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

Amir Abboud
Weizmann Institute of Science, Rehovot, Israel

Nathan Wallheimer
Weizmann Institute of Science, Rehovot, Israel

Abstract

A recent paper by Abboud and Wallheimer [ITCS 2023] presents self-reductions for various fundamental graph problems, which transform worst-case instances to expanders, thus proving that the complexity remains unchanged if the input is assumed to be an expander. An interesting corollary of their self-reductions is that if some problem admits such reduction, then the popular algorithmic paradigm based on expander-decompositions is useless against it. In this paper, we improve their core gadget, which augments a graph to make it an expander while retaining its important structure. Our new core construction has the benefit of being simple to analyze and generalize while obtaining the following results:

- A derandomization of the self-reductions, showing that the equivalence between worst-case and expander-case holds even for deterministic algorithms, and ruling out the use of expander-decompositions as a derandomization tool.
- An extension of the results to other models of computation, such as the Fully Dynamic model and the Congested Clique model. In the former, we either improve or provide an alternative approach to some recent hardness results for dynamic expander graphs by Henzinger, Paz, and Sricharan [ESA 2022].

In addition, we continue this line of research by designing new self-reductions for more problems, such as Max-Cut and dynamic Densest Subgraph, and demonstrating that the core gadget can be utilized to lift lower bounds based on the OMv Conjecture to expanders.

2012 ACM Subject Classification Theory of computation → Design and analysis of algorithms

Keywords and phrases Fine-grained complexity, expander graphs, self-reductions, worst-case to expander-case, expander decomposition, dynamic algorithms, exact and parameterized complexity, max-cut, maximum matching, k-clique detection, densest subgraph

Digital Object Identifier 10.4230/LIPIcs.ESA.2024.4

Related Version *Full Version*: `https://arxiv.org/abs/2403.08394`

Funding Supported by an Alon scholarship and a research grant from the Center for New Scientists at the Weizmann Institute of Science.

Acknowledgements We would like to thank Thatchaphol Saranurak for clarifying a typo in the mentioned lower bound for the Densest Subgraph problem [15, Table 5].

1 Introduction

When studying the complexity of any graph problem, it is natural to ask whether the problem can be solved faster on *expanders*, i.e., random-like, well-connected graphs that satisfy a certain definition of expansion.

▶ **Question 1.** *Are expanders worst-case instances of my problem?*

The motivation for such a question comes from multiple sources. First, it is inherently interesting to understand how the rich mathematical structure of expanders affects the complexity of fundamental problems such as shortest paths, cuts, matchings, subgraph detection, and so on. After all, expanders are among the most important graph families in computer science. Second, expanders exhibit some of the most algorithmically useful properties of *uniformly random* graphs, and so this question may help understand the *average-case* complexity. Third, graphs that arise in applications may be expanders (e.g., in network architecture). Moreover last but not least is the hope that if we solve a problem faster on expanders, we will also be able to solve it in the worst case by utilizing the popular *expander decomposition method*, which we discuss soon. Unless explicitly stated otherwise, we use the conductance-based notion of ϕ-expanders, whose precise definition can be found in Section 3, and we say that a graph is an expander if it is an $\Omega(1)$-expander.

It is possible to cook up problems for which the answer to Question 1 is negative. For instance, we can solve connectivity in constant time if the input is promised to be an expander, but it requires linear time in the worst case. A less obvious example with such gaps is counting spanning trees [23]. However, for many (perhaps most) interesting graph problems, the answer seems to be *positive*: expanders do not make the problem any easier. In other words, the *expander-case* is also worst-case. But how do we prove that? Let us discuss three methods and their drawbacks.

1. The first and most obvious method is to prove a lower bound for the problem on expander instances that matches the worst-case upper bound. Technically, this may follow directly from the existing lower bounds for the problem since they are often proved on random-like graphs (e.g., for distance oracles [29]), or it may require some modifications to the lower bound proofs (e.g., for dynamic graph problems [17]).

   The main drawback with this approach is that we are interested in answering Question 1 even when (or rather, especially when) we have not already resolved the worst-case time complexity of our problem, in which case we do not even have a matching lower bound (e.g., the Maximum Matching problem). Another drawback is that when asking for the *fine-grained complexity* of problems, the existing lower bounds are usually conditioned on strong assumptions, and one may hope to get an unconditional answer to Question 1.

   The next two approaches resolve these drawbacks since they are based on *worst-case to expander-case self-reductions* (WTERs). Such techniques show equivalence between the expander-case complexity and the worst-case complexity.

2. The second approach uses the *expander decomposition method.* This is a popular paradigm in recent years that suggests we can solve graph problems by (1) decomposing the graph into vertex-disjoint expanders with a small number of edges between them, (2) solving the problem on each expander separately, and (3) combining all the answers efficiently. Step (3) requires problem-specific techniques. If we can solve steps (1) and (3) for a problem, then we have effectively shown that any improvement on expanders will yield an improvement on worst-case graphs, giving a positive answer to Question 1. Efficient algorithms for computing such expander decompositions (for step (1)) are known both in the Word-RAM [3, 28, 19, 30, 25] and in other models of computation including dynamic [28], distributed [7], and recently even in streaming [11]. Applications of this paradigm have led to many breakthroughs in recent years to problems such as Maximum Flow [8], Dynamic Connectivity [14], Gomory-Hu Trees [1], Minimum Spanning Trees [24], and Triangle Enumeration [7].

   Self-reductions of this form are called *ED-WTER*s. A recent paper by Abboud and Wallheimer [2] proposed an alternative, simpler method of self-reductions that *do not* use expander decompositions. They call this method *Direct-WTER*s, which we discuss next, and it is not only simpler but also yields stronger qualitative and quantitative results.

3. The third and most direct method is to show that any graph can be turned into an expander without affecting the solution to the problem or increasing the size of the graph by too much. In their paper, Abboud and Wallheimer [2] gave the following definition, which we slightly reframe to fit our discussion more accurately:

   ▶ **Definition 1** (Direct-WTER [2, Definition 2])**.** *A direct worst-case to expander-case self-reduction to a graph problem $\mathcal{A}$, is an algorithm that given any instance G with n vertices and m edges, computes in $\tilde{O}(n+m)$ time a graph $G_{exp} := (V_{exp}, E_{exp})$ with the following guarantees:*
   - *G_{exp} is an $\Omega(1)$-expander with high probability.*
   - *The* blowup *in the number of vertices and edges in G_{exp} is $|V_{exp}| \leq K$ and $|E_{exp}| \leq M$ for some $K := K(n,m)$ and $M := M(n,m)$.*
   - *The solution $\mathcal{A}(G)$ can be computed from the solution $\mathcal{A}(G_{exp})$ in $\tilde{O}(m+n)$ time.*

   Direct-WTERs can be used to show equivalence between the complexity of polynomial-time problems and their complexity on $\Omega(1)$-expanders. Namely, if problem $\mathcal{A}$ is a polynomial-time problem that admits a Direct-WTER, then $\Omega(1)$-expanders are worst-case instances of $\mathcal{A}$ (ignoring poly-logarithmic factors).

   The main contribution of Abboud and Wallheimer was to show that some fundamental problems, such as k-Clique Detection and Maximum Matching, admit simple Direct-WTERs. In particular, their Direct-WTERs make a graph an expander by employing a *core gadget* that augments it with $O(n)$ vertices and $O(m + n\log n)$ random edges and then applies additional gadgets that control the solution. In particular, they obtain a near-linear blowup. Their results are surprising because such Direct-WTERs do not employ any of the heavy machinery that usually comes with expander decompositions, yet they output quantitatively better expanders: the outputs of Direct-WTERs are $\Omega(1)$-expanders (by definition), whereas expander decompositions can only produce $O(1/\log n)$-expanders (that are not as expanding) [28, 4].

