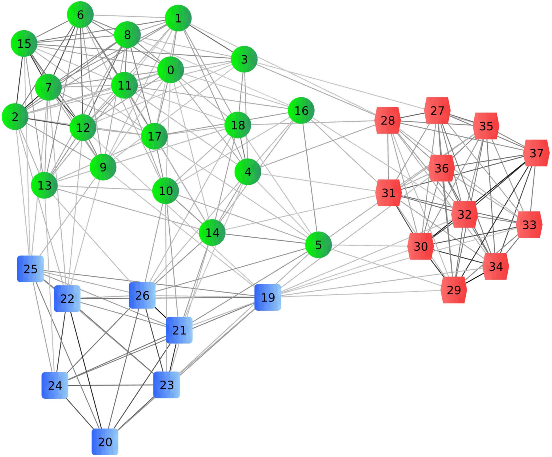
A Divisive clustering technique for maximizing the modularity

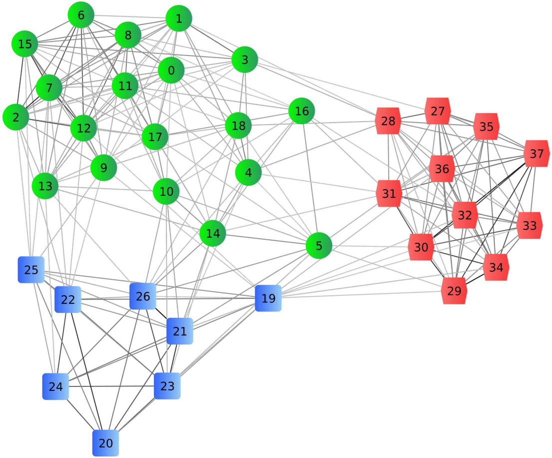
Ümit V. Çatalyürek, *Kamer Kaya*,
Johannes Langguth, and *Bora Uçar*

February 13, 2012

Clustering and Partitioning



Clustering and Partitioning



Clustering = k-way **Partitioning** with variable k

Graph Clustering

Graph Clustering

Given an undirected weighted graph $G = (V, E, w)$ find a "*light*" set of cut edges S such that the components of $G' = (V, E \setminus S, w)$ are "*heavy*".

Graph Clustering

Graph Clustering

Given an undirected weighted graph $G = (V, E, w)$ find a "*light*" set of cut edges S such that the components of $G' = (V, E \setminus S, w)$ are "*heavy*".

Features

- *light* and *heavy* are relative terms
- Want to avoid trivial cuts

Graph Clustering

Graph Clustering

Given an undirected weighted graph $G = (V, E, w)$ find a "*light*" set of cut edges S such that the components of $G' = (V, E \setminus S, w)$ are "*heavy*".

Features

- *light* and *heavy* are relative terms
- Want to avoid trivial cuts
- $\rightarrow$ Not yet a computational problem
- Need to capture compromise mathematically

Coverage

Clustering notation

Let $\mathcal{C}$ be a **Clustering** of **Clusters** $\mathcal{C}_1, \mathcal{C}_2, \ldots.\mathcal{C}_k$

$$\bigcup_{\mathcal{C}_i \in \mathcal{C}} = V \qquad \bigcap_{\mathcal{C}_i \in \mathcal{C}} = \emptyset$$

Light cut means maximize weight of edges $\{v, u\}$, $v \in \mathcal{C}_k, u \in \mathcal{C}_k$

Edge weight ω

Let

$$\omega(E) = \sum_{e \in E} w(e)$$

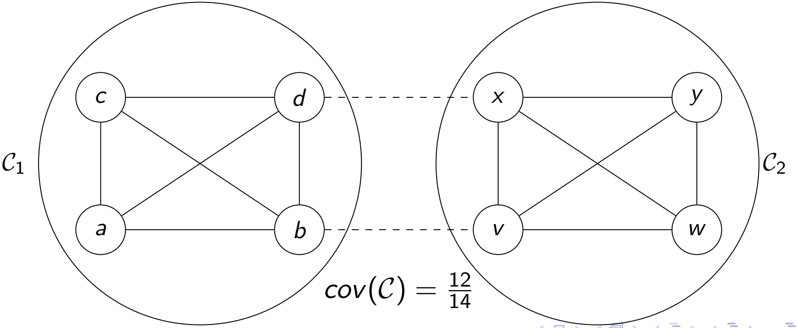
And

$$\omega(\mathcal{C}_i) = \omega(E_{G'[\mathcal{C}_i]})$$

Coverage

Coverage

$$cov(\mathcal{C}) = \frac{\sum_{\mathcal{C}_i \in \mathcal{C}} \omega(\mathcal{C}_i)}{\omega(E)}$$

