresponding to the indices of ordered w. The values of b are calculated greedily to ensure that the element of w with the largest value gets

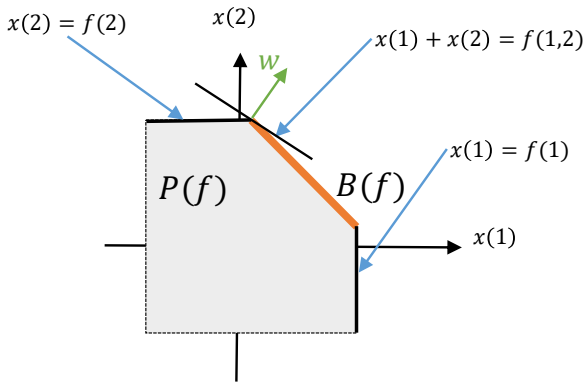

Figure 2.1: Base polyhedron and submodular polyhedron of the submodular function f defined on the set of two elements as drawn in [93].

maximum reward when multiplied with an element of b. This implies that the value of $b(1)$ should be set as high as possible. From the constraints that define the submodular polyhedron we have $b(1) \leq f(\{v_1\})$. Therefore, we can set $b(1)$ as

$$b(1) = f(\{v_1\}). \tag{2.5}$$

Now for the second value $b(2)$ we have the constraint $b(2) \leq f(\{v_2\})$ and $b(1) + b(2) \leq f(\{v_1, v_2\})$. Submodularity implies that $f(\{v_1, v_2\}) - f(\{v_2\}) \leq f(\{v_1\})$. Therefore the largest value $b(2)$ can be set is given by

$$b(2) = f(\{v_1, v_2\}) - f(\{v_2\}). \tag{2.6}$$

The greedy strategy described above is the basis for Edmond's Algorithm for computing extreme bases. As we will show below and as it is implied by the discussion above an extreme base corresponds to an ordering of w and the enforced ordering of the elements of $\mathscr{V}$ that it implies. We denote an ordering of the elements of $\mathscr{V}$ by a total order $\prec$ such that: $\prec : v_1 \prec \ldots \prec v_n$ and $b^{\prec}$ represents its corresponding extreme base. Let $i_{\prec}$ be the set of elements $\{1, \ldots, i\}$. The algorithm initializes the first element as $b^{\prec}(1) = f(\{v_1\})$ and rest of the elements as $b^{\prec}(k) = f(k_{\prec}) - f((k-1)_{\prec})$. Formally the Procedure (1) gives the detail of computing an extreme point.

Now we will show that the vector returned by Edmond's Algorithm lies in base polyhedron.

Lemma 2.1. *For a given ordering $\prec : v_1 \prec \ldots \prec v_n$, the vector $b^{\prec}$ returned by procedure 1 lies in base polyhedron of the function f.*

Procedure 1 Edmond's Algorithm

Input: A vector w.

Output: An extreme base $b^{\prec} = \arg\max_{x \in B(f)} w^T x$.

1: Define an ordering $\prec$ on the indices, such that $w_1 \geq w_2 \ldots \geq w_n$.

2: Denote by $i_{\prec}$ the set of elements $\{1, \ldots, i\}$. Set $b^{\prec}(i) = f(i_{\prec}) - f((i-1)_{\prec})$.

Proof. First let us consider the value of $b^{\prec}(\mathscr{V})$,

$$b^{\prec}(\mathscr{V}) = \sum_{v \in \mathscr{V}} b^{\prec}(v) \tag{2.7}$$

$$= \sum_{i \in [1, |\mathscr{V}|]} f(i_{\prec}) - f((i-1)_{\prec}) \tag{2.8}$$

$$= f(\mathscr{V}) \qquad \text{(Rest of the terms get cancelled out telescopically.)}$$

Now we will show that for any set $S \subset \mathscr{V}$ we have $b^{\prec}(S) \leq f(S)$ using induction. Let i be the largest index in $\mathscr{V} = \{v_1, v_2, ..., v_n\}$ such that $v_i \in \mathscr{V}$. For $i = 1$ it is trivial to see now let us consider the case for $i > 1$ as follows,

$$b^{\prec}(S) = b^{\prec}(v_i) + b^{\prec}(S - v_i) \tag{2.9}$$

$$= f(i_{\prec}) - f((i-1)_{\prec}) + b^{\prec}(S - v_i) \tag{2.10}$$

$$\leq f(i_{\prec}) - f((i-1)_{\prec}) + f(S - v_i) \tag{2.11}$$

$$\leq f(S). \qquad \text{(From submodularity)}$$

Hence proved. □

The reader is referred to [30, 93] for the formal proof that the output of Edmond's Algorithm is a solution to the LP problem given in Eq. (2.3).

For any vector $x \in \mathbb{R}^{\mathscr{V}}$, we denote by x^- the vectors in $\mathbb{R}^{\mathscr{V}}$ defined by $x^-(v) = \min\{0, x(v)\}$ for $v \in \mathscr{V}$. It is easy to see that for any base x and a subset U: $x^-(\mathscr{V}) \leq x(U) \leq f(U)$. The Min Max Theorem given below (for formal proof refer to [68, 86, 93]) shows that the inequalities provide a tight lower bound in the sense that the maximum value of this lower bound is also the minimum value of the function f.

Theorem 2.2 (Min Max Theorem)**.** *Given a submodular function $f : 2^{\mathscr{V}} \to R$, we have*

$$\max\{x^-(\mathscr{V}) \mid x \in B(f)\} = \min\{f(U) \mid U \subseteq \mathscr{V}\}.$$

Most of the algorithms with polynomial time complexity for SFM actually solve this dual problem [44, 74, 86]. We focus on the Min Norm Point (MNP) algorithm which is at the core of all the algorithms reported in this thesis. MNP algorithm reduces the problem of finding the minimum of f to finding a base x with minimum ℓ_2 norm in base polyhedron. The following lemma establishes the equivalence of the problem:

Lemma 2.3 (Min Norm Equivalence)**.** *Suppose x^* is a minimizer of the problem:*

$$\min \|x\|^2 \text{ subject to } x \in B(f).$$

Then a minimizer S^* for f can be obtained as follows:

$$S^* = \{u \in \mathscr{V} \mid x^-(u) \leq 0\}.$$

The reader is referred to [31, 93] for a formal proof of Lemma (2.3). Intuitively note that the sum of all the elements of x is a constant. Therefore maximising the negative elements of x is equivalent to minimising the positive elements of x. Taken together this is equivalent to minimising the norm $\|x\|$.

Procedure 2 Affineminimizer(S)

1: $B = [b_0, b_1, .., b_m]$;
\# given S=$\{b_0, b_1, .., b_m\}$
2: $B_0 = [b_1 - b_0\ b_2 - b_0\ ...\ b_m - b_0]$;
3: y= $b_0 - B_0[(B_0^T B_0)^{-1}(B_0^T b_0)]$;
\# Min-norm point in affine-hull(S) as per lemma 2.4
4: $\alpha = B^{-1}y$;
5: **return** (y,α);

Procedure 3 Project(S,x,α)

1: $\theta = \min_{i:\alpha_i<0} \lambda_i/(\lambda_i - \alpha_i)$;
\# Find θ such that $\theta\alpha_i + (1-\theta)\lambda_i \geq 0$
2: $x \longleftarrow \theta y + (1-\theta)x$;
3: $\lambda_i \longleftarrow \theta\alpha_i + (1-\theta)\lambda_i$;
4: $S = \{i : \lambda_i > 0\}$;
\# Delete points which have λ_i=0 this deletes atleast one point.

In the following section we will describe the MNP algorithm [31].

2.1 MNP Algorithm

At a very macro level the strategy of the Min Norm Point algorithm can be explained as follows. At any stage the algorithm maintains a set of extreme bases and a point with minimum ℓ_2 norm in the convex hull of these extreme bases. Note that this convex hull of the extreme bases is a sub space of the base polyhedron. This subspace is made bigger by adding a new extreme base, and the new point with minimum ℓ_2 norm in the new subspace of the base polyhedron is computed. If the new minimum norm point is not different than the previous minimum norm point then the algorithm terminates as it is not difficult to show

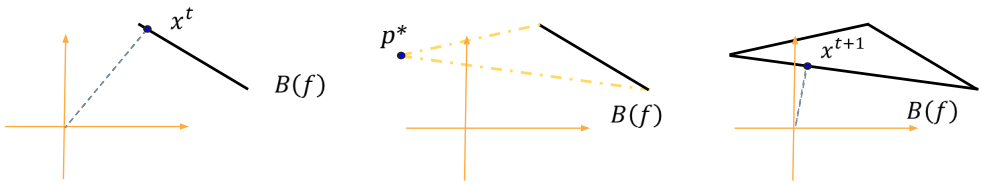

Figure 2.2: The visual example of an iteration of MNP algorithm.

that any further expansion of the subspace will not lead to further reduction in the norm. We now give details of how these steps get actually implemented.

Initialially any extreme base, denoted by q, is chosen as the starting sub space, denoted by S, of the base polyhedron. For this subspace the q itself is minimum norm point. We denote such a min norm point by x. A new extreme base, q, is computed using Edmond's Algorithm. That is, $q = \arg\min_{b \in B_f} x^T b$. The point q is added to the subspace S. Now the task is to compute the ℓ_2 norm minimizer in the new subspace, $S \cup \{q\}$. Since computing the ℓ_2 norm minimizer in the affine subspace of $S \cup \{q\}$ is easy (details given in Lemma 2.4) the minimum ℓ_2 norm in the affine hull is first computed.

Lemma 2.4. *Let $B=\{b_0, b_1, b_2 ... b_m\}$ be a set of extreme bases. Let $B_0 = [b_1 - b_0 \; b_2 - b_0 \; \ldots \; b_m - b_0]$ denote the extreme bases translated by the first element. Then, $y = b_0 - B_0 * \{(B_0^T * B_0)^{-1} * (B_0^T b_0)\}$ is the minimum norm point in the affine hull of B.*

Proof. A point in the affine-hull of B can be represented as:

$$b = \sum_i \alpha_i b_i, \tag{2.12}$$

$$s.t. \sum_i \alpha_i = 1.$$

and the problem of finding minimum norm point in the affine hull of B can be stated as:

$$\min_b b^T b.$$

Let $A = [\alpha_1, \alpha_2, \ldots, \alpha_m]$ denote the set of all the coefficients excluding α_0. Then putting $\alpha_0 = 1 - \sum_{i=1}^m \alpha_i$ in Eq.(2.12) gives

$$b = b_0 + B_0 A^T, \tag{2.13}$$

and the minimization objective can be equivalently written as:

$$\min_A (b_0 + B_0 A^T)^T (b_0 + B_0 A^T).$$

The objective is quadratic in A for which the minimum can be found by differentiating with respect to A and setting the result to 0. That is

$$(b_0 + B_0 A^T)^T (b_0 + B_0 A^T),$$

$$b_0^T b_0 + b_0^T B_0 A^T + A B_0^T b_0 + A B_0^T B_0 A^T.$$

By differentiating above expression w.r.t A and setting it to zero, we obtain following:

$$B_0^T b_0 + B_0^T B_0 A = 0,$$

$$A = -(B_0^T * B_0)^{-1} * (B_0^T b_0).$$

Putting above obtained value of A in Eq.(2.13) gives the minimum point as:

$$b = b_0 - B_0[(B_0^T B_0)^{-1}(B_0^T b_0)].$$

□

Procedure 4 MNP Algorithm

$q \longleftarrow$ Arbitrary extreme base;
$x \longleftarrow q$;
$S \longleftarrow \{q\}$;
while (true) **do**
  $q \longleftarrow \arg\min_{b \in B(f)} x^T b$;
  # find q using Edmond's Algorithm
  if $(\|x\|^2 \leq x^T q + \epsilon)$ **then**
    # Terminate when x is minimum norm point
    break;
  end if
  $S = S \cup \{q\}$;
  while (true) **do**
    (y, α) = Affineminimizer(S);
    if $\alpha_i \geq 0 \quad \forall i$ **then**
      # if y is in convex hull of S then terminate
      break;
    else
      (S,x)= Project(S,x,α)
    end if
  end while
end while

The affine minimizer (y) computed in the Lemma 2.4 may not lie in the convex subspace. If it doesn't then a point in the convex subspace is computed which is nearest to y along the line joining the y and the minimum norm point

$\lvert\mathscr{V}\rvert$	MNP [31]	IW [43]	IFF [44]	SCH [86]	FI [29]
100	0.00	0.41	1.00	2644.52	277.36
200	0.00	4.92	18.69	-	-
300	0.00	21.77	115.44	-	-
400	0.00	67.12	369.13	-	-
500	0.00	166.73	894.33	-	-
600	0.01	325.26	2820.83	-	-
700	0.01	568.54	-	-	-

Table 2.1: All running time are in seconds.

obtained at previous iteration. This step is known as projecting y onto the convex subspace. The MNP algorithm based on the above is given in the Procedure 4.

Note that, in the inner cycle or minor cycle, the algorithm eliminates the extreme points in a way that leave the remaining set of points affinely independent of each other. Singularity of B_0 (in Procedure 2) is only possible if the new extreme base already exists in the current set S. This would imply that the convex combination of the new extreme point and the current solution vector can improve the norm of the solution vector. This is a contradiction as the current set is already affinely independent with optimal coefficients.

Figure 2.2 shows graphically the main steps of the MNP algorithm. Procedure 2 gives the steps to compute the affine minimizer and Procedure 3 gives the details of the projection step.

The proof of convergence of the MNP algorithm is given in [20]. The only time complexity bound is given in [20] where it has been shown that MNP algorithm takes $O((n^5EO + n^7)F^2)$ for finding the minimizer of f. Here, $F = \max_{U,v} |f(U \cup \{v\}) - f(U)|$, where $U \subseteq \mathscr{V}, v \in V$. However, in practice it is the fastest algorithm reported. Fujishige [31] has given running time comparison of the MNP algorihtm with a number of other submodular polyhedron based algorithms. Table 2.1 gives the time comparison as reported in [31] for the following submodular function:

$$f(X) = |X||\mathscr{V} - X| - \sum_{j \in X}(5j - 2|\mathscr{V}|), \quad \forall X \subseteq \mathscr{V}. \tag{2.14}$$

However experimentally observed time complexity of the MNP algorithm is $O(n^{3.3})$ [31, 67].

Now we will look at how MRF-MAP inference problem can be posed as submodular function minimization problem.

2.2 MRF-MAP Inference as SFM

For a better readability, we re-write the MRF-MAP problem as follows:

$$l^*_{\mathscr{P}} = \arg\min_{l_{\mathscr{P}} \in \mathscr{L}^{|\mathscr{P}|}} E(l_{\mathscr{P}}) = \arg\min_{l_{\mathscr{P}} \in \mathscr{L}^{|\mathscr{P}|}} \sum_{p \in \mathscr{P}} D_p(l_p) + \sum_{\mathbf{c} \in \mathscr{C}} W_{\mathbf{c}}(\mathbf{l_c}), \tag{2.15}$$

For a 2 label problem, label 0 and 1 on a pixel can be seen as inclusion and exclusion of the pixel in a set. Therefore, clique potential $W_{\mathbf{c}}$ can be seen as a set function. If we assume the clique potentials to be submodular then energy function E becomes sum of submodular functions. Since sum of submodular function is also a submodular function [1] then in principal SFM techniques are applicable to MRF-MAP problem. Practically, such techniques suffer from high overhead of memory involved in operations and their running time as well. In the next chapter we will present how to overcome such problems and use SFM for an efficient inference. The inference for the multi label problems is often formulated as the inference on equivalent transformed 2-label problems. In the following section we describe one such encoding which transforms a multi-label problem into 2-label while preserving the submodularity.

2.2.1 Transforming a Multi-Label Problem into a 2-Label Problem

Here, we summarize the transformation to convert a multi-label to a 2-label problem as suggested in [8, 40]. Consider an *unordered* set of pixels $\mathscr{P} = \{p_1, \ldots, p_i, \ldots, p_n\}$, and an *ordered* set of labels $\mathscr{L} = \{1, \ldots, m\}$. To save the notation clutter, whenever obvious, we denote a pixel simply using variables p, q without the subscript index.

Definition 2.5 (Binary Encoding)**.** The encoding $E : \mathscr{L} \to \mathbb{B}^m$ maps a label $i \in \mathscr{L}$ to a m dimensional binary vector such that its first $m - i$ elements are 0 and the remaining elements are 1.

For example, $E(1) = (0, \ldots, 0, 0, 1)$, and $E(2) = (0, \ldots, 0, 1, 1)$. Let us denote the encoded label vector corresponding to a pixel p_i as $\gamma_i = (p_i^1, \ldots, p_i^m), p_i^j \in \{0, 1\}$. We denote by $\Gamma \in \mathbb{B}^{mn}$, the vector obtained by concatenating all encoded vectors $\gamma : \Gamma = (\gamma_1, \ldots, \gamma_i, \ldots, \gamma_n)$. The vector Γ represents encoding of labeling configuration over all the pixels. We also define a *universal set* containing all elements of $\Gamma : \mathscr{V} = \{p_1^1, \ldots, p_1^m, \ldots, p_n^1, \ldots, p_n^m\}$.

Definition 2.6 (Universal Ordering)**.** Assuming an arbitrary ordering among the pixels, the universal ordering, defines a total ordering of the elements p_i^j, where $i \in \mathbb{Z}_{1:n}, j \in \mathbb{Z}_{1:m}$:

$$\prec_0 : p_1^1 \prec \cdots \prec p_1^m \prec \cdots \prec p_n^1 \cdots \prec p_n^m.$$

We denote by $S \subseteq \mathscr{V}$, called *state*, set of all the elements, p_i^j of Γ labeled as 1. Note that there are 2^{mn} possible states, however only m^n of them correspond to valid Γ vector obtained by encoding labeling configurations over the pixels. We call such states as *valid states*. If label of a pixel p_i is denoted as $l_i \in \mathscr{L}$, a valid state may be represented as: $S = \{E(l_1), \ldots, E(l_i), \ldots, E(l_n)\}$. Similarly $S_p = \{E(l_p)\}$ includes elements corresponding to pixel p.

Definition 2.7 (Valid Ordering/Extreme Base)**.** An ordering $\prec$ is called a valid ordering, if for any $p_i^j, p_i^k \in \mathscr{V}$, $j > k \Rightarrow p_i^j \prec p_i^k$. An extreme base $b^{\prec}$ is called a valid extreme base, if it corresponds to a valid ordering.

The states, orderings or extreme-bases which are not valid are called *invalid*. We denote the set of all valid states by Z.

2.2.2 Binary Submodular Extension Function

Definition 2.8 (Covering State, Minimal Covering State)**.** For an arbitrary state, S, a valid state, $\hat{S} \in Z$, is called covering if $S \subseteq \hat{S}$. There may be multiple covering states corresponding to a S. The one with the smallest cardinality among them is referred to as the minimal covering state, and is denoted by $\overline{S}$. There is a unique minimal covering state corresponding to any S. For a valid state $S = \overline{S}$.

Let F be the original multi-label submodular function defined over the pixels $\mathscr{P}$. We refer the readers to [8] for definition of multi-label higher order submodular functions. We encode the multi-label function to a submodular pseudo-Boolean function f defined over set $\mathscr{V}$ of size mn as follows:

Definition 2.9 (The Extended Binary Set Function)**.**

$$f(S) = \begin{cases} F(\ldots, l_i, \ldots), & \text{if } S = \{\ldots, E(l_i), \ldots\} \\ f(\overline{S}) + (|\overline{S}| - |S|)L & \text{otherwise} \end{cases}$$

Here $l_i \in \mathscr{L}$ is label of pixel p_i, and $L \gg M = [\max_{S \in Z} f(S) - \min_{S \in Z} f(S)]$.

It is easy to see that $f(S)$ can also be defined as follows:

Definition 2.10 (The Extended Binary Set Function: Alternate Definition)**.**

$$f(S) = f(\overline{S}) + \sum_{p \in \mathscr{P}} (|\overline{S}_p| - |S_p|)L,$$

where $\overline{S}_p \subset \overline{S}$, and $S_p \subset S$ are the subsets containing elements corresponding to pixel p in $\overline{S}$ and S respectively.

Theorem 2.11. *The extended binary set function f as given by Definition 2.9 is submodular, and* $\min f(\cdot) = \min F(\cdot)$.

Please read the Section 5.1.1 of Chapter 5 for full proof.

Chapter 3

MNP Algorithm for Higher Order Inference

Over the last decade many computer vision problems ranging from image restoration [33], segmentation of videos [82] and images [69], super resolution [18], texture synthesis [77], stereo matching [37] to object detection [89] have been formulated as MRF-MAP inference problems. Our focus in this chapter is on developing an efficient MRF-MAP inference algorithm for problems defined over 2-labels involving submodular potentials defined over large cliques. 2-label problems are important as apart from having the applications in their own right, inference for the multi-label problems is often formulated as a series of inference problems defined over 2-labels [18, 28]. We start with a brief survey of the state of the art.

