Lecture 5
Overview of the Area

Thatchaphol Saranurak
U of Michigan

August 18, 2025
ADFOCS

Plan

1. History of expander decomposition and hierarchies
2. Recent developments in the last 10 years
3. Other central concepts in the area

Part 1
Early History

1980-2000: before first appearance

Areas that motivates expander decomposition and hierarchies

1. Approximate Sparsest Cuts and Multi Cuts
   - Flow/cut characterization [Leighton Rao 88]
   - Metric embedding [Linial Longdon Rabinonich 95]

2. Spectral graph theory
   - Eigenvalue characterization: Cheeger's Inequality [Alon Milman 85]
   - Random-walk characterization: [Sinclair Jerrum 89]

3. Graph minor theory
   - Disjoint path problem [Graph Minor XIII, Robertson Seymour 95]

4. Distributed algorithms
   - Low diameter decomposition [Awerbuch Peleg 90] [Linial Saks 93]

2000-2005: first appearance

- 1998 Goldreich Ron: **(Implicit) expander decomposition**
  - For property testing
- 2000 Kannan Vempala Vetta: **Define expander decomposition**
- 2002 Räcke: **Tree flow sparsifier** (also called Räcke tree)
  - via **Boundary-separator-expanding hierarchy** [Räcke 02; Bienkowski Korzeniowski Räcke 03]
  - Build a complicated version [Harrelson Hildrum Rao 03; Räcke Shah 14]
  - Applications:
    - Oblivious routing, Online multicut
    - Minimum bisection, Min max partition
  - Open problem: is optimal quality $q = \Theta(\log n)$?
    - Tree flow sparsifiers: $O(\log^2 n \log\log n)$, $\Omega(\log n)$
    - Tree cut sparsifiers: $O(\log n \log\log n)$ existential, $O(\log^{1.5} n \log\log n)$ poly-time, $\Omega(\log n)$

2000-2005: first appearance

- 2004 – 2006 Chekuri Khanna Shepherd (4 papers):

  Expander decomposition for any node weighting

  - Applications:
    - disjoint paths, all-or-nothing flow
    - polynomial grid minor theorem [Chekuri Chuzhoy'14]
  - Generalization: vertex expansion and directed expansion [Chekuri Ene'13 and '15]

- 2007 Patraşcu Thorup: **Separator-expanding hierarchy**
  - Connectivity oracles under edge faults

Before 2005

But these concepts were not formalized in a unified way.

Relationship between Räcke's and PT's hierarchies were not explicit documented anywhere… until 2025.

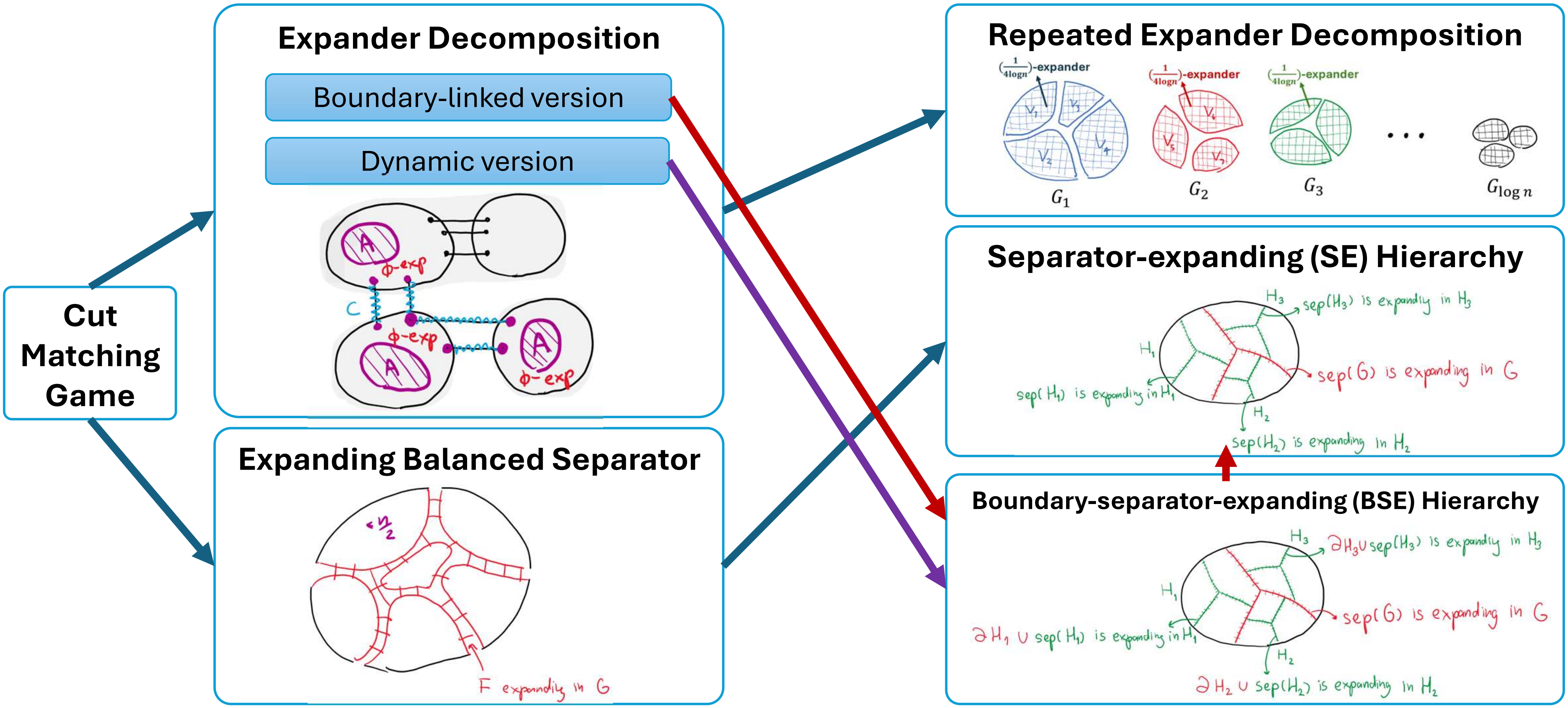
Expander decomposition, SE hierarchy, BSE hierarchy already appeared in literature.

It was unclear why they are useful for very fast algorithms yet.

Central Concepts

- Cut-matching Game

Recap key concepts



Cut-Matching Game

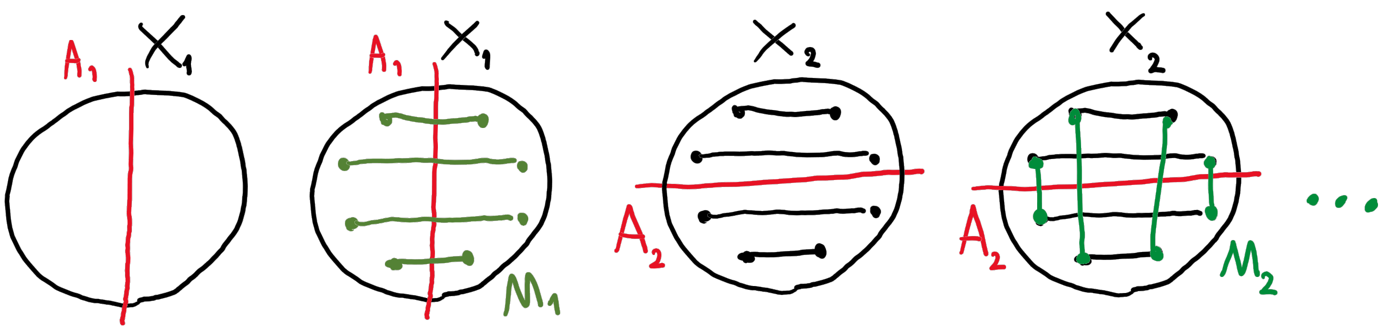
A process for building an expander
from few **adversarial** matchings

