Generalized Flows for Optimal Inference in Higher Order MRF-MAP

Chetan Arora Subhashis Banerjee Prem Kalra S.N. Maheshwari

Abstract—Use of higher order clique potentials in MRF-MAP problems has been limited primarily because of the inefficiencies of the existing algorithmic schemes. We propose a new combinatorial algorithm for computing optimal solutions to 2 label MRF-MAP problems with higher order clique potentials. The algorithm runs in time $O(2^k n^3)$ in the worst case (k is size of clique and n is the number of pixels). A special gadget is introduced to model flows in a higher order clique and a technique for building a flow graph is specified. Based on the primal dual structure of the optimization problem, the notions of the capacity of an edge and a cut are generalized to define a flow problem. We show that in this flow graph, when the clique potentials are submodular, the max flow is equal to the min cut, which also is the optimal solution to the problem. We show experimentally that our algorithm provides significantly better solutions in practice and is hundreds of times faster than solution schemes like Dual Decomposition [1], TRWS [2] and Reduction [3], [4], [5]. The framework represents a significant advance in handling higher order problems making optimal inference practical for medium sized cliques.

Index Terms—Markov Random Field (MRF), Maximum a posteriori (MAP), Higher Order Cliques, Optimal Inference

1 Introduction

Many problems in computer vision, statistical mechanics, natural language processing, protein chain placements etc. can be formulated as computation of minimum energy configurations. Historically, the first formulation of the energy minimization in the context of labeling problems in computer vision is due to Geman and Geman [6]. Assuming the labeling to be a Markov Random Field (MRF), finding a labeling configuration with Maximum a Posteriori Probability (MAP) can be formulated as:

$$E(\mathbf{l}_{\mathcal{P}}) = \min_{\mathbf{l}_{\mathcal{P}}} \left[\sum_{p \in \mathcal{P}} D_p(l_p) + \sum_{\mathbf{c} \in \mathcal{C}} W_{\mathbf{c}}(\mathbf{l}_{\mathbf{c}}) \right], \qquad (1)$$

where l_p denotes the label at pixel p. A *clique*, defined as the set of pixels whose labels are contextually dependent on each other, is denoted as $\mathbf{c}$. $\mathbf{l}_{\mathbf{c}}$ denotes a labeling configuration on clique $\mathbf{c}$. $\mathcal{C}$ denotes the set of all cliques $\mathbf{c}$. The first term, $D_p(l_p)$, also called the *data energy* or the *data term*, measures the cost of assigning label l_p to p. The term measures how good is the labeling with respect to the observed data. The second term, $W_{\mathbf{c}}(\mathbf{l}_{\mathbf{c}})$, called the *prior energy*, measures the cost of the labeling configuration $\mathbf{l}_{\mathbf{c}}$ of a clique $\mathbf{c}$ depending on how consistent the labeling is with respect to the prior knowledge. The penalty function, $W_{\mathbf{c}}(\cdot)$, is also called the clique potential function. The formulation as described in Eq. 1 is often referred to as MRF-MAP.

Over the last two decades computer vision researchers have focused on both MRF-MAP based modeling and algorithms for optimizing the resultant energy functions. Vision problems that have been formulated in the MRF-MAP framework have ranged from image restoration [6], segmentation of videos [7] and images [8], super resolution [9], texture synthesis [10], stereo matching [11] to object detection [12]. Research in algorithmic techniques has been influenced largely by the observation that while the general MRF-MAP optimization problem is NP-Hard, for 2-label 2-clique submodular potentials, the optimization problems have strongly polynomial time optimal algorithms [13]. This has initiated a new research area in which the focus has been to extend the class of energy functions for which either there are efficient optimal solutions or there are suboptimal solutions with well defined approximation guarantees. Submodular potentials are of particular interest because while real life problems involve non-submodular potentials, combinatorial techniques for handling non-submodularity so far have involved making some form of submodular approximation.

Our focus in this paper is on MRF-MAP labeling problems with 2-label and cliques of size more than 2. There have been essentially two lines of approach to deal with such problems.

- Message Passing or Decomposition Approaches: These techniques combine ideas from gradient based optimization [14], [15], belief propagation [16] or primal-dual based methodology of dual decomposition [17], [14], [18]. While convergence can in some cases be guaranteed for algorithms

- *Chetan Arora was with The Hebrew University of Jerusalem, Israel and is now with the Indraprastha Institute of Information Technology (IIIT) Delhi.*
- *Subhashis Banerjee, Prem Kalra and S.N. Maheshwari are with the Indian Institute of Technology (IIT) Delhi*

based on these ideas, it is only in the *limit* (if the algorithm is run for arbitrarily long time until convergence) and is not necessarily to the optimal solution.

- Reduction Based Approaches: These algorithms reduce the original problem to a sequence of 2-label 2-clique problems which are solved in an iterative manner by direct combinatorial algorithms. Reduction to 2-clique was first proposed by Kolmogorov and Zabih [19]. Since then the idea has attracted considerable attention [3], [4], [5], [20], [21], [22], [23]. Since reductions do not in general preserve submodularity, use of QPBO algorithm [24] has been advocated to provide a submodular approximation to the reduced problem.

The main problems with the reduction and decomposition based approaches are:

- Performance: Reduction based algorithms are inefficient because of the exponential number of terms that are added and the additional conversion time required to transform from the cost per labeling format to the polynomial form required for reduction. In decomposition based algorithms the cost of calculating messages increases exponentially with clique size and there is no fast definitive convergence towards agreement between the solutions of the decomposed problems.
- Quality: Reduction approaches may leave some or all nodes unlabeled depending upon problem specification. In the decomposition based approaches, even if the decomposed problems are solved optimally, the problem of how to combine the solutions to solve the original primal is not trivial.
- Approximation Factor: Neither reduction nor decomposition based approaches guarantee any bound on the approximation factor in a fixed number of steps. This is true even when the potential function is submodular for which algorithms for finding optimal solutions are known.

There have been several attempts to address the above drawbacks. Rother et al. [20] exploit sparsity of preferred labelings by creating submodular deviation functions for which, in some situations, the reduction algorithm creates compact but non-submodular quadratic forms. Trying to come up with novel reduction techniques which reduce the additional auxiliary variables introduced is another line of research that is being followed [4], [22]. There have also been attempts to generalize techniques like roof duality for submodular relaxations to higher order terms [5], [25]. While for cubic potentials generalized roof duality approximations can be obtained by solving a series of LP problems [5], for quartic potentials this gets limited to using only a subset of quartic submodular functions as not all submodular quartic polynomials can be reduced to an equivalent quadratic form.

The work reported here has been influenced by the realization that it has been known for a decade that submodular set functions can be minimized in strongly polynomial time [26], [27] and that MRF-MAP energy function minimization problem, when clique potentials are submodular, is essentially minimizing a sum of a set of submodular functions. However, the most efficient algorithm for minimizing a submodular function has $O(n^6)$ time complexity [26]. This makes direct use of these algorithms impractical for computer vision problems where n, which represents the number of pixels, can easily reach millions. Kolmogorov [28] has reported a scaling based scheme for minimizing sum of submodular functions using [29]. The algorithm is essentially of only theoretical interest as implementation overheads are very high. Development of efficient practical polynomial time algorithms for solving energy minimization problems involving higher order cliques with submodular potentials therefore is an important open problem. The problem is also important because, as mentioned earlier, non-submodular function energy optimization problems are solved by computing submodular approximations.

We report here an optimal algorithm for 2-label multi-clique energy minimization problems with submodular potentials which runs in $O(n|\mathcal{C}|^2k^32^k)$ steps, where n is the number of pixels, $|\mathcal{C}|$ is the number of cliques and k is the size of the maximal clique. As in the 2-clique version of the problem in which the energy minimization (the primal problem) can be viewed as a min cut and the dual as a max flow problem, we show that these concepts can be extended to higher order cliques ($k > 2$) also. The dual framework has resulted in a novel flow problem in which both the capacity of an edge and the cost of a cut have new but natural generalizations. We show that the proposed gadget based flow graph solves a relaxed form of the submodular higher-order MAP-MRF problem, in which flow augmentation is used to iteratively tighten the relaxation and guarantee that the optimal solution can be obtained in polynomial time. We call our algorithm Generic Cuts (GC). Compared to the Dual Decomposition based algorithms [17], and TRWS [2], GC is hundreds of times faster. Also, like the max flow based graph cut optimizer for second order potentials, the optimal solution provided by our algorithm is integral which LP based algorithms like [14] do not guarantee.

An earlier version of the work has appeared in [30]. The current paper contains a more comprehensive treatment with additional experiments and comparison with newer methods like [4], [5] that have appeared since then.

The organization of the paper is as follows. In Section 2 and 3 we develop the basic primal dual framework and the gadget which is used to model a clique in the flow graph. Section 4 and 5 describe important theoretical properties of the framework. In Section 6 we give the formal algorithm along with its complexity and convergence guarantees. Section 7 contains the experiments conducted for comparing the performance of our algorithm with the current state of the art. We conclude the paper with indications of future directions in Section 8.

2 Primal Dual Schema and Flow Interpretation of the Dual

The LP formulation for MRF-MAP labeling problem given below follows [31], [32]. Any pixel can take a label l from the set $\mathcal{L} = \{a, b\}$ of possible labels. $\mathbf{l_{c,p,l}}$ is a labeling configuration on clique $\mathbf{c}$ in which the label of pixel p is l. Note that there can be many such labelings (corresponding to the same clique or the other cliques containing p), and the set of all such labelings is denoted as $\{\mathbf{l_c}\}_{p,l}$. We introduce binary variables X_p^l for all combinations of pixels and labels. X_p^l takes value 1 whenever pixel p is assigned label l and is 0 otherwise. Similarly, binary variables $Y_\mathbf{c}^{\mathbf{l_c}}$ take value 1 whenever clique $\mathbf{c}$ is assigned labeling configuration $\mathbf{l_c}$ and is 0 otherwise. Let $W_\mathbf{c} : \mathcal{L}^k \to \mathbb{R}$ be the clique potential function giving the penalty of labeling pixels of clique $\mathbf{c}$ by $\mathbf{l_c}$. The MRF-MAP equation (1) can be equivalently written as the following integer program:

$$\min_{X_p^l, Y_\mathbf{c}^{\mathbf{l_c}}} \sum_{p\in\mathcal{P}} \sum_{l\in\mathcal{L}} D_p(l) X_p^l + \sum_{\mathbf{c}\in\mathcal{C}} \sum_{\mathbf{l_c}\in\mathcal{L}^k} W_\mathbf{c}(\mathbf{l_c}) Y_\mathbf{c}^{\mathbf{l_c}} \tag{2}$$

subject to

$$\sum_{l\in\mathcal{L}} X_p^l = 1, \qquad p \in \mathcal{P}, \tag{3}$$

$$\sum_{z\in\{\mathbf{l_c}\}_{p,l}} Y_\mathbf{c}^z = X_p^l, \qquad p\in\mathcal{P},\ l\in\mathcal{L}, \tag{4}$$

$$X_p^l \in \{0,1\} \quad , \quad Y_\mathbf{c}^{\mathbf{l_c}} \in \{0,1\}. \tag{5}$$

Equation (3) ensures that each pixel is assigned exactly one label, and (4) enforces consistency between pixel and clique labelings. Replacing (5) by (6) we get a relaxed LP formulation of the optimization problem.

