Sublinear Algorithms for Hierarchical Clustering

Sanjeev Khanna
University of Pennsylvania

Joint work with Arpit Agarwal (Columbia), Huan Li (Penn), and Prathamesh Patil (Penn).
Hierarchical Clustering

A technique to cluster data into a multilevel hierarchy based on similarity.
Hierarchical Clustering

A technique to *cluster data* into a *multilevel hierarchy* based on *similarity*. It arranges *data* as a *rooted tree* such that

---

- the *root* represents the *entire data set*, and each *leaf* corresponds to a *unique data point*.
- each *internal node* corresponds to a *cluster containing its descendant leaves*
Hierarchical Clustering

A technique to **cluster data** into a **multilevel hierarchy** based on **similarity**. It arranges **data** as a **rooted tree** such that:

-- **the root** represents the **entire data set**, and each **leaf** corresponds to a **unique data point**.

-- **each internal node** corresponds to a **cluster** containing its **descendant leaves**

*Clusters data at **multiple levels of granularity** simultaneously.*
The Hierarchical Clustering Problem

Dasgupta (2016) introduced the following formalization:

- **Input:** A weighted graph whose vertices correspond to data points and whose edges capture similarity between the data points.
- The cost of any HC tree $T$ is given by

$$\text{Cost}(T) = \sum_{\text{splits } s \rightarrow (S_l, S_r) \text{ in } T} (|S| \cdot w_G(S_l, S_r))$$

where $w_G(S_l, S_r) = \text{total weight of edges going from } S_l \text{ to } S_r$.

**Goal:** Find a tree that minimizes this cost.
The Hierarchical Clustering Problem

The cost function incentivizes cutting high weight similarity edges deeper down the tree.
Why this Cost Function?

- **Dasgupta (2016)** motivates this cost function as having several desirable properties:
  - When the data consists of a collection of connected components, an optimal tree starts by building a hierarchy that separates the components.
  - When the input graph is a clique, all trees should have the same cost – no particular cluster hierarchy is to be favored.
  - It recovers the desirable solution for some models of planted cluster partitions.

- **Cohen-Addad et al. (2019)** take an axiomatic approach to characterize good cost functions in general.

- We will focus on the Dasgupta objective in this talk.
The Hierarchical Clustering Problem

- The problem of finding the best HC tree is NP-hard.
- Assuming Small Set Expansion (SSE) conjecture, no $O(1)$–approximation possible [Charikar-Chatziafratis 17].
- A natural algorithm called recursive sparsest cut gives $O(\alpha)$–approximation where $\alpha = O(\sqrt{\log n})$ is the sparsest cut approximation guarantee [Charikar-Chatziafratis 17], [Cohen-Addad et al. 19].

Useful fact: At expense of an $O(1)$–loss in approximation ratio, we can assume that each binary partition is roughly balanced.
Can we match the best-known approximation guarantees for hierarchical clustering via sublinear algorithms?

Based on the computational platform, we may want sublinear query/time, space, or communication algorithms.

We will consider all three resources.
Sublinear Space Algorithms

Streaming Model of Computation

- The graph is presented as a stream of edges.
- The algorithm has limited memory to store information about the edges seen in the stream.
- A natural model when the input is either generated "on the fly" or is stored on a sequential access device, like a disk.
- The algorithm no longer has random access to the input.

Goal is to design algorithms that use space that is much smaller than the size of the graph.
Sublinear Query/Time Algorithms

Query Model of Computation

- Degree queries: What is the degree of a vertex $v$?
- Pair queries: Is $(u, v)$ an edge?
- Neighbor queries: Who is the $k_{th}$ neighbor of a vertex $v$?

Goal is to design algorithms that compute by performing only a few queries – much smaller than the size of the graph.

Additional goal: efficiently process the queries to recover a good HC tree.
Sublinear Communication Algorithms

MPC Model of Computation (Massively Parallel Computation)
- The edges of the graph are partitioned across multiple machines in an arbitrary manner.
- Each machine has small memory – much smaller than the input.
- Computation proceeds in rounds where in each round, a machine can send and receive limited information to other machines (not exceeding its memory).