   Furthermore, the simplicity of such Direct-WTERs leads to interesting and important messages to algorithm designers, as observed in [2]: The expander decomposition method is useless in the presence of Direct-WTERs because decomposing a graph into $o(1)$-expanders is meaningless when we can assume that the input graph is already an $\Omega(1)$-expander after a simple modification. This addressed (with a negative answer) a question that many researchers have wondered about as they looked for the next breakthrough to be obtained via the expander decomposition method:

   ▶ **Question 2.** *Are expander decompositions the key to solving my problem?*

This work

Motivated by the appeal of the method of Direct-WTERs towards answering Question 1 and Question 2, our goal is to develop this theory further. Toward that, we address the two main limitations that were highlighted in [2]: (1) the Direct-WTERs are *randomized* whereas ED-WTERs are deterministic [10], and (2) the results are restricted to the Word-RAM model, whereas expander decompositions are popular tools in other models as well. In addition, we continue their line of work by providing Direct-WTERs to additional problems. Let us motivate these two topics before stating our results formally.

1.1 Deterministic Direct-WTERs

The randomized Direct-WTERs of Abboud and Wallheimer [2] prove that $\Omega(1)$-expanders are worst-case instances of many problems *if* we allow algorithms to be randomized. They leave us wondering if perhaps $\Omega(1)$-expanders are truly easier for deterministic algorithms.

We remark that while it is believed that all algorithms can be derandomized by incurring a small polynomial blowup (as in $P = BPP$), it is far from clear that this blowup can be made $n^{o(1)}$ (see [9] for the state-of-the-art on such results). Can we provide a positive answer to Question 1 with respect to *deterministic* algorithms by designing *deterministic* Direct-WTERs?

Additional motivation comes from the hope of using the expander decomposition method in order to get breakthrough *derandomization* results, along the lines of Question 2, for problems where the current randomized algorithms are much faster than the current deterministic algorithms. Indeed, deterministic expander decompositions [10] have already played a major role in some of the most remarkable derandomization results of recent years, e.g. for Global Min-Cut [20, 27, 22, 21, 16].

▶ **Question 3.** *Are expander decompositions the key to derandomizing my algorithm?*

Similarly to the observation in [2], *deterministic Direct-WTER* also convey a message to algorithm designers: that the answer to the above question is negative, i.e., expander decompositions are useless for derandomizing the problem.

Motivated by this, our first result is a derandomization of the *core gadget* in [2], resulting in deterministic Direct-WTERs for various problems. In particular, we show that all problems admitting *randomized* Direct-WTERs in [2], and some additional problems, such as the Max-Cut problem, admit deterministic direct-WTERs.

▶ **Theorem 2.** *The following problems admit deterministic Direct-WTERs: Maximum Matching, Minimum Vertex Cover, k-Clique Detection, k-Clique Counting, Max-Clique, Max-Cut, Minimum Dominating Set, and H-Subgraph Detection ($m = \tilde{O}(n)$ and H does not contain pendant vertices).*

We provide formal definitions and an overview of all problems mentioned above in the full version of this paper.

An important feature of our deterministic core gadget is that it remains simple and efficient. Interestingly, this stands in contrast with other derandomization results in fine-grained complexity, which often tends to involve sophisticated methods and some slowdown (see, e.g., [6, 12]). As discussed earlier, simplicity is an important aspect of Direct-WTERs, not just because it makes them more accessible to the community, but also because it strengthens the message that expander decompositions become useless in the presence of Direct-WTERs. In Section 2, we provide an intuitive overview of how our derandomization is obtained by a modification to the core gadget of Abboud and Wallheimer [2] and in Section 4 we provide the construction itself.

Interestingly, our Direct-WTERs also improve upon the blowup in the number of added edges over the Direct-WTERs in [2], resulting in $\Omega(1)$-expanders with $O(n)$ vertices and $O(m + n)$ edges, whereas in [2], the expanders have $O(m + n \log n)$ edges. Note that this blowup is optimal since any expander is connected and, therefore, must contain $\Omega(n)$ edges. We also demonstrate that our core gadget preserves the following graph properties: (1) Bipartiteness-preserving; we can modify the core gadget so that if G is bipartite, then so is the expander, and (2) Degree-preserving; if the maximum degree in G is Δ, then the maximum-degree in G_{exp} is $2\Delta + O(1)$.

Remarks

1. For the exponential-time problems in Theorem 2: Minimum Vertex Cover, Minimum Dominating Set, Max-Clique, and Max-Cut, the blowup in the number of vertices in the output graph must be subject to stronger restrictions than for polynomial-time problems.

In particular, the blowup should be $n + o(n)$, to show that expanders are worst-case instances.[1] For such problems, we employ a generalized core gadget, which gives a tradeoff between the conductance and the blowup, resulting in Direct-WTERs providing, for every $0 < \varepsilon \leq 1$, conductance $\Omega(\varepsilon)$ and blowup εn.

2. For k-Clique Detection, we slightly improve the parameters over the Direct-WTERs given in [2] for this problem. In [2], the output is an $\Omega(1/k^2)$-expander G_{exp}, with a blowup of $\Theta(nk)$ vertices and $\Theta(k^2m)$ edges, where each k-clique in G corresponds to $k!$ k-cliques in G_{exp}. Our improved Direct-WTER produces an $\Omega(1)$-expander G_{exp} with $O(n)$ vertices and $O(m+n)$ edges (even if $k = \omega(1)$), such that every k-clique in G corresponds to $k+1$ k-cliques in G_{exp}. This enhancement makes our reduction more suitable for parameterized algorithms and larger (non-constant) values of k.

3. For the Max-Cut problem, there is an interesting related work on the approximation variant of the problem on expanders. A famous algorithm by Goemans and Williamson [13] obtains a > 0.878-approximation for the maximum cut in general graphs, based on a Semidefinite Programming relaxation. In search of other, perhaps simpler methods for approximating the max-cut beyond the trivially obtained 1/2-approximation[2], Trevisan [31] posed the following question: Is there a combinatorial algorithm that achieves better than 1/2-approximation?

   A positive answer to this question was given by Kale and Seshadhri [18], which remains the current-best combinatorial algorithm for this problem. A recent paper by Peng and Yoshida [26] addresses this question on expanders, providing a combinatorial algorithm for approximating the maximum cut on ϕ-expanders. Namely, the authors provide an algorithm that given ε, computes a $(1/2+\varepsilon)$-approximation, subject to $\varepsilon = O(\phi^2)$. In more detail, it computes a value x such that $(1/2+\varepsilon)MC(G) \leq x \leq MC(G)$, where $MC(\cdot)$ denotes the cardinality of the maximum cut in G. Moreover, its running time is sublinear when ϕ is a constant. Our Direct-WTER for Max-Cut in Theorem 2, on the other hand, is a reduction that given G and $0 < \phi \leq 1$, outputs an $\Omega(\phi)$-expander G_{exp}, and the maximum cut in G_{exp} is $MC(G_{exp}) \leq (1+4\phi)MC(G)$, assuming the input graph is not too sparse, say, $m = \omega(n)$.

