Sublinear Algorithms for Estimating Single-Linkage Clustering Costs

Pan Peng
University of Science & Technology of China

Christian Sohler
University of Cologne
Cologne, Germany

Yi Xu
University of Science & Technology of China

Outline

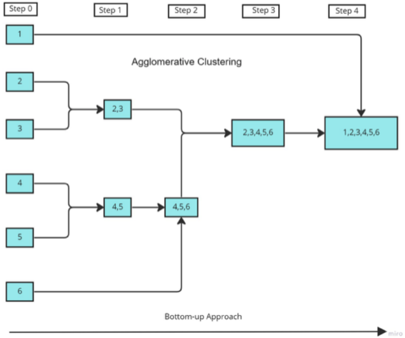
Outline

1. Background
2. Results
3. Proof Sketch
4. Extension to Similarity Case
5. Experiments
6. Conclusion

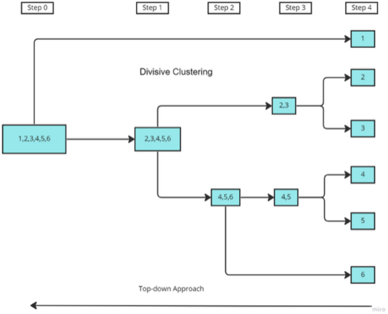
Hierarchical Clustering

Data mining, statistics: build a **hierarchy** of clusters **greedily**

- **Agglomerative:** bottom-up
- **Divisive:** top-down

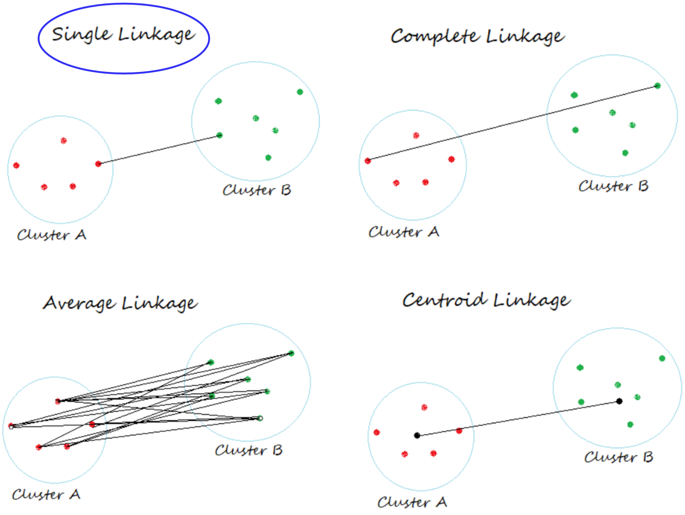


(a) Agglomerative

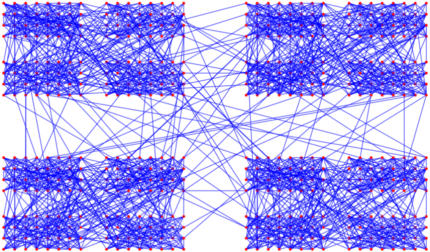


(b) Divisive

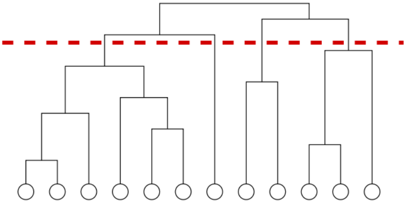
Hierarchical Clustering



Application - Community Detection



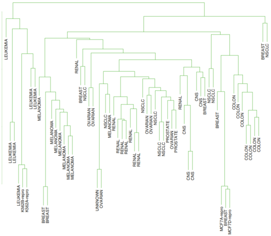
(a) Community



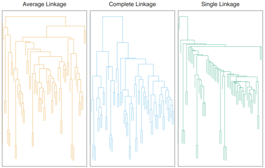
(b) Hierarchical tree

[For10]: Community detection in graphs
Citations: 13112

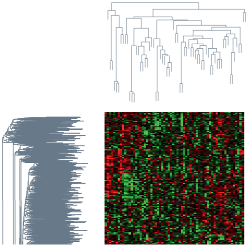
Application - Biology



(a) Human tumor

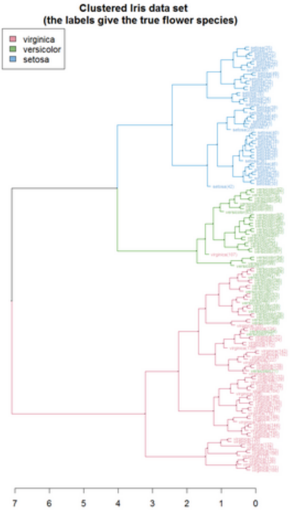


(b) Tumor: diff. linkages

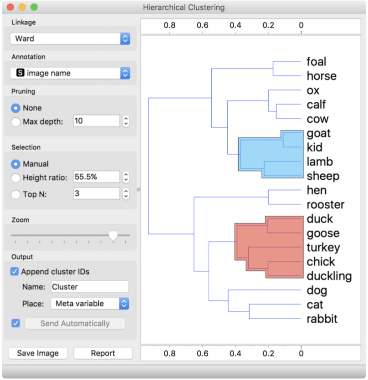


(c) DNA

[HTFF09]: Elements of statistical learning



(a) Iris clusters



(b) Software suite

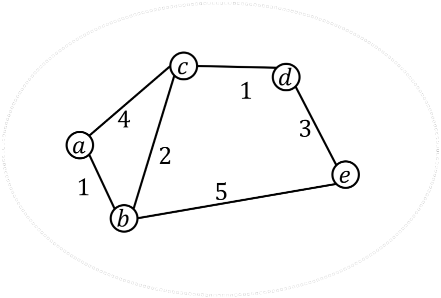
Single Linkage Clustering

- Input: connected, weighted graph $G = (V, E)$, **distance**
- SLC: bottom-up hierarchical clustering
  combine two **closest** clusters

