Worst-case to Expander-case Reductions

Amir Abboud, Nathan Wallheimer

Inspiration: The Expander Decomposition Method

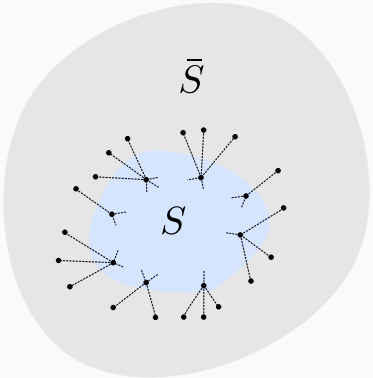
A powerful algorithmic technique that was used to achieve many breakthroughs in the recent years:

- Max-flow in $m^{1+o(1)}$ time.
- Decremental shortest paths in $m^{1+o(1)}$ total time.
- Computing a Gomory-Hu Tree in $n^{2+o(1)}$ time
- Dynamic minimum spanning tree in $n^{o(1)}$ update time.

$\vdots$

What is the expander decomposition method?

Expander Graphs



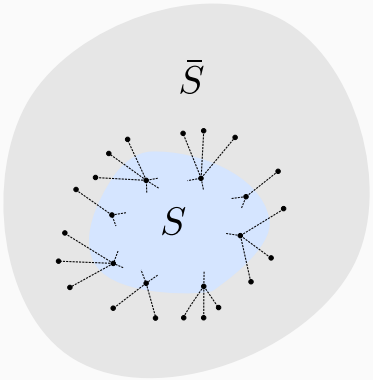
Definition

G is a ϕ-expander if for every non-empty cut $S \subseteq V(G)$, such that $vol(S) \leq vol(\bar{S})$, where $vol(S) = \sum_{v \in S} \deg(v)$, we have:

$$\phi(S) := e(S, \bar{S}) \geq \phi vol(S),$$

We refer to $\phi(S)$ as the *conductance* of S.

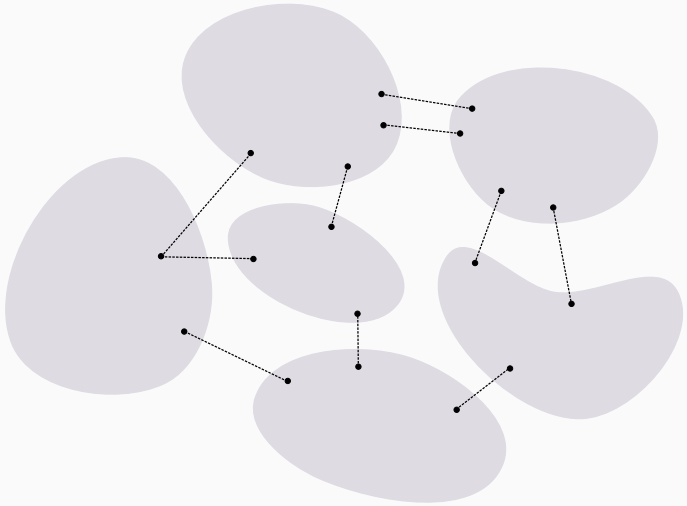
Expander Graphs



Expanders admit the following properties, that improve as $\phi \to 1$:

- Small diameter.
- Can't disconnect large portions of the graph without removing many edges.
- Random walks mix fast.

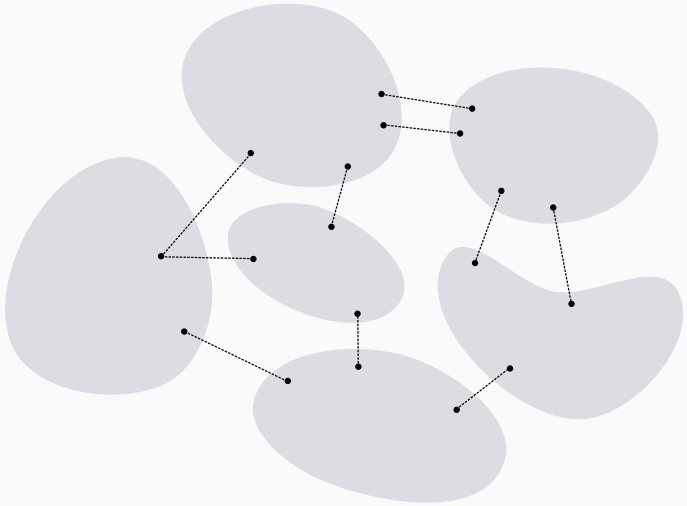
Expander Decompositions



Theorem

For any graph G and $0 < \phi < 1$, the vertex set $V(G)$ can be partitioned into $V_1, V_2, \ldots, V_k$, such that:

Expander Decompositions

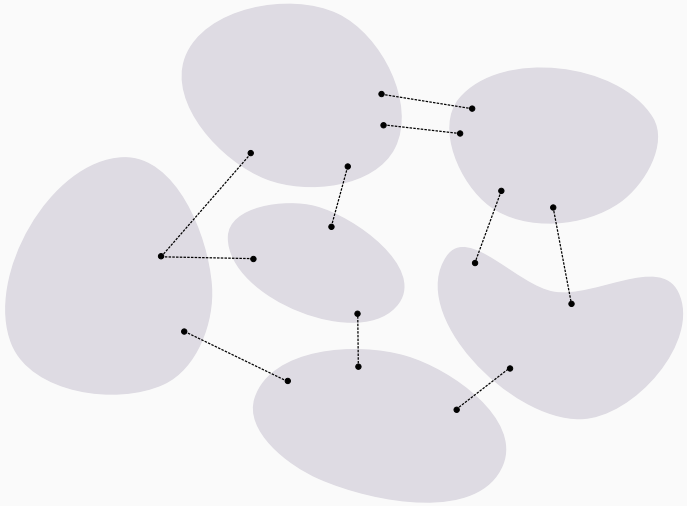


Theorem

For any graph G and $0 < \phi < 1$, the vertex set $V(G)$ can be partitioned into $V_1, V_2, \ldots, V_k$, such that:

- The induced subgraphs $G[V_1], G[V_2], \ldots, G[V_k]$ are ϕ-expanders.

Expander Decompositions

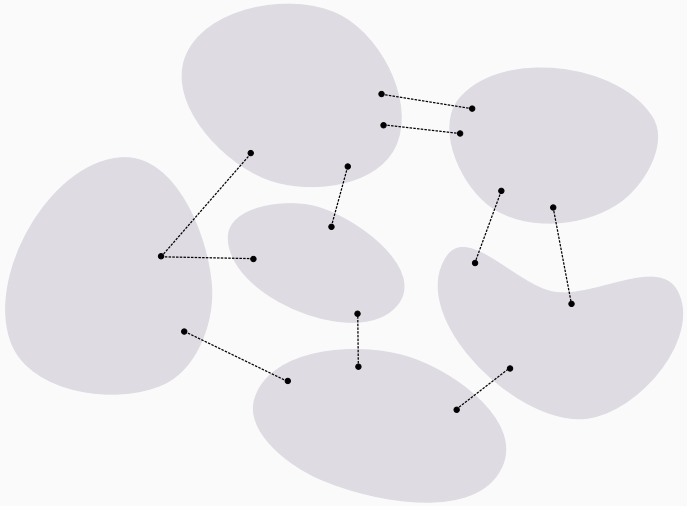


Theorem

For any graph G and $0 < \phi < 1$, the vertex set $V(G)$ can be partitioned into $V_1, V_2, \ldots, V_k$, such that:

- The induced subgraphs $G[V_1], G[V_2], \ldots, G[V_k]$ are ϕ-expanders.
- The total number of edges between the subgraphs is $\tilde{O}(\phi m)$.

Expander Decompositions

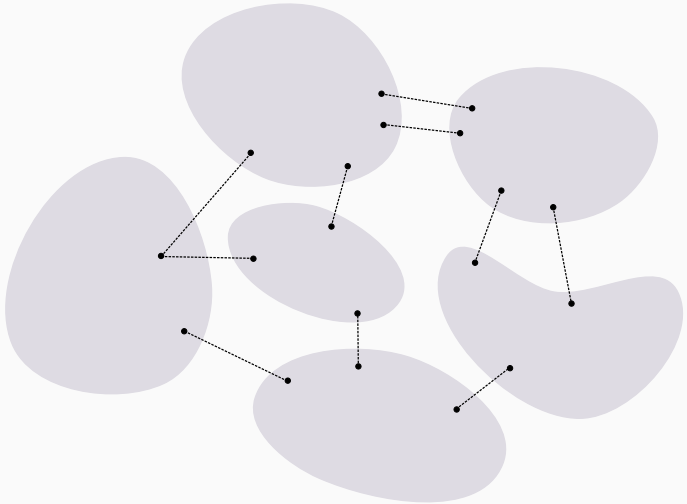


Theorem

For any graph G and $0 < \phi < 1$, the vertex set $V(G)$ can be partitioned into $V_1, V_2, \ldots, V_k$, such that:

- The induced subgraphs $G[V_1], G[V_2], \ldots, G[V_k]$ are ϕ-expanders.
- The total number of edges between the subgraphs is $\tilde{O}(\phi m)$.
- We can compute this partition in $\tilde{O}(m/\phi)$ time.

