Graph Cut - II

Anand Mishra

Center for Visual Information Technology
IIIT Hyderabad
http://researchweb.iiit.ac.in/∼anand.mishra/

May 2012

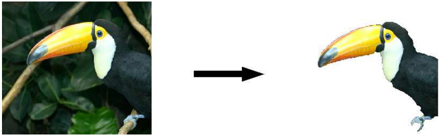
Outline

- Motivation
- Graph construction
- Implementation issues
- Sub modular functions
- Multi label problems and graph cut

Motivation

Let us consider the problem of image segmentation in energy minimization framework. The energy we need to minimize is of following form:

$$E(x) = \sum_i c_i x_i + \sum_{i,j} c_{i,j} x_i (1 - x_j)$$

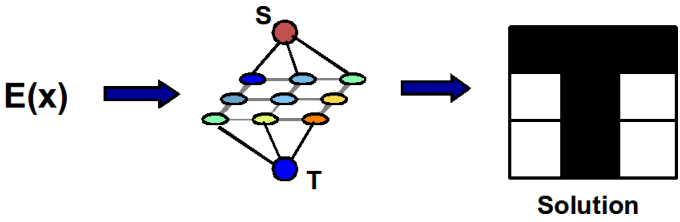


or we wish to find out the global minima of the above energy i.e. $x^* = \ argmin_x E(x)$

Graph construction

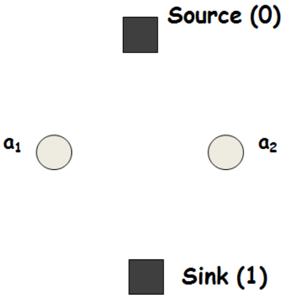
Construct a graph such that

- Any cut corresponds to an assignment of x
- The cost of the cut is equal to energy of x: E(x)



Let us understand how a graph is constructed for a given energy: (for simplicity let us assume there are two pixels a_1 and a_2

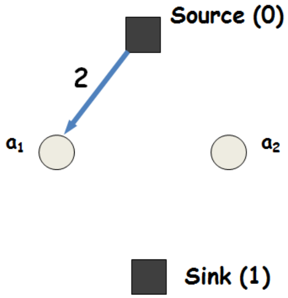
$E(a_1, a_2) =$



Graph construction

Let us understand how a graph is constructed for a given energy: (for simplicity let us assume there are two pixels a_1 and a_2

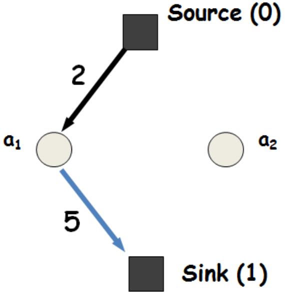
$E(a_1, a_2) = 2a_1$



Graph construction

Let us understand how a graph is constructed for a given energy: (for simplicity let us assume there are two pixels a_1 and a_2

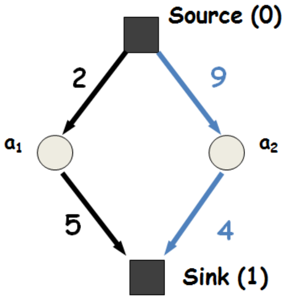
$E(a_1, a_2) = 2a_1 + 5\bar{a}_1$



Graph construction

Let us understand how a graph is constructed for a given energy: (for simplicity let us assume there are two pixels a_1 and a_2

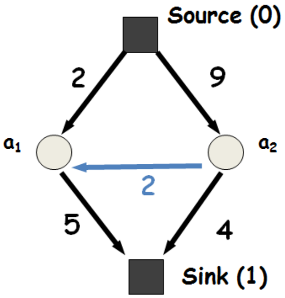
$E(a_1, a_2) = 2a_1 + 5\bar{a}_1 + 9a_2 + 4\bar{a}_2$