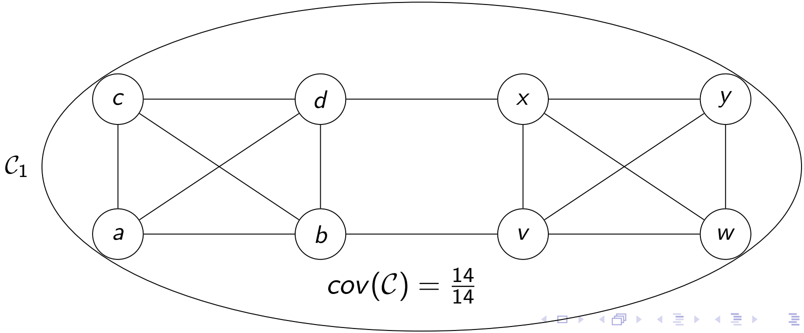


$$cov(\mathcal{C}) = \frac{12}{14}$$

Penalty term

Problem

- Trivial clustering $\mathcal{C} = \{\mathcal{C}_1\}$ maximizes coverage.
- We want *heavy* components (a.k.a. clusters)
- $\rightarrow$ Penalize large clusters



Penalty term

Problem

- Trivial clustering $\mathcal{C} = \{\mathcal{C}_1\}$ maximizes coverage.
- We want *heavy* components (a.k.a. clusters)
- $\rightarrow$ Penalize large clusters

Idea (Newman, 2003)

Introduce quadratic penalty term

Let

$$\psi(\mathcal{C}_i) = \sum_{v \in C_i} \Big(\sum_{\{v,u\} \in E} w\{v,u\} \Big)^2$$

Note

$$\psi(V) = 4 \times \omega(E)^2$$

Modularity

Modularity

$$p(\mathcal{C}) = cov(\mathcal{C}) - \frac{\sum_{\mathcal{C}_i \in \mathcal{C}} \psi(\mathcal{C}_i)^2}{4 \times \omega(E)^2}$$

Modularity

Modularity

$$p(\mathcal{C}) = cov(\mathcal{C}) - \frac{\sum_{\mathcal{C}_i \in \mathcal{C}} \psi(\mathcal{C}_i)^2}{4 \times \omega(E)^2}$$

Features

- $-0.5 \leq p(\mathcal{C}) \leq 1$
- Trivial clustering $\mathcal{T}$ has $p(\mathcal{T}) = 0$
- Isolated vertices have no effect
- Optimum solution is NP-hard, even for 2 clusters (Brandnes et al. 07)
- Also, APX-hard (DasGupta and Desai, 2011)

Approximation Algorithms

Agglomerative

Start with n clusters.
Join clusters until *penalty term* increase outweights *coverage* gain.
(i.e. modularity cannot be further increased.)

Approximation Algorithms

Agglomerative

Start with n clusters.
Join clusters until *penalty term* increase outweights *coverage* gain.
(i.e. modularity cannot be further increased.)

Divisive

Start with a single cluster.
Split clusters until *coverage* loss outweighs *penalty term* decrease.
(i.e. further modularity increase cannot be found.)

Agglomerative vs. Divisive Clustering

Assumption: split/join pairs.