$$X_p^l \ge 0 \quad , \quad Y_\mathbf{c}^{\mathbf{l_c}} \ge 0. \tag{6}$$

The Lagrangian dual of the relaxed LP can be written as:

$$\max_U \sum_{p\in\mathcal{P}} U_p \tag{7}$$

subject to

$$U_p \le h_p^l, \qquad p\in\mathcal{P},\ l\in\mathcal{L}, \tag{8}$$

where

$$h_p^l = D_p(l) + \sum_{\mathbf{c}:p\in\mathbf{c}} V_{\mathbf{c},p,l}, \tag{9}$$

and

$$\sum_{p\in\mathbf{c}} V_{\mathbf{c},p,\mathbf{l_c^p}} \le W_\mathbf{c}(\mathbf{l_c}), \qquad \mathbf{c}\in\mathcal{C},\ \mathbf{l_c}\in\mathcal{L}^k, \tag{10}$$

where $\mathbf{l_c^p}$ denote the label of pixel p in labeling $\mathbf{l_c}$. Note that $\mathbf{l_c^p} \in \mathcal{L}$. Complimentary slackness conditions can be written as

$$X_p^l > 0 \qquad \Rightarrow \qquad U_p = h_p^l, \tag{11}$$

and

$$Y_\mathbf{c}^{\mathbf{l_c}} > 0 \qquad \Rightarrow \qquad \sum_{p\in\mathbf{c}} V_{\mathbf{c},p,\mathbf{l_c^p}} = W_\mathbf{c}(\mathbf{l_c}). \tag{12}$$

We assume that the cost of assigning uniform labeling (all a's or all b's) to a clique is zero. The constraints to model this are

$$\sum_{p\in\mathbf{c}} V_{\mathbf{c},p,l} = 0, \qquad \mathbf{c}\in\mathcal{C},\ l\in\mathcal{L}. \tag{13}$$

This assumption is not restrictive as we show later that when costs are submodular, uniform labeling costs in a clique can be normalized through reparametrization to an equivalent state where they are zero.

Primal and dual solutions are called feasible if they satisfy equations (3,4,6) and (8,10) respectively. A primal solution is called integral if it satisfies equation (5) as well. The primal dual framework guarantees that any feasible primal and dual solution which satisfies all complimentary slackness conditions (11,12,13) is optimal. In what follows we propose a framework for higher order clique MRF-MAP problems which maintains, at all stages, feasible primal and dual solutions. When clique potential functions are submodular, we show that at termination all complimentary slackness conditions are satisfied thereby guaranteeing the optimality of the solution. The algorithm maintains an integral primal at all stages ensuring that the solution is optimal for the original integer program version of the primal.

The primal-dual framework developed here is very similar to the one introduced by Komodakis and Tziritas [33]. We use their model of balls and wells to motivate flow interpretation of the dual. Let us assume a well corresponding to every pixel in which balls representing labels a and b float. The ball a in the well corresponding to pixel p (or simply well p) is represented by p^a and it floats at height h_p^a in the well. Since the dual is a maximization problem and U_p has to be less than both h_p^a and h_p^b (see equation (8)), it can be set equal to the height of the lower of the two balls in the well. If U_p is equal to, say, h_p^a in the well corresponding to p, then setting the primal variable X_p^a to 1 (i.e. pixel p is assigned label a) keeps all complementary slackness conditions of type (11) satisfied. We call the ball a as *active* in the well p in such a scenario. In other words any dual optimization strategy which chooses the lower ball as active keeps equations (8) and (11) satisfied. Any change in the value of variable $V_{\mathbf{c},p,a}$ impacts the height of the ball a in the well p and may change the ball with minimum height, thereby forcing a change in U_p (equal to the height of minimum ball). The above strategy of increasing the value of variables U essentially ensures that variables V can be treated as the only free variables in the dual optimization problem.

Consider a clique $\mathbf{c_1}$ in which the labeling associated with its pixels satisfies the uniform labeling constraint given in equation (13). Let p and q be two wells in $\mathbf{c_1}$. Consider the operation on dual variables in which the ball p^a decreases its height by reducing the variable $V_{\mathbf{c_1},p,a}$ by δ without affecting the relative ordering of the balls in the well p (i.e. lower ball stays lower than the higher ball). To continue satisfying the constraint given in equation (13) requires increase in the value(s) of other V variables associated with clique $\mathbf{c_1}$. One possibility is to increase the variable $V_{\mathbf{c_1},q,a}$ corresponding to q^a by the same amount. In effect the height of p^a decreases and that of q^a increases. We may view this change in heights of p^a and q^a as a consequence of sending δ flow from p^a to q^a in clique $\mathbf{c_1}$. Note that while decreasing the value of variable $V_{\mathbf{c_1},p,a}$ fixes the well of flow origin in clique $\mathbf{c_1}$, the destination well could have been any of the other wells in clique $\mathbf{c_1}$. Consider another clique $\mathbf{c_2}$ containing wells q and r. If we further decrease the variable $V_{\mathbf{c_2},q,a}$ and increase $V_{\mathbf{c_2},r,a}$ by δ, then the combined effect will be that q^a remains at the same height while the height of p^a decreases by δ and that of r^a increases by δ. We can view this operation as a flow corresponding to ball a of amount δ originating at well p and ending at well r along a *path* consisting of "*edges*" in cliques $\mathbf{c_1}$ and $\mathbf{c_2}$ passing through well q.

The dual objective function requires maximization of the sum of U_p over all pixels. The above discussion suggests that this would involve raising the heights of active balls in the wells. This can be achieved by sending flow to an active ball (in the destination well) from a non active ball (in the source well) with the same label. The non active ball in the source well comes down by an amount equal to the flow sent and the active ball in the destination well increases its height by the same amount. All other balls on the path from the source to the destination well maintain their height. How much flow can be sent is a function of the relative heights of the balls in the source and destination wells in question, and the constraints defined by equation (10) containing V variables associated with edges on the flow path. In the destination well, the active ball should not float at a height higher than the non active ball (of the same well) when the flow is sent. Whereas in the source well, the height of the non active ball should not become less than that of the active ball (of the same well). This ensures that the active ball configuration remains feasible. Changes in the value of V variables associated with the *edges* along the path should ensure that no constraint given by equation (10) becomes infeasible.

Note that the algorithmic move in terms of flow pushing is constrained by both primal and dual optimization problems. Flow impacts the dual, and what balls are active at any time is controlled by the primal. Since this flow move can be between any two a balls or b balls it would seem that the equivalent flow graph should allow for both types of flows. We show in the next section that when potentials are submodular then the equivalent flow graph on which the max flow problem has to be solved needs to cater for flows in between balls of only one label. This simplifies the optimization problem significantly.

3 Flow Graph Construction

The primary problem in generalizing the flow graph construction used to solve a 2-label 2-clique problem to a 2-label multi-clique case lies in modeling of flow edges along which flow would take place between pairs of wells in a clique. Introducing explicit flow edges between all pairs of vertices in a clique is problematic because there is no one to one correspondence between the V variables associated with a pixel and such edges in a clique. We solve this problem by modeling a clique by a gadget whose construction we now describe.

3.1 Gadget

We model the flow carrying edges in a clique of size k by a gadget consisting of $k+2$ nodes[1]. The gadget consists of k *pixel* nodes $p, q, \ldots, r$ corresponding to the k pixels, and two *auxiliary* nodes n and m. We introduce directed edges called *conjugate* edges from, i) the node n to all pixel nodes, ii) from all pixel nodes to the node m. We also introduce a directed edge $m \rightarrow n$ from the node m to the node n. The gadget corresponding to a clique of size 4 is shown in Figure 1. Flow from pixel p to pixel q in the 4 clique is routed along the path $p \rightarrow m \rightarrow n \rightarrow q$ in the gadget (see Figure 1). The cliques, in general, can be overlapping with a node belonging to many cliques. Correspondingly a pixel node can participate in multiple gadgets. Figure 2 shows a gadget based flow graph corresponding to three 4 cliques. The flow path from p to r (see Figure 2) is an example of routing through multiple gadgets.

3.2 Conjugate Edges, Dual Feasibility Constraints (DFCs) and Terminal Edges

We assume that the flow graph models the movement of ball a and that all dual variables $V_{\mathbf{c},p,b}$ are initially

[1]Since there is one to one correspondence between a clique and the gadget of the flow graph, we use these terms interchangeably in the following discussion

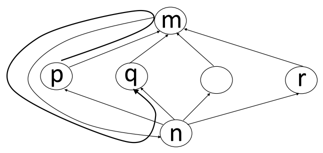


Fig. 1. A gadget for 4-clique

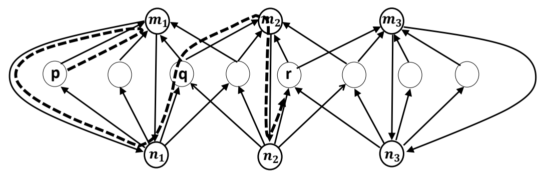


Fig. 2. Flow graph corresponding to three overlapping 4 cliques and a flow path in it

set to zero and remain so throughout the running of the algorithm[2]. Therefore, (10) simplifies to

$$\sum_{p\in\mathbf{c}:\mathbf{l}_\mathbf{c}^p=a} V_{\mathbf{c},p,a} \le W_\mathbf{c}(\mathbf{l_c}), \qquad \mathbf{c}\in\mathcal{C},\ \mathbf{l_c}\in\mathcal{L}^k. \tag{14}$$

Flow in the two conjugate edges, incident at and emanating from a pixel node p in the gadget corresponding to clique $\mathbf{c}$, is denoted by $f_{np}^{\mathbf{c}}$ and $f_{pm}^{\mathbf{c}}$ respectively. The effect of flow in the edge $n\to p$ ($p\to m$) can be looked upon as increasing (decreasing) the height of the ball a in well p by amount $f_{np}^{\mathbf{c}}$ ($f_{pm}^{\mathbf{c}}$). The relationship between the dual variable $V_{\mathbf{c},p,a}$ and the flow in the conjugate edges associated with node p is, therefore, defined by

$$V_{\mathbf{c},p,a} = f_{np}^{\mathbf{c}} - f_{pm}^{\mathbf{c}}. \tag{15}$$

Equation (15) implies that in the corresponding flow graph the constraints defined by (14) take the form

$$\sum_{p\in\mathbf{c}:\mathbf{l}_\mathbf{c}^p=a} (f_{np}^{\mathbf{c}} - f_{pm}^{\mathbf{c}}) \le W_\mathbf{c}(\mathbf{l_c}), \qquad \mathbf{c}\in\mathcal{C},\ \mathbf{l_c}\in\mathcal{L}^k. \tag{16}$$

If we look upon $f_{np}^{\mathbf{c}} - f_{pm}^{\mathbf{c}}$ as the effective flow in a pair of conjugate edges, then an inequality of the form (16) essentially implies that for a labeling $\mathbf{l_c}$, the sum of effective flows in the collection of conjugate edges corresponding to pixels labeled a in a clique $\mathbf{c}$ in the flow graph can not exceed $W_\mathbf{c}(\mathbf{l_c})$. We refer to a constraint of type (16) corresponding to a labeling $\mathbf{l_c}$ on clique $\mathbf{c}$ as a *dual feasibility constraint* (or simply DFC). $W_c(\mathbf{l_c})$ is also referred to as the *cost* of the DFC. The pixels of $\mathbf{c}$ which are assigned label a in $\mathbf{l_c}$, the conjugate edges, and the V variables corresponding to these pixels are said to be *participating* in the DFC. Conversely, a DFC is said to *cover* or *contain* all nodes, edges, and V variables participating in it. The quantity

$$W_\mathbf{c}(\mathbf{l_c}) - \sum_{p\in\mathbf{c}:\mathbf{l}_\mathbf{c}^p=a} (f_{np}^{\mathbf{c}} - f_{pm}^{\mathbf{c}}) \tag{17}$$

is referred to as the *slack* in the DFC.

