Expander Decomposition: Applications and How to use it

Thatchaphol Saranurak
Toyota Technological Institute at Chicago

ADGA 2019
Goal of this talk

1. Motivate dynamic algorithms
2. Expander Decomposition through dynamic graph applications.
3. How it is also used for centralized and distributed algorithms.
4. Quick survey of applications
Part 1
Dynamic Algorithms: What and Why?

*Also say Dynamic Data Structures as well.*
Analyze **dynamic networks**

- Road networks

- Track communities in social networks

---

**Subroutines** in static algorithms

- "Inside" Shortest Paths

- "Inside" Linear Programming
A common theme

We solve the **same** problem **repeatedly** where input keeps **changing**
Dynamic Algorithms

Science of how **not** to compute things from scratch

High-level goal:

“How to Efficiently Prepare for Changes”
# Example: Dynamic Problems

<table>
<thead>
<tr>
<th>Level</th>
<th>Textbook</th>
<th>Research</th>
</tr>
</thead>
<tbody>
<tr>
<td>Change</td>
<td>Insert/delete a number in a set $S$</td>
<td>Insert/delete an edge in a graph $G$</td>
</tr>
<tr>
<td>Maintain</td>
<td>Minimum number in $S$</td>
<td>Is $G$ connected?</td>
</tr>
<tr>
<td>Recompute</td>
<td>$O(</td>
<td>S</td>
</tr>
<tr>
<td>Can do</td>
<td>$O(\log</td>
<td>S</td>
</tr>
</tbody>
</table>

**Example:\text{polylog}(n)\text{ randomized}\**

[Sleator Tarjan STOC’81, Frederickson, STOC’83
Eppstein et al, FOCS’92, Henzinger King, STOC’95
Holm et al, STOC’98, Thorup, STOC’00
Patrascu Demaine, STOC’04, Patrascu Thorup, FOCS’07
Kapron et al, SODA’13, Wulff-Nilsen, SODA’13
Huang et al, SODA’17, Nanongkai S, STOC’17
Wulff-Nilsen, STOC’17, Nanongkai S Wulff-Nilsen, FOCS’17]

**Many open questions...**
### Dynamic Connectivity / MST
- Sleator Tarjan, STOC’81
- Frederickson, STOC’83
- Eppstein et al, FOCS’92
- Henzinger King, STOC’95
- Holm et al, STOC’98
- Thorup, STOC’00
- Patrascu Demaine, STOC’04
- Patrascu Thorup, FOCS’07
- Kapron et al, SODA’13
- Wulff-Nilsen, SODA’13
- Huang et al, SODA’17
- Nanongkai et al, STOC’17
- Wuill-Nilsen, STOC’17
  
  Nanongkai & Wuill-Nilsen, FOCS’17

### Dynamic (Directed) Reachability
- Even Shiloach, JACM’81
- Henzinger King, FOCS’95
- King Sagert, STOC’99
- Demetrescu Italiano, FOCS’00
- Rodditty Zwick, FOCS’02
- Sankowski, FOCS’04
- Lacki, SODA’11
- Henzinger et al, STOC’14
- Chechik et al, FOCS’16
- Italiano et al, STOC’17
- Bernstein et al, STOC’19

### Dynamic Maximum Matching
- Sankowski, SODA’07
- Onak Rubinfeld, STOC’10
- Baswana et al, FOCS’11
- Gupta Peng, FOCS’13
- Neiman Soloman, STOC’13
- Bosek et al, FOCS’14
- Gupta et al, SODA’14
- Bhattacharya et al, SODA’15
- Bernstein Stein, SODA’16
- Peleg Soloman, SODA’16
- Bhattacharya et al, STOC’16
- Soloman, FOCS’16
- Bhattacharya et al, SODA’17

### Dynamic Single-Source Shortest Path
- Even Shiloach, JACM’81
- Ausiello et al, SODA’90
- Rodditty Zwick, FOCS’04
- Bernstein Riditty, SODA’11
- Bernstein, STOC’13
- Henzinger et al, SODA’14
- Henzinger et al, STOC’14
- Henzinger et al, FOCS’14
- Bernstein Chechik, STOC’16
- Bernstein Chechik, SODA’17
- Chuzhoy Khanna, STOC’19
- Probst Wulff-Nilsen, SODA’20
- Probst Wulff-Nilsen, SODA’20

### Dynamic All-Pairs Shortest Path
- King, FOCS’99
- Demetrescu Italiano, FOCS’00
- Fakcharoenphol Rao, FOCS’01
- Baswana et al, STOC’02
- Baswana et al, SODA’03
- Rodditty Zwick, FOCS’04
- Thorup, STOC’05
- Bernstein, SODA’09
- Abraham et al, STOC’12
- Henzinger et al, FOCS’13
- Abraham et al, SODA’17
- Chechik, FOCS’18
- Probst Wulff-Nilsen, SODA’20
- Probst Wulff-Nilsen, SODA’20

---

More problems...
Dynamic Alg. **Inside** Static Alg.

- **Shortest path**
  - Priority Queue (e.g. Fibonacci Heap)

- **Kruskal’s MST**
  - Union-find

- **Linear Program**
  - Dynamic Linear System Solver
  - [Karmakar’84] [Vaidya’89]

- **Max flow**
  - Link-cut Tree
  - [Sleator Tarjan’82]

- **Traveling Salesman**
  - [Chekuri Quanrud FOCS’17]
  - Dyn. “Global Min Cut”

**Many more…**
Non-adaptive users: All updates are **fixed from the beginning**.

Adaptive users: Updates from users can **depend on previous answers**.

