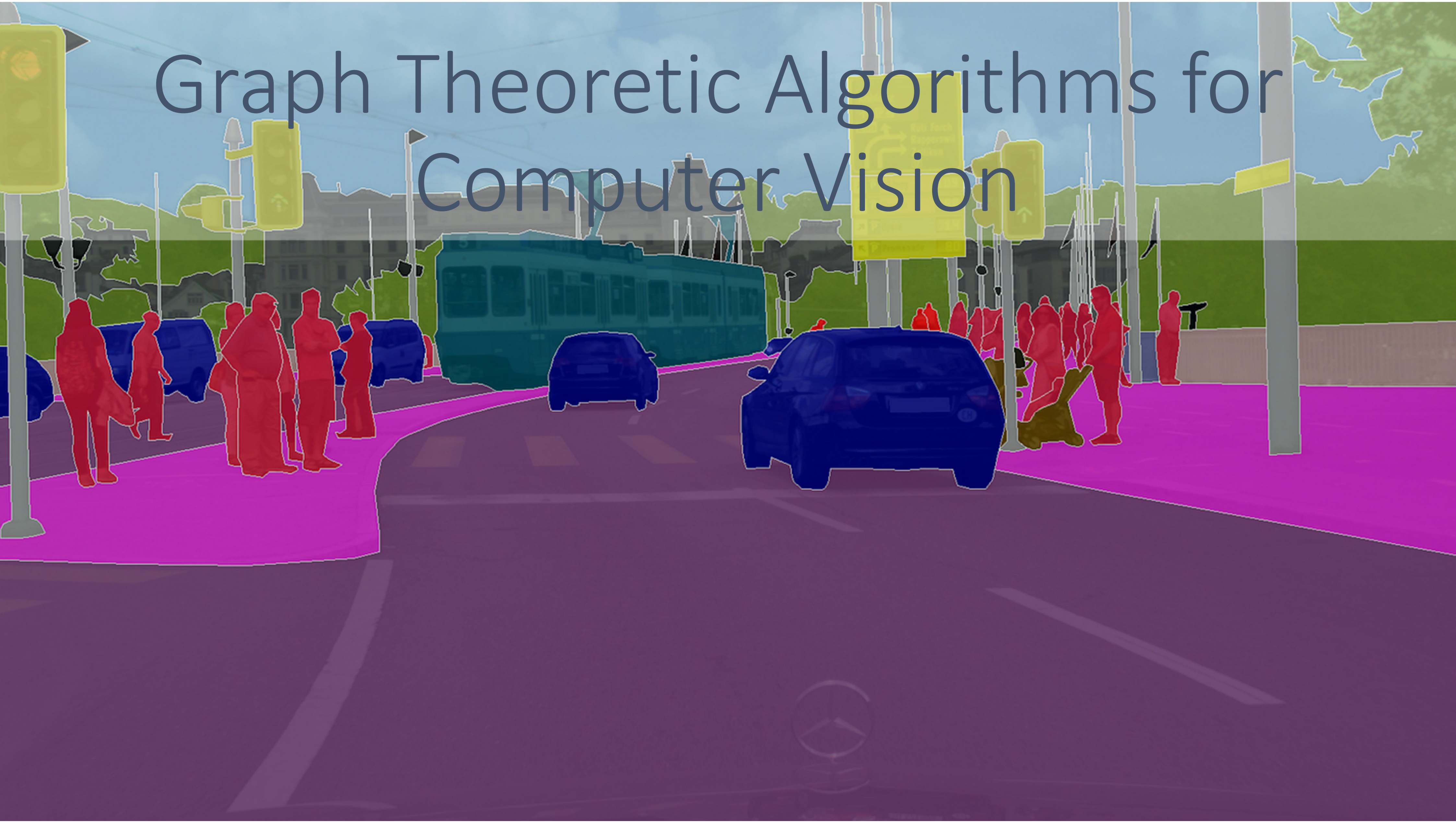
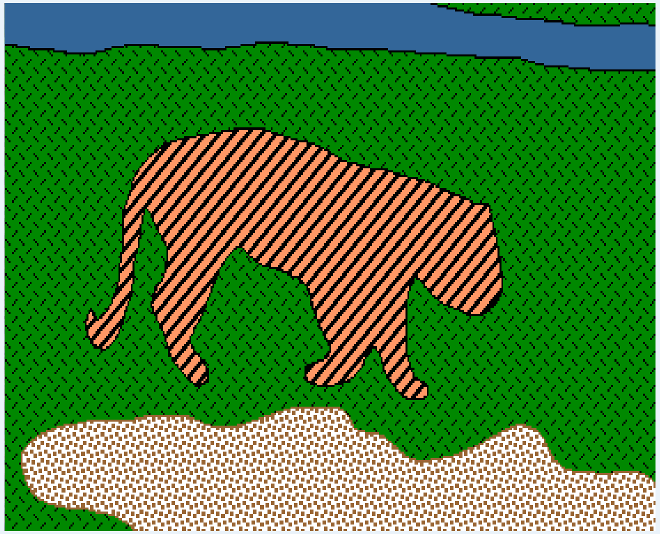
Graph Theoretic Algorithms for Computer Vision



Agenda

- Introduction
- Algorithms
- Conclusion

Introduction: Image Segmentation

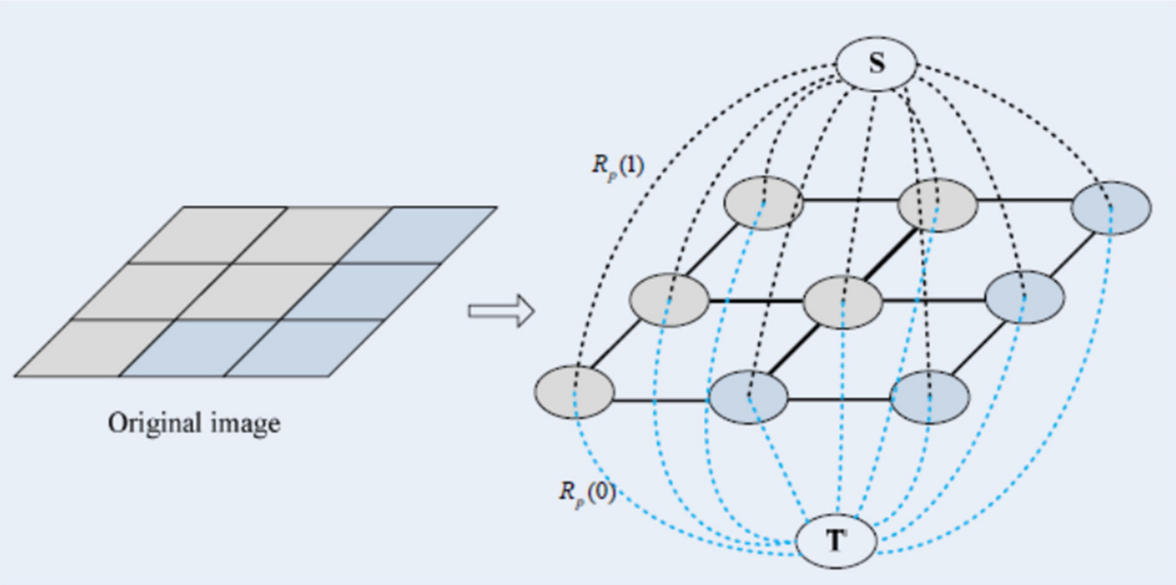


Identify groups of pixels that “go together”

Approaches and uses

Solutions: Intuition

- Turn the image into a ‘weighted graph’
- Divide the ‘graph’ into pieces under some constraint
- Cuts constitute borders between regions

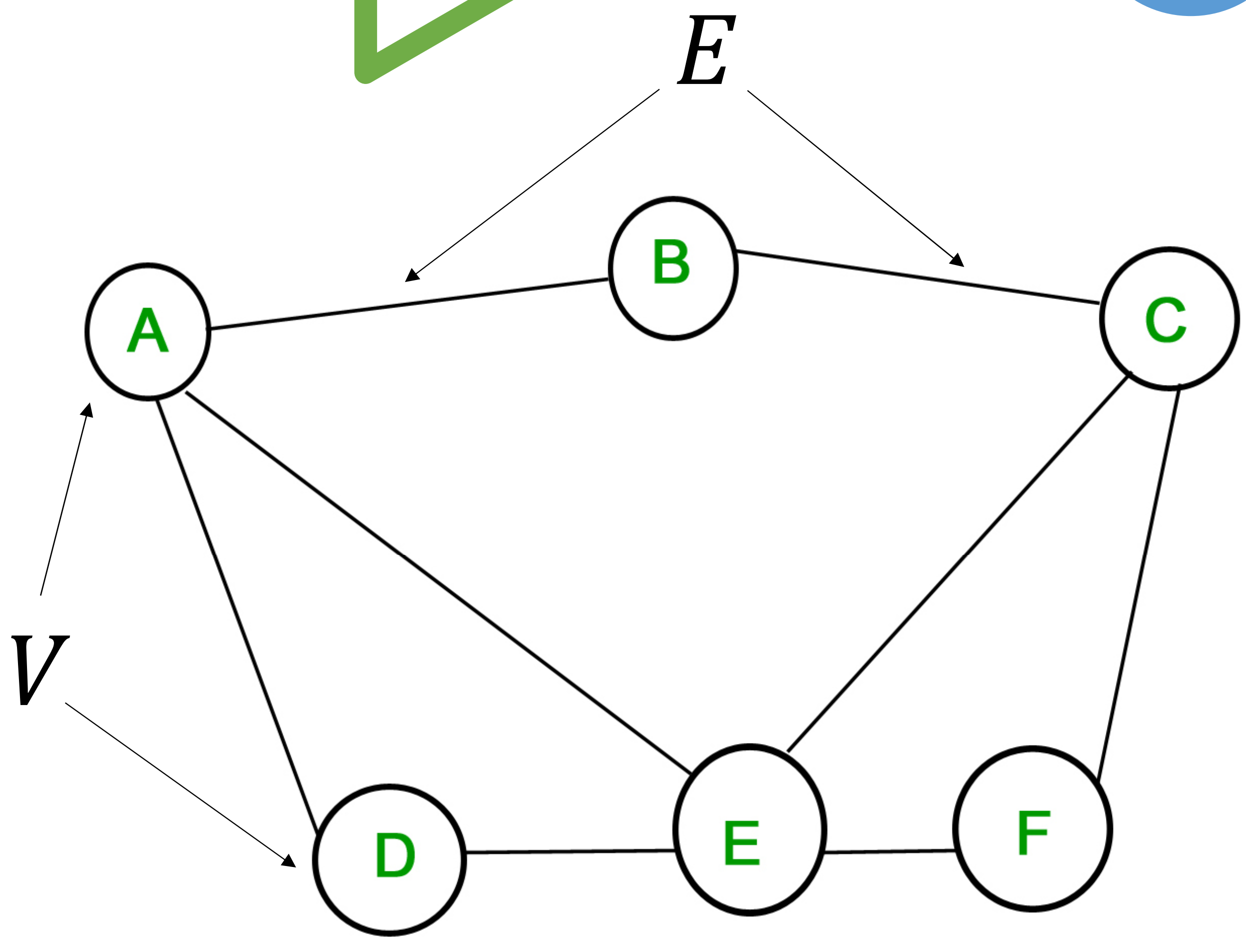


Normalized Cuts and Image Segmentation

- Graphs, weighted edges.
- Graph Cut, Min Cut, Association, Normalized Cut.
- Graph as matrix.
- Images as matrix.
- Nodes as pixels, edges as…
- Graph and Segmentation.

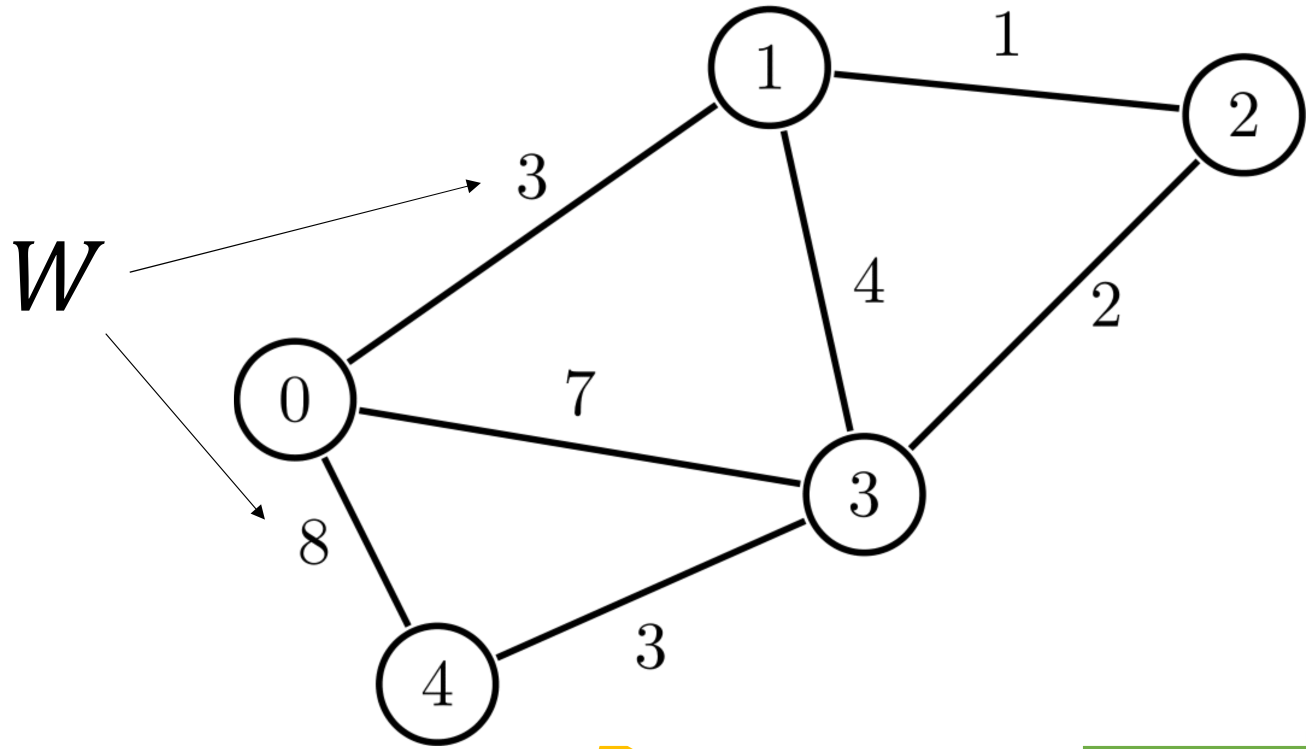
Graph

- $G = \langle V, E \rangle$
- V set of nodes
- $v_1, v_2, v_3 \in V$
- E set of edges
- $e_1, e_2, e_3 \in E$