Single Linkage Clustering

- Input: connected, weighted graph $G = (V, E)$, **distance**
- SLC: bottom-up hierarchical clustering
  combine two **closest** clusters
- Example: $V = \{a, b, c, d, e\}$

w_j: j-th **smallest** weight on MST
cost_k: **sum** of the costs of spanning trees within k clusters



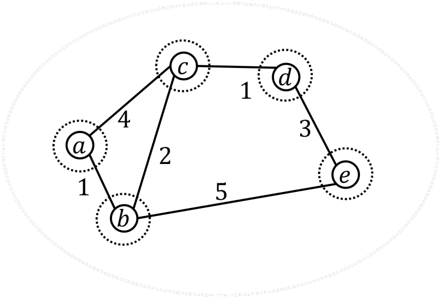
$$w_1 = 1,\ w_2 = 1,\ w_3 = 2,\ w_4 = 3$$

$$\mathrm{cost(MST)} = w_1 + w_2 + w_3 + w_4$$
$$= 7$$

Single Linkage Clustering

- Input: connected, weighted graph $G = (V, E)$, **distance**
- SLC: bottom-up hierarchical clustering
  combine two **closest** clusters
- Example: $V = \{a, b, c, d, e\}$

w_j: j-th **smallest** weight on MST
cost_k: **sum** of the costs of spanning trees within k clusters



$w_1 = 1,\ w_2 = 1,\ w_3 = 2,\ w_4 = 3$

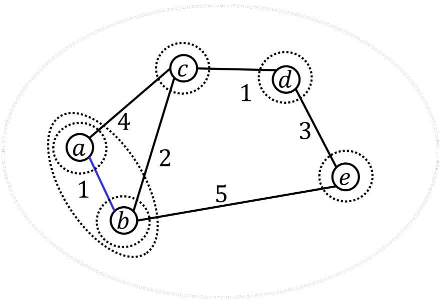
$\#$ of clusters $= 5$

$\mathrm{cost}_5 = 0$

Single Linkage Clustering

- Input: connected, weighted graph $G = (V, E)$, **distance**
- SLC: bottom-up hierarchical clustering
  combine two **closest** clusters
- Example: $V = \{a, b, c, d, e\}$

w_j: j-th **smallest** weight on MST
cost_k: **sum** of the costs of spanning trees within k clusters



$w_1 = 1,\ w_2 = 1,\ w_3 = 2,\ w_4 = 3$

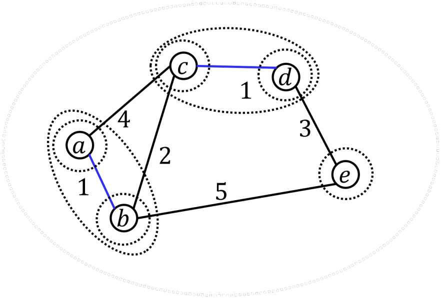
$\#$ of clusters $= 4$

$$\mathrm{cost}_4 = w_1$$
$$= 1$$

Single Linkage Clustering

- Input: connected, weighted graph $G = (V, E)$, **distance**
- SLC: bottom-up hierarchical clustering
  combine two **closest** clusters
- Example: $V = \{a, b, c, d, e\}$

w_j: j-th **smallest** weight on MST
cost_k: **sum** of the costs of spanning trees within k clusters



$w_1 = 1,\ w_2 = 1,\ w_3 = 2,\ w_4 = 3$

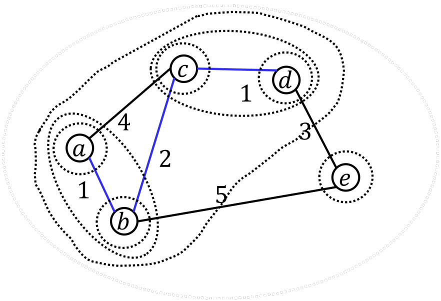
$\#$ of clusters $= 3$

$$\mathrm{cost}_3 = w_1 + w_2 = 2$$

Single Linkage Clustering

- Input: connected, weighted graph $G = (V, E)$, **distance**
- SLC: bottom-up hierarchical clustering
  combine two **closest** clusters
- Example: $V = \{a, b, c, d, e\}$

w_j: j-th **smallest** weight on MST
cost_k: **sum** of the costs of spanning trees within k clusters



$w_1 = 1,\ w_2 = 1,\ w_3 = 2,\ w_4 = 3$

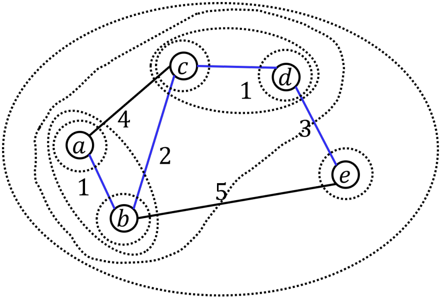
$\#$ of clusters $= 2$

$$\mathrm{cost}_2 = w_1 + w_2 + w_3$$
$$= 4$$

Single Linkage Clustering

- Input: connected, weighted graph $G = (V, E)$, **distance**
- SLC: bottom-up hierarchical clustering
  combine two **closest** clusters
- Example: $V = \{a, b, c, d, e\}$

w_j: j-th **smallest** weight on MST
cost_k: **sum** of the costs of spanning trees within k clusters



$w_1 = 1,\ w_2 = 1,\ w_3 = 2,\ w_4 = 3$

$\#$ of clusters $= 1$

$\mathrm{cost}_1 = w_1 + w_2 + w_3 + w_4$
$= 7$

Single Linkage Clustering

- Input: connected, weighted graph $G = (V, E)$, **distance**
- SLC: bottom-up hierarchical clustering
  combine two **closest** clusters

w_j: j-th **smallest** weight on MST
cost_k: **sum** of the costs of spanning trees within k clusters

Single Linkage Clustering

- Input: connected, weighted graph $G = (V, E)$, **distance**
- SLC: bottom-up hierarchical clustering
  combine two **closest** clusters

w_j: j-th **smallest** weight on MST
$\mathrm{cost}_k = \sum_{j=1}^{n-k} w_j$, sum of the costs of spanning trees within k clusters
$\mathrm{cost}(G) := \sum_{k=1}^{n} \mathrm{cost}_k = \sum_{j=1}^{n} (n-j) w_j$, total clustering cost