Example:

- **User**
  - Path $P$
  - Increase traffic on $P$
  - From A to B
  - Path $P'$

Usually **cannot** be used as subroutines inside static algo.
Dynamic Spanning Forest: Definition and Progress
Definition: Spanning Tree/Forest

**Spanning tree:** a smallest sub-network that connects all nodes together

**Spanning forest:** set of spanning trees on each connected component
Maintaining a *spanning forest* under changes

*Will say spanning tree and spanning forest interchangeably*
Example: Dynamic Spanning Forest

<table>
<thead>
<tr>
<th>Input:</th>
<th>Update in G</th>
</tr>
</thead>
<tbody>
<tr>
<td>Picture</td>
<td><img src="image" alt="Diagram" /></td>
</tr>
<tr>
<td>Output:</td>
<td>Change in F</td>
</tr>
</tbody>
</table>
Example: Dynamic Spanning Forest

<table>
<thead>
<tr>
<th>Input: Update in G</th>
<th>Delete(1,3)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Picture</strong></td>
<td></td>
</tr>
<tr>
<td><img src="image1" alt="Graph Description" /></td>
<td><img src="image2" alt="Graph Description" /></td>
</tr>
</tbody>
</table>

Output: Change in F
Example: Dynamic Spanning Forest

<table>
<thead>
<tr>
<th>Input: Update in G</th>
<th>Delete(1,3)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Picture</strong></td>
<td></td>
</tr>
<tr>
<td><img src="example.png" alt="Graph" /></td>
<td><img src="example.png" alt="Graph" /></td>
</tr>
<tr>
<td><strong>Output:</strong> Change in F</td>
<td>(1,3) removed (1,2) added</td>
</tr>
</tbody>
</table>

Input: Update in G
Delete(1,3)

Output: Change in F
(1,3) removed
(1,2) added
Example: Dynamic Spanning Forest

<table>
<thead>
<tr>
<th>Input: Update in G</th>
<th>Delete(1,3)</th>
<th>Insert(2,3)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Picture</td>
<td><img src="image1.png" alt="Graph" /></td>
<td><img src="image2.png" alt="Graph" /></td>
</tr>
<tr>
<td>Output: Change in F</td>
<td>(1,3) removed (1,2) added</td>
<td></td>
</tr>
</tbody>
</table>
Example: Dynamic Spanning Forest

<table>
<thead>
<tr>
<th>Input: Update in G</th>
<th>Delete(1,3)</th>
<th>Insert(2,3)</th>
<th>Delete(2,4)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Picture</td>
<td><img src="image" alt="Graph" /></td>
<td><img src="image" alt="Graph" /></td>
<td><img src="image" alt="Graph" /></td>
</tr>
<tr>
<td>Output: Change in F</td>
<td>(1,3) removed (1,2) added</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

*Diagram:*
- **Input:**
  - Update in G: Delete(1,3), Insert(2,3), Delete(2,4)
- **Output:**
  - Change in F: (1,3) removed, (1,2) added
**Example: Dynamic Spanning Forest**

<table>
<thead>
<tr>
<th>Input:</th>
<th>Delete(1,3)</th>
<th>Insert(2,3)</th>
<th>Delete(2,4)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Update in G</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Picture</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Output:</th>
<th>(1,3) removed</th>
<th>(2,4) removed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Change in F</td>
<td>(1,2) added</td>
<td>(2,3) added</td>
</tr>
</tbody>
</table>

---

![Graph Diagrams]

- 1
  - 2
  - 3
  - 4
  - 5

- 1
  - 2
  - 3
  - 4
  - 5

- 1
  - 2
  - 3
  - 4
  - 5

- 1
  - 2
  - 3
  - 4
  - 5
Goal: minimize update time

**Worst-case** time to output changes of $F$ for each update
Why this problem can be hard?
Why this problem can be hard?

Interesting when: delete a tree-edge

Want: Find a crossing edge

Question: Must scan all clique-edges? Scan the whole graph?
<table>
<thead>
<tr>
<th>Reference</th>
<th>Update time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Naïve</td>
<td>$m$</td>
</tr>
<tr>
<td>Frederickson [STOC’83]</td>
<td>$m^{1/2}$</td>
</tr>
<tr>
<td>EGIN [FOCS’92]</td>
<td>$n^{1/2}$</td>
</tr>
</tbody>
</table>

Progress

$n = \#$ of nodes, $m = \#$ of edges

Hide log factors from now
Progress

$n = \# \text{ of nodes}, m=\# \text{ of edges}$

<table>
<thead>
<tr>
<th>Reference</th>
<th>Update time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Naïve</td>
<td>$m$</td>
</tr>
<tr>
<td>Frederickson [STOC’83]</td>
<td>$m^{1/2}$</td>
</tr>
<tr>
<td>EGIN [FOCS’92]</td>
<td>$n^{1/2}$</td>
</tr>
</tbody>
</table>

20-year gap:
A lot of successes in closely related settings
(amortized update time)

Important development in amortized update time

Henzinger King [STOC’95]
Holm Lichtenberg Thorup [STOC’98]
Thorup [STOC’00]
Patrascu Demaine [STOC’04]
Wulff-Nilsen [SODA’13]
HHKP [SODA’17]
**Progress**

\[ n = \# \text{ of nodes}, \; m = \# \text{ of edges} \]

<table>
<thead>
<tr>
<th>Reference</th>
<th>Update time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Naïve</td>
<td>( m )</td>
</tr>
<tr>
<td>Frederickson [STOC’83]</td>
<td>( m^{1/2} )</td>
</tr>
<tr>
<td>EGIN [FOCS’92]</td>
<td>( n^{1/2} )</td>
</tr>
</tbody>
</table>

20-year gap:
A lot of successes in closely related settings (amortized update time)

Assume user is not adaptive

| Kapron King Mountjoy [SODA’13] | polylog \( n \) |

Important development in amortized update time:

- Henzinger King [STOC’95]
- Holm Lichtenberg Thorup [STOC’98]
- Thorup [STOC’00]
- Patrascu Demaine [STOC’04]
- Wulff-Nilsen [SODA’13]
- HHKP [SODA’17]
Non-adaptive users: All updates are **fixed from the beginning**.

