Extending a Scaling based flow algorithm for higher order MRF-MAP inference in vision.

Student Name: Prasoon

IIIT-D-MTech-CS-11-001
June 15, 2016

Indraprastha Institute of Information Technology
New Delhi

Thesis Committee
Internal: Vikram Goyal
External: S.N. Maheshwari (IIT Delhi)
Advisor:Chetan Arora

Submitted in partial fulfillment of the requirements
for the Degree of M.Tech. in Computer Science

Certificate

This is to certify that the thesis titled **"Extending a Scaling based flow algorithm for higher order MRF-MAP inference in vision. "** submitted by **Prasoon** for the partial fulfillment of the requirements for the degree of *Master of Technology* in *Computer Science & Engineering* is a record of the bonafide work carried out by her / him under my / our guidance and supervision in the Security and Privacy group at Indraprastha Institute of Information Technology, Delhi. This work has not been submitted anywhere else for the reward of any other degree.

Chetan Arora
Indraprastha Institute of Information Technology, New Delhi

Abstract

Several tasks in computer vision and Machine Learning are modeled as MRF-MAP inference problems. The problem can often be modeled as minimizing sum of submodular functions. Since, sum of submodular functions (SoS) is also submodular, therefore, existing Submodular Function Minimization (SFM) techniques can be employed for MRF-MAP inference. SFM algorithms do not exploit the sum property and because of their high complexity, cannot be directly employed for the inference task. There have been attempts to exploit sum property in Schrijver's and Minnorm algorithms [8], with reasonable success. In this work we explore exploiting SoS property in an augmenting path SFM algorithm embedded in a scaling framework. The flow based SFM algorithm developed by Iwata, Fleischer, Fujishige [3] is weakly polynomial with $O(n^5 \log M)$ complexity. The proposed adapted algorithm has a complexity of $O(k.n^2.m^3 \log M)$ where k is the number of cliques, m is the size of largest clique. We have created an implementation of both non SoS and the proposed algorithm. The proposed algorithm shows significant improvement over the original (non SoS) one, though not as efficient as the other adaptations of Schrijver's and Min Norm Point.

Contents

List of Figures

Chapter 1

Introduction

A Markov Random field is a stochastic process associated with set of random variables that satisfies markov property described by an undirected graph . The undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ where $\mathcal{V} = 1, 2, ..., N$ is set of nodes, each of which is associated with random variable $, U_j$ for $j = 1, 2, ...N$. An edge (i, j) is present in $\mathcal{G}$ if the random variables U_i and U_j are directly correlated. Let $\mathbf{U}_S$ be the set of random variables associated with set of nodes S . Then, the markov property described by graph $\mathcal{G}$ is as follows :
Given disjoint subsets of nodes A, B and C, U_A is conditionally independent of U_B given U_C if there is no path from any node in A to any node in B that does not pass through a node in C. We write it as $U_A \perp\!\!\!\perp U_B \mid U_C$. If such a path does not exist in $\mathcal{G}$, then the random variables are dependent. In words there exist a one to one correspondence between the Graph and the conditional independence among the random variables. We define the neighborhood of node i, $\mathcal{N}_i$ as set of nodes adjacent to i . $\mathcal{N}_i$ is also known as Markov Blanket of node i . The property adhered by the distribution is

$$\mathcal{P}(U_i \mid \{U_j\}_{j \in \mathcal{V}-i}) = \mathcal{P}(U_i \mid \{U_j\}_{j \in \mathcal{N}_i}) \tag{1.1}$$

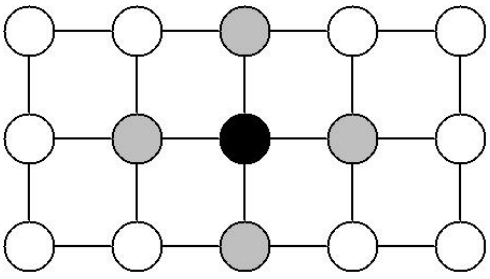

Figure 1.1: Given Grey nodes,Black node is conditionally independent of all other nodes.

1.0.1 Gibb's Distribution

Probability distribution $\mathbf{P}$ is called a Gibbs's distribution over $\mathcal{G}$ defined in above setting if it can be expressed in the form

$$\mathbf{P}(x) = \frac{1}{Z}\prod_{c\in\mathcal{C}}\phi_c(x_c) \tag{1.2}$$

Here ϕ_c is a positive function defined on maximal clique c .A clique is a subset of nodes in which every node is connected to every other node in $\mathcal{G}$.A maximal clique is a clique which cannot be extended. x_c is nodes in x that lies in clique c. (Note: Hereafter we will use nodes and Random Variables interchangeably) $\mathcal{C}$ is the set of all cliques in $\mathcal{G}$.
Z is called as partition function and has form

$$Z = \sum\prod_{c\in\mathcal{C}}\phi_c(x_c) \tag{1.3}$$

The ϕ_c is usually written as

$$\phi_c(x_c) = e^{-\frac{1}{T}V_c(x_c)} \tag{1.4}$$

T is called temperature and is often taken as 1. $P(x)$ thus is expressed as

$$P(x) = \frac{1}{Z}e^{-\frac{1}{T}\mathbb{E}(x)} \tag{1.5}$$

where

$$\mathbf{E}(x) = \sum_{c\in\mathcal{C}}V_c(x) \tag{1.6}$$

is known as energy function.
Hammersley-Clifford Theorem asserts that the process $\{X_i : i \in \mathcal{V}\}$ is a markov random field if and only if the corresponding distribution $\mathbf{P}$ is a Gibb's distribution. In other words, a probability distribution over random variables $(X = X_1, X_2, ..., X_n))$ that has a positive mass and satisfies (1) with respect to $\mathcal{G}$, then its density can be factorized over the cliques (or complete subgraphs) of $\mathcal{G}$.

1.0.2 MAP Problem

It stands for Maximum a posteriori. Given a distribution $\mathbf{P}$ defined over the set of random variables $X = \{X_1, X_2, ..., X_n\}$,

$$MAP(X) = \arg\max_x P(X = x) \tag{1.7}$$

In words, it is the assignment x for X whose probability is maximum. More specifically it is the configuration whose probability is maximum in the distribution.
In MRF, P is often expressed in terms of Energy function E (eqn 5). Now the assignment of X at which P is maximum, E will be minimum . So, MAP in MRFs is expressed as

$$MAP(X) = \arg\min_{x} E(X = x) \tag{1.8}$$

In case of MRF, we tend to minimize Energy function E rather than maximizing P as E is defined as sum whereas P is defined as product and hence P grows faster than E. **MAP** in **MRF** is also known as **Energy Minimzation Problem** .
In this work we are interested MRF-MAP inference in context of problems that appear in vision. It is shown that MRF with higher order clique potentials can encode complex dependencies between pixels [19,20,22,23,27,28] . There are plentiful techniques for Minimization of higher MRF potentials . Many of them are also applicable to optimization problem other than MRF. Gradient Descent is one such example. Some algorithms applicable only to MRF-MAP inference are Message passing [22, 24, 25], reduction techniques. In area such as Vision, there are problems like Image Denoising, Object Segmentation where each pixel can take some discrete values called labels. The domain of these labels are discrete and vary from problem to problem (It is different for Image denoising and Object Segmentation) . Our task is to find the true assignment of labels to each pixel (e.g. in Image Denoising we look for the assignment of labels to each pixel so that we get the original denoised image, in Object Segmentation we look for the assignment of labels to each pixel,that label the exact object to which each pixel belong in the image).

In such problems it is difficult to determine the true assignment because the pixels are largely affected by the other pixels in image and some other factors . So we go for a probabilistic framework where we make reasonable assumptions and try to find out the assignment that maximizes the probability under the given assumption. Markov random field turns out to be a good choice for modeling such problems.
In this distribution, a random variable is introduced for each pixel that can take any value from the list of available labels. We assume that only neighboring pixels affects each other. Again for a given pixel, you can have different number of neighbors. The choice of neighbors affecting the pixel is again a matter of choice. Graph consists of edges between two nodes if their corresponding pixels are neighbor and affects each other. All complete subgraphs in the graph constitutes a clique. A potential is assigned to each clique to completely define the distribution (see eqn 5 and 6). Certain submodular functions turns out to be a good choice for clique potential.

