University of Michigan–Ann Arbor

Department of Electrical Engineering and Computer Science

EECS 498 004 **Advanced Graph Algorithms**, Fall 2021

Lecture 17: Expander Decomposition: Existence and Construction

October 28, 2021

Instructor: Thatchaphol Saranurak

Scribe: Tian Zhang

We have seen a rich theory around expanders. They are great for so many reasons. **It would be wonderful if we can exploit expanders in non-expander graphs too.** Now come the punchline: **It turns out that we can!** This is where expander decomposition comes into play.

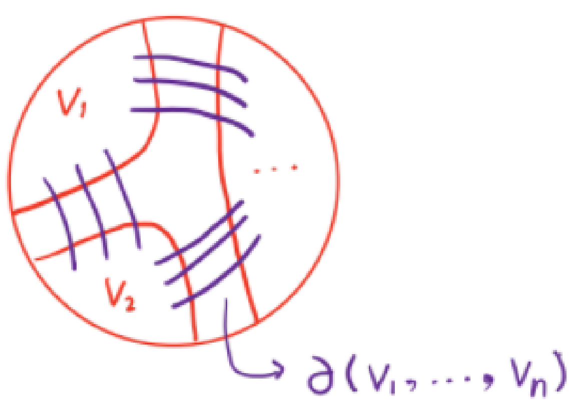
- Today: Existence and construction of expander decomposition.
- Next lecture: Applications of it.

1. Overview

Expander Decomposition = What Happens when Keep Finding Sparse Cuts. Let's say $G = (V, E)$ is unweighted. Let's fix parameter ϕ. Imagine the following process:

- DECOMP(G):
  - If $\Phi(G) \geq \phi$, return $\{V\}$
  - Else, there is a cut S where $\Phi_G(S) < \phi$, return DECOMP($G[S]$) $\cup$ DECOMP($G[V \setminus S]$)

If DECOMP(G) $= \{V_1, \ldots, V_k\}$, then we know each $\Phi(G[V_i]) \geq \phi$, i.e. $G[V_i]$ is a ϕ-expander.



- **Question:** How many edges we have „cut" throughout the whole process?
  - Formally, Let $\partial(V_1, \ldots, V_k)$ denote the edges crossing from V_i to V_j, $j \neq i$. What is $|\partial(V_1, \ldots, V_k)|$?
- **Answer:** $O(\phi m \log m)$ where $m = |E(G)|$.
- **Analysis:**
  - For each sparse cut $(S, V \setminus S)$ where $\mathrm{vol}(S) \leq \mathrm{vol}(V)/2$, we have that $|E(S, V \setminus S)| \leq \phi \mathrm{vol}(S)$.
  - Charge the edges in $E(S, V \setminus S)$ to nodes in the smaller side S.
    * where each node $u \in S$ is charged $\phi \deg(u)$.
  - How many times a node can be charged?
    * $O(\log m)$, as we charge to the side where the volume is halved.

The following theorem concludes what we get.

1.1. Theorem (Expander Decomposition)**.** *For any unweighted graph $G = (V, E)$ with m edges and any $\phi \in [0, 1]$, there is a partition of vertices* $\textsc{Decomp}(G) = \{V_1, \ldots, V_k\}$ *such that*

- *$G[V_i]$ is a ϕ-expander, i.e., $\Phi(G[V_i]) \geq \phi$, and*
- *at most $\phi \log m$-fraction of edges crosses the partition.*

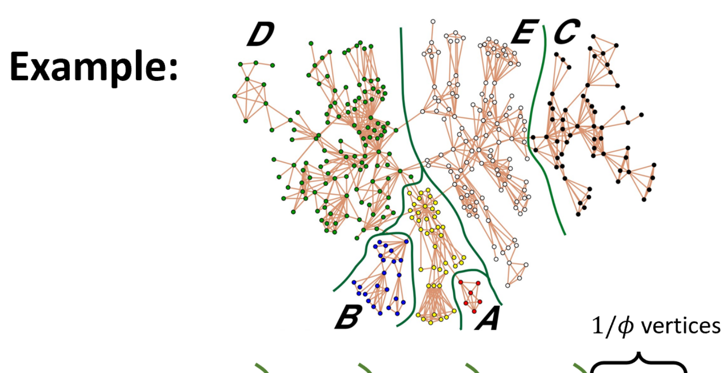
Intuitively, this is a maxim that you should keep