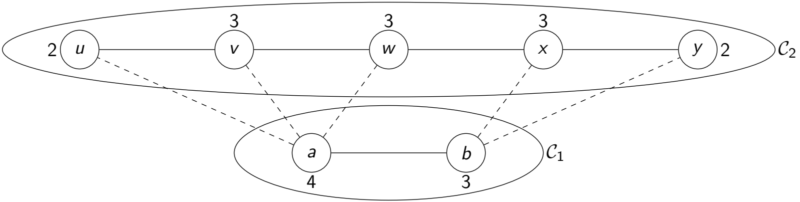
Agglomerative

- # possible moves starts high
- reduces over time
- Simple JOIN operation
- Modularity update has a single peak (Clauset et al. 2004)

Divisive

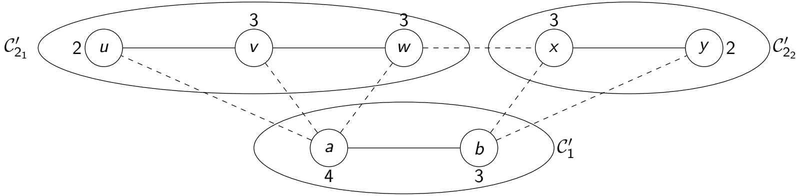
- # possible moves starts low
- increases over time
- Difficult BISECT operation
- Increase after a modularity-reducing bisection is possible.

Modularity Decrease



$$p(\mathcal{C}) = \frac{5}{10} - \frac{(3+4)^2 + (2+3+3+3+2)^2}{4 \times 10^2} = -\frac{18}{400}$$

Modularity Increase



$$p(\mathcal{C}') = \frac{4}{10} - \frac{(3+4)^2 + (2+3+3)^2 + (3+2)^2}{4 \times 10^2} = \frac{22}{400}\,.$$

Agglomerative vs. Divisive Clustering

Agglomerative

- # possible moves starts high
- reduces over time
- Simple JOIN operation
- Modularity update has a single peak (Clauset et al. 2004)

Divisive

- # possible moves starts low
- increases over time
- Difficult BISECT operation
- Increase after a modularity-reducing bisection is possible.

We follow the divisive approach

Algorithm Outline

Divisive Modularity Clustering

1. Start with a single (active) cluster
2. BISECT an active cluster: $\mathcal{C}_k \rightarrow \mathcal{C}'_k, \mathcal{C}''_k$
3. If modularity increases, keep $\mathcal{C}'_k, \mathcal{C}''_k$ and make them active.
4. Otherwise, keep $\mathcal{C}_k$ and make it inactive
5. Repeat until all clusters are inactive or singletons
6. REFINE CLUSTERING

Recursive bipartitioning strategy

Algorithm Outline

Divisive Modularity Clustering

1. Start with a single (active) cluster
2. BISECT an active cluster: $\mathcal{C}_k \to \mathcal{C}'_k, \mathcal{C}''_k$
3. If modularity increases, keep $\mathcal{C}'_k, \mathcal{C}''_k$ and make them active.
4. Otherwise, keep $\mathcal{C}_k$ and make it inactive
5. Repeat until all clusters are inactive or singletons
6. REFINE CLUSTERING

Recursive bipartitioning strategy

Algorithmic challenge: good BISECT method

BISECT

Subroutine BISECT

Input: a graph $G[C_i] = (C_i, E_G[C_i])$
Output: approx. $(2, 1+\epsilon)$ BALANCED PARTITION of $G[C_i]$

$(k, 1+\epsilon)$ BALANCED PARTITION

k - clustering $\mathcal{C}$ of G with $\max\{|\mathcal{C}_1|, |\mathcal{C}_2|\} \leq (1+\epsilon)\frac{|V|}{2}$ of maximum coverage

BISECT

Subroutine BISECT

Input: a graph $G[C_i] = (C_i, E_G[C_i])$
Output: approx. $(2, 1+\epsilon)$ BALANCED PARTITION of $G[C_i]$

$(k, 1+\epsilon)$ BALANCED PARTITION

k - clustering $\mathcal{C}$ of G with $\max\{|\mathcal{C}_1|, |\mathcal{C}_2|\} \leq (1+\epsilon)\frac{|V|}{2}$ of maximum coverage

Bipartition routine

- Use partitioner, e.g. PaToH, SCOTCH, or (modified) MeTiS
- Run multiple times and try different ϵ values