Given that sum of submodular is submodular, Energy Minimization in the above setting requires to minimize a submodular function.

Several optimization Problems in Machine Learning, computer vision and Natural Language Processing arise as a result of source of uncertainty that precludes the possibility of exact solution. Many such problems can be formulated as Energy Minimization Problem in MRF with clique potential as submodular functions.

There exists some algorithms that minimizes a submodular function. Applying such algorithms directly on these functions gives the point at which the function minimizes.But these algorithms does not exploit the fact that the function is actually sum of many small submodular functions.

In this report, we extend two general Submodular function minimization algorithms in Sum of Submodular settings

Chapter 2

Background

2.1 SUBMODULAR FUNCTIONS

We start with a general introduction on submodular function. Submodularity can be seen as discrete counterpart of convexity [10, 12, 13]. General background of submodular functions can be found in [9–11] Given a set V, a set function is a function that is defined over the subsets 2^v of a ground set $\mathbf{V}$. A submodular function f is a set function that satisfies

$$f(X) + f(Y) \geq f(X \cup Y) + f(X \cap Y) \tag{2.1}$$

where X, Y are subsets of V.
There are two natural polyhedra associated with submodular function f.

- *Submodular polyhedron* $\mathbf{P}(f)$.
  $\mathbf{P}(f) = \{ \text{ x} \in \mathbf{R}^V \mid \text{x}(S) \leq f(S) \forall S \subseteq V \}$
- It turns out that it is also useful to consider the face of $\mathbf{P}(f)$ for which $x(V) = f(V)$, which defines the *Base Polyhedron* $B(f)$.
  $B(f) = \{x \in P(f) | x(V) = f(V)\}$
  We will refer the elements of $B(f)$ as *bases*. *Base* is a point that lies on the polyhedron.

.

Lemma 2.1.1. *Given, a submodular function f and two sets $X1, X2$.*
If $x(X_i) = f(X_i), i = 1, 2$, then

$$f(X1 \cap X2) = x(X1 \cap X2) \tag{2.2}$$

$$f(X1 \cup X2) = x(X1 \cup X2) \tag{2.3}$$

Proof. From the definition of submodular polyhedron $\mathbf{P}(f)$, it is known that

$$x(X1 \cup X2) \leq f(X1 \cup X2)$$

Taking definition of submodularity

$$f(X1) + f(X2) \geq f(X1 \cap X2) + f(X1 \cup X2)$$

$$f(X1 \cap X2) \leq f(X1) + f(X2) - f(X1 \cup X2)$$

$$f(X1 \cap X2) \leq x(X1) + x(X2) - x(X1 \cup X2)$$

Since both inequalities exist therefore the equation follows as

$$f(X1 \cap X2) = x(X1 \cap X2)$$

Similarly for 11, we can prove the same.

□

At this point it is important to define the notion of $x(U)$ & $x^{-}(V)$.

$$x(U) = \sum_{u \in U} x(u), \forall U \subseteq V \tag{2.4}$$

$$x^{-}(V) = \sum_{v \in V, x(v) < 0} x(v) \tag{2.5}$$

It is worth noting that x^{-} captures the non-positive elements of a base x.

Min Max Theorem :*For a submodular function $f : 2^V \longrightarrow R$ holds*

$$max\{x^{-}(V) | x \in B(f)\} = min\{f(X) | X \subseteq V\} \tag{2.6}$$

Moreover, a minimizer A_o for f can be obtained from a maximizer $\widehat{x}$ over $B(f)$ as follows :

$$A_o = \{u \in V | \widehat{x}(u) < 0\} \tag{2.7}$$

Proof. The x in the equation (14) will generate the maximum value. Consider the u,$v \in$ V with $x(u) < 0$ and $x(v) > 0$ there exists a subset $X_{uv} \subseteq V \backslash$v such that $u \in X_{uv}$ and

$x(X_{uv}) = f(X_{uv})$. Now we prove the above statement. Consider the contradiction of the same. We know that the $x(X) < f(X)$ for all X defined as above. But this can't be true as we know that x is maximizer. Now we define a larger set X containing all the negative elements.

$$X = X_{uv}$$

The above equation can be made by union and intersection over all the sets which can be seen below.

$$X = \bigcup_{x(u)<0} \bigcap_{x(v)>0} X_{uv}$$

We know by lemma 2.1 that the below inequality will hold over intersections and unions.

$$x(X) = f(X)$$

Since it contains negative elements in V then $x^-(V) = f(X)$. □

Different SFM algorithms tends to increase x^- in order to minimize f.
Before digging into algorithm, its important to describe how these algorithms represents base x and what are the moves that can increase x^-.
A point on Base Polyhedron that cannot be described as a convex combination of other bases is called as *extreme base* .
Extreme bases can be generated using Edmond's greedy Algorithm. Given a total order $\prec$ of elements of V such that $\prec : v_1 \prec v_2 \prec ... \prec v_n$,if we denote extreme base as $b^{\prec}$ then it initializes $b^{\prec}(v_1) = f(V_1^{\prec})$ and remaining elements as $b^{\prec}(v_k) = f(V_k^{\prec}) - f(V_{k-1}^{\prec})$ (where $V_k^{\prec} = \{v_1, v_2, ..., v_k\}$) in ordering $\prec$.
Caratheodory's Theorem: The theorem states that if a point $x \in \mathbf{R}^n$ lies in the convex hull of a point set P, then there is a subset P' of P containing at most n+1 points such that x lies in the convex set P'. Now that we know how to generate extreme base, this theorem allows SFM algorithms to represent any base vector as a convex combination of at most n+1 extreme bases.
Formally, Base vector x is maintained as a convex combination of set of extreme bases $B = \{b^{\prec 1_1}, ..., b^{\prec_n}\}$ induced by a set of total orderings $\prec_1, ..., \prec_n$ of V :

$$x = \sum_{k=1}^{n} \lambda_k, \sum \lambda_k = 1, \lambda \geq 0 \tag{2.8}$$

Hereafter, we will use term SFM as abbreviation of submodular function minimization.

2.2 General Optimization Strategy

All SFM algorithms [14, 16] move one extreme point of the base polyhedron to an adjacent extreme point via an operation called exchange operation. Basically this operation increases base along one coordinate and decrease along others. One can attempt to iteratively maximize $x^-(V)$ - starting with an initial x, we maintain a representation for x as an affine combination of bases and try to optimize by modifying this set of bases. Intutively, one wants for a given x to increase some of the negative components while not making positive components negative.
For a given ordering $\prec$ and two consecutive elements v and u in this ordering, one can increase $b^{\prec}$ at u and decrease it at v by changing the order locally. We can apply this move to increase negative and decrease positive if u is positive and v is negative in $\prec$. Below Lemma specifies this move formally.

Definition 2.2.1. Suppose u immediately succeed v in an ordering $\prec$. Suppose further that $\prec'$ is obtained from $\prec$ by exchanging u and v. This operation is known as *Exchange operation* and the following relation holds

$$b^{\prec'} = b^{\prec} + C(\chi_u - \chi_v) \tag{2.9}$$

for a positive quantity C called Exchange capacity. Let S be the ordered set of elements that preceeds v in $\prec$. then $C = f(S \cup u) - f(S) - f(S \cup v) + f(S \cup v \cup u)$

Let $\prec$ be a total order on elements of V and $s, t \in V$. We denote by $(s, t]_{\prec}$ the set of elements which are between s and t(t inclusive) : $(s, t] = \{w \in V | s \prec w \preceq t\}$.
Let $u, v \in V$ and v $\prec$ u in ordering $\prec$.

SFM algorithms exploit these moves to increase $x^-(V)$, where $x \in B(f)$ and V represents the ground set over which submodular function f is defined.

Lemma 2.2.1. *Suppose that $U \subseteq V$ is the subset consisting of all elements which are negative or precede all negative elements for a given x:*

$$U = \{v \in V | \exists u \text{ with }, x(u) < 0 \text{ and an ordering } \prec \text{ with } v \preceq u\} \tag{2.10}$$

If U does not contain positive elements ,then U is a minimizer for f.