Graph construction

Let us understand how a graph is constructed for a given energy: (for simplicity let us assume there are two pixels a_1 and a_2

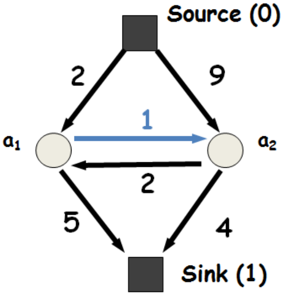
$E(a_1, a_2) = 2a_1 + 5\bar{a}_1 + 9a_2 + 4\bar{a}_2 + 2a_1\bar{a}_2$



Graph construction

Let us understand how a graph is constructed for a given energy: (for simplicity let us assume there are two pixels a_1 and a_2

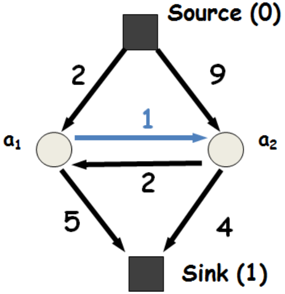
$E(a_1, a_2) = 2a_1 + 5\bar{a}_1 + 9a_2 + 4\bar{a}_2 + 2a_1\bar{a}_2 + \bar{a}_1 a_2$



Graph construction

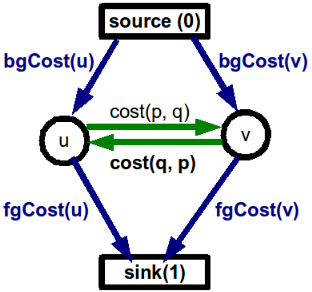
Let us understand how a graph is constructed for a given energy: (for simplicity let us assume there are two pixels a_1 and a_2

$$E(a_1, a_2) = 2a_1 + 5\bar{a}_1 + 9a_2 + 4\bar{a}_2 + 2a_1\bar{a}_2 + \bar{a}_1 a_2$$



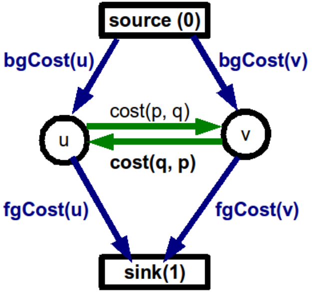
Implementation issues

```
graph *g

for all pixels p
        /* Add a node */
   node p = g->add_node()
   g -> add_tweights(p, fgCost(p), bgCost(p))
end

for all adjacent pixels p, q
    add_edge(p, cost(p,q), cost(q,p))
end

flow = g -> maxflow

for all pixels p
   Print  g -> what_segment(p)
end;
```



Implementation issues

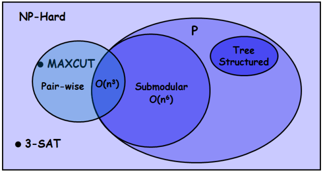
```
graph *g

for all pixels p
        /* Add a node */
   node p = g->add_node()
   g -> add_tweights(p, fgCost(p), bgCost(p))
end

for all adjacent pixels p, q
    add_edge(p, cost(p,q), cost(q,p))
end

flow = g -> maxflow

for all pixels p
   Print  g -> what_segment(p)
end;
```



Code on web:

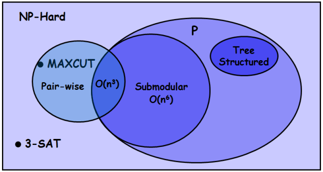
http://www.cs.ucl.ac.uk/staff/V.Kolmogorov/software.html

What energy functions can be minimized?



- General energy functions: NP hard to minimize, only approximate solutions available
- Easy energy functions (**sub-modular functions**) are graph representable and solvable in polynomial time

What energy functions can be minimized?



- General energy functions: NP hard to minimize, only approximate solutions available
- Easy energy functions (**sub-modular functions**) are graph representable and solvable in polynomial time

What is a sub-modular function?

