Worst-case to Expander-Case Reductions: Derandomized and Generalized

Amir Abboud, *Nathan Wallheimer*

WEIZMANN INSTITUTE OF SCIENCE

European Research Council
Established by the European Commission

Expander Graphs

Expanders are one of the most important graph families in computer science. Therefore, a natural question to ask is:

Question 1

Are expanders worst-case instances of our graph problem?

Expander Graphs

Expanders are one of the most important graph families in computer science. Therefore, a natural question to ask is:

Question 1

Are expanders worst-case instances of our graph problem?

Consider fundamental graph problems: *shortest paths*, *cuts*, *matchings*, *subgraph detection*, ...

Expander Graphs

Expanders are one of the most important graph families in computer science. Therefore, a natural question to ask is:

Question 1

Are expanders worst-case instances of our graph problem?

Consider fundamental graph problems: *shortest paths*, *cuts*, *matchings*, *subgraph detection*, . . .

Answering this question can be done in one of the following methods:

Expander Graphs

Expanders are one of the most important graph families in computer science. Therefore, a natural question to ask is:

Question 1

Are expanders worst-case instances of our graph problem?

Consider fundamental graph problems: *shortest paths*, *cuts*, *matchings*, *subgraph detection*, . . .

Answering this question can be done in one of the following methods:

- Adapting *conditional lower bounds* to expanders. Modify a reduction from a hard problem to output expanders instead of arbitrary worst-case graphs. (See [Henzinger et. al, ESA'22])

Expander Graphs

Expanders are one of the most important graph families in computer science. Therefore, a natural question to ask is:

Question 1

Are expanders worst-case instances of our graph problem?

Consider fundamental graph problems: *shortest paths*, *cuts*, *matchings*, *subgraph detection*, ...

Answering this question can be done in one of the following methods:

- Adapting *conditional lower bounds* to expanders. Modify a reduction from a hard problem to output expanders instead of arbitrary worst-case graphs. (See [Henzinger et. al, ESA'22]) **Limitation: Impossible to do when the problem is not conditionally hard (e.g., Maximum Matching).**

Expander Graphs

Expanders are one of the most important graph families in computer science. Therefore, a natural question to ask is:

Question 1

Are expanders worst-case instances of our graph problem?

Consider fundamental graph problems: *shortest paths, cuts, matchings, subgraph detection, . . .*

Answering this question can be done in one of the following methods:

- Adapting *conditional lower bounds* to expanders. Modify a reduction from a hard problem to output expanders instead of arbitrary worst-case graphs. (See [Henzinger et. al, ESA'22]) **Limitation: Impossible to do when the problem is not conditionally hard (e.g., Maximum Matching).**
- Self-reductions to expanders. Given an instance of the problem, output one or more expander instances of the same problem.

Expander Graphs

Expanders are one of the most important graph families in computer science. Therefore, a natural question to ask is:

Question 1

Are expanders worst-case instances of our graph problem?

Consider fundamental graph problems: *shortest paths, cuts, matchings, subgraph detection, . . .*

Answering this question can be done in one of the following methods:

- Adapting *conditional lower bounds* to expanders. Modify a reduction from a hard problem to output expanders instead of arbitrary worst-case graphs. (See [Henzinger et. al, ESA'22]) **Limitation: Impossible to do when the problem is not conditionally hard (e.g., Maximum Matching).**
- Self-reductions to expanders. Given an instance of the problem, output one or more expander instances of the same problem. **Our focus in this talk.**

Expander Graphs Definition

There are different notions of expansion in graphs. Our focus is on the notion of edge expansion in graphs with arbitrary degree sequences.

Expander Graphs Definition

There are different notions of expansion in graphs. Our focus is on the notion of edge expansion in graphs with arbitrary degree sequences.

ϕ-expander

Let $G = (V, E)$ be a graph. For any cut $S \subseteq V$, let $vol(S) = \sum_{v \in S} \deg(v)$. Then, G is called ϕ-expander if for every $S \subseteq V$, we have:

1. $vol(S) > 0$.
2. $e(S, V \setminus S) \geq \phi \cdot \min(vol(S), vol(V \setminus S))$.

Expander Graphs Definition

There are different notions of expansion in graphs. Our focus is on the notion of edge expansion in graphs with arbitrary degree sequences.

ϕ-expander

Let $G = (V, E)$ be a graph. For any cut $S \subseteq V$, let $vol(S) = \sum_{v \in S} \deg(v)$. Then, G is called ϕ-expander if for every $S \subseteq V$, we have:

1. $vol(S) > 0$.
2. $e(S, V \setminus S) \geq \phi \cdot \min(vol(S), vol(V \setminus S))$.

ϕ is called the *conductance* of the graph.

Expander Graphs Definition

There are different notions of expansion in graphs. Our focus is on the notion of edge expansion in graphs with arbitrary degree sequences.

