Correlation Clustering: from Theory to Practice

Francesco Bonchi
Yahoo Labs, Barcelona

David Garcia-Soriano
Yahoo Labs, Barcelona

Edo Liberty
Yahoo Labs, NYC

Plan of the talk

- Part 1: Introduction and fundamental results
  - Clustering: from the Euclidean setting to the graph setting
  - Correlation clustering: motivations and basic definitions,
  - Fundamental results
  - The Pivot Algorithm

- Part 2: Correlation clustering variants
  - Overlapping, On-line, Bipartite, Chromatic
  - Clustering aggregation

- Part 3: Scalability for real-world instances
  - Real-world application examples
  - Scalable implementation
  - Local correlation clustering

YAHOO!

Part I:
Introduction and fundamental results

Edo Liberty
Yahoo Labs, NYC

Clustering, in general

Partition a set of objects such that “similar” objects are grouped together and “dissimilar” objects are set apart.

Setting

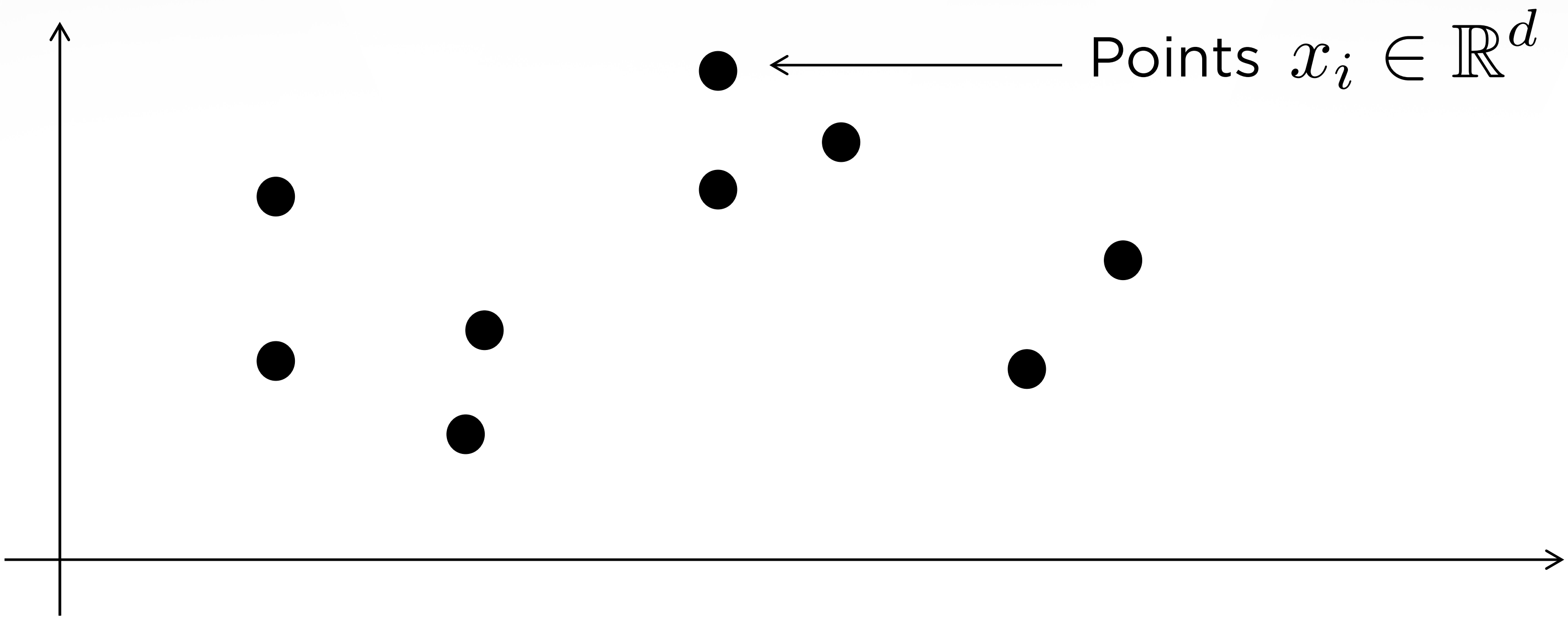
⇓

Objective function

⇓

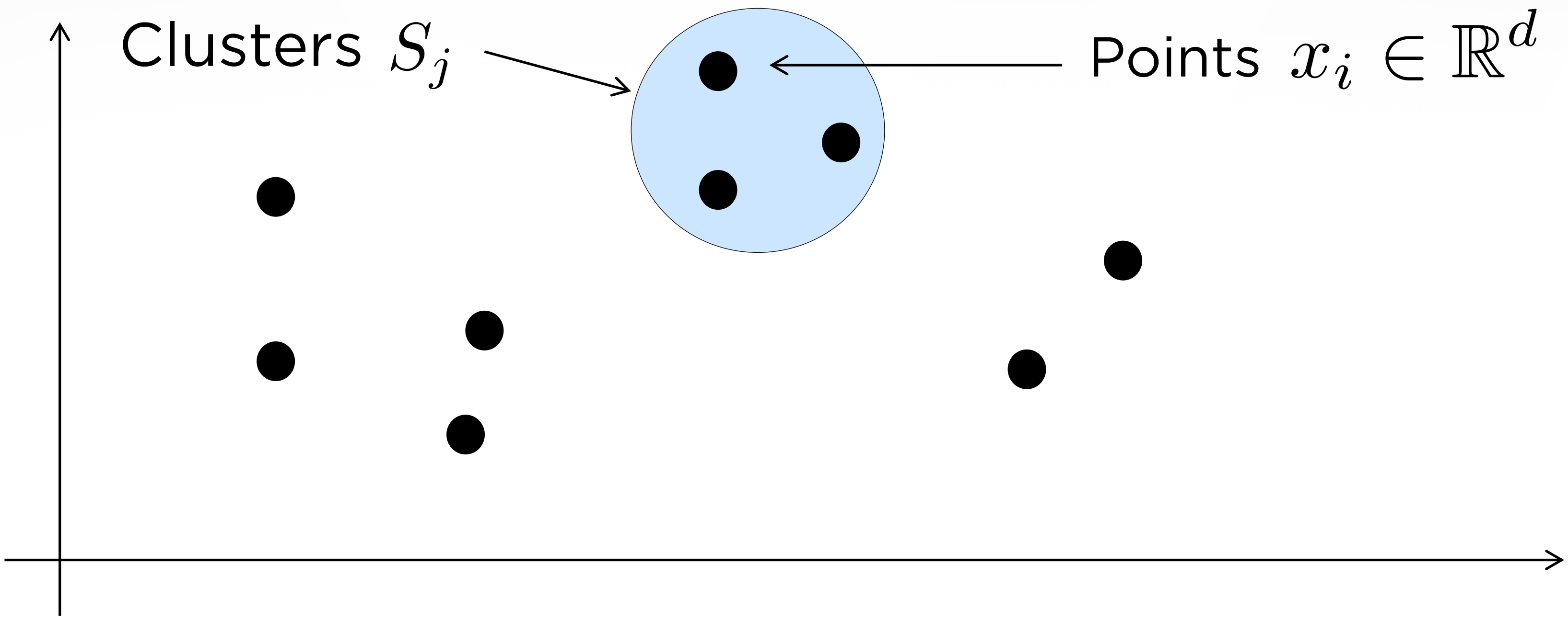
Algorithm

Euclidean Setting



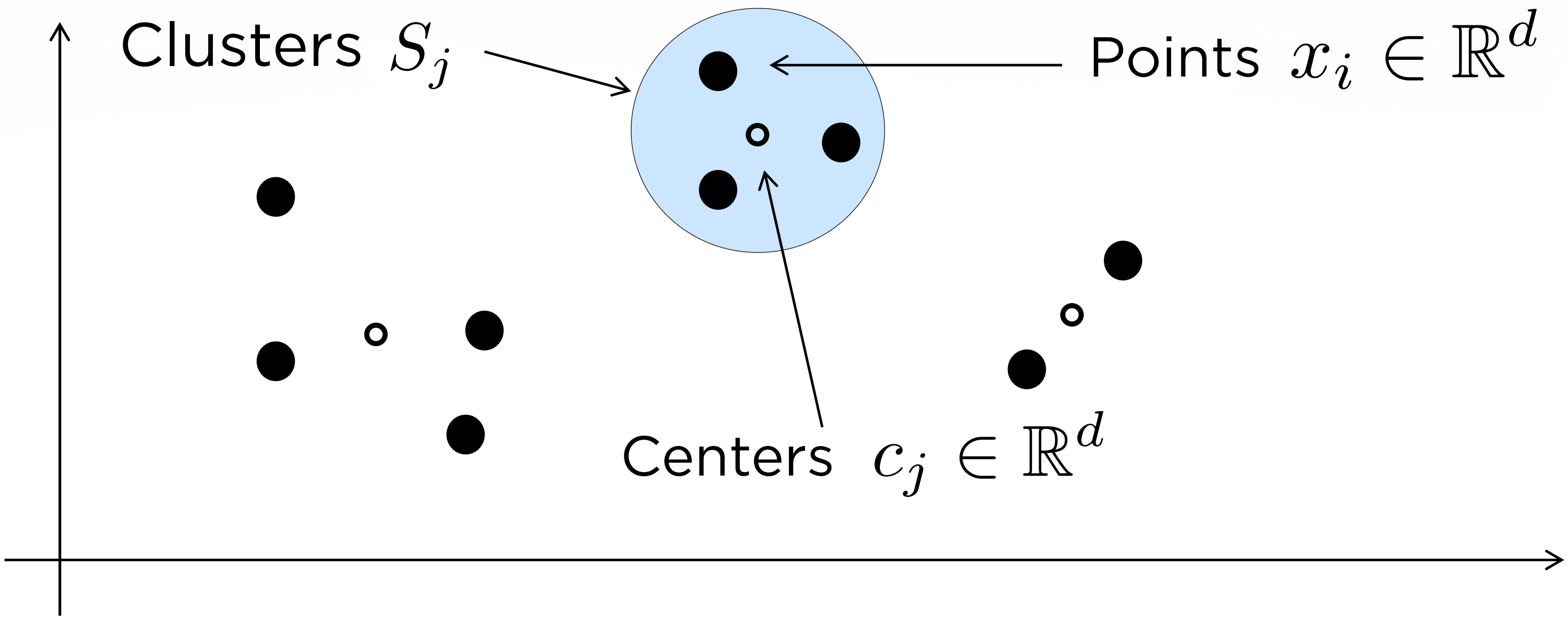
Small $\|x_i - x_j\|$ indicates the two points are “similar”

Euclidean Setting



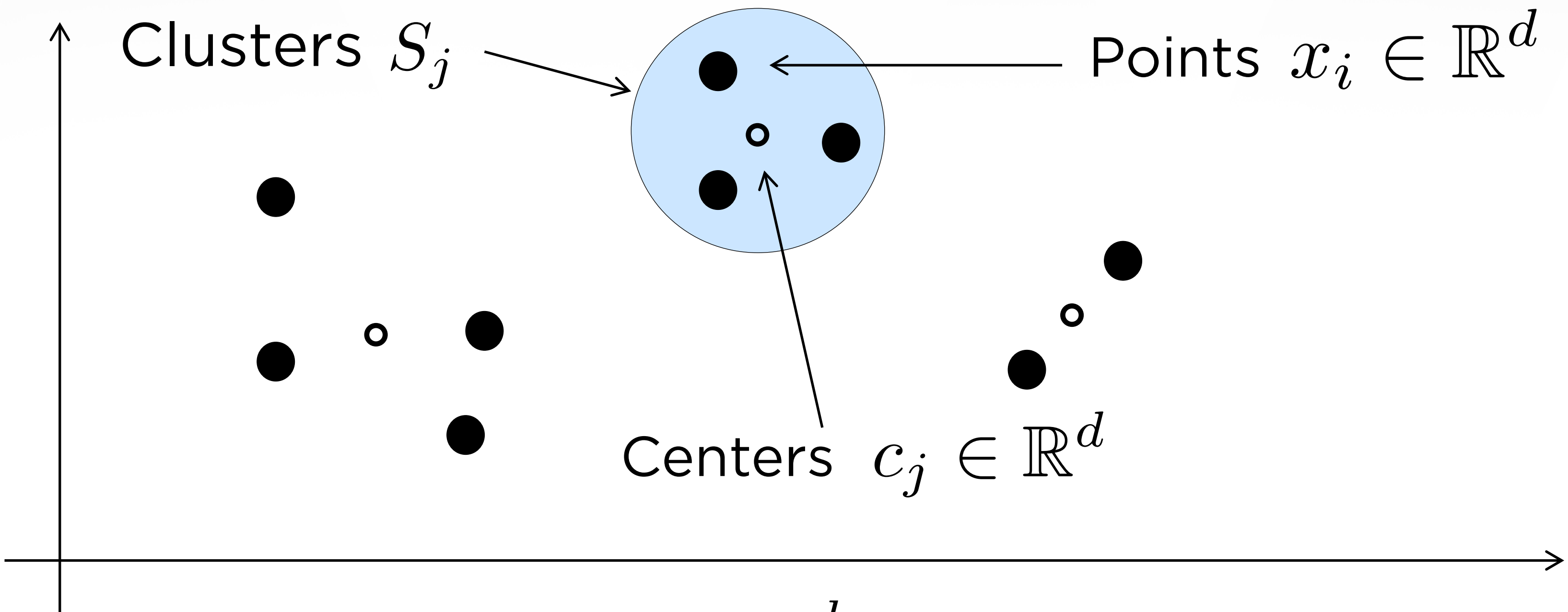
A cluster is a set of points

Euclidean Setting



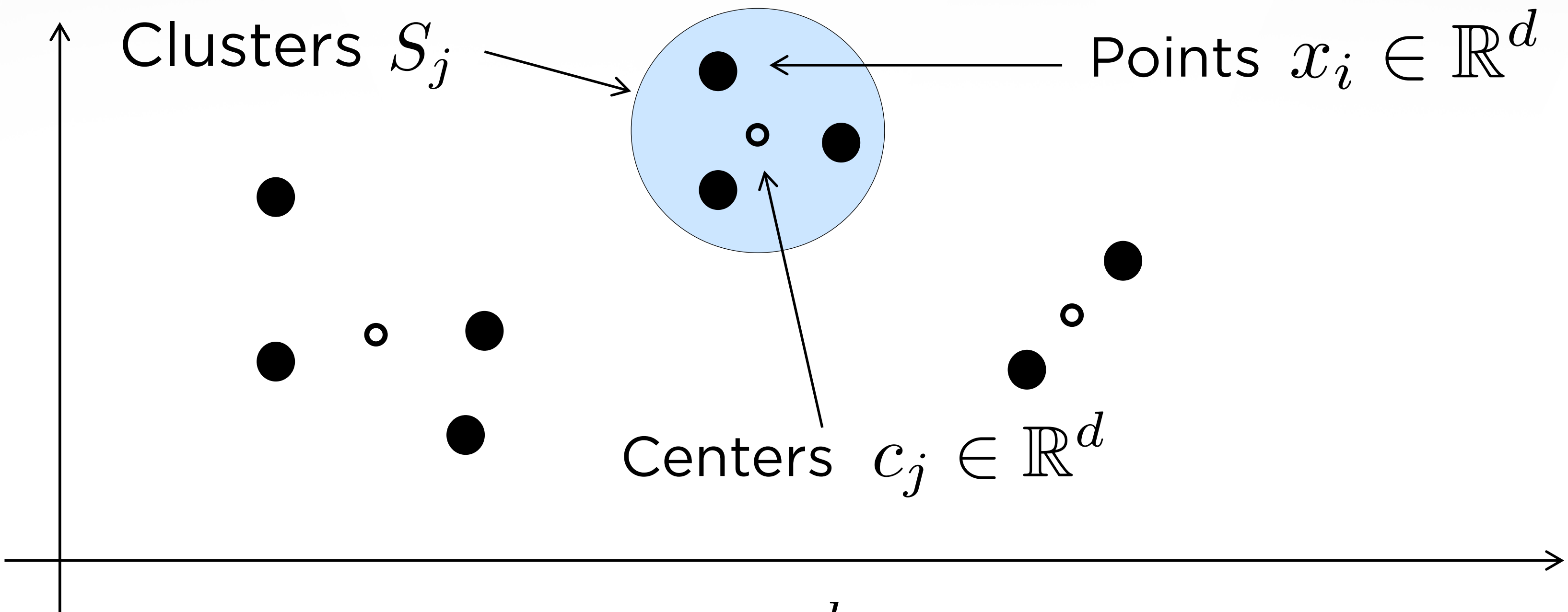
Each cluster has a cluster center

Euclidean objectives



K-means objective $$\sum_{j=1}^{k} \sum_{i \in S_j} \|x_i - c_j\|_2^2$$

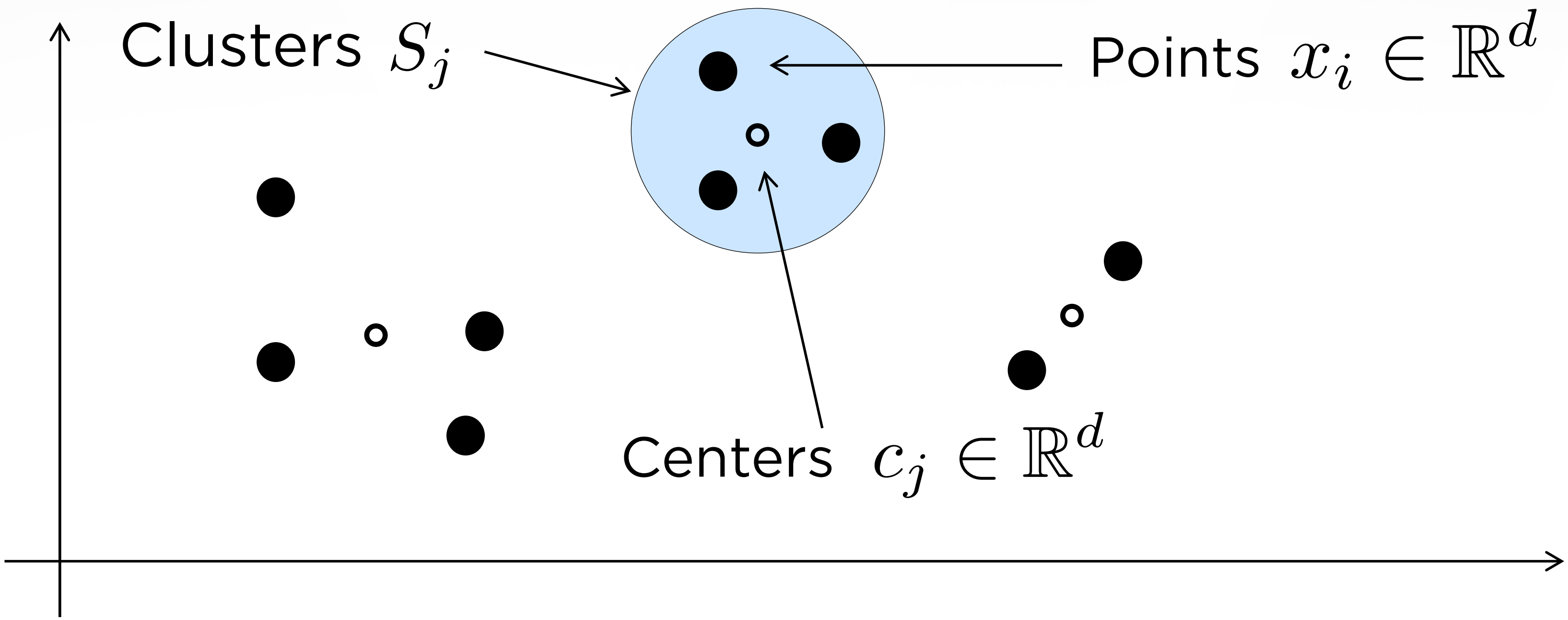
Euclidean objectives



K-median objective $$\sum_{j=1}^{k} \sum_{i \in S_j} \|x_i - c_j\|_2$$

YAHOO!

Euclidean objectives

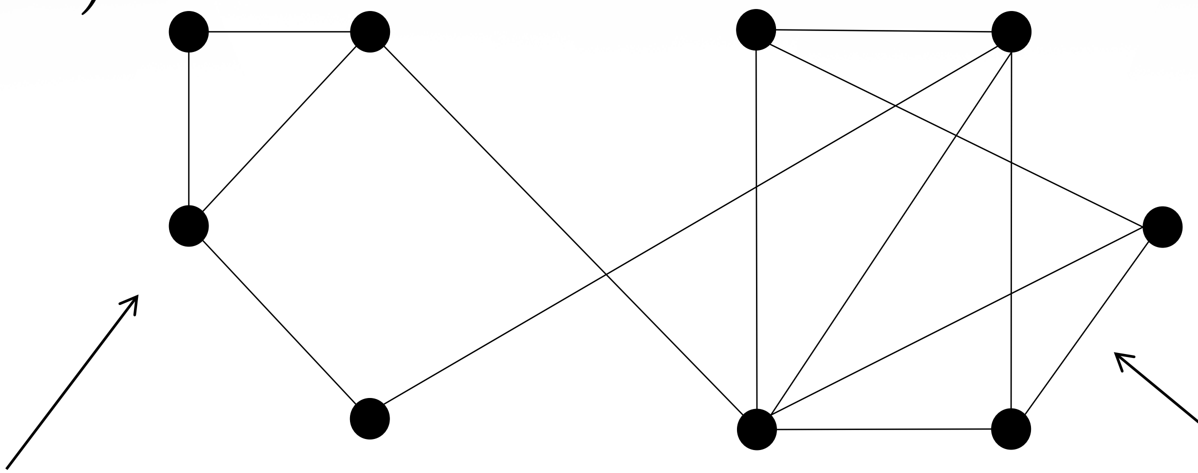


K-centers objective $\max_i \min_j \|x_i - c_j\|_2$

YAHOO!

Graph setting

$$G(V, E)$$

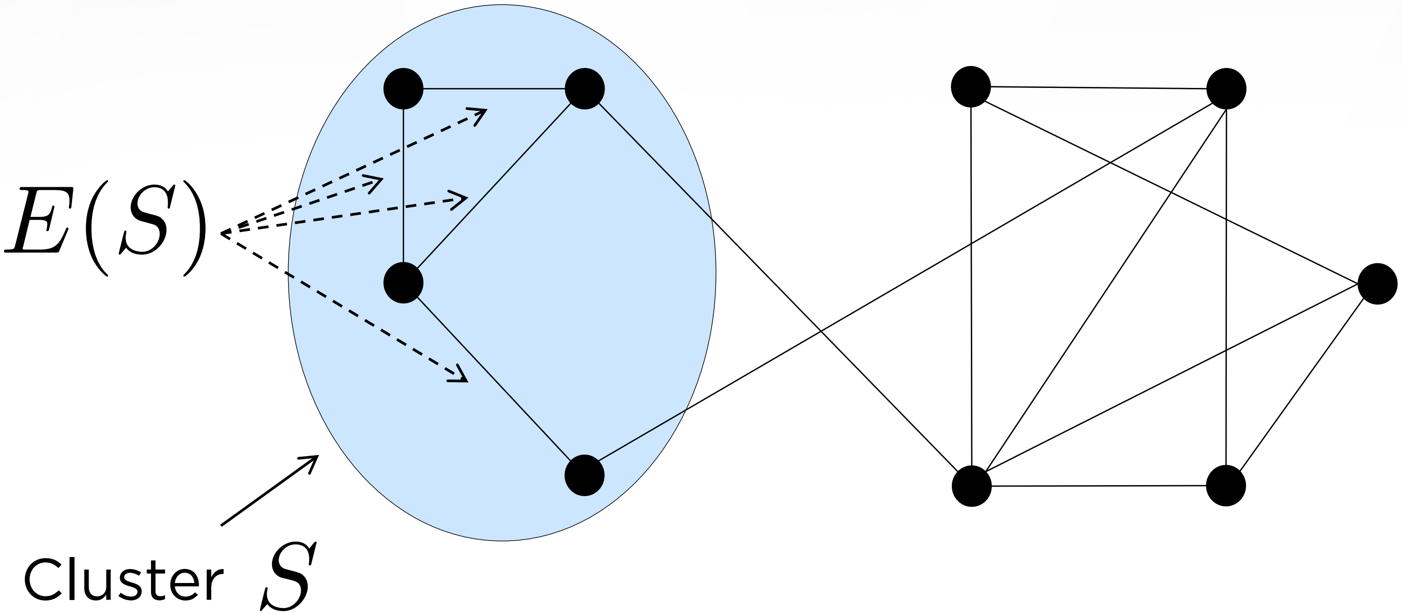


Nodes $v_i \in V$

Edges $e_{ij} \in E$

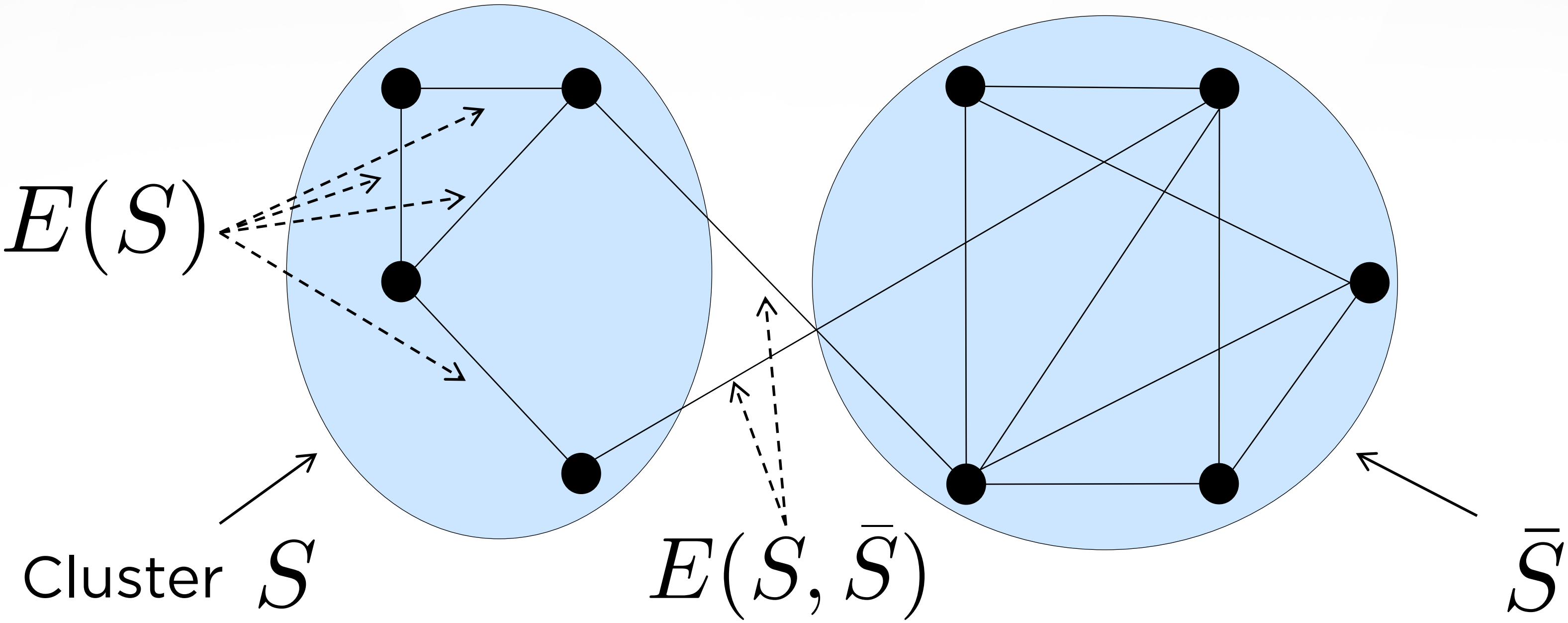
$e_{ij} \in E$ means the two nodes are “similar”

Graph setting



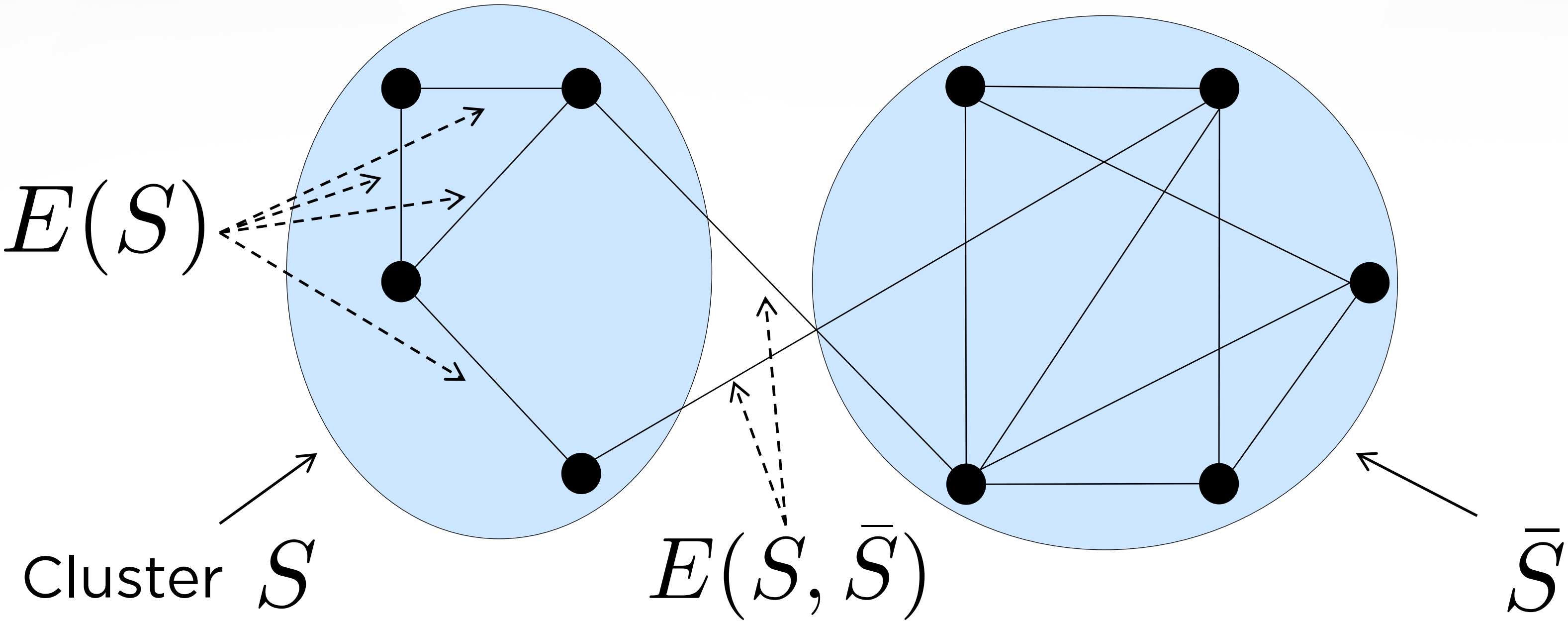
$e_{ij} \in E$ means the two nodes are "similar"

Graph setting



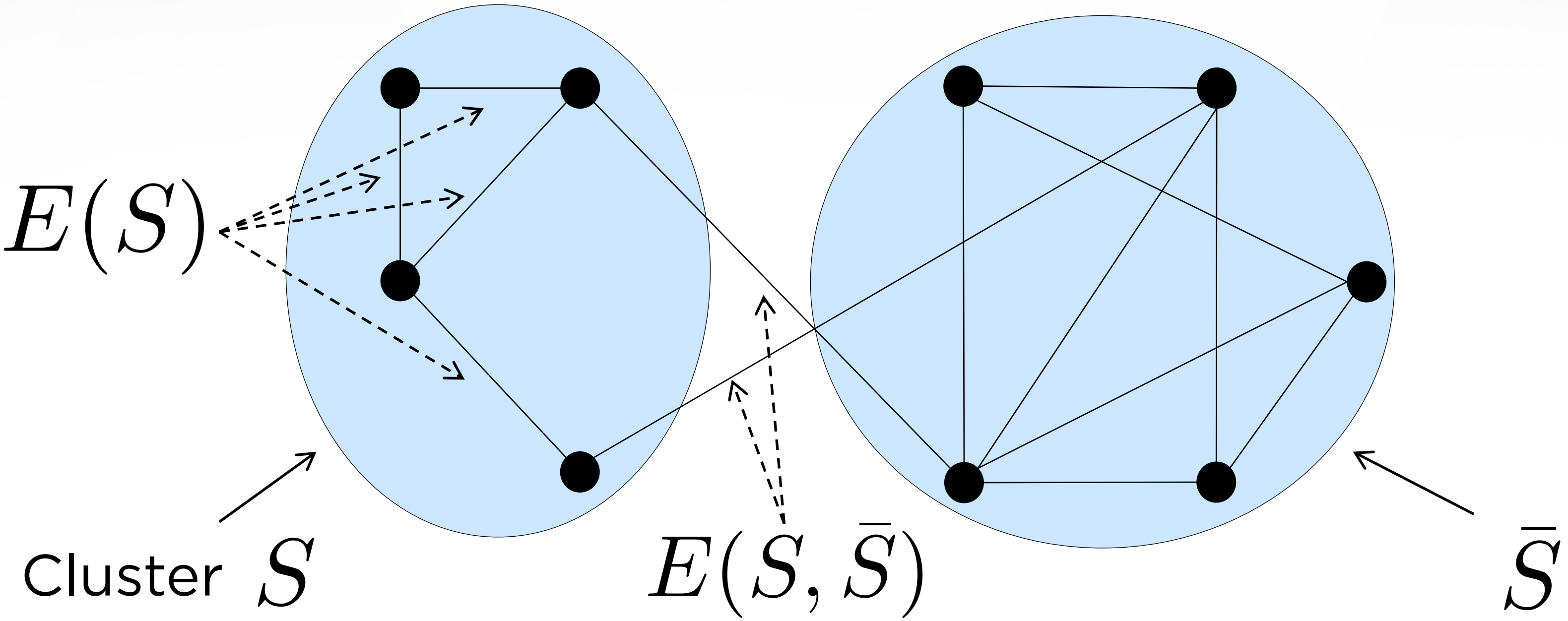
We want S and $\bar{S}$ large and $E(S, \bar{S})$ small