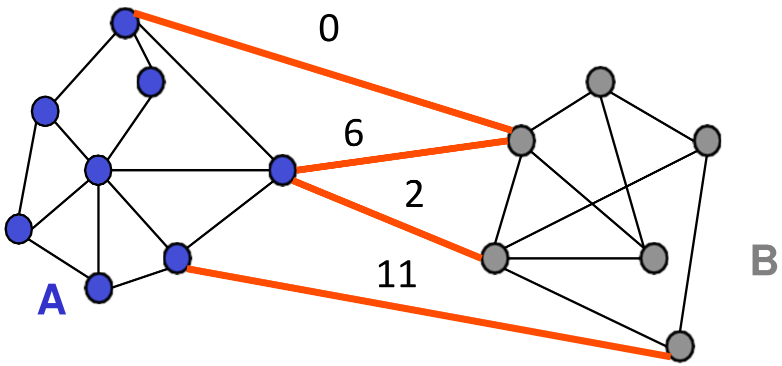


Graph with weights

- $e_1, e_2, e_3 \in E$
- $W(e_i) = w_i$
- $w_1, w_2, w_3 \in W$
- $w_i \in N \; or \; R$



Graph Cut



- Graph Cut = edges whose removal partitions a graph in two
- Formally: $A, B \mid A \cup B = V, A \cap B = \emptyset$
- Cost of a cut

  Sum of weights of cut edges:

$$cut(A,B) = \sum_{\substack{p \in A \\ q \in B}} w_{p,q}$$

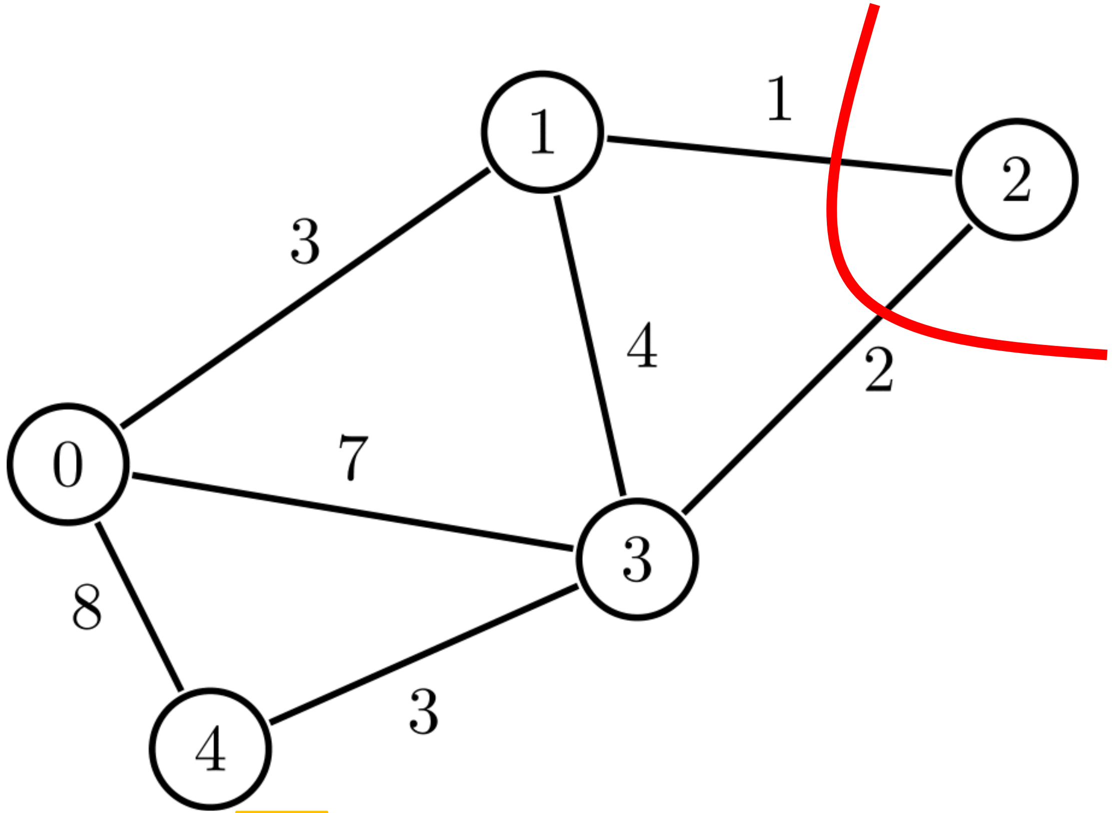
Min Cut

What is minimum cut?

$$\textit{min-cut}(A, B) =$$

$$min\left(\sum_{p\in A, q\in B} w_{p,q}\right)$$

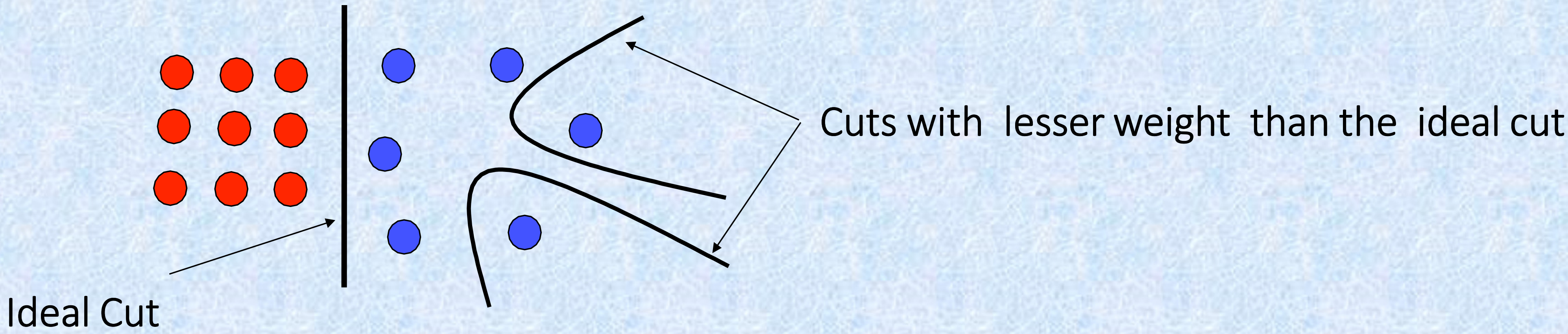
What is the minimum cut here?



Minimum Cut

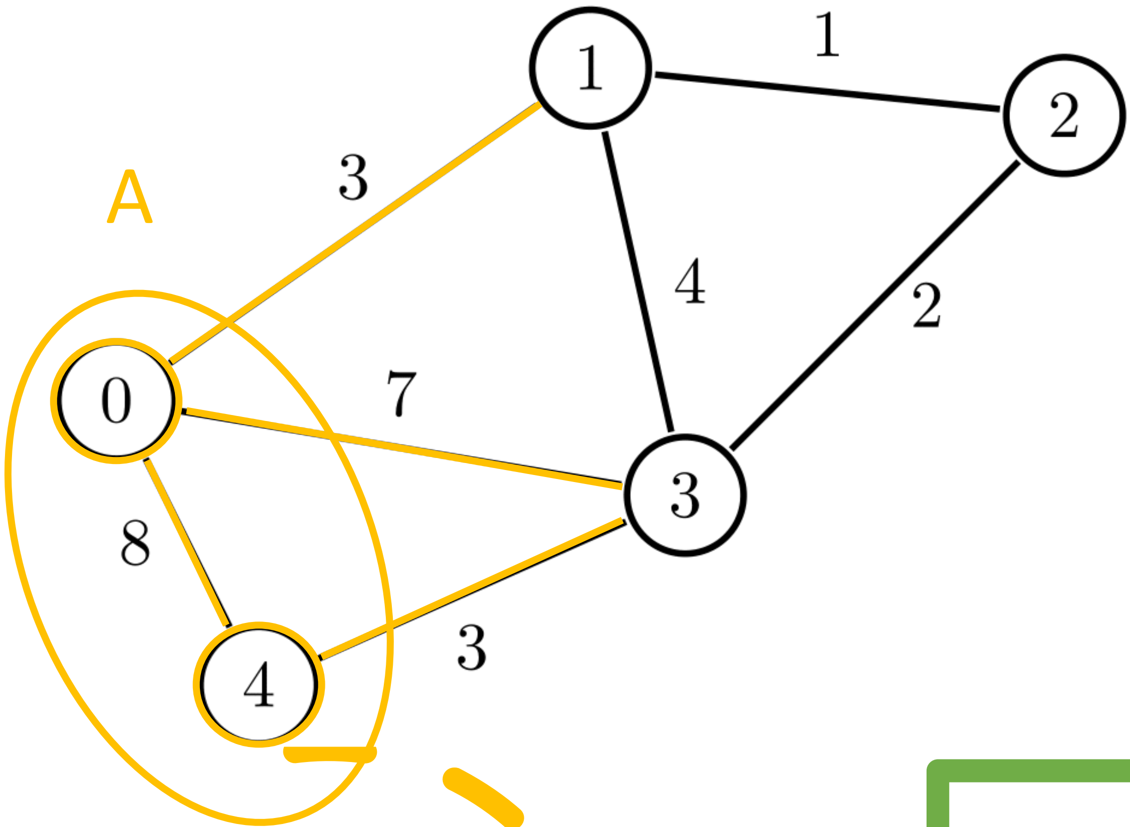
- We can do segmentation by finding the *minimum cut* in a graph
  - Min-cut can partition a graph into different objects in the image.
  - Efficient algorithms exist for finding min-cut.

- Drawback:
  - Weight of cut is proportional to number of edges in the cut.
  - Minimum cut tends to cut off very small, isolated components.



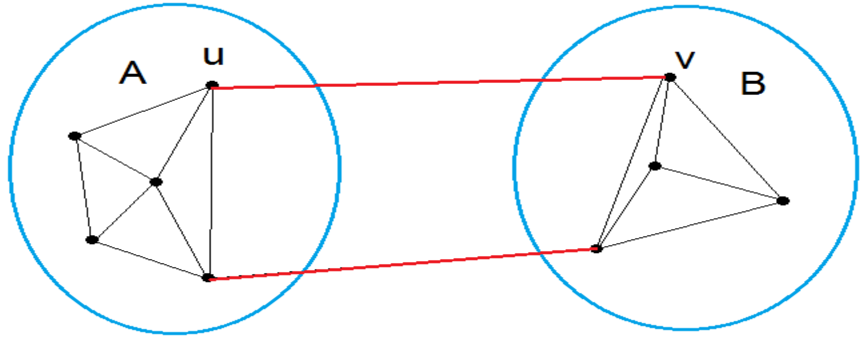
Set association

$$assoc(A,V) = \sum_{\substack{p \in A \\ q \in B}} w_{p,q} = 21$$

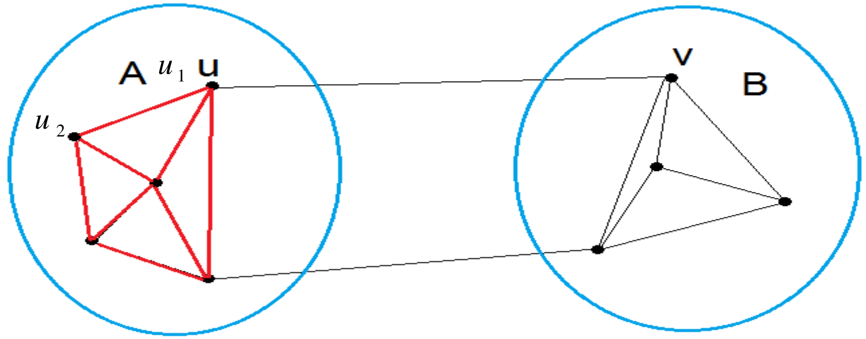


Vertex set association

$$Cut(A,B) = \sum_{u\in A, v\in B} w(u,v).$$

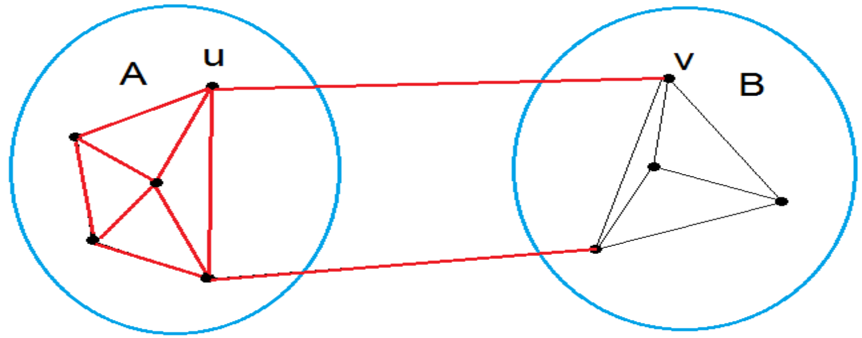


$$assoc(A,A) = \sum_{u_1,u_2\in A} w(u_1,u_2)$$



$$assoc(A,V) = assoc(A,A) + Cut(A,B).$$

$$assoc(A,V) = \sum_{u\in A, t\in V} w(u,t)$$



Normalized Cut Definition

- Cut costs are proportional to number of edges in cut
- Traditional graph cuts bias towards cutting small parts of the graph
- We can fix this by normalizing by $assoc$