Motivation:

- cost_k captures important **structure**
- SLC minimizes these costs

Single Linkage Clustering

- Input: connected, weighted graph $G = (V, E)$, **distance**
- SLC: bottom-up hierarchical clustering
  combine two **closest** clusters

w_j: j-th **smallest** weight on MST
$\mathrm{cost}_k = \sum_{j=1}^{n-k} w_j$, sum of the costs of spanning trees within k clusters
$\mathrm{cost}(G) := \sum_{k=1}^{n} \mathrm{cost}_k = \sum_{j=1}^{n} (n-j) w_j$, total clustering cost

Motivation:

- cost_k captures important **structure**
- SLC minimizes these costs

Naive solution: compute an MST in $\tilde{O}(nd)$ time
Question: estimate $\mathrm{cost}(G)$ and cost_k in **sublinear** time?

Outline

Main Results

W: max weight d: average degree query model: adj. list

$\mathrm{cost}(G)^{\dagger}$	cost_k	Lower bound
$\tilde{O}(\frac{\sqrt{W}}{\varepsilon^3}d)^{\ddagger}$	$\tilde{O}(\frac{\sqrt{W}}{\varepsilon^3}d)$	$\Omega(\frac{\sqrt{W}}{\varepsilon^2}d)$

† $(1+\varepsilon)$-estimate of $\mathrm{cost}(G)$

‡ Applying [CRT05], one can get: $(1+\varepsilon)$-estimate, $\tilde{O}(\frac{W}{\varepsilon^2}d)$ queries

Main Results

W: max weight $\quad$ d: average degree $\quad$ query model: adj. list

$\mathrm{cost}(G)^{\dagger}$	cost_k	Lower bound
$\tilde{O}(\frac{\sqrt{W}}{\varepsilon^3}d)^{\ddagger}$	$\tilde{O}(\frac{\sqrt{W}}{\varepsilon^3}d)$	$\Omega(\frac{\sqrt{W}}{\varepsilon^2}d)$

Succinct representation of the SLC estimates $(\widehat{\mathrm{cost}}_1, \ldots, \widehat{\mathrm{cost}}_n)$ s.t. $\forall k$, recover $\widehat{\mathrm{cost}}_k$ in a **short** time, and **on average** a $(1+\varepsilon)$ estimate

On average: $\sum_{k=1}^{n} |\widehat{\mathrm{cost}_k} - \mathrm{cost}_k| \le \varepsilon \cdot \mathrm{cost}(G) = \varepsilon \sum_{k=1}^{n} \mathrm{cost}_k$
Short time: in $O(\log\log W)$ time

† $(1+\varepsilon)$-estimate of $\mathrm{cost}(G)$
‡ Applying [CRT05], one can get: $(1+\varepsilon)$-estimate, $\tilde{O}(\frac{W}{\varepsilon^2}d)$ queries

Outline

Main Ideas

CC: **C**onnected **C**omponent

Step 1

Reduction $\Rightarrow$ estimating # of **CC**s

Main Ideas

CC: **C**onnected **C**omponent

Step 1

Reduction $\Rightarrow$ estimating # of **CC**s

Step 2

Estimate # of CCs $\Rightarrow$ sample & BFS

Main Ideas

CC: **C**onnected **C**omponent

Step 1

Reduction $\Rightarrow$ estimating # of **CC**s

Step 2

Estimate # of CCs $\Rightarrow$ sample & BFS

[CRT05]: $\tilde{O}(\frac{W}{\varepsilon^2}d)$ queries

Main Ideas

CC: **C**onnected **C**omponent

Step 1

Reduction $\Rightarrow$ estimating # of **CC**s

Step 2

Estimate # of CCs $\Rightarrow$ sample & BFS

Step 3

Apply binary search to accelerate

Main Ideas

CC: **C**onnected **C**omponent

Step 1

Reduction $\Rightarrow$ estimating # of **CC**s

Step 2

Estimate # of CCs $\Rightarrow$ sample & BFS

Step 3

Apply binary search to accelerate

$\tilde{O}(\frac{\sqrt{W}}{\varepsilon^2}d)$ queries

Step 1: Reduction

Definition

- G_j: the subgraph containing edges with weight $\leq j$
- c_j: # of CCs within G_j
- n_j: # of edges in the MST with weight $= j$

Step 1: Reduction

Definition

- G_j: the subgraph containing edges with weight $\leq j$
- c_j: # of CCs within G_j
- n_j: # of edges in the MST with weight $= j$

$$\mathrm{cost}(G) = \sum_{i=1}^{n-1} (n-i) \cdot w_i$$

Step 1: Reduction

Definition

- G_j: the subgraph containing edges with weight $\leq j$
- c_j: # of CCs within G_j
- n_j: # of edges in the MST with weight $= j$

$$\begin{aligned}\mathrm{cost}(G) &= \sum_{i=1}^{n-1} (n-i) \cdot w_i \\ &= \sum_{i=1}^{n_1} (n-i) \cdot 1 + \sum_{i=n_1+1}^{n_1+n_2} (n-i) \cdot 2 + \ldots\end{aligned}$$

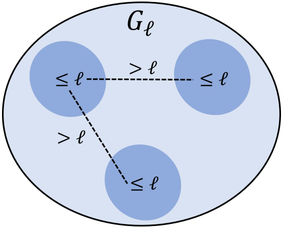
Step 1: Reduction

Definition

- G_j: the subgraph containing edges with weight $\leq j$
- c_j: # of CCs within G_j
- n_j: # of edges in the MST with weight $= j$

Observation

$\sum_{j>\ell} n_j = c_\ell - 1$



$$\mathrm{cost}(G) = \sum_{i=1}^{n-1} (n-i) \cdot w_i$$
$$= \sum_{i=1}^{n_1} (n-i) \cdot 1 + \sum_{i=n_1+1}^{n_1+n_2} (n-i) \cdot 2 + \ldots$$

