Worst-Case to Expander-Case Reductions

Amir Abboud ✉ 🆔
Weizmann Institute of Science, Rehovot, Israel

Nathan Wallheimer ✉ 🆔
Weizmann Institute of Science, Rehovot, Israel

Abstract

In recent years, the *expander decomposition method* was used to develop many graph algorithms, resulting in major improvements to longstanding complexity barriers. This powerful hammer has led the community to (1) believe that most problems are as easy on worst-case graphs as they are on expanders, and (2) suspect that expander decompositions are the key to breaking the remaining longstanding barriers in fine-grained complexity.

We set out to investigate the extent to which these two things are true (and for which problems). Towards this end, we put forth the concept of *worst-case to expander-case* self-reductions. We design a collection of such reductions for fundamental graph problems, verifying belief (1) for them. The list includes k-Clique, 4-Cycle, Maximum Cardinality Matching, Vertex-Cover, and Minimum Dominating Set. Interestingly, for most (but not all) of these problems the proof is via a simple gadget reduction, *not* via expander decompositions, showing that this hammer is effectively useless against the problem and contradicting (2).

1 Introduction

One of the most effective techniques in modern graph algorithms has been the *expander decomposition method*. To solve a difficult problem on a worst-case instance G on m edges, it roughly proceeds as follows.

1. Decompose G into the disjoint union of *expanders* $H_1, \ldots, H_k$ (also called *clusters*) plus $o(m)$ *outer edges*; such an expander decomposition can be computed in $m^{1+o(1)}$ time [12, 26, 36, 39, 46, 47, 49, 50].
2. Run some computation on each expander H_i, utilizing the special properties of expanders to gain a speed-up, e.g., because the search space is restricted on expanders or because some processes can be accelerated.
3. Run some computation on the $o(m)$ outer edges (e.g., recurse), utilizing the fact that there are not too many such edges.

The last two steps are problem-specific and often involve ingenious techniques. To keep this work focused, we choose to restrict ourselves to the following definition of expanders (that is most popular in our context).

▶ **Definition 1** (Expander). *A graph $G = (V, E)$ is called an ϕ-expander if its conductance is:*

$$\phi(G) := \min_{S \subset V} \frac{|E(S, V \setminus S)|}{\min(vol(S), vol(V \setminus S))} \geq \phi$$

where $vol(S)$ is the sum of degrees of all nodes in S.[1]

The expander decomposition method itself is common to several celebrated results breaking longstanding bounds for classical problems, e.g., [7, 27, 37, 41, 45, 51], leading the community to believe the intuition that "all graph problems are as easy on worst-case graphs as they are on expanders (up to sub-polynomial factors)". In other words, if we can solve the problem fast on expanders then we can also solve it fast on worst-case graphs. Moreover, the expander-decomposition method is viewed as the hammer proving this intuition. Of course, it's unlikely that this intuition is true for *all* problems; since there are problems for which the state of the art on expanders is better than it is on worst-case graphs.[2]

In this work, we set out to investigate the extent to which this intuition is true. Namely, we are looking to answer the following motivating question:

▶ **Question 1.** *Which problems are as easy on worst-case graphs as they are on expanders?*

To this end, we formalize a new notion of *worst-case to expander-case self-reductions.* If a problem admits such a reduction then the answer to this question is positive, meaning that the worst-case graphs of these problems *are* expanders. Essentially, the expander decomposition method can be viewed as a special case of such reductions, assuming that its three steps can be computed in significantly faster time than the best upper bound for the problem.

Since the question is irrelevant for easy problems that are already known to be solvable in near-linear time (that is, in $\tilde{O}(m)$ time where $\tilde{O}(\cdot)$ hides poly-logarithmic factors) in the worst-case, we are looking for difficult problems, for which the best known algorithms still run significantly slower than near-linear (say, in $\Omega(n^2)$ time). To find such problems we look at the state of the art in fine-grained complexity.

Fine-grained complexity, with its reductions-based approach, has been effective at mapping out the landscape of fundamental problems that are difficult. In the context of graph problems, there is a large number of tight conditional lower bounds that are based on the conjectured hardness of detecting a triangle in a graph (Triangle Detection) and the related Minimum-Weight Triangle, k-Clique, Shortest Cycle, etc. (see e.g., [1, 5, 9, 11, 40, 54] and the survey [53]). It is known that in order to make progress on any of a large number of problems, one must first come up with a groundbreaking algorithm for Triangle Detection (and its variants).

Notably, most reductions from Triangle Detection to other problems, e.g. Dynamic Maximum Matching, are *expansion preserving* in the sense that if the Triangle Detection instance is an expander, then so is the Dynamic Maximum Matching instance. The reason is that many of these reductions employ only a small number of local changes to the graph that do not reduce the expansion by much. Consequently, if the answer to the question for Triangle Detection is positive, meaning that the hardest instances are expanders, then it is also positive for many other Triangle-hard problems. Thus, it is most interesting to investigate Question 1 for Triangle Detection and other problems without conditional lower bounds.

---

[1] If G is a singleton then $\phi(G) := 1$.
[2] One interesting example is the problem of computing a Gomory-Hu tree of a graph. In weighted graphs, the expander case can be solved in $m^{1+o(1)}$ time (follows from Theorem 1.4 in [8]) but the worst-case bound is $\tilde{O}(n^2)$ [6].

But if the prevalent intuition is indeed true (due to expander decompositions), and the answer is positive for Triangle Detection, then all we have to do in order to refute the "Triangle conjecture" is to come up with a faster algorithm on expanders. This may appear much easier. After all, expanders are similar in many ways to random graphs and we can solve Triangle Detection on random $G(n, p)$ graphs in expected subquadratic time.[3]

Moreover, three previous results on Triangle Detection exploit its easiness on pseudo-random graphs: a combinatorial mildly subcubic algorithm using the Szemerédi Regularity Lemma [19], fine-grained reductions that make the graph random-like by removing dense pieces to prove hardness for approximate distance oracles [2], and an almost-optimal *distributed* algorithm using expander decompositions [23]. Indeed, many researchers have been wondering whether the "hammer" of expander decompositions is the right tool to refute these conjectures.

▶ **Question 2.** *Are expander decompositions the key towards resolving the open questions of fine-grained complexity?*

Despite the major advancements in fine-grained complexity over the last decade, there are still many remaining gaps in the complexity of classical problems, for which achieving tight lower bounds has been notoriously difficult. For example, we have the problems of Maximum Cardinality Matching, 4-Cycle Detection, and All-Pairs Max-Flow. For the latter problem, a cubic upper bound by Gomory and Hu from 1961 stood for almost 60 years until it was broken (for simple graphs) using the *expander decomposition method* [7]. This makes us even more motivated to study Question 2.

Finally, we note that both motivating questions are relevant even for NP-hard problems, where it is desirable to optimize the *exact* exponential time complexity. So far, the reductions-based approach of fine-grained complexity has only found limited success in this regime (see [28] and an interesting recent barrier [38]), leaving a state of the art that is full of gaps (see [55]). It is natural to wonder if expander decompositions can play a role in this regime too.

1.1 Our Contribution: Worst-Case to Expander-Case Self-Reductions

A *worst-case to expander-case* self-reduction (WTER) is a fine-grained self-reduction that translates some graph problem $\mathcal{A}$ to the same problem on ϕ-expanders, for some value ϕ. Such reductions effectively answer Question 1, and can indeed be achieved for many problems via the expander decomposition method. However, if we take a step back, forget all the technology of expander decompositions, and simply ask if a problem admits a WTER: it is natural to try a more naïve gadget reduction that turns any graph into an expander.

Direct-WTERs

The straightforward approach to design a WTER for some problem $\mathcal{A}$ is by showing an efficient transformation that takes an instance graph G and transforms it into a ϕ-expander G', such that the solution $\mathcal{A}(G)$ can be efficiently computed from $\mathcal{A}(G')$. We will call such reductions *Direct-WTERs*. Direct-WTERs are easy to construct and analyze, they do not

---

[3] Basic probability shows that with high probability, the graph contains at least $0.1n^3p^3$ triangles and that the maximum degree is bounded from above by $10np \log n$. Thus, we can sample triplets of vertices and check if they induce a triangle, while alternately running the $\Delta^2 n$ algorithm for graphs with bounded degree Δ (that checks for an edge between every pair of neighbors for every vertex). This algorithm terminates in expected $\tilde{O}(\min\{1/p^3, n^3p^2\})$ time and it balances to $\tilde{O}(n^{9/5})$ when $p = n^{-3/5}$.

rely on heavy algorithmic machinery, and they create only one expander instance. Moreover, if a problem admits a Direct-WTER then we can assume that the worst-case graphs to this problem are already expanders. It is therefore futile to invest time trying to solve the problem using the expander decomposition method, since applying expander decompositions on worst-case graphs (which are expanders), may as well simply return the graph itself. Effectively, if the problem admits a Direct-WTER then the answer to Question 2 is negative.

In Direct-WTERs, one should try to optimize the following parameters: 1) the conductance ϕ of the output graph G', 2) the running time of transformation, and 3) the *blowup* in the size of G', denoted by $(N, M) = (|V(G')|, |E(G')|)$. We remark that computing $\mathcal{A}(G)$ from $\mathcal{A}(G')$ will always be trivial in our examples (e.g., when $\mathcal{A}(G)$ is a linear function of $\mathcal{A}(G')$), so we do not consider it another parameter. The "gold standard" for Direct-WTERs is to obtain $\phi = \Omega(1)$, near-linear running time and near-linear blowup (meaning that $N = \tilde{O}(n)$ and $M = \tilde{O}(m)$). If a problem admits such gold standard reduction and it can be solved on $\Omega(1)$-expanders in $O(a(n,m))$ time, for some polynomially bounded function $a(n,m)$ (but not linear), then we can also solve on worst-case graphs in $\tilde{O}(a(n,m))$ time.

However, for problems whose best algorithms run in exponential time, we have to be more stringent in the efficiency requirements: even a blowup of $N = 2n$ weakens the result significantly because it means that to get any major speed-up over $2^{\alpha n}$ time on worst-case graphs (for some $\alpha > 0$), one needs to gain a major speed-up over $2^{\alpha n/2}$ in expanders. Namely, gold standard reductions do not necessarily answer Question 1 for exponential-time problems. We can thus speak about a "platinum standard", where in addition to the requirements of the gold standard, we also have $N = n + o(n)$. If a problem admits such platinum standard reduction and it can be solved in time $2^{\alpha(1-\varepsilon)N} \cdot N^{O(1)}$ on $\Omega(1)$-expanders (for any $\varepsilon > 0$), then we can solve this problem in time $2^{\alpha(1-\varepsilon)(n+o(n))} \cdot (n+o(n))^{O(1)} = 2^{n(\alpha(1-\varepsilon)+o(1))} \cdot n^{O(1)}$ on worst-case graphs.