ϕ-expander

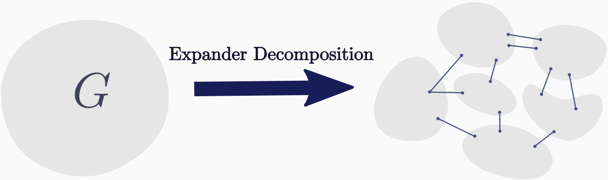
Let $G = (V, E)$ be a graph. For any cut $S \subseteq V$, let $vol(S) = \sum_{v \in S} \deg(v)$. Then, G is called ϕ-expander if for every $S \subseteq V$, we have:

1. $vol(S) > 0$.
2. $e(S, V \setminus S) \geq \phi \cdot \min(vol(S), vol(V \setminus S))$.

ϕ is called the *conductance* of the graph.

Whenever $\phi = \Omega(1)$, we simply say that G is an expander.

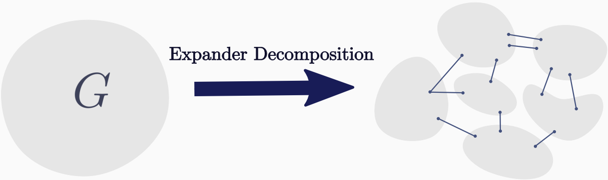
Expander Decompositions



Theorem

For any graph G and parameter $0 < \phi < 1$, the vertices of G can be partitioned into $V_1, V_2, \ldots, V_k$, such that:

Expander Decompositions

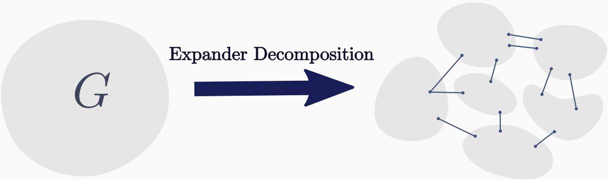


Theorem

For any graph G and parameter $0 < \phi < 1$, the vertices of G can be partitioned into $V_1, V_2, \ldots, V_k$, such that:

- For all $i \in [k]$, the induced subgraph $G[V_i]$ is a ϕ-expander.

Expander Decompositions

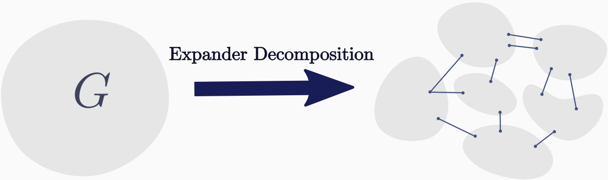


Theorem

For any graph G and parameter $0 < \phi < 1$, the vertices of G can be partitioned into $V_1, V_2, \dots, V_k$, such that:

- For all $i \in [k]$, the induced subgraph $G[V_i]$ is a ϕ-expander.
- The total number of edges between the subgraphs is $\tilde{O}(\phi m)$.

Expander Decompositions

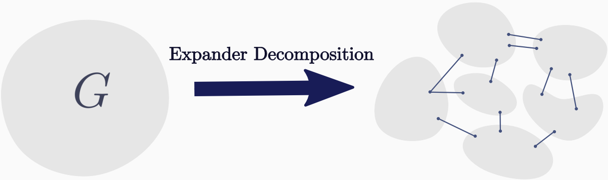


Theorem

For any graph G and parameter $0 < \phi < 1$, the vertices of G can be partitioned into $V_1, V_2, \ldots, V_k$, such that:

- For all $i \in [k]$, the induced subgraph $G[V_i]$ is a ϕ-expander.
- The total number of edges between the subgraphs is $\tilde{O}(\phi m)$.
- There is an algorithm computing this decomposition running in time $\tilde{O}(m/\phi)$.

Expander Decompositions



Theorem

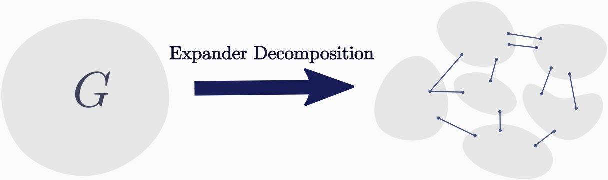
For any graph G and parameter $0 < \phi < 1$, the vertices of G can be partitioned into $V_1, V_2, \ldots, V_k$, such that:

- For all $i \in [k]$, the induced subgraph $G[V_i]$ is a ϕ-expander.
- The total number of edges between the subgraphs is $\tilde{O}(\phi m)$.
- There is an algorithm computing this decomposition running in time $\tilde{O}(m/\phi)$.

Note that the decomposition is non-trivial only for $\phi = \frac{1}{poly\log(n)}$ because of the second property.