Proof. By the definition we know that, in every ordering first $|U|$ elements are negative. Consider there is a k such that $U = V_k^{\prec}$. By edmonds greedy algorithm.

$$b^{\prec}(U) = x(V_k^{\prec})$$

By definition of x as convex combination of orderings.

$$x(V_k^{\prec}) = f(V_k^{\prec}) = f(U)$$

Further, $f(U) = x(U)$ where U contains only negative elements. Therefore $x(U) = x^{-}(V)$. We know by min-max theorem that $x^{-}(V) \leq f(S)$ for all $S \subseteq V$. It shows U is minimizer of f. □

Hereafter, f represents the submodular function, V represents the set of elements over which f is defined, x represents the base and is a vector of dimension $|V|$ which contains scalar corresponding to every node. In an abuse of notation we also define a set function denoted by x such that $x(S) = \sum_{v \in S} x(v)$. $P(f)$ and $B(f)$ represents the *Submodular Polyhedron* and *Base Polyhedron* respectively. $\prec$ denotes a total order on elements of V such that: $\prec$: $v_1 \prec \ldots v_n$. There is an extreme base corresponding to every ordering $\prec_i$, denoted as b_i, and can be created using Edmond's greedy algorithm.

2.3 Iwata Algorithms

2.3.1 Iwata Weakly Polynomial Time Algorithm

This section discusses an augmenting path SFM algorithm embedded in scaling framework. Cunninghman was the first to use such approach [14] to minimize submodular function in separation problem of matroid polyhedra. This approach was further adapted by Bixby, Cunninghman and Topkis [15] for SFM and improved by Cunningham in [16]. This algorithm belongs to class of combinatorial optimization algorithms. The first combinatorial algorithm for SFM was due to Grotschel, L. Lovasz, and A. Schrijver [18]

To start with, we have a set of elements V, an integer valued submodular function f , an initial base x corresponding to f, I ectreme bases such that $x = \sum_{i \in I} \lambda_i b_i$, a complete graph G with node set V . Formally $G = (V, A)$ with arc set $A = (V \times V)$. $\varphi : V \times V \rightarrow \mathbb{R}$

denotes flow in G. The *boundary* $\partial\varphi : V \to \mathbb{R}$ is defined as:

$$\partial\varphi(u) = \sum_{v \in V} \varphi(u,v) - \sum_{v \in V} \varphi(v,u)$$

$\partial\varphi(u)$ denotes net flow emanating from u. Each arc in G have capacity δ, where δ is a scaling parameter. Scaling parameter δ in flow framework ensures that algorithm transfers a flow of magnitude at least δ from source to sink. S denotes source, T denotes sink and is formally defined as $S = \{v \mid x(v) + \partial\varphi \leq -\delta\}$, $T = \{v \mid x(v) + \partial\varphi \geq +\delta\}$. A flow φ in G is called δ-feasible if $\varphi(u,v) \leq \delta$, $\forall u, v \in V$. Initially, $\phi(u,v) = 0$ $\forall u, v \in V$ and $\delta = \xi/n^2$ with $\xi = min\{|x^-(V)|, x^+(V)\}$. Algorithm ensures that at least one of $\phi(u,v)$ and $\phi(v,u)$ is 0 $\forall u, v \in V$. We define $G^o = (V, A^o)$ where V denotes the set of vertices and arc set $A^o = \{(u,v) | u, v \in V, u \neq v, \varphi(u,v) = 0\}$. W represents the set of elements reachable from S in G^0. The concept of this graph G^0 was first introduced by Iwata [1] while designing scaling based flow algorithm for submodular flows [17] .W represents the set of elements reachable from S in G^0. Initially $W = V$. A triple (i,u,v) of $i \in I, u \in W$ and $v \in V - W$ is called *active* if u immediately succeeds v in $\prec_i$. We define a vector $z = x + \partial\varphi$. Algorithm seeks to increase z^- by sending flow from S to T. It then increases $x^-(V)$ via $\delta-$ feasibility of φ.

Now that we have described Graph (G), flow (ϕ), scaling parameter (δ), source (S), sink (T), we can describe scaling phase. Each δ-scaling phase maintains a δ-feasible flow φ and a subgraph G^o. It aims at increasing $z^-(V)$ by sending flow from S to T in G^o along directed paths. We call such paths as δ-augmenting paths. Once a δ-augmenting path P is found,the algorithm augments flow along P by setting $\varphi(u,v) = \delta - \varphi(v,u)$ and $\varphi(v,u) = 0$, for each (u,v) in P.

If there is no δ-augmenting path, but there is a tuple (i,u,v) that is active, the algorithm performs an exchange operation (refer Lemma 2.2.1) on (u,v) in ordering $\prec_i$. Let $x = \sum_i \lambda_i b_i$ be the current convex combination of solution vector x that algorithm maintains at all the times. Let $\varphi(u,v)$ be the flow in edge (u,v) in graph G. Let C be the exchange capacity corresponding to exchanging u and v in $\prec_i$, and $\prec_j$ be the ordering obtained after exchanging u and v in $\prec_i$. We create a new solution vector $x = \sum_i \lambda_i b_{\prec_i}$, where coefficient λ_i and λ_j are set as follows. We find the α , minimum of old $\lambda_i C$ and φ We then increase the base x by α along the vector $(\chi_u - \chi_v)$ and decrease $\varphi(u,v)$ by α. It then assigns coefficient $\lambda_j = \lambda_i - \alpha/C$ to b_j and updates the coefficient $\lambda_i = \alpha/C$ of b_i. The algorithm then modifies $\partial\varphi$ so as to keep $z = x + \partial\varphi$ unchanged. We denote above exchange operation as *Double-Exchange(i,u,v)*.

Note that a double exchange operation may lead to $\varphi(u,v)$ becoming zero, thus bringing

back an edge in the graph and creating new augmenting path. Therefore the algorithm performs δ-augmenting phase again after double-exchange. A δ-scaling phase ends when there is neither a δ-augmenting path nor an active triple.

Lemma 2.3.1. *At the end of the δ-scaling phase,$z^-(V) \geq f(W) - n\delta$*

Proof. At the end of the δ-scaling phase, there are no active triples. It means that for any element $u \in W$, there does not exist $v \in V - W$ for which $v \prec_i u$, otherwise a double exchange would have been possible. Hence in each ordering $i \in I$ the first $|W|$ vertices in $\prec_i$ is W. Hence, $b_i(W) = f(W)$ (all the terms in the sum for $b_i(W)$ will telescope and cancel out except for $f(W)$). Therefore, $\sum_i \lambda_i b_i(W) = f(W)$ where $\sum_i \lambda_i = 1$. Since, $x(W) = \sum_i \lambda_i b_i(W)$, Therefore:

$$x(W) = f(W) \tag{2.11}$$

$z(W)$ is the sum of positive and negative elements in W. Since all positive elements are atmost δ:

$$\begin{aligned} z(W) &< z^-(W) + |W|\,\delta \\ \Rightarrow \quad z^-(W) &> z(W) - |W|\,\delta. \end{aligned} \tag{2.12}$$

Similarly, since:

$$\begin{aligned} z(v) &> -\delta, \quad \forall v \in V - W, \\ \Rightarrow z(V - W) &> -|V - W|\,\delta \end{aligned} \tag{2.13}$$

Combining equations 2.12 and 2.13 with the definition: $z^-(V) = z^-(W) + z^-(V - W)$, we get:

$$\begin{aligned} z^-(V) &\geq z(W) - \delta|W| - \delta|V - W|, && (2.14)\\ &= z(W) - \delta(|W| + |V - W|) = z(W) - n\delta, && (2.15)\\ &= x(W) + \partial\varphi(W) - n\delta. && (2.16)\\ & && (2.17) \end{aligned}$$

At the end of a scaling phase $\partial\varphi(W) \geq 0$. Also, due to equation 2.12, $x(W) = f(W)$ at this stage. Putting the two in the equation above:

$$z^-(V) \geq f(w) - n\delta. \tag{2.18}$$

□

Lemma 2.3.2. *At the end of the δ-scaling phase,$x^-(V) \geq f(W) - n^2\delta$*

Proof. By Lemma 2.3.1, the set W satisfies $z^-(V) \geq f(W) - n\delta$.
Let $N = \{v|x(v) < 0\}$. Thus, N satisfies

$$\begin{aligned} x^-(V) &= x(N) \\ &= z(N) - \partial\varphi(N) \\ &\geq z^-(V) - \partial\varphi(N) \end{aligned} \tag{2.19}$$

Clearly $\partial\varphi(v) = \sum_u \phi(v,u) - \sum_u \phi(u,v)$ and v is connected to $n-1$ nodes in flow graph. Hence,

$$\begin{aligned} \partial\varphi(v) &\leq (n-1)\delta \\ \partial\varphi(N) &\leq n\partial\varphi(v) \\ \therefore \partial\varphi(N) &\leq n(n-1)\delta \\ \therefore x^-(V) &\geq z^-(V) - n(n-1)\delta \\ &\geq f(W) - n^2\delta \end{aligned} \tag{2.20}$$

□

2.3.2 Iwata Strongly Polynomial

We motivate the Strongly Polynomial version with the following lemma.