Any Graph = Disjoint Expanders + Small Fraction of Edges

Let's see some examples:

1.2. Example. What is $\textsc{Decomp}(G)$ when G is...

- Stars or Cliques
- Grids
- Planar graphs



However, **the algorithm is slow for two reasons**

1. **Checking $\Phi(G)$ exactly is NP-hard.**
   Easy to fix. Use approximation algorithms. Let's say that we can find approximate sparse cut near-linear time (and we can). There is still another problem.

2. **We might need to recurse to the bigger side $\Omega(n)$ time.**
   So the running time can be $\Omega(mn)$.

2. Fast Algorithm: Reduction to Balanced Sparse Cut

To bound the recursion depth (hopefully $O(\log m)$), we want to find the most-balanced ϕ−sparse cut each time.

Let $\textsc{MaxBal}(G, \phi)$ be the volume of most-balanced ϕ-sparse cut.

$$\textsc{MaxBal}(G, \phi) = \max\{\mathrm{vol}(S) \mid \Phi_G(S) < \phi \text{ and } \mathrm{vol}(S) \leq \mathrm{vol}(V \setminus S)\}$$

If $\Phi(G) \geq \phi$ (no ϕ-sparse cut), define $\textsc{MaxBal}(G, \phi) = 0$.

2.1. Exercise. Show an algorithm $\textsc{BalCut}(G, \phi)$ that either outputs

- $S = \emptyset$ and reports that $\Phi(G) \geq \phi$
- $S \neq \emptyset$ where $\mathrm{vol}(S) \leq \mathrm{vol}(V \setminus S)$ such that
  - $\Phi_G(S) < \phi \cdot c_{\mathrm{exp}}$
  - $\mathrm{vol}(S) \geq \textsc{MaxBal}(G, \phi)/c_{\mathrm{bal}}$

where $c_{\mathrm{exp}}, c_{\mathrm{bal}} = O(\log^2 n)$ are approximation factor on expansion and balance. Note the algorithm described above approximately solves the most-balanced ϕ−sparse cut problem.

$\textsc{BalCut}(G, \phi)$ can be solved via the cut-matching framework and takes $\mathrm{polylog}(n)$ (approximate) max flow calls. A natural way for using $\textsc{BalCut}$ for $\textsc{Decomp}$ would be this.

- $\textsc{Decomp}(H)$:
  - Let $S \leftarrow \textsc{BalCut}(H, \phi)$
  - If $S = \emptyset$ (i.e. $\Phi(H) \geq \phi$),
    * return $\{V(H)\}$
  - Else (i.e. $\Phi_G(S) < c_{\mathrm{exp}} \cdot \phi$)
    * return $\textsc{Decomp}(H[S]) \cup \textsc{Decomp}(H[V \setminus S])$.
- Then, we call $\textsc{Decomp}(G)$.

2.1. Lucky Setting: Always Balanced Cuts

Suppose $\textsc{BalCut}(G, \phi)$ always returns a balanced cut S where $\mathrm{vol}(S), \mathrm{vol}(V \setminus S) \geq \mathrm{vol}(V)/10$.

- **Question:** What is the total running time?

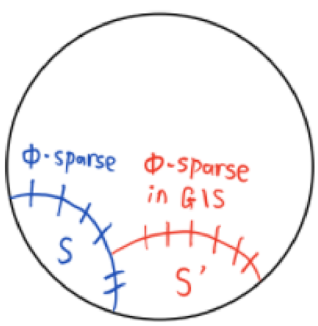
- **Answer:** There will be $O(\log m)$ recursion depth. Within the same depth, graphs are disjoint. Total time is $\tilde{O}(m)$.
- **Problem:** But what if all ϕ-sparse cuts are not balanced?
  - $\textsc{BalCut}(G, \phi)$ cannot return balanced cut.
  - So recursion depth is big?