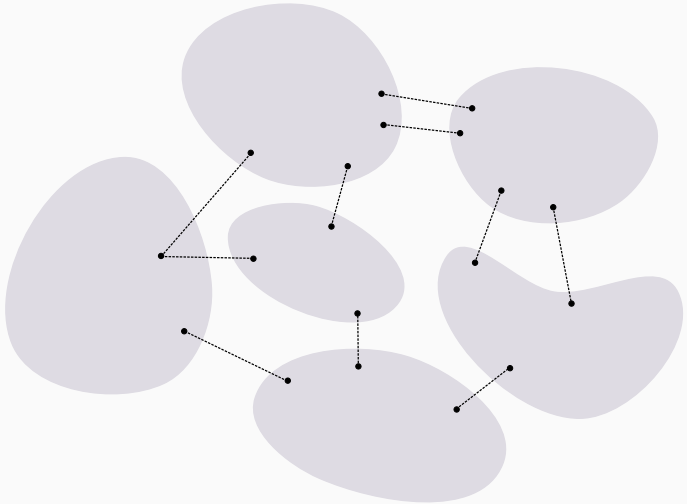
The Expander Decomposition Method



The expander decomposition method for problem $\mathcal{A}$:

1. Decompose G into ϕ-expanders $G_1, G_2, \ldots, G_k$, and $\tilde{O}(\phi m)$ outer edges.

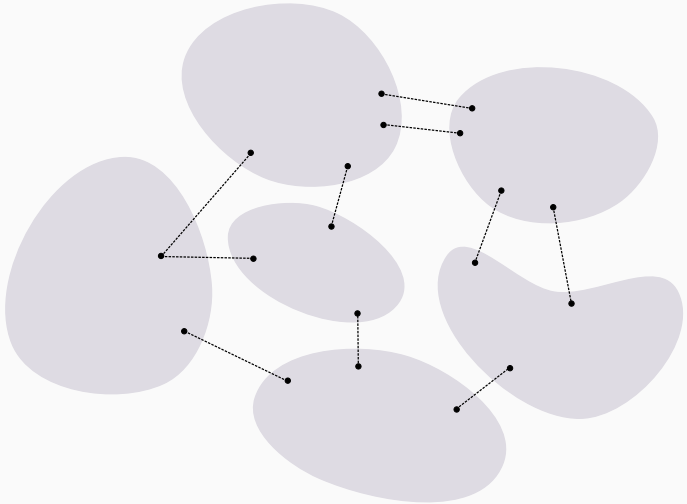
The Expander Decomposition Method



The expander decomposition method for problem $\mathcal{A}$:

1. Decompose G into ϕ-expanders $G_1, G_2, \ldots, G_k$, and $\tilde{O}(\phi m)$ outer edges.
2. Run some computation on each expander G_i, to obtain a solution $\mathcal{A}(G_i)$.

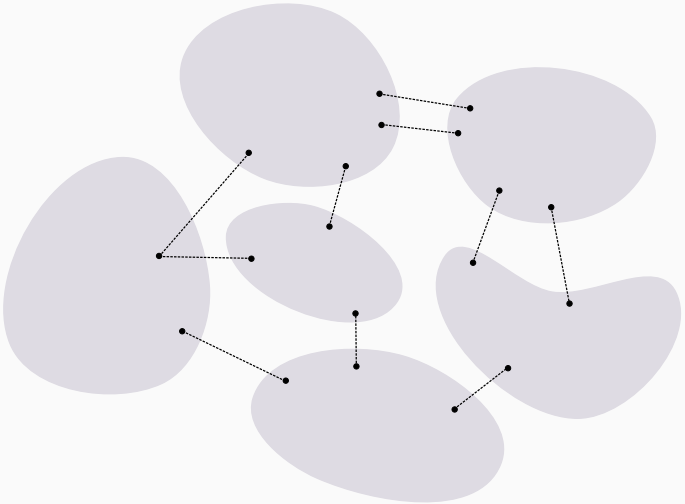
The Expander Decomposition Method



The expander decomposition method for problem $\mathcal{A}$:

1. Decompose G into ϕ-expanders $G_1, G_2, \ldots, G_k$, and $\tilde{O}(\phi m)$ outer edges.
2. Run some computation on each expander G_i, to obtain a solution $\mathcal{A}(G_i)$. Gain a speed-up in this step, utilizing the properties of expanders!

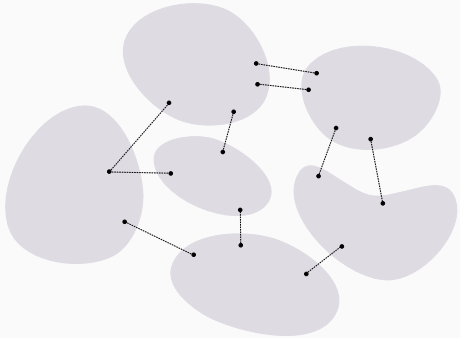
The Expander Decomposition Method



The expander decomposition method for problem $\mathcal{A}$:

1. Decompose G into ϕ-expanders $G_1, G_2, \ldots, G_k$, and $\tilde{O}(\phi m)$ outer edges.
2. Run some computation on each expander G_i, to obtain a solution $\mathcal{A}(G_i)$. Gain a speed-up in this step, utilizing the properties of expanders!
3. Run some computation on the remaining $\tilde{O}(\phi m)$ edges, and together with $\mathcal{A}(G_1), \ldots, \mathcal{A}(G_k)$, obtain the answer $\mathcal{A}(G)$.

Our Main Questions



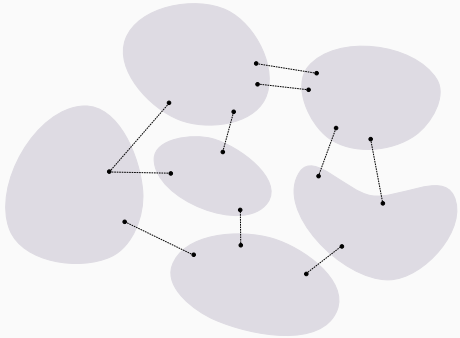
Intuition for the success of this paradigm:

"All problems are as easy on worst-case graphs as they are on expanders."

Question 1

Is this intuition true?

Our Main Questions



- Maximum Matching
- k-Clique
- 4-Cycle Detection
- ⋮

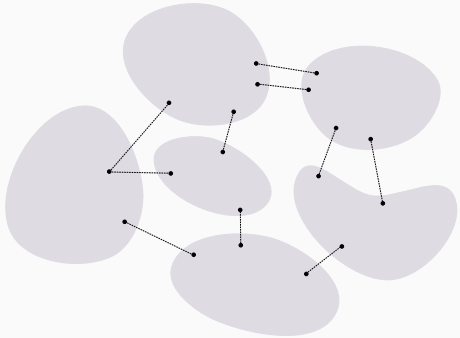
Intuition for the success of this paradigm:

"All problems are as easy on worst-case graphs as they are on expanders."

Question 1

Is this intuition true?

Our Main Questions



- Maximum Matching
- k-Clique
- 4-Cycle Detection
- ⋮

Intuition for the success of this paradigm:

"All problems are as easy on worst-case graphs as they are on expanders."

Question 1

Is this intuition true?

Question 2

Is the expander decompositon method the key to resolve the remaining open problems?

Our Contribution: Worst-case to Expander-case Reductions

To answer **Q1**, we need a way to reduce the worst-case instances of a problem into expander graphs.

Our Contribution: Worst-case to Expander-case Reductions

To answer **Q1**, we need a way to reduce the worst-case instances of a problem into expander graphs. We could try to apply the *expander decomposition method*!

Our Contribution: Worst-case to Expander-case Reductions

To answer **Q1**, we need a way to reduce the worst-case instances of a problem into expander graphs. We could try to apply the *expander decomposition method*!

However, we shall take a step back and start with a simpler approach.

Our Contribution: Worst-case to Expander-case Reductions

To answer **Q1**, we need a way to reduce the worst-case instances of a problem into expander graphs. We could try to apply the *expander decomposition method*!

However, we shall take a step back and start with a simpler approach.

Direct Worst-case to Expander-case Reductions:

An algorithm that given G, outputs G' such that:

Our Contribution: Worst-case to Expander-case Reductions

To answer **Q1**, we need a way to reduce the worst-case instances of a problem into expander graphs. We could try to apply the *expander decomposition method*!

However, we shall take a step back and start with a simpler approach.

Direct Worst-case to Expander-case Reductions:

An algorithm that given G, outputs G' such that:

- G' is a ϕ-expander.

Our Contribution: Worst-case to Expander-case Reductions

To answer **Q1**, we need a way to reduce the worst-case instances of a problem into expander graphs. We could try to apply the *expander decomposition method*!

However, we shall take a step back and start with a simpler approach.

Direct Worst-case to Expander-case Reductions:

An algorithm that given G, outputs G' such that:

- G' is a ϕ-expander.
- The solution $\mathcal{A}(G)$ can be easily computed from $\mathcal{A}(G')$.

Our Contribution: Worst-case to Expander-case Reductions

To answer **Q1**, we need a way to reduce the worst-case instances of a problem into expander graphs. We could try to apply the *expander decomposition method*!

However, we shall take a step back and start with a simpler approach.

Direct Worst-case to Expander-case Reductions:

An algorithm that given G, outputs G' such that:

- G' is a ϕ-expander.
- The solution $\mathcal{A}(G)$ can be easily computed from $\mathcal{A}(G')$.
- The *blowup* in the number of vertices and edges in G' is small, meaning that the number of vertices and edges in G' is $\tilde{O}(n), \tilde{O}(m)$.

Our Contribution: Worst-case to Expander-case Reductions

To answer **Q1**, we need a way to reduce the worst-case instances of a problem into expander graphs. We could try to apply the *expander decomposition method*!

However, we shall take a step back and start with a simpler approach.

Direct Worst-case to Expander-case Reductions:

An algorithm that given G, outputs G' such that:

- G' is a ϕ-expander.
- The solution $\mathcal{A}(G)$ can be easily computed from $\mathcal{A}(G')$.
- The *blowup* in the number of vertices and edges in G' is small, meaning that the number of vertices and edges in G' is $\tilde{O}(n), \tilde{O}(m)$.

Ideally: $\phi = \Omega(1)$, and the reduction runs in $\tilde{O}(m)$ time

Our Contribution: Worst-case to Expander-case Reductions

To answer **Q1**, we need a way to reduce the worst-case instances of a problem into expander graphs. We could try to apply the *expander decomposition method*!

However, we shall take a step back and start with a simpler approach.

Direct Worst-case to Expander-case Reductions:

If $\mathcal{A}$ admits such a reduction:

- The worst-case instances of $\mathcal{A}$ are expanders! Thus, we get a positive answer to **Q1**.
- The expander decomposition method is useless against $\mathcal{A}$, therefore we get a negative answer to **Q2**.