Adaptive users: Updates from users can **depend on previous answers**.

Example:

Usually **cannot** be used as subroutines inside static algo.
**Progress**

\[ n = \# \text{ of nodes, } m = \# \text{ of edges} \]

<table>
<thead>
<tr>
<th>Reference</th>
<th>Update time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Naïve</td>
<td>( m )</td>
</tr>
<tr>
<td>Frederickson [STOC’83]</td>
<td>( m^{1/2} )</td>
</tr>
<tr>
<td>EGIN [FOCS’92]</td>
<td>( n^{1/2} )</td>
</tr>
</tbody>
</table>

**Important development in amortized update time**

- Henzinger King [STOC’95]
- Holm Lichtenberg Thorup [STOC’98]
- Thorup [STOC’00]
- Patrascu Demaine [STOC’04]
- Wulff-Nilsen [SODA’13]
- HHKP [SODA’17]

**20-year gap:**

A lot of successes in closely related settings (amortized update time)

**Assume user is not adaptive**

- Kapron King Mountjoy [SODA’13] \( \text{polylog } n \)
### Progress

\[ n = \# \text{ of nodes}, \, m=\# \text{ of edges} \]

<table>
<thead>
<tr>
<th>Reference</th>
<th>Update time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Naïve</td>
<td>( m )</td>
</tr>
<tr>
<td>Frederickson [STOC’83]</td>
<td>( m^{1/2} )</td>
</tr>
<tr>
<td>EGIN [FOCS’92]</td>
<td>( n^{1/2} )</td>
</tr>
</tbody>
</table>

**20-year gap:**

A lot of successes in closely related settings
(amortized update time)

**Assume user is not adaptive**

<table>
<thead>
<tr>
<th>Reference</th>
<th>Update time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Kapron King Mountjoy [SODA’13]</td>
<td>polylog ( n )</td>
</tr>
<tr>
<td>KKPT [ESA’16]</td>
<td>( n^{1/2} \cdot \frac{\log \log n}{(\log n)^{1/2}} )</td>
</tr>
<tr>
<td>Reference</td>
<td>Update time</td>
</tr>
<tr>
<td>-----------------------------------</td>
<td>----------------------</td>
</tr>
<tr>
<td>Naïve</td>
<td>$m$</td>
</tr>
<tr>
<td>Frederickson [STOC’83]</td>
<td>$m^{1/2}$</td>
</tr>
<tr>
<td>EGIN [FOCS’92]</td>
<td>$n^{1/2}$</td>
</tr>
</tbody>
</table>

**20-year gap:**
A lot of successes in closely related settings
(amortized update time)

---

**Assume user is not adaptive**

<table>
<thead>
<tr>
<th>Reference</th>
<th>Update time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Kapron King Mountjoy [SODA’13]</td>
<td>polylog $n$</td>
</tr>
<tr>
<td>KKPT [ESA’16]</td>
<td>$n^{1/2} \cdot \frac{\log \log n}{(\log n)^{1/2}}$</td>
</tr>
<tr>
<td>Wulff-Nilsen [STOC’17]</td>
<td>$n^{0.499}$</td>
</tr>
<tr>
<td>Nanongkai S [STOC’17]</td>
<td>$n^{0.401}$</td>
</tr>
</tbody>
</table>

$n = \# \text{ of nodes}, m = \# \text{ of edges}$

**Independent works**
Progress

\( n = \# \text{ of nodes}, m=\# \text{ of edges} \)

<table>
<thead>
<tr>
<th>Reference</th>
<th>Update time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Naïve</td>
<td>( m )</td>
</tr>
<tr>
<td>Frederickson [STOC’83]</td>
<td>( m^{1/2} )</td>
</tr>
<tr>
<td>EGIN [FOCS’92]</td>
<td>( n^{1/2} )</td>
</tr>
</tbody>
</table>

20-year gap:
A lot of successes in closely related settings
(amortized update time)

Assume user is not adaptive

<table>
<thead>
<tr>
<th>Reference</th>
<th>Update time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Kapron King Mountjoy [SODA’13]</td>
<td>polylog ( n )</td>
</tr>
<tr>
<td>KKPT [ESA’16]</td>
<td>( n^{1/2} \cdot \frac{\log \log n}{(\log n)^{1/2}} )</td>
</tr>
<tr>
<td>Wulff-Nilsen [STOC’17]</td>
<td>( n^{0.499} )</td>
</tr>
<tr>
<td>Nanongkai S [STOC’17]</td>
<td>( n^{0.401} )</td>
</tr>
<tr>
<td>NSW [FOCS’17]</td>
<td>( n^{\circ(1)} )</td>
</tr>
</tbody>
</table>

Progress

\( n = \# \text{ of nodes}, m=\# \text{ of edges} \)

Will explain how to use
Expander decomposition via (simplification of) this work
Part 1.2
Dynamic Spanning Forest:
How to use Expander Decomposition
Recall: Why this problem can be hard?