Lemma 2.3.3. *If at the end of δ-scaling phase, $x(w) > n^2\delta$ where $n = |V|$, then the element w can not be included in any minimizer of f. Similarly if $x(w) < -n^2\delta$, then w must be included in any minimizer of f.*

Proof. At the end of a δ-scaling phase, we obtain a set W such that:

$$\begin{aligned} x^-(V) &> f(W) - n^2\delta \\ \Rightarrow \quad x^-(V) - f(W) &> -n^2\delta. \end{aligned} \tag{2.21}$$

Moreover, for any minimizer Y:

Algorithm 1 IwataWeaklyPolynomial(f)

$L \leftarrow$ a ordering on V.
$x \leftarrow$ an extreme base in $B(f)$ generated by L.
$\delta \leftarrow min\{|x^-(v)|, x^+(v)\}/n^2$
$I \leftarrow \{k\}, y_k \leftarrow x, \lambda_k \leftarrow 1, L_k \leftarrow L$
$\varphi \leftarrow 0$,
while $\delta \geq 1/n^2$ **do**
  $S \leftarrow \{v | x(v) + \partial\varphi \leq -\delta\}$
  $T \leftarrow \{v | x(v) + \partial\varphi \geq -\delta\}$
  $W \leftarrow$ the set of vertices reachable from S in G^o.
  while $W \cap T \neq \phi$ or there is an active triple **do**
    while While $W \cap T \neq \phi$ and there is an active triple **do**
      Apply **Double-Exchange** to an active triple (i, u, v)
      Update W
    end while
    if $W \cap T \neq \phi$ **then**
      Augment flow φ along a $\delta-$ augmenting path P
      Update G^o, S, T, W
    end if
    Apply $Reduce(x, I)$
  end while
  $\delta \leftarrow \delta/2$
  $\varphi \leftarrow \varphi/2$
end while
Return W

Algorithm 2 Double-Exchange(i, u, v)

$c \leftarrow$ capacity for exchanging u and v in $\prec_i$
$\alpha \leftarrow min\{\varphi(u, v), \lambda_{ic}\}$
$\varphi(u, v) \leftarrow \varphi(v, u) - \alpha$
if $\alpha < \lambda_{ic}$ **then**
  Introduce new extreme base with a new index k
  $\lambda_k \leftarrow \lambda_i - \alpha/c$
  $\lambda_i \leftarrow \alpha/c$
  $b^{\prec_k} \leftarrow b^{\prec_i}$
  $\prec_k \leftarrow \prec_i$
end if
$b^{\prec_k} = b^{\prec_i} + c(\chi_u - \chi_v)$
Update $\prec_i$ by exchanging u and v.

$$\begin{aligned} f(W) &\geq f(Y) \geq x(Y) \\ &\geq x^-(Y). \end{aligned} \tag{2.22}$$

For an element $w \in V$ and $x(w) < 0$

$$\begin{aligned} x(w) &\geq x^-(V-Y) = x^-(V) - x^-(Y) \\ &\geq x^-(V) - f(W) && \text{(Applying equation 2.22)} \\ &> -n^2\delta. && \text{(Applying equation 2.21)} \end{aligned}$$

Therefore any element for which $x(w) < -n^2\delta$ must be in every minimizer of f.
Also, $f(W) \geq x(Y)$.

$$\begin{aligned} x^-(Y) &\geq x^-(V) \\ &\geq f(W) - n^2\delta && \text{(From Lemma 2.3.2)} \\ \because f(W) &\geq x(Y) \\ \therefore x^-(Y) &\geq x(Y) - n^2\delta \end{aligned} \tag{2.23}$$

$\because x^-(V) \leq 0, \therefore$ from equation 2.23 if there exists a w such that $x(w) > n^2\delta$, then that w cannot be in any minimizer Y. □

Lemma 2.3.4. *Suppose we run Scaling phase starting with $\delta = \eta$ and a base y such that $y(w) \leq \eta$ for all $w \in V$, then*

1. *If $f(V) \geq \eta$, then after $\lceil 3 \log n \rceil$ we obtain w such that $x(w) \geq n^2\delta$.*
2. *If $f(V) \leq \eta$, then after $\lceil 3 \log n \rceil$ we obtain w such that $x(w) \leq -n^2\delta$.*

Proof. Let δ_i denotes the capacity at the ith scaling phase and $k = \lceil 3 \log n \rceil$. Initially $\delta = \eta$ and $\delta_k \geq \eta/n^3$. x denotes the base obtained at the end of k^{th} scaling phase. At first consider the first case. Here we obtain an x such that $x(V) = f(V) > \eta > n^3\delta_k$. Hence, there is an element w for which $x(w) > n^2\delta_k$. In second case we obtain an x such that $x^-(Y) \leq f(Y) - \eta < -n^3\delta_k$. Hence there is an element w such that $x(w) < -n^2\delta_k$. (Iwata [2].) □

Lemma 2.3.4 is implemented as a procedure **FIX**. We define some notations which are used to describe this procedure. $S_{\min} \subseteq V$ contain elements that is guaranteed to be included in every minimizer of f. $S_{\max} \subseteq V$ contain elements that is guaranteed to

be not included in any minimizer of f. This strongly polynomial time algorithm keeps redefining your submodular function to be minimized and the set over which this function is defined. During the course of algorithm, this new function and set is denoted by $\hat{f}$ and U respectively . At that point, the algorithm seeks to minimize $\hat{f}$ over U. Vetrex set $U = \{u_1, u_2, ...u_l\}$ corresponds to partition $\{V_1, V_2, ...V_l\}$ of $V - (S_{\min} \cup S_{\max})$ into disjoint nonempty subsets. We define a function Γ that maps $Y \subseteq U$ to the original elements in V. Throughout the course of execution, algorithm maintains the correspondence between the minimizer of f and $\hat{f}$ so that any minimizer of f is represented as $S_{\min} \cup \Gamma(Y)$ for some minimizer Y of $\hat{f}$. Initially $U = V, \hat{f} = f$. $D = (U, F)$ is a directed acyclic graph where an edge (u, v) in F is an ordered pair of vertices u and v in U such that v is in every minimizer of $\hat{f}$ containing u. If there is a cycle in the graph, we merge all the nodes of a cycle into one supernode such that graph remains acyclic at all stages. $R(u)$ denote the set of vertices reachable from u in graph D. An extreme base $b_{\prec}$ is said to be consistent with D, if for any $v \in R(u)$, $v \prec u$. $\hat{f}_u$ denote the contraction of $\hat{f}$ by u . $\hat{f}_u$ denote the submodular function on the ground set $U - R(u)$ defined by:

$$\hat{f}_u(Y) = \hat{f}(Y \cup R(u)) - \hat{f}(R(u)), \forall Y \subseteq U - R(u) \tag{2.24}$$

Contraction and deletion of vertices changes U and equation 2.24 is the operation that redefines $\hat{f}$.
Before defining FIX, we need to know a base y and η such that $y(w) \leq \eta$ for all $w \in U$.

