Visual Recognition: Inference in Graphical Models

Raquel Urtasun

TTI Chicago

Feb 28, 2012

Graphical models

- Applications
- Representation
- Inference
  - message passing (LP relaxations)
  - graph cuts
- Learning

Inference with graph cuts

Submodular Functions

- A Pseudo-boolean function $f : \{0,1\}^n \rightarrow \Re$ is submodular if

$$f(A) + f(B) \geq \underbrace{f(A \vee B)}_{OR} + \underbrace{f(A \wedge B)}_{AND} \qquad \forall A, B \in \{0,1\}^n$$

- Example: $n = 2,\ A = [1,0],\ B = [0,1]$

$$f([1,0]) + f([0,1]) \geq f([1,1]) + f([0,0])$$

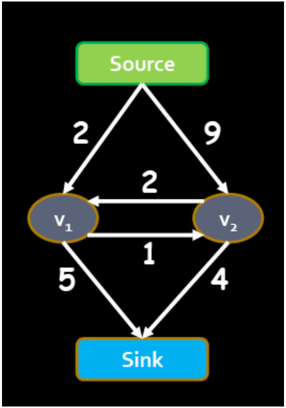
- Sum of submodular functions is submodular $\rightarrow$ Easy to proof.
- Some energies in computer vision can be submodular

Minimizing submodular Functions

- Pairwise submodular functions can be transformed to st-mincut/max-flow [Hammer, 65].
- Very low running time $\sim \mathcal{O}(n)$

The ST-mincut problem

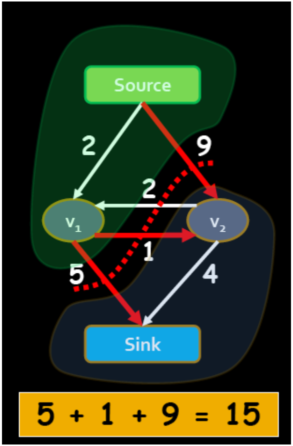
- Suppose we have a graph $G = \{V, E, C\}$, with vertices V, Edges E and costs C.



[Source: P. Kohli]

The ST-mincut problem

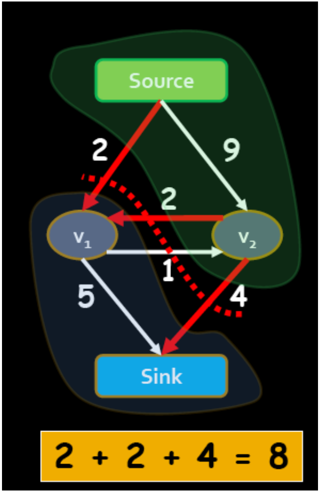
- An st-cut (S,T) divides the nodes between source and sink.
- The cost of a st-cut is the sum of cost of all edges going from S to T



[Source: P. Kohli]

The ST-mincut problem

- The st-mincut is the st-cut with the minimum cost

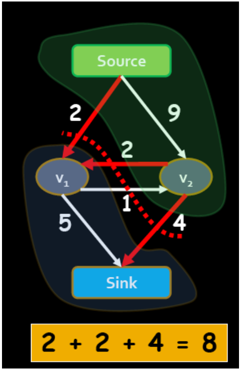


[Source: P. Kohli]

Back to our energy minimization

Construct a graph such that

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



[Source: P. Kohli]

St-mincut and Energy Minimization

$$E(x) = \sum_i \theta_i(x_i) + \sum_{i,j} \theta_{ij}(x_i, x_j)$$

For all ij $\theta_{ij}(0,1) + \theta_{ij}(1,0) \geq \theta_{ij}(0,0) + \theta_{ij}(1,1)$

Equivalent (transformable)

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

$c_{ij} \geq 0$

[Source: P. Kohli]

How are they equivalent?

$A = \theta_{ij}(0,0)$ $B = \theta_{ij}(0,1)$ $C = \theta_{ij}(1,0)$ $D = \theta_{ij}(1,1)$

$x_i \backslash x_j$	0	1
0	A	B
1	C	D

= A +

	0	1
0	0	0
1	C-A	C-A

+

	0	1
0	0	D-C
1	0	D-C

+

	0	1
0	0	B +C- A-D
1	0	0

if $x_1 = 1$ add C-A

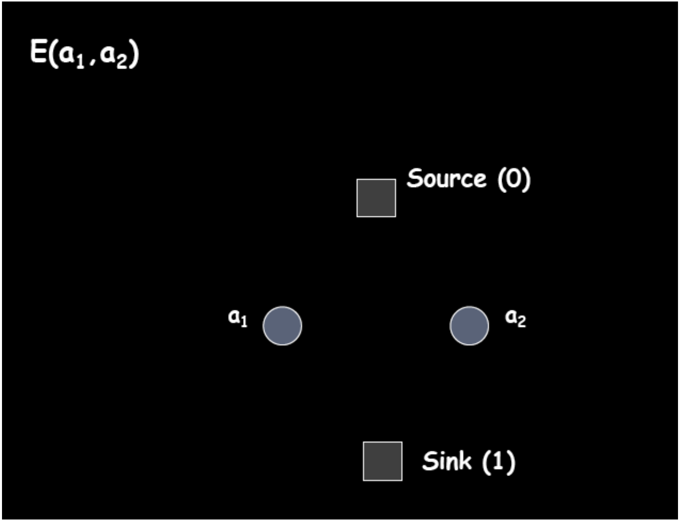
if $x_2 = 1$ add D-C

$$\theta_{ij}(x_i,x_j) = \theta_{ij}(0,0) + (\theta_{ij}(1,0)-\theta_{ij}(0,0))\, x_i + (\theta_{ij}(1,0)-\theta_{ij}(0,0))\, x_j + (\theta_{ij}(1,0) + \theta_{ij}(0,1) - \theta_{ij}(0,0) - \theta_{ij}(1,1))\,(1-x_i)\, x_j$$