Notably, all the Direct-WTERs in our results achieve $\phi = \Omega(1)$ and near-linear running time (but not necessarily near-linear blowup), so we keep the definition clean by not considering them as parameters:

▶ **Definition 2** (Direct-WTER). *A Direct-WTER to problem $\mathcal{A}$ with blowup (N, M) is a randomized, near-linear time transformation that takes an input graph G and outputs an $\Omega(1)$-expander graph G' w.h.p., such that $|V(G')| = N$, $|E(G')| = M$, and $\mathcal{A}(G)$ can be computed in constant time from $\mathcal{A}(G')$.*

In Section 1.2 we present our results which include very efficient Direct-WTERs for Triangle Detection, Maximum Cardinality Matching, and Vertex Cover, proving that the answer for each of them to Question 1 is positive and to Question 2 is (unfortunately) negative. For other problems, however, such as 4-Cycle, the only Direct-WTERs we could come up with have a super-linear blowup. In such cases, we ask whether a (not direct) WTER that does exploit expander decompositions can be obtained.

ED-WTERs

Before continuing with this discussion, we need to state formally what is an expander decomposition:

▶ **Theorem 3** (Theorems 5.1 and 5.3 in [12]). *Given a graph $G = (V, E)$ of m edges and a parameter ϕ, there is a randomized algorithm that with high probability finds a partitioning of V into $V_1, V_2, \ldots, V_k$ such that for all $i \in [k]$, the induced subgraph $G[V_i]$ has conductance $\phi(G[V_i]) \geq \phi$, and $\sum_{i=1}^{k} \delta(V_i) = O(\phi m \log^2 m)$. The running time of the algorithm is $O(m \log^7 m + \frac{m \log^4 m}{\phi})$.*

Now, the way that an expander decomposition based reduction works can be defined (loosely) in the following way:

▶ **Definition 4** (ED-WTER). *Problem $\mathcal{A}$ admits an* expander decomposition-WTER *with conductance $\phi(n)$ and running time $t(n,m)$ (abbreviated as ED-WTER), if there exists an algorithm with oracle access to $\mathcal{A}(A)$ that runs in time $t(n,m)$, that:*

1. *Applies the algorithm from Theorem 3 to decompose the graph into $\phi(n)$-expanders $G[V_1], \ldots, G[V_k]$.*

2. *Computes a list of $\Omega(\phi(n))$-expanders $G_1, G_2, \ldots, G_\ell$ for some ℓ. Typically, $\ell = k$ and each G_i is obtained from $G[V_i]$ by a simple procedure (possibly $G_i = G[V_i]$).*

3. *Makes oracle calls to $\mathcal{A}$ on $G_1, \ldots, G_\ell$, and receives the solutions $\mathcal{A}(G_1), \ldots, \mathcal{A}(G_\ell)$.*

4. *Does some computation on the decomposed graph, and together with $\mathcal{A}(G_1), \ldots, \mathcal{A}(G_\ell)$, it computes the solution $\mathcal{A}(G)$.*

ED-WTERs are not captured by Definition 2 mainly because they do not produce only one expander instance G', but many: $G_1, G_2, \ldots, G_\ell$. In fact, many fine-grained reductions work like this. In Section 5, we provide a formal definition of fine-grained WTERs that captures both definitions, but for the sake of clarity we will use only Definitions 2 and 4 throughout the paper.

There are three disadvantages to ED-WTERs compared to Direct-WTERs. First, they tend to be more complicated and use heavier machinery. Second, unlike Direct-WTERs, they do not give a negative answer to Question 2; if someone discovers a breakthrough algorithm on expanders then (without a Direct-WTER) the way to solve the worst-case problem is to use expander decompositions. Third, and perhaps most importantly, ED-WTERs cannot produce *true* expanders in the sense that $\phi = \Omega(1)$ but are only limited to proving hardness for graphs with $\phi = (\log n)^{-O(1)}$. This is because of the log factors in the expander decompositions (that provably cannot be avoided [13, 49]) which means that given a graph with expansion $\phi = 1/\log n$ we cannot expect a decomposition algorithm to decompose it further into true expanders with $\phi = \Omega(1)$.

From the viewpoint of Question 1, a WTER that achieves $\phi = (\log n)^{-O(1)}$ (or even $\phi = \Omega(n^{-\varepsilon})$ for $\varepsilon \to 0$, as we do in this work) is partially satisfying. On the one hand, it does not show that the problem remains hard on true expanders; it is conceivable that a problem can be solved in $O(2^{1/\phi} \cdot n)$ time which would be linear when $\phi = \Omega(1)$ but inefficient on the class of graphs produced by an ED-WTER. On the other hand, worst-case graphs have $\phi = O(1/n)$ and graphs with larger ϕ are already expander-like. Indeed, essentially all breakthroughs using the expander-decompositions method were obtained by exploiting the structures of "expanders" with $\phi = \Omega(n^{-\varepsilon})$.

Our results are presented in Section 1.2, and they include WTERs of both types. Let us conclude this section with two remarks.

- Our framework is general with respect to the definition of an expander (e.g. vertex vs. edge expansion, with or without demands, etc.), and it could even be applied more broadly to any graph with strong structural properties (e.g. a Szemerédi *regular* graph). The specific reductions, however, are sensitive to the exact definition and require modifications to satisfy each definition.

- The reductions we design in this work are *randomized*. This does not affect the message from our results because the problems we consider appear to be just as hard for randomized algorithms as well.

■ **Table 1** In this table we present our results with respect to the current best upper bounds for the problems. The third column contains the hardness results on ϕ-expander. In particular, any major improvement to the running times in the third column on ϕ-expanders will result in a major improvement to the running times of the first column on worst-case graphs. For clarity, we hide any poly-logarithmic factors.

Problem	Best Upper Bound	ϕ-Expander Lower Bound	ϕ
Triangle Detection (dense graphs)	n^ω	n^ω	$\Omega(1)$
Triangle Detection (sparse graphs)	$m^{\frac{2\omega}{\omega+1}}$	$m^{\frac{2\omega}{\omega+1}}$	$\Omega(1)$
k-Clique	$n^{\omega k/3}$	$n^{\omega k/3}$	$\Omega(1)$
4-Cycle ($m = \Theta(n^{1.5})$)	n^2	$n^{4/3}$	$\Omega(1)$
4-Cycle ($m = \Theta(n^{1.5})$)	n^2	n^2	$\Omega(n^{-\varepsilon})$
Subgraph Isomorphism (without pendant vertices)	$n^{f(H)}$	$n^{f(H)/2}$	$\Omega(1)$
Subgraph Isomorphism (without pendant vertices)	$m^{g(H)}$	$m^{g(H)}$	$\Omega(1)$
Maximum Cardinality Matching (dense graphs)	n^ω	n^ω	$\Omega(1)$
Maximum Cardinality Matching (sparse graphs)	$m \cdot \sqrt{n}$	$m \cdot \sqrt{n}$	$\Omega(1)$
Vertex Cover (bounded degree)	$1.2125^n \cdot n^{O(1)}$	$1.2125^n / n^{O(1)}$	$\Omega(1)$
Vertex Cover (bounded degree, parameterized)	$1.2738^k \cdot n^{O(1)}$	$1.2738^k / n^{O(1)}$	$\Omega(1)$
Minimum Dominating Set	$1.4969^n \cdot n^{O(1)}$	$1.4969^{n/(1+\varepsilon)} \cdot n^{O(1)}$	$\Omega(1)$

1.2   Applications

Let us now present the collections of WTERs designed in this paper.

Triangle Detection

The longstanding upper bound for detecting a triangle in an n-node, m-edge graph is $O(\min\{n^\omega, m^{\frac{2\omega}{\omega+1}}\})$ [16] where $\omega < 2.38$ is the fast matrix multiplication exponent [14]. Even if $\omega = 2$, the upper bound would only be $m^{4/3}$ on sparse graphs. There are different hardness assumptions of different strengths, stating that Triangle cannot be solved much faster, e.g. in $m^{1+o(1)}$ or $m^{4/3-\varepsilon}$ time (see [9]). The next theorem shows that any polynomial speed-up in the expander-case will break these assumptions in worst-case graphs.

▶ **Theorem 5** (Direct-WTER for Triangle Detection). *Triangle Detection admits a Direct-WTER with blowup $(N, M) = (O(n), \tilde{O}(m))$.*

In fact, our result for Triangle Detection is merely a special case of a result for the more general problem of detecting k-clique in a graph, known simply as the k-Clique problem. The best known upper bound for k-Clique is $O(n^{\omega k/3})$ when k is divisible by 3. Otherwise the upper-bound is slightly larger. One of the strongest conjectures in fine-grained complexity (meaning least likely to be true) is that we cannot do better (see [1]). For clarity, let us assume that $k \geq 3$ is constant. Then we can state our result in the following way:

▶ **Theorem 6** (Direct-WTER for k-Clique). *For all constants $k \geq 3$, there is a Direct-WTER with blowup $(N, M) = (O(n), \tilde{O}(m))$.*

For completeness, we remark that if $k = \omega(1)$, then the blowup would be $(N, M) = (O(nk), \tilde{O}(mk^2))$ and the conductance $\phi = \Omega(1/k^2)$.

Another problem that is closely related to Triangle Detection is 4-Cycle, for which achieving tight bounds has been notoriously difficult. The longstanding upper bound is $O(\min\{n^2, m^{4/3}\})$ [16]. Back in 1999, it had been conjectured that a subquadratic algorithm is not possible [56], but only very recently a super-linear lower bound of $\Omega(m^{1.11})$ (assuming the hardness of Triangle Detection) was obtained [2]. We remark that the hard cases of 4-Cycle are in the regime of $m = O(n^{1.5})$, as a simple counting argument shows that any graph with at least $n^{1.5}$ edges contains a 4-Cycle.

For 4-Cycle we design both a Direct-WTER and also an ED-WTER. Nonetheless, both WTERs are relevant because each has a different drawback.