Graph objectives



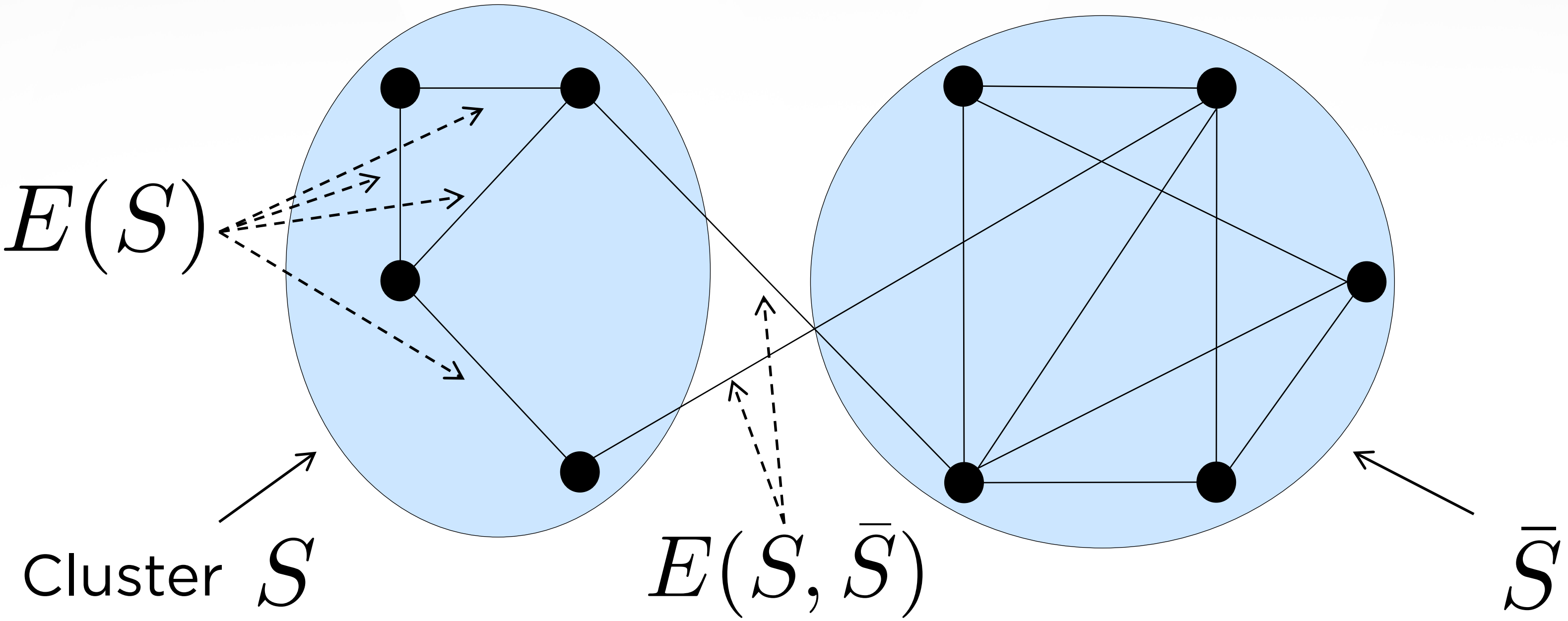
Sparsest cut objective $$\frac{|E(S,\bar{S})|}{|S| \cdot |\bar{S}|}$$

Graph objectives



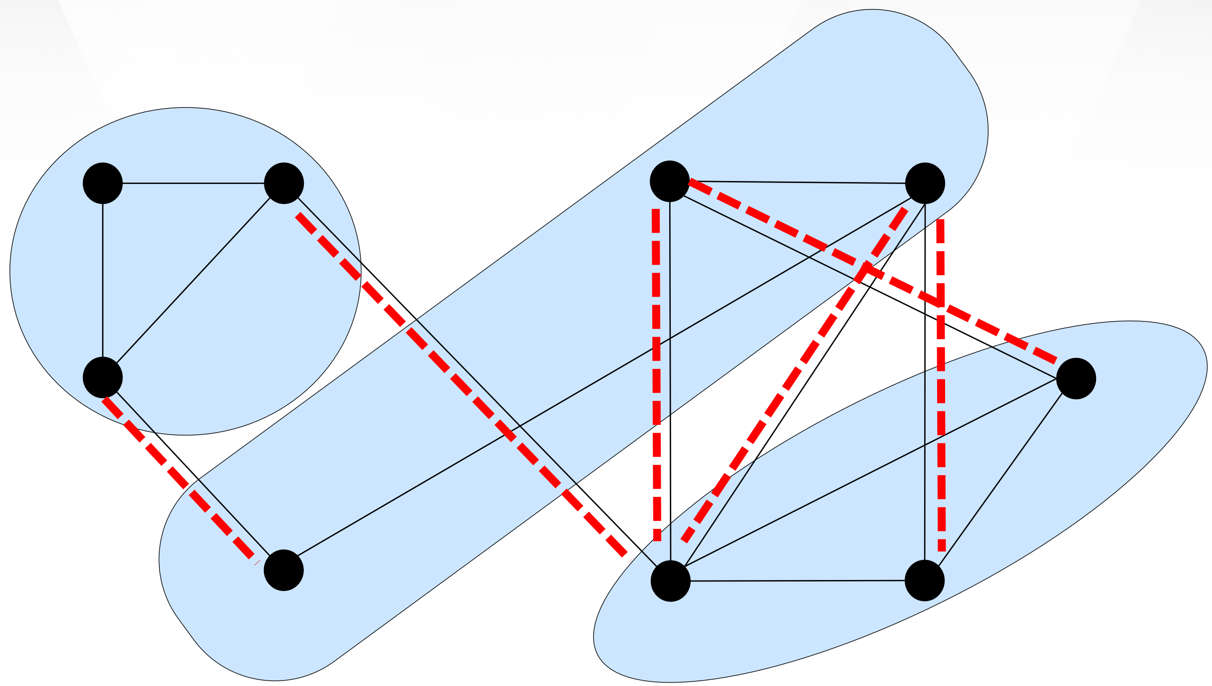
Edge expansion $\dfrac{|E(S, \bar{S})|}{|S|}$ s.t. $|S| \leq \dfrac{|V|}{2}$

Graph objectives



Graph Conductance $\dfrac{|E(S,\bar{S})|}{|E(S)|}$ s.t. $|E(S)| \leq \dfrac{|E|}{2}$

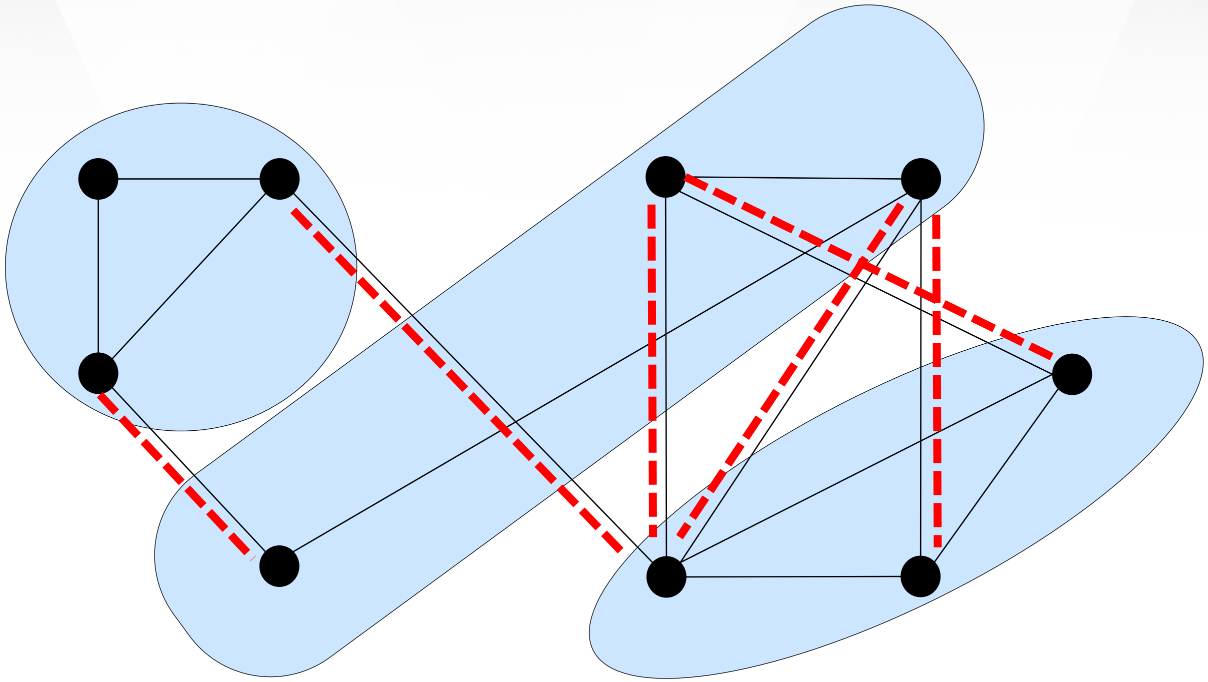
Graph objectives



k-balanced partitioning $E(S_1, \ldots S_k)$

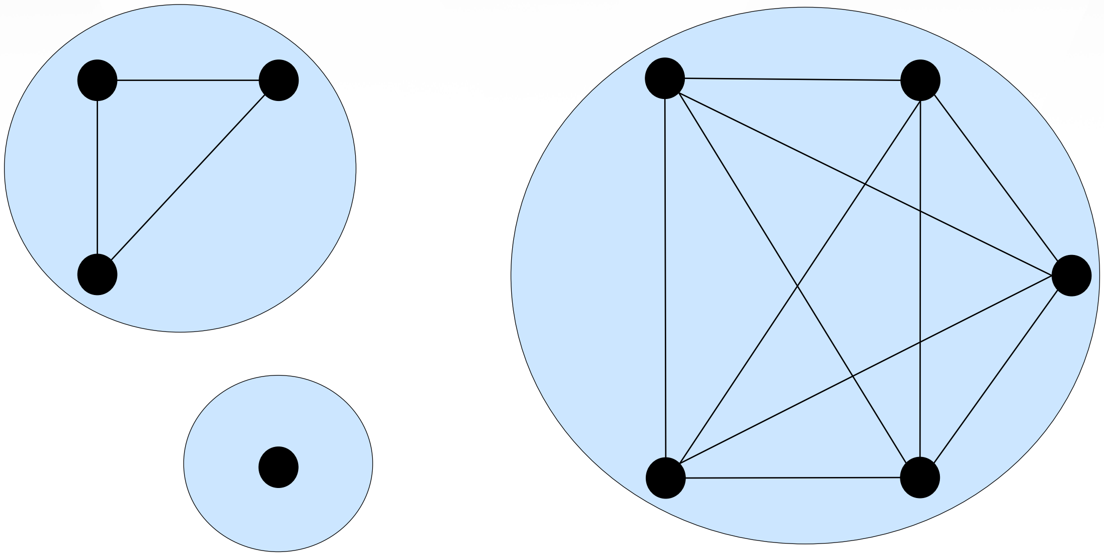
Where $|S_i| \approx n/k$

Graph objectives



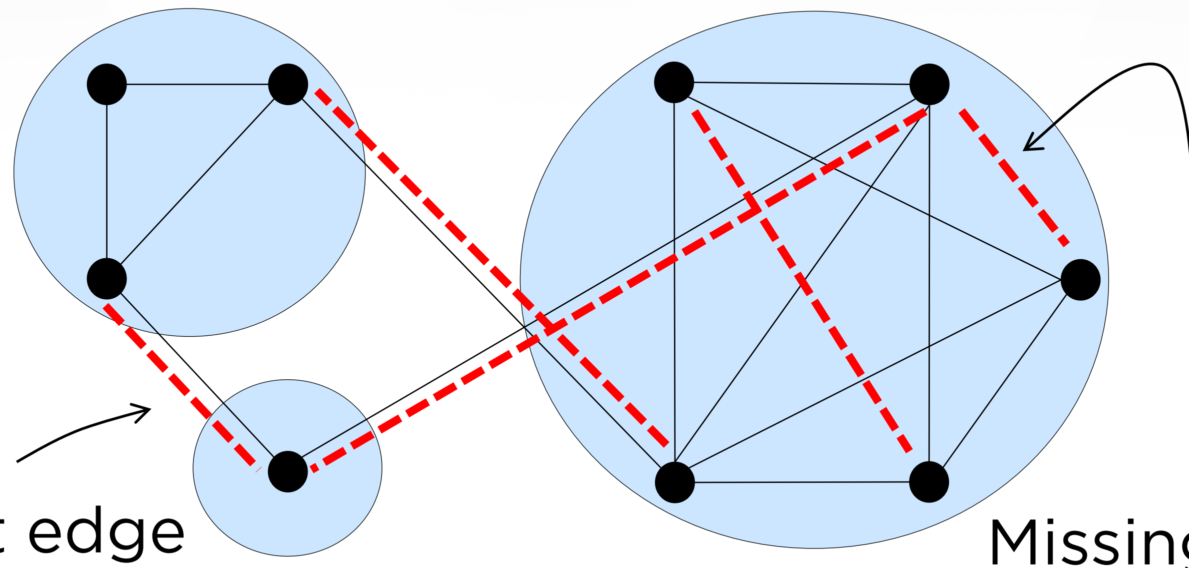
Multi-way spectral partitioning $$\max_{i \in 1,\ldots,k} \frac{E(S_i, \bar{S}_i)}{E(S_i)}$$

Correlation Clustering objective



Let C be a collection of cliques (clusters).

Correlation Clustering objective



Redundant edge

$$e \in E,\ e \notin C$$

Missing edge

$$e \notin E,\ e \in C$$

Find the clustering that correlates
the most with the input graph

4 Basic variants

	Unweighted	Weighted
Min-disagree	$\min_C \sum_{i,j} C_{ij}(1 - E_{ij}) + \sum_{i,j}(1 - C_{ij})E_{ij}$	$\min_C \sum_{i,j} w_{ij}C_{ij}(1 - E_{ij}) + \sum_{i,j} w_{ij}(1 - C_{ij})E_{ij}$
Max-agree	$\max_C \sum_{i,j} C_{ij}E_{ij} + \sum_{i,j}(1 - C_{ij})(1 - E_{ij})$	$\max_C \sum_{i,j} w_{ij}C_{ij}E_{ij} + \sum_{i,j} w_{ij}(1 - C_{ij})(1 - E_{ij})$

4 Basic variants

	Unweighted	Weighted
Min-disagree	$\min_C \sum_{i,j} C_{ij}(1-E_{ij}) + \sum_{i,j}(1-C_{ij})E_{ij}$	$\min_C \sum_{i,j} w_{ij}C_{ij}(1-E_{ij}) + \sum_{i,j} w_{ij}(1-C_{ij})E_{ij}$
Max-agree	$\max_C \sum_{i,j} C_{ij}E_{ij} + \sum_{i,j}(1-C_{ij})(1-E_{ij})$	$\max_C \sum_{i,j} w_{ij}C_{ij}E_{ij} + \sum_{i,j} w_{ij}(1-C_{ij})(1-E_{ij})$

Correlation Clustering objective

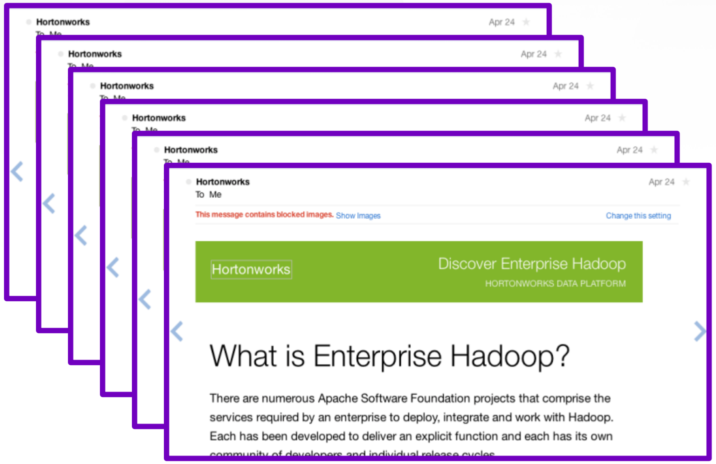
Important points to notice

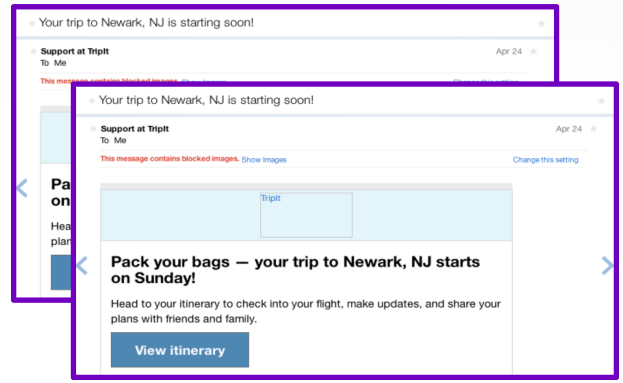
- There is no limitation on the number of clusters
- and no limitation on their sizes

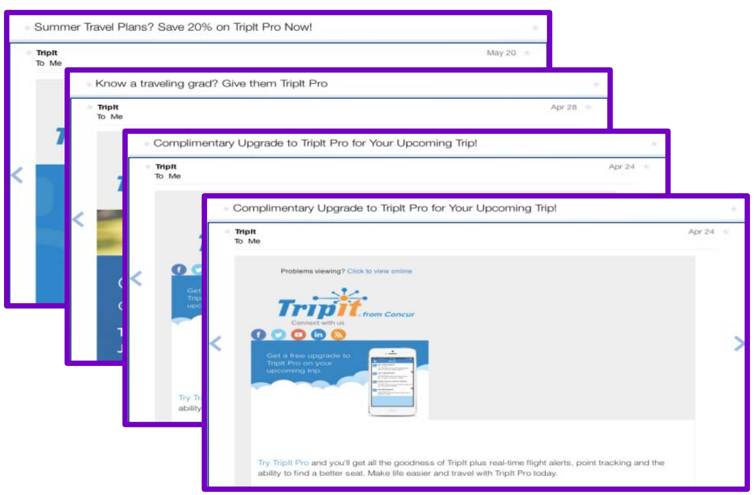
For example: the best solution could be

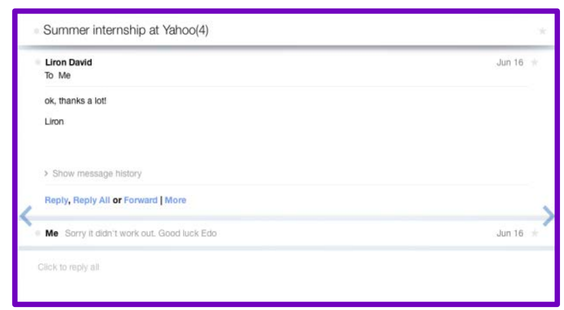
- 1 giant cluster
- n singletons

Document de-duplication









Document de-duplication

Thanks for your order, Martin Flimberton!

Want to manage your order online?
If you need to check the status of your order or make changes, please visit our home page at Amazon.com and click on

Purchasing Information:

E-mail Address: lostiddude@yahoo.com

Billing Address:
Martin Flimberton
1434 Main Street Road
Glenbert lows, Illinois 60121
United States

Order Grand Total: $53.99

Get the Amazon.com Rewards Visa Card and get **$30 instantly** as a

Order Summary:

Shipping Details : 8thdayconsulting

Order #:	104-3041649-8513858
Shipping Method:	Standard Shipping
Items:	$50.00
Shipping & Handling:	$3.99

Total Before Tax:	$53.99
Estimated Tax To Be Collected:*	$0.00

Order Total:	**$53.99**

Delivery estimate: Oct. 24, 2012 - Nov. 8, 2012
1 "Microsoft Office 2010: Essential (Shelly Cashman Series)"
Shelly, Gary B.; Paperback; $50.00
In Stock
Sold by: 8thdayconsulting

Preston

Thanks for your order, Preston Presterton!

Want to manage your order online?
If you need to check the status of your order or make changes, please visit our home page at Amazon.com and click on Y

Purchasing Information:

E-mail Address: mepartydj@yahoo.com

Billing Address:
Preston Presterton
259 Greenpoint DR
DALLAS, TX 75231-9126
United States

Order Grand Total: $97.41

Get the Amazon.com Rewards Visa Card and get **$30 instantly** as an Amazon.com Gift Card.

Order Summary:

Shipping Details : buybackselyria

Order #:	002-1903988-3076225
Shipping Method:	Standard Shipping
Items:	$30.68
Shipping & Handling:	$2.98

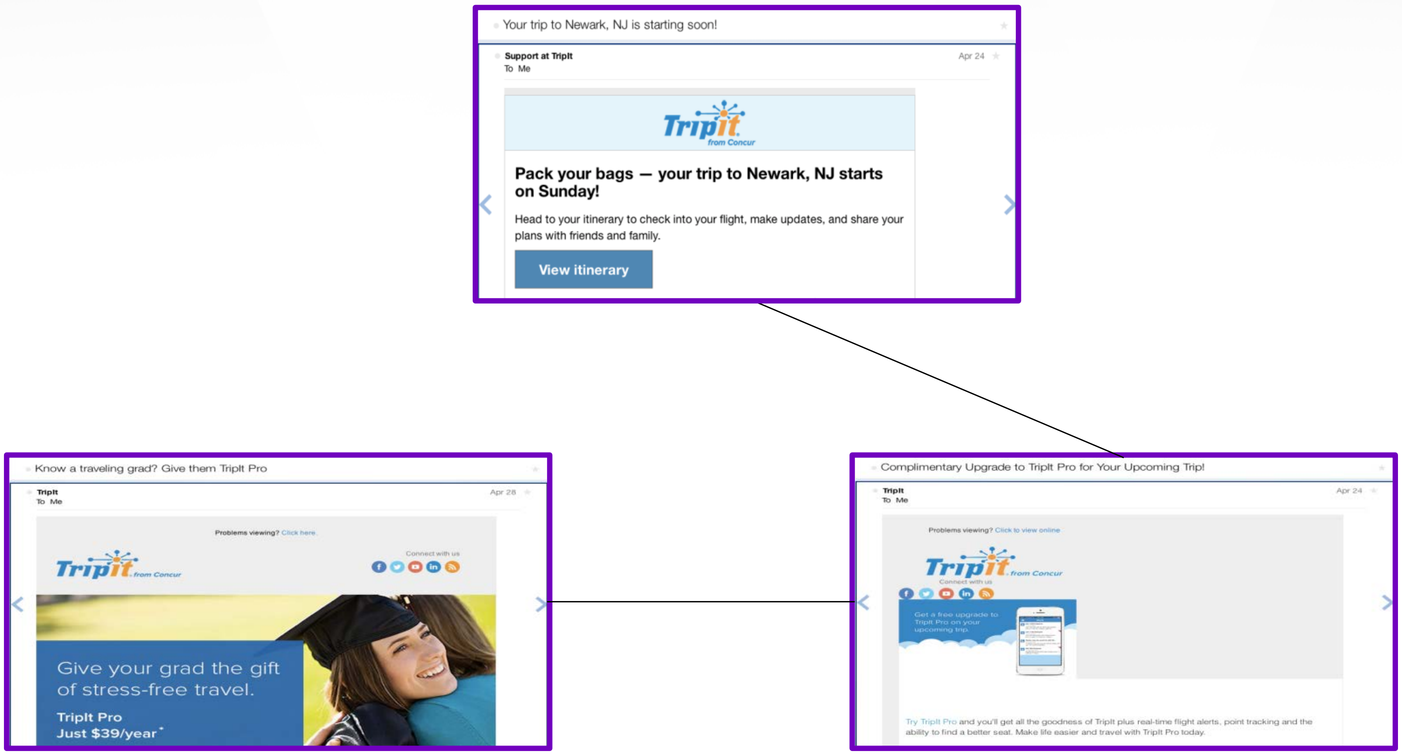
Total Before Tax:	$33.66
Estimated Tax To Be Collected:*	$0.00

Order Total:	**$33.66**

Delivery estimate: Oct. 16, 2012 - Oct. 31, 2012
1 "Fawlty Towers: The Complete Collection Remastered"
Cleese, John; DVD; $30.68
In Stock
Sold by: buybackselyria

They are not identical

Document de-duplication



And which is similar to which is not always clear…

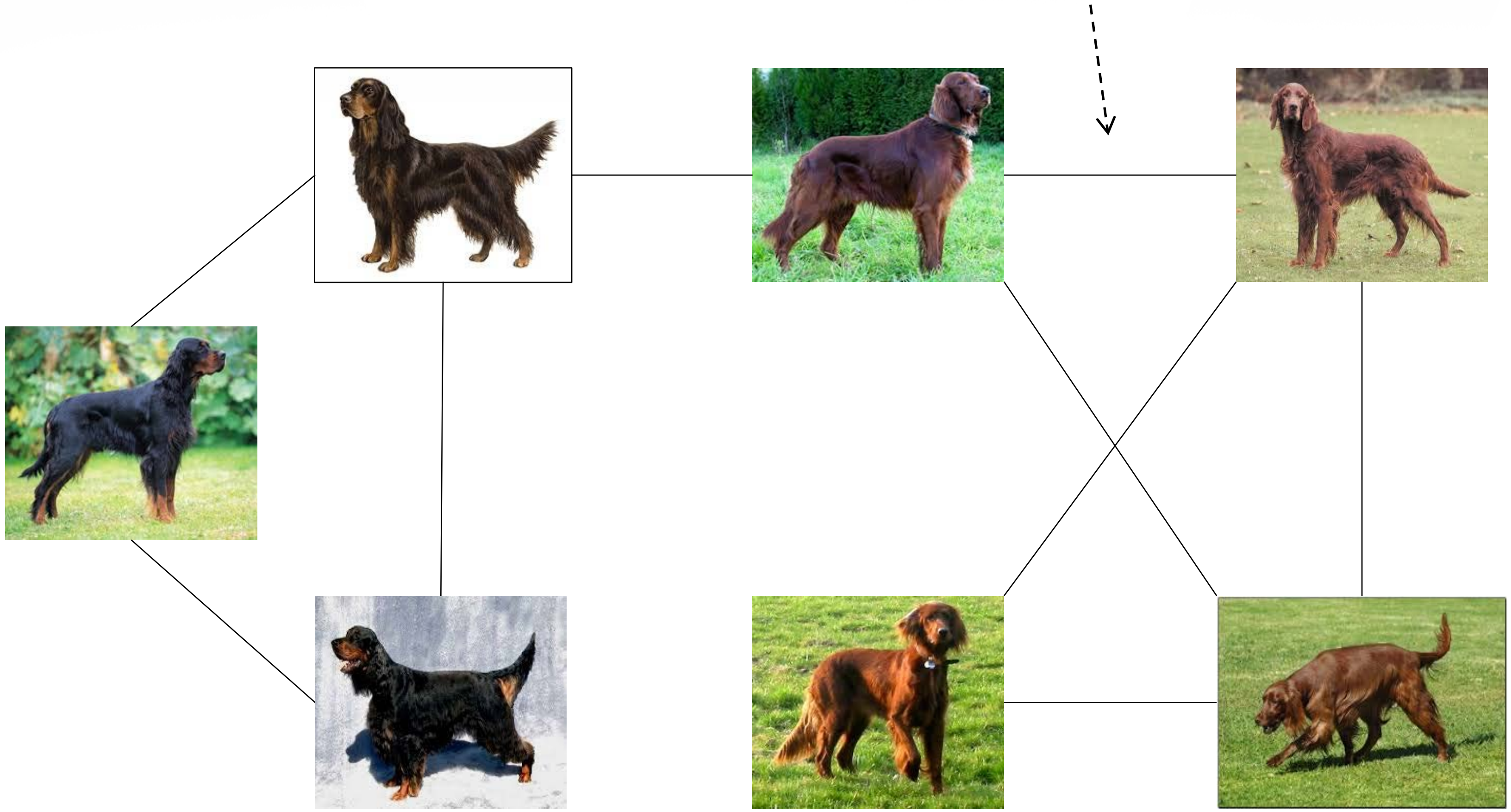
YAHOO!

Document de-duplication

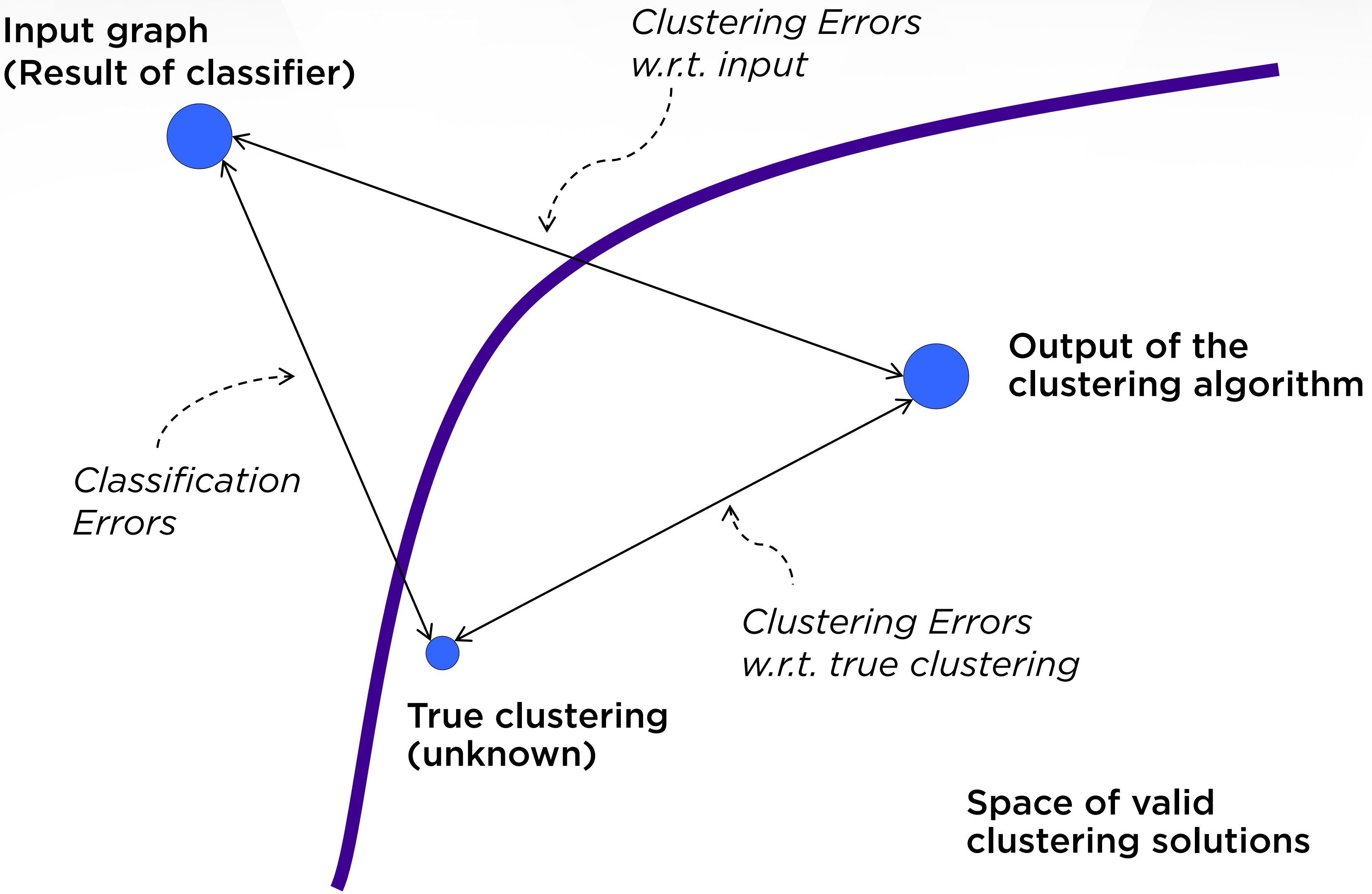
YAHOO!

Motivation from machine learning

$f($  ,  $) = 1$



Motivation from machine learning



YAHOO!

Some bad news : min-disagree

- Unweighted complete graphs - NP-hard (BBC02)
  - Reduction from “Partition into Triangles”

- Unweighted general graphs - APX-hard (DEFI06)
  - Reduction from multiway cuts.

- Weighted general graphs - APX-hard (DEFI06)
  - Reduction from multiway cuts.