$B+C-A-D \geq 0$ is true from the submodularity of θ_{ij}

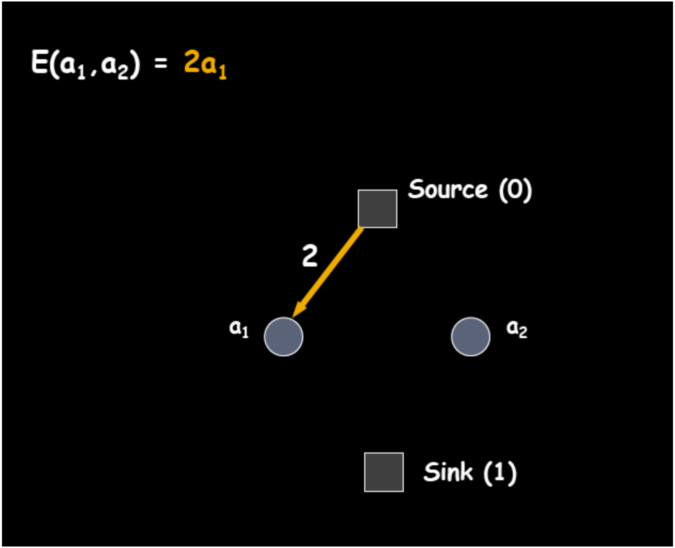
[Source: P. Kohli]

Graph Construction



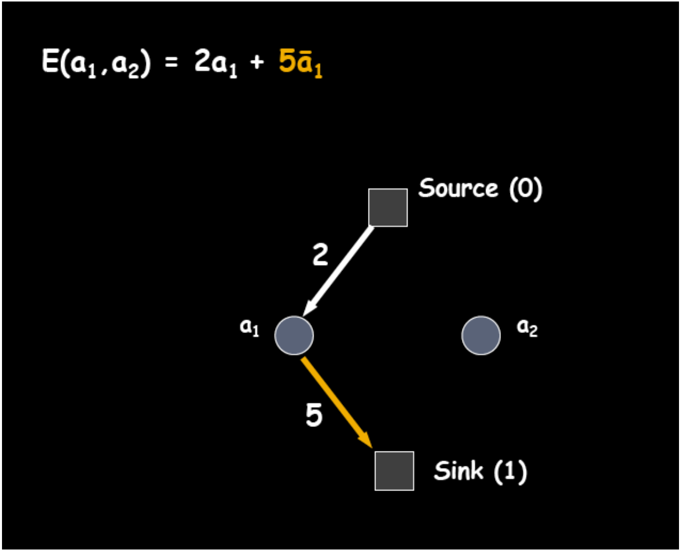
[Source: P. Kohli]

Graph Construction



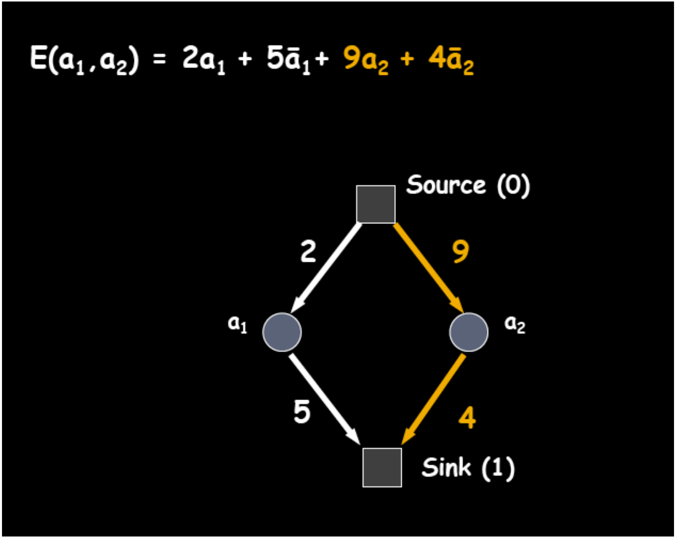
[Source: P. Kohli]

Graph Construction



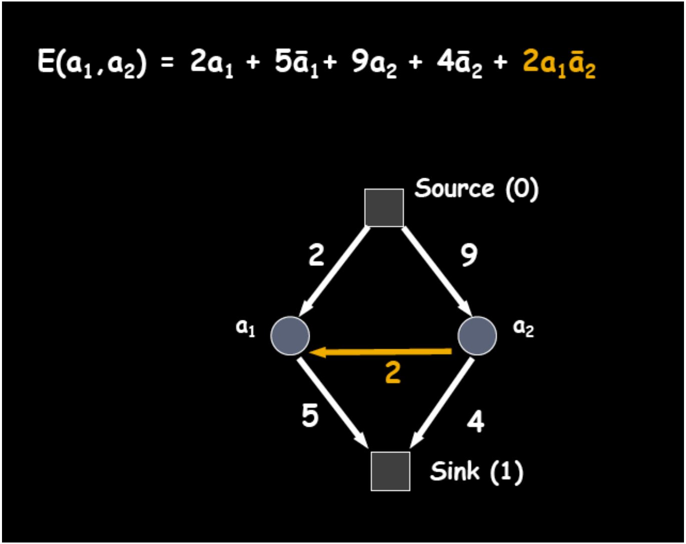
[Source: P. Kohli]

Graph Construction



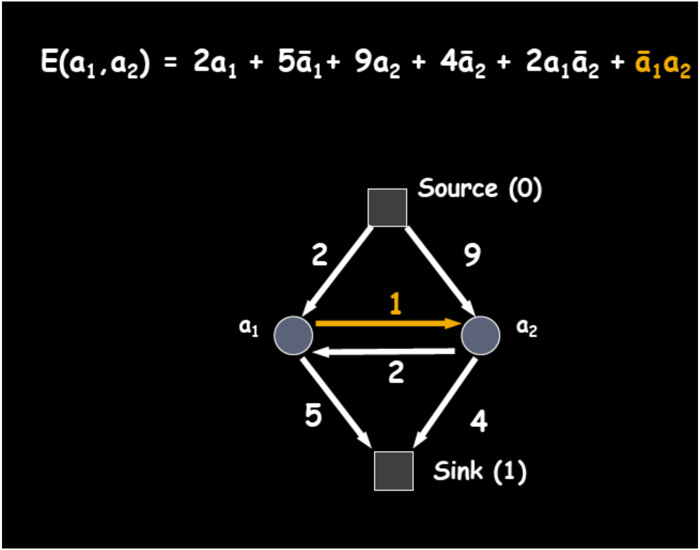
[Source: P. Kohli]