Cut-Matching Game

Init: an empty n-vertex graph $X_1 = (U, \emptyset)$

Round i:

- You choose a bisection (A_i, B_i) where $|A_i| = |B_i|$
- An adversary chooses a perfect matching M_i between A_i and B_i
- $X_{i+1} \leftarrow X_i \cup M_i$



Goal:
Choose (A_i, B_i) cleverly, so that after a few rounds X must be a (connected) expander

Cut-Matching Game

Init: an empty n-vertex graph $X_1 = (U, \emptyset)$

Round i:

- You choose a bisection (A_i, B_i) where $|A_i| = |B_i|$
- An adversary chooses a perfect matching M_i between A_i and B_i
- $X_{i+1} \leftarrow X_i \cup M_i$

Theorem (KKOV'07): $\exists$ deterministic strategy such that, after $R = O(\log n)$ rounds,
1_U is $(1/10)$-expanding in X_R.
(So X_R is a $\Omega(1/R)$-expander.)

[KKOV]: Slow, "Chicken & Egg"
[KRV, OSVV]: Fast, Randomized

Strategy: for each round i,

- $(S_i, V - S_i) \leftarrow$ most balanced cut s.t. $|E(S_i, V - S_i)| < \frac{1}{10}\min\{|S_i|, |V - S_i|\}$.
- Choose any A_i where $S_i \subseteq A_i$.

Approx Expansion via Cut-Matching Game

Assume: $\exists$cut strategy s.t. 1_U is ϕ-expanding in X_R after R rounds.

Theorem: For any G and $U \subseteq V$**,** can guarantee either
- 1_U is ϕ/R-expanding in G
- 1_U is not 1-expanding in G

Using R maxflow calls, plus time to run the cut strategy.

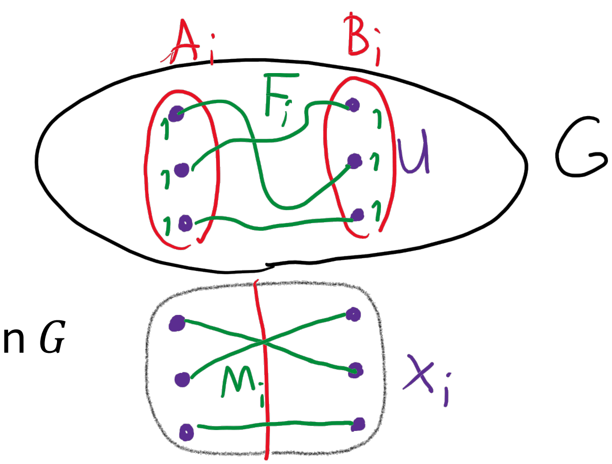
$\Rightarrow$ can $O(\frac{R}{\phi})$-approximate how much 1_U is expanding in G.
This extends to any node weighing A.

Approx Expansion via Cut-Matching Game

Assume: $\exists$cut strategy s.t. 1_U is ϕ-expanding in X_R after R rounds.

Algo:

- Init cut-matching game on $X_1 = (U, \emptyset)$.
- In round $i \leq R$
  - $(A_i, B_i) \leftarrow$ cut chosen by the cut player
  - $F_i \leftarrow$ (integral) max flow between 1_{A_i} and 1_{B_i}
  - If $\mathrm{val}(F_i) < |A_i|$, **report:** 1_U is not 1-expanding in G
  - If $\mathrm{val}(F_i) = |A_i|$
    - $M_i \leftarrow$ perfect (A_i, B_i)-matching routed by F_i
    - $X_{i+1} \leftarrow X_i \cup M_i$
- **Report:** 1_U is ϕ/R-expanding in G



D: any 1_U-respecting demand.
- D is routable in X_R with cong $1/\phi$ (by cut player)
- X_R is routable in G with cong R (by $F_1, \ldots, F_R$)

$\Rightarrow$ D is routable in G with cong R/ϕ

Central Concepts

- Probabilistic Tree Flow Sparsifiers (and j-Trees)

Recall: Tree Flow Sparsifiers

Def: A **tree flow sparsifier** T of G with quality q:

1. A capacitated tree spanning $V(G)$
2. For any $\deg_G$-respecting demand D
   - If D is routable in $G \Rightarrow D$ is routable in T
   - If D is routable in $T \Rightarrow D$ is routable in G with congestion q

Probabilistic Tree Flow Sparsifiers

Def: A probabilistic tree flow sparsifier $\mathcal{T}$ of G with quality q:

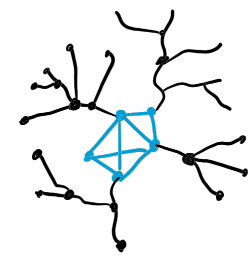
1. Distribution $\mathcal{T}$ over capacitated trees spanning $V(G)$
2. For any $\deg_G$-respecting demand D
   - If D is routable in $G \Rightarrow D$ is routable in each $T \in \mathcal{T}$
   - If D is routable in $\mathbb{E}_{T\in\mathcal{T}}[T] \Rightarrow D$ is routable in G with congestion q

[Räcke'08]: $\mathcal{T}$ with optimal quality $\Theta(\log n)$ in poly time

- Better than the single tree version.
- But fail in applications, e.g., min-max partition
- Also called "Räcke trees". Do not get confused.

Probabilistic Almost-Tree Flow Sparsifiers

Def: A j-tree = tree + graphs on j vertices. (**Think**: j is small. This is almost a tree)



Def: A probabilistic j-tree flow sparsifier $\mathcal{T}$ of G with quality q:

1. Distribution $\mathcal{T}$ over j-trees spanning $V(G)$
2. For any $\deg_G$-respecting demand D
   - If D is routable in $G \Rightarrow D$ is routable in each $T \in \mathcal{T}$
   - If D is routable in $\mathbb{E}_{T\in\mathcal{T}}[T] \Rightarrow D$ is routable in G with congestion q

[Madry'10]: $\mathcal{T}$ over $n^{o(1)}$-tree with quality $n^{o(1)}$ in $m^{1+o(1)}$ time

- The construction is not based on expanders

Part 2

History: 2005-2015

2005-2015: impact to fast algorithms

- 2004 Spielman Teng:

  (Weak) expander decomposition in $\tilde{O}(m/\phi^{O(1)})$ time

  - Approx expansion of G in near-linear time
  - Application: Laplacian solver in near-linear time (impactful!)

- 2006 Khandekar, Rao, and Vazirani:

  Cut matching game

  - Approx expansion of any A in *approx max flow* time
  - More versions [Orecchia Schulman Vazirani Vishnoi'08] [Khandekar Khot Orecchia Vishnoi'07]

- 2008 Räcke: **probabilistic tree flow sparsifiers** poly time

2005-2015: impact to fast algorithms

- 2010 Madry: **probabilistic almost-tree flow sparsifiers** $m^{1+o(1)}$ time
- 2013 Sherman and KLOS: **Approximate max flow** $m^{1+o(1)}$ time
  - Reduce to **probabilistic almost-tree flow sparsifiers.**
- 2014 Räcke Shah Täubig: **Tree flow sparsifiers** $m^{1+o(1)}$ time
  - Reduce to **approximate max flow**
- 2016 Peng: **Approximate max flow** $\tilde{O}(m)$ time
  - Resolve "chicken and egg"

2005-2015: impact to fast algorithms

10 years after [Spielman Teng],

it was clear that expanders are very powerful for fast algorithms.