Algorithms for unweighted min-disagree

An algorithms is a C approximation if:

$$\mathbb{E}[\mathrm{ALG}] \leq C \cdot \mathrm{OPT}$$

Paper	Approximation	Running time
[BBC02]	$\approx 20,000$	$O(n^2)$
[DEFI06]	$4\log(n)$	LP
[CGW03]	4	LP
[ACNA05]	2.5	LP
[ACNA05]	3	$O(m)$
[AL09]	< 3	$O(n) + OPT$

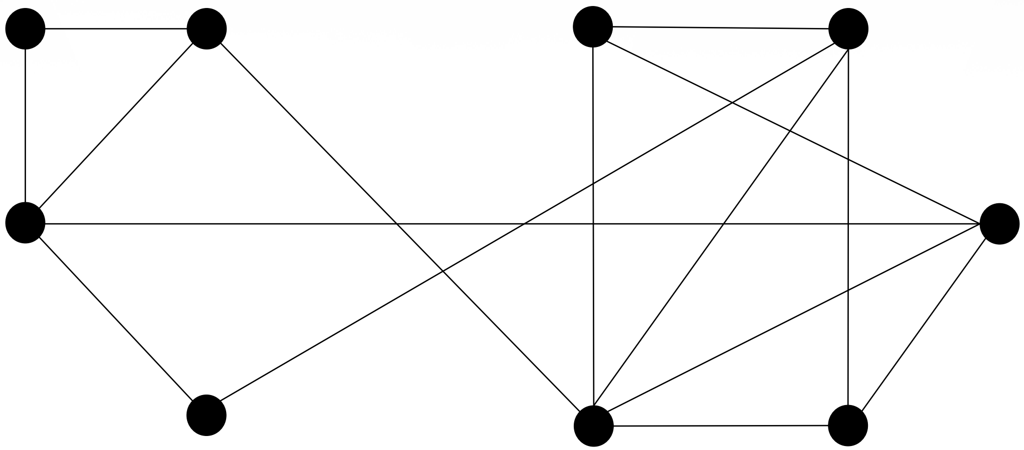
YAHOO!

Algorithm warm-up

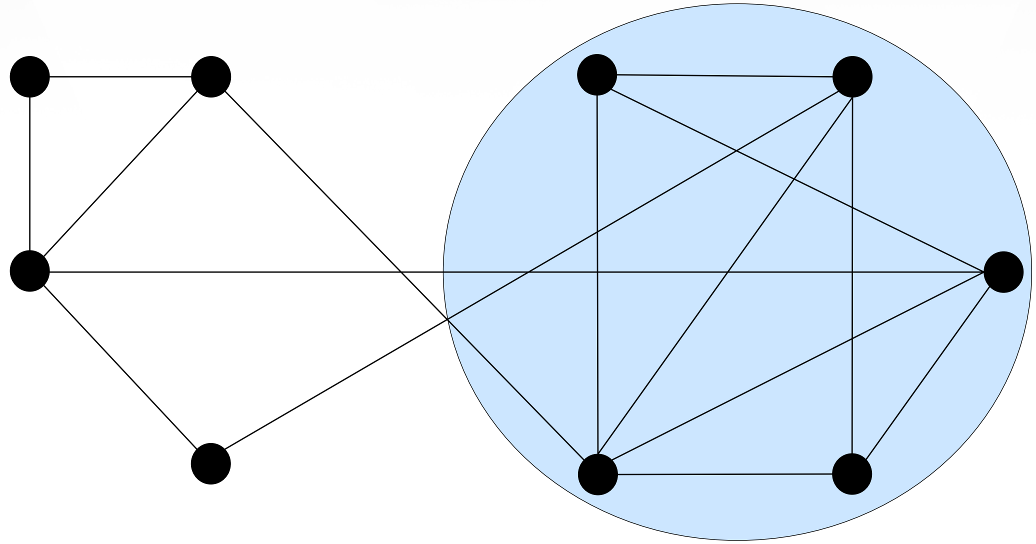
From

Correlation clustering, 2002
Nikhil Bansal, Avrim Blum, and Shuchi Chawla.

Algorithm warm-up

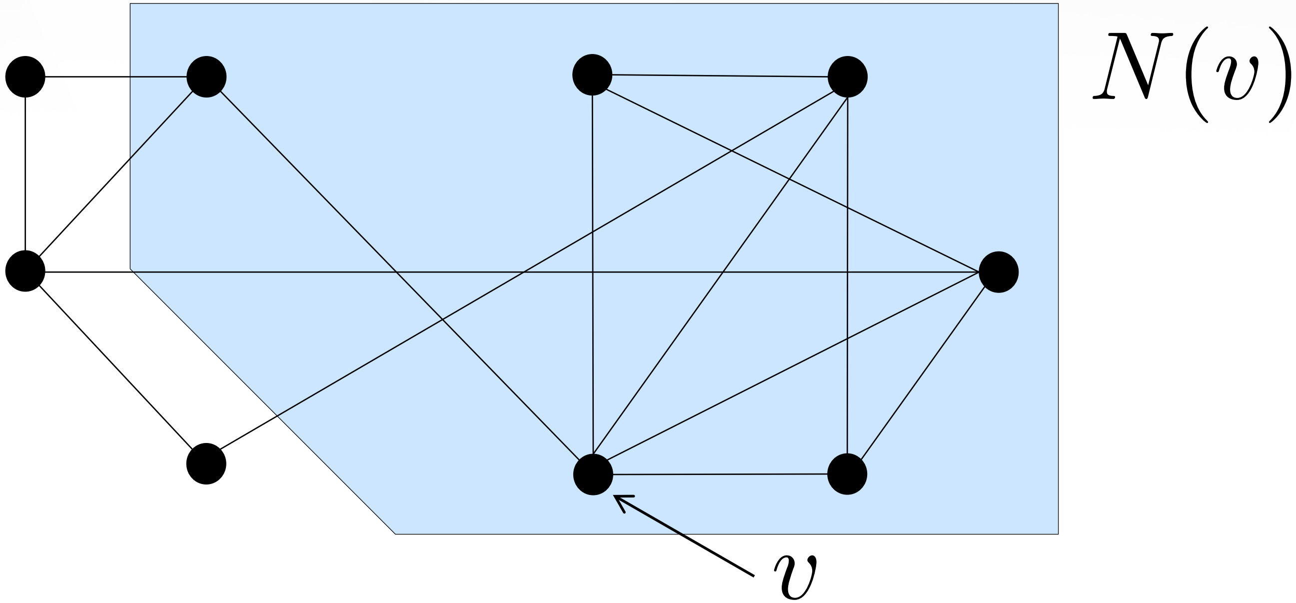


Algorithm warm-up



Consider only clustering to 2 clusters (for now...)

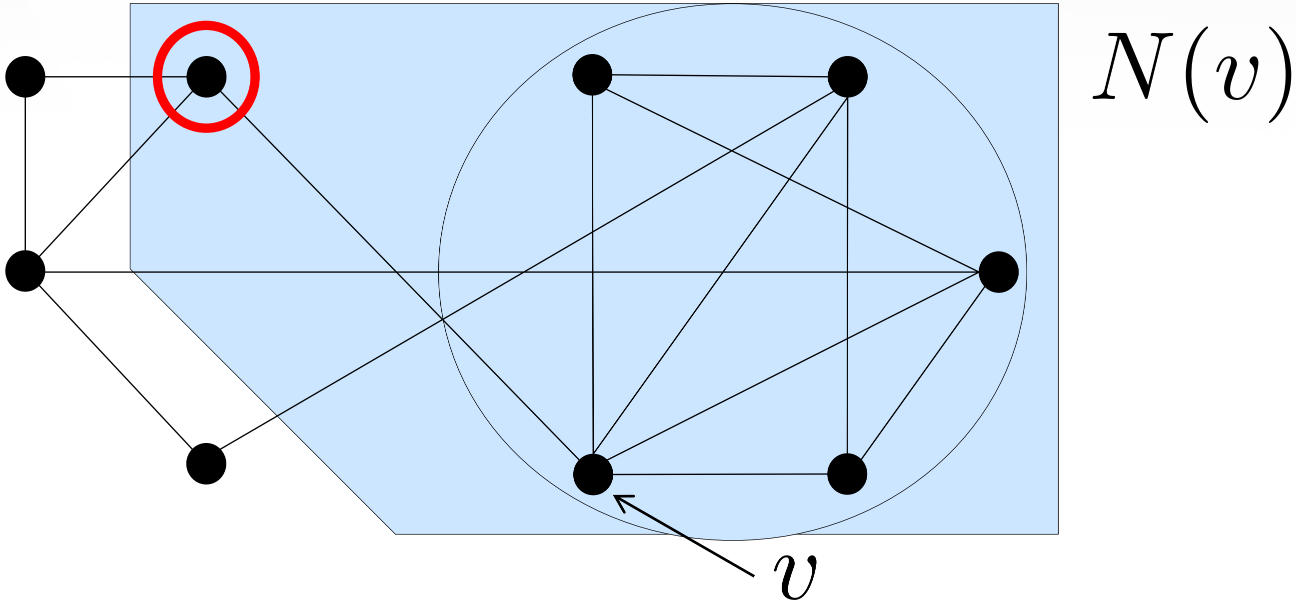
Algorithm warm-up



Consider all clustering to 2 clusters of the form

$$(N(v)\ ,\ \bar{N}(v))$$

Algorithm warm-up

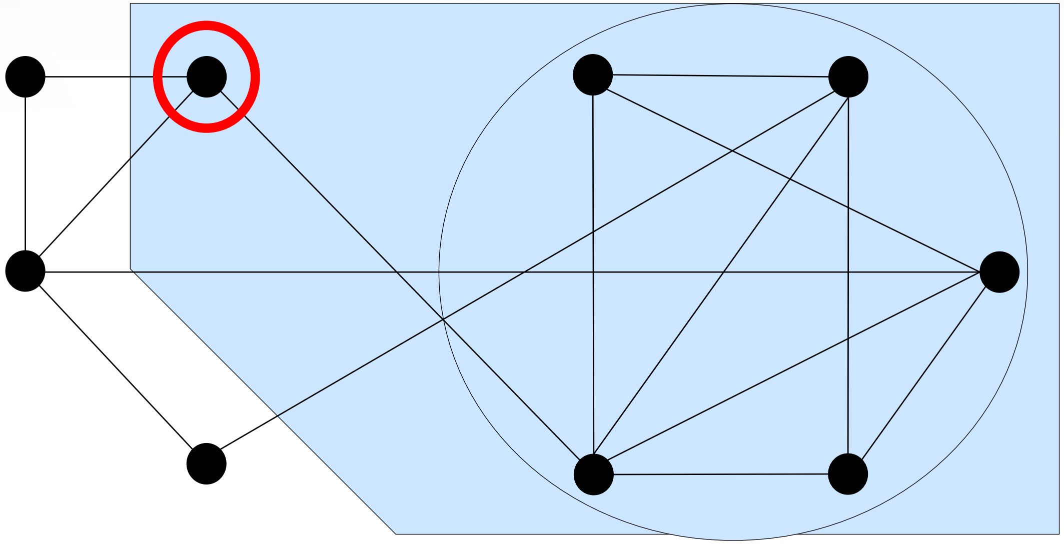


Consider the one whose neighborhood disagrees the least with the best clustering.

(Here $d = 1$)

YAHOO!

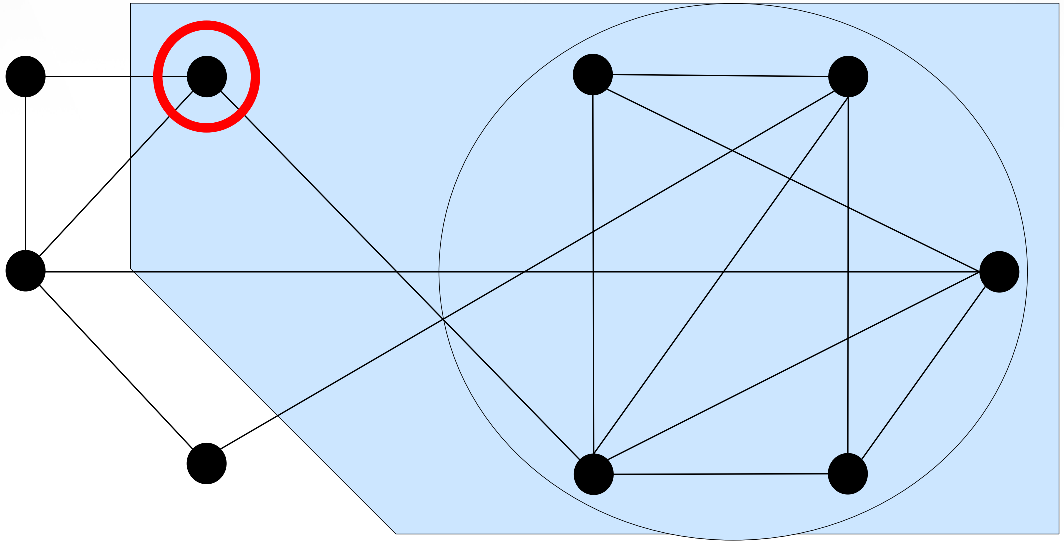
Algorithm warm-up



Each node "contributes" at least $d/2$ mistakes.

Therefore $\mathrm{OPT} \geq nd/2$

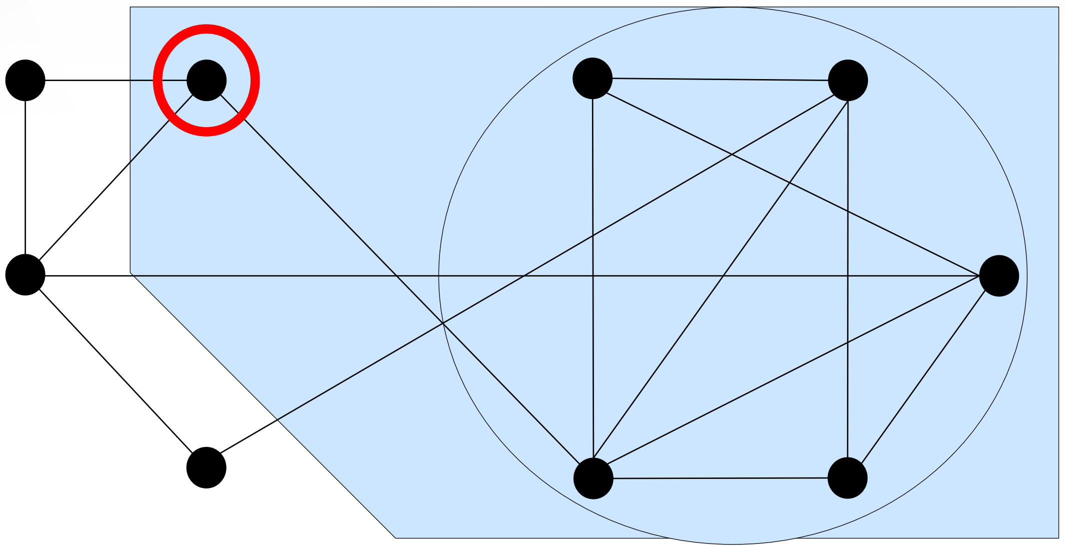
Algorithm warm-up



On the other hand $\mathrm{ALG} \leq \mathrm{OPT} + nd$

(Each of the d disagreements adds at most n errors)

Algorithm warm-up



Putting it all together $\mathrm{OPT} \geq nd/2$ and $\mathrm{ALG} \leq \mathrm{OPT} + nd$

Gives: $$\mathrm{ALG} \leq 3\,\mathrm{OPT}$$

LP based solutions

Erik D. Demaine, Dotan Emanuel, Amos Fiat, Nicole Immorlica
Correlation clustering in general weighted graphs, 2006

Moses Charikar, Venkatesan Guruswami, and Anthony Wirth.
Clustering with qualitative information, 2003

Nir Ailon, Moses Charikar, Alantha Newman 2005
Aggregating inconsistent information: ranking and clustering

LP relaxation

Minimize $$\sum_{(i,j)\in E} d_{ij} + \sum_{(i,j)\notin E} 1 - d_{ij}$$

s.t. $$\forall_{ij} \;\; d_{ij} \in \{0,1\}$$

$$\forall_{i,j,k} \;\; d_{ik} \leq d_{ij} + d_{jk}$$

LP relaxation

Minimize $$\sum_{(i,j)\in E} d_{ij} + \sum_{(i,j)\notin E} 1 - d_{ij}$$

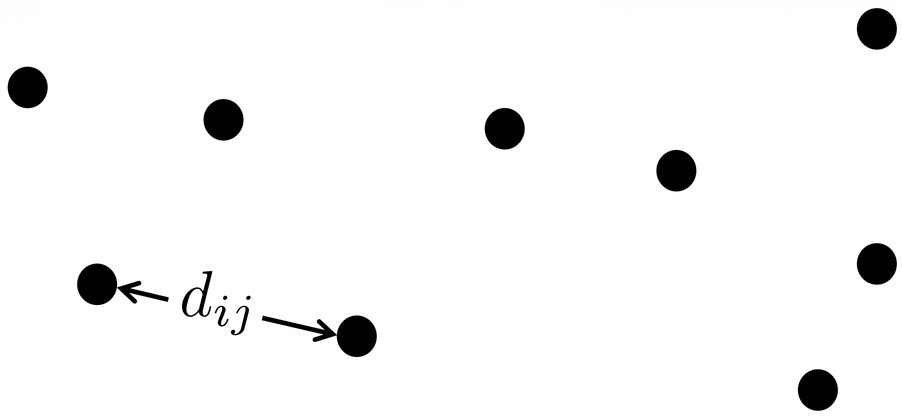
s.t. $$\forall_{ij} \;\; d_{ij} \in [0,1]$$ instead of $d_{ij} \in \{0,1\}$

$$\forall_{i,j,k} \;\; d_{ik} \leq d_{ij} + d_{jk}$$ triangle inequality

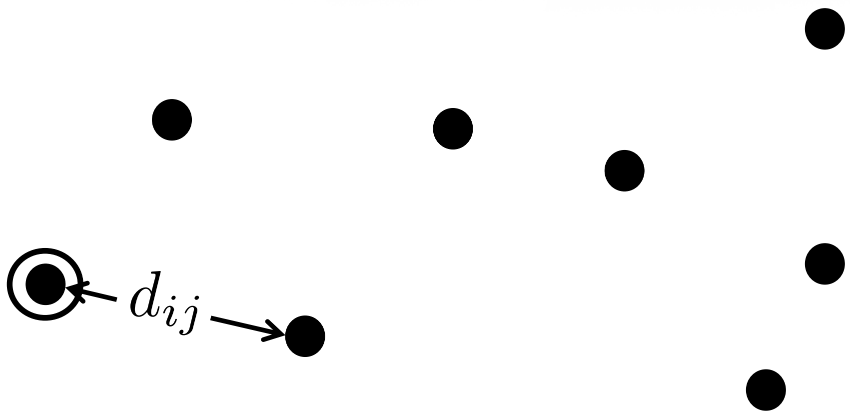
The solution is at least as good as OPT

But, it's fractional...

Region growing

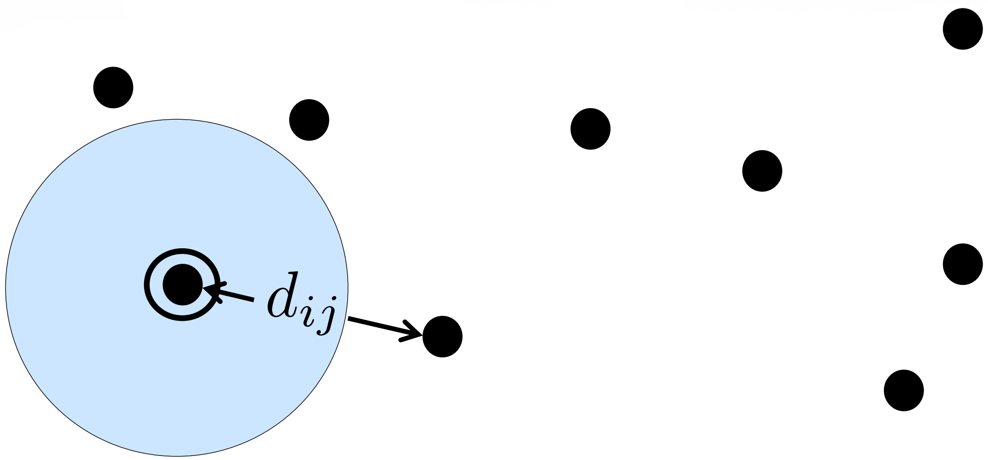


Region growing



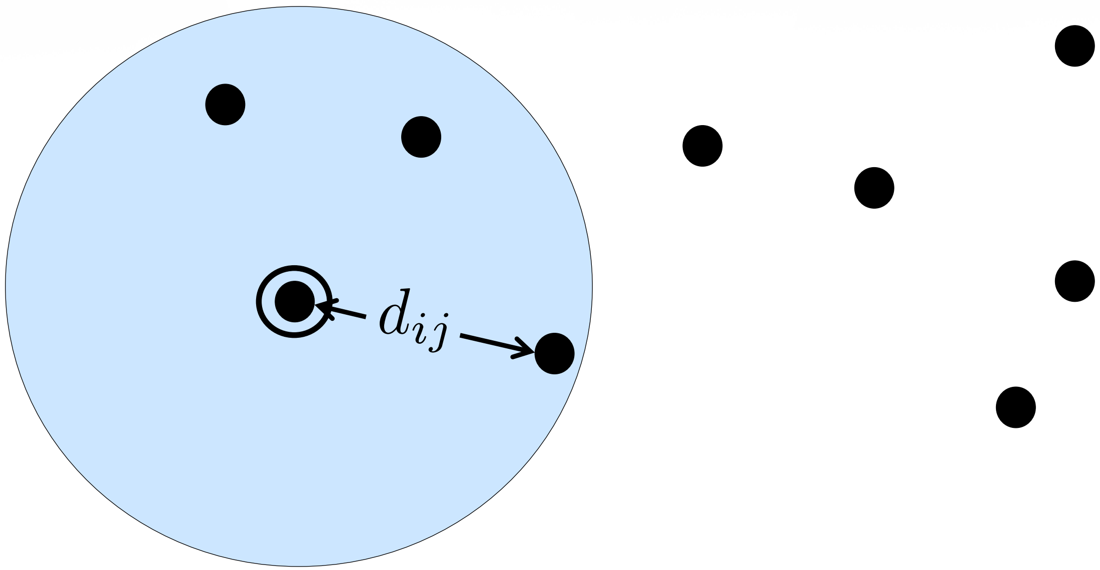
Pick an arbitrary node

Region growing



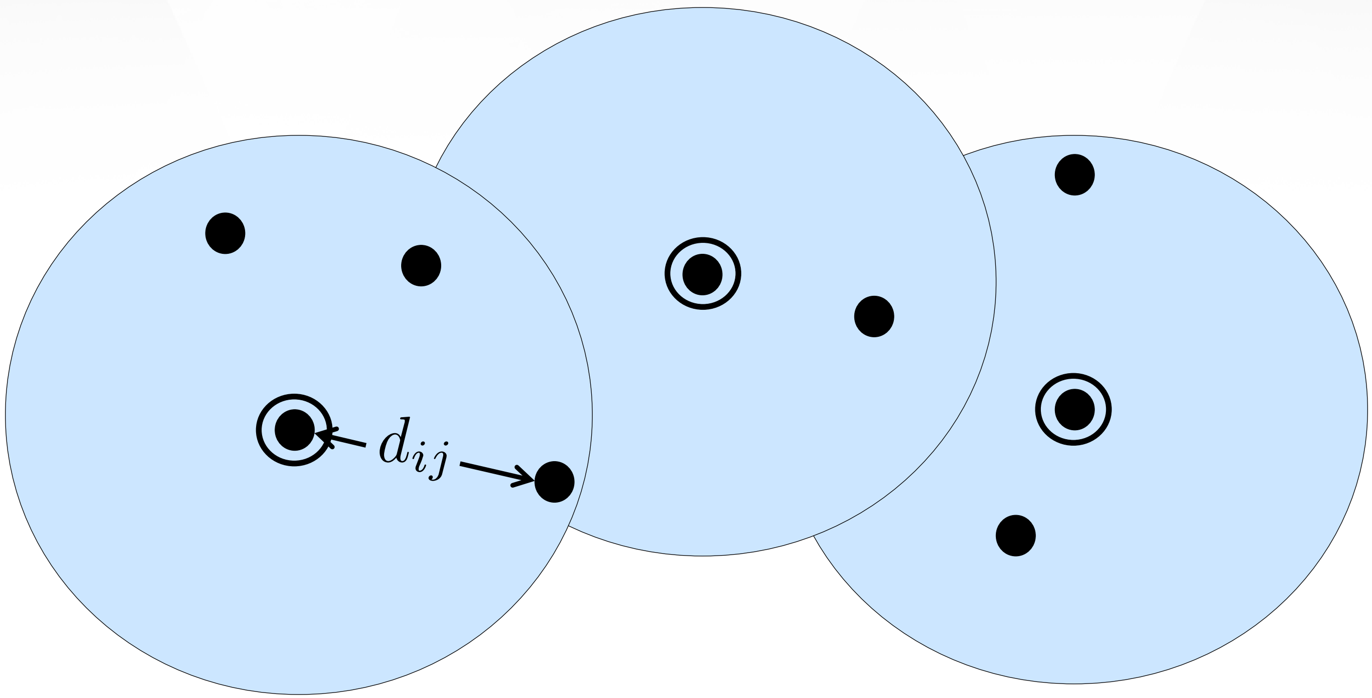
Start growing a ball around it

Region growing



Stop when some condition holds.

Region growing



And repeat until you run out of nodes.

YAHOO!

Some good and some bad news

Good news:

- [DEFI06] [CGW03] For weighted graphs we get:

$$\mathrm{ALG} \leq \mathrm{OPT} \cdot O(\log(n))$$

- [CGW03] For unweighted graphs we get:

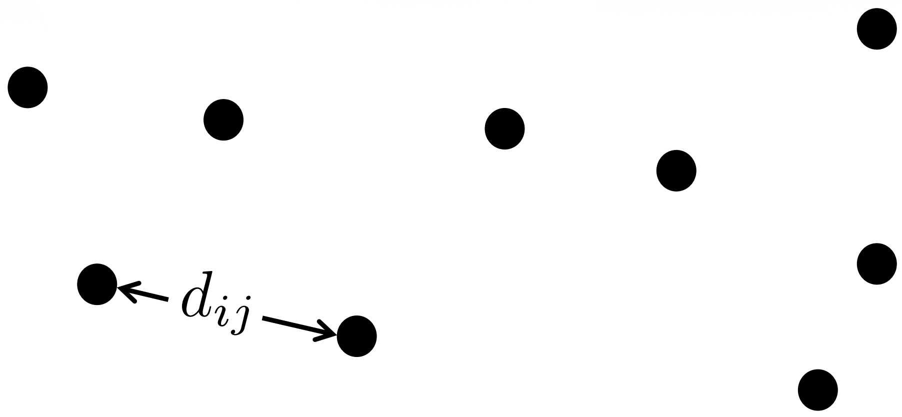
$$\mathrm{ALG} \leq 4\,\mathrm{OPT}$$

Pivot

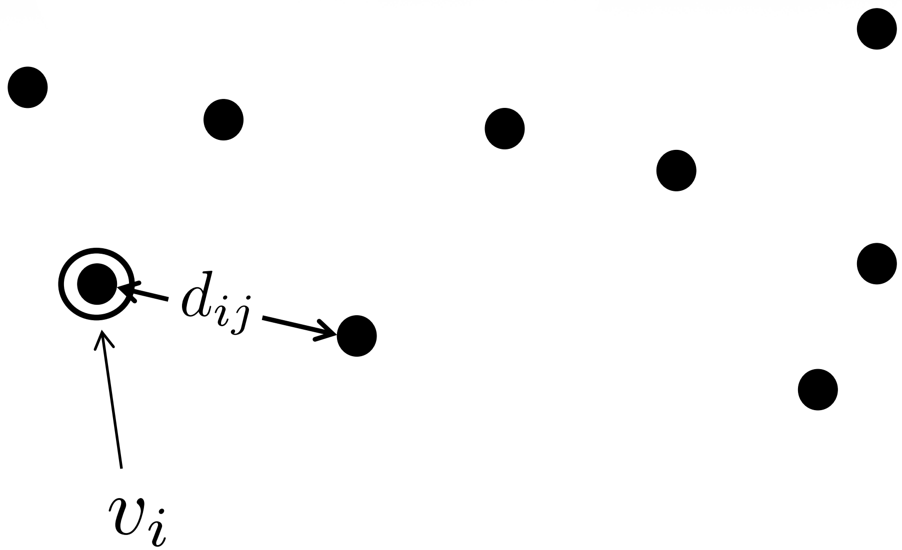
Nir Ailon, Moses Charikar, Alantha Newman 2005

Aggregating inconsistent information: ranking and clustering

Pivot



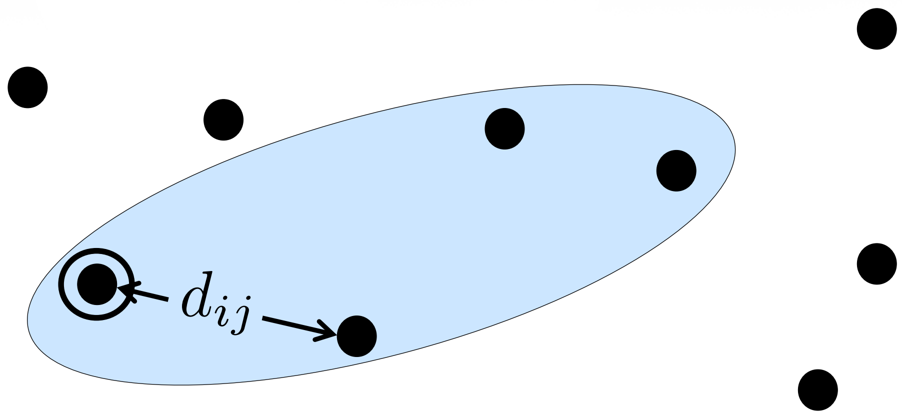
Pivot



Pick a node (v_i) uniformly at random

$$C = \{v_j\}$$

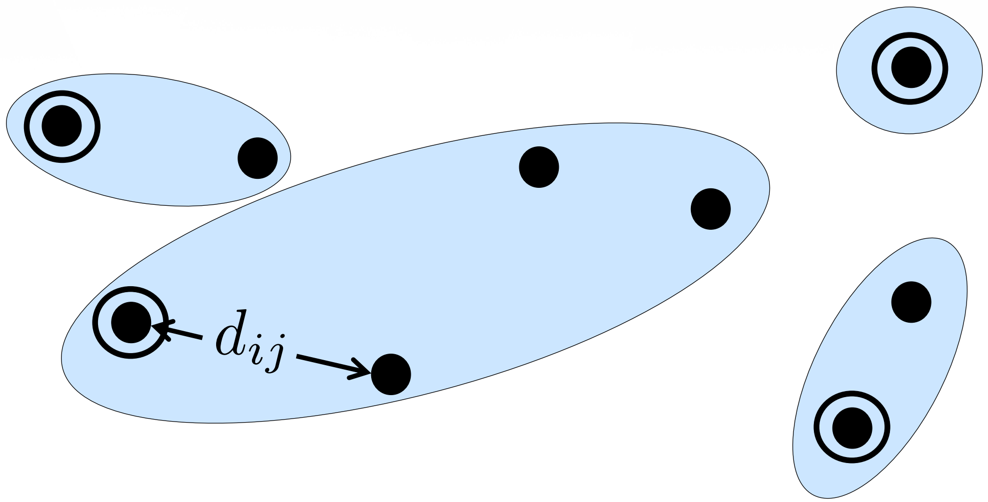
Pivot



With probability $1 - d_{ij}$ for all j

$$C \leftarrow C \cup \{v_j\}$$

Pivot



Recourse on the rest of the graph.

Some good and some bad news

Good news:

- The algorithm guaranties

$$\mathrm{ALG} \leq 2.5\,\mathrm{OPT}$$

- This is the best known approximation result!

Bad news:

- Solving large LPs is expensive.
- This LP has $\Omega(n^3)$ constraints… argh….

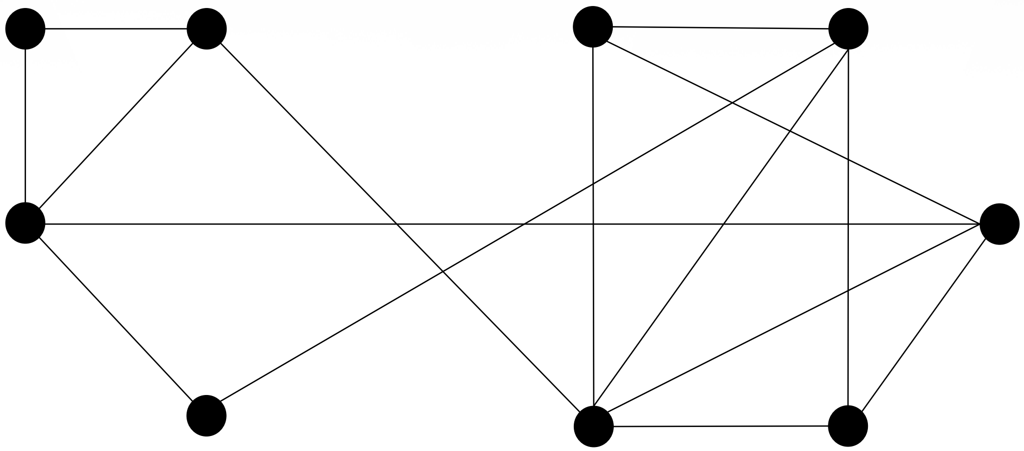
YAHOO!