Bipartition routine

Bipartition routine

- Use PaToH, SCOTCH, MeTiS,...
- Try different ϵ values

ϵ	ratio in %
0.05	52 : 48
0.10	55 : 45
0.20	60 : 40
0.40	70 : 30
0.80	90 : 10

Bipartition routine

Bipartition routine

- Use PaToH, SCOTCH, MeTiS,...
- REFINE BISECTION
- Try different ϵ values

ϵ	ratio in %
0.05	52 : 48
0.10	55 : 45
0.20	60 : 40
0.40	70 : 30
0.80	90 : 10

REFINE BISECTION

Heuristic refinement

For each ϵ value, improve result heuristically.
Based on the *Fiduccia-Mattheyses* heuristic

Fiduccia-Mattheyses heuristic

- Maximizes approx. coverage
- Based on *Kernighan-Lin* heuristic
- Idea: move single vertices between clusters if gain is positive

REFINE BISECTION

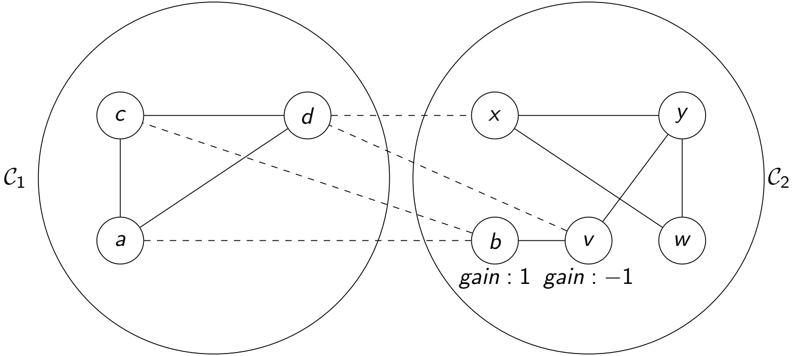
Heuristic refinement

For each ϵ value, improve result heuristically.
Based on the *Fiduccia-Mattheyses* heuristic

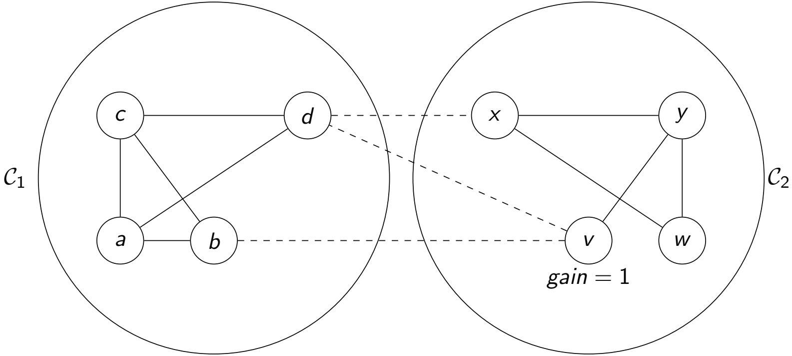
Fiduccia-Mattheyses heuristic

- Maximizes approx. coverage
- Based on *Kernighan-Lin* heuristic
- Idea: move single vertices between clusters if gain is positive
- Order of moves matters
- Keep vertices in priority queues according to gain
- Update neighbours after move

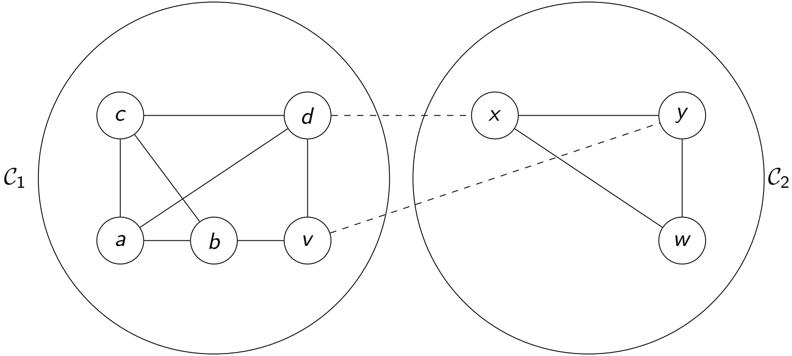
Refinement Heuristics



Refinement Heuristics



Refinement Heuristics



REFINE BISECTION

Heuristic refinement

For each ϵ value, improve result heuristically.