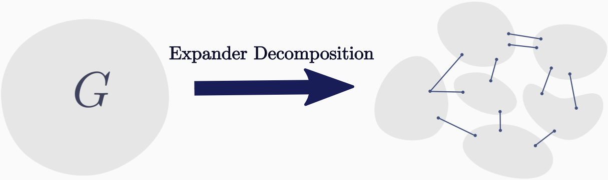
A Self-Reduction: The Expander Decomposition Method

One can self-reduce a problem to expanders using the expander decomposition method.



A Self-Reduction: The Expander Decomposition Method

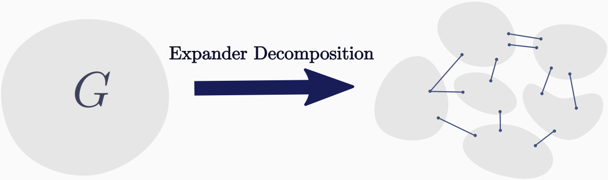
One can self-reduce a problem to expanders using the expander decomposition method.



The Expander Decomposition Method

A Self-Reduction: The Expander Decomposition Method

One can self-reduce a problem to expanders using the expander decomposition method.

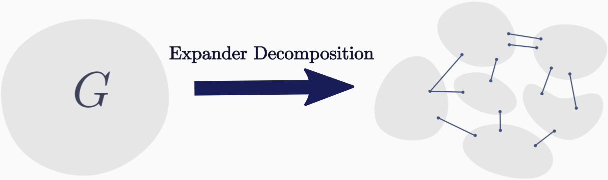


The Expander Decomposition Method

1. Decompose G with an appropriate choice of ϕ.

A Self-Reduction: The Expander Decomposition Method

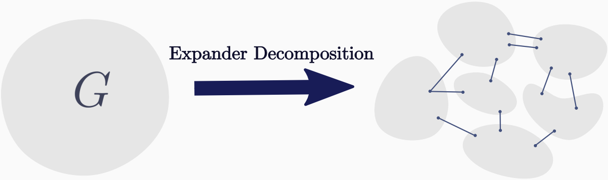
One can self-reduce a problem to expanders using the expander decomposition method.



The Expander Decomposition Method

1. Decompose G with an appropriate choice of ϕ.
2. Solve the problem on each expander.

One can self-reduce a problem to expanders using the expander decomposition method.

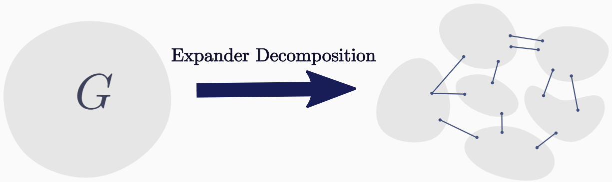


The Expander Decomposition Method

1. Decompose G with an appropriate choice of ϕ.
2. Solve the problem on each expander.
3. Combine the solutions using the property that there are few cross-edges.

A Self-Reduction: The Expander Decomposition Method

One can self-reduce a problem to expanders using the expander decomposition method.

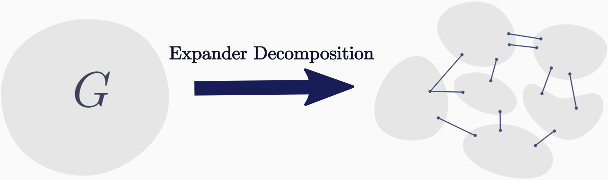


The expander decomposition method has been key to many breakthroughs in recent years:

- Max-flow in $m^{1+o(1)}$ time.
- *Dynamic* minimum spanning tree in $n^{o(1)}$ update time.
- *Derandomized* global min-cut in $m^{1+o(1)}$ time.

A Self-Reduction: The Expander Decomposition Method

One can self-reduce a problem to expanders using the expander decomposition method.



The expander decomposition method has been key to many breakthroughs in recent years:

- Max-flow in $m^{1+o(1)}$ time.
- *Dynamic* minimum spanning tree in $n^{o(1)}$ update time.
- *Derandomized* global min-cut in $m^{1+o(1)}$ time.

Question 2

Is expander decomposition the key to solving my problem?

A Direct Self-Reduction [Abboud and Wallheimer, 2023]

Surprisingly, we can answer both Question 1 and Question 2 at the same time!

A Direct Self-Reduction [Abboud and Wallheimer, 2023]

Surprisingly, we can answer both Question 1 and Question 2 at the same time!

$$G \longrightarrow \boxed{\textbf{Reduction}} \longrightarrow G_{exp}$$

A Direct Self-Reduction [Abboud and Wallheimer, 2023]

Surprisingly, we can answer both Question 1 and Question 2 at the same time!