Since a 2-label MRF-MAP problem with submodular clique potentials becomes minimization of sum of submodular functions the problem is essentially minimization of a submodular function defined on a large set. The MNP algorithm presented in Chapter 2 is applicable theoretically. In practice on a practical computer vision problem with a million of pixels the MNP algorithm can not be applied due to matrix inversion operation on million by million matrix. Optimal and practical state of the art inference methods for such MRFs can handle the maximum clique size upto 16 [7]. Higher order clique potentials can encode various structural and complex dependencies between pixels. It has been adequately shown by various authors [41, 54, 59, 81, 100] that using such complex dependencies greatly improves the solution quality. Therefore, our focus in this chapter is on extending the limits of the clique size that can be handled by current state of the art inference methods, specifically on 2-label MRF-MAP problem.

Considering the complexity of inference problem for general sum of submodular functions, researchers have worked on developing efficient algorithms for various special subsets of submodular functions [78, 81, 90]. One of the more successful efforts is that of Jegelka et al. [45] where use of two well known (block coordinate descent and Douglas Rachford splitting) has been explored. The potential functions are sum of special submodular functions for which $\min_{S \subseteq V} (f(S) - a(S))$

can be computed efficiently, where $a \in R^V$ is a constant vector.

In this work we consider the applicability of the submodular polyhedron based MNP algorithm. As shown in Lemma 2.3 of the previous chapter, MNP minimizes the norm $\|x\|^2$ where x is a vector in the base polyhedron. Taking the motivation from the work done on dual decomposition based methods [62] (shown in Section (1.3.4)) it is natural to ask if the objective of MNP can also be decomposed over the cliques. We describe in the next section that indeed it is possible and show how to solve each subproblem by the MNP algorithm. We also show that the decomposition presented in this work is tight and the proposed method converges to optimal (Section 3.3). We propose an algorithm for inference in 2-label higher order MRF-MAP problem (1.1), when the clique potentials are submodular. The proposed algorithm has following specific advantages/strength:

1. **Structure:** Our algorithm uses the ideas from min-norm-point algorithm [32] and adapts them for exploiting the sum of submodular structure in our problems.

2. **General:** Unlike contemporary approaches, the proposed algorithm can give optimal inference for general submodular functions with no extra conditions on the type of clique potential.

3. **Scalable:** In a significant improvement over state of the art, our algorithm can easily scale to problems with clique sizes ranging up to many hundreds compared to 16 for current state of the art [7].

4. **Efficient:** When the earlier state of the art can take hours for inference on clique size 16, our experiments show that the proposed algorithm converges with optimal inference in a matter of few seconds on problems with order of magnitude larger clique sizes.

The key to the scalability of the algorithms for computer vision problems is their ability to exploit the structure present in such problems. In this work, we adapt the min norm point algorithm [31] for exploiting the sum of submodular structure. We start with the theoretical foundations for our work, followed by the algorithm description. We show the effectiveness of our algorithm by running it on real computer vision problems in the Section 3.4. In the following section we describe the theoretical contribution and our proposed algorithm SoSMNP.

3.1 Theoretical Results

Let f be a submodular set function on a set $\mathscr{V}$ of the form: $f(S) = \sum_{\mathbf{c} \in \mathscr{C}} f_{\mathbf{c}}(S \cap \mathbf{c})$, where $\mathscr{C} \subseteq 2^{\mathscr{V}}$ is a set of subsets of $\mathscr{V}$, and $f_{\mathbf{c}} : 2^{\mathbf{c}} \rightarrow \mathbb{R}$ are submodular

functions (clique potentials). Our objective is to find the minimizer set $S^* = \arg\min_S f(S) = \arg\min_S \sum_{\mathbf{c}} f_{\mathbf{c}}(S \cap \mathbf{c})$. Since each $f_{\mathbf{c}}$ is submodular, we can associate, with each $f_{\mathbf{c}}$, a base polyhedron given by:

$$B(f_{\mathbf{c}}) := \Big\{ y_{\mathbf{c}} \in \mathbb{R}^{\mathbf{c}} \;\mid\; y_{\mathbf{c}}(U) \leq f_{\mathbf{c}}(U), \quad \forall\, U \subseteq \mathbf{c}; \tag{3.1}$$

$$y_{\mathbf{c}}(\mathbf{c}) = f_{\mathbf{c}}(\mathbf{c}) \Big\}. \tag{3.2}$$

As defined earlier, $y_{\mathbf{c}}$ denotes a vector of scalars for every element in $\mathbf{c}$: $y_{\mathbf{c}}(U) := \sum_{v \in U} y_{\mathbf{c}}(v), \quad \forall\, U \subseteq \mathbf{c}$.

We define a total order $\prec_{\mathbf{c}}^{j}$ for any $\mathbf{c} \in \mathscr{C}$ and correspondingly define the extreme base $q_{\mathbf{c},j} \in \mathbb{R}^{\mathbf{c}}$ using Edmond's algorithm. It follows that any convex combination of the extreme bases i.e., $y_{\mathbf{c}} = \sum_{j=1}^{k} \lambda_{\mathbf{c},j} q_{\mathbf{c},j}$, lies on the submodular polyhedron $B(f_{\mathbf{c}})$. We can now state the following:

Lemma 3.1. *Let $x(S) = \sum_{\mathbf{c}} y_{\mathbf{c}}(\mathbf{c} \cap S)$ where each $y_{\mathbf{c}}$ lies on the submodular polyhedron $B(f_{\mathbf{c}})$. Then, the vector x lies on the base polyhedron $B(f)$.*

Proof. Consider $U \subseteq \mathscr{V}$. To prove the lemma, it suffices to show that $x(U) \leq f(U)$ and $x(\mathscr{V}) = f(\mathscr{V})$. It is easy to see that:

$$x(U) = \sum_{\mathbf{c}} y_{\mathbf{c}}(\mathbf{c} \cap U) \leq \sum_{\mathbf{c}} f_{\mathbf{c}}(\mathbf{c} \cap U) = f(U).$$

The inequality follows from the fact that each $x_{\mathbf{c}}$ lies on the submodular polyhedron $B(f_{\mathbf{c}})$. Further,

$$\begin{aligned} x(\mathscr{V}) &= \sum_{\mathbf{c}} y_{\mathbf{c}}(\mathbf{c} \cap \mathscr{V}) = \sum_{\mathbf{c}} y_{\mathbf{c}}(\mathbf{c}) = \sum_{\mathbf{c}} f_{\mathbf{c}}(\mathbf{c}) \\ &= \sum_{\mathbf{c}} f_{\mathbf{c}}(\mathbf{c} \cap \mathscr{V}) = f(\mathscr{V}). \end{aligned}$$

Hence, proved. □

Lemma 3.2. *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., $y_{\mathbf{c}} \in B(f_{\mathbf{c}}) \forall \mathbf{c}$.*

Proof. Let us first prove the lemma for the case when x is an extreme base of $B(f)$. Using Edmond's algorithm, $\exists$ an ordering $\prec$ over the variables such that $x(i) = f(i_{\prec}) - f((i-1)_{\prec})$ (see Procedure 1 for details). Then, using the submodular decomposition of f, we have $x(i) = \sum_{\mathbf{c}} (f_{\mathbf{c}}(i_{\prec} \cap \mathbf{c}) - f((i-1)_{\prec} \cap \mathbf{c}))$. Given a clique $\mathbf{c}$, define $y_{\mathbf{c}}(i) = f_{\mathbf{c}}(\mathbf{c} \cap i_{\prec}) - f_{\mathbf{c}}(\mathbf{c} \cap (i-1)_{\prec})$. It is easy to see that $y_{\mathbf{c}}$ is an extreme base for $B(f_{\mathbf{c}})$, thus implying $x(i) = \sum_{\mathbf{c}} y_{\mathbf{c}}(i)$ for some $y_{\mathbf{c}} \in B(f_{\mathbf{c}}), \forall c$. The same can be extended to show that $x(S) = \sum_{\mathbf{c}} y_{\mathbf{c}}(\mathbf{c} \cap S)$ by adding the corresponding equations for each $i \in S$.

Next, let us consider the case when $x \in B(f)$ but may not be an extreme base. Since, every vector in the base polyhedron can be expressed as a convex combination of extreme bases, we have $x(S) = \sum_i \lambda_i b_i(S)$ where b_i is an extreme base of $B(f)$. Using the earlier proof for extreme bases, there exist $q_{\mathbf{c},i} \in B(f_{\mathbf{c}})$ such that $b_i(S) = \sum_{\mathbf{c}} q_{\mathbf{c},i}(\mathbf{c} \cap S)$. This means that $x(S) = \sum_i \lambda_i \sum_{\mathbf{c}} q_{\mathbf{c},i}(\mathbf{c} \cap S) = \sum_{\mathbf{c}} \sum_i \lambda_i q_{\mathbf{c},i}(\mathbf{c} \cap S)$. The inner sum is a convex combination of extreme bases $q_{\mathbf{c},i}$ and hence, lies inside the base polyhedron $B(f_{\mathbf{c}})$. Let $y_{\mathbf{c}} = \sum_i \lambda_i q_{\mathbf{c},i}(\mathbf{c} \cap S)$. Therefore, $x(S) = \sum_{\mathbf{c}} y_{\mathbf{c}}(\mathbf{c} \cap S)$ where $y_{\mathbf{c}} \in B(f_{\mathbf{c}})$. Hence, proved. □

3.2 SoSMNP

Our goal in this section is to devise a strategy for minimizing $\|x\|^2$ subject to $x \in B(f)$. From lemma 3.2, every vector $x \in B(f)$ can be expressed as a sum $x(S) = \sum_{\mathbf{c}} y_{\mathbf{c}}(\mathbf{c} \cap S)$, where each $y_{\mathbf{c}} \in B(f_{\mathbf{c}})$. In other words, every vector x can be written as a sum of vector $y_{\mathbf{c}}$'s which lie on respective base polyhedrons. Therefore, we can devise a strategy for minimizing $\|x\|^2$ by applying block coordinate descent where the blocks are represented by the variables $y_{\mathbf{c}}$'s as defined above.

Let $x_{\mathbf{c}}$ denote the restriction of the base vector x to clique $\mathbf{c}$. Further, let $\bar{x}_{\mathbf{c}}$ denote the restriction of x to variables $\{v \in \mathscr{V} \mid v \notin \mathbf{c}\}$. By definition, $\|x\|^2 = \|x_{\mathbf{c}}\|^2 + \|\bar{x}_{\mathbf{c}}\|^2$. Further, it is clear that $y_{\mathbf{c}}$ contributes to the to vector x only through x_c. Therefore, when trying to minimize $\|x\|^2$ with respect to the block $y_{\mathbf{c}}$, we can simply focus on the component $x_{\mathbf{c}}$. Let a_c denote the contribution of cliques other than $\mathbf{c}$ to the component x_c. Since, we are doing block coordinate descent over variables in $y_{\mathbf{c}}$, $a_{\mathbf{c}}$ can be treated as a constant. Further, noting that $y_{\mathbf{c}} = \sum_j \lambda_j q_{\mathbf{c},j}$, we have:

$$x_{\mathbf{c}} = y_{\mathbf{c}} + a_{\mathbf{c}} = \sum_j \lambda_{\mathbf{c},j} q_{\mathbf{c},j} + a_{\mathbf{c}} = \sum_j \lambda_{\mathbf{c},j}(q_{\mathbf{c},j} + a_{\mathbf{c}})$$

The last equality follows from the fact that $\sum_j \lambda_{\mathbf{c},j} q_{\mathbf{c},j}$ is a convex combination and hence, $\sum_j \lambda_{\mathbf{c},j} = 1$. Therefore, in minimizing $\|x_{\mathbf{c}}\|^2$ with respect to the variables in the block $y_{\mathbf{c}}$, we are looking for a convex combination of the points of the form $p = q_{\mathbf{c},j} + a_{\mathbf{c}}$ where each $q_{\mathbf{c},j} \in B(f_{\mathbf{c}})$ is an extreme base of $B(f_{\mathbf{c}})$ and $a_{\mathbf{c}}$ is a constant. Hence, we can use MNP algorithm for carrying out this minimization. The key question is how to efficiently carry out the step 2 of the algorithm. It is easy to see that:

$$\operatorname*{arg\,min}_{(q+a_{\mathbf{c}}):q \in B(f_{\mathbf{c}})} \langle \hat{x}, (q + a_{\mathbf{c}}) \rangle = a_{\mathbf{c}} + \operatorname*{arg\,min}_{q \in B(f_{\mathbf{c}})} \langle \hat{x}, q \rangle. \tag{3.3}$$

Equation (3.3) suggests we use Edmond's algorithm (Procedure 1) to select the extreme base $\hat{q}$ which minimizes the RHS above, followed by a translation $\hat{q}$ with $a_{\mathbf{c}}$. Other steps of the Wolfe's algorithm can remain as is.

Procedure 5 Min Norm Point Algorithm for Sum of Submodular Functions

Input: $\{f_{\mathbf{c}}\}$ such that $f = \sum f_{\mathbf{c}}$
Output: $x = \arg\min \|x\|^2$ subject to $x \in B(f)$.

```
# Initialize
for all (c ∈ 𝒞) do
    q_c ← Take any extreme base of f_c;
    S_c := {q_c};
    y_c := q_c
end for
x := Σ_c y_c;

# Perform Block Coordinate Descent with blocks specified by Cliques
while (‖x‖ decreases by more than δ) do
    for all (c ∈ 𝒞) do
        MinNormOverAClique(f_c,S_c,x_c,y_c)
    end for
end while
```

Procedure 5 describes our algorithm for finding the min norm point. This in turn requires Procedure 6 which minimizes $\|x_{\mathbf{c}}\|^2$ with respect to the variables in the block $y_{\mathbf{c}}$. We ensure that $y_{\mathbf{c}} \in B(f_{\mathbf{c}})$ during each minimization. Every time $\|x_{\mathbf{c}}\|^2$ is minimized, the set of extreme bases $S_{\mathbf{c}}$ involved in the minimiation (see Procedure 6) is stored and used to initialize the future iteration minimizing $x_{\mathbf{c}}$. This is important so that we do not waste computations done during the previous minimization steps. We note that the coordinate descent updates for non-overlapping blocks (i.e., $y_{\mathbf{c}}$'s) can be done in parallel and exploiting this is a direction for future work.

3.3 Convergence

The SOSMNP algorithm minimizes $\|x_{\mathbf{c}}\|^2$ using MNP over all the cliques $\mathbf{c} \in \mathscr{C}$ cyclically. This norm minimization step can be viewed as MNP minimizing $f'_{\mathbf{c}}(S) = f_{\mathbf{c}}(S) + a_{\mathbf{c}}(S), \forall S \subseteq \mathbf{c}$ where $a_{\mathbf{c}} = x_c - y_c$ (and $a_{\mathbf{c}}(S) = \sum_{e \in S} a_{\mathbf{c}}(e)$) is a denoting the contribution of the other cliques which remains constant while running MNP over this clique/block. It is easy to show the following result:

Lemma 3.3. *Let $q_{\mathbf{c}}$ be a extreme base vector in $B(f_{\mathbf{c}})$ corresponding to an ordering $\prec_{\mathbf{c}}$. Then the vector $q_{\mathbf{c}} + a_{\mathbf{c}}$ is an extreme base of $B(f'_{\mathbf{c}})$ corresponding to the same ordering $\prec_{\mathbf{c}}$.*

Proof. We can calculate the elements in extreme base vector ($q'_{\mathbf{c}} \in B(f')$) corre-

Procedure 6 MinNormOverAClique

Input: Clique function: $f_{\mathbf{c}}$
Input: Set of extreme bases selected in last iteration: $S_{\mathbf{c}}$
Input: Restriction of current solution vector x on $\mathbf{c}$: $x_{\mathbf{c}}$
Input: Current clique vector: $y_{\mathbf{c}}$
Output: Clique vector $y^*_{\mathbf{c}} \in B(f_c)$ minimizing $\|x_{\mathbf{c}}\|^2$
Output: Updated set $S^*_{\mathbf{c}}$ of extreme bases

1: **while** (TRUE) **do**
2: Find new translation $a_{\mathbf{c}} := x_{\mathbf{c}} - y_{\mathbf{c}}$;
3: Find extreme base $\hat{q}_{\mathbf{c}} := \arg\min_{q_{\mathbf{c}} \in B_{f_{\mathbf{c}}}} \langle x_{\mathbf{c}}, q_{\mathbf{c}} \rangle$ using Edmond's algorithm as given in Procedure (1)
4: Find translated extreme base $\hat{p}_{\mathbf{c}} = \hat{q}_{\mathbf{c}} + a_{\mathbf{c}}$.
5: **if** $(\|x_{\mathbf{c}}\|^2 \leq \langle x_{\mathbf{c}}, \hat{p} \rangle + \epsilon)$ **then**
6: break;
7: **end if**
8: $S_{\mathbf{c}} := S_{\mathbf{c}} \cup \hat{q}_{\mathbf{c}}$;
9: $P_{\mathbf{c}} = \{\hat{q_{\mathbf{c}}} + a_{\mathbf{c}} | q_{\mathbf{c}} \in S_{\mathbf{c}}\}$
10: Find $x_{\mathbf{c}}$ in affine hull of $P_{\mathbf{c}}$.
11: If $x_{\mathbf{c}}$ is not in convex hull $P_{\mathbf{c}}$, translate to nearest point in convex hull and update $S_{\mathbf{c}}$.
12: **end while**

sponding to ordering $\prec_{\mathbf{c}}$ by Edmond's Greedy Algorithm,

$$\begin{aligned}
q'(e) &= f'(S_e \cup e) - f'(S_e), \qquad (S_e \text{ is the set of elements before } e \in \mathbf{c} \text{ in } \prec_{\mathbf{c}}.)\\
&= f(S_e \cup e) + a_{\mathbf{c}}(S_e \cup e) - (f(S_e) + a_{\mathbf{c}}(S_e)),\\
&= f(S_e \cup e) + a_{\mathbf{c}}(S_e) + a_{\mathbf{c}}(e) - (f(S_e) + a_{\mathbf{c}}(S_e)),\\
&\qquad\qquad (a_{\mathbf{c}} \text{ can be seen as a modular function.})\\
&= f(S_e \cup e) - f(S_e) + a_{\mathbf{c}}(e),\\
&= q_{\mathbf{c}}(e) + a_{\mathbf{c}}(e). \qquad (\text{By Edmond's Greedy Algorithm.})
\end{aligned}$$

Hence, $q'_{\mathbf{c}} = q_{\mathbf{c}} + a_{\mathbf{c}}$. □

It is easy to see that:

$$\begin{aligned}
x_{\mathbf{c}} = y_{\mathbf{c}} + a_{\mathbf{c}} &= \sum_i \lambda_i q_{\mathbf{c}} + a_{\mathbf{c}} \qquad (\text{where } \textstyle\sum_i \lambda_i = 1, \text{ and } \lambda_i > 0)\\
&= \sum_i \lambda_i q_{\mathbf{c}} + \sum_i \lambda_i a_{\mathbf{c}} \qquad (\text{Since } \textstyle\sum_i \lambda_i = 1)\\
&= \sum_i \lambda_i (q_{\mathbf{c}} + a_{\mathbf{c}}) = \sum_i \lambda_i q'_{\mathbf{c}}
\end{aligned}$$

Hence, $x_{\mathbf{c}}$ is a base vector of f'. Therefore, minimizing the minimum norm over a block, the way SoSMNP does it, can be seen as minimizing the norm of $x_{\mathbf{c}}$: the restriction of x over the elements of clique $\mathbf{c}$ (and not $y_{\mathbf{c}}$). Let us suppose, we have reached a situation where the SoSMNP performs minimization over all blocks (cliques), and no change was observed in any of the blocks. The following lemma establishes the relationship between the extreme base of $f_{\mathbf{c}}$, and the one corresponding to f.

Lemma 3.4. *Let* $q_{\mathbf{c}} = \arg\min_{q \in B(f_{\mathbf{c}})} x_{\mathbf{c}}^T q,\ \forall \mathbf{c} \in \mathscr{C}$. *Then* $b = \sum_{\mathbf{c} \in \mathscr{C}} q_{\mathbf{c}}$ *also satisfies* $b = \arg\min_{b \in B(f)} x^T b$.

Proof. In the SoSMNP algorithm, the extreme base $q_{\mathbf{c}}$ is generated using Edmond's Greedy Algorithm [85] on the order $\prec_{\mathbf{c}}$ of the indices obtained by sorting the elements of $x_{\mathbf{c}}$ in the increasing order. We represent the extreme base so obtained by $q_{\mathbf{c}}^{\prec_{\mathbf{c}}}$. The SoSMNP algorithm for a block terminates when $x_{\mathbf{c}}^T x_{\mathbf{c}} = x_{\mathbf{c}}^T (q_{\mathbf{c}}^{\prec_{\mathbf{c}}} + a_{\mathbf{c}})$

Consider the termination situation of SoSMNP for the overall problem (comprising of all the cliques). In such a case the algorithms tries to minimize for all the blocks/cliques and no change is found on any of the cliques. Therefore, termination condition of each block is met, and $q_{\mathbf{c}}^{\prec_{\mathbf{c}}} = \arg\min_{q \in B(f_{\mathbf{c}})} x_{\mathbf{c}}^T q$.