For example, consider a 3-clique containing nodes p, q and r. There will be 8 DFCs corresponding to the 8 labeling configurations on the clique. The DFC corresponding to labeling $\{a, a, b\}$ covers pixels p, q and their conjugate edges. The DFC corresponding to labeling $\{a, b, a\}$ covers pixels p, r and their conjugate edges.

A DFC is said to be *violated* or become *infeasible* if its slack becomes negative. A DFC with slack of 0 is referred to as *tight*. All conjugate edges covered by a tight DFC are referred to as *tight under/in that DFC* or simply *tight* or *saturated* edges.

The slack of a DFC can be interpreted as the extent to which any pair of conjugate edges participating in it can have flow increased without violating it. In effect, the flow can be increased in a pair of conjugate edges as long as the DFCs covering them are not violated, i.e., it cannot exceed the minimum of the slacks of all the DFCs covering that pair of conjugate edges. This minimum value for a pair of conjugate edges can be considered as its capacity. We define the *residual capacity* in a pair of conjugate edges to be equal to the minimum of the slacks of all DFCs in which it participates, excluding those corresponding to uniform labeling. The DFCs corresponding to uniform labeling are excluded because the flow conservation constraints imply that for a clique $\mathbf{c}$

$$\sum_{p\in\mathbf{c}} f_{pm}^{\mathbf{c}} = \sum_{p\in\mathbf{c}} f_{np}^{\mathbf{c}} \Rightarrow \sum_{p\in\mathbf{c}} f_{np}^{\mathbf{c}} - \sum_{p\in\mathbf{c}} f_{pm}^{\mathbf{c}} = \sum_{p\in\mathbf{c}} V_{\mathbf{c},p,a} = 0, \tag{18}$$

In effect the slack in a DFC corresponding to uniform labeling is always 0. At the same time any flow pushing through the gadget never violates the DFC corresponding to uniform labeling. Hence, the uniform labeling constraints can be skipped in the residual capacity calculations.

The flow graph for the dual optimization problem is created as follows: The set of nodes consists of two distinguished nodes *source* (s) and *sink* (t), a pixel node corresponding to each pixel, and two auxiliary nodes for each clique. The pixels and the auxiliary nodes corresponding to a clique are connected by the gadget edges. Other than these there are edges from the node s to a pixel node and from a pixel node to the node t. We refer to the nodes s and t as *terminal nodes*, and the edges between the terminal nodes and the pixel nodes as *terminal edges*. The capacity of the terminal edges depends upon whether the flow graph models ball a's flow or ball b's flow. If flow is being modeled for the movement of ball a then there is a directed edge from s to each pixel node in whose well the ball p^a is above the ball p^b. The capacity of the edge is equal to the difference between the heights of the two balls. Similarly there is a directed edge from the pixel node, in whose well ball a is lower, to the node t and the capacity of the edge is the difference

[2]We show later that this has no effect on optimization problem if the potentials are submodular

in height between the two balls. Setting capacities of terminal edges in this way ensures that any non-active ball (connected to the source) can never go below its active ball. Similarly, the sink terminal edge capacities ensure that an active ball never goes above the non-active ball in its well. This maintains the invariance that an active ball is always below the non active ball in a well and equations (8) and (11) are always satisfied. The capacities of all $m \to n$ edges are set to infinity (infinity here implies a value much higher than the maximum clique potential value or highest unary cost).

3.3 Flow Redistribution

We make the implications of our definitions clear by an example. As mentioned earlier flow pushing in the flow graph simulates the movement of balls in the two wells of a clique. If in a clique $\mathbf{c}$, the ball a in the well p comes down by amount δ and the ball a of the well q in the same clique goes up by the same amount, then in the flow graph this is simulated by pushing δ flow in the path fragment $p \to m \to n \to q$ of the gadget corresponding to clique c. The effect of this flow push is to reduce the value of dual variable $V_{\mathbf{c},p,a}$ by δ and increase the value of $V_{\mathbf{c},q,a}$ by δ. Note that these changes in the values of the dual variables do not affect the DFCs in which either both $V_{\mathbf{c},p,a}$ and $V_{\mathbf{c},q,a}$ participate or both $V_{\mathbf{c},p,a}$ and $V_{\mathbf{c},q,a}$ do not participate. However, the slack of those DFCs in which only $V_{\mathbf{c},p,a}$ participates increases by δ and those in which only $V_{\mathbf{c},q,a}$ participates decreases by δ. Clearly the value of δ should not be allowed to increase by an amount which makes the slack of a DFC negative, i.e., "restrictions on residual capacity only need to take into account those DFCs which can become infeasible by pushing flow". Now let us consider a situation in which the residual capacity for the pair of conjugate edges for node q in clique c is 0. This implies that at least one DFC which covers the dual variable $V_{\mathbf{c},q,a}$ is tight. It will still be possible for the node p to send flow to q provided all the tight DFCs that cover $V_{\mathbf{c},q,a}$ also cover $V_{\mathbf{c},p,a}$. This is because any increase in $V_{\mathbf{c},q,a}$ is counter balanced by corresponding decrease in $V_{\mathbf{c},p,a}$ (path from p to q is $p \to m \to n \to q$) and the DFCs that were tight prior to pushing of flow continue to remain tight. Flow from p to q is therefore limited by the slack in DFCs which cover only $V_{\mathbf{c},q,a}$ (and do not cover $V_{\mathbf{c},p,a}$).

3.4 Residual Graph

The fact that the capacity constraints are on the effective flow in a pair of conjugate edges incident at a pixel node in a gadget, rather than on the flow on each individual conjugate edge, is dealt with by requiring that flow is non zero in only one of the edges in the conjugate pair. This gets taken care of by the rules used for creating the residual graph in presence of some valid flow. When there is non zero flow in a conjugate edge pair, the conjugate edge with zero flow is not included in the residual graph. The capacity constraint associated with the pair is associated with the edge with non zero flow. Three cases arise:

1) *The conjugate edge with non zero flow emanates from an n type auxiliary node*: In this case the residual graph has two edges. The edge from the auxiliary node (n type) to a pixel node has capacity equal to the residual capacity of the conjugate edge pair, and the edge from the pixel node to the auxiliary node (n type) has capacity equal to the flow in the conjugate edge pair. There are no edges in the residual graph between the pixel node and the auxiliary node of type m.
2) *The conjugate edge with non zero flow emanates from a pixel node to the m type auxiliary node*: In this case, the capacity of the residual edge from the pixel node to the auxiliary node (m type) is infinity, and the reverse direction edge has a capacity equal to the flow towards the auxiliary node (m type) in the conjugate edge pair under consideration.
3) *When there is no flow in either of the edges of a conjugate pair*: The residual graph has two edges corresponding to the two conjugate edges. The one emanating from the auxiliary node (n type) to the pixel node has capacity equal to the capacity of the conjugate edge pair. Capacity of the other, from the pixel node to the m type auxiliary node, is infinity. The requirement that only one of the conjugate edges has zero flow at any time is ensured by restricting the flow augmenting paths from s to t in the residual graph to include only one of the edges of a conjugate pair.

The rules for setting the residual capacities listed above enable a variable $V_{\mathbf{c},p,a}$ that participates in a DFC as given in equation (14) to have both positive and negative values. The value will be positive if $f^{\mathbf{c}}_{np}$ is positive and will be negative if $f^{\mathbf{c}}_{pm}$ is positive. Note that even when a DFC becomes tight because of pushing of additional flow in an edge and the residual capacity of that edge becomes zero, there may still exist the possibility of the edge being involved in an augmenting path as explained in the previous section.

Notwithstanding these complications, there does exist a flow value that is maximal for the flow graph. We show that, in general, this value is less than or equal to the "minimum capacity" (S, T) cut (also called min (S, T) cut) on the flow graph with a suitably generalized notion of cut capacity[3]. We show that when the clique potential is submodular, the capacity of the min (S, T) cut in the gadget based flow graph

[3]An (S, T) cut is given by a partition of the nodes of the flow graph into two sets S and T with $s \in S$ and $t \in T$ and consists of edges from S to T

Labeling	Slack/Input Potential	Slack After Flow Send	Labeling	Slack/Input Potential	Slack After Flow Send
bbbb	0	0	bbba	71	71
abbb	101	101	abba	160	160
babb	71	40	baba	71	40
aabb	31	0	aaba	101	70
bbab	71	102	bbaa	71	102
abab	101	132	abaa	101	132
baab	130	130	baaa	71	71
aaab	101	101	aaaa	0	0

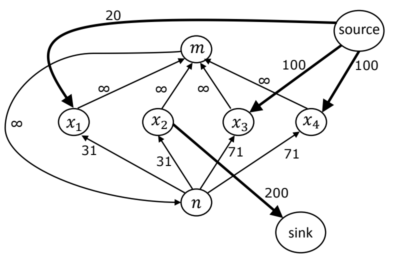
TABLE 1
Example clique potential

Node	$E(a)$	$E(b)$
x_1	50	30
x_2	0	200
x_3	100	0
x_4	100	0

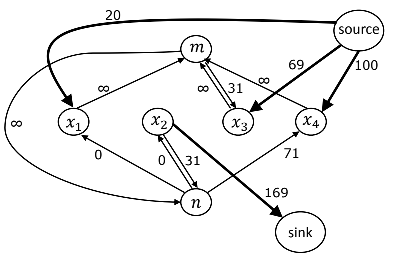
TABLE 2
Unary potential

Conj. edge	Residual cap
$n \to x_1$	0
$n \to x_2$	0
$n \to x_3$	71
$n \to x_4$	71

TABLE 3
Revised residual cap.



(a) Initial Flow Graph



(b) Revised Residual Flow Graph

Fig. 3. Example Flow Graph

equals max flow in the flow graph. We also show how the optimal labeling for the primal problem can be inferred from the min cut in the flow graph when max flow state has been reached. It must be noted that if the clique potential is non-submodular, then this one to one correspondence between optimal primal and optimal dual does not hold. The value of the optimal primal may be larger than the value of the optimal dual.

3.5 Example

To further clarify we show how flow is pushed in the gadget based flow graph corresponding to an MRF-MAP problem.

Consider a problem having single clique of size 4. The clique potential penalizes a labeling as given by the first column of Table 1. It can be verified that the clique potential is submodular. The unary costs are as given by Table 2. Note that initially (i.e., zero flow) the input potential values are also the slack values for the corresponding DFCs.

The flow graph with terminal and conjugate edge capacities created from potentials given in Tables 1 and 2 when flow is initially zero in all edges is as shown in Figure 3(a). Since the height of a and b balls in the well x_1 are 50 and 30 respectively, the *source* is connected to x_1 by a terminal edge with capacity equal to $50 - 30 = 20$, the difference in heights of the balls. Similarly the *source* is connected to x_3 and x_4 by terminal edges of capacity 100. There is a terminal edge from x_2 to *sink* of capacity 200 (ball b is 200 above ball a in well x_2). When flow in all edges is 0 then the slacks of all DFCs (there are 16 of them) are equal to the values in the column corresponding to 'Input Potential' in Table 1. We now show how the residual capacities of conjugate edges are calculated by working out the details for the pair of conjugate edges corresponding to the node x_1 (edges $n \to x_1$ and $x_1 \to m$). We know that the residual capacity of a pair of conjugate edges is the minimum of the slack of DFCs (excluding the uniform labeling one) that cover it. In this case the DFCs that cover correspond to labelings $abbb$, $aabb$, $abab$, $aaab$, $abba$, $aaba$, and $abaa$ and have slacks 101, 31, 101, 101, 160, 101, and 101 respectively. The residual capacity of x_1's pair of conjugate edges is therefore 31. This value is initially associated with the edge $n \to x_1$. The capacity of the other edge ($x_1 \to m$) is infinity (flow in it only increases the slack).