Graph Construction



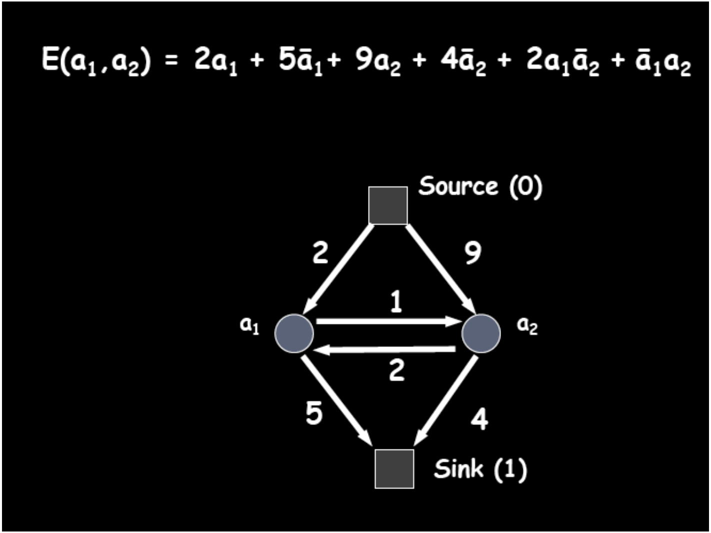
[Source: P. Kohli]

Graph Construction



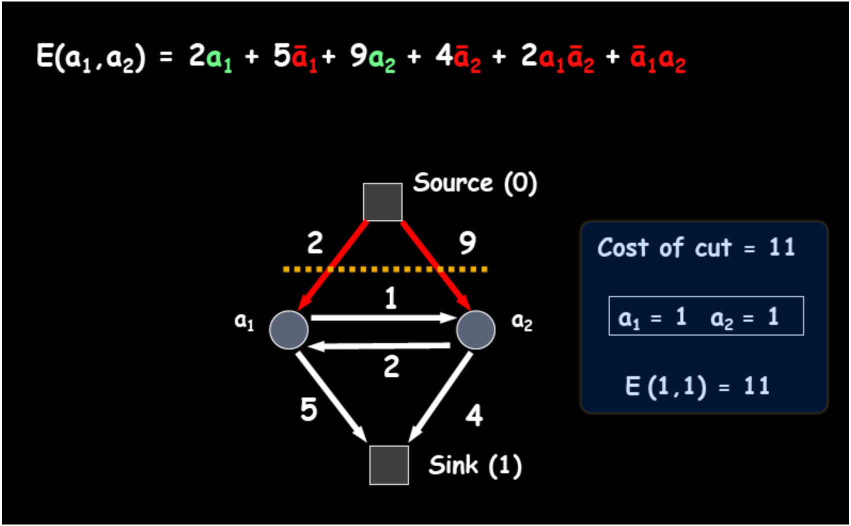
[Source: P. Kohli]

Graph Construction



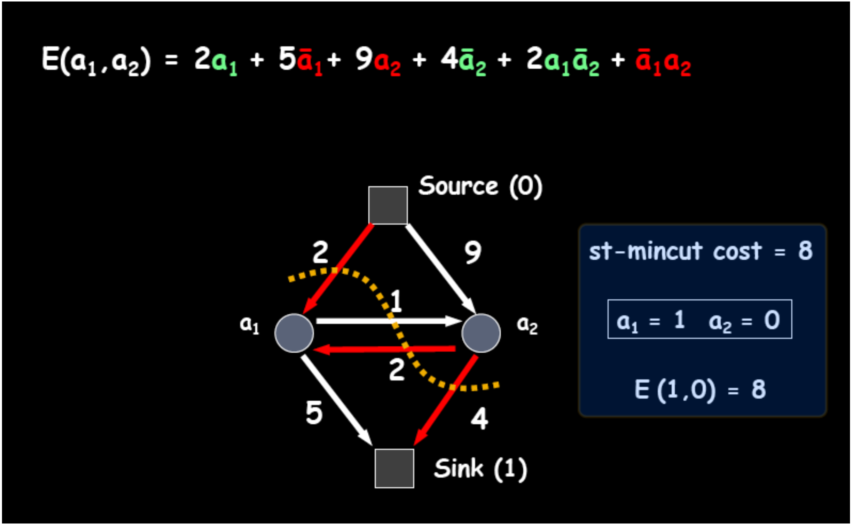
[Source: P. Kohli]

Graph Construction



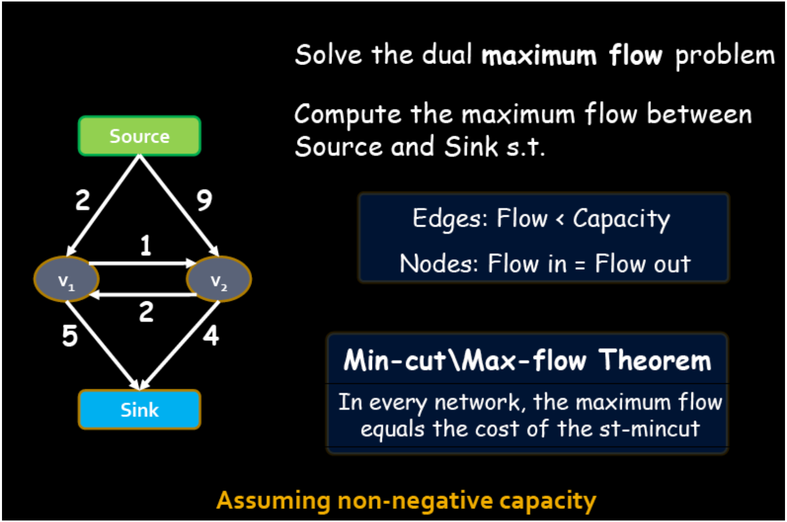
[Source: P. Kohli]

Graph Construction



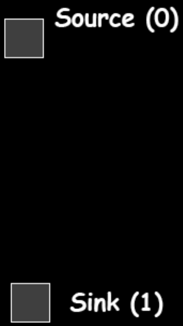
[Source: P. Kohli]

How to compute the St-mincut?