Pivot – skipping the LP

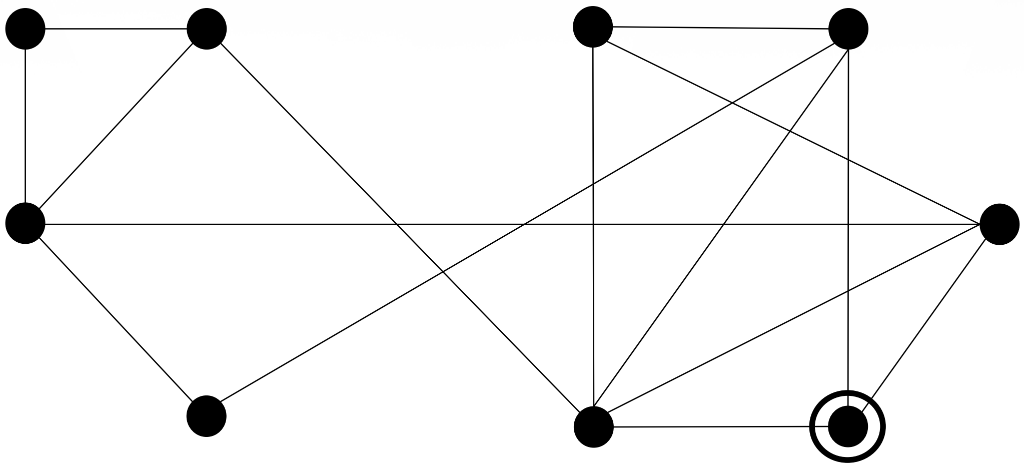
Nir Ailon, Moses Charikar, Alantha Newman 2005

Aggregating inconsistent information: ranking and clustering

Pivot

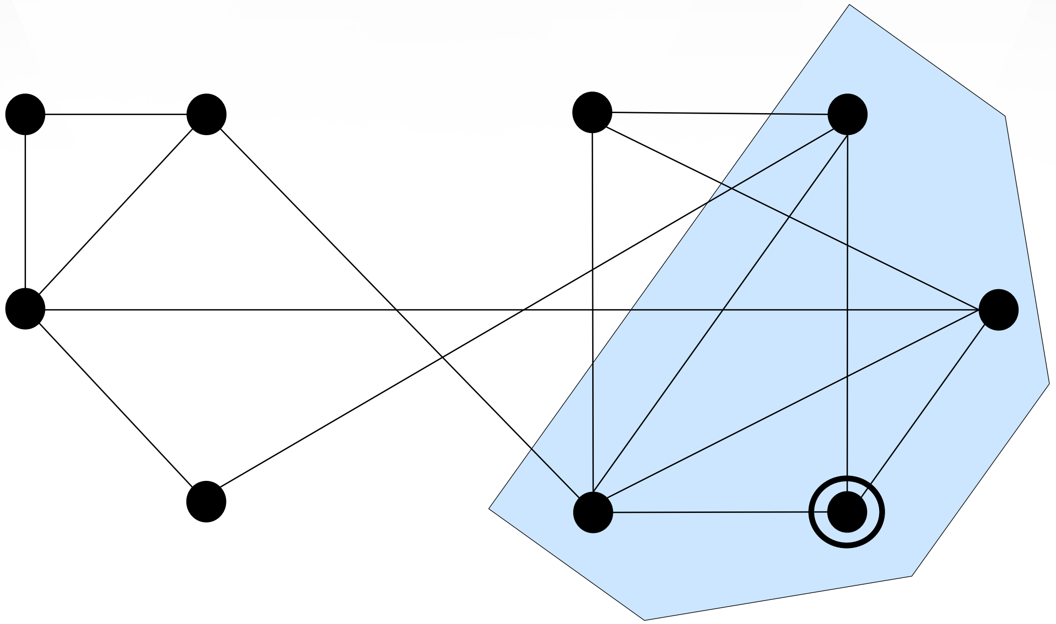


Pivot



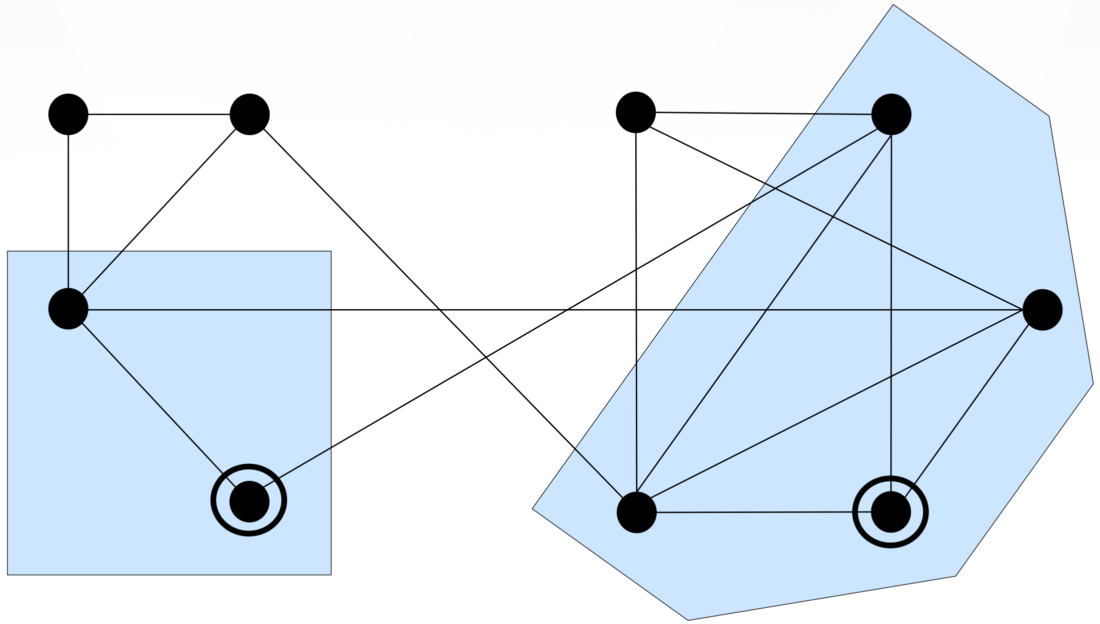
Pick a random node (uniformly!!!)

Pivot



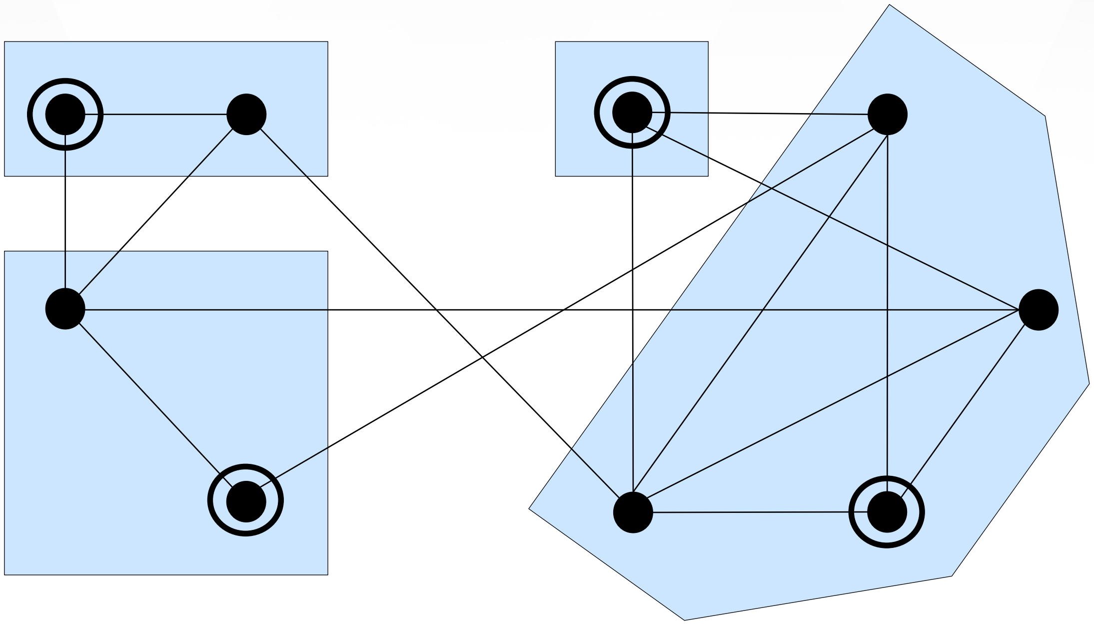
Declare itself and its neighbors as the first cluster.

Pivot



Pick a random node again (uniformly from the rest)

Pivot



And continue until you consume the entire graph.

Some good and some bad news

Good news:

- The algorithm guaranties

$$\mathbb{E}[\mathrm{ALG}] \leq 3\,\mathrm{OPT}$$

- Running time is $O(m)$, very efficient!!

Bad news:

- Works only for complete unweighted graphs

References

Nikhil Bansal, Avrim Blum, and Shuchi Chawla 2002
Correlation clustering

Erik Demaine, Dotan Emanuel, Amos Fiat, Nicole Immorlica 2006
Correlation clustering in general weighted graphs

Moses Charikar, Venkatesan Guruswami, Anthony Wirth 2003
Clustering with qualitative information.

Nir Ailon, Moses Charikar, Alantha Newman 2005
Aggregating inconsistent information: ranking and clustering

Further reading

Ioannis Giotis, Venkatesan Guruswami 2006
Correlation Clustering with a Fixed Number of Clusters

Nir Ailon, Edo Liberty 2009
Correlation Clustering Revisited: The "True" Cost of Error Minimization Problems.

Anke van Zuylen, David P. Williamson 2009
Deterministic Pivoting Algorithms for Constrained Ranking and Clustering Problems

Claire Mathieu, Warren Schudy 2010
Correlation Clustering with Noisy Input

Claire Mathieu, Ocan Sankur, Warren Schudy 2010
Online Correlation Clustering

Nir Ailon, Noa Avigdor-Elgrabli, Edo Liberty, Anke van Zuylen 2012
Improved Approximation Algorithms for Bipartite Correlation Clustering.

Nir Ailon and Zohar Karnin 2012
No need to choose: How to get both a PTAS and Sublinear Query Complexity

Part II:
Correlation clustering variants

Francesco Bonchi
Yahoo Labs, Barcelona

Correlation clustering variants

- Overlapping
- Chromatic
- On-line
- Bipartite
- Clustering aggregation

Overlapping correlation clustering

F. Bonchi, A. Gionis, A. Ukkonen: Overlapping Correlation Clustering ICDM 2011 overlapping clusters are very natural

- social networks
- proteins
- documents

From correlation clustering to overlapping correlation clustering

- Correlation clustering:
  - Set of objects $V = \{v_1, \ldots, v_n\}$
  - Similarity function $s : V \times V \rightarrow [0, 1]$
  - Labeling function $\ell : V \rightarrow L$

$$C_{cc}(\ell) = \sum_{(u,v) \in V \times V} |s(u, v) - I(\ell(u) = \ell(v))|$$

- Overlapping correlation clustering:
  - Labeling function $\ell : V \rightarrow 2^L \setminus \{\emptyset\}$
  - Similarity function between sets of labels $H : 2^L \times 2^L \rightarrow [0, 1]$

$$C_{occ}(\ell) = \sum_{(u,v) \in V \times V} |s(u, v) - H(\ell(u), \ell(v))|$$

OCC problem variants

$$(r, H, p)$$

- Based on these choices:
  - Similarity function *s* takes values in $[0,1]$ $\qquad r = \mathrm{f}$
  - Similarity function *s* takes values in $\{0,1\}$ $\qquad r = \mathrm{b}$
  - Similarity function *H* is the *Jaccard coefficient* $\qquad H = J$
  - Similarity function *H* is the *intersection indicator* $\qquad H = I$
  - Constraint on the maximum number of labels per object $\quad |\ell(v)| \leq p, \ \forall v \in V$
  - Special cases:

$p = 1$ — normal Correlation Clustering

$p = k \quad \text{where} \quad |L| = k$ — no constraint

YAHOO!

Some results

- (r, H, p) is NP-Hard [from hardness of $(r, H, 1)$]
- (r, I, p), with $p > 1$ is NP-Hard [from $\textsc{covering-by-cliques}$]
- (r, I, p) is hard to approximate [from $\textsc{covering-by-cliques}$]
- $(r, I, \Theta(n^2))$ the optimal solution can be found in polynomial time
- $(\mathrm{b}, I, \Theta(n^2))$ admits a zero-cost polynomial time solution

- Connection with graph coloring
- Connection with dimensionality reduction

Local-search algorithm

- We observe that cost can be rewritten as:

$$
\begin{aligned}
C_{\mathrm{occ}}(V,\ell) &= \frac{1}{2}\sum_{v\in V}\sum_{u\in V\setminus\{v\}} |H(\ell(v),\ell(u)) - s(v,u)| \\
&= \frac{1}{2}\sum_{v\in V} C_{v,p}(\ell(v) \mid \ell),
\end{aligned}
$$

where $C_{v,p}(\ell(v) \mid \ell) = \sum_{u\in V\setminus\{v\}} |H(\ell(v),\ell(u)) - s(v,u)|$

Algorithm 1 `LocalSearch`

1: initialize ℓ to a valid labeling;
2: **while** $C_{\mathrm{occ}}(V,\ell)$ decreases **do**
3:     **for** each $v \in V$ **do**
4:         find the label set L that minimizes $C_{v,p}(L \mid \ell)$;
5:         update ℓ so that $\ell(v) = L$;
6: **return** ℓ

YAHOO!

Local step for Jaccard

- $\textsc{Jaccard-Triangulation}$
  - Given $\{\langle S_j, z_j \rangle\}_{j=1\ldots n}$
  - Find $X \subseteq U$ that minimizes

$$d(X, \{\langle S_j, z_j \rangle\}_{j=1\ldots n}) = \sum_{j=1}^{n} |J(X, S_j) - z_j|$$

- $\textsc{Jaccard-Triangulation}$ is NP-Hard
  - generalization of “Jaccard median” problem*
  - non-negative least squares + post-processing of the fractional solution

F. Chierichetti, R. Kumar, S. Pandey, S. Vassilvitskii: Finding the Jaccard Median. SODA 2010

Local step for set intersection indicator

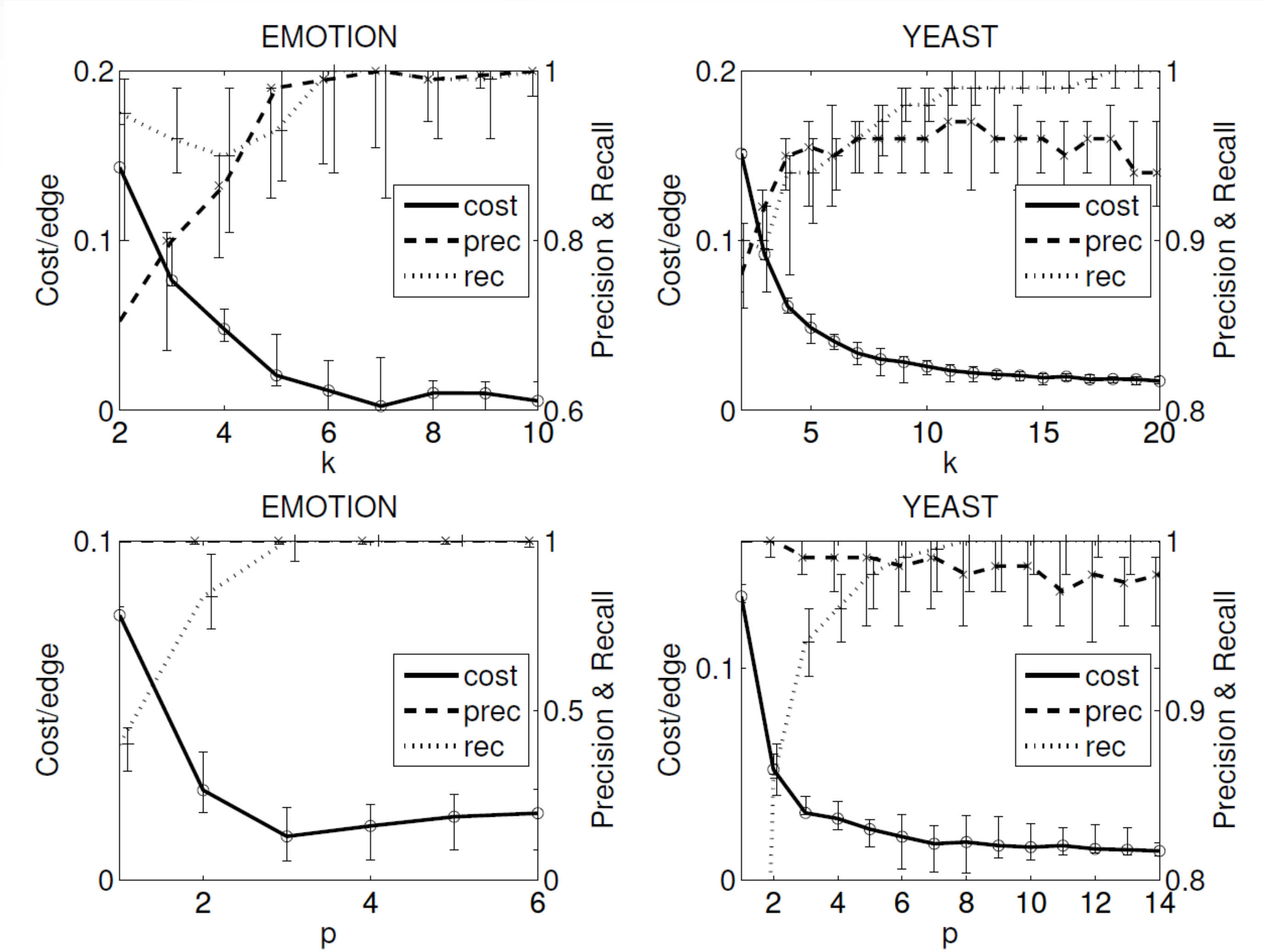
- $\textsc{Hit-n-Miss}$ problem
- Inapproximable within a constant factor
- $O(\sqrt{n}\log n)$ approximation by Greedy algorithm

Experiments on ground-truth overlapping clusters

- Two datasets from multilable classification
  - EMOTION: 593 objects, 6 labels
  - YEAST: 2417 objects, 14 labels
- Input similarity $s(u,v)$ is the Jaccard coefficient of the labels of u and v in the ground truth

YAHOO!

Experiments on ground-truth overlapping clusters



Application: overlapping clustering of trajectories

- Starkey project dataset containing the radio-telemetry locations of elks, deer, and cattle.
- 88 trajectories
  - 33 elks
  - 14 deers
  - 41 cattles
- 80K *(x,y,t)* observations (909 observations per trajectory in avg)
- Use EDR* as trajectory distance function, normalized to be in [0,1]

$$s(u, v) = 1 - edr(u, v)$$

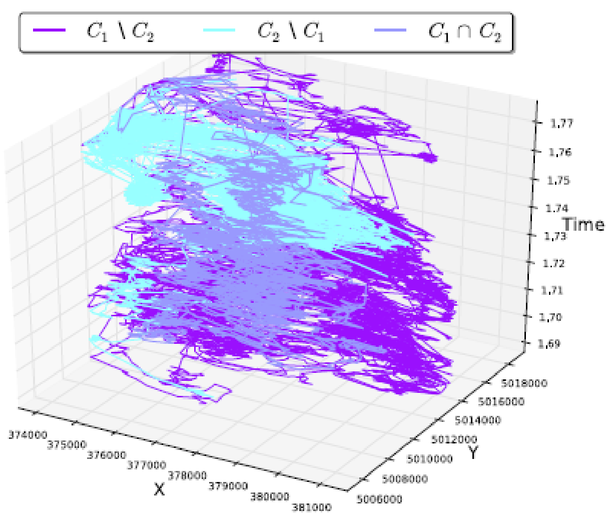
- Experiment setting: *k = 5, p = 2,* Jaccard

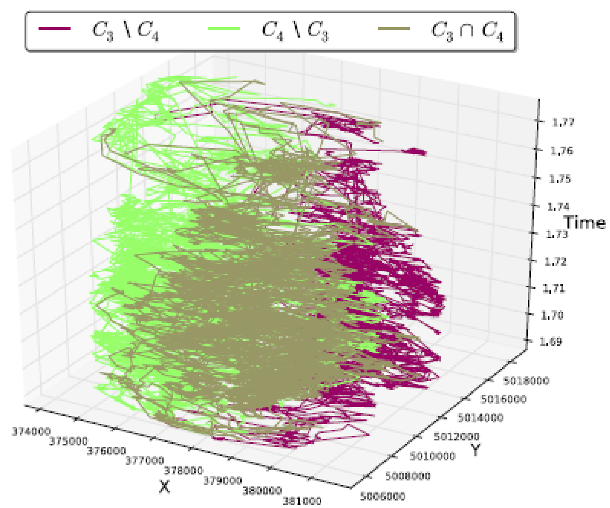
* L. Chen, M. T. Özsu, V. Oria: Robust and Fast Similarity Search for Moving Object Trajectories. SIGMOD 2005

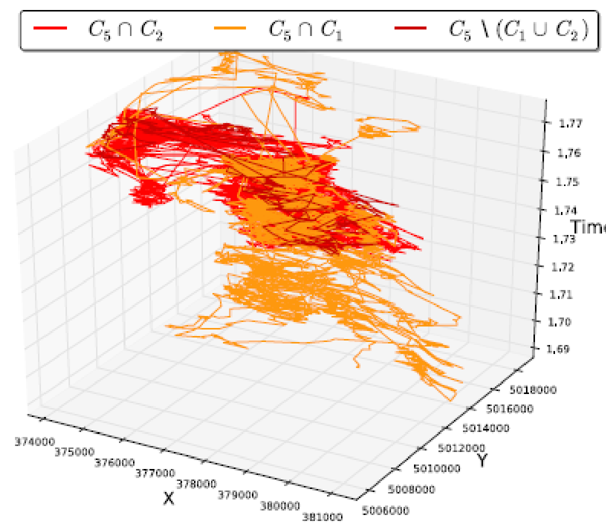
YAHOO!

Application: overlapping clustering of trajectories

C_1	C_2	C_3	C_4	C_5	clusters
16E 6D	3E 2D	3E	5E	3E 2D	C_1
	4E 2D 38C	$\emptyset$	1E	30C	C_2
		13E 6D	9E	$\emptyset$	C_3
			21E 2D	$\emptyset$	C_4
				3E 2D 33C	C_5







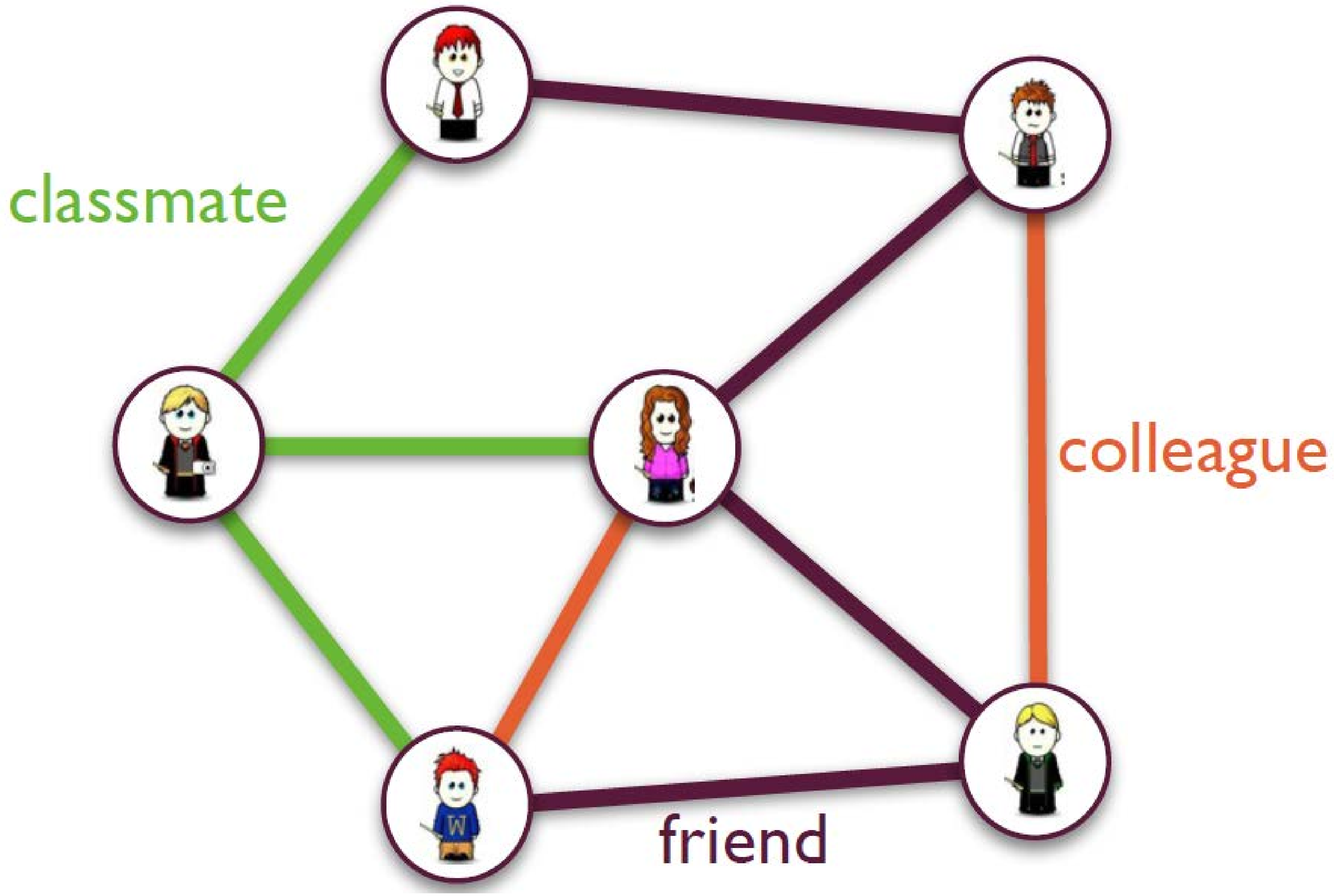
YAHOO!

Chromatic correlation clustering

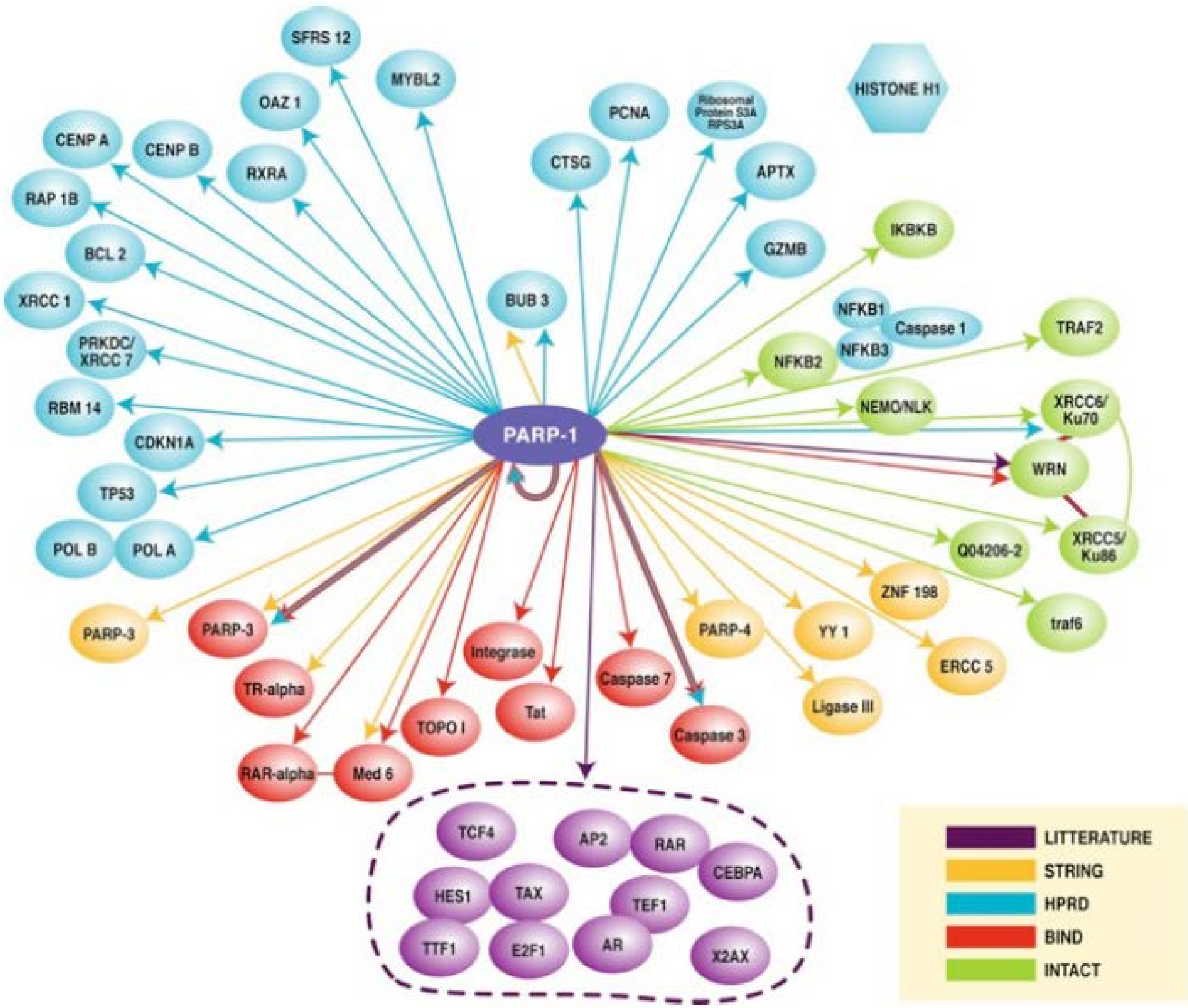
F. Bonchi, A. Gionis, F. Gullo, A. Ukkonen:
Chromatic correlation clustering
KDD 2012

- heterogeneous data
- objects of single type
- associations between objects are categorical
- can be viewed as edges with colors in a graph

Example: social networks



Example : protein interaction networks

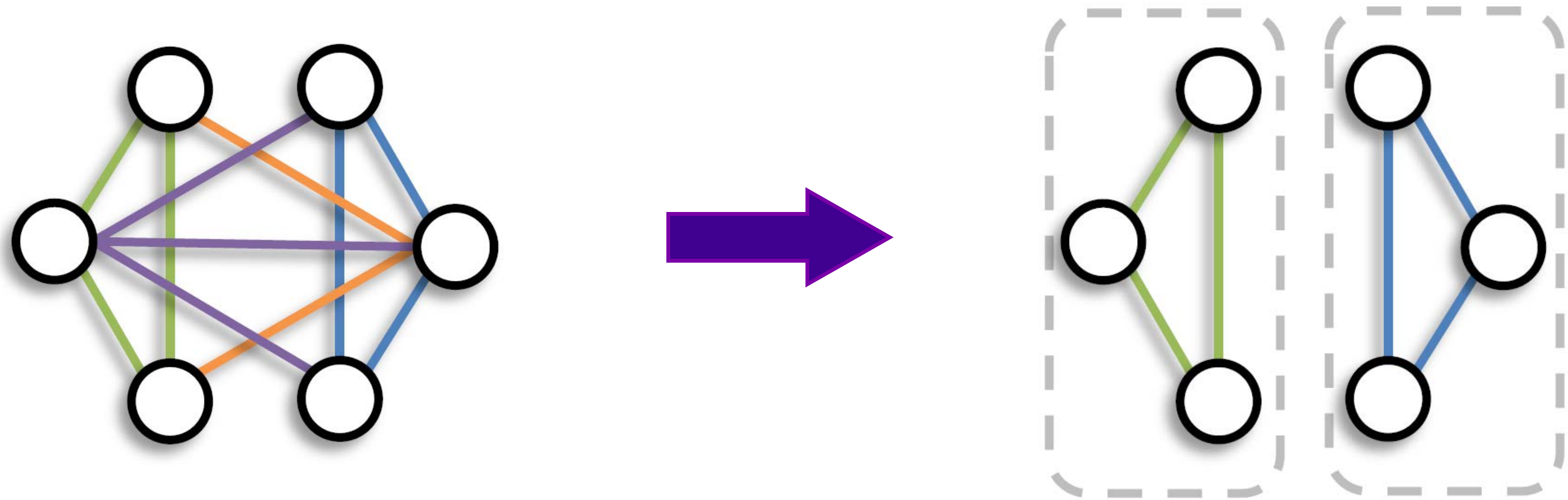


YAHOO!