Flow is first pushed along the path $source \to x_3 \to m \to n \to x_2 \to sink$. As per the residual capacities flow pushed that can be pushed is 31. This makes the DFC for labeling $aabb$ (or simply DFC $aabb$) tight (i.e., its slack becomes 0). Table 1 (column 3) contains the revised slacks of all the DFCs after this flow push. The revised residual capacities of the conjugate edges emanating from auxiliary node n are given in Table 3 and the residual graph with the residual capacities on individual edges is given in Figure 3(b). Note that the edges in the residual graph are as per the cases discussed in Section 3.4.

The tightness of the DFC $aabb$ still allows flow through path $source \to x_1 \to m \to n \to x_2 \to sink$ since both the nodes x_1 and x_2 participate in the DFC $aabb$. Sending flow from $x_1 \to x_2$ does not affect the slack in the DFC $aabb$. The capacity of the path fragment from x_1 to x_2 is contextually constrained at 40 due to the slack in the DFCs $babb$ and $baba$. However, because of the constraints posed by terminal edge capacities, flow of only 20 can be sent from the *source* to *sink*as that saturates the edge $source \rightarrow x_1$. No more flow can be sent in the flow graph and the total flow sent (also the value of the maximum flow) is $31 + 20 = 51$.

4 Max Flow Min Cut Relationship

Consider any (S, T) cut in a gadget based flow graph. The edges from S to T are either terminal edges, or are between auxiliary nodes, or are conjugate edges. Note that only one of a pair of conjugate edges can be from an S side node to a T side node. The capacity of an (S, T) cut is traditionally the sum of the capacities of the edges from nodes in S to nodes in T. For a gadget based flow graph we need the notion of *cover* given below to define the capacity of an (S, T) cut.

Definition 4.1. *Every DFC is a* cover *of all pairs of conjugate edges that participate in it.*

Definition 4.2. *A set of DFCs that covers all the conjugate edges of an (S, T) cut is called a* cut cover*. The* cost *or* capacity *of a cut cover is defined to be the sum of the costs of DFCs constituting the cut cover.*

In the (S, T) cut corresponding to Figure 4 there are 4 edge covers $EC1, \dots, EC4$. There are two cut covers possible, one including $EC1$ and $EC4$ and the other including $EC1$, $EC2$ and $EC3$. The capacity of the (S, T) cut corresponds to one of the two cut covers and is given by:

Definition 4.3. *The* cost *or* capacity *of an (S, T) cut is equal to the value of the* smallest cost *cut cover among all the cut covers covering the conjugate edges in the (S, T) cut plus the capacities of all the terminal edges in the cut.*

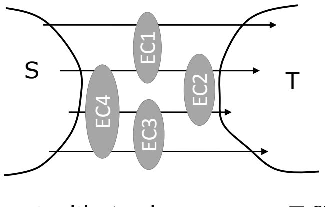


Fig. 4. (S, T) cut with 4 edge covers, $EC1, \dots, EC4$.

The definitions lead to the following relationship between cost of any (S, T) cut and the flow in the gadget based flow graph:

Lemma 4.4. *Flow in a gadget based flow graph cannot exceed the capacity of any (S, T) cut.*

The proof of the lemma as well as all following lemmas can be found in the Supplementary material.

Consider the scenario when the flow is equal to the maximum value flow in the gadget based flow graph. Since the flow can not be incremented any further, no path can exist between s and t in the residual graph. Let S be the set of nodes reachable[4] from s in this residual graph and let T be the rest of the nodes of the flow graph. Effective flow across a cut from S to T is equal to the sum of flow in the flow graph edges from S to T minus the sum of flow in the flow graph edges from T to S. A gadget is said to be *'on'* an (S, T) cut if some of the gadget nodes are in S and others in T. Lemma 4.5 describes some properties of gadgets on an (S, T) cut when max flow state has been reached.

Lemma 4.5. *For a gadget* on *an (S, T) cut when max flow state has been reached*

1) *Both auxiliary nodes m and n are in S.*
2) $f_{pm} = 0, \forall p \in T$.
3) $f = \sum_{i \in S} \sum_{j \in T} f_{ij}$.

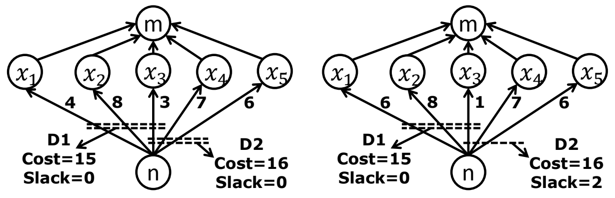
Inherent in the reasoning that has led us to the preceding results is the notion that flow can be increased provided there exists a path from s to t in a residual graph. However, it does not follow that flow is maximum if no s to t path exists in a residual graph. Consider, for example, the flow graph of Figure 5(a). The numbers written adjacent to an edge is the flow in the edge and the covering DFCs are marked by dashed edges (double dashed edges shows tight DFCs). All edges in this case have zero residual capacity because the two DFCs, $D1$ covering edges corresponding to x_1, x_2, x_3, and $D2$ covering edges corresponding to x_3, x_4, x_5 are tight and together cover all the edges. Suppose, it is possible to redistribute the flow by increasing the flow in the conjugate edge incident at x_1 to 6 and decreasing the flow in the conjugate edge incident at x_3 to 1. The total flow out of s remains the same but $D2$ (marked by single dashed edge in Figure 5(b)) now has a slack of 2. Conjugate edges corresponding to x_4 and x_5 may now have residual capacities of 2 each and there may now be a s to t path in the residual graph passing through these edges. The possibility of creating slack through redistribution is the reason for this max flow problem to be NP hard in general.

Let (S, T) be the cut in the flow graph when the flow is maximum. Using Lemma 4.5 we can say that the effective flow in the (S, T) cut is equal to the sum of the flow in the conjugate edges from the auxiliary nodes n to nodes in T plus the flow in the terminal edges from s to nodes in T. Note that each conjugate edge from S to T has zero residual capacity and therefore at least one DFC covering it is tight. In effect there are cut covers, corresponding to the (S, T) cut, in which all DFCs are tight. Is max flow equal to min capacity cut? Consider the smallest cost cut cover among all cut covers of the (S, T) cut. If each conjugate edge in the cut is covered by only one DFC in the smallest cost cut cover then it follows that max flow is equal to the capacity of the (S, T) cut. This

[4]A node p is said to be reachable from q, if it is possible to send flow from q to p, without violating any DFC

may not hold in general as there can be cases when a conjugate edge in the cut is covered by two or more tight DFCs in the smallest cost cut cover. In such cases the flow in the edge gets counted more than once in the cost of the cut cover, and max flow may be less than capacity of the min cost cut cover.

We now show that when the clique potential functions are submodular then not only is the max flow in the flow graph always equal to the value of the min cost cover, but also that the max flow can be obtained by standard flow algorithms (i.e., redistribution has no effect when clique potentials are submodular).



(a) Before redistribution (b) After redistribution

Fig. 5. Flow state before and after re-distributing flow in a gadget

For the purposes of the discussion that follows we specify any labeling of a clique's pixels by the set of pixels that are labeled a in the labeling. Since this set of pixels are also said to be participating in the DFC that covers them, a DFC can also be looked upon as being specified by a set. The clique potential $W_{\mathbf{c}}(\cdot)$, which assigns a cost for every labeling can now be considered to be a set function.

A clique potential function $W_{\mathbf{c}}(\cdot)$ is considered to be submodular, if for all $\mathbf{l}_{\mathbf{c}}^{'}, \mathbf{l}_{\mathbf{c}}^{''} \in \mathcal{L}^k$

$$W_{\mathbf{c}}(\mathbf{l}_{\mathbf{c}}^{'}) + W_{\mathbf{c}}(\mathbf{l}_{\mathbf{c}}^{''}) \geq W_{\mathbf{c}}(\mathbf{l}_{\mathbf{c}}^{'} \cup \mathbf{l}_{\mathbf{c}}^{''}) + W_{\mathbf{c}}(\mathbf{l}_{\mathbf{c}}^{'} \cap \mathbf{l}_{\mathbf{c}}^{''}),$$

where $\cup$ and $\cap$ are union and intersection on sets of a labeled pixels. Recall that $W_{\mathbf{c}}(\mathbf{l}_{\mathbf{c}}^{'})$ is also referred to as the cost of DFC corresponding to $\mathbf{l}_{\mathbf{c}}^{'}$ and a DFC is called tight if it's slack is zero or equivalently sum of flows in the conjugate edges participating in it is equal to DFC's cost. When potentials are submodular, tight DFCs exhibit the following property:

Lemma 4.6. *If $W_{\mathbf{c}}(\cdot)$ is submodular, then for every two tight DFCs corresponding to labeling $\mathbf{l}_{\mathbf{c}}^{'}$ and $\mathbf{l}_{\mathbf{c}}^{''}$, the DFCs corresponding to labeling $\mathbf{l}_{\mathbf{c}}^{'} \cup \mathbf{l}_{\mathbf{c}}^{''}$ and $\mathbf{l}_{\mathbf{c}}^{'} \cap \mathbf{l}_{\mathbf{c}}^{''}$ are also tight.*

Lemma 4.6 tells us that no advantage can be gained by redistribution of flow among the edges of tight DFCs as the DFC corresponding to the union will continue to remain tight and will force the residual capacities of all the edges covered by it to be zero. Lemma 4.6 also allows us to infer that there exists a cut cover of an (S, T) cut such that all the DFCs in it are tight and every conjugate edge of the cut is covered only once by the DFCs. This cut cover can be simply found by starting with any cut cover covering all edges of the cut (an edge may be covered twice in such a cut cover) and then replacing the two tight DFCs in it with the DFC corresponding to their union. Since all edges are covered only once and all DFCs are tight, the sum of flow in the edges is *exactly* equal to the sum of costs of DFCs in the cut cover. We, therefore, have:

Theorem 4.7. *In the flow graph corresponding to the dual optimization problem, max flow is equal to min cut under the assumption that clique potential functions on all cliques are submodular.*

Note that our flow formulation ensures that all complimentary slackness conditions other than equation (12) are always satisfied. In a maximum flow situation, equation (16) is satisfied with equality for all cliques on the cut[5]. For a clique not on the cut, we uniformly label all its pixel nodes either by a or b (label all pixels of cliques totally in S as b and all pixels of cliques totally in T as a). The DFC corresponding to uniform labeling is always tight due to flow conservation at an auxiliary node as shown in equation (18). Since all complimentary slackness conditions governing primal and dual solutions in our framework are satisfied and primal and dual solutions are feasible, the optimality of primal and dual solutions is guaranteed.

5 Uniform Labeling Cost and Submodularity Constraints

In the primal-dual framework as derived in Section 2, as well as for proving the optimality of the primal and dual solution in the previous section, we assumed that the cost of assigning uniform labeling (all a's or all b's) to pixels of a clique to be zero. Under this assumption an additional constraint resulting out of complimentary slackness conditions was given by equation (13). The gadget used for creating flow graphs ensures that this complimentary slackness condition continues to be satisfied by virtue of flow conservation at auxiliary nodes. Since the gadget does not explicitly capture equation (13), the flow graph formulation introduced is valid only for the cases where uniform labeling costs are zero. We now show that this assumption is not restrictive when clique potentials are submodular.