Lemma 2.3.5. *Any consistent extreme base $b \in B(\hat{f})$ satisfies $b(v) \leq \hat{f}(R(v)) - \hat{f}(R(v) - v)$ $\forall$ $v \in U$.*

Proof. Any v in the base $b \in B(\hat{f})$ satisfies

$$b(v) = \hat{f}(X) - \hat{f}(X - v) \text{ where } X\text{preceedes } v \text{ in } b$$
(From Edmonds Greedy Algorithm)

$$\therefore \hat{f}(X) - \hat{f}(X - v) \leq \hat{f}(R(v)) - \hat{f}(R(v) - v)$$
($\because b$ is consistent extreme base, $\therefore X \supseteq R(v)$ and $\hat{f}$ is submodular)

$$\therefore b(v) \leq \hat{f}(R(v)) - \hat{f}(R(v) - v)$$

□

Value of η is motivated by the above lemma. we set $\eta = \max\{\hat{f}(R(u)) - \hat{f}(R(u) - u)\}$. The definition of η can be re-written as $\eta = \max\{\hat{f}(R(u)) - \hat{f}(R(u) - u)\} = \max\{\hat{f}(U) - (\hat{f}(U) - \hat{f}(R(u))) - \hat{f}(R(u) - u)\}$. Therefore one of the following must hold:

1. $\hat{f}(U) \geq \eta/3$,
2. $-(\hat{f}(U) - \hat{f}(R(u))) \geq \eta/3 \Rightarrow \hat{f}(U) - \hat{f}(R(u)) \leq -\eta/3$ or
3. $-\hat{f}(R(u) - u) \geq \eta/3 \Rightarrow \hat{f}(R(u) - u) \leq -\eta/3$

As we show below, depending upon whichever case holds at the start of FIX, different guarantees hold at the end of FIX.

Lemma 2.3.6. *If $\hat{f}(U) \geq \eta/3$ then after $FIX(\hat{f}, D, \eta), \exists\ w \in U$ such that $x(w) > n^2\delta$. By Lemma 2.3.3, w must be excluded from any minimizer of f.*

Proof. Let δ denotes the delta at the end of FIX subroutine.

$$\begin{aligned} \because \delta &< \eta/(3n^3) \\ \implies n^3\delta &< \eta/3 \\ \because \hat{f}(U) &\geq \eta/3 \\ \implies \hat{f}(U) &> n^3\delta \\ \because \hat{f}(U) = \sum_{w \in U} x(w) & \\ \therefore \exists w, x(w) > n^2\delta & \end{aligned}$$

□

Lemma 2.3.7. *If $\hat{f}(U) - \hat{f}(R(u)) \leq -\eta/3$, then after $FIX((\hat{f}_u, D_u, \eta), \exists\ w$ such that $w \in R(u)$.*

Proof. Consider a submodular function $\hat{f}_u$, where $\hat{f}_u$ is $\hat{f}$ contracted by u. Let $U' = U - R(u)$. Therefore:

$$\hat{f}_u(U') = \hat{f}(U' + R(u)) - \hat{f}(R(u)) = \hat{f}(U) - \hat{f}(R(u)) \leq -\eta/3.$$

Let δ be the delta in the last phase. Therefore:

$$\delta < \eta/(3n^3) \Rightarrow -n^3\delta > -\eta/3.$$

Let $\hat{x}_u$ be the base that algorithm maintains while minimizing $\hat{f}_u$. At the end of scaling phase, $\sum_{w \in U'} \hat{x}_u(w) \leq \hat{f}_u(U') \leq -\eta/3$. Therefore there exists a w, such that, $\hat{x}_u(w) < -n^2\delta_k$. Thus, w lies in every minimizer of $\hat{f}_u$. For function $\hat{f}_u$ holds that if Y is a minimizer of $\hat{f}_u$ and $R(u)$ is in minimizer of $\hat{f}$, then $Y \cup R(u)$ is a minimizer of $\hat{f}$. Hence a minimizer that contains u must contain minimizer of $\hat{f}_u$. Hence $w \in R(u)$

□

Lemma 2.3.8. *If $\hat{f}(R(u)-u) < -\eta/3$ then after $FIX(\hat{f}, D, \eta)$, $\exists w \in R(u)-u$ such that $y(w) < -n^2\delta$. By Lemma 2.3.3, w must be included in any minimizer of f.*

Proof. Let δ be the delta in the last phase.

$$\begin{aligned} \because \delta &< \eta/(3n^3) \\ \implies -n^3\delta &> -\eta/3 \\ \because \hat{f}(R(u)-u) &< -\eta/3 \\ &< -n^3\delta \\ \therefore \exists w \in R(u)-u, x(w) &< -n^2\delta \end{aligned} \tag{2.25}$$

□

Algorithm 3 IwataStronglyPolynomial(f)

$X \leftarrow \phi, U \leftarrow V, \hat{f} \leftarrow f, F \leftarrow \phi$ {Initialization}
while $U \neq \phi$ **do**
  $\eta \leftarrow \max_u \left(\hat{f}(R(u)) - \hat{f}(R(u)-u)\right)$.
  Let $u \in U$ attain the maximum above.
  if $\eta \leq 0$ **then**
    $X \leftarrow X \cup \Gamma(U)$
    Return X
  else
    $X \leftarrow X \cup \Gamma(U)$
    Return X
  end if
end while

Lemma 2.3.9. *The number of augmentations per scaling phase is $O(n^2)$.*

Proof. It follows from Lemma 2.3.1 that at the beginning of each δ-scaling phase except for the first one, for some $X \subseteq V$, the below relation holds

$$z^-(V) \geq f(X) - 2n\delta$$

Now when the algorithm replaces φ by $\varphi/2$, it leads to decrease in $z(X)$. Lets $z_{reduced}$ denotes the decreased z. This decrease is $z(X)$ is at most $|X||V-X|\delta$.

$z^-(V)$ will also decrease.
Now,

$$\begin{aligned}
z(X) - z_{reduced}(X) &\leq |X||V-X|\delta \\
|X||V-X|\delta &\leq n^2\delta/4, \forall X \subseteq V \\
z^-_{reduced}(V) &\leq z_{reduced}(X) \\
z^-(V) &\leq z(X) \\
\therefore z^-(V) - z^-_{reduced}(V) &\leq n^2\delta/4, \forall X \subseteq V \\
\therefore f(X) - 2n\delta - n^2\delta/4 &\leq z^-_{reduced}(V) \\
\because x(X) &\leq f(X) \\
\& \ \partial\varphi(X) &\leq \delta|X||V-X| \\
\therefore z^-_{reduced}(V) &\leq z_{reduced}(X) \qquad (2.26) \\
&\leq f(X) + n^2\delta/4 \qquad (2.27)
\end{aligned}$$

Note that equation 2.27 holds throughout the δ-scaling phase.

Since each δ-augmentation increases $z^-(V)$ by δ, the number of δ-augmentations per phase is at most $2n + n^2/2$.

Now lets consider for the first scaling phase. Let x be the initial base or extreme base. $\because$ $z = x$ initially and $\delta = min\{|x^-(V)|, x^+(V)\}/delta$, z^- can increase by at most $min\{|x^-(V)|, x^+(V)\}$. An augmentation increases z^- by δ, $\therefore$ total number of augmentations in this phase is also $O(n^2)$.

□

Lemma 2.3.10. *Between δ-augmentation, number of new orderings increases by at most $n-1$.*

Proof. *Double-Exchange* may lead to addition of new ordering. Call such *Double-Exchange* as non-saturating *Double-Exchange*. Each non-saturating *Double-Exchange* leads to addition of new element in W. Addition of new element in W may connect some positive element to some negative element through a $\delta-$ augmenting path in flow graph G. In worst case algorithm can add at most n-1 vertices before discovering a δ-augmenting path. Hence number of new ordering increases by at most n-1. □

Lemma 2.3.11. *The algorithm performs the procedure* Double-Exchange $O(n^3)$ *times between δ-augmentation.*

Proof. Once the algorithm applies *Double-Exchange*,the vertices are interchanged in or-

dering i, and the triple (i, u, v) is no more active until definition of W changes. But definition of W changes only with a path augmentation. So, triple (i, u, v) is no more active until next augmentation or end of the phase. Reduce Base operation at the end of every augmentation ensures that algorithm always maintains n orderings at the begining of $\delta-$augmentation. Now, given that algorithm maintains n orderings at the starting of $\delta-$augmentation and from previous lemma number of new orderings may appear before $\delta-$ augmentation is at most n-1, total number of orderings in between $\delta-$augmentation is $2(n-1)$. Also there are at most $O(n^2)$ (u, v) pairs. Hence total number of such triples are at most $O(n^3)$ and in worst case *Double-Exchange* may end up in dealing with all such pairs before finding a $\delta-$augmenting path. □

Lemma 2.3.12. *The above algorithm is a strongly polynomial algorithm that performs* $O(n^7 log(n))$ *function evaluation and arithmetic operations.*

Proof. Each time the procedure FIX is called, the algorithm adds new arc to D or deletes a set of vertices. It can happen until there exists a pair between which an edge can be added or that can be deleted. Total number of such distinct pairs can be at most $O(n^2)$. So FIX can be called $O(n^2)$ times. Each call to FIX takes $O(log(n))$ phases and each phase has $O(n^2)$ augmentations, total number of path augmentation is at most $O(n^4 \log n)$. As per above lemma number of arithmetic operations and function evaluations per augmentation is $O(n^3)$,it yields an $O(n^7 log(n))$ bound on total number of steps. □

Chapter 3

Iwata Weakly Sum of Submodular

3.1 Introduction

In this setting we have $f(S) = \sum_C f_c(V_C \cap S)$, where f_c is function defined over clique c and V_c are the elements in the clique. Also, $V = \bigcup_C V_c$.