Part 3

Survey: 2015 - Now

Two dimensions of development

Expander-based techniques hugely extend in 2 dimensions

1. Across **models of computation**
2. Across **notions of expansion**

Part 3.1
Development across models of computation

Models of Computation

Static

Dynamic

Distributed

Models of Computation

Static

Dynamic

Distributed

Static Core Objects: Faster

- 2004 Spielman Teng: **(Weak) expander decomposition** $\tilde{O}(m/\phi^{O(1)})$ time

- 2017-Now: **Expander decomposition**
  - 2017 Nanongkai S Wulff-Nilsen: $m^{1+o(1)}$ time
  - 2019 S Wang: $O(m \log^4(n)\,/\phi)$ time, simple
  - 2023 Li Nanongkai Panigrahi S: $\tilde{O}(m)$

- All these are randomized

Static Core Objects: Deterministic

- 2020 CGLNPS: **Deterministic expander decomposition** $m^{1+o(1)}$ time
  - **Deterministic cut-matching game** $m^{1+o(1)}$ time
  - Applications:
    - Deterministic Laplacian solvers, sparsifiers,
    - **many** deterministic dynamic algos
- Open: **Deterministic expander decomposition** $\tilde{O}(m/\phi^{O(1)})$ time
  - Important! will speed up and simplify MANY algorithms.

Applications in Static Setting

Previous:

- **Approximate max flow** $\tilde{O}(m)$ time

New applications:

- **Deterministic global mincut** $\tilde{O}(m)$ time [KT'15, LP'20, Li'21, HLRW'25]
  - Use (or try to bypass) deterministic expander decomposition
- **Connectivity labeling scheme under edge faults** [LPS'25]
- **Parallel approximate max flow and Gomory Hu trees** [LNPS'21] [AKLPWWZ'24]
- Also, **exact max fow** but that is through the dynamic version. Will talk more about this.

Models of Computation

Static

Dynamic

Distributed

Dynamic Graph Problems

- **Input:** a graph G
- **Then: a sequence of updates** (edge insertions/deletions)

- **Task:**
  - Maintain objects (e.g. minimum spanning tree of G), or
  - Support queries (e.g. can query if u and v are connected in G)

- **Goal:** Fast **update time** (time to process each update)
  - Ideally, $\mathrm{polylog}(n)$ time

Dynamic Core Objects

- 2007: Exploit expanders for handling only **one batch of updates**
  - [Patrascu Thorup]'s connectivity oracles
  - But the dynamic setting has an online sequence of updates
- 2017 Nanongkai S Wulff-Nilsen**: Dynamic Expanders Decomposition**
- 2021 Goranci Räcke S Tan: **Dynamic Tree Flow Sparsifiers**
  - Dynamic BSE-hierarchy
- Last few years: Many uses of expanders in dynamic graphs
  [SW'19] [CK'19] [CGLNPS'20] [BGS'20] [GRST'21] [CS'21] [BGS'21] [JS'21] [BKMNSS'22] [BBGNSSS'22] [LS'22] [GHNSTW'23] [JST'24] [EHL'25] [HLS'25] [CP'25]

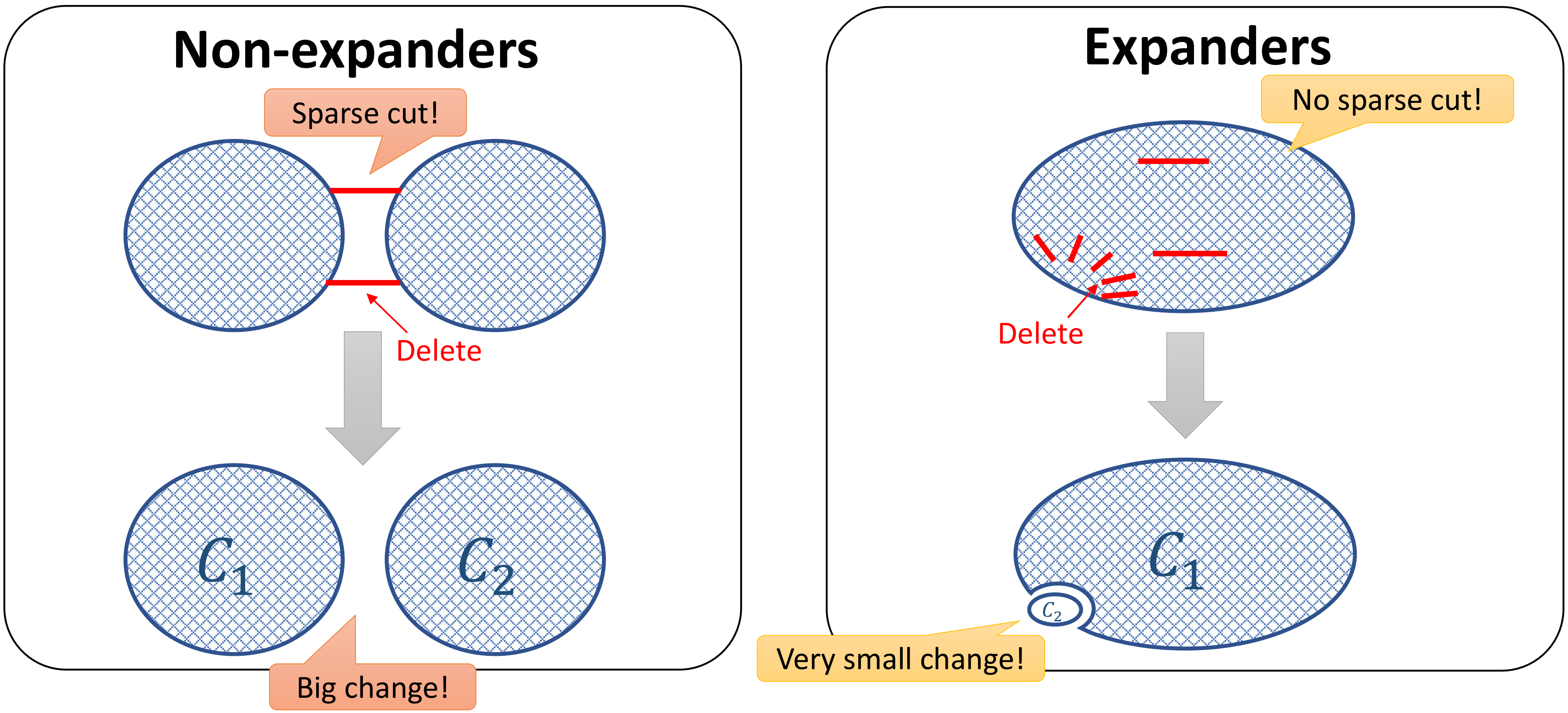
Applications in Dynamic Setting

- Fastest dynamic algorithms for
  - Minimum spanning tree [NSW'17]
  - k-edge connectivity [Jin Sun'21]
  - $n^{o(1)}$-approx max flow [GRST'21]
  - Exact/approx minimum cut [GHNSTW'23,JST'24,EHL'25]
- Fastest deterministic dynamic algorithms for
  - Decremental Reachability [BGS'20]
  - Decremental Shortest path [CK'19, CS'21, BGS'21]
  - Spanners [CKLPPS'22, CP'25]
  - Distance oracle [HLS'25]
- Everything relied on **Expander Pruning**

