MRF Optimization

Richard Hartley, ICCV Tutorial, 2009
Thanks to Phil Torr,
Oxford Brookes University,
Yuri Boykov
University of Western Ontario

also
Manish Jethwa, Pushmeet Kohli, Pawan Kumar, Yuri Boykov, George Vogliatzsis, Chria Bishop, Bill Freeman, Peter Carr and many others.
Markov Random Fields

Pair wise Random Field Model commonly used to represent images

\[ p(x_i | \mathbf{x}) = p(x_i | \{x_k\}), \quad x_k \in N_i \]
Basic Problem of Image Labelling

1. How to find the most likely labelling of the graph.
2. Equivalently, how to label the graph with the lowest cost assignment.
3. In 2-label problem, how to separate the nodes into two sets (0 and 1) with the least cost cut of edges in the graph.
Examples of MRFs and Graph Cuts
MRF for Image Segmentation

Boykov et al. [ICCV 2001], Blake et al. [ECCV 2004]

\[
E_{\text{MRF}}(x) = \sum_{i \in S} \left( \phi(D|x_i) + \sum_{j \in N_i} (\phi(D|x_i, x_j) + \psi(x_i, x_j)) \right) + \text{const}
\]

- Unary likelihood
- Contrast Term
- Potts Model Prior

MAP solution \( x^* = \arg \min_x E(x) \)

Data (D)  Unary likelihood  Pair-wise Terms  MAP Solution
Segmentation with Alpha Matting
Grabcut demo
Scan-line stereo

Example:

- Baker & Binford 1981
- Ohta & Kanade, 1985
- Geiger at al. 1992
- Belhumeur & Mumford 1992
- Cox at al. 1996
- Scharstein & Szelisky 2001

\[ E(d_1, d_2, \ldots, d_n) = \sum_{p \in S_{left}} E_p (d_p, d_{p+1}) \]

Disparities of pixels in the scan line

Consistency
Stereo example

Independent scan-lines (via DP)

Multi-scan line (via Graph Cuts)

Ground truth
Figure 2: (Left) Schematic showing the overlapping region between two patches. (Right) Graph formulation of the seam finding problem, with the red line showing the minimum cost cut.

\[ M(s, t, A, B) = \|A(s) - B(s)\| + \|A(t) - B(t)\| \]
Multi-way graph cuts

Graph-cut textures
(Kwatra, Schodl, Essa, Bobick 2003)
Shortest paths: Texture synthesis

“Image quilting”
Efros & Freeman, 2001
Shortest paths: Texture synthesis

“Image quilting”
Efros & Freeman, 2001
s-t graph cuts for video textures

Graph-cuts video textures
(Kwatra, Schodl, Essa, Bobick 2003)

3D generalization of “image-quilting” (Efros & Freeman, 2001)
Examples of Graph-Cuts in vision

- **Image Restoration** (e.g. Greig at.al. 1989)
- **Segmentation**
  - Wu & Leahy 1993
  - Nested Cuts, Veksler 2000
- **Multi-scan-line Stereo, Multi-camera stereo**
  - Roy & Cox 1998, 1999
  - Kolmogorov & Zabih 2002, 2004
- **Object Matching/Recognition** (Boykov & Huttenlocher 1999)
- **N-D Object extraction** (photo-video editing, medical imaging)
  - Boykov & Kolmogorov 2003
  - Rother, Blake, Kolmogorov 2004
- **Texture synthesis** (Kwatra, Schodl, Essa, Bobick 2003)
- **Shape reconstruction** (Snow, Viola, Zabih 2000)
- **Motion** (e.g. Xiao, Shah 2004)
MRFs and Energy Functions
Markov Random Fields

Consider a set of random variables $\mathcal{X} = \{X_1, \ldots, X_n\}$ taking values in a set $S$ (usually a discrete set of values). Assume a **neighbourhood structure**: for each variable $X_i$ there is defined a set of variables $\mathcal{N}_i \subset \mathcal{X} - \{i\}$ such that

1. $i \in \mathcal{N}_j \Rightarrow j \in \mathcal{N}_i$.
2. $P(X_i = x_i \mid X_j = x_j; j \neq i) = P(X_i = x_i \mid X_j = x_j; j \in \mathcal{N}_i)$.

In other words the conditional probability distribution of a given variable $X_i$ depends only the values of its neighbors.

This is called a *Markov Random Field* (MRF).

**Notation.** Denote the set (or vector) of all random variables by $\mathbf{x}$ and a set of values of the random variables by $\mathbf{x}$. 
MRFs and graphs

Define an undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ such that

1. The vertices $\mathcal{V}$ are in one-to-one correspondence with the random variables $X_i$. (In fact we will refer to the vertices as $X_i$.)
2. There is an edge from $X_i$ to $X_j$ if and only if $i \in \mathcal{N}_j$.

Example: Image graph (4-connected) has cliques of size 1 and 2.

- size 1 (vertices)
- size 2 (pairs of vertices joined by an edge).

6-connected and 8-connected graphs have cliques of size 3 and 4 respectively.
Gibbs distribution - The Hammersley-Clifford Theorem

Refer to paper Geman and Geman “Stochastic Relaxation, Gibbs Distributions, and the Bayesian Relaxation Restoration of Images.”

**Theorem.** The complete probability distribution of the MRF is given by

\[ P(x = x) = \frac{1}{Z} \prod_{C \in \mathcal{C}} \exp(-E_C(x)) \]

where \( \mathcal{C} \) represents the set of all cliques in the graph of the MRF.

1. \( E_C(x) \) is an **energy function** depending only on the values of the vertices \( X_i \in C \).
2. \( Z \) is a normalizing constant.
3. Requires condition that \( P(x) \neq 0 \) for all \( x \).
Maximizing the probability

Commonly, we want to find the assignment \( x \) that maximizes the probability (the most probable state of the MRF).