Lemma 3.1.1. *Let $x(S) = \sum_C x_C(C \cap S)$ where each x_C lies on the base polyhedron $B(f_C)$. Then x lies on the base polyhedron $B(f)$.*

Proof. At first we show that $x(U) \leq f(U) \forall U \subseteq V$.

$$\begin{aligned} x(U) &= \sum_C x_C(C \cap U) \\ &\leq \sum_C f_C(C \cap U) \\ &= f(U) \end{aligned}$$

Now we need to show $x(V) \leq f(V)$

$$\begin{aligned} x(V) &= \sum_C x_C(C \cap V) \\ &= \sum_C x_C(C) \\ &= \sum_C f_c \\ &= f(V) \end{aligned}$$

□

We used to have two graphs in case of original Iwata, a flow graph G and a directed, unweighted graph G_0. G used to be a complete graph in original Iwata. But here, in G only elements of cliques are connected such that every clique in G is a complete subgraph. In an abuse of notation, we use $G_{extended}$ to denote this new G. G_0, in original Iwata defined over V consist of those edges of G that have zero flows. Here we denote G_0 with $G_{0_{extended}}$.
We maintain ordering ($\prec_{k,c}$), extreme base ($b^{\prec_{k,c}}$) and base (x_c) corresponding to each clique, a base x as defined by lemma 3.1.1. Like original Iwata it searches for the δ augmenting path in $G_{0_{extended}}$ and augments path in $G_{extended}$. Definition of S, T and W remains same, i.e. S is defined as $\{v|x(v)+\partial\varphi \leq -\delta\}$, T is defined as $\{v|x(v)+\partial\varphi \geq -\delta\}$ and W is defined as the set of vertices reachable from S in $G_{0_{extended}}$.

But unlike original Iwata where a scaling phase continues until all the $|W|$ elements in the starting of every ordering of all extreme bases respects W (i.e. all the $|W|$ elements in the starting of every ordering is W), here in SoS, a scaling phase continues until all the extreme bases in all clique honors W (i.e. If let W_c denotes the $W \cap c$ for clique c, then all the extreme bases of clique c should have W_c in starting of their respective ordering). At first we describe the algorithm and then we will analyze the algorithm. We describe the algorithm as follows:

3.2 Analysis

Lemma 3.2.1. *At the end of the δ-scaling phase,$z^-(V) \geq f(W) - n(n+1)\delta/2$*

Proof. At the end of the δ-scaling phase, there are no active triples. Let W_c for a clique c denotes $W \cap c$. It implies that in all clique c, all the first $|W_c|$ elements in all the orderings are W_c, otherwise double exchange would have been possible. Hence, $b_{i,c}(W) = f_c(W)$ (all the terms in the sum for $b_{i,c}(W_c)$ will telescope and cancel out except for $f_c(W)$). Therefore, $\sum_i \lambda_{i,c} b_{i,c}(W_c) = f_c(W_c)$ where $\sum_i \lambda_{i,c} = 1$. Since, $x_c(W_c) = \sum_i \lambda_{i,c} b_{i,c}(W_c)$, Therefore:

$$x_c(W_c) = f_c(W_c) \tag{3.1}$$

and

$$x(W) = \sum_c x_c(W \cap c) = \sum_c x_c(W_c) = \sum_c f_c(W_c) = f(W) \tag{3.2}$$

At the end of δ-scaling phase, W also satisfies $S \subseteq W \subseteq V - T$ and $\partial\varphi(W) \geq 0$. Note that at the end of scaling-phase,no element of T is reachable from any element of S. It means no element v for which $z(v) \geq \delta$ is reachable from S. Since W denotes set of elements

Algorithm 4 IwataWeaklySoS($\{f_c\}$)

$L_c \leftarrow$ a ordering on V_c $\forall c \in C$.
$x_c \leftarrow$ an extreme base in $B(f_c)$ generated by L_c $\forall c \in C$.
$x = \sum_C x_C$
$\delta \leftarrow min\{|x^-(v)|, x^+(v)\}/n^2$
$I \leftarrow \{k\}, y_k \leftarrow x, \lambda_k \leftarrow 1, L_k \leftarrow L$
$\varphi \leftarrow 0$,
while $\delta \geq 1/n^2$ **do**
  $S \leftarrow \{v | x(v) + \partial\varphi \leq -\delta\}$
  $T \leftarrow \{v | x(v) + \partial\varphi \geq -\delta\}$
  $W \leftarrow$ the set of vertices reachable from S in $G_{0_{extended}}$.
  while $W \cap T \neq \phi$ or there is a $c \in C$ with an active triple **do**
    while While $W \cap T = \phi$ and there is a clique c with an active triple **do**
      Apply **Double-Exchange** to an active triple (i, u, v) in clique c
      Update W
    end while
    if $W \cap T \neq \phi$ **then**
      Augment flow φ along a $\delta-$ augmenting path P
      Update $G_{0_{extended}}, S, T, W$
    end if
    Apply $Reduce(x_c, I)$ $\forall c \in C$
  end while
  $\delta \leftarrow \delta/2$
  $\varphi \leftarrow \varphi/2$
end while
Return W

Algorithm 5 Double-Exchange(i, u, v)

$C \leftarrow$ capacity for exchanging u and v in $\prec_{i,c}$
$\alpha \leftarrow min\{\varphi(u, v), \lambda_{i,c}C\}$
$\varphi(u, v) \leftarrow \varphi(v, u) - \alpha$
if $\alpha < \lambda_{i,c}C$ **then**
  Introduce new extreme base with a new index k
  $\lambda_{k,c} \leftarrow \lambda_{i,c} - \alpha/C$
  $\lambda_{i,c} \leftarrow \alpha/C$
  $b^{\prec_{k,c}} \leftarrow b^{\prec_{i,c}}$
  $\prec_{k,c} \leftarrow \prec_{i,c}$
end if
$b^{\prec_{k,c}} = b^{\prec_{i,c}} + C(\chi_u - \chi_v)$
Update $\prec_{i,c}$ by exchanging u and v.

reachable from S and can not contain any v with $z(v) \geq \delta$, since those will definitely be in T. Therefore $z(v) < \delta, \forall v \in W$. By a similar reasoning $z(v) > -\delta, \forall v \in V - W$.