▶ **Theorem 7** (Direct-WTER for 4-Cycle). *There is a Direct-WTER to 4-Cycle with blowup* $(N, M) = (\tilde{O}(m), \tilde{O}(m))$.

The drawback of the Direct-WTER is a blowup in the number of vertices ($\tilde{O}(m)$ instead of $O(n)$), significantly weakening the result. In particular, worst-case graphs with $m = \Theta(n^{1.5})$ edges are transformed to expanders with $N = \Omega(n^{1.5})$ vertices. Therefore, in order to gain a speed-up of $\tilde{O}(n^{2-\varepsilon})$ on worst-case graphs, one needs a major speed-up of $\tilde{O}(N^{\frac{4}{3}(1-\varepsilon/2)}) = \tilde{O}(n^{2-\varepsilon})$ on $\Omega(1)$-expanders. In search of WTERs that avoid this major drawback, we manage to obtain the next ED-WTER:

▶ **Theorem 8** (ED-WTER for 4-Cycle). *For any $\varepsilon > 0$, there is a ED-WTER for 4-Cycle that runs in time $\tilde{O}(n^{\varepsilon}m + n^{2-\varepsilon/2})$, with conductance $\phi = n^{-\varepsilon}$.*

This WTER suffers from the disadvantages of ED-WTERs discussed in Section 1.1. However, it provides much better hardness results, in the sense that any $\tilde{O}(N^{2-\delta})$ speed-up on $\tilde{\Omega}(n^{-\varepsilon})$-expanders will imply a $\tilde{O}(n^{2-\delta})$ speed-up on worst-case graphs. An interesting open question to ask now is whether 4-Cycle does become easy on $\Omega(1)$-expanders while it remains hard on $\Omega(n^{-\varepsilon})$-expanders.

We now conclude this line of problems with a general result for the *Subgraph Isomorphism* problem.

Subgraph Isomorphism

The foregoing problems were special cases of the Subgraph Isomorphism problem that asks to detect a k-node pattern graph H inside a host graph G (not necessarily as an induced subgraph). The complexity of solving Subgraph Isomorphism can vary wildly depending on H. For example, while the best algorithms for k-Clique run in time $O(n^{\omega k/3})$, there are much more efficient algorithms for problems such as k-Cycle and k-Path, that run in $O(n^{\omega})$ [15]. Thus, for every pattern H we define a constant $f(H)$ (respectively, $g(H)$) that is the minimum number such that Subgraph Isomorphism (with respect to H) can be solved in $O(n^{f(H)+o(1)})$ time (respectively, $O(m^{g(H)+o(1)})$ time in sparse graphs). We will focus on a variant of Subgraph Isomorphism in which H does not contain pendant vertices (vertices whose degree is one). We note that this variant is still general enough to capture the hard and interesting cases such as k-Clique and k-Cycle. Our result for this problem is stated by the next theorem:

▶ **Theorem 9** (Direct-WTER for Subgraph Isomorphism without pendant vertices). *There is a Direct-WTER for Subgraph Isomorphism for patterns that do not contain pendant vertices. Assuming that k is constant, the blowup is $(N, M) = (\tilde{O}(m), \tilde{O}(m))$.*

We should note here that the Direct-WTER for 4-Cycle is, in fact, simply an application of the Direct-WTER for Subgraph Isomorphism. For completeness, we remark that for non-constant k the blowup is $(N, M) = (\tilde{O}(mk), \tilde{O}(mk))$ and the conductance is $\Omega(1/k)$. As we've noted in the case of 4-Cycle, there is a significant drawback that comes from the blowup to the number of vertices, and that weakens the result significantly.

Maximum Cardinality Matching

Next, we turn our attention to the fundamental problem that asks to find a set $M \subseteq E(G)$ of maximum cardinality, such that no two edges in M intersect. In a recent breakthrough, an almost-linear time algorithm for Max-flow [25] resulted in a $\tilde{O}(m)$ time algorithm for Maximum Matching in bipartite graphs. However, the best-known upper bounds for general graphs are still far from linear. For sparse graphs, there is a longstanding $O(m\sqrt{n})$ upper bound by Micali and Vazirani [42]. For dense graphs, we can do slightly better using an $\tilde{O}(n^{\omega})$ algorithm by Mucha and Sankowski [44]. The Micali and Vazirani algorithm is part of a long list of algorithms that work by finding *augmenting paths* that improve a non-maximum matching iteratively. A well-known fact says that a matching is non-maximum if and only if there exists an augmenting path in the graph.

An interesting line of work focused on the case where G is a random graph drawn from the distribution $G(n, p)$. Motwani [43] showed that if p is at least $\ln n / (n - 1)$, then with high probability, every non-maximum matching admits a short augmenting path of length $O(\log n)$, and therefore the Micali and Vazirani algorithm terminates in $\tilde{O}(m)$ time. Using similar arguments, Bast et al. [20] provided a simpler proof that the above holds for any $p \geq 33/n$. At the heart of their proofs, both papers rely on various properties of random graphs, most notably on vertex-expansion of sufficiently large sets of vertices.

A question that comes to mind is whether expansion (either vertex-expansion or other forms of expansion) alone is sufficient to make the problem easy, or if other properties of random graphs are necessary. If expansion alone is enough, perhaps we can use the expander decomposition method to improve the state-of-the-art on worst-case graphs. We note that the distinction between vertex-expansion and conductance (which is the notion used in the expander decomposition) does not seem to be very important for their analysis, as they rely on the fact that after expanding two subgraphs of G for $O(\log n)$ steps, the two must meet. This holds even in large-conductance graphs.

Alas, the next Direct-WTER shows that the expander decomposition method is useless against this problem.

▶ **Theorem 10** (Direct-WTER for Maximum Cardinality Matching)**.** *There is a Direct-WTER for Maximum Cardinality Matching with blowup* $(N, M) = (O(n), \tilde{O}(m))$. [4]

To answer the question about the properties that make random graphs easy for matching algorithms, we look closely at the analysis of Motwani [43] and of Bast et al. [20]. We find that they rely on the fact that with high probability, every pair of large, disjoint sets of vertices are connected by at least one edge.[5] In our reduction, we unavoidably introduce linear-sized sets of vertices that don't have an edge between them. Indeed, the graphs that are output by the reduction may not admit short augmenting paths if the original graph did not admit such. In particular, one cannot use the reduction as a way to speed-up algorithms on worst-case graphs. It is an interesting open question whether such approach, augmenting the graph to be random-like and "shortcut" augmenting paths, can be used to improve the running times on worst-case graphs.

We now move on to NP-hard problems, where we are interested in the exact (fine-grained) exponential time complexity.

---

[4] To further emphasize that the distinction between vertex-expansion and conductance is not very important, we note that it is possible to show that the graph output by this WTER is also a vertex-expander.

[5] See [43, Lemma 6.1] and [20, Lemma 3].

Minimum Vertex Cover

In the Minimum Vertex Cover problem, we wish to find a minimum-sized set of vertices such that every edge is incident to this set. This NP-Hard problem has received a lot of attention over the years, resulting in relatively fast algorithms (compared to other NP-hard problems). After a long line of works (e.g. [21, 31, 48]), the upper bound for Minimum Vertex Cover stands at $1.2125^n \cdot n^{O(1)}$ [22] and it is unclear if it can be improved further, e.g. to $1.001^n \cdot n^{O(1)}$. The problem has also received a lot of attention from the *parameterized complexity* community, and the current best bound for deciding if there is a vertex cover of size $\leq k$ is $1.2738^k \cdot n^{O(1)}$ [24]. In the next theorem we present WTERs that address both the exact complexity and the parameterized complexity of this problem on expanders.

▶ **Theorem 11** (Direct-WTER for Minimum Vertex Cover). *There is a Direct-WTER for Minimum Vertex Cover with blowup $(N, M) = (n + O(\log n + \Delta(G)), \tilde{O}(m))$, where $\Delta(G)$ is the maximum degree in G. Moreover, the size of minimum vertex cover in the output G' is $K = k + 5 \max\{\Delta(G), C \log n\}$ for some fixed constant C, where k is the size of the minimum vertex cover in G.*

At first sight, this WTER may appear unattractive because of the $\Delta(G)$ blowup in the number of vertices, which could be as large as $\Omega(n)$. However, by observing that the Vertex Cover problem becomes *easier* (in the sense that the 1.2125^n bound can be broken) in graphs with large degrees [22], we conclude that the hard cases have $\Delta(G) = O(1)$. Therefore, assuming that $\Delta(G) = O(1)$, the blowup is actually $(N, M) = (n + O(\log n), \tilde{O}(m))$ which achieves the so-called platinum standard. Furthermore, since the size of the minimum vertex cover in G' is $K = k + 5 \max\{\Delta(G), C \log n\} = k + O(\log n)$, then any speed-up of $(1.2738 - \epsilon)^K \cdot N^{O(1)}$ on $\Omega(1)$-expanders will result in a speed-up of $(1.2738 - \epsilon)^{k + O(\log n)} \cdot n^{O(1)} = (1.2738 - \epsilon)^k \cdot n^{O(1)}$ on worst-case graphs. We see this as a proof of concept that our framework can extend even beyond the traditional notions of complexity and have implications also in parameterized complexity.

Finally, let us present another fundamental NP-Hard problem.

Minimum Dominating Set

In the Minimum Dominating Set problem, the goal is to find a set of vertices of minimum size, such that every vertex in the graph is either a neighbor of some vertex in the set or is in the set itself. The Minimum Dominating Set problem can be solved trivially in $2^n \cdot n^{O(1)}$ time. In 2004, a first $c^n \cdot n^{O(1)}$ algorithm with $c < 2$ was discovered by Fomin [32], and since then, the complexity was further improved to $1.4969^n \cdot n^{O(1)}$ [52]. Unlike Minimum Vertex Cover, the Minimum Dominating Set problem is not fixed-parameter tractable, meaning that it is unlikely to be solved in $f(k) \cdot n^{O(1)}$ time, and in that sense it is considered harder.

For Minimum Dominating Set we show a Direct-WTER with blowup $(N, M) = ((1 + \varepsilon)n, \tilde{O}(m))$ for any constant $\varepsilon > 0$, and constant conductance that depends on ε. Observe that for any significant improvement to the exponent in the running time on $o(1)$-expanders, we can choose ε to be small enough to get an improvement on worst-case graphs. Interestingly, this WTER requires a slightly different approach that is a combination of the techniques we used for k-Clique and Maximum Cardinality Matching, together with a subsampling trick. We find it interesting and see it as an indicator that our techniques can be applied to more open questions.