Goal is to compute in a small number of rounds using only machines with small memory.
Our Results

- There are efficient sublinear algorithms for hierarchical clustering in all three models of computation.
- There are also nearly matching lower bounds that show these algorithms are essentially best possible.

**Notation:** We will use $n$ to denote the number of vertices and $m$ to denote the number of edges.
Theorem 0: Given a weighted graph $G$ as a stream of edges, there is an $\tilde{O}(n)$ space algorithm to find a $(1 + o(1))$-approximate hierarchical clustering of $G$.

- The approximation guarantee above is better than $O(\sqrt{\log n})$ because the model allows unbounded computation time. It is $O(\sqrt{\log n})$ in poly-time.
- It is also easy to show that $\Omega(n)$ space is necessary to obtain any $\tilde{O}(1)$-approximation.
- The algorithm also works for dynamic streams.
Results 1: Sublinear Communication Algorithms (MPC Model)

**Theorem 1:** Given a weighted graph $G$ with edges partitioned across machines with $\tilde{O}(n)$ memory, can find a $(1 + o(1))$–approximate hierarchical clustering of $G$ in 2 rounds.

**Theorem 2:** No randomized 1-round protocol using machines with $n^{4/3-\epsilon}$ memory for any $\epsilon > 0$, can output an $\tilde{O}(1)$–approximate hierarchical clustering even on unweighted graphs.
Results 2: Sublinear Query/Time Algorithms

Theorem 3: Given an unweighted graph $G$ with $m$ edges, there is an algorithm that outputs a $(1 + o(1))$–approximate hierarchical clustering of $G$ using

- $\tilde{O}(n+m)$ queries if $m \leq n^{4/3}$.
- $\tilde{O}(n + m/\alpha^3)$ queries if $m = \alpha \cdot n^{4/3}$ for some $\alpha \geq 1$.

The query bound starts becoming sublinear once $m$ exceeds $n^{4/3}$, and then drops to $\tilde{O}(n)$ queries once $m \geq n^{3/2}$.
By investing an additional $n^{1+\tau+o(1)}$ time over the query complexity, we can get an $O(\sqrt{\log n/\tau})$-approximate solution [Sherman 09] and [Chen, Kyng, Liu, Peng, Probst Gutenberg, Sachdeva 22].

We can get similar guarantees for the weighted case, assuming a suitable graph representation.

**Theorem 4:** The query complexity achieved by the algorithm in Theorem 3 is essentially optimal for every edge density.
Related Recent Work

Assadi, Chatziafratis, Lacki, Mirrokkni, and Wang (2022)
- Focuses on estimating the HC value in sublinear in \( n \) space, and shows several negative results.
- Also gives algorithms for finding a \( \Theta(1) \)–approximate HC tree in the streaming and the MPC model – this is slightly weaker than \( (1 + o(1)) \)–approximation that we get.

Kapralov, Kumar, Lattanzi, Mousavifar (2022)
- Focuses on estimating the HC value in sublinear queries in \((k, \epsilon)\)-clusterable graphs: input is \( k \) expanders with outer conductance bounded by \( \epsilon \).
- \( O(\sqrt{\log k}) \)–approximation in \( \text{poly}(k) \cdot n^{1/2+O(\epsilon)} \) queries.
Sublinear Algorithms
Graph Sparsification for HC

Given any HC tree $T$, the cost of $T$ is given by

$$\text{Cost}(G,T) = \sum \text{splits } s \rightarrow (S_l, S_r) \text{ in } T \left( |S| \cdot w_G(S_l, S_r) \right)$$

where $w_G(S_l, S_r) = \text{total weight of edges going from } S_l \text{ to } S_r$.