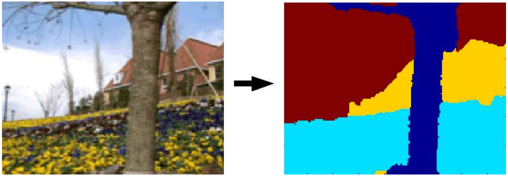
What is a sub-modular function

Let f be a function defined over set of boolean variables $x = \{x_1, x_2, ..., x_n\}$, then

1. All functions of one boolean variables are sub modular.
2. A function f of two boolean variable is sub-modular if
$$f(0,0) + f(1,1) \leq f(0,1) + f(1,0)$$
3. In general a function is sub-modular if all its projection to two variables are sub-modular.

Graph cuts: multi label cases

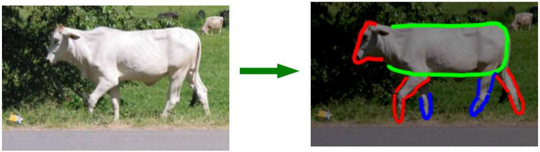
You might be wondering I am till now only talking about bi-label problems where a pixel can take either source (foreground) or sink(background). But in practice there are many vision problems which can be posed in a multi labelling framework.
Example 1: Find the labels for tree, sky, house and ground.



Graph cuts: multi label cases

You might be wondering I am till now only talking about bi-label problems where a pixel can take either source (foreground) or sink(background). But in practice there are many vision problems which can be posed in a multi labelling framework.

Example 2: I am interested in finding parts of an object.



Graph cuts: multi label cases

More examples:

- Stereo correspondence
- Image de-noising
- Image Inpainting

Graph cuts: multi label cases

Question: Can we solve such problem in graph cut framework?

Graph cuts: multi label cases

Question: Can we solve such problem in graph cut framework?
Answer: Yes :). We can solve it efficiently.

Back to energy function

Same as bi-label cases, the goal is to find a labelling that assigns each pixel $p \in \mathcal{P}$ a label $f_p \in \mathcal{L}$ where f is both consistent with observed data and piecewise smooth. i.e. we wish to find the global minima of energy function of following form:

$$E(f) = E_{data}(f) + E_{smooth}(f)$$

The form of $E_{data}(f)$ is typically, $E_{data}(f) = \sum_{p \in \mathcal{P}} D_p(f_p)$ Where D_p measures the agreement of inferred label from the observed data. In image restoration, for example, $D_p(f_p)$ is $(f_p - i_p)^2$ where i_p is the observed intensity at pixel p.

Back to energy function

$$E(f) = E_{data}(f) + E_{smooth}(f)$$

$E_{smooth}(f)$ has following form typically,

$$E_{smooth}(f) = \sum_{(p,q) \in \mathcal{N}} V_{pq}(f_p, f_q)$$

The smoothness term E_{smooth} is used to impose spatial smoothness. It should have **discontinuity preserving property**.

Question: Can we minimize this form of energy always using graph cut?

Back to energy function

- We have seen for $|\mathcal{L}|$ =2, this energy is exactly minimization.
- We can also proof that we can find global minima of such energy in case when $|\mathcal{L}|$ =finite but $V_{pq}(f_p, f_q) = |f_p - f_q|$.
- Unfortunately, such V_{pq} is not discontinuity preserving and thus can not be applied for many vision application.
- In general, minimizing such energy function is an NP hard problem. Although we can get approximate solution with known factor of global minima in case when V_{pq} is either a metric or semi-metric.