Consider the dual optimization problem as developed in Section 2. Suppose we textually replace all occurrences of $V_{\mathbf{c},p,l}$, the V variable for the label l associated with pixel p of clique $\mathbf{c}$, by $V_{\mathbf{c},p,l}+\delta$ in equations of type (8) and (10). Then, effectively, the only changes that have occurred in the dual optimization problem are the following:

1) The r.h.s. of equation (8) which defines the value of dual variable h_p^l has an additional term δ, and
2) The l.h.s of all equations of type (10) in which $V_{\mathbf{c},p,l}$ occurs, has an additional δ.

[5]Note that equation (16) is nothing but another representation of equation (12)

It is as if we have increased the unary/data cost $D_p(l)$ of assigning label l to a pixel p in clique $\mathbf{c}$ and compensated for that increase by decreasing costs in the clique potential for the labelings in which p was labeled l. It is easy to establish that from the perspective of inference, the cost/energy of the optimal solution and the assignment of labels in that optimal solution in both versions of the problem remain the same. Choosing δ such that one of the labeling costs becomes zero can be looked upon as tightening of a DFC. We call this process of tightening at least one DFC using variable $V_{\mathbf{c},p,l}$ as *reparametrization using* $V_{\mathbf{c},p,l}$. It can be shown (proof is in the Supplementary material) that reparametrization preserves submodularity of clique potentials.

Lemma 5.1. *A submodular clique potential* $W_{\mathbf{c}}$*, remains submodular after reparametrization using any* $V_{\mathbf{c},p,a}$ *or* $V_{\mathbf{c},p,b}$ *for all pixels* $p \in \mathbf{c}$*.*

It can be shown that if the DFCs corresponding to uniform labeling are not tight, there must exist at least one V variable which can be reparametrized. Note that if no V variable can be reparametrized, then each V variable must be participating in at least one tight DFC. The union of all such tight DFCs corresponds to uniform labeling which must be tight as well (follows from equation (18)). The argument applies for the DFCs corresponding to uniform labeling a as well as b. We can summarize the result as follows:

Lemma 5.2. *Any 2-label multi-clique inference problem with submodular clique potential having non-zero uniform labeling cost can be reparametrized to an equivalent problem with a new clique potential function which has uniform labeling costs zero.*

For tightening the DFCs corresponding to uniform labelings a and b for a clique $\mathbf{c}$, at every iteration we choose any variable $V_{\mathbf{c},p,l}$ and perform reparametrization on it if possible[6]. The process ends when the DFCs corresponding to the uniform labelings a and b have been tightened. Note that this may require reparametrization using variables of type $V_{\mathbf{c},p,a}$ as well as $V_{\mathbf{c},p,b}$.

6 Max Flow Algorithm

The proof of the "max flow min cut" theorem for gadget based flow graphs with submodular clique potentials is not algorithmic in the sense that how the max flow state can be arrived at has not been specified. In this section we give an algorithm to find maximum flow in a gadget based flow graph.

6.1 Augmenting Path Availability

It is easy to see that any augmenting path in a gadget based flow graph starts at a pixel node connected to the source, passes alternately through a pixel and a auxiliary node and ends at a pixel node connected to the sink. A path fragment is a consecutive triplet of pixel, auxiliary and pixel nodes in an augmenting path. Note that two consecutive path fragments have a pixel node in common and that all three nodes of a path fragment should belong to one clique. The complete augmenting path can be visualized as a series of path fragments with a terminal edge at the start and the end. We refer to the pixel node of the path fragment closer to the source as the *sending node* and the other as the *receiving node*. The corresponding conjugate edges are similarly called the sending and the receiving edges. The amount of flow that can be sent from the sending node to the receiving node of a path fragment without violating any DFC covering its edges is referred to as the *residual capacity* of the path fragment. A path fragment with residual capacity zero is termed to be *tight/saturated/blocked*. Note that any flow sent through a path fragment does not effect the slack of a DFCs which either covers both the sending and the receiving node or does not cover any of the two nodes. Any flow sent through a path fragment increases the slack of all DFCs covering only the sending node and decreases the slack of all DFCs covering only the receiving node. The residual capacity of a path fragment is therefore governed by the minimum slack of all DFCs which cover the receiving node and do not cover the sending node. We can now specify when a conjugate edge can not be a part of any augmenting path (proofs are in the Supplementary material).

Lemma 6.1. *A saturated conjugate edge* $n \to p$ *corresponding to a clique* $\mathbf{c}$ *in presence of flow* f *cannot be in an augmenting path as receiving edge of the path fragment if the intersection of all tight DFCs covering it contains no other edge* [7,8]*.*

We add "include all saturated conjugate edges in the residual graph to which Lemma 6.1 does not apply" as an additional rule for adding edges to the residual graph. Such a saturated edge of the residual graph can be included in an $s-t$ augmenting path provided there is an immediately preceding conjugate edge in the path covered by the same tight DFC. The residual capacity of such a saturated edge is contextually defined as the minimum slack of all DFCs which cover the saturated receiving edge and does not cover the preceding edge. We can now state the following (proof is in the Supplementary material):

Lemma 6.2. *If a flow is not maximum in a flow graph corresponding to the dual optimization problem with sub-*

[6]Note that reparametrization for $V_{\mathbf{c},p,l}$ is not possible if it participates in any tight DFC

[7]Note that we only talk about saturated conjugate edges of type $n \to p$. Conjugate edges of type $p \to m$ are never saturated since increase in flow in $p \to m$ causes slack of any DFC covering it to increase.

[8]If there is only one DFC, the intersection is defined as the DFC itself

modular clique potential functions, then there will exist an $s-t$ augmenting path in the residual graph created with respect to the flow.

6.2 Shortest Path

Until now we have been able to generalize most of the properties defined for max flow in traditional flow graphs to gadget based flow graphs. We generalized the notions of residual capacity and augmenting paths. We also proved the max flow min cut theorem for gadget based flow graphs. We now show that the idea of flow augmentation along "shortest augmenting path" also extends to gadget based flow graphs and there are algorithms which converge after polynomial number of such flow augmentation iterations. We define the length of an augmenting path as the number of path fragments in it. Let $\delta^i(s,t)$ be length of the shortest path from s to t after i shortest augmenting path flow augmentations. It can be shown that (proof is in the Supplementary material):

Lemma 6.3. *If flow augmentation is always done along a shortest augmenting path, then $\delta^{i+1}(s,t) \geq \delta^i(s,t)$.*

It should be pointed out that unlike standard max flow problems for which Edmonds and Karp's shortest path based augmentation strategy [36] based on Lemma 6.3 results in a strongly polynomial max flow algorithm, the same strategy may not even result in a terminating algorithm for gadget based max flow problems.

Consider a scenario when the shortest augmenting paths from s to t pass through a clique containing nodes x_1, x_2, x_3 and x_4. There are paths from s to x_1 and s to x_3 of high residual capacity. These paths from s to x_1 and x_3 are of the same length. There are paths from x_2 to t, and x_4 to t of the same length of high residual capacity. Consider the situation where the DFC covering x_1 and x_2 is tight, blocking the path fragment x_3 to x_2. The augmenting path chosen is s to $x_1 \rightarrow m \rightarrow n \rightarrow x_4$ to t. The flow gets augmented by ϵ and the DFC covering x_4 and x_3 becomes tight blocking further flow in this path fragment. However the DFC covering x_1 and x_2 has non-zero slack now. This enables ϵ flow to be sent from s to t along path s to $x_3 \rightarrow m \rightarrow n \rightarrow x_2$ to t. This creates a slack again in the DFC covering x_3 and x_4 unblocking path fragment x_1 to x_4 enabling the possibility of repeated augmentations which send very little flow and do not bring any change in the shortest path structure.

If, however, the shortest augmenting paths were lexicographically ordered then a path fragment can not be "unblocked" more than k times (once by each node) without a change of direction of flow in it. Therefore, Edmonds and Karp's shortest path augmentation strategy has to be used along with ordering of paths of the same length lexicographically to bound the number of flow augmentation iterations of the same length. A max flow algorithm based on the above and its complexity analysis is given below.

6.3 Flow Algorithm and Complexity Analysis

We call our algorithm Generic Cuts (GC). Algorithms 1 and 2 describe the two main components of GC. Note that flow can be augmented using any heuristic used in traditional max flow algorithms. The augmenting path framework of steps 4 to 6 could even be replaced by the Push Relabel technique of [35].

Algorithm 1 GC Max Flow Algorithm

```
1: for All cliques $\mathbf{c} \in \mathcal{C}$ do
2:     Reparametrize the clique potential until the DFCs for uniform labeling become tight;
3: Build the residual graph $R$;
4: while There exists an $s$-$t$ augmenting path in $R$ find the lexicographically shortest augmenting path; do
5:     Augment flow in that path;
6:     Build the residual graph $R$;
```

Algorithm 2 GC Residual Graph Construction

```
1: for All cliques $\mathbf{c}$ do
2:     for All DFCs of $\mathbf{c}$ do
3:         Calculate the slack in presence of current flow;
4:         Set the residual capacity of a conjugate edge pair as the minimum of slacks of all DFCs in which it participates;
5:     for All path fragments of type $p$ to $q$ do
6:         if Edge corresponding to $q$ have have non zero residual capacity then
7:             Add the path fragment to the residual graph;
8:             Set the residual capacity of the path fragment as the residual capacity of receiving edge;
9:         else
10:            if The intersection of the tight DFCs covering $q$ also cover $p$ then
11:                Add the path fragment to the residual graph;
12:                Set the residual capacity of the added fragment as minimum slack of all DFCs which cover $q$ and not $p$;
```

Since the number of DFCs per clique can be $O(2^k)$, the residual capacity of each edge can be computed naively in $O(2^k)$ steps. Time to find the shortest augmenting path and updating the residual capacities of the edges in the residual graph is $O(2^k|E|)$. The total number of iterations before the path length increases is bounded by $O(k|E|)$. The total number of iterations is $O(k|V||E|)$ and the overall time complexity is $O(2^k k|V||E|^2)$. If the set of cliques is denoted by $\mathcal{C}$, and the number of pixels by n, then $|V|$ and $|E|$ are $O(n+|\mathcal{C}|)$ and $O(k|\mathcal{C}|)$ respectively. Therefore, the complexity of the max flow algorithm using Edmonds and Karp's shortest path heuristic is $O(2^k k^3 n(|\mathcal{C}|)^2)$. Under the assumption that submodular functions for computer vision problems are locally defined with $n \approx |\mathcal{C}|$, the complexity can also be written as $O(2^k k^3 n^3)$.

It should be noted that strongly polynomial algorithms for general submodular function optimization [26], [27] can be used to recompute the residual capacities of the conjugate edges of a gadget in which flow has been augmented. The time complexity of these algorithms is at least $O(n^6)$ and computing the residual capacities of the k conjugate edges of a gadget using them will take $O(k^6)$ steps. Keeping the complexity of these algorithms in mind, the use of brute force $O(2^k)$ algorithm is advisable.

The framework developed here also allows to exploit sparseness in clique potentials. The flow problem created for higher order cliques needs to keep track of slack in $O(2^k)$ DFCs which gets reflected in the running time and memory requirements of the algorithm. In case the problem specification guarantees that some of the labeling costs are high and the corresponding DFCs will never be tight, the algorithm can simply skip tracking slacks in those DFCs. The $O(2^k)$ multiplicative factor in the time complexity analysis can be replaced by the number of DFCs which the algorithm actually needs to track.