   Can we apply the Direct-WTER and then use the algorithm of Peng and Yoshida to get a combinatorial algorithm that $(1/2+\varepsilon)$-approximates the maximum cut in general graphs? Perhaps we may even improve upon the algorithm by Kale and Seshadhri, as both the Direct-WTER and the algorithm of Peng and Yoshida are very efficient. However, the approximate value we get from this approach is $(1/2+\varepsilon)MC(G) \leq x \leq (1+4\phi)MC(G)$, or equivalently $(1/2+\varepsilon)/(1+4\phi)MC(G) \leq x \leq MC(G)$. For this approximation ratio to be larger than 1/2, we need to pick $\varepsilon > 2\phi$, but recall the constraint $\varepsilon \leq O(\phi^2)$. Hence, this approach fails. This is not so surprising, since our Direct-WTER is quite elementary so we do not expect to make use of it as a subroutine inside another algorithm. Instead, this result should be interpreted as a limitation to algorithms for Max-Cut on ϕ-expanders; that one cannot obtain a $(1/2+\varepsilon)$-approximation for some $2\phi < \varepsilon \leq 1/2$ unless this (unlikely) approach works.

[1] Otherwise, an exponential speed-up on expanders does not necessarily translate to an exponential speed-up on general graphs.
[2] Which follows from the fact that the maximum cut is at least $m/2$ in any graph with m edges.

1.2 Direct-WTERs in the Fully Dynamic setting

Before this work, Direct-WTERs were limited to the (randomized) Word-RAM model, whereas ED-WTERs could address many other models. One particular area in which expanders, expander decomposition, and derandomization are important subjects is the area of dynamic graph algorithms. Let us focus on the Fully Dynamic model of computation, where the goal is to maintain the solution of a problem in a graph undergoing edge updates, i.e., edge insertions and deletions. Dedicated tools have been developed for maintaining an expander decomposition in this model and subsequently achieved major breakthroughs (e.g., [24]). Notably, derandomization is a central concern in the dynamic setting because deterministic algorithms are essentially the only ones that work against adaptive adversaries (see, e.g., [5]). For these reasons, the three main questions outlined above are particularly interesting in this model. Our first question, in this context, is whether Direct-WTERs can be adapted to the Fully Dynamic setting.

An important related work is a recent paper by Henzinger, Paz, and Sricharan [17] (abbreviated as HPS), who initiated the study of Question 1 in the dynamic model, regarding the complexity of fundamental problems on *dynamic expanders*. A dynamic expander is a dynamic graph that undergoes edge updates but remains an $\Omega(1)$-expander at any point in time. They adapt lower bound proofs from fine-grained complexity, so that they hold even on dynamic expanders. Their techniques differ from the self-reduction approach of Direct-WTERs and correspond to the first of the three methods outlined above to answer Question 1. In particular, they base their results on the *Online Matrix Vector* (OMv) conjecture, which was introduced by Henzinger et al. [15] to prove lower bounds for various dynamic problems. HPS obtained their results by adapting these lower bound proofs to the case of constant-degree expanders, proving that dynamic expanders whose maximum degree remains bounded by a constant are OMv-hard. The problems they consider in this context are Maximum Matching, Densest Subgraph, and *st*-Shortest Path (abbreviated as *st*-SP). This leaves us wondering with the following questions:

1. Are expanders in higher density regimes, whose maximum degree is not bounded by a constant, also OMv-hard instances?
2. Can the techniques that are used in the static setting to prove Theorem 2 contribute to this study by, e.g., providing a simpler, or alternative method to prove that certain problems remain OMv-hard on expanders?

Our second main result is an adaptation of the deterministic core gadget to the dynamic setting, resulting in a deterministic, dynamic algorithm for maintaining a dynamic $\Omega(1)$-expander, whose running time is amortized $\tilde{O}(1)$ per edge update. Subsequently, we show *Dynamic Direct-WTERs* (abbreviated as DD-WTERs) to various problems,[3] thus proving that $\Omega(1)$-expanders are worst-case instances and that the expander decomposition method is useless against them. We present and discuss the formal definition of DD-WTERs in the full version of this paper.

▶ **Theorem 3.** *The following problems admit a DD-WTER: Maximum Matching, Bipartite Perfect Matching, Densest Subgraph (in graphs with $m > 42n$ edges), k-Clique Detection, k-Clique Counting, and H-Subgraph Detection (where $m = \tilde{O}(n)$ and H does not contain pendant vertices).*

[3] Namely, to all problems in the previous theorem, except that we do not discuss the exponential-time problems in the dynamic model in this work.

In addition, our results also have some interesting implications related to the work of HPS and the questions above.

1. For the Densest Subgraph problem, Henzinger et al. [15] prove a lower bound of $n^{1/3-o(1)}$ per update under OMv for *general graphs*. By modifying their reduction, HPS were able to show a weaker $n^{1/4-o(1)}$ lower bound for *expanders*.[4]
   As a consequence of our DD-WTER, we conclude that the $n^{1/3-o(1)}$ lower bound for general graphs also holds for expanders. One subtlety towards this result is that the reduction of Henzinger et al. produces very sparse graphs with $m \leq 2n$ edges while our DD-WTER assumes that $m > 42n$. In the full version of this paper, we discuss how to modify the original reduction so that denser graphs are produced. Another strength of the DD-WTER compared to HPS is that it is a self-reduction, hence it does not depend on the OMv Conjecture to get a lower bound (at least when the $m > 42n$ assumption is made)[5].
2. For the Maximum Matching problem on graphs of maximum degree $O(n^t)$, for any $0 \leq t \leq 1$, we show that this problem is OMv-hard on $\Omega(1)$-expanders as well. This improves upon HPS, who only prove it for the constant-degree case (i.e. $t = 0$) [17, Theorem 12], while implicitly leaving an open question regarding $t > 0$. While intuitively, the constant-degree case should be the most difficult to prove a lower bound for, it is not immediate, as techniques that artificially increase the degrees in the graph (e.g. attaching a star to every vertex) tend to also increase the number of vertices, thus resulting in weaker lower bounds. Instead, our result is obtained by combining a lower bound by HPS for graphs (not expanders) of maximum degree $O(n^t)$ [17, Theorem 12], with our degree-preserving DD-WTER for Maximum Matching. Hence, we get the following corollary:
   ▶ **Corollary 4.** *For any $0 \leq \varepsilon, t \leq 1$ and any constant $\varepsilon > 0$, there is no dynamic algorithm for maintaining a maximum matching on $\Omega(1)$-expanders with maximum degree $O(n^t)$, with amortized $O(n^{(1+t)/2-\varepsilon})$ update time and $O(n^{1+t-\varepsilon})$ query time, unless the OMv Conjecture is false.*
3. We demonstrate that our core gadget is useful even outside the context of self-reductions by showing a DD-WTER for *Graphical OMv*, an equivalent graph formulation of the OMv problem. This implies that we can lift OMv-based lower bounds to $\Omega(1)$-expanders for many problems, in particular, for problems such as Maximum Matching and *st*-SP, which HPS considered. We demonstrate the power of this technique by proving that *st*-SP is OMv-hard on $\Omega(1)$-expanders.[6]
   ▶ **Proposition 5** (*st*-SP is OMv-hard on $\Omega(1)$-expanders)**.** *For any $\varepsilon > 0$, there is no dynamic algorithm for the dynamic st-SP problem on $\Omega(1)$-expanders, with polynomial preprocessing time, $O(m^{1/2-\varepsilon})$ update time, and $O(m^{1-\varepsilon})$ query time, assuming the OMv Conjecture.*

Finally, let us emphasize another aspect of how our work and HPS differ. One of the main motivations for our work is to address Question 2 and Question 3 about the applicability of expander decompositions in algorithms, whereas HPS main motivation is to gain a better

[4] We remark that the proof in Henzinger et al. [15, Corollary 3.26] gives an $n^{1/3-o(1)}$ lower bound under OMv, while the introductions of both [15] and HPS [17] mention, erroneously, an $n^{1/2-o(1)}$ lower bound.
[5] We remark that the limitation of $m > 42n$ in this approach is due to the fact that to augment a graph to become an expander requires at least adding some amount of edges, which unavoidably affects the densest subgraph.
[6] Note that in comparison with HPS, we do not prove hardness for graphs of constant degree, which is outside the scope of our paper.

understanding of the complexity of dynamic problems on various graph families, including expanders, along the lines of Question 1. As we have seen, DD-WTERs imply that expander decompositions are useless because they show equivalence between the worst-case and the expander-case complexities. We remark that the results of HPS imply the same, assuming their obtained lower bounds are tight. Hence, it implies a *conditional* answer to these questions, whereas self-reductions imply an *unconditional* one.