Direct Worst-case to Expander-case Reductions

Our Results

Theorem. The following problems admit direct reductions:

- Maximum Cardinality Matching
- Triangle Detection
- 4-Cycle Detection in sparse graphs (i.e. $m = \tilde{O}(n)$)
- Minimum Vertex Cover, Minimum Dominating Set

Q1: Their worst-case instances are $\Omega(1)$-expanders!

Q2: The expander decomopsition method is useless against them!

Maximum Matching and Triangle Detection

Maximum Matching and Triangle Detection

Maximum Cardinality Matching

Goal: find a maximum cardinality set $M \subseteq E(G)$, such that no two edges in M intersect.

The best upper bound is $O(m \cdot \sqrt{n})$, obtained by the Micali and Vazirani algorithm. For dense graphs, there is also an $\tilde{O}(n^{\omega})$ upper bound.

Maximum Matching and Triangle Detection

Maximum Cardinality Matching

Goal: find a maximum cardinality set $M \subseteq E(G)$, such that no two edges in M intersect.

The best upper bound is $O(m \cdot \sqrt{n})$, obtained by the Micali and Vazirani algorithm. For dense graphs, there is also an $\tilde{O}(n^{\omega})$ upper bound.

In bipartite graphs, this problem can be solved in near-linear time using the expander decomposition method.

Maximum Matching and Triangle Detection

Maximum Cardinality Matching

Goal: find a maximum cardinality set $M \subseteq E(G)$, such that no two edges in M intersect.

The best upper bound is $O(m \cdot \sqrt{n})$, obtained by the Micali and Vazirani algorithm. For dense graphs, there is also an $\tilde{O}(n^{\omega})$ upper bound.

In bipartite graphs, this problem can be solved in near-linear time using the expander decomposition method.

Triangle Detection

Goal: Detect whether G contains a triangle or not.

The best upper bound is $O(\min\{n^{\omega}, m^{\frac{2\omega}{\omega+1}}\})$.

Maximum Matching and Triangle Detection

Maximum Cardinality Matching

Goal: find a maximum cardinality set $M \subseteq E(G)$, such that no two edges in M intersect.

The best upper bound is $O(m \cdot \sqrt{n})$, obtained by the Micali and Vazirani algorithm. For dense graphs, there is also an $\tilde{O}(n^{\omega})$ upper bound.

In bipartite graphs, this problem can be solved in near-linear time using the expander decomposition method.

Triangle Detection

Goal: Detect whether G contains a triangle or not.

The best upper bound is $O(\min\{n^{\omega}, m^{\frac{2\omega}{\omega+1}}\})$.

Triangle Detection Conjecture: the above running time is the best we can do.

Maximum Matching and Triangle Detection

Maximum Cardinality Matching

Goal: find a maximum cardinality set $M \subseteq E(G)$, such that no two edges in M intersect.

The best upper bound is $O(m \cdot \sqrt{n})$, obtained by the Micali and Vazirani algorithm. For dense graphs, there is also an $\tilde{O}(n^{\omega})$ upper bound.

In bipartite graphs, this problem can be solved in near-linear time using the expander decomposition method.

Triangle Detection

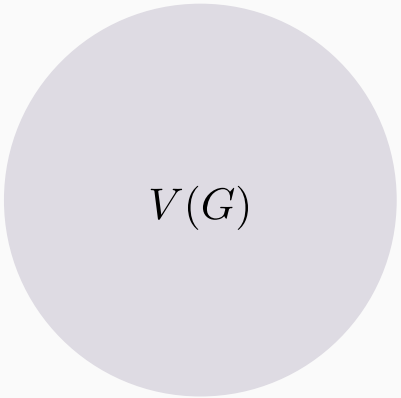
Goal: Detect whether G contains a triangle or not.

The best upper bound is $O(\min\{n^{\omega}, m^{\frac{2\omega}{\omega+1}}\})$.

Triangle Detection Conjecture: the above running time is the best we can do.

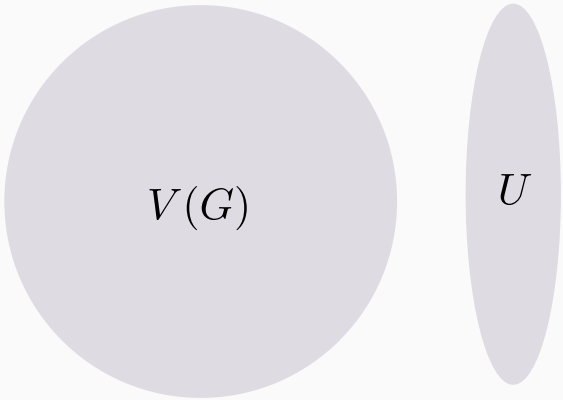
Perhaps the expander decomposition method is the key to resolve the complexity of these problems?

Basic Expander Construction



Given a graph G, construct an expander graph G' as follows.

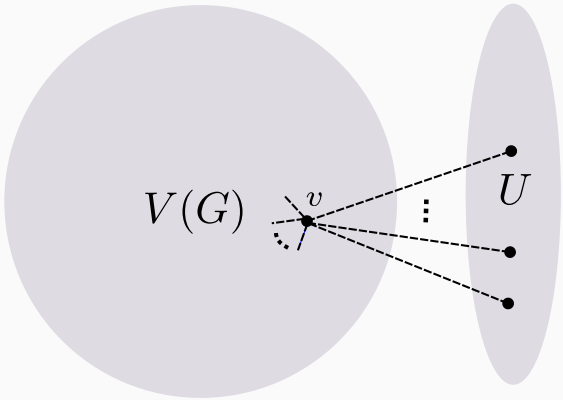
Basic Expander Construction



Given a graph G, construct an expander graph G' as follows.

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

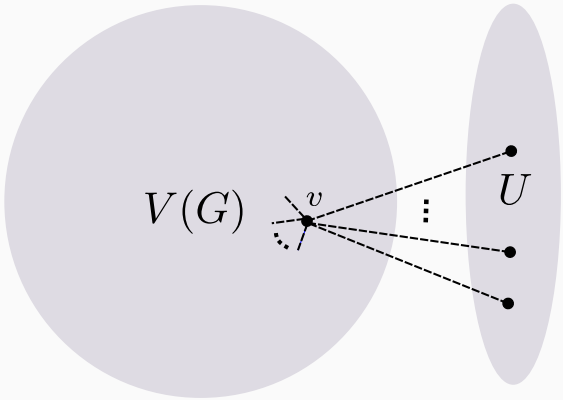
Basic Expander Construction



Given a graph G, construct an expander graph G' as follows.

1. Add an *expansion layer* U, that consists of $\Theta(n)$ vertices.
2. For every $v \in V(G)$, sample $\deg(v)$ vertices from U and add edges to them.

Basic Expander Construction

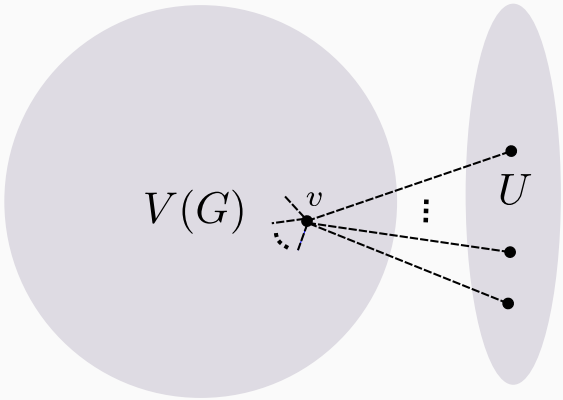


Given a graph G, construct an expander graph G' as follows.

1. Add an *expansion layer* U, that consists of $\Theta(n)$ vertices.
2. For every $v \in V(G)$, sample $\deg(v)$ vertices from U and add edges to them.

Claim: The number of added edges is $O(m)$.
Claim: Assuming that the degree of every vertex is at least $\Omega(\log n)$, then w.h.p., G' is an $\Omega(1)$-expander.

Basic Expander Construction

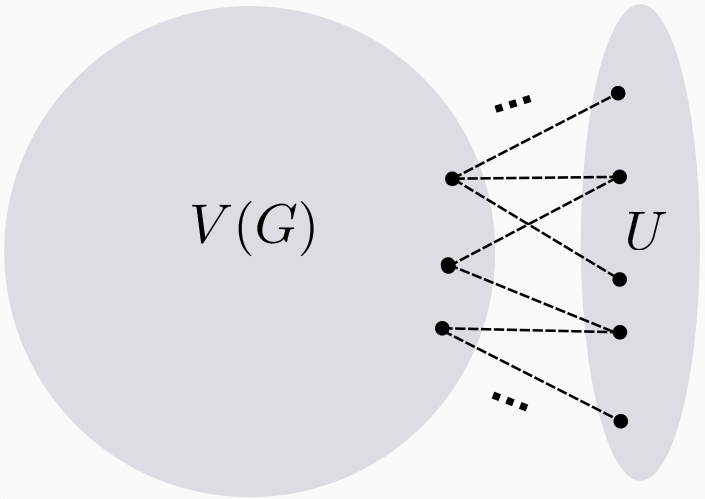


Given a graph G, construct an expander graph G' as follows.

1. Add an *expansion layer* U, that consists of $\Theta(n)$ vertices.
2. For every $v \in V(G)$, sample $\deg(v)$ vertices from U and add edges to them.