So far the technique of choice for MRF-MAP for binary labels with high order cliques has required reduction of higher order Boolean functions to second order and use of QPBO for optimization [3]. This process involves creation of $O(2^k)$ auxiliary nodes per clique in the reduction phase. Since the resultant flow graph is dense, the number of edges can be $O(2^{2k})$ per clique. If $\mathcal{C}$ is the set of cliques and $n \approx |\mathcal{C}|$, the max flow part of the algorithm's time complexity will be $O(2^{5k} n|\mathcal{C}|^2)$ or $O(2^{5k} n^3)$, assuming that the max flow algorithm is based on Edmonds and Karp [36]. This analysis has been done primarily to estimate the type of speed up one should expect using our algorithm over algorithms based on the reduction technique. In the section where we report experimental results we show that this is indeed the case in practice.

7 Experiments and Results

All experiments were conducted on a computer with 2.5 GHz dual core processor, 2 GB of RAM running Windows 7 operating system with 64 bit addressability. The experiments have focussed on comparing time taken and errors in labeling observed using GC, message passing methods like Max Product Inference (MPI) [37], Dual Decomposition (DD) [17], TRWS [2], reduction techniques proposed by Ishikawa (IQ) [3], Fix et al. (FZ) [4] and the Generalized Roof Duality based algorithm (KS) of [5]. Implementations of IQ, FZ and KS are from [38], [39], [40] and [41] respectively. The rest are from the Darwin framework [42].

Experiments, reported here, put in perspective comparative performance and quality of the prevalent techniques on 2-label multi-clique problems when potentials are submodular. It should be noted that non combinatorial direct methods [2], [17], [37]have been developed primarily to handle non-submodular potentials. They are computationally intensive in general. Our experiments show they are computationally expensive even when run on submodular potentials.

The first experiment has focussed on the optimality of GC using a segmentation problem on a synthetic image with Gaussian noise added as the base. The terminal weights (unary potential) for black and white labeling were chosen as difference of pixel intensity from their respective ideal values (0 and 255). The segmentation problem has been solved using rectangular cliques of size 4 anchored on all pixels. The number of cliques in such a system is equal to the number of pixels in the image. The clique potential chosen penalizes as per the square root of number of edges present in the labeling (SQRT). This clique potential is submodular. That GC outputs optimal energy was confirmed by comparing the primal and dual values outputted. Figure 6 shows the results. The numbers below each figure are the primal energy followed by time taken in seconds. Note that the solution outputted by GC is better both in terms of the value of primal as well the error from the ground truth. Also, note that the time taken by GC is hundreds of time faster than that taken by non combinatorial methods like DD and TRWS.

Tables 4 and 5 provide details of the energy outputted and the time taken by various algorithms on 4 clique problems of various sizes. It should be noted that algorithms which output near optimal results take couple of order of magnitude more time than that taken by GC. MPI is only two to three times slower than GC but has very poor energy output. MPI seems to move into a local minima fast and does not have the ability to move out of it. DD which has good energy convergence took 20 seconds at image size of 120×120 and failed to run when image size was increased to 160×160. In terms of energy minimization, performance of KS is nearest to that of GC. For the 160×160 image KS has outputted the same optimal energy as GC (see Table 4). However, the time taken is larger by a factor of thousands. Table 6 shows relative performance with different clique sizes. For all practical purposes the upper limit on clique size is 6 for methods other than GC. On images of size 100×100 GC is hundreds of times faster than non combinatorial methods like DD.

Figure 7 reports the time comparison of IQ and GC at log scale on a fixed image and varying clique sizes .

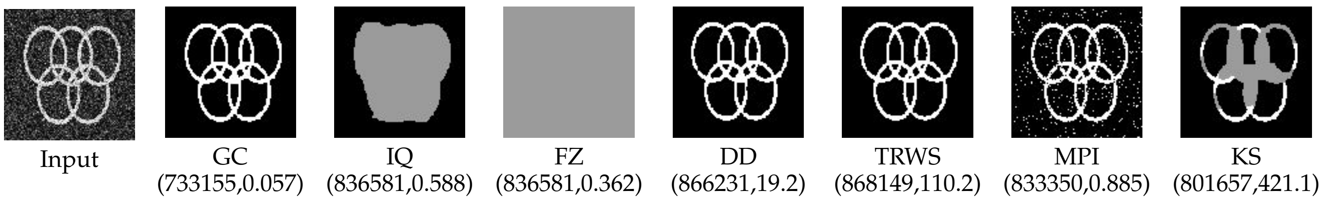


Fig. 6. Segmentation: σ of Gaussian noise added is 60. Numbers in parentheses show primal value and time taken (in seconds) by each algorithm.

Image Size	GC	DD	TRWS	MPI	KS	IQ	FZ
40×40	126680.0	139724.0	147520.0	143689.0	128518.0	128518.0	128518.0
80×80	463844.0	567254.0	566877.0	522711.0	490074.0	519967.0	519967.0
120×120	978232.0	1146288.0	1149290.0	1112566.0	980315.0	1164596.0	1181320.0
160×160	1674408.0	DNR	1898237.0	1921017.0	1674408.0	1991083.0	2091985.0

TABLE 4
Energy of the inferred solution for a 4 clique problem at different image sizes

Image Size	GC	DD	TRWS	MPI	KS	IQ	FZ
40×40	0.016	1.153	3.641	0.029	9.328	0.055	0.040
80×80	0.029	4.765	46.76	0.381	138.1	0.283	0.196
120×120	0.102	20.05	222.1	1.801	686.5	0.743	0.549
160×160	0.277	DNR	691.8	5.453	2447	1.395	0.942

TABLE 5
Inference time in seconds for a 4 clique problem

Image Size	Clique Size	DD	GC	IQ
100×100	4	15.081	0.014	0.519
100×100	6	36.683	0.103	7.398
50×50	9	44.872	0.435	206.756
50×50	10	88.577	1.102	DNR
50×50	12	400.543	7.125	DNR

TABLE 6
Time comp. at various clique sizes

The almost linear increase in the ratio with increasing clique size confirms the exponential divergence in the running time of the two algorithms that the theoretical analysis predicts. To put the impact of this exponential divergence in perspective, note that the run time of IQ for clique size of 4 is 63 ms. GC takes 25 ms on the same problem. As clique size increases to 11, IQ starts to take 235 seconds whereas the time for GC increases only to 407 ms.

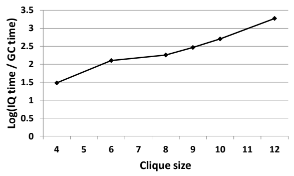


Fig. 7. GC and IQ comparison at log scale.

8 Conclusion

We consider GC to be the first step towards development of a truly practical strongly polynomial algorithm for optimizing submodular higher order energy functions that arise in problems modeled by MRF-MAP. Its primary limitation is in the multiplicative factor of 2^k that shows up in the time complexity analysis. While this limits its realistic practical (in terms of time) use on images of 500×500 to cliques of size 9, the worst case time complexity will still be less than $O(n^6)$ of [26] for $k = 16$. On submodular clique potentials GC gives optimal results and can be the basis for approximating non-submodular clique potentials for real life problems with nonzero uniform costs. There is a generalization of GC's gadget to handle such potentials. We have reported preliminary results in [43] where we have shown that the output not only is of very good quality without significant increase in time taken, it also has the desirable property of persistency.

References

[1] N. Komodakis, N. Paragios, and G. Tziritas, "MRF Energy Minimization and Beyond via Dual Decomposition," *TPAMI*, vol. 33, no. 3, pp. 531 –552, march 2011.

[2] V. Kolmogorov, "Convergent tree-reweighted message passing for energy minimization," *TPAMI*, vol. 28, no. 10, pp. 1568–1583, 2006.

[3] H. Ishikawa, "Transformation of General Binary MRF Minimization to the First-Order Case," *TPAMI*, vol. 33, pp. 1234–1249, June 2011.

[4] A. Fix, A. Gruber, E. Boros, and R. Zabih, "A graph cut algorithm for higher-order markov random fields," in *ICCV*, 2011, pp. 1020–1027.

[5] F. Kahl and P. Strandmark, "Generalized roof duality for pseudo-boolean optimization," in *ICCV*, 2011, pp. 255–262.

[6] S. Geman and D. Geman, "Stochastic relaxation, Gibbs distributions and the Bayesian restoration of images," *TPAMI*, vol. 6, no. 6, pp. 721–741, 1984.

[7] A. Criminisi, G. Cross, A. Blake, and V. Kolmogorov, "Bilayer Segmentation of Live Video," in *CVPR*, 2006, pp. 53 – 60.

[8] C. Rother, V. Kolmogorov, and A. Blake, "GrabCut: interactive foreground extraction using iterated graph cuts," *SIGGRAPH*, vol. 23, pp. 309–314, 2004.