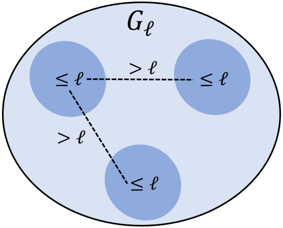
Step 1: Reduction

Definition

- G_j: the subgraph containing edges with weight $\leq j$
- c_j: # of CCs within G_j
- n_j: # of edges in the MST with weight $= j$

Observation

$\sum_{j>\ell} n_j = c_\ell - 1$



$$\begin{aligned}
\mathrm{cost}(G) &= \sum_{i=1}^{n-1} (n-i) \cdot w_i \\
&= \sum_{i=1}^{n_1} (n-i) \cdot 1 + \sum_{i=n_1+1}^{n_1+n_2} (n-i) \cdot 2 + \ldots \\
&= \sum_{i=1}^{n-c_1} (n-i) \cdot 1 + \ldots
\end{aligned}$$

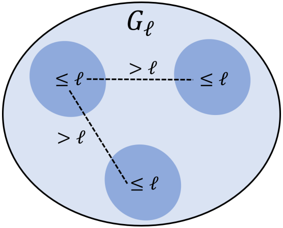
Step 1: Reduction

Definition

- G_j: the subgraph containing edges with weight $\leq j$
- c_j: # of CCs within G_j
- n_j: # of edges in the MST with weight $= j$

Observation

$\sum_{j>\ell} n_j = c_\ell - 1$



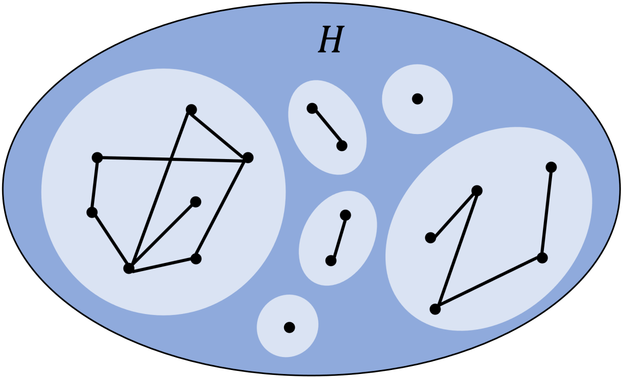
$$\mathrm{cost}(G) = \sum_{i=1}^{n-1} (n-i) \cdot w_i$$

$$= \frac{n(n-1)}{2} + \frac{1}{2} \cdot \sum_{j=1}^{W-1} (c_j^2 - c_j)$$

Step 2: Estimate # of CCs

Input: graph G and subgraph H
Output: $\hat{c}$, estimated # of CCs in H

Algorithm: [CRT05]

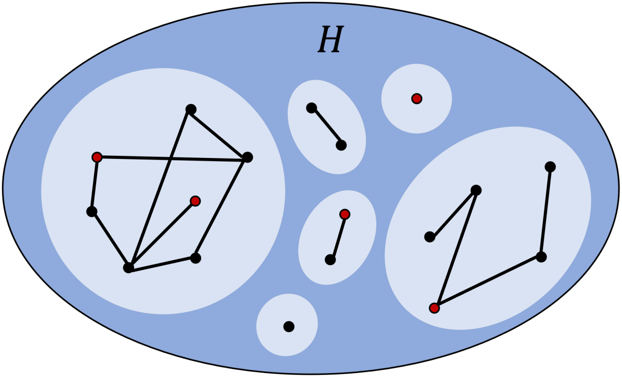


Step 2: Estimate # of CCs

Input: graph G and subgraph H
Output: $\hat{c}$, estimated # of CCs in H

Algorithm: [CRT05]

1. Sample

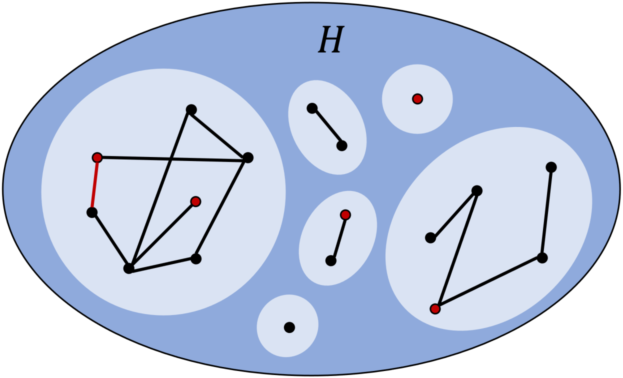


Step 2: Estimate # of CCs

Input: graph G and subgraph H
Output: $\hat{c}$, estimated # of CCs in H

Algorithm: [CRT05]

1. Sample
2. BFS

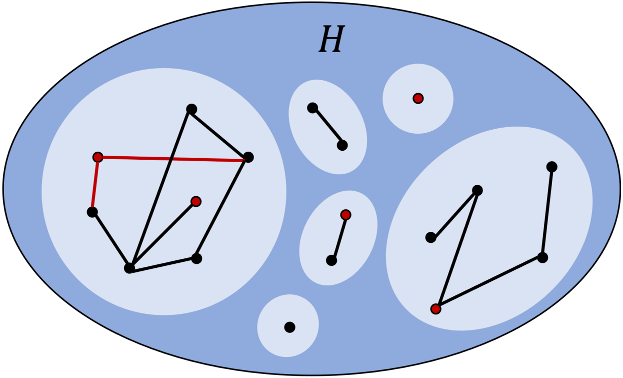


Step 2: Estimate # of CCs

Input: graph G and subgraph H
Output: $\hat{c}$, estimated # of CCs in H

Algorithm: [CRT05]