Let’s solve the problem on *graphs* that this situation cannot happen...
Expanders

Intuition
- Well-connected
- Hard to separate into two equal sides

Random Graphs (Erdős-Rényi)

Power-law Graphs (preferential attachment)

Hypercubes

$\mathbb{F}_p$-cycles with inverse chords

Gkantsidis, Mihail, Saberi SIGMETRICS'03
Mihail, Papadimitriou, Saberi FOCS'03
Definition: Expander

\( G = (V, E) \) is an **expander** if

\[
\forall S \subseteq V \quad \frac{|E(S, \overline{S})|}{\min\{\text{vol}(S), \text{vol}(\overline{S})\}} \geq \frac{1}{\text{polylog}(n)}
\]

**Sum of degree:**

\( \text{vol}(S) = \sum_{u \in S} \deg_G u \)

In general, a \( \phi \)-expander:

\[
\frac{|E(S, \overline{S})|}{\min\{\text{vol}(S), \text{vol}(\overline{S})\}} \geq \phi
\]
Expander Paradigm

1. Solve it on expanders.

2. Combine the solutions.
Expander Paradigm

1. Solve it on expanders.
Warm-up: One update to Expander

Suppose that $G$ is an expander, and there is one update. 

Goal: maintain a spanning tree $T$ of $G$. 
Warm-up: One update to Expander

Suppose that $G$ is an expander, and there is one update. 
**Goal:** maintain a **spanning tree $T$ of $G$**

**Interesting only when:** delete a tree-edge

**Want:** edge crossing $S$ to reconnect

**Alg:** sample an **edge with an endpoint in $S$** (can do fast)

**By expansion:** get edge crossing $S$ w.p. $\frac{1}{\text{polylog}(n)}$

**Repeat:** $\tilde{O}(1)$ times. Done w.h.p.
What if there are more updates?

After many edge deletions, \textit{not expander} anymore!

Let's "repair" the expander
Expander Paradigm

1. Solve it on **expanders**.

   - Problem specific: 
     - e.g. Random Sampling

   - General tool: 
     - **Expander Pruning**

2. **Combine** the solutions.


**Expander Pruning** [NSW’17]

$G_0$: expander

$G_1 = G_0 - e_1$

$G_2 = G_1 - e_2$

\[ G_i = G_{i-1} - e_i \]

$G_k = G_{k-1} - e_k$

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

**Guarantee:**

1. Time to update $P_{i-1}$ to $P_i$ is $n^{o(1)}$
2. So $vol(P_i) = i \cdot n^{o(1)}$
3. $G_i[V - P_i]$ is a $\frac{1}{n^{o(1)}}$-expander

*We show something slightly weaker*
Expander Pruning [NSW’17]

$G_0$: expander

$G_1 = G_0 - e_1$

$G_2 = G_1 - e_2$

$G_i = G_{i-1} - e_i$

$G_k = G_{k-1} - e_k$

$G_i[V - P_i]$ is a $(\frac{1}{n^{o(1)}})$-expander*

Expanders can be quickly “repaired” under edge updates.
Expander Paradigm

1. Solve it on expanders.

Problem specific:
- e.g. Random Sampling

General tool:
- Expander Pruning
Expander Paradigm

1. Solve it on **expanders**.

- **Problem specific:**
  - *e.g.* Random Sampling

- **General tool:**
  - Expander Pruning

In this talk, will only show how to solve a **relaxed problem** (contains all conceptual ideas)
Relaxed Problem: Dynamic Spanning Subgraphs

1. Maintain Any Spanning Subgraph with $\tilde{O}(n)$ edges (Easier than Spanning Forest.)

2. There are only $n^{1-o(1)}$ updates (can assume w.l.o.g. by standard techniques.)
What if there are more updates?

Suppose that $G$ is an expander, but there are many updates.

Expander Pruning:
1. Time to update $P_i$ is $n^{o(1)}$
2. So $\text{vol}(P_i) = i \cdot n^{o(1)}$
3. $G_i[V - P_i]$ is a $\frac{1}{n^{o(1)}}$-expander
What if there are more updates?

Suppose that $G$ is an expander, but there are many updates.

**Algo:** maintain spanning tree $T$ of $G[V - P]$ union with $E(P, V)$

**Update time:** $n^{o(1)}$

- Updating $E(P, V)$: $n^{o(1)}$ by Expander Pruning.
- Updating $T$: $n^{o(1)}$ by Random Sampling
  - $G[V - P]$ is $\frac{1}{n^{o(1)}}$-expander at any time.

**Expander Pruning:**
1. Time to update $P_i$ is $n^{o(1)}$
2. So $\text{vol}(P_i) = i \cdot n^{o(1)}$
3. $G_i[V - P_i]$ is a $\frac{1}{n^{o(1)}}$-expander

Work with adaptive users!
What if there are more updates?