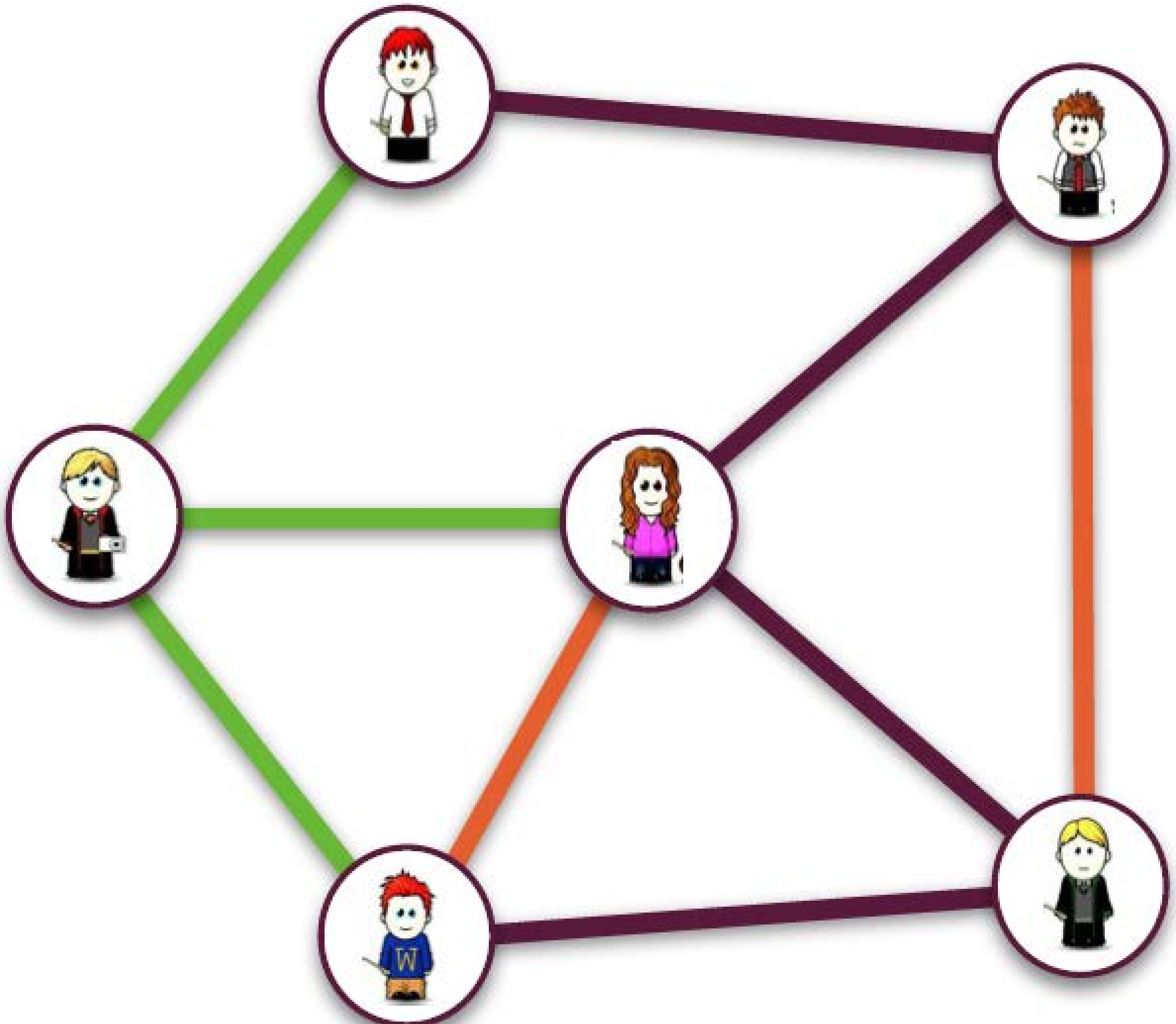
Research question

- how to incorporate edge types in the clustering framework?

- Intuitively:

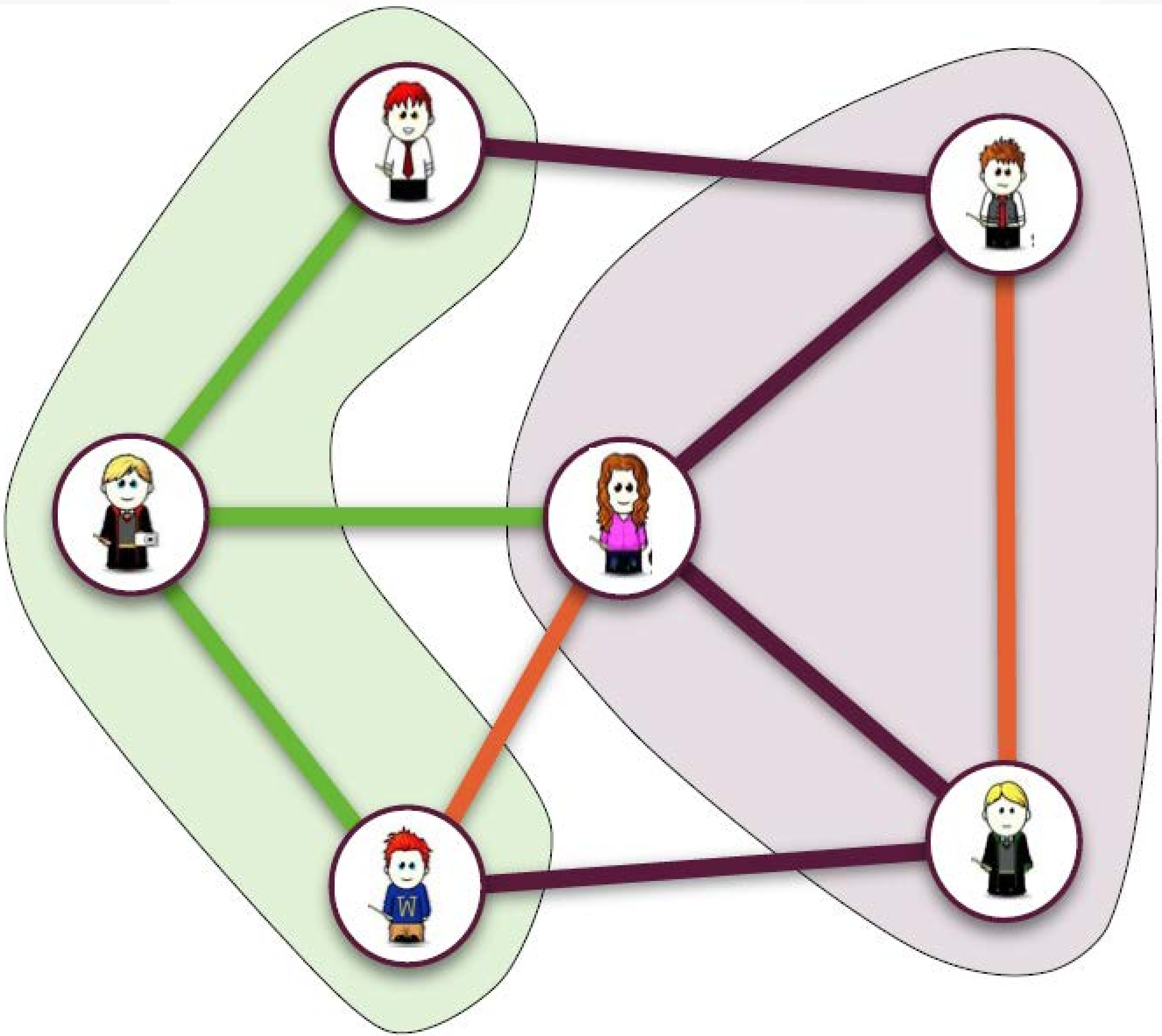


Chromatic correlation clustering

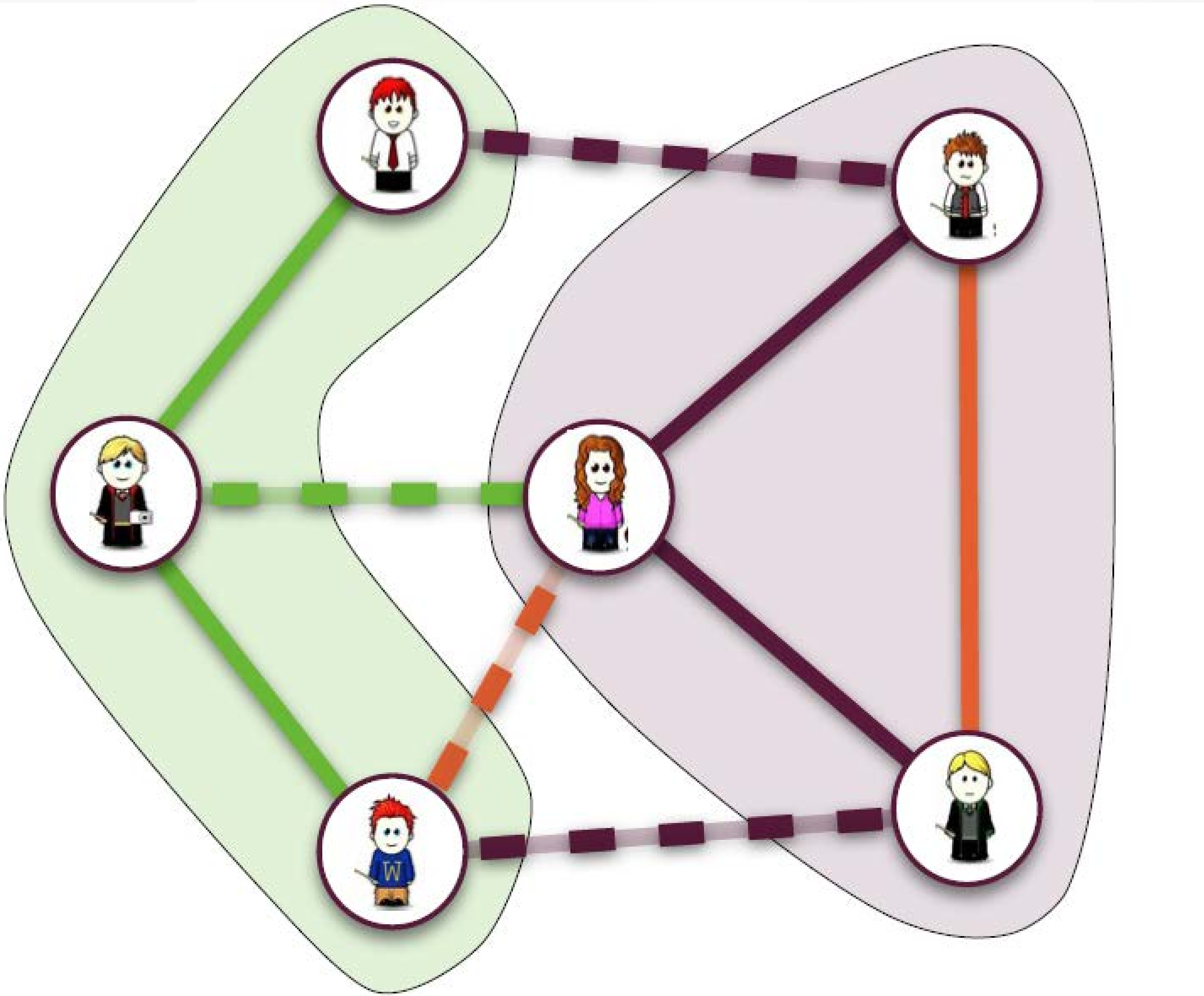


YAHOO!

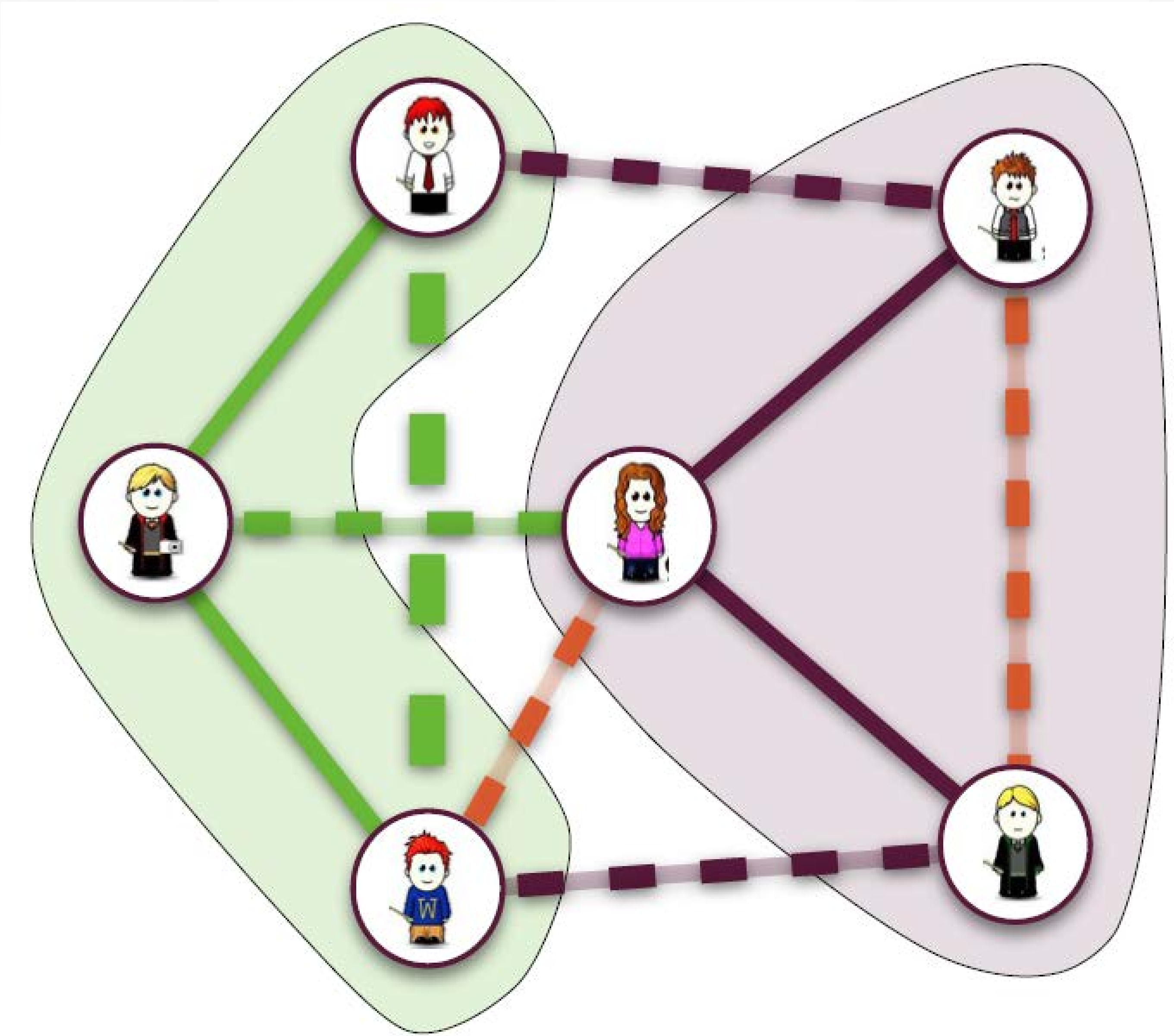
Chromatic correlation clustering



Cost of chromatic correlation clustering



Cost of chromatic correlation clustering



YAHOO!

From correlation clustering to chromatic correlation clustering

- <u>Correlation clustering</u>:
  - Set of objects $V = \{v_1, \ldots, v_n\}$
  - Similarity function $\mathrm{sim} : V \times V \to [0,1]$
  - Clustering $\mathcal{C} : V \to \mathbb{N}$

$$\mathrm{cost}(\mathcal{C}) = \sum_{\substack{(x,y)\in V\times V \\ \mathcal{C}(x)=\mathcal{C}(y)}} (1 - \mathrm{sim}(x,y)) + \sum_{\substack{(x,y)\in V\times V \\ \mathcal{C}(x)\neq\mathcal{C}(y)}} \mathrm{sim}(x,y)$$

- <u>Chromatic correlation clustering</u>:
  - Pairwise labeling function $\ell : V \times V \to L \cup \{l_0\}$
  - Clustering $\mathcal{C} : V \to \mathbb{N}$
  - Cluster labeling function $c\ell : \mathcal{C}[V] \to L$

$$\mathrm{cost}(\mathcal{C}, c\ell) = \sum_{\substack{(x,y)\in V\times V, \\ \mathcal{C}(x)=\mathcal{C}(y)}} (1 - \mathrm{I}[\ell(x,y) = c\ell(\mathcal{C}(x))]) + \sum_{\substack{(x,y)\in V\times V, \\ \mathcal{C}(x)\neq\mathcal{C}(y)}} \mathrm{I}[\ell(x,y) \neq l_0].$$

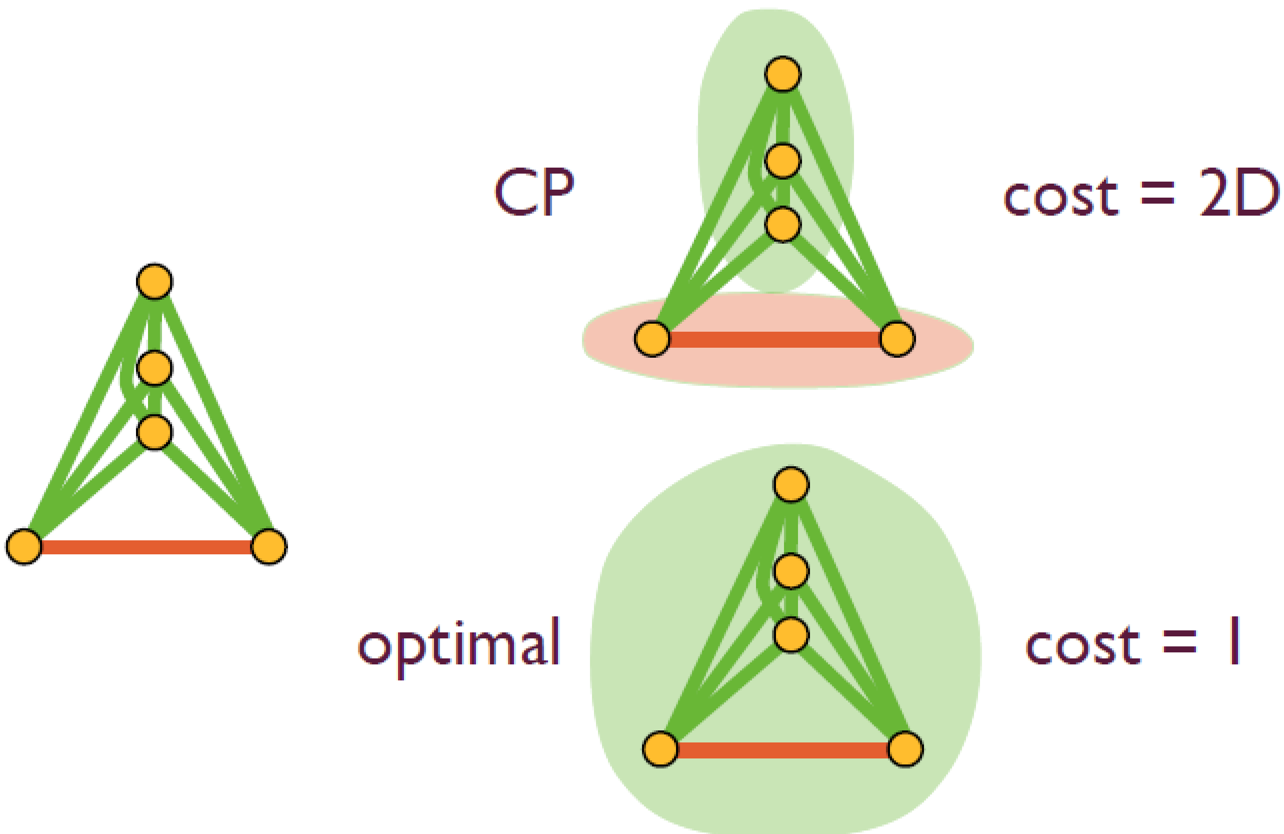
chromatic PIVOT algorithm

- Pick a random edge *(u,v)*, of color *c*
- Make a cluster with *u,v* and all neighbors *w*, such that *(u,v,w)* is monochromatic
- assign color *c* to the cluster
- repeat until left with empty graph

- approximation guarantee *6(2D-1)*
  - where *D* is the maximum degree
- Time complexity $\mathcal{O}(|E|)$

YAHOO!

how good is this bound ?



Lazy chromatic pivot

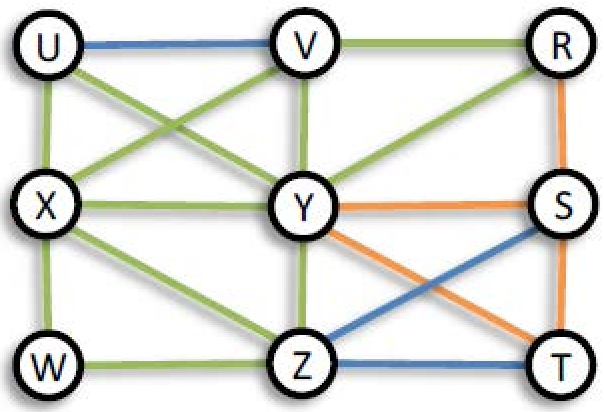
- Same scheme as Chromatic Pivot with two differences:

- The way how the pivot *(x,y)* is picked:
  not uniformly at random, but with probability proportional to the maximum chromatic degree

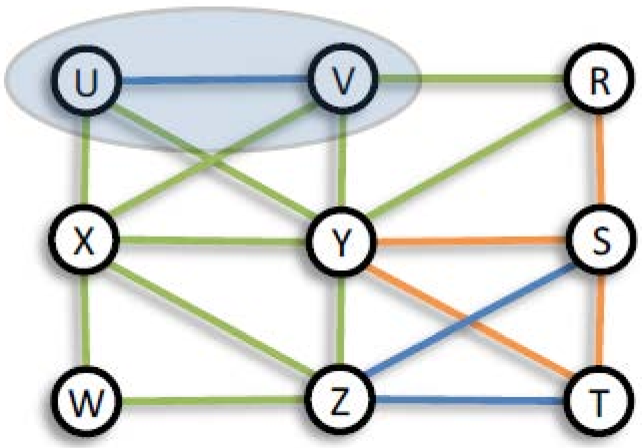
- The way how the cluster is built around *(x,y)*:
  not only vertices forming monochromatic triangles with the pivots, but also vertices forming monochromatic triangles with non-pivot vertices belonging to the cluster.

- Time complexity $\mathcal{O}((|L| + \log|V|)|E|)$

YAHOO!

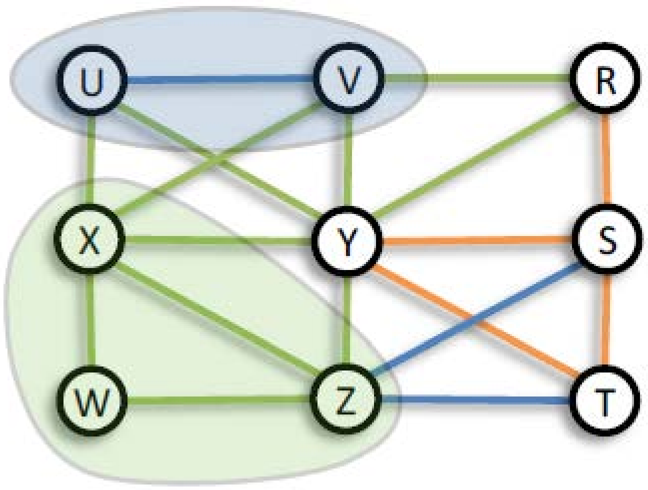


chromatic pivot

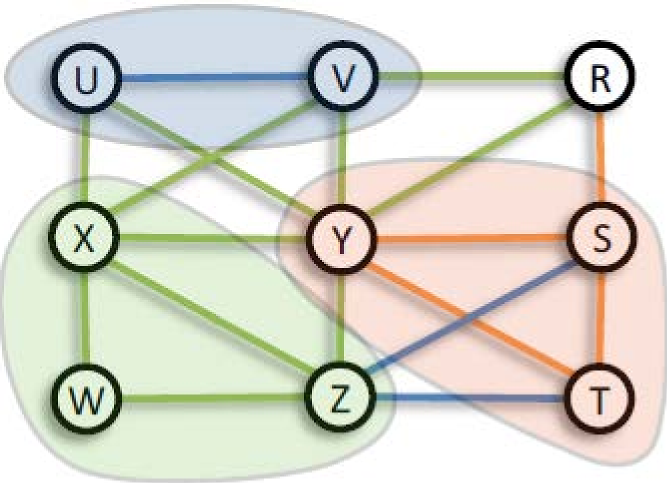


chromatic pivot

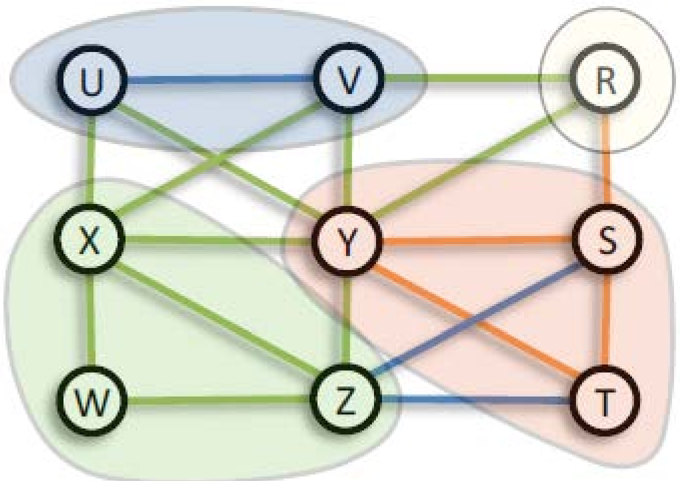
YAHOO!



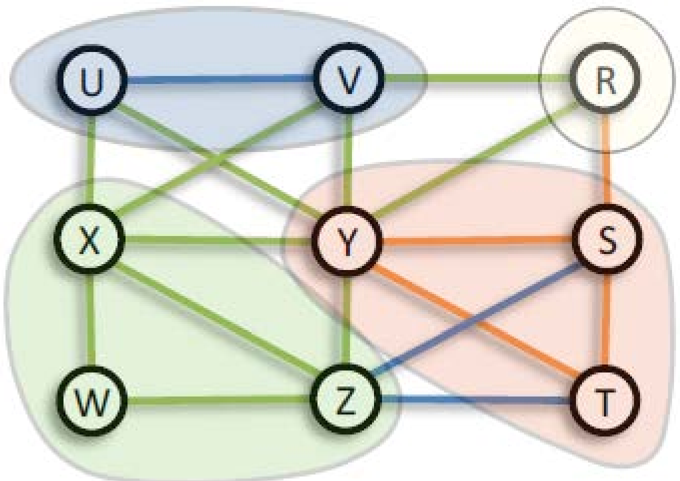
chromatic pivot



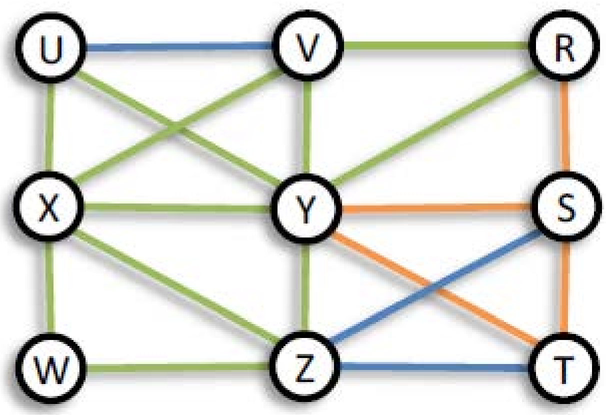
chromatic pivot



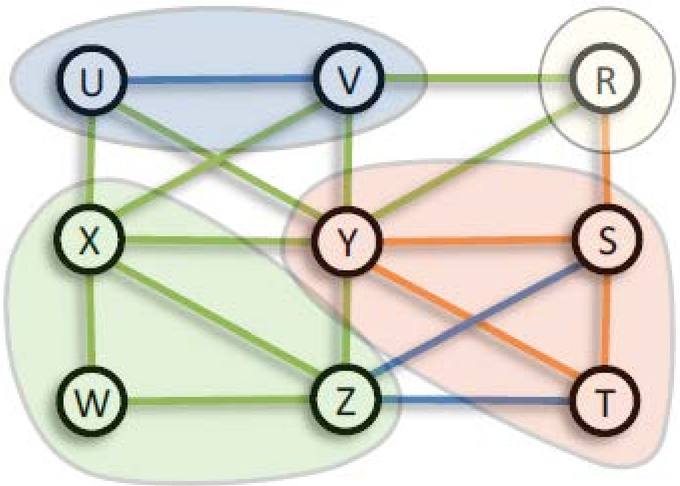
chromatic pivot



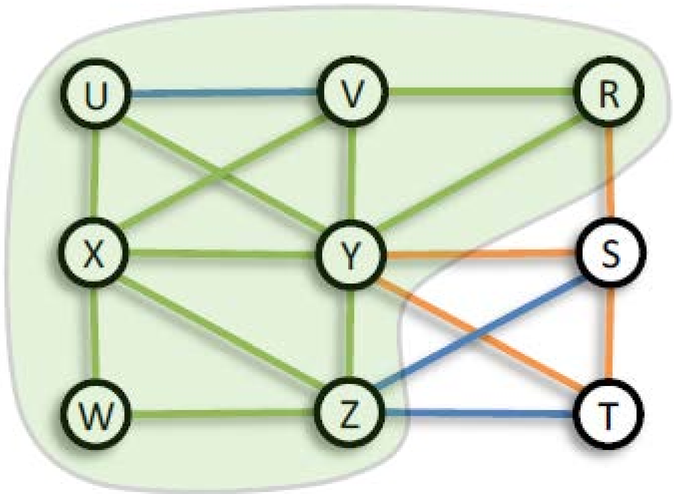
chromatic pivot



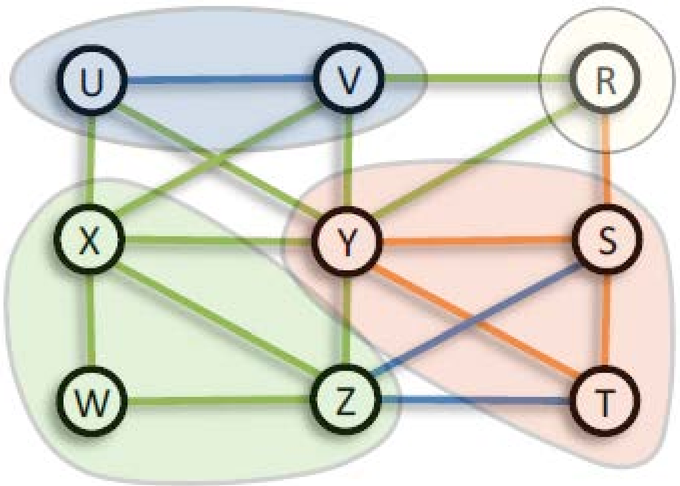
lazy chromatic pivot



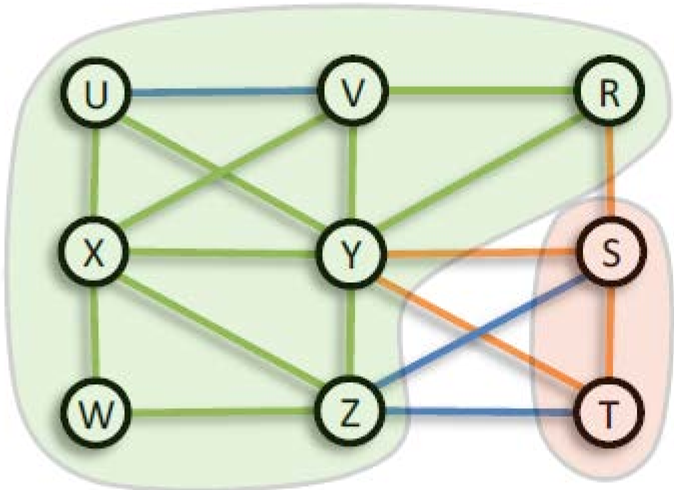
chromatic pivot



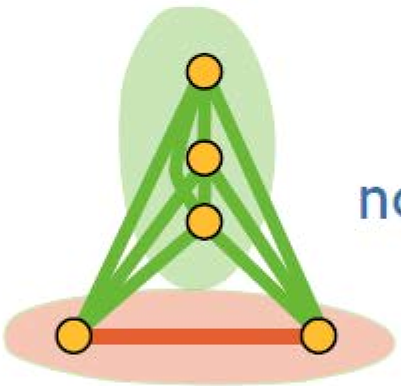
lazy chromatic pivot



chromatic pivot



lazy chromatic pivot



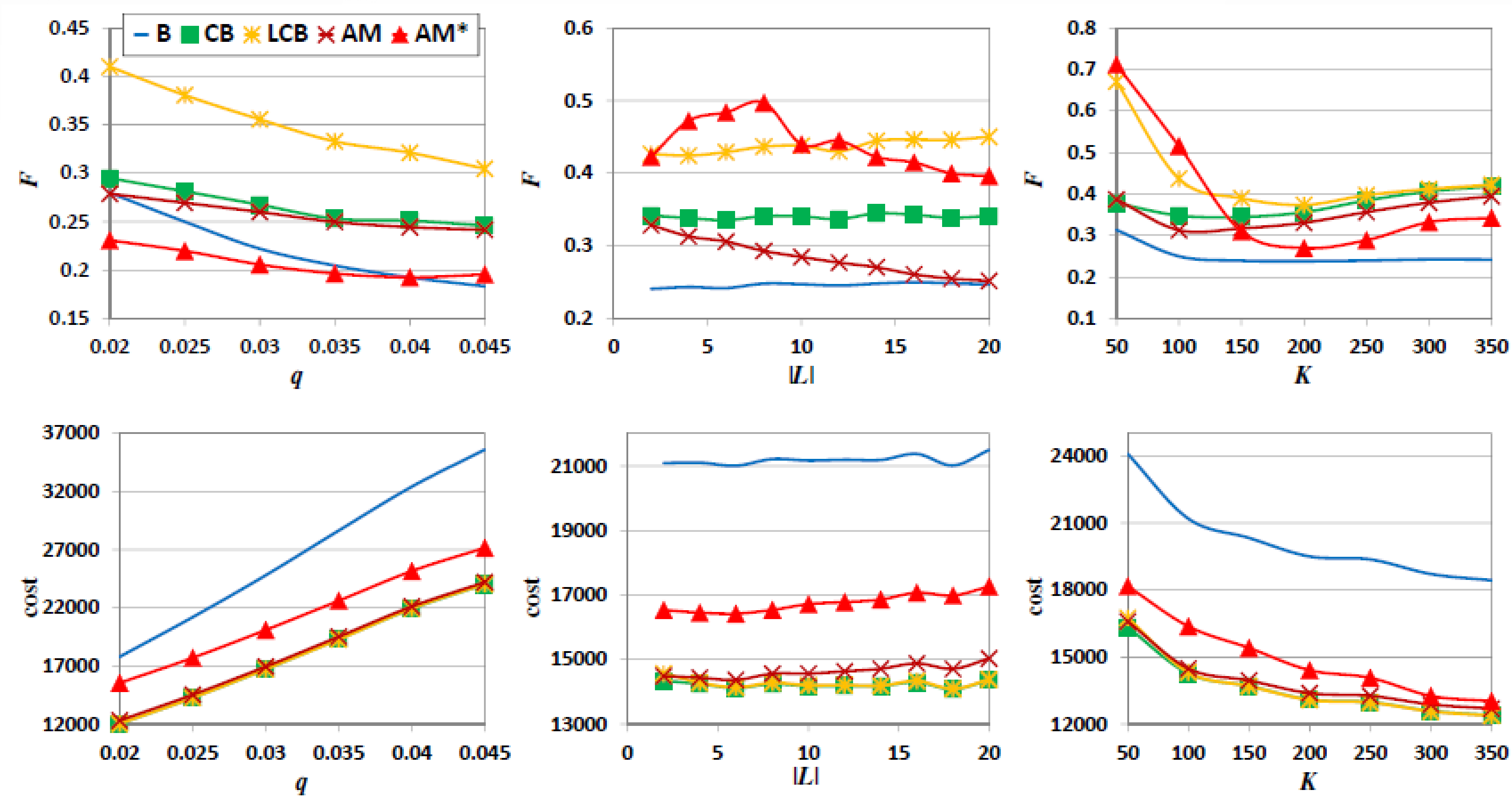
not a counter-example anymore

An algorithm for finding a predefined number of clusters

- Based on the *alternating-minimization* paradigm:
  - Start with a random clustering with K clusters
  - Keep fixed vertex-to-cluster assignments and optimally update label-to-cluster assignments
  - Keep fixed label-to-cluster assignments and optimally update vertex-to-cluster assignments
  - Alternately repeat the two steps until convergence

- Guaranteed to converge to a local minimum of the objective function

Experiments on synthetic data with planted clustering
q = level of noise, |L| = number of labels,
K = number of ground truth clusters



Experiments on real data

dataset	cost			
	B	CB	LCB	*AM*
String	163 305	160 060	155 881	156 976
Youtube	23 550 213	18 956 000	22 644 858	19 670 899
DBLP	2 260 065	1 633 149	1 678 714	2 018 952

dataset	runtime (secs)			
	B	CB	LCB	*AM*
String	1.95	2.14	5.02	82.07
Youtube	5.89	6.78	16.15	273.36
DBLP	1.79	1.89	5.23	886.79

dataset	#clusters			
	B	CB	LCB	*AM*
String	1 086	1 451	784	1 451
Youtube	568	1 078	672	1 078
DBLP	66 276	123 197	99 948	123 197

Extension: multi-chromatic correlation clustering
(to appear)

- object relations can be expressed by *more than one label*
- i.e., the input to our problem is an edge-labeled graph whose edges may have multiple labels.