1. Sample
2. BFS
   - Recursively:
   - Flip a coin
   - Double # of visited edges

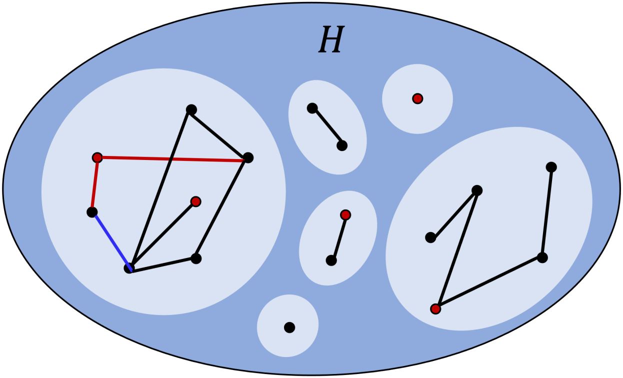


Step 2: Estimate # of CCs

Input: graph G and subgraph H
Output: $\hat{c}$, estimated # of CCs in H

Algorithm: [CRT05]

1. Sample
2. BFS
   - Recursively:
   - Flip a coin
   - Double # of visited edges

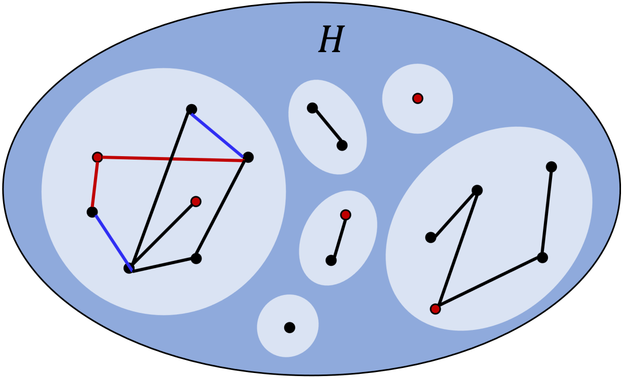


Step 2: Estimate # of CCs

Input: graph G and subgraph H
Output: $\hat{c}$, estimated # of CCs in H

Algorithm: [CRT05]

1. Sample
2. BFS
   - Recursively:
   - Flip a coin
   - Double # of visited edges

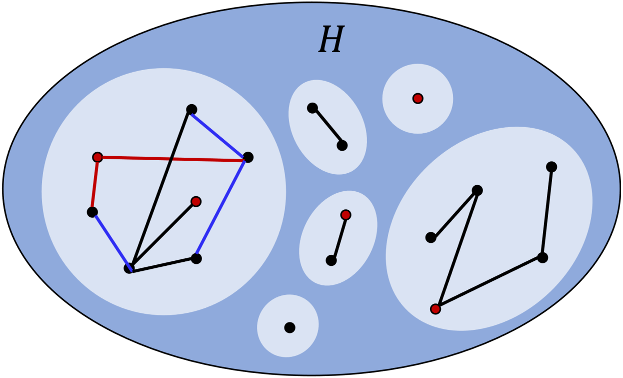


Step 2: Estimate # of CCs

Input: graph G and subgraph H
Output: $\hat{c}$, estimated # of CCs in H

Algorithm: [CRT05]

1. Sample
2. BFS
   - Recursively:
   - Flip a coin
   - Double # of visited edges



Step 2: Estimate # of CCs

Input: graph G and subgraph H

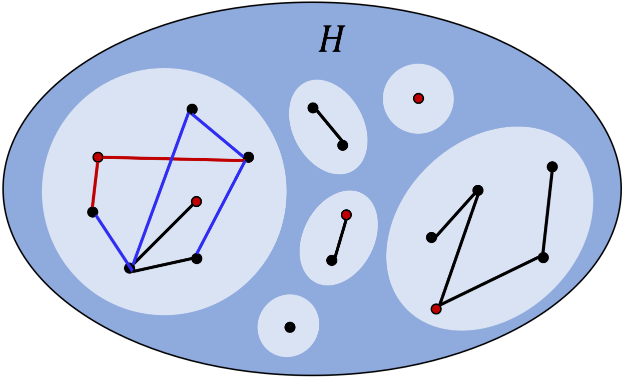
Output: $\hat{c}$, estimated # of CCs in H

Algorithm: [CRT05]

1. Sample
2. BFS
   - Recursively:
   - Flip a coin
   - Double # of visited edges

Guarantee: [CRT05]

- $|\hat{c} - c| \leq \varepsilon n$
- $\tilde{O}(\frac{d}{\varepsilon^2})$



Step 2: Estimate # of CCs

Input: graph G and subgraph H

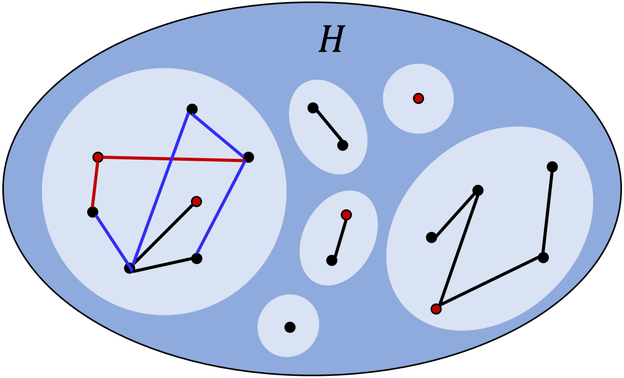
Output: $\hat{c}$, estimated # of CCs in H

Algorithm: [CRT05]

1. Sample
2. BFS
   - Recursively:
   - Flip a coin
   - Double # of visited edges

Guarantee: ours

- $|\hat{c} - c| \leq \varepsilon \cdot \max\{\frac{n}{\sqrt{W}}, c\}$
- $\tilde{O}(\frac{\sqrt{W}}{\varepsilon^2} d)$ queries



Step 3: Binary Search

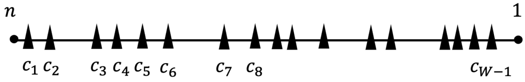
$\mathrm{cost}(G) = \frac{n(n-1)}{2} + \sum_{j=1}^{W-1} \frac{c_j^2 - c_j}{2}$

- Observation: $c_1, \ldots, c_W \in [1, n]$ is **non-increasing**

Step 3: Binary Search

$\mathrm{cost}(G) = \frac{n(n-1)}{2} + \sum_{j=1}^{W-1} \frac{c_j^2 - c_j}{2}$

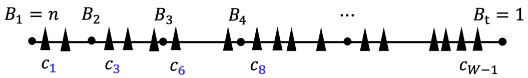
- Observation: $c_1, \ldots, c_W \in [1, n]$ is **non-increasing**