Suppose that $G$ is an expander, but there are many updates.
**Algo:** maintain spanning tree $T$ of $G[V - P]$ union with $E(P, V)$

**Correctness:**
- $T \cup E(P, V)$ spans $G$
- $|T \cup E(P, V)| = O(n)$
  - $|T| \leq n$
  - $|E(P, V)| = \text{vol}(P) = O(n)$
  - Recall: #updates is $n^{1-o(1)}$

**Expander Pruning:**
1. Time to update $P_i$ is $n^{o(1)}$
2. So $\text{vol}(P_i) = i \cdot n^{o(1)}$
3. $G_i[V - P_i]$ is a $\frac{1}{n^{o(1)}}$-expander
Expander Paradigm

1. Solve it on **expanders**.

- **General tool:** Expander Pruning
- **Problem specific:** e.g. Random Sampling

How to work with general graphs?
Expander Paradigm

1. Solve it on **expanders**.
   - Problem specific:
     - e.g. Random Sampling
   - General tool:
     - Expander Pruning

2. **Combine** the solutions.
   - General tool:
     - Expander Decomposition
Expander Decomposition

Input: $G = (V, E)$

Output: A partition $(V_1, \ldots, V_k)$ of $V$

$G[V_i]$ is expander $\leq m/2$ inter-cluster edges

“Graph = Disjoint Expanders + Few Edges”

[S Wang SODA’19]: $\tilde{O}(m)$-time w.h.p.
Repeated Expander Decomposition

Expander decomposition
Repeated Expander Decomposition

\[ \leq \frac{m}{2} \text{ inter-cluster edges} \]
Repeated Expander Decomposition

\[ \leq \frac{m}{2} \text{ inter-cluster edges} \]
Repeated Expander Decomposition

$\leq \frac{m}{4}$ inter-cluster edges
Repeated Expander Decomposition

\[ \leq \frac{m}{4} \text{ inter-cluster edges} \]
Repeated Expander Decomposition

Graphs $G_1$, $G_2$, $G_3$, ..., $G_O(\log n)$
Repeated Expander Decomposition

Input: $G = (V, E)$

Output: $(G_1, \ldots, G_{O(\log n)})$ such that

- $G_i$ = disjoint union of expanders
- $E = E(G_1) \cup \ldots \cup E(G_{O(\log n)})$

Time: $\tilde{O}(m)$
Dynamic Spanning Subgraph: General Graphs

1. **Preprocess**: Compute repeated expander decomposition \((G_1, \ldots, G_{O(\log n)})\)
2. **Algo**: Each expander \(G'\), maintain spanning subgraph \(H'\) of \(G'\)
   - \(H'\) has \(O(|V(G')|)\) edges
   - Update time \(n^{o(1)}\) (if the update is on \(G'\))
Dynamic Spanning Subgraph: General Graphs

1. **Preprocess**: Compute repeated expander decomposition \((G_1, \ldots, G_{O(\log n)})\)

2. **Algo**: Each expander \(G'\), maintain spanning subgraph \(H'\) of \(G'\)
   - \(H'\) has \(O(|V(G')|)\) edges
   - Update time \(n^{o(1)}\) (if the update is on \(G'\))

3. **Claim**: Union of all \(H'\) is a spanning subgraph of \(G\) with \(O(n \log n)\) edges.
Dynamic Spanning Subgraph: General Graphs

1. **Preprocess**: Compute repeated expander decomposition \((G_1, \ldots, G_{O(\log n)})\)

2. **Algo**: Each expander \(G'\), maintain spanning subgraph \(H'\) of \(G'\)
   - \(H'\) has \(O(|V(G')|)\) edges
   - Update time \(n^{o(1)}\) (if the update is on \(G'\) )

3. **Claim**: Union of all \(H'\) is a spanning subgraph of \(G\) with \(O(n \log n)\) edges.
Dynamic Spanning Subgraphs

Conclusion:
Given $G$ undergoing edge updates,
• maintain spanning subgraph
• with $O(n \log n)$ edges
• in $n^{o(1)}$ update time
Part 2
Centralized Algorithms
1. Solve it on **expanders**.

- **Problem specific:** e.g. Random Sampling

- **General tool:** Expander Pruning

2. **Combine** the solutions.

- **General tool:** Expander Decomposition
Definition: Spanner

**Informal:** Subgraph that preserves all distances.
Definition: Spanner

Let $G = (V, E)$. 

$H = (V, E')$ is a \textit{k-spanner} of $G$ if

1. $E' \subset E$
2. $\forall (u, v) \in E$, $\text{dist}_{H}(u, v) \leq k$
Spanners of Expanders

$G$: expander

$T$: a shortest path tree in $G$ (rooted at an arbitrary node $r$).

**Observe**: $T$ is a polylog$(n)$-spanner of $G$

**Proof**: $\forall (u, v) \in E$, $\text{dist}_T(u, v) \leq \text{dist}_T(u, r) + \text{dist}_T(r, v)$

= polylog$(n)$

**Fact**: Diameter of expanders is polylog$(n)$. 

$G$: $\phi$-expander
Spanners of General Graphs

Spanner(G):

1. Compute repeated expander decomposition: \((G_1, \ldots, G_{O(\log n)})\)
2. \(H_i = \text{Shortest path tree on each expander of } G_i\)
Spanners of General Graphs

Spanner(G):

1. Compute repeated expander decomposition: \((G_1, \ldots, G_{O(\log n)})\)
2. \(H_i = \text{Shortest path tree on each expander of } G_i\)
3. Return \(H = \cup_i H_i\)

\[\forall (u, v) \in E, \quad \text{dist}_T(u, v) \leq \text{polylog}(n)\]

Total time: \(\tilde{O}(m)\)

\[|E(H_i)| \leq n \text{ (forest), so } |E(H)| = O(n \log n).\]
Spanners of General Graphs

Conclusion:

Given $G$,
• a polylog($n$)-spanner
• with $O(n \log n)$ edges
• in $\tilde{O}(m)$ time
Expander Paradigm

1. Solve it on **expanders**.
   - Problem specific: **Shortest path tree**
   - Problem specific: **Random sampling**

2. **Combine** the solutions.
   - General tool: **Expander Decomposition**

**More applications:**
- **Cut sparsifiers**: preserve cut sizes
- **Spectral sparsifiers**: preserve eigenvalues
Part 3
Distributed Algorithms
Definition: CONGEST model
Definition: CONGEST model

- **Local knowledge:** A node knows only its neighbors.
- **Local communication:** A node can send messages to only its neighbors 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.
Expander Paradigm (Distributed)

1. Solve it on **expanders**.

   - Problem specific: e.g. Random Sampling
   - General tool: Expander Routing

2. **Combine** the solutions.