[9] U. Mudenagudi, S. Banerjee, and P. Kalra, "Space-Time Super-Resolution Using Graph-Cut Optimization," *TPAMI*, vol. 33, no. 5, pp. 995 –1008, 2011.
[10] V. Kwatra, A. Schödl, I. Essa, G. Turk, and A. Bobick, "Graph-cut Textures: Image and Video Synthesis Using Graph Cuts," *SIGGRAPH*, vol. 22, no. 3, pp. 277–286, 2003.
[11] Y. Boykov, O. Veksler, and R. Zabih, "Fast Approximate Energy Minimization via Graph Cuts," *TPAMI*, vol. 23, pp. 1222–1239, 2001.
[12] R. Qian and T. Huang, "Object detection using hierarchical MRF and MAP estimation," in *CVPR*, 1997, pp. 186 –192.
[13] D. M. Greig, B. T. Porteous, and A. H. Seheult, "Exact maximum a posteriori estimation for binary images," *J. R. Statist. Soc.*, vol. 51, pp. 271–279, 1989.
[14] D. Sontag, A. Globerson, and T. Jaakkola, *Introduction to Dual Decomposition for Inference*. MIT Press, 2011.
[15] T. Hazan and A. Shashua, "Norm-product belief propagation: primal-dual message-passing for approximate inference," *IEEE Trans. Inf. Theor.*, vol. 56, no. 12, pp. 6294–6316, 2010.
[16] J. Pearl, *Probabilistic reasoning in intelligent systems: networks of plausible inference*. Morgan Kaufmann Publishers Inc., 1988.
[17] N. Komodakis and N. Paragios, "Beyond pairwise energies: Efficient optimization for higher-order MRFs," in *CVPR*, vol. 0, 2009, pp. 2985–2992.
[18] T. Werner, "High-arity interactions, polyhedral relaxations, and cutting plane algorithm for soft constraint optimisation (MAP-MRF)," in *CVPR*, 2008, pp. 1–8.
[19] V. Kolmogorov and R. Zabih, "What energy functions can be minimized via graph cuts?" in *ECCV*, 2002, pp. 65–81.
[20] C. Rother, P. Kohli, W. Feng, and J. Jia, "Minimizing sparse higher order energy functions of discrete variables," in *CVPR*, 2009, pp. 1382–1389.
[21] D. Freedman and P. Drineas, "Energy minimization via graph cuts: Settling what is possible," in *CVPR*, 2005, pp. 939–946.
[22] S. Ramalingam, C. Russell, L. Ladicky, and P. H. S. Torr, "Efficient Minimization of Higher Order Submodular Functions using Monotonic Boolean Functions," *CoRR*, vol. abs/1109.2304, 2011.
[23] A. C. Gallagher, D. Batra, and D. Parikh, "Inference for order reduction in Markov random fields." in *CVPR*, 2011, pp. 1857–1864.
[24] V. Kolmogorov and C. Rother, "Minimizing nonsubmodular functions with graph cuts-a review," *TPAMI*, vol. 29, no. 7, pp. 1274–1279, 2007.
[25] V. Kolmogorov, "Generalized roof duality and bisubmodular functions," *Discrete Applied Mathematics*, vol. 160, no. 4-5, pp. 416–426, 2012.
[26] S. Iwata and J. B. Orlin, "A simple combinatorial algorithm for submodular function minimization," in *SODA*, 2009, pp. 1230–1237.
[27] A. Schrijver, "A combinatorial algorithm minimizing submodular functions in strongly polynomial time," *J. Comb. Theory Ser. B*, vol. 80, pp. 346–355, November 2000.
[28] V. Kolmogorov, "Minimizing a sum of submodular functions," *Discrete Appl. Math.*, vol. 160, no. 15, pp. 2246–2258, 2012.
[29] S. Iwata, "A capacity scaling algorithm for convex cost submodular flows," in *SODA*, 1996, pp. 482–489.
[30] C. Arora, S. Banerjee, P. Kalra, and S. Maheshwari, "Generic Cuts: An efficient optimal algorithm for submodular MRF-MAP problems with higher order cliques," in *ECCV*, 2012, pp. 17–30.
[31] J. Kleinberg and E. Tardos, "Approximation algorithms for classification problems with pairwise relationships: metric labeling and markov random fields," *J. ACM*, vol. 49, pp. 616–639, September 2002.
[32] C. Chekuri, S. Khanna, J. S. Naor, and L. Zosin, "Approximation algorithms for the metric labeling problem via a new linear programming formulation," 2001, pp. 109–118.
[33] N. Komodakis and G. Tziritas, "Approximate labeling via graph cuts based on linear programming," *TPAMI*, vol. 29, no. 8, pp. 1436–1453, 2007.
[34] V. Vazirani, *Approximation algorithms*. Springer, 2001.
[35] A. V. Goldberg and R. E. Tarjan, "A new approach to the maximum flow problem," in *STOC*, 1986, pp. 136–146.
[36] J. Edmonds and R. M. Karp, "Theoretical improvements in algorithmic efficiency for network flow problems," *J. ACM*, vol. 19, pp. 248–264, April 1972.
[37] D. Koller and N. Friedman, *Probabilistic Graphical Models: Principles and Techniques*, ser. Adaptive Computation and Machine Learning. MIT Press, 2009.
[38] H. Ishikawa, "Higher-order clique reduction software version 1.02," http://www.f.waseda.jp/hfs/software.html.
[39] V. Kolmogorov, "QPBO version 1.3," http://pub.ist.ac.at/∼vnk/software.html.
[40] A. Fix, "Higher order energy reduction, Last checked Feb 7, 2013," www.cs.cornell.edu/∼afix/.
[41] P. Strandmark, "Pseudo-Boolean Optimization - Last Checked Feb 25, 2013," http://www.maths.lth.se/matematiklth/personal/petter/pseudoboolean.php.
[42] S. Gould, "Darwin 1.1.2," http://drwn.anu.edu.au/.
[43] C. Arora, S. Banerjee, P. Kalra, and S. Maheshwari, "Fast approximate inference in higher order MRF-MAP labeling problems," in *CVPR*, 2014, pp. 1338–1345.

Chetan Arora received his Bachelor's degree in Electrical Engineering 1999 and the Ph.D. degree in Computer Sc. in 2012 both from IIT Delhi. From 2000-2009 he was an entrepreneur involved in setting up companies working on various computer vision based products. From 2012 to 2014 he was post doctoral researcher at Hebrew University. He is currently an Assistant Professor at IIIT Delhi. His broad areas of research include computer vision and discrete optimization.

Subhashis Banerjee is a professor in the Department of Computer Science and Engineering at the Indian Institute of Technology, Delhi, India. His broad areas of research include computer vision, robotics, real-time systems, image processing, and pattern recognition.

Prem Kumar Kalra is a professor in the Department of Computer Science and Engineering at the Indian Institute of Technology, Delhi, India. His broad areas of research include computer graphics, animation, image processing, and pattern recognition.

S.N. Maheshwari is a Professor Emeritus in the Department of Computer Science and Engineering at the Indian Institute of Technology, Delhi, India. His broad areas of research include combinatorial optimization, algorithms and complexity, and computational biology.

APPENDIX

SUPPLEMENTARY MATERIAL

Proof of Lemma 4.4

Let $f = (f_{ij})$ be a flow in the flow graph and let (S, T) be any cut. Summing up the flow conservation equations for all nodes $i \in S$ we get

$$\begin{aligned} v &= \sum_{i \in S} (\sum_{j} f_{ij} - \sum_{j} f_{ji}) \\ &= \sum_{i \in S} \sum_{j \in S} (f_{ij} - f_{ji}) + \sum_{i \in S} \sum_{j \in T} (f_{ij} - f_{ji}) \\ &= \sum_{i \in S} \sum_{j \in T} (f_{ij} - f_{ji}), \qquad (19) \end{aligned}$$

where v is the flow entering t. In effect v is the net flow through any (S, T) cut. Since $f_{ji} \geq 0$

$$v \leq \sum_{i \in S} \sum_{j \in T} f_{ij}.$$

Note that flow in all edges other than conjugate edges in the (S, T) cut is less than or equal to their capacities. Also, since the sum of flows in all the conjugate edges covered by a DFC is less than or equal to the cost of DFC, the sum of flows in all the conjugate edges in the (S, T) cut is less than or equal to the capacity of the smallest cost Cut Cover. Therefore, it follows that flow in the gadget based flow graph is always less than or equal to the capacity of any (S, T) cut.

Proof of Lemma 4.5

1) Note that any node, say p, in a gadget can always send flow to m. In case there is zero flow in the conjugate edge pair incident at node p or in the edge $n \to p$, then by definition, there is infinite capacity in the edge $p \to m$. If there is non zero flow in $n \to p$, then there is non zero residual capacity in $p \to n$ and $n \to m$ at-least equal to the flow in $n \to p$, and flow can thus be sent from p to$m$ through n. Consider a gadget *on* the cut. By definition at-least one of the nodes in it is in S. Since there is always non zero residual capacity on the edge from any node to m, m must be in S as well. Also, since the edge $m \to n$ has infinite capacity by definition, n must also be in S.
2) Lemma 4.5(1) states that for all such gadgets, m must be in S. Any flow in $p \to m$ for some p in T results in non zero residual capacity in the edge $m \to p$ and thus creates an st augmenting path. This violates the assumption that flow is maximum.
3) Equation (19) states that:

$$f = \sum_{i \in S} \sum_{j \in T} (f_{ij} - f_{ji}).$$

However, Lemma 4.5(2) implies that when the flow is maximum, there is no flow from any node in T to a node in S. Therefore:

$$f = \sum_{i \in S} \sum_{j \in T} (f_{ij}).$$

Proof of Lemma 4.6

Since $W_c(\cdot)$ is submodular, we have:

$$W_{\mathbf{c}}(\mathbf{l}_{\mathbf{c}}^{'}) + W_{\mathbf{c}}(\mathbf{l}_{\mathbf{c}}^{''}) \geq W_{\mathbf{c}}(\mathbf{l}_{\mathbf{c}}^{'} \cup \mathbf{l}_{\mathbf{c}}^{''}) + W_{\mathbf{c}}(\mathbf{l}_{\mathbf{c}}^{'} \cap \mathbf{l}_{\mathbf{c}}^{''}).$$

For two tight DFCs corresponding to $\mathbf{l}_{\mathbf{c}}^{'}$ and $\mathbf{l}_{\mathbf{c}}^{''}$ we have:

$$\sum_{e \in \mathbf{l}_{\mathbf{c}}^{'}} f_e = W_{\mathbf{c}}(\mathbf{l}_{\mathbf{c}}^{'}), \text{ and } \sum_{e \in \mathbf{l}_{\mathbf{c}}^{''}} f_e = W_{\mathbf{c}}(\mathbf{l}_{\mathbf{c}}^{''}),$$

where e refers to a conjugate edge pair and notation $e \in \mathbf{l}_{\mathbf{c}}^{'}$ refers to a conjugate edge pair e participating in DFC corresponding to $\mathbf{l}_{\mathbf{c}}^{'}$. f_e denotes the effective flow in conjugate edge pair e. Therefore

$$\sum_{e \in \mathbf{l}_{\mathbf{c}}^{'}} f_e + \sum_{e \in \mathbf{l}_{\mathbf{c}}^{''}} f_e \geq W_{\mathbf{c}}(\mathbf{l}_{\mathbf{c}}^{'} \cup \mathbf{l}_{\mathbf{c}}^{''}) + W_{\mathbf{c}}(\mathbf{l}_{\mathbf{c}}^{'} \cap \mathbf{l}_{\mathbf{c}}^{''}).$$

Since cost of a DFC is always greater than or equal to the sum of flows in the conjugate edges participating in it,

$$W_{\mathbf{c}}(\mathbf{l}_{\mathbf{c}}^{'} \cap \mathbf{l}_{\mathbf{c}}^{''}) \geq \sum_{e \in (\mathbf{l}_{\mathbf{c}}^{'} \cap \mathbf{l}_{\mathbf{c}}^{''})} f_e.$$

Therefore

$$\sum_{e \in \mathbf{l}_{\mathbf{c}}^{'}} f_e + \sum_{e \in \mathbf{l}_{\mathbf{c}}^{''}} f_e - \sum_{e \in (\mathbf{l}_{\mathbf{c}}^{'} \cap \mathbf{l}_{\mathbf{c}}^{''})} f_e \geq W_{\mathbf{c}}(\mathbf{l}_{\mathbf{c}}^{'} \cup \mathbf{l}_{\mathbf{c}}^{''}).$$

The l.h.s. of the above is nothing but the sum of flow in the conjugate edges covered by the DFC corresponding to $\mathbf{l}_{\mathbf{c}}^{'} \cup \mathbf{l}_{\mathbf{c}}^{''}$.

$$\sum_{e \in (\mathbf{l}_{\mathbf{c}}^{'} \cup \mathbf{l}_{\mathbf{c}}^{''})} f_e \geq W_{\mathbf{c}}(\mathbf{l}_{\mathbf{c}}^{'} \cup \mathbf{l}_{\mathbf{c}}^{''}).$$

Since cost of a DFC is always greater than or equal to the sum of flows in the conjugate edges participating in it:

$$\sum_{e \in (\mathbf{l}_{\mathbf{c}}^{'} \cup \mathbf{l}_{\mathbf{c}}^{''})} f_e = W_{\mathbf{c}}(\mathbf{l}_{\mathbf{c}}^{'} \cup \mathbf{l}_{\mathbf{c}}^{''}).$$

The DFC corresponding to $(\mathbf{l}_{\mathbf{c}}^{'} \cup \mathbf{l}_{\mathbf{c}}^{''})$ is therefore tight. Tightness for DFC corresponding to $(\mathbf{l}_{\mathbf{c}}^{'} \cap \mathbf{l}_{\mathbf{c}}^{''})$ can be similarly proved.

Proof of Lemma 5.1

We describe the proof using variable of type $V_{\mathbf{c},p,a}$. The argument for $V_{\mathbf{c},p,b}$ can be derived symmetrically.