Semi-Metric and Metric on space of labels

A function $V(\cdot, \cdot)$ is called a semi-metric on the space of labels $\alpha, \beta \in \mathcal{L}$ if it satisfies following two properties:

1. $V(\alpha, \beta) = V(\beta, \alpha)$
2. $V(\alpha, \beta) = 0 \iff \alpha = \beta$

If $V(\cdot, \cdot)$ also satisfies triangle inequality i.e.

$$V(\alpha, \beta) = V(\alpha, \gamma) + V(\gamma, \beta)$$

for $\alpha, \beta, \gamma \in \mathcal{L}$ then it is a metric.
Example: function $V_{pq} = min(K, |f_P - f_q|^2)$ is a semi-metric whereas Potts model $V = \delta(f_p - f_q)$ is a metric.

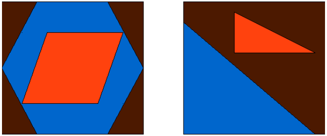
Move algorithm for approximate solution

Now we will describe two move algorithms which find out the approximate solution for the energy function.

- First move algorithm is known as $\alpha - \beta$ **swap** which works when V is a semi-metric.
- Second move algorithm we call as α **expansion**, it works only when V is metric.

Move algorithm for approximate solution

Before we formally define these moves let us first understand how labelling is associated with partitioning of pixels.
Any labelling f can be uniquely represented by a partition of image pixels $P = \{\mathcal{P}_l : l \in \mathcal{L}\}$ where $\mathcal{P}_l$ is subset of pixels assigned label l.



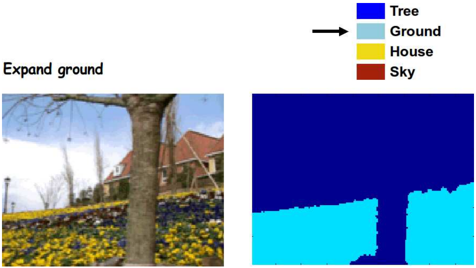
Here two different labelling (or equivalently partitioning) is shown.

$\alpha - \beta$ **swap:** Given a pair of labels α, β a move from partition P to partition P' is called an $\alpha - \beta$ swap if $\mathcal{P}_l = \mathcal{P}'_l$ for any label $l \neq \alpha, \beta$

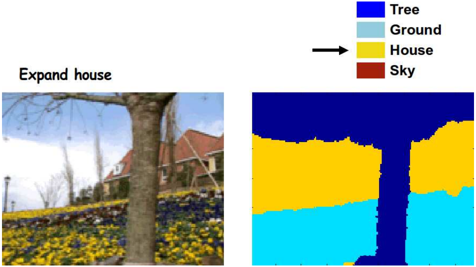
$\alpha - \beta$ **swap:** Given a pair of labels α, β a move from partition P to partition P' is called an $\alpha - \beta$ swap if $\mathcal{P}_l = \mathcal{P}'_l$ for any label $l \neq \alpha, \beta$

α **expansion:** Given a label α a move from partition P to partition P' is called an α expansion if $\mathcal{P}_\alpha \subset \mathcal{P'}_\alpha$ and $\mathcal{P'}_l \subset \mathcal{P}_l$ for any label $l \neq \alpha$.

α **expansion:** Given a label α a move from partition P to partition P' is called an α expansion if $\mathcal{P}_\alpha \subset \mathcal{P}'_\alpha$ and $\mathcal{P}'_l \subset \mathcal{P}_l$ for any label $l \neq \alpha$.

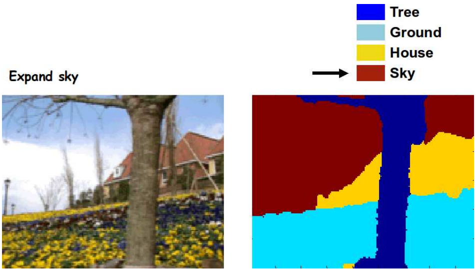


α **expansion:** Given a label α a move from partition P to partition P' is called an α expansion if $\mathcal{P}_\alpha \subset \mathcal{P}'_\alpha$ and $\mathcal{P}'_l \subset \mathcal{P}_l$ for any label $l \neq \alpha$.