   - General tool: Expander Decomposition
Expander Routing (Informal)
[Ghaffari Kuhn Su PODC'17] [Ghaffari Li DISC'18]

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

**Expanders** allow global communication with small overhead.

Local communication
In any graph, can exchange with only neighbors in 1 round.
Expander Routing

[Ghaffari Kuhn Su PODC'17] [Ghaffari Li DISC'18]

**Input:** underlying graph $G = (V, E)$ and demand graph $D = (V, E')$

- $G$: expander
- $\deg_D(u) \leq \deg_G(u) \forall u \in V$

**Output:**

- for all $(u, v) \in E'$ simultaneously,
- $u$ and $v$ can exchange a message in $n^{o(1)}$ rounds (in $G$)

*Expanders allow global communication with small overhead*
Expander Paradigm (Distributed)

1. Solve it on expanders.
   - General tool: Expander Routing
   - Problem specific: e.g. Random Sampling

2. Combine the solutions.
   - General tool: Expander Decomposition
   - Round complexity:
     - $n^{1-\epsilon}$ [Chang Pettie Zhang SODA’19] (with caveat)
     - $n^\epsilon$ [Chang S PODC’19]
     - $\text{polylog}(n)$ [Chang S]

Can import ideas from algorithms in CONGESTED-CLIQUE model
Part 4
Conclusion:
Survey and Open Problems
Centralized Setting
Expander Paradigm (Centralized)

1. Solve it on **expanders**.
   - Problem specific: e.g. Random Sampling

2. **Combine** the solutions.
   - General tool: Expander Decomposition
Expander Decomposition

Input: \( G = (V, E) \)

Output: A partition \((V_1, \ldots, V_k)\) of \( V \)

\[ G[V_i] \text{ is expander} \quad \leq m/2 \text{ inter-cluster edges} \]

"Graph = Disjoint Expanders + Few Edges"

[S Wang SODA’19]: \( \tilde{O}(m) \)-time w.h.p.
## Fast Centralized Algorithms

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Time (Randomized)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Laplacian system solvers</td>
<td>$\tilde{O}(m)$</td>
</tr>
<tr>
<td>[Spielman Teng STOC’04]</td>
<td></td>
</tr>
<tr>
<td>Spectral sparsifiers</td>
<td>$\tilde{O}(m)$</td>
</tr>
<tr>
<td>[Spielman Teng STOC’04]</td>
<td></td>
</tr>
<tr>
<td>Approx. max flow</td>
<td>$\tilde{O}(m)$</td>
</tr>
<tr>
<td>[Kelner Lee Orecchia Sidford SODA’14]</td>
<td></td>
</tr>
<tr>
<td>Approx. vertex max flow</td>
<td>$\tilde{O}(n^2)$</td>
</tr>
<tr>
<td>[Chuzhoy Khanna STOC’19]</td>
<td></td>
</tr>
<tr>
<td>Bipartite Matching, Shortest Path, Max flow</td>
<td>$\tilde{O}(m^{10/7})$</td>
</tr>
<tr>
<td>[Cohen Madry Sankowski Vladu SODA’17]</td>
<td></td>
</tr>
</tbody>
</table>

**New!** [Chuzhoy Gao Li Nanongkai Peng S]:
Expander decomposition in $m^{1+o(1)}$ deterministic time

Expander Paradigm is the key to all these results
Fast Centralized Algorithms

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Time (Randomized)</th>
<th>Time (Deterministic)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Laplacian system solvers</td>
<td>$\tilde{O}(m)$</td>
<td>$m^{1+o(1)}$</td>
</tr>
<tr>
<td>[Spielman Teng STOC’04]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Spectral sparsifiers</td>
<td>$\tilde{O}(m)$</td>
<td>$m^{1+o(1)}$</td>
</tr>
<tr>
<td>[Spielman Teng STOC’04]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Approx. max flow</td>
<td>$\tilde{O}(m)$</td>
<td>$m^{1+o(1)}$</td>
</tr>
<tr>
<td>[Kelner Lee Orecchia Sidford SODA’14]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Approx. vertex max flow</td>
<td>$\tilde{O}(n^2)$</td>
<td>$n^{2+o(1)}$</td>
</tr>
<tr>
<td>[Chuzhoy Khanna STOC’19]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bipartite Matching, Shortest Path, Max flow</td>
<td>$\tilde{O}(m^{10/7})$</td>
<td>$m^{10/7+o(1)}$</td>
</tr>
<tr>
<td>[Cohen Madry Sankowski Vladu SODA’17]</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Open: Expander decomposition in $\tilde{O}(m)$ **deterministic** time (would remove all $m^{o(1)}$ below)
Dynamic Setting
Non-adaptive users:
All updates are fixed from the beginning.