A *direct worst-case to expander-case self-reduction* is a linear time algorithm that, given G, outputs G_{exp}, such that:

A Direct Self-Reduction [Abboud and Wallheimer, 2023]

Surprisingly, we can answer both Question 1 and Question 2 at the same time!

$$G \longrightarrow \boxed{\textbf{Reduction}} \longrightarrow G_{exp}$$

A *direct worst-case to expander-case self-reduction* is a linear time algorithm that, given G, outputs G_{exp}, such that:

- G_{exp} is an $\Omega(1)$-expander.

A Direct Self-Reduction [Abboud and Wallheimer, 2023]

Surprisingly, we can answer both Question 1 and Question 2 at the same time!

$$G \longrightarrow \boxed{\textbf{Reduction}} \longrightarrow G_{exp}$$

A *direct worst-case to expander-case self-reduction* is a linear time algorithm that, given G, outputs G_{exp}, such that:

- G_{exp} is an $\Omega(1)$-expander.
- The solution to G can be computed from the solution to G_{exp} in linear time.

A Direct Self-Reduction [Abboud and Wallheimer, 2023]

Surprisingly, we can answer both Question 1 and Question 2 at the same time!

$$G \longrightarrow \boxed{\textbf{Reduction}} \longrightarrow G_{exp}$$

A *direct worst-case to expander-case self-reduction* is a linear time algorithm that, given G, outputs G_{exp}, such that:

- G_{exp} is an $\Omega(1)$-expander.
- The solution to G can be computed from the solution to G_{exp} in linear time.
- The blowup in the size of G_{exp} is linear in the size of G.

A Direct Self-Reduction [Abboud and Wallheimer, 2023]

Surprisingly, we can answer both Question 1 and Question 2 at the same time!

$$G \longrightarrow \boxed{\textbf{Reduction}} \longrightarrow G_{exp}$$

Implications of Direct Reductions

If problem $\mathcal{A}$ admits a Direct Reduction, then:

A Direct Self-Reduction [Abboud and Wallheimer, 2023]

Surprisingly, we can answer both Question 1 and Question 2 at the same time!

$$G \longrightarrow \boxed{\textbf{Reduction}} \longrightarrow G_{exp}$$

Implications of Direct Reductions

If problem $\mathcal{A}$ admits a Direct Reduction, then:

- Hardness - $\Omega(1)$-expanders are worst-case instances of $\mathcal{A}$.

A Direct Self-Reduction [Abboud and Wallheimer, 2023]

Surprisingly, we can answer both Question 1 and Question 2 at the same time!

$$G \longrightarrow \boxed{\textbf{Reduction}} \longrightarrow G_{exp}$$

Implications of Direct Reductions

If problem $\mathcal{A}$ admits a Direct Reduction, then:

- Hardness - $\Omega(1)$-expanders are worst-case instances of $\mathcal{A}$.
- Conceptual message - The expander decomposition method is useless against $\mathcal{A}$.

A Direct Self-Reduction [Abboud and Wallheimer, 2023]

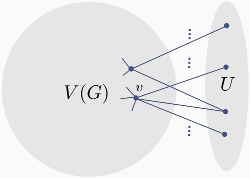
Surprisingly, we can answer both Question 1 and Question 2 at the same time!

$$G \longrightarrow \boxed{\textbf{Reduction}} \longrightarrow G_{exp}$$

[Abboud and Wallheimer, 2023]
The following problems admit direct reductions:

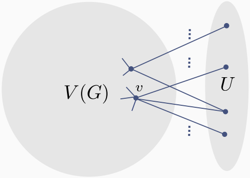
- k-Clique Detection
- Maximum Matching
- Vertex Cover
- Minimum Dominating Set

The Randomized Core Gadget of AW'23



Given a graph G, construct an expander graph G_{exp} as follows.

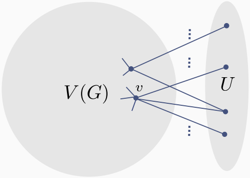
The Randomized Core Gadget of AW‘23



Given a graph G, construct an expander graph G_{exp} as follows.

1. Add an *expansion layer U*, that consists of $\Theta(n)$ vertices.

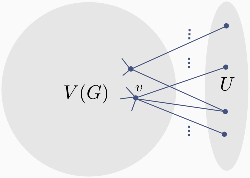
The Randomized Core Gadget of AW'23



Given a graph G, construct an expander graph G_{exp} as follows.

1. Add an *expansion layer* U, that consists of $\Theta(n)$ vertices.
2. For every $v \in V(G)$, add $\deg_G(v)$ random neighbors in U.

The Randomized Core Gadget of AW'23

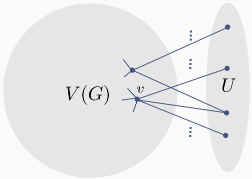


Given a graph G, construct an expander graph G_{exp} as follows.