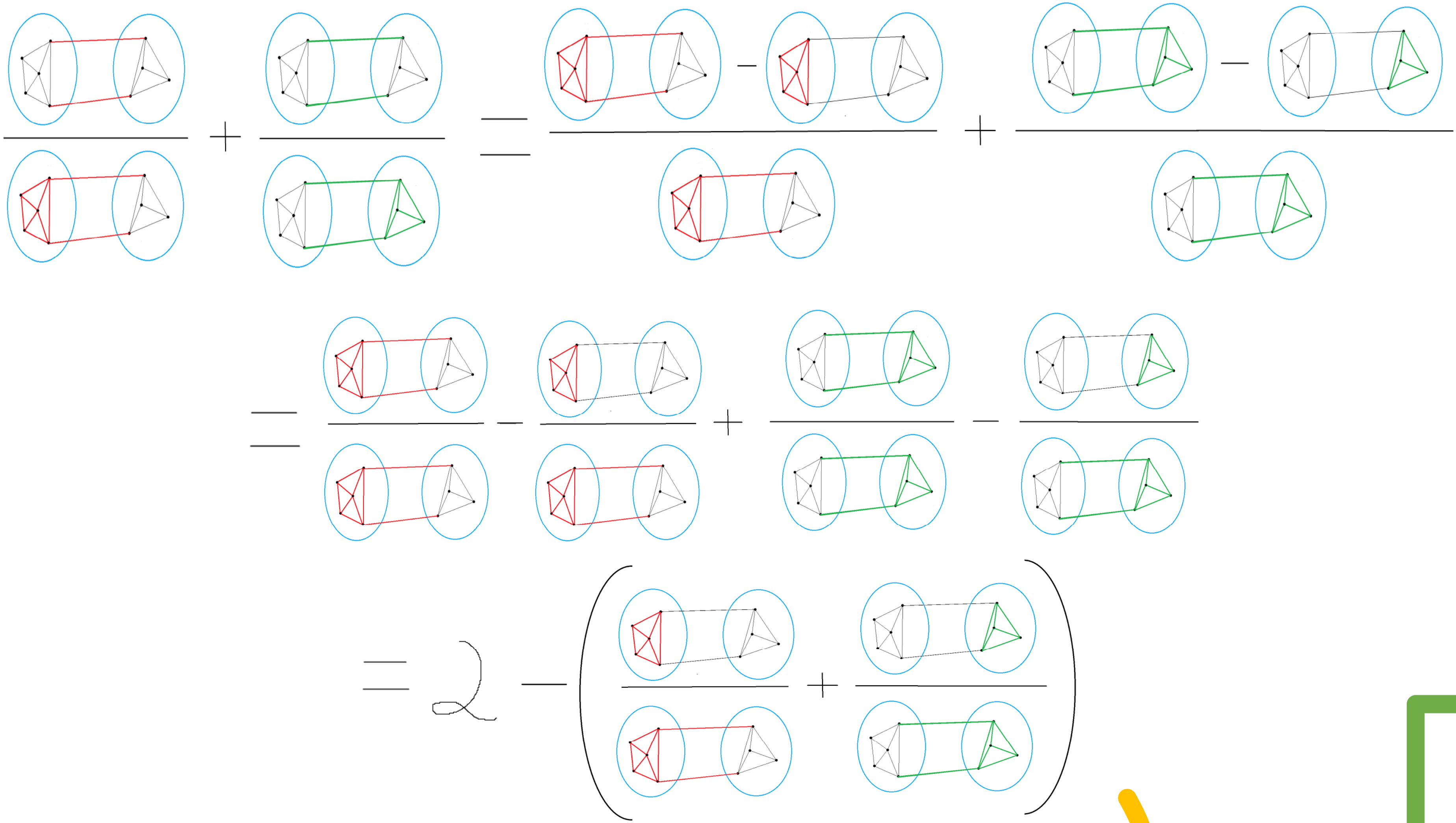
$$Ncut(A,B) = \frac{Cut(A,B)}{assoc(A,V)} + \frac{Cut(A,B)}{assoc(B,V)}$$

Normalized Cut and Assoc

$$Ncut(A,B) = \frac{cut(A,B)}{assoc(A,V)} + \frac{cut(A,B)}{assoc(B,V)}$$

$$Nassoc(A,B) = \frac{assoc(A,A)}{assoc(A,V)} + \frac{assoc(B,B)}{assoc(B,V)}$$

Relationship between Ncut and Nassoc



Normalized Cut (NCut)

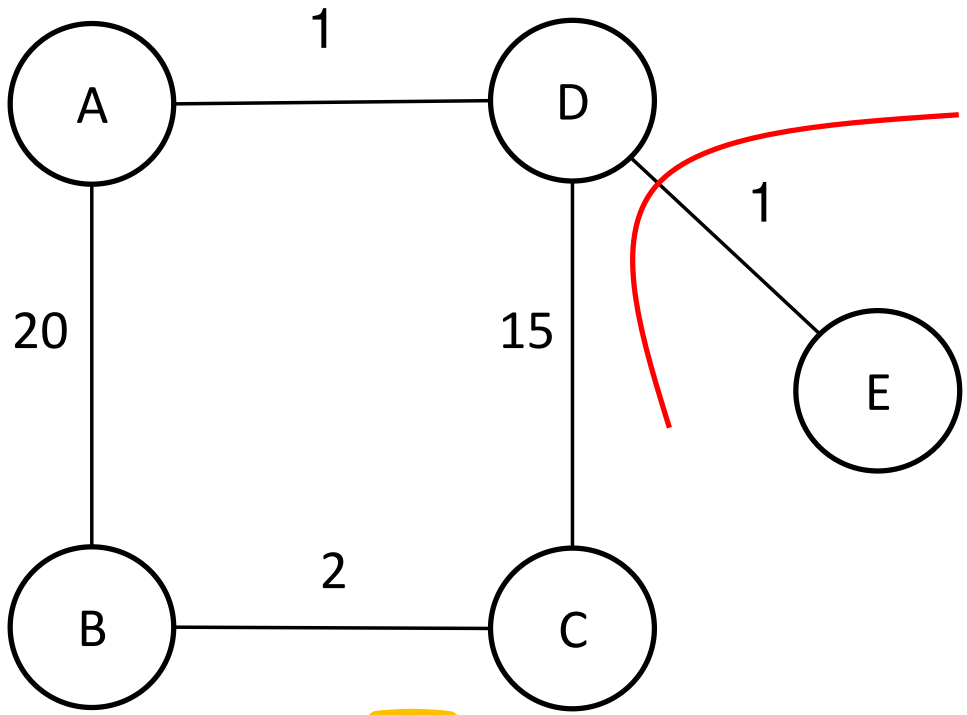
$$Ncut(A,B) = \frac{cut(A,B)}{assoc(A,V)} + \frac{cut(A,B)}{assoc(B,V)}$$

$$= \frac{assoc(A,V) - assoc(A,A)}{assoc(A,V)} + \frac{assoc(B,V) - assoc(B,B)}{assoc(B,V)}$$

$$= 2 - \left(\frac{assoc(A,A)}{assoc(A,V)} + \frac{assoc(B,B)}{assoc(B,V)}\right) = 2 - Nassoc(A,B)$$

Normalized Cut (NCut)

What is the Min-Cut here?

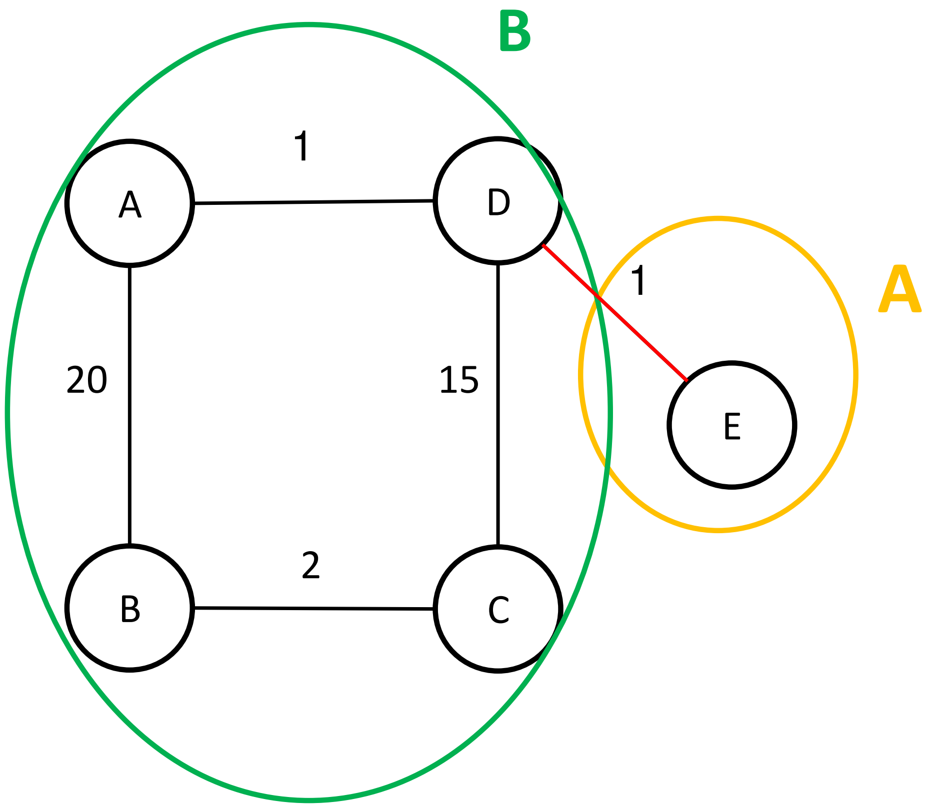


Normalized Cut example 1

$$Nassoc(A,B) = \frac{assoc(A,A)}{assoc(A,V)} + \frac{assoc(B,B)}{assoc(B,V)}$$

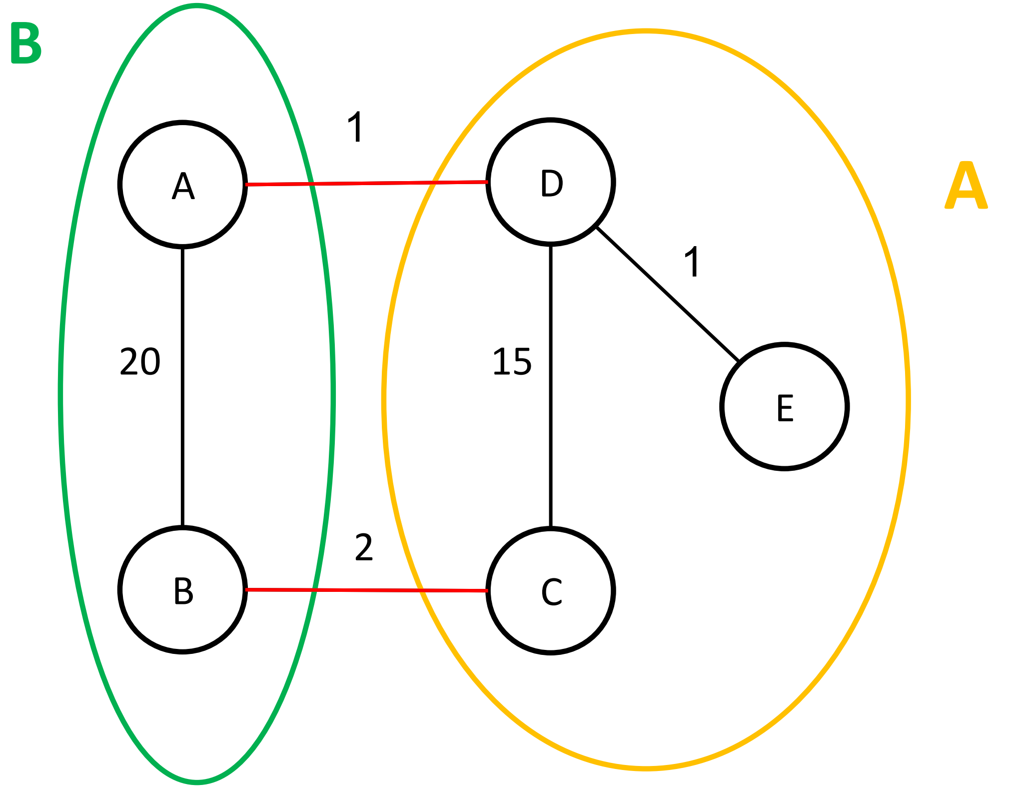
$$Ncut(A,B) = 2 - Nassoc(A,B)$$

$$NCut(A,B) = 2 - \left(\frac{0}{1} + \frac{38}{39}\right) = \sim 1.025$$



Normalized Cut example 2

$$Nassoc(A,B) = \frac{assoc(A,A)}{assoc(A,V)} + \frac{assoc(B,B)}{assoc(B,V)}$$



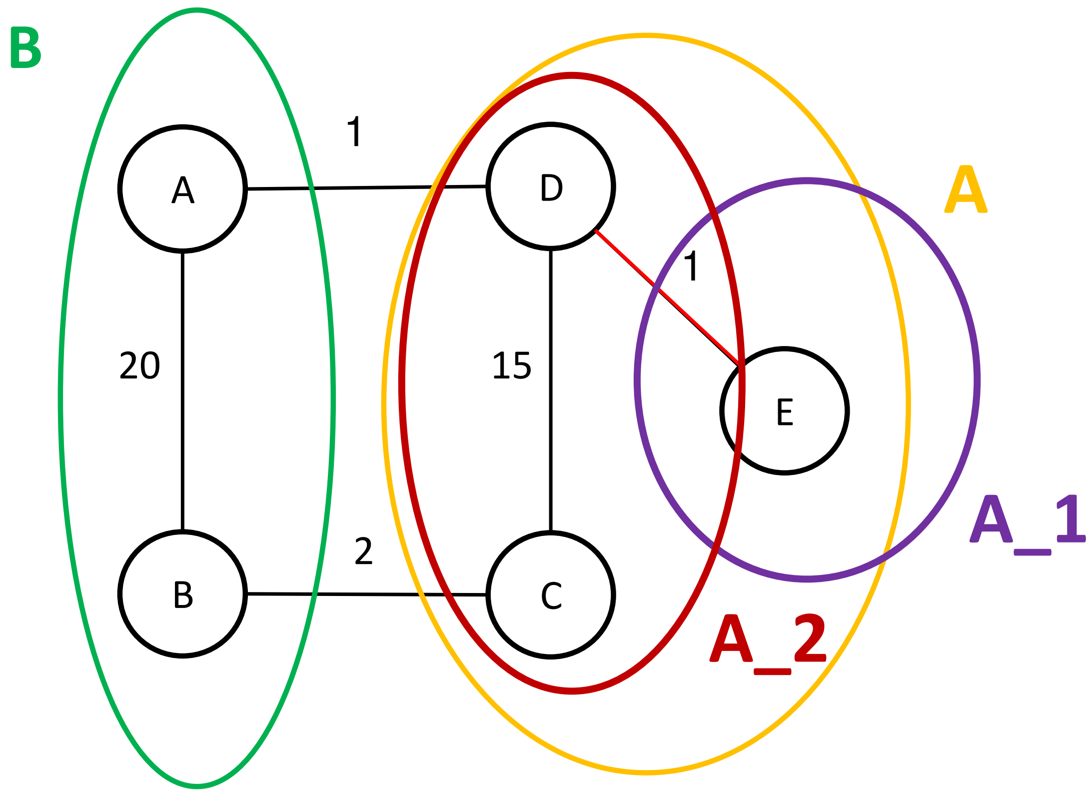
$$NCut(A,B) = 2 - \left(\frac{20}{23} + \frac{16}{19}\right) = \sim 0.288$$