Let $\prec_f$ be the ordering of elements of x in the increasing order. It is easy to see that the ordering over x and $x_{\mathbf{c}}$ will be consistent with each other, in the sense that $x(e_1) \prec_f x(e_2) \Rightarrow x_{\mathbf{c}}(e_1) \prec_{\mathbf{c}} x_{\mathbf{c}}(e_2)$.

Let us create an extreme base of f, corresponding to the ordering $\prec_f$, and denote as $b^{\prec_f}$. Since $\prec_f$ denotes the ordering over elements of x, therefore, from Edmond's algorithm, we have: $b^{\prec_f} = \arg\min_{b \in B(f)} x^T b$. Further, we also have:

$$
\begin{aligned}
b^{\prec_f}(e) &= f(S_e \cup e) - f(S_e), \\
&\qquad \text{(As per Edmond's algorithm. } S_e \text{ is the set of elements before } e \text{ in } \prec_f) \\
&= \sum_{\mathbf{c} \in \mathscr{C}} f_{\mathbf{c}}(S_e \cup e \cap \mathbf{c}) - f(S_e \cap \mathbf{c}), \qquad \text{(Since } f(S) = \textstyle\sum_{\mathbf{c} \in \mathscr{C}} f_{\mathbf{c}}(S \cap \mathbf{c})\text{)} \\
&= \sum_{\mathbf{c} \in \mathscr{C}} q_{\mathbf{c}}^{\prec_{\mathbf{c}}}(e \cap \mathbf{c}). \qquad \text{(Since } \prec_c \text{ is the restriction of } \prec\text{)}
\end{aligned}
$$

Since above holds for all the elements $e \in \mathscr{V}$, therefore:

$$
b^{\prec_f} = \sum_{\mathbf{c} \in \mathscr{C}} q_{\mathbf{c}}^{\prec_{\mathbf{c}}}.
$$

Hence, we have proved both the properties of $b^{\prec_f}$ □

We can now give the convergence proof of the SoSMNP with the following lemma:

Lemma 3.5. *If in a complete cycle of SoSMNP over all the cliques, we can not improve the norm $x_{\mathbf{c}}$ for any $\mathbf{c}$, then we have $x \in B(f)$ such that $\|x\|^2 = x^T x = x^T b$, where $b = \arg\min_{b \in B(f)} x^T b$.*

Proof. Recall that for a clique $\mathbf{c}$, SoSMNP can be seen as minimizing the norm of $x_{\mathbf{c}}$ which is a base vector of $f'_{\mathbf{c}} = f_{\mathbf{c}} + a_{\mathbf{c}}$. Further $q'_{\mathbf{c}} = q_{\mathbf{c}} + a_{\mathbf{c}}$ is an extreme base of f'. Therefore, from the termination of basic MNP algorithm, the following must hold:

$$x_{\mathbf{c}}^T x_{\mathbf{c}} = x_{\mathbf{c}}^T (q_{\mathbf{c}} + a_{\mathbf{c}}). \qquad (q_{\mathbf{c}} = \arg\min_{q \in B(f_{\mathbf{c}})} x_{\mathbf{c}}^T q)$$

Summing over all the cliques we get

$$\begin{aligned} \sum_{\mathbf{c} \in \mathscr{C}} x_{\mathbf{c}}^T x_{\mathbf{c}} &= \sum_{\mathbf{c} \in \mathscr{C}} x_{\mathbf{c}}^T (q_{\mathbf{c}} + a_{\mathbf{c}}), \\ \sum_{\mathbf{c} \in \mathscr{C}} x_{\mathbf{c}}^T (y_{\mathbf{c}} + a_{\mathbf{c}}) &= \sum_{\mathbf{c} \in \mathscr{C}} x_{\mathbf{c}}^T (q_{\mathbf{c}} + a_{\mathbf{c}}), \\ \sum_{\mathbf{c} \in \mathscr{C}} x_{\mathbf{c}}^T y_{\mathbf{c}} &= \sum_{\mathbf{c} \in \mathscr{C}} x_{\mathbf{c}}^T q_{\mathbf{c}}. \qquad (\textstyle\sum_{\mathbf{c} \in \mathscr{C}} x_{\mathbf{c}}^T a_{\mathbf{c}} \text{ cancels out}) \end{aligned}$$

Since vector $y_{\mathbf{c}}$ and $q_{\mathbf{c}}$ have non-zero values only for elements in $\mathbf{c}$. Therefore we can write $x_{\mathbf{c}}^T y_{\mathbf{c}} = x^T y_{\mathbf{c}}$ and $x_{\mathbf{c}}^T q_{\mathbf{c}} = x^T q_{\mathbf{c}}$. Substituting the values, we get:

$$\begin{aligned} \sum_{\mathbf{c} \in \mathscr{C}} x^T y_{\mathbf{c}} &= \sum_{\mathbf{c} \in \mathscr{C}} x^T q_{\mathbf{c}}, \\ x^T \sum_{\mathbf{c} \in \mathscr{C}} y_{\mathbf{c}} &= x^T \sum_{\mathbf{c} \in \mathscr{C}} q_{\mathbf{c}} \\ x^T x &= x^T b \qquad (\text{where } b = \arg\min_{b \in B(f)} x^T b, \text{ by Lemma 3.4}) \end{aligned}$$

The equation above is the termination condition of basic MNP when run over the overall function f [20]. Therefore, the lemma essentially proves that the basic MNP terminating with optimal solutions for all cliques/blocks implies that the the base vector obtained by summing up the base vectors of all the cliques/blocks is the optimal solution for the overall objective function. □

When MNP algorithm is run in the block co-ordinate descent mode it is easy to show that any decrease in the $\|x_c\|^2$ of a clique decreases the over all $\|x\|^2$ by the same amount because $\|x_{\mathscr{C}'}\|$ is untouched when optimizing for $\mathbf{c}$. Since at each cycle there is at least one clique for which $\|x_c\|^2$ decreases, we can say that $\|x\|^2$ decreases monotonically at each cycle. Note that Theorem 4 of [20] gives us a lower bound on the improvement in every MNP iteration. It follows that MNP algorithm running in block co-ordinate descent mode will have a provable rate of convergence.

For the sake of completeness we will also like to point out that the optimal solution obtained when MNP is run globally also corresponds to the individual blocks having reached their local optimal.

3.4 Experiments

In this section, we report our experiments comparing our proposed SoS MinNorm algorithm with the state of the art. The implementation of our algorithm in C++, is available at http://www.iiitd.edu.in/~ishants/sosminnorm.html. All the tests have been performed on a standard workstation with 3.0 GHz CPU and 8 GB RAM running Ubuntu 14.04.

3.4.1 Experimental Setup

Using the terminology in the standard vision literature, we refer to our problems as 0/1 labeling problems. Each pixel can be assigned a value of 0 or 1 corresponding to exclusion and inclusion in a set, respectively. The total labeling cost is defined as the sum of labeling costs over pixels appearing in each clique. Labeling costs for a clique are referred to as clique potentials which directly correspond to the functions $f_{\mathbf{c}}$ in the sum we want to minimize. Therefore, if the clique potentials are submodular, then the problem of finding the minimum cost labeling configuration can be expressed as an SoS minimization problem.

We compare with the following algorithms:

1. Standard MinNorm point (MinNorm) which does not use sum of submodular property. We did not find any public implementation of the MinNorm and have used or own implementation in C++ for comparison.

2. Generic Cuts (GC) [7]: a flow based approach exploiting sum of submodular structure. We have used the implementation available on authors' website.

3. Jegelka et al. [45] approach using decomposition strategy. This algorithm is restricted to a subclass of submodular functions for which $\min_{S \subseteq V} f(S) - a(S)$ can be computed efficiently, where $a \in R^V$ is a constant vector. The assumption is exploited in an inner loop of their algorithm. Since we would like to experiment with a general class of submodular functions, we have replaced this step dealing with specialized functions with a more general QP solver routine from cvxopt library [23].

4. For object detection experiments we have used the TextonBoost [88] approach to generate per pixel confidence. To generate simple baseline we use confidence directly for prediction (without any MRF structure), which we refer to as output from TextonBoost.

5. We also compare with pairwise cliques formulation using QPBO [57, 70] for the inference.

	Problem Size				
	16	36	64	100	144
SoS-MinNorm	0.000	0.001	0.003	0.005	0.006
MinNorm [32]	0.002	0.023	0.073	0.221	0.776

Table 3.1: Comparing performance of SoS-MinNorm with the vanilla Min Norm Point algorithm [32]. We keep clique size = 2×2 with edge based costs for this experiment. The numbers denote the time taken in seconds. The proposed algorithm clearly outperforms the vanilla appproach.

The proposed algorithm as well as the compared algorithm [7, 45] give optimal inference for submodular functions tested in this paper. The focus of the experiments in this paper is therefore on scalability and efficiency.

Fix et al. [27] have suggested an algorithm to learn submodular functions for the inference problems they have considered. In our experiments the approach fails to scale beyond clique size 9. Therefore we have used simple hand tuned clique potentials. It may be noted that the clique potentials are not the focus of this paper and are merely used as a proxy for real world potentials normally seen in computer vision problems. One of the future directions of our research is to use our algorithm to learn submodular clique potentials for large cliques.

Given a clique C of size $k_w \times k_h$, we consider the following potentials:

- **Edge Based Costs:** As described by Arora et al. [7], we generate a submodular potential over a 2×2 clique $\mathbf{c}$ by defining $f_{\mathbf{c}}(S \cap \mathbf{c})$ as the square root of the number of edges where an edge is a pair of neighboring vertices (top,down,left and right nodes) assigned different labels. In order to generate functions of size greater than 4, we translate the template in a non-overlapping fashion and add the costs from various templates.
- **Count Based Costs:** These potentials are inspired by the ones used by Stobbe and Krause [90]. We define a submodular potential over $\mathbf{c}$ as $f_{\mathbf{c}}(S) = |S \cap \mathbf{c}||\mathbf{c} \setminus S|$. For a fixed $\mathbf{c}$, $f_{\mathbf{c}}(S)$ is a concave function of the number of pixels in $\mathbf{c}$ labeled 1. Uniform labeling is favored while equal number of pixels with $1's$ and $0's$ are penalized the most.

As is the standard for several formulations, we also incorporated additional per pixel costs, called the unary potentials. The overall function can be defined as follows:

$$f(S) = \sum_{v_i \in \mathscr{V}} w_i \mathscr{I}[v_i \in S] + \alpha \sum_{\mathbf{c} \in \mathscr{C}} f_{\mathbf{c}}(\mathbf{c} \cap S).$$

Unary potentials can be seen as encoding pixel-wise evidence, whereas clique potentials represent labeling priors. Note that unary costs can equivalently be

	Clique Size			
	2×2	4×2	4×4	4×6
SoS-MinNorm	0.01	0.01	0.02	0.02
GC [7]	0.00	0.26	731.67	DNR
SoS-Jegelka [45]	861.07	12217.37	TO	TO

Table 3.2: Comparing SoS-MinNorm with GC [7] and SoS-Jegelka [45] on varying cliques using edge based potential. The problem size was fixed at 400. The numbers denote the time taken in seconds. DNR shows that the algorithm crashed on the test. TO denotes a time out for decomposition approach after 4 hours of running. SoS-MinNorm significantly outperforms both the existing algorithms, none of which can scale beyond clique size 16.

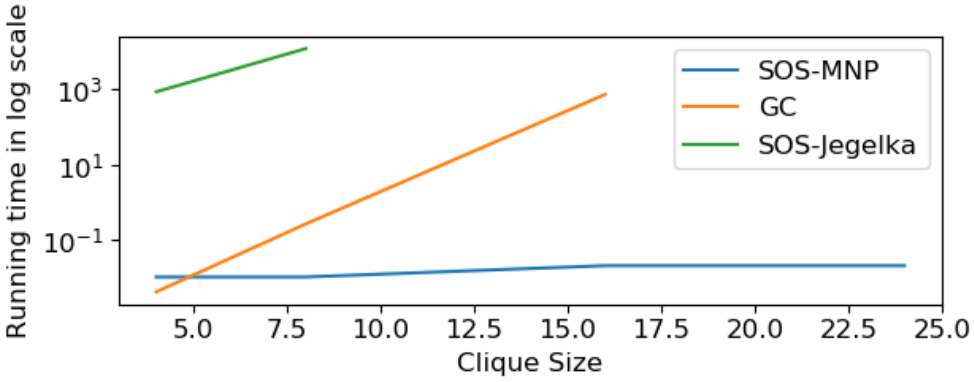

Figure 3.1: Log plot for running time comparison between the algorithms:SOS-MNP, GC and SOS-Jegelka.

absorbed in the cost of a clique, and hence, do not lead to any additional complications in the model.

3.4.2 Experiments on Synthetic Problems

We perform our evaluations in two parts. In the first part, we experiment with synthetically generated submodular potentials. Our synthetic potentials are inspired by those used in real world vision applications. Synthetic data allows us to test the scalability of our approach with varying clique as well image sizes. Synthetic problems are generated over grid graphs of size $n = n_w \times n_h$. Cliques represent sub-grids of size $k = k_w \times k_h$. We vary n and k in our experiments. Unary costs have been generated randomly.

Table 3.1 compares SoS-MinNorm with vanilla Min Norm Point algorithm [32].

	Problem Size			
	100	400	900	1600
SoS-MinNorm	0.01	0.01	0.04	0.09
GC [7]	36	467	2744	2868

Table 3.3: Comparing SoS-MinNorm with GC [7] for varying problem sizes. Clique size is fixed at 16. We use edge based potentials for the experiment. Our approach significantly outperforms both the other approaches at all the problem sizes. SoS-Jegelka [45] had a timeout (time more than 4 hours) for all values.

Clearly, we significantly outperform the vanilla approach by virtue of exploiting the sum of submodular property.

Next, we compare our approach with GC [7] and SoS-Jegelka [45]. Both the algorithms are state of the art and exploit the sum of submodular property of the underlying function. Table 3.2 shows the comparison results. Our proposed algorithm clearly outperforms SoS-Jegelka which did not scale well in our experiments. GC required few hours to solve the problems with clique size 16. Our algorithm could solve these problems in less than a minute. Further, the proposed algorithm can easily scale to problems of clique size 32 whereas GC can not go beyond clique size 16.

We also compared our algorithm with GC and SoS-Jegelka using different problem sizes. We fixed the clique size at $k = 16$ (maximum possible to which GC can scale). The image size was varied from 100 to 1600. Table 3.3 shows the results. Our approach is at least an order of magnitude faster than GC at all problem sizes. SoS-Jegelka timed out (process killed externally after 4 hours) at this clique size for all problem sizes.

3.4.3 Comparison on Real Datasets

The contribution of this paper is essentially algorithmic in nature and our algorithm can be used for any problem formulated as binary MRF-MAP or structured prediction problem with sum of submodular structure. However, we have done some indicative experiments with pixel level object detection and interactive object segmentation problems to show the efficacy of our approach on real datasets.

We experimented with pixel level object detection using the dataset provided by [73]. We generate per pixel confidence using the probabilities generated by TextonBoost [88]. We use these probabilities for setting unary potentials in our formulation. We compare with the formulation using unary cost alone (thresholding) and pairwise cliques approach [75]. We experimented with multiple values for the relative weighing of clique potential for the pairwise approach

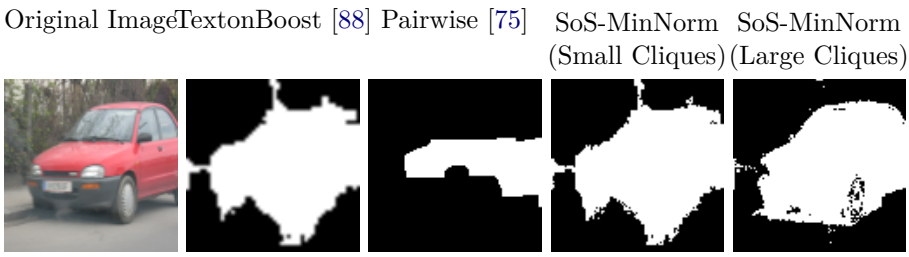

Figure 3.2: Pixel level object detection: We generate per pixel confidence using the probabilities generated by TextonBoost [88]. The second column titled TextonBoost has been generated based upon these confidence alone. The third column, shows the results using cliques of size of 2 only [75]. For generating higher order cliques to be used with our algorithm, we use region growing as suggested by Stobbe and Krause [90]. The fourth column shows the results using our algorithm with region growing restricted to 50 resulting in average clique size of 31. For testing with larger cliques, we allowed region growing until size 300 generating cliques with average size of 230. We used count based cost. The image size is 100×100 and the time for small and large cliques is 0.6 and 53 seconds respectively. The quality of results seems to improve with increasing clique size.

and chose the one which gave the best visual results. For generating higher order cliques to be used with our algorithm, we first grow regions using HSV channels as suggested by Stobbe and Krause [90]. The image was divided into 5×5 grids and each grid intersection was taken as a seed for region growing. To cover any remaining regions, we then generate 50 random seeds and grow regions from them. Any seed appearing in already grown region was ignored. To show the improvement using higher order clique we use cliques of two different sizes. For generating smaller cliques, the region growing was restricted to 50 resulting in average clique size of 31. For larger cliques, region growing was allowed until 300, generating cliques with average size of 230. We use count based cost for this experiment. Figure 3.2 compares the results of TextonBoost, Pairwise Cliques and our algorithm using small and large cliques. Not only, our algorithm can scale to such large sized cliques, the quality of results seems to improve with increasing clique size.

Next we have experimented with interactive object segmentation as used by Jegelka et al. [45]. The setup resembles the method proposed by Rother et al. [82]. From inference perspective, we generate the higher order cliques in the same way as described for object detection. However, the unary costs are now based upon the user interaction. Figure 3.3 shows the result. Similar to object detection, we see an improvement with increasing clique size with our method.

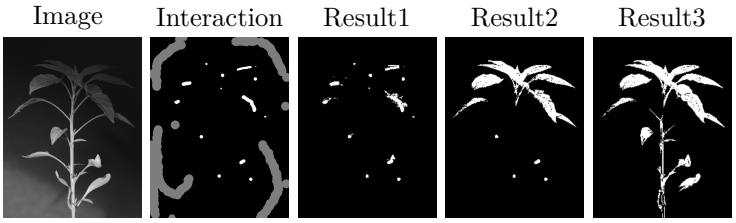

Figure 3.3: Interactive object segmentation: The unary cost for the inference is based upon user interaction, whereas the higher order cliques have been generated using the region growing as is done for the object detection problem. First and second columns show the input image and user inputs respectively, whereas columns third to fifth show the results using our approach with increasing average clique size. The image size is 103×132 and the time for the 3 results shown are 1.15,0.47 and 0.11 seconds respectively. Similar to the object detection problem, our experiments show improved visual quality with increasing clique size.

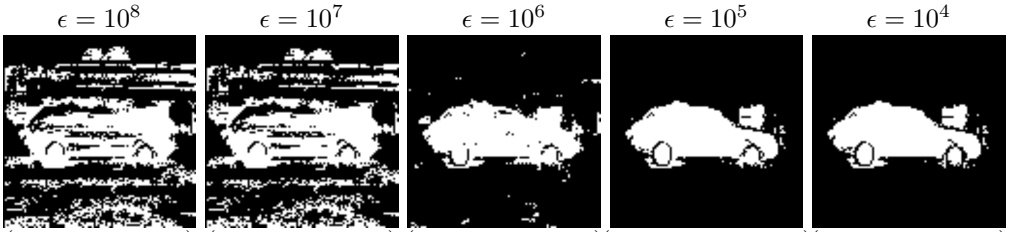

Figure 3.4: Our algorithm can be used for faster approximate inference by changing the value of ϵ in Step 9 Procedure 6. The number below the each figure shows time taken (in seconds), followed by primal and dual values (in thousands). We have used count based clique potential with clique size ~ 250. The approximation strategy should be useful for applications with limited time budget.

3.4.4 Approximation Strategy

Our algorithm is guaranteed to converge to the optimum for all submodular functions. However, the convergence may be slow for large problems. The algorithm can be potentially used as an approximation algorithm also. For optimal inference ϵ in Step 9 Procedure 6 should be a very small value. Increasing the value of ϵ can lead to faster convergence but may cause algorithm to terminate before optimality is reached. We have experimented with various values of ϵ for the pixel level object detection experiment. Figure 3.4 shows the results. As expected the time taken for the inference improves as we increase the value of ϵ. Correspondingly, we observe increasing primal dual gap with increasing values of ϵ. Interestingly, the visual quality of results also degrades gradually. This indicates the possibility of using ϵ as a tunable parameter for controlling quality vs time taken in problems with limited time budget.