2.2. Ideal Setting: Exact Algorithm

Suppose magically we can solve the exact version of $\textsc{BalCut}(G, \phi)$ where $c_{\mathrm{exp}}, c_{\mathrm{bal}} = 1$ in linear time.

Let $S \leftarrow \textsc{BalCut}(G, \phi)$ s.t. $\mathrm{vol}(S) = \textsc{MaxBal}(G, \phi)$ and $\Phi_G(S) < \phi$. So S is the most-balanced cut among all ϕ-sparse cut.

- **The Setting:**
  - Suppose S is not very balanced (say, $\mathrm{vol}(S) \le \mathrm{vol}(V)/10$). (Otherwise it is good.)
  - Let's look at we get when we recurse $\textsc{Decomp}(G[V \setminus S], \phi)$ on the bigger side.
  - Let $S' \leftarrow \textsc{BalCut}(G[V \setminus S], \phi)$.
- **Question:** Can S' be not balanced again (say, $\mathrm{vol}(S') \le \mathrm{vol}(V \setminus S)/10$)?
- **Answer**: No.
  - This is because $S \cup S'$ would be a ϕ-sparse cut in G which is more balanced than the ϕ-sparse cut S.



  - Contradict the guarantee of S.
- **Conclusion:** That is, we cannot get very unbalanced cut two consective times on the bigger side. So the recursion depth is still $O(\log m)$!

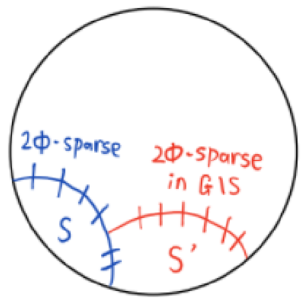
2.3. How Things Goes Wrong with Approximation

As we have seen,

- If we are always lucky, then the algorithm is fast.

- Else if we are not lucky but we have an exact algorithm that returns the most -balanced sparse cut, it's also fine.
- However, the reality is that we are not only unlucky but also have an approximate algorithm only.

The argument in 2.2 does not work even when $c_{\text{exp}} = 2$ (or anything > 1) and $c_{\text{bal}} = 1$. It's because We know that there is no ϕ-sparse cut which is more balanced than S, but S itself and $S \cup S'$ is only 2ϕ-sparse. There is no contradiction and the recursion depth seems to be still $\Omega(n)$.



2.4. The Trick: Adjusting Parameters

In reality, we have $c_{\text{exp}}, c_{\text{bal}} = O(\log^2 n)$. Let's see how bad it can be...

Suppose $S \leftarrow \textsc{BalCut}(G, \phi)$ is super unbalanced, say $\text{vol}(S) = O(1)$. Ok, S might not be useful. **But the algorithm just tells us something quite strong:** $\textsc{MaxBal}(G, \phi) \le O(c_{\text{bal}}) = O(\log^2 n)$.

Now comes **the key trick**:

- What if we run $\textsc{BalCut}(G, \phi/c_{\text{exp}})$ instead of $\textsc{BalCut}(G, \phi)$?
  - The cut returned must have conductance at most $c_{\text{exp}}(\phi/c_{\text{exp}}) = \phi$.
  - Can $\textsc{BalCut}(\cdot, \phi/c_{\text{exp}})$ keep returning $O(1)$-volume on the bigger side of the recursion?
  - Not more than $O(c_{\text{bal}})$ times! Otherwise the union of these cuts will of have volumes more than $O(c_{\text{bal}}) \ge \textsc{MaxBal}(G, \phi)$.
  - Contradiction.