Step 3: Binary Search

$\mathrm{cost}(G) = \frac{n(n-1)}{2} + \sum_{j=1}^{W-1} \frac{c_j^2 - c_j}{2}$

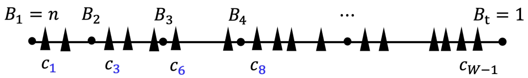
- Observation: $c_1, \ldots, c_W \in [1, n]$ is **non-increasing**
- Idea:
  Divide $[1, n]$ into intervals $1 = B_t < \cdots < B_1 = n$



Step 3: Binary Search

$\text{cost}(G) = \frac{n(n-1)}{2} + \sum_{j=1}^{W-1} \frac{c_j^2 - c_j}{2}$

- Observation: $c_1, \ldots, c_W \in [1, n]$ is **non-increasing**
- Idea:
  Divide $[1, n]$ into intervals $1 = B_t < \cdots < B_1 = n$

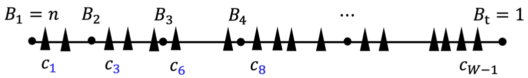


$\text{cost}(G) \approx \frac{n(n-1)}{2} + \sum_{i=1}^{t-1} \{\#\text{ of elements within } i\text{-th interval}\} \cdot \frac{B_i^2 - B_i}{2}$

Step 3: Binary Search

$\mathrm{cost}(G) = \frac{n(n-1)}{2} + \sum_{j=1}^{W-1} \frac{c_j^2 - c_j}{2}$

- Observation: $c_1, \ldots, c_W \in [1, n]$ is **non-increasing**
- Idea:
  Divide $[1, n]$ into intervals $1 = B_t < \cdots < B_1 = n$
  Map each c_j to its **nearest** interval endpoint

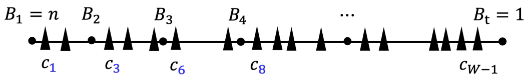


$\mathrm{cost}(G) \approx \frac{n(n-1)}{2} + \sum_{i=1}^{t-1} \{\#\text{ of elements within } i\text{-th interval}\} \cdot \frac{B_i^2 - B_i}{2}$

Step 3: Binary Search

$\mathrm{cost}(G) = \frac{n(n-1)}{2} + \sum_{j=1}^{W-1} \frac{c_j^2 - c_j}{2}$

- Observation: $c_1, \ldots, c_W \in [1, n]$ is **non-increasing**
- Idea:
  Divide $[1, n]$ into intervals $1 = B_t < \cdots < B_1 = n$
  Map each c_j to its **nearest** interval endpoint



$$\mathrm{cost}(G) \approx \frac{n(n-1)}{2} + \sum_{i=1}^{t-1} \{\#\text{ of elements within } i\text{-th interval}\} \cdot \frac{B_i^2 - B_i}{2}$$

Challenge: estimated $\hat{c}_1, \ldots, \hat{c}_W$ may **not** be non-increasing

Step 3: Binary Search

For $c_j \in (B_{i+1}, B_i]$, let $|\hat{c}_j - c_j| = \varepsilon \cdot \max\{\frac{n}{\sqrt{W}}, c_j\}$.
We further need: $\forall c_j \in (B_{i+1}, B_i],\ B_i - B_{i+1} = \Theta(|\hat{c}_j - c_j|)$

Step 3: Binary Search

For $c_j \in (B_{i+1}, B_i]$, let $|\hat{c}_j - c_j| = \varepsilon \cdot \max\{\frac{n}{\sqrt{W}}, c_j\}$.
We further need: $\forall c_j \in (B_{i+1}, B_i],\ B_i - B_{i+1} = \Theta(|\hat{c}_j - c_j|)$

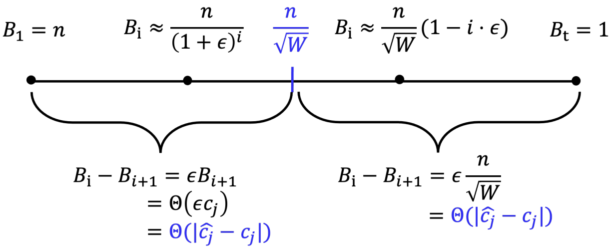
- Too **large**: can't estimate c_j well;
- Too **small**: $\hat{c}_j$ will locate in an interval far away

Step 3: Binary Search

For $c_j \in (B_{i+1}, B_i]$, let $|\hat{c}_j - c_j| = \varepsilon \cdot \max\{\frac{n}{\sqrt{W}}, c_j\}$.

We further need: $\forall c_j \in (B_{i+1}, B_i]$, $B_i - B_{i+1} = \Theta(|\hat{c}_j - c_j|)$

- Too **large**: can't estimate c_j well;
- Too **small**: $\hat{c}_j$ will locate in an interval far away

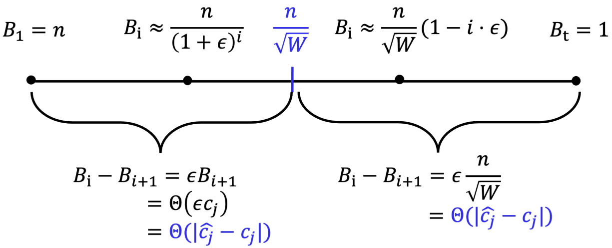


Step 3: Binary Search

For $c_j \in (B_{i+1}, B_i]$, let $|\hat{c}_j - c_j| = \varepsilon \cdot \max\{\frac{n}{\sqrt{W}}, c_j\}$.