1.3 Expanders in distributed models

The above results demonstrate that the impact of Direct-WTERs goes beyond the classic Word-RAM model of computation. A natural continuation of this is to apply these techniques to more models of computation, where expanders, expander decompositions, and derandomization are important subjects. In the full version of this paper, we discuss the applicability of Direct-WTERs in distributed models of computation. We show that while there are limitations in adapting Direct-WTERs to CONGEST, it is possible to do so in CONGESTED-CLIQUE and MPC (Massively Parallel Computation).

Roadmap

A technical overview is given in Section 2, where we also explain the differences compared to [2]. Then, after some preliminaries in Section 3, we provide in Section 4 the complete details of our derandomized core gadget and its dynamic adaptation. Section 5 presents Direct-WTERs for Max-Cut, Densest Subgraph, and Graphical OMv. In the full version of this paper, we show adaptations of the new core gadget to obtain deterministic (and dynamic) Direct-WTERs to problems that appeared in [2] and other related problems.

2 Technical Overview

In this section, we summarize the core gadget of Abboud and Wallheimer [2] (henceforth, AW) that is used in all their Direct-WTERs, and then present our modification. Roughly, their construction boils down to the following procedure.

AW's core gadget

Given a graph $G = (V, E)$, add a set U of n vertices called *expansion layer*. Then, for every vertex $v \in V$, take a sample of $\deg_G(v) + O(\log n)$ vertices in U and make them neighbors of v. Clearly, the size of this graph and the running time are both $O(m + n \log n)$. In addition, they prove that it is an $\Omega(1)$-expander with high probability by showing that the probability that a cut in G_{exp} ends up being sparse is very low. However, there is no guarantee that the problem's solution to G can be computed easily from the solutions to G_{exp}. To this end, AW provide additional, mostly simple gadgets to control the solution in G_{exp} in a predictable manner while preserving the conductance of G_{exp} up to a constant factor.

Derandomizing the core gadget

The first limitation of AW's approach that we aim to resolve in this paper is their use of randomness. To do so, one could attempt to introduce an *explicit*, d-regular bipartite $\Omega(1)$-expander X between V and U. However, which degree d should we choose? For starters, we consider picking some constant $d \geq 3$. However, this approach might fail if G contains cuts with many internal edges and few out-going edges. For instance, if G includes a cut

S of $\sqrt{n}$ vertices that induce an isolated $\sqrt{n}$-clique, then in G_{exp} we have $vol(S) = \Theta(n)$, while the number of out-going edges added by the expander is $O(|S|d) = O(\sqrt{n})$, thus $\phi(S) = O(1/\sqrt{n})$. On the opposite end, we could select a complete bipartite graph with $\Theta(n^2)$ edges. While this will address the problem and indeed result in an $\Omega(1)$-expander, this approach is undesirable due to the excessive number of added edges. Hence, the most natural strategy to consider is incorporating a d-regular expander for $d = \lceil \frac{m}{n} \rceil$. To be more precise, since explicit constructions exist only for $d \geq 3$, the approach is to pick $d = \lceil \frac{m}{n} \rceil + 3$ (as it could be the case that, for example, $m = 0$). While the number of added edges is now $O(m + n)$, this approach is slightly naïve, as it does not even address the above problem of isolated $\sqrt{n}$-clique in a graph with $O(n)$ edges. The crux of the issue lies in the fact that G is not regular. If G were d-regular, then the inclusion of a d-regular expander would resolve our problem. One can look for expanders with a certain degree sequence, matching the one of G, but constructing those seems challenging.

The starting point for our modification is the observation that the randomness in AW's constructions is coming to accomplish *two things at once*: first and foremost, choosing random edges breaks structure and creates a "random like" graph. However, second, and less obviously, it is a way to achieve a certain "balanced allocation" of neighbors in U, resulting in nearly uniform degrees in U, which is crucial for their analysis to work. This leads to our modification, in which we substitute their single expansion layer U with two layers: one for balanced allocation denoted L, connected to V using a load-balancing algorithm, ensuring that the degrees in this layer are about $\frac{2m}{n}$, and another layer for expansion denoted R, connected to L using the edges of a bipartite, $(\frac{2m}{n} + 3)$-regular $\Omega(1)$-expander. A random graph can accomplish the construction of the first layer, as AW did, but it could also be accomplished *in any other way*, such as the standard *Round-Robin* algorithm without any randomness. Then, all we have to do is use an explicit construction (described in the full version of this paper) of a *regular* $\Omega(1)$-expander. This is the whole deterministic core gadget in Section 4. Using it, we follow the steps of AW and add various gadgets to G_{exp}, such that the solution for G can be retrieved from the solution for G_{exp}.

Dynamizing the core gadget

The above core gadget is not suitable for the fully dynamic setting, where G undergoes edge insertions and deletions. The main issue arising is that when the degree of a vertex $v \in V$ increases due to edge insertions, we need to allocate to v additional neighbors from L while preserving approximately balanced degrees in L. One attempt to solve this issue is to add an edge from v to a minimum-degree vertex in L, which can be computed quickly using a priority queue. While this ensures balanced degrees in L, there is a subtle issue that the minimum-degree vertex in L might already be a neighbor of v; hence we can not add another edge to it because that would create a parallel edge (which we aim to avoid). Therefore, we slightly modify this heuristic, and instead, our suggested approach is compute the successor of the minimum-degree vertex in L repeatedly, until a minimum-degree vertex in $L \setminus N(v)$ is reached, and then make it a neighbor of v. The required computation here is proportional to the degree of v, which might not be constant, but using lazy updates this approach results in amortized cost $O(1)$. However, a possible issue arising in this algorithm is that the minimum-degree vertex in $L \setminus N(v)$ is not necessarily a minimum-degree vertex in L, so it needs to be clarified that the degrees in L remain balanced, as otherwise the graph might not be an $\Omega(1)$-expander. Nonetheless, we prove that the degrees in L become overly imbalanced only after $\Omega(m + n)$ edge insertions. At this stage, there is enough credit to reconstruct G_{exp} from scratch, resulting in a total amortized cost of $O(1)$ per edge insertion.

In general, edge deletions do not require further computation. However, too many edge deletions or insertions will make X too dense or too sparse compared to G, resulting either in a large blowup or small conductance. This may happen only after $\Omega(m+n)$ updates, thus it can also be handled in amortized cost $O(1)$ by periodic reconstruction.

3 Preliminaries

Let $G = (V, E)$ be an undirected simple graph. Throughout the paper, we use n and m to denote the number of edges in vertices in G, respectively. Denote the neighborhood of v by $N(v) := \{u \in V \mid uv \in E\}$ and the degree of v by $\deg(v)$. We say that G is d-regular if $\deg(v) = d$ for all $v \in V$. A vertex v is called a *pendant vertex* if $\deg(v) = 1$. The set of all edges with one endpoint in $S \subseteq V$ and another endpoint in $T \subseteq V$ is denoted by $E(S,T) := \{uv \in E \mid u \in S, v \in T\}$, and its cardinality is denoted by $e(S,T) := |E(S,T)|$. We call $E(S, V \setminus S)$ the *out-going edges* of S. We employ subscripts to indicate which graphs we refer to when it is not clear from the context. For instance, $\deg_H(v)$ denotes the degree of vertex v in a graph H.