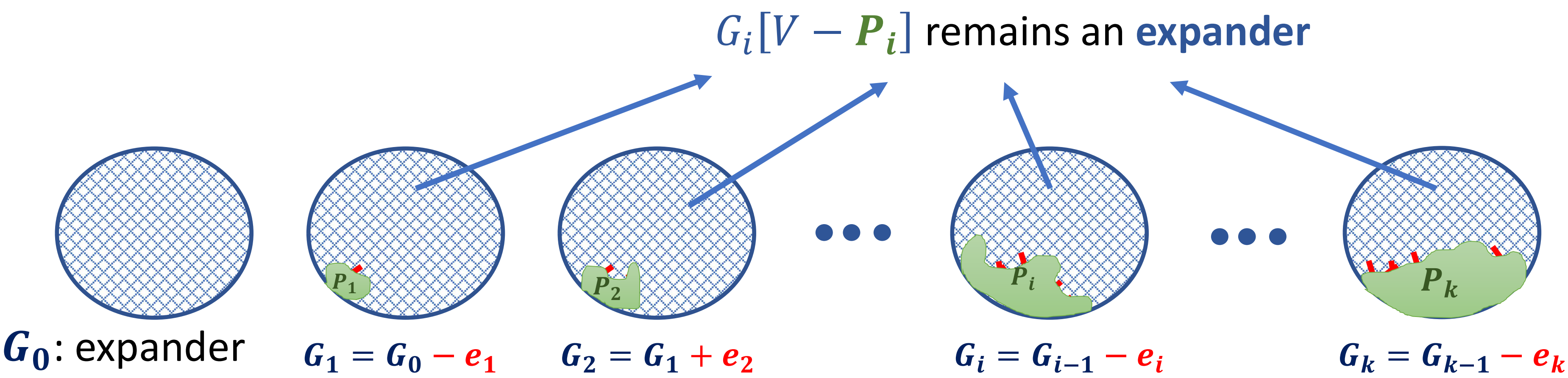
Central Concepts

- Expander Pruning

Intuition: Expanders are Robust under updates.



Expander Pruning [NSW'17] [SW'19] [MPS'25]



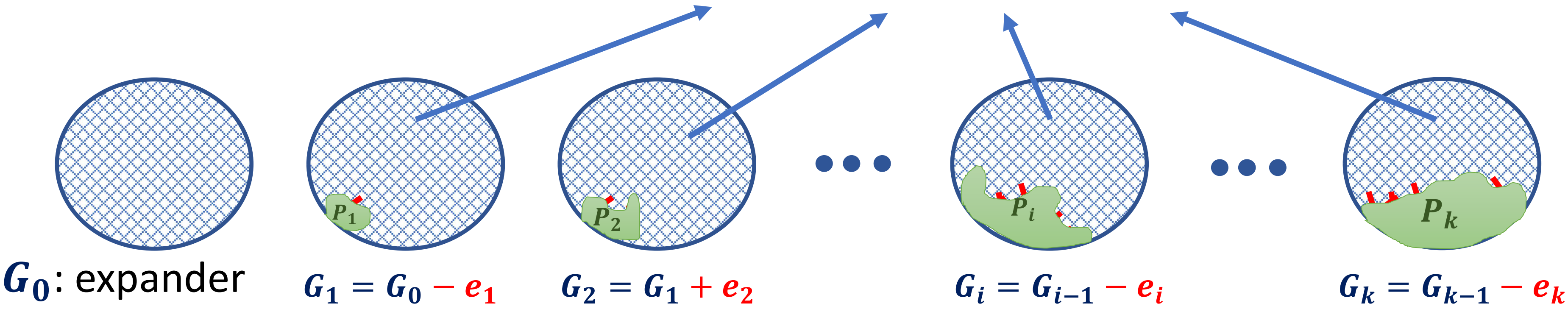
Alg. maintains a vertex set P such that:

1. P grows very slowly: $\deg(P_i) \leq i \cdot \mathbf{polylog(n)}$
2. Update P_{i-1} to P_i in $\mathbf{polylog(n)}$ time

where $k \leq m/n^{o(1)}$

Expander Pruning [NSW'17] [SW'19]

$G_i[V - P_i]$ remains an **expander**



Each update can only cause **"small problem"**.
The remaining part is still an **expander**.

More Applications:
Dynamic Expanders for Exact Max Flow

All modern max flow algorithms exploit dynamic algo for expanders

Exact max flow:

- BLNPSSSW'20, BLLSSSW'21 $\tilde{O}(m + n^{1.5})$ time
  - **dynamic spectral sparsifiers (via dynamic ED)**
- CHKLPPS'22, CKLMP'24 $m^{1+o(1)}$ time
  - **dynamic spanners (via dynamic ED)**
- BCKLMPS'24: $m^{1+o(1)}$ time (simplest)
  - Its only core component: **dynamic tree flow sparsifiers**

Open:

Dynamic tree flow sparsifiers with quality $\mathrm{polylog}(n)$ in $\mathrm{polylog}(n)$ update time

$\Rightarrow \tilde{O}(m)$-time max flow!

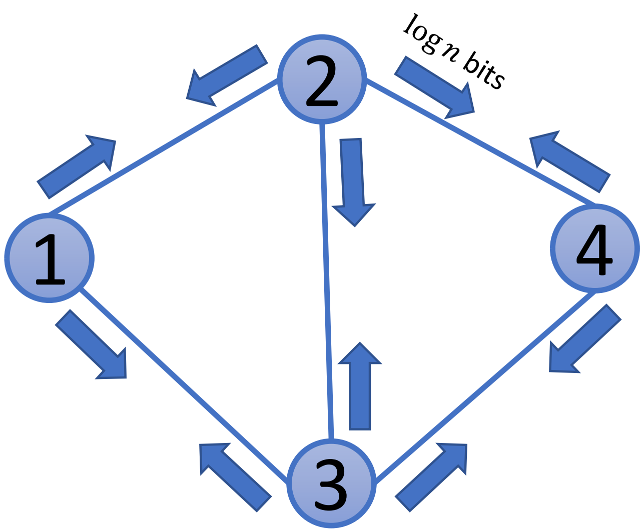
Models of Computation

Static

Dynamic

Distributed

Distributed Model: CONGEST

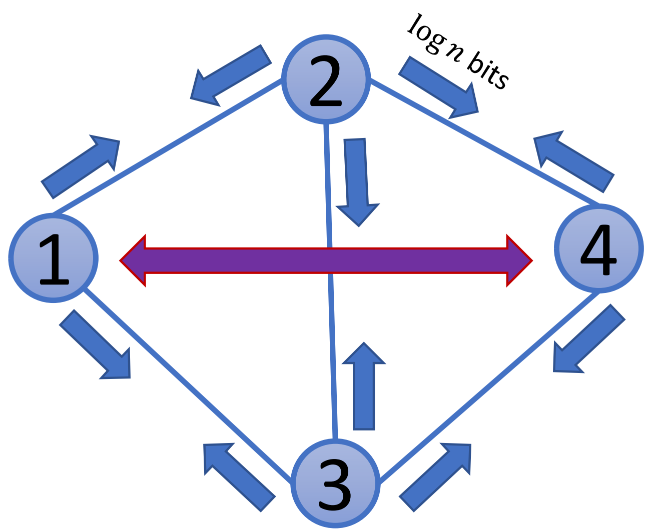


- **Local communication**:
  A node can send a message <u>to each of its neighbors</u> in each *round*
- **Bounded Bandwidth:**
  Each message has size $O(\log n)$-bit