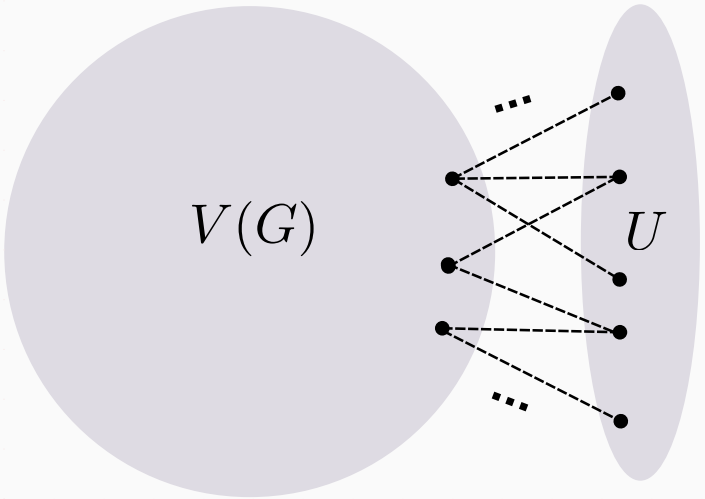
Claim: The number of added edges is $O(m)$.
Claim: Assuming that the degree of every vertex is at least $\Omega(\log n)$, then w.h.p., G' is an $\Omega(1)$-expander.
Idea: Random walks mix well in this graph. Whenever the random walk is in G, it escapes to U with probability $1/2$.

Reduction for Maximum Cardinality Matching



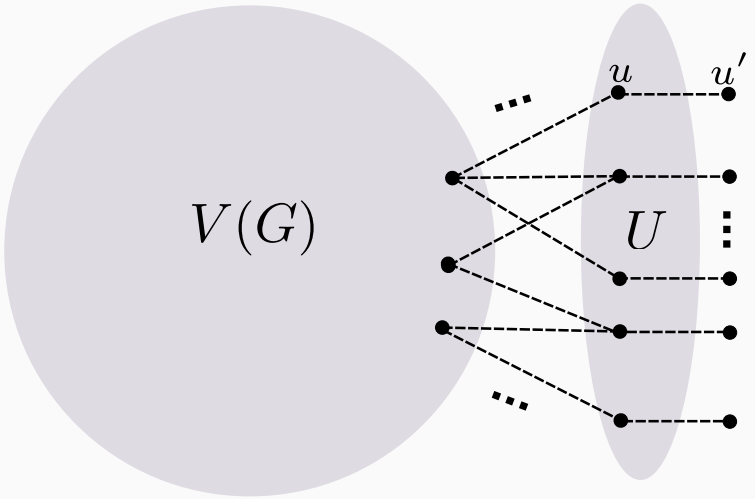
1. Add an expansion layer U.

Reduction for Maximum Cardinality Matching



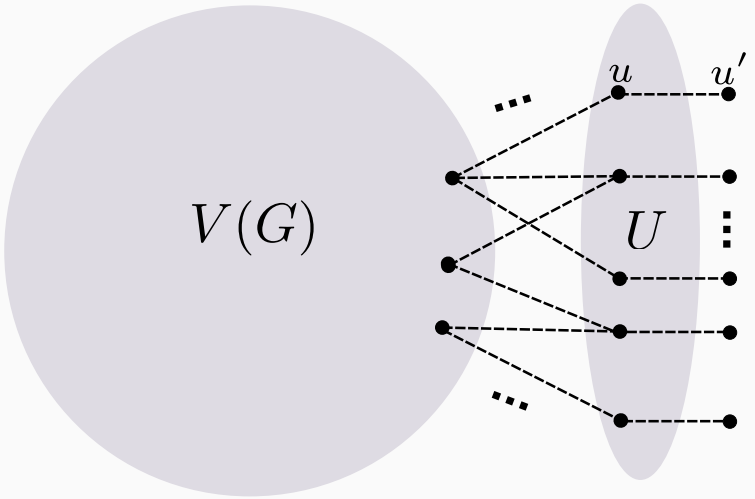
1. Add an expansion layer U. Problem: $MCM(G')$ is not fixed.

Reduction for Maximum Cardinality Matching



1. Add an expansion layer U. Problem: $MCM(G')$ is not fixed.
2. Add a "twin" vertex u' to every $u \in U$.

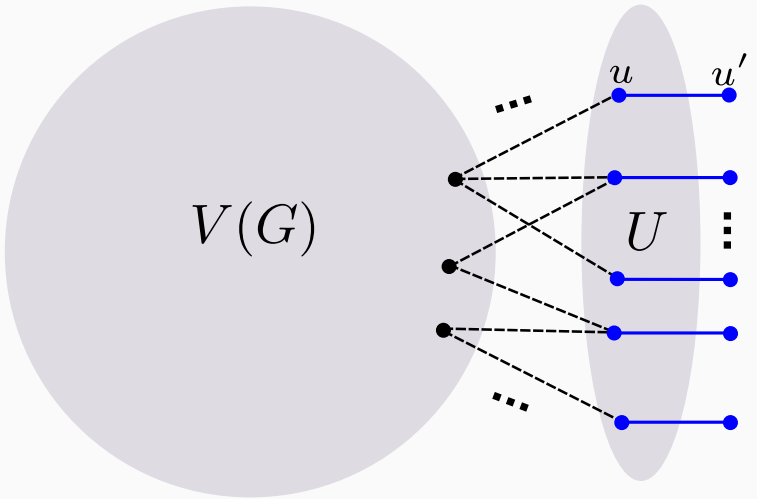
Reduction for Maximum Cardinality Matching



1. Add an expansion layer U. Problem: $MCM(G')$ is not fixed.
2. Add a "twin" vertex u' to every $u \in U$.

Claim: $MCM(G') = MCM(G) + |U|$

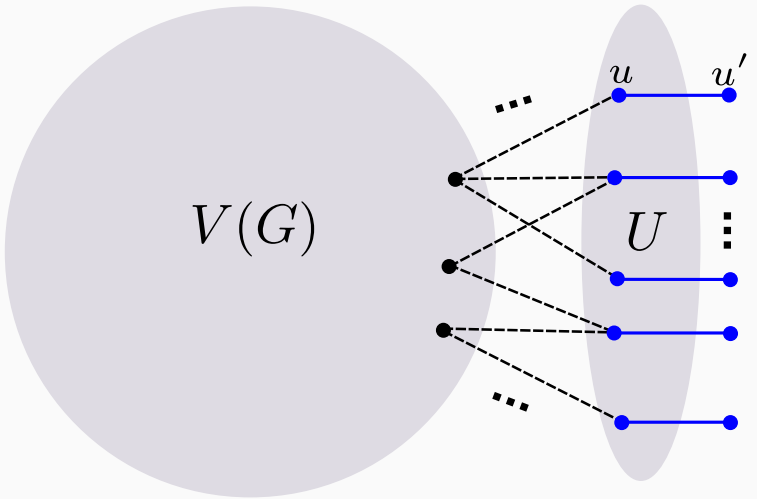
Reduction for Maximum Cardinality Matching



1. Add an expansion layer U. Problem: $MCM(G')$ is not fixed.
2. Add a "twin" vertex u' to every $u \in U$.

Claim: $MCM(G') = MCM(G) + |U|$

Reduction for Maximum Cardinality Matching

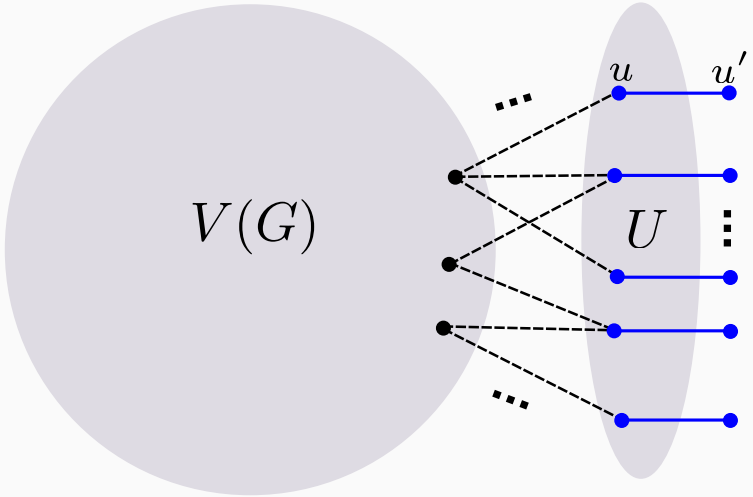


1. Add an expansion layer U. Problem: $MCM(G')$ is not fixed.
2. Add a "twin" vertex u' to every $u \in U$.

Claim: $MCM(G') = MCM(G) + |U|$

Claim: w.h.p., G' is an $\Omega(1)$-expander.

Reduction for Maximum Cardinality Matching



Theorem: Maximum Cardinality Matching admits a Direct Worst-case to Expander-case Reduction.

Question 1

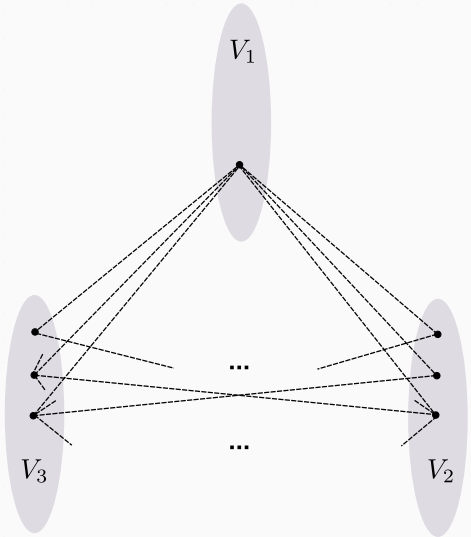
Is this problem as easy on worst-case graphs as it is on expanders?

Question 2

Is the expander decompositon method the key to resolve this problem?

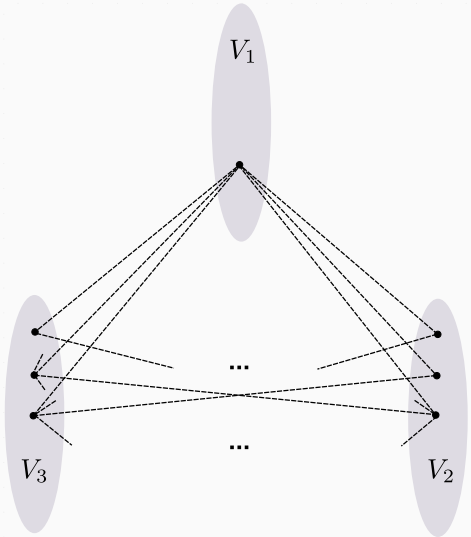
We get a positive answer to **Q1** and a negative answer to **Q2**.