α expansion

α **expansion:** Given a label α a move from partition P to partition P' is called an α expansion if $\mathcal{P}_\alpha \subset \mathcal{P}'_\alpha$ and $\mathcal{P}'_l \subset \mathcal{P}_l$ for any label $l \neq \alpha$.



$\alpha - \beta$ swap: basic algorithm

1. Start with an arbitrary labelling f
2. Set success $\leftarrow 0$
3. for each pair of label $\{\alpha, \beta\} \in \mathcal{L}$
    1. Find $\hat{f} = \underset{f'}{\operatorname{argmin}} E(f')$ within one $\alpha - \beta$ swap of f
    2. If $E(\hat{f}) < E(f)$ then
       Set $f \leftarrow \hat{f}$
       Set success $\leftarrow 1$
4. if success == 1 then goto step 2 else return f.

α expansion: basic algorithm

1. Start with an arbitrary labelling f
2. Set success $\leftarrow 0$
3. for every label $\alpha \in \mathcal{L}$
   1. Find $\hat{f} = \underset{f'}{\operatorname{argmin}} E(f')$ within one α expansion of f
   2. If $E(\hat{f}) < E(f)$ then
      Set $f \leftarrow \hat{f}$
      Set success $\leftarrow 1$
4. if success == 1 then goto step 2 else return f.

Important properties of swap and expansion algorithm

1. A cycle in a swap move algorithm takes $O(|\mathcal{L}|^2)$ iterations, and a cycle in expansion move takes $O(|\mathcal{L}|)$ iterations.
2. The algorithms are guaranteed to terminate in finite number of cycles.
3. Expansion move produces a labeling f such that $E(f^*) \leq E(f) \leq 2k.E(f^*)$ where f^* is the global minimum and constant $k = \frac{max\{V(\alpha,\beta):\alpha\neq\beta\}}{min\{V(\alpha,\beta):\alpha\neq\beta\}}$

$\alpha - \beta$ swap: graph cuts

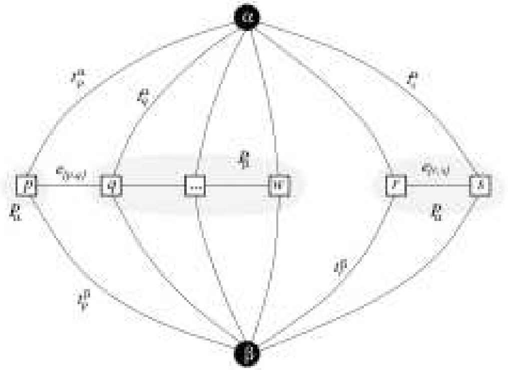
Goal: Given an input labelling f(partition P) and a pair of labels α, β, we wish to find a labelling $\hat{f}$ that minimizes E over all labelling within one $\alpha - \beta$ swap of f.

- This swap move can be performed using graph cut. Let us understand it by an example.
- Let us suppose there are two partitions of pixels one with labels $\alpha(\mathcal{P}_\alpha)$ and the other one with labels $\beta(\mathcal{P}_\beta)$. Suppose

$$\mathcal{P}_\alpha = \{p, q, r\}$$
$$\mathcal{P}_\beta = \{s, t, ..., w\}$$

$\alpha - \beta$ swap: graph cuts

The graph is constructed to find out the $\alpha - \beta$ swap. The source and target of the graph are α and β respectively. The cut of the graph determines which pixels will change their labels and which others will retain it.



In the similar way α expansion is performed using graph cut. For this a graph is constructed with source and target as α and non-α respectively, and graph cut is performed for each label in $\mathcal{L}$

References

- Vladimir Kolmogorov, Ramin Zabih: What Energy Functions Can Be Minimized via Graph Cuts? PAMI 2004
- Yuri Boyhood, Olga Veksler, Ramin Zabih: Fast Approximate Energy Minimization via Graph Cut. PAMI 2001