2.5. Formal Algorithm

- **Parameters**:
  - $\varepsilon = 0.01$ (actually, we later we will set $\varepsilon = \Theta(\sqrt{\log n})$, but let's say $\varepsilon = 0.01$ for now)
  - $L = 1/\varepsilon$
  - Let $\phi_1 \ge \cdots \ge \phi_{L+1} = \phi$ where $\phi_{\ell+1} = \phi_\ell / c_{\text{exp}}$. So $\phi_1 = \phi \cdot \log^{O(L)} n$
  - $m = |E(G)|$
- $\textsc{Decomp}(H, \ell)$:

- Let $S \leftarrow \textsc{BalCut}(H, \phi_{\ell+1})$
- If $S = \emptyset$ (i.e. $\Phi(H) \geq \phi_{\ell+1}$),
  * return $\{V(H)\}$
- Else (i.e. $\Phi_H(S) < c_{\exp} \cdot \phi_{\ell+1} = \phi_\ell$ and $\mathrm{vol}(S) \geq \textsc{MaxBal}(H, \phi_{\ell+1})/c_{\mathrm{bal}}$))
  * If $\mathrm{vol}(S) \geq m^{1-\ell\varepsilon}/c_{\mathrm{bal}}$,
    · return $\underbrace{\textsc{Decomp}(H[S], 1)}_{\text{"left" recursion}} \cup \underbrace{\textsc{Decomp}(H[V \setminus S], \ell)}_{\text{"right" recursion}}$.
  * Else,
    · return $\underbrace{\textsc{Decomp}(H, \ell + 1)}_{\text{"down" recursion}}$

- Then, we call $\textsc{Decomp}(G, 1)$.
- Compare the difference:
  - We now have **levels**.
  - How we recurse depends on how balanced the cut S is.

2.6. Analysis

It is enough to analyze the depth of the recursion. (Why? Answer: The sum of the subgraphs at each level is just the original graph)

Consider a recursion tree $\mathcal{T}$:

- Each edge is labeled
  - „left” : small side
  - „right”: big side
  - „down”: same graph but increment level.
- Each leaf $x \in \mathcal{T}$ corresponds to an expander (because we stop the recursion).

Then, consider a recursion tree $\mathcal{T}$:

2.2. Lemma. *P can contains at most $O(\log m)$ left edges.*

Bizonyítás. Every time we go left, the volume is halved. □

2.3. Lemma. *Between two consecutive left edges in P, there are at most L many down edges.*

Bizonyítás. Between two consecutive left edges in P, the level never decreases. It increments for each down edge. After L down edges, we are at level $\ell = L + 1$. The condition

$$\mathrm{vol}(S) \geq m^{1-\ell\varepsilon}/c_{\mathrm{bal}} = 1/c_{\mathrm{bal}}$$

always holds. So we cannot have any more down recursion. □

- It remains to prove that between any left/down edges in P, there cannot be too many right edges.
- That is, there cannot be too many consecutive right edges in P.
- Now, we observe that the algorithm maintains this invariant:

2.4. Lemma. *When* $\textsc{Decomp}(H, \ell)$ *is called, we have* $\textsc{MaxBal}(H, \phi_\ell) < m^{1-(\ell-1)\varepsilon}$.

Bizonyítás. For $\ell = 1$, $\textsc{MaxBal}(H, \phi_1) \le m$ trivially holds (note $\mathrm{vol}(V)/2 = m$).

For $\ell > 1$, note that ℓ is incremented only by „down" edge. Before ℓ is increased, we have

$$\begin{aligned}&\mathrm{vol}(S) \ge \textsc{MaxBal}(H, \phi_{\ell+1})/c_{\mathrm{bal}}, \text{ and}\\ &\mathrm{vol}(S) < m^{1-\ell\varepsilon}/c_{\mathrm{bal}}\end{aligned}$$

and then we set $\ell \leftarrow \ell + 1$. So the lemma holds. □

2.5. Lemma. *There is at most* $R = m^{\varepsilon} \cdot c_{\mathrm{bal}}$ *consecutive right edges in* P.

Bizonyítás.