$$0.288 < 1.025$$

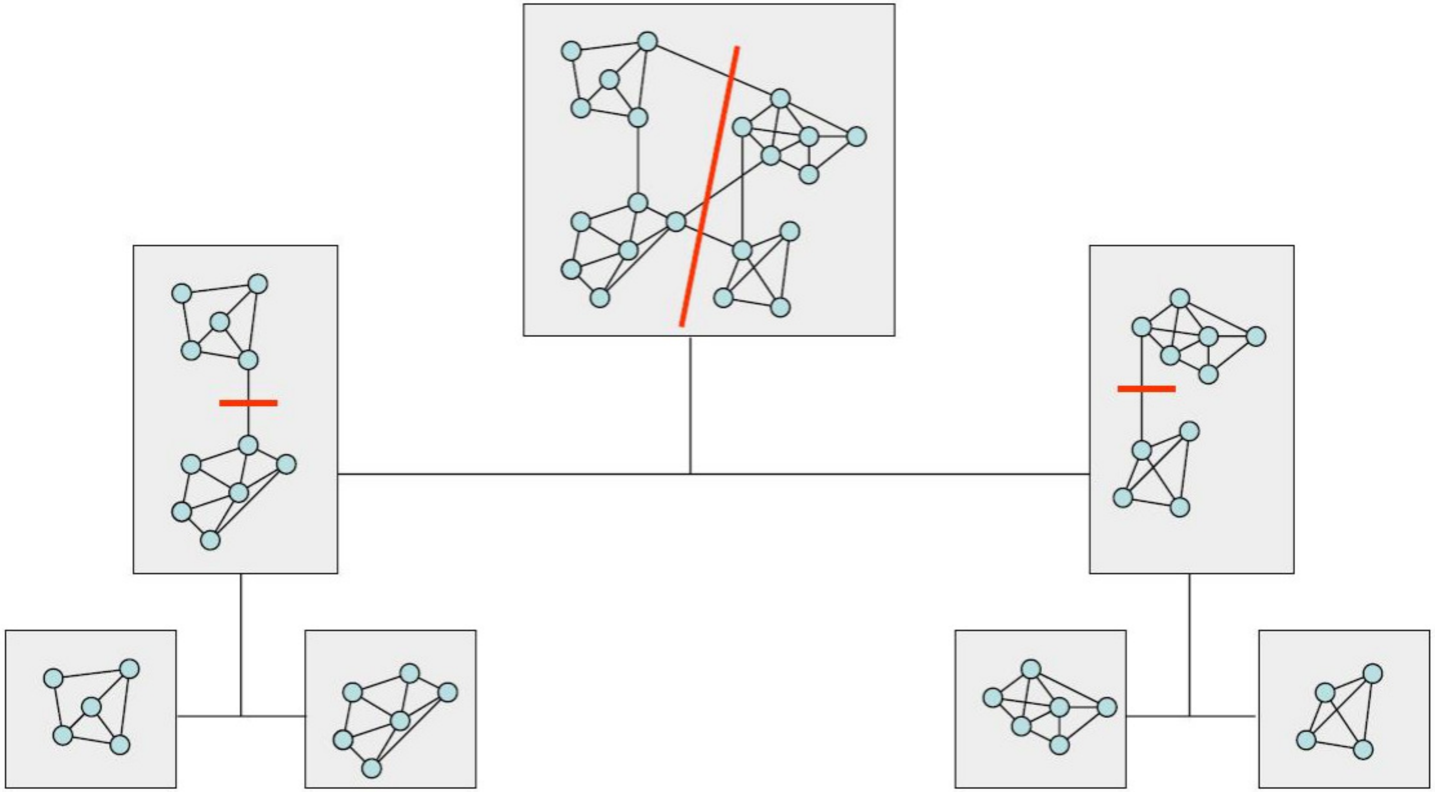
Recursive NCut

- More partitions can be done by subdividing the segmented parts.

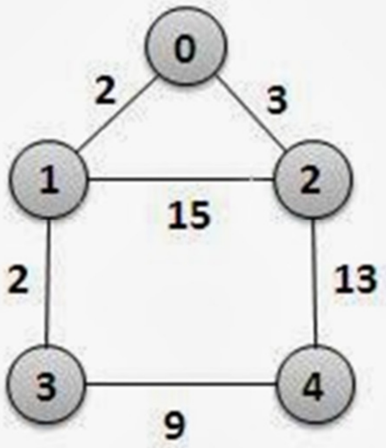
- Basically, finding new NCuts in the new parts A and B.



Recursive Two-Way Ncut example

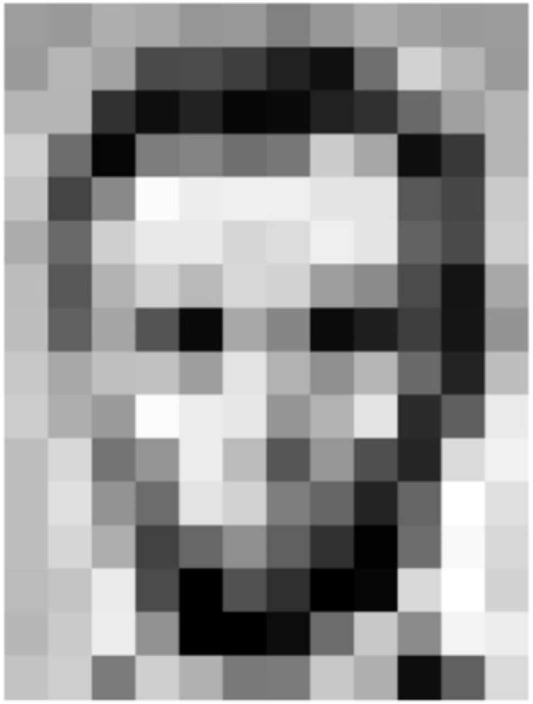


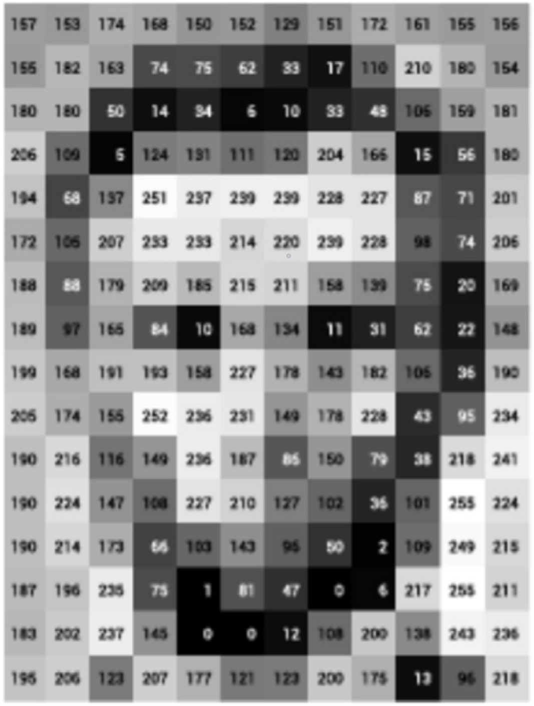
Graph as a matrix



	0	1	2	3	4
0	0	2	3	0	0
1	2	0	15	2	0
2	3	15	0	0	13
3	0	2	0	0	9
4	0	0	13	9	0

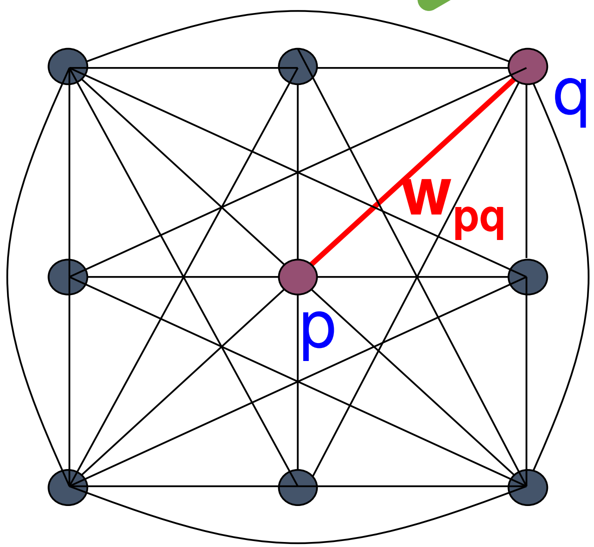
Image as a matrix

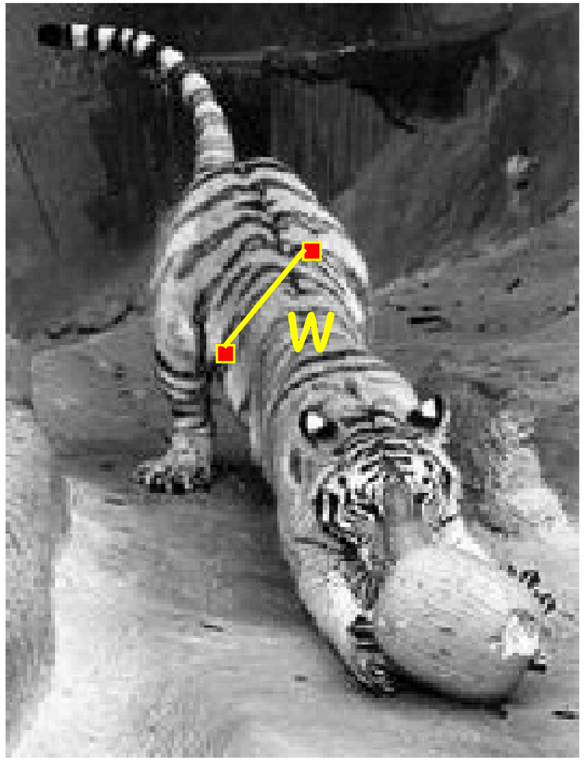




157	153	174	168	150	152	129	151	172	161	155	156
155	182	163	74	75	62	33	17	110	210	180	154
180	180	50	14	34	6	10	33	48	106	159	181
206	109	5	124	131	111	120	204	166	15	56	180
194	68	137	251	237	239	239	228	227	87	71	201
172	105	207	233	233	214	220	239	228	98	74	206
188	88	179	209	185	215	211	158	139	75	20	169
189	97	165	84	10	168	134	11	31	62	22	148
199	168	191	193	158	227	178	143	182	106	36	190
205	174	155	252	236	231	149	178	228	43	95	234
190	216	116	149	236	187	86	150	79	38	218	241
190	224	147	108	227	210	127	102	36	101	255	224
190	214	173	66	103	143	96	50	2	109	249	215
187	196	235	75	1	81	47	0	6	217	255	211
183	202	237	145	0	0	12	108	200	138	243	236
195	206	123	207	177	121	123	200	175	13	96	218

Images as graphs





Fully-connected graph

- Node for every pixel
- Link between *every* pair of pixels, **p,q**
- Affinity weight $\mathbf{w_{pq}}$ for each link (edge)
- $\mathbf{w_{pq}}$ measures *similarity*
- Similarity is *inversely proportional* to differences (in color and position…)

The weight function

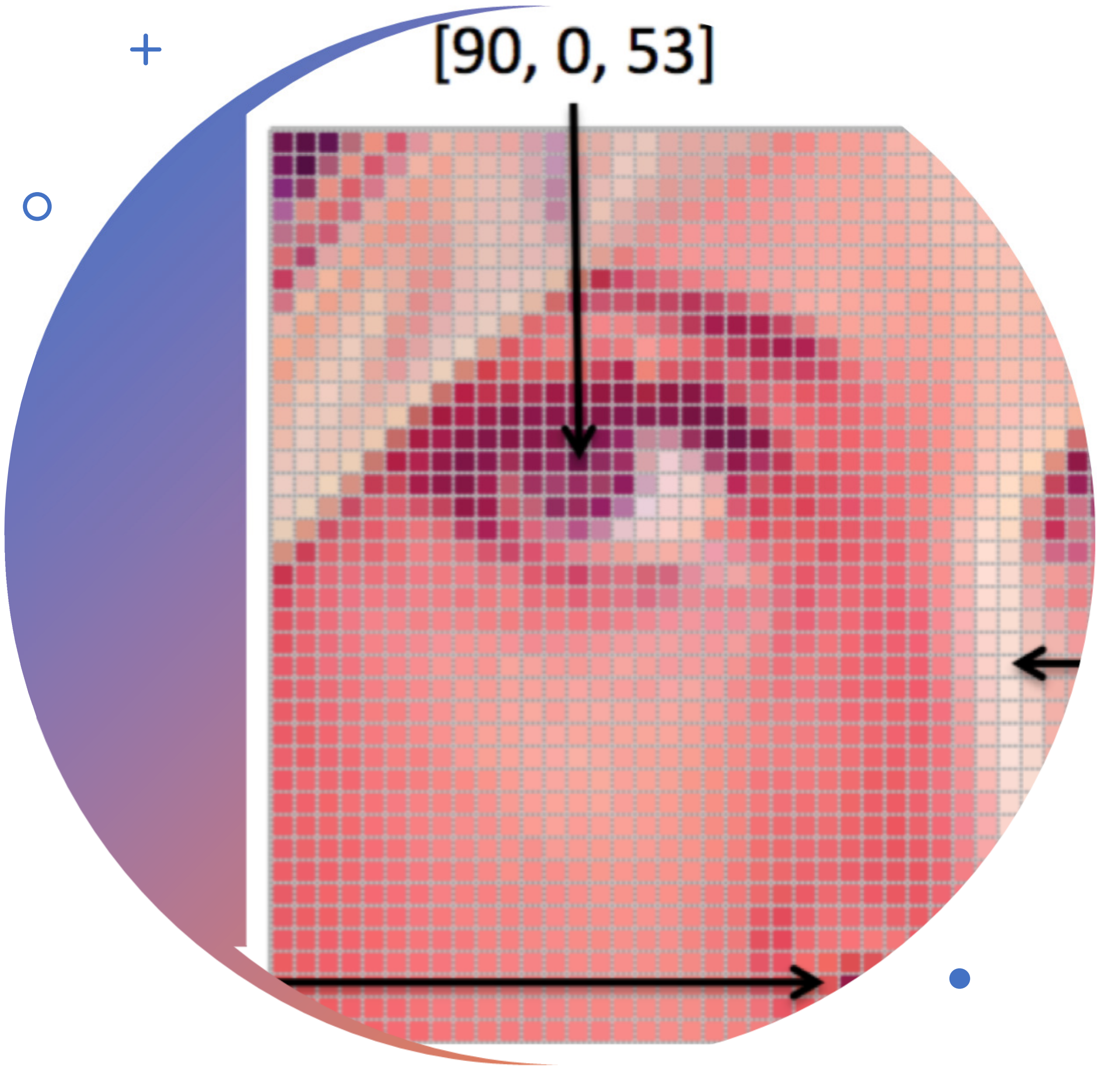
- A function that translate image feature into weights on edges
- For example: brightness (Intensity).