We see that

\[
\log P(x = x) = \text{const} - \sum_{C \in C} E_C(x)
\]

hence

\[
\arg\max_x P(x = x) = \arg\min_x \sum_{C \in C} E_C(x)
\]

Hence, finding the most probable state of the MRF is equivalent to minimizing the **energy function**

\[
\sum_{C \in C} E_C(x)
\]

where each \( E_C(x) \) is a function only of the variables in the clique \( C \).
4-connected graph

- Each vertex connected to 4 neighbors.
- Cliques of size 1 (vertices) and 2 (edges).

8-connected graph

- Each vertex connected to 8 neighbors.
- Cliques of size 1, 2, 3 and 4.
- Energy function might still only use cliques of size 1 and 2.
Energy functions and probability
Practical considerations

Two basic problems.

- How to choose the right energy function.
- If we have the right energy functions, can we minimize it?
- The general energy minimization problem is NP-hard.
Probability distribution of an MRF

Let \( m(x) \) be a data measurement associated with an MRF \( x \).

We compute:

\[
P(x = x \mid m(x)) = \frac{P(m(x) \mid x = x)P(x = x)}{P(m(x))}
\]

\[
\approx P(m(x) \mid x = x)P(x = x),
\]

where we treat \( P(m(x)) \) as a constant.

Taking logarithms:

\[
\log(P(x \mid m(x))) = K + \sum_i \log(P(m(x_i) \mid x_i)) + \log(P(X))
\]

- \( P(m(x_i) \mid x_i = x_i) \) is the probability of the measurement of the variable \( x_i \) given that the actual label is \( x_i \).
- \( P(x = x) \) may be considered as a prior distribution for the random variable \( X \) in the absence of any data.
Interpretation

1. The unary terms in the MRF may be considered as logarithms of the measurement given the label.

2. However,

$$\log(P(X)) \neq \prod_{i,j \in \mathcal{N}} P(X_i, X_j).$$

In fact, clearly the different probabilities $P(X_i, X_j)$ are not independent.

**Question:** How do we assign binary terms so that their sum may be meaningfully interpreted as $\log(P(X))$?

Question is very difficult!
The Ising Model

1. The Ising model is the simplest model of two-label MRF.
2. Used in Physics to model magnetism.
3. Edge weight $E_{ij}(x_i, x_j) = k$ if $x_i = x_j$, and 0 otherwise. Model interaction potential of two neighboring particles.
4. Vertex weights are interpreted as action of external magnetic field.
5. 2D Zero field case solved by Lars Onsager (Nobel prize for Chemistry 1968).
6. Unsolved in closed form in the 3D case – intractible.

Ising

Onsager
Partition Function for the 2D Ising Model

- Recall
  \[ P(x) = \frac{1}{Z} e^{-E(x)} \]

- \( Z \) is called the partition function.

- The zero-field solution partition function is given by
  \[
  \frac{-\log Z}{N} = -\log 2 - \frac{1}{8\pi^2} \int_0^{2\pi} \int_0^{2\pi} \log \left[ \cosh^2 2k - \sinh 2k (\cos \eta + \cos \zeta) \right] \, d\eta d\zeta
  \]
  where \( k \) is the edge weight, \( N \) the number of nodes.

- **Critical temperature** \((T_C = 2.269)\). There is a “critical” value of the weight
  \[
  k_C = \log(1 + \sqrt{2})/2 = 0.440686
  \]

- For smaller values of \( k \), a sample is expected to have equal numbers of 1 and 0.

- For larger values of \( k \), samples are expected to be unbalanced (expected non-zero magnetization).
Samples of zero-field Ising model.

Is this a good model for natural images?
What functions can be minimized
Binary (pseudo-boolean) functions
Definition of pseudo-boolean function.

Define $\mathcal{B} = \{0, 1\}$. A pseudo-boolean function is a mapping

$$f : \mathcal{B}^n \to \mathbb{R}.$$  

Variables: $x_i$. The set of variables will be denoted $X = \{x_i; i = 1, \ldots, n\}$.

Literals: Literals are $x_i, \bar{x}_i$, where $\bar{x}_i = 1 - x_i$. Use $u$ to represent a literal. The set of literals is denoted by $L = \{x_i, \bar{x}_i; i = 1, \ldots, n\}$. 
What do energy functions represent

- Consider a cost function of the form

\[ C(x) = \sum_i E_i(x_i) + \sum_{(i,j) \in N} E_{ij}(x_i x_j) \]

- What do the terms \( E_i(x_i) \) and \( E_{ij}(x_i, x_j) \) represent?
- How should they be chosen to give the results we want?
Three ways of representing a pseudo-boolean function.

1. Tableau. This lists all $2^n$ values of the function.
2. Posiform.

$$a_0 + \sum_i a_i u_i + \sum_{i,j} a_{ij} u_i u_j + \ldots$$

where all coefficients are positive, except perhaps $a_0$.
3. Polynomial.

$$c_0 + \sum_{i=1}^n c_i x_i + \sum_{1 \leq i < j \leq n} c_{ij} x_i x_j + \ldots$$
Transformation between different forms.