Conductance and edge-expansion

Let $S \subseteq V$ be a cut. The *volume* of S is defined as $vol(S) := \sum_{v\in S} \deg(v)$. The *conductance* of S is defined as $\phi(S) := e(S, V \setminus S)/\min(vol(S), vol(V \setminus S))$. If $\min(vol(S), vol(V \setminus S)) = 0$, we define $\phi(S) = 0$. The conductance of the entire graph G is defined as $\phi_G := \min_{S\subseteq V} \phi(S)$. Throughout this paper, unless explicitly indicated otherwise, we adhere to the following definition of expander graphs, in which $0 \le \phi \le 1$.

▶ **Definition 6.** *G is a ϕ-expander if $\phi_G \ge \phi$.*

Another related notion of expansion that we use in our proofs is edge expansion.

▶ **Definition 7.** *The edge expansion of G is:*

$$h_G := \min_{\emptyset\neq S\subseteq V, |S|\le n/2} \frac{e(S, V \setminus S)}{|S|}.$$

We will say that G is an h-edge expander if $h_G \ge h$.

A known fact states that conductance and edge expansion are interchangeable in regular graphs.

▶ **Fact 8.** *If G is a d-regular h-edge expander, then it is also a $\frac{h}{d}$-expander. In particular, if G is an $\Omega(d)$-edge expander, then it is also an $\Omega(1)$-expander.*

4 The Core Gadget

In this section, we present a derandomized core gadget on which we base our results. The core gadget is a deterministic algorithm that takes a graph G and augments it with new vertices and edges to output an $\Omega(1)$-expander G_{exp} in $\tilde{O}(m+n)$ time. We will utilize an explicit construction of d-regular, bipartite, ϕd-edge expanders on $2N$ vertices for some constant $\phi > 0$. Assume that given any $d \ge 3$, and sufficiently large N, we can construct such graph in $\tilde{O}(Nd)$ time. See the full version of this paper for more details about the construction.

4.1 The core gadget

Given a graph G, augment G with an initially empty bipartite graph featuring N vertices on each side, denoted $L := \{x_1, x_2, \ldots, x_N\}$ and $R := \{y_1, y_2, \ldots, y_N\}$. N is a parameter that can vary depending on the application, but for the most basic construction, we set it to $N = (1 + o(1))n$. The gadget consists of the following two-step construction:

1. (Degree balancing) For every $v \in V$ add, using a Round-Robin algorithm, $\deg_G(v) + 3$ neighbors in L. Namely, follow a circular order over L to pick neighbors one at a time. Assuming $N \geq n + 2$, every vertex is connected to $\deg_G(v) + 3$ *distinct* neighbors in L without wrapping around. The Round-Robin algorithm guarantees that the degrees within L are almost balanced. In more detail, since the total number of edges that we add in this step is $\sum_{v \in V}(\deg_G(v) + 3) = 2m + 3n$, the degree of every vertex in L is either $\lfloor \frac{2m+3n}{N} \rfloor$ or $\lceil \frac{2m+3n}{N} \rceil$.
2. (Expander construction) Set $d := \lceil \frac{2m+3n}{N} \rceil$. Let X be a bipartite, d-regular, ϕd-edge expander on $L \cup R$.

Denote the resulting graph by $G_{exp} := (V_{exp}, E_{exp})$, illustrated in Figure 1. Note that the running time of this gadget is $\tilde{O}(m + n + dN) = \tilde{O}(m + n)$.

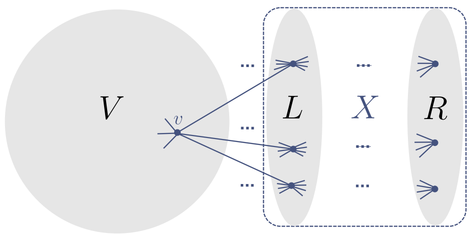

Figure 1 The core gadget augments G with $O(m + n)$ edges connected to a bipartite, d-regular, ϕd-edge expander, resulting in an $\Omega(1)$-expander G_{exp}.

▶ **Lemma 9.** *The graph G_{exp} is an $\Omega(1)$-expander.*

The proof of this lemma is deferred to the full version of this paper. We remark that the blowup of the core gadget, i.e., the number of added vertices and edges, is $2N = O(n)$ vertices and $O(Nd) = O(m + n)$ edges. Furthermore, if G is a graph of maximum degree Δ, then G_{exp} is a graph of maximum degree $2\Delta + 3$.

Before continuing further, let us state two variants of the core gadgets which are useful throughout the paper.

4.2 Variants of the core gadget

The first variant we consider is a generalization of the core gadget, which allows greater variations in the degrees of the vertices in G_{exp}.

▶ **Lemma 10.** *For every $0 < \varepsilon \leq 1$, $\alpha \geq 1$, and an integer $d_X \geq 3$, consider the following generalization of the core gadget: (1) Every $v \in V$ has at least $\varepsilon \deg_G(v) + 1$ neighbors in L, (2) X is a d_X-regular, ϕd_X-edge expander, for some some constant $\phi > 0$. and (3) The degrees of all the vertices in L are within $[d_X, \alpha d_X]$. Then G_{exp} is an $\phi\varepsilon/(5\alpha)$-expander.*

The proof of this lemma is deferred to the full version of this paper.

Using this generalization, one can also obtain a tradeoff between the blowup in the number of vertices and the conductance of G_{exp} as follows. Given $0 < \varepsilon \leq \delta \leq 1$, modify the core gadget by setting $N = \lceil \delta n(1 + o(1)) \rceil$, and for every $v \in V$, instead of adding $\deg_G(v) + 3$

edges from v to L, add $\lceil \varepsilon \deg_G(v) \rceil + 3$ such edges. By Lemma 10, the resulting graph is an $\Omega(\varepsilon)$-expander. Moreover, the blowup in the number of vertices is $2N \leq 2\delta n + O(1)$, and the blowup in the number of edges is $2\varepsilon m + 3n$. Rescaling appropriately, we obtain that given any $0 < \varepsilon \leq \delta \leq 1$, we can construct an $\Omega(\varepsilon)$-expander with a blowup of δn in the number of vertices and $\varepsilon m + 3n$ in the number of edges.

Finally, one can modify the core gadget to preserve bipartiteness as follows. Given a bipartite graph $G = (A \cup B, E)$, where $|A| = |B| = n$, instead of adding edges from $A \cup B$ to L, add edges from A to L and from B to R. Namely, for every $v \in A$, add $\deg_G(v) + 3$ edges to L, and for every $v \in B$, add $\deg_G(v) + 3$ edges to R. The rest of the construction stays the same, i.e., we construct a d-regular expander X between L and R for the same value d as before. Clearly, the blowup in the number of edges and vertices is still linear, and the graph is bipartite, with sides $A \cup R$ and $B \cup L$. The proof that the graph is an $\Omega(1)$-expander appears in the full version of this paper.

4.3 Fully-dynamic core gadget

Our algorithm makes use of three procedures: (1) `UPDATE`, which computes and adds a batch of edges from a vertex in G to L, (2) `BALANCE`, which rebalances the degrees in L using Round-Robin, and (3) `RECOMPUTE`, which recomputes the graph using the static core gadget. In this section, we use G^{exp} to denote a dynamic $\Omega(1)$-expander output by the dynamic core gadget. We use subscripts to indicate the state of a dynamic graph at a certain time, e.g., G_t is the dynamic graph G at time t. Let us describe the algorithm.