$X(i)$ spatial location of node i

$F(i)$ intensity function

$$w_{ij} = e^{\frac{-\|\boldsymbol{F}(i)-\boldsymbol{F}(j)\|_2^2}{\sigma_I}} * \begin{cases} e^{\frac{-\|\boldsymbol{X}(i)-\boldsymbol{X}(j)\|_2^2}{\sigma_X}} & \text{if } \|\boldsymbol{X}(i)-\boldsymbol{X}(j)\|_2 < r \\ 0 & \text{otherwise,} \end{cases}$$

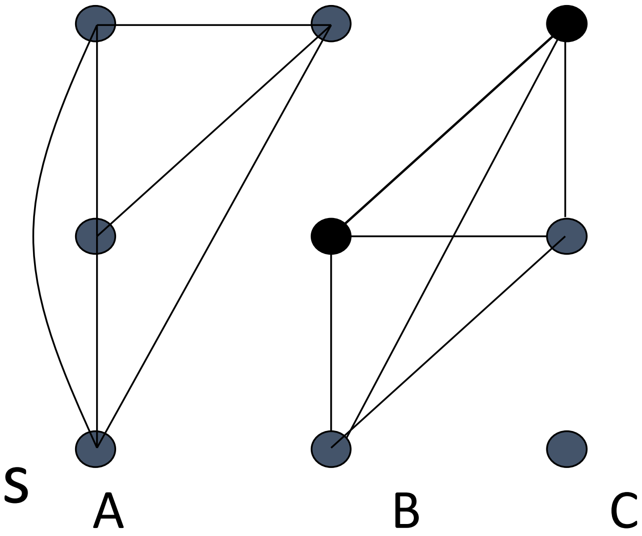
r is spatial distance



Graph and Segmentation

- A graph with weights can represent an image.
- We can address image processing tasks with graph processing tools.
- For example… segmentation.

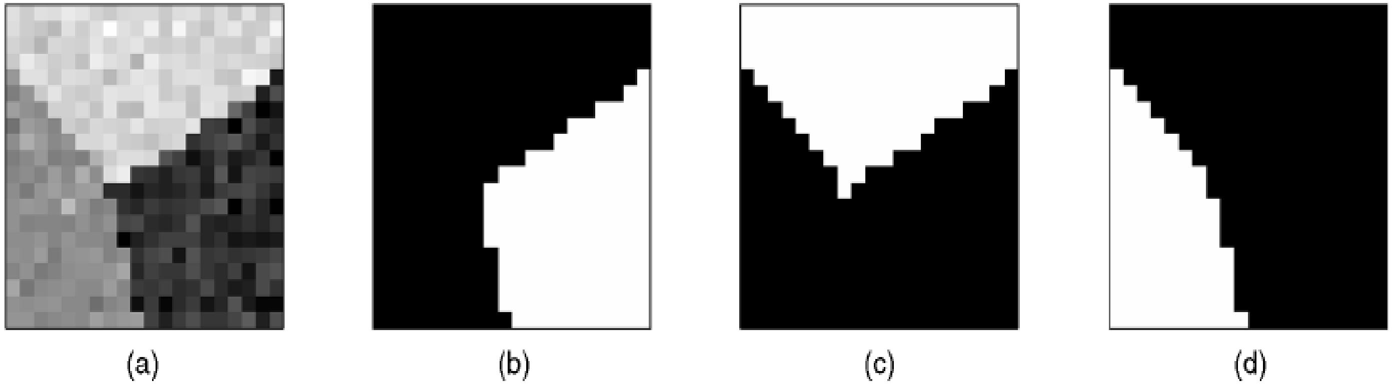
Segmentation by Normalized Cut

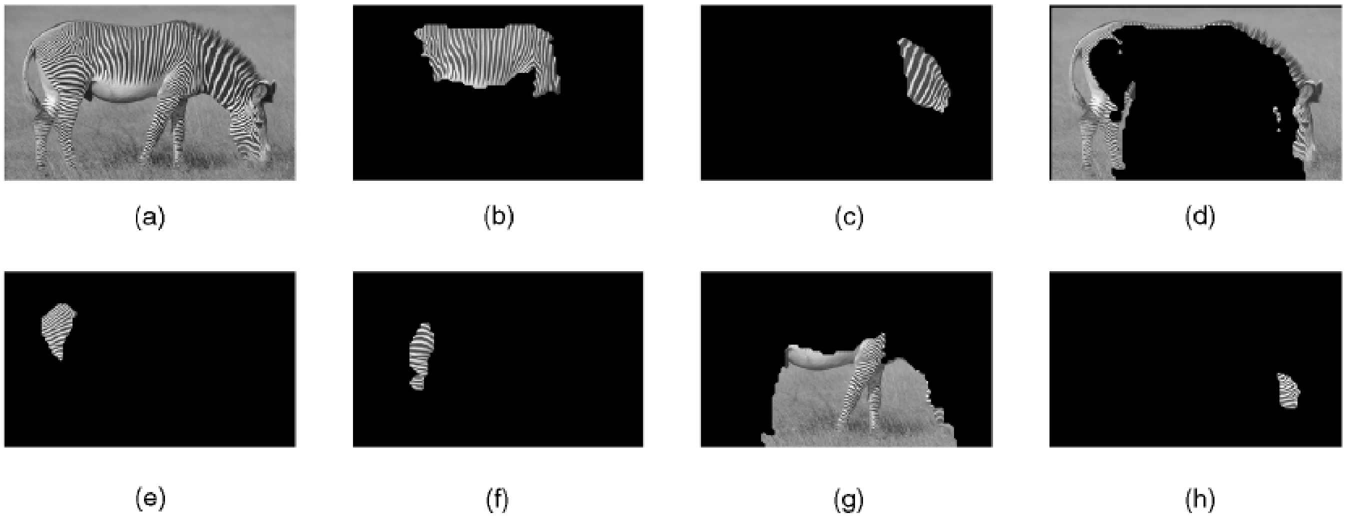


Break Graph into Segments

- Delete links that cross between segments
- Easiest to break links that have low affinity
- similar pixels should be in the same segments
- dissimilar pixels should be in different segments

Intensity and DOOG based graph cut





Normalized Cuts and Image Segmentation

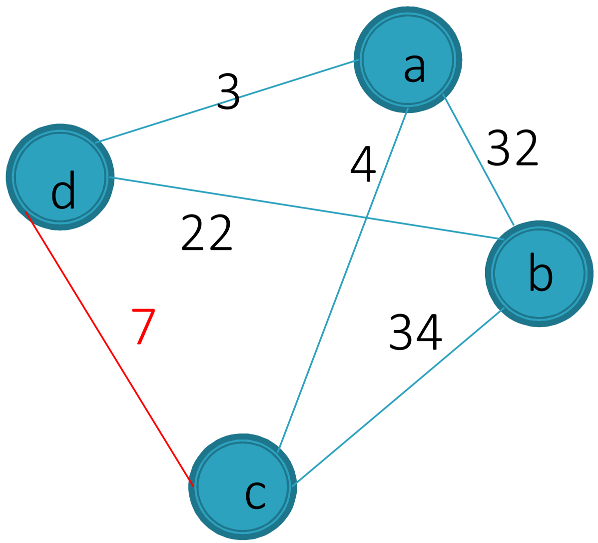
- Minimizing NCut is NP-complete.
- Approximate discrete solution can be found efficiently in the real value domain.
- How? Eigenvectors!
- Some definitions first…

Normalized Cuts and Image Segmentation

- Adjacency, Diagonal and Laplacian matrix.
- Rayleigh Quotient.
- Segment $'y'$ as Indicator vector
- Normalized cut algorithm.

Adjacency matrix

W is the Adjacency matrix of the graph, where every w_{ij} is the weight of edge $e(i,j)$ where $i,j \in V$.



$$W \Rightarrow \begin{array}{c c} & \begin{array}{cccc} a & b & c & d \end{array} \\ \begin{array}{c} a \\ b \\ c \\ d \end{array} & \begin{pmatrix} 0 & 32 & 4 & 3 \\ 32 & 0 & 34 & 22 \\ 4 & 34 & 0 & 7 \\ 3 & 22 & 7 & 0 \end{pmatrix} \end{array}$$

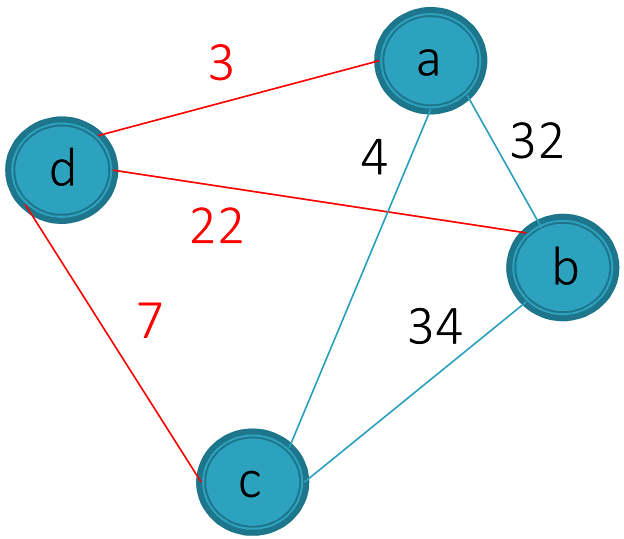
$$w(d,c) = 7 = w_{d,c}$$

Diagonal matrix

D is the Diagonal matrix of the graph with diagonal entries

$$D(i,i) = \sum_j w(i,j)$$

($D(i)$ = the sum of all edge with one end in $i \in V$)



$$D \Rightarrow \begin{array}{c c} & \begin{array}{cccc} a & b & c & d \end{array} \\ \begin{array}{c} a \\ b \\ c \\ d \end{array} & \begin{pmatrix} 39 & 0 & 0 & 0 \\ 0 & 88 & 0 & 0 \\ 0 & 0 & 41 & 0 \\ 0 & 0 & 0 & \color{red}{32} \end{pmatrix} \end{array}$$

$$D(d,d) = 32 = D_{d,d} = D_d$$

Laplacian matrix

L is the Laplacian matrix L : $L = (D - W)$

$$\underbrace{\begin{pmatrix} 39 & 0 & 0 & 0 \\ 0 & 88 & 0 & 0 \\ 0 & 0 & 41 & 0 \\ 0 & 0 & 0 & 32 \end{pmatrix}}_{D} - \underbrace{\begin{pmatrix} 0 & 32 & 4 & 3 \\ 32 & 0 & 34 & 22 \\ 4 & 34 & 0 & 7 \\ 3 & 22 & 7 & 0 \end{pmatrix}}_{W} \; L \Rightarrow \underbrace{\begin{pmatrix} 39 & -32 & -4 & -3 \\ -32 & 88 & -34 & -22 \\ -4 & -34 & 41 & -7 \\ -3 & -22 & -7 & 32 \end{pmatrix}}_{L}$$

Laplacian matrix properties

$$L = (D - W)$$

- L is real symmetric and positive semi-definite

$$x^T L x \geq 0$$

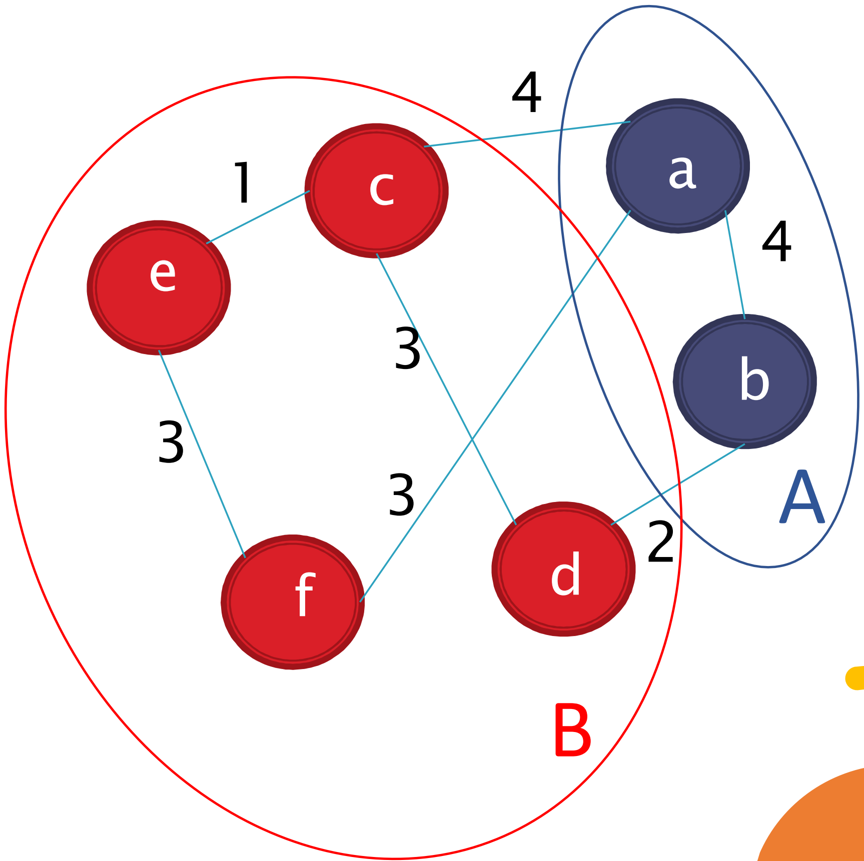
Eigenvectors are perpendicular and Eigenvalues are all non-negative.