Goal:
- Compute something about the underlying network
- Minimize the number of rounds

Distributed Model: CONGESTED CLIQUE

Stronger model than **CONGEST**



- **All-to-all communication**:
  A node can send a message to everyone in each *round*
- **Bounded Bandwidth:**
  Each message has size $O(\log n)$-bit

Goal:
- Compute something about the underlying network
- Minimize the number of rounds

Without Expander Decomposition

CONGEST

Triangle Listing:
$O(n^{3/4})$ [IG'17]

k-Clique Listing:
$O(n^{2-\Theta(\frac{1}{k})})$ [EFF+'19] (detection only)

CONGESTED CLIQUE

Triangle Listing:
$\Theta(n^{1/3})$
[DLP'12, IG'17, PRS'18]

k-Clique Listing:
$\Theta(n^{1-2/k})$
[DLP'12, FGKO'18]

Using Expander Decomposition and Expander Routing

Will define soon!

CONGEST

Triangle Listing:

$O(n^{1/2})$[CPZ'19],

$\Theta(n^{1/3})$ [CS'19 '20]

k-Clique Listing:

$O(n^{1-2/(k+2)})$[CGL'19]

$\Theta(n^{1-2/k})$[CCGL'21] [CLV'22]

CONGESTED CLIQUE

Triangle Listing:

$\Theta(n^{1/3})$

[DLP'12, IG'17, PRS'18]

k-Clique Listing:

$\Theta(n^{1-2/k})$

[DLP'12, FGKO'18]

Bypassing Locality

CONGEST model ← **CONGESTED CLIQUE** model

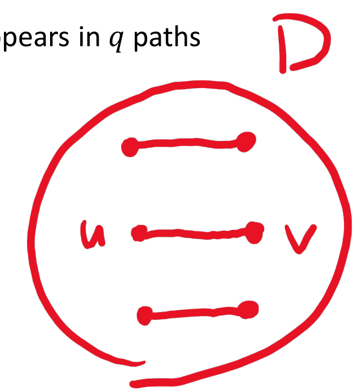
Expander Decomposition & Routing
can transfer strong results
from CONGESTED CLIQUE (without locality constraint)
to CONGEST (with locality constraint)

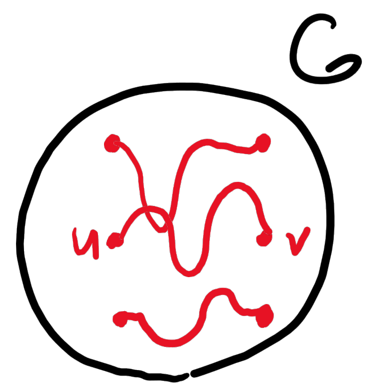
Central Concepts

- Expander Routing

Expander Routing with quality q

- **Input:** $\widetilde{\Omega}(1)$-expander G
- **Query:**
  - Given, a $\deg_G$-respecting demand D,
  - Find a flow F routing D
    - **(Short):** Each flow path has length q
    - **(Low congestion):** $\mathrm{cong}(F) \leq q$ Each edge appears in q paths





- Think $q = n^{o(1)}$

Survey: Distributed Expander-based Algorithms

- **Expander Routing**
  - **Known**: quality $q = n^{o(1)}$ in $m^{1+o(1)}$ time (even $n^{o(1)}$ rounds in CONGEST)
    [Ghaffari Kuhn Su'17] [Ghaffari Li'18] [Chang S'20] [Chuzhoy S'21] [Chang Huang Su'24]
  - **Open:** quality $q = \text{polylog}(n)$ in $\tilde{O}(m)$ time (or $\tilde{O}(1)$ rounds in CONGEST)
    - Very important!
    - Bottleneck for so MANY problems (static, dynamic, distributed)
- **Distributed Expander Decomposition**
  - Randomized: $\tilde{O}\big(1/\text{poly}(\phi)\big)$ rounds [Chang Pettie Zhang'19] [Chang S'19] [Chen Meierhans Probst S'25]
  - Deterministic: $n^{o(1)}$ rounds for $\phi \geq 1/n^{o(1)}$ [Chang S'20]

Why Expander Routing is Useful in CONGEST?

A node u can exchange $\deg_G(u)$ messages
with **any set of nodes**
in $q = n^{o(1)}$ **rounds** in an **expander**

Expanders allow **all-to-all communication**
like in CONGESTED CLIQUE
(but with small overhead and at most $\deg_G(u)$ messages).

Local communication
u can exchange $\deg_G(u)$ messages with **only neighbors** in 1 round

Part 3.2
Development across notions of expansion

Notions of Expansion

Vertex Expansion

Directed Expansion

Length-Constrained Expansion

Unbreakability

Hybrid Expansion

Promising Research Directions

- Applications of expander hierarchies for other notions of expansion

- SE hierarchy
  - It admits $m^{1+o(1)}$-time algorithms for many notions of expansion
  - This notion is not well-known.
  - More applications for other notions of expansion?

- BSE hierarchy
  - Previous algorithms only works for edge-capacitated undirected graphs
  - Our construction works on many notions of expansion
  - More applications for other notions of expansion?

Notions of Expansion

Vertex Expansion

Directed Expansion

Length-Constrained Expansion

Unbreakability

Hybrid Expansion

Using Vertex Expander Decomp/Hierarchies

Using Expander Decomposition

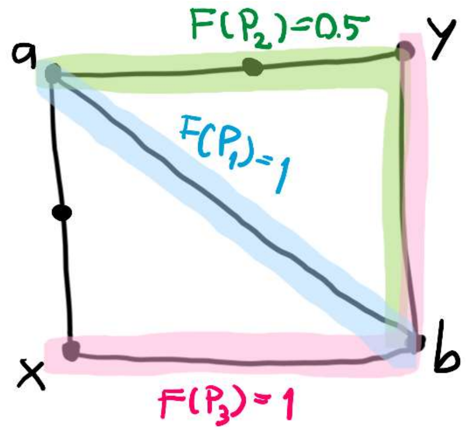
- **Deterministic vertex connectivity** [SY'22] [JNSY'25]

Using Expander hierarchies

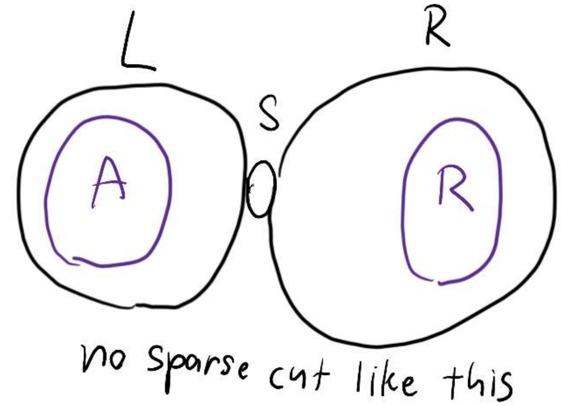
- **Connectivity oracles under vertex faults** [LS'22] [LY'24]
- **Connectivity labeling scheme under vertex faults** [LPS'25]

Vertex Expansion

- Vertex congestion:
  - $\mathrm{vcong}_F(v) = F(e)/\mathrm{cap}(v)$
  - $\mathrm{vcong}(F) = \max_v \mathrm{vcong}_F(v)$



$\mathrm{vcong}(F) = 2.5$



- A is ϕ-vertex-expanding in G if
  - **Flow view**: every A-respecting demand is routable with v-congestion $1/\phi$
  - **Cut view:** for every vertex cut (L, S, R) (i.e. $L \uplus S \uplus R = V$ and $E(L, R) = \emptyset$)

$$\mathrm{cap}(S) \geq \phi \min\{A(L \cup S), A(R \cup S)\}$$

- G is a ϕ-vertex-expander if $\vec{1}_V$ is ϕ-vertex-expanding in G

Quiz: which ones are vertex expanders?