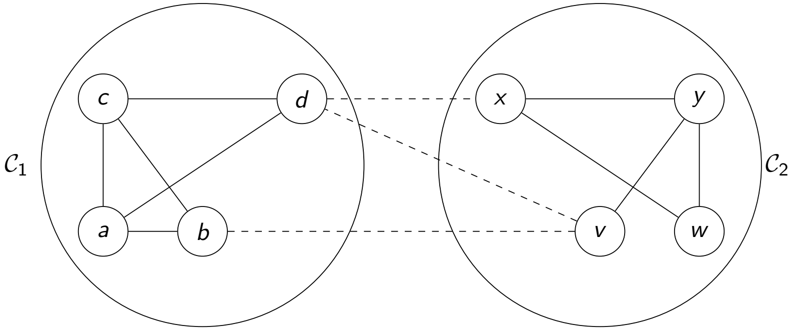
Based on the *Fiduccia-Mattheyses* heuristic

Nonlocal property

Coverage gains are local (affect only neighbours)

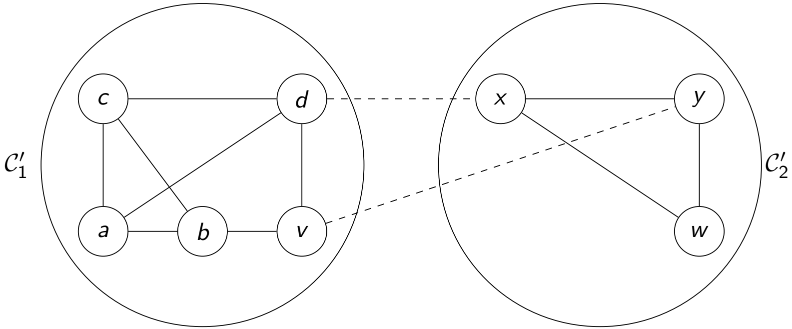
Changes in penalty term are global

Refinement Heuristics



$$p(\mathcal{C}) = \frac{9}{12} - \frac{(3+3+2+2)^2 + (1+2+2+3)^2}{4 \times 12^2} = \frac{268}{576}$$

Refinement Heuristics



$$p(\mathcal{C}') = \frac{10}{12} - \frac{(3+3+3+3+2)^2 + (2+2+2)^2}{4 \times 12^2} = \frac{248}{576}$$

REFINE BISECTION

Heuristic refinement

For each ϵ value, improve result heuristically.
Based on the *Fiduccia-Mattheyses* heuristic

Heuristic refinement of modularity

- Full update cost: $O(|V|)$ per move
- Use priority queues by coverage gain
- Retrieve top elements from both queues
- Compute actual gain for both vertices for each move
- $O(1)$ passes over all vertices

Algorithm Outline

Divisive Modularity Clustering

1. Start with a single (active) cluster
2. BISECT an active cluster: $\mathcal{C}_k \to \mathcal{C}'_k, \mathcal{C}''_k$
3. If modularity increases, keep $\mathcal{C}'_k, \mathcal{C}''_k$ and make them active.
4. Otherwise, keep $\mathcal{C}_k$ and make it inactive
5. Repeat until all clusters are inactive or singletons
6. REFINE CLUSTERING

REFINE CLUSTERING

REFINE CLUSTERING

- Run refinement heuristic after main algorithm terminates.
- Similar to REFINE BIPARTITION
- Includes all vertices in all clusters
- Choose vertices in random order.

Complexity

BISSECT

Complexity: $O(|E| \log |V|)$ Frequency: $O(|V| \log |V|)$
Amortized: $O(|E| \log^2 |V|)$

Complexity

BISSECT

Complexity: $O(|E| \log |V|)$ Frequency: $O(|V| \log |V|)$
Amortized: $O(|E| \log^2 |V|)$

REFINE BISSECTION

Complexity: $O(|E| \log |V|)$ Frequency: $O(|V| \log |V|)$
Amortized: $O(|E| \log^2 |V|)$

Complexity

BISSECT

Complexity: $O(|E|\log|V|)$ Frequency: $O(|V|\log|V|)$
Amortized: $O(|E|\log^2|V|)$

REFINE BISSECTION

Complexity: $O(|E|\log|V|)$ Frequency: $O(|V|\log|V|)$
Amortized: $O(|E|\log^2|V|)$

REFINE CLUSTERING

Complexity: $O(K|V|+|E|)$ Frequency: $O(1)$
$K =$ number of clusters

Total complexity:

$$O(|E|\log^2|V|)$$

Contains large constants!