Preprocessing

In the preprocessing step, given G_0, apply the static core gadget to construct an $\Omega(1)$-expander G_0^{exp}. Store the vertices of L sorted according to their degrees, in a data structure supporting updates and successor queries in $O(1)$ time.[7]

Edge insertions and deletions

For every insertion of an edge uv to G, begin by inserting uv to G^{exp}. Denote by m_t the number of edges in G at the last time we applied the `RECOMPUTE` procedure (or m_0 if we did not apply it yet). If $m \geq 2m_t + n$, apply the `RECOMPUTE` procedure to recompute the graph using the static core gadget and finish.

Otherwise, let us describe the process we apply to v and similarly do to u. Denote by $\deg_L(v)$ the number of neighbors that v has in L, i.e., $\deg_L(v) = |N(v) \cap L|$.

1. If $\deg_G(v) < 2\deg_L(v)$, finish. Otherwise, apply the `UPDATE` procedure to v to add a new batch of neighbors of v in L, after which we have $\deg_L(v) = \deg_G(v) + 3$.
2. Check if the degrees in L became unbalanced, namely, if $\Delta_L \geq 2\delta_L$, where Δ_L and δ_L are the maximum and minimum-degree vertices in L, respectively. If so, apply the `BALANCE` procedure, after which $\Delta_L \in \{\delta_L, \delta_L + 1\}$.

For every deletion of an edge uv from G, delete uv from G^{exp}. Then, if $n \leq m \leq 0.5m_t$, apply the `RECOMPUTE` procedure to recompute X.

Let us now describe the three procedures used above.

[7] Naïvely this would take $\tilde{O}(1)$ time using standard data structures, but it can be optimized to $O(1)$ since the degrees are integers in the range $[1, 2N]$, and our updates only increase or decrease the degree of a vertex by 1.

- `UPDATE` Let $k := \deg_L(v)$. Compute $k+3$ minimum-degree vertices in $L \setminus N(v)$, denoted $x_1, x_2, \ldots, x_{k+3}$, by repeatedly making successor queries to the minimum degree vertex in L and skipping vertices which belong to $N(v)$. For every x_i, insert an edge vx_i to G^{exp}. Note that now we have $\deg_L(v) = \deg_G(v) + 3$.
- `BALANCE` Compute, using Round-Robin, a new set of V-to-L edges, denoted A, and then replace $E(V, L)$ with A. This is done by inserting the edges of A in the order given by the Round-Robin algorithm before removing $E(V, L) \setminus A$, to ensure that the degrees do not vary too much in the intermediate graphs.
- `RECOMPUTE`. Apply the static core gadget to compute a set of edges A, and X', where A is the set of V-to-L edges define above, and X' is the expander on $L \cup R$. In particular, X' is a $d_{X'}$-regular, $\Omega(d_{X'})$-edge expander for $d_{X'} = \lceil \frac{2m+3n}{N} \rceil$. Replace X with X' by first inserting the edges of X', and then removing the leftover edges of X which do not belong to X'. Then insert the edges of A and remove $E(V, L)$, as we did above.

For the analysis of the dynamic core gadget, see the full version of this paper.

5 Direct-WTERs for Max-Cut, Densest Subgraph, and Graphical OMv

In this section, we develop further this line of research by providing a Direct-WTER for Max-Cut, a DD-WTER for Densest Subgraph, and a DD-WTER for Graphical OMv instances.

5.1 Direct-WTER for Max-Cut

In the Max-Cut problem, the goal is to compute the maximum cut in a graph G, which we denote by $MC(G) := \max_{S \subseteq V} e(S, V \setminus S)$. To make the graph an $\Omega(1)$-expander, simply applying the core gadget does not work because it might affect the maximum cut in G_{exp} unpredictably. To this end, we add a gadget that ensures that any maximum cut in G_{exp} will separate L from R. Additionally, we modify the core gadget so that every vertex in G will have the same number of neighbors in L and R. These two gadgets together ensure that the vertices in G do not get a preference to be in the part of L or R, due to symmetry. Consequently, the maximum cut in G_{exp} induces a maximum cut in G.

Let us now describe the reduction in more detail. Given $0 < \varepsilon \leq 1$, apply the core gadget with a tradeoff between the conductance and the blowup, as described in Section 4.2, to get an $\Omega(\varepsilon^2)$-expander with a blowup of $2N = \varepsilon n$ in the number of vertices. This is achieved by picking $N = \varepsilon/2n$ (roughly), and adding for every $v \in V$, $\varepsilon^2 \deg_G(v) + 3$ neighbors in L. Add a symmetric copy of the V-to-L edges between V and R as well.

Observe that for every vertex in $L \cup R$, the number of neighbors it has in V is at most $d \leq \frac{2\varepsilon^2 m + 4n}{N} = 4\varepsilon m/n + O(1) \leq 4\varepsilon n + O(1)$, where d is also the degree of the expander between L and R. Now, add two bi-cliques of size $N \times 3d$ as follows: add two sets of F_L and F_R containing $3d$ vertices each, and all L-to-F_L and R-to-F_R edges. The purpose of this gadget is to ensure that L and R are separated in a maximum cut. Denote the resulting graph by G_{exp}. See Figure Figure 2.

Observe that the blowup in the number of vertices is $2N + 6d \leq 25\varepsilon n + O(1)$. Moreover, we claim that the additional gadgets: the V-to-R edges and the bi-cliques, do not ruin the graph's expansion, and that G_{exp} is an $\Omega(\varepsilon^2)$-expander. To see why, observe that the induced graph on $L \cup F_R \cup F_L \cup R$ is a bipartite, $\Omega(d)$-edge expander, hence the proof of Lemma 10 holds for it as well (by letting $L \cup R$ play the role of L).

Finallly, the next claim shows that the maximum cut in G_{exp} encodes the maximum cut in G.

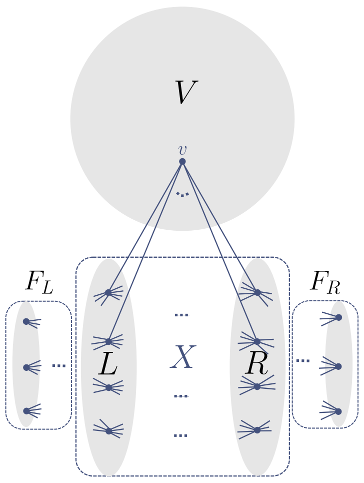

Figure 2 A Direct-WTER for the Max-Cut problem.

▷ Claim 11. The maximum cut in G_{exp} is $MC(G_{exp}) = MC(G) + 7dN$.

Proof of Claim 11. Let $(S, V_{exp} \setminus S)$ be a maximum cut in G_{exp}. We call the vertices of S *red* and the vertices of $V_{exp} \setminus S$ *blue*. Consider the following observations:

1. F_L must be monochromatic because otherwise, we could improve the cut by giving F_L the opposite color of the majority color in L. A similar observation applies to F_R as well.
2. L is monochromatic and has the opposite color to F_L. To see why, assume w.l.o.g. that F_L is red, and for contradiction, assume that some $x \in L$ is also red. By changing the color of x to blue, we gain $|F_L| = 3d$ edges and lose at most $\deg_V(x) + \deg_R(x) \leq 2d$ edges, a contradiction to S being a maximum cut. A similar observation applies to F_R as well.
3. L and R have different colors. To see why, assume w.l.o.g. that L is red, and assume for contradiction that R is also red. Based on the previous observation, F_L is blue. By flipping the colors of L and F_L, we gain the L-to-R edges; there are $N \cdot d$ such edges, and lose the L-to-V edges; there are at most $N \cdot d$ such edges. Hence, we can assume without loss of generality that this observation holds.