1. Add an *expansion layer* U, that consists of $\Theta(n)$ vertices.
2. For every $v \in V(G)$, add $\deg_G(v)$ random neighbors in U.

Claim: G_{exp} is an $\Omega(1)$-expander w.h.p., and its size is $O(m+n)$.

The Randomized Core Gadget of AW‘23



Given a graph G, construct an expander graph G_{exp} as follows.

1. Add an *expansion layer* U, that consists of $\Theta(n)$ vertices.
2. For every $v \in V(G)$, add $\deg_G(v)$ random neighbors in U.

Claim: G_{exp} is an $\Omega(1)$-expander w.h.p., and its size is $O(m+n)$.
High level idea: Random walks mix well in this graph.

This Work: Overcoming the Limitations of AW‘23

The Direct-Reductions of AW‘23 had two limitations:

This Work: Overcoming the Limitations of AW'23

The Direct-Reductions of AW'23 had two limitations:

- The core gadget is randomized.

This Work: Overcoming the Limitations of AW‘23

The Direct-Reductions of AW‘23 had two limitations:

- The core gadget is randomized.
  1. The hardness results do not hold for deterministic algorithms.

Question 1

Are expanders worst-case instances also for deterministic algorithms?

This Work: Overcoming the Limitations of AW'23

The Direct-Reductions of AW'23 had two limitations:

- The core gadget is randomized.

  1. The hardness results do not hold for deterministic algorithms.

Question 1

Are expanders worst-case instances also for deterministic algorithms?

  2. Derandomization via deterministic expander decomposition?

This Work: Overcoming the Limitations of AW‘23

The Direct-Reductions of AW‘23 had two limitations:

- The core gadget is randomized.

1. The hardness results do not hold for deterministic algorithms.

Question 1

Are expanders worst-case instances also for deterministic algorithms?

2. Derandomization via deterministic expander decomposition?

Question 2

Is the expander decomposition method the key to derandomizing the best algorithms for our problem?

This Work: Overcoming the Limitations of AW'23

The Direct-Reductions of AW'23 had two limitations:

- The core gadget is randomized.

1. The hardness results do not hold for deterministic algorithms.

Question 1

Are expanders worst-case instances also for deterministic algorithms?

2. Derandomization via deterministic expander decomposition?

Question 2

Is the expander decomposition method the key to derandomizing the best algorithms for our problem?

This Work: Overcoming the Limitations of AW'23

The Direct-Reductions of AW'23 had two limitations:

- The core gadget is restricted to the sequential model, wheres expander decompositions are useful in other computational models, such as the fully-dynamic model.

This Work: Overcoming the Limitations of AW'23

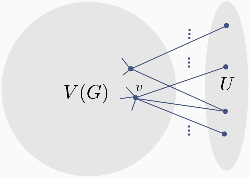
The Direct-Reductions of AW'23 had two limitations:

- The core gadget is restricted to the sequential model, wheres expander decompositions are useful in other computational models, such as the fully-dynamic model.

Question 3

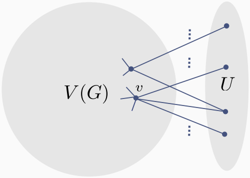
Are expanders worst-case instances in the fully-dynamic setting?

Attempt to Derandomize the Core Gadget



Replace the random V-to-U edges with a *fully explicit expander*.

Attempt to Derandomize the Core Gadget

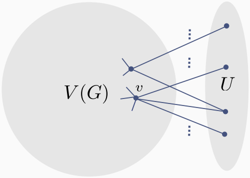


Replace the random V-to-U edges with a *fully explicit expander*.

Fully explicit, d-regular expander

For every degree $d \geq 3$ and sufficiently large n, there is an algorithm that constructs, deterministically, a d-regular $\Omega(1)$-expander in near-linear time.

Attempt to Derandomize the Core Gadget



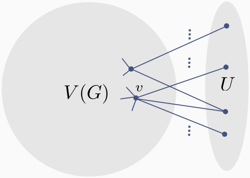
Replace the random V-to-U edges with a *fully explicit expander*.

Fully explicit, d-regular expander

For every degree $d \geq 3$ and sufficiently large n, there is an algorithm that constructs, deterministically, a d-regular $\Omega(1)$-expander in near-linear time.

Which d should we pick?

Attempt to Derandomize the Core Gadget



Replace the random V-to-U edges with a *fully explicit expander*.

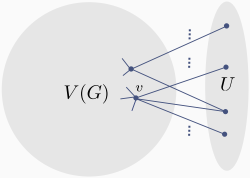
Fully explicit, d-regular expander

For every degree $d \geq 3$ and sufficiently large n, there is an algorithm that constructs, deterministically, a d-regular $\Omega(1)$-expander in near-linear time.

Which d should we pick?

- $d = 3$. The resulting graph is not necessarily an expander.