- The smallest eigenvalue is always 0 with eigenvector $\mathbf{1}$.

$$0 = \lambda_1 \leq \lambda_2 \leq \lambda_3 \leq \ldots \leq \lambda_n \qquad \mathbf{1} = \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix}$$

Indicator vector

For a graph partition into two groups A and B, an indicator vector **y** is an n=|V| dimensional binary vector such that $y_i = 1$ for every $i \in A$ and $y_i = -1$ for $i \in B$.



$$\mathbf{y} = \begin{array}{c} \begin{pmatrix} 1 \\ 1 \\ -1 \\ -1 \\ -1 \\ -1 \end{pmatrix} \end{array} \begin{array}{c} a \\ b \\ c \\ d \\ e \\ f \end{array}$$

Normalized cut

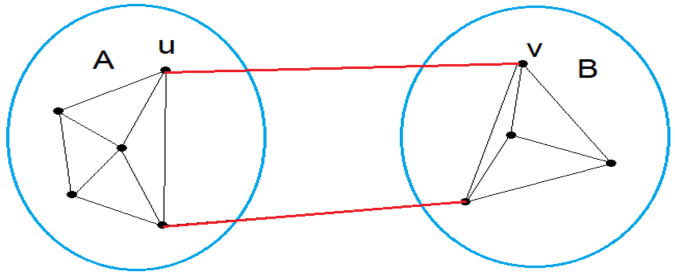
- *W* is the adjacency matrix of the graph
- *D* is the diagonal matrix of graph
- y is the indicator vector
- Then the normalized cut cost can be written as

$$\frac{y^T (D - W) y}{y^T D y} = \frac{1}{2} \sum_{ij} \frac{W_{ij} \left(y_i - y_j \right)^2}{D_{ii}}$$

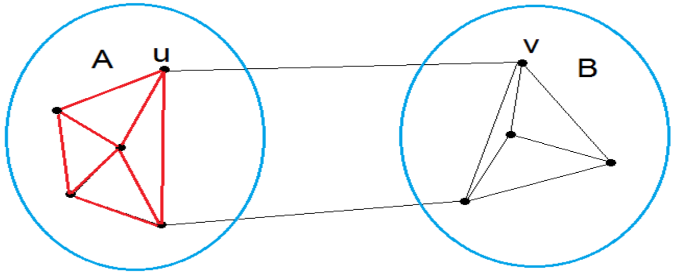
Goal is to minimize this constrained to $y_i \in \{1, -b\}$

Eigensystem min - intuition

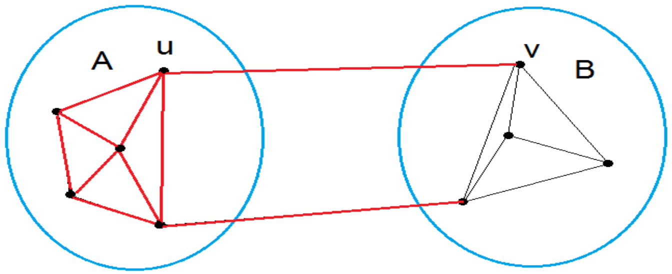
$$Cut(A,B) = \sum_{i\in A, j\in B} w_{ij}$$



$$assoc(A,A) = \sum_{i,j\in A} w_{ij} = y^T W y$$



$$assoc(A,V) = \sum_{i\in A, j\in V} w_{ij} = y^T D y$$



$$Cut(A,B) = assoc(A,V) - assoc(A,A)$$

$$NCut(A,B) = Cut(A,B) / assoc(A,V) = \frac{y^T (D - W) y}{y^T D y}$$

Rayleigh Quotient

- J. Shi and J. Malik proved (2000) using the Laplacian matrix properties, that:

$$\min_y Ncut(y) = \min_y \frac{y^T(D-W)y}{y^T D y}$$

L = D-W

- Where $y_i \in \{1, -b\}$ with some constant b:

$$b = \frac{\sum_{y_i>0} d_{ii}}{\sum_{y_i<0} d_{ii}}$$

Normalized cut

Relaxing y and allowing real values we can solve

$$y = \arg\min_y \frac{y^T (D - W) y}{y^T D y}$$

by using the *generalized eigenvalue problem:*

$$(D - W) y = \lambda D y$$

where y_i are the eigenvectors and the eigenvalues represent the cut cost.

Normalized cut

The smallest eigenvector is always 0 , because we can have a partition of $A = V$ and $B = \emptyset$ $(\boldsymbol{y} = \mathbf{1})$ thus $Ncut(A, B) = 0$

$$\boldsymbol{1}^T \, L \, \boldsymbol{1} = 0$$

<u>**Second**</u> smallest eigenvector is the **real-valued y** that minimizes Ncut and is the solution for

$$(D - W)\, y = \lambda D y$$

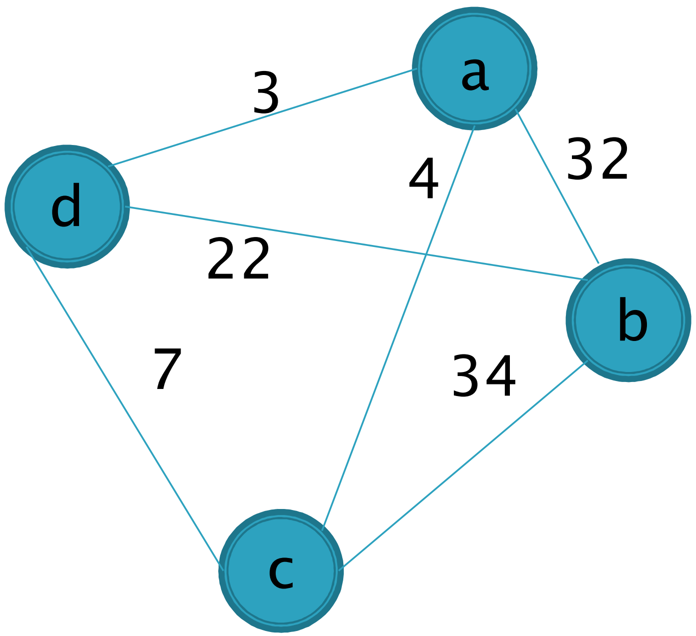
Use a threshold to differentiate between the two segments.

Normalized cut

Returning to discrete world.

The eigenvector *y* will hopefully have similar values for nodes with high similarity - high $w(i,j)$

Thus, a threshold T on the eigenvector entries creates a binary classification of nodes.



$\lambda = 0.99$	
a	−1.6
b	−0.9
c	0.9
d	2

$T = 0$

$\Rightarrow$

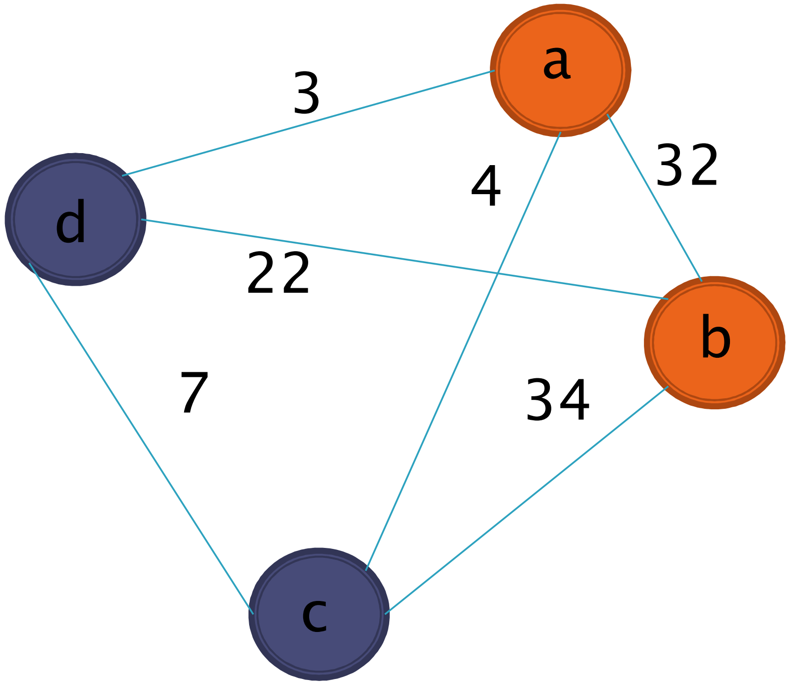
a	1
b	1
c	−b
d	−b

Normalized cut

Returning to discrete world.

The eigenvector y will hopefully have similar values for nodes with high similarity - high $w(i,j)$

Thus, a threshold T on the eigenvector entries creates a binary classification of nodes.



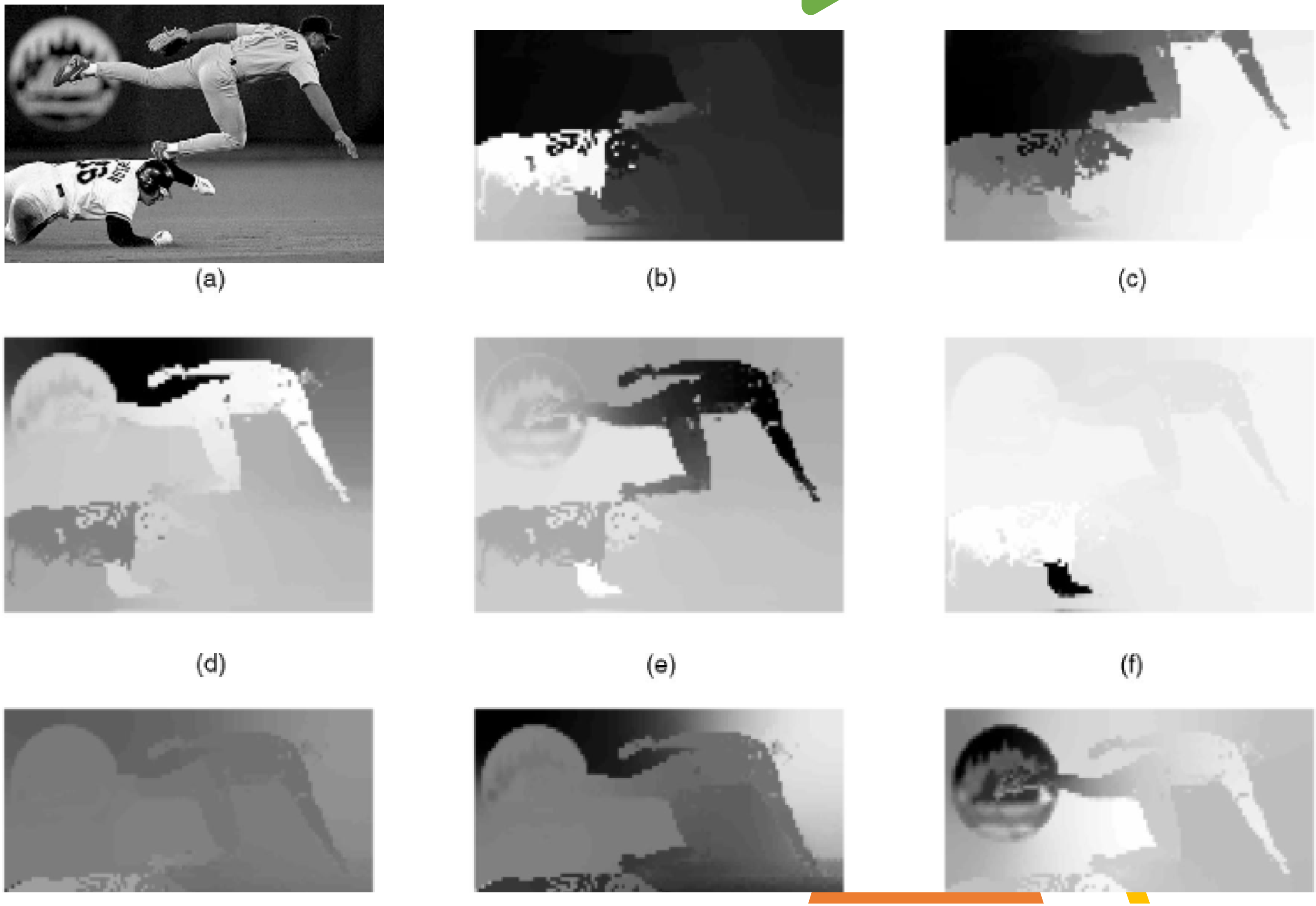
$\lambda = 0.99$	
a	−1.6
b	−0.9
c	0.9
d	2

$$T = 0$$

$$\Rightarrow$$

a	1
b	1
c	−b
d	−b

Eigenvectors corresponding to eigenvalues consecutively



Approximate discrete solution

- Rayleigh quotient: minimize $\frac{x^T A x}{x^T x}$ where A is symmetric
- Let $x_1 \dots x_n$ be eigenvectors of A with $\lambda_1 \leq \cdots \leq \lambda_n$
- Under the constraint $x \perp x_1 \dots x_{j-1}$ the minimizing solution is x_j
- Since we are constraining $y \perp \mathbf{1}$, and $\mathbf{1}$ is x_1, the solution to our problem is x_2

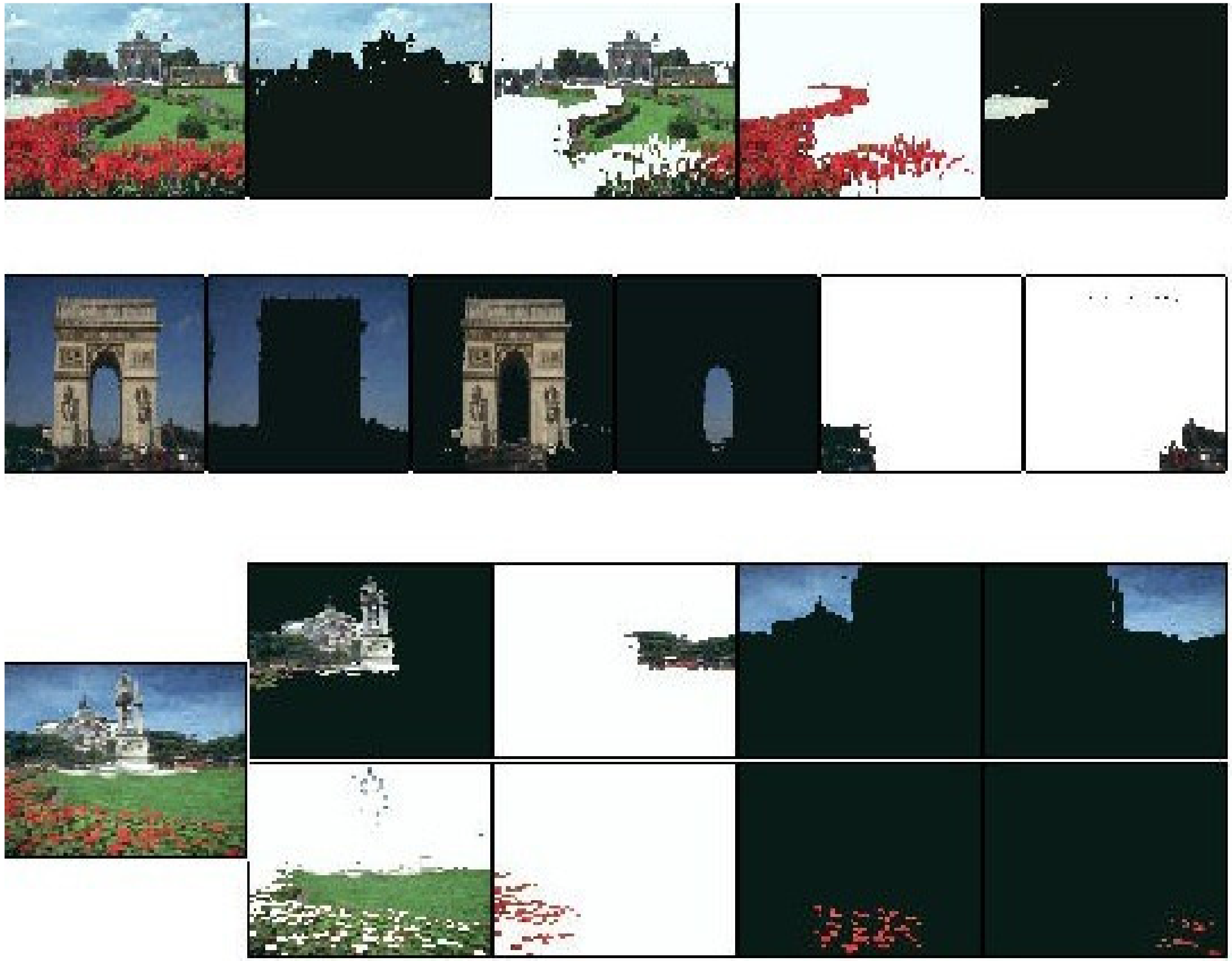
Normalized cut algorithm

1. Represent the image as a weighted graph $G = (V,E)$, compute the weight of each edge, and summarize the information in D and W

2. Solve $(D - W)y = \lambda Dy$ for the eigenvector with the second smallest eigenvalue

3. Use the entries of the eigenvector to bipartition the graph

Image segmentation by color with NCuts



Fast approximate energy minimization via graph cuts

- What does this have to do with our problem?
- Energy should encode how "bad" the solution is
- The weight of a cut can be thought of as the energy of a solution
- We will see a rigorous proof of this connection
- …But what exactly *is* a solution?