Example – Tableau to posiform

<table>
<thead>
<tr>
<th>$x_1$</th>
<th>$x_2$</th>
<th>$x_3$</th>
<th>3</th>
<th>$x_1\bar{x}_2\bar{x}_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>$x_1\bar{x}_2x_3$</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>-5</td>
<td>$\bar{x}_1x_2\bar{x}_3$</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>$\bar{x}_1x_2x_3$</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>-4</td>
<td>$x_1\bar{x}_2\bar{x}_3$</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>-2</td>
<td>$x_1\bar{x}_2x_3$</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>3</td>
<td>$x_1x_2\bar{x}_3$</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>-7</td>
<td>$x_1x_2x_3$</td>
</tr>
</tbody>
</table>

Corresponding posiform computed as follows.

$$3\bar{x}_1\bar{x}_2\bar{x}_3 + 2\bar{x}_1\bar{x}_2x_3 - 5\bar{x}_1x_2\bar{x}_3 + 1\bar{x}_1x_2x_3 - 4x_1\bar{x}_2\bar{x}_3 - 2x_1\bar{x}_2x_3 + 3x_1x_2\bar{x}_3 - 7x_1x_2x_3$$

Replace the terms with negative coefficients.

$$-x_1\bar{x}_2x_3 = -(1 - \bar{x}_1)\bar{x}_2x_3$$
$$= \bar{x}_1x_2x_3 - \bar{x}_2x_3$$
$$= \bar{x}_1\bar{x}_2x_3 - (1 - x_2)x_3$$
$$= \bar{x}_1\bar{x}_2x_3 + x_2x_3 - x_3$$
$$= \bar{x}_1\bar{x}_2x_3 + x_2x_3 + \bar{x}_3 - 1$$

All terms with negative coefficients can be replaced in this way.
Transform to posiform

Replace the terms with negative coefficients.

\[-x_1 \bar{x}_2 x_3 = -(1 - x_1) \bar{x}_2 x_3\]
\[= x_1 \bar{x}_2 x_3 - x_2 x_3\]
\[= \bar{x}_1 \bar{x}_2 x_3 - (1 - x_2) x_3\]
\[= \bar{x}_1 \bar{x}_2 x_3 + x_2 x_3 - x_3\]
\[= \bar{x}_1 \bar{x}_2 x_3 + x_2 x_3 + \bar{x}_3 - 1\]

All terms with negative coefficients can be replaced in this way.

Transform to polynomial

Replace each \(\bar{x}_i\) by \(1 - x_i\) and multiply out. Example.

\[\bar{x}_1 \bar{x}_2 x_3 = (1 - x_1)(1 - x_2)x_3\]
\[= x_3 - x_1 x_3 - x_2 x_3 + x_1 x_2 x_3\]
Ambiguity of posiform representation.

\[
x_1 \bar{x_2} = (1 - \bar{x_1})(1 - x_2) \\
= 1 - \bar{x_1} - x_2 + \bar{x_1}x_2 \\
= 1 - (1 - x_1) - (1 - \bar{x_2}) + \bar{x_1}x_2 \\
= -1 + x_1 + \bar{x_2} + \bar{x_1}x_2
\]
Uniqueness of polynomial representation.

Polynomial representation is unique.

**Proof:** We can compute the coefficients of the polynomial representation by evaluating function.

Example: \( f(x) = a_0 + \sum_i a_i x_i + \sum_{ij} x_i x_j + \ldots \)

1. Determination of the constant term: \( f(0) = a_0 \).
2. Determination of \( a_i \). Evaluate at \( x_i = 1, x_j = 0 \) for \( i \neq j \), gives \( f(0, \ldots, 1, \ldots, 0) = a_0 + a_i \), allows us to determine \( a_i \).
3. Determine the coefficients \( a_{ij} \) by evaluating by \( x_i = x_j = 1 \) and \( x_k = 0 \), otherwise. Gives \( a_0 + a_i + a_j + a_{ij} \), allows us to determine \( a_{ij} \).
Quadratic functions

For a quadratic pseudo-boolean function, we can formulate an alternative form.

$$f(x) = a_0 + \sum_i a_i u_i + \sum_{1 \leq i < j \leq n} a_{ij} \bar{x}_i x_j$$

where $u_i = x_i$ or $\bar{x}_i$, and $a_i \geq 0$ for all $i$.

This representation is unique.
Cost representation

Sometimes, we express the function to be minimized in terms of costs of the form $E_{i;p}$ or $E_{ij;pq}$, where $i$ indexes the variable, and $p, q$ are boolean values, 0 or 1.

Thus, cost $E_{i;p}$ is incurred if variable $x_i$ takes value $p$, and not otherwise. Similarly, $E_{ij;pq}$ is incurred if $x_i = p$ and $x_j = q$. Thus, the costs associated with two variables $x_i$ and $x_j$ are made up of linear terms such as

$$E_{i;1} x_i + E_{i;0} \bar{x}_i$$

plus quadratic terms

$$E_{ij;00} \bar{x}_i \bar{x}_j + E_{ij;11} x_i x_j + E_{ij;01} \bar{x}_i x_j + E_{ij;10} x_i \bar{x}_j .$$

Higher-degree terms may also be expressed in a similar way.
NP hardness.

We can show that in general, minimizing a posiform is an NP hard problem (with respect to the number of variables $n$). Consider a problem in which all terms are of degree 3 and all coefficients are 1 in a posiform representation. This is

$$f(x) = \sum_{i,j,k} u_i u_j u_k$$

where $u_i$, $u_j$ and $u_k$ are literals. For example:

$$f(x_1, x_2, x_3) = x_1 \bar{x}_2 x_3 + \bar{x}_1 x_2 \bar{x}_3 + x_1 x_2 \bar{x}_3.$$ 

Clearly $f(x) \geq 0$. To determine whether $f(x) = 0$ for some $x$ is the 3-SAT problem: Can we make a boolean assignment so that all terms $u_i u_j u_k$ are zero.

Another way to see this is to observe that a posiform or polynomial representation of a general pseudo-boolean function can have up to $2^n$ terms. Thus, given an arbitrary function over $n$ variables, presented in polynomial format, it requires exponential time just to read the coefficients of the function.
Simple Graph Representability.

Let $G = (\mathcal{V}, \mathcal{W})$ be a directed weighted graph with vertices $\mathcal{V}$ and weights $\mathcal{W}$. The weights are real numbers (not necessarily positive).