Reduction for Triangle Detection



Assumption: G is 3-partite, and every vertex has the same number of neighbors in every part except its own.

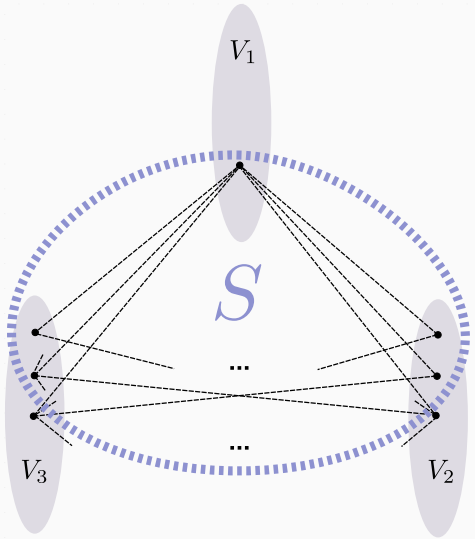
Reduction for Triangle Detection



Assumption: G is 3-partite, and every vertex has the same number of neighbors in every part except its own.

To achieve this assumption, we take 3 copies of $V(G)$, and a copy of $E(G)$ between each pair.

Reduction for Triangle Detection



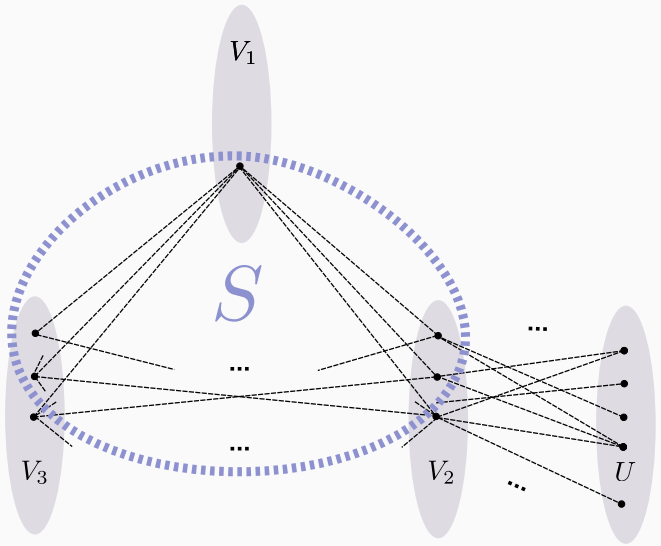
Assumption: G is 3-partite, and every vertex has the same number of neighbors in every part except its own.

To achieve this assumption, we take 3 copies of $V(G)$, and a copy of $E(G)$ between each pair.

Claim: Any sparse cut S must have about the same volume in every part:

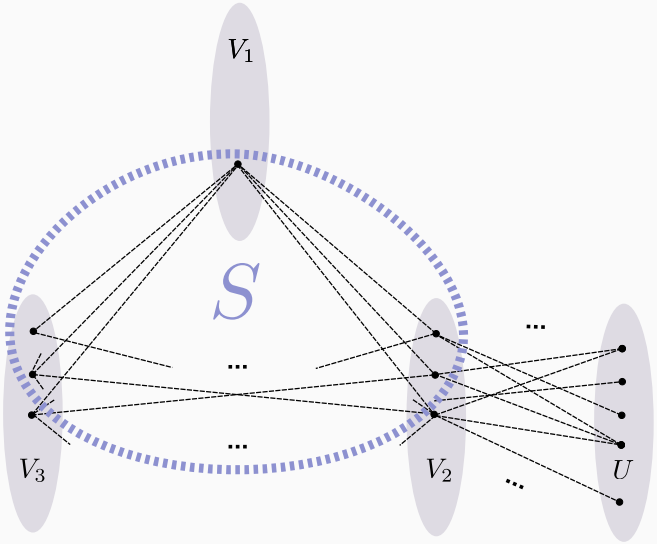
$$vol(S \cap V_i) = \Theta(vol(S)/3)$$

Reduction for Triangle Detection



Construction: Add an expansion layer adjacent to one of the parts.

Reduction for Triangle Detection

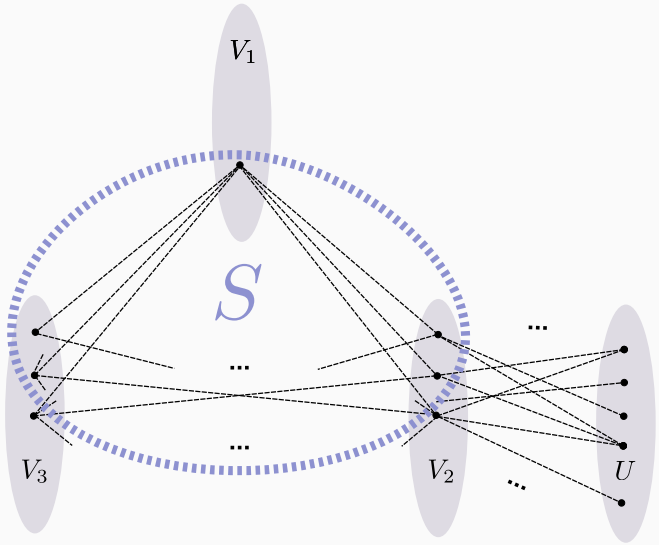


Construction: Add an expansion layer adjacent to one of the parts.
Idea: Since $vol(S \cap V_2) = \Theta(vol(S))$, we have:

$$e(S \cap V_2, U) = \Omega(vol(S)),$$

and therefore $\phi(S) = \Omega(1)$.

Reduction for Triangle Detection



Theorem: Triangle Detection admits a Direct Worst-case to Expander-case Reduction.

Question 1

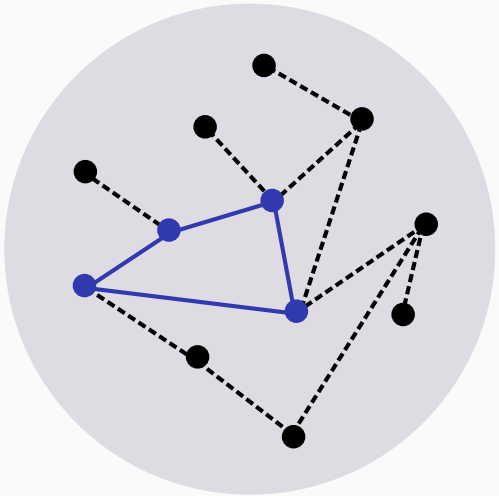
Is this problem as easy on worst-case graphs as it is on expanders?

Question 2

Is the expander decompositon method the key to resolve this problem?

We get a positive answer to **Q1** and a negative answer to **Q2**.

The 4-Cycle Detection Problem

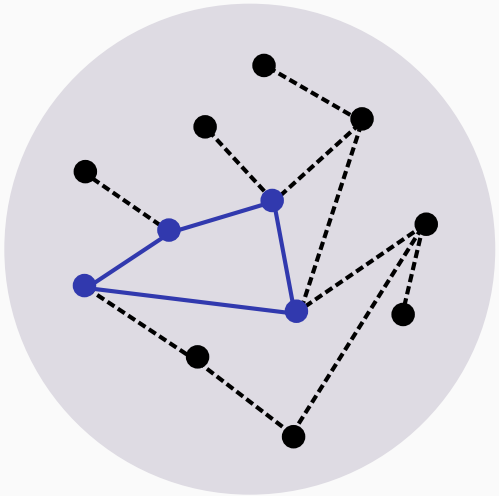


4-Cycle Detection

Goal: Detect whether G contains a 4-cycle.

Complexity: If $m \geq n^{3/2}$, the graph must contain a 4-cycle and the problem becomes trivial. For smaller m, the best known upper bound is $O(m^{4/3}) = O(n^2)$. There is a recent lower bound of $\Omega(m^{1.1})$, assuming the hardness of Triangle Detection.

The 4-Cycle Detection Problem



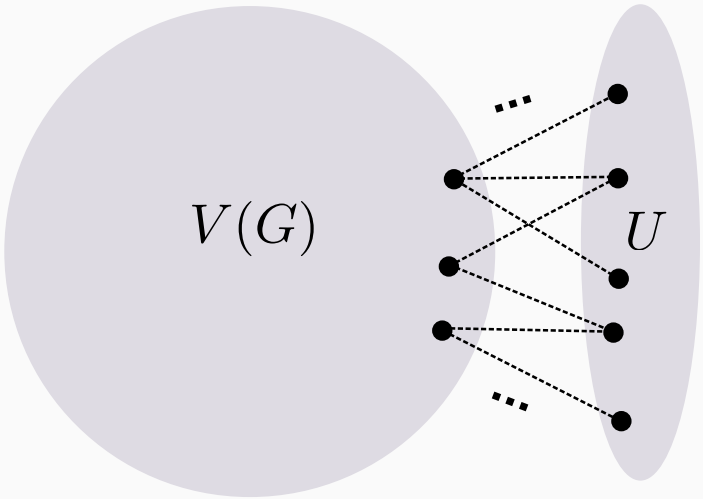
4-Cycle Detection

Goal: Detect whether G contains a 4-cycle.

Complexity: If $m \geq n^{3/2}$, the graph must contain a 4-cycle and the problem becomes trivial. For smaller m, the best known upper bound is $O(m^{4/3}) = O(n^2)$. There is a recent lower bound of $\Omega(m^{1.1})$, assuming the hardness of Triangle Detection.