✔ 1. Cliques
✔ 2. Hypercubes
✘ 3. Stars
✘ 4. Paths

Vertex Expander Decomposition

- **ϕ-Expander decomposition** of G: there is $C \subseteq V$ where
  - Each component of $G - C$ is a ϕ-v-expander
  - $|C| \leq \phi n \log n$

Vertex Expander Hierarchy

- **Separator-expanding hierarchy:** ∃ partition $V_0, \ldots, V_{\ell=O(\log n)}$ of V
  - $\vec{1}_{V_i}$ is ¼-v-expanding in $G - V_{>i}$ for each i.

Not clear how to define boundary-separator-expanding hierarchy

Notions of Expansion

Vertex Expansion

Directed Expansion

Length-Constrained Expansion

Unbreakability

Hybrid Expansion

Using Directed Expander Decomp/Hierarchies

Using Expander Decomposition

- **Decremental single-source reachability** [BGS’20]

Using Expander Hierarchies

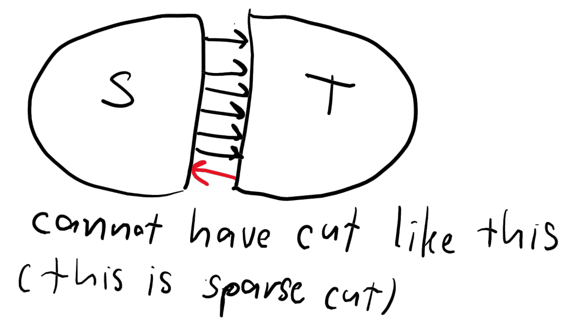
- **Combinatorial max flow** [BBST’24, BBLST’25]
  - Approach: just augmenting path
  - Technical part: ~~(100+ pages)~~ 25 pages
  - Use basic tools (Dijkstra, SCC, etc.) except expander decomposition
- **Fault-tolerant strong-connectivity preservers** [HSW’25]

Directed Expansion

- A is ϕ-expanding in directed graph G if
  - **Flow view**: every A-respecting *(symmetric)* demand is routable with cong $1/\phi$
  - **Cut view:** for every cut $(S, V - S)$

$$\text{cap}(S, V - S)) \geq \phi \min\{A(S), A(V - S)\}$$

- G is a ϕ-expander if $\deg_G$ is ϕ-expanding in G
  - $\deg_G$ counts both in and out degree.



Exactly same notation as in the undirected case.

Directed Expander Decomposition

- **ϕ-Expander decomposition** of G: there is $C \subseteq E$ where
  - Each strong component of $G - C$ is a ϕ-expander
  - $|C| \leq \phi m \log n$

Directed Expander Hierarchies

- **Separator-expanding hierarchy:** $\exists$ partition $E_0, \ldots, E_{\ell=O(\log\ \ )}$ of V
  - E_i is ¼-expanding in $G - E_{>i}$ for each i.
- **BSE hierarchy:** $\exists$ partition $E_0, \ldots, E_{\ell=O(\log n)}$ of V
  - $E_{\geq i}$ is $\Omega(1/\log n)$-expanding in $G - E_{>i}$ for each i.

Notions of Expansion

Vertex Expansion

Directed Expansion

Length-Constrained Expansion

Unbreakability

Hybrid Expansion

Flow Problems

encompass many problems on graphs

Distance
(ℓ_1)

Shortest paths
Transshipments
Steiner trees
LOCAL model

Distance & Congestion
(ℓ_1 & ℓ_∞)

Min cost flow
Min-cost multi-commodity flow
Network design (e.g. k-ECSS)
CONGEST model

Congestion
(ℓ_∞)

Max flow
Multi-commodity flow
Tree packing
CONGESTED CLIQUE model

Low diameter decomposition
since 1980, 100s of papers

Length-Constrained Expander decomp
since 2020

Expander decomposition
since 2000, 100s of papers

Graph Decomposition

key algorithmic technique

Using Length-Constrained Expander Decomp/Hierarchies

- **Universally optimal distributed algorithms** [HRG'22]

Using flow shortcut (see Exercise 2) via expander hierarchies

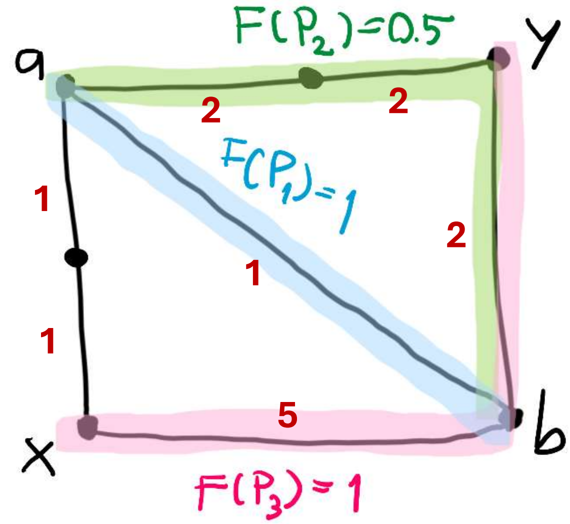
- **Dynamic distance oracle** [HLS'24]
  - First deterministic: $O_\epsilon(1)$-approx in n^ϵ update time
- $\boldsymbol{O_\epsilon(1)}$**-approx Multi-commodity flow** in $m^{1+\epsilon}$ time [HHLRS'24]
- **Parallel** $\boldsymbol{(1+\epsilon)}$**-approx min-cost flow** in $m^{1+o(1)}$ time [HJLSW'25]

- **Fault-tolerant distance oracles/labeling schemes** [HLRS'26]
- Approachable open problems: fault-tolerant roundtrip spanners

Flow and Embedding

- Graphs are
  - **undirected**, with **edge-lengths**, **unit-capacity** (for simplicity)

- (Multi-commodity) flow F
  - Congestion: $\mathrm{cong}(F) = \max_e F(e)$
  - Length: $\mathrm{len}(F) =$ max total length in flow paths