Labelings and Partitions

- A solution $f\colon \mathcal{P} \to \mathcal{L}$ is a function which gives each pixel in the picture some label from the set $\mathcal{L}$
- Labels could be $\{background, foreground\}$, $\{6,7,8,9\}$ and so on
- Any labeling of pixels in a photo defines a partition of the photo into different parts, and vice versa

Energy functions

- In general, the "badness" of a solution can be split into two parts:
  - How bad is it that this red pixel is a cat?
  - How bad is it that these two very different pixels are both a cat?

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

$$E(f) = \sum_{\{p,q\}\in\mathcal{N}} V\left(f_p, f_q\right) + \sum_{p\in\mathcal{P}} D_p\left(f_p\right)$$

unlikely

Energy Functions – Cont'

- Our energy functions will be constrained by V being metric:

$$V(\alpha, \beta) = 0 \Leftrightarrow \alpha = \beta$$

$$V(\alpha, \beta) = V(\beta, \alpha) \geq 0$$

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

- This will come in handy later when we consider paths on graphs

- For example: $V(\alpha, \beta) = 15 \cdot \min\{3, |f_p - f_q|\}$

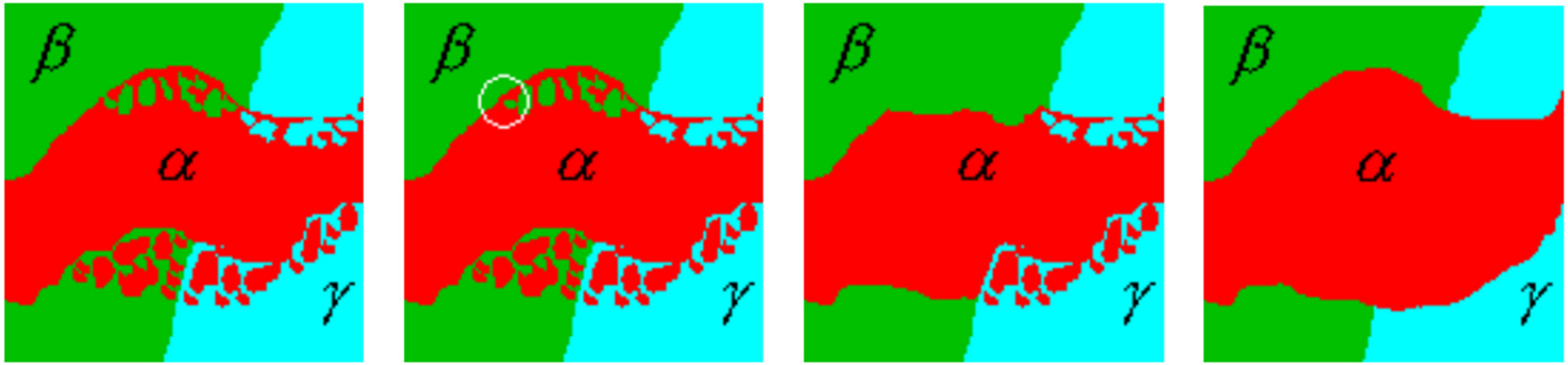
Local minima and "movesets"

- Since we are in a discrete setting, local minima can be defined discretely:
- A labeling f is a local minima if any small change increases its energy

$$E(f) \leq E(f') \; for \; any \; f' \; near \; to \; f$$

- A close labeling is one which we can arrive at with only one move

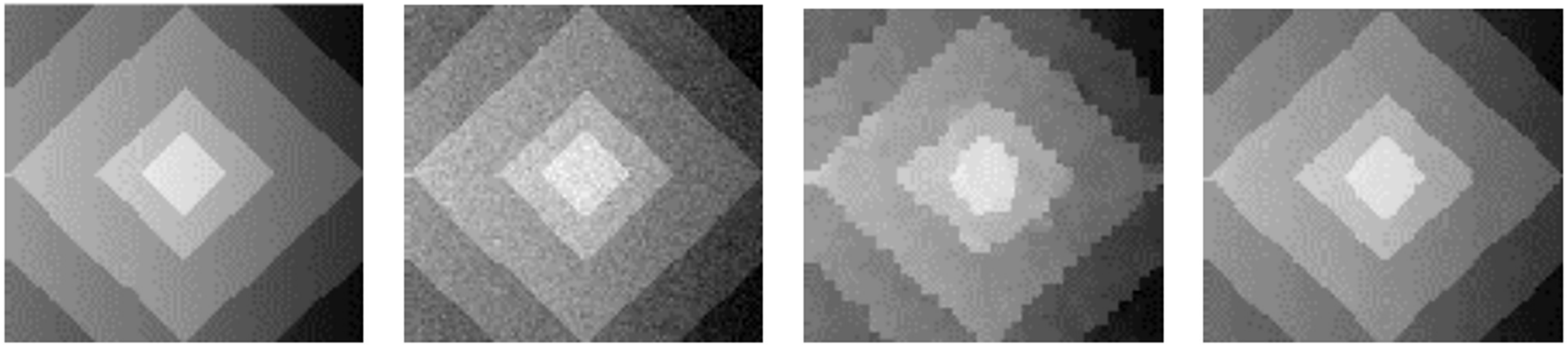
Movesets - examples



(a) initial labeling (b) standard move (c) α-β-swap (d) α-expansion

- $\mathcal{L} = \{\alpha, \beta, \gamma\}$
- Each of the 3 rightmost figures shows a move achievable by some moveset

Movesets - examples



(a) original image (b) observed image (c) local min w.r.t. standard moves (d) local min w.r.t. α-expansion moves

- Example of labelings which are local minima w.r.t different movesets
- Fig (d) looks better because it's labeling "competes" with more possibilities

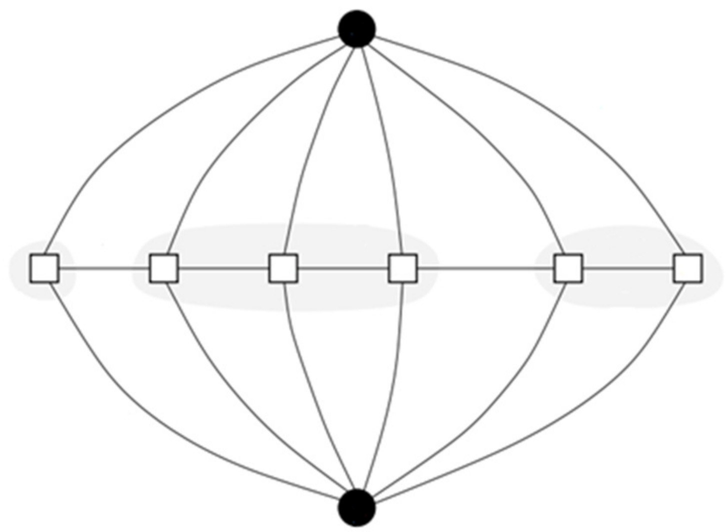
Pseudocode and terminology

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

- Each execution of 3.1 – 3.2 is called an **iteration**
- Each execution of 2 – 4 is called a **cycle**
- A similar algorithm exists for minimization w.r.t α-expansion, but we will not focus on it

Finding the optimal swap move

- We would like to show an efficient way to compute $\hat{f}$ as defined in 3.1
- To do this, we will use graph cuts
- Example graph:



Graph Cuts - Revisited

- We say that a set of edges $C \subseteq E$ is a cut of G if it the two terminals are separated from each other in the induced graph $\langle V, E - C \rangle$

- We will also require that no proper subset of C separates the two terminals, for reasons you will see later

- The cost of a cut C will be defined as the sum of the edge weights of the cut

Building the graph

- Vertices of the graph are all pixels labeled α or β, plus the two terminals:

$$V_{\alpha\beta} = \mathcal{P}_\alpha \cup \mathcal{P}_\beta \cup \{\alpha, \beta\}$$

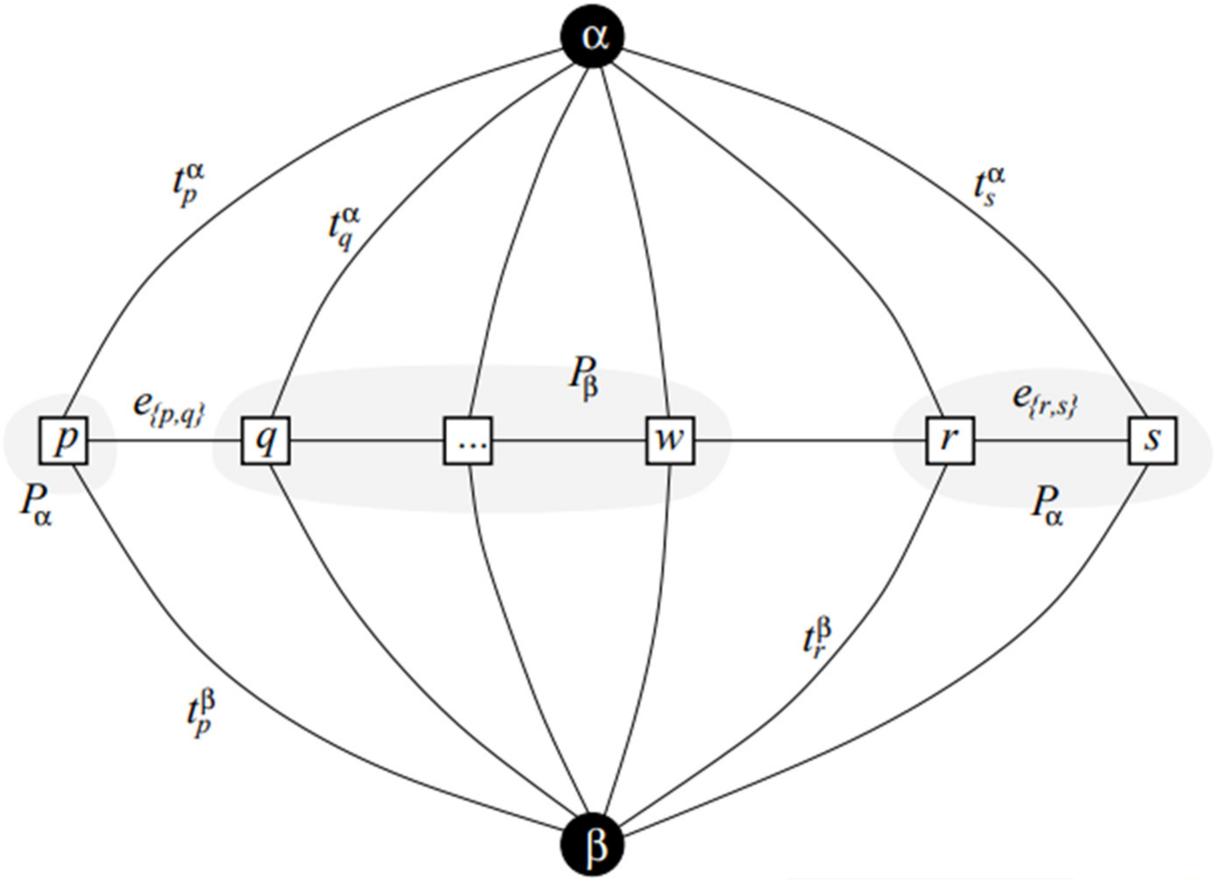
- Each pixel is connected to its neighbors in the picture and the terminals

- Edges between pixels are n-links
- Edges between pixels and terminals are t-links:

$$E_{\alpha\beta} = \bigcup_{\substack{\{p,q\}\in\mathcal{N} \\ p,q\in V_{\alpha\beta}}} \{\{p,q\}\} \bigcup_{p\in\mathcal{P}_\alpha\cup\mathcal{P}_\beta} \{\{p,\alpha\},\{p,\beta\}\}$$

Example graph again:

$$G_{\alpha\beta} = \langle V_{\alpha\beta}, E_{\alpha\beta} \rangle$$



Defining the weights

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

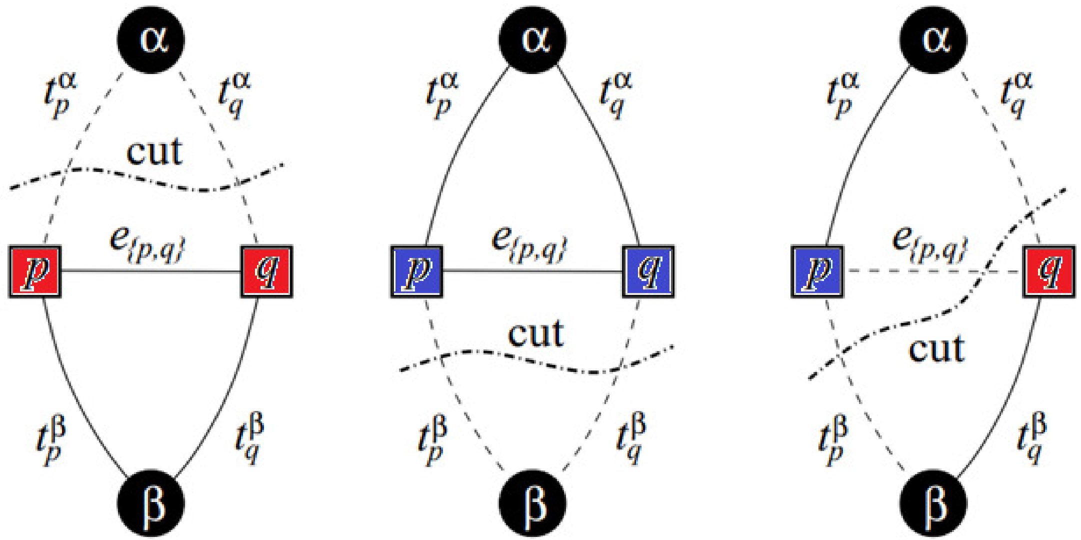
Big scary table – Cont’

- The weight of a t-link $\{\alpha, p\}$ is the cost of assigning α to p
- The weight of an n-link is the cost of having a boundary in the partition between the two pixels

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

Graph weights - explanation

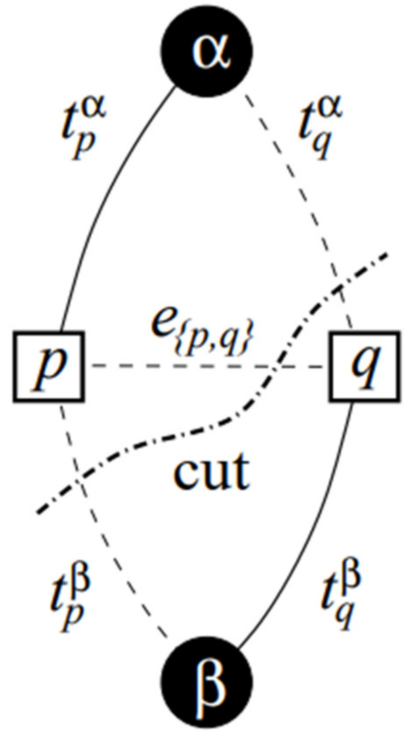
- An n-link appears in the cut **only if** its two endpoints are assigned different labels:



What labeling does a cut define?