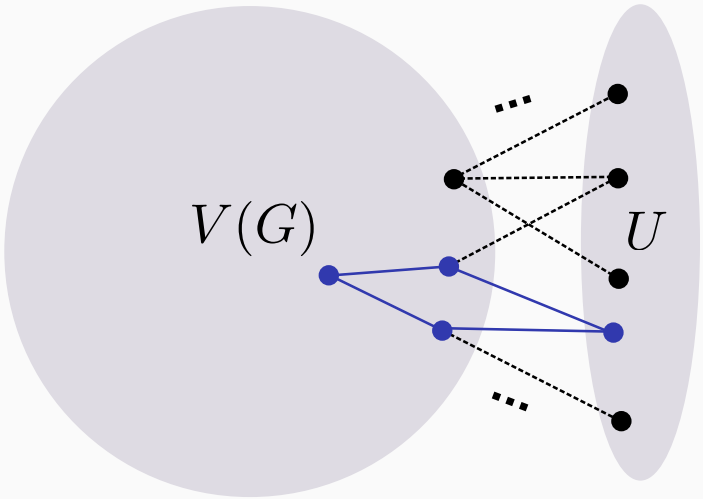
Big open problem: Can we solve 4-Cycle Detection in $n^{2-\Omega(1)}$ time on graphs with $m = \Theta(n^{1.5})$ edges?

Reduction for 4-Cycle Detection



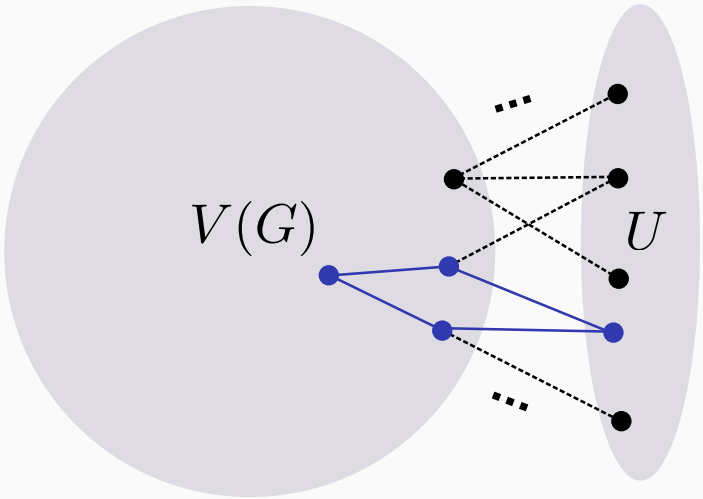
1. Add an expansion layer U.

Reduction for 4-Cycle Detection



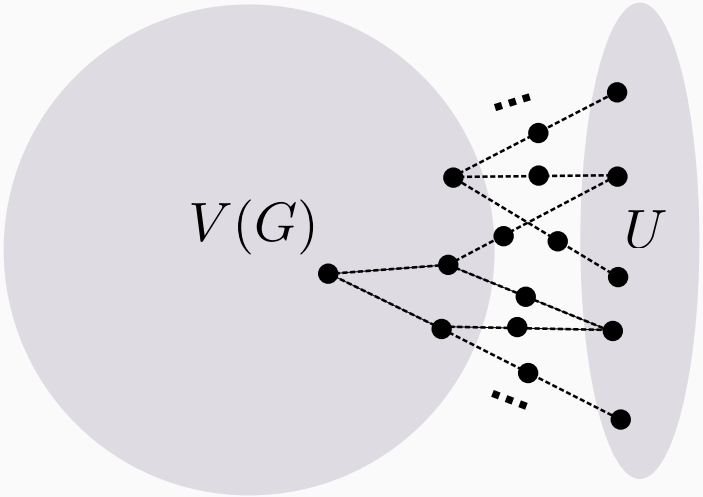
1. Add an expansion layer U. Problem: we might have added 4-cycles.

Reduction for 4-Cycle Detection



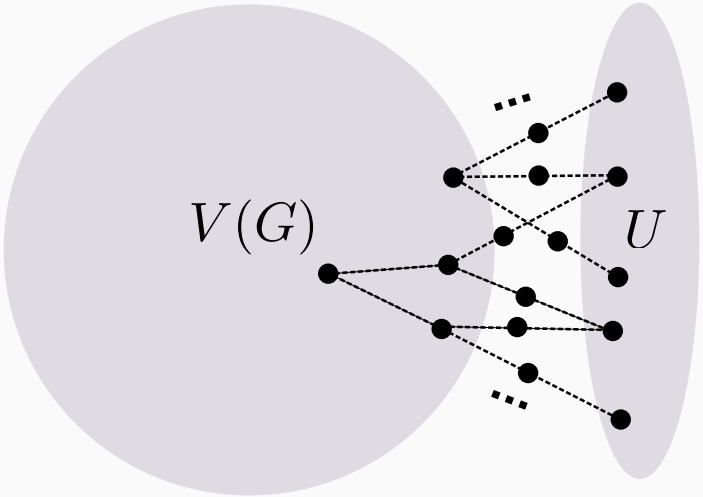
1. Add an expansion layer U. Problem: we might have added 4-cycles.
2. Subdivide each new edge once.

Reduction for 4-Cycle Detection



1. Add an expansion layer U. Problem: we might have added 4-cycles.
2. Subdivide each new edge once.

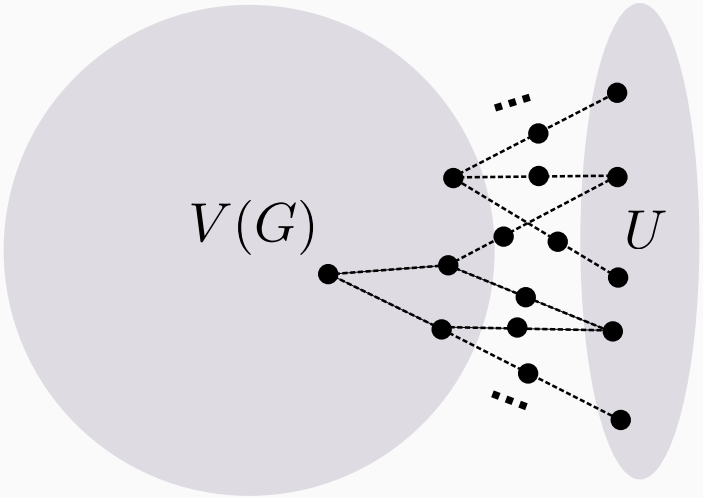
Reduction for 4-Cycle Detection



1. Add an expansion layer U. Problem: we might have added 4-cycles.
2. Subdivide each new edge once.

Caveat: the blowup in the number of vertices is $\Omega(m)$.

Reduction for 4-Cycle Detection



Theorem: 4-Cycle admits a Direct Worst-case to Expander-case Reduction with blowup $\Omega(m)$ to the number of vertices.

Question 1

Is this problem as easy on worst-case graphs as it is on expanders?

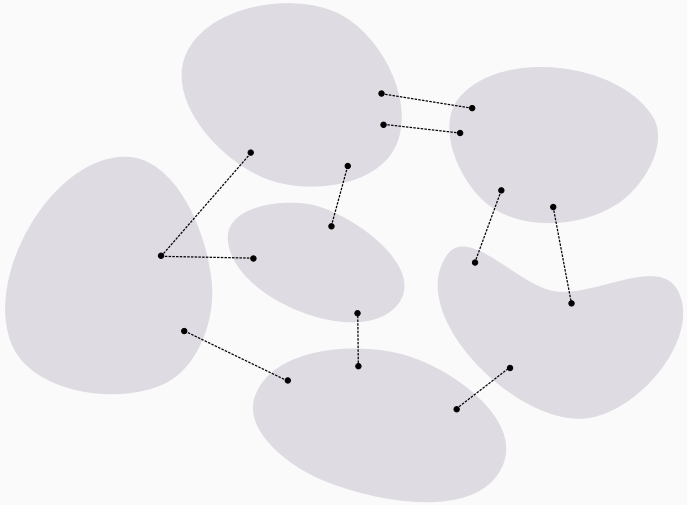
Question 2

Is the expander decompositon method the key to resolve this problem?

We do not get a definitive answer to questions **Q1** and **Q2**, unless the graph is sparse.

Expander Decomposition Based Reductions

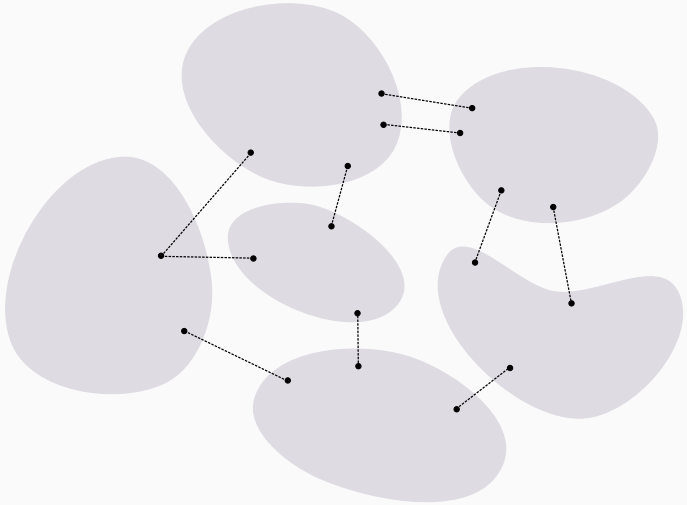
Expander Decomposition Based Reductions



Shortcomings of Expander Decomposition Based Reductions:

- They produce ϕ-expanders, where $\phi \leq (\log n)^{-O(1)}$. This is because of known limits to expander decompositions. Therefore, they cannot prove hardness on *true*, $\Omega(1)$-expanders.