3.5 Conclusion

Many problems in computer vision modelled as MRF-MAP labeling problems can be reduced to minimizing a sum of submodular functions. The state of the art algorithms suggested in computer vision scale well with image size but not with clique size. On the other hand algorithms proposed in mathematical optimization community scales well with clique size but not with image size. In this work we have tried to take the best of both the worlds. We suggest a new block coordinate descent algorithm which adapts Min Norm Point algorithm for minimizing a sum of submodular functions. Being based upon MNP algorithm, the algorithm scales well with clique size, whereas by exploiting sum of submodular structure, the algorithm works well with large problem sizes also. In our experiments, we have run it for inference problems with number of nodes running into many thousands and clique size of multiple hundreds. The algorithm achieves state of the art accuracy both in terms of efficiency (time taken for the inference) as well as scalability (with image and clique size). Block coordinate descent framework developed in this work is general. In a sense that different methods which optimize ℓ_2-norm of solution vector can be applied to optimize the objective over different cliques. For different configuration of clique sizes, some inference methods have advantage over others. For example, for large number of small cliques flow based methods work faster than polyhedron based inference method developed in this work. For small number of large cliques flow based methods does not scale but polyhedron based method works well. In the next chapter we show how one can use flow based methods and polyhdron methods for solving MRF-MAP problem with small and large cliques.

Chapter 4

Hybrid Framework for Small and Large Cliques

In the previous chapter we show SoSMNP for doing inference in 2-label MRF-MAP problems with higher order submodular clique potentials. While higher order cliques penalize deviation from the region/patch level statistics (e.g. a prior on the expected object size) the smaller cliques can be used to enforce regularization. Such problems have 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). In principal SoSMNP can be applied to solve this problem but it converges very slowly due to the huge number of small cliques.

One practical inference algorithm which works on the problems with a million of cliques is GC [7]. But this algorithm does not scale on the problems with cliques of size larger than 16. Therefore such problems with large number of small cliques and small number of large cliques are not solvable practically either by flow based algorithms like [7] or submodular polyhedron based algorithm SOSMNP. At this point it is natural to ask if we can apply GC to solve the subproblems over the small cliques and MNP to solve the subproblems over large cliques. Later combine the results obtained from the two frameworks to find the solution for the overall problem. We show that this is indeed possible by showing the mapping between the free variables of both the frameworks. The specific contributions of this work 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.

In the following section we discuss the theoretical results which is the basis for developing the hybrid framework.

4.1 Theoretical results

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 [7], 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 4.1. *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 (3.1) and (3.2) give the conditions for any vector to be in the base polyhedron. By virtue of Eq. (1.13), it is easy to see that the flow vector satisfies Eq. (3.1). 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. [7]), thus satisfying Eq. (3.2) as well. □

Lemma 4.1 serves to indicate that equations (3.1) and (1.13) 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 4.2. *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.*

4.2 Hybrid Algorithms

Lemmas 4.1 and 4.2 enable us to put in place a block coordinate descent strategy consistent with the framework of [87]. We make a coordinate block corresponding to sum of large clique potentials to be solved by a polyhedral method maximizing $x^-(\mathscr{V})$. 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. (1.1) can be rewritten as

$$\mathbf{l}_{\mathscr{V}}{}^* = \arg\min_{\mathbf{l}_{\mathscr{V}}} \sum_{\mathbf{c}\in\mathscr{L}} f_{\mathbf{c}}(\mathbf{l}_{\mathbf{c}}) + \sum_{\mathbf{c}\in\mathscr{S}} f_{\mathbf{c}}(\mathbf{l}_{\mathbf{c}}). \tag{4.1}$$

Here $\mathscr{L}$ and $\mathscr{S}$ are the set of large and small cliques respectively such that $\mathscr{C} = \mathscr{L} \cup \mathscr{S}$. If x is a base vector corresponding to the submodular function as given in Eq. (4.1), then using Lemma 3.2 it can also be written as:

$$x = \sum_{\mathbf{c}\in\mathscr{L}} y_{\mathbf{c}}(\mathbf{l}_{\mathbf{c}}) + \sum_{\mathbf{c}\in\mathscr{S}} y_{\mathbf{c}}(\mathbf{l}_{\mathbf{c}}), \text{ or} \tag{4.2}$$

$$x = x_l + x_s, \tag{4.3}$$

where $y_{\mathbf{c}}(\mathbf{l}_{\mathbf{c}}), \mathbf{c} \in \mathscr{L}$ and $y_{\mathbf{c}}(\mathbf{l}_{\mathbf{c}}), \mathbf{c} \in \mathscr{S}$ 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 $x^-(\mathscr{V})$ by alternately carrying out a block coordinate descent iterations on the blocks corresponding to $\mathscr{L}$ and $\mathscr{S}$. While maximizing $x_l^-(\mathscr{V})$ we keep x_s (and its associated y's) as constant. Any base polyhedron based SFM algorithm which works on the principle of maximizing $x^-(\mathscr{V})$ 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 $x_s^-(\mathscr{V})$ 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 1.3.6 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 4.1.

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 [53] using graph cuts. Our current implementation, however, does not exploit this.

4.3 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 [67, 87]. We give below the reasoning in detail and the changes required to use MNP to handle large cliques.

4.3.1 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 $x^-(\mathscr{V})$ and MNP which minimizes $\|x\|^2$.

Polyhedral methods that maximize $x^-(\mathscr{V})$ can be considered to move from one base vector to another in which the value of $x^-(\mathscr{V})$ increases. The extent of increase is a function of the next base vector that the algorithm chooses. For example, in Schrijver's algorithm [85] 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 $\mathscr{V}$, 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 $x^-(\mathscr{V})$ 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 $x^-(\mathscr{V})$ 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 $x^-(\mathscr{V})$ 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 $x^-(\mathscr{V})$ 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 4.2 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 $x^{-}(\mathscr{V})$ 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.

4.3.2 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 4.3. *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 4.3 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 4.3, 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.

4.3.3 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 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 [87] 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

Procedure 7 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 $x \in B(f)$.
1: Set $y_m = f_m$ and $y_{\mathbf{c}} = 0, \mathbf{c} \in \mathscr{S}$.
2: $\forall \mathbf{c} \in \mathscr{L}$ initialize $y_{\mathbf{c}}$ from a random base vector.
3: **while** Norm of x decreases **do**
4: Call SOSMNP with $y_m + x_s$ as unary cost and f_l as the clique potential;
5: MinNormGC($\mathscr{V}, f_s, y_m + x_l, 0$);
6: **end while**

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 4.2. The above detailing of the initialization process takes into account modular function costs which were subsumed earlier. Algorithms 7 and 8 give the details of proposed algorithm in pseudocode format.

4.3.4 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 SOSMNP algorithm presented in Section 3.3 and polynomial time convergence of GC [7]. 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. Also note that every time a GC block is handled the ℓ_2 norm can not increase for that block. Therefore, termination and convergence of the HybridMNGC algorithm follows along the same lines as that for the SOSMNP algorithm.

Experiments reported in the next section indicate that HybridMNGC is significantly faster than SOSMNP which, experiments reported earlier [87] indicate, is the algorithm of choice for polyhedral based algorithms working with large cliques.

Procedure 8 MinNormGC($\mathscr{V}, f_s, U, b$)