Attempt to Derandomize the Core Gadget



Replace the random V-to-U edges with a *fully explicit expander*.

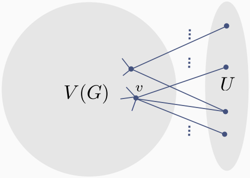
Fully explicit, d-regular expander

For every degree $d \geq 3$ and sufficiently large n, there is an algorithm that constructs, deterministically, a d-regular $\Omega(1)$-expander in near-linear time.

Which d should we pick?

- $d = 3$. The resulting graph is not necessarily an expander.
- $d = \Theta(n)$. The blowup is too big.

Attempt to Derandomize the Core Gadget



Replace the random V-to-U edges with a *fully explicit expander*.

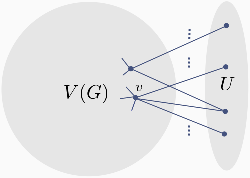
Fully explicit, d-regular expander

For every degree $d \geq 3$ and sufficiently large n, there is an algorithm that constructs, deterministically, a d-regular $\Omega(1)$-expander in near-linear time.

Which d should we pick?

- $d = 3$. The resulting graph is not necessarily an expander.
- $d = \Theta(n)$. The blowup is too big.
- $d = \Theta(m/n)$. Same problem as before. E.g., a graph with $m = O(n)$ edges that contains an isolated $\Theta(\sqrt{n})$-clique.

Attempt to Derandomize the Core Gadget



Replace the random V-to-U edges with a *fully explicit expander*.

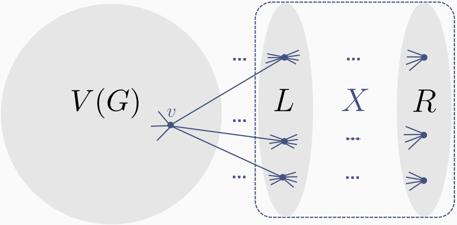
Fully explicit, d-regular expander

For every degree $d \geq 3$ and sufficiently large n, there is an algorithm that constructs, deterministically, a d-regular $\Omega(1)$-expander in near-linear time.

Open Question

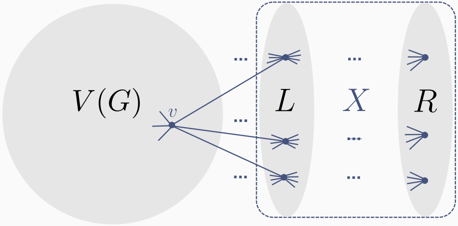
Is there an explicit construction of expanders that, given a degree sequence $(\deg(v_1), \deg(v_2), \ldots, \deg(v_n))$, construct an $\Omega(1)$-expander with this degree sequence?

Workaround: Add a degree-balancing layer L of size n.



Derandomized Core Gadget

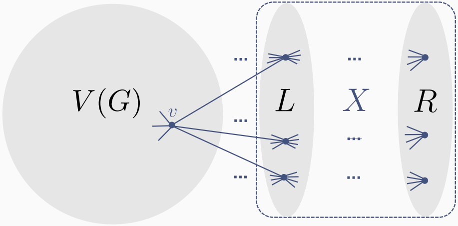
Workaround: Add a degree-balancing layer L of size n.



1. For every $v \in V$, add $\deg_G(v)$ neighbors in L using a deterministic load-balancing rule.

Derandomized Core Gadget

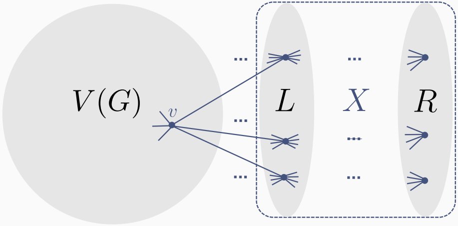
Workaround: Add a degree-balancing layer L of size n.



1. For every $v \in V$, add $\deg_G(v)$ neighbors in L using a deterministic load-balancing rule. Now L is almost d-regular, where $d = 2m/n$.

Derandomized Core Gadget

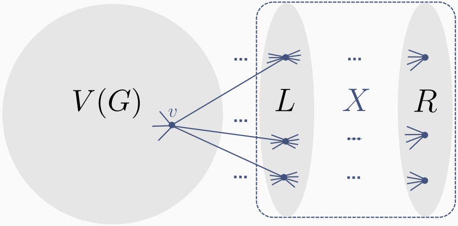
Workaround: Add a degree-balancing layer L of size n.



1. For every $v \in V$, add $\deg_G(v)$ neighbors in L using a deterministic load-balancing rule. Now L is almost d-regular, where $d = 2m/n$.
2. Construct deterministically a d-regular expander X between L and R.