▶ **Theorem 12** (Direct-WTER for Minimum Dominating Set). *There is a Direct-WTER for Minimum Dominating Set with blowup $(N, M) = (n + \varepsilon n, \tilde{O}(m))$ and conductance $\phi(G') \geq \Omega(1)$, for any constant $\varepsilon > 0$.*

Our results only reveal the tip of the iceberg. There is a plethora of problems for which Questions 1 and 2 may be relevant, from open questions in fine-grained complexity to NP-hard problems. We are hopeful that our framework will help in understanding the role that expanders have in algorithms and beyond.

1.3   Related Work

An interesting line of research in fine-grained complexity it to show conditional lower bounds for "realistic" input classes such as planar graphs, bounded treewidth graphs, and so on. There has been some progress on designing reductions that apply to such restricted families as well (see e.g. [3, 4, 10, 30]). Perhaps the next extension to this line of work should be random-like graphs, e.g. expanders? Indeed, a recent independent paper is concerned with the complexity of various problems on families of dynamic graphs, that also include expanders [35]. Our work is not motivated by these kinds of questions, but our techniques may have implications for that line of work as well.

Another area of research, that somewhat resembles ours, is the research on worst-case to average-case reductions. These reductions show that a problem is as easy on worst-case inputs as it is on *random* instances (e.g., taken from the uniform distribution). In recent years, much effort has gone into proving worst-case to average-case reductions for problems in P (see [17, 18, 29, 34]). However, it has only been accomplished for counting or algebraic problems but not decision problems, while we are interested in graph decision problems on pseudo-random instances. In fact, trying to show any worst-case to average-case reductions on these problems might as well fail, since they (e.g., Triangle Detection) typically do become easier on random graphs.

2   Preliminaries

Let $G = (V, E)$. For every $v \in V(G)$ define the neighborhood of v to be $N(v) := \{u \in V \mid \{u, v\} \in E(G)\}$. The degree of a vertex $v \in V(G)$ is denoted by $\deg_G(v) := |N(v)|$, and for $u \notin V(G)$ define $\deg_G(u) := 0$. We denote the maximum degree among the vertices of G by $\Delta(G)$. For every $S \subseteq V$, denote the *volume* of S by $vol_G(S) = \sum_{v \in S} \deg_G(v)$. We say that an edge $\{u, v\}$ is an *internal* edge is both u and v are in S. Denote by $E_G(S, S')$ the set of edges between disjoint sets of vertices $S, S' \subseteq V(G)$ and by $e_G(S, S') := |E_G(S, S')|$. Denote by $\delta_G(S) := e_G(S, \bar{S})$ and note that $\delta(S) = \delta(\bar{S})$. The *conductance* of S is defined as $\phi_G(S) := \frac{\delta_G(S)}{\min\{vol_G(S), vol_G(\bar{S})\}}$ and for $S = \emptyset$ we define $\phi_G(S) := 1$. The conductance of G is defined as $\phi(G) = \min_{S \subseteq V} \phi_G(S)$. The *expansion* of a cut or a graph is simply its conductance. We omit the subscripts when they are clear from the context. We denote by $G[S]$ the induced subgraph of G on set of vertices S. Throughout that paper, we use the Chernoff bound, stated as:

▶ **Fact 13** (Chernoff Bound). *Let X be the sum of n independent indicators, each taking the value 1 with probability $0 < p < 1$, and μ be the expectation of X. Then for any $0 \leq \delta \leq 1$, $\mathbb{P}[|X - \mu| \geq \delta\mu] \leq \exp(-\delta^2\mu/3)$.*

3   Technical Overview

Recall that our goal when designing a Direct-WTER is to take a worst-case graph G and turn it into an expander G' without changing the answer to the problem uncontrollably. Consider the Triangle Detection problem as an example for now. The goal is to add edges to a graph in order to make it an expander while not introducing a triangle to the graph if the original graph was triangle-free.

Let us start by recalling the intuition that an expander graph should be *mixing* [33] in the sense that a short random walk from any vertex is equally likely to reach any other vertex. Keeping this in mind will help the reader follow the informal arguments in this section, but it is not necessary for understanding our actual proofs that only work with the definition of expander in Definition 1.

The first idea that comes to mind is to take an expander graph X on $n := |V(G)|$ nodes (e.g. a random sparse graph) and *add it* to G, so that $G' = G \cup X$. There are several issues with this idea. First, the resulting graph will not be an expander if X is just an arbitrary expander. For example, if G is dense and X is sparse, then the new edges of X will not have much effect on random walks in the combined graph G'. This issue can be resolved by using an expander X in which each vertex v has the same degree that it has in G; a random graph with specified degrees does the job. The second issue is that the answer to the problem can change arbitrarily when taking the union with another graph X; e.g. an edge $\{u, v\} \in E(G)$ may become a triangle because u, v have a common neighbor in the newly introduced X. For some problems, e.g., the Subgraph Isomorphism problem for pattern H without degree-1 vertices, this issue can be overcome by *subdividing* the edges of X into paths of length $|H| + 1$ before adding them to G. This makes the new edges useless for forming any copy of H (or a triangle in particular) because any subgraph of size $|H|$ (or 3) that uses a new edge must have a leaf node of degree 1. This does result in a Direct-WTER, but it is not as fine-grained as we would like: the number of nodes in G' becomes $O(m)$ where $m := |E(G)|$ because of the subdivided edges. Our actual reductions use additional ideas and are actually quite different, but the intuition behind them is similar.

Instead, the basic idea in our reductions is to add a small *expansion layer* U with $O(n)$ nodes and connect each node in $V(G)$ with random edges to U. That is, we add a random bipartite graph whose one side is $V(G)$, and the other side is U. If a node $v \in V(G)$ has degree d in G then we add d random edges to U. Then, a random walk starting at v will go with probability $1/2$ to the expansion layer U, and the next step would send it back to an essentially random vertex $w \in V(G)$; thus simulating a "random edge" from v to w. We prove that adding such an expansion layer to any graph that has a sufficiently large (logarithmic) minimum degree makes it an expander. We introduce some gadgets to artificially increase the degrees in our problems.

The advantage of adding an expansion layer over simply taking the union with a random graph is that it makes it easier to control the change in the solution of the problem. This is accomplished with different gadgetry for each problem and with a different argument for why the gadgets do not harm the conductance of the graph by too much. For example, in the WTER for maximum matching, we add a unique "twin" vertex to each node in the expansion layer U and connect them by an edge. The maximum matching is forced to match every node in U to its twin, thus making the newly added edges between $V(G)$ and U unusable in a maximum matching. The WTER for Vertex Cover uses a similar idea. A different idea is used for Triangle (and k-Clique), where we connect the expansion layer U only to a (sufficiently large) independent set V_1 in G. Consequently, the newly added edges cannot form any triangle. The WTER for Minimum Dominating Set combines ideas from both WTERs for Triangle Detection and Vertex Cover, together with a sampling trick. Notably, for the 4-Cycle problem, we could not avoid the subdivision trick so we complement our result with an ED-WTER that avoids the blowup.

Designing ED-WTERs is fundamentally different. To reduce 4-Cycle into expanders, we use the expander decomposition to partition G into induced subgraphs with conductance $\phi = n^{-\varepsilon}$ and a small set of outer edges. Then, assuming that we can check (in subquadratic

time) whether there is a 4-cycle inside each subgraph, the problem boils down to finding a 4-cycle that uses one of the outer edges. Hence, we provide an algorithm that finds a 4-cycle that uses one of the outer edges or outputs that there is no such cycle. Using the fact that the number of outer edges is small (namely, $O(n^{1.5-\varepsilon})$), this algorithm runs in subquadratic time.

4   The Reductions

Our main building block is a randomized algorithm that takes as an input an n-vertex graph G where for every $v \in V(G)$, $\deg_G(v) \geq C \log n$ for some large enough constant C, and outputs a graph G' such that: (1) $|E(G')| = O(|E(G)|)$ and $|V(G')| = O(n)$, and (2) $\phi(G') = \Omega(1)$. We note that this algorithm applies even to graphs with self-loops. We later modify and use this algorithm in different ways to show all the above results.

4.1   The Expansion Layer Construction

Construct an expander graph G' from G as follows.
1. Add to G an *expansion layer* U that consists of $5n$ vertices.
2. For every vertex $v \in V(G)$, sample without replacement $\deg_G(v)$ vertices from the expansion layer U and add all edges between them and v. Thus, the total number of vertices and edges added in this step is $O(n)$ and $O(|E(G)|)$, respectively.
3. Increase the degrees of the vertices of U deterministically in the following way. Partition each of $V(G)$ and U into $\frac{10n}{C \log n}$ parts of equal size (i.e, partition $V(G)$ into parts of size $\frac{C}{10} \log n$ and partition U into parts of size $\frac{C}{2} \log n$). Then, take an arbitrary perfect matching between the parts in the partitions of $V(G)$ and U, and add a bi-clique between every matched pair of parts (i.e. add all possible edges between every matched pair). Thus, the total number of edges added in this step is $O(\frac{n}{\log n} \cdot \log^2 n) = O(n \log n)$. Note that after this step, the degrees of all vertices in U are at least $\frac{C}{10} \log n$ and the degrees of all vertices in $V(G)$ are at least $\frac{3}{2} \cdot C \log n$ (as we assume that $\deg_G(v) \geq C \log n$ and then add an additional $\frac{C}{2} \log n$ edges).

See Figure 1. Observe that $|E(G')| = O(|E(G)|)$, $|V(G')| = O(n)$ and that the reduction is randomized and runs in linear time.
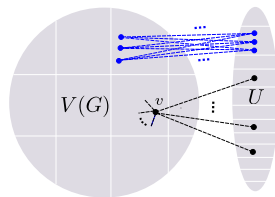
Figure 1 The basic construction of the algorithm. The blue edges represent the deterministically added bi-cliques. The black edges represent the edges of G and the new random edges. For each vertex v we sample $\deg_G(v)$ neighbors in U and add edges between them.