[Source: P. Kohli]

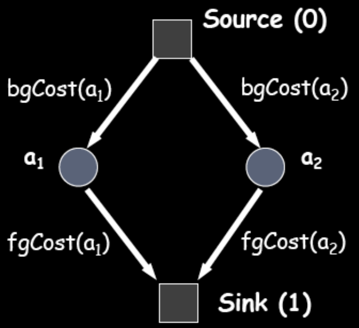
```
Graph *g;

For all pixels p

    /* Add a node to the graph */
    nodeID(p) = g->add_node();

    /* Set cost of terminal edges */
    set_weights(nodeID(p), fgCost(p), bgCost(p));

end

for all adjacent pixels p,q
    add_weights(nodeID(p), nodeID(q), cost(p,q));
end

g->compute_maxflow();

label_p = g->is_connected_to_source(nodeID(p));
// is the label of pixel p (0 or 1)
```



[Source: P. Kohli]

How does the code look like

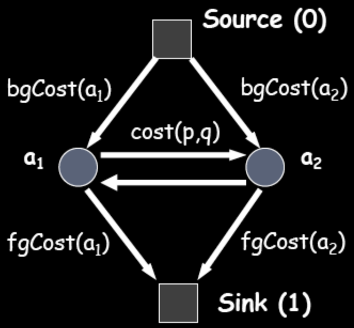
```
Graph *g;

For all pixels p

        /* Add a node to the graph */
        nodeID(p) = g->add_node();

        /* Set cost of terminal edges */
        set_weights(nodeID(p), fgCost(p), bgCost(p));

end

for all adjacent pixels p,q
        add_weights(nodeID(p), nodeID(q), cost(p,q));
end

g->compute_maxflow();

label_p = g->is_connected_to_source(nodeID(p));
// is the label of pixel p (0 or 1)
```



[Source: P. Kohli]

How does the code look like

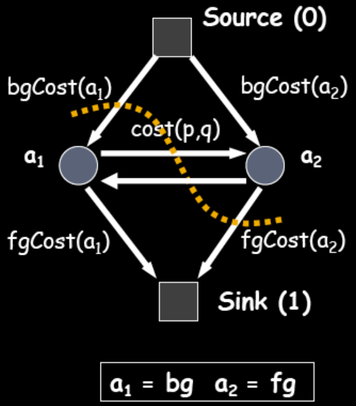
```
Graph *g;

For all pixels p

        /* Add a node to the graph */
        nodeID(p) = g->add_node();

        /* Set cost of terminal edges */
        set_weights(nodeID(p), fgCost(p), bgCost(p));

end

for all adjacent pixels p,q
        add_weights(nodeID(p), nodeID(q),  cost(p,q));
end

g->compute_maxflow();

label_p = g->is_connected_to_source(nodeID(p));
// is the label of pixel p (0 or 1)
```



[Source: P. Kohli]

How does the code look like

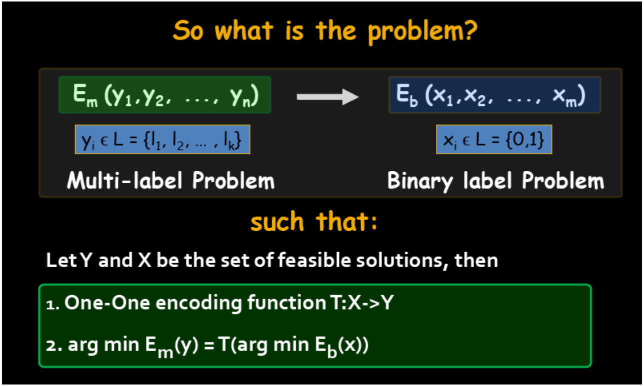
```
Graph *g;

For all pixels p

    /* Add a node to the graph */
    nodeID(p) = g->add_node();

    /* Set cost of terminal edges */
    set_weights(nodeID(p), fgCost(p), bgCost(p));

end

for all adjacent pixels p,q
    add_weights(nodeID(p), nodeID(q), cost(p,q));
end

g->compute_maxflow();

label_p = g->is_connected_to_source(nodeID(p));
// is the label of pixel p (0 or 1)
```



[Source: P. Kohli]

Graph cuts for multi-label problems

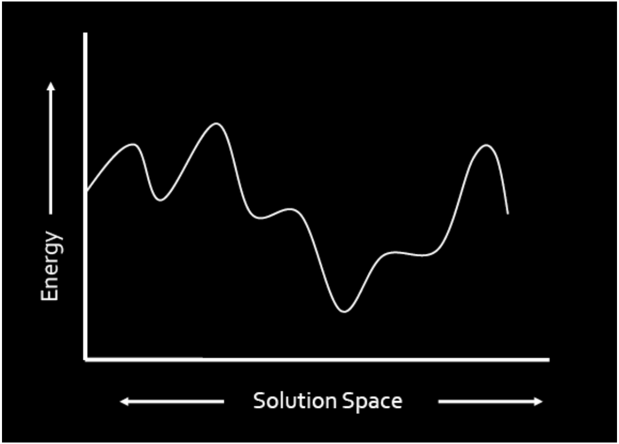
- Exact Transformation to QPBF [Roy and Cox 98] [Ishikawa 03] [Schlesinger et al. 06] [Ramalingam et al. 08]



- Very high computational cost

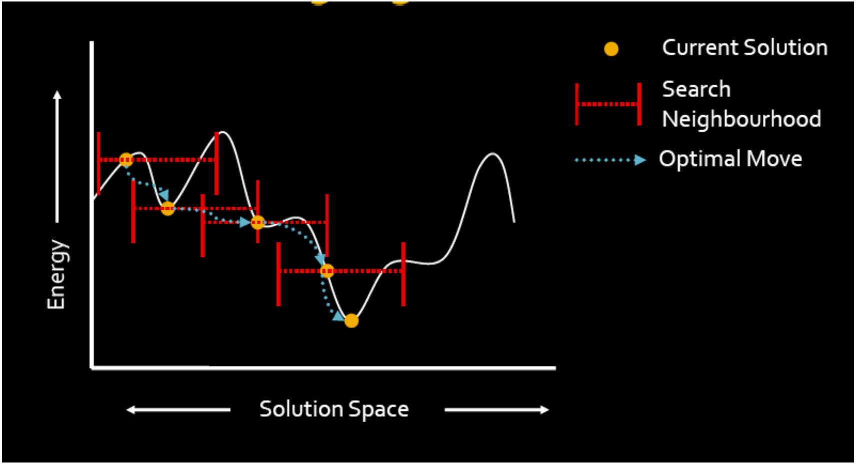
[Source: P. Kohli]