Derandomized Core Gadget

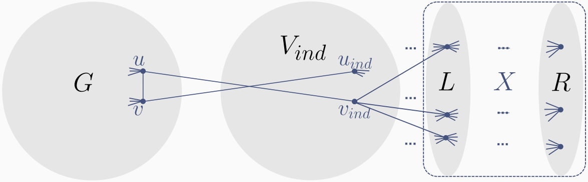
Workaround: Add a degree-balancing layer L of size n.



1. For every $v \in V$, add $\deg_G(v)$ neighbors in L using a deterministic load-balancing rule. Now L is almost d-regular, where $d = 2m/n$.
2. Construct deterministically a d-regular expander X between L and R.

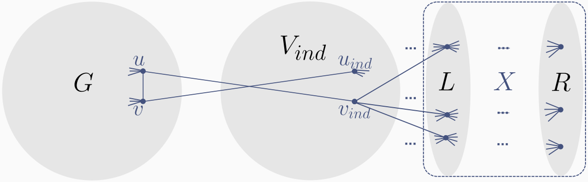
Claim: G_{exp} is an $\Omega(1)$-expander, and its size is $O(m + n)$.

Example: Direct Reduction for k-Clique Detection



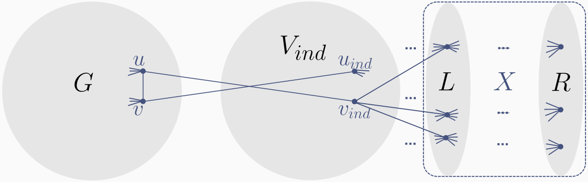
1. Add a copy $v_{ind} \in V_{ind}$ to every $v \in V$.

Example: Direct Reduction for k-Clique Detection



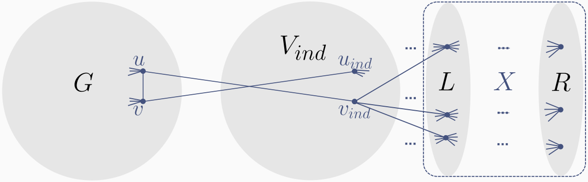
1. Add a copy $v_{ind} \in V_{ind}$ to every $v \in V$.
2. For every edge $uv \in E(G)$, add edges uv_{ind} and $u_{ind}v$.

Example: Direct Reduction for k-Clique Detection



1. Add a copy $v_{ind} \in V_{ind}$ to every $v \in V$.
2. For every edge $uv \in E(G)$, add edges uv_{ind} and $u_{ind}v$.
3. Add an expansion layer to V_{ind}.

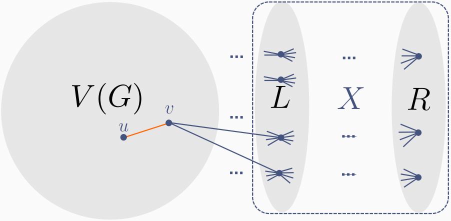
Example: Direct Reduction for k-Clique Detection



1. Add a copy $v_{ind} \in V_{ind}$ to every $v \in V$.
2. For every edge $uv \in E(G)$, add edges uv_{ind} and $u_{ind}v$.
3. Add an expansion layer to V_{ind}.

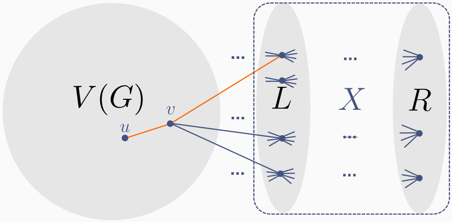
Claim: G_{exp} is an $\Omega(1)$-expander and the number of k-cliques in G_{exp} is $k\times$(the number of k-cliques G).

Attempt to Dynamize the Core Gadget



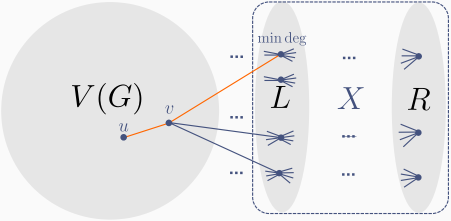
- For every insertion of an edge uv in G, add two edges from u, v to L.

Attempt to Dynamize the Core Gadget



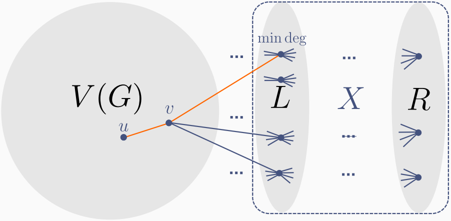
- For every insertion of an edge uv in G, add two edges from u, v to L.
- To maintain balanced degrees, choose the minimum-degree vertex in L.

Attempt to Dynamize the Core Gadget



- For every insertion of an edge uv in G, add two edges from u, v to L.
- To maintain balanced degrees, choose the minimum-degree vertex in L.