We further need: $\forall c_j \in (B_{i+1}, B_i]$, $B_i - B_{i+1} = \Theta(|\hat{c}_j - c_j|)$

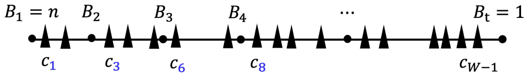
- Too **large**: can't estimate c_j well;
- Too **small**: $\hat{c}_j$ will locate in an interval far away



Remark: $t = O(\log W / \varepsilon)$

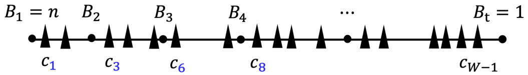
Estimate cost(G)

$$\widehat{\text{cost}}(G) = \frac{n(n-1)}{2} + \sum_{i=1}^{t-1} \{\# \text{ of elements within } i\text{-th interval}\} \cdot \frac{B_i^2 - B_i}{2}$$



Estimate cost(G)

$\widehat{\text{cost}}(G) = \frac{n(n-1)}{2} + \sum_{i=1}^{t-1} \{\#$ of elements within i-th interval$\} \cdot \frac{B_i^2 - B_i}{2}$

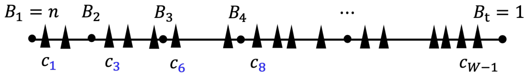


Complexity

$t = O(\log W/\varepsilon)$

Estimate cost(G)

$\widehat{\text{cost}}(G) = \frac{n(n-1)}{2} + \sum_{i=1}^{t-1}\{\#$ of elements within i-th interval$\} \cdot \frac{B_i^2 - B_i}{2}$

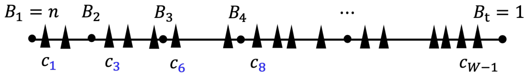


Complexity

$t = O(\log W/\varepsilon) \Rightarrow \tilde{O}(t \cdot \frac{\sqrt{W}}{\varepsilon^2} d) = \tilde{O}(\frac{\sqrt{W}}{\varepsilon^3} d)$ queries

Estimate cost(G)

$\widehat{\text{cost}}(G) = \frac{n(n-1)}{2} + \sum_{i=1}^{t-1} \{\#\text{ of elements within } i\text{-th interval}\} \cdot \frac{B_i^2 - B_i}{2}$



Complexity

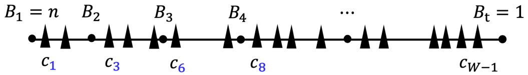
$t = O(\log W/\varepsilon) \Rightarrow \tilde{O}(t \cdot \frac{\sqrt{W}}{\varepsilon^2} d) = \tilde{O}(\frac{\sqrt{W}}{\varepsilon^3} d)$ queries

Correctness

$\forall c_j \in (B_{i+1}, B_i],\ B_i - B_{i+1} = \Theta(|\hat{c}_j - c_j|)$

Estimate cost(G)

$\widehat{\text{cost}}(G) = \frac{n(n-1)}{2} + \sum_{i=1}^{t-1} \{\#$ of elements within i-th interval$\} \cdot \frac{B_i^2 - B_i}{2}$



Complexity

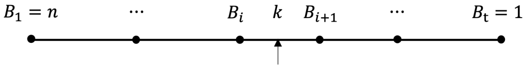
$t = O(\log W/\varepsilon) \Rightarrow \tilde{O}(t \cdot \frac{\sqrt{W}}{\varepsilon^2} d) = \tilde{O}(\frac{\sqrt{W}}{\varepsilon^3} d)$ queries

Correctness

$\forall c_j \in (B_{i+1}, B_i],\ B_i - B_{i+1} = \Theta(|\hat{c}_j - c_j|)$
$\Rightarrow \widehat{\text{cost}}(G)$ achieves $(1+\varepsilon)$ approximation ratio

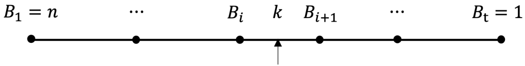
Estimating the Profile

Profile vector: $(\mathrm{cost}_1, \mathrm{cost}_2, \ldots, \mathrm{cost}_n)$



Estimating the Profile

Profile vector: $(\mathrm{cost}_1, \mathrm{cost}_2, \ldots, \mathrm{cost}_n)$



Main idea:

1. Pre-process in $\tilde{O}(\frac{\sqrt{W}}{\varepsilon^3} d)$ time
2. $\textsc{ProfileOracle}(k)$: return $\widehat{\mathrm{cost}}_k$ in $O(\log t) = O(\log\log W)$ time
3. $\sum_{k=1}^{n} |\widehat{\mathrm{cost}}_k - \mathrm{cost}_k| = \varepsilon \mathrm{cost}(G) = \varepsilon \sum_{k=1}^{n} \mathrm{cost}_k$

Outline

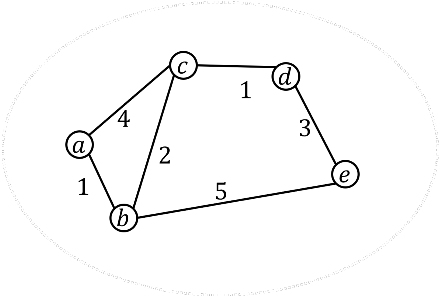
SLC in the Similarity Case

- Input: weight represents **similarity** between two nodes
- SLC: combine two **most similar** clusters

SLC in the Similarity Case

- Input: weight represents **similarity** between two nodes
- SLC: combine two **most similar** clusters

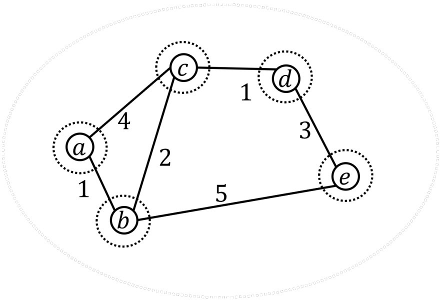
w_j: j-th **largest** weight on MaxST
cost_k: **sum** of the costs of spanning trees within k clusters



$$w_1 = 5,\ w_2 = 4,\ w_3 = 3,\ w_4 = 2$$
$$\mathrm{cost}(\mathrm{MaxST}) = w_1 + w_2 + w_3 + w_4$$
$$= 14$$

- Input: weight represents **similarity** between two nodes
- SLC: combine two **most similar** clusters

w_j: j-th **largest** weight on MaxST
cost_k: **sum** of the costs of spanning trees within k clusters