Alternative: Move making



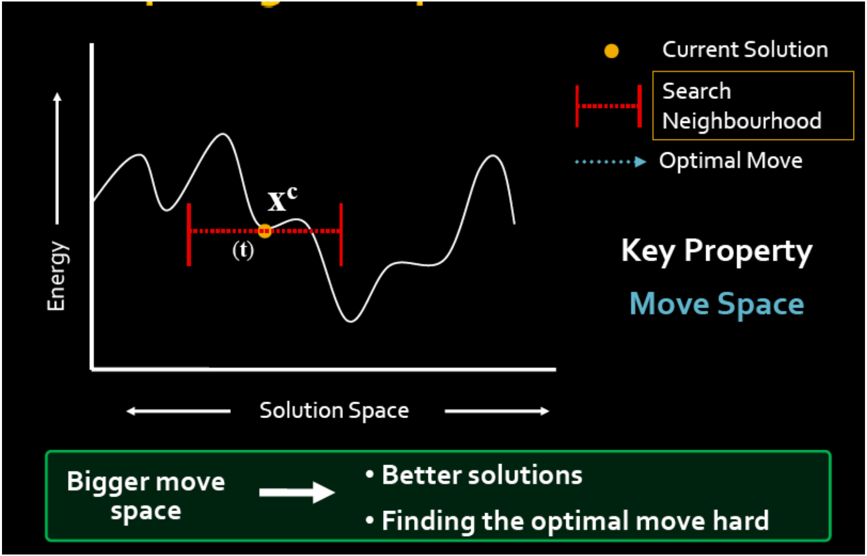
[Source: P. Kohli]

Alternative: Move making



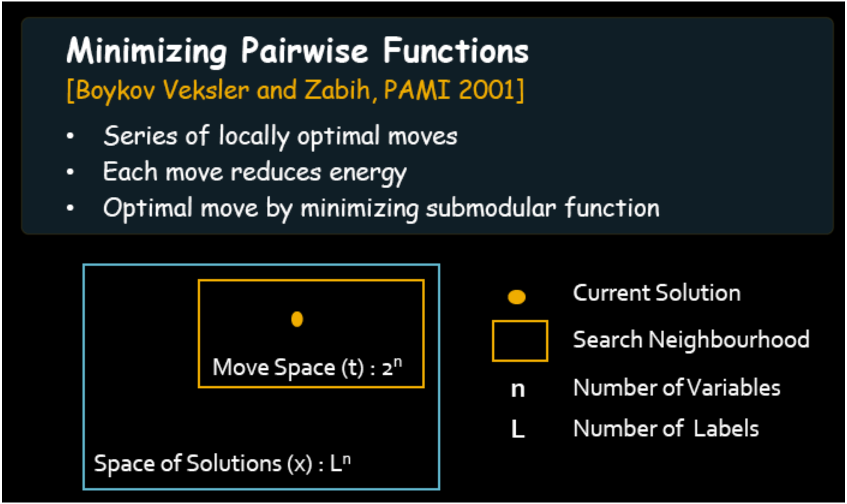
[Source: P. Kohli]

Computing the Optimal Move



[Source: P. Kohli]

Move Making Algorithms



[Source: P. Kohli]

Energy Minimization

- Consider pairwise MRFs

$$E(f) = \sum_{\{p,q\} \in \mathcal{N}} V_{p,q}(f_p, f_q) + \sum_p D_p(f_p)$$

with $\mathcal{N}$ defining the interactions between nodes, e.g., pixels

- D_p non-negative, but arbitrary.

Energy Minimization

- Consider pairwise MRFs

$$E(f) = \sum_{\{p,q\} \in \mathcal{N}} V_{p,q}(f_p, f_q) + \sum_p D_p(f_p)$$

with $\mathcal{N}$ defining the interactions between nodes, e.g., pixels

- D_p non-negative, but arbitrary.
- Same as before, where $V_{p,q} \equiv -\theta_\alpha$ and $D_p \equiv -\theta_p$.

Energy Minimization

- Consider pairwise MRFs

$$E(f) = \sum_{\{p,q\} \in \mathcal{N}} V_{p,q}(f_p, f_q) + \sum_p D_p(f_p)$$

with $\mathcal{N}$ defining the interactions between nodes, e.g., pixels

- D_p non-negative, but arbitrary.
- Same as before, where $V_{p,q} \equiv -\theta_\alpha$ and $D_p \equiv -\theta_p$.
- This is the graph-cuts notation.
- Important to notice it's the same thing.

Energy Minimization

- Consider pairwise MRFs

$$E(f) = \sum_{\{p,q\} \in \mathcal{N}} V_{p,q}(f_p, f_q) + \sum_p D_p(f_p)$$

with $\mathcal{N}$ defining the interactions between nodes, e.g., pixels

- D_p non-negative, but arbitrary.
- Same as before, where $V_{p,q} \equiv -\theta_\alpha$ and $D_p \equiv -\theta_p$.
- This is the graph-cuts notation.
- Important to notice it's the same thing.

Metric vs Semimetric

Two general classes of pairwise interactions

- **Metric** if it satisfies for any set of labels α, β, γ

$$\begin{aligned} V(\alpha,\beta) = 0 \quad &\leftrightarrow \quad \alpha = \beta \\ V(\alpha,\beta) \quad &= \quad V(\beta,\alpha) \geq 0 \\ V(\alpha,\beta) \quad &\leq \quad V(\alpha,\gamma) + V(\gamma,\beta) \end{aligned}$$

- **Semi-metric** if it satisfies for any set of labels α, β, γ

$$\begin{aligned} V(\alpha,\beta) = 0 \quad &\leftrightarrow \quad \alpha = \beta \\ V(\alpha,\beta) \quad &= \quad V(\beta,\alpha) \geq 0 \end{aligned}$$

Metric vs Semimetric

Two general classes of pairwise interactions

- **Metric** if it satisfies for any set of labels α, β, γ

$$\begin{aligned} V(\alpha,\beta) = 0 \quad &\leftrightarrow \quad \alpha = \beta \\ V(\alpha,\beta) \quad &= \quad V(\beta,\alpha) \geq 0 \\ V(\alpha,\beta) \quad &\leq \quad V(\alpha,\gamma) + V(\gamma,\beta) \end{aligned}$$

- **Semi-metric** if it satisfies for any set of labels α, β, γ

$$\begin{aligned} V(\alpha,\beta) = 0 \quad &\leftrightarrow \quad \alpha = \beta \\ V(\alpha,\beta) \quad &= \quad V(\beta,\alpha) \geq 0 \end{aligned}$$

Examples for 1D label set

- Truncated quadratic is a semi-metric

$$V(\alpha, \beta) = \min(K, |\alpha - \beta|^2)$$

with K a constant.