Consider a reparametrization by δ step using variable $V_{\mathbf{c},p,a}$ corresponding to ball p_a. Since, potential function was submodular prior to this transformation we had

$$W(\mathbf{l}_{\mathbf{c}}^{'}) + W(\mathbf{l}_{\mathbf{c}}^{''}) \geq W(\mathbf{l}_{\mathbf{c}}^{'} \cup \mathbf{l}_{\mathbf{c}}^{''}) + W(\mathbf{l}_{\mathbf{c}}^{'} \cap \mathbf{l}_{\mathbf{c}}^{''}).$$

Note that there is an inequality of the above type for any two labelings $\mathbf{l}_\mathbf{c}^{'}$ and $\mathbf{l}_\mathbf{c}^{''}$ on clique $\mathbf{c}$. The following cases arise:

1) p was labeled b in both $\mathbf{l}_\mathbf{c}^{'}$ and $\mathbf{l}_\mathbf{c}^{''}$
In this case the value of $W(\mathbf{l}_\mathbf{c}^{'})$, $W(\mathbf{l}_\mathbf{c}^{''})$, $W(\mathbf{l}_\mathbf{c}^{'} \cup \mathbf{l}_\mathbf{c}^{''})$ and $W(\mathbf{l}_\mathbf{c}^{'} \cap \mathbf{l}_\mathbf{c}^{''})$ does not change. Both l.h.s and r.h.s. remain the same and the inequality continues to remain satisfied.
2) p was labeled a in only one of $\mathbf{l}_\mathbf{c}^{'}$ or $\mathbf{l}_\mathbf{c}^{''}$
In this case p will be labeled in a in labeling corresponding to $\mathbf{l}_\mathbf{c}^{'} \cup \mathbf{l}_\mathbf{c}^{''}$ but not in labeling corresponding to $\mathbf{l}_\mathbf{c}^{'} \cap \mathbf{l}_\mathbf{c}^{''}$. Therefore $W(\mathbf{l}_\mathbf{c}^{'})$ or $W(\mathbf{l}_\mathbf{c}^{''})$ decreases by δ and $W(\mathbf{l}_\mathbf{c}^{'} \cup \mathbf{l}_\mathbf{c}^{''})$ decreases by δ. Since the l.h.s and r.h.s of the inequality decrease by the same amount (δ) the inequality continues to hold.
3) p was labeled a in both $\mathbf{l}_\mathbf{c}^{'}$ and $\mathbf{l}_\mathbf{c}^{''}$
In this case p will be labeled a in labeling corresponding to $\mathbf{l}_\mathbf{c}^{'} \cup \mathbf{l}_\mathbf{c}^{''}$ as well as $\mathbf{l}_\mathbf{c}^{'} \cap \mathbf{l}_\mathbf{c}^{''}$. Therefore $W(\mathbf{l}_\mathbf{c}^{'})$, $W(\mathbf{l}_\mathbf{c}^{''})$, $W(\mathbf{l}_\mathbf{c}^{'} \cup \mathbf{l}_\mathbf{c}^{''})$ and $W(\mathbf{l}_\mathbf{c}^{'} \cap \mathbf{l}_\mathbf{c}^{''}))$ decrease by amount δ. The l.h.s and r.h.s. of the inequality decrease by the same amount (2δ) and so the inequality remains satisfied.

Therefore normalizing any V variable is nothing but an equivalent energy transformation which maintains submodularity property. For a clique of size k, there are 2^k DFCs for a clique and time taken for normalization per clique will be $O(2^k)$. Under the assumption that the number of cliques is of the same order as n, the number of pixels, the total normalization time is $O(n2^k)$.

Proof of Lemma 6.1

Suppose the intersection of all tight DFC's covering $n \rightarrow p$ contains at least one more edge. This edge is either an n type edge $n \rightarrow q$ or an m type edge $r \rightarrow m$. If the edge is $n \rightarrow q$ then path fragment $q \rightarrow n \rightarrow p$ has residual capacity greater than zero. This is so because all tight DFCs covering receiving edge of the path fragment also cover the sending node. Therefore a path from s to t which contains the path fragment $q \rightarrow n \rightarrow p$ (with all other path fragments in the path having residual capacities greater than zero) will be an augmenting path. On the other hand if the intersection of all tight DFCs covering $n \rightarrow p$ contains no other edge, then any path fragment $q \rightarrow n \rightarrow p$ will violate the tight DFCs covering $n \rightarrow p$ and hence can not be in any augmenting path.

Proof of Lemma 6.2

Suppose flow is not maximum and no augmenting path exists in the residual graph consistent with that flow. Let $\mathbf{S}$ be the set of nodes reachable [9] from s at that stage and let $\mathbf{T}$ be the rest. Following the arguments given in Lemma 4.5, it is always possible to send flow from a pixel node to node m, and from m to n node. Therefore, the only reason of for no augmenting path existing is because all conjugate edges out of n node to nodes in T are saturated. For all such saturated conjugate edges Lemma 6.1 applies and all tight DFCs covering these edges do not cover edges in S (otherwise a path fragment from S node to T would be available), or conversely can only cover edges of nodes in T. Using arguments given for proving Theorem 4.7, we can find a cut cover covering all such edges once. All DFCs of such a cut must be tight and the sum of flows in the edges covered by these DFCs must be equal to the cost of cut. that is flow in the cut will therefore be equal to cost of cut. Lemma 4.4 states that flow must always be less than equal to cost of the cut. The flow must therefore be maximum contradicting the initial assumption.

[9]Recall that a node p is said to be reachable from q, if non zero flow can be pushed from q to p without violating any DFC

Proof of Lemma 6.3

For the purposes of the proof we specify the shortest augmenting path length from s to a node x and from x to t in clique $\mathbf{c}$ after i such iterations by $\delta^i(s, \mathbf{c}, x)$ and $\delta^i(t, \mathbf{c}, x)$ respectively. We refer to the path fragment as belonging to a clique $\mathbf{c}$ to which auxiliary node of the path fragment belongs.

Note that flow in a path fragment can only affect the slacks of the DFCs corresponding to the clique to which they belongs. Consider the flow graph after the i^{th} flow augmentation. If the path fragments in the shortest augmenting path for the $(i+1)^{th}$ augmentation are not in the cliques through which the i^{th} augmenting path passed then that path must have remained unchanged during i^{th} augmentation and existed even at i^{th} augmentation. Since we always augment along a shortest path, $\delta^{i+1}(s,t) \geq \delta^i(s,t)$. For the possibility that the $(i+1)^{th}$ shortest augmenting path to be smaller than i^{th} shortest augmenting path it is necessary that there be at least one clique in which the i^{th} and $(i{+}1)^{th}$ paths use different path fragments. Call these path fragments as i^{th} and $(i+1)^{th}$ path fragments. For the $(i+1)^{th}$ path to be shorter than i^{th} path, $(i+1)^{th}$ path fragment must have become available only after sending flow through i^{th} path fragment (since if it was shorter and available before, it should have been used).

We will consider the case when the two paths share exactly one clique in which the path fragments are different in the two paths. Let that clique be $\mathbf{c}$ and let the i^{th} path fragment be from x_1 to x_2 (i.e. with x_1 as start node and x_2 as end node). One possibility is that after the i^{th} augmentation the i^{th} path fragment is saturated and the $(i{+}1)^{th}$ path fragment is from x_2 to x_1. In this case since $\delta^{i+1}(s, \mathbf{c}, x_2) \geq \delta^i(s, \mathbf{c}, x_2)$ and $\delta^{i+1}(t, \mathbf{c}, x_1) \geq \delta^i(t, \mathbf{c}, x_1)$, implies $\delta^{i+1}(s,t) \geq \delta^i(s,t)$. Therefore, assume that the $(i+1)^{th}$ path fragment is from x_3 to x_4. There can be 9 possible cases. The first four cases namely

1) $\delta^i(s,\mathbf{c},x_1) = \delta^{i+1}(s,\mathbf{c},x_3)$ and $\delta^i(t,\mathbf{c},x_2) = \delta^{(i+1)}(t,\mathbf{c},x_4)$, or
2) $\delta^i(s,\mathbf{c},x_1) = \delta^{i+1}(s,\mathbf{c},x_3)$ and $\delta^i(t,\mathbf{c},x_2) < \delta^{(i+1)}(t,\mathbf{c},x_4)$), or
3) $\delta^i(s,\mathbf{c},x_1) < \delta^{i+1}(s,\mathbf{c},x_3)$ and $\delta^i(t,\mathbf{c},x_2) < \delta^{(i+1)}(t,\mathbf{c},x_4)$), or
4) $\delta^i(s,\mathbf{c},x_1) < \delta^{i+1}(s,\mathbf{c},x_3)$ and $\delta^i(t,\mathbf{c},x_2) = \delta^{(i+1)}(t,\mathbf{c},x_4)$),

are routine. In these cases $\delta^i(s,t) \leq \delta^{i+1}(s,t)$. In cases

5) $\delta^i(s,\mathbf{c},x_1) > \delta^{i+1}(s,\mathbf{c},x_3)$ and $\delta^i(t,\mathbf{c},x_2) > \delta^{(i+1)}(t,\mathbf{c},x_4)$), or
6) $\delta^i(s,\mathbf{c},x_1) < \delta^{i+1}(s,\mathbf{c},x_3)$ and $\delta^i(t,\mathbf{c},x_2) > \delta^{(i+1)}(t,\mathbf{c},x_4)$), or
7) $\delta^i(s,\mathbf{c},x_1) = \delta^{i+1}(s,\mathbf{c},x_3)$ and $\delta^i(t,\mathbf{c},x_2) > \delta^{(i+1)}(t,\mathbf{c},x_4)$,

we need to ask why the path fragment from x_1 to x_4 was not chosen in the i^{th} iteration as that would have resulted in a shorter augmenting path from s to t. This implies that the conjugate edge $n \to x_4$ was saturated earlier being covered by some tight DFC. If flow sent through x_1 causes the slack of any such tight DFC to become non-zero, then x_1 must have been participating in that DFC and by Lemma 6.1, the flow in the path fragment from x_1 to x_4 would have been possible in the i^{th} iteration also. But that would contradict the assertion that flow in the i^{th} iteration was pushed on the shortest path. These cases, therefore, can not arise.

In cases

8) $\delta^i(s,\mathbf{c},x_1) > \delta^{i+1}(s,\mathbf{c},x_3)$ and $\delta^i(t,\mathbf{c},x_2) = \delta^{(i+1)}(t,\mathbf{c},x_4)$), or
9) $\delta^i(s,\mathbf{c},x_1) > \delta^{i+1}(s,\mathbf{c},x_3)$ and $\delta^i(t,\mathbf{c},x_2) < \delta^{(i+1)}(t,\mathbf{c},x_4)$

we need to ask similarly why the path fragment from x_3 to x_2 was not used in the i^{th} iteration. The only reason why this may not be possible is when x_2 is covered by a tight DFC not covering x_3. This tight DFC must be covering x_1, since x_1 to x_2 was used instead. Also, since the x_3 to x_4 fragment was not used before and became available in $(i+1)^{th}$ iteration after sending flow through x_1, the implication is that x_4 was covered by some tight DFC also covering x_1 and not covering x_3. From the submodularity constraint (Lemma 4.6) we observe that if the DFC covering x_1 and x_2 and the one covering x_1 and x_4 are tight then DFC covering x_1, x_2, x_4 must also be tight. This DFC must stay tight even after sending flow from x_1 to x_2 and will block any flow from x_3 to x_4 in $(i+1)^{th}$ iteration. These cases, therefore, also can not happen.