▶ **Theorem 14.** *The conductance of G' is $\phi(G') \geq 0.01$ with probability at least $1 - O(\frac{1}{n})$.*

Before proving this result, let us introduce some notation and give an overview. Throughout the section, we omit the subscripts from $\deg_{G'}(\cdot), vol_{G'}(\cdot), E_{G'}(\cdot, \cdot), e_{G'}(\cdot, \cdot), \delta_{G'}(\cdot)$ and $\phi_{G'}(\cdot)$. For any cut $S \subseteq V(G')$, we denote by S_V and S_U the parts $S \cap V(G)$ and $S \cap U$, respectively.

To prove the theorem, we will focus on the complement probability. That is, we show that $\mathbb{P}[\phi(G') < 0.01] = O(\frac{1}{n})$. We will apply the union bound over all possible cuts. More precisely, we show that for any cut $S \subseteq V(G')$, the *failure probability* $\mathbb{P}[\phi(S) < 0.01]$ is sufficiently small so that by summing the failure probabilities of all possible cuts we still get a value much smaller than 1.

For any cut $(S, \bar{S})$, assume w.l.o.g. that S is the part for which the inequality $|S_U| \leq |U|/2$ holds (as it must hold either for S or $\bar{S}$). Observe that in order to show that $\mathbb{P}[\phi(S) < 0.01]$ is small it suffices to show that $\mathbb{P}[\frac{\delta(S)}{vol(S)} < 0.01]$ is small, since:

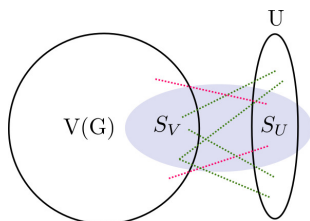$$\phi(S) = \frac{\delta(S)}{\min\{vol(S), vol(\bar{S})\}} = \max\left\{\frac{\delta(S)}{vol(S)}, \frac{\delta(S)}{vol(\bar{S})}\right\}$$

and therefore $\mathbb{P}[\phi(S) < 0.01] \leq \mathbb{P}[\frac{\delta(S)}{vol(S)} < 0.01]$.

The main lemma is the following one:

▶ **Lemma 15.** *For any cut $S \subseteq V(G')$ such that $|S_U| \leq |U|/2$, we have:*

$$\mathbb{P}\left[\frac{\delta(S)}{vol(S)} < 0.01\right] \leq n^{-\frac{C}{800} \cdot |S|}.$$

■ **Figure 2** A depiction of a graph G with expansion layer U. The gray area denotes a set S. To prove Lemma 15, we lower bound the number of pink edges and the number of green edges, and show that at least one of them is proportional to $vol(S)$.

We prove the lemma in the full version of the paper. See Figure 2 for a short explanation about the proof.

Proof of Theorem 14. Fix C to be large enough so that the failure probability in Lemma 15 is less than $n^{-2|S|}$, for any $S \subseteq V(G')$ with $|S_U| \leq |U|/2$ ($C = 1600$ will do). Since the number of different cuts that consist of k vertices is $O(n^k)$, we can apply the union-bound over all cuts such that $|S_U| \leq |U|/2$. In more detail, we sum over $k = 1, 2, \ldots, 3.5n$ (any cut larger than that must have $|S_U| > |U|/2$) the number of cuts of size k times the failure probability of such cuts, and it is easy to see that the sum is bounded by $O(1/n)$. ◀

Let us now explain why we can assume that the degrees in G are at least $C \log n$. If for some $v \in V(G)$ we have $deg_G(v) < C \log n$, then we can add $C \log n - deg_G(v)$ self-loops to v in a preprocessing step. After constructing the expander G' we can remove all those self-loops from G'. Notice that by removing self-loops from an expander we only increase the expansion of the graph.

4.2 Direct-WTERs for Maximum Cardinality Matching and Minimum Vertex Cover

We are now ready to show our first WTERs for the problems of computing the Maximum Cardinality Matching (abbreviated as MCM) and Minimum Vertex Cover (abbreviated as MVC) of a graph G. We use $MCM(G)$ and $MVC(G)$ to denote their values, respectively.

Naïvely, we could try to simply apply Algorithm 4.1 on G, resulting in an expander graph G'. However, this approach will fail because there is no obvious way to compute $MCM(G)$ $(MVC(G))$ from $MCM(G')$ $(MVC(G'))$. Therefore, we employ a simple trick that helps us control the values of the MCM and MVC. The trick is to add *twin vertices* to a selected subset of the vertices of the graph. Formally, we define a twin of a vertex u to be a degree-1 vertex u' that is adjacent to u (that is, u' is a pendant vertex). By the next fact, we can add twins to the graph to force a selected subset of edges or vertices to be inside the MCM or MVC, respectively.

▶ **Fact 16.** *If a graph contains a twin vertex u' of u, then there exists an MCM that contains the edge $\{u, u'\}$, and there exists an MVC that contains the vertex u.*

We note that we need to be careful that by adding twins to an expander graph, we do not reduce the expansion by much. Formally, we prove this in the next lemma:

▶ **Lemma 17.** *Let H be a connected graph and let $A \subseteq V(H)$ be arbitrary. Let H' be a graph constructed from H by adding a twin vertex $u' \in V(H')$ and an edge $\{u, u'\} \in E(H')$ for every $u \in A$. Then, $\phi(H') \geq \phi(H)/2$.*

Proof. Let $S \subseteq V(H')$. Observe that $\delta_{H'}(S) \geq \delta_H(S)$. Additionally, since every vertex in S has at most one twin then $\deg_{H'}(v) \leq \deg_H(v) + 1$ for every $v \in S$, and therefore $vol_{H'}(S) \leq |S| + vol_H(S)$. Since H is connected, then $|S| \leq vol_H(S)$ and therefore $vol_{H'}(S) \leq |S| + vol_G(S) \leq 2vol_H(S)$. Thus, since the above holds for $\bar{S}$ as well, we get:

$$\phi_{H'}(S) = \frac{\delta_{H'}(S)}{\min\{vol_{H'}(S), vol_{H'}(\bar{S})\}} \geq \frac{\delta_H(S)}{2\min\{vol_H(S), vol_H(\bar{S})\}} = \frac{\phi_H(S)}{2},$$

and therefore $\phi(H') \geq \frac{\phi(H)}{2}$.   ◀

We are now ready to show a Direct-WTER for MCM.

Proof of Theorem 10 (Direct-WTER for MCM). Apply Algorithm 4.1 on G to get an $\Omega(1)$-expander $\hat{G}$. Then add a twin vertex u' for every $u \in U$, where U is the expansion layer in $\hat{G}$. Denote the resulting graph by G'. See Figure 3. Note that the number of twins added is $|U| = 5n$ and that the number of added edges is $O(m + n \log n)$. Hence, the blowup is $(N, M) = (O(n), \tilde{O}(m))$. Moreover, by Lemma 17 we have $\phi(G') = \Omega(1)$.
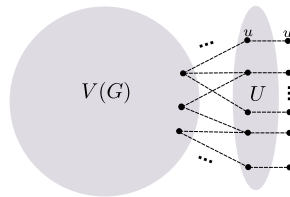
Figure 3 The graph G' obtained from our construction for MCM and MVC.

Next, we prove that computing $MCM(G)$ from $MCM(G')$ is trivial.

▷ **Claim 18.** The size of the maximum cardinality matching in G' is $MCM(G') = MCM(G) + 5n$.

Proof. Denote by $X \subseteq E(G')$ the set of all edges incident to twins. By Fact 16, there exists in G' a maximum matching M that contains X. Notice that M does not contain any U-to-$V(G)$ edges and therefore $|M| = |M \cap E(G)| + |X| = |M \cap E(G)| + 5n$. Moreover, $M \cap E(G)$ must be a maximum matching of G because if there was a larger matching we could take it with X to obtain a larger matching for G'. Thus, $|M| = |M \cap E(G)| + 5n = MCM(G) + 5n$.   ◁

We've shown a transformation from G to a $\Omega(1)$-expander G' with blowup $(N, M) = (O(n), \tilde{O}(m))$, such that $MCM(G)$ can be computed efficiently from $MCM(G')$.   ◄

Next, we show a Direct-WTER for MVC. Since the best upper bounds for MVC are exponential then we aim to keep the blowup in the number of vertices small. To this end, we slightly modify Algorithm 4.1 to avoid adding an expansion layer of size $5n$.

Modifications to Algorithm 4.1

Instead of adding an expansion layer of size $5n$, we add an expansion layer of size $5 \max\{\Delta(G), C \log n\}$. Notice that U is sufficiently large to allow every vertex in $V(G)$ to sample $\max\{\deg_G(v), C \log n\}$ vertices from U without replacement. We claim that Theorem 14 holds under this modification. To see why, observe that throughout the construction and analysis we only needed U to be large enough so that: (1) every vertex $v \in V(G)$ can sample without replacement $\max\{\deg(v), C \log n\}$ vertices from U, and (2) $|U| - |S_U| - \max\{\deg(v), C \log n\} \geq 1.5|U|$ which indeed holds as long as $|S_U| \leq 2.5|U|$.

Proof of Theorem 11 (Direct-WTER for MVC). Apply the modified Algorithm 4.1 on G and add a twin vertex u' for every $u \in U$. Denote the resulting graph by G'. The main difference between the previous construction and this one is that we now have $|U| = 5 \max\{\Delta(G), C \log n\}$ and therefore, the blowup in G' is $(N, M) = (n + O(\log n + \Delta(G)), \tilde{O}(m))$. Moreover,

▷ **Claim 19.** The size of the minimum vertex cover in G' is:

$$MVC(G') = MVC(G) + 5 \max\{\Delta(G), C \log n\}$$

The proof is similar to the proof for MCM and thus omitted. Hence, we've shown a transformation from G to a $\Omega(1)$-expander G' with blowup $(N, M) = (n + O(\log n + \Delta(G)), \tilde{O}(m))$, such that $MVC(G)$ can be computed efficiently from $MVC(G')$.   ◄

4.3 Direct-WTER for k-Clique

The construction for k-Clique can be described roughly in the following way. We apply a simple transformation that makes the graph k-partite, and such that every vertex has the same number of neighbors in every part (except the one it belongs to). This property is crucial as it implies that any sparse cut in the graph must contain vertices from all parts. We apply Algorithm 4.1 to only one of the parts and obtain a graph G' (we later describe this step in more detail). Since any sparse cut in the k-partite graph must contain vertices from the part that has an expansion layer, there are no sparse cuts in G'. Specifically, the graph G' has conductance $\Omega(1/k^2)$ and the number of k-Cliques in G' is $k!$ times the number of k-cliques in G.