- Truncated absolute distance is a metric

$$V(\alpha, \beta) = \min(K, |\alpha - \beta|)$$

with K a constant.

Examples for 1D label set

- Truncated quadratic is a semi-metric

$$V(\alpha, \beta) = \min(K, |\alpha - \beta|^2)$$

  with K a constant.

- Truncated absolute distance is a metric

$$V(\alpha, \beta) = \min(K, |\alpha - \beta|)$$

  with K a constant.

- For multi-dimensional, replace $|\cdot|$ by any norm.

Examples for 1D label set

- Truncated quadratic is a semi-metric

$$V(\alpha, \beta) = \min(K, |\alpha - \beta|^2)$$

  with K a constant.

- Truncated absolute distance is a metric

$$V(\alpha, \beta) = \min(K, |\alpha - \beta|)$$

  with K a constant.

- For multi-dimensional, replace $|\cdot|$ by any norm.
- Potts model is a metric

$$V(\alpha, \beta) = K \cdot T(\alpha \neq \beta)$$

  with $T(\cdot) = 1$ if the argument is true and 0 otherwise.

Examples for 1D label set

- Truncated quadratic is a semi-metric

$$V(\alpha, \beta) = \min(K, |\alpha - \beta|^2)$$

  with K a constant.

- Truncated absolute distance is a metric

$$V(\alpha, \beta) = \min(K, |\alpha - \beta|)$$

  with K a constant.

- For multi-dimensional, replace $|\cdot|$ by any norm.
- Potts model is a metric

$$V(\alpha, \beta) = K \cdot T(\alpha \neq \beta)$$

  with $T(\cdot) = 1$ if the argument is true and 0 otherwise.

Binary Moves

- $\alpha - \beta$ moves works for semi-metrics
- α expansion works for V being a metric

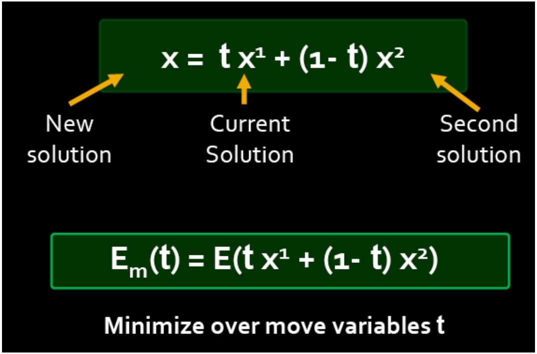


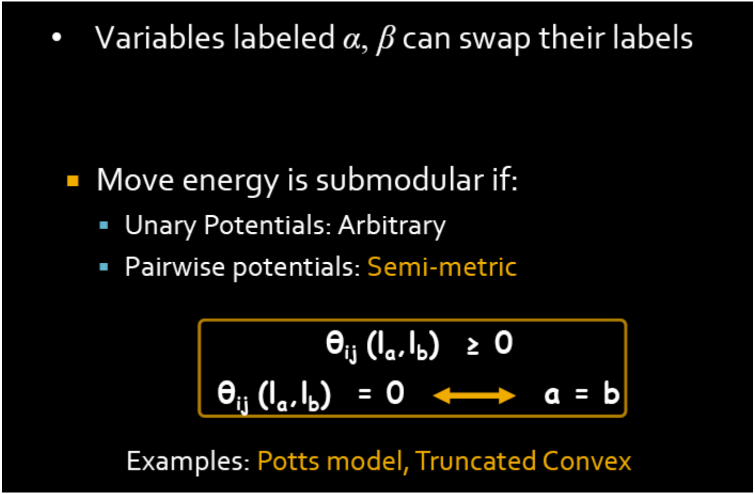
Figure: Figure from P. Kohli tutorial on graph-cuts

- For certain x^1 and x^2, the move energy is sub-modular QPBF

Swap Move

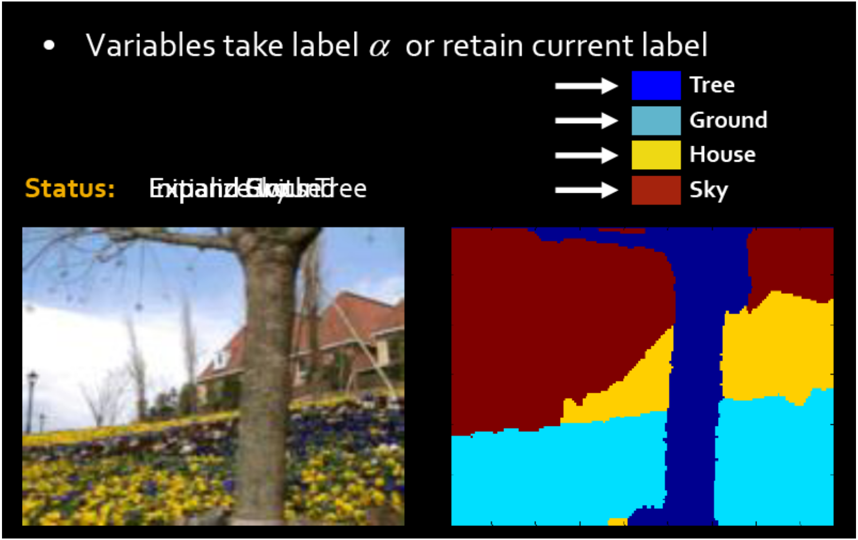
[Source: P. Kohli]

Swap Move



[Source: P. Kohli]

Expansion Move



[Source: P. Kohli]

Expansion Move

- Variables take label α or retain current label

- Move energy is submodular if:
  - Unary Potentials: Arbitrary
  - Pairwise potentials: Metric

Semi metric + Triangle Inequality

$$\theta_{ij}(l_a, l_b) + \theta_{ij}(l_b, l_c) \geq \theta_{ij}(l_a, l_c)$$

Examples: Potts model, Truncated linear

Cannot solve truncated quadratic

[Source: P. Kohli]

More formally

- Any labeling can be uniquely represented by a partition of image pixels $\mathbf{P} = \{\mathcal{P}_l | l \in \mathcal{L}\}$, where $\mathcal{P}_l = \{p \in \mathcal{P} | f_p = l\}$ is a subset of pixels assigned label l.
- There is a one to one correspondence between labelings f and partitions $\mathcal{P}$.