$z(W)$ is the sum of positive and negative elements in W. Since all positive elements are atmost δ:

$$\begin{aligned} z(W) &< z^-(W) + |W|\,\delta \\ \Rightarrow \quad z^-(W) &> z(W) - |W|\,\delta. \end{aligned} \tag{3.3}$$

Similarly, since:

$$\begin{aligned} z(v) &> -\delta, \quad \forall v \in V - W, \\ \Rightarrow z(V - W) &> -|V - W|\,\delta \end{aligned} \tag{3.4}$$

Combining equations 3.3 and 3.4 with the definition: $z^-(V) = z^-(W) + z^-(V - W)$, we get:

$$z^-(V) \geq z(W) - \delta|W| - \delta|V - W|, \tag{3.5}$$

$$= z(W) - \delta(|W| + |V - W|) = z(W) - n\delta, \tag{3.6}$$

$$= x(W) + \partial\varphi(W) - n\delta. \tag{3.7}$$

$$\because \partial\varphi(W) \geq -\delta|W||V - W|$$

$$\geq -n(n-1)\delta/2 \tag{3.8}$$

$$\therefore z^-(V) \geq f(W) - n(n+1)\delta/2 \tag{3.9}$$

□

Lemma 3.2.2. *At the end of the δ-scaling phase,$x^-(V) \geq f(W) - n^2\delta$*

Proof. By Lemma 3.2.1, the set W satisfies $z^-(V) \geq f(W) - n\delta$.
Let $N = \{v|x(v) < 0\}$. Thus, N satisfies

$$x^-(V) = x(N)$$

$$= z(N) - \partial\varphi(N)$$

$$\geq z^-(V) - \partial\varphi(N) \tag{3.10}$$

$$\because \partial\varphi(N) \leq \delta|N||V - N|$$

$$\leq n(n-1)\delta/2 \tag{3.11}$$

$$\therefore x^-(V) \geq z^-(V) - n(n-1)\delta \tag{3.12}$$

$\therefore$ from equation 3.12 and lemma 3.2.1, $x^-(V) \geq f(W) - n^2\delta$ $\square$

Lemma 3.2.3. *The number of augmentations per scaling phase is $O(n^2)$.*

Proof. It follows from Lemma 2.3.1 that at the beginning of each δ-scaling phase except for the first one, for some $X \subseteq V$, the below relation holds

$$z^-(V) \geq f(X) - n(n+1)\delta$$

Now when the algorithm replaces φ by $\varphi/2$, it leads to decrease in $z(X)$. Lets $z_{reduced}$ denotes the decreased z. This decrease is $z(X)$ is at most $|X||V - X|\delta$.

$z^-(V)$ will also decrease.
Now,

$$\begin{aligned}
z(X) - z_{reduced}(X) &\leq |X||V - X|\delta \\
|X||V - X|\delta &\leq n^2\delta/4, \forall X \subseteq V \\
z^-_{reduced}(V) &\leq z_{reduced}(X) \\
z^-(V) &\leq z(X) \\
\therefore z^-(V) - z^-_{reduced}(V) &\leq n^2\delta/4, \forall X \subseteq V \\
\therefore f(X) - n(n+1)\delta - n^2\delta/4 &\leq z^-_{reduced}(V) \\
\because x(X) &\leq f(X) \\
\& \ \partial\varphi(X) &\leq \delta|X||V - X| \\
\therefore z^-_{reduced}(V) &\leq z_{reduced}(X) \qquad (3.13) \\
&\leq f(X) + n^2\delta/4 \qquad (3.14)
\end{aligned}$$

Since each δ-augmentation increases $z^-(V)$ by δ, the number of δ-augmentations per phase is at most $n(n+1)\delta + n^2\delta/4$.

Now lets consider for the first scaling phase. Let x be the initial base or extreme base. $\because$ $z = x$ initially and $\delta = min\{|x^-(V)|, x^+(V)\}/delta$, z^- can increase by at most $min\{|x^-(V)|, x^+(V)\}$. An augmentation increases z^- by δ, $\therefore$ total number of augmentations in this phase is also $O(n^2)$. $\square$

Lemma 3.2.4. *The algorithm obtains a minimizer of f at the end of $\delta-$scaling phase with $\delta < 1/n^2$*

Proof. For W at the end of this scaling phase, we have

$$\begin{aligned} x^-(V) &\geq f(W) - n^2\delta \\ &> f(W) - 1 \\ \because x^-(V) &< f(U), \forall U \subseteq V \\ \therefore f(W) - 1 &< f(U), \forall U \subseteq V \end{aligned}$$

Hence W is a minimizer of f. □

Lemma 3.2.5. *Between δ-augmentation, number of new orderings in clique c increases by at most $|V_c|$ where V_c denote the elements in clique c.*

Proof. Double-Exchange may lead to addition of new ordering in clique c. Call such *Double-Exchange* as non-saturating *Double-Exchange*. Since each non-saturating *Double-Exchange* leads to addition of new ordering and expansion of W. Thus there are at most $|V_c|$ applications of nonsaturating *Double-Exchange* between augmentations. Hence the number of indices added between augmentations is at most $|V_c|$. □

Lemma 3.2.6. *The algorithm performs the procedure* Double-Exchange $O(km^3)$ *times between δ-augmentation where k is the number of cliques and m is size of the largest clique.*

Proof. In clique c with V_c nodes, each triple (i_c, u, v) can be active at most once between augmentation. There can be at most $O(|V_c|^2)$ (u, v) pairs and from lemma 3.2.5, i_c can grow at most $|V_c|$ times. If k is the number of cliques and m is size of the largest clique.k is the number of cliques and m is size of the largest clique then total number of *Double-Exchange* possible between $\delta-$augmentation is $O(k.m^3)$. □

Lemma 3.2.7. *Iwata weakly SoS SFM is a polynomial time algorithm that performs $O(k.n^2.m^3.\log M)$ function evaluations and arithmetic operations where k denotes number of cliques and m denotes size of largest clique.*

Proof. From Lemma 3.2.3, a scaling phase can have at most $O(n^2)$ augmentations. Also from Lemma 3.2.6, it can have at most $O(k.m^3)$ Double-Exchange between two augmentations. Hence, total number of function evaluations and arithmetic operations in a scaling phase is $O(k.n^2.m^3)$. $\log M$ represents the scaling phase when $\delta < 1/n^2$. Thus, algorithm performs $O(k.n^2.m^3.\log M)$ function evaluations and arithmetic operations □

Chapter 4

Experiments

All three algorithms i.e. Iwata Weakly without SoS,Iwata Strongly and Iwata Weakly SoS have been implemented in Visual C++ in windows7. All the experiments have been performed in workstation with 3.0 GHZ processor and 8 GB RAM.

Problem Size				
Algorithm	10	30	60	100
Iwata Weakly(without SoS)	1 sec	3 sec	98 sec	3901 sec
Iwata Strongly(without SoS)	1.5 sec	39 sec	1204 sec	4801 sec

Below experiment is done on synthetic problem. In these problems, we have a set of elements. Number of elements in the set is mentioned in problem size in the table below. Elements in the set is distributed among different cliques. Number of cliques in the problem is mentioned in the table below. The elements are assigned to the clique randomly. Cliques are overlapping and are of random sizes. We mention cliques of maximum size only in the table below.

we set unaries in a range such that neither unary cost nor clique potential alone decides the minimizer. The choice of function representing clique potential in real image while working with binary image segmentation and in this test example is same. In binary image segmentation, unary potentials of pixels lies in the range [-125, 125]. We set unaries randomly in this example in the range that appears in binary image segmentation.

Problem Size	num of cliques	max Clique size	Time(SoS)	Time(nonSoS)
100	97	4	30 sec	2876 sec
100	93	8	178 sec	3209 sec
200	197	4	214 sec	4135 sec
200	193	8	621 sec	4689 sec

Below is an example of Interactive object Segmentation. This problem is modeled as MRF-MAP inference problem and energy function in this binary case can be modeled as Sum of Submodular functions. Since sum of submodular functions is submodular, energy minimization in the above case turns out to be Submodular function minimzation problem. Since this submodular function has sum property, we apply the Iwata Weakly SoS to exploit this structure. The input is an image of size 18×24 and 15×20.