Results

Test set: *coAuthorsCiteseer, coPapersCiteseer, citationCiteseer, coAuthorsDBLP, cond-mat, hep-th, preferentialAttachment, cond-mat-2005, netscience, email, football, karate, polbooks, astro-ph, as-22july06, chesapeake, smallworld, G_n_pin_pout, celegansneural, caidaRouterLevel, jazz, lesmis, power, adjnoun, dolphins, polblogs, cnr-2000, PGPgiantcompo, cond-mat-2003, celegans_metabolic, cond-mat-2003-component*

Average modularity found

- PaToH: 0.6507
- SCOTCH: 0.6430
- MeTiS: 0.6373

Improvement from REFINE CLUSTERING

	Before	After	Improvement
PaToH:	0.6465	0.6507	0.0042
SCOTCH:	0.6329	0.6430	0.0101

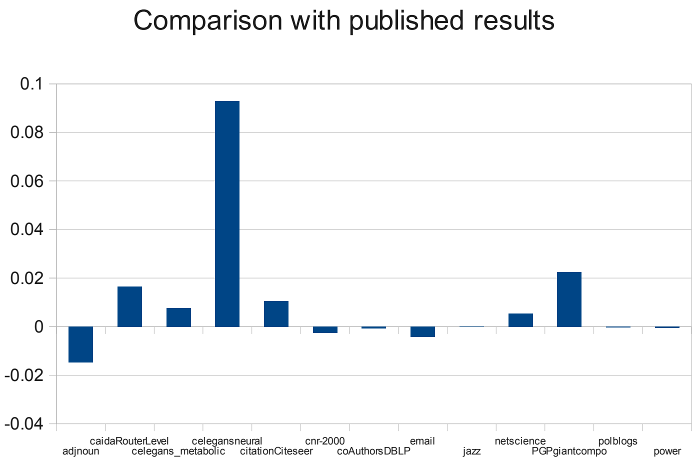
Improvement from repeated BISECT runs

	1 pass	5 passes	Improvement
PaToH:	0.6507	0.6514	0.0008
SCOTCH:	0.6430	0.6455	0.0025

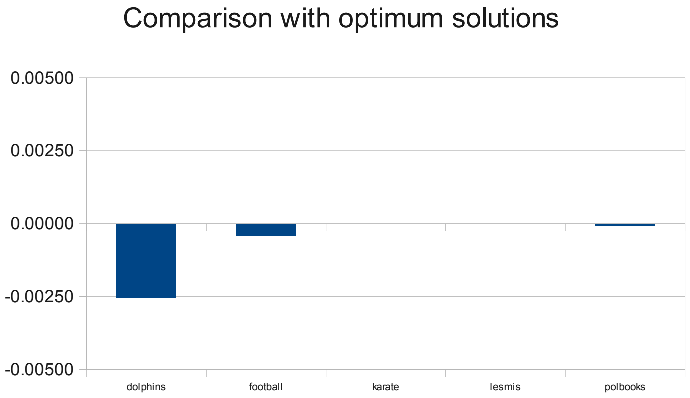
Improvement from repeated REFINE BISECTION

	not used	5 passes	Improvement
PaToH:	0.6507	0.6488	-0.0018
SCOTCH:	0.6430	0.6429	0.0001

Results


Comparison with published results
0.1
0.08
0.06
0.04
0.02
0
-0.02
-0.04
adjnoun
caidaRouterLevel
celegans_metabolic
celegansneural
citationCiteseer
cnr-2000
coAuthorsDBLP
email
jazz
netscience
PGPgiantcompo
polblogs
power

Results



Comparison with optimum solutions
0.00500
0.00250
0.00000
-0.00250
-0.00500
dolphins
football
karate
lesmis
polbooks

Conclusions

Divisive clustering technique

- Yields high modularity
- Straightforward implementation
- Theoretically fast
- To do: parallel implementation