Natural idea: Work with an approximate cut sparsifier of $G$. For any pair of disjoint sets $X, Y$, we can express $w_G(X, Y)$ in terms of cuts in $G$:

$$w_G(S_l, S_r) = \frac{1}{2} \cdot (w_G(S_l, S_l) + w_G(S_r, S_r) - w_G(S_l \cup S_r, S_l \cup S_r))$$

Problem: Expressing $w_G(S_l, S_r)$ as difference of approximately preserved values, can result in unbounded error.
Graph Sparsification for HC

\[ w_G(S_l, S_r) = \frac{1}{2} \cdot (w_G(S_l, \overline{S_l}) + w_G(S_r, \overline{S_r}) - w_G(S_l \cup S_r, \overline{S_l \cup S_r})). \]

Observation: If we fix any HC tree, the negative term at any node appears with a strictly larger positive coefficient at the parent of the node.

\[ |A| \cdot \frac{1}{2} \cdot (w_G(B, \overline{B}) + w_G(C, \overline{C}) - w_G(A, \overline{A})) \]

\[ |B| \cdot \frac{1}{2} \cdot (w_G(D, \overline{D}) + w_G(E, \overline{E}) - w_G(B, \overline{B})) \]

Note that \(|A| > |B|\).
Graph Sparsification for HC

**Upshot:** The cost of any tree $T$ can be written as

$$\sum_{\text{splits } s \rightarrow (S_l, S_r) \text{ in } T} \frac{1}{2} \cdot (|S_r| \cdot w_G(S_l, S_r) + |S_l| \cdot w_G(S_r, S_r)) + \sum_v w_G(v, \bar{v})$$

We get a blackbox reduction to cut sparsifiers.

To get a $(1 + o(1))$-approximate hierarchical clustering, it suffices to construct a $(1 + o(1))$–approximate cut sparsifier.

Now we can just focus on accomplishing this task in various models of computation.
Immediate Applications

Corollary (Thm 0): There is an $\tilde{O}(n)$ space dynamic streaming algorithm that outputs a $(1 + o(1))$–approximate hierarchical clustering of a weighted graph.

Corollary (Thm 1): There is a 2-round MPC algorithm with $\tilde{O}(n)$ space per machine that outputs a $(1 + o(1))$–approximate hierarchical clustering of a weighted graph.

Both results basically follow from [Ahn, Guha, McGregor 12].
Application to Sublinear Time?

Constructing a cut sparsifier necessarily requires $\Omega(m)$ queries (even for connectivity).

We will work with a relaxed notion of cut sparsifiers that will prove much easier to construct.
A Relaxed Notion of Cut Sparsifiers

A graph $H(V, E')$ is an $(\epsilon, \delta)$-sparsifier of a graph $G(V, E)$ if for any cut $(S, \bar{S})$, we have

$$(1 - \epsilon)w_G(S) \leq w_H(S) \leq (1 + \epsilon)w_G(S) + \delta \cdot \min\{|S|, |\bar{S}|\}$$

The usual notion of cut sparsifiers gives an $(\epsilon, 0)$-sparsifier.

**Lemma:** If $H$ is an $(\epsilon, \delta)$-sparsifier of a graph $G$ then for any HC tree $T$, we have

$$(1 - \epsilon)\text{cost}_G(T) \leq \text{cost}_H(T) \leq (1 + \epsilon)\text{cost}_G(T) + O(\delta \cdot n^2)$$
High-level Plan for Sublinear Time

We will focus on unweighted graphs.

- Show that larger the δ, the easier it is to compute an \((\epsilon, \delta)\)-sparsifier.
- But how large can we make δ to still get a \((1 + o(1))\)–approximation?
- Identify an easy to compute lower bound \(C\) for optimal HC cost, and set \(\delta = o\left(\frac{C}{n^2}\right)\) to get \((1 + o(1))\)–approximation.
High-level Plan for Sublinear Time

Lemma: The cost of hierarchical clustering on any unweighted graph $G$ with $n$ vertices and $m$ edges is $\Omega\left(\frac{m^2}{n}\right)$.