Figure 4.1: Input: 18×24 avg clique size: 20

Figure 4.2: Unary: 18×24

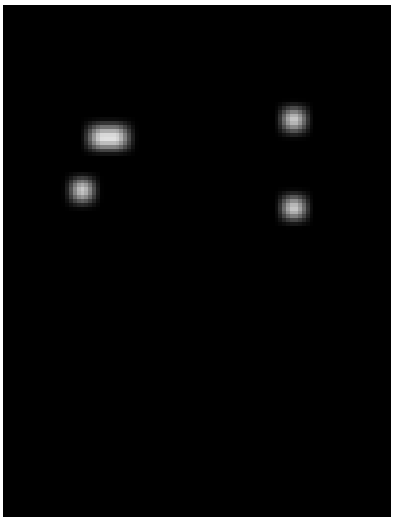

Figure 4.3: Result time (sec): 3891

Figure 4.4: Input: 15 × 20
avg clique size: 12

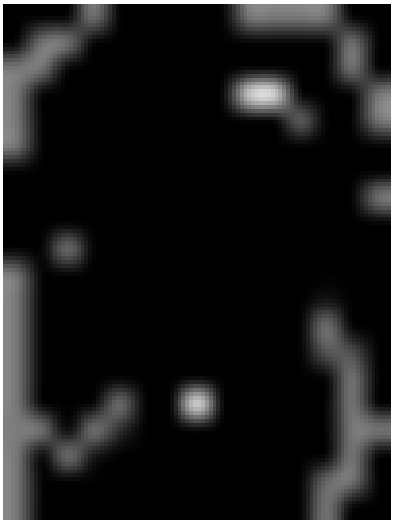

Figure 4.5: Unary: 15 × 20

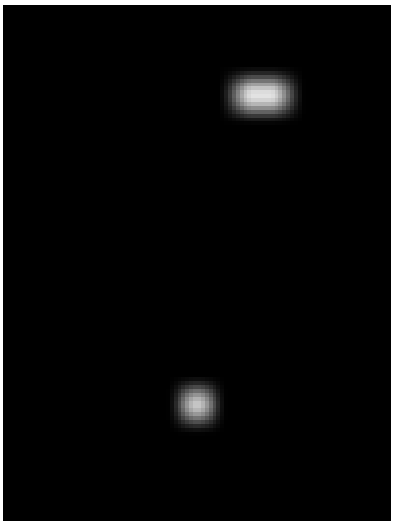

Figure 4.6: Result time (sec): 1801

Below experiment is done on the same problem and same input as above. But here,unlike previous case where we were halving the flow after each scaling phase, we are cutting down the flow by 1/4 after each scaling phase. In this setting, number of scaling phases decrease but path augmentation per scaling phase increases. Convergence of algorithm is guaranteed in this case.

Figure 4.7: Input: 18 × 24
avg clique size: 20

Figure 4.8: Unary: 18 × 24

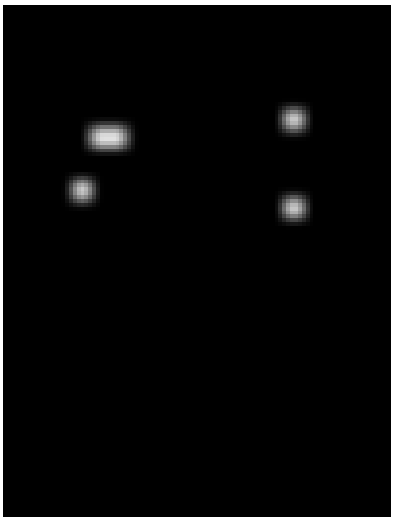

Figure 4.9: Result time (sec): 4235

Figure 4.10: Input: 15 × 20 avg clique size: 12

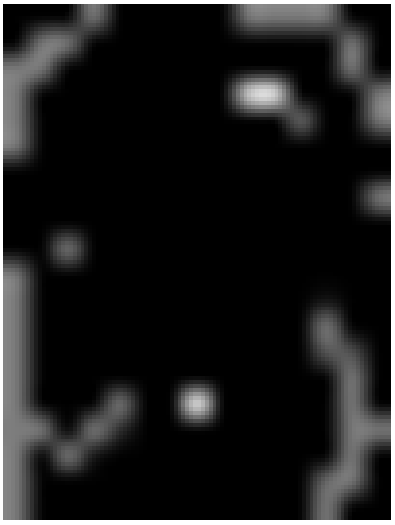

Figure 4.11: Unary: 15 × 20

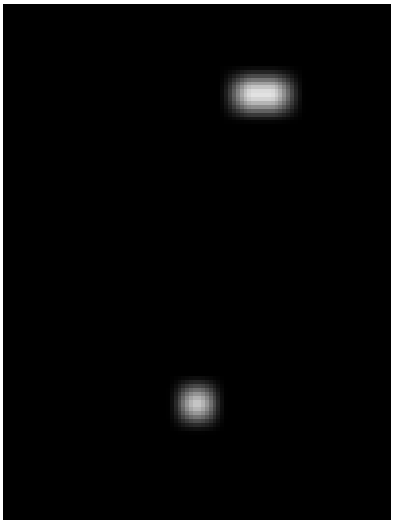

Figure 4.12: Result time (sec): 2092

Bibliography

[1] S. Iwata: A capacity scaling algorithm for convex cost submodular flows, *Math. Programming*, **76** (1997), 299 − 308.

[2] S. Iwata: A fully combinatorial algorithm for submodular function minimization, *J.Combinatorial Theory*

[3] Submodular Function Minimization
Alexander Toshev Written Preliminary Examination II
Computer and Information Science, University of Pennsylvania, Philadelphia

[4] S. Iwata, S. T. McCormick, and M. Shigeno: A strongly polynomial cut canceling algorithm for the submodular flow problem, *Proceedings of the Seventh MPS Conference on Integer Programming and Combinatorial Optimization* (1999), 259 − 272.

[5] A. Schrijver: A combinatorial algorithm minimizing submodular functions in strongly polynomial time, *J. Combinatorial Theory*, **B80** (2000), 346 − 355.

[6] L. Fleischer, S. Iwata, and S. T. McCormick: A faster capacity scaling algorithm for submodular flow, *Math. Programming*

[7] L.Fleischer and S. Iwata: improved algorithms for submodular function minimization and submodular flow, *Proceedings of the 32th Annual ACM Symposium on Theory of Computing,* (2000) 107 − 1

[8] Min Norm Point Algorithm for Higher Order MRF-MAP Inference. *Ishant Shanu Chetan Arora Parag Singhla, CVPR 2016*

[9] A. Frank and E. Tardos: Generalized polymatroids and submodular flows, *Math. Programming,***42** (1988), 489 − 563

[10] L. Lovasz: Submodular functions and convexity.*Mathematical Programming The State of the Art,* A. Bachem, M. Grotschel and B. Korte, eds., Springer-Verlag, 1983, 235-257

[11] S. Fujishige: *Submodular Functions and Optimization,* North-Holland, 1991.

[12] S. Fujishige: Theory of submodular programs: A Fenchel-type min-max theorem and subgradients of submodular functions, Math. Programming, **29** (1984), 142 − 155.

[13] A. Frank: An algorithm for submodular functions on graphs, *Ann. Discrete Math.,* **16** (1982), 189 − 212

[14] W. H. Cunningham: Testing membership in matroid polyhedra, *J. Combinatorial Theory,* **B36** (1984), 161 − 188.

[15] R. E. Bixby, W. H. Cunningham, and D. M. Topkis: Partial order of a polymatroid extreme point, *Math. Oper. Res.,* **10** (1985), 367 − 378.

[16] W. H. Cunningham: On submodular function minimization, *Combinatorica*, **5** (1985), 185 − 192.

[17] J. Edmonds and R. Giles, "A min-max relation for submodular functions on graphs, Studies in Integer Programming", *Proceedings of Workshop on Programming, Bonn, 1975. Ann. Discrete Math.* **1** (1977) 185 − 204

[18] Grotschel, L. Lovasz, and A. Schrijver. The ellipsoid method and its consequences in combinatorial optimization. *Combinatorica*, 1(2):169 − 197, 1981

[19] H. Ishikawa. Transformation of general binary MRF minimization to the first-order case. *IEEE TPAMI*, 33(6):123-1249,2011

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

[21] V. Kolmogorov. A new look at reweighted message passing. IEEE TPAMI, 37(5):919 − 930, 2015

[22] Komodakis and N. Paragios. Beyond pairwise energies: Efficient optimization for higher-order MRFs. *In Proc. of CVPR*, pages 2985 − 2992, 2009

[23] S. Roth and M. J. Black. Fields of experts. *IJCV*, 82(2):205229, 2009.

[24] V. Kolmogorov. A new look at reweighted message passing. *IEEE TPAMI*, 37(5):919-930, 2015.

[25] S. Jegelka, F. Bach, and S. Sra. Reflection methods for userfriendly submodular optimization. In Proc. of NIPS, pages 1313-1321, 2013

[26] Primal-dual message-passing for approximate inference. *Information Theory*, 56(12):6294-6316, 2010. 1

[27] T. Hazan and A. Shashua. Norm-product belief propagation: C. Rother, P. Kohli, W. Feng, and J. Jia. Minimizing sparse higher order energy functions of discrete variables. In *Proc. of CVPR*, pages 1382-1389, 2009.

[28] O. Woodford, P. Torr, I. Reid, and A. Fitzgibbon. Global stereo reconstruction under second order smoothness priors. *In Proc. of CVPR*, pages 18, 2008. 1