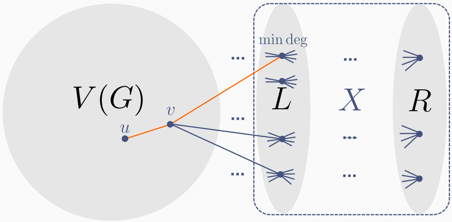
Attempt to Dynamize the Core Gadget



- For every insertion of an edge uv in G, add two edges from u, v to L.
- To maintain balanced degrees, choose the minimum-degree vertex in L.

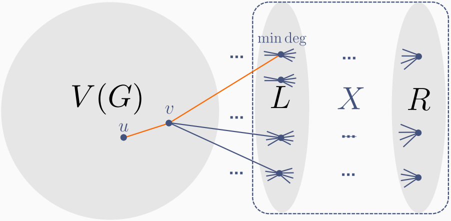
Subtle issue - we may introduce parallel edges to the graph.

Fully-Dynamic Core Gadget



Solution: (greedy + lazy) strategy.

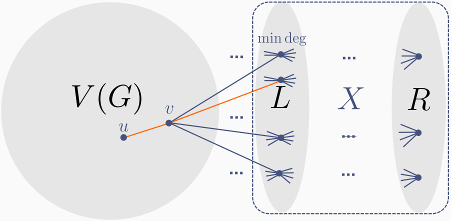
Fully-Dynamic Core Gadget



Solution: (greedy + lazy) strategy.

- Greedy - choose the minimum degree vertex in $L \setminus N(v)$. This requires $O(\deg(v))$ successor queries.

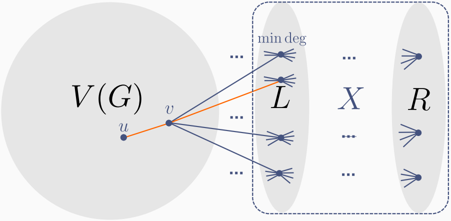
Fully-Dynamic Core Gadget



Solution: (greedy + lazy) strategy.

- Greedy - choose the minimum degree vertex in $L \setminus N(v)$. This requires $O(\deg(v))$ successor queries.

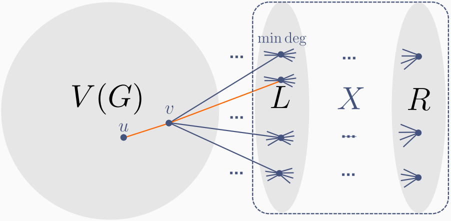
Fully-Dynamic Core Gadget



Solution: (greedy + lazy) strategy.

- Greedy - choose the minimum degree vertex in $L \setminus N(v)$. This requires $O(\deg(v))$ successor queries.
- Lazy - wait until the degree of v doubles itself before adding new neighbors in L.

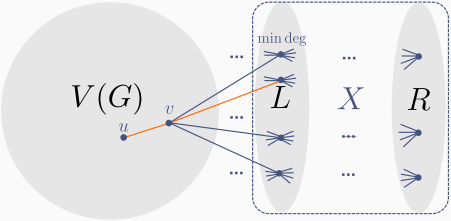
Fully-Dynamic Core Gadget



Solution: (greedy + lazy) strategy.

- Greedy - choose the minimum degree vertex in $L \setminus N(v)$. This requires $O(\deg(v))$ successor queries.
- Lazy - wait until the degree of v doubles itself before adding new neighbors in L.
- Periodic recomputation - recompute all V-to-L edges when the degrees in L become overly unbalanced.

Fully-Dynamic Core Gadget



Solution: (greedy + lazy) strategy.

- Greedy - choose the minimum degree vertex in $L \setminus N(v)$. This requires $O(\deg(v))$ successor queries.
- Lazy - wait until the degree of v doubles itself before adding new neighbors in L.
- Periodic recomputation - recompute all V-to-L edges when the degrees in L become overly unbalanced.

Claim: The degrees in L are in $\Theta(2m/n)$ for at least $\Omega(m+n)$ updates.

Summary and Open Questions

Our Results

The following problems admit deterministic, fully-dynamic, worst-case to expander-case reductions:
k-Clique Detection, Maximum Matching, Dynamic Densest Subgraph, Max-Cut, Minimum Vertex Cover, Minimum Dominating Set.

Summary and Open Questions

Our Results

The following problems admit deterministic, fully-dynamic, worst-case to expander-case reductions:
k-Clique Detection, Maximum Matching, Dynamic Densest Subgraph, Max-Cut, Minimum Vertex Cover, Minimum Dominating Set.

Open Question

- Demanding that the reduction outputs a ϕ-expander, where $\phi = \Omega(1)$, might be too coarse. Some problems (e.g., Triangle Detection) remain hard on $1/100$-expanders but become easy on $(1/2 - o(1))$-expanders. Find the exact threshold.
- Algorithms parameterized by ϕ?