Proof of Theorem 6 (Direct-WTER for k-Clique). We would like to assume that G is k-partite with parts $V_1, V_2, \ldots, V_k$, such for every vertex $v \in V_i$, v has the same number of neighbors in every part V_j for $j \neq i$. We can take k copies $V_1, \ldots, V_k$ of $V(G)$ such

that $V_i = \{v_i \mid v \in V(G)\}$. Now, for every edge $\{u,v\} \in E(G)$ we create $\binom{k}{2}$ copies: $\{\{u_i, v_j\} \mid i \neq j\}$. Observe that the resulting graph is k-partite, the number of vertices in this graph is kn, and the number of edges is $k \cdot m^2$. Most importantly, the number of k-cliques in this graph is $k!$ times the number of k-cliques in the original graph. For clarity, we henceforth assume that G has this k-partite form from the beginning, but we state our results as if it had not (e.g., we state that the blowup is $(O(nk), \tilde{O}(mk^2))$).

We wish to add an expansion layer to G without introducing any k-clique. Hence, we cannot naïvely apply Algorithm 4.1 on G because, with some probability, some vertices from the expansion layer will participate in a k-clique. Instead, we add an expansion layer that is adjacent only to V_1, but with respect to the degrees of the vertices of V_1 in G. That is, we apply Algorithm 4.1 in the following way: for every $v \in V_1$, v will sample $\max\{\deg_G(v), C \log n\}$ edges to the expansion layer U. Denote the resulting graph by G'. See Figure 4.
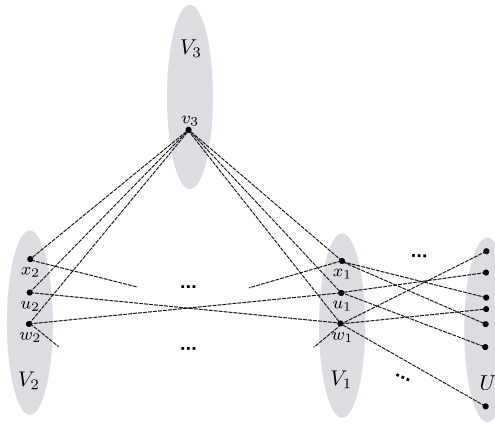
Figure 4 The construction of k-Clique for $k = 3$. In this example, we can see that v_3 participates in the triangles u_1, w_2, v_3 and w_1, u_2, v_3.

▶ **Lemma 20.** *With probability $1 - O(\frac{1}{n})$, the conductance of G' is $\phi(G') = \Omega(1/k^2)$.*

The proof is present in the full version of the paper.   ◀

4.4   Direct-WTER for Subgraph Isomorphism

Fix H to be a k-node pattern graph without pendant vertices.

Proof of Theorem 9. Apply Algorithm 4.1 on G and then subdivide every U-to-$V(G)$ edge k times. Denote by G' be the resulting graph.

▷ **Claim 21.** H is a subgraph of G if and only if it is a subgraph of G'.

Proof. Since H does not contain pendant vertices and $|V(H)| < k + 1$, then any copy of H in G' does not use an edge that was added in the subdivision step. Therefore it is also a copy of H in G.   ◁

▷ **Claim 22.**   The expansion of G' is $\phi(G') = \Omega(\frac{1}{k}) = \Omega(1)$.

Proof. For any cut $S \subseteq V(G')$ denote by $\hat{S}$ the corresponding cut in $\hat{G}$, where $\hat{G}$ is the graph obtained from G before the subdivision. More precisely, $\hat{S}$ consists of $S \cap V(G)$ and $S \cap U$. By Theorem 14, we have $\phi_{\hat{G}}(S) = \Omega(1)$. Now, observe that $\delta(S) \geq \delta_{\hat{G}}(\hat{S})$ and that $vol(S) \leq 2k \cdot vol_{\hat{G}}(S)$. Therefore:

$$\phi(S) \geq \frac{\delta_{\hat{G}}(\hat{S})}{2k \cdot \min\{vol_{\hat{G}}(\hat{S}), vol_{\hat{G}}(\hat{S})\}} = \Omega\left(\frac{1}{k}\right) \qquad \lhd$$

◀

4.5 ED-WTER for 4-Cycle

In this WTER we use the expander decomposition from Theorem 3 to obtain a WTER for 4-Cycle with conductance $n^{-\varepsilon}$, for any constant $0 < \varepsilon < 0.5$, and running time $\tilde{O}(n^{2-0.5\varepsilon} + n^{1.5+\varepsilon})$. Observe that the number of outer edges in such decomposition is $\tilde{O}(n^{-\varepsilon}m) = \tilde{O}(n^{1.5-\varepsilon})$. We will call the vertices that are incident to outer edges *portals*. We also assume that we have access to an oracle for 4-Cycle, such that when given any $n^{-\varepsilon}$-expander X it answers "YES" if and only if X contains a 4-Cycle. Note that if a single call to the oracle takes $O(n_i^{2-\delta})$ time for some $\delta > 0$, where n_i is the number of nodes in the ith expander and $\sum_i n_i = n$, then the total time for all calls will also be $\sum_i n_i^{2-\delta} = O(n^{2-\delta})$; therefore, this WTER proves that any polynomial speedup on expanders implies a polynomial speedup in worst-case graphs.

Proof of Theorem 8 (ED-WTER for 4-Cycle). Decomopse G into expanders by using the expander decomposition of Theorem 3 with parameter $\phi = n^{-\varepsilon}$. Next, query the oracle on every expander in the decomposition, and if at some point the oracle answers "YES" we are done. Otherwise, we need to check if there is a 4-Cycle that uses one of the $\tilde{O}(n^{1.5-\varepsilon})$ outer edges.

To this end, we use a "high degree - low degree" trick. First, we check whether there are 4-cycles that contain any high-degree vertex in the graph (not necessarily a portal). We define a high-degree vertex to be a vertex of degree at least $n^{0.5(1+\varepsilon)}$. Denote by $A \subseteq V(G)$ the set of all high-degree vertices in the graph. A simple counting argument shows that $|A| = O(n^{1.5}/n^{0.5(1+\varepsilon)}) = O(n^{1-0.5\varepsilon})$. Now, for every $v \in A$ we can check in $O(n)$ time whether v participates in any 4-Cycle in the following way: we mark every neighbor-of-a-neighbor of v, until we either find a collision between two marked nodes (that is, a 4-Cycle) or until we marked all neighbors-of-a-neighbor. Note that in any case, we spend at most $O(n)$ time because after marking n nodes, we must find a collision. Hence, we can check in $O(|A| \cdot n) = O(n^{2-0.5\varepsilon})$ time whether there is a vertex in A that participates in a 4-Cycle.

Next, we need to check if there is a 4-Cycle that consists only of low-degree vertices and at least one outer edge. We denote by P the set of all portals whose degree is less than $n^{0.5(1+\varepsilon)}$. For every vertex $v \in P$, we mark the endpoints of every 2-path whose center is v, and that consists of at least one outer edge. If we denote by $b(v)$ the number of outer edges incident to v, then the number of such paths (and the time it takes to mark their endpoints) is at most :

$$\sum_{v \in B} \deg(v) \cdot b(v) \leq n^{0.5(1+\varepsilon)} \sum_{v \in B} b(v) = O(n^{0.5(1+\varepsilon)} \cdot n^{1.5-\varepsilon}) = O(n^{2-0.5\varepsilon}).$$

If we've encountered a collision between the two endpoints of different 2-paths, we've found a 4-Cycle, and we are done. Otherwise, we claim the graph is 4-Cycle free. To see why, observe that any 4-Cycle that contains an outer edge must contain at least two outer edges.

Therefore, there is a pair of distinct portals u, v that cover those two edges. Any such 4-Cycle can be broken into two 2-paths that pass through u and v, each consisting of at least one outer edge. Therefore, this 4-Cycle must be detected by the algorithm.

The running time of this reduction is therefore $\tilde{O}(n^{1.5+\varepsilon} + n^{2-0.5\varepsilon})$.   ◀

4.6   Direct-WTER for Minimum Dominating Set

Our Direct-WTER for Minimum Dominating Set (abbreviated as MDS) uses twins trick. That is, by adding a unique neighbor u' to some vertex u, we can more easily control the size of the dominating set in the graph. Formally, we have:

▶ **Fact 23.** *If u' is a twin of u, there exists a minimum dominating set that contains u.*

However, if we apply the same construction that worked for MCM (and MVC); adding an expansion layer U and a twin to every $u \in U$, the vertices of U will have to be in a minimum dominating set and therefore will also dominate $V(G)$. Essentially, the size of the minimum dominating set in G' will become $MDS(G') := |U|$, losing all information about $MDS(G)$. To deal with this problem, we add an intermediate layer R that consists of independent vertices that are "copies" of vertices in G. That is, for each $v_r \in R$ we have some $v \in V(G)$ and v_r is adjacent in G only to the neighborhood of v. Hence, v_r doesn't dominate any vertices in G that cannot be dominated by v.

The vertices of G that will have copies in R are chosen by a randomized algorithm, such that every sparse cut in G will have some copies in R, while keeping the size of R less than εn. Then, we add an expansion layer U to R, and to control the size of the dominating set we add twins to U. Note that R becomes useless to any minimum dominating set because we can always replace copies with their originals. Let us now present the construction in full detail.

Proof of Theorem 12 (Direct-WTER for Minimum Dominating Set). Let $0 < \varepsilon$ be a constant and assume w.l.o.g. that ε is sufficiently small (say, $\varepsilon \leq 0.01$). We construct the graph G' using the following algorithm. The first four steps deal with sampling vertices from G that hit all sparse cuts, and the last steps take copies of the sampled vertices and make the graph expander by adding an expansion layer.

1. Let $Q = \emptyset$. For every $v \in V(G)$, add v to Q with probability 10ε, independently from other vertices.
2. For every vertex $v \in V(G) \setminus Q$, we say that v is *bad* if $|N(v) \cap Q| < \varepsilon \deg(v)$. Add to Q all the bad vertices that have degree at least $\log(1/\varepsilon)/\varepsilon$.
3. Add vertices to Q according to the next deterministic procedure. Decompose G into connected, edge-disjoint subgraphs of size (where size is the number of vertices) in range $[1/\varepsilon, 3/\varepsilon]$. From every subgraph in the decomposition pick one arbitrary vertex to Q. See the full version of the paper for more details about the decomposition.
4. Add to the graph a set of new vertices $R = \{v_r \mid v \in Q\}$, and for every $v_r \in R$ add all edges between $N_G(v)$ and v_r.
5. Add an expansion layer U adjacent to R, with the following modifications to Algorithm 4.1. Denote by $\Delta_R := \max\{\deg_G(v) \mid v_r \in R\}$.