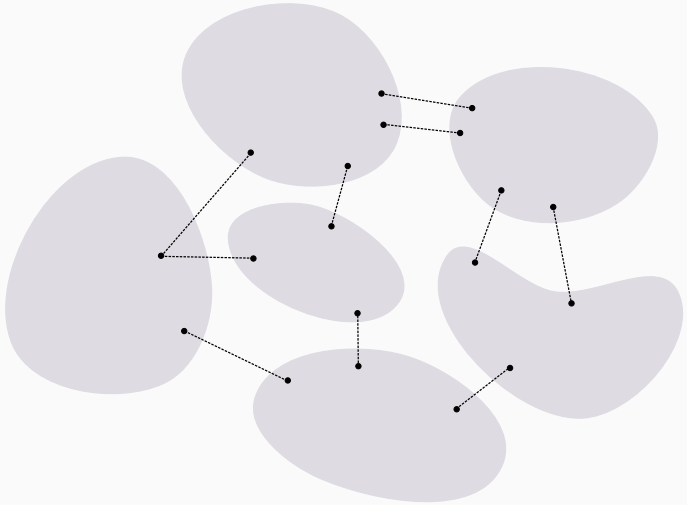
Expander Decomposition Based Reductions



Shortcomings of Expander Decomposition Based Reductions:

- They produce ϕ-expanders, where $\phi \leq (\log n)^{-O(1)}$. This is because of known limits to expander decompositions. Therefore, they cannot prove hardness on *true*, $\Omega(1)$-expanders.
- They rely on the heavy machinery involved with expander decompositions.

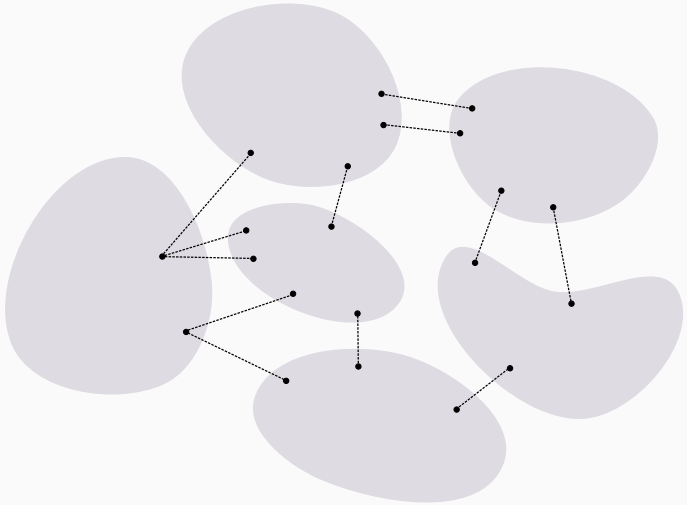
Expander Decomposition Based Reductions



Shortcomings of Expander Decomposition Based Reductions:

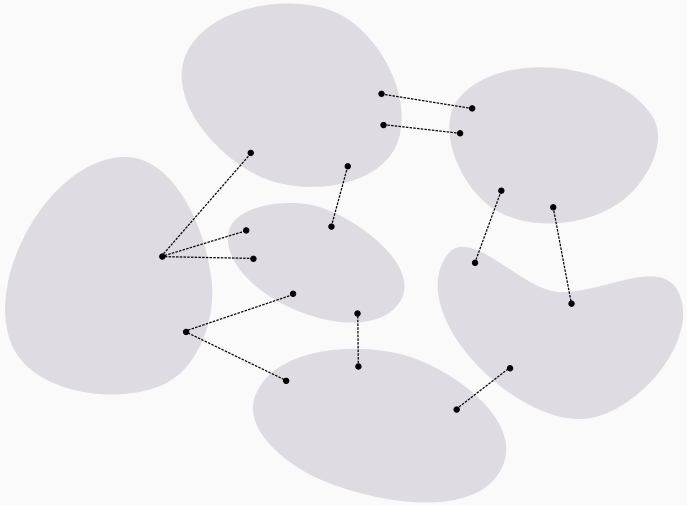
- They produce ϕ-expanders, where $\phi \leq (\log n)^{-O(1)}$. This is because of known limits to expander decompositions. Therefore, they cannot prove hardness on *true*, $\Omega(1)$-expanders.
- They rely on the heavy machinery involved with expander decompositions.
- Their running time is strongly super-linear.

Expander Decomposition Based Reduction for 4-Cycle



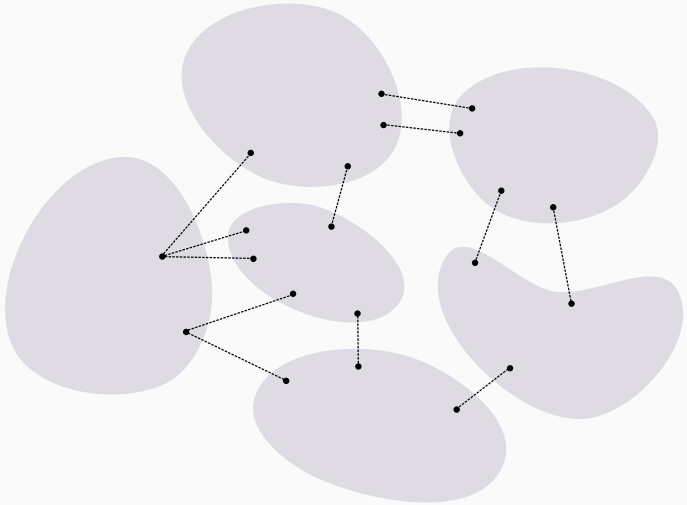
1. Decompose G into $n^{-\varepsilon}$-expanders, where $0 < \varepsilon < 1$ is a sufficiently small constant.

Expander Decomposition Based Reduction for 4-Cycle



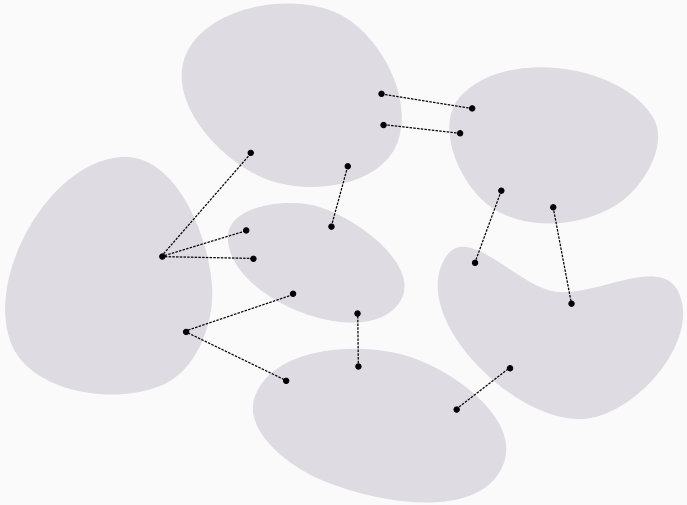
1. Decompose G into $n^{-\varepsilon}$-expanders, where $0 < \varepsilon < 1$ is a sufficiently small constant. The number of outer edges is $\tilde{O}(\phi m) = \tilde{O}(n^{1.5-\varepsilon})$.

Expander Decomposition Based Reduction for 4-Cycle



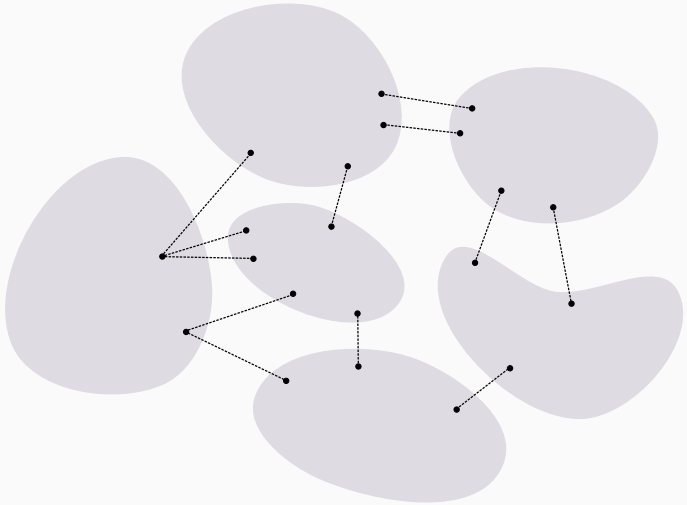
1. Decompose G into $n^{-\varepsilon}$-expanders, where $0 < \varepsilon < 1$ is a sufficiently small constant. The number of outer edges is $\tilde{O}(\phi m) = \tilde{O}(n^{1.5-\varepsilon})$.
2. Detect 4-cycles that use an outer edge.

Expander Decomposition Based Reduction for 4-Cycle



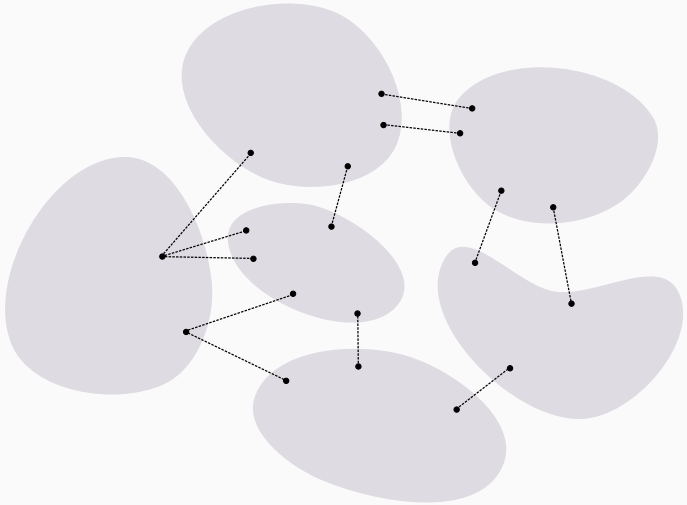
1. Decompose G into $n^{-\varepsilon}$-expanders, where $0 < \varepsilon < 1$ is a sufficiently small constant. The number of outer edges is $\tilde{O}(\phi m) = \tilde{O}(n^{1.5-\varepsilon})$.
2. Detect 4-cycles that use an outer edge.
   Claim: Can be done in time $\tilde{O}(n^{2-0.5\varepsilon})$.