Input: $\mathscr{V}$: Set of Nodes/Elements for which norm is to computed.
Input: U: Function specifying unary/modular cost for each pixel in $\mathscr{V}$.
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 ∀p ∈ 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 ∈ 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 ∈ T and sink
  node t;
  Denote average of U_T by b_T;
  MinNormGC(T, f_s, U_T, b_T);
end if
```

4.4 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 4.4.1. Quality comparison has been done on the pixelwise object segmentation problem where the images for the demonstration are taken from Pascal VOC dataset [24]. Outputs of HybridM-

	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 4.1: 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 4.2: Similar comparison as in Table 4.1 with the proviso that clique potentials are edge based. All running time are in seconds.

NGC are compared with that of SegNet [10] (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 [87], we consider edge based clique potentials based on counting the 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.

4.4.1 Performance Comparison

Tables 4.1 and 4.2 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. We take SOSMNP as the baseline because it already outperforms SOS-Jegelka [45]. 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 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

	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 4.3: 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 4.4: 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.

trends where performance decreases with increase in clique size. Similar trend is observed with both count as well as edge based potentials.

Tables 4.3 and 4.4 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.

4.4.2 Comparing Object Segmentation Quality

We show some indicative results in Figures 4.2 and 4.3 for pixel wise object segmentation. Each image has a single object consistent with our binary labeling formulation. In the images in Figure 4.3 Gaussian noise with zero mean and intensity dependent variance has been added. We use Segnet [10] output in two ways. First as the basis for comparison of deep network based object segmentation, and second as a source for generating unary potentials (priors) for individual pixel labels. Learning such priors for use in MRF-MAP optimisation is an area of research in itself. Use of machine learning techniques has been advocated [45, 90, 104]. Whatever else that segnet does it does make available the priors. The important thing to notice is that our algorithm improves the quality of inference and does so better than other existing algorithms with which comparison has been shown. 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

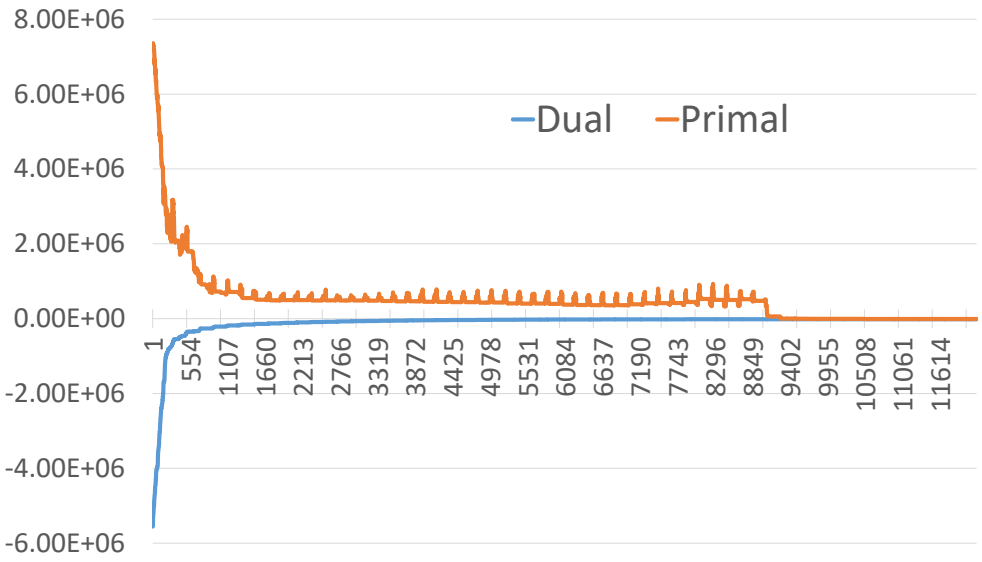

Figure 4.1: Energy as a function of the number of iterations of the HybridMNGC algorithm. Primal is the function value for the set defined by the negative elements of the current base vector.

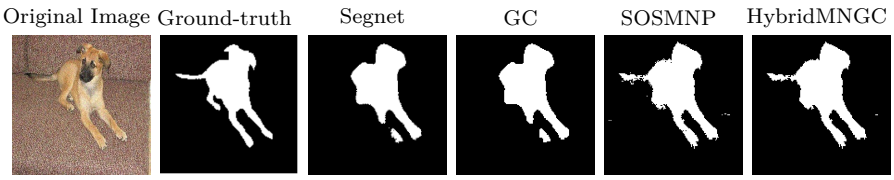

Figure 4.2: Object Segmentation results from various methods using input image with no noise.

large cliques. The pairwise cliques span the image in 4-connected neighborhood style. Big cliques are region grown as in [90]. 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 [20]. We run these algorithms with ϵ as 100. This condition seems to be arrived, on the average, after about 12000 iterations. Experimental evidence indicates Figure 4.1 that the primal labeling stabilizes much before the termination condition is reached. Figure 4.3 has the comparative output images. Note the significant improvement in HybridMNGC output in comparison to that of GC and SOSMNP.

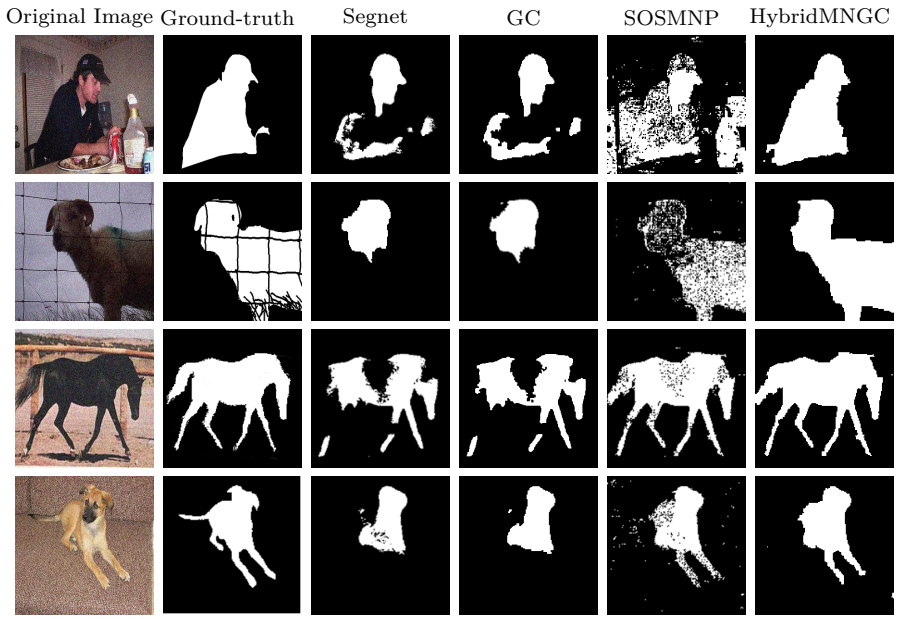

Figure 4.3: Object Segmentation results from various method using input images with Gaussian noise.

Figure 4.2 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.

4.5 Conclusions

Our objective in this work has been to establish the efficacy of combining algorithms like Generic Cuts [7] and SOSMNP [87] 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. We have tested with only Gaussian noise to validate our algorithm's robustness to noise. A more thorough investigation experimenting with different type of noises and potentials is recommended for future research. In

the next chapter we show that multi-label problem can be converted into a 2-label problem which has both small and large cliques, on which our proposed HybridMNGC algorithm can be applied.

Chapter 5

Inference Algorithm for Multi-Label Problems

In the previous two chapters we have looked at the inference methods for 2-label MRF-MAP problem. Binary or 2-label MRFs are limited in terms of capturing the semantic structure of the problem. Often many computer vision problems like stereo, multi object segmentation requires each pixel to be assigned a label from a set of more than 2 labels. Such problems can be modeled as multi-label MRF-MAP inference problems.

Traditionally, the technique of choice for working with multi-label problems has been using the α-expansion [18] heuristic. α-expansion involves cycling through all the labels and making an expansion move with the current label α. The expansion move consists of minimizing the MRF potential with respect to the current label (in effect solving a two label problem). α-expansion with higher order cliques involves an additional step of using reduction to transform a higher order problem to a first order problem. α-expansion converges to local minimum even with submodular potentials.

α-expansion is an example of an "indirect" method that operates in the 2-label solution space without formally mapping a multi-label problem to an equivalent 2-label problem. As explained earlier, one approach to handle multi-label potentials is to use encoding [8, 40] to convert a multi-label problem to an equivalent 2-label problem while preserving submodularity. The encoding expands the size of the solution space from m^k to 2^{mk} [8]. Note that only m^k of the 2^{mk} binary configurations correspond to the original m^k labeling configurations and the rest are *invalid* in the problem context. If potentials for the invalid state are kept very large, with those for the valid states, same as in the original multi-label version, the minimum is always among the valid states. In the flow based algorithm given in [8], the invalid states problem is avoided by observing that residual capacity calculation step in the algorithm requires working with valid states only. However, flow based algorithms do not scale well [7], and have been shown to work only up to clique size 4 for 4-label MRF-MAP problems. The focus of this work is on developing optimal inference algorithm for multi-label, submodular,

higher order MRF-MAP problems.

In principle, this encoding allows one to leverage algorithms developed for 2-label problems in previous two chapters and perform optimal inference for the converted problems as well. However there are some practical challenges due to very large values at invalid states, in practice the SoSMNP algorithm suffers due to the floating point issues in the matrix inversion step of MNP algorithm. In addition to that the expansion in the state space made after the transformation would make the convergence of the SoSMNP very slow. We handle both of the problems by showing that the convex combination of the exponential number of extreme bases computed over the invalid states can be represented as the linear combination of the linear number of basis vectors. We use this result and present an efficient as well as numerically stable inference algorithm for multi-label MRF-MAP problems. The proposed algorithm raises the bar significantly in that using it we can handle multi-label MRF-MAP problems with 16 labels, and clique size upto 100.

At this stage we would like to contrast out mapping technique with that of [94], which has exploited the linear relationship between a tree and order in labels, to map multi-label submodular functions to the more general class of tree based $L^\natural$-convex functions. However, these algorithms have high degree polynomial time complexity (based on [44, 71, 74]), limiting them to be of theoretical interest only. Our focus on the other hand is to extend the frontiers of practical optimal algorithms.

In the following section for enhanced readability we discuss about the transformation of the multi-label potentials into 2-label potentials already present in Chapter 2.

5.1 Converting Multi-label Problem to 2-Label

We first summarize the transformation to convert a multi-label to a 2-label problem as suggested in [8, 40]. Consider an *unordered* set of pixels $\mathscr{P} = \{p_1, \ldots, p_i, \ldots, p_n\}$, and an *ordered* set of labels $\mathscr{L} = \{1, \ldots, m\}$. To save the notation clutter, whenever obvious, we denote a pixel simply using variables p, q without the subscript index.

Definition 5.1 (Binary Encoding)**.** The encoding $E : \mathscr{L} \to \mathbb{B}^m$ maps a label $i \in \mathscr{L}$ to a m dimensional binary vector such that its first $m - i$ elements are 0 and the remaining elements are 1.

For example, $E(1) = (0, \ldots, 0, 0, 1)$, and $E(2) = (0, \ldots, 0, 1, 1)$. Let us denote the encoded label vector corresponding to a pixel p_i as $\gamma_i = (p_i^1, \ldots, p_i^m), p_i^j \in \{0, 1\}$. We denote by $\Gamma \in \mathbb{B}^{mn}$, the vector obtained by concatenating all encoded vectors $\gamma : \Gamma = (\gamma_1, \ldots, \gamma_i, \ldots, \gamma_n)$. The vector Γ represents encoding of labeling

configuration over all the pixels. We also define a *universal set* containing all elements of $\Gamma : \mathscr{V} = \{p_1^1, \ldots, p_1^m, \ldots, p_n^1, \ldots, p_n^m\}$.

Definition 5.2 (Universal Ordering)**.** Assuming an arbitrary ordering among the pixels, the universal ordering, defines a total ordering of the elements p_i^j, where $i \in \mathbb{Z}_{1:n}, j \in \mathbb{Z}_{1:m}$:

$$\prec_0 : p_1^1 \prec \cdots \prec p_1^m \prec \cdots \prec p_n^1 \cdots \prec p_n^m.$$

We denote by $S \subseteq \mathscr{V}$, called *state*, set of all the elements, p_i^j of Γ labeled as 1. Note that there are 2^{mn} possible states, however only m^n of them correspond to valid Γ vector obtained by encoding labeling configurations over the pixels. We call such states as *valid states*. If label of a pixel p_i is denoted as $l_i \in \mathscr{L}$, a valid state may be represented as: $S = \{E(l_1), \ldots, E(l_i), \ldots, E(l_n)\}$. Similarly $S_p = \{E(l_p)\}$ includes elements corresponding to pixel p.

Definition 5.3 (Valid Ordering/Extreme Base)**.** An ordering $\prec$ is called a valid ordering, if for any $p_i^j, p_i^k \in \mathscr{V}$, $j > k \Rightarrow p_i^j \prec p_i^k$. An extreme base $b^{\prec}$ is called a valid extreme base, if it corresponds to a valid ordering.

The states, orderings or extreme-bases which are not valid are called *invalid*. We denote the set of all valid states by Z.

5.1.1 Binary Submodular Extension Function

Definition 5.4 (Covering State, Minimal Covering State)**.** For an arbitrary state, S, a valid state, $\hat{S} \in Z$, is called covering if $S \subseteq \hat{S}$. There may be multiple covering states corresponding to a S. The one with the smallest cardinality among them is referred to as the minimal covering state, and is denoted by $\overline{S}$. There is a unique minimal covering state corresponding to any S. For a valid state $S = \overline{S}$.

Let F be the original multi-label submodular function defined over the pixels $\mathscr{P}$. We refer the readers to [8] for definition of multi-label higher order submodular functions. We encode the multi-label function to a submodular pseudo-Boolean function f defined over set $\mathscr{V}$ of size mn as follows:

Definition 5.5 (The Extended Binary Set Function)**.**

$$f(S) = \begin{cases} F(\ldots, l_i, \ldots), & \text{if } S = \{\ldots, E(l_i), \ldots\} \\ f(\overline{S}) + (|\overline{S}| - |S|)L & \text{otherwise} \end{cases}$$

Here $l_i \in \mathscr{L}$ is label of pixel p_i, and $L \gg M = [\max_{S \in Z} f(S) - \min_{S \in Z} f(S)]$.

It is easy to see that $f(S)$ can also be defined as follows:

Definition 5.6 (The Extended Binary Set Function: Alternate Definition)**.**

$$f(S) = f(\overline{S}) + \sum_{p \in \mathscr{P}} (|\overline{S}_p| - |S_p|)L,$$

where $\overline{S}_p \subset \overline{S}$, and $S_p \subset S$ are the subsets containing elements corresponding to pixel p in $\overline{S}$ and S respectively.

Theorem 5.7. *The extended binary set function f as given by Definition 5.5 is submodular, and* $\min f(\cdot) = \min F(\cdot)$.

Proof. Recall that we define the extended binary submodular function for the valid states as equal to the original multi-label function and for the invalid states as the following:

$$f(S) = f(\overline{S}) + (|\overline{S}| - |S|)L.$$

Here $\overline{S}$ is the minimum covering state for an invalid state, S, which is defined as the smallest cardinality valid state, $\overline{S} \in Z$, such that $S \subset \overline{S}$. For a valid state $S = \overline{S}$.

Let us factorize $f(S) = g(S) + h(S)$, where $h(S) = f(\overline{S}) + |\overline{S}|L$, and $g(S) = -|S|L$. Since, g is modular, it is sufficient to show that h is submodular. We will need the following result to prove the Theorem.

Lemma 5.8. *For sets X, Y, and $(X \cap Y) \subseteq \mathscr{V}$ and their minimum covering states $\overline{X}, \overline{Y}$, and $\overline{X \cap Y}$ respectively:*

$$f(\overline{X \cap Y}) \leq f(\overline{X} \cap \overline{Y}))$$

Proof. Recall that for any valid state S, $\overline{S} = S$. Consider, two valid states $A, B \subseteq V$ with $A \subseteq B$. It is easy to see that:

$$h(B) - h(A) = L(|B| - |A|) + f(B) - f(A) \geq 0. \tag{5.1}$$

Since, $A \subseteq B$, therefore, $|B| - |A| \geq 0$. Also $L \gg f(B) - f(A)$ by definition. Therefore, $h(A) \leq h(B)$. Further, it has been shown in section 6 of [85] that for two $X, Y \in \mathscr{V}$, $\overline{X \cup Y} = (\overline{X} \cup \overline{Y})$ and $\overline{X \cap Y} \subseteq (\overline{X} \cap \overline{Y})$ holds. Therefore, using Eq. (5.1), $f(\overline{X \cap Y}) \leq f(\overline{X} \cap \overline{Y}))$. □

We can now give the proof of the theorem as follows. For the valid states, the extended function f, has been shown to be submodular in [8]. Therefore, here, we show only for the cases when S is an invalid state. Now, for two arbitrary (valid or invalid) sets, $X, Y \subseteq \mathscr{V}$

$$
\begin{aligned}
h(X) + h(Y) &= f(\overline{X}) + f(\overline{Y}) + |\overline{X}|L + |\overline{Y}|L \\
&\geq f(\overline{X} \cup \overline{Y}) + f(\overline{X} \cap \overline{Y}) + |\overline{X}|L + |\overline{Y}|L \\
&\qquad \text{(Using submodularity over } \overline{X}\text{, and } \overline{Y}\text{)} \\
&= f(\overline{X} \cup \overline{Y}) + f(\overline{X} \cap \overline{Y}) + |\overline{X} \cup \overline{Y}|L + |\overline{X} \cap \overline{Y}|L \\
&\qquad \text{(Since } |\overline{X}| + |\overline{Y}| = |\overline{X} \cup \overline{Y}| + |\overline{X} \cap \overline{Y}|\text{)} \\
&= f(\overline{X \cup Y}) + |\overline{X \cup Y}|L + f(\overline{X} \cap \overline{Y}) + |\overline{X} \cap \overline{Y}|L \\
&\qquad \text{(Since } \overline{X \cup Y} = (\overline{X} \cup \overline{Y})\text{)} \\
&\geq f(\overline{X \cup Y}) + |\overline{X \cup Y}|L + f(\overline{X \cap Y}) + |\overline{X \cap Y}|L \\
&\qquad \text{(Using Lemma 5.8)} \\
&= h(X \cup Y) + h(X \cap Y).
\end{aligned}
$$

The above shows that h is submodular. It is easy to see that, g, as defined above is modular. Since addition of a modular function and a submodular function is submodular, therefore, $f = g + h$ is submodular. □

Note that any value of $L \gg M = [\max_{S \in Z} f(S) - \min_{S \in Z} f(S)]$, keeps the function, f, submodular. However, as we show later, choosing such a large value of L, makes the contribution of some extreme bases very small causing precision issues in the computation. We also show that including those extreme bases with very small contribution is extremely important for achieving the optimal inference. The major contribution of this paper is in showing that one can perform an efficient inference bypassing L altogether. Therefore, the use of L is merely conceptual in our framework. There is no impact of actual value of L on the algorithm's performance.

5.2 Representing Invalid Extreme Bases

In the discussion that follows, we refer to any scalar as *small* or *finite* if their absolute value is $\ll L$, and *large* or *infinite* if the absolute value is $\propto L$. We write the solution vector x as:

$$x = x_v + x_i = \sum_{b^{\prec_j} \in R} \lambda_j b^{\prec_j} + \sum_{b^{\prec_i} \in Q} \lambda_i b^{\prec_i}. \tag{5.2}$$

R and Q are the sets of valid and invalid extreme bases, and x_v, and x_i, their contribution in x respectively. It is easy to see that, all the elements of x_i must be much smaller than L [1]. We first focus on the relationship between λ and L in the block of invalid extreme bases.

[1] We start the algorithm with a valid extreme base, where the condition is satisfied. In all further iterations the norm of x decreases monotonically, and the condition continues to remain satisfied.

Lemma 5.9. *For any element, e, of an invalid extreme base, $b^{\prec}$: $b^{\prec}(e) = a_e L + b_e$, where $|a_e|, |b_e| \ll L$.*

Proof. Let S_2 be the set of all elements smaller than e as per $\prec$. Let $S_1 = S_2 \cup \{e\}$.

$$\begin{aligned} b^{\prec}(e) &= f(S_1) - f(S_2) && \text{(Definition of extreme base)} \\ &= \Big(f(\overline{S}_1) + (|\overline{S}_1| - |S_1|)L\Big) - \Big(f(\overline{S}_2) + (|\overline{S}_2| - |S_2|)L\Big) && \text{(Definition 5.5)} \\ &= \Big(f(\overline{S}_1) - f(\overline{S}_2)\Big) + \Big(|\overline{S}_1| - |S_1| - |\overline{S}_2| + |S_2|\Big) L \\ &= a_e L + b && \text{(where } |a_e|, |b_e| \ll L\text{)} \end{aligned}$$

□

Lemma 5.10. *Consider two base vectors x_1 and x_2 such that $\|x_1\|^2, \|x_2\|^2 < |\mathscr{V}|M^2$. If $x_2 = (1-\lambda)x_1 + \lambda b^{\prec}$ and $b^{\prec}$ is an invalid extreme base, then $\lambda \leq |\mathscr{V}|\frac{M}{L}$.*

Proof. Recall that in our algorithm, base vector is represented as the sum of contributions from valid and invalid extreme bases separately: $x = x_v + x_i$, where x_v and x_i are the base vectors collecting contributions of valid and invalid extreme bases respectively. Further, we start from a valid extreme and in each iteration of the algorithm, keep on decreasing the norm of the overall base vector. Note that, all the elements of a valid extreme base are smaller than M. Therefore the squared ℓ_2 norm of the overall base vector is less than $|\mathscr{V}|M^2$ at any point in the algorithm.

We will prove the lemma by contradiction, and show that unless the λ for the invalid extreme base is less than $|\mathscr{V}|M/L$, the squared norm of the overall base vector is more than $|\mathscr{V}|M^2$, which is a contradiction.

We will first need to prove the following result:

Lemma 5.11. *Consider an invalid ordering $\prec$, and its corresponding invalid extreme base $b^{\prec}$. Let e be the smallest element (as per $\prec$), for which validity condition is violated. Then, $\exists\ a_e \in \mathbb{R}$, and $a_e \geq (1 - M/L)$, s.t. $b^{\prec}(e) = a_e L$.*

Proof. Let S_2 be the set of all elements smaller than e as per $\prec$. Let $S_1 = S_2 \cup \{e\}$. Notice that S_2 is a valid and S_1 is an invalid state.

$$
\begin{aligned}
b^{\prec}(e) &= f(S_1) - f(S_2) && \text{(Definition of extreme base)} \\
&= \Big(f(\overline{S}_1) + (|\overline{S}_1| - |S_1|)L\Big) - f(S_2) && \text{(Definition 5.5)} \\
&\geq \min_{S \in Z} f(S) - f(S_2) + (|\overline{S}_1| - |S_1|)L \\
&\qquad\qquad (\overline{S}_1 \text{ is a valid state, therefore } f(\overline{S}_1) \geq \min_{S \in Z} f(S)) \\
&\geq \min_{S \in Z} f(S) - \max_{S \in Z} f(S) + (|\overline{S}_1| - |S_1|)L \\
&\qquad\qquad (S_2 \text{ is a valid state, therefore } f(S_2) \leq \max_{S \in Z} f(S)) \\
&= (|\overline{S}_1| - |S_1| - M/L)L && \text{(Defintion of } M)
\end{aligned}
$$

Note that for any invalid state S_1, $(|\overline{S}_1| - |S_1|) \geq 1$. Therefore there exists $a_e \geq (1 - M/L)$ such that $b^{\prec}(e) = a_e L$. □

To prove our main result by contradiction, assume $\lambda > |\mathscr{V}|M/L$. Let e be the smallest element (as per $\prec$ of invalid extreme base $b^{\prec}$), for which validity condition is violated. Consider:

$$
\begin{aligned}
(x_2(e))^2 &= ((1-\lambda)x_1(e) + \lambda b^{\prec}(e))^2 \\
&> ((1-\lambda)x_1(e) + b^{\prec}(e)|\mathscr{V}|M/L)^2 && (\lambda > |\mathscr{V}|M/L) \\
&= ((1-\lambda)x_1(e) + a_e|\mathscr{V}|M))^2. && \text{(Using lemma 5.11)}
\end{aligned}
$$

Two cases are possible:

1. $x_1(e) \geq 0$:

$$
\begin{aligned}
(x_2(e))^2 &\geq ((1-\lambda)x_1(e) + a_e|\mathscr{V}|M))^2 \\
&\geq (a_e|\mathscr{V}|M)^2 && \text{(Since } (1-\lambda) \geq 0) \\
&= (a_e|\mathscr{V}|)(|\mathscr{V}|M^2)
\end{aligned}
$$

Since $M \ll L$, and $a_e \geq (1 - M/L)$, therefore $a_e \approx 1$. The smallest problem size that we consider is of 3 pixels and 2 labels for which $|\mathscr{V}| = 6$. Hence, for our case, $a_e|\mathscr{V}| > a_e\sqrt{|\mathscr{V}|} > 2$. This implies:

$(x_2(e))^2 > |\mathscr{V}|M^2$.

2. $x_1(e) < 0$:

Note that for x_1 and any element $e \in \mathscr{V}$ we have $x_1(e)^2 \leq \|x_1\|^2 \leq |\mathscr{V}|M^2$. This implies that $x_1(e) \geq -\sqrt{|\mathscr{V}|}M$.

$$\begin{aligned}(x_2(e))^2 &\geq ((1-\lambda)x_1(e) + a_e|\mathscr{V}|M)^2 \\ &\geq (-(1-\lambda)\sqrt{|\mathscr{V}|}M + a_e|\mathscr{V}|M)^2 \\ &\geq (-\sqrt{|\mathscr{V}|}M + a_e|\mathscr{V}|M)^2 \qquad \text{(Since } 1 \geq (1-\lambda) \geq 0\text{)} \\ &= M^2|\mathscr{V}|(a_e\sqrt{|\mathscr{V}|} - 1) \quad \text{(As described in the first case } a_e\sqrt{|\mathscr{V}|} > 2\text{)} \\ &> |\mathscr{V}|M^2.\end{aligned}$$