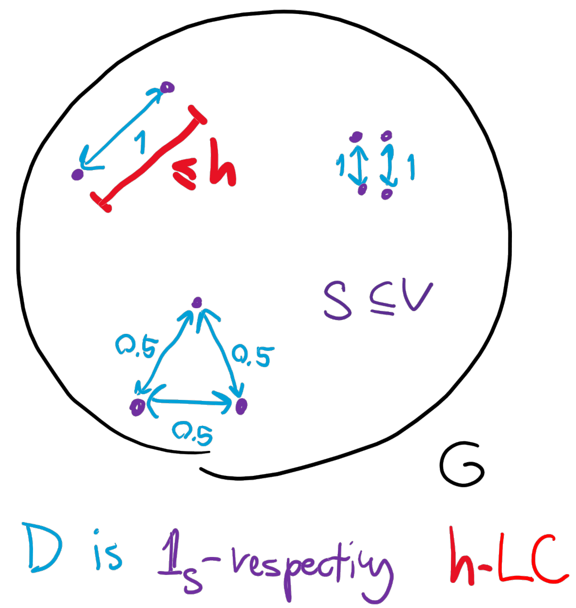


Example:
- $\mathrm{cong}(F) = 1.5$
- $\mathrm{len}(F) = 7$

Length-Constrained Demands

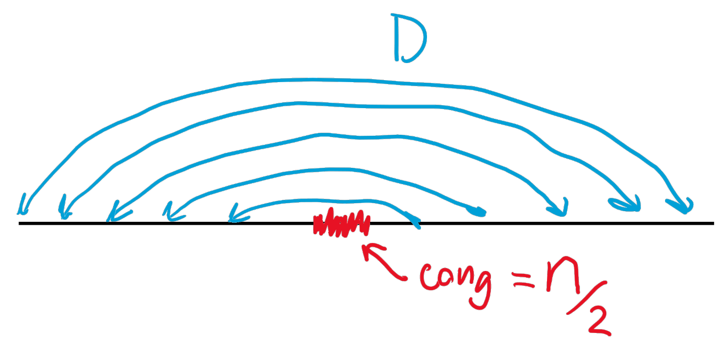
Demand D is ***h*-LC** if,

for each $D(a,b) > 0$, $\mathrm{dist}_G(a,b) \leq h$



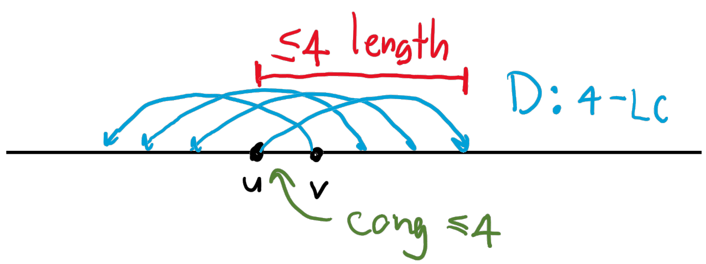
Length-Constrained Demands are Easier

G is a path and $A = 1_{V(G)}$



- A is **not** routable with congestion $< n/2$

- A is 4-LC routable with length 4 and congestion 4



LC-Expanders: Flow-based Definitions

Recall: G is ϕ-expander $\Leftrightarrow$
$\deg_G$ is routable with congestion $1/\phi$

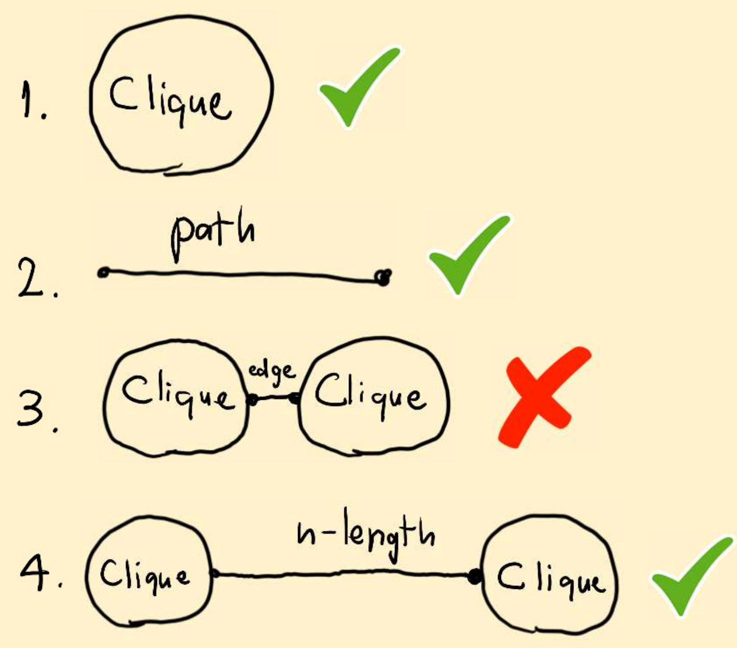
G is (h, s)-LC ϕ-expander $\Leftrightarrow$
$\deg_G$ is h-LC routable with length hs and congestion $1/\phi$

A is (h, s)-LC ϕ-expanding $\Leftrightarrow$
A is h-LC routable with length hs and congestion $1/\phi$

Quiz:

Let $h, s = O(1)$ and $\phi = \Omega(1)$

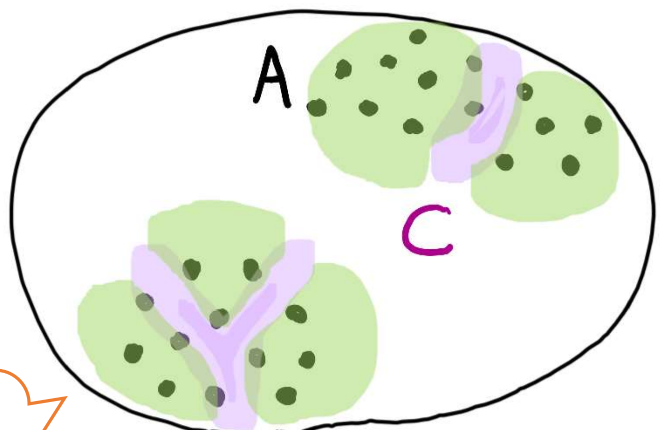
Which is not (h, s)-LC ϕ-expander ?



Expander Decomposition

Theorem: Given $G = (V, E), A, \phi$, there exists an hs-LC cut $C \subseteq E$

- A is (h, s)-LC ϕ-expanding in $G - C$.
- $|C| \leq (\phi n^{O(1/s)} \log n) \cdot |A|$



C does not decompose graphs into connected components anymore! But our notations work!