We consider a pseudo-boolean function $f(x_1, \ldots, x_n)$ and consider graphs with $n + 2$ vertices. There are two vertices called 0 and 1 (or sometimes $s$ and $t$) and other vertices that are labelled $x_i$, in one-to-one correspondence with the variables. There are certain edges with weights.

A partition of the graph is a division of the vertices into $\mathcal{V}_0$ and $\mathcal{V}_1$, where $0 \in \mathcal{V}_0$ and $1 \in \mathcal{V}_1$. If $x$ is a particular value of the variable $x = (x_1, \ldots, x_n)$, then $\mathcal{V}_0(x)$ is the set of vertices $x_i$ with value 0, and $\mathcal{V}_1(x)$ is the set of vertices $x_i$ with value 1. Thus $(\mathcal{V}_0(x), \mathcal{V}_1(x))$ is a particular partition of the graph.
Graph-representable functions

The **cost** of a partition \((\mathcal{V}_0, \mathcal{V}_1)\) is the sum of weights of all edges going from \(\mathcal{V}_0\) to \(\mathcal{V}_1\). Formally,

\[
\text{Cost}(\mathcal{V}_0, \mathcal{V}_1) = \sum_{u \in \mathcal{V}_0, v \in \mathcal{V}_1} w(u, v)
\]

where \(w(u, v)\) is the weight of the edge from vertex \(u\) to \(v\).

**Definition** The function \(f(x)\) is *simple graph representable* if there is a graph \(\mathcal{G} = (\mathcal{V}, \mathcal{W})\), such that for all \(x\),

\[
f(x) = \text{Cost}(\mathcal{V}_0(x), \mathcal{V}_1(x)) .
\]

Cost of a partition is the sum of weights of edges passing from \(\mathcal{V}_0\) to \(\mathcal{V}_1\).
Cost \((x_1=0, \, x_2=x_3=1)\) = \(\Sigma\) red edges.
Representation of quadratic pseudo-boolean functions.

The easiest way to find a graph-representation of a quadratic pseudo-boolean function is as follows.

**Step 1.** Represent the function in the form

\[ f(x) = L + \sum_{i,j=1}^{n} a_{i,j} \bar{x}_i x_j \]

where \( L \) represents linear terms in literals \( x_i \) and \( \bar{x}_i \).

**Step 2.** Draw a graph with vertices labelled 0, 1 and \( x_i \), and assign edges as follows:

1. For the constant term \( a_0 \), add an edge from 0 to 1 with weight \( a_0 \).
2. For a term \( a_i x_i \), add an edge from 0 to \( x_i \) with weight \( a_i \).
3. For a term \( a_i \bar{x}_i \), add an edge from \( x_i \) to 1 with weight \( a_i \).
4. For a term \( a_{i,j} \bar{x}_i x_j \), add an edge from \( x_i \) to \( x_j \) with weight \( a_{i,j} \).
Graph Cuts

Consider the case of two segments.
Graph Cuts

\[ W(x_1, p_h) \]

\[ W(x_j, x_k) \]

\[ W(x_n, p_t) \]
Energy Minimization using Graph cuts

What really happens? Building the graph

$$E_{MRF}(a_1, a_2)$$

Source (0)

$$a_1$$  $$a_2$$

Sink (1)
Energy Minimization using Graph cuts

What really happens? Building the graph

If we cut a link from 0 to 1, incur the cost of that edge
Energy Minimization using Graph cuts

What really happens? Building the graph

\[ E_{MRF}(a_1, a_2) = 2a_1 \]
Energy Minimization using Graph cuts

What really happens? Building the graph

$$E_{MRF}(a_1, a_2) = 2a_1 + 5\bar{a}_1$$
Energy Minimization using Graph cuts

What really happens? Building the graph

\[ E_{MRF}(a_1, a_2) = 2a_1 + 5\bar{a}_1 + 9a_2 + 4\bar{a}_2 \]
Energy Minimization using Graph cuts

What really happens? Building the graph

\[ E_{\text{MRF}}(a_1, a_2) = 2a_1 + 5\bar{a}_1 + 9a_2 + 4\bar{a}_2 + 2a_1\bar{a}_2 \]
Energy Minimization using Graph cuts

What really happens? Building the graph

\[ E_{\text{MRF}}(a_1, a_2) = 2a_1 + 5\bar{a}_1 + 9a_2 + 4\bar{a}_2 + 2a_1\bar{a}_2 + \bar{a}_1a_2 \]
Energy Minimization using Graph cuts

What really happens? Building the graph

\[ E_{\text{MRF}}(a_1, a_2) = 2a_1 + 5\bar{a}_1 + 9a_2 + 4\bar{a}_2 + 2a_1\bar{a}_2 + \bar{a}_1a_2 \]
Energy Minimization using Graph cuts

What really happens? Building the graph

\[ E_{\text{MRF}}(a_1, a_2) = 2a_1 + 5\bar{a}_1 + 9a_2 + 4\bar{a}_2 + 2a_1\bar{a}_2 + \bar{a}_1a_2 \]

Cost of st-cut = 11
\[ E_{\text{MRF}}(1,1) = 11 \]
Energy Minimization using Graph cuts

What really happens? Building the graph

\[ E_{\text{MRF}}(a_1, a_2) = 2a_1 + 5\bar{a}_1 + 9a_2 + 4\bar{a}_2 + 2a_1\bar{a}_2 + \bar{a}_1a_2 \]

\[ a_1 = 1 \quad a_2 = 0 \]

Cost of st-cut = 8

\[ E_{\text{MRF}}(1, 0) = 8 \]
Energy Minimization using Graph cuts

What really happens? Building the graph

\[ E_{MRF}(a_1, a_2) = 2a_1 + 5\bar{a}_1 + 9a_2 + 4\bar{a}_2 + 2a_1\bar{a}_2 + \bar{a}_1a_2 \]

+ CONSTANT TERM K

\[ a_1 = 1 \quad a_2 = 0 \]

Cost of st-cut = 8

\[ E_{MRF}(1, 0) = 8 \]
Notes:

1. An edge from vertex $u$ to vertex $v$ with weight $a$ represents the term $a\overline{u}v$, where $\overline{0} = 1$.

2. It is unnecessary to have both terms $x_i$ and $\bar{x}_i$ occurring. Furthermore, non-zero linear terms can be of the form $a_ix_i$ with $a_i > 0$, or $a_i\bar{x}_i$ with $a_i > 0$.