Both the cases imply that if $\lambda > |\mathscr{V}|M/L$ then norm $||x_2||^2 > |\mathscr{V}|M^2$ which is a contradiction. Hence for any invalid extreme base its contribution λ in the overall base vector must be less than $|\mathscr{V}|M/L$. □

Conceptually, Lemma 5.9 shows that all elements of an invalid extreme base are either small or are proportional to L (and not proportional to, say L^2, or other higher powers of L). Whereas Lemma 5.10 shows that since $\mathscr{V}$ and M are effectively constants, λ the multiplicative factor associated with in the contribution of invalid extreme bases, λ is proportional to $1/L$. Therefore, as $L \to \infty$, the value of $\lambda \to 0$. However, it is important to note that the value of $\lambda b^{\prec}(e)$, is always finite. It is easy to see that, whenever $a_e = 0$, $\lambda b^{\prec}(e) \to 0$, and when $a_e \neq 0$, the L present in the $b^{\prec}(e)$ and $1/L$ present in λ cancel each other, leading to a finite contribution. The above motivates our overall approach in this paper that, for a numerically stable norm minimization algorithm, focus should be on manipulating the finite valued product $\lambda b^{\prec}$, and not the individual λ and $b^{\prec}(e)$. We show in the following sections that this is indeed possible.

We start by showing that it is possible to find a small set of what we call *elementary* invalid extreme bases whose linear combination contains as a subset the space of vectors x_i as given in Eq. (5.2). Crucial to doing this is the notion of Canonical Orderings.

5.2.1 Canonical Ordering and Its Properties

In an arbitrary, valid or invalid, ordering $\prec$ consider two adjacent elements u and v such that $u \prec v$. We term swapping of order locally between u and v in $\prec$ as an *exchange operation*. The operation will result in a new ordering $\prec_{\text{new}}$ such that u and v are still adjacent but $v \prec_{\text{new}} u$.

Consider a strategy in which starting with $\prec$ we carry out exchange operations till all the elements corresponding to a pixel come together, and repeat this

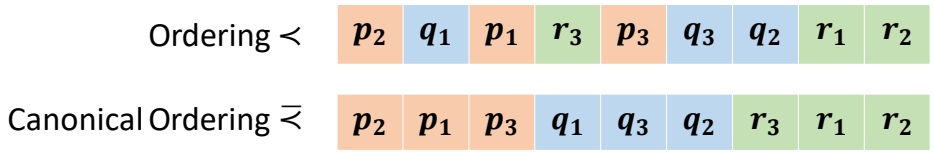

Figure 5.1: Top: An ordering of elements in $\mathscr{P} = \{p, q, r\}$, for a label of size 3. Bottom: Corresponding canonical ordering.

for all pixels. Note that we do not change the relative ordering between elements corresponding to the same pixel. We call the resultant ordering the *canonical* form of the original ordering $\prec$ and denote it by $\overline{\prec}$. The corresponding extreme base is called *canonical extreme base*. Figure 5.1 contains an example of an arbitrary ordering and one of its canonical orderings.

Note that a valid (invalid) ordering leads to a valid (invalid) canonical ordering. For any p^j and p^k, in a valid canonical ordering, if $j = k+1$, then p^j, p^k are adjacent in the ordering and $p^k \ \overline{\prec}\ p^j$. Further, a canonical ordering is agnostic to any relative order among pixels. For example, for pixels p and q, a canonical ordering only requires that all elements of p (or q) are contiguous. An ordering in which elements corresponding to p come before those of q will define a different canonical ordering from the one in which the relative ordering of elements of p and q is vice-versa. In general a canonical ordering $\overline{\prec}$ corresponding to a $\prec$ can be any one of the possible canonical orderings.

Lemma 5.12. *Let $\prec$ be an invalid ordering and $\overline{\prec}$ be its canonical ordering. Then, $b^{\prec}(e) - b^{\overline{\prec}}(e) \ll L, \forall e \in \mathscr{V}$.*

Proof. Let S and S' be the set of elements preceding $p^i, p \in \mathscr{P}$, in ordering $\prec$ and $\overline{\prec}$ respectively. Consider the term $b^{\prec}(p^i)$:

$$\begin{aligned}
b^{\prec}(p^i) &= f(S \cup \{p^i\}) - f(S) \\
&= L \sum_{q \in \mathscr{P}} \Big(|(\overline{S \cup \{p^i\}})_q| - |(S \cup \{p^i\})_q| \Big) + f\Big(\overline{S \cup \{p^i\}}\Big) \\
&\quad - L \sum_{q \in \mathscr{P}} \Big(|\overline{S}_q| - |S_q| \Big) - f\Big(\overline{S}\Big) \qquad \text{(Def. 5.6)} \\
&= L\Big(|\overline{S_p \cup \{p^i\}}| - |S_p \cup \{p^i\}| \Big) - L\Big(|\overline{S}_p| - |S_p| \Big) \\
&\quad + f(\overline{S \cup \{p^i\}}) - f(\overline{S})
\end{aligned}$$

Similarly we obtain:

$$\begin{aligned}
b^{\overline{\prec}}(p^i) &= L\Big(|\overline{S'_p \cup \{p^i\}}| - |S'_p \cup \{p^i\}| \Big) - L\Big(|\overline{S'}_p| - |S'_p| \Big) \\
&\quad + f\Big(\overline{S' \cup \{p^i\}}\Big) - f\Big(\overline{S'}\Big)
\end{aligned}$$

Note that a canonical ordering does not change intersay ordering between elements corresponding to a particular pixel. Therefore, $S_p = S'_p$, and:

$$b^{\prec}(p^i) - b^{\overline{\prec}}(p^i) = (f(\overline{S \cup \{p^i\}}) - f(\overline{S}) - f(\overline{S' \cup \{p^i\}}) + f(\overline{S'})$$

Since, all terms in the r.h.s. of the equation above, correspond to valid sets, therefore $b^{\prec}(p^i) - b^{\overline{\prec}}(p^i) \ll L$. □

The above result serves to indicate that by changing an invalid extreme base to canonical one, the change in value of any element of the extreme base is much less than L. Therefore, due to Lemma 5.10, one can conclude that the contribution of an invalid extreme base or its canonical extreme base in a base vector is going to be the same.

Lemma 5.13. *For a canonical invalid ordering $\overline{\prec}$, let p^i and p^j be two adjacent elements corresponding to a pixel p, s.t. $p^i \overline{\prec} p^j$. Let $\overline{\prec}_p^{i,j}$ be the ordering obtained by swapping p^i and p^j. Then $b^{\overline{\prec}_p^{i,j}} - b^{\overline{\prec}} = (\chi_p^j - \chi_p^i)(aL + b)$, where χ_p^i is an indicator vector for the element p^i, and $a, b \ll L$.*

Proof. Recall that for an extreme base $b^{\prec} : b^{\prec}(k) = f(k_{\prec}) - f((k-1)_{\prec})$, where $k_{\prec}$ is the first k elements in the ordered set $\{v_1, \ldots, v_k, \ldots, v_n\}$. Since the swap between p^i, and p^j leaves the set of preceding elements unchanged for all other elements, therefore, $b^{\overline{\prec}_p^{i,j}} - b^{\overline{\prec}}$ is non-zero corresponding to only p^i, and p^j.

Let S be the set of elements preceding p^i in $\overline{\prec}$. Now:

$$\begin{aligned} b^{\overline{\prec}}(p^j) &= f(S \cup \{p^j\} \cup \{p^i\}) - f(S \cup \{p^i\}) \\ &= L(|\overline{S \cup \{p^j\} \cup \{p^i\}}| - |S \cup \{p^j\} \cup \{p^i\}|) + f(\overline{S \cup \{p^j\} \cup \{p^i\}}) \\ &\quad - L(|\overline{S \cup \{p^i\}}| - |S \cup \{p^i\}|) - f(\overline{S \cup \{p^i\}}) \end{aligned} \tag{5.3a}$$

Similarly we have,

$$b^{\overline{\prec}_p^{i,j}}(p^j) = L(|\overline{S \cup \{p^j\}}| - |S \cup \{p^j\}|) - L(|\overline{S}| - |S|) + f(\overline{S \cup \{p^j\}}) - f(\overline{S}), \tag{5.3b}$$

Subtracting Eq. (5.3a) from Eq. (5.3b) and using $(|S \cup \{p^i\}| + |S \cup \{p^j\}| - |S \cup \{p^i\} \cup \{p^j\}| - |S|) = 0$ we have

$$\begin{aligned} b^{\overline{\prec}_p^{i,j}}(p^j) - b^{\overline{\prec}}(p^j) &= L(|\overline{S \cup \{p^i\}}| + |\overline{S \cup \{p^j\}}| - |\overline{S \cup \{p^i\} \cup \{p^j\}}| - |\overline{S}|) \\ &\quad + (f(\overline{S \cup \{p^j\}}) - f(\overline{S}) - f(\overline{S \cup \{p^j\} \cup \{p^i\}}) + f(\overline{S \cup \{p^i\}}), \\ &= aL + b, \end{aligned}$$

where:

$$\begin{aligned} a &= |\overline{S \cup \{p^i\}}| + |\overline{S \cup \{p^j\}}| - |\overline{S \cup \{p^i\} \cup \{p^j\}}| - |\overline{S}|, \text{ and} \\ b &= (f(\overline{S \cup \{p^j\}}) - f(\overline{S}) - f(\overline{S \cup \{p^j\} \cup \{p^i\}}) + f(\overline{S \cup \{p^i\}}). \end{aligned}$$

Note that b is sum of function values at valid states and is $\ll L$. Two cases arise for the value of a:

1. $i < j$:
   In this case $|\overline{S_p \cup \{p^i\} \cup \{p^j\}}| = |\overline{S_p \cup \{p^j\}}|$, and $a = |\overline{S \cup \{p^i\}}| - |\overline{S}|$. Therefore, $a \ll L$.

2. $j < i$:
   In this case $|\overline{S_p \cup \{p^i\} \cup \{p^j\}}| = |\overline{S_p \cup \{p^i\}}|$, and $a = |\overline{S \cup \{p^j\}}| - |\overline{S}|$. Therefore, $a \ll L$

Hence $b^{\overline{\prec_p^{i,j}}}(p^j) - b^{\overline{\prec}}(p^j) = aL + b$, such that $a, b \ll L$. Further, since $b^{\overline{\prec_p^{i,j}}}$ and $b^{\overline{\prec}}$ are extreme bases, and the sum of all the elements in them is constant, therefore, the reverse must hold for $b^{\overline{\prec_p^{i,j}}}(p^i) - b^{\overline{\prec}}(p^i)$. Hence $b^{\overline{\prec_p^{i,j}}} - b^{\overline{\prec}} = (\chi_p^j - \chi_p^i)(aL + b)$. □

Lemma 5.13 relates the two extreme bases when one pair of their elements is swapped. It is useful to note that in a valid extreme base all elements have small values. With each swap in an invalid canonical ordering we either move the canonical ordering towards validity or away from it. In each swap the change in the value of an element is proportional to L (positive or negative). Since conversion of an invalid canonical ordering to a valid one may involve swaps between a number of elements, the extreme base corresponding to the invalid ordering will contain multiple elements with values proportional to L. The special cases are the ones in which only one swap has been done. In these cases there will be only two elements with values proportional to L (positive and negative). We show that using such extreme bases as the basis to represent canonical invalid extreme bases. In the next section we show that it is indeed possible.

5.2.2 Elementary Invalid Extreme Base

Definition 5.14 (Elementary Invalid Extreme Base)**.** The ordering obtained by swapping two elements p^j and p^{j+1}, corresponding to a pixel p, in a canonical valid ordering, is called an *elementary invalid ordering*. Its corresponding extreme base is called *elementary invalid extreme base*, and is denoted as $b^{\widetilde{\prec_p^j}}$.

Lemma 5.15. *Consider an elementary invalid extreme base $b^{\widetilde{\prec_p^i}}$, obtained by swapping two adjacent elements (p^{i+1}, p^i) in the universal ordering, $\prec_0$ (Def. 5.2). Then:*

$$b^{\widetilde{\prec_p^i}} - b^{\prec_0} = (\chi_p^i - \chi_p^{i+1})(L + b),$$

where $b^{\prec_0}$ is the valid extreme base corresponding to $\prec_0$.

Proof. Recall:

- The universal ordered sequence, $\prec_0$, which is a valid ordering, and also defines a particular ordering among the pixels.
- The elementary invalid ordering, $\widetilde{\prec}$, which is defined as the ordering obtained by making one swap between adjacent elements of a valid ordering. The corresponding extreme base is denoted as $b^{\widetilde{\prec}}$.

Further, recall from the proof of Lemma 5.13, we showed that: $b^{\overline{\prec}_p^{i,j}} - b^{\overline{\prec}} = (\chi_p^j - \chi_p^i)(aL+b)$, such that $a = |\overline{S \cup \{p^i\}}| - |\overline{S}|$ (if $i < j$), or $a = |\overline{S \cup \{p^j\}}| - |\overline{S}|$ (if $j < i$). Now consider an elementary invalid extreme base $b^{\widetilde{\prec}_p^i}$, obtained by swapping two adjacent elements (p^{i+1}, p^i) in the universal ordering. The term $(\chi_p^i - \chi_p^{i+1})$ may be looked upon as corresponding to the creation of the elementary extreme base $b^{\widetilde{\prec}_p^i}$ from $b^{\prec_0}$. It is easy to see that for such special elementary invalid extreme bases created from universal ordering, $a = 1$, and we have:

$$b^{\widetilde{\prec}_p^i} - b^{\prec_0} = (\chi_p^i - \chi_p^{i+1})(L + b), \tag{5.4}$$

Hence, proved. □

Lemma 5.16. *An invalid canonical extreme base, $b^{\overline{\prec}}$, can be represented as a linear combination of elementary invalid extreme base vectors such that:*

$$b^{\overline{\prec}} = \sum_{p \in \mathscr{P}} \sum_{i=1}^{m-1} \alpha_p^i b^{\widetilde{\prec}_p^i} + \Lambda,$$

where $0 < \alpha_p^i \ll L$, and Λ is a vector with all its elements much smaller than L.

Proof. Consider the canonical invalid ordering $\overline{\prec}$ and let $\prec_s$ be the starting canonical valid ordering from which it can be obtained by a series of swaps between adjacent elements. Note that since in the canonical ordering all the elements of a pixel are already together, therefore all the swaps required are between elements corresponding to same pixels. Let us assume that total number of such swaps required are T. Starting from $\prec_s$, let $\prec_j$ represents the ordering obtained after j such swaps. Hence, $\prec_T = \overline{\prec}$ by definition. Let j^{th} swap happens between elements p^{k_j} and p^{l_j}, where $p \in \mathscr{P}$.

$$\begin{aligned}
b^{\overline{\prec}} - b^{\prec_s} &= b^{\prec_T} - b^{\prec_s} \\
&= \sum_{j=1}^{T} \left(b^{\prec_j} - b^{\prec_{j-1}} \right) \\
&= \sum_{j=1}^{T} \left(\chi_p^{l_j} - \chi_p^{k_j} \right)(a_j L + b_j) && \text{(Using Lemma 5.13)} \\
&= \sum_{j=1}^{T} (\chi_p^{l_j} - \chi_p^{k_j}) a_j L + \sum_{j=1}^{T} (\chi_p^{l_j} - \chi_p^{k_j}) b_j.
\end{aligned}$$

Since $(\chi_p^{l_j} - \chi_p^{k_j}) = \sum_{i=l_j}^{k_j}(\chi_p^i - \chi_p^{i+1})$ we can write:

$$b^{\overline{\prec}} - b^{\prec_s} = \sum_{j=1}^{T} a_j \sum_{i=l_j}^{k_j}(\chi_p^i - \chi_p^{i+1})L + \sum_{j=1}^{T} b_j \sum_{i=l_j}^{k_j}(\chi_p^i - \chi_p^{i+1}). \tag{5.5}$$

Recall from Lemma 5.15:

$$\begin{aligned} & b^{\widetilde{\prec}_p^i} - b^{\prec_0} = (\chi_p^j - \chi_p^{i+1})(L + b) \\ \Rightarrow \quad & (\chi_p^i - \chi_p^{i+1})L = b^{\prec_p^i} - b^{\prec_0} - (\chi_p^i - \chi_p^{i+1})b_p^i. \qquad \text{(where } b_p^i \ll L\text{)} \end{aligned}$$

Substituting the value of $(\chi_p^i - \chi_p^{i+1})L$ in Eq. (5.5), we get:

$$b^{\overline{\prec}} - b^{\prec_s} = \sum_{j=1}^{T} a_j \sum_{i=l_j}^{k_j}(b^{\widetilde{\prec}_p^i} - b^{\prec_0} - (\chi_p^i - \chi_p^{i+1})b_p^i) + \sum_{j=1}^{T} b_j \sum_{i=l_j}^{k_j}(\chi_p^i - \chi_p^{i+1}).$$

Since both $\prec_s$, and $\prec_0$ are valid orderings, we can write $b^{\prec_s} = b^{\prec_0} + \vec{d}$, where elements of $\vec{d}$ are much smaller than L. Therefore we get

$$\begin{aligned} b^{\overline{\prec}} = & \sum_{j=1}^{T}\sum_{i=l_j}^{k_j} a_j b^{\widetilde{\prec}_p^i} + \Big(1 - \sum_{j=1}^{T}\sum_{i=l_j}^{k_j} a_j\Big)b^{\prec_0} - \sum_{j=1}^{T}\sum_{i=l_j}^{k_j} a_j(\chi_p^i - \chi_p^{i+1})b_p^i \\ & + \sum_{j=1}^{T}\sum_{i=l_j}^{k_j}(\chi_p^i - \chi_p^{i+1})b_j + \vec{d} \\ b^{\overline{\prec}} = & \sum_{j=1}^{T}\sum_{i=l_j}^{k_j} a_j b^{\widetilde{\prec}_p^i} + \Lambda, \end{aligned} \tag{5.6}$$

where Equation (5.6) has been derived summing the last 4 terms into a vector Λ. Note that all the elements of Λ are $\ll L$. It is easy to see that the first term in the equation essentially is a linear combination of some elementary invalid extreme bases, allowing us to simplify:

$$b^{\overline{\prec}} = \sum_{p\in P}\sum_{i=1}^{m-1} \alpha_p^i b^{\widetilde{\prec}_p^i} + \Lambda, \tag{5.7}$$

where coefficients α_p^i corresponding to elementary extreme bases not present in Equation (5.6) can be simply set to zero. □

Due to Lemma 5.12, the above result is also true for representing the invalid extreme bases (and not only the canonical ones), with a different Λ. Lemma (5.15) allows us to further simplify the result of Lemma (5.16) to the following:

Lemma 5.17 (Invalid Extreme Base Representation)**.** *An invalid extreme base can be represented as* $b^{\prec} = \sum_{p\in\mathscr{P}}\sum_{i=1}^{m-1}\alpha_p^i L(\chi_p^i - \chi_p^{i+1}) + \Lambda$*, where* χ_p^i *is an indicator vector corresponding to element* p^i*,* $0 < \alpha_p^i \ll L$*, and* Λ *is some vector whose all elements are* $\ll L$*.*

Proof. Using Equation (5.7), we have:

$$b^{\overline{\prec}} = \sum_{p\in P}\sum_{i=1}^{m-1}\alpha_p^i b^{\widetilde{\prec}_p^i} + \Lambda.$$

Substituting representation of elementary extreme base from Equation (5.4), we have:

$$\begin{aligned} b^{\prec} &= \sum_{p\in P}\sum_{i=1}^{m-1}\alpha_p^i\Big(b^{\prec_0} + (\chi_p^i - \chi_p^{i+1})(L + b_p^i)\Big) + \Lambda. \\ &= \sum_{p\in P}\sum_{i=1}^{m-1}\alpha_p^i b^{\prec_0} + \sum_{p\in P}\sum_{i=1}^{m-1}\alpha_p^i L(\chi_p^i - \chi_p^{i+1}) + \sum_{p\in P}\sum_{i=1}^{m-1}(\chi_p^i - \chi_p^{i+1})\alpha_p^i b_p^i + \Lambda. \\ &= \sum_{p\in P}\sum_{i=1}^{m-1}\alpha_p^i L(\chi_p^i - \chi_p^{i+1}) + \Lambda. \end{aligned}$$

Note that we have replaced Λ with $\sum_{p\in P}\sum_{i=1}^{m-1}\alpha_p^i b^{\prec_0} + \sum_{p\in P}\sum_{i=1}^{m-1}(\chi_p^i - \chi_p^{i+1})\alpha_p^i b_p^i + \Lambda$. □

Recall from Eq. (5.2): $x = x_v + x_i$, where $x_v = \sum_{b^{\prec_j}\in R}\lambda_j b^{\prec_j}$, and $x_i = \sum_{b^{\prec_i}\in Q}\lambda_i b^{\prec_i}$. Using Lemma 5.16, as $L\to\infty$, $\lambda_i\to 0$, and $\sum\lambda_j\to 1$ to replace the second term, one observes that the term $\sum_{b_i\in Q}\lambda_i\Gamma_i$ in the expansion can be made smaller than the precision constant by increasing the value of L ($\lambda < |\mathscr{V}|M/L$ by Lemma 5.10) and can be dropped. As one of the final theoretical results of this paper, we can show the following:

Theorem 5.18 (Main Result)**.**

$$\sum_{\forall b^{\prec_i}\in Q}\lambda_i b^{\prec_i} = \sum_{p\in\mathscr{P}}\sum_{k=1}^{m-1}\beta_p^k L(\chi_p^k - \chi_p^{k+1}),$$

where $\lambda_i \geq 0$*,* $\beta_p^k = \sum_{b_i\in Q}\alpha_p^k\lambda_i$*.*

Proof. Consider the expansion of the term $x_i = \sum_{b^{\prec_i}\in Q}\lambda_i b^{\prec_i}$ in Eq.(5.2). Using Theorem (5.17) we get:

$$x_i = \sum_{b^{\prec_i}\in Q}\sum_{p\in\mathscr{P}}\sum_{k=1}^{m-1}\lambda_i\alpha_p^k L(\chi_p^k - \chi_p^{k+1}) + \sum_{b^{\prec_i}\in Q}\lambda_i\Lambda_i.$$

Recall from Lemma (5.10) that, for all $b^{\prec_i} \in Q$, the coefficient λ_i can be made arbitrarily small. Therefore, we can drop the term $\sum_{b^{\prec_i} \in Q} \lambda_i \Lambda_i$ and rewrite the above equation as:

$$x_i = \sum_{p \in P} \sum_{k=1}^{m-1} \sum_{b^{\prec_i} \in Q} \lambda_i \alpha_p^k L(\chi_p^k - \chi_p^{k+1}).$$

Replacing by $\beta_p^k = \sum_{b^{\prec_i} \in Q} \lambda_i \alpha_p^k$, we get:

$$x_i = \sum_{p \in P} \sum_{k=1}^{m-1} \beta_p^k L(\chi_p^k - \chi_p^{k+1}).$$

□

Note that the above result incorporates *all* the invalid extreme bases, not merely the ones involved in the representation of base vector x in any iteration of MNP. Using the result in Eq. (5.2), we get the following:

$$\|x\|^2 = \left\| \sum_{b^{\prec_j} \in R} \lambda_j b^{\prec_j} + \sum_{p \in \mathcal{P}} \sum_{k=1}^{m-1} \beta_p^k L(\chi_p^k - \chi_p^{k+1}) \right\|^2.$$

5.3 The Multi-label Hybrid Algorithm

In this section we give the algorithm for minimizing the norm of the base vector corresponding to a single clique in the original MRF-MAP problem, where the pseudo-Boolean function is generated from encoding the multi-label function. For solving the overall MRF-MAP problem with multiple cliques, the proposed algorithm can be used in the inner loop of the BCD strategy SoSMNP as suggested in Chapter 3.

Theorem (5.18) opens up the possibility of minimizing $\|x\|^2$ for a single clique also by the BCD strategy. We will have two blocks in this case. The first block, called the *valid block*, is a convex combination of valid extreme bases $b^{\prec_j}$, where standard MNP algorithm can be used to optimize the block. The other block, called the *invalid block*, corresponds to the sum of the mn terms of type: $\beta_p^k L(\chi_p^k - \chi_p^{k+1})$, representing the invalid extreme bases. Below we show how the min ℓ_2 norm can be calculated for the invalid block.

For the purpose of minimizing the norm of the overall base vector using the invalid block, we hold the contribution from the valid block, x_v, constant [2]. Each vector $\beta_p^k L(\chi_p^k - \chi_p^{k+1})$ may be looked upon as capturing the β_p^k increase/decrease due to the exchange operation between the two adjacent elements which define

[2]Recall that we start from a valid extreme base. Therefore, at initialization $x = x_v$

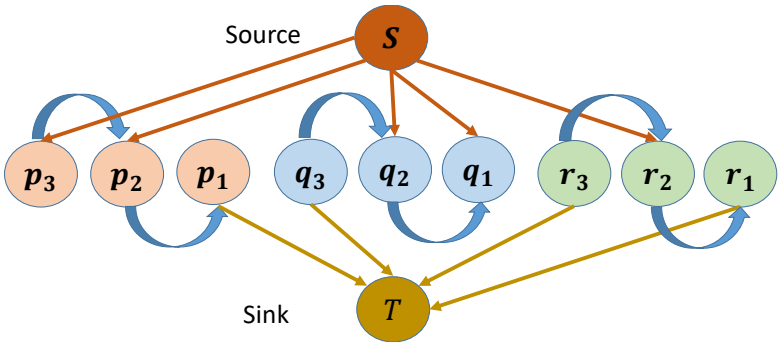

Figure 5.2: Flow graph corresponding to the exchange operations for optimizing the block containing invalid extreme bases.

an elementary extreme base. This exchange operation can be viewed as flow of $\beta_p^k L$ from the element p^{k+1} to p^k.

We model the optimization problem for the invalid block using a flow graph whose nodes consists of $\{p^k \mid p \in \mathscr{P}, 1 \leq k \leq m-1\} \cup \{s,t\}$. We add two type of edges:

- **Type 1:** If $x_v(p^k)$, corresponding to the valid block contribution, is > 0, then we add a directed edge from $s \rightarrow p^k$, else we add the edge from $p^k \rightarrow t$ with capacity $x_v(p^k)$.
- **Type 2:** The directed edges p^{k+1} to p^k, $1 \leq k \leq (m-1)$ with capacity $|\mathscr{V}|M$ to ensure that the capacity is at least as large as $\beta_p^k L$: much larger than any permissible value of $x_v(p^k)$. Thus, any feasible flow augmentation in a path from from s to t can saturate only the first or the last edge in the augmenting path (i.e. the edge emanating from s or the edge incident at t in the path).

Figure 5.2 is an example of a flow graph for 3 pixel and 3 label problem. Since the starting state is x_v the "initial flow" prior to pushing flow for flow maximization requires setting flow in a type 1 edge incident at p^k equal to the value of $x_v(p^k)$ and that in type 2 edges as 0. This is because sum of flow on all edges incident at a node may be looked upon as the value of the corresponding element in the base vector [3]. In effect initially there are non zero excesses on the non s,t nodes in the flow graph defined as the sum of net in-flow on all edges incident at a node. The excess at node p_k is denoted by $e(p_k)$. Max flow state can be looked upon as that resulting from repeatedly sending flow from a positive excess vertex to a negative excess vertex till that is no more possible. Values

[3] we refer the reader to Chapter 4 for details about the flow to base vector correspondence

Procedure 9 Computing Min ℓ_2 Norm from the Flow Output

Input: Vector e the output of the max flow algorithm.
Output: The transformed vector e with minimum ℓ_2 norm.