LC cut **(i.e. length Increase)**

- hs-LC cut is

$$C : E \to \left\{0, \frac{1}{hs}, \frac{2}{hs}, \ldots, 1\right\}$$

- Length in $G - C$

$$\text{len}_{G-C}(e) = \text{len}_G(e) + hs \cdot C(e)$$

Notions of Expansion

Vertex Expansion

Directed Expansion

Length-Constrained Expansion

Unbreakability

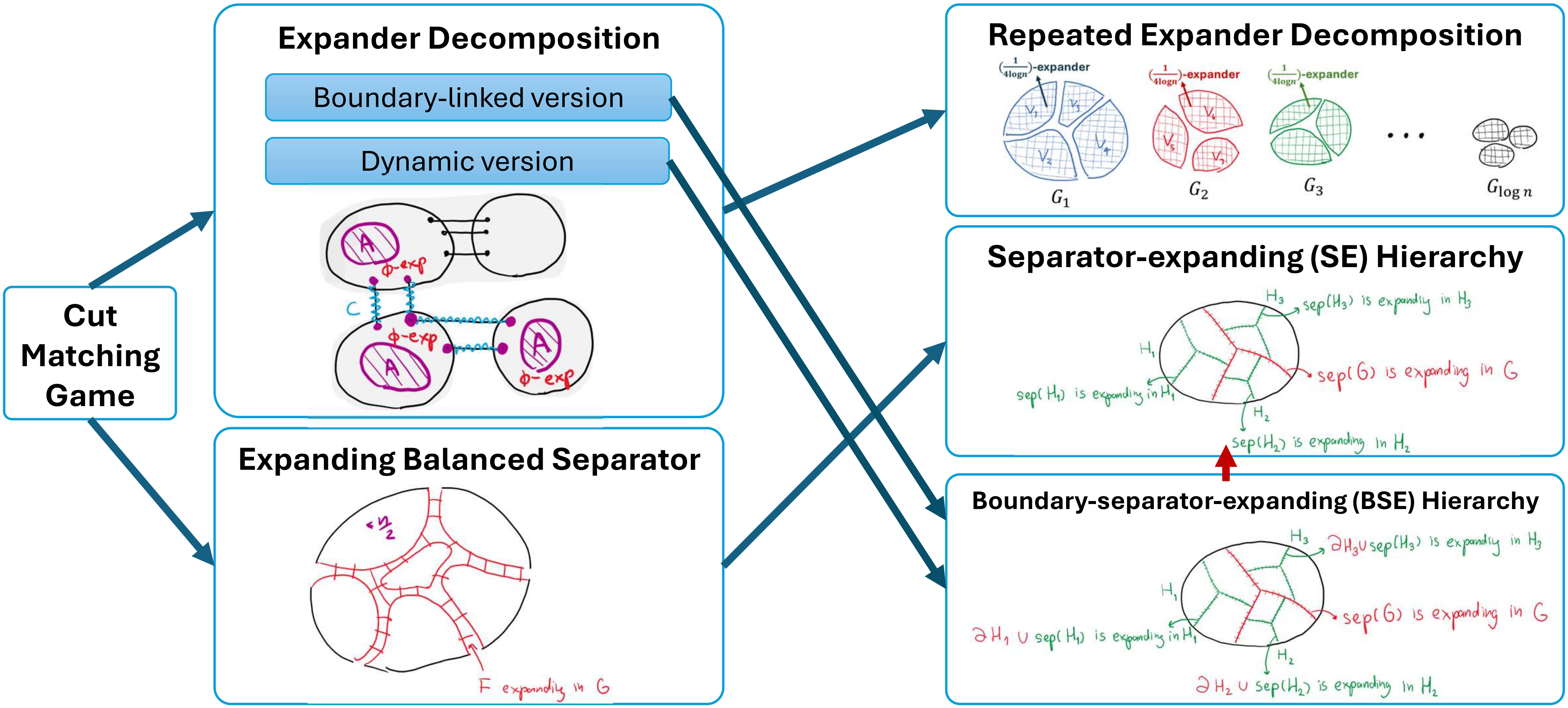
Hybrid Expansion

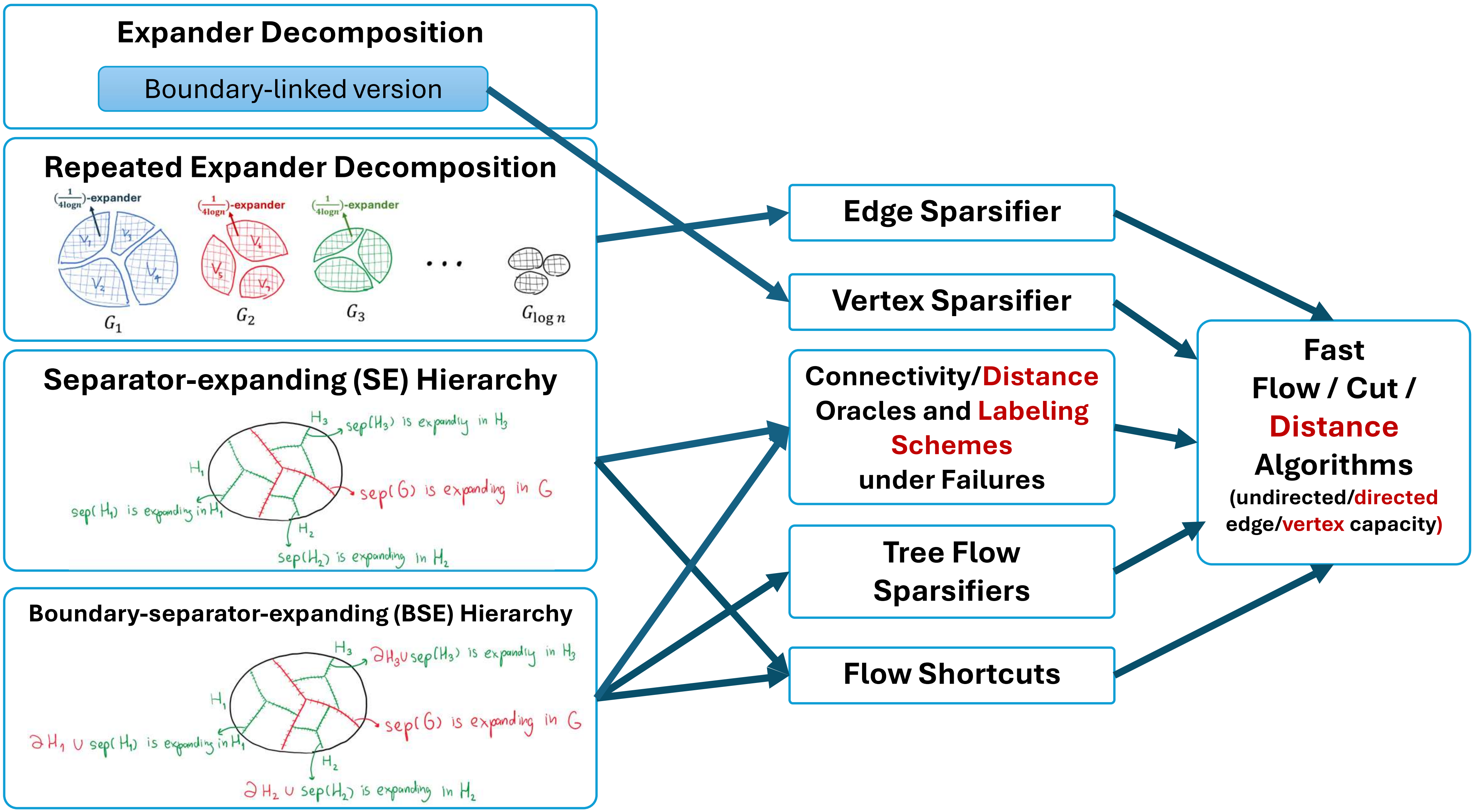
Notions of Expansion

- Vertex Expansion
- Directed Expansion
- Length-Constrained Expansion
- Unbreakability
- Hybrid Expansion

Summary

Recap key concepts




Expander Decomposition
Boundary-linked version
Repeated Expander Decomposition
$(\frac{1}{4\log n})$-expander
$(\frac{1}{4\log n})$-expander
$(\frac{1}{4\log n})$-expander
G_1
G_2
G_3
$G_{\log n}$
Separator-expanding (SE) Hierarchy
sep(H₃) is expanding in H₃
sep(G) is expanding in G
sep(H₁) is expanding in H₁
sep(H₂) is expanding in H₂
Boundary-separator-expanding (BSE) Hierarchy
∂H₃ ∪ sep(H₃) is expanding in H₃
sep(G) is expanding in G
∂H₁ ∪ sep(H₁) is expanding in H₁
∂H₂ ∪ sep(H₂) is expanding in H₂
Edge Sparsifier
Vertex Sparsifier
Connectivity/Distance Oracles and Labeling Schemes under Failures
Tree Flow Sparsifiers
Flow Shortcuts
Fast Flow / Cut / Distance Algorithms
(undirected/directed edge/vertex capacity)