Example: Suppose $G$ is any graph with $m \gg n^{3/2}$ edges, then optimal tree cost is $\gg n^2$.
So if we set $\delta = O(1)$, then the $O(\delta \cdot n^2)$ additive error term is negligible because optimal tree cost is $\gg n^2$.

Let us focus on this density regime, and we will design a $\tilde{O}(n/\varepsilon^2)$ query algorithm to construct an $(\varepsilon, O(1))$-sparsifier.
Constructing an $(\epsilon, O(1))$-sparsifier

[Spielman-Srivastava 11]

One way to construct an $(\epsilon, 0)$-sparsifier of $G$:
sample $O(n \log n/\epsilon^2)$ times each edge $e = (u, v)$ with probability $p_e$ proportional to $R(u, v) = \text{effective resistance between } u \text{ and } v$.

**Difficulty:** How to estimate effective resistances in sublinear time?

**Fix:** Add a constant degree expander $G'$ to $G$. 
Constructing an $(\epsilon, O(1))$-sparsifier

Observation: Any $(\epsilon, 0)$-sparsifier for the graph $H = G \cup G'$ is an $(\epsilon, O(1))$-sparsifier for the graph $G$.

For any cut $(S, \bar{S})$, its size in any $(\epsilon, 0)$-sparsifier of $H$

- is at least $(1 - \epsilon)w_G(S)$, and
- at most $(1 + \epsilon)w_G(S) + O(1 + \epsilon). \min\{|S|, |\bar{S}|\}$

New Goal: Construct an $(\epsilon, 0)$-sparsifier of the graph $H$. 
An \((\epsilon, 0)\)-sparsifier of the Graph \(H\)

What have we gained by shifting the focus to \(H\) instead of \(G\)?

**Observation:** For any edge \(e = (u, v)\), its effective resistance \(R(u, v)\) in \(H\) satisfies

\[
\frac{1}{\min\{d_H(u), d_H(v)\}} \leq R(u, v) \leq \frac{O(\log n)}{\min\{d_H(u), d_H(v)\}}
\]

\(R(u, v) \geq \frac{1}{\min\{d_H(u), d_H(v)\}}\) is easy.
An \((\varepsilon, 0)\)-sparsifier of the Graph \(H\)

More interesting direction: \(R(u, v) \leq \frac{o(\log n)}{\min\{d_H(u), d_H(v)\}}\)

In a constant degree expander, for any 2 sets \(X\) and \(Y\), there are \(\approx \min\{|X|, |Y|\}\) edge-disjoint paths of \(O(\log n)\) length between \(X\) and \(Y\) [Frieze 01].
Constructing an \((\epsilon, O(1))\)-sparsifier

We now have a very simple algorithm to construct an \((\epsilon, 0)\)-sparsifier for the graph \(H = G \cup G'\).

Repeat the following for \(\tilde{O}(n/\epsilon^2)\) steps:
- sample a random vertex \(v\).
- sample a random edge incident on \(v\), and add it to the sparsifier.

Thus in \(\tilde{O}(n/\epsilon^2)\) queries, we get a sparsified graph that gives a \((1 + \epsilon)\)-approximation to hierarchical clustering whenever the input graph contains \(m \gg n^{3/2}\) edges.
General Case: An \((\varepsilon, \delta)\)-sparsifier

Add constant degree expander \(G'\) with edges of weight \(\delta\).

Observation: For any edge \((u, v)\) in \(H = G \cup G'\), we have

\[
\frac{1}{\min\{d_H(u), d_H(v)\}} \leq R(u, v) \leq \frac{O(\log n)}{\min\{d_H(u), d_H(v)\}} \cdot \frac{1}{\delta}
\]

Now construct an \((\varepsilon, 0)\)-sparsifier for the graph \(H = G \cup G'\) by sampling as before for \(\tilde{O}(n/\delta\varepsilon^2)\) steps.