```
for ∀p ∈ 𝒫 do
  for i = 2 : m do
    repeat
      find smallest k, i ≥ k ≥ 1, such that
      e(p^i) > e(p^{i-1}) = e(p^{i-2}) ··· = e(p^k) or e(p^i) = e(p^{i-1}) =
      e(p^{i-2}) ··· = e(p^{k+1}) > e(p^k);
      let av_k be the average of e(p^i), e(p^{i-1}), ..., e(p^k);
      set e(p^i), e(p^{i-1}), ..., e(p^k) equal to av_k;
    until e(p^{k+1}) ≤ e(p^k)
  end for
end for
```

in the optimal base vector (optimal subject to the given x_v) at the end of this iteration will be the excesses at nodes when max flow state has been reached.

5.3.1 Computing Min ℓ_2 Norm By Flow

Since there is no edge between any two nodes corresponding to different pixels max flow can be calculated independently for each pixel. When max flow state is reached in the flow graph associated with a pixel, a vertex which still has a negative excess will be to the left of vertices with positive excess (planar flow graph laid out as in Figure (5.2) otherwise flow could be pushed from a positive excess vertex to a negative excess vertex.

Note that the optimal base vector is not unique. Consider two adjacent vertices, p^{k+1} and p^k, in the flow graph when the max flow state has been reached. If $e(p^{k+1})$ is larger than $e(p^k)$ then increasing the flow in the edge from p^{k+1} to p^k by δ decreases $e(p^{k+1})$ by δ and increases $e(p^k)$ by δ. The result of this "exchange operation" is to create another optimal base vector but with a smaller ℓ_2 norm.

An optimal base vector with minimum ℓ_2 norm will correspond to the max flow state in the flow graph in which $e(p^{k+1}) \leq e(p^k)$ for all adjacent pairs of type 2 vertices. If this is not so then there would exist at least a pair $e(p^{k+1})$ and $e(p^k)$ such that $e(p^{k+1}) > e(p^k)$. Doing an exchange operation between p^{k+1} and p^k involving setting $e(p^{k+1})$ and $e(p^k)$ to the average of the old values will create a new optimal base vector with lower value of the ℓ_2 norm. Algorithm 9 gives an efficient procedure to transform the optimal base vector outputted by the max flow algorithm to one with minimum ℓ_2 norm. Note that the proposed algorithm simply updates the base vector in one pass without any explicit flow pushing. In contrast, the corresponding algorithm for general flow graphs given

in Chapter 4 requires $O(n \log n)$ additional max flow iterations over an n vertex flow graph.

5.3.2 Overall Algorithm

The proposed Multi-label Hybrid (MLHybrid) algorithm (Proc. 11) is quite similar to the algorithm in [87] in its over all structure. Just like [87], we also create blocks corresponding to each clique, and optimize each block independently (taking the contribution of other blocks as suggested in [87]) in an overall block coordinate descent strategy. The only difference between SoSMNP and MLHybrid is the way we optimize one block. While SoSMNP uses standard MNP, we optimize using a special technique, as outlined in previous section, with (sub)blocks of valid and invalid extreme bases, within each block/clique. Hence, the convergence and correctness of overall algorithm follows from block coordinate descent similar to [87]. What we need to show is that for a single clique/block, the algorithmic strategy of alternating between valid and invalid blocks converges to the optimal for that clique/block.

Recall that in a standard MNP algorithm iteration, given the current base vector x, an extreme base q, that minimizes $x^\intercal q$ is added to the current set. Hence, steps to convergence of MNP is bounded by the number of extreme bases that may be added. In our case we have shown in Lemma 5.19 that when we start with a valid extreme base, the extreme base generated in the valid block after using the latest contribution from the invalid block, will come out to be a valid extreme base. This implies that the number of iterations involving invalid blocks can not exceed the number of valid extreme bases added as in the standard MNP algorithm. This ensures convergence of the optimization step for each block. The formal convergence proof for the MLHybrid algorithm is given in the Section 5.4.

The correctness of our optimization for each block follows from the fact that the optimization for valid blocks proceeds in the standard way, and results in a new extreme base given the current base vector. The correctness of the optimization step of the invalid block, which finds a minimum norm base vector given a valid block, has already been explained in the previous section.

5.4 Convergence of ML-hybrid Algorithm

Note that in SoSMNP each block is optimized using the MNP algorithm. In MLhybrid, on the other hand, each block is further subdivided. One corresponds to the set of valid extreme bases (the valid block) and the other to the set of invalid extreme bases (the invalid block) whose convex combination defines the base vector $x_{\mathbf{c}}$. MNP is run on the valid block. If at any iteration MNP [31] inserts an invalid extreme base, the flow based Algorithm 10 is run on the invalid

Procedure 10 ComputeInvalidContribution

Input: Vector $x_{\mathbf{c}}$ the output of the max flow algorithm.
Output: The transformed vector $x_{\mathbf{c}}$ with minimum ℓ_2 norm .

1: **for** $\forall p \in \mathbf{c}$ **do**
2: **for** $i = 2 : m$ **do**
3: **repeat**
4: find smallest k, $i \geq k \geq 1$, such that $x_{\mathbf{c}}(p^i) > x_{\mathbf{c}}(p^{i-1}) = x_{\mathbf{c}}(p^{i-2}) \cdots = x_{\mathbf{c}}(p^k)$ or $x_{\mathbf{c}}(p^i) = x_{\mathbf{c}}(p^{i-1}) = x_{\mathbf{c}}(p^{i-2}) \cdots = x_{\mathbf{c}}(p^{k+1}) > x_{\mathbf{c}}(p^k)$;
5: let av_k be the average of $x_{\mathbf{c}}(p^i), x_{\mathbf{c}}(p^{i-1}), \ldots, x_{\mathbf{c}}(p^k)$;
6: set $x_{\mathbf{c}}(p^i), x_{\mathbf{c}}(p^{i-1}), \ldots, x_{\mathbf{c}}(p^k)$ equal to av_k;
7: **until** $x_{\mathbf{c}}(p^{k+1}) \leq x_{\mathbf{c}}(p^k)$
8: **end for**
9: **end for**

block. We show below that when MNP is run on the valid block now (that is just after a run of the flow based algorithm on the invalid block) the extreme base generated will be valid.

Lemma 5.19. *Algorithm 10 returns a vector $x_{\mathbf{c}}$ for clique $\mathbf{c}$ such that extreme base q_c given as $q_c = \arg\min_{q \in B(f_{\mathbf{c}})} x_{\mathbf{c}}^T q$ is valid.*

Proof. It is easy to show that when Algorithm 10 terminates, for any pair of indices i, j corresponding to any $p \in \mathscr{P}$ if $i > j$ then $e(p^i) \leq e(p^j)$. Note that by construction the excess vector e is the base vector $x_{\mathbf{c}}$. This implies that the order $\prec_{\mathbf{c}}$ of the indices obtained by sorting the elements of $x_{\mathbf{c}}$ will satisfy $p^i \prec_{\mathbf{c}} p^j, \forall i > j$, and $\forall p \in \mathscr{P}$. This is the condition that has to be satisfied for an ordering to be valid (Cf. Def. 5.3). Recall that in the MNP algorithm the extreme base is found by computing the ordering of sorted elements of x. Hence, the extreme base $q_c = \arg\min_{q \in B(f_{\mathbf{c}})} x_{\mathbf{c}}^T q$ will be a valid one. □

Lemma 5.19 implies that an iteration on the invalid block will be followed by the MNP algorithm making progress in the form of generation of a valid extreme base. Also note that the ℓ_2 norm decreases when the flow based algorithm is run on the invalid block. Therefore, termination and convergence of the MLhybrid algorithm running on a clique/block follows along the same lines as that for the standard MNP algorithm [20].

Now we show that termination over a clique/block results in $x_{\mathbf{c}}$ using which minimizer obtained comes on a valid state.

Note that generation of an invalid extreme base can always be followed by generation of a valid extreme base (by running the flow based algorithm on the invalid block). Therefore, at termination it is guaranteed that the order $\prec_{\mathbf{c}}$ of the indices obtained by sorting the elements of $x_{\mathbf{c}}$ is valid. That is the optimal

Procedure 11 HybridML: Algorithm for minimizing a sum of multilabel submodular functions

Input: $\{f_{\mathbf{c}}\}$ such that $f = \sum f_{\mathbf{c}}$.
Output: $x = \arg\min \|x\|^2$ subject to $x \in B(f)$.