   Modifications to Algorithm 4.1.   The modifications are similar to the ones we did in the constructions for MVC and k-Clique. For every $v_r \in R$, we wish to sample $deg_G(v) + C \log n$ edges from v_r to U. Since we can not afford to add an expansion layer of size $|U| = 5(\Delta_R + C \log n)$ (because Δ_R can be as large as n), we add an expansion layer of size $|U| = 5(\varepsilon \Delta_R + C \log n)$. Then, every vertex $v_r \in R$ will sample

$\varepsilon \deg_G(v) + C \log n$ neighbors from U (instead of $\deg_G(v) + C \log n$). Notice that we replaced the degree-increasing step by sampling $C \log n$ more neighbors from U. Thus, by Lemma 15, the induced subgraph on $R \cup U$ is an $\Omega(1)$-expander with high probability.

6. To control the size of the dominating set, for every $u \in U$ add a twin vertex u'.
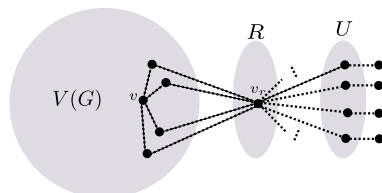
Figure 5 The proposed WTER for dominating set. The vertices of R are copies of vertices of G, thus sharing the same neighborhood in G.

Denote the resulting graph by G'. See Figure 5. It is not difficult to see that with high probability, the blowup in the number of vertices in $O(\varepsilon n)$ and that $MDS(G') = MDS(G) + |U|$.

To prove that $\phi(G') = \Omega(1)$, we show that for any cut $S \subseteq V(G)$, if S is not expanding inside G (that is, $\delta_G(S) << vol_G(S)$) then there are many edges between S and R. Since R is adjacent to U then we obtain expansion between S and R or between R and U. The first part of this claim is formalized by the next lemma:

▶ **Lemma 24.** *There exists a constant c_ε such that for every cut $S \subseteq V(G)$, either $\delta_G(S) \geq c_\varepsilon vol_G(S)$ or $e(S, R) \geq c_\varepsilon vol_G(S)$.*

The proof of this claim appears in the full version of the paper. Assuming this claim, it is not hard to see that any cut must have many cross-edges either in the induced subgraph on $V(G) \cup R$, or in the induced subgraph on $R \cup U$. In particular, we obtain the next lemma whose proof also appears in the full version of the paper:

▶ **Lemma 25.** *With high probability, for any cut $S \subseteq V(G')$ we have $\phi(S) = \Omega(1)$.* ◀

5 Definition

Let us conclude by formalizing the notion of a (fine-grained) WTER that captures Definitions 2 and 4.

▶ **Definition 26** (($\phi(n), a(n, m), b(n, m)$)-WTER). *Problem $\mathcal{A}$ has a worst-case to expander-case self-reduction with conductance $\phi(n)$ and times $a(n, m), b(n, m)$, if there exists a randomized algorithm with oracle access to $\mathcal{A}$, that solves $\mathcal{A}$ on any n-nodes, m-edges graph by making at most $k = k(n)$ calls to the oracle on instances $G_1, \ldots, G_k$, such that:*
- *each G_i is a $\phi(n)$-expander.*
- *for every $\varepsilon > 0$ there is a $\delta > 0$ such that the reduction runs in time $O(a(n, m)^{1-\delta})$ and $\sum_{i=1}^{k} b(n_i, m_i)^{1-\varepsilon} = O(a(n, m))^{1-\delta}$, where n_i, m_i are the number of nodes and edges in G_i, respectively.*

We will abbreviate by saying that $\mathcal{A}$ admits a $(\phi(n), a(n, m), b(n, m))$-WTER. The intuition behind this definition can be described as follows. If a problem admits a $(\phi(n), a(n, m), b(n, m))$-WTER, then any "fast" (i.e. $b(n, m)^{1-\varepsilon}$ time) algorithm on $\phi(n)$-expanders can be used to obtain a "faster" $(a(n, m)^{1-\delta}$ time) algorithm on worst-case graphs. For example, if a problem admits a $((\log n)^{-1}, n^3, n^2)$-WTER. Then the existence of any

$O(n^{2(1-\varepsilon)})$-time algorithm that solves the problem on n-vertex $(\log n)^{-1}$-expanders implies that the problem can be solved on worst-case graphs in time $O(n^{3(1-\delta)})$. In this work we mainly focused on $(\Omega(1), a(n, m), a(n, m))$-WTERs, where $a(n, m)$ is the best known upper bound to solve the problem, essentialy showing the problem is as easy on worst-case graphs as on expanders.

For completeness, we formalize the intuition in the next theorem:

▶ **Theorem 27.** *If problem $\mathcal{A}$ has a $(\phi(n), a(n, m), b(n, m))$-WTER, then for all $\varepsilon > 0$ there is a $\delta > 0$ such that if there is an algorithm that solves $\mathcal{A}$ on $\phi(n)$-expanders in time $O(b(n, m)^{1-\varepsilon})$, then $\mathcal{A}$ can be solved on worst-case graphs in time $O(a(n, m)^{1-\delta})$.*

Proof. Suppose that problem $\mathcal{A}$ has a $(\phi(n), a(n, m), b(n, m))$-WTER and that there is an algorithm that runs in time $O(b(n, m)^{1-\varepsilon})$ that solves the problem on $\phi(n)$-expanders. Then we can simulate the algorithm in the WTER on worst-case graphs by replacing the oracle with the fast algorithm on expanders. The total running time is $O(a(n, m)^{1-\delta})$, for some $\delta > 0$. ◀

References

1 Amir Abboud, Arturs Backurs, and Virginia Vassilevska Williams. If the current clique algorithms are optimal, so is Valiant's parser. *SIAM J. Comput.*, 47(6):2527–2555, 2018. `doi:10.1137/16M1061771`.

2 Amir Abboud, Karl Bringmann, Seri Khoury, and Or Zamir. Hardness of approximation in P via short cycle removal: cycle detection, distance oracles, and beyond. In Stefano Leonardi and Anupam Gupta, editors, *STOC '22: 54th Annual ACM SIGACT Symposium on Theory of Computing, Rome, Italy, June 20 - 24, 2022*, pages 1487–1500. ACM, 2022. `doi:10.1145/3519935.3520066`.

3 Amir Abboud, Vincent Cohen-Addad, and Philip N. Klein. New hardness results for planar graph problems in P and an algorithm for sparsest cut. In Konstantin Makarychev, Yury Makarychev, Madhur Tulsiani, Gautam Kamath, and Julia Chuzhoy, editors, *Proccedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing, STOC 2020, Chicago, IL, USA, June 22-26, 2020*, pages 996–1009. ACM, 2020. `doi:10.1145/3357713.3384310`.

4 Amir Abboud and Søren Dahlgaard. Popular conjectures as a barrier for dynamic planar graph algorithms. In Irit Dinur, editor, *IEEE 57th Annual Symposium on Foundations of Computer Science, FOCS 2016, 9-11 October 2016, Hyatt Regency, New Brunswick, New Jersey, USA*, pages 477–486. IEEE Computer Society, 2016. `doi:10.1109/FOCS.2016.58`.

5 Amir Abboud, Fabrizio Grandoni, and Virginia Vassilevska Williams. Subcubic equivalences between graph centrality problems, apsp and diameter. In *Proceedings of the twenty-sixth annual ACM-SIAM symposium on Discrete algorithms*, pages 1681–1697. SIAM, 2014.

6 Amir Abboud, Robert Krauthgamer, Jason Li, Debmalya Panigrahi, Thatchaphol Saranurak, and Ohad Trabelsi. Gomory-hu tree in subcubic time. *arXiv preprint arXiv:2111.04958*, 2021.

7 Amir Abboud, Robert Krauthgamer, and Ohad Trabelsi. Subcubic algorithms for gomory-hu tree in unweighted graphs. In Samir Khuller and Virginia Vassilevska Williams, editors, *STOC '21: 53rd Annual ACM SIGACT Symposium on Theory of Computing, Virtual Event, Italy, June 21-25, 2021*, pages 1725–1737. ACM, 2021. `doi:10.1145/3406325.3451073`.

8 Amir Abboud, Robert Krauthgamer, and Ohad Trabelsi. Friendly cut sparsifiers and faster Gomory-Hu trees. In *Proceedings of the 2022 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 3630–3649. SIAM, 2022.

9 Amir Abboud and Virginia Vassilevska Williams. Popular conjectures imply strong lower bounds for dynamic problems. In *55th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2014, Philadelphia, PA, USA, October 18-21, 2014*, pages 434–443. IEEE Computer Society, 2014. `doi:10.1109/FOCS.2014.53`.

10 Amir Abboud, Virginia Vassilevska Williams, and Joshua R. Wang. Approximation and fixed parameter subquadratic algorithms for radius and diameter in sparse graphs. In Robert Krauthgamer, editor, *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016, Arlington, VA, USA, January 10-12, 2016*, pages 377–391. SIAM, 2016. `doi:10.1137/1.9781611974331.ch28`.

11 Udit Agarwal and Vijaya Ramachandran. Fine-grained complexity for sparse graphs. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing*, pages 239–252, 2018.

12 Daniel Agassy, Dani Dorfman, and Haim Kaplan. Expander decomposition with fewer inter-cluster edges using a spectral cut player. *CoRR*, abs/2205.10301, 2022. `doi:10.48550/arXiv.2205.10301`.

13 Vedat Levi Alev, Nima Anari, Lap Chi Lau, and Shayan Oveis Gharan. Graph clustering using effective resistance. *arXiv preprint arXiv:1711.06530*, 2017.

14 Josh Alman and Virginia Vassilevska Williams. A refined laser method and faster matrix multiplication. In *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms, SODA 2021*, pages 522–539, 2021. `doi:10.1137/1.9781611976465.32`.