A variation of this expander idea was used by [Lee 14] for efficiently answering a single cut query with bounded additive error – we need this guarantee to hold for all cut queries.
Lower Bounds
Query Lower Bounds

**Theorem:** For any $\gamma \in (0, \frac{1}{2})$, there is a family of unweighted graphs with $m = \Theta(n^{1+\gamma})$ edges such that any randomized algorithm that outputs an $\tilde{O}(1)$–approximate hierarchical clustering for this family, requires $n^{\min\{1+\gamma, 2-2\gamma\} - o(1)}$ queries.

The lower bound
- remains $m^{1-o(1)}$ as $m$ increases from $n$ to $n^{4/3}$; and
- then gradually decreases from $n^{4/3-o(1)}$ to $n^{1-o(1)}$ as $m$ increases from $n^{4/3}$ to $n^{3/2}$.

We will illustrate the lower bound idea for $\gamma = 1/3$, and show a lower bound of $n^{4/3-o(1)}$ queries.
$n^{4/3-o(1)}$ Query Lower Bound for $m = n^{4/3}$

$n^{2/3}$ randomly matched pairs of cliques
An Optimal Tree

Optimal clustering cost: $\Theta(n^{5/3})$
Consider any $\tilde{O}(1) -$approximation algorithm $A$.

- Assume w.l.o.g. that the top-level partition is roughly balanced in the solution output by $A$.
- $A$ must not cut too many clique matching edges at the top partition since penalty for each edge cut is $n$. So $A$ must "discover" most of the meta-matching among the cliques.
- It takes about $n^{2/3 - o(1)}$ queries to discover match of a given clique under $M$.
- We need to discover $\Omega(n^{2/3})$ matches in $M$, giving us an $n^{4/3 - o(1)}$ query lower bound.
MPC Lower Bound

Theorem 2: No randomized 1-round protocol using machines with $n^{4/3-\varepsilon}$ memory for any $\varepsilon > 0$, can output an $\tilde{O}(1)$–approximate hierarchical clustering even on unweighted graphs.

- The input graph is partitioned across $\approx n^{1/3}$ machines with $n^{4/3-\varepsilon}$ memory for an arbitrarily small $\varepsilon > 0$.
- We want to rule out recovery of an $\tilde{O}(1)$–approximate HC tree in one round of communication.
MPC Lower Bound

\[ |V_1| = |V_2| = n. \]

Union of \( \approx n^{1/3} \) bipartite cliques of size \( \approx n^{2/3} \)

Union of \( \approx n^{2/3} \) bipartite cliques of size \( \approx n^{1/3} \)

So \( \Theta(n^{5/3}) \) edges are partitioned across \( \approx n^{1/3} \) machines.
MPC Lower Bound

Key idea: each machine gets a graph isomorphic to $G[V_2]$. We do this by tiling the bi-cliques in $G[V_1]$ by graphs that are isomorphic to $G[V_2]$.

A machine can not tell locally whether it received the blue cliques, the red cliques, or the graph $G[V_2]$ itself.
MPC Lower Bound

Key idea: each machine gets a graph isomorphic to $G[V_2]$.
We do this by tiling the bi-cliques in $G[V_1]$ by graphs that are isomorphic to $G[V_2]$.

Any $\tilde{O}(1)$–approximate solution must discover how the vertices are partitioned across the cliques in $G[V_2]$. 
MPC Lower Bound

Key idea: each machine gets a graph isomorphic to $G[V_2]$. We do this by tiling the bi-cliques in $G[V_1]$ by graphs that are isomorphic to $G[V_2]$.

So each of the $n^{1/3}$ machines needs to send $\Omega(n)$ bits of information to the coordinator – this is much more than the coordinator’s memory.
Concluding Remarks

- We designed near-optimal sublinear algorithms for hierarchical clustering in the query model, streaming, and MPC model.
- The main algorithmic ingredient:
  - a relaxed notion of cut sparsifiers that is easy to compute in various computational models.
- We also establish lower bounds that almost match the performance of our algorithms.
- An interesting direction is to understand if there is a separation between the queries needed to estimate the value and finding a clustering in general graphs.
Thank you!