```
# Initialize
for all (c ∈ 𝒞) do
    q_c ← Take any extreme base of f_c;
    S_c := {q_c};
    y_c := q_c;
end for
x := Σ_c y_c;
# Perform Block Coordinate Descent with blocks specified by Cliques
while (‖x‖ decreases by more than δ) do
    for all (c ∈ 𝒞) do
        MLHybridOverAClique(f_c,S_c,x_c,y_c);
    end for
end while
```

solution corresponds to a valid primal state. Hence, it follows, using Lemma 3.5, that the MLHybrid algorithm run in the block coordinate descent manner converges to the optimal.

5.5 Experiments

We have experimented with pixel-wise object segmentation and stereo correspondence problems. All experiments have been conducted on a computer with Intel Core i7 CPU, 8 GB of RAM running Windows 10. For the segmentation experiments, the input images are from Pascal VOC dataset [24] with a small amount of Gaussian noise added. We have experimented with two types of submodular clique potentials:

- Decomposable: Sum of absolute difference of labels for all pixel pairs in a clique. Denoted by ABS.
- Non-decomposable: Cconcave-of-Cardinality potential defined in [104] as: $\sum_{l \in \mathscr{L}}(\text{number of pixels} - \text{number of pixels which have their label as } l)^{\alpha}$. We have used $\alpha = 0.5$ in our experiments.

For both the potentials, two types of clique sizes namely "Small" (cliques ranging from 60 to 80 elements) and "Big" (cliques ranging from 300 to 400 elements) have been used for the experiments. Overlapping of cliques has been ensured by running SLIC algorithm [2] with different seeds.

Procedure 12 MLHybridOverAClique

Input: Clique function: $f_{\mathbf{c}}$
Input: Set of valid extreme bases selected in last iteration: $S_{\mathbf{c}}$
Input: Restriction of current solution vector x on $\mathbf{c}$: $x_{\mathbf{c}}$
Input: Current clique vector: $y_{\mathbf{c}}$
Output: Clique vector $y^*_{\mathbf{c}} \in B(f_c)$ minimizing $\|x_{\mathbf{c}}\|^2$
Output: Updated set $S^*_{\mathbf{c}}$ of valid extreme bases

while (TRUE) **do**
  Find new translation $a_{\mathbf{c}} := x_{\mathbf{c}} - y_{\mathbf{c}}$;
  Find extreme base $\hat{q}_{\mathbf{c}} := \arg\min_{q_{\mathbf{c}} \in B_{f_{\mathbf{c}}}} \langle x_{\mathbf{c}}, q_{\mathbf{c}} \rangle$ using Edmond's algorithm.
  if Extreme base $\hat{q}_{\mathbf{c}}$ is invalid according to Definition 5.3 **then**
    ComputeInvalidContribution($x_{\mathbf{c}}$);
    continue;
  end if
  Find translated extreme base $\hat{p}_{\mathbf{c}} = \hat{q}_{\mathbf{c}} + a_{\mathbf{c}}$;
  if $(\|x_{\mathbf{c}}\|^2 \leq \langle x_{\mathbf{c}}, \hat{p} \rangle + \epsilon)$ **then**
    break;
  end if
  $S_{\mathbf{c}} := S_{\mathbf{c}} \cup \hat{q}_{\mathbf{c}}$;
  $P_{\mathbf{c}} = \{\hat{q}_{\mathbf{c}} + a_{\mathbf{c}} | q_{\mathbf{c}} \in S_{\mathbf{c}}\}$;
  Find $x_{\mathbf{c}}$ in affine hull of $P_{\mathbf{c}}$;
  If $x_{\mathbf{c}}$ is not in convex hull $P_{\mathbf{c}}$, translate to nearest point in convex hull and update $S_{\mathbf{c}}$;
end while

Figure 5.5 shows the IOU values as bars for Deeplabv3+ [21] fine-tuned on noisy images (red), running MLHybrid with small cliques (green) and with big cliques (blue) on all the classes of the VOC dataset for the segmentation problem. The likelihood of a label on each pixel, required for our algorithm, is estimated using the score from the Deeplabv3+. We use the pre-trained version of Deeplabv3+ from [21]. Deeplabv3+ gives overall pixel accuracy of 82.79 and with MLHybrid we get pixel accuracy of 84.07 and 85.11 respectively for small and big cliques. Mean IOU values (three bars at the right end) are 0.544, 0.566, and 0.579 respectively. MLHybrid has been run with non-decomposable clique potentials and the same standard fixed hyper parameters on the VOC dataset.

The performance of MLHybrid improves with fine tuning of hyper parameters. Figure 5.3 shows the visual results on four pictures from the data set when the hyper parameters have been tuned. To show the extent of improvement we have also included in Figure 5.3 the MlHybrid output with the standard hyper parameters (standard-hyp). We have also included the IOU values in the images (upper left hand corner) corresponding to Deeplabv3+, MlHybrid (Big(concave))

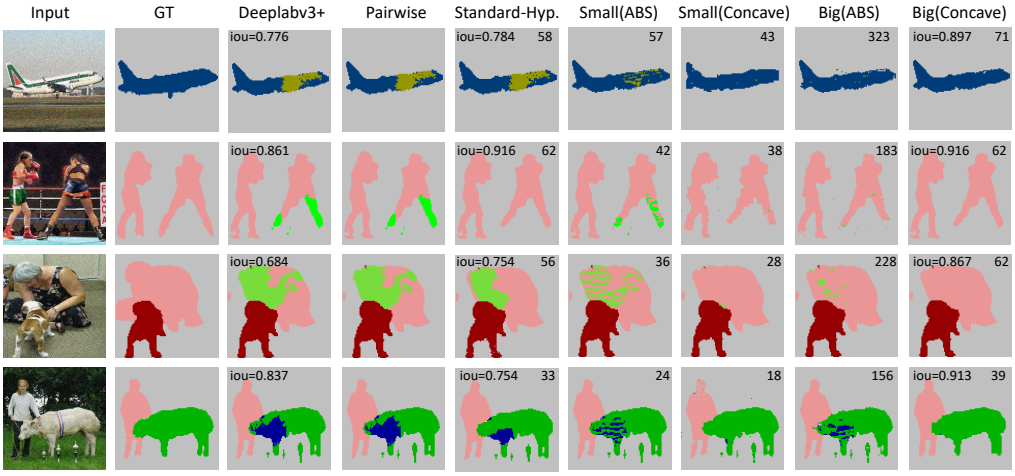

Figure 5.3: Pixel-wise object segmentation comparison. Input images from the Pascal VOC dataset.

run with standard and fine tuned hyper parameters respectively. For all the four images IOU values hover around 0.9 when MLHybrid is run with big cliques and concave potentials. Run time for MLhybrid in seconds are shown at the upper right corner of the respective images. Deeplabv3+ takes approximately 0.5 seconds per image excluding the training time. Hyper parameters for α-expansion running on pairwise cliques are the optimized parameters used for MLHybrid as are the likelyhood labels for the pixels.

Note that the quality of output is distinctly better for the non-decomposable concave potential in comparison to the decomposable ABS potential for both Small and Big clique configurations. The output for Big(Concave) matches the ground truth significantly. The time taken for concave potentials is distinctly less than ABS potentials for the same size and number of cliques. This difference is because the number of iterations taken for convergence is proportionately less for non-decomposable potentials. It is reasonable to infer that the segmentation quality improves with clique size. Since for large cliques, potentials will need to be predefined and not learnt, designing clique potentials calls for further investigation. Also, since fine tuning of hyper parameters improves quality of segmentation results significantly an area of research with high pay off is how to automate the process of fine tuning the hyper parameters for the segmentation problem.

For stereo correspondence, the images are from Middelbury dataset [84] and are of size 200×200. The cliques are generated, as earlier, using SLIC algorithm. Label likelihood is calculated using Birchfield/Tomasi cost given in [12]. There are 16 disparity labels considered and clique potential used is the same as for the segmentation problem. Figure 5.4 shows the output. We have com-

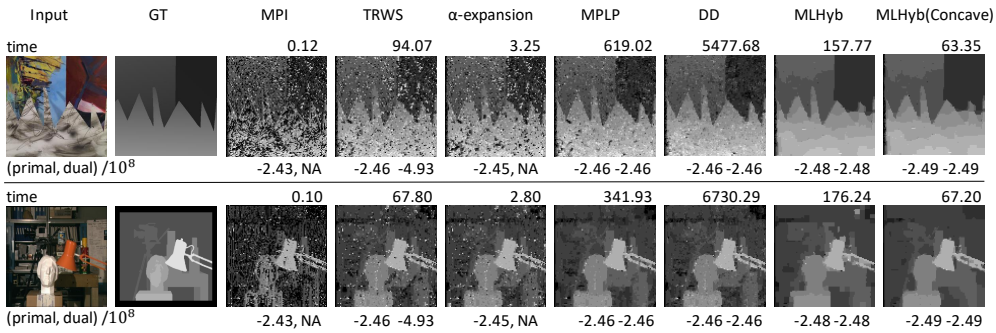

Figure 5.4: Stereo matching problem. Input images from the Middlebury dataset.

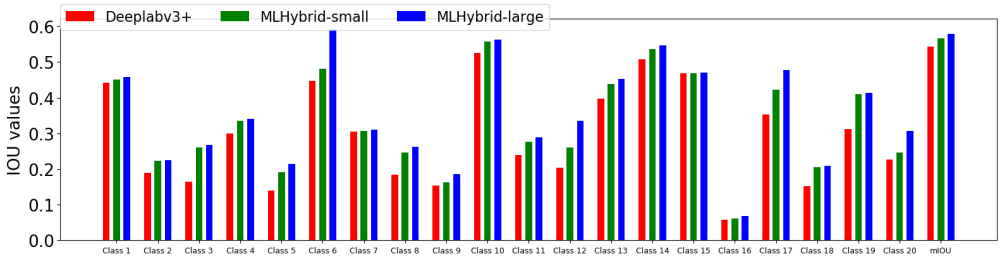

Figure 5.5: Shows IOU values across all the classes of PASCAL VOC dataset.

pared with implementations of Max Product Inference (MPI) [55], TRWS [56], MPLP [34], α-expansion [18] available in Darwin framework [36]. We use the pair wise absolute difference of labels potential with a pixel covered by maximum of four cliques. Other than α-expansion, other methods could not handle pairwise potentials emanating out of all pairs of variables in a clique of size 50 or larger.

Our final experiments are to show efficacy of convergence of the MLHybrid algorithm. Table 5.1 shows the performance of SOS-MNP [87] on the extended pseudo-boolean submodular function at different values of L. The number x in brackets stands for multiplication by 10^x. Note that primal and dual do not converge even when the value of L is as large as 10^{15} after running the algorithm for approximately 50 minutes. When L is small (i.e. $O(10^9)$) errors accumulate so fast that the primal value hovers around 10^{15}.

In contrast Figure 5.6 shows the convergence performance of the MLHybrid algorithm for solving a stereo problem on the sawtooth sample with sum of absolute difference potential. The figure shows that on the same potential function and same problem size, time taken for effective convergence by the MLHybrid

$L =$	10^9	10^{11}	10^{13}	10^{15}
Primal	1.26(15)	1.26(17)	−1.75(8)	−1.77(8)
Dual	−5.37(8)	−5.37(8)	−5.58(8)	−5.60(8)

Table 5.1: Primal dual for SoS-MNP for different values of L.

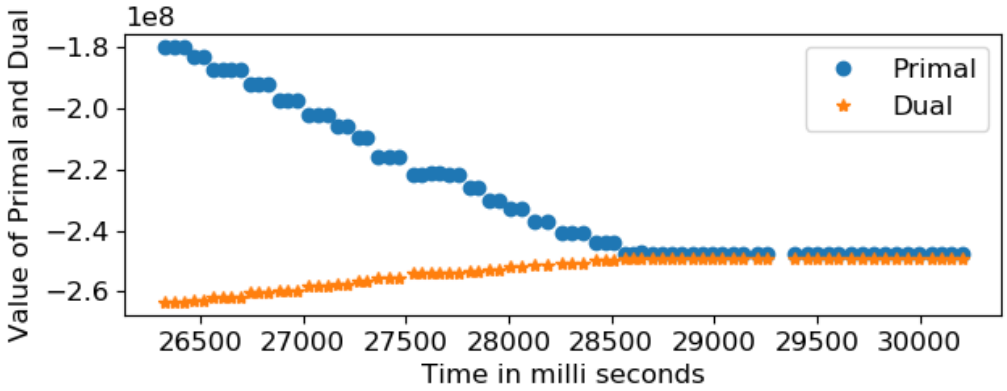

Figure 5.6: Convergence of MLHybrid.

algorithm is only around 28 seconds. It must be pointed out that one of the factors contributing to speed gain is the way invalid extreme bases are being handled. The flow graph created at each iteration handles a fixed number of (only $n(m-1)$) elementary extreme bases which span the space of all invalid extreme bases. The run-time at each iteration is essentially independent of the number of invalid extreme bases added by Wolfe's algorithm.

5.6 Discussion

The experiments show that MLHybrid enables solving multi-label problems optimally on large cliques (16 labels with clique sizes of up to 100) when potentials are submodular. This has been made possible by exploiting the structure of the potentials used to make the extension function submodular. Which allows the overall objective to decompose into two blocks of valid and invalid extreme bases. The valid block can be optimized by MNP algorithm. Further, we show that block of invalid extreme bases represented by a simple flow graph. The min ℓ_2-norm solution to the block of invalid extreme bases can be found by max

flow techniques on the flow graph. In principal the HybridMNGC algorithm developed in the previous chapter is applicable to the converted problem. Since the flow graph has simple structure which can be exploited, we have shown an efficient algorithm for computing min ℓ_2-norm of invalid block.

Chapter 6

Conclusion

In this thesis we have exploited polyhedron based submodular function minimization techniques for the structure present in MRF-MAP inference problems. We show that the dual of the problem can be decomposed into smaller size multiple subproblems which can be solved efficiently in block coordinate style. We have extended the limits of the clique size in 2-label MRF-MAP problems that could be handled by state of the art from 16 to 1000. We also show how to handle very large number of small clique and small number of large cliques efficiently. We combine the two seemingly far apart frameworks into a common framework to solve this problem. At last we show that ideas developed for 2-label problem can be applied to multi-label problem by transforming the multi-label problem into 2-label problem. We show how to avoid the computation over the large set of extra states introduced by the transformation. Our proposed inference method for multi-label MRF-MAP problem extends the limit of the maximum configuration handled by state of the art from 4-label 4-clique to 16-labels 100-clique.

6.1 Future Work

Pseudo boolean functions can be converted to piece-wise continuous functions using Lovasz extension [9]. This extension is convex if the original function is submodular otherwise it is non-convex. The advantage with continuous functions is that even for non-convex functions a local optimum can be found by doing gradient descent steps. The continuous extension can be applied to convert the 2-label MRF-MAP problem into a continuous optimization problem. Such problem will be the sum of smaller size continuous functions in which situation dual decomposition techniques can be applied on the dual of the problem. Note that such extensions can only be used to find approximate solutions for MRF-MAP problems with nonsubmodular potentials. Problems with multi-label nonsubmodular potentials can be solved by converting them into 2 label. In such converted problem, handling invalid states arising from the transformation is the subject of future investigation. Optimization method developed for non submodular po-

tentials are useful and have their applicability for scenario explained in the next paragraph.

The hybrid framework proposed in Chapter 4, for handling small and large cliques allows us to see the similarity between different frameworks proposed for MRF-MAP problems. It may be useful to model the problem with MRFs which has a mix of submodular and nonsubmodular potentials. Given that there exist frameworks for handling the submodular or non submodular potentials individually. If a mapping can be found between the variables of their corresponding frameworks then it will allow to fuse both the framework for handling respective clique potentials.

Now let us look at handling sparse potentials with some of the ideas developed for solving multi-label problem. If the useful states of the sparse potential given in a 2-label problem satisfies that if two states are useful then their intersection and union are also useful states (such group of useful states is called ring family [86]). Then The 2-label sparse potentials can be looked upon as the potentials obtained by transforming some multi-label potentials. Handling the invalid states in the context of multi-label problem can be looked upon as a way to optimize the problem which involves working on the useful sparse states only. Such methods for handling extra or unuseful states can be applied in those applications where sparse potentials are being used [81].

Our experiments have shown that both quality and time taken are a function of the potentials used. Choosing of the appropriate potential for the specific problem at hand is an important issue. This issue is all the more important because when cliques are of large size the potentials cannot be learnt. Not only there are no known techniques that ensure that the potential learnt is submodular, exponential size of the potential function table rules out use of learnt potentials. Potentials necessarily have to be predefined based on easily computable functions. It is also a fact different type of potentials are required for different applications. For example, smoothing concave potentials perform well in image segmentation tasks but may not be suitable for object detection tasks. Now that usefulness of large cliques has been established, designing suitable potential functions is a direction for future research.

Some of the recent works include doing MRF-MAP inference using neural networks [105]. In such works iteration of MAP inference are converted to the forward pass in a neural network. Neuralizing the MRF-MAP inference techniques is also important to learn better potentials as well as good approximations. Current works model the clique potential as a tensor of exponential size in terms of clique size (for general clique potentials). It is useful to ask if the exponential dependency on the size of network can be reduced for special clique potentials like, submodular clique potentials. An important notion in neural networks is the expressivity which is if a neural network is seen as a tensor product of the tensors then the rank of matrix obtained as the matricization of this

tensor is called expressivity. Since the range of submodular potentials spans a restricted space bounded by submodularity constraints. Restriction on the range space will affect the expressivity of the network to be low therefore a possibility of low rank decomposition arises for the tensors corresponding to submodular clique potentials.

Bibliography

[1] *Minimizing a sum of submodular functions*, Discrete Applied Mathematics, 160 (2012), pp. 2246 – 2258.

[2] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, and S. Süsstrunk, *Slic superpixels compared to state-of-the-art superpixel methods*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 34 (2012), pp. 2274–2282.

[3] A. M. Ali, A. A. Farag, and G. L. Gimel'Farb, *Optimizing binary MRFs with higher order cliques*, in European Conference on Computer Vision, Springer, 2008, pp. 98–111.

[4] C. Arora, *Primal dual algorithms for map inference in MRF labeling problems*, PhD thesis, 2012.

[5] C. Arora, S. Banerjee, P. Kalra, and S. Maheshwari, *An efficient graph cut algorithm for computer vision problems*, in European Conference on Computer Vision, Springer, 2010, pp. 552–565.

[6] C. Arora, S. Banerjee, P. Kalra, and S. N. Maheshwari, *Fast approximate inference in higher order MRF-MAP labeling problems*, in IEEE Conference on Computer Vision and Pattern Recognition, 2014, pp. 1338–1345.

[7] C. Arora, S. Banerjee, P. K. Kalra, and S. N. Maheshwari, *Generalized flows for optimal inference in higher order MRF-MAP*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 37 (2015), pp. 1323–1335.

[8] C. Arora and S. Maheshwari, *Multi label generic cuts: Optimal inference in multi label multi clique mrf-map problems*, in IEEE Conference on Computer Vision and Pattern Recognition, 2014, pp. 1346–1353.

[9] F. R. Bach, *Learning with submodular functions: A convex optimization perspective*, CoRR, abs/1111.6453 (2011).

[10] V. Badrinarayanan, A. Kendall, and R. Cipolla, *Segnet: A deep convolutional encoder-decoder architecture for image segmentation*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 39 (2017), pp. 2481–2495.

[11] J. Besag, *On the statistical analysis of dirty pictures*, Journal of the Royal Statistical Society. Series B (Methodological), (1986), pp. 259–302.

[12] S. Birchfield and C. Tomasi, *A pixel dissimilarity measure that is insensitive to image sampling*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 20 (1998), pp. 401–406.

[13] S. Birchfield and C. Tomasi, *Multiway cut for stereo and motion with slanted surfaces*, in IEEE International Conference on Computer Vision, vol. 1, 1999, pp. 489–495.

[14] E. Boros and A. Gruber, *On quadratization of pseudo-boolean functions*, arXiv preprint arXiv:1404.6538, (2014).

[15] E. Boros and P. L. Hammer, *Pseudo-boolean optimization*, Discrete applied mathematics, 123 (2002), pp. 155–225.

[16] Y. Boykov and V. Kolmogorov, *Computing geodesics and minimal surfaces via graph cuts*, in null, IEEE, 2003, p. 26.

[17] Y. Boykov, O. Veksler, and R. Zabih, *Markov random fields with efficient approximations*, in IEEE Conference on Computer Vision and Pattern Recognition (Cat. No. 98CB36231), 1998, pp. 648–655.

[18] Y. Boykov, O. Veksler, and R. Zabih, *Fast approximate energy minimization via graph cuts*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 23 (2001), pp. 1222–1239.

[19] Y. Y. Boykov and M.-P. Jolly, *Interactive graph cuts for optimal boundary & region segmentation of objects in nd images*, in Proceedings eighth IEEE international conference on computer vision., vol. 1, IEEE, 2001, pp. 105–112.

[20] D. Chakrabarty, P. Jain, and P. Kothari, *Provable submodular minimization using wolfes algorithm*, in Advances in Neural Information Processing Systems 27, 2014, pp. 802–809.

[21] L.-C. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam, *Encoder-decoder with atrous separable convolution for semantic image segmentation*, arXiv:1802.02611, (2018).

[22] S. Chen, H. Tong, and C. Cattani, *Markov models for image labeling*, Mathematical Problems in Engineering, (2012).

[23] J. Dahl and L. Vandenberghe, *Cvxopt*, 2007. http://mloss.org/software/view/34/.

[24] 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.

[25] O. Faugeras and R. Keriven, *Variational principles, surface evolution, PDE's, level set methods and the stereo problem*, IEEE, 2002.

[26] P. F. Felzenszwalb and D. P. Huttenlocher, *Efficient belief propagation for early vision*, International Journal of Computer Vision, 70 (2006), pp. 41–54.

[27] A. Fix, T. Joachims, S. Min Park, and R. Zabih, *Structured learning of sum-of-submodular higher order energy functions*, in Proceedings of the IEEE International Conference on Computer Vision, 2013, pp. 3104–3111.

[28] A. Fix, C. Wang, and R. Zabih, *A primal-dual algorithm for higher-order multilabel markov random fields*, in IEEE European Conference on Computer Vision, 2014, pp. 1138–1145.

[29] L. Fleischer and S. Iwata, *A push-relabel framework for submodular function minimization and applications to parametric optimization*, Discrete Applied Mathematics, 131 (2003), pp. 311–322.

[30] S. Fujishige, *Submodular functions and optimization*, vol. 58, Elsevier, 2005.

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

[32] S. Fujishige and S. Isotani, *A submodular function minimization algorithm based on the minimum-norm base*, Pacific Journal of Optimization, 7 (2011), pp. 3–17.

[33] S. Geman and D. Geman, *Stochastic relaxation, gibbs distributions, and the bayesian restoration of images*, IEEE Transactions on Pattern Analysis and Machine Intelligence, (1984), pp. 721–741.

[34] A. Globerson and T. S. Jaakkola, *Fixing max-product: Convergent message passing algorithms for map lp-relaxations*, in Advances in neural information processing systems, 2008, pp. 553–560.

[35] A. V. Goldberg, *Two-level push-relabel algorithm for the maximum flow problem*, in International Conference on Algorithmic Applications in Management, Springer, 2009, pp. 212–225.

[36] S. Gould, *Darwin: A framework for machine learning and computer vision research and development*, The Journal of Machine Learning Research, 13 (2012), pp. 3533–3537.

[37] D. Greig, B. Porteous, and A. H. Seheult, *Exact maximum a posteriori estimation for binary images*, Journal of the Royal Statistical Society. Series B (Methodological), (1989), pp. 271–279.

[38] D. M. Greig, B. T. Porteous, and A. H. Seheult, *Exact maximum a posteriori estimation for binary images*, Journal of the Royal Statistical Society: Series B (Methodological), 51 (1989), pp. 271–279.

[39] R. Hartley and A. Zisserman, *Multiple view geometry in computer vision*, Cambridge university press, 2003.

[40] H. Ishikawa, *Exact optimization for markov random fields with convex priors*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 25 (2003), pp. 1333–1336.

[41] H. Ishikawa, *Transformation of general binary MRF minimization to the first-order case*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 33 (2011), pp. 1234–1249.

[42] H. Ishikawa and D. Geiger, *Segmentation by grouping junctions*, in cvpr, vol. 98, Citeseer, 1998, p. 125.

[43] S. Iwata, *A faster scaling algorithm for minimizing submodular functions*, SIAM Journal on Computing, 32 (2003), pp. 833–840.

[44] S. Iwata, L. Fleischer, and S. Fujishige, *A combinatorial strongly polynomial algorithm for minimizing submodular functions*, Journal of the ACM (JACM), 48 (2001), pp. 761–777.

[45] S. Jegelka, F. Bach, and S. Sra, *Reflection methods for user-friendly submodular optimization*, in NIPS, 2013, pp. 1313–1321.

[46] H. W. Jensen, *Realistic image synthesis using photon mapping*, AK Peters/CRC Press, 2001.

[47] O. Juan and Y. Boykov, *Active graph cuts*, in IEEE Conference on Computer Vision and Pattern Recognition, vol. 1, 2006, pp. 1023–1029.

[48] O. Juan and Y. Boykov, *Capacity scaling for graph cuts in vision*, in IEEE International Conference on Computer Vision, 2007, pp. 1–8.

[49] Z. Kato, J. Zerubia, et al., *Markov random fields in image segmentation*, Foundations and Trends in Signal Processing, 5 (2012), pp. 1–155.

[50] P. Kohli, *Graph cuts for minimizing robust higher order potentials.*

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

[52] P. Kohli, L. Ladický, and P. Torr, *Robust higher order potentials for enforcing label consistency*, International Journal of Computer Vision, 82 (2009), pp. 302–324.

[53] P. Kohli and P. H. Torr, *Dynamic graph cuts for efficient inference in markov random fields*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 29 (2007), pp. 2079–2088.

[54] P. Kohli, P. H. Torr, et al., *Robust higher order potentials for enforcing label consistency*, International Journal of Computer Vision, 82 (2009), pp. 302–324.

[55] D. Koller, N. Friedman, and F. Bach, *Probabilistic graphical models: principles and techniques*, MIT press, 2009.

[56] V. Kolmogorov, *Convergent tree-reweighted message passing for energy minimization*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 28 (2006), pp. 1568–1583.

[57] V. Kolmogorov and C. Rother, *Minimizing nonsubmodular functions with graph cuts-a review*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 29 (2007), pp. 1274–1279.

[58] V. Kolmogorov and R. Zabih, *Computing visual correspondence with occlusions via graph cuts*, tech. rep., Cornell University, 2001.

[59] N. Komodakis and N. Paragios, *Beyond pairwise energies: Efficient optimization for higher-order MRFs*, in Computer Vision and Pattern Recognition, 2009, pp. 2985–2992.

[60] N. Komodakis, N. Paragios, and G. Tziritas, *MRF optimization via dual decomposition: Message-passing revisited*, in 2007 IEEE 11th International Conference on Computer Vision, IEEE, 2007, pp. 1–8.

[61] N. Komodakis and G. Tziritas, *Approximate labeling via graph cuts based on linear programming*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 29 (2007), pp. 1436–1453.

[62] N. Komodakis, G. Tziritas, and N. Paragios, *Fast, approximately optimal solutions for single and dynamic MRFs*, in IEEE Conference on Computer Vision and Pattern Recognition, 2007, pp. 1–8.

[63] A. Krizhevsky, I. Sutskever, and G. E. Hinton, *Imagenet classification with deep convolutional neural networks*, in Advances in neural information processing systems, 2012, pp. 1097–1105.

[64] X. Lan, S. Roth, D. Huttenlocher, and M. J. Black, *Efficient belief propagation with learned higher-order markov random fields*, in European Conference on Computer Vision, Springer, 2006, pp. 269–282.

[65] H. Langseth, *The hammersley-clifford theorem and its impact on modern statistics.*

[66] J. J. McAuley, T. S. Caetano, A. J. Smola, and M. O. Franz, *Learning high-order MRF priors of color images.*

[67] S. McCormick, *Submodular function minimization*, (2013).

[68] S. T. McCormick, *Submodular function minimization*, Handbooks in operations research and management science, 12 (2005), pp. 321–391.

[69] U. Mudenagudi, S. Banerjee, and P. K. Kalra, *Space-time super-resolution using graph-cut optimization*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 33 (2010), pp. 995–1008.

[70] A. C. Müller and S. Behnke, *pystruct - learning structured prediction in python*, JMLR, 15 (2014), pp. 2055–2060.

[71] K. Murota, *On steepest descent algorithms for discrete convex functions*, SIAM Journal on Optimization, 14 (2004), pp. 699–707.

[72] N. Komodakis, and G. Tziritas and N. Paragios, *Performance v/s computational efficiency for optimizing single and dynamic MRFs: Setting the state of the art with primal-dual strategies*, Computer Vision and Image Understanding, 112 (2008), pp. 14–29.

[73] A. Opelt, A. Pinz, M. Fussenegger, and P. Auer, *Generic object recognition with boosting*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 28 (2006), pp. 416–431.

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

[75] W. Pieczynski and A.-N. Tebbache, *Pairwise markov random fields and segmentation of textured images*, Machine Graphics and Vision, 9 (2000), pp. 705–718.

[76] B. Potetz and T. S. Lee, *Efficient belief propagation for higher-order cliques using linear constraint nodes*, Computer Vision and Image Understanding, 112 (2008), pp. 39–54.

[77] R. J. Qian and T. S. Huang, *Object detection using hierarchical MRF and MAP estimation*, in IEEE Conference on Computer Vision and Pattern Recognition, 1997, pp. 186–192.

[78] S. Ramalingam, C. Russell, L. Ladicky, and P. H. Torr, *Efficient minimization of higher order submodular functions using monotonic boolean functions*, arXiv preprint arXiv:1109.2304, (2011).

[79] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, *You only look once: Unified, real-time object detection*, in IEEE conference on computer vision and pattern recognition, 2016, pp. 779–788.

[80] S. Roth and M. J. Black, *Fields of experts*, International Journal of Computer Vision, 82 (2009), p. 205.

[81] C. Rother, P. Kohli, W. Feng, and J. Jia, *Minimizing sparse higher order energy functions of discrete variables*, in IEEE Conference on Computer Vision and Pattern Recognition, 2009, pp. 1382–1389.

[82] C. Rother, V. Kolmogorov, and A. Blake, *" grabcut" interactive foreground extraction using iterated graph cuts*, ACM transactions on graphics (TOG), 23 (2004), pp. 309–314.

[83] S. Roy and I. J. Cox, *A maximum-flow formulation of the n-camera stereo correspondence problem*, in Sixth International Conference on Computer Vision (IEEE Cat. No. 98CH36271), IEEE, 1998, pp. 492–499.

[84] D. Scharstein and R. Szeliski, *A taxonomy and evaluation of dense two-frame stereo correspondence algorithms*, International Journal of Computer Vision, 47 (2002), pp. 7–42.

[85] A. Schrijver, *A combinatorial algorithm minimizing submodular functions in strongly polynomial time*, Journal of Combinatorial Theory, Series B, 80 (2000), pp. 346–355.

[86] A. Schrijver, *Combinatorial Optimization - Polyhedra and Efficiency*, Springer, 2003.

[87] I. Shanu, C. Arora, and P. Singla, *Min norm point algorithm for higher order MRF-MAP inference*, in IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 5365–5374.

[88] J. Shotton, J. Winn, C. Rother, and A. Criminisi, *Textonboost: Joint appearance, shape and context modeling for multi-class object recognition and segmentation*, in European Conference on Computer Vision, Springer, 2006, pp. 1–15.

[89] D. Sontag, A. Globerson, and T. Jaakkola, *Introduction to dual decomposition for inference*, Optimization for Machine Learning, 1 (2011), pp. 219–254.

[90] P. Stobbe and A. Krause, *Efficient minimization of decomposable submodular functions*, in Advances in Neural Information Processing Systems, 2010, pp. 2208–2216.

[91] R. Szeliski, *Computer vision: algorithms and applications*, Springer Science & Business Media, 2010.

[92] M. F. Tappen and W. T. Freeman, *Comparison of graph cuts with belief propagation for stereo, using identical MRF parameters*, in null, IEEE, 2003, p. 900.

[93] A. Toshev and W. P. Examination II, *Submodular function minimization*, (2010).

[94] V. Kolmogorov, *Submodularity on a tree: Unifying ℓ_1-convex and bisubmodular functions*, in International Symposium on Mathematical Foundations of Computer Science, Springer, 2011, pp. 400–411.

[95] M. J. Wainwright, T. S. Jaakkola, and A. S. Willsky, *Map estimation via agreement on trees: message-passing and linear programming*, IEEE transactions on information theory, 51 (2005), pp. 3697–3717.

[96] C. Wang, N. Komodakis, and N. Paragios, *Markov random field modeling, inference & learning in computer vision & image understanding: A survey*, Computer Vision and Image Understanding, 117 (2013), pp. 1610–1627.

[97] C. Wang, O. Teboul, F. Michel, S. Essafi, and N. Paragios, *3d knowledge-based segmentation using pose-invariant higher-order graphs*, in International Conference on Medical Image Computing and Computer-Assisted Intervention, Springer, 2010, pp. 189–196.

[98] Y. Weiss and W. T. Freeman, *On the optimality of solutions of the max-product belief-propagation algorithm in arbitrary graphs*, IEEE Transactions on Information Theory, 47 (2001), pp. 736–744.

[99] T. Werner, *A linear programming approach to max-sum problem: A review*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 29 (2007), pp. 1165–1179.

[100] O. Woodford, P. Torr, I. Reid, and A. Fitzgibbon, *Global stereo reconstruction under second order smoothness priors*, in Proceedings of Computer Vision and Pattern Recognition, 2008, pp. 1–8.

[101] O. Woodford, P. Torr, I. Reid, and A. Fitzgibbon, *Global stereo reconstruction under second-order smoothness priors*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 31 (2009), pp. 2115–2128.

[102] J. Yang and T. Huang, *Image super-resolution: Historical overview and future challenges*, Super-resolution imaging, (2010), pp. 20–34.

[103] Y. Zeng, C. Wang, Y. Wang, X. Gu, D. Samaras, and N. Paragios, *Dense non-rigid surface registration using high-order graph matching*, in IEEE Conference on Computer Vision and Pattern Recognition, 2010, pp. 382–389.

[104] J. Zhang, J. Djolonga, and A. Krause, *Higher-order inference for multi-class log-supermodular models*, in Proceedings of the International Conference on Computer Vision, 2015, pp. 1859–1867.

[105] Z. Zhang, F. Wu, and W. S. Lee, *Factor graph neural network*, 2019.