- Extending chromatic correlation clustering by:

1. allowing to assign a set of labels to each cluster (instead of a single label)
2. measuring the intra-cluster label homogeneity by means of a distance function between sets of labels

From chromatic correlation clustering to multi-chromatic correlation clustering

- Chromatic correlation clustering:
  - Set of objects $V = \{v_1, \ldots, v_n\}$
  - Pairwise labeling function $\ell : V \times V \rightarrow L \cup \{l_0\}$
  - Clustering $\mathcal{C} : V \rightarrow \mathbb{N}$
  - Cluster labeling function $c\ell : \mathcal{C}[V] \rightarrow L$

$$\mathrm{cost}(\mathcal{C}, c\ell) = \sum_{\substack{(x,y) \in V \times V, \\ \mathcal{C}(x) = \mathcal{C}(y)}} (1 - \mathrm{I}[\ell(x,y) = c\ell(\mathcal{C}(x))]) + \sum_{\substack{(x,y) \in V \times V, \\ \mathcal{C}(x) \neq \mathcal{C}(y)}} \mathrm{I}[\ell(x,y) \neq l_0].$$

- Multi-chromatic correlation clustering:
  - Pairwise labeling function $\ell : V_2 \rightarrow 2^L \cup \{l_0\}$
  - Distance between set of labels $d_\ell : 2^L \cup \{l_0\} \times 2^L \cup \{l_0\} \rightarrow \mathbb{R}^+$
  - Cluster labeling function $c\ell : \mathcal{C}[V] \rightarrow 2^L$

$$\mathrm{cost}(G, \mathcal{C}, c\ell) = \sum_{\substack{(x,y) \in V_2, \\ \mathcal{C}(x) = \mathcal{C}(y)}} d_\ell(\ell(x,y), c\ell(\mathcal{C}(x)) + \sum_{\substack{(x,y) \in V_2, \\ \mathcal{C}(x) \neq \mathcal{C}(y)}} d_\ell(\ell(x,y), \{l_0\})$$

- As distance between sets of labels we adopt Hamming distance

$$d_\ell(L_1, L_2) = |L_1 \setminus L_2| + |L_2 \setminus L_1|$$

- A consequence is that inter-cluster edges cost the number of labels they have plus one

$$d_\ell(\ell(x, y), \{l_0\}) = |\ell(x, y)| + 1, \forall (x, y) \in E$$

Multi-chromatic pivot

- Pick randomly a pivot (x, y)
- Add all vertices z such that $\ell(x, y) = \ell(x, z) = \ell(y, z)$
- The cluster is assigned the set of colors $\ell(x, y)$

- approximation guarantee *6|L|(D-1)*
  - where *D* is the maximum degree

Online correlation clustering

C. Mathieu, O. Sankur, W. Schudy:
Online correlation clustering
STACS 2010

Online correlation clustering

- Vertices arrive one by one.
- The size of the input is unknown.
- Upon arrival of a vertex *v*, an online algorithm can
  - Create a new cluster *{v}*.
  - Add *v* to an existing cluster.
  - Merge any pre-existing clusters.
  - ~~Split a pre-existing cluster~~

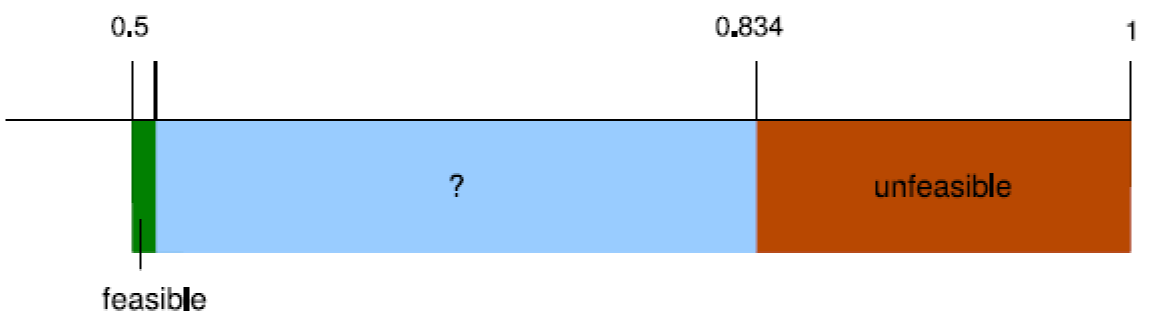
Main results

- An online algorithm is c-competitive if on any input *I* , the algorithm outputs a clustering *ALG(I)* s.t.

$$profit(ALG(I)) \geq c \cdot profit(OPT(I))$$

where *OPT(I)* is the offline optimum.

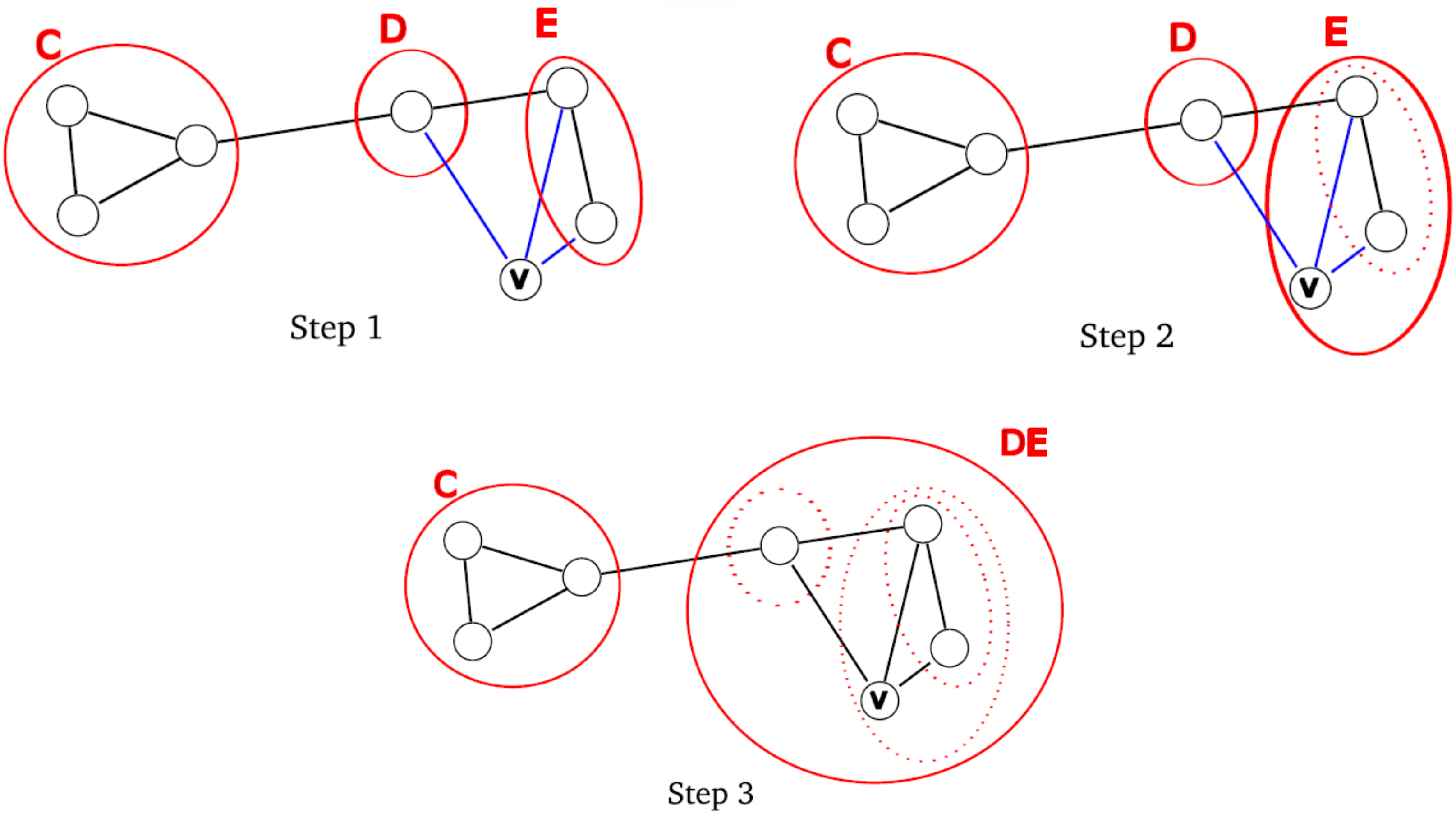
- Main results:
  - $\textsc{MinDisAgree}$ is hopeless: *O(n)*-competitive and this is proved optimal.
  - For $\textsc{MaxAgree}$
    - Greedy 0.5-competitive
    - No algorithm can be better than 0.834-competitive
    - (0.5+c)-competitive randomized algorithm



YAHOO!

Algorithm 1 Algorithm GREEDY

Upon arrival of vertex v **do**
  Put v in new cluster $\{v\}$.
  while $\exists$ clusters C, D s.t. merging C and D improves the profit **do**
    Merge C and D
  end while
end for


C
D
E
v
Step 1
C
D
E
v
Step 2
C
DE
v
Step 3

- If *profit(OPT) ≤ (1 − α)|E|*, GREEDY has competitive ratio > 0.5
- <u>IDEA:</u> design an algorithm with competitive ratio > 0.5 when

profit(OPT) > (1 − α)|E| DENSE

Algorithm 2 GREEDYORDENSE

With probability p, run GREEDY,
With probability $1 - p$, run DENSE.

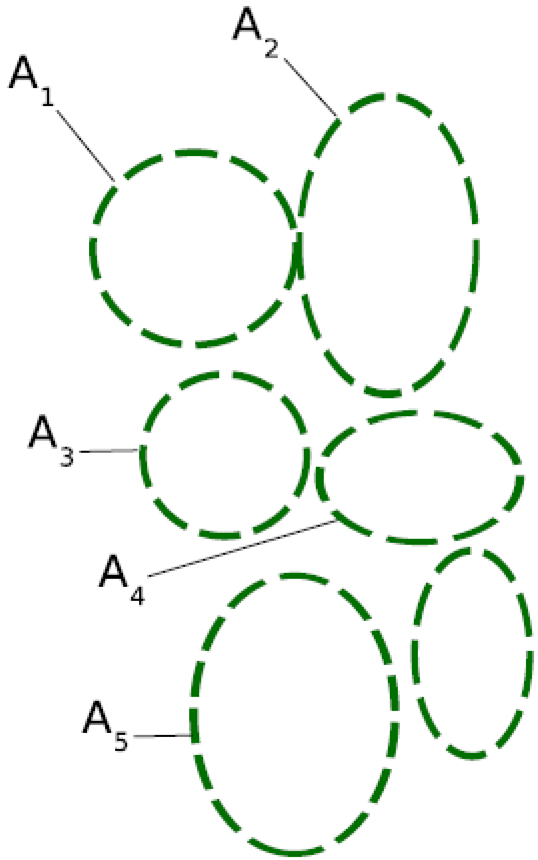
- GREEDYORDENSE is (0.5 + ε)-competitive.

Algorithm Dense

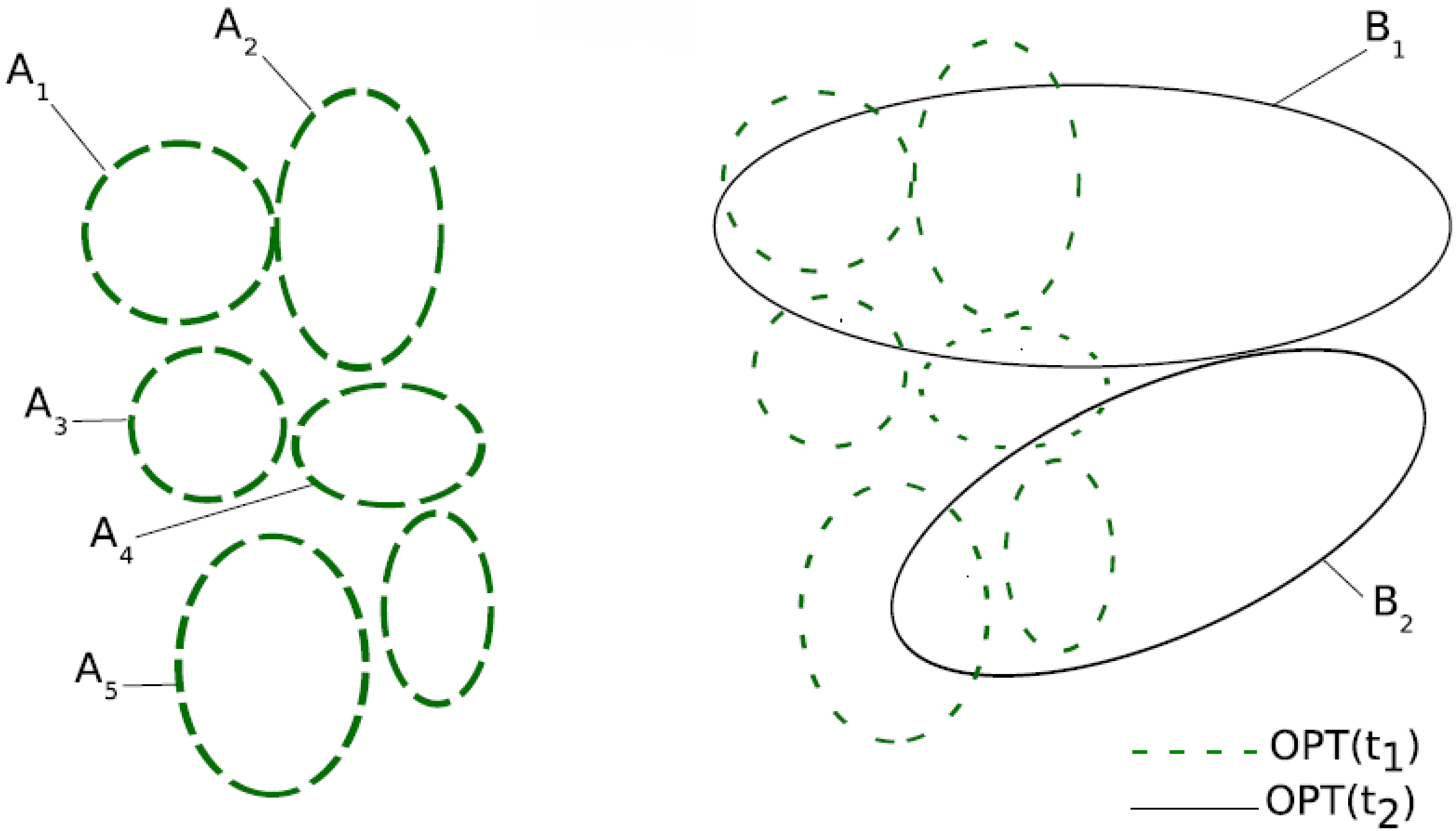
- Reminder: focus on instances where *profit(OPT) > (1 − α)|E|*
- Fix $\tau = 1.1$
- When new vertices arrive put them in a singleton cluster
- At times $t_i = \tau^i$
- Compute (near) $\mathrm{OPT}(t_i)$
- Merge clusters as explained next

YAHOO!

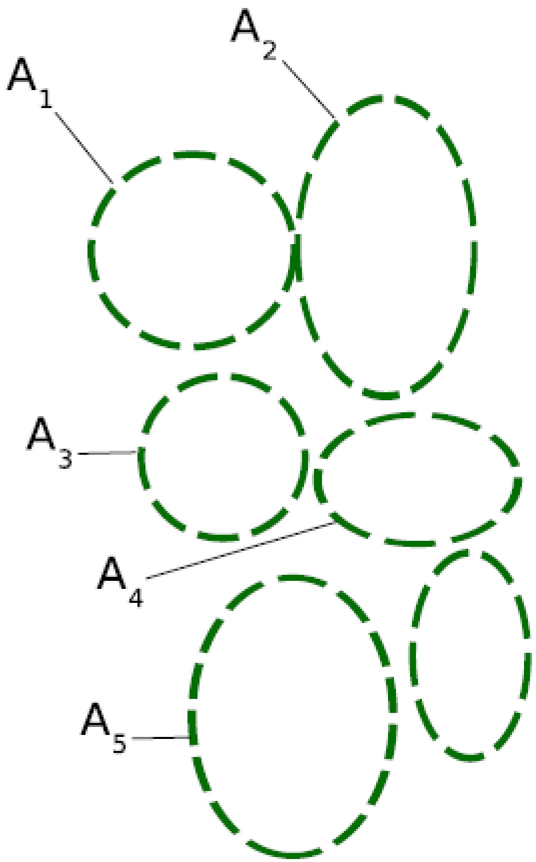
- Suppose we start with OPT at time t1.
- Until time t2, we put all new vertices to singletons.

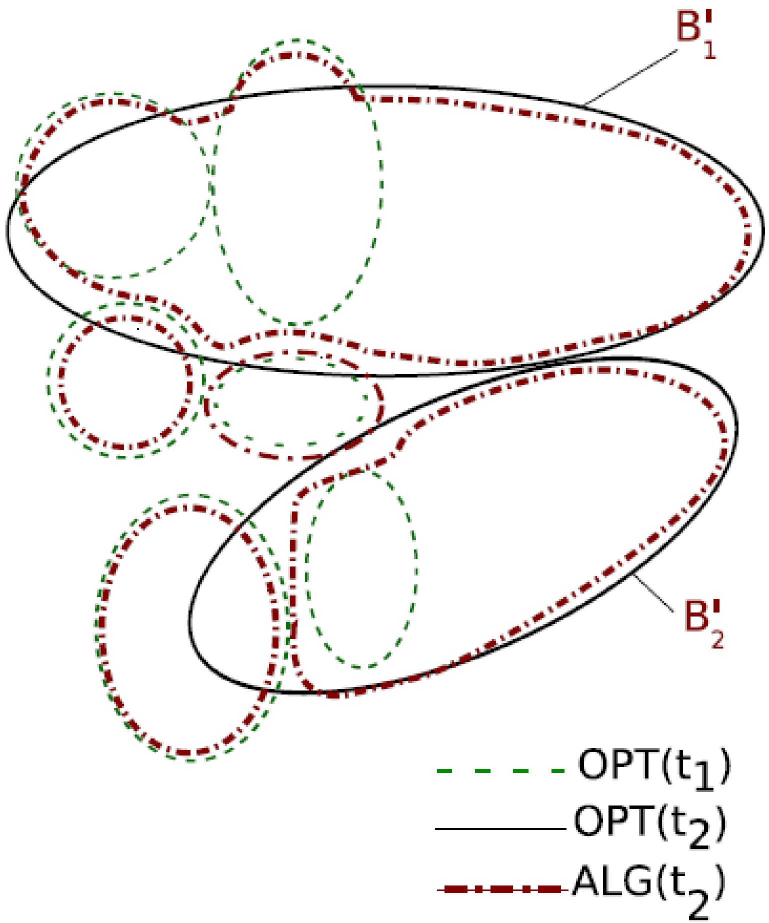


- At time t2, we run the merging procedure.
- First, compute OPT(t2).
- Then try to recreate OPT(t2).

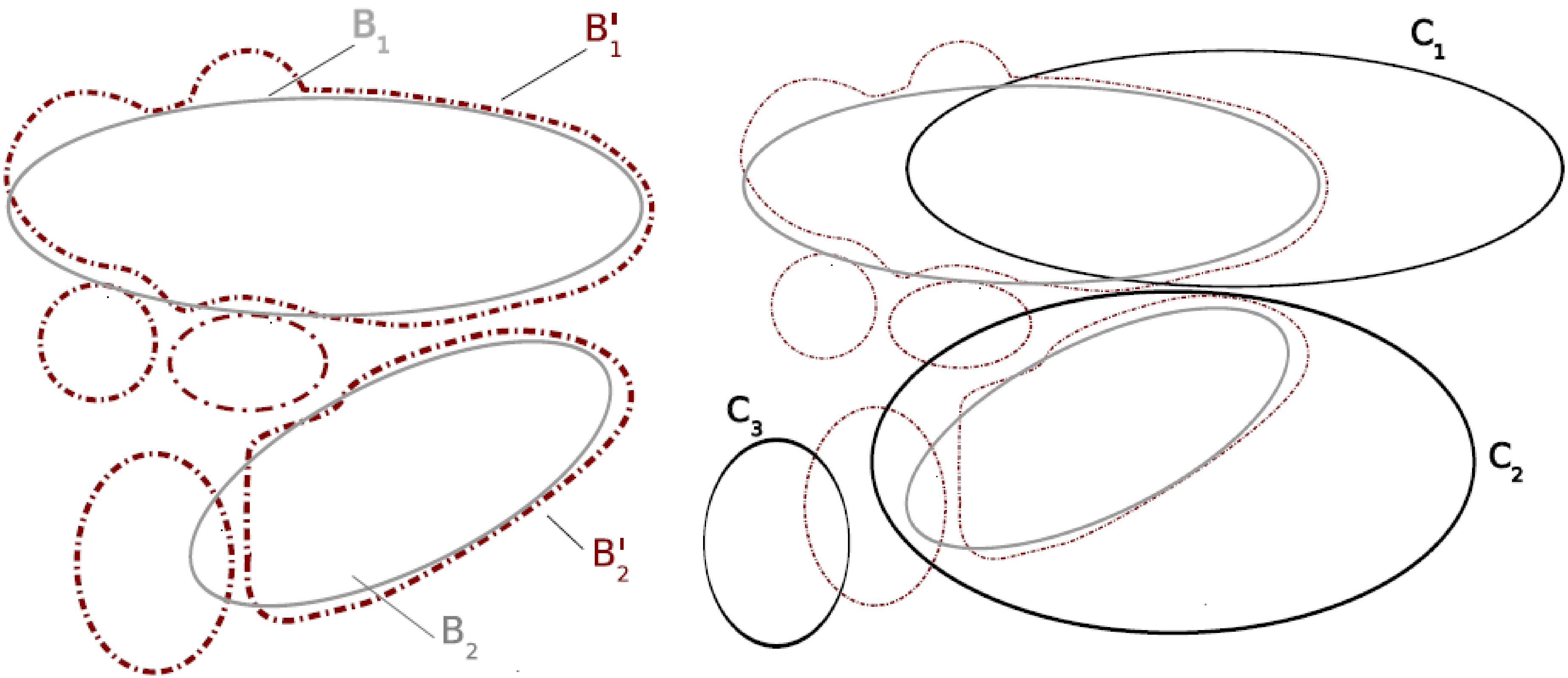


- Clusters at the previous step, that are more than half covered by a cluster in the new optimal clustering are merged in the cluster.

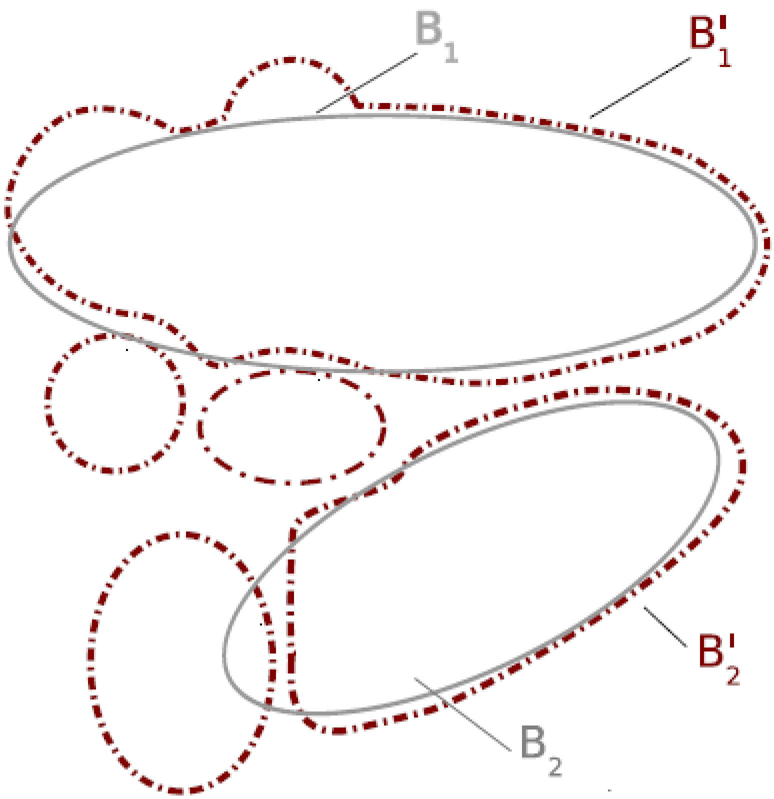


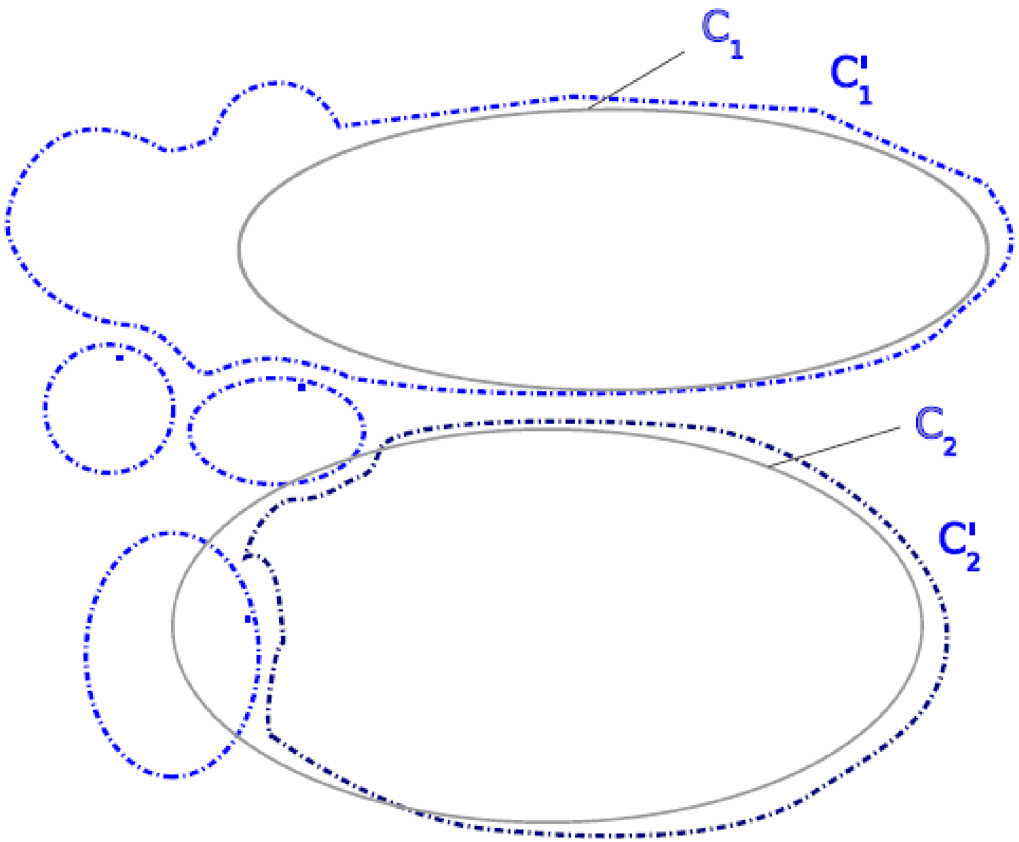


- B1 and B2 are kept as *ghost clusters*.
- At time 3, the new optimal cluster are compared to the ghost clusters at the previous step



- B1 and B2 are kept as *ghost clusters*.
- At time 3, the new optimal cluster are compared to the ghost clusters at the previous step





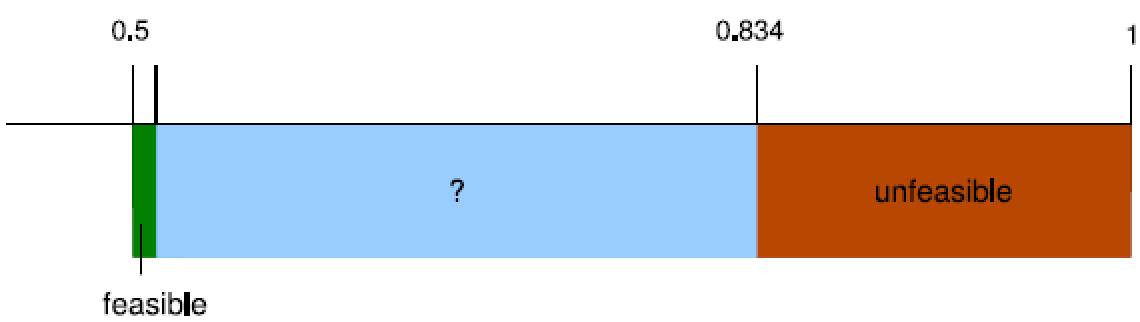
Main results

- An online algorithm is c-competitive if on any input *I* , the algorithm outputs a clustering *ALG(I)* s.t.

$$profit(ALG(I)) \geq c \cdot profit(OPT(I))$$

where *OPT(I)* is the offline optimum.