3. It is unnecessary to have both $\bar{x}_i x_j$ and $\bar{x}_j x_i$ occurring. Thus, the function can be written as

$$f(x) = L + \sum_{1 \leq i < j \leq n} a_{ij}\bar{x}_i x_j.$$
Flow on a graph.

Given a graph with vertices $v_i$, a flow is a function $\phi : V \times V \rightarrow \mathbb{R}$ such that

1. $\phi(v_i, v_j) = -\phi(v_j, v_i)$
2. For all $i$, $\sum_j \phi(v_i, v_j) = 0$.

Think of Kirchoff’s Current Law. Total flow into a vertex equals total flow out.
Permissible Flow

Later will will be interested in permissible flow on weighted graphs which will satisfy one other axiom.

1. If $w_{ij}$ is the weight of an edge from vertex $v_i$ to $v_j$, then $\phi(v_i, v_j) \leq w_{ij}$ for all $i, j$.

It is easily seen that there is no permissible flow on a weighted graph unless $w_{ij} + w_{ji} \geq 0$ for all $i, j$. In fact this is a necessary and sufficient condition.
Graph Cuts Basics
(simple 2D example)

Goal: divide the graph into two parts separating red and blue nodes

Red/blue nodes can be “identified” into two super nodes (terminals)
Graph Cuts Basics
(simple 2D example)

Goal: divide the graph into two parts separating red and blue nodes

- Cut cost is a sum of severed edge weights
- Minimum cost $s$-$t$ cut can be found in polynomial time
“Augmenting Paths”

- Find a path from S to T along non-saturated edges
- Increase flow along this path until some edge saturates

A graph with two terminals
“Augmenting Paths”

- Find a path from S to T along non-saturated edges
  - Increase flow along this path until some edge saturates
  - Find next path…
  - Increase flow…
“Augmenting Paths”

- Find a path from $S$ to $T$ along non-saturated edges
- Increase flow along this path until some edge saturates

Iterate until ... all paths from $S$ to $T$ have at least one saturated edge

A graph with two terminals

MAX FLOW $\Leftrightarrow$ MIN CUT
Max flow algorithm

Cost = 2

Cost = 3

Cost = 1

$x_1 = x_2 = 0$
$x_3 = 1$

Total Cost = 6
Implementation notes (sequential version)

- empirical comparison of different versions of \textit{augmenting paths} and \textit{push-relabel} algorithms on grid-graphs typical in vision
- “tuned” version of augmenting paths is proposed (\textit{freely available implementation})
- graph cuts can be used for problems in vision in near real time
- empirical complexity is near linear with respect to image size
Summary.

1. All quadratic pseudo-boolean functions \( f(x) \) can be expressed as graphs \( G = (\mathcal{V}, \mathcal{W}) \) such that the function value is equal to the cost of the corresponding partition:

\[
f(x) = \text{Cost}(\mathcal{V}_0(x), \mathcal{V}_1(x))
\]

2. If all weights are positive, then the min-cut on the graph (minimum of the function) is equal to the maximal permissible flow.

3. A graph with weights \( w_{uv} \) can be reparametrized to a graph with non-negative weights if and only if

\[
w(x_i, x_j) + w(x_j, x_i) \geq 0
\]

for all pairs of variables \( x_i, x_j \).

4. If all the weights are non-negative, then

\[
\text{mincut} = \min_x f(x) = \text{maxflow}
\]

and the minimization problem can be solved using a max-flow algorithm in polynomial time.
**Regular functions.**

A quadratic function is **regular** (or **submodular**) if it can be written in the form

\[
f(x_1, \ldots, x_n) = a_0 + \sum_{i=0}^{n} a_i x_i - \sum_{1 \leq i < j \leq n} a_{ij} x_i x_j \tag{1}
\]

with \( a_{ij} \geq 0 \) for all \( i, j \).

A quadratic submodular function can be written as a posiform

\[
a_0 + \sum_u a_u u + \sum_{i,j} a_{ij} x_i x_j \tag{2}
\]

and hence is representable by a graph with non-negative weights.
Condition for sub-modularity

If the function is written in terms of $E_{ij;pq}$, then

$$E_{ij;00} \bar{x}_i \bar{x}_j + E_{ij;11} x_i x_j + E_{ij;01} \bar{x}_i x_j + E_{ij;10} x_i \bar{x}_j$$

$$= L + (E_{ij;00} + E_{ij;11} - E_{ij;01} - E_{ij;10}) x_i x_j$$

where $L$ are linear terms.

Hence,

$$a_{ij} = E_{ij;01} + E_{ij;10} - (E_{ij;00} + E_{ij;11})$$

Function is regular if

$$E_{ij;01} + E_{ij;10} - (E_{ij;00} + E_{ij;11}) \geq 0$$
**Theorem.** A quadratic pseudo-boolean function can be minimized using max-flow on its simple graph representation if and only if it is of the form (1) or equivalently (2).

**Method.**

1. Write the function in the form (2).
2. Coefficients in the posiform are exactly the edges on the graph.

**Example.**

\[
7 + 7x_1 + 3\bar{x}_2 + x_3 - 5x_1x_2 - 2x_2x_3 + 4x_1\bar{x}_3
\]
\[
= 1 + 7x_1 + 8\bar{x}_2 + \bar{x}_3 + 5\bar{x}_1x_2 + 2\bar{x}_2x_3 + 4x_1\bar{x}_3
\]
**General Regular functions.**

**Definition.** A pseudo-boolean function of any degree is called regular if any restriction to a two-variable function is regular, hence of the form \( L + a_{ij}x_i \overline{x}_j \) with \( L \) linear and \( a_{ij} \geq 0 \).