Adaptive users:
Updates from users can depend on previous answers

Example:

From A to B
Path $P$
Increase traffic on $P$
From A to B
Path $P'$

Usually cannot be used as subroutines inside static algo.
**Frontier of Dynamic Graph Algorithms**

---

**We DON’T know how to serve adaptive users!**

<table>
<thead>
<tr>
<th>Problems</th>
<th>Non-adaptive users</th>
<th>Adaptive users</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spanning Forests</td>
<td>$\text{polylog } n$</td>
<td>$\sqrt{n}$</td>
</tr>
<tr>
<td>(worst case)</td>
<td>[Kapron King Mountjoy SODA’13]</td>
<td>[EGIN FOCS’92]</td>
</tr>
</tbody>
</table>
# Frontier of Dynamic Graph Algorithms

We DON’T know how to serve adaptive users!

<table>
<thead>
<tr>
<th>Problems</th>
<th>Non-adaptive users</th>
<th>Adaptive users</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Spanning Forests</strong></td>
<td>polylog $n$</td>
<td>$\sqrt{n}$</td>
</tr>
<tr>
<td>(worst case)</td>
<td>[Kapron King Mountjoy SODA’13]</td>
<td>[EGIN FOCS’92]</td>
</tr>
<tr>
<td><strong>Spanners</strong></td>
<td>polylog $n$</td>
<td>$m$</td>
</tr>
<tr>
<td>(amortized)</td>
<td>[BKS ESA06, SODA’08]</td>
<td>[trivial]</td>
</tr>
<tr>
<td><strong>Single Source Shortest Paths</strong></td>
<td>$m^{1+o(1)}$</td>
<td>$mn$</td>
</tr>
<tr>
<td>(decremental approximate amortized)</td>
<td>[HKKN FOCS’14]</td>
<td>[Even Shiloah’81]</td>
</tr>
<tr>
<td><strong>Single Source Reachability</strong></td>
<td>$m$</td>
<td>$mn$</td>
</tr>
<tr>
<td>(decremental amortized)</td>
<td>[BPW STOC’19]</td>
<td>[Even Shiloah’81]</td>
</tr>
<tr>
<td><strong>Cut Sparsifiers</strong></td>
<td>polylog $n$</td>
<td>$m$</td>
</tr>
<tr>
<td>(worst-case)</td>
<td>[ADKKP FOCS’16]</td>
<td>[trivial]</td>
</tr>
<tr>
<td><strong>Maximal Matching</strong></td>
<td>$O(1)$</td>
<td>$\sqrt{m}$</td>
</tr>
<tr>
<td></td>
<td>[Solomon FOCS’16]</td>
<td>[Neiman Solomon STOC’13]</td>
</tr>
</tbody>
</table>
### Frontier of Dynamic Graph Algorithms

Expander Paradigm can help in many cases!

<table>
<thead>
<tr>
<th>Problems</th>
<th>Non-adaptive users</th>
<th>Adaptive users</th>
<th>Adaptive users (by Expander Decomposition)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spanning Forests (worst case)</td>
<td>polylog $n$</td>
<td>$\sqrt{n}$</td>
<td>$n^{o(1)}$</td>
</tr>
<tr>
<td></td>
<td>[Kapron King Mountjoy SODA’13]</td>
<td>[EGIN FOCS’92]</td>
<td>[NSW FOCS’17]</td>
</tr>
<tr>
<td>Spanners (amortized)</td>
<td>polylog $n$</td>
<td>$m$</td>
<td>$n^{o(1)}$</td>
</tr>
<tr>
<td></td>
<td>[BKS ESA06, SODA’08]</td>
<td>[trivial]</td>
<td>[BNSSS FOCS’17]</td>
</tr>
<tr>
<td>Single Source Shortest Paths (decremental approximate amortized)</td>
<td>$m^{1+o(1)}$</td>
<td>$mn$</td>
<td>$n^{2+o(1)}$</td>
</tr>
<tr>
<td></td>
<td>[HKN FOCS’14]</td>
<td>[Even Shiloah’81]</td>
<td>[Bernstein Chechik STOC’16] [CS]</td>
</tr>
<tr>
<td>Single Source Reachability (decremental amortized)</td>
<td>$m$</td>
<td>$mn$</td>
<td>_</td>
</tr>
<tr>
<td></td>
<td>[BPW STOC’19]</td>
<td>[Even Shiloah’81]</td>
<td>_</td>
</tr>
<tr>
<td>Cut Sparsifiers (worst-case)</td>
<td>polylog $n$</td>
<td>$m$</td>
<td>_</td>
</tr>
<tr>
<td></td>
<td>[ADKKP FOCS’16]</td>
<td>[trivial]</td>
<td>_</td>
</tr>
<tr>
<td>Maximal Matching</td>
<td>$O(1)$</td>
<td>$\sqrt{m}$</td>
<td>_</td>
</tr>
<tr>
<td></td>
<td>[Solomon FOCS’16]</td>
<td>[Neiman Solomon STOC’13]</td>
<td>_</td>
</tr>
</tbody>
</table>

We saw this (simplified)
Expander Paradigm (Dynamic)

1. Solve it on **expanders**.
   - Problem specific: e.g. Random Sampling
   - General tool: Expander Pruning

2. Combine **the solutions**.
   - General tool: Expander Decomposition
**Expander Pruning** [NSW’17]

$G_0$: expander

$G_1 = G_0 - e_1$

$G_2 = G_1 - e_2$

$G_i = G_{i-1} - e_i$

$G_k = G_{k-1} - e_k$

$G_i[V - P_i]$ is a $\left(\frac{1}{n^{o(1)}}\right)$-expander

**Expanders** can be quickly “repaired” under edge updates.
Expander Pruning [NSW’17]

\[ G_0: \text{expander} \]
\[ G_1 = G_0 - e_1 \]
\[ G_2 = G_1 - e_2 \]
\[ G_i = G_{i-1} - e_i \]
\[ G_k = G_{k-1} - e_k \]