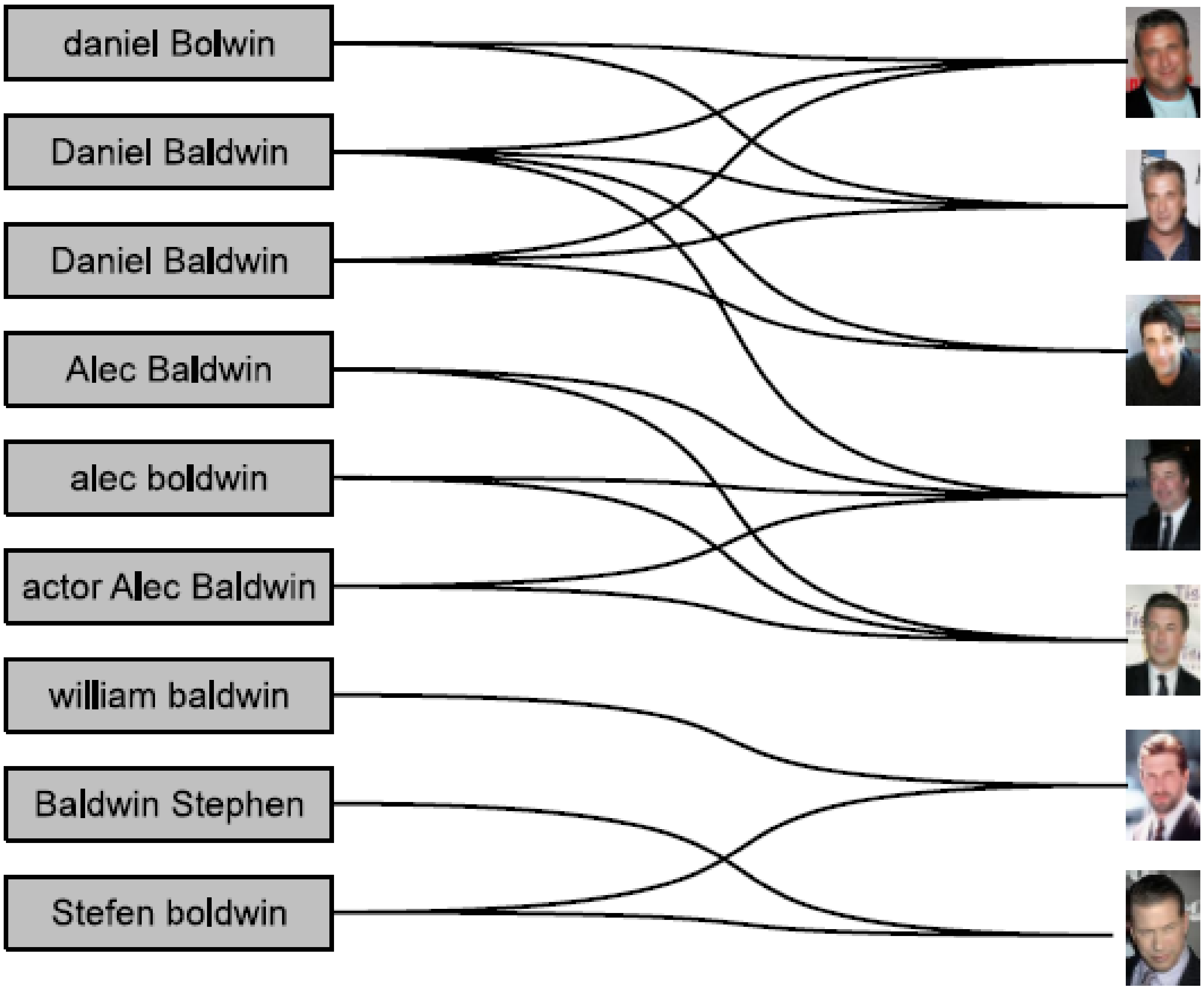
- Main results:
  - MINDISAGREE is hopeless: *O(n)*-competitive and this is proved optimal.
  - For MAXAGREE
    - Greedy 0.5-competitive
    - No algorithm can be better than 0.834-competitive
    - (0.5+c)-competitive randomized algorithm



Bipartite correlation clustering

N. Ailon, N. Avigdor-Elgrabli, E. Liberty, A. van Zuylen
Improved Approximation Algorithms for Bipartite Correlation Clustering
ESA 2011

Correlation bi-clustering

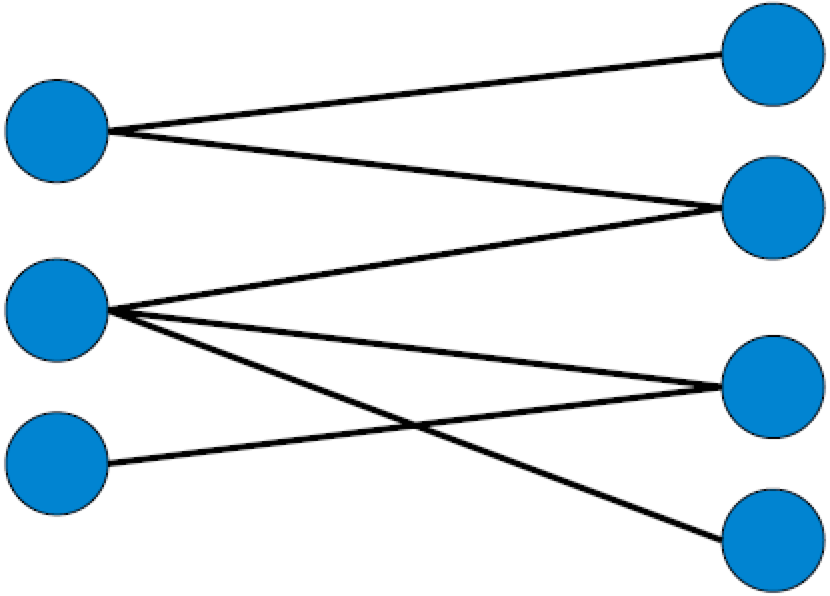


YAHOO!

Correlation bi-clustering

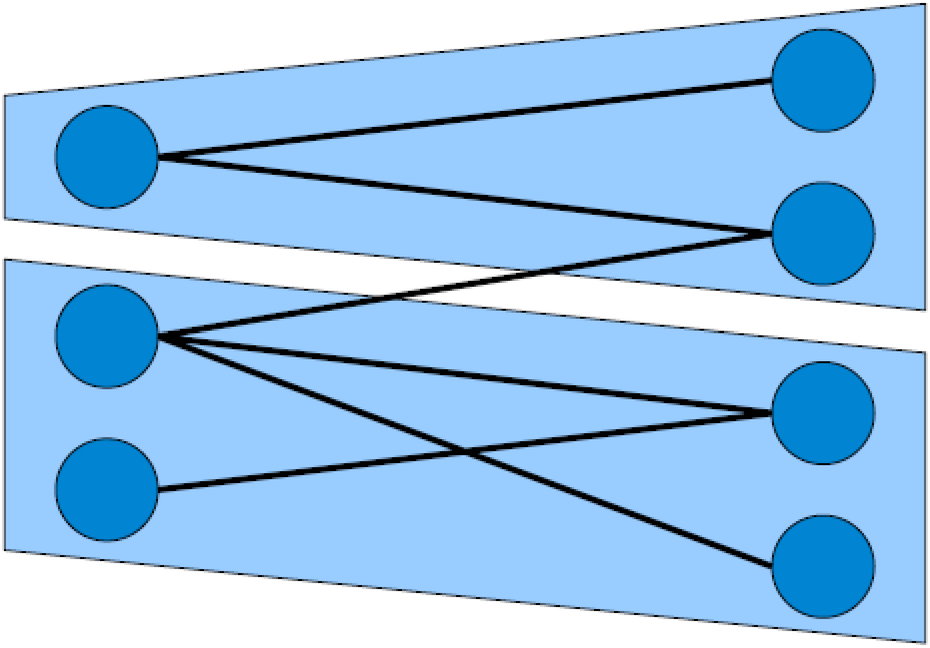
- Users – Items
- Raters – Movies
- B-cookies – User_Id
- Web Queries - URLs

Input for correlation bi-clustering



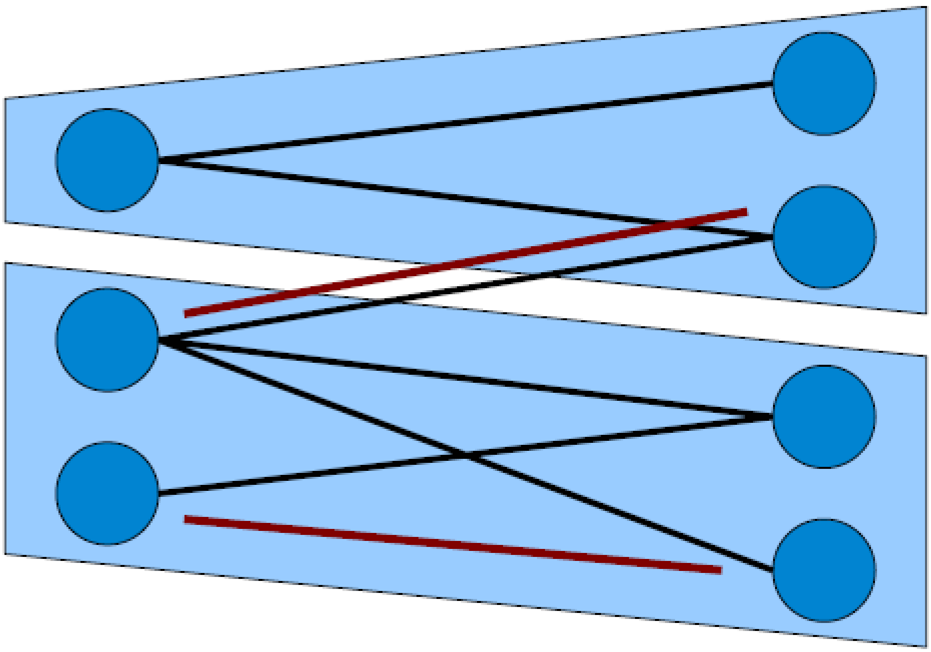
The input is an undirected unweighted bipartite graph.

Output of correlation bi-clustering



The output is a set of bi-clusters.

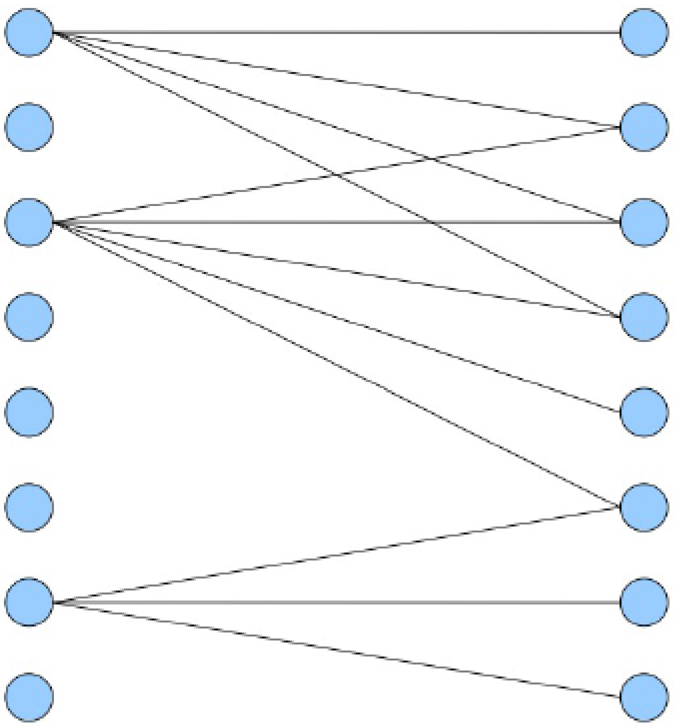
Cost of a correlation bi-clustering solution



The cost is the number of erroneous edges.

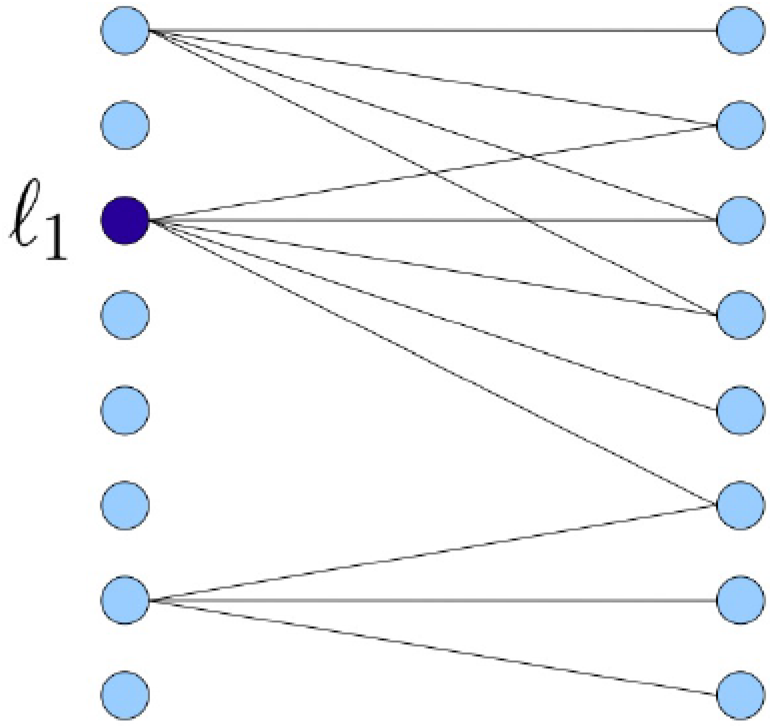
YAHOO!

PivotBiCluster



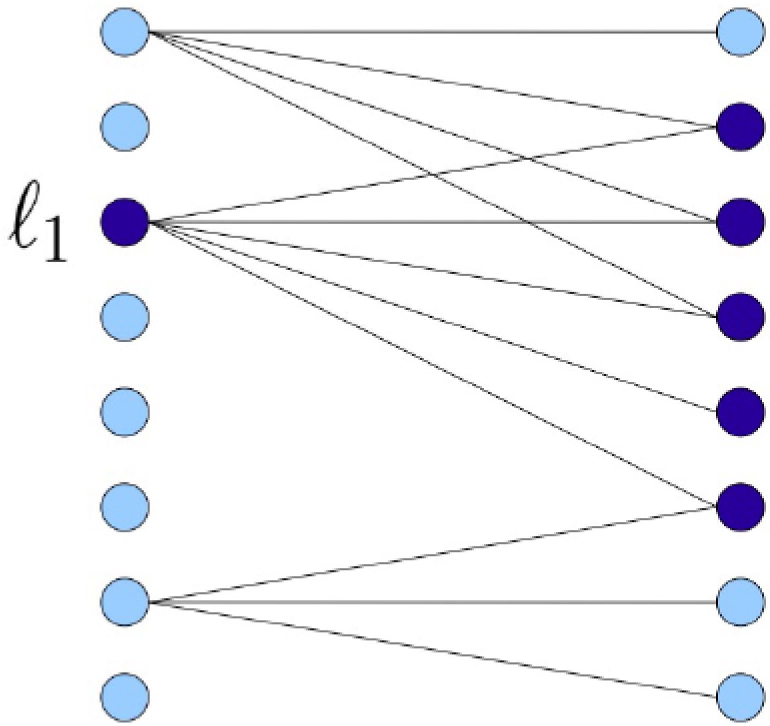
Consider the following graph

PivotBiCluster



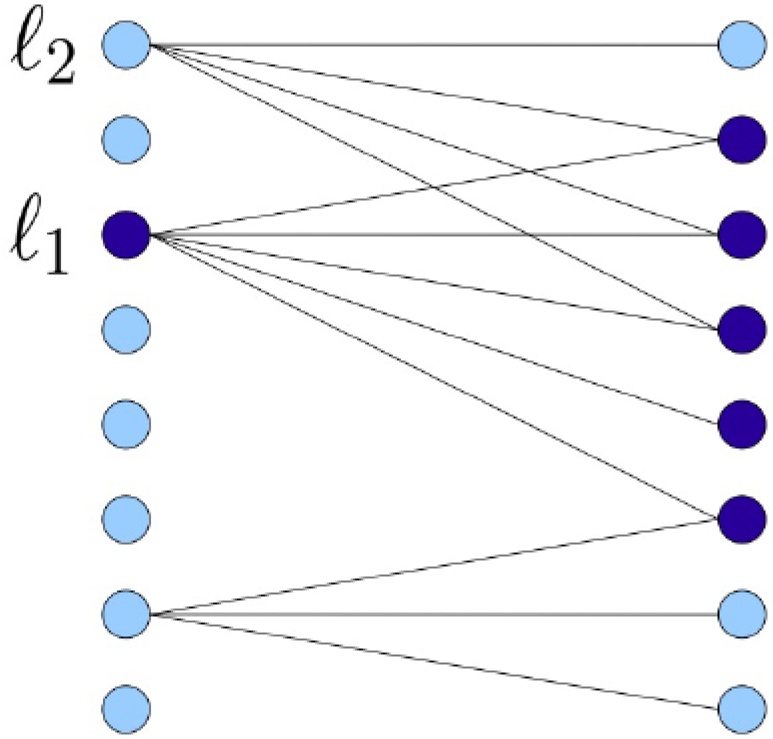
Choose ℓ_1 uniformly at random from the left side.

PivotBiCluster



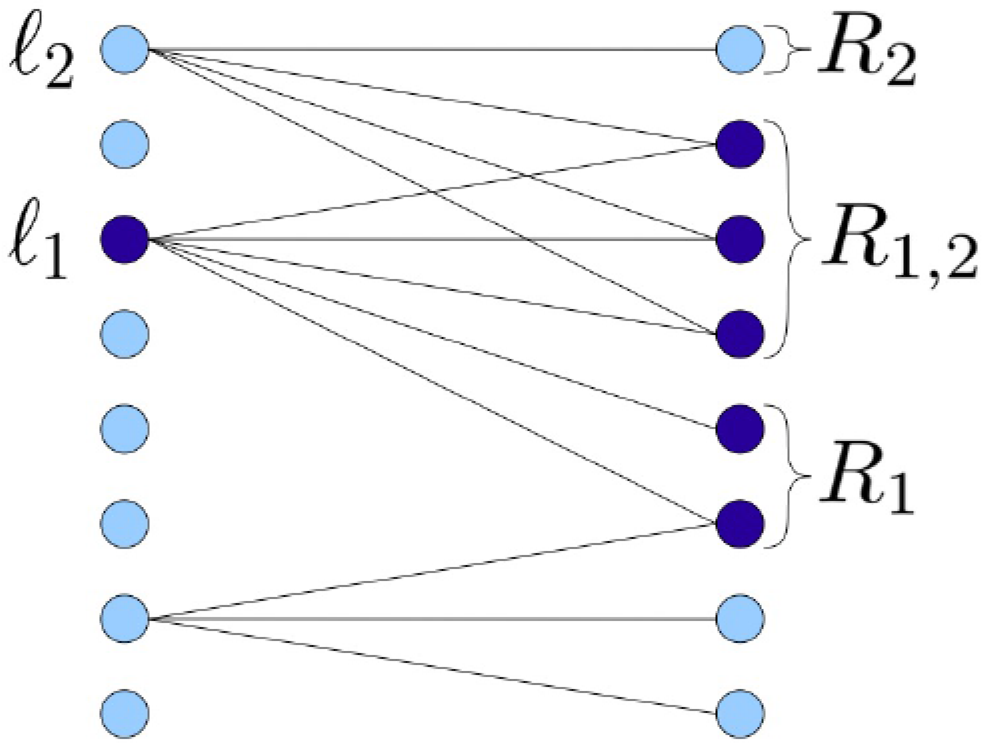
Add the neighborhood of ℓ_1 to the cluster

PivotBiCluster



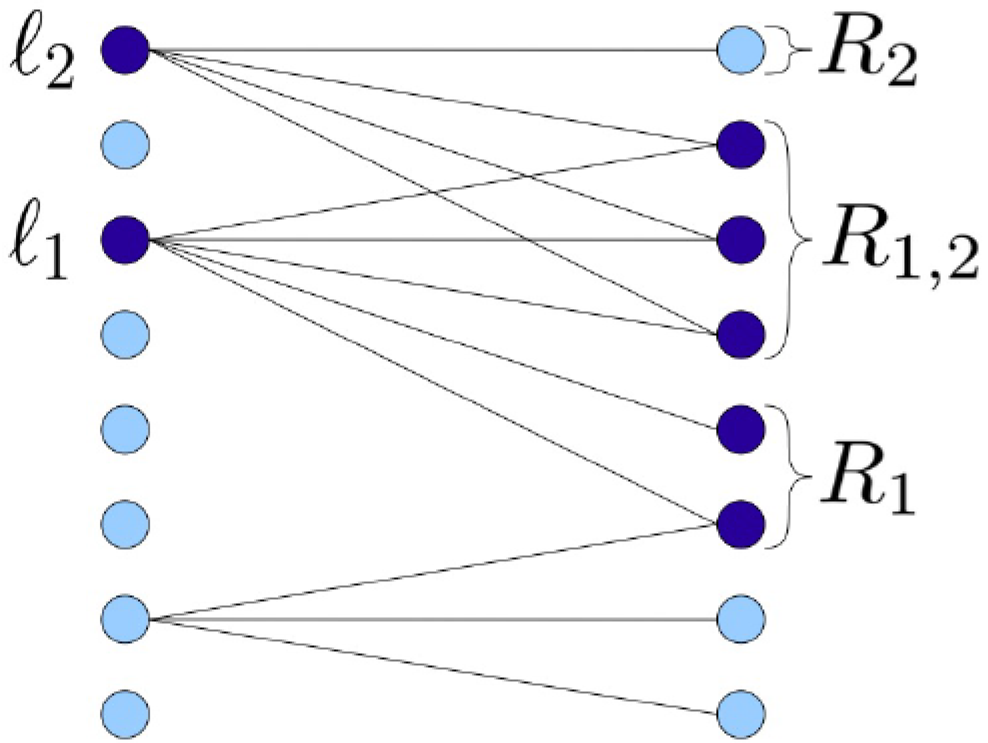
For each other node on the left (ℓ_2) do the following:

PivotBiCluster



w.p. $\min(|R_{1,2}|/|R_2|, 1)$ add ℓ_2 to the cluster if $|R_{1,2}| \geq |R_1|$.

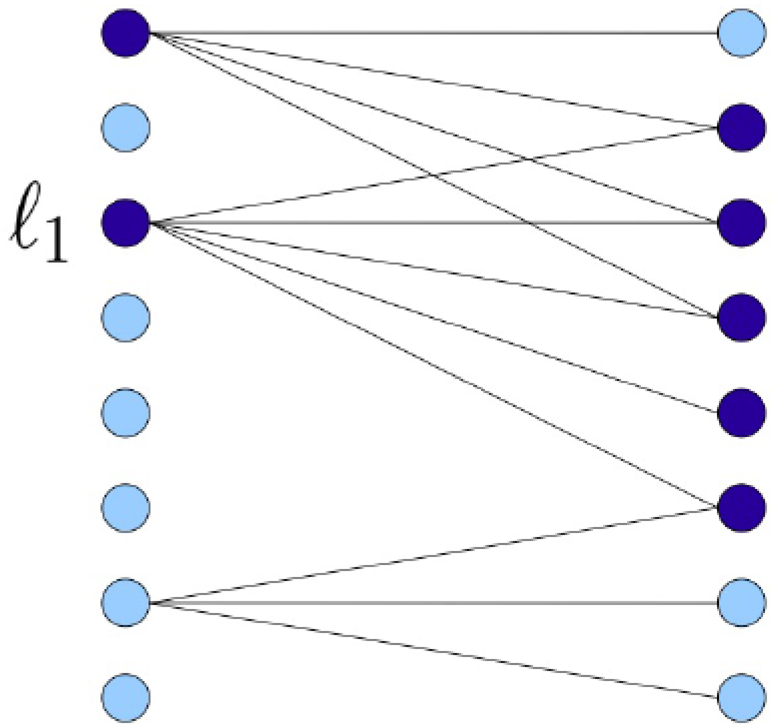
PivotBiCluster



Here ℓ_2 joins the cluster because $R_{1,2} \geq R_1$.

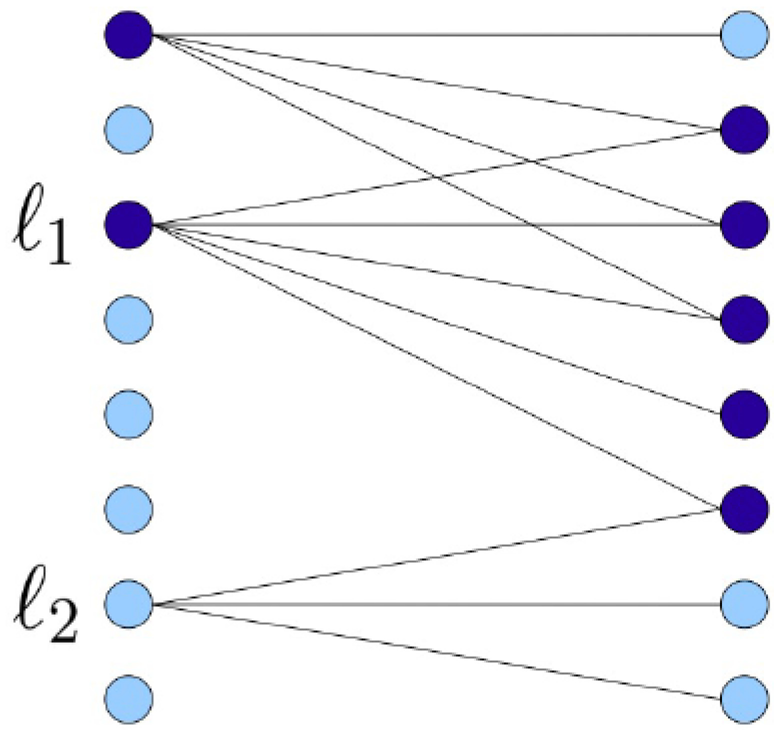
YAHOO!

PivotBiCluster



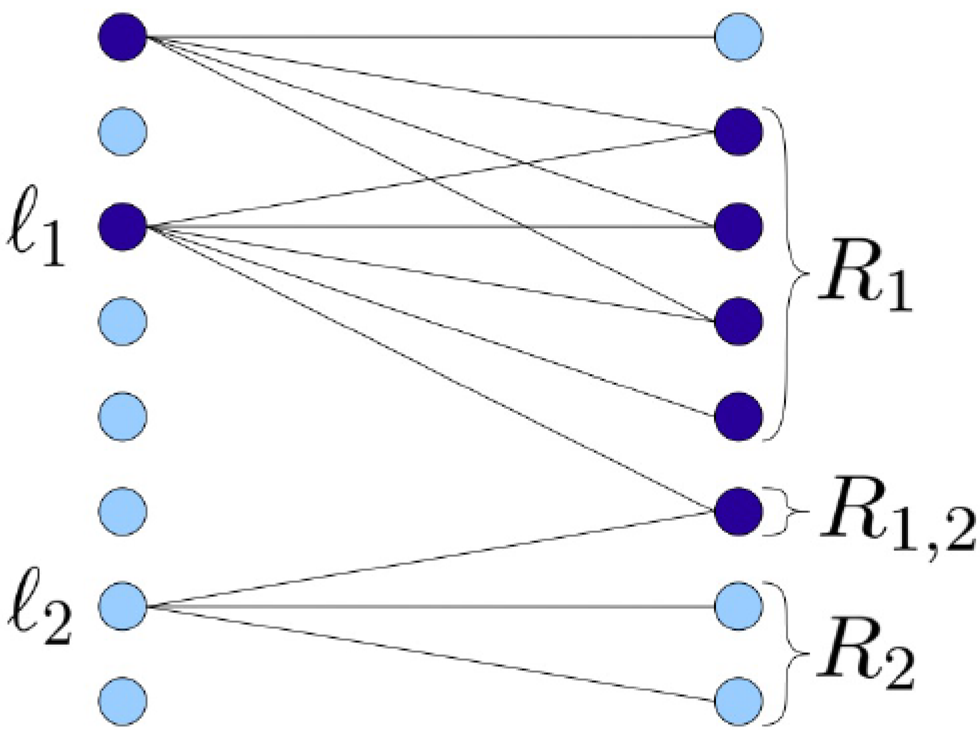
Let's consider another example

PivotBiCluster



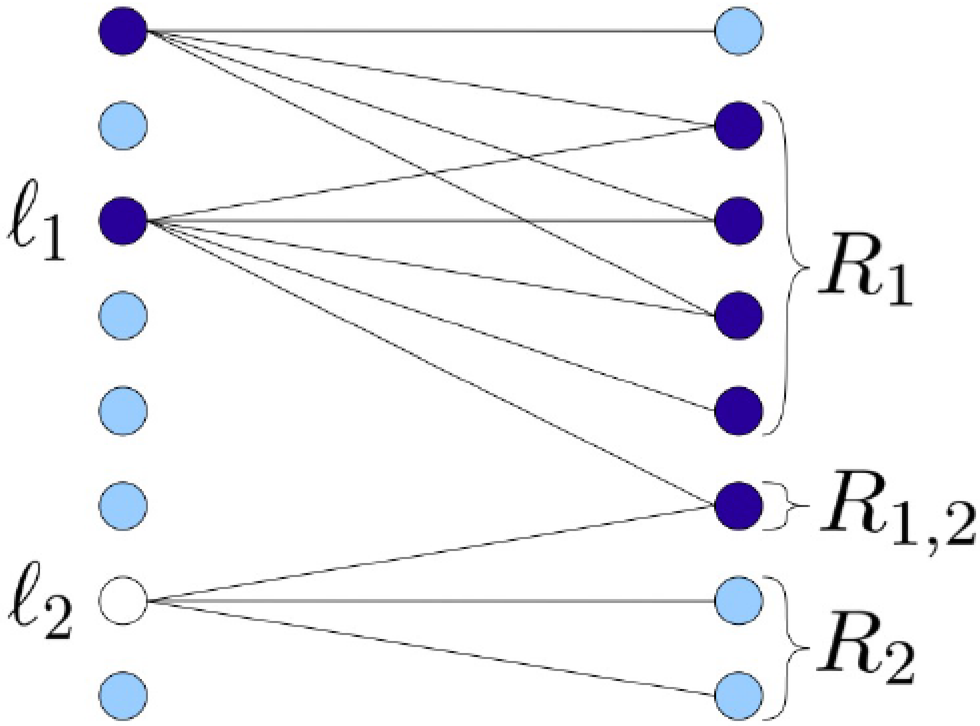
Let's consider another example

PivotBiCluster



Since $|R_{1,2}|/|R_2| = 1/2$ with probability $1/2$ we decide what to do with ℓ_2

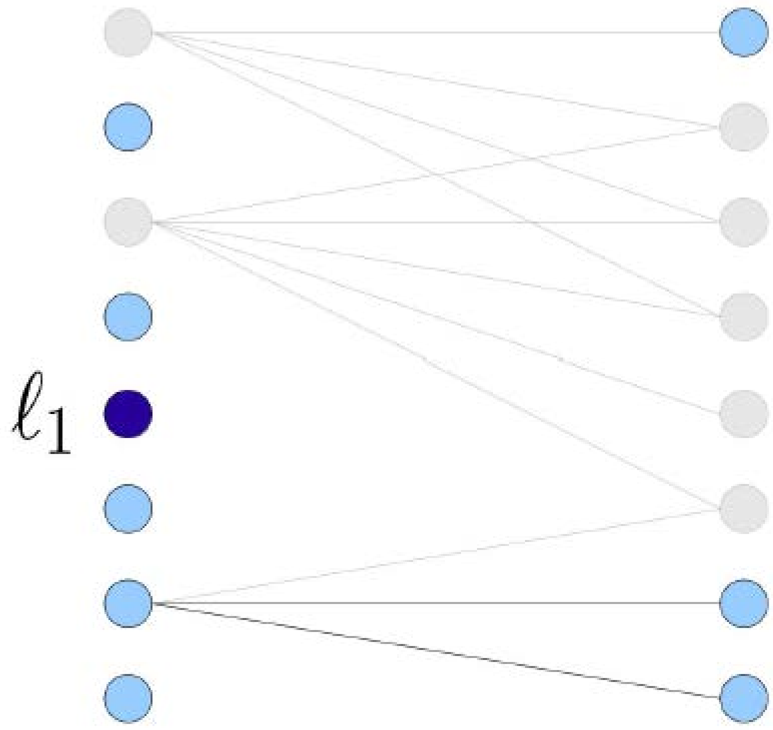
PivotBiCluster



Since $|R_{1,2}| < |R_1|$ that decision should be to make ℓ_2 a singleton

Otherwise (w.p. 1/2) we decide nothing about ℓ_2 and continue.

PivotBiCluster



We remove the clustered nodes from the graph and repeat.