15 Noga Alon, Raphael Yuster, and Uri Zwick. Color-coding. *Journal of the ACM (JACM)*, 42(4):844–856, 1995.

16 Noga Alon, Raphael Yuster, and Uri Zwick. Finding and counting given length cycles. *Algorithmica*, 17(3):209–223, 1997. `doi:10.1007/BF02523189`.

17 Vahid R Asadi, Alexander Golovnev, Tom Gur, and Igor Shinkar. Worst-case to average-case reductions via additive combinatorics. *arXiv preprint arXiv:2202.08996*, 2022.

18 Marshall Ball, Alon Rosen, Manuel Sabin, and Prashant Nalini Vasudevan. Average-case fine-grained hardness. In Hamed Hatami, Pierre McKenzie, and Valerie King, editors, *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2017, Montreal, QC, Canada, June 19-23, 2017*, pages 483–496. ACM, 2017. `doi:10.1145/3055399.3055466`.

19 Nikhil Bansal and Ryan Williams. Regularity lemmas and combinatorial algorithms. *Theory Comput.*, 8(1):69–94, 2012. `doi:10.4086/toc.2012.v008a004`.

20 Holger Bast, Kurt Mehlhorn, Guido Schafer, and Hisao Tamaki. Matching algorithms are fast in sparse random graphs. *Theory of Computing Systems*, 39(1):3–14, 2006.

21 Richard Beigel. Finding maximum independent sets in sparse and general graphs. In *SODA*, volume 99, pages 856–857. Citeseer, 1999.

22 Nicolas Bourgeois, Bruno Escoffier, Vangelis T Paschos, and Johan MM van Rooij. Fast algorithms for max independent set. *Algorithmica*, 62(1):382–415, 2012.

23 Yi-Jun Chang, Seth Pettie, Thatchaphol Saranurak, and Hengjie Zhang. Near-optimal distributed triangle enumeration via expander decompositions. *J. ACM*, 68(3):21:1–21:36, 2021. `doi:10.1145/3446330`.

24 Jianer Chen, Iyad A Kanj, and Ge Xia. Improved parameterized upper bounds for vertex cover. In *International symposium on mathematical foundations of computer science*, pages 238–249. Springer, 2006.

25 Li Chen, Rasmus Kyng, Yang P. Liu, Richard Peng, Maximilian Probst Gutenberg, and Sushant Sachdeva. Maximum flow and minimum-cost flow in almost-linear time. *CoRR*, abs/2203.00671, 2022. `doi:10.48550/arXiv.2203.00671`.

26 Julia Chuzhoy, Yu Gao, Jason Li, Danupon Nanongkai, Richard Peng, and Thatchaphol Saranurak. A deterministic algorithm for balanced cut with applications to dynamic connectivity, flows, and beyond. In *61st IEEE Annual Symposium on Foundations of Computer Science, FOCS 2020*, pages 1158–1167, 2020. `doi:10.1109/FOCS46700.2020.00111`.

27 Julia Chuzhoy and Sanjeev Khanna. A new algorithm for decremental single-source shortest paths with applications to vertex-capacitated flow and cut problems. In Moses Charikar and Edith Cohen, editors, *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing, STOC 2019, Phoenix, AZ, USA, June 23-26, 2019*, pages 389–400. ACM, 2019. `doi:10.1145/3313276.3316320`.

28   Marek Cygan, Holger Dell, Daniel Lokshtanov, Dániel Marx, Jesper Nederlof, Yoshio Okamoto, Ramamohan Paturi, Saket Saurabh, and Magnus Wahlström. On problems as hard as CNF-SAT. *ACM Trans. Algorithms*, 12(3):41:1–41:24, 2016. `doi:10.1145/2925416`.

29   Mina Dalirrooyfard, Andrea Lincoln, and Virginia Vassilevska Williams. New techniques for proving fine-grained average-case hardness. In Sandy Irani, editor, *61st IEEE Annual Symposium on Foundations of Computer Science, FOCS 2020, Durham, NC, USA, November 16-19, 2020*, pages 774–785. IEEE, 2020. `doi:10.1109/FOCS46700.2020.00077`.

30   Jacob Evald and Søren Dahlgaard. Tight hardness results for distance and centrality problems in constant degree graphs. *CoRR*, abs/1609.08403, 2016. `arXiv:1609.08403`.

31   Fedor V Fomin, Fabrizio Grandoni, and Dieter Kratsch. Measure and conquer: A simple $O(2^{0.288n})$ independent set algorithm. In *SODA*, volume 6, pages 18–25. Citeseer, 2006.

32   Fedor V Fomin, Dieter Kratsch, and Gerhard J Woeginger. Exact (exponential) algorithms for the dominating set problem. In *International Workshop on Graph-Theoretic Concepts in Computer Science*, pages 245–256. Springer, 2004.

33   Oded Goldreich. Basic facts about expander graphs. In *Studies in Complexity and Cryptography. Miscellanea on the Interplay between Randomness and Computation*, pages 451–464. Springer, 2011.

34   Monika Henzinger, Andrea Lincoln, and Barna Saha. The complexity of average-case dynamic subgraph counting. In Joseph (Seffi) Naor and Niv Buchbinder, editors, *Proceedings of the 2022 ACM-SIAM Symposium on Discrete Algorithms, SODA 2022, Virtual Conference / Alexandria, VA, USA, January 9 - 12, 2022*, pages 459–498. SIAM, 2022. `doi:10.1137/1.9781611977073.23`.

35   Monika Henzinger, Ami Paz, and AR Sricharan. Fine-grained complexity lower bounds for families of dynamic graphs. *arXiv preprint arXiv:2208.07572*, 2022.

36   Ravi Kannan, Santosh Vempala, and Adrian Vetta. On clusterings: Good, bad and spectral. *Journal of the ACM*, 51(3):497–515, 2004. `doi:10.1145/990308.990313`.

37   Jonathan A Kelner, Yin Tat Lee, Lorenzo Orecchia, and Aaron Sidford. An almost-linear-time algorithm for approximate max flow in undirected graphs, and its multicommodity generalizations. In *Proceedings of the 25th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 217–226. SIAM, 2014.

38   Alexander S. Kulikov and Ivan Mihajlin. Polynomial formulations as a barrier for reduction-based hardness proofs. *CoRR*, abs/2205.07709, 2022. `doi:10.48550/arXiv.2205.07709`.

39   Jason Li and Thatchaphol Saranurak. Deterministic weighted expander decomposition in almost-linear time. *CoRR*, abs/2106.01567, 2021. `arXiv:2106.01567`.

40   Andrea Lincoln, Virginia Vassilevska Williams, and R. Ryan Williams. Tight hardness for shortest cycles and paths in sparse graphs. In Artur Czumaj, editor, *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2018, New Orleans, LA, USA, January 7-10, 2018*, pages 1236–1252. SIAM, 2018. `doi:10.1137/1.9781611975031.80`.

41   Yaowei Long and Thatchaphol Saranurak. Near-optimal deterministic vertex-failure connectivity oracles. *CoRR*, abs/2205.03930, 2022. `doi:10.48550/arXiv.2205.03930`.

42   Silvio Micali and Vijay V. Vazirani. An $O(\sqrt{|v|} \cdot |E|)$ algorithm for finding maximum matching in general graphs. In *21st Annual Symposium on Foundations of Computer Science, Syracuse, New York, USA, 13-15 October 1980*, pages 17–27. IEEE Computer Society, 1980. `doi:10.1109/SFCS.1980.12`.

43   Rajeev Motwani. Average-case analysis of algorithms for matchings and related problems. *Journal of the ACM (JACM)*, 41(6):1329–1356, 1994.

44   Marcin Mucha and Piotr Sankowski. Maximum matchings via gaussian elimination. In *45th Annual IEEE Symposium on Foundations of Computer Science*, pages 248–255. IEEE, 2004.

45   Danupon Nanongkai, Thatchaphol Saranurak, and Christian Wulff-Nilsen. Dynamic minimum spanning forest with subpolynomial worst-case update time. In *2017 IEEE 58th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 950–961. IEEE, 2017.

46 Lorenzo Orecchia, Sushant Sachdeva, and Nisheeth K. Vishnoi. Approximating the exponential, the Lanczos method and an $\tilde{O}(m)$-time spectral algorithm for balanced separator. In *Proceedings of the 44th Annual ACM Symposium on Theory of Computing*, pages 1141–1160, 2012. `doi: 10.1145/2213977.2214080`.

47 Lorenzo Orecchia and Nisheeth K Vishnoi. Towards an SDP-based approach to spectral methods: A nearly-linear-time algorithm for graph partitioning and decomposition. In *Proceedings of the 22nd Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 532–545. SIAM, 2011.

48 Igor Razgon. Faster computation of maximum independent set and parameterized vertex cover for graphs with maximum degree 3. *Journal of Discrete Algorithms*, 7(2):191–212, 2009.

49 Thatchaphol Saranurak and Di Wang. Expander decomposition and pruning: Faster, stronger, and simpler. In *Proceedings of the 30th Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2019*, pages 2616–2635, 2019. `doi:10.1137/1.9781611975482.162`.

50 Daniel A Spielman and Shang-Hua Teng. A local clustering algorithm for massive graphs and its application to nearly linear time graph partitioning. *SIAM Journal on computing*, 42(1):1–26, 2013.

51 Daniel A Spielman and Shang-Hua Teng. Nearly linear time algorithms for preconditioning and solving symmetric, diagonally dominant linear systems. *SIAM Journal on Matrix Analysis and Applications*, 35(3):835–885, 2014.

52 Johan MM Van Rooij and Hans L Bodlaender. Exact algorithms for dominating set. *Discrete Applied Mathematics*, 159(17):2147–2164, 2011.

53 Virginia Vassilevska Williams. On some fine-grained questions in algorithms and complexity. In *Proceedings of the International Congress of Mathematicians: Rio de Janeiro 2018*, pages 3447–3487. World Scientific, 2018.

54 Virginia Vassilevska Williams and R. Ryan Williams. Subcubic equivalences between path, matrix, and triangle problems. *J. ACM*, 65(5):27:1–27:38, 2018. `doi:10.1145/3186893`.

55 Gerhard J Woeginger. Exact algorithms for NP-hard problems: A survey. In *Combinatorial optimization—eureka, you shrink!*, pages 185–207. Springer, 2003.

56 Raphael Yuster and Uri Zwick. Finding even cycles even faster. *SIAM J. Discret. Math.*, 10(2):209–222, 1997. The conference version appeared in ICALP 1994. `doi: 10.1137/S0895480194274133`.