Guarantee:
1. Time to update \( P_{i-1} \) to \( P_i \) is \( n^{o(1)} \)
2. So \( vol(P_i) = i \cdot n^{o(1)} \)
3. \( G_i[V - P_i] \) is a \( \frac{1}{n^{o(1)}} \)-expander

Open:
Improve \( n^{o(1)} \) to polylog\( (n) \)

imply polylog\( (n) \) worst-case update time for many
problems (e.g. spanning subgraphs, spectral
sparsifiers)
Distributed Setting
Expander Paradigm (Distributed)

1. Solve it on **expanders**.
   - **Problem specific:** e.g. Random Sampling
   - **General tool:** Expander Routing

2. **Combine** the solutions.
   - **General tool:** Expander Decomposition
Expander Routing

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

Expanders allow global communication with small overhead.

Open:

Improve $n^{o(1)}$ to polylog($n$) (Many applications even in centralized setting (chat offline))
## Distributed CONGEST algorithm

<table>
<thead>
<tr>
<th></th>
<th>Upper bound</th>
<th>Lower bound</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Triangle (3-clique) listing</strong></td>
<td>$\tilde{O}(n^{1/3})$</td>
<td>$\Omega(n^{1/3})$</td>
</tr>
<tr>
<td></td>
<td>[Chang, Pettie, Zhang SODA’18]</td>
<td>[Izumi, LeGall PODC’17]</td>
</tr>
<tr>
<td></td>
<td>[Chang SPODC’19]</td>
<td></td>
</tr>
<tr>
<td><strong>4-clique listing</strong></td>
<td>$\tilde{O}(n^{5/6})$</td>
<td>$\tilde{O}(n^{1/2})$</td>
</tr>
<tr>
<td></td>
<td>[Eden, Fiat, Fischer, Kuhn, Oshman DISC’19]</td>
<td>[Fischer, Gonen, Kuhn, Oshman SPAA’18]</td>
</tr>
<tr>
<td><strong>5-clique listing</strong></td>
<td>$\tilde{O}(n^{21/22})$</td>
<td>$\tilde{O}(n^{3/5})$</td>
</tr>
<tr>
<td></td>
<td>[Eden et al. DISC’19]</td>
<td>[Fischer et al. SPAA’18]</td>
</tr>
<tr>
<td><strong>$k$-vertex subgraph detection</strong></td>
<td>$n^{2-\Omega(1/k)}$</td>
<td>$n^{2-o(1/k)}$</td>
</tr>
<tr>
<td></td>
<td>[Eden et al. DISC’19]</td>
<td>[Fischer et al. SPAA’18]</td>
</tr>
</tbody>
</table>

**Expander Paradigm used in all upper bounds**
## Distributed CONGEST algorithm

<table>
<thead>
<tr>
<th></th>
<th>Upper bound</th>
<th>Lower bound</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Triangle (3-clique) listing</strong></td>
<td>$\tilde{O}(n^{1/3})$ [Chang Pettie Zhang SODA'18] [Chang S PODC'19]</td>
<td>$\tilde{\Omega}(n^{1/3})$ [Izumi LeGall PODC'17]</td>
</tr>
<tr>
<td><strong>4-clique listing</strong></td>
<td>$\tilde{O}(n^{5/6})$ [Eden Fiat Fischer Kuhn Oshman DISC'19]</td>
<td>$\tilde{\Omega}(n^{1/2})$ [Fischer Gonen Kuhn Oshman SPAA'18]</td>
</tr>
<tr>
<td><strong>5-clique listing</strong></td>
<td>$\tilde{O}(n^{21/22})$ [Eden et al. DISC'19]</td>
<td>$\tilde{\Omega}(n^{3/5})$ [Fischer et al. SPAA'18]</td>
</tr>
<tr>
<td><strong>$k$-vertex subgraph detection</strong></td>
<td>$n^{2-\Omega(1/k)}$ [Eden et al. DISC'19]</td>
<td>$n^{2-O(1/k)}$ [Fischer et al. SPAA'18]</td>
</tr>
<tr>
<td><strong>$k$-clique enumeration</strong></td>
<td>?</td>
<td>$\tilde{\Omega}(n^{1-2/k})$ [Fischer et al. SPAA'18]</td>
</tr>
</tbody>
</table>

**Open:** Application which is not subgraph detection/listing
## History: Distributed Expander Decomposition

<table>
<thead>
<tr>
<th>Reference</th>
<th>Rounds</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>[Chang Pettie Zhang SODA’19]</td>
<td>$n^{1-\delta}$</td>
<td>Output an extra part: a subgraph with arboricity $n^{\delta}$</td>
</tr>
<tr>
<td>[Chang S PODC’19]</td>
<td>$n^{\varepsilon}$</td>
<td></td>
</tr>
<tr>
<td>[Chang S in progress]</td>
<td>polylog$(n)$</td>
<td></td>
</tr>
<tr>
<td>[Chang S in progress]</td>
<td>$n^{\varepsilon}$</td>
<td>Deterministic</td>
</tr>
<tr>
<td><strong>Open:</strong></td>
<td>polylog$(n)$</td>
<td><strong>Deterministic</strong></td>
</tr>
</tbody>
</table>