$w_1 = 5,\ w_2 = 4,\ w_3 = 3,\ w_4 = 2$

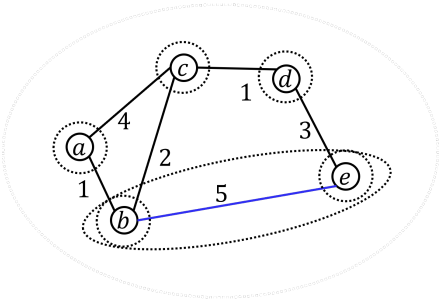
$\#$ of clusters $= 5$

$\text{cost}_5 = 0$

SLC in the Similarity Case

- Input: weight represents **similarity** between two nodes
- SLC: combine two **most similar** clusters

w_j: j-th **largest** weight on MaxST
cost_k: **sum** of the costs of spanning trees within k clusters



$w_1 = 5, \; w_2 = 4, \; w_3 = 3, \; w_4 = 2$

$\#$ of clusters $= 4$

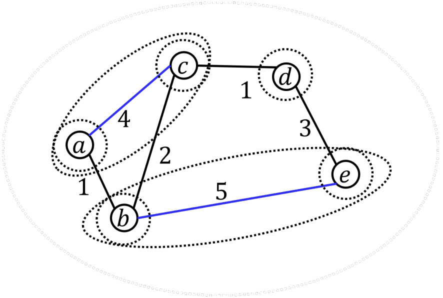
$\mathrm{cost}_4 = w_1$

$= 5$

SLC in the Similarity Case

- Input: weight represents **similarity** between two nodes
- SLC: combine two **most similar** clusters

w_j: j-th **largest** weight on MaxST
cost_k: **sum** of the costs of spanning trees within k clusters



$$w_1 = 5,\ w_2 = 4,\ w_3 = 3,\ w_4 = 2$$

$$\#\text{ of clusters} = 3$$

$$\mathrm{cost}_3 = w_1 + w_2$$
$$= 9$$

Results in the Similarity Case

cost(G)	cost_k	Lower bound
$\tilde{O}(\frac{W}{\varepsilon^3}d)$	$\tilde{O}(\frac{W}{\varepsilon^3}d)$	$\Omega(\frac{W}{\varepsilon^2}d)$

Results in the Similarity Case

$\text{cost}(G)$	cost_k	Lower bound
$\tilde{O}(\frac{W}{\varepsilon^3} d)$	$\tilde{O}(\frac{W}{\varepsilon^3} d)$	$\Omega(\frac{W}{\varepsilon^2} d)$

Main idea:

1. G_j: subgraph with weights $\geq j$ $\quad c_j$: $\#$ of CCs in G_j

Results in the Similarity Case

$\mathrm{cost}(G)$	cost_k	Lower bound
$\tilde{O}(\frac{W}{\varepsilon^3}d)$	$\tilde{O}(\frac{W}{\varepsilon^3}d)$	$\Omega(\frac{W}{\varepsilon^2}d)$

Main idea:

1. G_j: subgraph with weights $\geq j$ c_j: $\#$ of CCs in G_j
2. Reduction: $\mathrm{cost}(G) = \sum_{j=1}^{W} \frac{(c_j+n-1)(n-c_j)}{2}$.

Results in the Similarity Case

$\text{cost}(G)$	cost_k	Lower bound
$\tilde{O}(\frac{W}{\varepsilon^3}d)$	$\tilde{O}(\frac{W}{\varepsilon^3}d)$	$\Omega(\frac{W}{\varepsilon^2}d)$

Main idea:

1. G_j: subgraph with weights $\geq j$ c_j: $\#$ of CCs in G_j
2. Reduction: $\text{cost}(G) = \sum_{j=1}^{W} \frac{(c_j+n-1)(n-c_j)}{2}$.

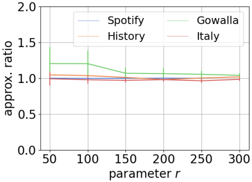
Remark:

- W appears in every k-cluster
- need **new** algorithm on $n - c_j$

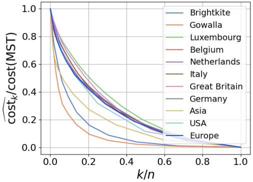
Outline

Experiments

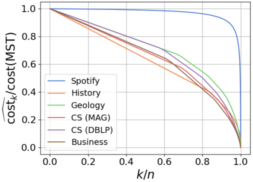
r: sample size for each estimate of # CC; $\widehat{\mathrm{cost}}_k$: estimated value for cos_k.



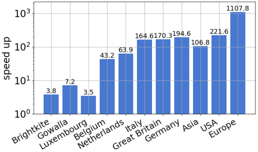
(a) Approx. ratio



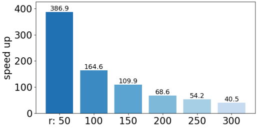
(b) Distance profiles



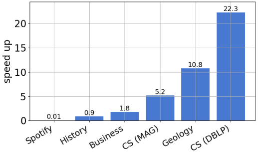
(c) Similarity profiles



(d) Distance with $r = 100$



(e) On one dataset with various r



(f) Similarity with $r = 100$

Outline

Recall: Our Results

Summary

W: max weight d: average degree query model: adj. list

Setting	$\mathrm{cost}(G)^{\dagger}$	$\mathrm{cost}_k^{\ddagger}$	Lower bound
Distance Case	$\tilde{O}(\frac{\sqrt{W}}{\varepsilon^3}d)$	$\tilde{O}(\frac{\sqrt{W}}{\varepsilon^3}d)$	$\Omega(\frac{\sqrt{W}}{\varepsilon^2}d)$
Similarity Case	$\tilde{O}(\frac{W}{\varepsilon^3}d)$	$\tilde{O}(\frac{W}{\varepsilon^3}d)$	$\Omega(\frac{W}{\varepsilon^2}d)$

Open questions:

- Dependency on ε
- Extend to other hierarchical clustering
  - Average linkage, centroid linkage

† Approximation ratio is $(1+\varepsilon)$

‡ On *average* is a $(1+\varepsilon)$-estimate