- Consider the *right subpath* $P' \subseteq P$ of length R.
- Let H_1 be the graph corresponds to the node at the closest to root of P'. $H_2, \ldots, H_{R+1}$ are defined similarly.
- The algorithm calls $\textsc{Decomp}(H_1, \ell), \ldots, \textsc{Decomp}(H_R, \ell)$.
  - Note that the level ℓ never changes with the right edges.
  - We have $\textsc{MaxBal}(H_1, \phi_\ell) < m^{1-(\ell-1)\varepsilon}$ by the invariant of the algorithm.
- Let $S_i = \textsc{BalCut}(H_i, \phi_{\ell+1})$.
  - $\Phi_{H_i}(S_i) < c_{\exp} \cdot \phi_{\ell+1} = \phi_\ell$.
  - $\mathrm{vol}_{H_i}(S_i) \ge m^{1-\ell\varepsilon}/c_{\mathrm{bal}}$ for all i.
- Let $\overline{S} = S_1 \cup \cdots \cup S_R$.
- Observe that $\Phi_{H_1}(\overline{S}) < \phi_\ell$.
  - Because union of ϕ_ℓ-sparse cut is a ϕ_ℓ-sparse cut.
  - This is true if $\mathrm{vol}_{H_1}(\overline{S}) \le \mathrm{vol}_{H_1}(V(H_1))/2$.
  - But this might not be true. (**Exercise: How to fix this?)**
- Suppose $R > m^{\varepsilon} \cdot c_{\mathrm{bal}}$. Then,

$$\mathrm{vol}_{H_1}(\overline{S}) \ge R \cdot m^{1-\ell\varepsilon}/c_{\mathrm{bal}} > m^{1-(\ell-1)\varepsilon} = \textsc{MaxBal}(H_1, \phi_\ell).$$

- This is a contradiction. So $R \le m^{\varepsilon} \cdot c_{\mathrm{bal}}$.

$\square$

2.6. Corollary. *The length of root-to-leaf path P is at most $O(\log n) \times L \times m^{\varepsilon} \cdot c_{\text{bal}} = \tilde{O}(m^{\varepsilon})$.*

- From here, we can conclude that expander decomposition can be computed in almost-linear time if we pay a $m^{o(1)}$ factor

2.7. Theorem (Fast Expander Decomposition)**.** *For any unweighted graph $G = (V, E)$ with m edges and any $\phi \in [0, 1]$ and a parameter $\varepsilon > 0$, we can compute a partition of vertices* $\textsc{Decomp}(G) = \{V_1, \ldots, V_k\}$ *such that*

- *$G[V_i]$ is a ϕ-expander, i.e., $\Phi(G[V_i]) \geq \phi$, and*
- *at most $\phi \log^{O(1/\varepsilon)} m$-fraction of edges crosses the partition.*

The algorithm call $\textsc{BalCut}$ *in graphs of $\tilde{O}(m^{1+\varepsilon})$ total number of edges. So it takes $\tilde{O}(m^{1+\varepsilon})$ total time.*

If we set $\varepsilon = O(1/\sqrt{\log m})$, then

- the running time is $\tilde{O}(m) \cdot 2^{O(\sqrt{\log m})} = m^{1+o(1)}$
- the fraction of crossing edges is $\phi \log^{O(\sqrt{\log m})} m = \phi m^{o(1)}$. (Instead of $\phi \log m$).

3. State of The Art

You actually saw the fastest algorithm.

There is another incomparable algorithm:

3.1. Theorem (Another Fast Expander Decomposition)**.** *For any unweighted graph $G = (V, E)$ with m edges and any $\phi \in [0, 1]$ and a parameter $\varepsilon > 0$, we can compute in $\tilde{O}(m/\phi)$ time a partition of vertices* $\textsc{Decomp}(G) = \{V_1, \ldots, V_k\}$ *such that*

- *$G[V_i]$ is a ϕ-expander, i.e., $\Phi(G[V_i]) \geq \phi$, and*
- *at most $\phi \log^3 m$-fraction of edges crosses the partition.*
- This is faster and better for large ϕ, say $\phi \geq 1/\text{polylog}(n)$.

4. Flexibility of The Algorithm

Once you define a notion of sparse cut. The definition of expander decomposition follows. (Just keep finding a sparse cut).

4.1. Exercise. Try to state expander decomposition for these versions

- ***d***-expansion
- hypergraph

- vertex expansion
- directed graphs!

So the definition itself is flexible.But the algorithm we saw today is flexible too. Why because it is just a reduction to BALCUT,

- BALCUT can be solved based on the cut-matching game.
- The cut-matching game is very flexible to all notation of expansion.

The upshot is that, based on the lecture today, you compute all variant of expander decomposition using polylog(n) approx max flow calls!