Thus, set all variables but two to given values, you get a function of two variables, which must be regular.

**Example.**

\[-4x_1x_2 - 3x_1x_3 - 2x_2x_3 + 2\overline{x}_1x_2x_3 + 3x_1\overline{x}_2x_3 + 4x_1x_2\overline{x}_3\]

1. Set \( x_3 = 1 \) gives

\[-4x_1x_2 + 2\overline{x}_1x_2 + 3x_1\overline{x}_2 + L\]

\[= -9x_1x_2 + L \text{ which is submodular}\]

2. Set \( x_3 = 0 \) gives \(-8x_1x_2 + L\)

3. Set \( x_2 = 0, 1, \) and \( x_1 = 0, 1 \) and test if submodular.
Cubic terms
Cubic functions.

Observation A cubic pseudo-boolean function can be written uniquely in the form

\[ L + \sum_{1 \leq i < j \leq n} a_{ij} x_i x_j + \sum_{1 \leq i < j < k \leq n} a_{ijk} u_i u_j u_k \]  \hspace{1cm} (1)

such that

1. \( L \) is a linear posiform without terms in both \( x_i \) and \( \bar{x}_i \).
2. For each \( i, j, k \), either \( u_i u_j u_k = x_i x_j x_k \), or \( u_i u_j u_k = \bar{x}_i \bar{x}_j \bar{x}_k \).
3. \( a_{ijk} \leq 0 \) for all \( i, j, k \).
4. Coefficients \( a_{ij} \) may be positive or negative.
5. The coefficients are uniquely determined by these conditions.

Proof. First, get the cubic terms in the right form, then the quadratic terms, and finally the linear and constant terms.
**Theorem.** The cubic posiform in the form (??) is submodular if and only if the coefficients \( a_{ij} \) of the quadratic terms are all negative.

**Proof.** First, we show that this form is submodular. For two indices \( i, j \), a restriction to \( x_i \) and \( x_j \) will be of the form

\[
L + a_{ij} x_i x_j + \sum_{ij} b_{ij} x_i x_j + \sum_{ij} c_{ij} \bar{x}_i \bar{x}_j
\]

where the \( b_{ij} \) and \( c_{ij} \) are sums of some of the coefficients \( a_{ijk} \) and hence are negative. However, we may write

\[
\sum_{ij} c_{ij} \bar{x}_i \bar{x}_j = L + \sum_{ij} c_{ij} x_i x_j
\]

and so the quadratic terms of the form \( \bar{x}_i x_j \) have negative coefficients.
Conversely, suppose that the function is submodular. Consider two indices $i, j$. We wish to show that $a_{ij} \leq 0$. We restrict the function $f(x)$ by assigning values to each $x_k$ except $x_i$ and $x_j$. This gives $L + a_{ij}x_ix_j + C(x)$ where $C(x)$ represents cubic terms.

We wish to make an assignment to all other $x_k$ so that the cubic terms vanish. There may exist a cubic term $a_{ijk}x_ix_jx_k$ or $a_{ijk}x_i\overline{x}_j\overline{x}_k$ but not both.

Suppose a cubic term $a_{ijk}x_ix_jx_k$. Then setting $x_k = 0$ will cause this term to vanish.

If term $a_{ijk}x_i\overline{x}_j\overline{x}_k$ occurs, then setting $x_k = 1$ will cause this term to vanish.

Thus, for this assignment of variables $x_k$, the function $f(x)$ reduces to $L + a_{ij}x_ix_j$. Since this must be submodular, $a_{ij} \leq 0$. 
Graph representation of cubic functions.

Given a function of the form \((??)\), we show how to represent the function as a graph. This is related closely to reducing the function in some sense to a quadratic function.

Handling the cubic terms: observe that for all values of \(x_i, x_j\), and \(x_k\),

\[
-x_i x_j x_k = \min_y (\bar{x}_i + \bar{x}_j + \bar{x}_k - 1)y
\]

\[
= \min_y (\bar{x}_i + \bar{x}_j + \bar{x}_k)y + \bar{y} - 1
\]

So, given a term \(-a_{ijk} x_i x_j x_k\), with \(a_{ijk} > 0\), replace by

\[
a_{ijk} (\bar{x}_i y + \bar{x}_j y + \bar{x}_k y + \bar{y})
\]

which differs by a constant.

Similarly, replace term \(-a_{ijk} \bar{x}_i \bar{x}_j \bar{x}_k\), with \(a_{ijk} > 0\) by

\[
a_{ijk} (x_i \bar{y} + x_j \bar{y} + x_k \bar{y} + y).
\]
Graph for term $-x_i x_j x_k$

Graph for term $-\bar{x}_i \bar{x}_j \bar{x}_k$
Higher Order Cliques

The construction for cubic terms carries on to higher order terms of the form

\[ ax_1 x_2 \ldots x_n \text{ or } a \overline{x}_1 \overline{x}_2 \ldots \overline{x}_n \]

with \( a < 0 \).

Thus we can assign a bonus (decreased cost) if all nodes \( x_1 \) to \( x_n \) are assigned the same value.

Graph for term \(-x_i x_j x_k\)

Graph for term \(-\overline{x}_I \overline{x}_J \overline{x}_K\)
Higher Order Energy Functions

\[ E(x) = \sum_{i} \psi_i(x_i) + \sum_{i \in \mathcal{V}, j \in \mathcal{N}_i} \psi_{ij}(x_i, x_j) + \sum_{c \in \mathcal{C}} \psi_c(x_c) \]

Unary

Pairwise

Higher order

MRF for Image Denoising

Original

Pairwise MRF

Higher order MRF

Images Courtesy: Lan et al. ECCV06
Other algorithms
The Roof-dual Algorithm
Graph representation of non-submodular functions.