Expander Decomposition Based Reduction for 4-Cycle



1. Decompose G into $n^{-\varepsilon}$-expanders, where $0 < \varepsilon < 1$ is a sufficiently small constant. The number of outer edges is $\tilde{O}(\phi m) = \tilde{O}(n^{1.5-\varepsilon})$.
2. Detect 4-cycles that use an outer edge.
   Claim: Can be done in time $\tilde{O}(n^{2-0.5\varepsilon})$.
3. For the clusters, use a 4-Cycle Detection oracle on $n^{-\varepsilon}$-expanders.

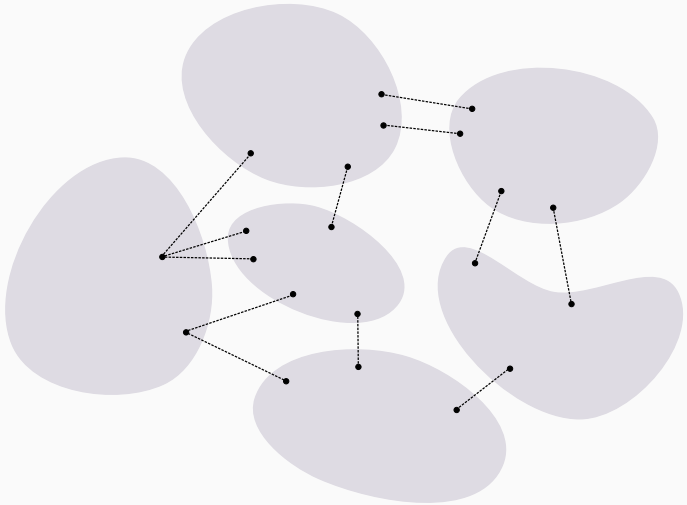
Expander Decomposition Based Reduction for 4-Cycle



1. Decompose G into $n^{-\varepsilon}$-expanders, where $0 < \varepsilon < 1$ is a sufficiently small constant. The number of outer edges is $\tilde{O}(\phi m) = \tilde{O}(n^{1.5-\varepsilon})$.
2. Detect 4-cycles that use an outer edge.
   Claim: Can be done in time $\tilde{O}(n^{2-0.5\varepsilon})$.
3. For the clusters, use a 4-Cycle Detection oracle on $n^{-\varepsilon}$-expanders.

Total *reduction* time: $\tilde{O}(n^{1.5+\varepsilon} + n^{2-0.5\varepsilon})$. Any $\varepsilon < 0.25$ gives $n^{2-\Omega(1)}$.

Expander Decomposition Based Reduction for 4-Cycle

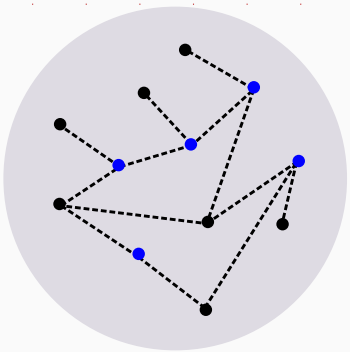


Theorem: 4-Cycle Detection admits a worst-case to expander-case reduction to $n^{-\varepsilon}$-expanders, that uses expander decomposition.

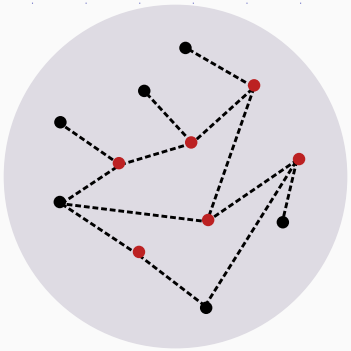
Q1: Any $n^{2-\Omega(1)}$ algorithm for $n^{-\varepsilon}$-expanders implies an $n^{2-\Omega(1)}$ algorithm for general graphs.

Q2: Expander decomposition might be the key.

Exponential Time Problems



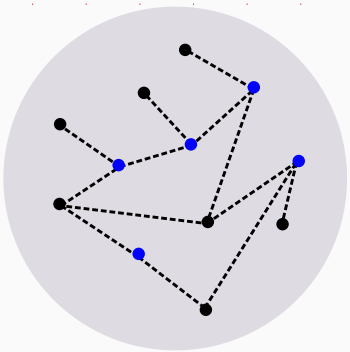
Minimum Dominating Set



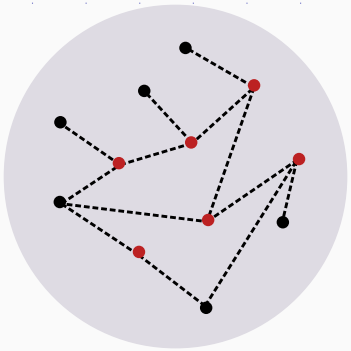
Minimum Vertex Cover

Question: What is the smallest constant c, such that $\mathcal{A}$ can be solved in $c^n \cdot n^{O(1)}$ time.

Exponential Time Problems



Minimum Dominating Set

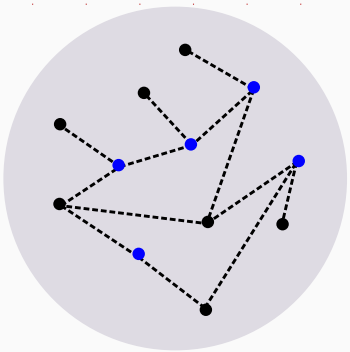


Minimum Vertex Cover

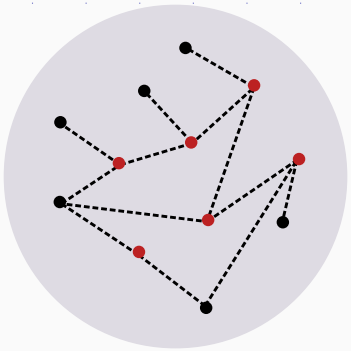
Question: What is the smallest constant c, such that $\mathcal{A}$ can be solved in $c^n \cdot n^{O(1)}$ time.

Caveat: An expansion layer of $|U| = \Theta(n)$ vertices is too expensive.

Exponential Time Problems



Minimum Dominating Set



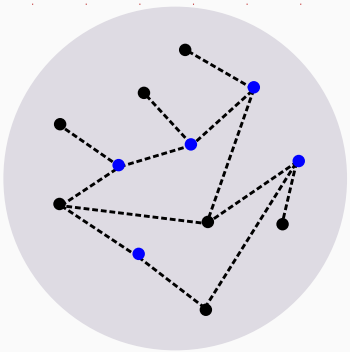
Minimum Vertex Cover

Question: What is the smallest constant c, such that $\mathcal{A}$ can be solved in $c^n \cdot n^{O(1)}$ time.

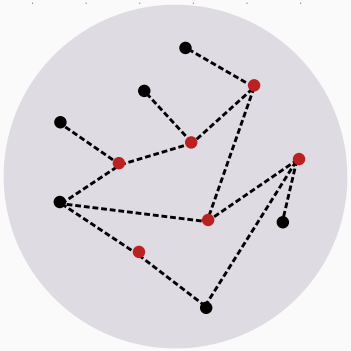
Caveat: An expansion layer of $|U| = \Theta(n)$ vertices is too expensive.

- For Vertex Cover, we obtain a reduction with $|U| = O(\log n)$ to $\Omega(1)$-expanders. For Dominating Set, we obtain reduction with $|U| = \varepsilon n$ to $poly(\varepsilon)$-expanders.

Exponential Time Problems



Minimum Dominating Set



Minimum Vertex Cover

Question: What is the smallest constant c, such that $\mathcal{A}$ can be solved in $c^n \cdot n^{O(1)}$ time.

Caveat: An expansion layer of $|U| = \Theta(n)$ vertices is too expensive.

- For Vertex Cover, we obtain a reduction with $|U| = O(\log n)$ to $\Omega(1)$-expanders. For Dominating Set, we obtain reduction with $|U| = \varepsilon n$ to $poly(\varepsilon)$-expanders.
- Parameterized Complexity of Vertex Cover: We manage to get $VC(G') = VC(G) + O(\log n)$. Thus, the hardness holds even against parameterized algorithms!

Open Questions

1. An improved reduction for 4-Cycle, that produces $\Omega(1)$-expanders?

Open Questions

1. An improved reduction for 4-Cycle, that produces $\Omega(1)$-expanders?
2. Adapt the reductions to other models of computation?

Open Questions

1. An improved reduction for 4-Cycle, that produces $\Omega(1)$-expanders?
2. Adapt the reductions to other models of computation?
   Dynamic, Approximate, Streaming, Distributed, Sublinear.

Open Questions

1. An improved reduction for 4-Cycle, that produces $\Omega(1)$-expanders?
2. Adapt the reductions to other models of computation?
   Dynamic, Approximate, Streaming, Distributed, Sublinear.
3. Provide *worst-case to pseudorandom-case* reductions, for other notions of pseudo-randomness?

Open Questions

1. An improved reduction for 4-Cycle, that produces $\Omega(1)$-expanders?
2. Adapt the reductions to other models of computation?
   Dynamic, Approximate, Streaming, Distributed, Sublinear.
3. Provide *worst-case to pseudorandom-case* reductions, for other notions of pseudo-randomness?
   (p, β)-jumbled graphs?

Open Questions

1. An improved reduction for 4-Cycle, that produces $\Omega(1)$-expanders?
2. Adapt the reductions to other models of computation?
   Dynamic, Approximate, Streaming, Distributed, Sublinear.
3. Provide *worst-case to pseudorandom-case* reductions, for other notions of pseudo-randomness?
   (p, β)-jumbled graphs?
4. Provide worst-case to pseudorandom-case reductions, but not for graphs?

Open Questions

1. An improved reduction for 4-Cycle, that produces $\Omega(1)$-expanders?
2. Adapt the reductions to other models of computation?
   Dynamic, Approximate, Streaming, Distributed, Sublinear.
3. Provide *worst-case to pseudorandom-case* reductions, for other notions of pseudo-randomness?
   (p, β)-jumbled graphs?
4. Provide worst-case to pseudorandom-case reductions, but not for graphs?
   Pseudorandom strings?