More formally

- Any labeling can be uniquely represented by a partition of image pixels $\mathbf{P} = \{\mathcal{P}_l | l \in \mathcal{L}\}$, where $\mathcal{P}_l = \{p \in \mathcal{P} | f_p = l\}$ is a subset of pixels assigned label l.
- There is a one to one correspondence between labelings f and partitions $\mathcal{P}$.
- Given a pair of labels α, β, a move from a partition $\mathcal{P}$ (labeling f) to a new partition $\mathcal{P}$' (labeling f') is called an $\alpha - \beta$ **swap** if $\mathcal{P}_l = \mathcal{P}'$ for any label $l \neq \alpha, \beta$.

More formally

- Any labeling can be uniquely represented by a partition of image pixels $\mathbf{P} = \{\mathcal{P}_l | l \in \mathcal{L}\}$, where $\mathcal{P}_l = \{p \in \mathcal{P} | f_p = l\}$ is a subset of pixels assigned label l.
- There is a one to one correspondence between labelings f and partitions $\mathcal{P}$.
- Given a pair of labels α, β, a move from a partition $\mathcal{P}$ (labeling f) to a new partition $\mathcal{P}$' (labeling f') is called an $\alpha - \beta$ **swap** if $\mathcal{P}_l = \mathcal{P}'$ for any label $l \neq \alpha, \beta$.
- The only difference between $\mathcal{P}$ and $\mathcal{P}'$ is that some pixels that were labeled in $\mathcal{P}$ are now labeled in $\mathcal{P}'$, and vice-versa.

More formally

- Any labeling can be uniquely represented by a partition of image pixels $\mathbf{P} = \{\mathcal{P}_l | l \in \mathcal{L}\}$, where $\mathcal{P}_l = \{p \in \mathcal{P} | f_p = l\}$ is a subset of pixels assigned label l.
- There is a one to one correspondence between labelings f and partitions $\mathcal{P}$.
- Given a pair of labels α, β, a move from a partition $\mathcal{P}$ (labeling f) to a new partition $\mathcal{P}$' (labeling f') is called an $\alpha - \beta$ **swap** if $\mathcal{P}_l = \mathcal{P}'$ for any label $l \neq \alpha, \beta$.
- The only difference between $\mathcal{P}$ and $\mathcal{P}'$ is that some pixels that were labeled in $\mathcal{P}$ are now labeled in $\mathcal{P}'$, and vice-versa.
- Given a label l, a move from a partition $\mathcal{P}$ (labeling f) to a new partition $\mathcal{P}'$ (labeling f') is called an α-**expansion** if $\mathcal{P}_\alpha \subset \mathcal{P}'_\alpha$ and $\mathcal{P}'_l \subset \mathcal{P}_l$.

More formally

- Any labeling can be uniquely represented by a partition of image pixels $\mathbf{P} = \{\mathcal{P}_l | l \in \mathcal{L}\}$, where $\mathcal{P}_l = \{p \in \mathcal{P} | f_p = l\}$ is a subset of pixels assigned label l.
- There is a one to one correspondence between labelings f and partitions $\mathcal{P}$.
- Given a pair of labels α, β, a move from a partition $\mathcal{P}$ (labeling f) to a new partition $\mathcal{P}$' (labeling f') is called an $\alpha - \beta$ **swap** if $\mathcal{P}_l = \mathcal{P}'$ for any label $l \neq \alpha, \beta$.
- The only difference between $\mathcal{P}$ and $\mathcal{P}'$ is that some pixels that were labeled in $\mathcal{P}$ are now labeled in $\mathcal{P}'$, and vice-versa.
- Given a label l, a move from a partition $\mathcal{P}$ (labeling f) to a new partition $\mathcal{P}'$ (labeling f') is called an α-**expansion** if $\mathcal{P}_\alpha \subset \mathcal{P}'_\alpha$ and $\mathcal{P}'_l \subset \mathcal{P}_l$.
- An α-**expansion** move allows any set of image pixels to change their labels to α.

More formally

- Any labeling can be uniquely represented by a partition of image pixels $\mathbf{P} = \{\mathcal{P}_l | l \in \mathcal{L}\}$, where $\mathcal{P}_l = \{p \in \mathcal{P} | f_p = l\}$ is a subset of pixels assigned label l.
- There is a one to one correspondence between labelings f and partitions $\mathcal{P}$.
- Given a pair of labels α, β, a move from a partition $\mathcal{P}$ (labeling f) to a new partition $\mathcal{P}$' (labeling f') is called an $\alpha - \beta$ **swap** if $\mathcal{P}_l = \mathcal{P}'$ for any label $l \neq \alpha, \beta$.
- The only difference between $\mathcal{P}$ and $\mathcal{P}'$ is that some pixels that were labeled in $\mathcal{P}$ are now labeled in $\mathcal{P}'$, and vice-versa.
- Given a label l, a move from a partition $\mathcal{P}$ (labeling f) to a new partition $\mathcal{P}'$ (labeling f') is called an α-**expansion** if $\mathcal{P}_\alpha \subset \mathcal{P}'_\alpha$ and $\mathcal{P}'_l \subset \mathcal{P}_l$.
- An α-**expansion** move allows any set of image pixels to change their labels to α.