First, we represent a general quadratic pseudo-boolean function.

\[ 2x_1 + 3x_3 + x_4 + x_5 + x_2\bar{x}_3 + x_4\bar{x}_5 + 2\bar{x}_1\bar{x}_2 + 2\bar{x}_3\bar{x}_4 + 2\bar{x}_3\bar{x}_4 \]
After maximum-flow algorithm, we can reduce it to a form where there is no path from 0 to 1.
What happens for submodular functions?

For a submodular function

$$f(x) = L + \sum_{1 \leq i < j \leq n} \bar{x}_i x_j$$

the graph separates into two parts, disjoint except at 0 and 1.
Max-flow provides a partial solution.

Nodes connected to the vertex 0 will have value 0 in any optimal solution.

2-SAT labelling

- Remaining unlabelled nodes form an expression with only quadratic terms.
- Follow up with Tarjan’s algorithm for partial labelling of the (?) nodes.
- Typically labels over 90% of nodes.
- Some nodes will remain unlabelled.
\[ a_{ij} u_i u_j \equiv \frac{1}{2} a_{ij}(\bar{u}_i \rightarrow u_j) + \frac{1}{2} a_{ij}(\bar{u}_j \rightarrow u_i) \]

\[ f(x) = -4 + 4x_1 + 4\bar{x}_3 + 4\bar{x}_4 + 2x_2x_3 + 2x_2\bar{x}_4 + 4x_3x_4 \]
The Elimination Algorithm
Elimination.

Consider the problem of minimizing a pseudo-boolean function $f_0(x_1, x_2, \ldots, x_n)$. We may write

$$f_0(x_1, x_2, \ldots, x_n) = x_1 \Delta(x_2, \ldots, x_n) + h(x_2, \ldots, x_n).$$

We call $\Delta$ the derivative of $f_0$ with respect to $x_1$.

Now, suppose values of $x_2, \ldots, x_n$ are given, and let

$$x_1^* = \arg\min_{x_1} f_0(x_1, x_2, \ldots, x_n).$$

We see that

$$x_1^* = \begin{cases} 1 & \text{if } \Delta(x_2, \ldots, x_n) < 0 \\ 0 & \text{if } \Delta(x_2, \ldots, x_n) > 0 \end{cases}$$

and if $\Delta(x_2, \ldots, x_n) = 0$, then $x_1^*$ can have either value.
Back substitution

Note, $x_1^*$ is a function of $(x_2, \ldots, x_n)$. Substitute back $x_1^*$ in $f_0$ and define

$$f_1(x_2, \ldots, x_n) = \min_{x_1} f_0(x_1, \ldots, x_n)$$
$$= x_i^* \Delta(x_2, \ldots, x_n) + h(x_2, \ldots, x_n)$$

and

$$\min_{x_1, \ldots, x_n} f_0(x_1, \ldots, x_n) = \min_{x_2, \ldots, x_n} \min_{x_1} f_0(x_1, \ldots, x_n)$$
$$= \min_{x_2, \ldots, x_n} f_1(x_2, \ldots, x_n).$$
Example.

\[ f_0(x_1, x_2, x_3) = 4x_1 + 3x_2 - 2x_3 + 2x_1x_2 - 5x_1x_3 + 7x_2x_3 + 3x_1x_2x_3 \]
\[ = x_1(4 + 2x_2 - 5x_3 + 3x_2x_3) + (3x_2 - 2x_3 + 7x_2x_3) \]
\[ = x_1 \Delta(x_2, x_2) + h(x_2, x_3) \]

Then, we may compute

<table>
<thead>
<tr>
<th>(x_2)</th>
<th>(x_3)</th>
<th>(\Delta)</th>
<th>(x_1^*)</th>
<th>(x_1^* \Delta)</th>
<th>term</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>(\bar{x}_2 \bar{x}_3)</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>-1</td>
<td>1</td>
<td>-1</td>
<td>(\bar{x}_2 x_3)</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>6</td>
<td>0</td>
<td>0</td>
<td>(x_2 \bar{x}_3)</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>(x_2 x_3)</td>
</tr>
</tbody>
</table>

Hence,

\[ x_1^* \Delta(x_2, x_3) = -\bar{x}_2 x_3 = x_2 x_3 - x_3 \]

Substitute in \(f_1(x_2, x_3)\) gives

\[ f_1(x_2, x_3) = 3x_2 - 3x_3 + 8x_2x_3 = x_2(3 + 8x_3) - 3x_3 \].
Continuing with

\[ f_1(x_2, x_3) = x_2(3 + 8x_3) - 3x_3. \]

Then

<table>
<thead>
<tr>
<th>( x_3 )</th>
<th>( \Delta )</th>
<th>( x_2^* )</th>
<th>( x_2^* \Delta )</th>
<th>term</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>( x_3 )</td>
</tr>
<tr>
<td>1</td>
<td>11</td>
<td>0</td>
<td>0</td>
<td>( x_3 )</td>
</tr>
</tbody>
</table>

Hence, \( f_2(x_3) = -3x_3 \). This is minimized when \( x_3^* = 1 \), and the minimum is \(-3\). Substituting back gives \( x_2^* = 0 \), and \( x_1^* = 1 \).
Elimination on thin graphs

**Basic Rule:** When you eliminate a node, all the adjoining edges must be connected together (in a clique).
Elimination on thin graphs

**Basic Rule:** When you eliminate a node, all the adjoining edges must be connected together (in a clique).
Elimination on thin graphs

Basic Rule: When you eliminate a node, all the adjoining edges must be connected together (in a clique).
Elimination on thin graphs