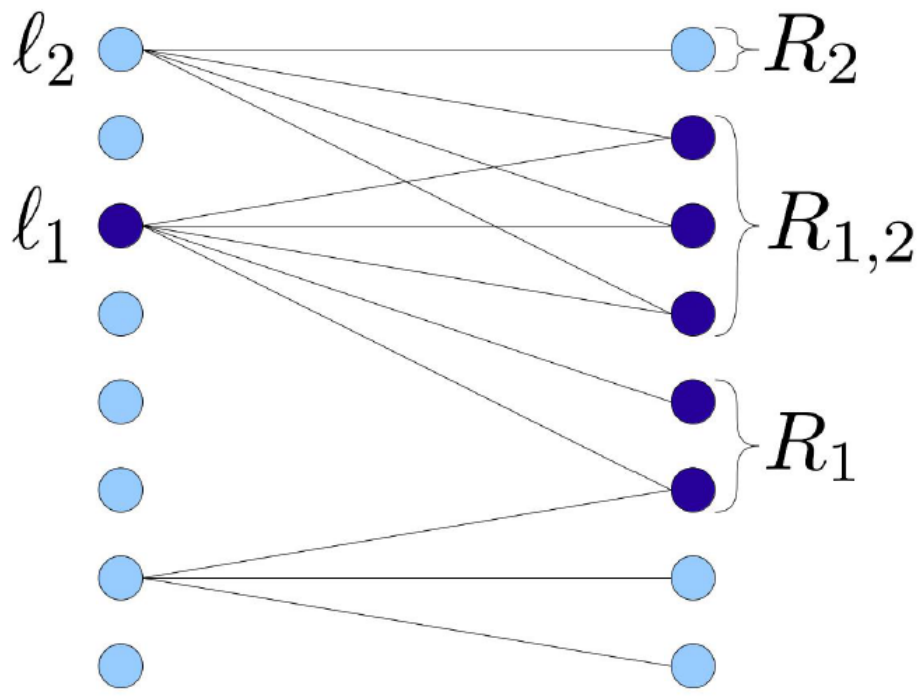
PivotBiCluster

- Let OPT denote the best possible bi-clustring of G.
- Let B be a random output of PivotBiCluster.
- Then:

$$E[cost(B)] \leq 4cost(OPT)$$

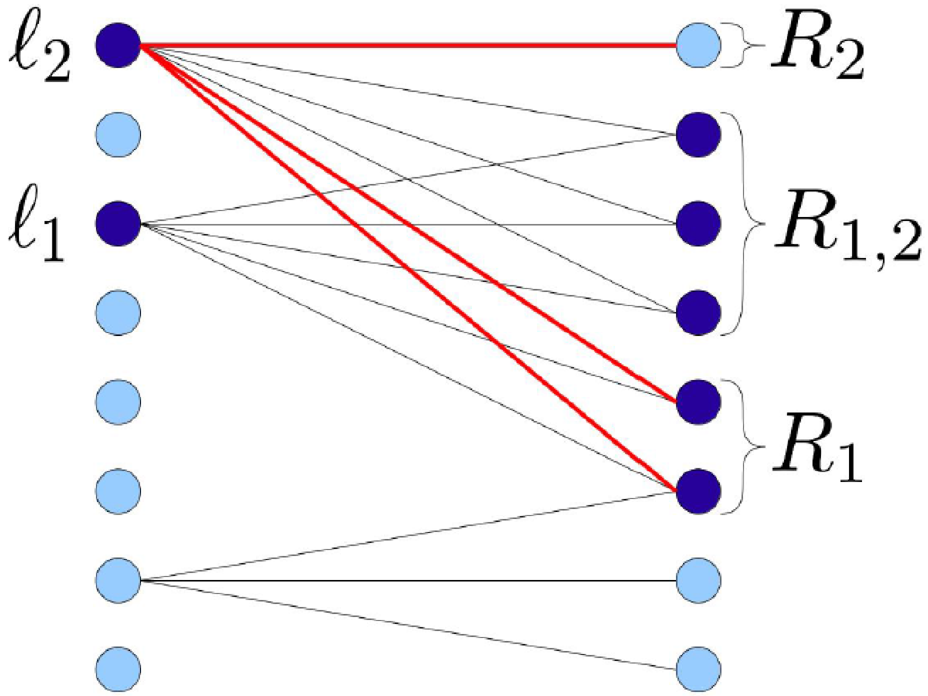
- Let's see how to prove this…

Tuples, bad events, and violated pairs



A "bad event" (X_T) happens to tuple $T = (\ell_1, \ell_2, R_1, R_{1,2}, R_2)$.

Tuples, bad events, and violated pairs



We "blame" bad event X_T for the violated (red) pairs, $\mathbb{E}[cost(T)|X_T] = 3$.

YAHOO!

Tuples, bad events, and violated pairs

- Since every violated pair can be blamed on (or colored by) one bad event happening we have:

$$\mathbb{E}_{B \sim PivotBiCluster}[cost(B)] \leq \sum_{T} q_T \cdot \mathbb{E}[cost(T) | X_T]$$

where qT denotes the probability that a bad event happened to tuple T.

- Note: the number of tuples is exponential in the size of the graph.

Proof sketch

1. We have (previous slide)

$$ALG \le \sum_T q_T \cdot \mathbb{E}[cost(T)|X_T]$$

2. Write the dual linear program

$$OPT \ge \sum_T \beta(T) \text{ s.t. constrains on } \beta(T)$$

3. Set a feasible solution $\beta(T) \leftarrow q_T f(T)$.
4. Show that:

$$\mathbb{E}[cost(T)|X_T] + E[cost(\overline{T})|X_{\overline{T}}] \le 4(f(T) + f(\overline{T}))$$

5. Which gives

$$ALG \le \sum_T q_T \cdot \mathbb{E}[cost(T)|X_T] \le 4 \sum_T q_T f(T) \le 4 \cdot OPT$$

Clustering aggregation

A. Gionis, H. Mannila, P. Tsaparas
Clustering aggregation
ICDE 2004 & TKDD

Clustering aggregation

- Many different clusterings for the same dataset!
  - Different objective functions
  - Different algorithms
  - Different number of clusters
- Which clustering is the best?
  - Aggregation: we do not need to decide, but rather find a reconciliation between different outputs

The clustering-aggregation problem

- Input
  - n objects $V = \{v1,v2,\ldots,vn\}$
  - m clusterings of the objects $C1,\ldots,Cm$

- Output
  - a single partition C, that is as close as possible to all input partitions

- How do we measure closeness of clusterings?
  - disagreement distance

YAHOO!

Disagreement distance

$$d_{u,v}(\mathcal{C}_1, \mathcal{C}_2) = \begin{cases} 1 & \text{if } \mathcal{C}_1(u) = \mathcal{C}_1(v) \text{ and } \mathcal{C}_2(u) \neq \mathcal{C}_2(v), \\ & \text{or } \mathcal{C}_1(u) \neq \mathcal{C}_1(v) \text{ and } \mathcal{C}_2(u) = \mathcal{C}_2(v), \\ 0 & \text{otherwise.} \end{cases}$$

$$d_V(\mathcal{C}_1, \mathcal{C}_2) = \sum_{(u,v) \in V \times V} d_{u,v}(\mathcal{C}_1, \mathcal{C}_2).$$

U	*C*	*P*
x_1	*1*	*1*
x_2	*1*	*2*
x_3	*2*	*1*
x_4	*3*	*3*
x_5	*3*	*4*

$d(C,P) = 3$

YAHOO!

Clustering aggregation

Problem 1 (*Clustering Aggregation*). Given a set of objects V and m clusterings $\mathcal{C}_1, \ldots, \mathcal{C}_m$ on V, compute a new clustering $\mathcal{C}$ that minimizes the total number of disagreements with all the given clusterings, that is, it minimizes

$$D(\mathcal{C}) = \sum_{i=1}^{m} d_V(\mathcal{C}_i, \mathcal{C}).$$

Why clustering aggregation?

- Clustering categorical data

U	*City*	*Profession*	*Nationality*
x_1	*New York*	*Doctor*	*U.S.*
x_2	*New York*	*Teacher*	*Canada*
x_3	*Boston*	*Doctor*	*U.S.*
x_4	*Boston*	*Teacher*	*Canada*
x_5	*Los Angeles*	*Lawer*	*Mexican*
x_6	*Los Angeles*	*Actor*	*Mexican*

- The two problems are equivalent

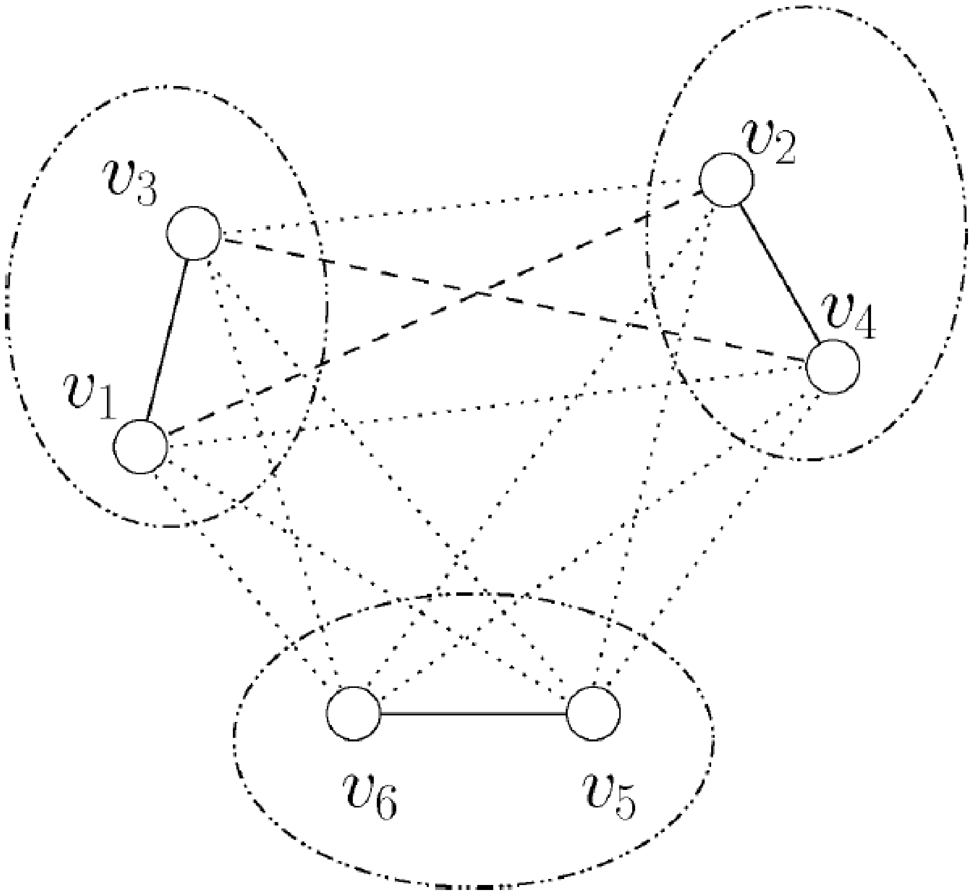
Why clustering aggregation?

- Clustering heterogenous data
  - E.g., imcomparable numeric attributes
- Identify the correct number of clusters
  - the optimization function does not require an explicit number of clusters
- Detect outliers
  - outliers are defined as points for which there is no consensus
- Improve the robustness of clustering algorithms
  - different algorithms have different weaknesses.
  - combining them can produce a better result.
- Privacy preserving clustering
  - different companies have data for the same users. They can compute an aggregate clustering without sharing the actual data.

YAHOO!

Clustering aggregation
=
Correlation clustering with fractional similarities satisfying triangle inequality

	$\mathcal{C}_1$	$\mathcal{C}_2$	$\mathcal{C}_3$
v_1	1	1	1
v_2	1	2	2
v_3	2	1	1
v_4	2	2	2
v_5	3	3	3
v_6	3	4	3



Metric property for disagreement distance

- $d(C,C) = 0$
- $d(C,C') \geq 0$ for every pair of clusterings C, C'
- $d(C,C') = d(C',C)$
- Triangle inequality?
- It is sufficient to show that for each pair of points $x,y \in V$: $d_{x,y}(C1,C3) \leq d_{x,y}(C1,C2) + d_{x,y}(C2,C3)$
- $d_{x,y}$ takes values 0/1; triangle inequality can only be violated when

  $d_{x,y}(C1,C3) = 1$ and $d_{x,y}(C1,C2) = 0$ and $d_{x,y}(C2,C3) = 0$

  - Is this possible?

YAHOO!

A 3-approximation algorithm

- The *BALLS* algorithm:
  - Sort points in increasing order of weighted degree
  - Select a point x and look at the set of points B within distance ½ of x
  - If the average distance of x to B is less than ¼ then create the cluster $B \cup \{x\}$
  - Otherwise, create a singleton cluster $\{x\}$
  - Repeat until all points are exhausted

- The *BALLS* algorithm has approximation factor *3*

Other algorithms

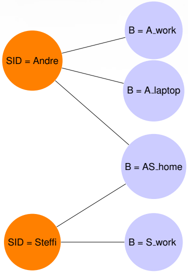
- Picking the best clustering among the input clusterings, provides 2(1-1/m) approximation ratio.
  - However, the runtime is *O(m^2n)*

- Ailon et al. (STOC 2005) propose a similar pivot-like algorithm (for correlation clustering) that for the case of similarity satisfying triangle inequality gives an approximation ratio of 2.
- For the specific case of clustering aggregation they show that chosing the best solution between their algorithm and the best of the input clusterings, yelds a solution with expected approximation ratio of 11/7.

Part III:
Scalability for real-world instances

David Garcia-Soriano
Yahoo Labs, Barcelona

Application 1: B-cookie de-duplication



- Each visit to Yahoo sites is tied to a browser B-cookie.
- We also know the hashed Yahoo IDs (SIDs) of users who are logged in.
- Many-many relationship between B-cookies and SIDs.

Problem

How to identify the set of distinct users and/or machines?

Application 1: B-cookie de-duplication (II)

- Data for a few days may occupy tens of Gbs and contain hundreds of millions of cookies/SIDs.
- It is stored across multiple machines.

Application 1: B-cookie de-duplication (II)

- Data for a few days may occupy tens of Gbs and contain hundreds of millions of cookies/SIDs.
- It is stored across multiple machines.
- We have developed a general **distributed** and **scalable** framework for correlation clustering in Hadoop.
- The problem may be modeled as correlation bi-clustering, but we choose to use standard CC for scalability reasons.

B-cookie de-duplication: graph construction

- We build a weighted graph of B-cookies.
- Assing a (multi)set $SIDs(B)$ to each B-cookie.
- The weight (similarity) of edge $B_1 \leftrightarrow B_2$ is

$$w(B_1, B_2) = J(SIDs(B_1), SIDs(B_2)) \triangleq \frac{|SIDs(B_1) \cap SIDs(B_2)|}{|SIDs(B_1) \cup SIDs(B_2)|} \in [0, 1].$$

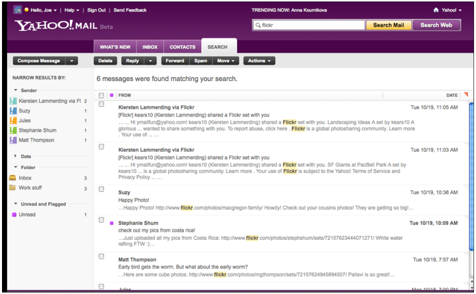
- We use correlation clustering to find $\ell : V \to \mathbb{N}$ minimizing

$$\sum_{\ell(B_1) \neq \ell(B_2)} J(B_1, B_2) + \sum_{\ell(B_1) = \ell(B_2)} [1 - J(B_1, B_2)] .$$

Application 2 (under development): Yahoo Mail

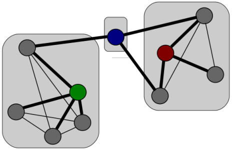
Spam detection

- Spammers tend to send groups of groups emails very similar contents.
- Correlation clustering can be applied to detect them.



How is the graph given?

How fast we can perform correlation clustering depends on how the edge information is accessed.
For simplicity we describe the case of 0-1 weights.



1. **Neighborhood oracles**: given $v \in V$, return its *positive* neighbours:

$$E^+(v) = \{w \in V \mid (v, w) \in E^+\}.$$

2. **Pairwise queries**: given a pair $v, w \in V$, determine if $(v, w) \in E^+$.

Part 1: CC with neighborhood oracles

Constructing neighborhood oracles

- Easy if the input is the graph of positive edges explicitly given.
- Otherwise, *locality sensitive hashing* may be used for certain distance metrics such as Jaccard similarity.
- This technique involves computing a set $\mathcal{H}_v$ of hashes for each node v based on its features, and building an inverted index.
- Given a node v, we can retrieve the nodes whose similarity with v exceeds a certain threshold by inspecting the nodes w with $\mathcal{H}(w) \cap \mathcal{H}(v) \neq \emptyset$.

Large-scale correlation clustering with neighborhood oracles

- We show a system that achieves an expected 3-approximation guarantee with a small number of MapReduce rounds.
- A different approach with high-probability bounds has been developed by Chierichetti, Dalvi and Kumar: *Correlation Clustering in MapReduce*, KDD'14 (Monday 25th, 2pm).

Running time of Pivot with neighborhood oracles

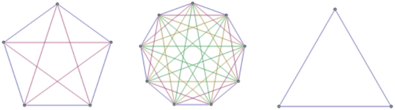
Algorithm Pivot

while $V \neq \emptyset$ **do**
  $v \leftarrow$ uniformly random node from V
  Create cluster $C_v = \{v\} \cup E^+(v)$
  $V \leftarrow V \setminus C_v$
  $E^+ \leftarrow E^+ \cap (V \times V)$

- Recall that Pivot attains an expected 3-approximation.
- Its running time is $O(n + m^+)$, i.e., linear in the size of the *positive* graph.
- Later we'll see that a certain variant runs in $O(n^{3/2})$, regardless of m^+.

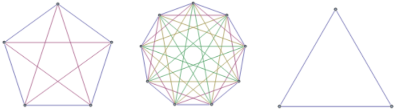
Running time of Pivot (II)

Observe that if the input graph can be partitioned into a set of cliques, Pivot actually runs in $O(n)$.



Running time of Pivot (II)

Observe that if the input graph can be partitioned into a set of cliques, Pivot actually runs in $O(n)$.



Can it be faster than $O(n+m)$ if the graph is just **close** to a union of cliques?

Running time of Pivot (III)

Theorem (Ailon and Liberty, ICALP'09)

The expected running time of Pivot with a neighborhood oracle is $O(n + OPT)$, where OPT is the cost of the optimal solution.

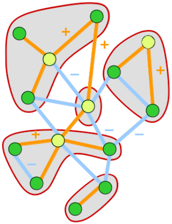
Running time of Pivot (III)

Theorem (Ailon and Liberty, ICALP'09)

The expected running time of Pivot with a neighborhood oracle is $O(n + OPT)$, where OPT is the cost of the optimal solution.

Proof:

- Each edge from a center either captures a cluster element or disagrees with the final clustering C.
- There are at most $n - 1$ edges of the first type, and $cost(C) \leq 3 \cdot OPT$ of the second.



So where's the catch?

- The algorithm needs $\Omega(n)$ memory to store the set of pivots found so far (including singleton clusters.)
- It is inherently sequential (needs to check if the new candidate pivot has connections to previous ones).
- We would like to be able to create many clusters *in parallel*.

Running Pivot in parallel

Observation #1: after *fixing a random vertex permutation* π, Pivot becomes deterministic.

Algorithm Pivot

$\pi \leftarrow$ random permutation of V
for $v \in V$ by order of π **do**
  if v is smaller than all of $E^+(v)$ according to π **then**
    Create cluster $C_v = \{v\} \cup E^+(v)$ # v is a center, $E^+(v)$ are spokes
    $V \leftarrow V \setminus C_v$
    $E^+ \leftarrow E^+ \cap (V \times V)$

Running Pivot in parallel

Observation #1: after *fixing a random vertex permutation* π, Pivot becomes deterministic.

Algorithm Pivot

$\pi \leftarrow$ random permutation of V
for $v \in V$ by order of π **do**
  if v is smaller than all of $E^+(v)$ according to π **then**
    Create cluster $C_v = \{v\} \cup E^+(v)$ # v is a center, $E^+(v)$ are spokes
    $V \leftarrow V \setminus C_v$
    $E^+ \leftarrow E^+ \cap (V \times V)$

Observation #2: If a vertex comes *before all its neighbours* (in the order defined by π), it is a cluster center. We can find them in parallel in one round.

Running Pivot in parallel

Observation #1: after *fixing a random vertex permutation* π, Pivot becomes deterministic.

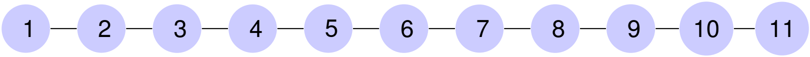
Algorithm Pivot

$\pi \leftarrow$ random permutation of V
for $v \in V$ by order of π **do**
  if v is smaller than all of $E^+(v)$ according to π **then**
    Create cluster $C_v = \{v\} \cup E^+(v)$ # v is a center, $E^+(v)$ are spokes
    $V \leftarrow V \setminus C_v$
    $E^+ \leftarrow E^+ \cap (V \times V)$

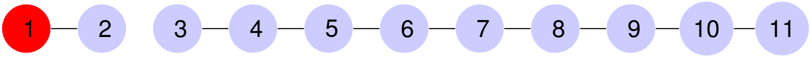
Observation #2: If a vertex comes *before all its neighbours* (in the order defined by π), it is a cluster center. We can find them in parallel in one round.

Observation #3: We should remove *remove edges as soon as possible*, i.e., when we know for sure whether or not a vertex is a cluster *center*.

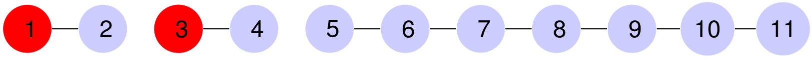
Example: clustering a line



Example: clustering a line

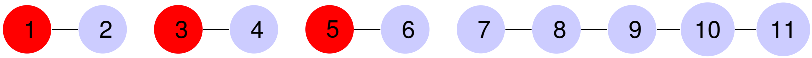


Example: clustering a line



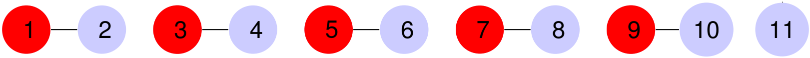
YAHOO!

Example: clustering a line

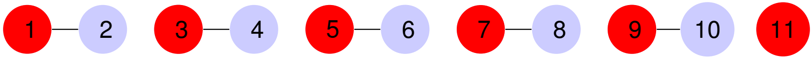


Example: clustering a line

Example: clustering a line



Example: clustering a line

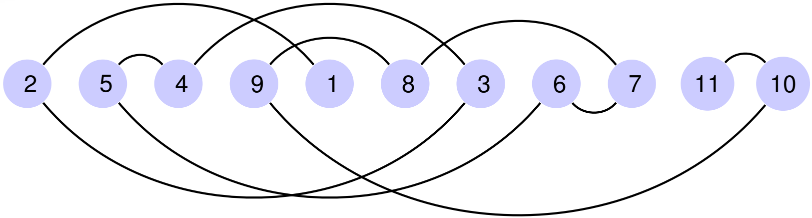


YAHOO!

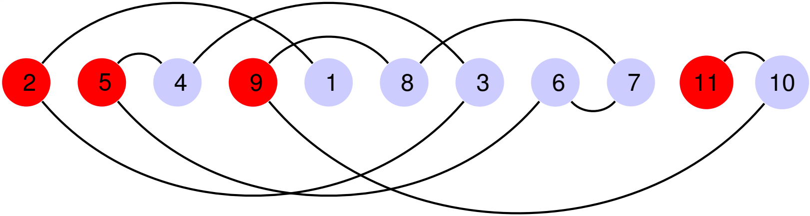
Example: clustering a line

- If $\pi = id$, a single cluster of size 2 is found per round $\Rightarrow \lceil n/2 \rceil$ rounds.
- But π was chosen at random!

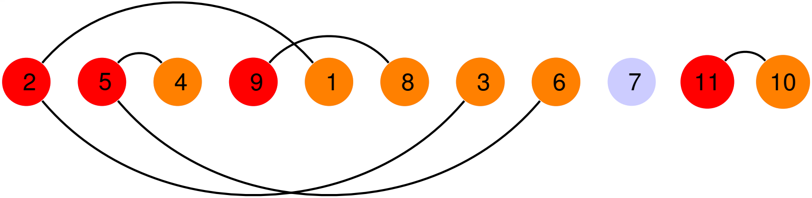
Clustering a line: random permutation



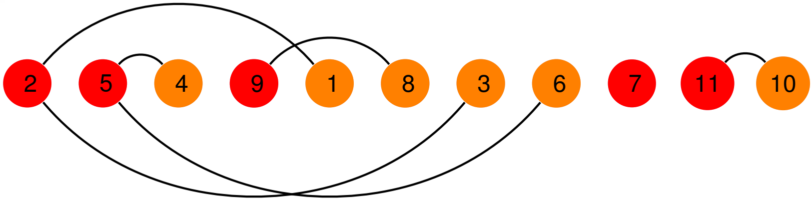
Clustering a line: random permutation



Clustering a line: random permutation



Clustering a line: random permutation



YAHOO!

Clustering a line: random permutation (II)

Some intuition:

- For a line, we expect to find 1/3 of the vertices to be pivots in the first round.
- The ”longest dependency chain“ has expected size $O(\log n)$.
- Thus we expect to cluster the line in about $\log n$ rounds.

Pseudocode for $\mathrm{ParallelPivot}$

```
Pick a random bijection π : V → |V|          # π encodes a random vertex permutation
C = ∅                                         # C is the set of vertices known to be cluster centers
S = ∅                                         # S is the set of vertices known not to be cluster centers
E = E⁺ ∩ {(i, j) | π(i) < π(j)}               # Only keep "+" edges respecting the permutation order
while C ∪ S ≠ V do
    # For each round, pick pivots in parallel and update C, S and E.
    for i ∈ V \ (C ∪ S) do
        # i's status is unknown
        N(i) = {j ∈ V | (i, j) ∈ E}           # Remaining neighbourhood of i
        if N(i) = ∅ then
            # i has no smaller neighbour left; it is a cluster center.
            # Also, none of the remaining neighbours of i is a center (but they may be assigned to another center).
            C = C ∪ {i}
            S = S ∪ N(i)
            E = E \ E({i} ∪ N(i))
```

- Each vertex can be a cluster *center* or a *spoke* (attached to a center).
- When a vertex finds out about its own status, it notifies its neighbours.
- Otherwise it asks about the status of the neighbours it needs to know.

$\mathrm{ParallelPivot}$: analysis

We obtain the exact same clustering that Pivot would find for a given vertex permutation π. Hence the same approximation guarantees hold.

The ith round (iteration of the **while** loop) requires $O(n + m_i)$ work, where $m_i = |E^+|$ is the number of edges remaining (which is strictly decreasing).

Question: How many rounds before termination?

Pivot and Maximal Independent Sets (MISs)

Focus on the set of cluster centers found:

Algorithm Pivot

$\pi \leftarrow$ random permutation of V
$C \leftarrow \emptyset$
for $v \in V$ in order of π **do**
  if v has no earlier neighbours in C **then**
    $C \leftarrow C \cup \{v\}$
    $C_v = \{v\} \cup E^+(v)$ # v is a center, $E^+(v)$ are spokes
    $V \leftarrow V \setminus C_v$

- C is an *independent set*: there are no edges between two centers.
- It is also *maximal*: cannot be extended by adding more vertices to C.
- Finding set of pivots $\equiv$ finding a *lexicographically smallest* MIS (after applying π).

Lexicographically Smallest MIS

- The lexicographically smallest MIS is *P*-hard to compute [Cook'67].
- This means that it is very unlikely to be parallelizable.
- Bad news?

Lexicographically Smallest MIS

- The lexicographically smallest MIS is *P*-hard to compute [Cook'67].
- This means that it is very unlikely to be parallelizable.
- Bad news?

- Recall that π is not an arbitrary permutation, but was chosen at *random*.
- For this case, a result of Luby (STOC'85) implies that the number of rounds of Pivot is $O(\log n)$ in expectation. ✓

MapReduce implementation details

- Each round of $\mathrm{ParallelPivot}$ uses two MapReduce jobs.
- Each vertex uses key-value pairs to send messages to its neighbours whenever it discovers that it is/isn't a cluster center.
- These two rounds do not need to be separated.

B-cookie de-duplication: some figures

- We take data for a few weeks.
- The graph can be built in 3 hours.
- Our system computes a high-quality clustering in 25 minutes, after 12 Map-Reduce rounds.
- The average number of erroneous edges per vertex (in the CC measure) is less than 0.2.
- The maximum cluster size is 68 and the average size among non-singletons is 2.89.
- For a complete evaluation we wold need some ground truth data.

Part 2: CC with pairwise queries

Correlation clustering with pairwise queries

Pairwise queries are useful when we don’t have an explicit input graph.

Correlation clustering with pairwise queries

Pairwise queries are useful when we don't have an explicit input graph.

Problem

Making all $\binom{n}{2}$ pairwise queries may be too costly to compute or store.

Can we get approximate solutions with fewer queries?

Correlation clustering with pairwise queries

Pairwise queries are useful when we don't have an explicit input graph.

Problem

Making all $\binom{n}{2}$ pairwise queries may be too costly to compute or store.

Can we get approximate solutions with fewer queries?

Constant-factor approximations require $\Omega(n^2)$ pairwise queries...

Query complexity/accuracy tradeoff

Theorem

With a "budget" of q queries, we can find a clustering C with $cost(C) \leq 3 \cdot OPT + \frac{n^2}{q}$ in time $O(nq)$.

This is nearly optimal.

Query complexity/accuracy tradeoff

Theorem

With a "budget" of q queries, we can find a clustering C with $cost(C) \leq 3 \cdot OPT + \frac{n^2}{q}$ in time $O(nq)$.

This is nearly optimal.

- We call this a $(3, \varepsilon)$ approximation (where $\varepsilon = \frac{1}{q}$).
- Restating, we can find a $(3, \varepsilon)$-approximtion in time $O(n/\varepsilon)$.
- This allows to find good clusterings up to a fixed an accuracy threshold ε.
- We can use this result about *pairwise* queries to give a faster $O(1)$-approximation algorithm for *neighborhood* queries that runs in $O(n^{3/2})$.

This result is a consequence of the existence of **local algorithms** for correlation clustering.

Bonchi, García-Soriano, Kutzkov: *Local correlation clustering*, arXiv:1312.5105.

Local correlation clustering (LCC)

Definition

A clustering algorithm $\mathcal{A}$ is said to be *local* with time complexity t if having oracle access to any graph G, and taking as input $|V(G)|$ and a vertex $v \in V(G)$, $\mathcal{A}$ returns a cluster label $\mathcal{A}^G(v)$ in time $O(t)$. Algorithm $\mathcal{A}$ implicitly defines a clustering, described by the labelling $\ell(v) = \mathcal{A}^G(v)$.

- Each vertex queries t edges.
- Outputs a label identifying its own cluster in time $O(t)$.

LCC $\rightarrow$ explicit clustering

An LCC algorithm can output a explicit clustering by:

1. Computing $\ell(v)$ for each v in time $O(t)$;
2. Putting together all vertices with the same label ℓ (in $O(n)$).

Total time: $O(nt)$.

LCC $\to$ explicit clustering

An LCC algorithm can output a explicit clustering by:

1. Computing $\ell(v)$ for each v in time $O(t)$;
2. Putting together all vertices with the same label ℓ (in $O(n)$).

Total time: $O(nt)$.
In fact we can use LCC to cluster the part of the graph we're interested in without having to cluster the whole graph.

LCC $\rightarrow$ Local clustering reconstruction

Queries of the form “are x, y in the same cluster”? can be answered in time $O(t)$.

- How: compute $\ell(x)$ and $\ell(y)$ in $O(t)$, and check for equality.
- No need to partition the whole graph!
- This is is like “correcting” the missing/extraneous edges in the input data on the fly.
- It fits into the paradigm of “property-preserving data reconstruction” (Ailon, Chazelle, Seshadhri, Liu’08).

LCC $\rightarrow$ Distributed clustering

The computation can be distributed:

1. We can assign vertices to diffent processors.
2. Each processor computes $\ell(v)$ in time $O(t)$.
3. All processors must share the same source of randomness.

LCC $\to$ Streaming clustering

Edge streaming model: edges arrive in arbitrary order.

1. For a fixed random seed, the set of v's neighbours the LCC can query has size at most 2^t.
2. This set can be compute before any edge arrives.
3. We only need to store $O(n \cdot 2^t)$ edges (this can be improved further.)

This has applications in clustering dynamic graphs.

LCC $\rightarrow$ Quick cluster edit distance estimators

The *cluster edit distance* of a graph is the smallest number of edges to change for it to admit a perfect clustering (i.e., a union of cliques). Equivalently, it is the cost of the optimal correlation clustering.

- We can estimate the cluster edit distance by sampling random pairs of vertices and checking whether $\ell(v) = \ell(w)$.
- This also gives *property testers* for clusterability.
- This allows us to quickly reject instances where even the optimal clustering is too bad.
- Another application may be in quickly evaluating the impact of decisions of a clustering algorithm.

Local correlation clustering: results

Theorem

Given $\varepsilon \in (0, 1)$, a $(3, \varepsilon)$-approximate clustering can be found locally in time $O(1/\varepsilon)$ per vertex, (after $O(1/\varepsilon^2)$ preprocessing.) Moreover, finding an $(O(1), \varepsilon)$-approximation with constant success probability requires $\Omega(1/\varepsilon)$ queries.

This is particularly useful where the graph contains a relatively small number of “dominant” clusters.

Local correlation clustering: algorithm

Algorithm $\mathrm{LocalCluster}(v, \varepsilon)$

$P \leftarrow \mathrm{FindGoodPivots}(\varepsilon)$
return $\mathrm{FindCluster}(v, P)$

Algorithm $\mathrm{FindCluster}(v, P)$

if $v \notin E^+(P)$ **then**
  return v
else
  $i \leftarrow \min\{j \mid v \in E^+(P_j)\}$;
  return P_i

YAHOO!

Algorithm $\mathrm{FindGoodPivots}(\varepsilon)$

for $i \in [16]$ **do**
  $P^i \leftarrow \mathrm{FindPivots}(\varepsilon/12)$;
  $\tilde{d}^i \leftarrow$ estimate of the cost of P^i with $O(1/\varepsilon)$ local clustering calls
$j \leftarrow \arg\min\{\tilde{d}^i \mid i \in [16]\}$
return P^j

Algorithm $\mathrm{FindPivots}(\varepsilon)$

$Q \leftarrow$ random sample of $O(1/\varepsilon)$ vertices.
$P \leftarrow$ [] (empty sequence)
for $v \in Q$ **do**
  if $\mathrm{FindCluster}(v, P) = v$ **then**
    append v to P
return P

Part IV: Challenges and directions for future research

Edo Liberty
Yahoo Labs, NYC

Future challenges

- Can we have efficient algorithms for weighted or partial graphs with provable approximation guaranties?
- In practice, greedy algorithms work very well but provably fail sometimes. Can we characterize when that happens?
- Practically solving Correlation Clustering problems in large scale is still a challenge.
- Better conversion and representation of data as graphs will enable fast and efficient clustering.
- Can we develop machine learned pairwise similarities that can support neighborhood queries over sets of objects?

Thank you!
Questions?