Example

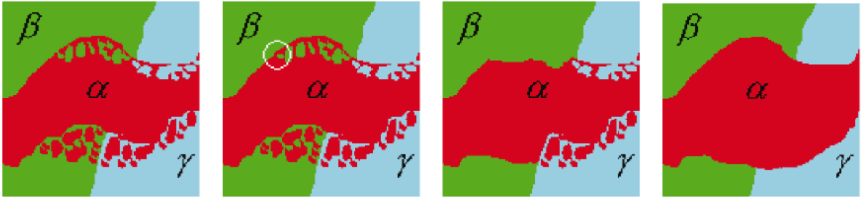


Figure: (a) Current partition (b) local move (c) $\alpha - \beta$-swap (d) α-expansion.

Algorithms

1. Start with an arbitrary labeling f
2. Set success := 0
3. For each pair of labels $\{\alpha, \beta\} \subset \mathcal{L}$
   3.1. Find $\hat{f} = \arg\min E(f')$ among f' within one α-β swap of f
   3.2. If $E(\hat{f}) < E(f)$, set f := $\hat{f}$ and success := 1
4. If success = 1 goto 2
5. Return f

---

1. Start with an arbitrary labeling f
2. Set success := 0
3. For each label $\alpha \in \mathcal{L}$
   3.1. Find $\hat{f} = \arg\min E(f')$ among f' within one α-expansion of f
   3.2. If $E(\hat{f}) < E(f)$, set f := $\hat{f}$ and success := 1
4. If success = 1 goto 2
5. Return f

Finding optimal Swap move

- Given an input labeling f (partition $\mathcal{P}$) and a pair of labels α, β we want to find a labeling $\hat{f}$ that minimizes E over all labelings within one $\alpha - \beta$-swap of f.
- This is going to be done by computing a labeling corresponding to a minimum cut on a graph $\mathcal{G}_{\alpha\beta} = (\mathcal{V}_{\alpha\beta}, \mathcal{E}_{\alpha\beta})$.

Finding optimal Swap move

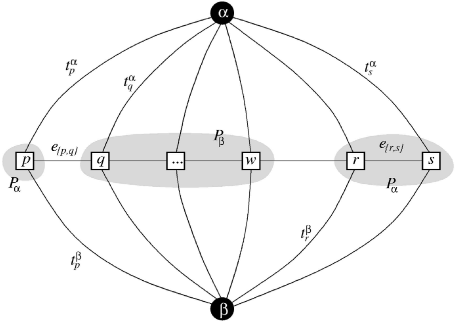
- Given an input labeling f (partition $\mathcal{P}$) and a pair of labels α, β we want to find a labeling $\hat{f}$ that minimizes E over all labelings within one $\alpha - \beta$-swap of f.
- This is going to be done by computing a labeling corresponding to a minimum cut on a graph $\mathcal{G}_{\alpha\beta} = (\mathcal{V}_{\alpha\beta}, \mathcal{E}_{\alpha\beta})$.
- The structure of this graph is dynamically determined by the current partition $\mathcal{P}$ and by the labels α, β.

Finding optimal Swap move

- Given an input labeling f (partition $\mathcal{P}$) and a pair of labels α, β we want to find a labeling $\hat{f}$ that minimizes E over all labelings within one $\alpha - \beta$-swap of f.
- This is going to be done by computing a labeling corresponding to a minimum cut on a graph $\mathcal{G}_{\alpha\beta} = (\mathcal{V}_{\alpha\beta}, \mathcal{E}_{\alpha\beta})$.
- The structure of this graph is dynamically determined by the current partition $\mathcal{P}$ and by the labels α, β.

Graph Construction

- The set of vertices includes the two terminals α and β, as well as image pixels p in the sets $\mathcal{P}_\alpha$ and $\mathcal{P}_\beta$ (i.e., $f_p \in \{\alpha, \beta\}$).
- Each pixel $p \in \mathcal{P}_{\alpha\beta}$ is connected to the terminals α and β, called t-links.
- Each set of pixels $p, q \in \mathcal{P}_{\alpha\beta}$ which are neighbors is connected by an edge $e_{p,q}$



edge	weight	for
t_p^α	$D_p(\alpha) + \sum_{q \in \mathcal{N}_p, q \notin \mathcal{P}_{\alpha\beta}} V(\alpha, f_q)$	$p \in \mathcal{P}_{\alpha\beta}$
t_p^β	$D_p(\beta) + \sum_{q \in \mathcal{N}_p, q \notin \mathcal{P}_{\alpha\beta}} V(\beta, f_q)$	$p \in \mathcal{P}_{\alpha\beta}$
$e_{\{p,q\}}$	$V(\alpha, \beta)$	$\{p,q\} \in \mathcal{N}$, $p,q \in \mathcal{P}_{\alpha\beta}$

Computing the Cut

- Any cut must have a single t-link not cut.
- This defines a labeling

$$f_p^{\mathcal{C}} = \begin{cases} \alpha & \text{if } \; t_p^{\alpha} \in \mathcal{C} \; \text{ for } \; p \in \mathcal{P}_{\alpha\beta} \\ \beta & \text{if } \; t_p^{\beta} \in \mathcal{C} \; \text{ for } \; p \in \mathcal{P}_{\alpha\beta} \\ f_p & \text{for } \; p \in \mathcal{P}, \; p \notin \mathcal{P}_{\alpha\beta}. \end{cases}$$

- There is a one-to-one correspondences between a cut and a labeling.
- The energy of the cut is the energy of the labeling.
- See Boykov et al, "*fast approximate energy minimization via graph cuts*" PAMI 2001.

Properties

- For any cut, then

$$
\begin{array}{llllll}
(a) & \textit{If} & t_p^\alpha, t_q^\alpha \in \mathcal{C} & \textit{then} & e_{\{p,q\}} \notin \mathcal{C}. \\
(b) & \textit{If} & t_p^\beta, t_q^\beta \in \mathcal{C} & \textit{then} & e_{\{p,q\}} \notin \mathcal{C}. \\
(c) & \textit{If} & t_p^\beta, t_q^\alpha \in \mathcal{C} & \textit{then} & e_{\{p,q\}} \in \mathcal{C}. \\
(d) & \textit{If} & t_p^\alpha, t_q^\beta \in \mathcal{C} & \textit{then} & e_{\{p,q\}} \in \mathcal{C}.
\end{array}
$$