**Basic Rule:** When you eliminate a node, all the adjoining edges must be connected together (in a clique).
Elimination on thin graphs

**Basic Rule:** When you eliminate a node, all the adjoining edges must be connected together (in a clique).
Elimination on thin graphs

**Basic Rule:** When you eliminate a node, all the adjoining edges must be connected together (in a clique).

- Include all the added edges in the original graph ...
- Resulting graph is known as a “chordal graph”.
- Every cycle of length at least 3 has a chord.
- Necessary and sufficient condition for elimination to work without addition of extra edges.
Elimination will also work on thin trees

Work from the leaves of the tree towards the centre.
3-wide trees
Multi-label Optimization
Common edge functions
Potts Model

Quadratic cost
Alpha-Expansion

Ishikawa’s Construction
Ishikawa’s Method – convex functions
$\mathcal{L} = \{0, 1\}$
\[ \mathcal{L} = \{0, 1, 2, 3\} \]

\[ x_i = ? \quad x_j = 2 \]
\[ \mathcal{L} = \{0, 1, 2, 3\} \]

\[ + E_{ij}(2, 0) \]
\[ + E_{ij}(1, 1) \]
\[ - E_{ij}(1, 0) \]
\[ - E_{ij}(2, 1) \]
Label-swap algorithms
Alpha Expansion
In each $a$-expansion a given label “$a$” grabs space from other labels.

For each move we choose expansion that gives the largest decrease in the energy: binary optimization problem.
a-expansion algorithm

1. Start with any initial solution
2. For each label “a” in any (e.g. random) order
   1. Compute optimal a-expansion move (s-t graph cuts)
   2. Decline the move if there is no energy decrease
3. Stop when no expansion move would decrease energy
When is alpha expansion possible

1. Each variable has the option to switch to a given value $\alpha$.
2. Problem is encoded as a binary optimization problem.
3. $u_i = 0$ means retain present value.
4. $u_j = 1$ means, switch to $\alpha$.

<table>
<thead>
<tr>
<th>$u_i$</th>
<th>$u_j$</th>
<th>binary cost</th>
<th>multilabel cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>$B^\alpha(0,0)$</td>
<td>$E(x_i,x_j)$</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>$B^\alpha(0,1)$</td>
<td>$E(x_i,\alpha)$</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>$B^\alpha(1,0)$</td>
<td>$E(\alpha,x_j)$</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>$B^\alpha(1,1)$</td>
<td>$E(\alpha,\alpha)$</td>
</tr>
</tbody>
</table>

Submodularity condition:

$$B^\alpha(0,1) + B^\alpha(1,0) \geq B^\alpha(0,0) + B^\alpha(1,1)$$

Corresponding condition on multilabel functions:

$$E(x_i,\alpha) + E(\alpha,x_j) \geq E(x_i,x_j) + E(\alpha,\alpha)$$

The triangle inequality
\( a \)-expansion for energies with \textit{metric} interactions

- \( a \)-expansion algorithm applies to pair-wise interactions that are \textit{metrics} on the space of labels (BVZ, PAMI’01)

\[
V(a, a) = 0
\]
\[
V(a, b) \geq 0
\]
\[
V(a, b) \leq V(a, c) + V(c, b)
\]

- Example: any truncated metric is also a metric

- \( \implies \text{Metric} \) case includes many \textit{robust} interactions
$a$-expansions: examples of *metric* interactions

"noisy shaded diamond"

Truncated "linear" $V$

$V(\alpha, \beta)$

Potts $V$ $\alpha - \beta$
Convex alpha expansion

1. Each variable has the option to switch to a given value $\alpha$.
2. Problem is encoded as a binary optimization problem.
3. $u_i = 0$ means take lesser of current value $x_i$ or $\alpha$.
4. $u_j = 1$ means take greater of current value $x_i$ or $\alpha$.
5. Suppose $x_i < \alpha < x_j$.

<table>
<thead>
<tr>
<th>$u_i$</th>
<th>$u_j$</th>
<th>binary cost</th>
<th>multilabel cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>$B^\alpha(0,0)$</td>
<td>$E(x_i, \alpha)$</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>$B^\alpha(0,1)$</td>
<td>$E(x_i, x_j)$</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>$B^\alpha(1,0)$</td>
<td>$E(\alpha, \alpha)$</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>$B^\alpha(1,1)$</td>
<td>$E(\alpha, x_j)$</td>
</tr>
</tbody>
</table>

Submodularity condition:
\[ B^\alpha(0,1) + B^\alpha(1,0) \geq B^\alpha(0,0) + B^\alpha(1,1) \]

Corresponding condition on multilabel functions:
\[ E(\alpha, \alpha) + E(x_i, x_j) \geq E(x_i, \alpha) + E(\alpha, x_j) \]

- The inverse triangle inequality - holds for convex functions.
- Better to take two small steps than one large step.
- Alpha-Expansion can handle convex functions.

- Multilabel Swap makes Ishikawa’s construction feasible for real problems.
  - Veksler (EMMCVPR 2009)
  - Carr / Hartley (DICTA 2009)
  - Stephen Gould (CVPR 2009)
Alpha-beta Swap
Experimental Results

Original

Pairwise

Swap (4.7 sec)

Ground Truth

Expansion (3.7 sec)

Higher Order

Swap (5.0 sec)

Expansion (4.4 sec)
Multi-label Swap
- Alpha-Expansion can handle convex functions.
- Multilabel Swap makes Ishikawa’s construction feasible for real problems.
  - Veksler (EMMCVPR 2009)
  - Carr / Hartley (DICTA 2009)
  - Stephen Gould (CVPR 2009)
- Elimination algorithms present feasible approach to non-submodular problems.
- FastPD (Komodakis, Paragios, Tziritas)
  - Addresses arbitrary multilabel problems.
Break