Now, we claim that $S \cap V$ is a maximum cut in G. To see why, observe that since L and R have different colors, and for every $v \in V$, we have $\deg_L(v) = \deg_R(v)$, then replacing $S \cap V$ with any other cut in V will only change the number of edges cut inside G. Therefore, we get $MC(G_{exp}) = MC(G) + N(|F_L| + |F_R| + d) = MC(G) + 7dN$. ◁

5.2 DD-WTER for Densest Subgraph

In the Densest Subgraph problem, we define the *density* of non-empty set $S \subseteq V$ to be $\rho(S) := m_S/|S|$, where m_S is the number of edges in the subgraph induced by S. The goal is to compute $\rho(G) := \max_{S \subseteq V} \rho(S)$. In our DD-WTER, we will make use of the following claim whose proof can be found in the full version of this paper.

▷ Claim 12. Any set S^* that maximizes ρ in a graph with m edges and n vertices does not contain any vertex of degree less than m/n.

In addition, we assume that the density in G is sufficiently large, namely, $m > 42n$.[8] To make the graph an $\Omega(1)$-expander, we apply the core gadget in a way that only introduces vertices of degree smaller than m_{exp}/n_{exp}, where m_{exp} and n_{exp} are the number of edges and vertices in G_{exp}, respectively. To this end, consider the graph obtained by applying the core gadget with a tradeoff between the conductance and the blowup, as described in Section 4.2. Specifically, we pick the parameters such that in G_{exp}, every $v \in V$ is connected to $\lceil \varepsilon \deg_G(v) \rceil + 3$ vertices in L, and $N = |L| = |R| = n(1 + o(1))$. The conductance of this graph is $\Omega(\varepsilon)$. The parameter ε is chosen to be a sufficiently small constant which will be determined later.

▷ Claim 13. The maximum density in G_{exp} is $\rho(G_{exp}) = \rho(G)$.

Proof of Claim 13. We will show that $m_{exp}/n_{exp} > \mu$, where μ is the maximum degree in L (and therefore also in $L \cup R$). Hence, by Claim 12, it will follow that any maximum density subgraph in G_{exp} does not contain vertices from $L \cup R$, so it must induce a maximum density subgraph in G.

Note that the number of V-to-L edges is bounded by $2\varepsilon m + 3n \leq e(V, L) \leq 2\varepsilon m + 4n$, and that the degree of the expander between L and R is $d = \lceil e(V, L)/N \rceil$. Hence, we have $m_{exp} = m + e(V, L) + Nd \geq m + (2\varepsilon m + 3n) + (2\varepsilon m + 3n) = (1 + 4\varepsilon)m + 6n$. In addition, note that the maximum degree in L is:

$$\mu \leq 2d \leq 2\left(\frac{e(V, L)}{N} + 1\right) \leq \frac{4\varepsilon m + 9n + o(n)}{n}.$$

Now, observe that the inequality:

$$\mu \leq \frac{4\varepsilon m + 9n + o(n)}{n} < \frac{(1 + 4\varepsilon)m + 6n}{3n + o(n)} \leq \frac{m_{exp}}{n_{exp}},$$

holds when n is sufficiently large, $1 - 8\varepsilon > 0$, and $21n/(1 - 8\varepsilon) < m$. For example, by picking $\varepsilon < 1/16$, we get that $\mu < m_{exp}/n_{exp}$ when $m > 42n$. ◁

To adapt the Direct-WTER to a DD-WTER, we replace the core gadget with the dynamic core gadget. However, note that in the dynamic core gadget there are some variations in the degrees due to lazy updates and rebalances. Nonetheless, these variations can be compensated for by picking a smaller ε, specifically $\varepsilon = 1/44$ will suffice.

5.3 DD-WTER for Graphical OMv instances

In this subsection, we demonstrate how our core gadget from Section 4 can be used to prove OMv-hardness to various OMv-hard problems by making the typical "OMv-hard" instances of a problem $\Omega(1)$-expanders. The definitions of the OMv and OuMv problems and the OMv Conjecture appear in the full version of this paper.

A *Graphical OMv* instance is constructed from an OuMv instance as follows. Given a $k \times k$ binary matrix M, construct bipartite graph $G_M := (A \cup B, E)$, where A and B are equally-sized parts, denoted by $A = \{a_1, \ldots, a_k\}$ and $B = \{b_1, \ldots, b_k\}$. The edges of G_M are defined according to the 1's of the matrix, i.e., $E := \{a_i b_j | M[i, j] = 1\}$. Next, add some problem-specific gadgets to G_M: for many OMv-hard problems, such as *st*-SubConn, *st*-SP, and more, the gadgets consist of $O(k)$ vertices and $O(k)$ edges that are connected to $L \cup B$ in a certain (dynamic) way.

[8] For clarity, we do not attempt to optimize this constant, although it can be reduced.

Making Graphical OMv instances $\Omega(1)$-expanders

To make such instances $\Omega(1)$-expanders, we apply the bipartiteness-preserving (see Section 4.2) core-gadget on G_M before adding the problem-specific gadgets. In some cases, as we will soon demonstrate for the *st*-SP problem, such gadgets preserve expansion, or they can be easily adapted to preserve expansion. Therefore, this adaptation proves that their OMv-hard instances are $\Omega(1)$-expanders. Let us now demonstrate this technique for the problem of *st*-SP and prove Proposition 5 which states that *st*-SP is OMv-hard on $\Omega(1)$-expanders.

Proof of Proposition 5. We will prove the proposition for an easier variant of the problem called *st*-SP (3 vs. 5), where the goal is only to distinguish between $dist(s,t) = 3$ and $dist(s,t) \geq 5$. Henzinger et al. [15] proved a lower bound to this problem via a reduction from OuMv: construct G_M, add vertices s and t to G_M, and then update the edges between s-to-A and t-to-B according to the input vectors. It then followed that whenever $u^T M v = 1$, then $dist(s,t) = 3$, and whenever $u^T M v = 0$, $dist(s,t) \geq 5$. By picking $k = \sqrt{m}$, the graph has $O(m)$ edges, and the lower bound the follows was $m^{1/2-\varepsilon}$ per update and $m^{1-\varepsilon}$ per query. We now modify their construction as follows.

Given M, apply the bipartiteness-preserving core gadget to G_M before adding vertices s and t. Then, pick a non-edge in the expander X, i.e., $xy \notin E(X)$ for some $x \in L$ and $y \in R$ arbitrarily. The purpose of this modification is to ensure that s and t are connected to the graph without introducing a path of length < 5 between them. Now, we procceed as in [15]; namely, given vectors $u = (u_1, u_2, \ldots, u_k)$ and $v = (v_1, v_2, \ldots, v_k)$, we update the graph by adding the edges sa_i iff $u_i = 1$, and tb_j iff $v_j = 1$. Note that $u^T M v = 1$ iff $dist(s,t) = 3$, and otherwise $dist(s,t) \geq 5$. In addition, we claim that the graph is an $\Omega(1)$-expander. This follows from the next claim.

$\triangleright$ Claim 14. If G is a ϕ-expander for some $\phi > 0$, then adding a vertex to G and connecting it arbitrarily to $\ell \geq 1$ vertices, results in an $\phi/4$-expander.

The proof of this claim is deferred to the full version of this paper.

In our setting, we add two vertices to G_M, and at all times, each of them is connected to at least one vertex in G_M. Therefore, since G_M is a ϕ-expander for some constant ϕ, then the resulting graph is a $\phi/16$-expander at all times according to this claim. Hence, assuming the OMv Conjecture, there is no algorithm for *st*-SP (3 vs. 5) on $\Omega(1)$-expanders, whose preprocessing time is polynomial, update time $m^{1/2-\varepsilon}$, and query time $m^{1-\varepsilon}$, for any $\varepsilon > 0$. ◀

References

1 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.