- Let f_p^C denote the label given to pixel p by cut C

- If $p \notin \mathcal{P}_{\alpha\beta}$, then $f_p^C = f_p$
- If $t_p^\alpha \in C$ then $f_p^C = \alpha$
- If $t_p^\beta \in C$ then $f_p^C = \beta$



- since C is a cut, no vertex is **reachable from both** terminals, and no vertex is **isolated from both** either

Main theorem

$$|C| = E(f^c) - K$$

- For any α-β cut on $G_{\alpha\beta}$
- Specifically, the **minimum cut** is the **minimal energy labeling** one α-β swap away from the initial f
- Why?

Main theorem - Proof

$$|C \cap t - links| = \sum_{p \in \mathcal{P}_{\alpha\beta}} \left(D_p\left(f_p^C\right) + \sum_{\substack{q \in \mathcal{N}_p \\ q \notin \mathcal{P}_{\alpha\beta}}} V\left(f_p^C, f_q^C\right) \right)$$

- Since the weight of a t-link is defined accordingly:

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

Main theorem - Proof

$$|C \cap n - links| = \sum_{\substack{\{p,q\} \in \mathcal{N} \\ p,q \in \mathcal{P}_{\alpha\beta}}} V\left(f_p^C, f_q^C\right)$$

- Note that since V is metric, then if $f_p^C = f_q^C$, there is no boundary, the edge is not in the cut, and $V\left(f_p^C, f_q^C\right) = 0$

$e_{\{p,q\}}$	$V(\alpha, \beta)$	$\{p,q\} \in \mathcal{N}$ $p,q \in \mathcal{P}_{\alpha\beta}$

Main theorem - Proof

- Finally, putting it all together:

$$|C| = \sum_{p \in \mathcal{P}_{\alpha\beta}} \left(D_p\left(f_p^C\right) + \sum_{\substack{q \in \mathcal{N}_p \\ q \notin \mathcal{P}_{\alpha\beta}}} V\left(f_p^C, f_q^C\right) \right) + \sum_{\substack{\{p,q\} \in \mathcal{N} \\ p,q \in \mathcal{P}_{\alpha\beta}}} V\left(f_p^C, f_q^C\right)$$

$$|C| = \sum_{p \in \mathcal{P}_{\alpha\beta}} D_p\left(f_p^C\right) + \sum_{\substack{\{p,q\} \in \mathcal{N} \\ p \text{ or } q \in \mathcal{P}_{\alpha\beta}}} V\left(f_p^C, f_q^C\right)$$

Proof – Cont'

$$|C| = \sum_{p \in \mathcal{P}_{\alpha\beta}} D_p\left(f_p^C\right) + \sum_{\substack{\{p,q\} \in \mathcal{N} \\ p \text{ or } q \in \mathcal{P}_{\alpha\beta}}} V\left(f_p^C, f_q^C\right)$$

$$|C| = E(f^C) - \sum_{p \notin \mathcal{P}_{\alpha\beta}} D_p\left(f_p^C\right) - \sum_{\substack{\{p,q\} \in \mathcal{N} \\ p,q \notin \mathcal{P}_{\alpha\beta}}} V\left(f_p^C, f_q^C\right)$$

$$|C| = E(f^C) - K$$

Main proof - Corollary

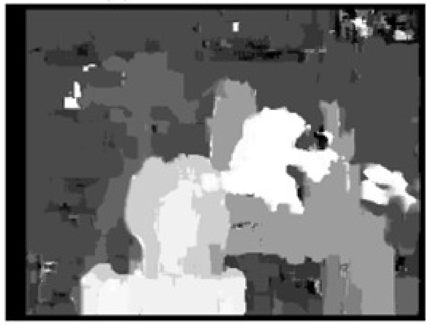
- The minimal weight labeling one α-β swap away from f is defined by f^C where C is the minimal weight cut on $G_{\alpha\beta}$
- To implement the algorithm, we can repeatedly apply minimum cut operations until the labeling stops changing, something which we know how to do efficiently
- Enough math, let's see some results!

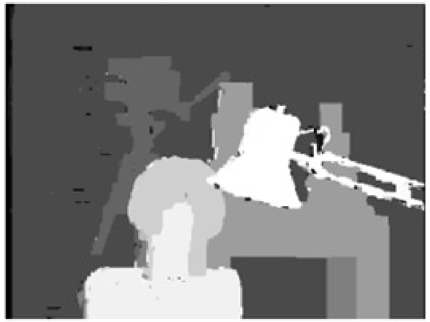
Results - Stereo

Left Image

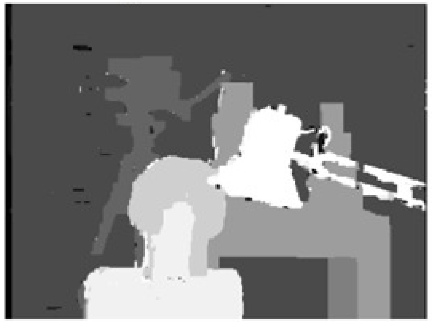
Ground truth

Normalized Correlation

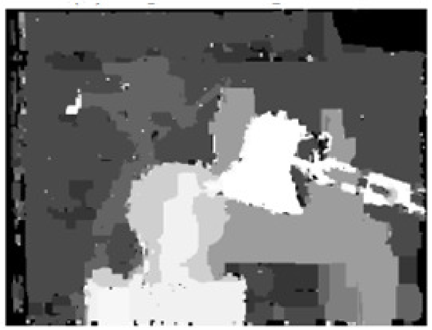




Swap algorithm



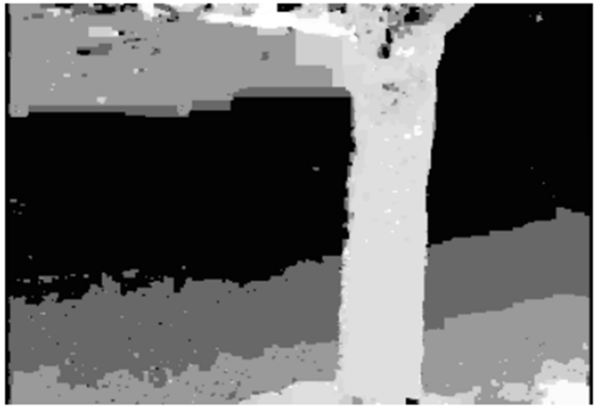
Expansion algorithm



Simulated annealing

Results – Flower garden sequence

Stereo pair



Horizontal movement

- $V(f_p, f_q) = T(f_p \neq f_q) \cdot 80$
- $D_p(f_p)$... Its complicated 😞

Results – Moving cat

Moving cat

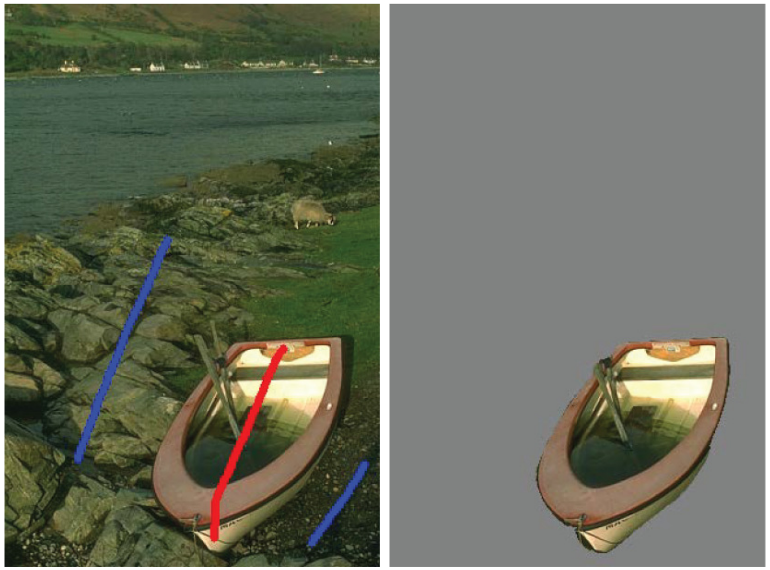
Horizontal movement

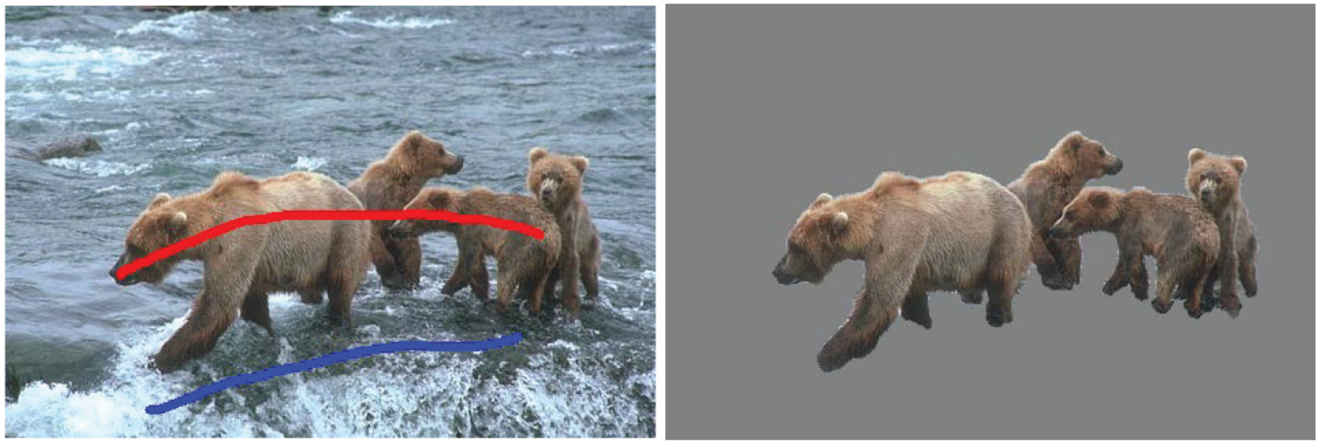
Vertical movement

- $V\left(f_p, f_q\right) = 40 \cdot \min\left\{8, \left(f_p^h - f_q^h\right)^2 + \left(f_p^v - f_q^v\right)^2\right\}$

- f_p^h and f_p^v are horizontal and vertical components of f_p

Interactive Graph Cuts …





Interactive Graph Cuts

- We would like the ability to impose hard constraints.

- The user marks certain pixels as "object" or "background" to provide hard constraints for segmentation.

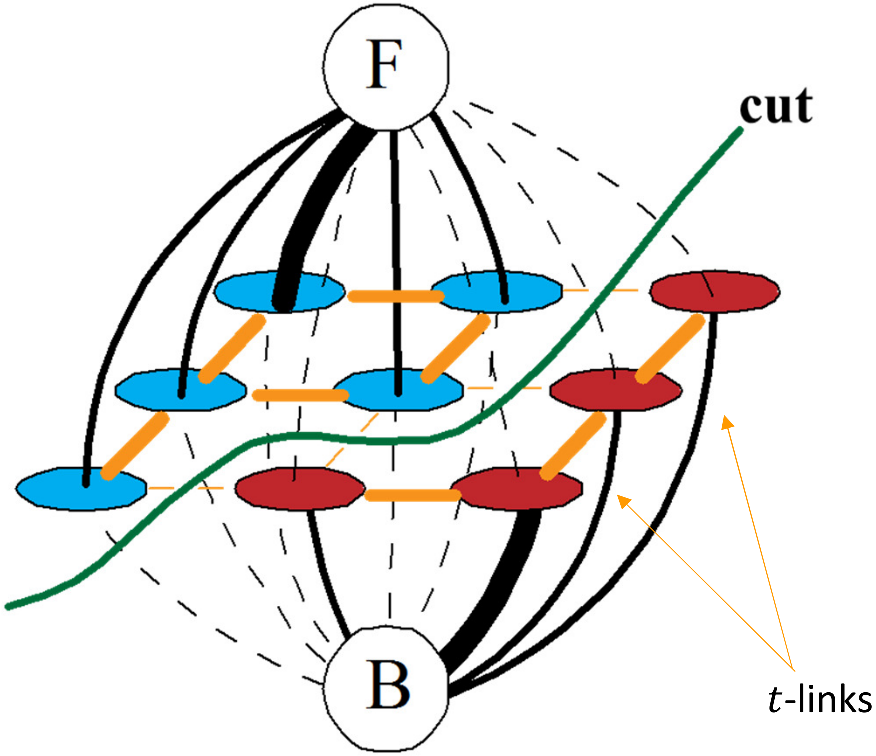
- ***red*** is object
  Blue is background

(a)

(b)

We can use the same system as earlier

- To impose hard constraints, we can force very heavy t-links



Heavy $\boldsymbol{t}$-links force choice of label

- Remember from the last part – this time we have only one possible α-β swap.

- If we add a very heavy weight to $\{p, \alpha\}$, it will not be in the cut

- If $\{p, \alpha\} \notin C$, then $f_p^C = \beta$

Summary

- Large intersection between vision and graph theory

- Today we saw several of the algorithms for segmentation
  - Normalized graph cuts
  - Graph cuts minimizing large moves

- We also saw a solution for some adjacent problems like
  - Motion segmentation

References

Boykov, Yuri, Olga Veksler, and Ramin Zabih. "Fast approximate energy minimization via graph cuts." *IEEE Transactions on pattern analysis and machine intelligence* 23.11 (2001): 1222-1239.

Shi, Jianbo, and Jitendra Malik. "Normalized cuts and image segmentation." *IEEE Transactions on pattern analysis and machine intelligence* 22.8 (2000): 888-905.

Boykov, Yuri Y., and M-P. Jolly. "Interactive graph cuts for optimal boundary & region segmentation of objects in ND images." *Proceedings eighth IEEE international conference on computer vision. ICCV 2001*. Vol. 1. IEEE, 2001.

Summary

- Large intersection between vision and graph theory

- Today we saw several of the algorithms for segmentation
  - Normalized graph cuts
  - Graph cuts minimizing large moves

- We also saw a solution for some adjacent problems like
  - Motion segmentation

Thanks for listening!