2 Amir Abboud and Nathan Wallheimer. Worst-case to expander-case reductions. In Yael Tauman Kalai, editor, *14th Innovations in Theoretical Computer Science Conference, ITCS 2023, January 10-13, 2023, MIT, Cambridge, Massachusetts, USA*, volume 251 of *LIPIcs*, pages 1:1–1:23. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2023. doi:10.4230/LIPIcs.ITCS.2023.1.

3 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.

4 Vedat Levi Alev, Nima Anari, Lap Chi Lau, and Shayan Oveis Gharan. Graph clustering using effective resistance. In Anna R. Karlin, editor, *9th Innovations in Theoretical Computer Science Conference, ITCS 2018, January 11-14, 2018, Cambridge, MA, USA*, volume 94 of *LIPIcs*, pages 41:1–41:16. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2018. `doi:10.4230/LIPIcs.ITCS.2018.41`.

5 Amos Beimel, Haim Kaplan, Yishay Mansour, Kobbi Nissim, Thatchaphol Saranurak, and Uri Stemmer. Dynamic algorithms against an adaptive adversary: Generic constructions and lower bounds. In *Proceedings of the 54th Annual ACM SIGACT Symposium on Theory of Computing*, pages 1671–1684, 2022. `doi:10.1145/3519935.3520064`.

6 Timothy M Chan and Qizheng He. Reducing 3sum to convolution-3sum. In *Symposium on Simplicity in Algorithms*, pages 1–7. SIAM, 2020. `doi:10.1137/1.9781611976014.1`.

7 Yi-Jun Chang and Thatchaphol Saranurak. Improved distributed expander decomposition and nearly optimal triangle enumeration. In *Proceedings of the 2019 ACM Symposium on Principles of Distributed Computing*, pages 66–73, 2019. `doi:10.1145/3293611.3331618`.

8 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. In *2022 IEEE 63rd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 612–623. IEEE, 2022. `doi:10.1109/FOCS54457.2022.00064`.

9 Lijie Chen and Roei Tell. Guest column: New ways of studying the bpp= p conjecture. *ACM SIGACT News*, 54(2):44–69, 2023. `doi:10.1145/3604943.3604950`.

10 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 *2020 IEEE 61st Annual Symposium on Foundations of Computer Science (FOCS)*, pages 1158–1167. IEEE, 2020. `doi:10.1109/FOCS46700.2020.00111`.

11 Arnold Filtser, Michael Kapralov, and Mikhail Makarov. Expander decomposition in dynamic streams. In Yael Tauman Kalai, editor, *14th Innovations in Theoretical Computer Science Conference, ITCS 2023, January 10-13, 2023, MIT, Cambridge, Massachusetts, USA*, volume 251 of *LIPIcs*, pages 50:1–50:13. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2023. `doi:10.4230/LIPIcs.ITCS.2023.50`.

12 Nick Fischer, Piotr Kaliciak, and Adam Polak. Deterministic 3sum-hardness. In Venkatesan Guruswami, editor, *15th Innovations in Theoretical Computer Science Conference, ITCS 2024, January 30 to February 2, 2024, Berkeley, CA, USA*, volume 287 of *LIPIcs*, pages 49:1–49:24. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2024. `doi:10.4230/LIPIcs.ITCS.2024.49`.

13 Michel X Goemans and David P Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *Journal of the ACM (JACM)*, 42(6):1115–1145, 1995. `doi:10.1145/227683.227684`.

14 Gramoz Goranci, Harald Räcke, Thatchaphol Saranurak, and Zihan Tan. The expander hierarchy and its applications to dynamic graph algorithms. In *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 2212–2228. SIAM, 2021. `doi:10.1137/1.9781611976465.132`.

15 Monika Henzinger, Sebastian Krinninger, Danupon Nanongkai, and Thatchaphol Saranurak. Unifying and strengthening hardness for dynamic problems via the online matrix-vector multiplication conjecture. In *Proceedings of the forty-seventh annual ACM symposium on Theory of computing*, pages 21–30, 2015. `doi:10.1145/2746539.2746609`.

16 Monika Henzinger, Jason Li, Satish Rao, and Di Wang. Deterministic near-linear time minimum cut in weighted graphs. In *Proceedings of the 2024 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 3089–3139. SIAM, 2024. `doi:10.1137/1.9781611977912.111`.

17 Monika Henzinger, Ami Paz, and A. R. Sricharan. Fine-grained complexity lower bounds for families of dynamic graphs. In Shiri Chechik, Gonzalo Navarro, Eva Rotenberg, and Grzegorz Herman, editors, *30th Annual European Symposium on Algorithms, ESA 2022, September 5-9, 2022, Berlin/Potsdam, Germany*, volume 244 of *LIPIcs*, pages 65:1–65:14. Schloss Dagstuhl –

Leibniz-Zentrum für Informatik, Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPIcs.ESA.2022.65.

18 Satyen Kale and C. Seshadhri. Combinatorial approximation algorithms for maxcut using random walks. In Bernard Chazelle, editor, *Innovations in Computer Science - ICS 2011, Tsinghua University, Beijing, China, January 7-9, 2011. Proceedings*, pages 367–388. Tsinghua University Press, 2011. URL: http://conference.iiis.tsinghua.edu.cn/ICS2011/content/papers/20.html.

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

20 Ken-ichi Kawarabayashi and Mikkel Thorup. Deterministic edge connectivity in near-linear time. *J. ACM*, 66(1):4:1–4:50, 2019. doi:10.1145/3274663.

21 Jason Li. Deterministic mincut in almost-linear time. In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*, pages 384–395, 2021. doi:10.1145/3406325.3451114.

22 Jason Li and Debmalya Panigrahi. Deterministic min-cut in poly-logarithmic max-flows. In Sandy Irani, editor, *61st IEEE Annual Symposium on Foundations of Computer Science, FOCS 2020, Durham, NC, USA, November 16-19, 2020*, pages 85–92. IEEE, 2020. doi:10.1109/FOCS46700.2020.00017.

23 Lawrence Li and Sushant Sachdeva. A new approach to estimating effective resistances and counting spanning trees in expander graphs. In *Proceedings of the 2023 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 2728–2745. SIAM, 2023. doi:10.1137/1.9781611977554.ch102.

24 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. doi:10.1109/FOCS.2017.92.

25 Lorenzo Orecchia, Leonard J Schulman, Umesh V Vazirani, and Nisheeth K Vishnoi. On partitioning graphs via single commodity flows. In *Proceedings of the fortieth annual ACM symposium on Theory of computing*, pages 461–470, 2008. doi:10.1145/1374376.1374442.

26 Pan Peng and Yuichi Yoshida. Sublinear-time algorithms for max cut, max e2lin (q), and unique label cover on expanders. In *Proceedings of the 2023 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 4936–4965. SIAM, 2023. doi:10.1137/1.9781611977554.ch180.

27 Thatchaphol Saranurak. A simple deterministic algorithm for edge connectivity. In *Symposium on Simplicity in Algorithms (SOSA)*, pages 80–85. SIAM, 2021. doi:10.1137/1.9781611976496.9.

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

29 Christian Sommer, Elad Verbin, and Wei Yu. Distance oracles for sparse graphs. In *2009 50th Annual IEEE Symposium on Foundations of Computer Science*, pages 703–712. IEEE, 2009. doi:10.1109/FOCS.2009.27.

30 Daniel A Spielman and Shang-Hua Teng. Nearly-linear time algorithms for graph partitioning, graph sparsification, and solving linear systems. In *Proceedings of the thirty-sixth annual ACM symposium on Theory of computing*, pages 81–90, 2004. doi:10.1145/1007352.1007372.

31 Luca Trevisan. Max cut and the smallest eigenvalue. In *Proceedings of the forty-first annual ACM symposium on Theory of computing*, pages 263–272, 2009. doi:10.1145/1536414.1536452.